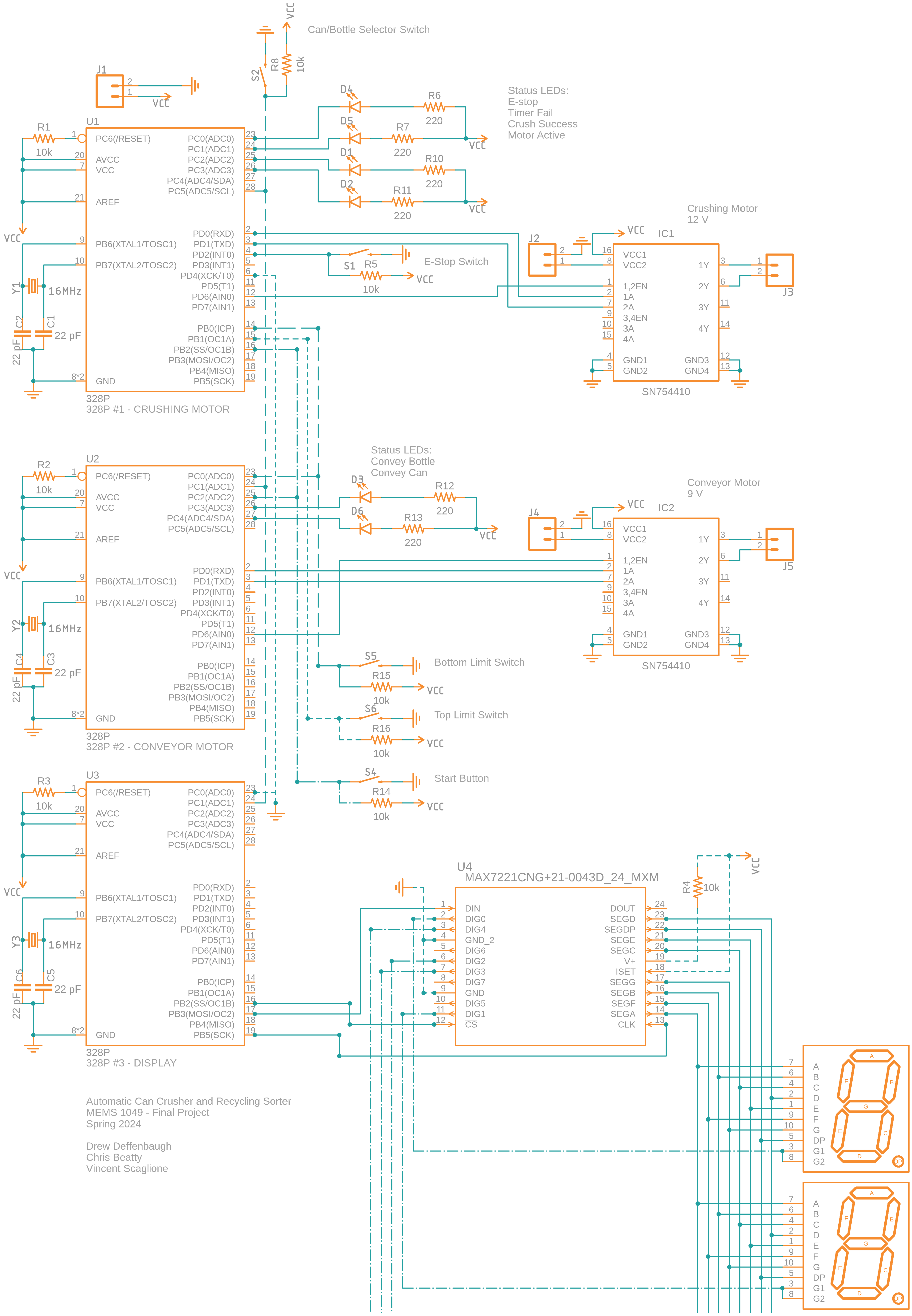
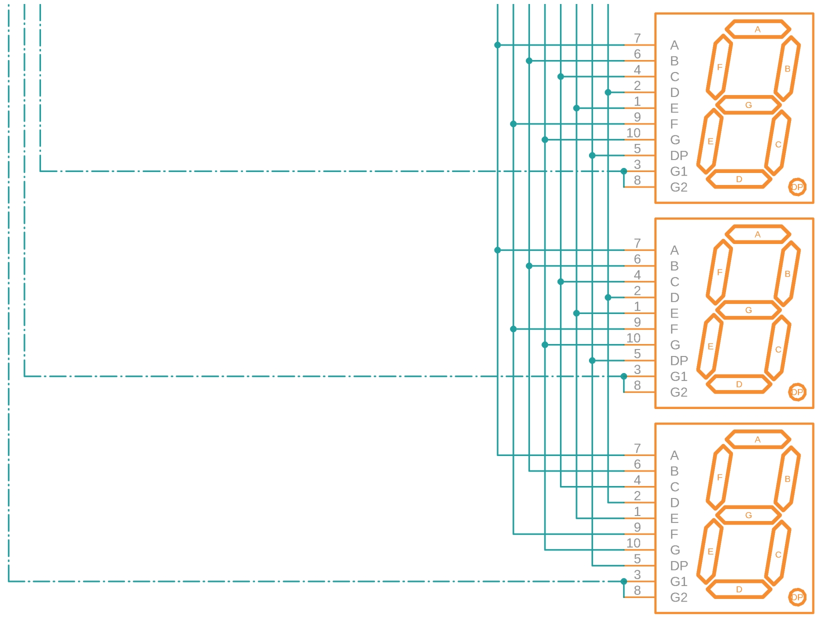
Mechatronics Project - Final Circuit Design And C Code

Final Circuit



Automatic Can Crusher and Recycling Sorter
MEMS 1049 - Final Project
Spring 2024

Drew Deffenbaugh
Chris Beatty
Vincent Scaglione



Final Code

328P #1 - Crushing Motor

```
/*
 * CrusherCode.c
 *
 * Created: 4/2/2024 7:43:06 AM
 * Author : Van
 */

#include <avr/io.h>
#include <avr/interrupt.h>

//Global
int count;
int workTime;
int flag;

//Functions
void wait(volatile int, volatile char);
void delay_T_msec_timer1(volatile char);
void crushUp(void);
void crushDown(void);
void activeCrush(int);
void activeLift(int);
void goHome(void);

//Interrupt Service
ISR(INT0_vect)
{
        //Dead stop system until reset
        OCR0A = 0;

        PORTC = PORTC & 0b11111110;
        PORTC = PORTC | 0b00001000;

        while(1) {}
}

int main(void)
{
// Setup

        //Set up PWM
        TCCR0A = 0b10000011; //PD6 on non-inverting and part of
fast PWM setting
        TCCR0B = 0b00000011; //Prescaler of 64 to get 1kHz freq
and rest of fast PWM setting
        DDRD = 0b01000011; //PD6 set as output pin for PWM, PD0
PD1 set as output for direction control of motor
        OCR0A = 0; //Initializing duty cycle at 0% (?)

        //Set up interrupt conditions
        EICRA = 1<<ISC01 | 0<<ISC00; //Interrupt 0 to activate
on falling edge

        EIMSK = 1<<INT0; //Enable interrupt 0

        sei(); //Enable global interrupt

        //Set switch pins
        DDRB = 0b00000000; //All switch pins are input: PB1 -
```

```
upper limit, PB2 - Start button, PB0 - lower limit

        //Set LED pins
        DDRC = 0b00011111; //PC0-3 for LED pins: PC2 - Success,
PC1 - Timer failure, PC0 - Safety interrupt, PC3 - Device
running, PC4 - NON LED: Crushing active output

        //Main loop
        while(1)
    {

                PORTC = 0b00001111;

                wait(250,2);
                while(1)
                {
                        if (!(PINB & 0b00000100))
                        {
                                break;
                        }
                } //Holds until user presses start switch

                if (PINC & 0b00100000) //if bottle mode is on
                {
                        wait(2000,2);
                }
                else
                {
                        PORTC = PORTC & 0b11110111;

                        //Startup function
                        goHome(); //Home positioning before
start
