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: Isabel Song
Pennkey: isong25
GitHub Repository: lab2-isong25

1. Code to turn all LEDs on:

```
#include <avr/io.h>
void q1(void) {
    DDRB |= 0x1E;
    PORTB |= 0x1E;
}
```

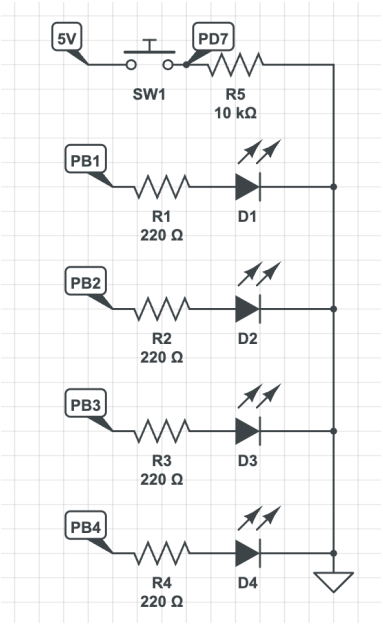
2. To turn on LED when button pressed, off when released:

```
void q2(void) {
    DDRD &= ~(1<<DDD7);
    PORTD |= (1<<PORTD7);
    DDRB |= (1<<DDB1);
    while(1) {
        if (PIND & (1<<PIND7)) { // button pressed
            PORTB |= (1<<PORTB1);
        } else { // button not pressed
            PORTB &= ~(1<<PORTB1);
        }
    }
}
```

3. To alternate which of four LEDs is lit with successive button presses:

```c
void q3(void) {
    int ctr = 0;
    DDRD &= ~(1<<DDD7); // configure input pin
    PORTD |= (1<<PORTD7); // pull high
    DDRB |= 0x1E; // configure output pins
    PORTB = 0x00;
    PORTB |= (1<<PORTB1); // start first LED on
    while (1) {
        while (!(PIND & (1 << PIND7))) {}
        ctr = (ctr + 1) % 4;
        switch(ctr) {
            case 0:
                PORTB = 0x02;
                break;
            case 1:
                PORTB = 0x04;
                break;
            case 2:
                PORTB = 0x08;
                break;
            case 3:
                PORTB = 0x10;
                break;
        }
        _delay_ms( ms: 500);
    }
}
```

4. CircuitLab schematic:



5. One disadvantage of using interrupts over polling was that polling yields a consistently faster response than interrupts. One advantage of using interrupts was that all the CPU did not need to focus on constantly checking and comparing the state of the input pin in order to determine when the output pin should be activated; instead, the CPU was just interrupted when the state changed.

6. 30ms=480,000 ticks (0 to 479,999); 200ms=3,200,000 ticks (0 to 3,199,999); 400ms=6,400,000 ticks (0 to 6,399,999)
7. Using a prescaler allows us to work with higher frequencies without the use of additional (internal or external) resources necessitated by timer overflows. For example if we only have an 8-bit timer available, it can only count to 255 and any frequencies higher than 256 without prescaling may overflow many times just in setting the frequency. If we instead use a prescaler that allows the frequency to be counted within the available bits, scaling down the desired frequency, we can handle the frequency setting without using a larger register or keeping track of overflows in addition to the clock counting each second.
8. Link to demo video: https://drive.google.com/file/d/13RoR32hikZg2VzPGOahuO77yqDa1Mom1/view?usp=sharing

In order to make tapping out the characters more consistent, the bounds for character time were adjusted to 500 ms threshold between dot and dash and 1000 ms for space since the originals were too short to be tapped out manually with consistency.

9. “Someday I will rule you all”