Joseph Daniel Moreno, U#60156204

Embedded Systems – Assignment 7

Problem Description:
Use the XC8 compiler in MPLAB X IDE to control a PWM signal for operating a motor. The design must allow the user to control the duty cycle and it must include the components: 4x4 keypad, 2x16 LCD screen, and a motor (DC, stepper, or servo).

Pseudocode:

```
pwm_freq = ((500000 / 156) / 64);          // 50 Hz
pwm_period = (1 / pwm_freq);               // Clock cycle time

keypad[11] = KEYSTROKE[11];                // Keypad buttons.
motor_dc = (pwm_period * 0) seconds;       // The motor’s duty cycle.

lcd = “Assignment 7”;                      // What the LCD is displaying.

while(true) {
        if (keypad[0]) {
                motor_dc = (pwm_period * 0) seconds;
                lcd = “Duty Cycle = 0%”;
        }

        if (keypad[1]) {
                motor_dc = (pwm_period * 0.1) seconds;
                lcd = “Duty Cycle = 10%”;
        }

        if (keypad[2]) {
                motor_dc = (pwm_period * 0.2) seconds;
                lcd = “Duty Cycle = 20%”;
        }

        if (keypad[3]) {
                motor_dc = (pwm_period * 0.3) seconds;
                lcd = “Duty Cycle = 30%”;
        }

        if (keypad[4]) {
                motor_dc = (pwm_period * 0.4) seconds;
                lcd = “Duty Cycle = 40%”;
        }

        if (keypad[5]) {
                motor_dc = (pwm_period * 0.5) seconds;
                lcd = “Duty Cycle = 50%”;
        }
```

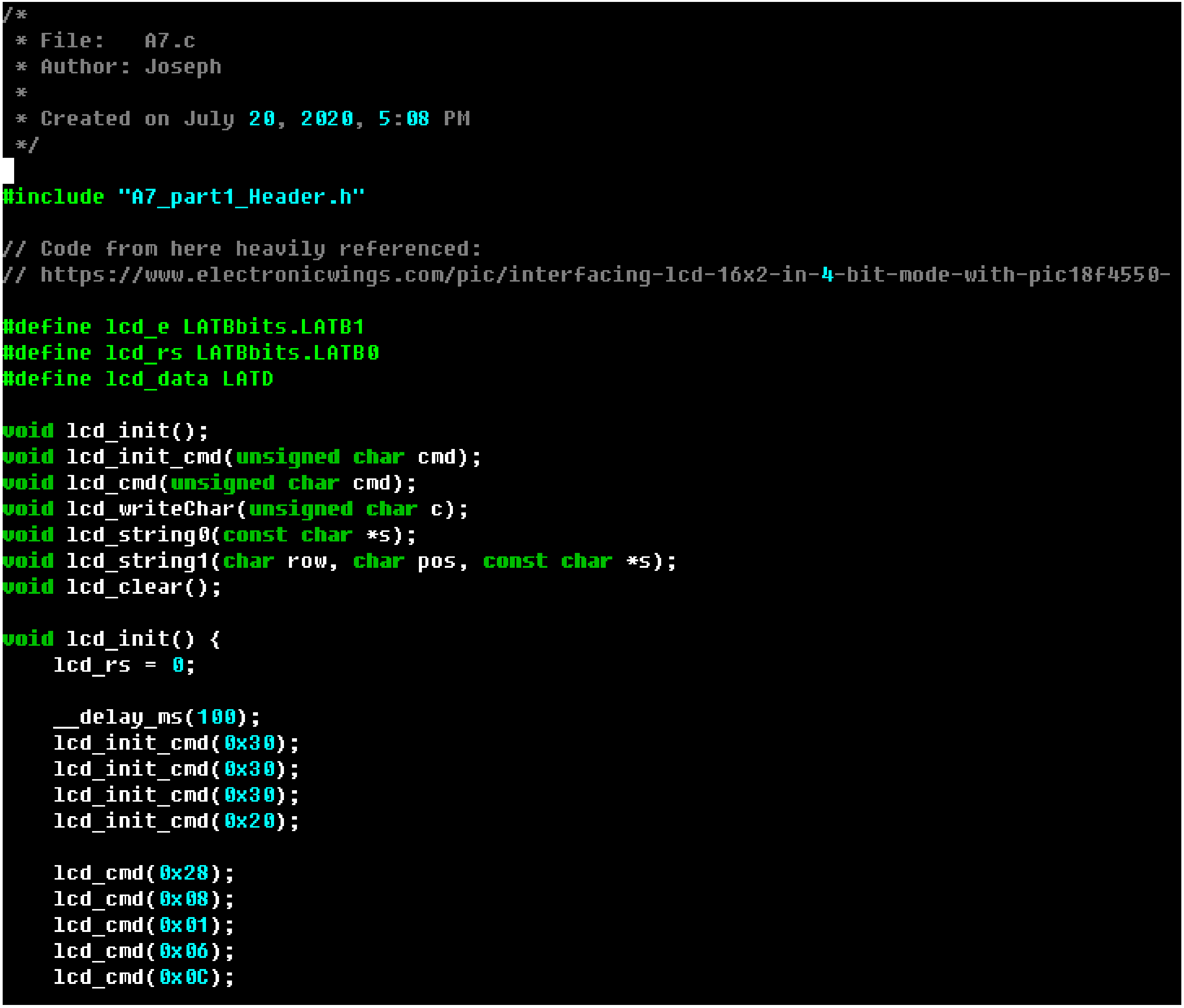
Demo Video Link:

https://youtu.be/yJZos7zkTfs

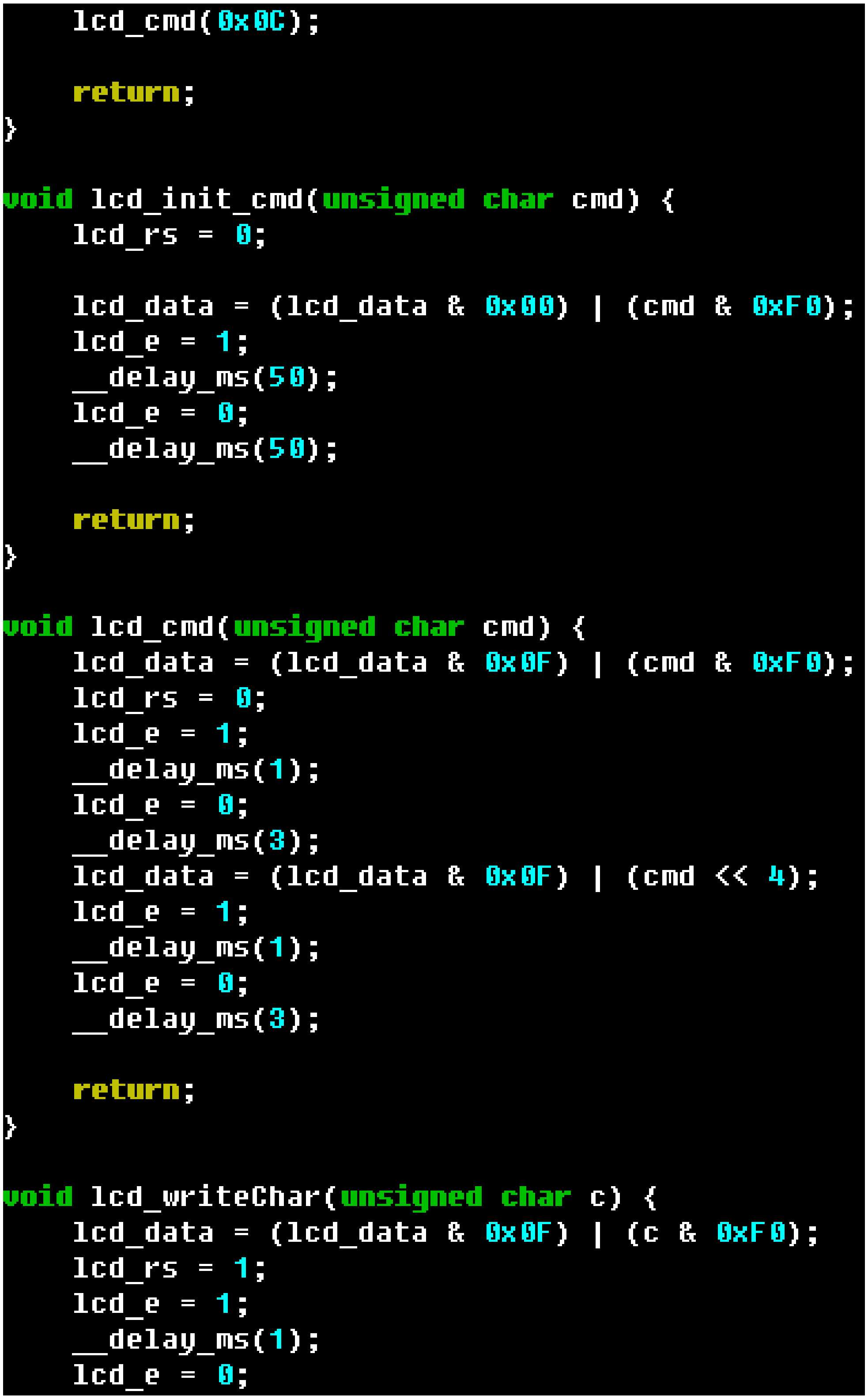
```
        if (keypad[6]) {
                motor_dc = (pwm_period * 0.6) seconds;
                lcd = “Duty Cycle = 60%”;
        }

        if (keypad[7]) {
                motor_dc = (pwm_period * 0.7) seconds;
                lcd = “Duty Cycle = 70%”;
        }

        if (keypad[8]) {
                motor_dc = (pwm_period * 0.8) seconds;
                lcd = “Duty Cycle = 80%”;
        }

        if (keypad[9]) {
                motor_dc = (pwm_period * 0.9) seconds;
                lcd = “Duty Cycle = 90%”;
        }

        if (keypad[10]) {
                motor_dc = (pwm_period * 1) seconds;
                lcd = “Duty Cycle = 100%”;
        }
}
```