                        wait(2000,2);

                        //Send crusher down
                        crushDown();
                        OCR0A = 255; //Send at high speed
                        activeCrush(22000);


                        OCR0A = 0;
                        PORTC = PORTC & 0b11101111; //Change crush mode output off
                        wait(1000,2);

                        //Send crusher back up
                        crushUp();
                        OCR0A = 200;


                        activeLift(40000);

                }

        } //end main loop

} //end main function

void crushDown(void)
{
        PORTD = PORTD & 0b11111110; //Clear other dir. control bit
        PORTD = PORTD | 0b00000010; //Set forward motor pin (PD1)
}

void crushUp(void)
{
        PORTD = PORTD & 0b11111101; //Clear other dir. control bit
        PORTD = PORTD | 0b00000001; //Set backward motor pin (PD0)
}

void activeLift(int workTime)
```

```
{
        //Hold while loop until upper limit switch falling edge
or timer expires
        count = 0;
        while (PINB & 0b00000010)
        {
                wait(1,2);
                count++;

                if (count == workTime)
                {
                        PORTC = PORTC & 0b11111101; //Time fail
LED on
                        OCR0A = 0;
                        break;
                }
        }

        if (!(count == workTime)) //only if timer failure didn't
happen
        {
                goHome(); //Home positioning
                wait(5000,2);
        }

}

void activeCrush(int workTime)
{
        //Set pin to output crush signal
        PORTC = PORTC | 0b00010000;

        //Hold while loop until lower limit switch falling edge
or timer expires
        count = 0;
        while (PINB & 0b00000001)
        {
                wait(1,2);
                count++;
        if (count == workTime)
        {
            PORTC = PORTC & 0b11111101; //Time fail LED on
            break;
        }
    }

    if ((PORTC & 0b00000010)) //If time fail LED off
    {
        PORTC = PORTC & 0b11111011; //Success LED on
    }
}

void goHome(void)
{
    if (PINB & 0b00000010) //Limit Switch not active
    {
        //Send up slowly until upper limit switch falling edge
        crushUp();
        OCR0A = 150;

        wait(500,2);

        while(1)
        {
            if (!(PINB & 0b00000010))
            {
                break;
            }
        } //Breaks on limit switch press

        OCR0A = 0;
        wait(500,2);

