DIGITAL ELECTRONICS 2 LAB ASSIGNMENT 4

Name: Demirkan Korbey Baglamac

1)

Module	Number of bits	1	8	32	64	128	256	1024
Timer/Counter0	8	16u	128u	--	1ms	--	4ms	16ms
Timer/Counter1	16	4ms	33ms	--	262ms	--	1s	4s
Timer/Counter2	8	16u	128u	512u	1ms	2ms	4ms	16ms

2)

Module	Operation	I/O register(s)	Bit(s)
Timer/Counter0	Prescaler	TCCR0B	CS02, CS01, CS00 (000: stopped, 001: 1, 010: 8, 011: 64, 100: 256, 101: 1024)
	8-bit data value	TCNT0	TCNT0[7:0]
	Overflow interrupt enable	TIMSK0	TOIE0 (1: enable, 0:disable)
Timer/Counter1	Prescaler	TCCR1B	CS12, CS11, CS10 (000: stopped, 001: 1, 010: 8, 011: 64, 100: 256, 101: 1024)
	16-bit data value	TCNT1H, TCNT1L	TCNT1[15:0]
	Overflow interrupt enable	TIMSK1	TOIE1 (1: enable, 0: disable)
Timer/Counter2	Prescaler	TCCR2B	CS22, CS21, CS20 (000: stopped, 001: 1, 010: 8, 011: 32, 100: 64, 101:128, 110: 256, 111: 1024)
	8-bit data value	TCNT2	TCNT2[7:0]
	Overflow interrupt enable	TIMSK2	TOIE2 (1: enable, 0:disable)

Time.h file:

```
#ifndef TIMER_H
#define TIMER_H

/* Includes ------------------------------------------------------------*/
#include <avr/io.h>
/* Defines -------------------------------------------------------------*/
/**
 * @brief Defines prescaler CPU frequency values for Timer/Counter0.
 * @note  F_CPU = 16 MHz
 */
#define TIM0_stop()             TCCR0B &= ~((1<<CS02) | (1<<CS01) | (1<<CS00));
#define TIM1_overflow_16u()     TCCR0B &= ~((1<<CS02) | (1<<CS01)); TCCR0B |= (1<<CS00);
#define TIM1_overflow_128u()    TCCR0B &= ~((1<<CS02) | (1<<CS00)); TCCR0B |= (1<<CS01);
#define TIM1_overflow_1ms()     TCCR0B &= ~(1<<CS02); TCCR0B |= (1<<CS01) | (1<<CS00);
#define TIM1_overflow_4ms()     TCCR0B &= ~(1<<CS01) | (1<<CS00); TCCR0B |= (1<<CS02);
#define TIM1_overflow_16ms()    TCCR0B &= ~(1<<CS01); TCCR0B |= (1<<CS02) | (1<<CS00);

/**
 * @brief Defines interrupt enable/disable modes for Timer/Counter0.
 */
#define TIM0_overflow_interrupt_enable()    TIMSK0 |= (1<<TOIE0);
#define TIM0_overflow_interrupt_disable()   TIMSK0 &= ~(1<<TOIE0);


/**
 * @brief Defines prescaler CPU frequency values for Timer/Counter1.
 * @note  F_CPU = 16 MHz
 */
#define TIM1_stop()             TCCR1B &= ~((1<<CS12) | (1<<CS11) | (1<<CS10));
#define TIM1_overflow_4ms()     TCCR1B &= ~((1<<CS12) | (1<<CS11)); TCCR1B |= (1<<CS10);
#define TIM1_overflow_33ms()    TCCR1B &= ~((1<<CS12) | (1<<CS10)); TCCR1B |= (1<<CS11);
#define TIM1_overflow_262ms()   TCCR1B &= ~(1<<CS12); TCCR1B |= (1<<CS11) | (1<<CS10);
#define TIM1_overflow_1s()      TCCR1B &= ~((1<<CS11) | (1<<CS10)); TCCR1B |= (1<<CS12);
#define TIM1_overflow_4s()      TCCR1B &= ~(1<<CS11); TCCR1B |= (1<<CS12) | (1<<CS10);

/**
 * @brief Defines interrupt enable/disable modes for Timer/Counter1.
 */
#define TIM1_overflow_interrupt_enable()    TIMSK1 |= (1<<TOIE1);
#define TIM1_overflow_interrupt_disable()   TIMSK1 &= ~(1<<TOIE1);


/**
 * @brief Defines prescaler CPU frequency values for Timer/Counter2.
 * @note  F_CPU = 16 MHz
 */
#define TIM2_stop()             TCCR2B &= ~((1<<CS22) | (1<<CS21) | (1<<CS20));
#define TIM2_overflow_16u()     TCCR2B &= ~((1<<CS22) | (1<<CS21)); TCCR2B |= (1<<CS20);
#define TIM2_overflow_128u()    TCCR2B &= ~((1<<CS22) | (1<<CS20)); TCCR2B |= (1<<CS21);
#define TIM2_overflow_512u()    TCCR2B &= ~(1<<CS22); TCCR2B |= (1<<CS21) | (1<<CS20);
#define TIM2_overflow_1ms()     TCCR2B &= ~((1<<CS21) | (1<<CS20)); TCCR2B |= (1<<CS22);
#define TIM2_overflow_2ms()     TCCR2B &= ~(1<<CS21); TCCR2B |= (1<<CS22) | (1<<CS20);
#define TIM2_overflow_4ms()     TCCR2B &= ~(1<<CS20); TCCR2B |= (1<<CS21) | (1<<CS22);
#define TIM2_overflow_16ms()    TCCR2B |= (1<<CS22) | (1<<CS21) | (1<<CS20);

/**
 * @brief Defines interrupt enable/disable modes for Timer/Counter2.
 */
#define TIM2_overflow_interrupt_enable()    TIMSK2 |= (1<<TOIE2);
#define TIM2_overflow_interrupt_disable()   TIMSK2 &= ~(1<<TOIE2);


#endif
```