Joseph Daniel Moreno, U#60156204

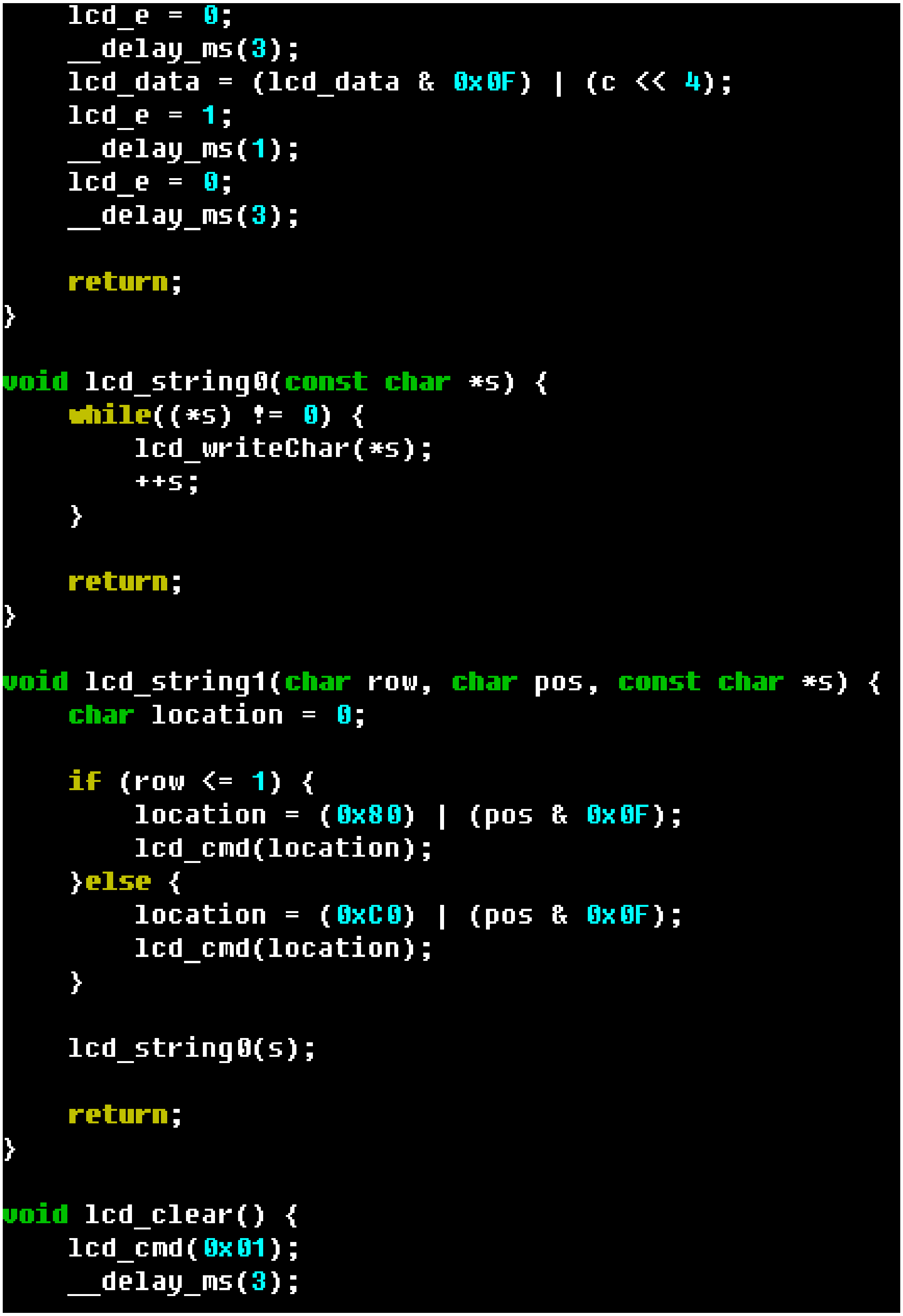
C Code (Servo Project):
Code from "ES_A7_part1.c"; included in .zip file.



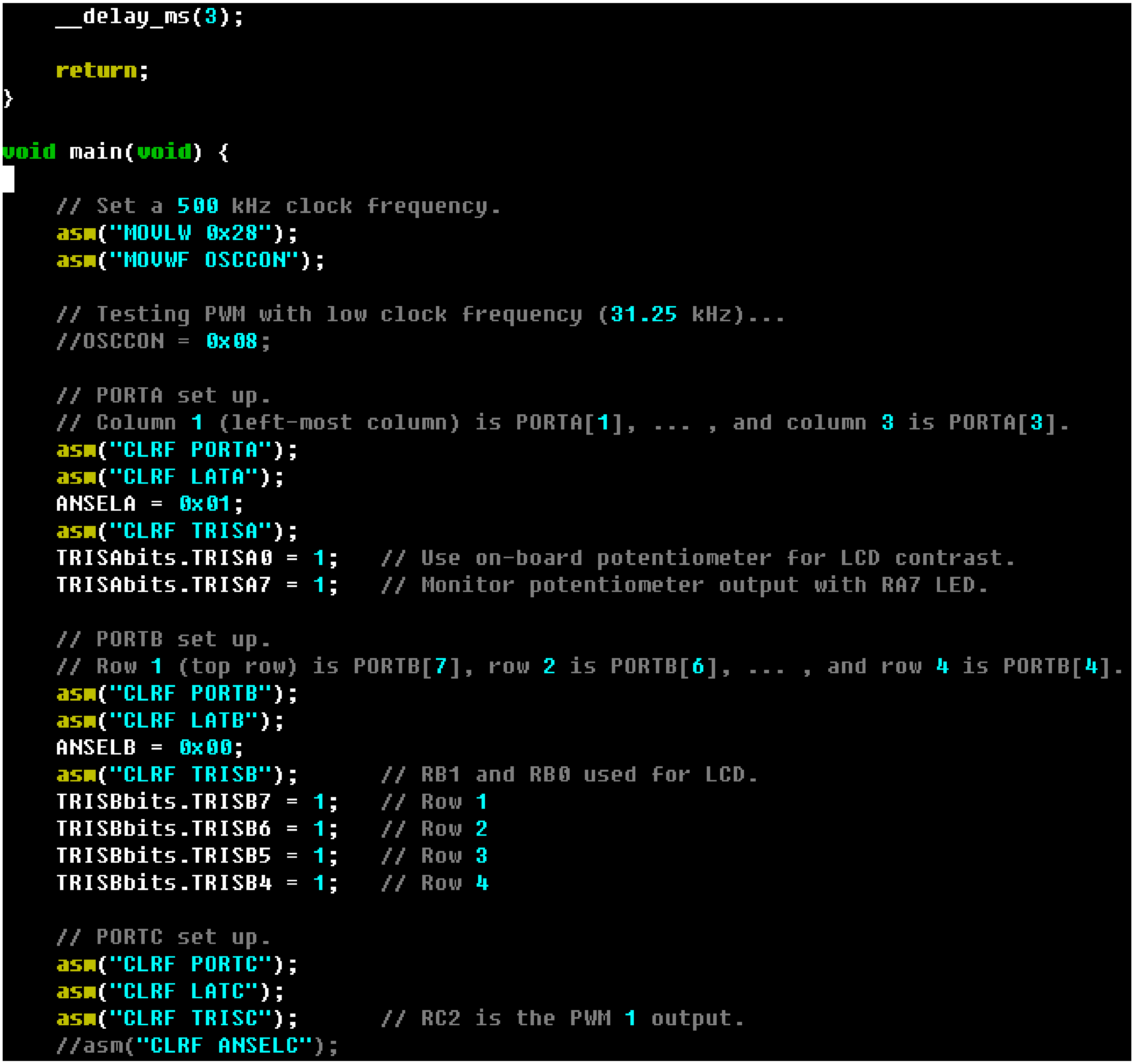
```
/*
 * File:   A7.c
 * Author: Joseph
 *
 * Created on July 20, 2020, 5:08 PM
 */

#include "A7_part1_Header.h"

// Code from here heavily referenced:
// https://www.electronicwings.com/pic/interfacing-lcd-16x2-in-4-bit-mode-with-pic18f4550-

#define lcd_e LATBbits.LATB1
#define lcd_rs LATBbits.LATB0
#define lcd_data LATD

void lcd_init();
void lcd_init_cmd(unsigned char cmd);
void lcd_cmd(unsigned char cmd);
void lcd_writeChar(unsigned char c);
void lcd_string0(const char *s);
void lcd_string1(char row, char pos, const char *s);
void lcd_clear();

void lcd_init() {
    lcd_rs = 0;

    __delay_ms(100);
    lcd_init_cmd(0x30);
    lcd_init_cmd(0x30);
    lcd_init_cmd(0x30);
    lcd_init_cmd(0x20);

    lcd_cmd(0x28);
    lcd_cmd(0x08);
    lcd_cmd(0x01);
    lcd_cmd(0x06);
    lcd_cmd(0x0C);
```