        //Send down slowly until upper limit switch rising edge
```

```
                crushDown();
                OCR0A = 150;

                wait(1500,2);

                while(1)
                {
                        if (PINB & 0b00000010)
                        {
                                break;
                        }
                } //Breaks when switch is laid off of

                OCR0A = 0; //Turn off motor
        }
        else
        {
                //Send down slowly until upper limit switch rising edge
                OCR0A = 0;
                wait(500,2);

                crushDown();
                OCR0A = 150;

                wait(1500,2);

                while(1)
                {
                        if (PINB & 0b00000010)
                        {
                                break;
                        }
                } //Breaks when switch is laid off of

                OCR0A = 0; //Turn off motor
        }
} //end goHome
```

```
void wait(volatile int multiple, volatile char time_choice) {
        //*** wait ***
        /* This subroutine calls others to create a delay.
                 Total delay = multiple*T, where T is in msec
and is the delay created by the called function.

                Inputs: multiple = number of multiples to delay,
where multiple is the number of times an actual delay loop is
called.
                Outputs: None
        */

        while (multiple > 0) {
                delay_T_msec_timer1(time_choice); // we are
choosing case 2, which is a 1 msec delay
                multiple--;
        }
} // end wait()

void delay_T_msec_timer1(volatile char choice) {
        //
        // ***Note that since the Timer1 register is 16 bits,
the delays can be much higher than shown here.
        // This subroutine creates a delay of T msec using
TIMER1 with prescaler on clock, where, for a 16MHz clock:
        //T = 0.125 msec for prescaler set to 8 and count of 250
(preload counter with 65,535-5)
        //T = 1 msec for prescaler set to 64 and count of 250
(preload counter with 65,535-5)
        //T = 4 msec for prescaler set to 256 and count of 250
(preload counter with 65,535-5)
        //T = 16 msec for prescaler set to 1,024 and count of
250 (preload counter with 65,535-5)
        //Default: T = .0156 msec for no prescaler and count of
250 (preload counter with 65,535-5)

        //Inputs: None
        //Outputs: None
        TCCR1A = 0x00; // clears WGM00 and WGM01 (bits 0 and 1) to ensure Timer/Counter is in normal mode.
