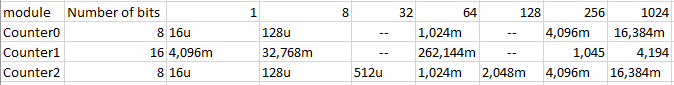
1.



2.

#ifndef TIMER\_H

#define TIMER\_H

/\* Includes ----------------------------------------------------------\*/

#include <avr/io.h>

/\* 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));

#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() TCCR0B &= ~(1<<CS01); TCCR0B |= (1<<CS02) | (1<<CS00);

/\*\*

\* @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 prescaler CPU frequency values for Timer/Counter2.

\* @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/Counter0.

\*/

#define TIM0\_overflow\_interrupt\_enable() TIMSK0 |= (1<<TOIE0);

#define TIM0\_overflow\_interrupt\_disable() TIMSK0 &= ~(1<<TOIE0);

/\*\*

\* @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);

/\*\*

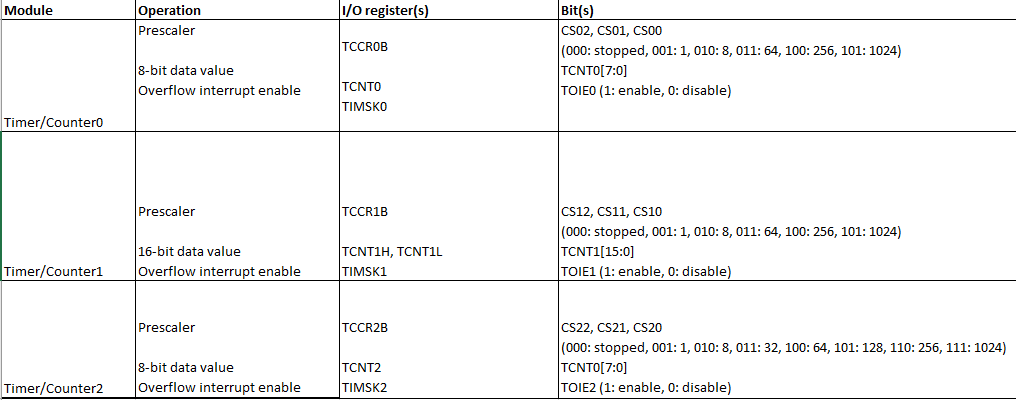
\* @brief Defines interrupt enable/disable modes for Timer/Counter2.

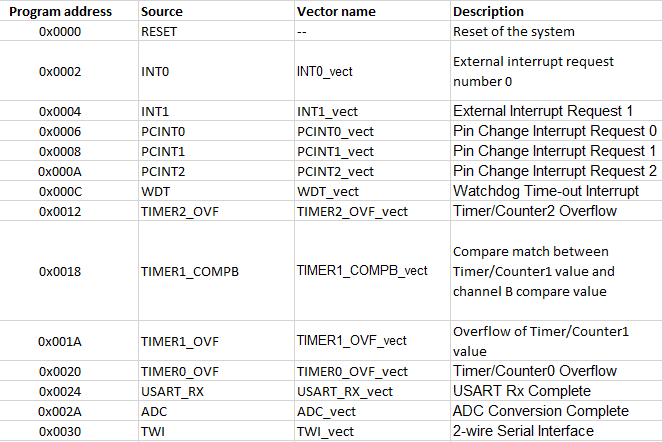
\*/

#define TIM2\_overflow\_interrupt\_enable() TIMSK2 |= (1<<TOIE2);

#define TIM2\_overflow\_interrupt\_disable() TIMSK2 &= ~(1<<TOIE2);

#endif





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

\*

\* 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

\* Set prescaler and enable overflow interrupt \*/

TIM0\_overflow\_1ms();

TIM0\_overflow\_interrupt\_enable();

/\* Configuration of 16-bit Timer/Counter1

\* Set prescaler and enable overflow interrupt \*/

TIM1\_overflow\_4ms();

TIM1\_overflow\_interrupt\_enable();

/\* Configuration of 8-bit Timer/Counter2

\* Set prescaler and enable overflow interrupt \*/

TIM2\_overflow\_2ms();

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 D1 LED on

\* Multi-function shield. \*/

ISR(TIMER0\_OVF\_vect)

{

// WRITE YOUR CODE HERE

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

}

ISR(TIMER1\_OVF\_vect)

{

// WRITE YOUR CODE HERE

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

}

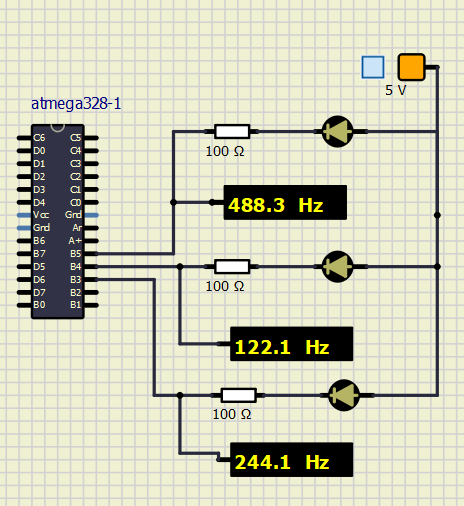
ISR(TIMER2\_OVF\_vect)

{

// WRITE YOUR CODE HERE

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

}



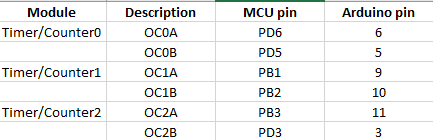
**common C function**

Musí se volat v programu. Může mýt nějaký vstup a návratovou hodnotu.

**interrupt service routine**

Provede se jen v případě, že nastane událost přerušení. To zastaví právě vykonávaný program a provede se tělo ISR. Po skončení se vykonává dále program, který byl přerušen. Nemá vstupní proměnnou a nemůže vracet hodnotu.

3.



**behavior of Clear Timer on Compare**

TCNT0 je hodnota čítače. OCR0A registr určuje hodnotu nulování. TCNT0 se resetuje při rovnosti s OCR0A. Při rovnosti nastane přerušení.

**Fast PWM mode**

Při shodě TCNT0 a OCR0x se nastaví nízká úroveň (vznikne přerušení). Při přetečení TCNT0 se nastaví vysoká úroveň (přerušením).