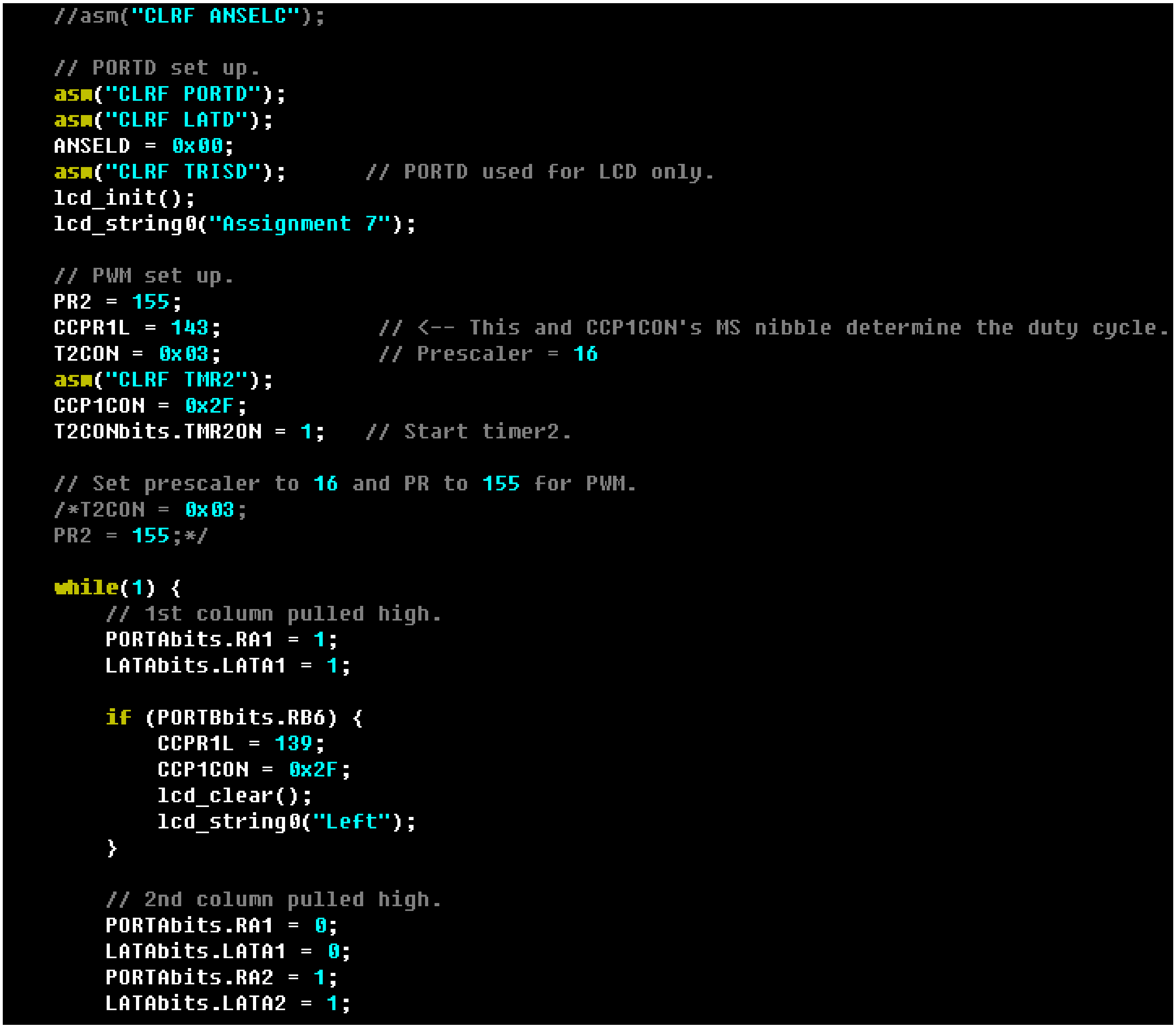
```
    lcd_cmd(0x0C);

    return;
}

void lcd_init_cmd(unsigned char cmd) {
    lcd_rs = 0;

    lcd_data = (lcd_data & 0x00) | (cmd & 0xF0);
    lcd_e = 1;
    __delay_ms(50);
    lcd_e = 0;
    __delay_ms(50);

    return;
}

void lcd_cmd(unsigned char cmd) {
    lcd_data = (lcd_data & 0x0F) | (cmd & 0xF0);
    lcd_rs = 0;
    lcd_e = 1;
    __delay_ms(1);
    lcd_e = 0;
    __delay_ms(3);
    lcd_data = (lcd_data & 0x0F) | (cmd << 4);
    lcd_e = 1;
    __delay_ms(1);
    lcd_e = 0;
    __delay_ms(3);

    return;
}

void lcd_writeChar(unsigned char c) {
    lcd_data = (lcd_data & 0x0F) | (c & 0xF0);
    lcd_rs = 1;
    lcd_e = 1;
    __delay_ms(1);
    lcd_e = 0;
```



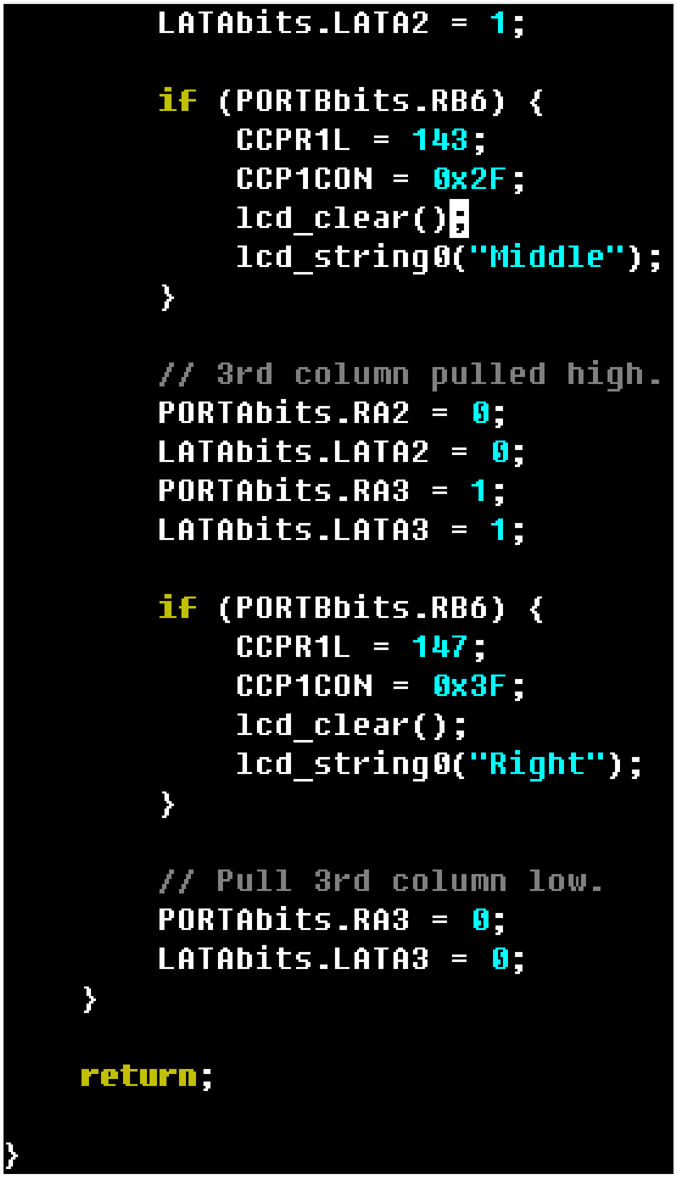
```
    lcd_e = 0;
    __delay_ms(3);
    lcd_data = (lcd_data & 0x0F) | (c << 4);
    lcd_e = 1;
    __delay_ms(1);
    lcd_e = 0;
    __delay_ms(3);

    return;
}

void lcd_string0(const char *s) {
    while((*s) != 0) {
        lcd_writeChar(*s);
        ++s;
    }

    return;
}

void lcd_string1(char row, char pos, const char *s) {
    char location = 0;

    if (row <= 1) {
        location = (0x80) | (pos & 0x0F);
        lcd_cmd(location);
    }else {
        location = (0xC0) | (pos & 0x0F);
        lcd_cmd(location);
    }

    lcd_string0(s);

    return;
}

void lcd_clear() {
    lcd_cmd(0x01);
    __delay_ms(3);
```



```
    __delay_ms(3);

    return;
}

void main(void) {

    // Set a 500 kHz clock frequency.
    asm("MOVLW 0x28");
    asm("MOVWF OSCCON");

    // Testing PWM with low clock frequency (31.25 kHz)...
    //OSCCON = 0x08;

    // PORTA set up.
    // Column 1 (left-most column) is PORTA[1], ... , and column 3 is PORTA[3].
    asm("CLRF PORTA");
    asm("CLRF LATA");
    ANSELA = 0x01;
    asm("CLRF TRISA");
    TRISAbits.TRISA0 = 1;   // Use on-board potentiometer for LCD contrast.
    TRISAbits.TRISA7 = 1;   // Monitor potentiometer output with RA7 LED.

    // PORTB set up.
    // Row 1 (top row) is PORTB[7], row 2 is PORTB[6], ... , and row 4 is PORTB[4].
    asm("CLRF PORTB");
    asm("CLRF LATB");
    ANSELB = 0x00;
    asm("CLRF TRISB");      // RB1 and RB0 used for LCD.
    TRISBbits.TRISB7 = 1;   // Row 1
    TRISBbits.TRISB6 = 1;   // Row 2
    TRISBbits.TRISB5 = 1;   // Row 3
    TRISBbits.TRISB4 = 1;   // Row 4

    // PORTC set up.
    asm("CLRF PORTC");
    asm("CLRF LATC");
    asm("CLRF TRISC");      // RC2 is the PWM 1 output.
    //asm("CLRF ANSELC");
```