        TCNT1 = 0;  // preload load TIMER1 with 5 if counting to 255 (count must reach 65,535-5 = 250)
        // or preload with 0 and count to 250

        switch ( choice ) { // choose prescaler
                case 1:
                TCCR1B = 1<<CS11;//TCCR1B = 0x02; // Start TIMER1, Normal mode, crystal clock, prescaler = 8
                break;
                case 2:
                TCCR1B =  1<<CS11 | 1<<CS10;//TCCR1B = 0x03;  // Start TIMER1, Normal mode, crystal clock, prescaler = 64
                break;
                case 3:
                TCCR1B = 1<<CS12;//TCCR1B = 0x04; // Start TIMER1, Normal mode, crystal clock, prescaler = 256
                break;
                case 4:
                TCCR1B = 1<<CS12 | 1<<CS10;//TCCR1B = 0x05; // Start TIMER1, Normal mode, crystal clock, prescaler = 1024
                break;
                default:
                TCCR1A = 1<<CS10;//TCCR1B = 0x01; Start TIMER1, Normal mode, crystal clock, no prescaler
                break;
        }

        //while ((TIFR1 & (0x1<<TOV1)) == 0); // wait for TOV1 to roll over at 255 (requires preload of 65,535-5 to make count = 250)
        // How does this while loop work?? See notes
        while (TCNT1 < 0xfa); // exits when count = 250 (requires preload of 0 to make count = 250)

        TCCR1B = 0x00; // Stop TIMER1
        //TIFR1 = 0x1<<TOV1;  // Clear TOV1 (note that this is an odd bit in that it
        //is cleared by writing a 1 to it)

} // end delay_T_msec_timer1()
```

328P #2 - Conveyor Motor

```
/*
 * Sorter-Code.c
 *
 * Created: 4/1/2024 4:38:25 PM
 * Author : cjbea
 */

#include <avr/io.h>

int PWM_value = 0;
float voltage_read = 0;

void wait(volatile int multiple, volatile char time_choice);
void delay_T_msec_timer1(char choice);

char switch_value = 'c';                //Set switch value
variable (checking if bottle or can)

int main(void)
{
        // Setup
        DDRC = 0b00011000;              //setting PORTC to all
inputs except for 3 and 4 for LEDs
        PORTC = 0b00011000;             //Active low LEDs
        DDRB = 0xFF;
        PORTB = 0x01;
        DDRD = 0b01100011; //Setting PD5 and PD6 to output for
PWM, and setting PD0 and PD1 to output for motor control
        PORTD = 0b00000001;

        OCR0A = 0x00;       // Load $00 into OCR0 to set initial
duty cycle to 0 (motor off)
```

```
        TCCR0A = 0b10000011; //1<<COM0A1 | 0<<COM0A0 | 1<<WGM01
| 1<<WGM00;      // Set non-inverting mode on OC0A pin (COMA1:0
= 10; Fast PWM (WGM1:0 bits = bits 1:0 = 11) (Note that we are
not affecting OC0B because COMB0:1 bits stay at default = 00)
        TCCR0B = 0b00000011; //0<<CS02 | 1<<CS01 | 1<<CS00; //
Set base PWM frequency (CS02:0 - bits 2-0 = 011 for prescaler of
64, for approximately 1kHz base frequency)
         //PWM is now running on selected pin at selected base
frequency.  Duty cycle is set by loading/changing value in OCR0A
register.

        while(1)
        {
                PWM_value = 255;

                while(PINC & 0b00000100){}      //wait for start
button

                if((PINC & 0b00000010))         //If bottle,
turn one direction
                {

                        OCR0A = PWM_value;              //Set
motor Speed

                        PORTD = 0b00000010;             //turn
on belt
                        PORTC = PORTC & 0b11101111;
//Bottle LED

                } else   //if can, wait for crusher
                {

                        while ((PINC & 0b00000001))
//check limit switch being pressed
                        {}

                                wait(3000,2);
                                OCR0A = PWM_value;
//Set motor Speed

                                PORTD = 0b00000001;
//turn on belt
                                PORTC = PORTC & 0b11110111;
//Can LED

                }


                        wait(9000,2);           //Wait for item
to deposit
                        PORTD = 0x00;           //Turn off motor
                        PORTC = 0xFF;           //turn off LED's
                        OCR0A = 0;

        } // end main while
} // end main

void wait(volatile int multiple, volatile char time_choice) {
        /* This subroutine calls others to create a delay.
                 Total delay = multiple*T, where T is in msec
and is the delay created by the called function.

                Inputs: multiple = number of multiples to delay,
where multiple is the number of times an actual delay loop is
called.
                Outputs: None
        */

        while (multiple > 0) {
                delay_T_msec_timer1(time_choice);
                multiple--;
        }
} // end wait()
void delay_T_msec_timer1(volatile char choice) {
        //
        // ***Note that since the Timer1 register is 16 bits,
the delays can be much higher than shown here.
