

# Lab 4: Interrupts, timers

- Table with overflow times.

Module	Number of bits	1	8	32	64	128	256	1024
Timer/Counter 0	8	16u	128u	--	1.024m	--	4.096m	16.384m
Timer/Counter 1	16	4.096m	32.76m	--	262.14m	--	1.048	4.19
Timer/Counter 2	8	16u	128u	131.07m	262.14m	524.2m	1.048	4.19

- 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_4ms()  TCCR0B &= ~(1<<CS02) | (1<<CS01); TCCR0B |=
(1<<CS00);
#define TIM0_overflow_33ms() TCCR0B &= ~(1<<CS02) | (1<<CS00); TCCR0B |=
(1<<CS01);
#define TIM0_overflow_262ms() TCCR0B &= ~(1<<CS02); TCCR0B |= (1<<CS01) |
(1<<CS00);
#define TIM0_overflow_1s()   TCCR0B &= ~(1<<CS02) | (1<<CS00); TCCR0B |=
(1<<CS02);
#define TIM0_overflow_4s()   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);

#endif

```

- Listing of library header file timer.h,

<b>Program address</b>	<b>Source</b>	<b>Vector name</b>	<b>Description</b>
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 final application main.c

Blink only LED 2

```

/*****
 *
 * Control LEDs using functions from GPIO and Timer libraries. Do not
 * use delay library any more.
 * ATmega328P (Arduino Uno), 16 MHz, AVR 8-bit Toolchain 3.6.2
 *
 * Copyright (c) 2018-2020 Tomas Fryza
 * Dept. of Radio Electronics, Brno University of Technology, Czechia
 * This work is licensed under the terms of the MIT license.
 *
 *****/

/* Defines -----*/
#define LED_D1 PB5
#define LED_D2 PB4
#define LED_D3 PB3

/* 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.
 */
int main(void)
{
    /* Configuration of three LEDs */
    GPIO_config_output(&DDRB, LED_D1);
    GPIO_write_low(&PORTB, LED_D1);

    GPIO_config_output(&DDRB, LED_D2);
    GPIO_write_low(&PORTB, LED_D2);

    GPIO_config_output(&DDRB, LED_D3);
    GPIO_write_low(&PORTB, LED_D3);

    /* Configuration of 8-bit Timer/Counter0 */

```

```

TIM0_overflow_1s();
TIM0_overflow_interrupt_enable();

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

/* Configuration of 8-bit Timer/Counter2 */
TIM2_overflow_33ms();
TIM2_overflow_interrupt_enable();

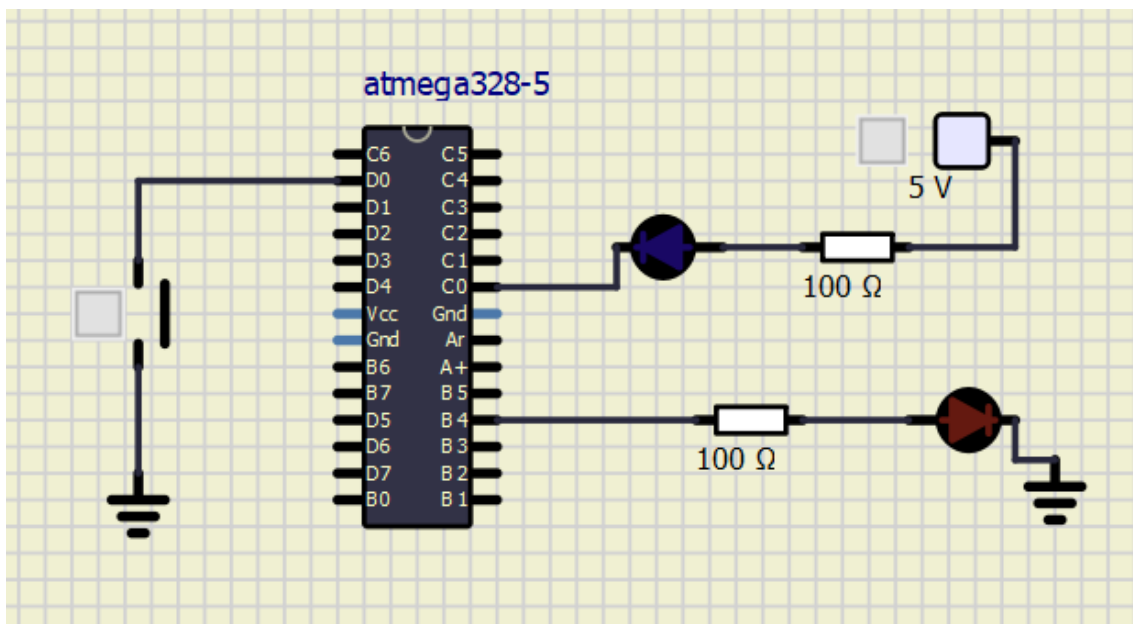
// Enables interrupts by setting the global interrupt mask
sei();

