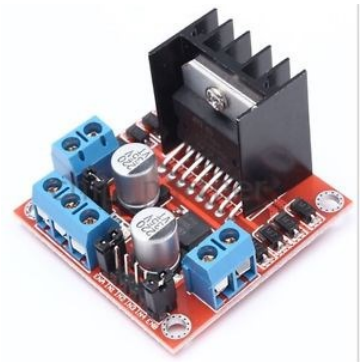
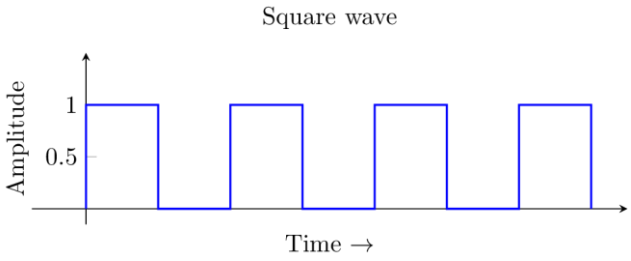
University of the West Indies
Faculty of Engineering

ENGR2205
Engineering Laboratory and Design IV

Laboratory #1

Waveform Generation and DC Motor Control





Written by: Dr Lindon Falconer

Laboratory Date: February 23, 2024

Due Date: February 23, 2024

Table of Contents

1 Objectives

- The aim of this laboratory is to expose students to the practical implementation of waveform generation and DC motor control using the AVR microcontroller.

- Students will use the hardware waveform generation features of the microcontroller to:
    1. Generate square waves of different frequencies to produce sound using a buzzer.
    2. Generate PWM waves to control the direction of a servo motor.
    3. Generate PWM waves to control the speed of a permanent magnet DC motor.

- The motors will be connected to an obstacle avoidance robot so that students can control the direction and speed of the robot.

2 Preparation

- Review the following topics in the ECSE2104 lectures.
    - Timer0 and Timer1
- Review the Appendix in this manual
- Install Proteus application

3 Files Provided by Lecturer

- Template for controlling the servo motor and DC motors on the obstacle avoidance robot in file robot.c

4 Task 1: Create Sound using AVR Hardware Waveform Generation

In this section you will use the ATMega32 to generate a square wave of different frequencies on pin OC0 (PB3). PINB3 is connected to buzzer circuit as shown in Figure 1. Review Non-PWM Waveform Generation on the ATMega32 Microcontroller in the Appendix.

Write an AVR C-program that uses the hardware waveform generating feature of the microcontroller to continuously create a square wave of 500Hz, 1 kHz, 5 kHz and 10 kHz on PINB3 of the ATmega32 microcontroller. Use a delay of five seconds between each sound. Assume the microcontroller is clocked by an 8 MHz crystal.

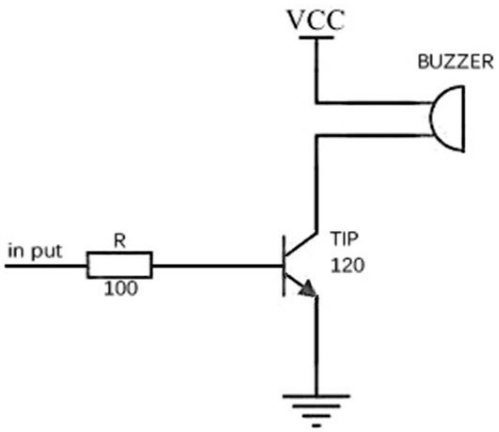


Figure 1: Buzzer Circuit.

5 Task 2: Control Servo Motor

In this section of the experiment, students are required to modify a code that controls the direction of a servo motor. Servo motors are controlled by sending a continuous PWM signal to its input. An example of a Servo motor used in this laboratory is shown in Figure 2.

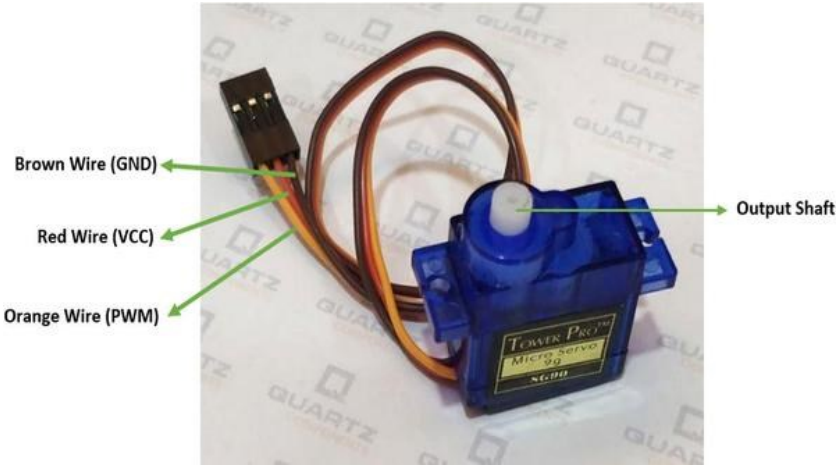


Figure 2: Servo Motor

The servo motor is part of an obstacle avoidance robot. An ultrasonic sensor is placed on top of the motor as shown in Figure 3.

