Čtvrtá laboratorní úloha z DE2

xpastu02

Vysoké učení technické v Brně

Fakulta elektroniky
a komunikačních technologií

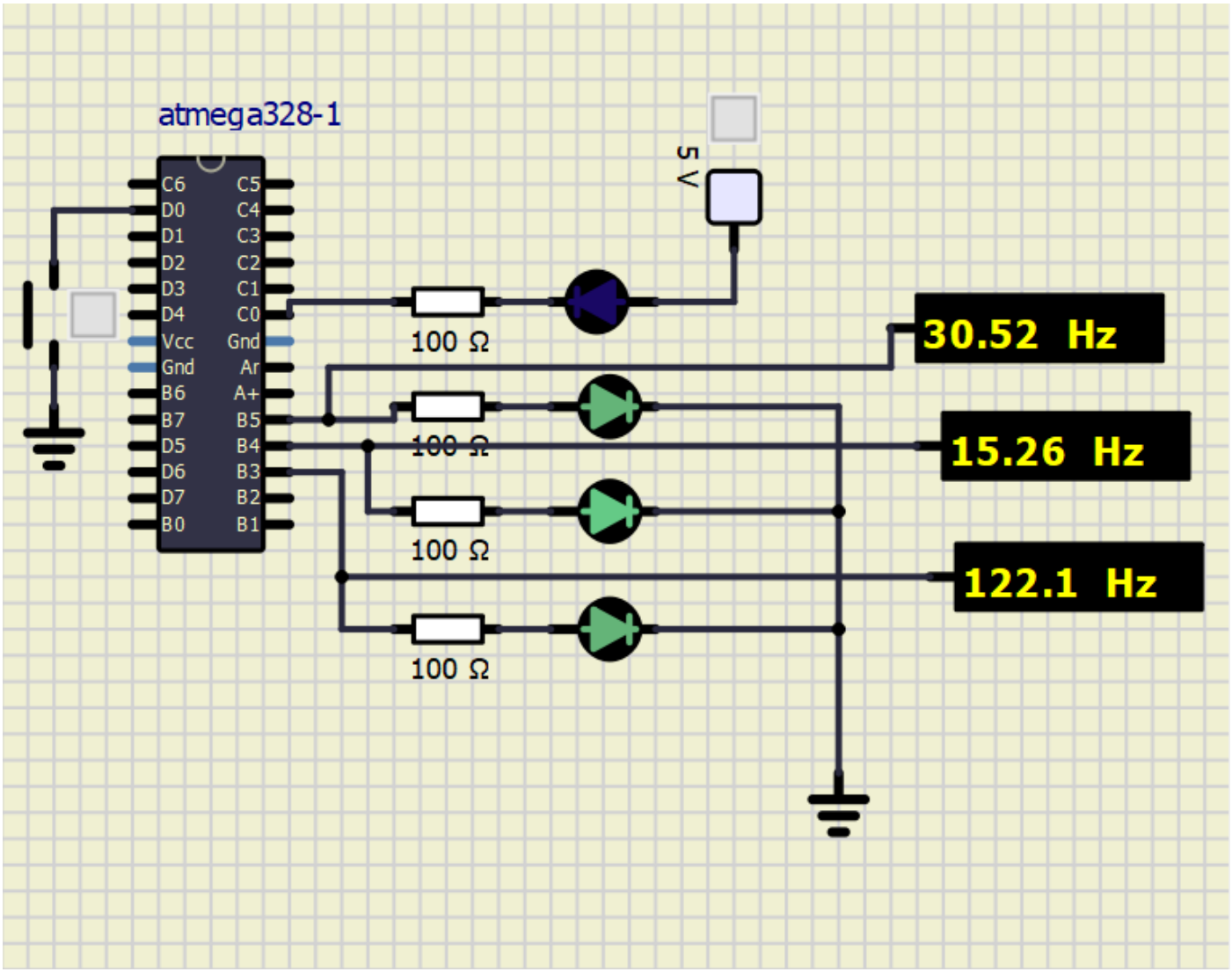
14. 10. 2020

		microseconds						
Module	**Number of bits**	**1**	**8**	**32**	**64**	**128**	**256**	**1024**
Timer/Counter0	**8**	**16u**	**128u**	**--**	**1024u**	**--**	**4096u**	**16384u**
Timer/Counter1	**16**	**4096u**	**32768u**	**--**	**262144u**	**--**	**1048576u**	**4194304u**
Timer/Counter2	**8**	**16u**	**128u**	**512u**	**1024u**	**2048u**	**4096u**	**16384u**

