Giancarlo Pacenza
Matthew Daniel
Lab 2
1 March 2019

Lab 2 Report

Part 1

The logic behind Part 1 was fairly straightforward, with the vast majority of the code serving to set up the PIT and the LED. The main difficulty was finding the proper syntax for different operations. In the first few lines, we followed the steps laid out in the lab to allow us to toggle the red LED. We also loaded and enabled the timer. After this, we had an infinite while loop where the polling took place. At every iteration of the loop, we checked the value Timer Flag Register of the zeroth PIT to see if a timeout occurred. If there was a timeout, we toggled the red LED and cleared the Timer Flag Register. If there was no timeout, we did nothing. This process looped forever, toggling the LED approximately every second.

Part 2

In the second part of the lab, we used the same logic from Part 1 to toggle the blue and green LEDs. With regard to LED control, we just had to figure out the corresponding ports and pins for the different colors by looking at the lab writeup. Once we figured out the proper connections, we performed the same setup process for the green and blue LEDs as we did for the red LED in part 1 (enabling clock to port, etc.) . We also enabled the clock to the PIT module and loaded the timer as we had done for Part 1.

We struggled more in calling the interrupt and actually jumping to the interrupt service routine. Our only initial modification to enable the interrupt was the following line of code: PIT->CHANNEL[0].TCTRL = (1<<1);. Unfortunately, this did not succeed in producing the desired action - that is, the code in the interrupt handler never executed. We ultimately solved this problem by using just one command, PIT->CHANNEL[0].TXTRL = 3, to enable interrupts and the timer at the same time.

Once the above issue was settled, the rest of the main function consisted of an infinite while loop. Within the while loop, we would first run a for loop to create a delay of about 1 second, then we would toggle the blue LED.

In the interrupt handler, we toggled the green LED to turn it on, entered a .1 second for loop, then toggled the green LED off again. Finally, we cleared the interrupt flag to correctly exit the interrupt code after flashing the green LED.

Work Distribution

When meeting in person, given that Matt has a Windows machine, we did most of the coding on his computer. While Matt did the physical typing, both team members were actively reviewing and contributing to the code. After a meeting, the person with more free time would take the K64F with them to work separately. Any code developed separately by either party was generally exchanged over email. Most progress was made in office hours, which both team members attended. Both team members contributed to all sections of the writeup. Apart from emailing code, all other communication between members was through text message.