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 number and title....)

STUDENT A: Luan Vo

Spire ID: 30553600

STUDENT B: Jeffrey Matheson

Spire ID:

Report submission date: 25 March 2020

**Introduction**:

The objective of this lab is to understand the working of a shift register and a 7-segment timer display. Combine this knowledge with our understanding of AVR and the ATmega328p. With the combined knowledge we will then program a timer/stopwatch that counts upward and then be able to toggle/reset the stopwatch at the push of two switches. Ultimately, we want to create a watch that runs as accurately as possible through the use of multiplexing the remaining digits with the first digit.

**Materials and Hardware Description**:

C3 & C4 22 pF (pico-farad) ceramic capacitors (non-polarized)

(22 pF (picofarad) = 22 trillionths (10−12) of a farad)

* The capacitors are used to smoothening out any noises in input signals since we want the current and voltage to be constantly similar throughout the circuits that’s why it is necessary to add these capacitors to provide better accuracy.
* Sources: Charles Platt, *Encyclopedia of electronics components Vol. 1* (O’REILLY 2013) 97

R1 - R9 390 ohms each, eight current-limiting resistors

(eight for the anodes of the LED module and one for the test LED)

* These eight resistors will limit the currents going into the 7-segment timer display, so that not making the display too bright, damaging the equipment and
* Sources: Charles Platt, Encyclopedia of electronics components Vol. 1 (O’REILLY 2013) 75

R10 10 kohms, to greatly diminish the current (brightness) of the power on LED.

* This resistor makes it so that the LED won’t be too bright
* Sources: Charles Platt, *Encyclopedia of electronics components Vol. 1* (O’REILLY 2013) 75

SW1 momentary switch, N.O. (normally open)

and

SW2

* This switch is the toggle between stop/start and reset of the timer.

T1-T4 2N7000, N-Channel Field Effect Transistor (FET), up to 60V, up to 200mA,

Through Hole Part TO-92-3 package

[Digi-key Part Number: 2N7000FS-ND](https://www.digikey.com/product-detail/en/on-semiconductor/2N7000/2N7000FS-ND/244278)

* These transistors’ sources are connected to the cathode of the 7-segment display then the gates are connected to the output of the microprocessor and the drains are connected to ground.
* Sources: from the lab2\_hardware\_transistor\_fet\_2n7000.PDF in piazza

U1 ATmega328P microcontroller (aka 328P or simply the AVR)

[Digi-key Part Number: ATMEGA328P-PU-ND](https://www.digikey.com/product-detail/en/microchip-technology/ATMEGA328P-PU/ATMEGA328P-PU-ND/1914589)

* This is the main-out of the circuit, used to write instructions to the secondary-in shift register, and the outputs are being connected to the transistors so that we can control which digits are being shown/displayed.
* Sources: ATmega328P\_full\_datasheet\_662pages.pdf piazza

U2 74HC595N CMOS shift register (8-Bit shift register with 3-State output registers)

[Digi-key Part Number: 296-36142-5-ND](https://www.digikey.com/product-detail/en/texas-instruments/SN74HC595NE4/296-36142-5-ND/1571270)

* The shift register (sequential logic circuit) is responsible for storing and movement of data within the circuit. Used to convert parallel data to serial or serial to parallel.
* Sources: lab2\_hardware\_shift\_register\_74hc595.pdf piazza

U3 TDCR1060M Clock Display (Four-Digit Seven-Segment LED module by Vishay)

* To display of timer of the circuit.
* Sources: lab2\_hardware\_4\_digit\_led\_module\_tdcr1060M\_by\_vi piazza

X1 16 MHz quartz crystal

* The crystal is the one that determines the frequency of the circuit through crystal’s vibration.
* Sources: Abracon, Half Size DIP Low Voltage 5.0V Crystal Clock Oscillator Datasheet (Abracon, 2016).
* **Miscellaneous equipment:**

* + USB Logic Analyzer - 25MHz/8-Channel from Sparkfun

[Sparkfun part number: TOL15033](https://www.sparkfun.com/products/15033)

* + Atmel ICE debugger with Atmel ICE flat cable

Digi-Key Part Number: ATATMEL-ICE-ND

* + Two (2) A-Male to Micro B Charger Cables
  + One (1) A-Male to Mini B Charger Cable (for Logic Analyzer)
  + Adafruit USB micro-B breakout board (to power your circuit)

Digi-key Part Number: 1528-1383-ND

* + Solderless Breadboard Terminal Strip (No Frame)
  + Digi-Key Part Number: 1528-2143-ND

**Circuit Description:**

- 5 vdc powers the circuit and inputs to pin 7,20, 21 of microprocessor. As well as going into pin 10,16 of shift register.

