Timers

Alexandru Malgras

20/11/2021

Introduction

The idea of the assignment is that on a button press the light of the LED is toggled using interrupts. When the LED is turned on it will start to blink every few of seconds using timer interrupts. A second button is pressed to change the interval that the LED is blinking. There are four different intervals. When the LED is turned off the blink will stop until the LED is turned on again.

Research

Community research

At the start of the development, I started informing myself how the timer interrupts work. Browsing the internet, I found a website with documentation on the subject. Reading throughout the documentation I found a lot of similarities with the interrupts and had the occasion to refresh my knowledge on how they work. In the documentation there were also different examples on situations the timers can be used. The examples were accompanied with code, and it made it easier for me to understand the steps of integrating a timer in my application.

The website I used for timer interrupts documentation:
https://www.robotshop.com/community/forum/t/arduino-101-timers-and-interrupts/13072

To make my code efficient I also used the datasheet of the Arduino Uno which can be found at:
https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

Non-functional test

A non-functional test was performed in order to see if the final version of the application meets the requirements listed in the assignment.

- The application is operable using a manual control panel. There is a button and an LED located on the breadboard.
- A button is used to control when the timer is used.
- An LED is used to interact with the timer. The LED blinks within a specific time.
- The calculation of the one second between blinks is done using the timer.
- A timer frequency is set for accurate behavior.
- The application is always responsive to user actions and button presses.
- The datasheet was used to make the code free of memory leaks.

Testing the application

Step 1

The application was uploaded to the Arduino.

Step 2

The button in charge of toggling the LED was pressed to check if the LED turns on.

Step 3

The LED was observed to see if the timer behaved as expected.

Step 4

The button in charge of changing the blinking interval was pressed multiple times to see if the blinking intervals are working as expected.

Step 5

The button was pressed again to turn the LED off.

Step 6

The LED was observed to see if the process ends successfully.

The testing process was recorded and it can be found pressing here.

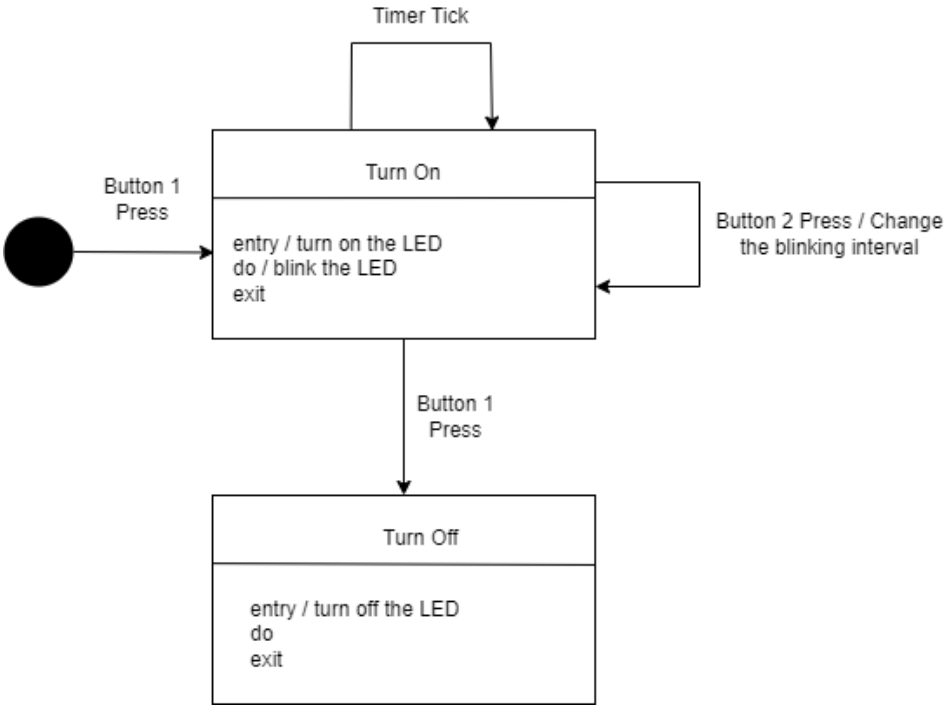
Toggling the LED

In order to toggle the LED, I used the system interrupts.

