

ISTANBUL TECHNICAL UNIVERSITY
COMPUTER ENGINEERING DEPARTMENT

BLG 351E
MICROCOMPUTER LABORATORY
EXPERIMENT REPORT

EXPERIMENT NO : 6
EXPERIMENT DATE : 21.12.2020
LAB SESSION : FRIDAY - 08.30
GROUP NO : G10

GROUP MEMBERS:

150180028 : ABDÜLKADİR PAZAR
150180103 : EGEMEN GÜLSERLİLER
150180716 : AHMET SELÇUK TUNCER

FALL 2020

Contents

FRONT COVER

CONTENTS

1	INTRODUCTION	1
2	MATERIALS AND METHODS	1
2.1	Part 1	1
2.2	Part 2	4
2.3	Part 3	6
2.4	Part 4	8
3	RESULTS	10
3.1	Part 1	10
3.2	Part 2	10
3.3	Part 3	11
3.4	Part 4	12
4	DISCUSSION	13
5	CONCLUSION	14

1 INTRODUCTION

In this experiment, we implemented timer interrupt functions and observed these timer interrupts in LED's, 7 segment displays and LCD.

2 MATERIALS AND METHODS

2.1 Part 1


At part 1, we used LED's and we displayed two patterns using timer interrupt and interrupt functions.

```
void setup()
{
  DDRB = 255;
  DDRD = 0b11110000;
  //set timer1 interrupt at 1Hz
  TCCR1A = 0; // set entire TCCR1A register to 0
  TCCR1B = 0; // same for TCCR1B
  TCNT1 = 0; // initialize counter value to 0
  // set compare match register for 1hz increments
  OCR1A = 15624; // = (16*10^6) / (1*1024) - 1
  // turn on CTC mode
  TCCR1B |= (1 << WGM12);
  // Set CS12 and CS10 bits for 1024 prescaler
  TCCR1B |= (1 << CS12) | (1 << CS10);
  // enable timer compare interrupt
  TIMSK1 |= (1 << OCIE1A);

  attachInterrupt(digitalPinToInterrupt(3), swap, RISING);
}
```

Figure 1: Setup Phase

We needed two int to create each pattern. A boolean variable (first_one) to determine which pattern we are on and another boolean variable (which_way) to use in pattern 2 since the pattern will continue in opposite direction.



```
int seq1 = 16, seq2 = 8, seq3 = 128, seq4 = 1, display = 0;
bool first_one = true, which_way = true;
```

Figure 2: Variables

Figure below is the swap function which is being called by the external interrupt to change the pattern that is being displayed on the LEDs.

```

void swap()
{
    first_one = !first_one;
}

```

Figure 3: External Interrupt Function

Since we circularly shift numbers at both of the patterns. We opted to write two functions that circularly shift numbers.

```

int csr(int a)
{
    a = a/2;
    if(a<1)
    {
        a = 128;
    }
    return a;
}

int csl(int a)
{
    a = a*2;
    if(a>128)
    {
        a = 1;
    }
    return a;
}

```

Figure 4: Shift Functions

And finally below figures are the implementations of patterns. Both are a part of the timer interrupt.

```

ISR(TIMER1_COMPA_vect)
{
    if(first_one)
    {
        display = seq1+seq2;
        PORTD = (display%4)*64;
        PORTB = display/4;
        seq3 = 128;
        seq4 = 1;
        seq1 = csl(seq1);
        seq2 = csr(seq2);
    }
}

```

Figure 5: First Pattern

```

else
{
    display = seq3+seq4;
    PORTD = (display%4)*64;
    PORTB = display/4;
    seq3 = (seq3+128)%256;
    if(which_way)
    {
        seq4 = csl(seq4);
        if(seq4==128)
            which_way = !which_way;
    }
    else
    {
        seq4 = csr(seq4);
        if(seq4==1)
            which_way = !which_way;
    }

    seq1 = 16;
    seq2 = 8;
}
}

