**Lab 4: Interrupts, timers**

David Garcia Torre

* Table with overflow times.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Module | Number of bits | 1 | 8 | 32 | 64 | 128 | 256 | 1024 |
| Timer/Counter0 | 8 | 16us | 128us | -- | 1ms | -- | 4ms | 16ms |
| Timer/Counter1 | 16 | 4ms | 33ms | -- | 262ms | -- | 1s | 4s |
| Timer/Counter2 | 8 | 16us | 128us | 512us | 1ms | 2ms | 4ms | 16ms |

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\_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<<CS02) | (1<<CS00)); TCCR0B |= (1<<CS02);

#define TIM0\_overflow\_16ms() 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);

/\*\*

\* @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<<CS20);

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

/\*\*

\* @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

|  |  |  |  |
| --- | --- | --- | --- |
| 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 Knight Rider application main.c,

/\* 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 LED\_RED6 PC5 // AVR pin where red LED 6 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

/\* Functions ---------------------------------------------------------

\* Main function where the program execution begins.

\*/

int leds[] ={LED\_RED1,LED\_RED2,LED\_RED3,LED\_RED4,LED\_RED5,LED\_RED6};

int a=0,b=0;

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);

GPIO\_config\_output(&DDRC, LED\_RED6);

GPIO\_write\_low(&DDRC, LED\_RED6);

/\* Configuration of 16-bit Timer/Counter1

\* Set prescaler and enable overflow interrupt \*/

TIM1\_overflow\_262ms();

TIM1\_overflow\_interrupt\_enable();

// Enables interrupts by setting the global interrupt mask

sei();

// Infinite loop

for (;;){

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* leds[] ={LED\_RED1,LED\_RED2,LED\_RED3,LED\_RED4,LED\_RED5,LED\_RED6};

*uint8\_t* a = 0;

*uint8\_t* b = 0;

if(b == 5){

a = 1;

GPIO\_write\_high(&DDRC,leds[5]);

}else if(b == 0){

a = 0;

GPIO\_write\_high(&DDRC,leds[5]);

}

if(a == 0){

b++;

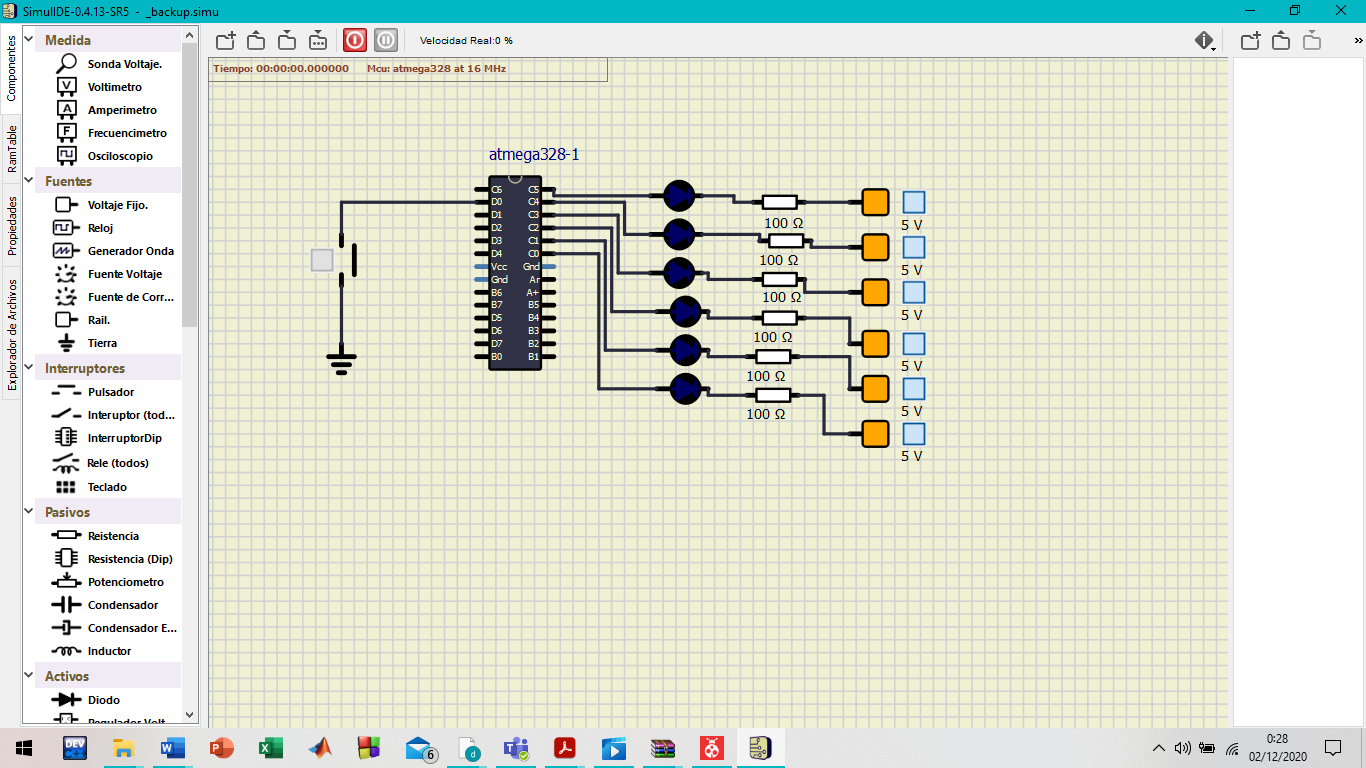
}else{

b--;

}

GPIO\_write\_low(&DDRC,leds[b]);

}



The difference between a normal function in c and an interruption, is that the normal function follows the order established in the code, when the call to the function arrives, then the function enters, in case of the interruption, is that when it arrives, it skips directly to the process regardless of the order.

* 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 |

