University of Massachusetts Amherst
College of Engineering
Department of Electrical and Computer Engineering

ECE 231 - Introduction to Embedded System - Spring 2020
Instructors: B. Soules and F. Anwar

Lab Report

Lab 2
Stopwatch

STUDENT A: Ashton Gray
Spire ID: 32000589

STUDENT B: Bradford Gill
Spire ID: 31871838

Report Submission Date: 9 April 2020

Description:
The objective of this lab was to understand how to work with a cpu and peripherals. Specifically in this case, the task was to understand and successfully use a counter in the cpu to communicate with a shift register that sent signals to a 4 digit led display. The whole system would count and display time passed, pause the clock, and reset the clock.

The code is really straight forward. An infinite while loop is used to cycle through each time interval (100ms), first you display the minutes, this is done by setting the correct binary value to the SPDR that correspond to the current minute, turning off every digit on the led besides the minutes digit by setting portc, transferring the data from the SPDR to the shift register, and then pausing for a period of time so the number shows. This is repeated for seconds(first digit), seconds(second digit), and tenths of seconds. At the end of the while loop the program checks to see if the tenths should roll over to 0 and increase the next significant bit. This is the part that counts. If the user inputs to pause the function there is a variable that increase by 1 every time you hit the button. If the variable is odd, the program will not increment any time values(tenths of seconds), hence the display is paused. If the user wants to reset the clock they can press a button which causes an if statement to reset all values of tenths, seconds and minutes.

Conclusion:
From this project, the student was able to make seven working test programs, one final script, as well as a fully functioning circuit. The circuit allows the user to hit a button that will start and pause a timer on four, seven-segment displays, showing the minutes, tens of seconds, seconds, and tenths of seconds, along with a secondary button that can restart the counter and set all values to zero. While constructing the circuit and building the test programs, the student had to learn how to use a shift register, which will load one bit at a time and output these bits one at a time per the clock cycle. We would send this data from our shift register to the display, where we had the next task of multiplexing. Due to the fact that the values on the display were changing and could be different, the values projected on each of the seven segment displays had to be updated separately. This means, for example. The minutes display updated first, while the other displays were off, then the tens of seconds display updated while the others were off, and this

continued each of the displays. This was able to be done by using transistors that acted like switches, allowing to switch only one of the four seven-segment displays on at a time. This cycle then had to be done fast enough to where it seemed like each of the displays were on at the same time to the human eye.

In the future, this embedded system could be enhanced by using a different display that used a ":" to distinguish between minutes and seconds, as well as could have used more displays to go to tens of minutes, hours, or even hundredths of seconds which many stopwatches use such as on a phone. A feature where the stopwatch could stop at an inputted time and set off some sort of alarm, such as with a timer could also be implemented.

Appendix A.

```c
/*
 * Project1GoodVersion.c
 *
 * Created: 2/10/2020 2:25:47 PM
 * Author : Ashton, brad
 */

#include <avr/io.h>

// Test 8 Final
// note: brad is awful at spelling

int main(void)
{
    // ---- initialize outputs and variables ----- //
    DDRB = (1<<DDB3) | (1<<DDB5) | (1<<DDB2); // output for MOSI, SCK, and SS
    DDRC = (1<<DDC0) | (1<<DDC1) | (1<<DDC2) | (1<<DDC3); // output C
    SPCR = (1<<SPIE) | (1<<SPE) | (1<<MSTR) | (0<<SPR0); //changing clock will
      effect speed pf output
    SPSR = (1<<SPI2X);

    uint8_t number_map[10] = {0b00111111, 0b00000110, 0b01011011, 0b01001111,
      0b01100110, 0b01101101, 0b01111101, 0b00000111, 0b01111111, 0b01100111}; //
      0-9

    uint8_t minutes = 0;
    uint8_t seconds_tens = 0;
    uint8_t seconds_ones = 0;
    uint8_t tenths = 0;


    // ----- timing ----- //

    TCCR1B = 0b00000010;// set pre-scaler value (for clock)
    TCNT1 = 0; // clock counter, important to note TCNT1 cant get bigger than 2^16
      (65536)(16 bits)
    uint16_t interval = 25000; // how many clock cycles in 1/10th of a second,
    uint16_t pause = 400; // cannot be larger than 2^16 bc of how it works with
      TCNT1 (16 bits)
    uint16_t t_hold; // value to use in hold

    char stopped = 1; // if stopped is odd time will not count, if it is even it
      will

    while(1)
    {

        // display minutes
        SPDR = number_map[minutes]; // loading data register
```

```
        PORTC = 0b00000001;
        PORTB = 0b00000000; // SS low, start sending
        while(!(SPSR & (1<<SPIF))); // sends until stops data transfer
        PORTB = 0b00000100; // SS High, stop sending

        t_hold = TCNT1; // pause
        while(pause + t_hold> TCNT1);

        // seconds (tens)
        SPDR = number_map[seconds_tens];
        PORTC = 0b00000010;
        PORTB = 0b00000000; // SS low, start sending
        while(!(SPSR & (1<<SPIF))); // stops data transfer
        PORTB = 0b00000100; // SS High, stop sending

        t_hold = TCNT1;
        while(pause + t_hold> TCNT1);

        // seconds (ones)
        SPDR = number_map[seconds_ones] + 0b10000000; // 2^7 to put in the decimal
        PORTC = 0b00000100;
        PORTB = 0b00000000; // SS low, start sending
        while(!(SPSR & (1<<SPIF))); // stops data transfer
        PORTB = 0b00000100; // SS High, stop sending

        t_hold = TCNT1;
        while(pause + t_hold> TCNT1);

        // tenths
        SPDR = number_map[tenths];
        PORTC = 0b00001000;
        PORTB = 0b00000001; // SS low, start sending
        while(!(SPSR & (1<<SPIF))); // stops data transfer
        PORTB = 0b00000100; // SS High, stop sending

        t_hold = TCNT1;
        while(pause + t_hold> TCNT1);

        //reset
        PORTD = 0b00001001; // pd3
        if(PIND == 0b00000001){
            minutes = 0;
            seconds_tens = 0;
            seconds_ones = 0;
            tenths = 0;
            stopped = 0;
        }

        //pause
```

```
        PORTD = 0b00001001; // pd0
        if(PIND == 0b000001000){
            stopped ++;
            t_hold = TCNT1;
            while(t_hold + 30000 > TCNT1); // prevent double pressing
        }

        if(TCNT1 > interval && stopped % 2 == 1){
            TCNT1 = 0; // if this wasn't here TCNT1 could over flow causing minor
              issues
        }

        if(TCNT1 > interval && stopped % 2 == 0){
            TCNT1 = 0;
            tenths++; // increase data
            if (tenths >= 10){ // reset to 0 if 10
                tenths = 0;
                seconds_ones ++;
                if (seconds_ones >= 10){
                    seconds_ones = 0;
                    seconds_tens++;
                    if (seconds_tens >= 6){
                        seconds_tens = 0;
                        minutes ++;
                        if (minutes >= 10){
                            minutes = 0;
                        }
                    }
                }
            }
        }
    }
}
```