Module	Operation	I/O registers	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	TCN2[7:0]
	Overflow interrupt enable	TIMSK2	TOIE2 (1: enable, 0: disable)

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	Timer/Counter2 Overflow
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	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



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

- Hlavní rozdíl je ve volání funkce a ISR, ISR narozdíl od funkce nemá návratový typ.
- Funkce se většinou volá v hlavní funkci main, z jiné funkce nebo stejné (rekurzivní) funkce.
- ISR je speciální procedura, která je v operačním systému vyvolána při obsluze přerušení a volá se mimo funkci main. Přerušení je asynchronní událost, při jejímž příchodu je přerušena činnost procesoru, je vyvolána obsluha přerušení, a poté činnost procesoru pokračuje na místě, kde bylo přerušení vyvoláno.

2) Describe the behavior of Clear Timer on Compare and Fast PWM modes.

- Časovač v normálním režimu počítá a nastavuje porovnávací bit ve stavovém registru časovače, když se čítač shoduje s porovnávacím registrem, a nastavuje bit přetečení, když se zalomí zpět na nulu. Pokud jsou aktivovány bity aktivace přerušení pro jeden nebo druhý z těchto bitů, avr bude přerušeno. V režimu Fast PWM je výstup pro porovnání výstupu spojený s tímto časovačem nastaven, když se časovač dostane na hodnotu reg srovnání výstupu a resetuje se na 0.
- Clear timer v režimu porovnání nebo v režimu CTC se registr OCR0A používá k manipulaci s rozlišením čítače. V režimu CTC je čítač vymazán na nulu, když hodnota čítače (TCNT0) odpovídá OCR0A. OCR0A definuje vrchol hodnoty čítače, tedy také jeho rozlišení. Tento režim umožňuje větší kontrolu nad porovnání shody výstupní frekvence.

```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
#define LED_D4  PB2

/* 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 one LED
 * on the Multi-function shield using the internal 8- or 16-bit
 * Timer/Counter.
 */
int main(void)
{
    /* Configuration of LED(s) */
    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_16ms();
    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_4ms();
    TIM2_overflow_interrupt_enable();