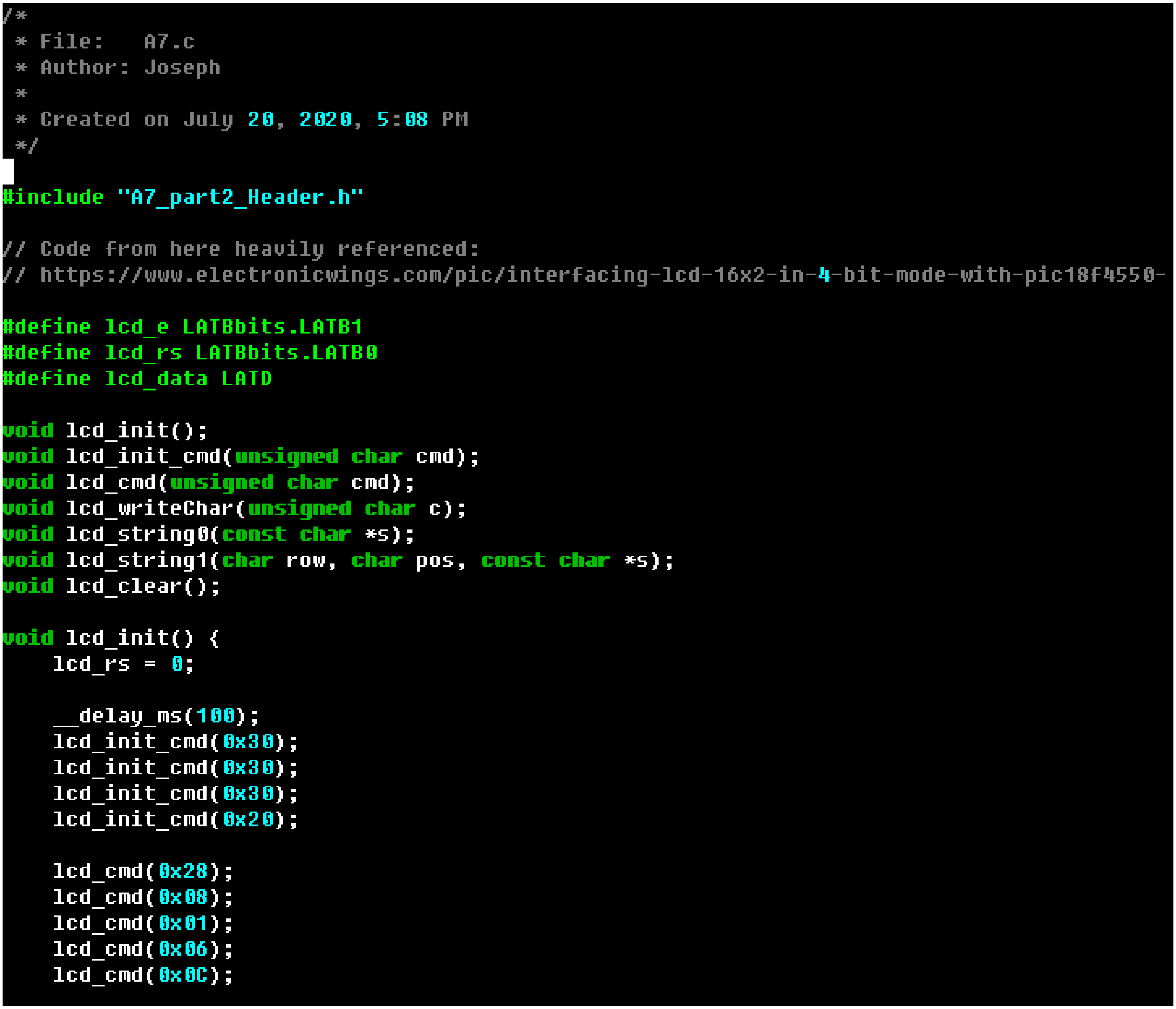
```
    //asm("CLRF ANSELC");

    // PORTD set up.
    asm("CLRF PORTD");
    asm("CLRF LATD");
    ANSELD = 0x00;
    asm("CLRF TRISD");      // PORTD used for LCD only.
    lcd_init();
    lcd_string0("Assignment 7");

    // PWM set up.
    PR2 = 155;
    CCPR1L = 143;            // <-- This and CCP1CON's MS nibble determine the duty cycle.
    T2CON = 0x03;            // Prescaler = 16
    asm("CLRF TMR2");
    CCP1CON = 0x2F;
    T2CONbits.TMR2ON = 1;   // Start timer2.

    // Set prescaler to 16 and PR to 155 for PWM.
    /*T2CON = 0x03;
    PR2 = 155;*/

    while(1) {
        // 1st column pulled high.
        PORTAbits.RA1 = 1;
        LATAbits.LATA1 = 1;

        if (PORTBbits.RB6) {
            CCPR1L = 139;
            CCP1CON = 0x2F;
            lcd_clear();
            lcd_string0("Left");
        }

        // 2nd column pulled high.
        PORTAbits.RA1 = 0;
        LATAbits.LATA1 = 0;
        PORTAbits.RA2 = 1;
        LATAbits.LATA2 = 1;
```



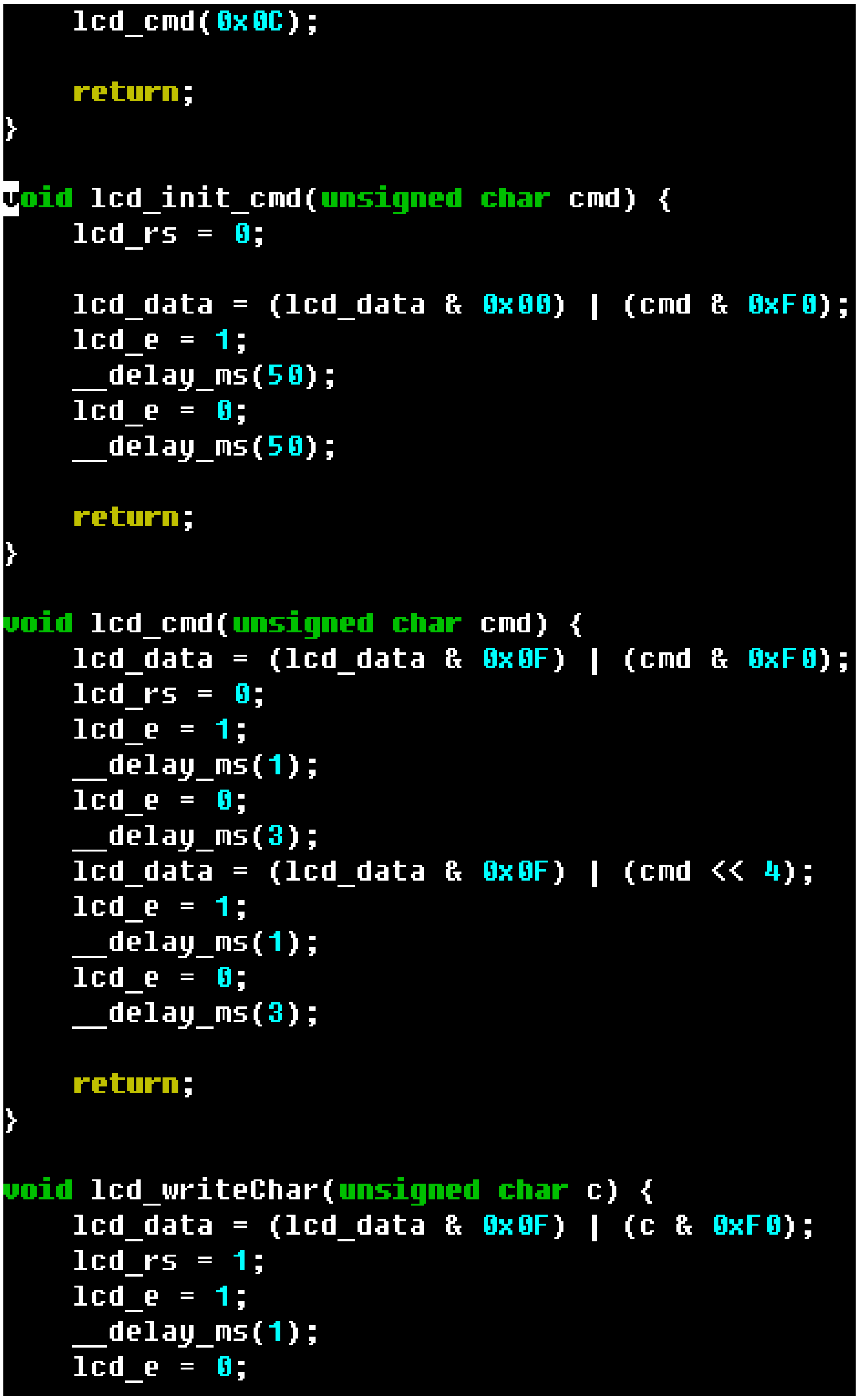
```c
        LATAbits.LATA2 = 1;

        if (PORTBbits.RB6) {
            CCPR1L = 143;
            CCP1CON = 0x2F;
            lcd_clear();
            lcd_string0("Middle");
        }

        // 3rd column pulled high.
        PORTAbits.RA2 = 0;
        LATAbits.LATA2 = 0;
        PORTAbits.RA3 = 1;
        LATAbits.LATA3 = 1;

        if (PORTBbits.RB6) {
            CCPR1L = 147;
            CCP1CON = 0x3F;
            lcd_clear();
            lcd_string0("Right");
        }

        // Pull 3rd column low.
        PORTAbits.RA3 = 0;
        LATAbits.LATA3 = 0;
    }

    return;

}
```

