Experiment 6: Using Timer Interrupt on Arduino

Due date : 21.12.2020 12:00

Abdullah Ekrem Okur
okurabd@itu.edu.tr

1 Introduction

In this experiment, our aim is to discover the timer interrupt mechanism and implement different timer interrupt applications. Please investigate why we need timer interrupt mechanism and mention it briefly in your report. Circuit designs of each part will be given in sections below.

- **You can not use any loop statement and recursive functions except void loop.**
- **You can not use block programming provided by tinkercad.**
- **You should add comments for necessary cases. Otherwise, you will get zero point.**
- **You can not use digitalWrite and pinMode functions.**
- **You can not use delay function. Use millis or micros instead of delay.**
- **You can use waitMicros and waitMillis function in only initLCD, sendCMD, and sendChar functions.**
- **You can not use any library for 16x2 LCD Display such as "LiquidCrystal.h".**

For your questions: Abdullah Ekrem Okur (okurabd@itu.edu.tr)

2 Part-1

In this part, you will program the circuit whose design is presented in Figure 1 (Design link). You will write a code that generates the sequences in the Table 1. Your program should generate the first pattern by default. Whenever the button is clicked, the displayed pattern should be changed. You should use the timer interrupt to generate

patterns and void loop should be empty. It is enough to leave a noticeable range while showing the patterns on LEDs. Also, we expect you to implement interrupt subroutine for button press.

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

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

Table 1: Example sequence of leds with 1 meaning led is open and 0 meaning led is closed.

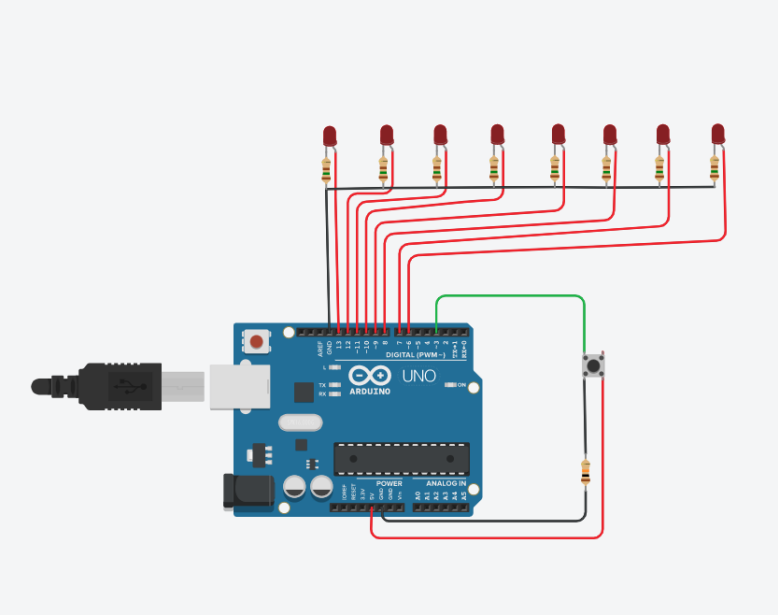


Figure 1: Circuit diagram of Part-1

3 Part-2

In this part, you are going to to implement a stopwatch that counts time in centisecond which is elapsed after you click the "Start" button (first button) on the design is given in Figure 2 (Design link). Two left most 7-segment displays should show the seconds and the other two should show the centiseconds of the stopwatch. Second button should work as a lap button. When the lap button is clicked, lap number, lap time and the total time will be displayed on the serial monitor. The third button is the reset button. Stopwatch should be reset as soon as the reset button is clicked. Timer interrupt should be used to keep track of how much time has passed and only display instructions should be in the void loop function. In addition, first and second button should be implemented as external interrupts. The third button can be simply checked with an if-else statement.