```

```
54
55      // Enables interrupts by setting the global interrupt mask
56      sei();
57
58      // Infinite loop
59      while (1)
60      {
61          /* Empty loop. All subsequent operations are performed exclusively
62           * inside interrupt service routines ISRs */
63      }
64
65      // Will never reach this
66      return 0;
67  }
68
69  /* Interrupt service routines ----------------------------------------*/
70  /**
71   * ISR starts when Timer/Counter0 overflows. Toggle D1 LED on
72   * Multi-function shield.
73   */
74  ISR(TIMER0_OVF_vect)
75  {
76      // Configuring a GPIO pin for output and toggling it.
77      GPIO_toggle(&PORTB, LED_D1);
78  }
79
80  /**
81   * ISR starts when Timer/Counter1 overflows. Toggle D2 LED on
82   * Multi-function shield.
83   */
84  ISR(TIMER1_OVF_vect)
85  {
86      // Configuring a GPIO pin for output and toggling it.
87      GPIO_toggle(&PORTB, LED_D2);
88  }
89
90  /**
91   * ISR starts when Timer/Counter2 overflows. Toggle D3 LED on
92   * Multi-function shield.
93   */
94  ISR(TIMER2_OVF_vect)
95  {
96      // Configuring a GPIO pin for output and toggling it.
97      GPIO_toggle(&PORTB, LED_D3);
98  }
```

```
1  #ifndef TIMER_H
2  #define TIMER_H
3
4  /***********************************************************************
5   *
6   * Timer library for AVR-GCC.
7   * ATmega328P (Arduino Uno), 16 MHz, AVR 8-bit Toolchain 3.6.2
8   *
9   * Copyright (c) 2019-2020 Tomas Fryza
10  * Dept. of Radio Electronics, Brno University of Technology, Czechia
11  * This work is licensed under the terms of the MIT license.
12  *
13  **********************************************************************/
14
15 /**
16  * @file  timer.h
17  * @brief Timer library for AVR-GCC.
18  *
19  * @details
20  * The library contains macros for controlling the timer modules.
21  *
22  * @note
23  * Based on Microchip Atmel ATmega328P manual and no source file is
24  * needed for the library.
25  *
26  * @copyright (c) 2019-2020 Tomas Fryza
27  * Dept. of Radio Electronics, Brno University of Technology, Czechia
28  * This work is licensed under the terms of the MIT license.
29  */
30
31 /* Includes ----------------------------------------------------------*/
32 #include <avr/io.h>
33
34 /* Defines -----------------------------------------------------------*/
35 /**
36  * @brief Defines prescaler CPU frequency values for Timer/Counter0.
37  * @note  F_CPU = 16 MHz
38  */
39 #define TIM0_stop()             TCCR0B &= ~((1<<CS02) | (1<<CS01) |
      (1<<CS00));
40 #define TIM0_overflow_16us()        TCCR0B &= ~((1<<CS02) | (1<<CS01));
      TCCR0B |= (1<<CS00);
41 #define TIM0_overflow_128us()    TCCR0B &= ~((1<<CS02) | (1<<CS00)); TCCR0B |
      = (1<<CS01);
42 #define TIM0_overflow_1ms()   TCCR0B &= ~(1<<CS02); TCCR0B |= (1<<CS01) |
      (1<<CS00);
43 #define TIM0_overflow_4ms()      TCCR0B &= ~((1<<CS01) | (1<<CS00)); TCCR0B |
      = (1<<CS02);
44 #define TIM0_overflow_16ms()      TCCR1B &= ~(1<<CS01); TCCR0B |= (1<<CS02) |
       (1<<CS00);
45
46 /**
47  * @brief Defines interrupt enable/disable modes for Timer/Counter0.
```

```
48   */
49  #define TIM0_overflow_interrupt_enable()    TIMSK0 |= (1<<TOIE0);
50  #define TIM0_overflow_interrupt_disable()   TIMSK1 &= ~(1<<TOIE0);
51
52
53  /**
54   * @brief Defines prescaler CPU frequency values for Timer/Counter1.
55   * @note  F_CPU = 16 MHz
56   */
57  #define TIM1_stop()             TCCR1B &= ~((1<<CS12) | (1<<CS11) |
      (1<<CS10));
58  #define TIM1_overflow_4ms()     TCCR1B &= ~((1<<CS12) | (1<<CS11)); TCCR1B |=
      (1<<CS10);
59  #define TIM1_overflow_33ms()    TCCR1B &= ~((1<<CS12) | (1<<CS10)); TCCR1B |=
      (1<<CS11);
60  #define TIM1_overflow_262ms()   TCCR1B &= ~(1<<CS12); TCCR1B |= (1<<CS11) |
      (1<<CS10);
61  #define TIM1_overflow_1s()      TCCR1B &= ~((1<<CS11) | (1<<CS10)); TCCR1B |=
      (1<<CS12);
62  #define TIM1_overflow_4s()      TCCR1B &= ~(1<<CS11); TCCR1B |= (1<<CS12) |
      (1<<CS10);
63
64  /**
65   * @brief Defines interrupt enable/disable modes for Timer/Counter1.
66   */
67  #define TIM1_overflow_interrupt_enable()    TIMSK1 |= (1<<TOIE1);
68  #define TIM1_overflow_interrupt_disable()   TIMSK1 &= ~(1<<TOIE1);
69
70
71  /**
72   * @brief Defines prescaler CPU frequency values for Timer/Counter2.
73   * @note  F_CPU = 16 MHz
74   */
75  #define TIM2_stop()               TCCR1B &= ~((1<<CS22) | (1<<CS21) |
      (1<<CS20));
76  #define TIM2_overflow_16us()      TCCR1B &= ~((1<<CS22) | (1<<CS21));
      TCCR2B |= (1<<CS20);
77  #define TIM2_overflow_128us()     TCCR1B &= ~((1<<CS22) | (1<<CS20));
      TCCR1B |= (1<<CS21);
78  #define TIM2_overflow_512us()     TCCR1B &= ~(1<<CS22); TCCR2B |= (1<<CS21)
      | (1<<CS20);
79  #define TIM2_overflow_1ms()       TCCR1B &= ~((1<<CS21) | (1<<CS20));
      TCCR2B |= (1<<CS22);
80  #define TIM2_overflow_2ms()       TCCR1B &= ~(1<<CS21); TCCR2B |= (1<<CS22)
      | (1<<CS20);
81  #define TIM2_overflow_4ms()       TCCR1B &= ~(1<<CS20); TCCR2B |= (1<<CS22)
      | (1<<CS21);
82  #define TIM2_overflow_16ms()      TCCR1B |= (1<<CS21) | (1<<CS22) |
      (1<<CS20);
83
84  /**
85   * @brief Defines interrupt enable/disable modes for Timer/Counter2.
86   */
```

```
87  #define TIM2_overflow_interrupt_enable()    TIMSK2 |= (1<<TOIE2);
88  #define TIM2_overflow_interrupt_disable()   TIMSK2 &= ~(1<<TOIE2);
89
90  #endif
```