Joseph Daniel Moreno, U#60156204

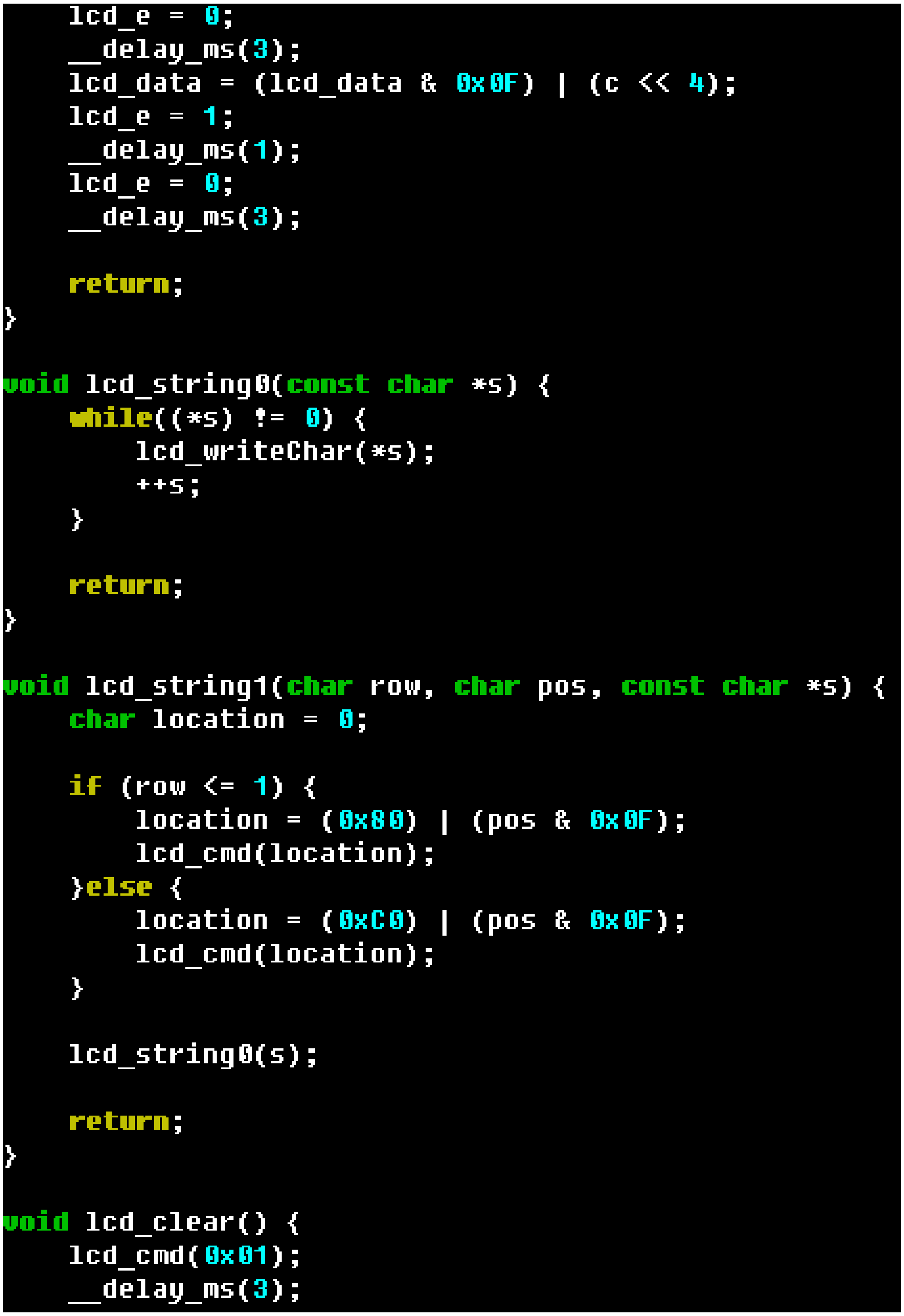
C Code (PWM Through LED Project):
Code from "ES_A7_part2.c"; included in .zip file.



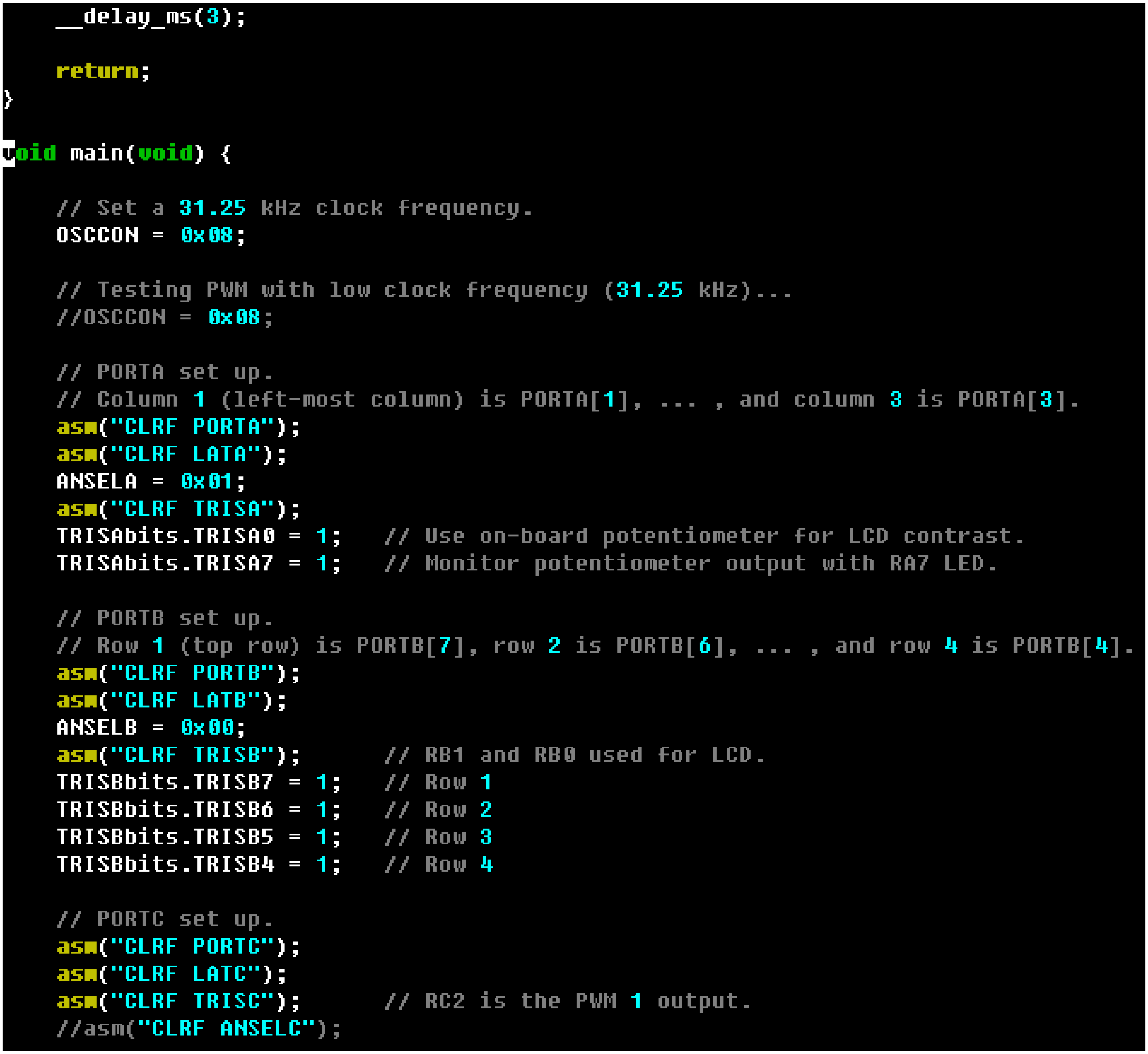
```
/*
 * File:   A7.c
 * Author: Joseph
 *
 * Created on July 20, 2020, 5:08 PM
 */

#include "A7_part2_Header.h"

// Code from here heavily referenced:
// https://www.electronicwings.com/pic/interfacing-lcd-16x2-in-4-bit-mode-with-pic18f4550-

#define lcd_e LATBbits.LATB1
#define lcd_rs LATBbits.LATB0
#define lcd_data LATD

void lcd_init();
void lcd_init_cmd(unsigned char cmd);
void lcd_cmd(unsigned char cmd);
void lcd_writeChar(unsigned char c);
void lcd_string0(const char *s);
void lcd_string1(char row, char pos, const char *s);
void lcd_clear();

void lcd_init() {
    lcd_rs = 0;

    __delay_ms(100);
    lcd_init_cmd(0x30);
    lcd_init_cmd(0x30);
    lcd_init_cmd(0x30);
    lcd_init_cmd(0x20);

    lcd_cmd(0x28);
    lcd_cmd(0x08);
    lcd_cmd(0x01);
    lcd_cmd(0x06);
    lcd_cmd(0x0C);
```