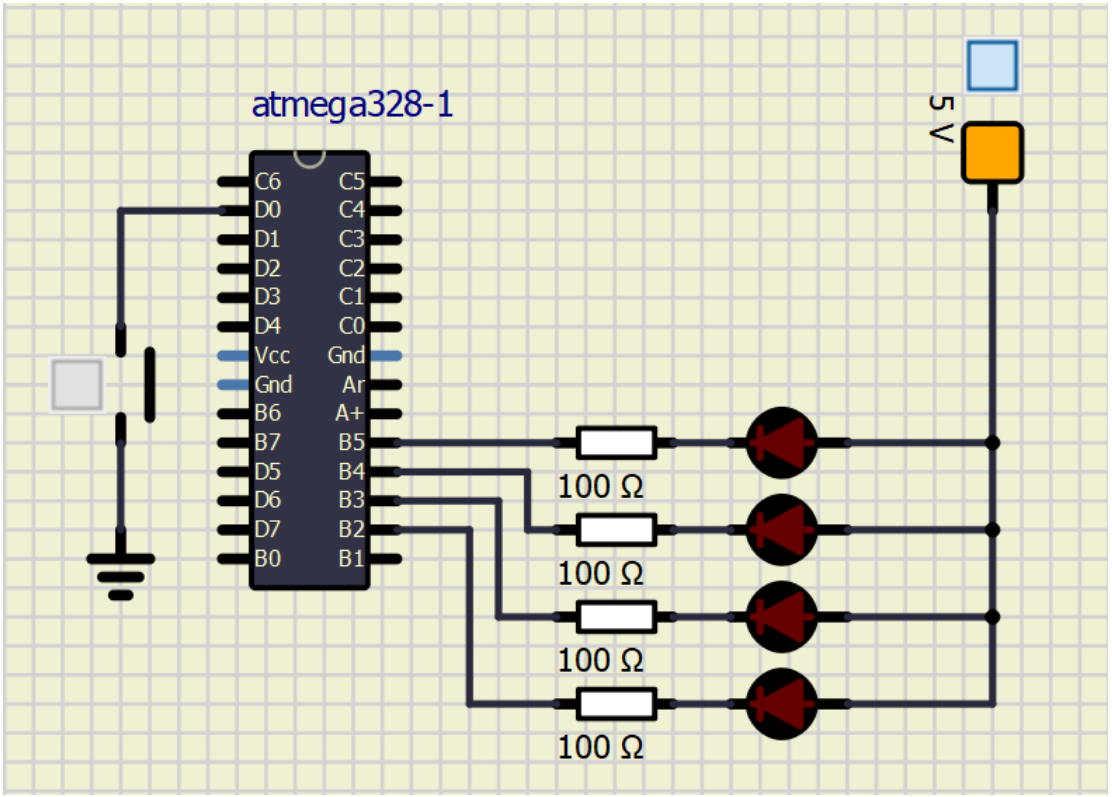
Program address	Source	Vector name	Description
0x0000	RESET	--	Reset of the system
0x0002	INT0	INT0_vect	External interrupt request number 0
0x0004	INT1	INT1_ vect	External interrupt request number 1
0x0006	PCINT0	PCINT0_vect	Pin change interrupt request 0
0x0008	PCINT1	PCINT1_vect	Pin change interrupt request 1
0x000A	PCINT2	PCINT2_vect	Pin change interrupt request 2
0x000C	WDT	WDT_vect	Watchdog Time-out Interrupt
0x0012	TIMER2_OVF	TIMER2_OVF_vect	Overflow of Timer/Counter2 value
0x0018	TIMER1_COMPB	TIMER1_COMPB_vect	Compare match between Timer/Counter1 value and channel B compare value
0x001A	TIMER1_OVF	TIMER1_OVF_vect	Overflow of Timer/Counter1 value
0x0020	TIMER0_OVF	TIMER0_OVF_vect	Overflow of Timer/Counter0 value
0x0024	USART_RX	USART_RX_vect	USART, Tx Complete
0x002A	ADC	ADC_ vect	ADC Conversion Complete
0x0030	TWI	TWI_vect	2-wire Serial Interface

