Lab Assignment 4

# Masauso Lungu 209533

### Preparation task.

Table with overflow Times

| **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 |

### Timer Library

| **Module** | **Operation** | **I/O register(s)** | **Bit(s)** |
| --- | --- | --- | --- |
| Timer/Counter0 | Prescaler  8-bit data value Overflow interrupt enable | TCCR0B  TCNT0 TIMSK0 | CS02, CS01, CS00 (000: stopped, 001: 1, 010: 64, 100: 256, 101: 1024) TCNT0[7:0] TOIE0(1: enable, 0: disable) |
| Timer/Counter1 | Prescaler  16-bit data value Overflow interrupt enable | TCCR1B  TCNT1H, TCNT1L TIMSK1 | CS12, CS11, CS10 (000: stopped, 001: 1, 010: 8, 011: 64, 100: 256, 101: 1024) TCNT1[15:0] TOIE1 (1: enable, 0: disable) |
| Timer/Counter2 | Prescaler  8-bit data value Overflow interrupt enable | TCCR2B  TCNT2 TIMSK2 | CS22, CS11, CS10 (000: stopped, 001: 1, 010: 8, 011: 32, 100: 64, 101: 128, 110: 256, 111: 1024) TCNT2[7:0] TOIE2(1: enable, 0; disable) |

**timer.h** file.

/\*

\* timer.h

\*

\* Created: 10/14/2020 11:23:08

\* Author : masau

\*/

#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>

/\* Defines -----------------------------------------------------------\*/

//Timer/counter0

/\*\*

\* @brief Defines prescaler CPU frequency values for Timer/Counter1.

\* @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<<CS01) | (1<<CS00)); TCCR0B |= (1<<CS02);

#define TIM0\_overflow\_16ms() TCCR1B &= ~(1<<CS01); 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);

//Timer/counter1

/\*\*

\* @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);

//Timer/counter2

/\*\*

\* @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<<CS00);

#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<<CS20) | (1<<CS21);

/\*\*

\* @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

**Table with ATmega328P selected interrupt sources**

| **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 1 |
| 0x0006 | PCINT0 | PCINT0\_vect | Pin change interrupt request 0 |
| 0x0008 | PCINT1 | PCINT1\_vect | Pin change interrupt request 1 |
| 0x00A | PCINT2 | PCINT2\_vect | Pin change interrupt request 2 |
| 0x00C | WDT | WDT\_vect | Watchdaog 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\_vect | USART\_Rx complete |
| 0x002A | ADC | ADC\_vect | ADC Conversion Complete |
| 0x0030 | TWI | TWI\_vect | 2-wire Serial interface |

**Main.c**

/\*

\* timers.c

\*

\* Created: 10/14/2020 11:23:08

\* Author : masau

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* 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 PC0

/\* 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);

// WRITE YOUR CODE HERE

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();

// 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/Counter0 overflows. Toggle LED D1 on

\* Multi-function shield. \*/

ISR(TIMER0\_OVF\_vect)

{

GPIO\_toggle(&PORTB, LED\_D1);

}

/\* Interrupt service routines ----------------------------------------\*/

/\*\*

\* ISR starts when Timer/Counter1 overflows. Toggle LED D2 on

\* Multi-function shield. \*/

ISR(TIMER1\_OVF\_vect)

{

// WRITE YOUR CODE HERE

GPIO\_toggle(&PORTB, LED\_D2);

}

/\* Interrupt service routines ----------------------------------------\*/

/\*\*

\* ISR starts when Timer/Counter2 overflows. Toggle LED D3 on

\* Multi-function shield. \*/

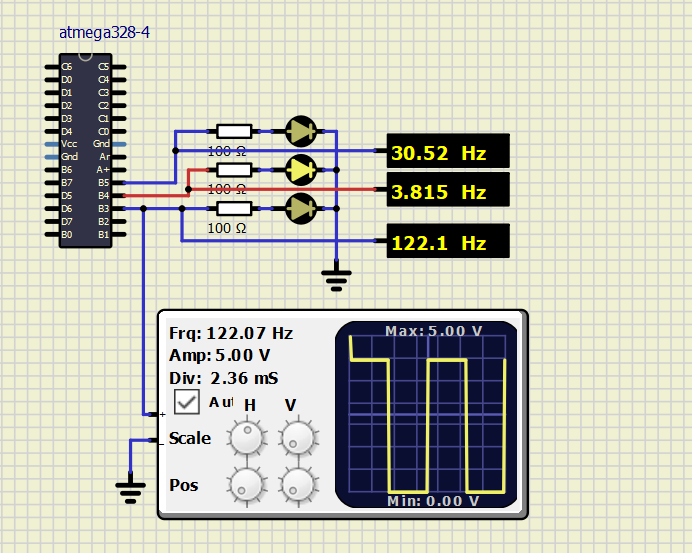
ISR(TIMER2\_OVF\_vect)

{

GPIO\_toggle(&PORTB, LED\_D3);

}

**SimulIDE circuit**

****

**Difference between a C function and an Interrupt service routine(ISR)**

Both ISR and C function are a set of instruction to perform a particular task. The difference between C function and ISR is in the way they get called.

The C function is generally called in the main program code and get executed.

While an ISR code get executed automatically whenever an interrupt is triggered in the hardware peripheral or processor exception signals. ISR takes no parameters and returns no results and unlike the normal function, ISR can be called at anytime when an interrupt occur.

### PWM

**Table with channels of ATmega328P**

| **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 |

**The behavior of Clear Timer on Compare and Fast PWM modes**

In Clear Timer on Compare (CTC) mode, the OCR0 Register is used to manipulate the counter resolution.

-The counter is cleared to zero when the counter value (TCNT0) matches the OCR0.

-There is greater control of the Compare Match output frequency.

In fast fast PWM mode, the counter is incremented until the counter value matches the TOP value, then cleared at the following timer clock cycle.

* The counter counts from BOTTOM to TOP then restarts from BOTTOM