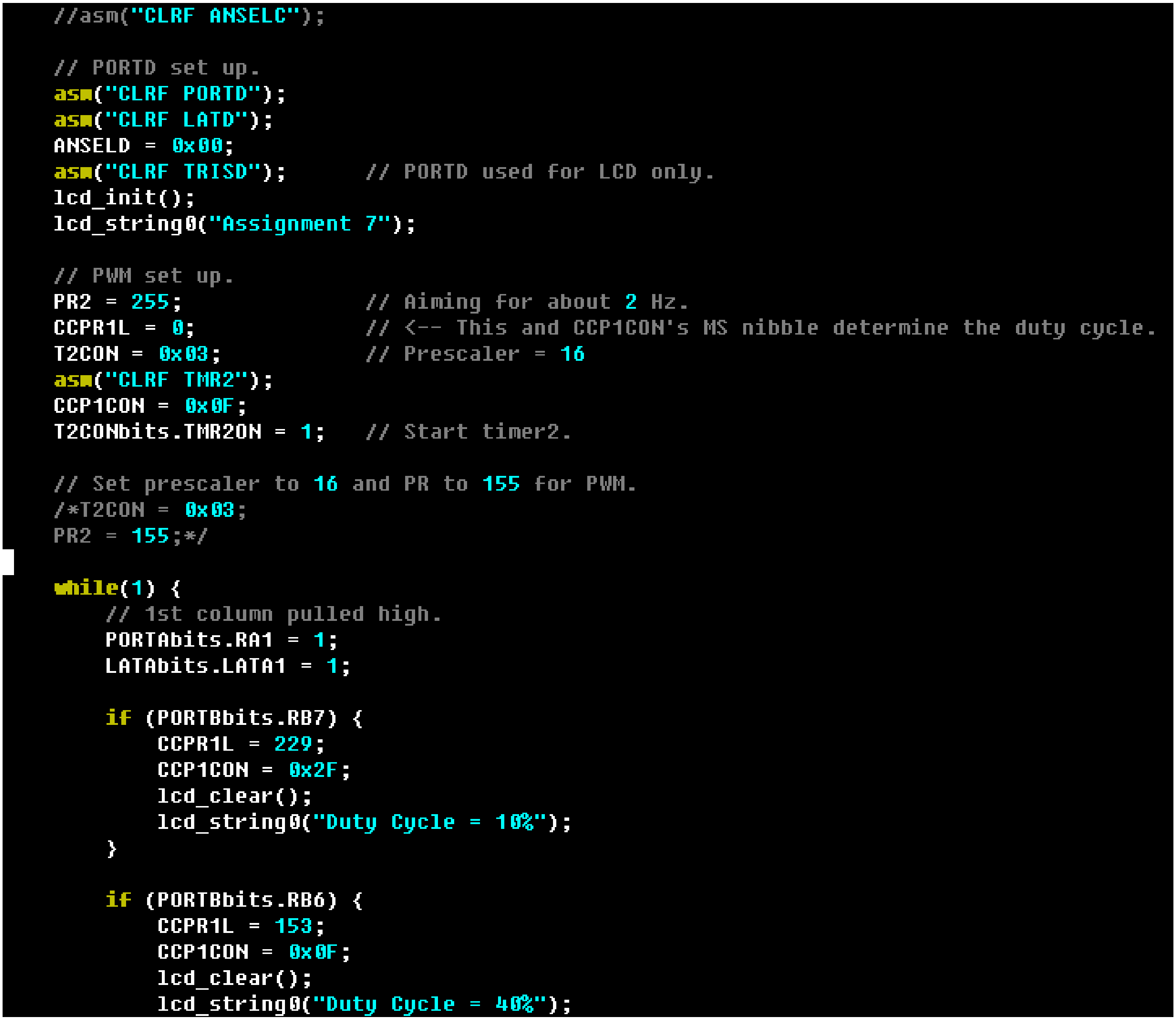
```
    lcd_cmd(0x0C);

    return;
}

void lcd_init_cmd(unsigned char cmd) {
    lcd_rs = 0;

    lcd_data = (lcd_data & 0x00) | (cmd & 0xF0);
    lcd_e = 1;
    __delay_ms(50);
    lcd_e = 0;
    __delay_ms(50);

    return;
}

void lcd_cmd(unsigned char cmd) {
    lcd_data = (lcd_data & 0x0F) | (cmd & 0xF0);
    lcd_rs = 0;
    lcd_e = 1;
    __delay_ms(1);
    lcd_e = 0;
    __delay_ms(3);
    lcd_data = (lcd_data & 0x0F) | (cmd << 4);
    lcd_e = 1;
    __delay_ms(1);
    lcd_e = 0;
    __delay_ms(3);

    return;
}

void lcd_writeChar(unsigned char c) {
    lcd_data = (lcd_data & 0x0F) | (c & 0xF0);
    lcd_rs = 1;
    lcd_e = 1;
    __delay_ms(1);
    lcd_e = 0;
```



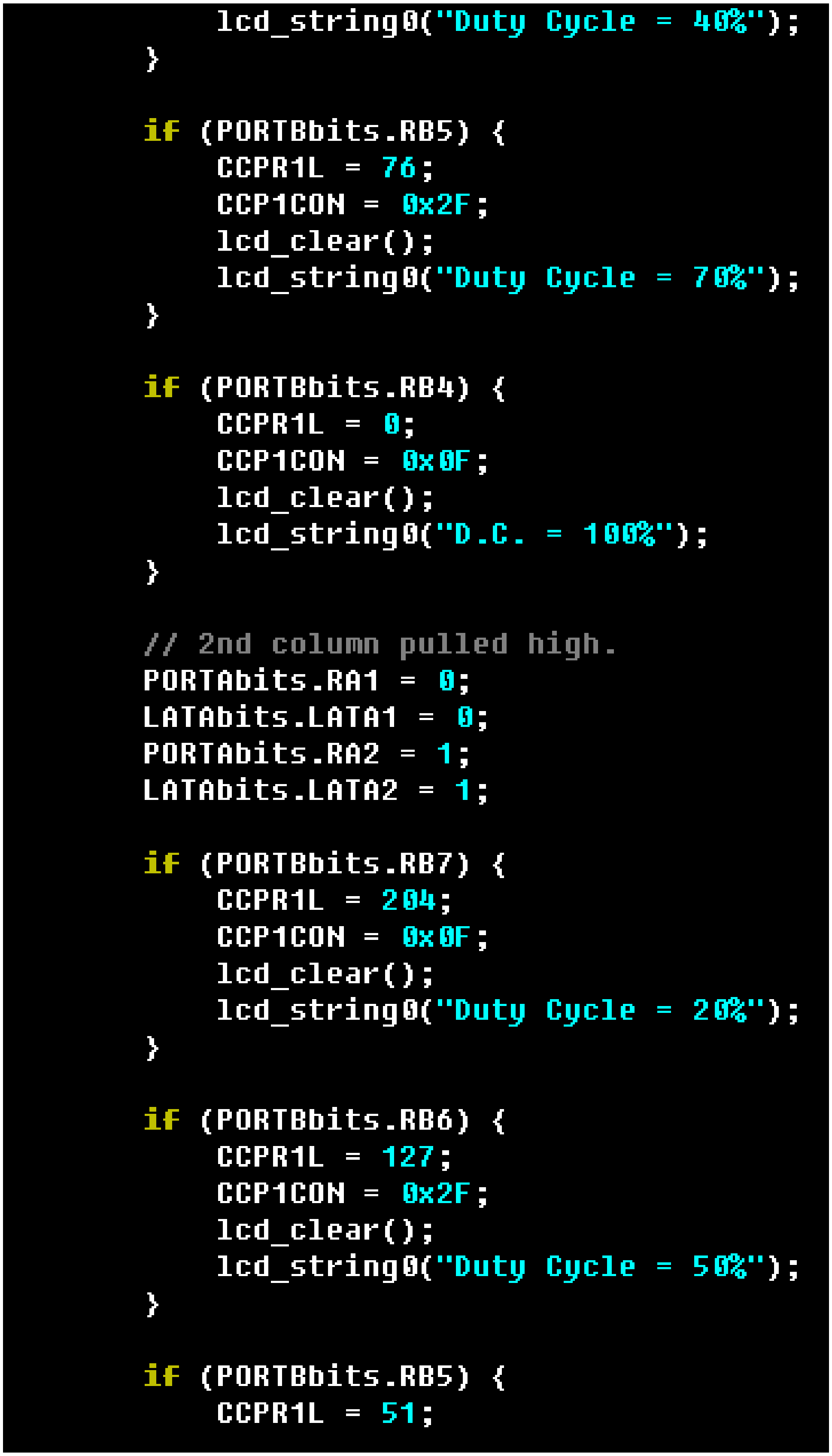
```
    lcd_e = 0;
    __delay_ms(3);
    lcd_data = (lcd_data & 0x0F) | (c << 4);
    lcd_e = 1;
    __delay_ms(1);
    lcd_e = 0;
    __delay_ms(3);

    return;
}

void lcd_string0(const char *s) {
    while((*s) != 0) {
        lcd_writeChar(*s);
        ++s;
    }

    return;
}

void lcd_string1(char row, char pos, const char *s) {
    char location = 0;

    if (row <= 1) {
        location = (0x80) | (pos & 0x0F);
        lcd_cmd(location);
    }else {
        location = (0xC0) | (pos & 0x0F);
        lcd_cmd(location);
    }

    lcd_string0(s);

    return;
}

void lcd_clear() {
    lcd_cmd(0x01);
    __delay_ms(3);
```



