ECEN 3002

Lab #7

Shared Resources

Derek Wright

All answers are given in blue right below the part/number that asks for a response.

Part II – Functional Testing

1. Create a functional test plan and procedure to test all functionality, per the requirements. Only a black-box test is required (verification of internal behavior, not visible to user, is not required)
   1. Startup the board
   2. Wait 5 seconds, and the LED1 will come on. The LCD will print mph 0, dir straight.
   3. Press either the left/right side of the capsense, LED1 will turn off. The LCD dir will update
   4. Wait 5 seconds, LED1 will turn on.
   5. Now press and hold the far left side for 5 seconds. The LCD will update to say Hard Left, and after 5 seconds the LED1 will turn on.
   6. Now touch the far left side of the slider. The direction will read “Hard Left”
   7. Now touch the near left side. The direction will read “Left”
   8. Now touch the near right side. The direction will read “Right”
   9. Now touch the far right side of the slider. The direction will read “Hard Right”
   10. Now touch both sides of the slider. The direction will read “Straight”, the same as when you aren’t touching at all.
   11. Now touch the full right side. The direction will read “Right.”
   12. Now touch the full right side and only the far left side of the slider. The direction will read “Right”, because you’re more favored towards turning right.
   13. Now push the right pushbutton (BTN0). The speed will go to 5mph.
   14. Now, press the left pushbutton (BTN1). The speed will go back to 0 mph.
   15. Now, set the speed to 50 mph.
   16. Press the right side of the capsense. LED0 will turn on.
   17. Release, LED0 will turn off.
   18. Press the left side of the capsense. LED0 will turn on.
   19. Now, set the speed to 80 mph. LED0 will turn on.
   20. Repeat XV-XIX using negative speeds instead. The same behavior will result.
2. T

Part III – System Analysis

\*\*Note to grader\*\*

I am one of the people who can’t use uC probe because I have a Mac, and the lab computers are not set up to run uC probe. The image shown below is courtesy of fellow classmate Jacob Sheiffler, for illustrative purposes. My implementation is slightly different, but it can be assumed we would see a similar thing. Information from Segger SystemView confirms the operation of my tasks.

1. Micrium uC-Probe
   1. Press one of the pushbuttons. Which tasks are awakened? Record the number of context switches that occur.
      1. The monitor task is awakened, after the scheduler of course.
      2. I’m unsure of the context switches because I don’t have uC Probe
   2. Touch one side of the touch slider. Which tasks are awakened? Record the number of context switches that occur.
      1. The LCD Task and the Monitor Task are awakened. I have the LCD task set to a higher priority, so it is called first, but then immediately after the monitor is called.
   3. Do the number of context switches recorded above match your expectations? Explain.
      1. Again, I don’t have access to this data because of the reasons listed above. Please refer to the Segger SystemView responses, as it is no fault of my own that the lab computers don’t have the correct SDK to run uC Probe.
   4. Save a screenshot of the application, showing the Task(s) tab.

A screenshot of a cell phone

Description automatically generated

This image is courtesy of Jacob Scheiffler, as mentioned above.

1. Segger SystemView
2. Check the scheduling of each task. Is each task scheduled as expected? Explain.
   1. Yes the scheduling is as expected. The buttons wake up the speed setpoint task, which are nominally running at 0 Hz (waiting for input). The Speed setpoint task in turn wakes up the monitor task which in turn wakes up the LED task.
   2. The direction task does a similar thing, waking up the monitor task, and then the LED task
   3. The LCD task is periodically
3. Record a screenshot of the events leading up to the scheduling of each task.

A picture containing screenshot

Description automatically generated

The above image shows the speed setpoint task being triggered. The next task to be awakened by the scheduler is the monitor task (purple), and then the LED task, just as expected.

A screenshot of a cell phone

Description automatically generated

The above image shows the Direction task being triggered. The next task to be awakened by the scheduler is the LCD task, but this is coincidental, because the LCD task is a periodic task using OSTimeDly, and is not awoken by the direction task. The monitor task (purple) is awakened by the direction task, but the LCD has a higher priority in my implementation. And then the LED task, just as expected.

A picture containing screenshot

Description automatically generated

Above is when a change is detected by the direction task. The LCD awakens (higher priority than the others), recognizes the change, and writes to the screen. As you can see, this takes a while!!