Lab 4: Interrupts, timers

David Garcia Torre

- Table with overflow times.

Module	Number of bits	1	8	32	64	128	256	1024
Timer/Counter 0	8	16us	128us	--	1ms	--	4ms	16ms
Timer/Counter 1	16	4ms	33ms	--	262ms	--	1s	4s
Timer/Counter 2	8	16us	128us	512us	1ms	2ms	4ms	16ms

Listing of library header file `timer.h`

```
#ifndef TIMER_H
#define TIMER_H

/************************************************************************
 *
 * Timer library for AVR-GCC.
 * ATmega328P (Arduino Uno), 16 MHz, AVR 8-bit Toolchain 3.6.2
 *
 * Copyright (c) 2019-2020 Tomas Fryza
 * Dept. of Radio Electronics, Brno University of Technology, Czechia
 * This work is licensed under the terms of the MIT license.
 *
 ************************************************************************/

/**
 * @file  timer.h
 * @brief Timer library for AVR-GCC.
 *
 * @details
 * The library contains macros for controlling the timer modules.
 *
 * @note
 * Based on Microchip Atmel ATmega328P manual and no source file is
 * needed for the library.
 *
```

```
 * @copyright (c) 2019-2020 Tomas Fryza
 * Dept. of Radio Electronics, Brno University of Technology, Czechia
 * This work is licensed under the terms of the MIT license.
 */

/* Includes ------------------------------------------------------------*/
#include <avr/io.h>

/**
 * @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 TIM0_overflow_16us()    TCCR0B &= ~((1<<CS02) | (1<<CS01)); TCCR0B |=
(1<<CS00);
#define TIM0_overflow_128us()   TCCR0B &= ~((1<<CS02) | (1<<CS00)); TCCR0B |=
(1<<CS01);
#define TIM0_overflow_1ms()     TCCR0B &= ~(1<<CS02); TCCR0B |= (1<<CS01) |
(1<<CS00);
#define TIM0_overflow_4ms()     TCCR0B &= ~((1<<CS02) | (1<<CS00)); TCCR0B |=
(1<<CS02);
#define TIM0_overflow_16ms()    TCCR0B &= ~(1<<CS02); TCCR0B |= (1<<CS02) |
(1<<CS00);


/**
 * @brief Defines interrupt enable/disable modes for Timer/Counter1.
 */
#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/Counter1.
 * @note  F_CPU = 16 MHz
 */
#define TIM2_stop()             TCCR2B &= ~((1<<CS22) | (1<<CS21) | (1<<CS20));
#define TIM2_overflow_16us()    TCCR2B &= ~((1<<CS22) | (1<<CS21)); TCCR2B |=
(1<<CS20);
```

```
#define TIM2_overflow_128us()   TCCR2B &= ~((1<<CS22) | (1<<CS20)); TCCR2B |=
(1<<CS21);
#define TIM2_overflow_512us()   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<<CS22) |
(1<<CS21);
#define TIM2_overflow_16ms()    TCCR2B |= (1<<CS22) | (1<<CS21) | (1<<CS20);

/**
 * @brief Defines interrupt enable/disable modes for Timer/Counter1.
 */
#define TIM2_overflow_interrupt_enable()    TIMSK2 |= (1<<TOIE2);
#define TIM2_overflow_interrupt_disable()   TIMSK2 &= ~(1<<TOIE2);

#endif
```

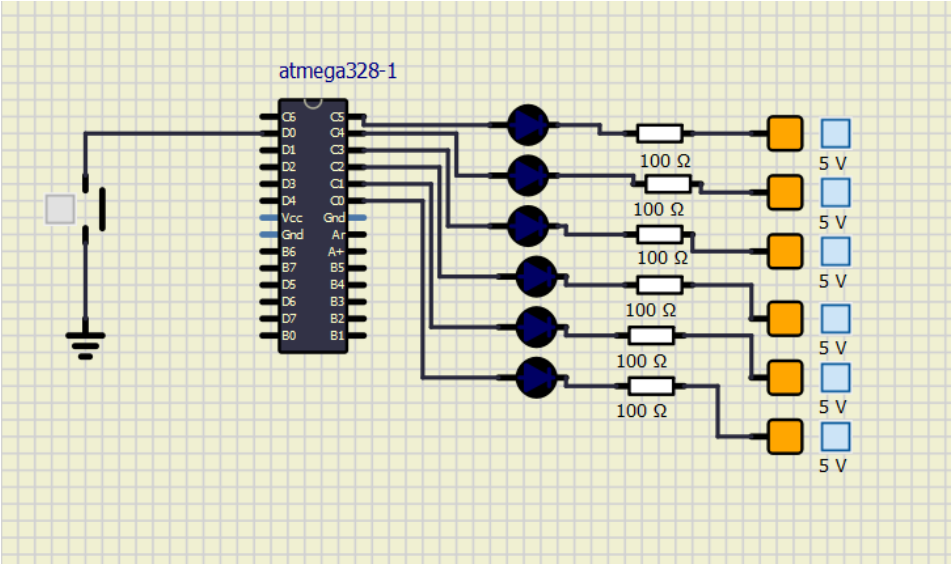
Program address	Source	Vector name	Description
0x0000	RESET	--	Reset of the system
0x0002	INT0	`INT0_vect`	External interrupt request number 0
0x0004	INT1	`INT1_vect`	External interrupt request number 1
0x0006	PCINT0	PCINT0_vect	Pin change interrupt request 0
0x0008	PCINT1	PCINT1_vect	Pin change interrupt 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	USARTRX complete
0x002A	ADC	ADC_vect	ADC conversion complete