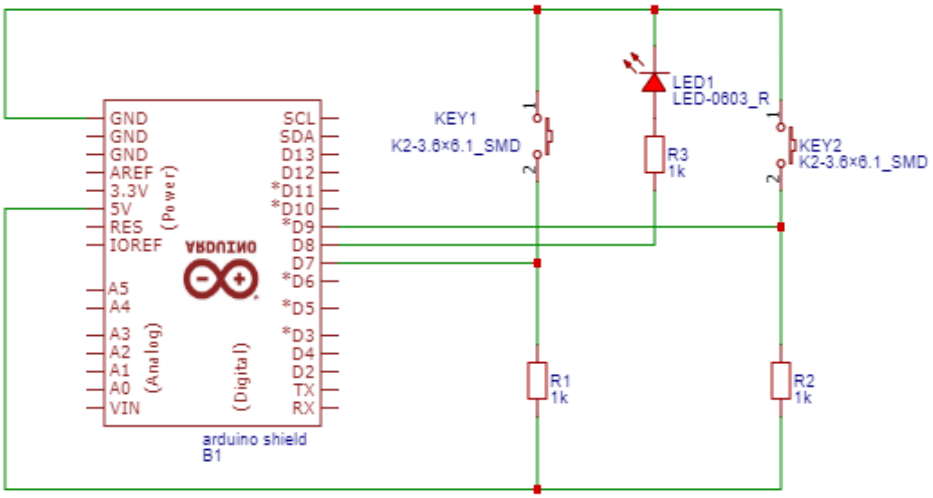
Blinking the LED

In order to blink the LED, I changed the pre-scaler of the timer that I am using. From what I understood the timers have pre-scalers. The pre-scalers represent the number of cycles until an action is performed. In order to change the blinking speed of the LED I changed between 4 pre-scalers, 1024, 256, 64, 8. The higher the pre-scaler number is the longer it will take for the LED to perform a blink.

State Diagram



The circuit



Components:

1x Arduino UNO

2x Push button switch

1x White LED

3x 1K Ω resistors

The code

```
#include <Arduino.h>


bool isON = false;
int prescaler = 1024;


ISR(PCINT2_vect)
{
  //Toggle the LED on button release
  if (!(PIND & (1 << PIND7)))
  {
    isON = !isON;
  }
  PORTB &= ~(1 << PB0);
}


ISR(PCINT0_vect)
{
  //Set prescaler to 256 from 1024
  if(!(PINB & (1 << PINB1)) && prescaler == 1024)
  {
      TCCR1B &= ~(1 << WGM13) & ~(1 << WGM12) & ~(1 << WGM11) & ~(1 << WGM10);


      TCCR1B |= (1 << CS12);
      TCCR1B &= ~(1 << CS11) & ~(1 << CS10);
```

```
    prescaler = 256;
}
//Set prescaler to 64 from 256
else if(!(PINB & (1 << PINB1)) && prescaler == 256)
{
    TCCR1B &= ~(1 << WGM13) & ~(1 << WGM12) & ~(1 << WGM11) & ~(1 << WGM10);


    TCCR1B |= (1 << CS11) | (1 << CS10);
    TCCR1B &= ~(1 << CS12);


    prescaler = 64;
}
//Set prescaler to 8 from 64
else if(!(PINB & (1 << PINB1)) && prescaler == 64)
{
    TCCR1B &= ~(1 << WGM13) & ~(1 << WGM12) & ~(1 << WGM11) & ~(1 << WGM10);


    TCCR1B |= (1 << CS11);
    TCCR1B &= ~(1 << CS12) & ~(1 << CS10);


    prescaler = 8;
}
else if(!(PINB & (1 << PINB1)) && prescaler == 8)
{
    TCCR1B &= ~(1 << WGM13) & ~(1 << WGM12) & ~(1 << WGM11) & ~(1 << WGM10);
```

```
      TCCR1B |= (1 << CS12) | (1 << CS10);
      TCCR1B &= ~(1 << CS11);

      prescaler = 1024;
  }
}

ISR(TIMER1_COMPA_vect)
{
  if(isON)
  {
    PORTB ^= (1 << PB0);
  }
}

void setup()
{
  // put your setup code here, to run once:

  //Initiate timer1

  // disable all interrupts
  noInterrupts();

  TCCR1A = 0;
```

```
TCCR1B = 0;


//Pin mode
DDRD |= (1 << DDD7);
DDRB &= ~(1 << DDB0);
DDRB |= (1 << DDB1);

//Set pin 8 to off
PORTB &= ~(1 << PB0);


//Enable the interrupts for port D
PCICR |= (1 << PCIE2);
//Enable the interrupts for port B
PCICR |= (1 << PCIE0);


//Enable interrupts for pin 7
PCMSK2 |= (1 << PCINT23);
//Enable interrupts for pin 9
PCMSK0 |= (1 << PCINT1);


//Set timer frequency
OCR1A = 62500;


//Set timer mode
TCCR1B &= ~(1 << WGM13) & ~(1 << WGM12) & ~(1 << WGM11) & ~(1 << WGM10);
```

```
  //Set pre-scaler to 1024
  TCCR1B |= (1 << CS12) | (1 << CS10);
  TCCR1B &= ~(1 << CS11);

  //Enable timer1 compare interrupt
  TIMSK1 |= (1 << OCIE1A);

  //Enable all interrupts
  interrupts();
}

void loop()
{
  // put your main code here, to run repeatedly:
}
```