DIGITAL ELECTRONICS 2

LAB ASSIGNMENT 4

Name:NADIR OSMAN AL-WATTAR
ID:225796
GITHUB:https://github.com/Nadir011995/Digital-electronics-2.git

Table 1:Overflow times

Module	Number of bits	1	8	32	64	128	256	1024
Timer/Counter0	8	16u	128u	--	1024u	--	4096u	16384u
Timer/Counter1	16	4096u	32768u	--	0,262144	--	1,0485	4,194
Timer/Counter2	8	16u	128u	512u	1024u	2048u	4096u	16384u

Table 2: Timer 0/1/2

Module	Operation	I/O register(s)	Bits
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/Counter 1	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/Counter 2	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)

Table 3: ATmega 328P

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 Request2
0x000C	WDT	WDT_vect	Watchdog Timeout 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_RX_vect	USART Rx Complete
0x002A	ADC	ADC_vect	ADC Conversion Complete
0x0030	TWI	TWI_vect	2-wire Serial Interface

Table 4: Pwm Channels of Atmega32P

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

1.Describe the difference between a common C function and interrupt service routine.

An interrupt is a completely different concept than a common C Function. A common C function is called when in the program there is a call for it. An Interrupt is a feature of the processor hardware, however, is linked to an event. If this event occurs, then the interrupt function will be called. Furthermore, interrupts are outside the C standard, so there is no standardized language construct for it.

2.Describe the behavior of Clear Timer on Compare and Fast PWM modes.

Clear Timer on Compare (CTC):

The counter counts up until it matches OCRnx (BOTTOM-> OCRnx: Match!) And is then set to zero. The maximum value can therefore be easily determined using the OCRnx register. In concrete terms, this means that the base frequency generated by the Prescaler in this mode is divided again by the value of OCRnx.

Fast PWM:

The counter counts from BOTTOM to TOP, in which TOP can either be 0xFF or OCRnx.

When a match is

a) non-inverting mode of the counter cleared and set at BOTTOM

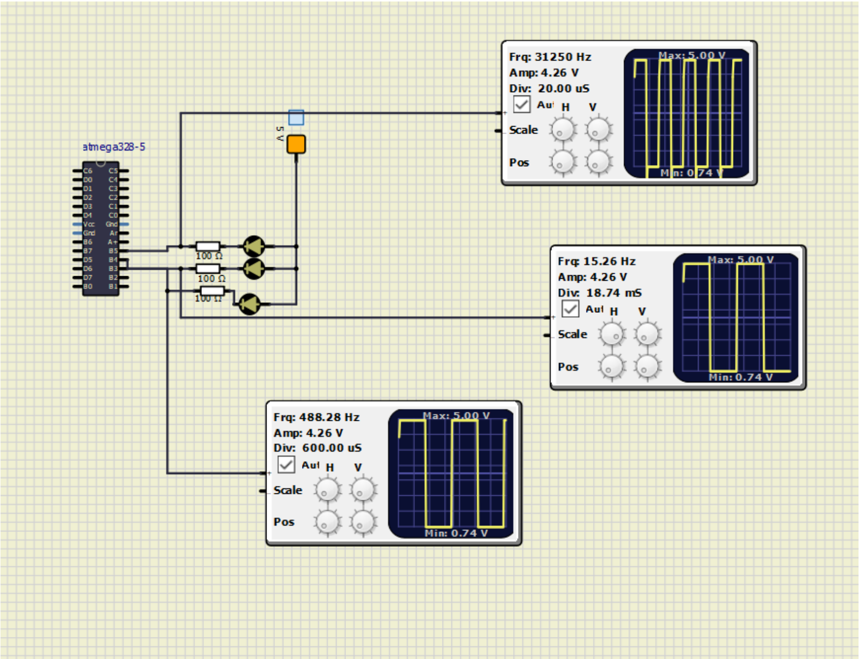
b) inverting mode of the counter set, and cleared at BOTTOM.Sounds complicated in theory, in practice it just inverts the output.

This mode has an asymmetrical output form because the output is switched periodically and then inverted again after the variable pulse length has elapsed. There is also a toggle mode, which is only available for output OC0A.

/****   HEADER FILE   ****/

```
#ifndef TIMER_H
#define TIMER_H

/* Includes ----------------------------------------------------------*/
#include <avr/io.h>

/* Defines  TIM0 -----------------------------------------------------------*/

#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_4ms()     TCCR0B &= ~((1<<CS00)  | (1<<CS01)); TCCR0B
|=(1<<CS02);//100
#define TIM0_overflow_16ms()    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);


/*
 * @brief Defines prescaler CPU frequency values for Timer/Counter1.
 * @note  F_CPU = 16 MHz
 */
/* Defines  TIM1 -----------------------------------------------------------*/
#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);


/* Defines  TIM2 -------------------------------------------------------------*/


#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<<CS20)  | (1<<CS21)); TCCR2B
|=(1<<CS22);//100
#define TIM2_overflow_2ms()      TCCR2B &= ~(1<<CS21); TCCR2B |= (1<<CS22) |
(1<<CS20);//101
#define TIM2_overflow_4ms()      TCCR2B &= ~(1<<CS20); TCCR2B |= (1<<CS22) |
(1<<CS21);//110
#define TIM2_overflow_16ms()      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
```