0x0030	TWI	TWI_vect	2 wire serial interface

- Listing of the Knight Rider application `main.c`,

```
/* Defines -----------------------------------------------------------*/
#define LED_RED1    PC0     // AVR pin where red LED 1 is connected
#define LED_RED2    PC1     // AVR pin where red LED 2 is connected
#define LED_RED3    PC2     // AVR pin where red LED 3 is connected
#define LED_RED4    PC3     // AVR pin where red LED 4 is connected
#define LED_RED5    PC4     // AVR pin where red LED 5 is connected
#define LED_RED6    PC5     // AVR pin where red LED 6 is connected
#define BUTTON          PD0     // AVR pin where the button is connected


/* 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


/* Functions ---------------------------------------------------------
 * Main function where the program execution begins.
 */

int leds[] ={LED_RED1,LED_RED2,LED_RED3,LED_RED4,LED_RED5,LED_RED6};
int a=0,b=0;

int main(void){


       /* Configuration of  LEDs */
       GPIO_config_output(&DDRC, LED_RED1);
       GPIO_write_low(&DDRC, LED_RED1);

       GPIO_config_output(&DDRC, LED_RED2);
       GPIO_write_low(&DDRC, LED_RED2);

       GPIO_config_output(&DDRC, LED_RED3);
       GPIO_write_low(&DDRC, LED_RED3);

       GPIO_config_output(&DDRC, LED_RED4);
       GPIO_write_low(&DDRC, LED_RED4);

       GPIO_config_output(&DDRC, LED_RED5);
       GPIO_write_low(&DDRC, LED_RED5);

       GPIO_config_output(&DDRC, LED_RED6);
       GPIO_write_low(&DDRC, LED_RED6);

       /* Configuration of 16-bit Timer/Counter1
    * Set prescaler and enable overflow interrupt */
    TIM1_overflow_262ms();
    TIM1_overflow_interrupt_enable();


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

```
    // Infinite loop
    for (;;){

          if(bit_is_clear(PIND,BUTTON)){

                TIM1_overflow_262ms();

                }else{

                TIM1_overflow_1s();
          }

    }

    // Will never reach this
    return 0;
    }

/* Interrupt service routines ----------------------------------------*/
/**
 * ISR starts when Timer/Counter1 overflows. Toggle LED D2 on
 * Multi-function shield. */

ISR(TIMER1_OVF_vect)
{
       uint8_t leds[] ={LED_RED1,LED_RED2,LED_RED3,LED_RED4,LED_RED5,LED_RED6};
       uint8_t a = 0;
       uint8_t b = 0;


       if(b == 5){

             a = 1;
             GPIO_write_high(&DDRC,leds[5]);

       }else if(b == 0){

             a = 0;
             GPIO_write_high(&DDRC,leds[5]);

       }

       if(a == 0){

             b++;

       }else{

             b--;

       }

       GPIO_write_low(&DDRC,leds[b]);

}
```



The difference between a normal function in c and an interruption, is that the normal function follows the order established in the code, when the call to the function arrives, then the function enters, in case of the interruption, is that when it arrives, it skips directly to the process regardless of the order.

- Table with PWM channels of ATmega328P

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

In Clear Timer on Compare or CTC mode (WGM22:0 = 2), the OCR2A Register is used to manipulate the counter resolution. In CTC mode the counter is cleared to zero when the counter value (TCNT2) matches the OCR2A. The OCR2A defines the top value for the counter, hence also its resolution. This mode allows greater control of the compare match output frequency. It also simplifies the operation of counting external events.