```c
// Samuel DuBois and Michael Brauninger
// ECE 231: Embedded Systems
// Project 2 Final Test Program 8
// 03.04.2020

#include <asf.h>

// Definitions of Digit 'identities' (Bit representations of numbers on CLK display)
#define zero    0b00111111
#define one     0b00000110
#define two     0b01011011
#define three   0b01001111
#define four    0b01100110
#define five    0b01101101
#define six     0b01111101
#define seven   0b00000111
#define eight   0b01111111
#define nine    0b01101111

// Variable to monitor whether the program is stopped
int isStopped = 0;

// An array of our Digit Identities
uint32_t identities[10] = { zero, one, two, three, four, five, six, seven, eight, nine };

// Program incrementer *Should move to another function location*
static uint32_t inc = 0b00000000;

// MARK: Handle any trigger sent by the TIMER_COMP_vect
ISR(TIMER1_COMPA_vect) {
    if (isStopped == 0) {
        inc += 1;
    }
}

// MARK: Handle any triggered flag sent by the SPI_STC_vect
ISR(SPI_STC_vect) {

    // Set the SS flag to end transmission
    PORTB = 0b00101100;
}

// MARK: Handle any input from the first button
ISR(INT1_vect) {
    if (isStopped == 0) {
        isStopped = 1;
    }
```

```
    else {
        isStopped = 0;
    }
}

// MARK: Handle any input from the second button
ISR(PCINT2_vect) {
    inc = 0b00000000;
}

// Description: returns the correct identity value of the number we want to display, given the value of the incrementer
int calculateIdentityFor(int digit) {

    int ds; // deciseconds
    int os; // ones place of seconds
    int ts; // tens place of seconds
    int m;  // minutes

    switch (digit) {
        case 0:
            ds = inc % 10;
            return identities[ds];
        case 1:
            os = inc / 10;
            os = os % 10;
            return identities[os];
        case 2:
            ts = ((inc - (inc % 100)) % 600) / 100;
            return identities[ts];
        case 3:
            m = inc / 600;
            return identities[m];
        default:
            return zero;
    }
}

void checkForOverflow() {
    if (inc >= 5999)
        inc -= 5999; // Maximum Value before overflow
}

// Description: Shifts through each 'index' of the clock display so that it creates the illusion that all of the lights on the screen are lit up at the same time
// function shiftThroughDisplayIndices(time: int) -> Void
void shiftThroughDisplayIndices(uint32_t time) {
    uint32_t delay = 0;
```

```
    // Enable first index
    SPDR = calculateIdentityFor(3) + 128;
    // Turn off the SS Pin to configure communication and start transmission
    PORTB = 0b00101000;
    // Set value of the 3rd transistor
    PORTC = 0b00100000;
    while (delay < time) {
        delay += 1;
    }

    // enable second pin
    SPDR = calculateIdentityFor(2);
    // Turn off the SS Pin to configure communication and start transmission
    PORTB = 0b00101000;
    // Set value of the 2nd transistor
    PORTC = 0b00010000;
    delay = 0;
    while (delay < time) {
        delay += 1;
    }

    SPDR = calculateIdentityFor(1) + 128;
    // Turn off the SS Pin to configure communication and start transmission
    PORTB = 0b00101000;
    // Set value of the 1st transistor
    PORTC = 0b00001000;
    delay = 0;
    while (delay < time) {
        delay += 1;
    }

    SPDR = calculateIdentityFor(0);
    // Turn off the SS Pin to configure communication and start transmission
    PORTB = 0b00101000;
    // Set value of the 0th transistor
    PORTC = 0b00000100;
    delay = 0;
    while (delay < time) {
        delay += 1;
    }
}

int main (void)
{
    // MARK: Set the Button and interrupt flag to record input

    // Set the EIMSK to enable INT1
    EIMSK = 0b00000010;
```

```
    // Set the EICRA to detect every falling edge of INT1 and schedule interrupt
    EICRA = 0b00001000;
    // set the PCMK2 tells it to enable PCINT16
    PCMSK2 = 0b00000001;
    // Set the PCICR to enable the internal flag of PCINT16
    PCICR = 0b00000100;
    // Configure Internal Resistor of PD3 and PD0
    PORTD = 0b00001001;

    // MARK: Set the SPI outputs

    // Set the MOSI, SS, and SCK pin as outputs
    DDRB = 0b00101100;
    // Set the SPI to Master Mode, Set the SPI to 1 (enable), Set SPIE to 1 to
      enable interrupt
    SPCR = 0b11010000;
    // Lets us divide our clock frequency by 2 to get the 1MHz
    SPSR = 0b00000001;
    // Set the SREG to global interrupt to allow our ISR function to activate
    sei();

    // MARK: Set out clock pints to active

    // Set the compare value
    OCR1A = 25000;
    // Set the counter to CTC Mode
    TCCR1B |= (1 << WGM12);
    //Set Interrupt Compare Match
    TIMSK1 |= (1 << OCIE1A);
    // Set Pre-scaler to 8 and start the timer
    TCCR1B |= (1 << CS01);

    // MARK: Set up DDRC Pins for transistors

    // Enable PC2, PC3, PC4, and PC5 as outputs
    DDRC =  0b00111100;

    while (1) {

        // check counter for overflow
        checkForOverflow();

        // go through each display
        shiftThroughDisplayIndices(950);

    }


}
```