// Infinite loop
while (1)
{
    /* Empty loop. All subsequent operations are performed exclusively
     * inside interrupt service routines ISRs */
}

// Will never reach this
return 0;
}

/* Interrupt service routines -----*/
/**
 * ISR starts when Timer/Counter1 overflows. Toggle LED D2 on
 * Multi-function shield. */
ISR(TIM1_OVF_vect)
{
    GPIO_toggle(&PORTB, LED_D2);
}

```



## Knight Rider style

```
/*
 * Control LEDs Knight Rider
 * ATmega328P (Arduino Uno), 16 MHz, AVR 8-bit Toolchain 3.6.2
 *
 * Copyright (c) 2018-2020 Tomas Fryza
 * Dept. of Radio Electronics, Brno University of Technology, Czechia
 * This work is licensed under the terms of the MIT license.
 */

/* 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 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

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

/* Configuration of 8-bit Timer/Counter0 */
TIM0_overflow_1s();
TIM0_overflow_interrupt_enable();

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

/* Configuration of 8-bit Timer/Counter2 */

TIM2_overflow_33ms();
TIM2_overflow_interrupt_enable();

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

// Infinite loop
while (1)
{
    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 r = 0;
    uint8_t i = 0;

    uint8_t leds[] = {LED_RED1,LED_RED2,LED_RED3,LED_RED4,LED_RED5};

    if(i == 4){

        r = 1;
        GPIO_write_high(&DDRC,leds[4]);

    }else if(i == 0){

        r = 0;
        GPIO_write_high(&DDRC,leds[4]);

    }
}

```

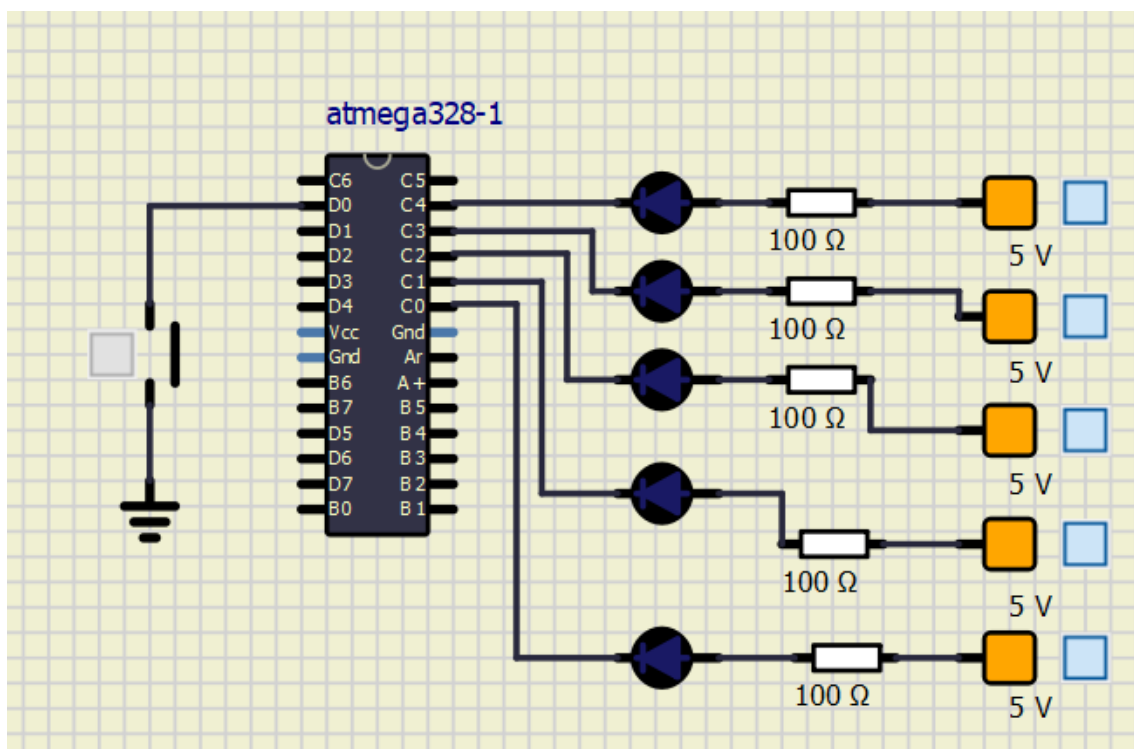
```

}

if(r == 0){
    i++;
}else{
    i--;
}

GPIO_write_low(&DDRC, leds[i]);
}

```



The difference between a common C function and an interrupt is that when an interrupt happen, the microprocessor hast to attend it immediately, in the other hand, a common function could be left without finishing because there is an interrupt for example.

- 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.

The fast Pulse Width Modulation or fast PWM mode ( $WGM22:0 = 3$  or  $7$ ) provides a high frequency PWM waveform generation option. The fast PWM differs from the other PWM option by its single-slope operation. The