Main.c file:

```c
/* Defines -----------------------------------------------------------*/
#define LED_D1  PB5
#define LED_D2  PB4
#define LED_D3  PB3
#define LED_D4  PB2
#define BTN           PD0

/* Includes ----------------------------------------------------------*/
#include <avr/io.h>         // AVR device-specific IO definitions
#include <avr/interrupt.h>  // Interrupts standard C library for AVR-GCC
#include "gpio.h"           // GPIO library for AVR-GCC
#include "timer.h"          // Timer library for AVR-GCC

/* Function definitions ----------------------------------------------*/
/**
 * Main function where the program execution begins. Toggle three LEDs
 * on Multi-function shield with internal 8- and 16-bit timer modules.
 */

// Global Variables for leds
uint8_t leds[4] = {LED_D1, LED_D2, LED_D3, LED_D4};
int led_count = 0;
int back = 0;

int main(void)
{

    /* Configuration of three LEDs */
    GPIO_config_output(&DDRB, leds[0]);
    GPIO_write_low(&PORTB, leds[0]);

    GPIO_config_output(&DDRB, leds[1]);
    GPIO_write_high(&PORTB, leds[1]);
```

```
       GPIO_config_output(&DDRB, leds[2]);
    GPIO_write_high(&PORTB, leds[2]);

       GPIO_config_output(&DDRB, leds[3]);
       GPIO_write_high(&PORTB, leds[3]);


       /*Setting up the push button*/
       GPIO_config_input_pullup(&DDRD, BTN);

    // Enables interrupts by setting the global interrupt mask
    sei();

    // Infinite loop
    while (1)
    {
              //Determining if the button is pressed or not and adjusting the speed
according to that
        if(GPIO_read(&PIND, BTN))
              {
                     /* Configuration of 16-bit Timer/Counter1
                      * Set prescaler and enable overflow interrupt */
                      TIM1_overflow_1s();
                   TIM1_overflow_interrupt_enable();
              }
              else
              {
                      /* Configuration of 16-bit Timer/Counter1
                      * Set prescaler and enable overflow interrupt */
                      TIM1_overflow_262ms();
                   TIM1_overflow_interrupt_enable();
              }
    }

    // Will never reach this
    return 0;
}

/* Interrupt service routines ----------------------------------------*/
ISR(TIMER1_OVF_vect)
{
       //Toggling off the previous led
       GPIO_toggle(&PORTB, leds[led_count]);

       //Changing the direction
       if(led_count == 3) {
              back = 1;
       }
       if(led_count == 0) {
              back = 0;
       }

       //Adjusting the led_count value
       if(back == 0){
              led_count = led_count + 1;
       }
       else if(back == 1){
              led_count = led_count - 1;
       }

       //Toggling on the new led
       GPIO_toggle(&PORTB, leds[led_count]);
}
```

Screenshot of SimulIDE circuit:



In your words, describe the difference between a common C function and interrupt service routine:

Interrupt service routine is a speficitation of the microcontroller hardware but C function is just a executable code.

3)

Module	Description	MCU pin	Arduino pin
Timer/Counter0	OC0A	PD6	~6
	OC0B	PD5	~5
Timer/Counter1	OC1A	PB1	~9
	OC1B	PB2	10
Timer/Counter2	OC2A	PB3	~11
	OC2B	PD3	~3

Clear timer on Compare Mode: This mode gives us, more control on the Compare Match output frequency. Also it simplifies external event counting.

Fast PWM Mode: This mode provides high frequency PWM waveforms. This high frequency can be used in rectification, power regulation and more.