```

Figure 6: Second Pattern

2.2 Part 2

In this part, we implemented a stopwatch using Timer Interrupts and External Interrupts. Since we are counting centiseconds timer interrupt frequency should be 100Hz.

```
int counter = 0, lap_counter = 0, lap_number = 0;
int bout[16] = {0,4,1,0,4,2,2,0,0,0,0,6,3,4,3,3}; //mapping array for PortB
int dout[16] = {16,240,32,96,192,64,0,240,0,64,128,0,16,32,0,128}; //mapping array for PortD
bool started = false;
int B = 0;
void setup()
{
  DDRB = 0b000111;
  DDRD = 0b11110000;
  DDRC = 255;
  // TIMER 1 for interrupt frequency 100 Hz:
  cli(); // stop interrupts
  TCCR1A = 0; // set entire TCCR1A register to 0
  TCCR1B = 0; // same for TCCR1B
  TCNT1 = 0; // initialize counter value to 0
  // set compare match register for 100 Hz increments
  OCR1A = 19999; // = 16000000 / (8 * 100) - 1
  // turn on CTC mode
  TCCR1B |= (1 << WGM12);
  // Set CS12, CS11 and CS10 bits for 8 prescaler
  TCCR1B |= (0 << CS12) | (1 << CS11) | (0 << CS10);
  // enable timer compare interrupt
  TIMSK1 |= (1 << OCIE1A);
  sei(); // allow interrupts
  attachInterrupt(digitalPinToInterrupt(2), start, RISING);
  attachInterrupt(digitalPinToInterrupt(3), lap, RISING);
  Serial.begin(9600);
}
```

Figure 7: Setup Phase

Mapping logic of PORTD and PORTB values to seven segment displays are shown in the table below.

	A	B	C	D	E	F	G	PORTB (A+B+C)	PORTD (D+E+F+G<<4)
0	0	0	0	0	0	0	1	0	16
1	1	0	0	1	1	1	1	4	240
2	0	0	1	0	0	1	0	1	32
3	0	0	0	0	1	1	0	0	96
4	1	0	0	1	1	0	0	4	192
5	0	1	0	0	1	0	0	2	64
6	0	1	0	0	0	0	0	2	0
7	0	0	0	1	1	1	1	0	240
8	0	0	0	0	0	0	0	0	0

Table 1: Mapping logic for seven segment displays

We use loop function to display the counter value on the seven segment displays and check if the reset button is clicked. To check if reset button is clicked we mask the PINB value to variable B. If button is pressed B=32 else B=0. When button is pressed it calls the reset function.

```

void loop()
{
  if(millis()%40 == 9)
  {
    PORTC = 28;
    PORTB = bout[counter/1000];
    PORTD = dout[counter/1000];
  }
  if(millis()%40 == 19)
  {
    PORTC = 44;
    PORTB = bout[(counter/100)%10];
    PORTD = dout[(counter/100)%10];
  }
  if(millis()%40 == 29)
  {
    PORTC = 52;
    PORTB = bout[(counter/10)%10];
    PORTD = dout[(counter/10)%10];
  }
  if(millis()%40 == 39)
  {
    PORTC = 56;
    PORTB = bout[counter%10];
    PORTD = dout[counter%10];
  }
  B = PINB&32;
  if(B)
  {
    reset();
  }
}

```

Figure 8: Loop

The start function which is attached to an external interrupt only changes the boolean variable "started". But as long as started is false nothing is happening on the circuit in terms of counting, we display 4 zeros since stopwatch isn't counting. With timer interrupt we increment the counter and lap_counter values by one since we are counting every centisecond. Lap function which is attached to another external interrupt resets the lap_counter, increments lap_number and prints lap number lap time and total time to Serial Monitor. And the reset function resets all variables to their initial values and stops the counting.

```

ISR(TIMER1_COMPA_vect)
{
    if(started)
    {
        counter++;
        lap_counter++;
    }
}

void start()
{
    started = true;
}

void reset()
{
    counter = 0;
    started = false;
    lap_counter = 0;
    lap_number = 0;
}

void lap()
{
    if(started)
    {
        lap_number++;
        Serial.print("lap number: ");
        Serial.println(lap_number);
        Serial.print("lap time: ");
        Serial.println(lap_counter);
        Serial.print("total time: ");
        Serial.println(counter);
        lap_counter = 0;
    }
}

```

Figure 9: Start, Stop and Lap Functions

2.3 Part 3

In this part, we displayed our names on an LCD screen. We used the given `initLCD`, `waitMillis` and `WaitMicros` functions. Setup phase is also fairly standard. It just sets all pins in `PORTB` and `PORTD` as output. We implemented `sendChar` and `sendCMD` functions. They are pretty similar. Only difference between them is `sendCMD` sets RS value as 0 whereas `sendChar` sets it as 1. In both functions we take the data as characters and shift them 4 bits so that they are in the correct place to be sent over to the LCD.

```

void sendChar(unsigned char data)
{
    PORTD = 0b00000100 | (data << 4); //rs = 1 and data is shifted
    triggerEnable(); //enable lcd
    waitMicros(55); //55 us delay
}

void sendCMD(unsigned char data)
{
    PORTD = 0b00000000 | (data << 4); //rs = 0 and data is shifted
    triggerEnable(); //enable lcd
    waitMicros(55); //55 us delay
}