        // This subroutine creates a delay of T msec using TIMER1 with prescaler on clock, where, for a 16MHz clock:
        //T = 0.125 msec for prescaler set to 8 and count of 250 (preload counter with 65,535-5)
        //T = 1 msec for prescaler set to 64 and count of 250 (preload counter with 65,535-5)
        //T = 4 msec for prescaler set to 256 and count of 250 (preload counter with 65,535-5)
        //T = 16 msec for prescaler set to 1,024 and count of 250 (preload counter with 65,535-5)
        //Default: T = .0156 msec for no prescaler and count of 250 (preload counter with 65,535-5)

        //Inputs: None
        //Outputs: None

        TCCR1A = 0x00; // clears WGM00 and WGM01 (bits 0 and 1) to ensure Timer/Counter is in normal mode.
        TCNT1 = 0;  // preload load TIMER1 with 5 if counting to 255 (count must reach 65,535-5 = 250)
        // or preload with 0 and count to 250

        switch ( choice ) { // choose prescaler
                case 1:
                TCCR1B = 1<<CS11;//TCCR1B = 0x02; // Start TIMER1, Normal mode, crystal clock, prescaler = 8
                break;
                case 2:
                TCCR1B =  1<<CS11 | 1<<CS10;//TCCR1B = 0x03;  // Start TIMER1, Normal mode, crystal clock, prescaler = 64
                break;
                case 3:
                TCCR1B = 1<<CS12;//TCCR1B = 0x04; // Start TIMER1, Normal mode, crystal clock, prescaler = 256
                break;
                case 4:
                TCCR1B = 1<<CS12 | 1<<CS10;//TCCR1B = 0x05; // Start TIMER1, Normal mode, crystal clock, prescaler = 1024
                break;
                default:
                TCCR1A = 1<<CS10;//TCCR1B = 0x01; Start TIMER1, Normal mode, crystal clock, no prescaler
                break;
        }

        //while ((TIFR1 & (0x1<<TOV1)) == 0); // wait for TOV1 to roll over at 255 (requires preload of 65,535-5 to make count = 250)
        // How does this while loop work?? See notes
        while (TCNT1 < 0xfa); // exits when count = 250 (requires preload of 0 to make count = 250)

        TCCR1B = 0x00; // Stop TIMER1
        //TIFR1 = 0x1<<TOV1;  // Clear TOV1 (note that this is an odd bit in that it
        //is cleared by writing a 1 to it)

} // end delay_T_msec_timer1()
```

328P #3 - Display

```
/*
 * 1049 - Deffenbaugh - Seven_Segmen.c
 *
 * Created: 4/2/2024 9:38:55 AM
 * Author : dldef
 */

#include <avr/io.h>
#include <avr/interrupt.h>

// functions
void print_seven_seg(unsigned char command, unsigned char data);
void display_number(int number);
void display_letter(char letter, int digit);
void wait(int);
void increment_count(void);

// global variables
unsigned char data;
unsigned char command;
int count = 0;
int letter_code;
int i;

//interrupt service routines
ISR(INT0_vect)
{
        // INCREMENT COUNT
        increment_count();

        // FLASH CRUSHING
        if(!(PINC & 0b00000010)) // toggle switch set to can
        {
                while(!(PINC & 0b00000001)){}

                while(PINC & 0b00000001){
                        display_letter('C', 0);
                        display_letter('R',1);
                        display_letter('U',2);
                        display_letter('S',3);
                        display_letter('H',4);
                        wait(1000);

                        display_letter('Q', 0);
                        display_letter('Q',1);
                        display_letter('Q',2);
                        display_letter('Q',3);
                        display_letter('Q',4);
                        wait(1000);
                }
        }

        // FLASH THANK YOU
        for(i=0; i<4; i++)
        {
                display_letter('T', 0);
                display_letter('H',1);
                display_letter('A',2);
                display_letter('N',3);
                display_letter('K',4);
                wait(500);

                display_letter('Q', 0);
                display_letter('Y',1);
                display_letter('O',2);
                display_letter('U',3);
                display_letter('Q',4);
                wait(500);
        }

        // FLASH COUNT
        for(i=0; i<4; i++)
        {
                display_number(count);
                wait(500);

                display_letter('Q', 0);
                display_letter('Q',1);
                display_letter('Q',2);
                display_letter('Q',3);
                display_letter('Q',4);
                wait(500);
        }

        EIFR = 0b00000001;

} // RETURN TO STANDBY

int main(void)
{
        //set up inputs
        DDRC = 0b00000000;      // PC0 and 1 input
        DDRD = 0b00000000;  // set bits of PORTD as input (only need PD2 and PD3 as input for the interrupts)

        // Set up count variable
        //count = 0;

        // Set up Main SPI
        DDRB = 0b00101100; // DDRB = 1<<PORTB5 | 1<<PORTB3 | 1<<PORTB2;  // Set pins SCK, MOSI, and SS as output
        SPCR = 0b01010001; // (SPIE = 0, SPE = 1, DORD = 0, MSTR = 1, CPOL = 0, CPHA = 0, SPR1 = 0, SPR0 = 1 // enable the SPI, set to Main mode 0, SCK = Fosc/16, lead with MSB

        // Set up Interrupts
        EICRA = 0b00000011; //set INT0 to RAISING edge
        EIMSK = 0b00000001; //enable INT0 and INT 1
        sei();  //enable global interrupt

        // Set up seven segment display
        command = 0b00001011;   // set scan limit to 5
        data = 0b00000100;
        print_seven_seg(command, data);

