

ELEC 2220 Computer Systems
Homework #16
("Double" assignment/points)
Due: Monday, July 13

PART 1 – Microcontroller Hardware and Input/Output Ports

For this assignment, provide answers to the following questions. The required information can be found in the *ARM Cortex-M4 User Guide*, the *STM32F4xx Microcontroller Technical Reference*, and the *STM32F4-Discovery User Manual*. (All available from the course web page.)

1. List the range of memory addresses for the STM32F407 microcontroller's
 - a. Flash memory
 - b. SRAM
 - c. STM microcontroller-specific peripherals
 - d. Cortex-M4 peripherals
2. What is the base address of the block of registers that comprise the Reset and Clock Control (RCC) block?
3. What RCC register contains bits to enable clocks to drive the GPIO ports, and what is the address offset of this register within the RCC block?
4. What are the base addresses of the blocks of registers associated with
 - a. GPIOA
 - b. GPIOB
5. What are the register address offsets within the GPIO blocks for the following GPIO registers:
 - a. GPIOx_MODER
 - b. GPIOx_ODR
 - c. GPIOx_IDR
 - d. GPIOx_BSRR
6. To which GPIO pins are the four LEDs and the user button connected on the STM32F4-Discovery board?
7. Using the information from questions 2-6, write a sequence of assembly language instructions that would do the following.

Create and assemble the program to remove syntax errors.

I suggest testing it on the board, but this is not required.

You will, however, use these operations in PART 2 of the assignment.

- a. Turn on (enable) clocks to drive the two GPIO ports connected to the LEDs and button.
- b. Configure the GPIO pin connected to the Green LED as an output.
- c. Write a 1 to the pin to turn on the Green LED by writing to the output data register (ODR) of that GPIO port.
- d. Write a 0 to the pin to turn off the Green LED by writing to the corresponding bit set/reset register (BSRR).
- e. Configure the GPIO pin connected to the push button as an input.
- f. Read the state of the button by reading the input data register (IDR) of that GPIO port.
- g. Repeat the operation(s) in the last step until the button value is 0.

Submit the answers to questions 1-6 and the assembly language code for question 7.

PART 2 – Blinking LED Program

The purpose of this part is to work with simple GPIO ports and to get a project to run on the STM32F4-Discovery board, using information from Part 1.

I suggest that you build and debug the project in two steps.

Step 1 – Blinking LED

Write a program that continuously blinks the blue LED ON and OFF with a frequency of 2 seconds – ON for one second and OFF for one second. Implement this with a main program and three subroutines.

- **Subroutine 1** (InitLED) - Initialize I/O ports and LED state.
- **Subroutine 2** (LEDOnoff) – Turn the LED ON or OFF, according to a parameter passed to the subroutine in register R0.
- **Subroutine 3** (Delay) – Implement a one-second time delay (i.e. “do nothing” for about one second.)
- **Main Program** – Call the three subroutines to produce the behavior described in the problem statement.

Step 2 – Button-controlled blinking LED

Modify the program from Step 1 so that the LED state is “frozen” at its current state (ON or OFF) if the user button is being held down, and then continues blinking when the button is released. Add two additional subroutines.

- **Subroutine 4** (InitButton) – Initialize the I/O port pin connected to the User button.
- **Subroutine 5** (CheckButton) – Check the state of the user button and return 1 in register R0 if the button is pressed or 0 if the button is not pressed.

Deliverables:

1. Submit the source program. (There will be no debug window to submit.)
2. To demonstrate the operation of the program, create and provide a short video to demonstrate meeting the requirements of Parts 1 and 2. (Narrate the video to tell me what you are showing.) Alternatively, you can demonstrate the program to me in a short Zoom session.