```


And finally the `triggerEnable` function. Simply enables the LCD waits for 50 microseconds then disables it back.

```
void triggerEnable()//enable wait then disable lcd
{
    PORTB = 1;
    waitMicros(50);
    PORTB = 0;
}
```

At loop we send necessary functions one by one to write our names on the LCD. We called `initLCD()` before sending the data to the LCD so that the screen would be empty when writing a new name.

```
sendChar(0b0100); //E
sendChar(0b0101);

sendChar(0b0100); //G
sendChar(0b0111);

sendChar(0b0100); //E
sendChar(0b0101);

sendChar(0b0100); //M
sendChar(0b1101);

sendChar(0b0100); //E
sendChar(0b0101);

sendChar(0b0100); //N
sendChar(0b1110);

//newline
sendCMD(0b1010);
sendCMD(0b1000);

sendChar(0b0100); //G
sendChar(0b0111);

sendChar(0b0101); //U
sendChar(0b0101);

sendChar(0b0100); //L
sendChar(0b1100);

sendChar(0b0101); //S
sendChar(0b0011);

//
```

Figure 10: A part of Loop3

2.4 Part 4

Here we used the 1Hz timer interrupt to count down second by second similar to part 1. We used all the functions related with LCD from Part 3. This time we printed a countdown and similar to part3 we did this in loop. Timer interrupt counts down the counter every second until counter hits 0. And to display seconds minutes and hours we used display_number and display_time functions below.

```
ISR(TIMER1_COMPA_vect)//decrement counter once per second
{
    if(counter)//decrement if countdown hasnt ended
        counter--;
}

void display_time()//display time in hh:mm:ss
{
    int hour = counter / 3600;//hour of counter
    int minute = (counter % 3600) / 60;//minute of counter
    int second = counter % 60;//second of counter

    display_number(hour/10);//tens digit of hour
    display_number(hour%10);//ones digit of hour

    sendChar(0b0011);//':' character
    sendChar(0b1010);|

    display_number(minute/10);//tens digit of minute
    display_number(minute%10);//ones digit of minute

    sendChar(0b0011);//':' character
    sendChar(0b1010);

    display_number(second/10);//tens digit of second
    display_number(second%10);//ones digit of second
}

void display_number(unsigned char num)//display digits 0-9
{
    sendChar(0b0011);//upper 4 bits are same for digits
    sendChar(num);//lower 4 bits are number in binary
}
```

Figure 11: Timer Interrupt and Displaying Timer

And at last we check in loop if the counter has reached 0. If it has we send a signal to the piezo.

```

//T
sendChar(0b0101);
sendChar(0b0100);
//I
sendChar(0b0100);
sendChar(0b1001);
//M
sendChar(0b0100);
sendChar(0b1101);
//E
sendChar(0b0100);
sendChar(0b0101);
//R
sendChar(0b0101);
sendChar(0b0010);
//:
sendChar(0b0011);
sendChar(0b1010);

//newline
sendCMD(0b1010);
sendCMD(0b1000);

display_time();//display time on lcd display
waitMillis(1000);

if(!counter)//send signal to piezo when counter = 0
{
    PORTB = 8;
}
}

```

Figure 12: A part of Loop4

3 RESULTS

3.1 Part 1

At the end of the first task, the team is succeeded in implementing given patterns on the leds. Implementation of this part is mentioned in detail at the material and methods section. Basically, by default our program generates the first pattern showed in figure 1.a and if button is clicked it switched to second pattern which is showed in figure 1.b.

time	led pattern
0	00011000
1	00100100
2	01000010
3	10000001
4	10000001
5	01000010
6	00100100
7	00011000
8	00011000
9	00100100
10	01000010
11	10000001
12	10000001
13	01000010
14	00100100

a)

time	led pattern
0	10000001
1	00000010
2	10000100
3	00001000
4	10010000
5	00100000
6	11000000
7	10000000
8	11000000
9	00100000
10	10010000
11	00001000
12	10000100
13	00000010
14	10000001

b)

Figure 13: Part 1

3.2 Part 2

In the second part, the team implemented a stopwatch that counts time in centiseconds. Implementation of this part is mentioned in detail at the material and methods section. As asked, two left most 7-segment display is reserved for seconds and other two for centiseconds. First button achieved to starting of stopwatch, second works as a lap button and the last one is reset button. In addition, serial monitor displays desired values; lap time, lap count and total time.