- 2 switches with pull-up resistors are connected to the microprocessor pin 2, 5 so when they are pressed, it will change the state of the input/output of microprocessor by changing the current going to ground instead.

- 2 capacitors are connected with the crystal quartz so that it will increase the accuracy of the fluctuation and only smoothening out the input to the microprocessor.

- All resistors are used to limit the current going into the equipment such as the LEDs and the 7-segment timer display, so that it will not damage these tools.

- we connect MOSI, SS, CLK from the Atmega328p (defined by Atmel ICE), to the SER, RCLK and SRCLK of the shift register. We also connect Vcc and SRLCK compliment to 5Vdc as these needs to be constantly high input. In addition, OE complimented and ground of shift register need to be connected to ground as these pins need to always be at low state.

- We then connect the outputs of the shift register from (Q\_A to Q\_H) to the anodes and cathodes of the 7-segment timer display. (where the anodes is where the current flow in, and the cathode is where currents flow out) this set up help us manipulate the data flow to the timer (from shift register) and the digits being displayed.

- The output ports (defined in our case) are portC array, from pin 25 to 28 as the output ports that manipulate the gate voltage of the NMOS so that we can allow the segment to get turned on or off and to manipulate the changing in digit of the timer.

**Software Description**:

* For the final program we initialized the SPI with this function:

void setSPI(void) {

//init values

SPCR = (1<<SPE)|(1<<MSTR)|(0<<SPR0|(0<<SPR1)); //SPE enables SPI, MSTR makes it main, SPR0 and SPR1 select freq

SPSR = (1<<SPI2X); //with SPR0 and SPR1 selects freq

DDRB = (1<<DDB3)|(1<<DDB5)|(1<<DDB2); //makes SCK, MOSI, and SS' outputs

DDRC = (1<<DDC5)|(1<<DDC4)|(1<<DDC3)|(1<<DDC2);

TCCR1B = 0b00000011;  //sets up the 64 prescalar value, initiates timer1

}

* + Where SPCR is the initialization of the role (Main or Secondary) and the set up of SPI as well as frequency through clock division. In our case, the clock division is 2: (¼)\*2
  + DDRC sets the respective port on, ready for data transmission.
  + DDRB sets the respective ports as out put for SCK, MOSI, SS’
  + TCCR1B is used to set up the prescaler (in our case is 64)
* For the digits being displayed, we found through mathematical expression that each segment (A,B,C,D,E,F,G,dot) (from datasheet) is turned on respectively by 2^n, where A segment is 2^0, B segment is 2^1 and so on until n = 7.
  + So, for example: 7 = A + B + C = 1 + 2 + 4 = 7 (in decimal)
  + The decimal for dot is: 128
  + If we want a certain digit to be displayed, we just need to add up all the segments to get the desired digit.
  + We created 3 functions: for digits of 0 to 9, digits of 0 to 9 with dot, digits of 0 to 5 (for ten second place only goes up to 5)
  + The sample code is here for digits with dot:

void digitDisplay\_dot(int a, char digitDisplay[])

{

switch(digitDisplay[a]) {

case(191):  // if its 0, with dot

digitDisplay[a]=134;  //make it 1, with dot

break;

case(134):  //if its 1, with dot

digitDisplay[a]=219;  //make it 2, with dot

break;

case(219):  //if its 2, with dot

digitDisplay[a]=207;  //make it 3, with dot

break;

case(207):  //if its 3, with dot

digitDisplay[a]=230;  //make it 4, with dot

break;

case(230):  //if its 4, with dot

digitDisplay[a]=237;  //make it 5, with dot

break;

case(237):  //if its 5, with dot

digitDisplay[a]=253;  //make it 6, with dot

break;

case(253):  //if its 6, with dot

digitDisplay[a]=135;  //make it 7, with dot

break;

case(135):  //if its 7, with dot

digitDisplay[a]=255;  //make it 8, with dot

break;

case(255):  //if its 8, with dot

digitDisplay[a]=231;  //make it 9, with dot

break;

case(231):  //if its 9, with dot

digitDisplay[a]=191;  //make it 0, with dot

break;

}

}

* This is our function for data transmission:

void dataTransmission(*uint8\_t* data){

SPDR = data;

while(!(SPSR & (1<<SPIF)))  //waits for the current transmission to finish

{

}

}

* + Where the while loop is the wait time, so that we can transmit the data at a frequency by the formula: t = 1/f
* This is our function for multiplexing, switch digit display and port toggling

PORTC = portON[digit];

setSPI();  //sets up the SPI

dataTransmission(digitDisplay[temp]); //sets up the data for transmission

//have the timer TCNT1 to be zero as to reset the value to zero whenever the loop is done running

