Part 1

Nov 20, 2019

1. Read through the design requirements and procedures
    a. Since this project is in parts I will write the documents pertaining to the part instead of all together.
    b. Deliverables for part one:
        i. Design log (this document)
        ii. Algorithm (both high and low level)
        iii. .s file
2. Research the GPIO pins for the project and write high level algorithm
    a. Four user LEDs:
        i. USR0: GPIO1_21 PROC PIN: V15
        ii. USR1: GPIO2_22 PROC PIN: U15
        iii. USR2: GPIO2_23 PROC PIN: T15
        iv. USR3: GPIO2_24 PROC PIN: V16
    b. A logic 1 will turn on the LEDs
    c. High level program requirements:
        i. Switch between LEDs 0, 3 and 1, 2 for two seconds
        ii. Using the text I will create the high level algorithm including all registers and addresses needed for the GPIO pins
        iii. Since a requirement for using an infinite loop or a required amount of loops, I will use 10 loops for pulsing LEDs
3. Continue on to write the low level algorithm while researching values to use
    a. According to the text I need to use 2sec/2ns for finding the value of a 2 second delay loop which is 1e01 or 0x3B9ACA00
    b. To turn on/off LED0 I need to write 0x00200000 to GPIO1
    c. To turn on/off LED3 I need to write 0x01000000 to GPIO2
    d. To turn on/off the other two together I write 0x00C00000 to GPIO2
    e. 0x4804C000 base for GPIO1
    f. 0x481AC000 base for GPIO2
    g. Add 0x194 for SETDATAOUT 0x190 for CLEARDATAOUT
    h. Write 0x02 to GPIO clocks 1 and 2 for this project
    i. 0xFFDFFFFF for enable GPIO1_21 as output
    j. 0xFE3FFFFF for enable GPIO2_22, GPIO2_23, and GPIO2_24 as outputs
    k. Add 134 for output enable

Nov 21, 2019

1. Starting to writing assembly code with comments
    a. Followed the example from the text

Nov 23, 2019

1. Implementing and debugging the final code
    a. Ran into a few issues

Part 2

Dec 3, 2019

1. There are some errors I have to debug and figure out before continuing on to part 2

a. One error with initializing GPIO2
b. **I'm not sure which document I looked at to refer the user LEDs but it was wrong. Looked at the datasheet from digikey instead and the user LEDs are all GPIO1 not 2.**
c. **Further research of the manuals brought the conclusion. The revision of the manual I originally looked at was A5.2 created in 2013. The correct manual is rev b made in 2014.**
d. Another error with the timing calculation of the pulse
e. Since a timer will be used instead later on. I will not spend time researching the clock speed for the correct delay loop calculation
f. Went back and fixed the algorithms and code

2. Then I continue on with the design of part 2 (adding a push button start/stop system)
    a. I took most of this time gathering resources and putting things straight in my head. Probably, the most complex thing so far.
    b. I will be referring to the high level algorithm from the text.
    c. Started the algorithms

Dec 5, 2019

1. Start coding part 2 of the program
    a. Didn't really have any issues. Straightforward after following the templates from the text. Simple to add my own altercations for the program design.
    b. Realized I can use an unconditional branch to infinitely loop the LED rotation
    c. Realized that I can use a if-else conditional in the initial wait loop to determine whether the LEDs should rotate or not
    d. Used register R0 to update the status of LEDs and whether to rotate them or not. Picked R0 because it can return values from procedures
2. Started to debug. The system and code worked as expected.
    a. The errors found were logic errors. Also, I realized that TST doesn't work the way I first thought.
    b. Had to rearrange procedures and restructure conditional branches

Dec 10, 2019

1. Starting part 3.
    a. Noted that timer 7 will be used
        i. Int value is 95 for timer 7 which corresponds to bit 31 of mir2
            1. So unmask value is 0x80000000
        ii. CM_PER_TIMER7_CLKCTRL = 7C
            1. Has the same enable value as timer2
        iii. PRCMCLKSEL_TIMER7 offset is 4
            1. Same value for clock of 32KHz
        iv. Timer7 base address is 0x4804A000
        v. A two second delay according to the equation from the text will have a TLDR value of 0xFFFF0000
    b. I made the algorithms simpler by excluding part 2 from it
        i. Otherwise it gets complicated to look at for me

Dec 13, 2019

1. Starting to debug the last program
    a. The logic seems sound. However, I cannot get the interrupt from the timer whatsoever. Though the button works fine.
    b. Could not figure out what the issue was, followed the template given and changed the necessary values. However, I could not find the problem with the timer initializing.
    c. Nevermind, I figured it out. I was missing the auto reload code after the button push.
    d. It works with some glitches with the button. I think this is due to the timer being much faster than me or human use. Also, probably the processor in the middle of instruction fetches so fast.