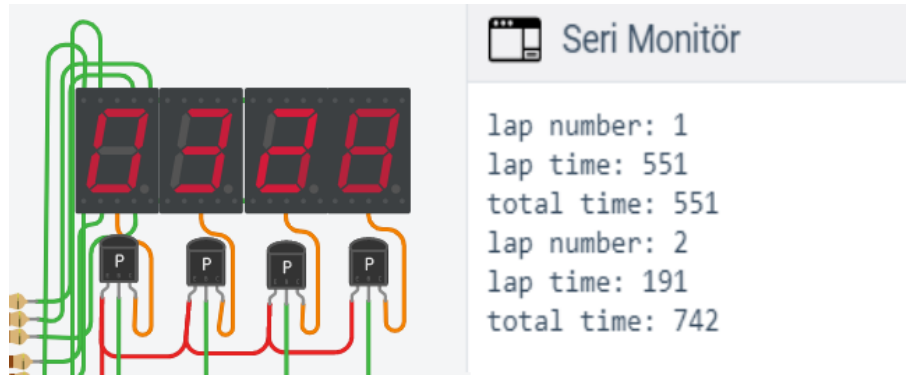


Figure 14: Part 2

3.3 Part 3

In the third part, the team implemented the program that drives 16x2 dot matrix LCD. Two desired task is achieved respectively, at first we configure the LCD display in order to communicate in 4-bit(D4-D7) mode and then we send 8-bit ASCII characters as nibbles (4 bits) to display using the specific instruction. We use five subroutines that given in the homework file as a guide to complete this task. At the end of the third part our names are displayed on the LCD.

