Lab 2: Morse Code Decoder

ESE3500: Embedded Systems & Microcontroller Laboratory
University of Pennsylvania

In this document, you'll fill out your responses to the questions listed in the Lab 2 Manual. Please fill out your name and link your Github repository below to begin. Be sure that your code on the repo is up-to-date before submission!

For all the questions that require a video, provide a link to the video (e.g. youtube, google drive, etc.).

Student Name: Lixuan Luo
Pennkey: 58944286
GitHub Repository: https://github.com/ese3500/lab2-luoli

1. & 2.

```
void Initialize(){
    // Outputs
    DDRB |= (1 << DDB1);
    DDRB |= (1 << DDB2);
    DDRB |= (1 << DDB3);
    DDRB |= (1 << DDB4);
}

int main(void)
{
    Initialize();
        while (1)
        {
            PORTB |= (1 << PORTB1);
            PORTB |= (1 << PORTB2);
            PORTB |= (1 << PORTB3);
            PORTB |= (1 << PORTB4);
        }
}
```

Figure 1. Code for Part A1

```
void Initialize(){
    // Outputs
    DDRB |= (1 << DDB1);

    // Inputs
    DDRD &= ~(1 << DDD7);
}

int main(void)
{
    Initialize();
    while (1)
    {
        // Toggles LED
        if(PIND & (1 << PIND7)){
            PORTB |= (1 << PORTB1);
        } else{
            PORTB &= ~(1 << PORTB1);
        }

    }
}
```

Figure 2. Code for Part B2

3.

```c
volatile int counter = 0;

void Initialize(){
    // Outputs
    DDRB |= (1 << DDB1);
    DDRB |= (1 << DDB2);
    DDRB |= (1 << DDB3);
    DDRB |= (1 << DDB4);

    // Inputs
    DDRD &= ~(1 << DDD7);
}
```

```c
int main(void)
{
    Initialize();
    while (1)
    {
        if(PIND & (1 << PIND7)){
            counter++;
            _delay_ms(5000);
        }
        if(counter % 4 == 0){
            PORTB |= (1 << PORTB1);
            PORTB &= ~(1 << PORTB4);
        } else if(counter % 4 == 1){
            PORTB |= (1 << PORTB2);
            PORTB &= ~(1 << PORTB1);
        } else if(counter % 4 == 2){
            PORTB |= (1 << PORTB3);
            PORTB &= ~(1 << PORTB2);
        } else{
            PORTB |= (1 << PORTB4);
            PORTB &= ~(1 << PORTB3);
        }

    }
}
```

Figure 3a & 3b. Code for Part B3

4.

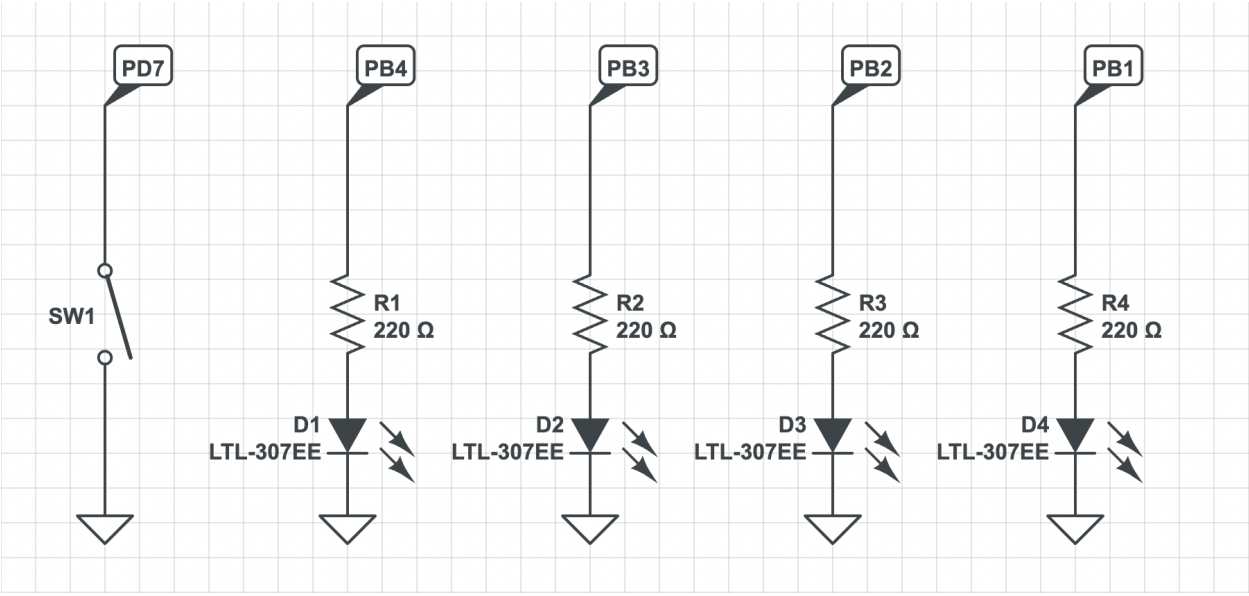

Figure 4. Circuit for part B4

5. One advantage of using interrupts in this scenario is that it allows the CPU to simply ignore the device and perform other tasks until the actual interrupt occurs. On the other hand, if polling is used, since the CPU would have to constantly check for a device input, it can provide a very quick response time and also ensures that other parts of the program won't be messed up since nothing is ever "interrupted."

6. For a 16MHz clock, it essentially means that there are 16000000 ticks / second. Performing some calculations reveals that for 30ms, there are $\frac{16000000\ ticks}{1\ second} \times \frac{1\ second}{1000\ ms} \times 30 = 480000\ ticks$; for 200ms, there are $\frac{16000000\ ticks}{1\ second} \times \frac{1\ second}{1000\ ms} \times 200 = 3200000\ ticks$; and for 400ms, there are $\frac{16000000\ ticks}{1\ second} \times \frac{1\ second}{1000\ ms} \times 400 = 6400000\ ticks$.

7. A prescaler essentially allows us to "slow down" the clock on the microcontroller and therefore work with a wider range of frequencies. Since the largest clock on the AtMega328P is only 16 bits, it means that it could only count up to 2^16 - 1 "ticks" before it overflows and starts over – this is merely 4ms and it makes it very difficult to for us to keep track of the time that has passed in 4ms intervals (even if we do have an overflow counter). However, the prescaler (whether clock or system) essentially "slows down" the clock by counting 1 tick for every x specified ticks (x usually being 8, 64, 256, or 1024). This way, we are able to work with larger units of time, such as 4.1943 s intervals if we scale it by 1024, as now only 16000000 / 1024 ticks occur every second and we are still able to keep track of 2^16 - 1 ticks.

8. https://youtu.be/0_aqf-dxZdY (Note: I am not sure why some of the LED blinks were not captured in the video, but they definitely blinked in reality!) Furthermore, note that although the time intervals for each press (to represent the dash and dots) remained as the same time interval as mentioned in the lab, I changed the amount of time between presses to indicate a space from 400ms to 1000ms to ensure that i will have time to properly think of what to press next.

9. "Someday I will rule you all"

10. https://youtu.be/pAInPOge-ZY