        command = 0b00001010;   // set intensity to max
        data = 0b00001111;
        print_seven_seg(command, data);

        command = 0b00001100;   // turn on display
        data = 0b00000001;
        print_seven_seg(command, data);


        //command = 0b00001001; // set decoding mode to yes
        //data = 0b00011111;
        //print_seven_seg(command, data);
        //command = 0x0F;       // test display
        //data = 0b00000001;
        //print_seven_seg(command, data);
        //while(1){};
```

```
while (1) {
            // --- display count
                    display_number(count);
                    wait(2000);

            // --- display "You Feed I Crush"
                    display_letter('Q', 0);
                    display_letter('Y',1);
                    display_letter('O',2);
                    display_letter('U',3);
                    display_letter('Q',4);
                    wait(750);

                    display_letter('F', 0);
                    display_letter('E',1);
                    display_letter('E',2);
                    display_letter('E',3);
                    display_letter('D',4);
                    wait(750);

                    display_letter('Q', 0);
                    display_letter('Q',1);
                    display_letter('I',2);
                    display_letter('Q',3);
                    display_letter('Q',4);
                    wait(750);

                    display_letter('C', 0);
                    display_letter('R',1);
                    display_letter('U',2);
                    display_letter('S',3);
                    display_letter('H',4);
                    wait(750);

            // --- display count
                    display_number(count);
                    wait(2000);
```

```
// --- display "me so hungry"
        display_letter('Q', 0);
        display_letter('Q',1);
        display_letter('I',2);
        display_letter('Q',3);
        display_letter('Q',4);
        wait(750);

        display_letter('Q', 0);
        display_letter('S',1);
        display_letter('O',2);
        display_letter('Q',3);
        display_letter('Q',4);
        wait(300);
        display_letter('Q', 0);
        display_letter('Q',1);
        display_letter('S',2);
        display_letter('O',3);
        display_letter('Q',4);
        wait(300);
        display_letter('Q', 0);
        display_letter('S',1);
        display_letter('O',2);
        display_letter('Q',3);
        display_letter('Q',4);
        wait(300);

        display_letter('H', 0);
        display_letter('U',1);
        display_letter('N',2);
        display_letter('G',3);
        display_letter('R',4);
        wait(350);
        display_letter('U', 0);
        display_letter('N',1);
        display_letter('G',2);
        display_letter('R',3);
        display_letter('Y',4);
```

```
                wait(350);

                display_letter('H', 0);
                display_letter('U',1);
                display_letter('N',2);
                display_letter('G',3);
                display_letter('R',4);
                wait(350);
                display_letter('U', 0);
                display_letter('N',1);
                display_letter('G',2);
                display_letter('R',3);
                display_letter('Y',4);
                wait(350);

        // --- display count
                display_number(count);
                wait(2000);

        // --- display "the void beckons"
                display_letter('Q', 0);
                display_letter('T',1);
                display_letter('H',2);
                display_letter('E',3);
                display_letter('Q',4);
                wait(750);

                display_letter('V', 0);
                display_letter('O',1);
                display_letter('I',2);
                display_letter('D',3);
                display_letter('Q',4);
                wait(750);

                display_letter('Q', 0);
                display_letter('Q',1);
                display_letter('Q',2);
                display_letter('B',3);
                display_letter('E',4);
```

```
                        wait(250);
                        display_letter('Q', 0);
                        display_letter('Q',1);
                        display_letter('B',2);
                        display_letter('E',3);
                        display_letter('C',4);
                        wait(250);
                        display_letter('Q', 0);
                        display_letter('B',1);
                        display_letter('E',2);
                        display_letter('C',3);
                        display_letter('K',4);
                        wait(250);
                        display_letter('B', 0);
                        display_letter('E',1);
                        display_letter('C',2);
                        display_letter('K',3);
                        display_letter('O',4);
                        wait(250);
                        display_letter('E', 0);
                        display_letter('C',1);
                        display_letter('K',2);
                        display_letter('O',3);
                        display_letter('N',4);
                        wait(250);
                        display_letter('C', 0);
                        display_letter('K',1);
                        display_letter('O',2);
                        display_letter('N',3);
                        display_letter('S',4);
