Kamil Káčer 211777

		Milisec.						
Module	Number of bits	1	8	32	64	128	256	1024
Timer/Counter0	8	0.016	0.128	--	1.024	--	4.096	16.384
Timer/Counter1	16	4.096	32.768	--	262.144	--	1048.576	4194.304
Timer/Counter2	8	0.016	0.128	0.512	1.024	2.048	4.096	16.384

Operation	I/O registers	Bit(s)
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
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)
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

Program adress	Source	Vector name	Description
0x0000	RESET	..	Reset of the system
0x0002	INT0	INT0_vect	External interrupt request 0
0x0004	INT1	INT1_vect	External interrupt request 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	Watching Time-out interrupt
0x0012	TIMER2_OVF	TIMER2_OVF_vect	Timer/Counter2 Overflow
0x0018	TIMER1_COMPB	TIMER1_COMPB_vect	Timer/Counter1 compare match B
0x001A	TIMER1_OVF	TIMER1_OVF_vect	Timer/Counter1 Overflow
0x0020	TIMER0_OVF	TIMER0_OVF_vect	Timer/Counter0 Overflow
0x0024	USART_RX	USART_RX_vect	USART Rx complete
0x002A	ADC	ADC_vect	ADC conversion complete
0x0030	TWI	TWI_vect	2-wire Serial interface

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

Normal mode:

Counting direction is always up overruns when passes the last bit

CTC mode:

Allows greater controf of the compare match output frequency. Counting external events.

The counter value increases until compare match occurs between TCNT0 and OCR0A then counter is cleared (interrupt can be enabled)

PWM mode:

In PWM mode counter is incremented until the counter value matches the top value. Counter is cleared in the next cycle.

Two modes- non-inverting compare output mode and inverting compare output mode.

Phase correct PWM mode

Counts until the counter value reatches top then it changes the counter direction

Interrupt flag can be used to generate interrupt each time the counter reaches the bottom.

Funkcia musí byt explicitne zavolaná kdežto ISR je zavolaná na základe nejakého externej udalosti.

Keď je zavolaný napríklad interrupt tak sa zapíše do pamäti čo sa vykonávalo a prejde sa rovno na tú danú interrupciu.

Main c.

```c
/***********************************************************************
 *
 * 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_D2);
       GPIO_write_low(&PORTB, LED_D2);

       GPIO_config_output(&DDRB, LED_D1);
       GPIO_write_low(&PORTB, LED_D1);

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



    /* Configuration of 8-bit Timer/Counter0 */
       TIM0_overflow_16ms();
       TIM0_overflow_interrupt_enable();

       /* Configuration of 16-bit Timer/Counter1
       * Set prescaler and enable overflow interrupt */
       TIM1_overflow_1s();
       TIM1_overflow_interrupt_enable();

       /* Configuration of 8-bit Timer/Counter2 */
       TIM2_overflow_128us();
       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
```

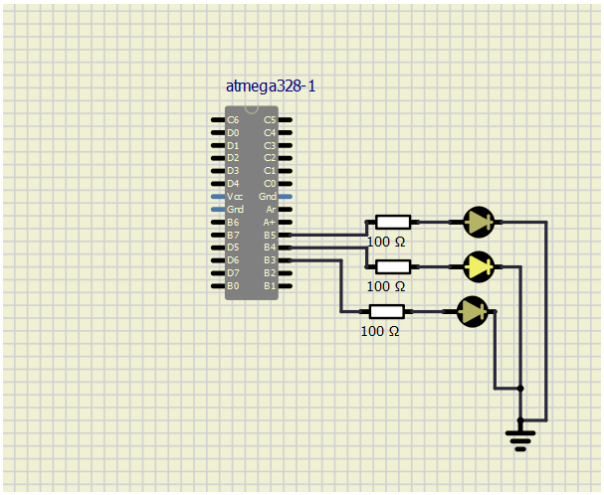
```
         * inside interrupt service routines ISRs */
    }

    // Will never reach this
    return 0;
}

/* Interrupt service routines ---------------------------------------*/
/**
 * ISR starts when Timer/Counter1 overflows. Toggle LEDs on
 * Multi-function shield. */
       ISR(TIMER0_OVF_vect)
       {
       GPIO_toggle(&PORTB, LED_D1);
       }

       ISR(TIMER1_OVF_vect)
       {
       GPIO_toggle(&PORTB, LED_D2);
       }

       ISR(TIMER2_OVF_vect)
       {
       GPIO_toggle(&PORTB, LED_D3);
       }
```

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.
 */
```

```c
/* 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 TIM0_overflow_16us()      TCCR0B &= ~((1<<CS2) | (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<<CS01) | (1<<CS00)); TCCR0B |=
(1<<CS02);
#define TIM0_overflow_16ms()    TCCR0B &= ~(1<<CS01); TCCR0B |= (1<<CS02) | (1<<CS00);
/**

 * @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 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_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_1024us()   TCCR2B &= ~((1<<CS21) | (1<<CS20)); TCCR2B |=
(1<<CS22);
#define TIM2_overflow_2048us()   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<<CS20) | (1<<C21));

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

#endif
```



Cannot see two other leds blinking because prescalar is still too small for human eye to recognize its change.