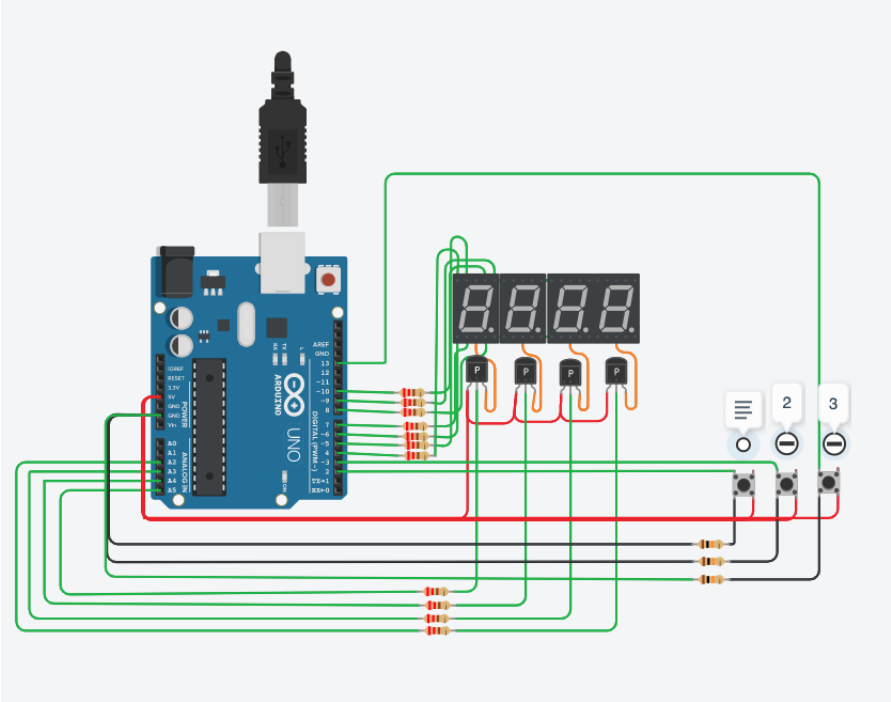


Figure 2: Circuit diagram of Part-2

4 Part-3

In the experiment, you are going to implement the program that drives 16x2 dot matrix LCD on the design is given in Figure 3 (Design link). You are going to implement a print function which uses char characters as inputs and show them on the LCD display. You should write your names to the first line and your surnames to the second line.

Background Information

There are two working modes of LCD display which require 8-bit and 4-bit connection bandwidth respectively. Since, the only upper nibble (D4-D7) of data bus are wired to microcontroller, we should use 4-bit working mode to utilize the LCD. This link contains list of commands for driving LCD. Please, remember that the LCD works in 8-bit mode by default. There are two things you need to do in this lab: Firstly you should configure the LCD display in order to communicate in 4-bit mode. Secondly you should send 8-bit ASCII characters as nibbles (4 bits) to display using the specific instruction. The flow chart that shows the steps of initialization and configuration of the LCD is given at the end of the experiment. More detail about this flow chart could be found in this link. In addition to this, the datasheet of the LCD display will be available in homework files.

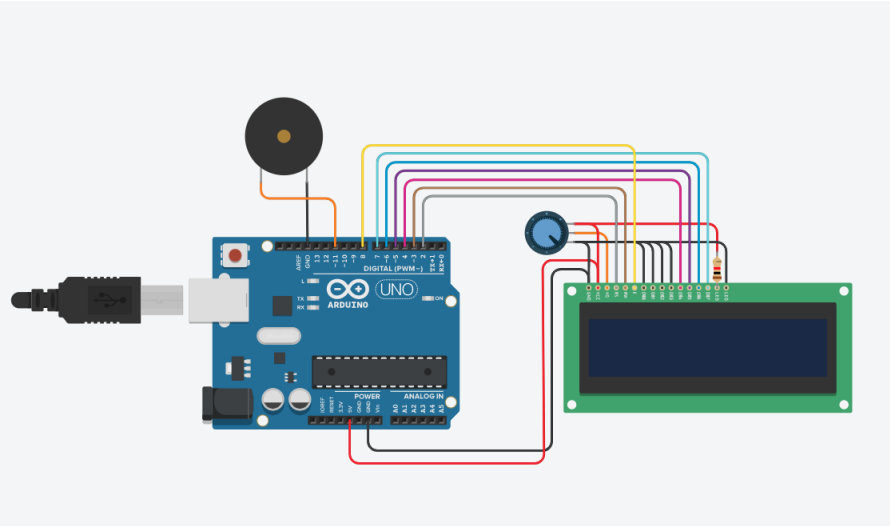


Figure 3: Circuit diagram of Part-3

The list of subroutines are given below to assist you while implementing your program. You can construct your program by using these subroutines as your building blocks or you may choose different approach to implement the requested program. In other words, these subroutines are not mandatory to implement, you are free to choose any other approach/subroutine set to implement your program.