                        wait(250);
                        display_letter('K', 0);
                        display_letter('O',1);
                        display_letter('N',2);
                        display_letter('S',3);
                        display_letter('Q',4);
                        wait(250);
                        display_letter('O', 0);
                        display_letter('N',1);
```

```
                display_letter('S',2);
                display_letter('Q',3);
                display_letter('Q',4);
                wait(250);
                display_letter('N', 0);
                display_letter('S',1);
                display_letter('Q',2);
                display_letter('Q',3);
                display_letter('Q',4);
                wait(250);
                display_letter('S', 0);
                display_letter('Q',1);
                display_letter('Q',2);
                display_letter('Q',3);
                display_letter('Q',4);
                wait(250);
                display_letter('Q', 0);
                display_letter('Q',1);
                display_letter('Q',2);
                display_letter('Q',3);
                display_letter('Q',4);
                wait(500);

        // --- display count
                display_number(count);
                wait(2000);

        // --- display "boo...  ... ahhh"
                display_letter('B',0);
                display_letter('O',1);
                display_letter('O',2);
                display_letter('Q',3);
                display_letter('Q',4);
                wait(750);

                display_letter('Q',0);
                display_letter('Q',1);
                display_letter('Q',2);
                display_letter('A',3);
```

```
                        display_letter('H',4);
                        wait(250);
                        display_letter('Q',0);
                        display_letter('Q',1);
                        display_letter('A',2);
                        display_letter('H',3);
                        display_letter('H',4);
                        wait(250);
                        display_letter('Q',0);
                        display_letter('A',1);
                        display_letter('H',2);
                        display_letter('H',3);
                        display_letter('H',4);
                        wait(250);
                        display_letter('A',0);
                        display_letter('H',1);
                        display_letter('H',2);
                        display_letter('H',3);
                        display_letter('H',4);
                        wait(250);
                        display_letter('H',0);
                        display_letter('H',1);
                        display_letter('H',2);
                        display_letter('H',3);
                        display_letter('H',4);
                        wait(300);

                        display_letter('Q',0);
                        display_letter('Q',1);
                        display_letter('Q',2);
                        display_letter('Q',3);
                        display_letter('Q',4);
                        wait(250);

                        display_letter('S',0);
                        display_letter('C',1);
                        display_letter('A',2);
                        display_letter('R',3);
                        display_letter('Y',4);
                        wait(750);

    }//end while
}

//MAX7221 transmit interface
void print_seven_seg(unsigned char command, unsigned char data){
        // Transmit the data
        PORTB &= ~(0b00000100); //(1 << PORTB2);  // Clear the SS bit to enable Secondary

        SPDR = command; //Send the command
        while (!(SPSR & 0b10000000)); // Check the SPIF bit and wait for it to be set => transmit complete

        SPDR = data; //Send the data
        while (!(SPSR & 0b10000000)); // Check the SPIF bit and wait for it to be set => transmit complete

        PORTB = PORTB | 0b00000100; //Return PB2 to 1, set the SS bit to disable secondary (end transmission)
}

//number display (able to print numbers 0-29)
void display_number(int number){
        command = 0b00001001;   // set decoding mode to yes
        data = 0b00011111;
        print_seven_seg(command, data);

        if(count <10){
                command = 0x01; // set digit 0
                data = number;  // display ones place
                print_seven_seg(command, data);

                command = 0x02; //set digit 1
                data = 0b00001111; // display BLANK
                print_seven_seg(command, data);

                command = 0x03; //set digit 2
                data = 0b00001111; // display BLANK
                print_seven_seg(command, data);

                command = 0x04; //set digit 3
                data = 0b00001111; // display BLANK
                print_seven_seg(command, data);

                command = 0x05; //set digit 4
                data = 0b00001111; // display BLANK
                print_seven_seg(command, data);
        }

        else if(count <20){
                command = 0x01; // set digit 0
                data = 0b00000001; // display '1'
                print_seven_seg(command, data);

                command = 0x02; //set digit 1
                data = (number-10);     // display ones place
                print_seven_seg(command, data);

                command = 0x03; //set digit 2