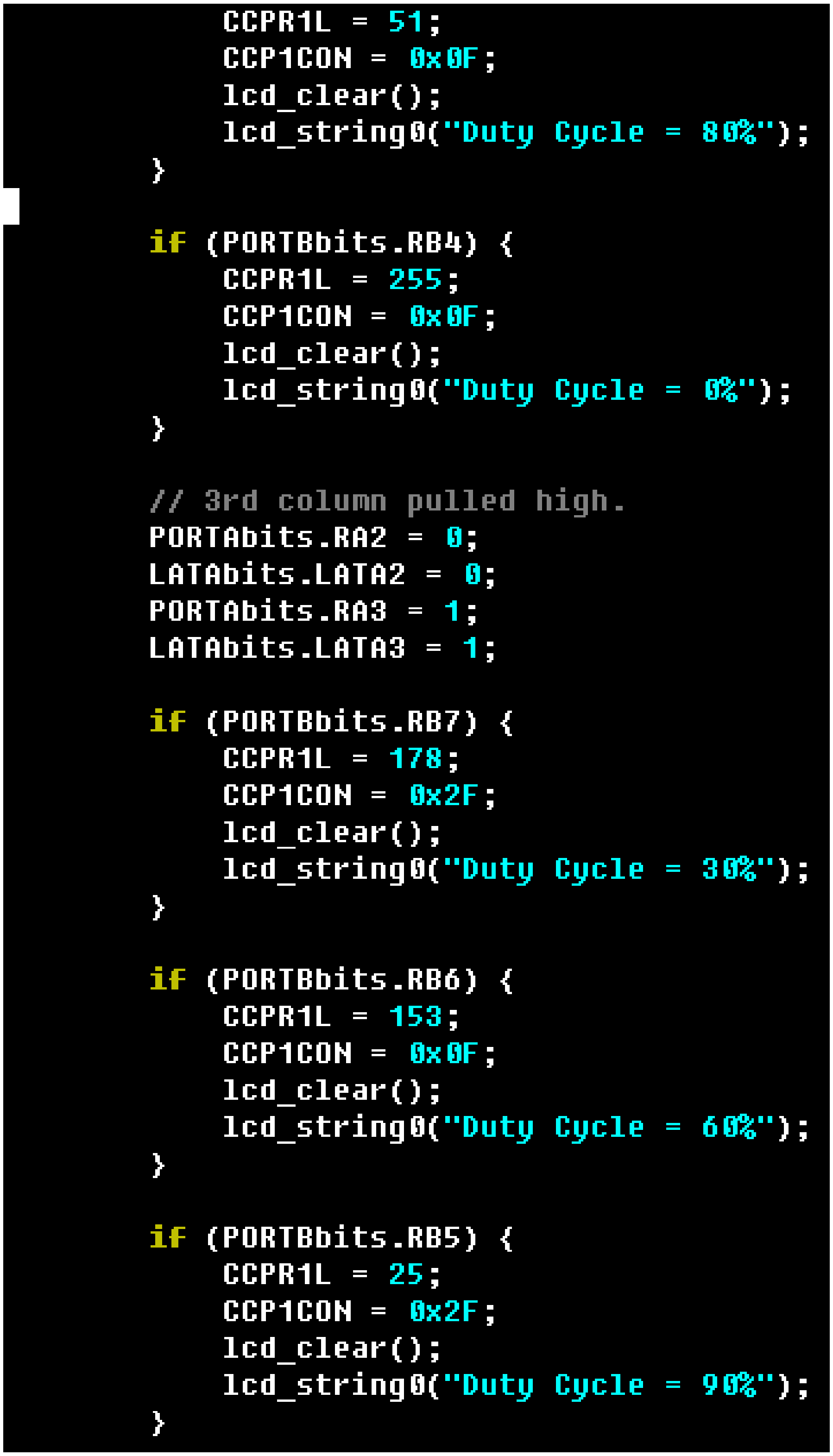
```
    __delay_ms(3);

    return;
}

void main(void) {

    // Set a 31.25 kHz clock frequency.
    OSCCON = 0x08;

    // Testing PWM with low clock frequency (31.25 kHz)...
    //OSCCON = 0x08;

    // PORTA set up.
    // Column 1 (left-most column) is PORTA[1], ... , and column 3 is PORTA[3].
    asm("CLRF PORTA");
    asm("CLRF LATA");
    ANSELA = 0x01;
    asm("CLRF TRISA");
    TRISAbits.TRISA0 = 1;   // Use on-board potentiometer for LCD contrast.
    TRISAbits.TRISA7 = 1;   // Monitor potentiometer output with RA7 LED.

    // PORTB set up.
    // Row 1 (top row) is PORTB[7], row 2 is PORTB[6], ... , and row 4 is PORTB[4].
    asm("CLRF PORTB");
    asm("CLRF LATB");
    ANSELB = 0x00;
    asm("CLRF TRISB");      // RB1 and RB0 used for LCD.
    TRISBbits.TRISB7 = 1;   // Row 1
    TRISBbits.TRISB6 = 1;   // Row 2
    TRISBbits.TRISB5 = 1;   // Row 3
    TRISBbits.TRISB4 = 1;   // Row 4

    // PORTC set up.
    asm("CLRF PORTC");
    asm("CLRF LATC");
    asm("CLRF TRISC");      // RC2 is the PWM 1 output.
    //asm("CLRF ANSELC");
```



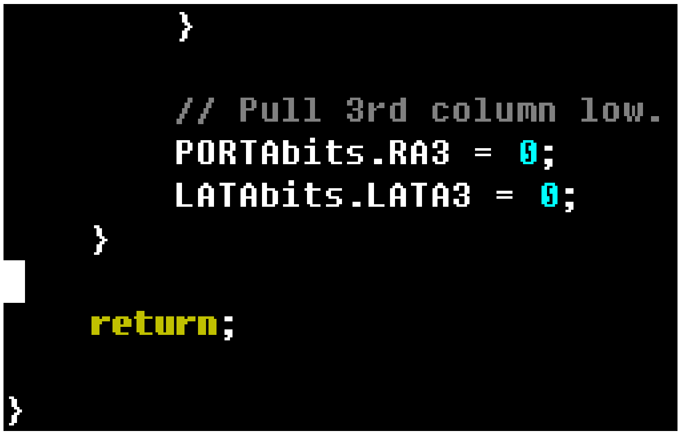
```c
    //asm("CLRF ANSELC");

    // PORTD set up.
    asm("CLRF PORTD");
    asm("CLRF LATD");
    ANSELD = 0x00;
    asm("CLRF TRISD");      // PORTD used for LCD only.
    lcd_init();
    lcd_string0("Assignment 7");

    // PWM set up.
    PR2 = 255;              // Aiming for about 2 Hz.
    CCPR1L = 0;             // <-- This and CCP1CON's MS nibble determine the duty cycle.
    T2CON = 0x03;           // Prescaler = 16
    asm("CLRF TMR2");
    CCP1CON = 0x0F;
    T2CONbits.TMR2ON = 1;   // Start timer2.

    // Set prescaler to 16 and PR to 155 for PWM.
    /*T2CON = 0x03;
    PR2 = 155;*/

    while(1) {
        // 1st column pulled high.
        PORTAbits.RA1 = 1;
        LATAbits.LATA1 = 1;

        if (PORTBbits.RB7) {
            CCPR1L = 229;
            CCP1CON = 0x2F;
            lcd_clear();
            lcd_string0("Duty Cycle = 10%");
        }

        if (PORTBbits.RB6) {
            CCPR1L = 153;
            CCP1CON = 0x0F;
            lcd_clear();
            lcd_string0("Duty Cycle = 40%");
```



```
            lcd_string0("Duty Cycle = 40%");
        }

        if (PORTBbits.RB5) {
            CCPR1L = 76;
            CCP1CON = 0x2F;
            lcd_clear();
            lcd_string0("Duty Cycle = 70%");
        }

        if (PORTBbits.RB4) {
            CCPR1L = 0;
            CCP1CON = 0x0F;
            lcd_clear();
            lcd_string0("D.C. = 100%");
        }

        // 2nd column pulled high.
        PORTAbits.RA1 = 0;
        LATAbits.LATA1 = 0;
        PORTAbits.RA2 = 1;
        LATAbits.LATA2 = 1;

        if (PORTBbits.RB7) {
            CCPR1L = 204;
            CCP1CON = 0x0F;
            lcd_clear();
            lcd_string0("Duty Cycle = 20%");
        }

        if (PORTBbits.RB6) {
            CCPR1L = 127;
            CCP1CON = 0x2F;
            lcd_clear();
            lcd_string0("Duty Cycle = 50%");
        }

        if (PORTBbits.RB5) {
            CCPR1L = 51;
```