TCNT1 = 0;

//setting the digits to be on (by making them flashing really fast)

while(TCNT1 < 100){}

//needed for transmission and to start next transmission by toggling the portB twice so that the data is stored and written down to the shift register

//this port is the SS

PORTB ^= (1<<PORTB2);

PORTB ^= (1<<PORTB2);

//multiplexing part, by selecting other digits based on the change in the last digit

// selects the different display digits

if(digit !=3){digit++;}

else {digit = 0;}

// selects the different digit value

if(temp != 3){temp++;}

else {temp = 0;}

* + After this step, we multiplex other digits based on the first digit, i.e the 10th of a sec, because we only have 1 shift register, this is the best way to utilize the resources. I.e, once the last digit hit 9, we will increase the next digit by 1, this is the code for it:

//millisecond digit

if(mSec >= 1561){    //1561 is 0.1s (1000000/64/10)-1

mSec = 0;  //resets the temp

digitDisplay\_noDot(3,digitDisplay);

}

//seconds digit

if(seconds >= 15619){

if(digitDisplay[3] == 63){  //checks if the milliseconds have hit 0

seconds = 0;  //resets the temp

digitDisplay\_dot(0,digitDisplay);

}

//ten-seconds digit

if(1){

if(digitDisplay[0] == 191 && digitDisplay[3] == 63){  //checks if the milliseconds and seconds have hit 0

sec10th = 0;   //resets the temp

minutes = 1;  //sets minutes so that the the minutes can start next cycle

digitDisplay\_10sec(1,digitDisplay);

}

}

//minute digit

if(minutes == 1){  //minutes == 1 is used because minutes = 0 only occurs at the very beginning, which is when minutes should be 0

if(digitDisplay[1] == 63 && digitDisplay[0] == 191){  // checks if the seconds are at 0

minutes = 0;  //resets the temp

digitDisplay\_dot(2,digitDisplay);

}

}

}

mSec = mSec + 53;

seconds = seconds + 53;

sec10th = sec10th + 52;

* + Note that, since we used the hardware timer TCNT1 for the quick flashing in previous code, we cannot use that again. Instead of reusing TCNT1, we calculate the number of ticks needed to for 100ms in term of 64 prescaler and that is 1561 ticks, and since seconds is just 10\* 100ms, so the ticks for that would be 15619. From then we would create a code that replaces the normal operation of tcnt1 by making a while loop that wait for that amount of time, by the use of incrementing the mSec, secs, sec10th respectively by given amount so that the loop will be as close to 100ms as possible.

**Problems Encountered and Solutions**:

- We encountered the issue of unable to enter programing mode, and we solved it by fixing the wires from the logic analyzer to have both grounds so that the currents have a ground to flow to.

- We encountered a problem that the timer turns on all digits while display, at any given time the problem is that the wires that comes from the microprocessor’s outputs to transistor and from the shift register to the timer’s cathodes are off by 1 pin each.

- Also, we encountered that the segments sometimes are blurred/lost in display, the problem is that the shift register is “loose” as we have to keep pinning the shift register down to the board tightly whenever we get fragmented display.

- We found a problem with our program that sometimes, the display will be faster or slower when compared to the real timer on phone. Therefore we made the adjustment that we change the interval of the loop for seconds, msec and sec10th so that we achieve the best result. We theorized that since the loop is running too fast if we have all at +50 for all, since this would be the equivalent of 30 loops (by 1561/50) to 100ms. But since it is too slow, we changed from all loop operating at 50 to 53 and that was too quick. So by changing the increment of sec10th (the most inner loop) to be 1 lower than the actual value of 53, we deliberately make the innermost loop goes slower to compensate for it being too fast, thus slower the speed down to almost 100ms.

**Conclusions**:

This lab taught us how to correctly set up a 7-segment display and how to use multiplexing in practice. Also, we learned about the data transmission procedure of the shift register, such as the reason behind toggling SS twice, the Main out Secondary in, and the significant behind rising and falling edge of the clock. We also learned how to identify the difference between the anode and cathode of the timer, hence the reason why we have two test LED to simulate the simpler version of the timer. Also, it is important to clamped down the current with resistors if we don’t want to damage these tools, and most importantly how capacitors can greatly increase the accuracy and consistency of the circuit through its noise cancelling. Most importantly is that we learned how a shift register operate by using the logic analyzer to tap into the output and input of the register, and that we learned how to transmit data from the microprocessor to shift register as a mean to perform serial data transmission hence, the need for SPI (serial peripheral interface).

**References**: List works you made use of in the course of completing the lab. (10 points)