1. **initLCD:** The steps of initialization and configuration (are given in the flow chart) can be implemented in this subroutine. However, you should be aware of the initial state of the LCD while writing this function. Initially, the LCD works in 8-bit mode until you configure it otherwise. Thus, you should send your first commands accordingly.

2. **sendCMD:** Works in 4 bit mode, first loads the upper nibble of the command to the output port and calls triggerEnable function to send the data to LCD, then repeats the same steps for the lower nibble.

3. **sendChar:** Works in 4 bit mode, differs from sendCMD in terms of RS input, similarly it first it loads the upper nibble of the command to the output port and calls triggerEnable function to send the data to LCD, then repeats the same steps for the lower nibble.

4. **triggerEnable:** first it changes the value of EN to high (1) then it changes it back to low (0).

5. **waitMillis and waitMillis:** busy-wait (loop) function to create necessary delays which is mandatory for LCD to function properly.

Remember to use the triggerEnable subroutine while sending the configuration commands in the flowchart.
Note: You should add 55 microseconds at the end of the sendCMD and sendChar functions.

You can find the implementation of initLCD and delay functions in Figure 5.

```
void initLCD(){
  PORTD= PORTD&B11110011; //Clear RS and R/W
  waitMillis(100);  //Wait 100 ms

  PORTD= (PORTD&B00001111)|B00110000; //Special case of 'Function Set'
  triggerEnable();      //Send Enable Signal
  waitMillis(5);   //Wait 5ms

  PORTD= (PORTD&B00001111)|B00110000; //Special case of 'Function Set'
  triggerEnable();      //Send Enable Signal
  waitMicros(150);

  PORTD= (PORTD&B00001111)|B00110000; //Function set, Interface is 8 bit longs
  triggerEnable();      //Send Enable Signal
  waitMicros(150);

  PORTD= (PORTD&B00001111)|B00100000; //Initial 'Function Set' to change interface
  triggerEnable();      //Send Enable Signal
  waitMicros(150);

  PORTD= (PORTD&B00001111)|B00100000; //'Function Set' DL=0 // Dataline 8bits
  triggerEnable();      //Send Enable Signal
  PORTD= (PORTD&B00001111)|B10000000; //'Function Set' N=1 //2 lines
                                      // F =0 5x8 dots
  triggerEnable();      //Send Enable Signal
  waitMicros(55);

  PORTD= (PORTD&B00001111); //Display On Of control
  triggerEnable();      //Send Enable Signal
  PORTD= (PORTD&B00001111)|B10000000;
  triggerEnable();      //Send Enable Signal
  waitMicros(55);

  PORTD= (PORTD&B00001111); //Clear display
  triggerEnable();      //Send Enable Signal
  PORTD= (PORTD&B00001111)|B00010000;
  triggerEnable();      //Send Enable Signal
  waitMillis(5);
```

```
  PORTD= (PORTD&B00001111); //Entry mode set
  triggerEnable();      //Send Enable Signal
  PORTD= (PORTD&B00001111)|B01100000; //ID=1 Increment, S=0
  triggerEnable();      //Send Enable Signal
  waitMicros(55);

  PORTD= (PORTD&B00001111); //Display On Of control
  triggerEnable();      //Send Enable Signal
  PORTD= (PORTD&B00001111)|B11100000; //Display =1
                                      //Cursor =1
                                      //Blink cursor=0
  triggerEnable();      //Send Enable Signal
  waitMicros(55);
}

void waitMicros(int time){
  long start_time=micros();
  while(micros()-start_time>=time);
}

void waitMillis(int time){
  long start_time=millis();
  while(millis()-start_time>=time);
}
```

