


FilipPaul lab4 ...		2 minutes ago	🕒 History
..			
📁 .vscode	readme		9 days ago
📁 include	readme		9 days ago
📁 lib	labs4		5 days ago
📁 pictures	lab4		4 minutes ago
📁 src	labs4		5 days ago
📁 test	readme		9 days ago
📄 .gitignore	readme		9 days ago
📄 platformio.ini	readme		9 days ago
📄 readme.md	lab4		2 minutes ago
📄 simulation.simu	labs4		5 days ago

readme.md 

Lab 4

Preparation tasks

Module	Number of bits	1	8	32	64	128	256	1024
Timer/Counter0	8	16us	128us	--	1.024ms	--	4.096ms	16.384ms
Timer/Counter1	16	4.096ms	32.768ms	--	262.144ms	--	1.048576s	4.194304s
Timer/Counter2	8	16us	128us	512us	1.024ms	2.048ms	4.096ms	16.384ms

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)

Program address	Source	Vector name	Description
0x0000	RESET	--	Reset of the system
0x0002	INT0	External Interrupt 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

Program address	Source	Vector name	Description
0x000A	PCINT2	PCINT2_vect	Pin Change Interrupt Request 2
0x0012	TIMER2_OVF	TIMER2_OVF_vect	Timer/Counter2 Overflow
0x001A	TIMER1_OVF	TIMER1_OVF_vect	Timer/Counter1 Overflow
0x0020	TIMER0_OVF	TIMER0_OVF_vect	Timer/Counter0 Overflow
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	PB5	5
Timer/Counter1	OC1A	PB1	9
	OC1B	PB2	10
Timer/Counter2	OC2A	PB3	11
	OC2B	PD3	3

Source code

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>
//-----Timer 1 -----
/* Defines -----*/
/*
 * @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) | (1<<CS11) | (1<<CS10));
#define TIM1_overflow_1s()    TCCR1B &= ~((1<<CS11) | (1<<CS10)); TCCR1B |= (1<<CS12);
#define TIM1_overflow_4s()    TCCR1B &= ~((1<<CS11) | (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);

//-----Timer 0 -----
/* 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)); //000
#define TIM0_overflow_16us() TCCR0B &= ~(1<<CS02) | (1<<CS01)); TCCR0B |= (1<<CS00); //001
#define TIM0_overflow_128us() TCCR0B &= ~(1<<CS02) | (1<<CS00)); TCCR0B |= (1<<CS01); //010
#define TIM0_overflow_1024us() TCCR0B &= ~(1<<CS02); TCCR0B |= (1<<CS01) | (1<<CS00); //011
#define TIM0_overflow_4096us() TCCR0B &= ~(1<<CS01) | (1<<CS00)); TCCR0B |= (1<<CS02); //100
#define TIM0_overflow_16384us() TCCR0B &= ~(1<<CS01); TCCR0B |= (1<<CS02) | (1<<CS00); //101

/*
 * @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);

//-----Timer 2 -----
/* Defines -----*/
/*
 * @brief Defines prescaler CPU frequency values for Timer/Counter2.
 * @note F_CPU = 16 MHz
 */

#define TIM2_stop()          TCCR2B &= ~(1<<CS22) | (1<<CS21) | (1<<CS20)); //000
#define TIM2_overflow_16us() TCCR2B &= ~(1<<CS22) | (1<<CS21)); TCCR2B |= (1<<CS20); //001
#define TIM2_overflow_128us() TCCR2B &= ~(1<<CS22) | (1<<CS20)); TCCR2B |= (1<<CS21); //010
#define TIM2_overflow_512us() TCCR2B &= ~(1<<CS22); TCCR2B |= (1<<CS21) | (1<<CS20); //011
#define TIM2_overflow_1024us() TCCR2B &= ~(1<<CS21) | (1<<CS20)); TCCR2B |= (1<<CS22); //100
#define TIM2_overflow_2048us() TCCR2B &= ~(1<<CS21); TCCR2B |= (1<<CS22) | (1<<CS20); //101
#define TIM2_overflow_4096us() TCCR2B &= ~(1<<CS20); TCCR2B |= (1<<CS22) | (1<<CS21); //110
#define TIM2_overflow_16384us() TCCR2B |= (1<<CS22) | (1<<CS21) | (1<<CS20); //111

/*
 * @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

```

main.cpp

```

#include <Arduino.h>
/*****
 *
 * 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);
    // WRITE YOUR CODE HERE

    /* Configuration of 8-bit Timer/Counter0 */
    TIM0_overflow_4096us();
    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_16384us();
TIM2_overflow_interrupt_enable();

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

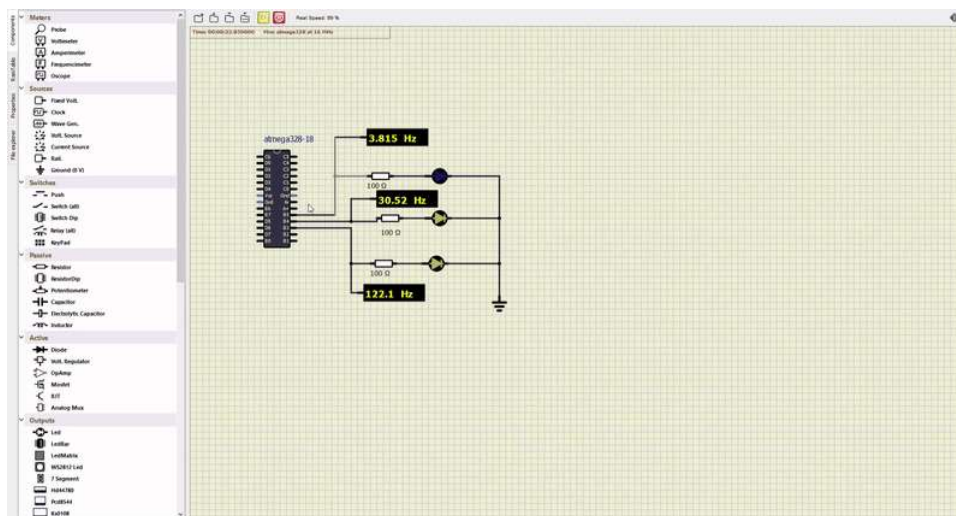
// Infinite loop
while (1)
{
    /* Empty loop. All subsequent operations are performed exclusively
     * inside interrupt service routines ISRs */
}

// 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_D1);
}
ISR(TIM2_OVF_vect)
{
    GPIO_toggle(&PORTB, LED_D2);
}
ISR(TIM0_OVF_vect)
{
    GPIO_toggle(&PORTB, LED_D3);
}

```

Simulation



Answers to questions

ISR() is function that is called whenever specific condition meets interrupts condition, input of this function is predefined vector, that defines type of interrupt. For EX: ISR(TIMERO_OVF_vect) execute its code whenever Timer 1 overflow. This function is called asynchronously.

Common C function is called from source code from line, where the function is called. This function is called synchronously.

Clear Timer on Compare - this method counts into given number of Timer, if the number of counts is equal to our given number, it resets counter value and start again from 0. We can define what should be executed in ISR function.

Fast PWM modes, uses Timers for changing automatically logical levels at GPIOs without calling ISR function from script.