Output Compare Match (PWM)

Executed by:Kevin Anum
Mailbox:108
Embedded System Lab
Natural Science Laboratory, Jacobs University Bremen

September 5, 2017

Introduction

In this lab we varied the brightness of the LED using the CTC (Clear Time on Compare match) mode and the PWM (Pulse Width Modulation) mode.

In order to use the CTC mode ,we use the the timer (TCTN1)and the Output Compare Register (OCR1A).

In Clear Time on Compare match (CTC) mode, the Timer (TCTN1) is increased by one in each clock cycle and will be automatically clear to zero when it matches OCR1A.

PWM are used to control the brightness using the time ratio.This is done by varying the value of the OCR1A.

By adjusting the on-time ratio, one can control the brightness of the LED. This type of digital-to-analog converting (convert a digital 16-bits value to an analog on-time ratio) is called Pulse Width Modulation (PWM).

Mode	WGM13	(CTC1)	(PWM11)	(PWM10)	Operation	TOP	OCR1x at	Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Immediate	MAX

Figure 1: The table shows the modes used to generate a PWM and a CTC.

Pre-lab

1. **Read the the corresponding content in the data sheet and study the functions of necessary registers in order to enable**

the Output Compare Interrupt.

TIMSK1 – Timer/Counter1 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
(0x6F)	–	–	ICIE1	–	–	OCIE1B	OCIE1A	TOIE1	TIMSK1
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figure 2: The figure shows the Timer/Counter 1.This is the register in which the Output Compare Interrupt (for CTC).We set bit 1 to 1 to enable the interrupt

Table 16-5. Clock Select Bit Description

CS12	CS11	CS10	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/1 (No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T1 pin. Clock on falling edge.
1	1	1	External clock source on T1 pin. Clock on rising edge.

Figure 3: The table shows the Clock bit description.

TCCR1A – Timer/Counter1 Control Register A

Bit	7	6	5	4	3	2	1	0	
(0x80)	COM1A1	COM1A0	COM1B1	COM1B0	–	–	WGM11	WGM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

TCCR1B – Timer/Counter1 Control Register B

Bit	7	6	5	4	3	2	1	0	
(0x81)	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figure 4: The figure shows Time control register A and Time control register B.

We use Fig.3 and Fig.5 to determine what we input in the Time control A and Time control B.In this lab we used a prescaler of clk/1024 and we used a CTC with the TOP as OCR1A.

2. **Calculate a proper combination of the value in OCR1A and counting frequency to have an Output Compare Match every 1 second.**
   The CPU clock rate is $8 \times 10^6 Hz$ since this value is big we use a prescaler to reduce the value. In this lab we used 1024 therefore:
   $\frac{8 \times 10^6}{1024} = 7812.5$

3. Design a program that you can generate PWM signal on the pins of PORTD.

```
#include <avr/io.h>
#define F_CPU 8000000UL
#include<util/delay.h>
#include<avr/interrupt.h>

int main(void)
{
        DDRB = 0xFF;
        TCCR1B |= (1<<3);
        TCCR1A |= (1<<0);
        TIMSK1 |= (1<<1);
        TCCR1A |= (1<<7);
        TCNT1 = 0x00;
        TCCR1B |= (1<<0);

        while(1){
                for(int i=0; i<255; i++){
                        OCR1A = i;
                        _delay_ms(20);
                }
        }

}
```

4. **Find out how to toggle OC1A pin in CTC mode and how to generate a fast PWM signal on OC1A pin by setting proper registers.**

Table 16-2. Compare Output Mode, Fast PWM[1]

COM1A1/COM1B1	COM1A0/COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	WGM13:0 = 14 or 15: Toggle OC1A on Compare Match, OC1B disconnected (normal port operation). For all other WGM1 settings, normal port operation, OC1A/OC1B disconnected.
1	0	Clear OC1A/OC1B on Compare Match, set OC1A/OC1B at BOTTOM (non-inverting mode)
1	1	Set OC1A/OC1B on Compare Match, clear OC1A/OC1B at BOTTOM (inverting mode)

Figure 5: The table shows the compare output mode for fast PWM.From the table we use the third row.