Figure 4: Some subroutines and their implementations

5 Part-4

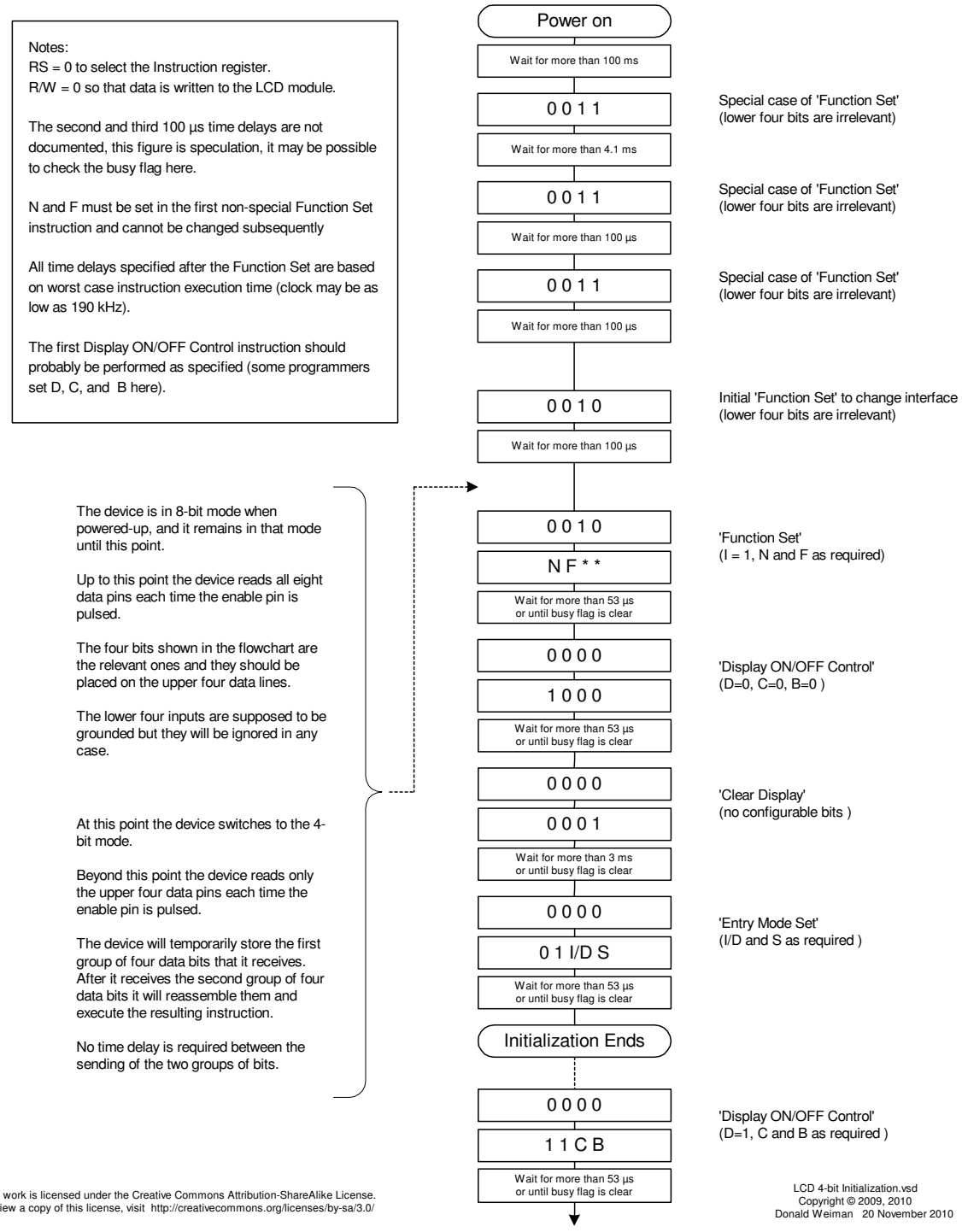
In this part, you are going to create a countdown timer that counts down in seconds, minutes, and hours using LCD and timer interrupt on the design which is given in Part-3. LCD display should show the countdown timer as hh:mm:ss. You can specify the countdown number as a variable. However, your program should work with different countdown numbers. A sound from the buzzer should ring at end of the timer. You can utilize the codes you have implemented in previous parts. Timer interrupt should be used to keep track of how much time has passed and only LCD instructions should be in the void loop.

Figure 5: Example of countdown timer

6 Submission

- You should upload your experiment codes and report on Ninova, and please, do not send your experiment files via e-mail.
- You must upload each part's code seperately to the ninova.
- Your reports must be written with Latex format. Latex report template is available on Ninova. You can use any Latex editor whichever you want. If you upload your report without Latex file, you directly get 0 as your report grade. You should upload both .pdf and .tex files of your report.
- You must add the calculation of the period of the timer interrupt for each part that needs timer interrupt to the report.
- Finally, please do not forget that late submissions are not accepted.

Character Mode Liquid Crystal Display Module
Initialization by Instruction - 4-bit data interface



LCD 4-bit Initialization.vsd

Donald Weiman 20 November 2010