                data = 0b00001111; // display BLANK
                print_seven_seg(command, data);

                command = 0x04; //set digit 3
                data = 0b00001111; // display BLANK
                print_seven_seg(command, data);

                command = 0x05; //set digit 4
                data = 0b00001111; // display BLANK
                print_seven_seg(command, data);
        }

        else if(count <30){
                command = 0x01; // set digit 0
                data = 0b00000010; // display '2'
                print_seven_seg(command, data);
                command = 0x02; //set digit 1
                data = (number-20);     // display ones place
                print_seven_seg(command, data);

                command = 0x03; //set digit 2
                data = 0b00001111; // display BLANK
                print_seven_seg(command, data);

                command = 0x04; //set digit 3
                data = 0b00001111; // display BLANK
                print_seven_seg(command, data);

                command = 0x05; //set digit 4
                data = 0b00001111; // display BLANK
                print_seven_seg(command, data);
        }
}

//letter display
void display_letter(char letter, int digit){
        command = 0b00001001;   // set decoding mode to no
        data = 0b00000000;
        print_seven_seg(command, data);

        switch (letter){
                case 'Q':
                        letter_code = 0b00000000;   // SPACE
(blank)
                        break;
                case 'A':
                        letter_code = 0b01110111;   // letter A
                        break;
                case 'B':
                        letter_code = 0b00011111;   // letter B
                        break;
                case 'C':
                        letter_code = 0b01001110;       //
letter C
                        break;
                case 'D':
                        letter_code = 0b00111101;       //
letter D
                        break;
                case 'E':
                        letter_code = 0b01001111;       //
letter E
                        break;
                case 'F':
                        letter_code = 0b01000111;       //
letter F
                        break;
                case 'G':
                        letter_code = 0b01011110;       //
letter G
                        break;
                case 'H':
                        letter_code = 0b00110111;       //
letter H
                        break;
                case 'I':
                        letter_code = 0b00110000;       //
letter I
                        break;
                case 'J':
                        letter_code = 0b00111100;       //
letter J
                        break;
                case 'K':
                        letter_code = 0b01010111;       //
letter K
                        break;
                case 'L':
                        letter_code = 0b00001110;       //
letter L
                        break;
                case 'M':
                        letter_code = 0b11111111;       //
letter M BROKEN
```

```
                        break;
                case 'N':
                        letter_code = 0b00010101;        //letter
N
                        break;
                case 'O':
                        letter_code = 0b01111110;        //
letter O
                        break;
                case 'P':
                        letter_code = 0b01100111;        //
letter P
                        break;
                case 'R':
                        letter_code = 0b00000101;        //
letter R
                        break;
                case 'S':
                        letter_code = 0b01011011;        //
letter S
                        break;
                case 'T':
                        letter_code = 0b00001111;        //
letter T
                        break;
                case 'U':
                        letter_code = 0b000111110;       //
letter U
                        break;
                case 'V':
                        letter_code = 0b000111110;       //
letter V
                        break;
                case 'X':
                        letter_code = 0b00110111;        //
letter X
                        break;
                case 'Y':
                        letter_code = 0b00111011;        //
```

```
letter Y
                        break;
                case 'Z':
                        letter_code = 0b01101101;       //
letter Z
                        break;
        }

        command = (digit+1);    // display on digit "digit"
        data = letter_code;     // display character
        print_seven_seg(command, data);
}

void increment_count(void){

        count = count +1;

}

//wait function
void wait(volatile int multiple)
{
        //creates a delay equal to multiple*T (T is 1 msec)
        //assumes 16MHz clock frequency, change exit value in
while loop to change
        while (multiple > 0)
        {
                TCCR0A = 0x00; //clears WGM00 and WGM01 (bits 1
& 2)
                TCNT0 = 0; //preload value for testing on count
= 250
                TCCR0B = 0b00000011; //1<<CS01 | 1<<CS00; &nbsp;
&nbsp;TCCR0B = 0x03; &nbsp;//start TIMER0, normal mode, crystal
clock, prescale = 64
                while (TCNT0 < 0xFA);  //exits when count = 250
CHANGE THIS FOR DIFFERNT CLOCKS
                TCCR0B = 0x00; //stop TIMER0
                multiple--;
        }
}
```