LAB #3—Timer & Interrupts

Kizito NKURIKIYEYEZU

October 24, 2022

1 LAB OBJECTIVES

- Understand how we might use them to control digital outputs.
- Understand the limitations switch polling
- Using timers and interrupts to mitigate the limitations of switch polling
- Understand the potential problem (e.g., race conditions, shared mutable variables, etc) and potential solutions to these problem.

2 LAB EXERCISE

Four LEDs are connected to an ATMega328 MCU as follows:

- LED 1 (D1) is green and is connected to PB0 of the MCU
- LED 2 (D2) is red and is connected to PB1 of the MCU
- LED 3 (D3) is blue and is connected to PB2 of the MCU
- LED 4 (D4) is yellow and is connected to PB3 of the MCU
- A push button switch (SW1) is connected to PB4 of the MCU
- A push button switch (SW2) is connected to PB5 of the MCU

2.1 Exercise 1—blocking polling

In this exercise, you're asked to write a program that read the digital input. Your program should work as follows:

- All LEDs are OFF at the start of the program
- Once the program begins, LED1 and LED4 blink continuously at 1 Hz
- While the switch SW1 is pressed, LED2 is turned ON and stays ON until the switch SW1 is released.
- When SW1 is pressed and then released, turn ON LED3 for one second, and turn it OFF

To complete this exercise, you should do the following:

- Sketch a flowchart of the program
- Write, debug and simulate the program using Proteus VSM
- Does your program work as intended? If not, describe the problem, if any, and explain why this is the case.

NOTE—At the end of this exercise, you should note that:

- While the digital input SW1 is pressed, this code blocks the CPU in a tight polling loop. This is also known as spinning.
- Switch pooling is not power or CPU efficient.
- Once the switch is pressed, the CPU is blocked and cannot do anything else.
- While it is not always bad practice to monitor micro-controller peripheral hardware this way (especially when the polling duration is short), this is highly discouraged is several case because the switch presses have an indeterminate time. Thus, spinning becomes wasteful of CPU time and most of all, power.

2.2 Exercise 2—non-blocking polling

Although switch polling is simple, it has several disadvantages:

- It uses a lot of CPU resource even when there are no changes in the inputs. In reality, most modern embedded systems are often idle and polling would unnecessary drain their battery.
- The loop cycle time is variable (e.g. due to conditional statements). Therefore the sampling interval of the inputs is not constant and is said to jitter.
- If the loop cycle is ever too long, input changes could be missed and data lost. The delay gets longer as more code is added.

This exercise aims to address some of these issues. It uses timers to create non-blocking delays and interrupts to asynchronously detect when a switch is pressed and released.

2.2.1 reading materials and exercise

before attempting the lab, please review the lecture notes on interrupts and timers. You should also read the following online resources:

- Read to review the basics of AVR timer
- Read this online resource to understand the basics of AVR interrupts
- Read this online resource to understand external hardware interrupts in AVR ATmega32

2.3 Lab exercise

Complete the previously completed exercise using interrupts and and timers. It is helpful to keep the following in mind:

- To detect whether a switch has been pressed or not, please use external interrupts. It is recommended to revised discussion in the lecture about this topic. You should also read this online resource to understand external hardware interrupts in AVR ATmega32
- From your readings and the discussion in the class, it should be clear that that not all pin support external interrupts. Thus, you should re-arrange the connection of the two push button SW1 and SW2 to take advantage of external interrupts.
- You should not use the built-in delay $\_delay_ms()$ function when creating delays. Instead, you need to create no-blocking delays using a timer. Please review the lecture slides and read to review the basics of AVR timer to understand how to create a delay with a timer.

3 Lab submission

- I will check your work (Proteus simulation and the source code) at the beginning of the next lecturer.
- You can do the lab in a group of no more than 3 students
- The lab is handed out on October 24, 2022 and is due on **November 1, 2022**.
- Because the course will only lasts for 11 weeks, this is a hard deadline and **late submissions are not accepted after the deadline**.
- If you're not available in the lab, you will get a zero for the assignment (unless you can provide a documented evidence for your absence)