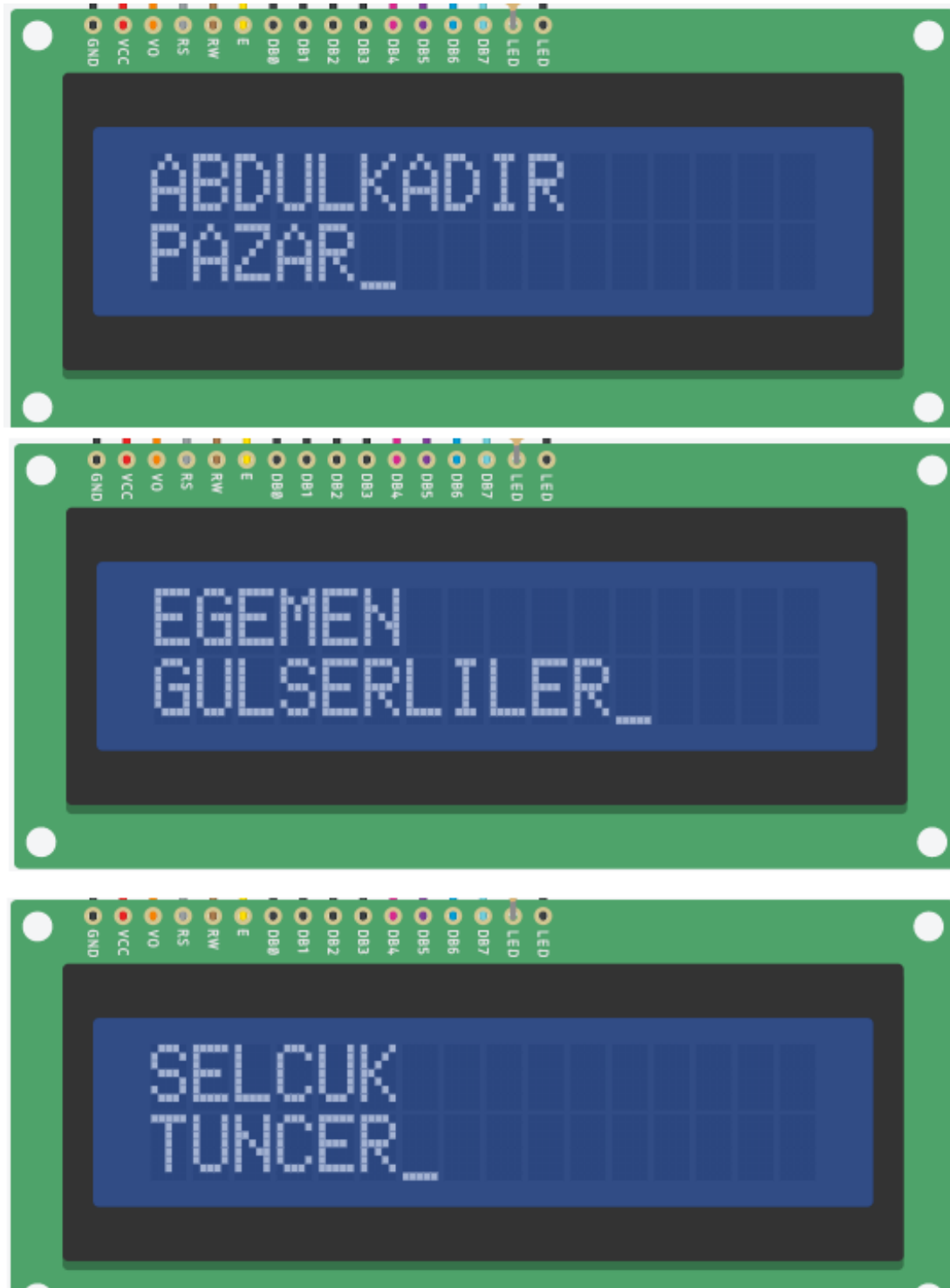


Figure 15: Part 3

3.4 Part 4

In the last part of the experiment, the team successfully implemented the countdown timer that can counts down in seconds, minutes, and hours using LCD and timer interrupt. Implementation of this part is mentioned in detail at the material and methods section.

At the end of the part 4, countdown timer is programmed such a way that it can be used with different countdown numbers and a sound from the buzzer rings at end of the timer.

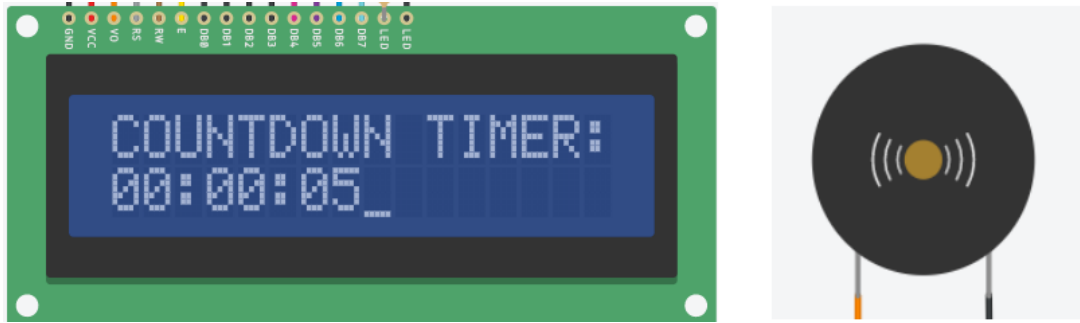


Figure 16: Part 4

4 DISCUSSION

In the first part of the experiment, timer interrupts were used alongside external interrupts. In the timer interrupt requested patterns were produced and displayed on LEDs. The external interrupt was a simple change in a boolean variable to change patterns. `csr()` and `csl()` functions we wrote made our task easy as we were able to produce patterns easily using these functions. This task was easy to achieve compared to the other tasks as we only had to deal with LED lights.

In the second part of the experiment, the given task was similar to the first task in that we used timer interrupts alongside external interrupts. We first tried to implement the 3rd button as an external interrupt but since `attachInterrupt()` function on Arduino Uno boards doesn't support buttons connected to pin 13, we used a simple if statement to check the button press.

In the third part of the experiment, we used an LCD display to write our names. We used the given `initLCD()` function to initialize the LCD display. Given delay functions didn't work as the compiler optimized away the while loop in the code so we wrote a new delay function. We implemented `sendCMD()` and `sendChar()` functions to send characters to the LCD display 4 bits at a time. We used the given manual to find the binary equivalents of letters of our names and used the loop function to display each group member continuously with a 2 second delay.

In the fourth and last part of the experiment, we used the LCD display and the piezo to implement a countdown timer that gives an alert sound when the countdown is finished. We entered the timer value in seconds as a variable to the program, implemented `display_time()` function to display time in hh:mm:ss format. The counter is decremented using timer interrupt subroutine. The sound is played when the counter reaches 0. This

part was easy to implement as we learned the basics of LCD display after completing part 3.

5 CONCLUSION

In this experiment we learned how to use timer interrupts. We used timer interrupts alongside external interrupts in first two parts. Learning the basics of LCD display proved tricky as most resources on the Internet used external libraries. This experiment overall was a good practice to understand the basics of Timer interrupts and interacting with LCD displays.