**Lab 4: Interrupts, timers**

* Table with overflow times.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Module** | **Number of bits** | **1** | **8** | **32** | **64** | **128** | **256** | **1024** |
| Timer/Counter0 | 8 | 16u | 128u | -- | 1.024m | -- | 4.096m | 16.384m |
| Timer/Counter1 | 16 | 4.096m | 32.76m | -- | 262.14m | -- | 1.048 | 4.19 |
| Timer/Counter2 | 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,

|  |  |  |  |
| --- | --- | --- | --- |
| **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 | 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

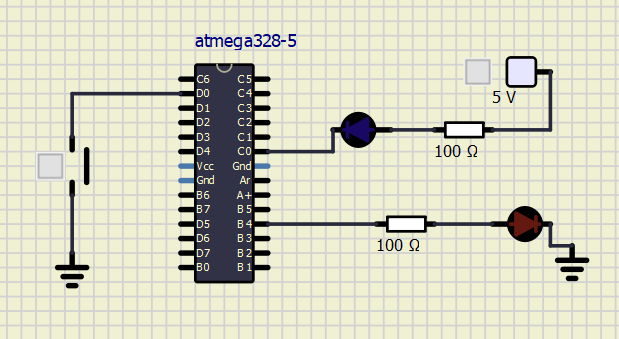
\* Multi-function shield. \*/

ISR(TIMER1\_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++;

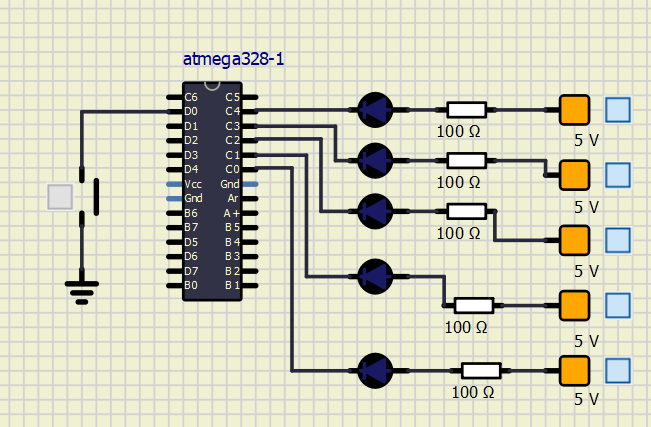
}else{

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 |