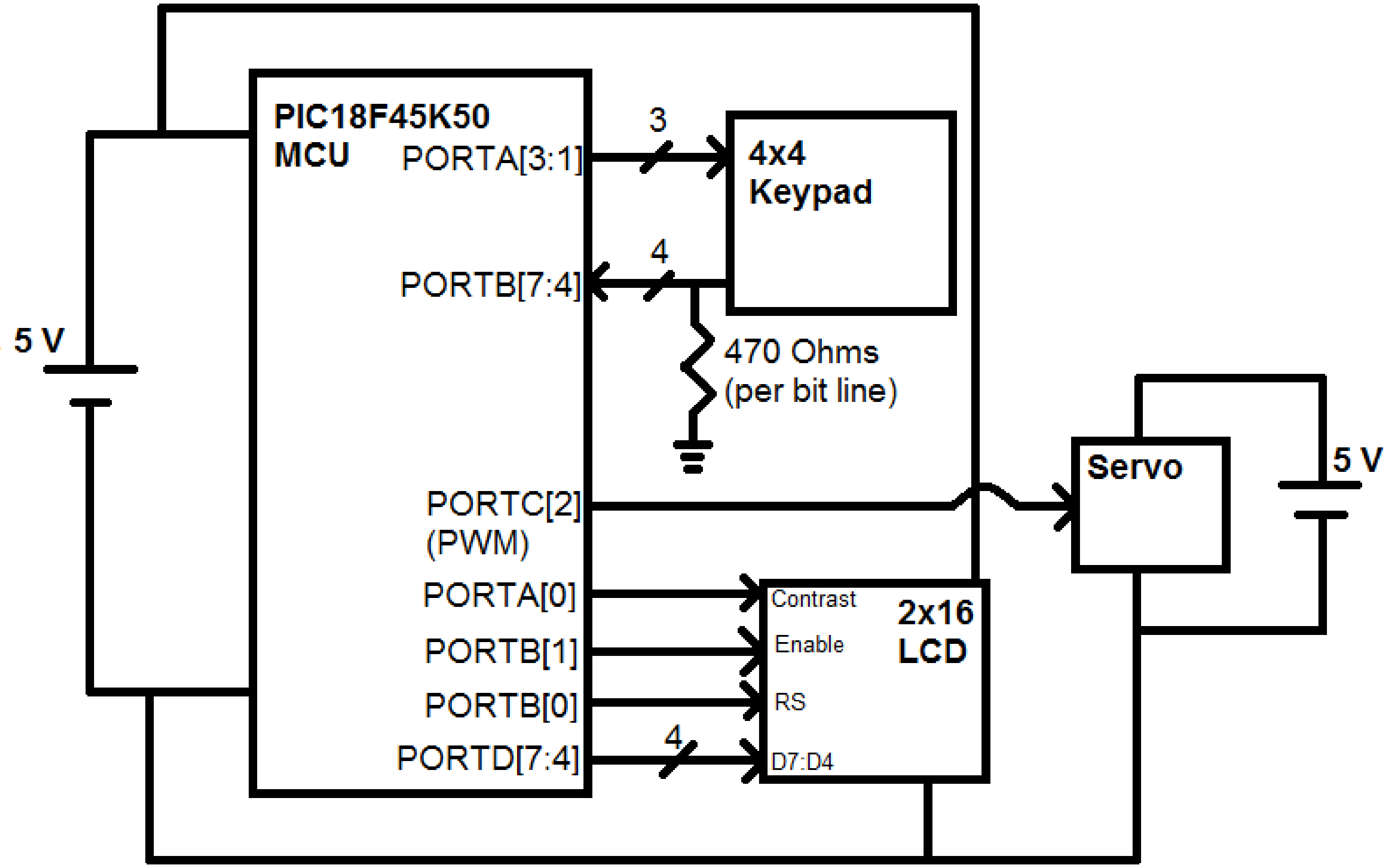
```
            CCPR1L = 51;
            CCP1CON = 0x0F;
            lcd_clear();
            lcd_string0("Duty Cycle = 80%");
        }

        if (PORTBbits.RB4) {
            CCPR1L = 255;
            CCP1CON = 0x0F;
            lcd_clear();
            lcd_string0("Duty Cycle = 0%");
        }

        // 3rd column pulled high.
        PORTAbits.RA2 = 0;
        LATAbits.LATA2 = 0;
        PORTAbits.RA3 = 1;
        LATAbits.LATA3 = 1;

        if (PORTBbits.RB7) {
            CCPR1L = 178;
            CCP1CON = 0x2F;
            lcd_clear();
            lcd_string0("Duty Cycle = 30%");
        }

        if (PORTBbits.RB6) {
            CCPR1L = 153;
            CCP1CON = 0x0F;
            lcd_clear();
            lcd_string0("Duty Cycle = 60%");
        }

        if (PORTBbits.RB5) {
            CCPR1L = 25;
            CCP1CON = 0x2F;
            lcd_clear();
            lcd_string0("Duty Cycle = 90%");
        }
```



```
        }

        // Pull 3rd column low.
        PORTAbits.RA3 = 0;
        LATAbits.LATA3 = 0;
    }

    return;

}
```

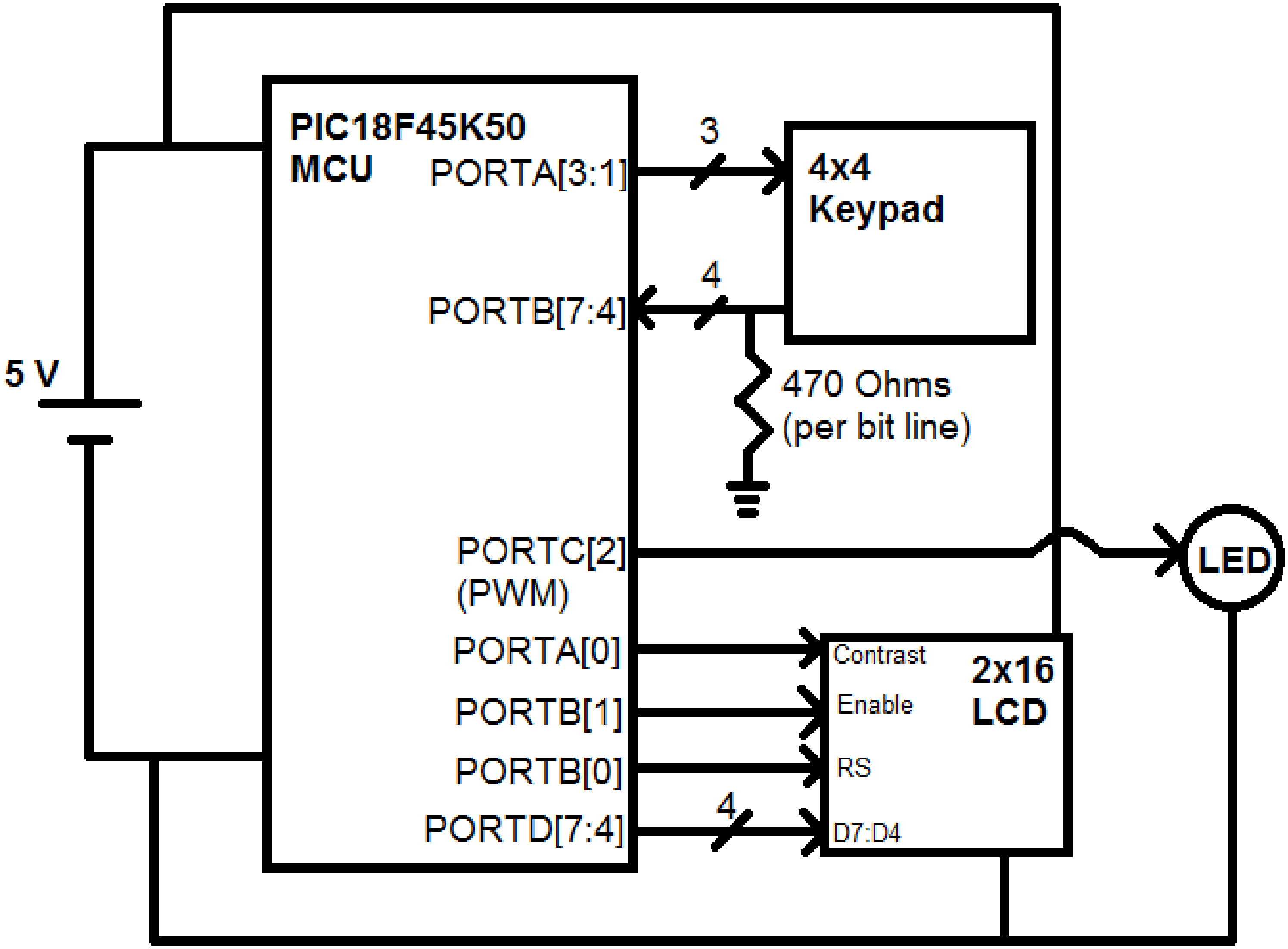
Joseph Daniel Moreno, U#60156204

Wiring Diagram:
Servo Project



Joseph Daniel Moreno, U#60156204

PWM Through LED Project



Joseph Daniel Moreno, U#60156204

System's Picture: