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: Rafael Sakamoto
Pennkey: kenzosak
GitHub Repository: lab-2-morse-kenzoSakamoto

1.

```c
#include <avr/io.h>


int main(void) {

    // Sets pins PB1-4 as output
    DDRB |= 0x1E;

    // Turns pins PB1-4 HIGH
    PORTB |= 0x1E;

    return 0;
}
```

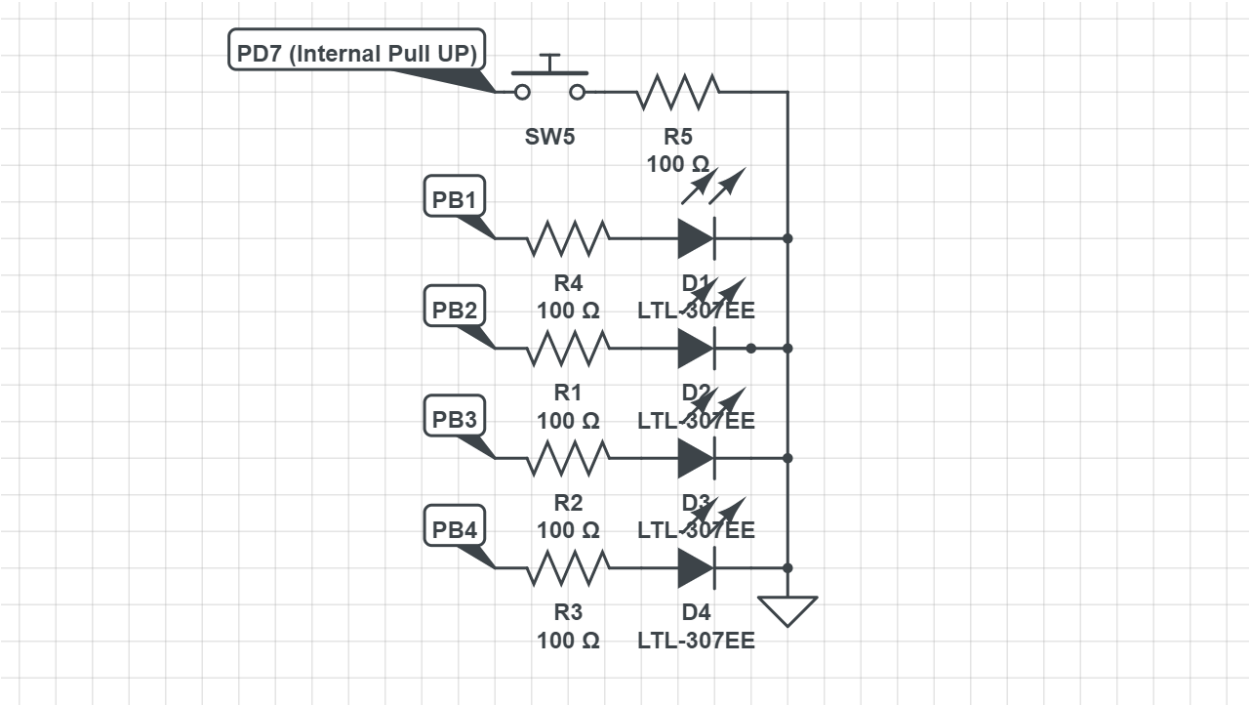
2.

```c
int main(void) {

    // Sets pins PB1-4 as output and PD7 as input
    DDRB |= 0x1E;
    DDRD &= ~(1 << DDD7);

    // Enable pull-up resistor for PD7
    PORTD |= (1 << PORTD7);

    // Turn PB1 on if PD7 is LOW and off if PD7 is HIGH (active low)
    while (1) {
        if (PIND & (1 << PIND7)) PORTB = 0x0;
        else PORTB = 0x02;
    }

    return 0;
}
```

3.

```c
int main(void) {

    // Sets pins PB1-4 as output and PD7 as input
    DDRB |= 0x1E;
    DDRD &= ~(1 << DDD7);

    // Enable pull-up resistor for PD7
    PORTD |= (1 << PORTD7);


    // Keep track of the LED that should light up next
    int num_pressed = 3;

    // Loop to turn off LEDs if button not pressed and turn on the right one if pressed
    while (1) {
        if (PIND & (1 << PIND7)) PORTB = 0x0;
        else {
            num_pressed ++;
            num_pressed = (num_pressed) % 4;
            switch (num_pressed) {
                case 0:
                    PORTB |= (1 << PORTB1);
                    break;
                case 1:
                    PORTB |= (1 << PORTB2);
                    break;
                case 2:
                    PORTB |= (1 << PORTB3);
                    break;
                default:
                    PORTB |= (1 << PORTB4);
            }

            // Wait for button to be unpressed and a bit of delay for debounce issues
            while (!(PIND & (1 << PIND7))) _delay_ms(10);
        }
    }

    return 0;
}
```

4.



5.

Advantage: By using interrupts instead of polling in this task, we can do useful work while the button is not changing states. In polling, we have to constantly check the state of the input pin. On the other hand, using interrupts the function is only called when there is a change in state in the input pin. While we could also potentially do other work in the polling loop, we would spend time checking the pin every cycle which is not very efficient when compared to using interrupts.

Disadvantage: On top of being more complex to implement, when using interrupts we have to keep in mind that we do not know when they will be "triggered". Thus, we have to design our code around the possibility of having our "main" code stopped and having it interrupted at any time by the handler (although we can also disable interrupts at certain critical times if needed).

6. $\#ticks = \Delta t \cdot f$
   - 30ms: $30 \cdot 10^{-3} \times 16 \cdot 10^{6} = 4.8 \cdot 10^{5}\ ticks$
   - 200ms: $200 \cdot 10^{-3} \times 16 \cdot 10^{6} = 3.2 \cdot 10^{6}\ ticks$
   - 400ms: $400 \cdot 10^{-3} \times 16 \cdot 10^{6} = 6.4 \cdot 10^{6}\ ticks$

7. A prescaler can be useful because our variables can only hold up to a certain number of ticks (e.g. an unsigned int_16 can only hold up to $2^{16} - 1$). Thus, depending on the time scale we are interested in, the number of ticks may not fit in our variable. For instance, in this lab, we are interested in time intervals of around a few hundred milliseconds. If we didn't use a prescaler, our variable would only be able to count up to around 4ms. So we can see how a prescaler is useful here: by scaling down our counting (i.e. only incrementing the counter every 1024 clock ticks, for example), we can now count up to around 4.2s. Thus, this basically means that we can "adjust" the count frequency based on the frequency of our microcontroller to our needs in order to be able to count to the range we would like to.
8. Video link: https://photos.app.goo.gl/p6T1MuMRdrKaaDLY9
9. Decoded message: "SOME DAY I WILL RULE YOU ALL"
10. Video link: https://photos.app.goo.gl/sHDvAGrpJHzEqJSTA