Frq: 31250 Hz
Amp: 4.26 V
Div: 20.00 uS

Frq: 15.26 Hz
Amp: 4.26 V
Div: 18.74 mS

Frq: 488.28 Hz
Amp: 4.26 V
Div: 600.00 uS

Figure 1:Three different overflow times and ISRs

/*****   MAIN.C   *****/

```c
/***********************************************************************
 *
 * 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
#define BIN         0

/* 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.
 */

uint8_t LEDs_array[4] = {LED_D1,LED_D2,LED_D3,LED_D4};
       int LED= 0;
       int move= 0;

int main(void)
{
       int perform=0;

    /* Configuration of LED(s) */


    GPIO_config_output(&DDRB, LEDs_array[0]); // this led is going to blink first.
    GPIO_write_low(&PORTB, LEDs_array[0]);

    GPIO_config_output(&DDRB, LEDs_array[1]);
    GPIO_write_high(&PORTB, LEDs_array[1]);


       GPIO_config_output(&DDRB, LEDs_array[2]);
       GPIO_write_high(&PORTB, LEDs_array[2]);


       GPIO_config_output(&DDRB, LEDs_array[3]);
       GPIO_write_high(&PORTB, LEDs_array[3]);
       GPIO_config_input_pullup(&DDRD, BIN);

              /* Configuration of 16-bit Timer/Counter0
            * Set prescaler and enable overflow interrupt */



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


    // Infinite loop
    while (1)
    {
              perform=GPIO_read(&PIND, BIN);

              if(perform==1)// if button is not pressed the Leds will blink slowly!
              {
                    TIM1_overflow_1s();
                    TIM1_overflow_interrupt_enable();

              }
              else
              {
                    TIM1_overflow_262ms();//if button is pressed the Leds will blink
faster!
                    TIM1_overflow_interrupt_enable();

              }


    }

    // Will never reach this
    return 0;
}


ISR(TIMER1_OVF_vect)
{
    GPIO_toggle(&PORTB,LEDs_array[LED]);

       if(LED==0) //check for PB5
       {
              move=0; // it has to move down
       }else if(LED==3)
       {
              move=1; // it has to move up
       }
       if(move==0) // moving in down direction
       {
              LED++;

       }
       else if(move==1)// moving in up direction
       {
              LED--;
       }
    GPIO_toggle(&PORTB,LEDs_array[LED]);
}
```

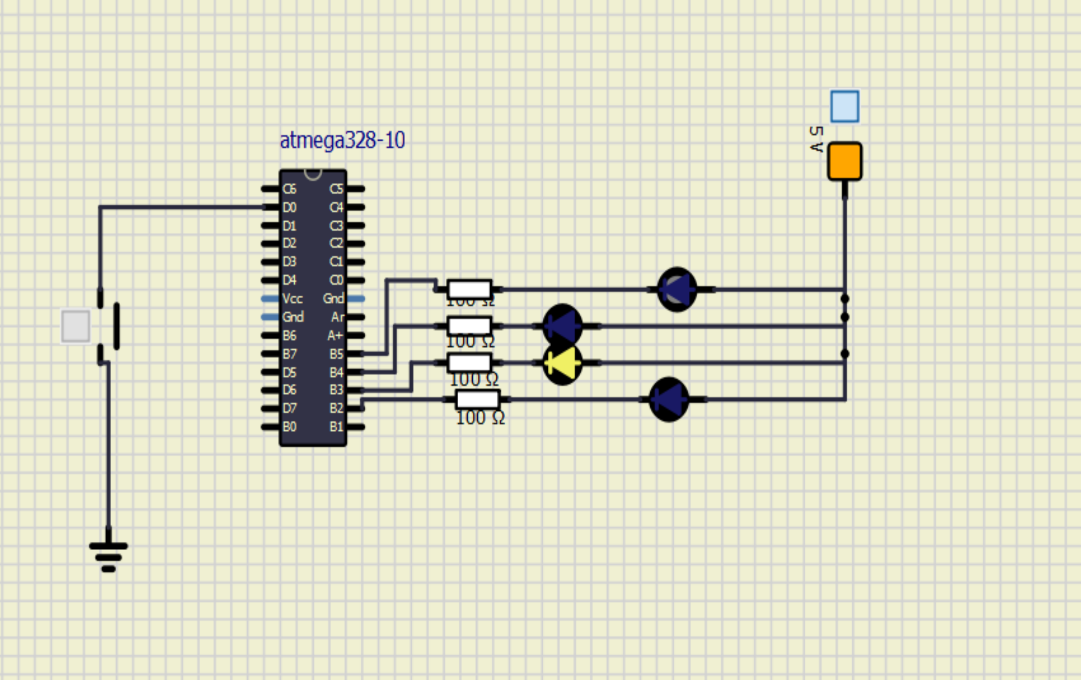


Figure 2: Circuit for Knight Rider Style