Circuit Design

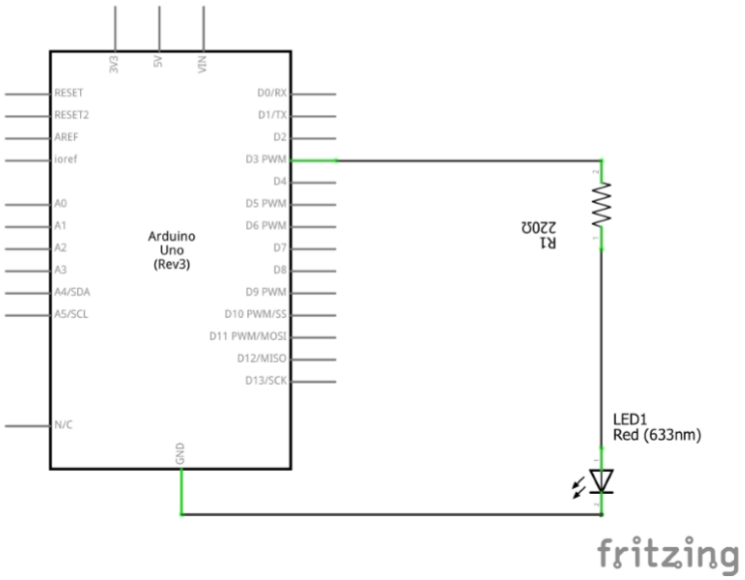

Figure 6: The figure shows the circuit of part 1 and part 2.In part 1 and 2 are connected to port B pin 9.

C Code

Lab 5.1.1

```
#include <avr/io.h>
#define F_CPU 8000000UL
```

```c
#include <util/delay.h>
#include <avr/interrupt.h>

int main(void)
{
        DDRB = 0xFF;// turns Port B as an output
        PORTB |= (1 << 9);
        TCCR1B = 0b00001101;//pre-scalar of 1024 and set CTC mode
        TIMSK1 = 0b00000010;// enables Output Compare interrupt
        TCNT1 = 0x00;//intial value of counter
        OCR1A = 0x1E84;//intial value of ORC1A
        sei();//enables global interrupt


        while(1){
        }

}

ISR(TIMER1_COMPA_vect){
        PORTB ^= 0xFF;//toggle PORTD
        TCNT1 = 0x00;
}
```

Lab.5.1.2

```c
#include  <avr/io.h>
#define F_CPU 8000000UL
#include <util/delay.h>
#include <avr/interrupt.h>

int count;
int on = 1;
int rise = 10;
int main(void)
{
        DDRD = 0xFF;
        TCCR1B = 0b00001001;
        TIMSK1 = 0b00000010;
        TCNT1 = 0x00;
        OCR1A = 0xFF;
```

```
        sei();
        PORTD |= (1 << 7);
        while(1){
        }

}

ISR(TIMER1_COMPA_vect){
          if(on) {
                OCR1A = 20000;//brightest value
                on = 0;
        } else {
                OCR1A = rise;//slowing increases the brightness
                on = 1;
        }
        rise += 10;//increases the brightness
        if(rise > 20000) rise = 10;
        PORTD ^= (1 << 7);
}
```

Lab.5.2

```
#include <avr/io.h>
#define F_CPU 8000000UL
#include<util/delay.h>
#include<avr/interrupt.h>

int main(void)
{
        DDRB = 0xFF;
        TCCR1B |= (1<<3);//enable PWM mode(WGM12)
        TCCR1A |= (1<<0);//enable PWM mode(WGM10)
        TIMSK1 |= (1<<1);//enable Output Compare interrupt
        TCCR1A |= (1<<7);// enable toggling of OC1A
        TCNT1 = 0x00;
        TCCR1B |= (1<<0);

        while(1){
                for(int i=0; i<255; i++){
                        OCR1A = i;//gradual increase to max brightness
                        //when i is 255
```

```
                    _delay_ms(20);
                //delay of 20 ms so observe increase in brightness
                }
        }

}
```

Conculsion

In this lab we learned how to control the brightness of an LED using the CTC and PWM.