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: Victor Gao
Pennkey: vgao
GitHub Repository: lab2_vgao

1.

```c
int main(void)
{
    DDRB |= (1<<DDB1); //Configure PB1, PB2, PB3, and PB4 to output
    DDRB |= (1<<DDB2);
    DDRB |= (1<<DDB3);
    DDRB |= (1<<DDB4);

    while (1)
    {
        PORTB |= (1<<PORTB1); //Drive PB1, PB2, PB3, PB4 to high
        PORTB |= (1<<PORTB2);
        PORTB |= (1<<PORTB3);
        PORTB |= (1<<PORTB4);
    }
}
```

2.

```c
void Initialize()
{
    DDRD &= ~(1<<DDD7); //Configure PD7 as input
    DDRB |= (1<<DDB1); //Configure PB1, PB2, PB3, and PB4 to output
    DDRB |= (1<<DDB2);
    DDRB |= (1<<DDB3);
    DDRB |= (1<<DDB4);
}
```

```c
int main(void)
{
    Initialize();

    while (1)
    {
        if (PIND & (1<<PIND7)) //If value at D7 is high (button depressed)
        {
            PORTB |= (1<<PORTB1); //Set PB1 to high
        }
        else
        {
            PORTB &= ~(1<<PORTB1); //else set PB1 to low
        }
    }
}
```

3.

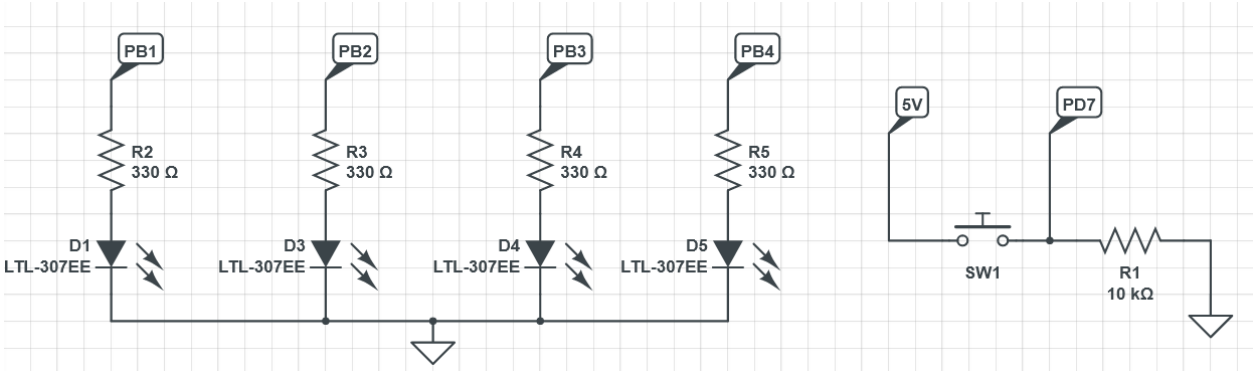
```c
void Initialize()
{
    DDRD &= ~(1<<DDD7); //Configure PD7 as input
    DDRB |= (1<<DDB1); //Configure PB1, PB2, PB3, and PB4 to output
    DDRB |= (1<<DDB2);
    DDRB |= (1<<DDB3);
    DDRB |= (1<<DDB4);
}

int main(void)
{
    Initialize();

    int i = 0; //counter variable to keep track of cycle

    while (1)
    {
        if (PIND & (1<<PIND7)){ //If value at D7 is high (button depressed)
            i++;
        }
        if (i%4 == 0){
            PORTB |= (1<<PORTB1); //Set PB1 to high
            PORTB &= ~(1<<PORTB4); //Set PB4 to low
        }
        else if(i%4 == 1){
            PORTB |= (1<<PORTB2); //Set PB2 to high
            PORTB &= ~(1<<PORTB1); //Set PB1 to low
        }
        else if(i%4 == 2){
            PORTB |= (1<<PORTB3); //Set PB3 to high
            PORTB &= ~(1<<PORTB2); //Set PB2 to low
        } else {
            PORTB |= (1<<PORTB4); //Set PB4 to high
            PORTB &= ~(1<<PORTB3); //Set PB3 to low
        }
        _delay_ms(5000); //delay for 5s
    }
}
```

4.



5.
   - Advantage: the CPU is only interrupted when the button is pressed, as opposed to polling, which perpetually checking if the button is pressed. Thus using interrupts is much more efficient.
   - Disadvantage: if the button is depressed for a long period of time, then the program will be interrupted for a long time - this means that the main program will be at a standstill, which is undesirable if other lines of code need to be run. This is opposed to polling, in which we can simultaneously check that the button is depressed and run other lines of code in the main() section.
6.

$$16\,MHz \;=\; \frac{1}{16\times10^{6}}\;s\;(tick\;duration)$$

   - For 30ms
     - # of ticks = $30 \times 10^{-3} \cdot 16 \times 10^{6} = 480,000$ ticks
   - For 200ms
     - # of ticks = $200 \times 10^{-3} \cdot 16 \times 10^{6} = 3,200,000$ ticks
   - For 400ms
     - # of ticks = $400 \times 10^{-3} \cdot 16 \times 10^{6} = 4,800,000$ ticks

7. A prescaler allows us to work with frequencies beyond the pre-determined frequency of the system clock. For example, if we had a 16 MHz clock connected to a timer with no prescaler, the timer would count at the same frequency of the clock (16 million ticks per second). This would be way too fast in most scenarios, so we can use a cock prescaler to scale down the frequency. For example, a prescaler of 256 would decrease the frequency to 62.5 kHz - this would be slow enough to blink an LED every second (using a 16 bit timer). Thus, using various prescaler values would allow us to obtain a frequency we desire.
8. Link to Video
   - **Modified dash and space durations**
     - Dash: from 200ms to 1000ms
     - Space: anything larger than 1000ms
     - Dot: still from 30ms to 200ms
9. SOMEDAY I WILL RULE YOU ALL
10. Link to Video
   - The code in the repository currently has the word “VICTOR”. The word can be changed by updating the length and elements of the “word” char array
   - Dash and dot durations are modified, see comments in bonus.c file