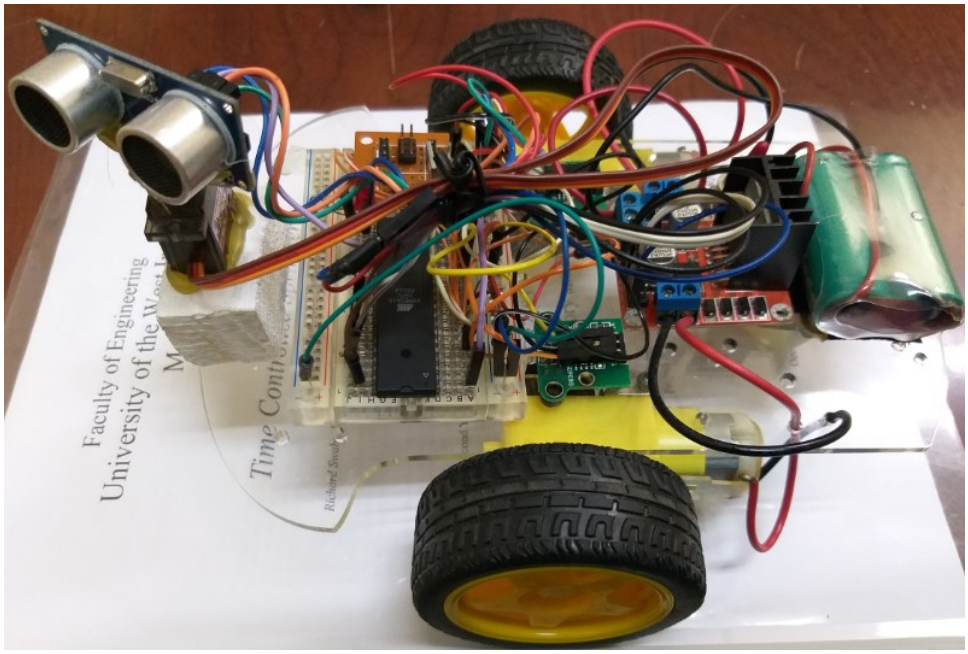


Figure 3: Obstacle Avoidance Robot

You are given the task to turn the motor 0^0, 90^0, and 180^0 as show in the diagram in Figure 4.

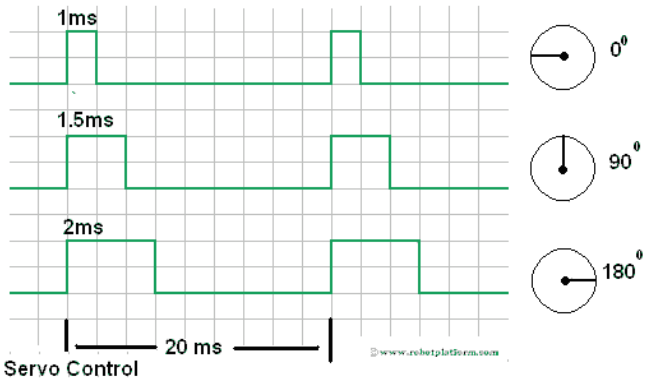


Figure 4: PWM Signal and Servo Motor Direction

The OC2 Pin on the ATmega32 microcontroller is used to generate the PWM signal for the servo motor. Review PWM generation on OC2 pin in the Appendix and derive the values needed for TCCR2 and OCR2 for creating the PWM signals in Figure 4.

Your task is to apply the following sequence to the servo motor on power up:

Turn left (0^0) - delay 2 second – **Turn Center (90^0)** – delay 2 second – **Turn right (180^0)** – delay 2 second – **Turn Center** (90^0) – **Repeat.**

5.1 Submission

Modify the code *robot.c* file with your solution and submit for evaluation.

6 Task 3: Permanent Magnet Motor Direction and Speed Control

In this section students are required to control the speed and direction of two permanent magnet DC motors. The motor controls the wheels on the robot in Figure 3.

The LM298 Motor Driver Module is used to control the motors. The circuit in Figure 5 illustrates the connection for the motor driver and the microcontroller.

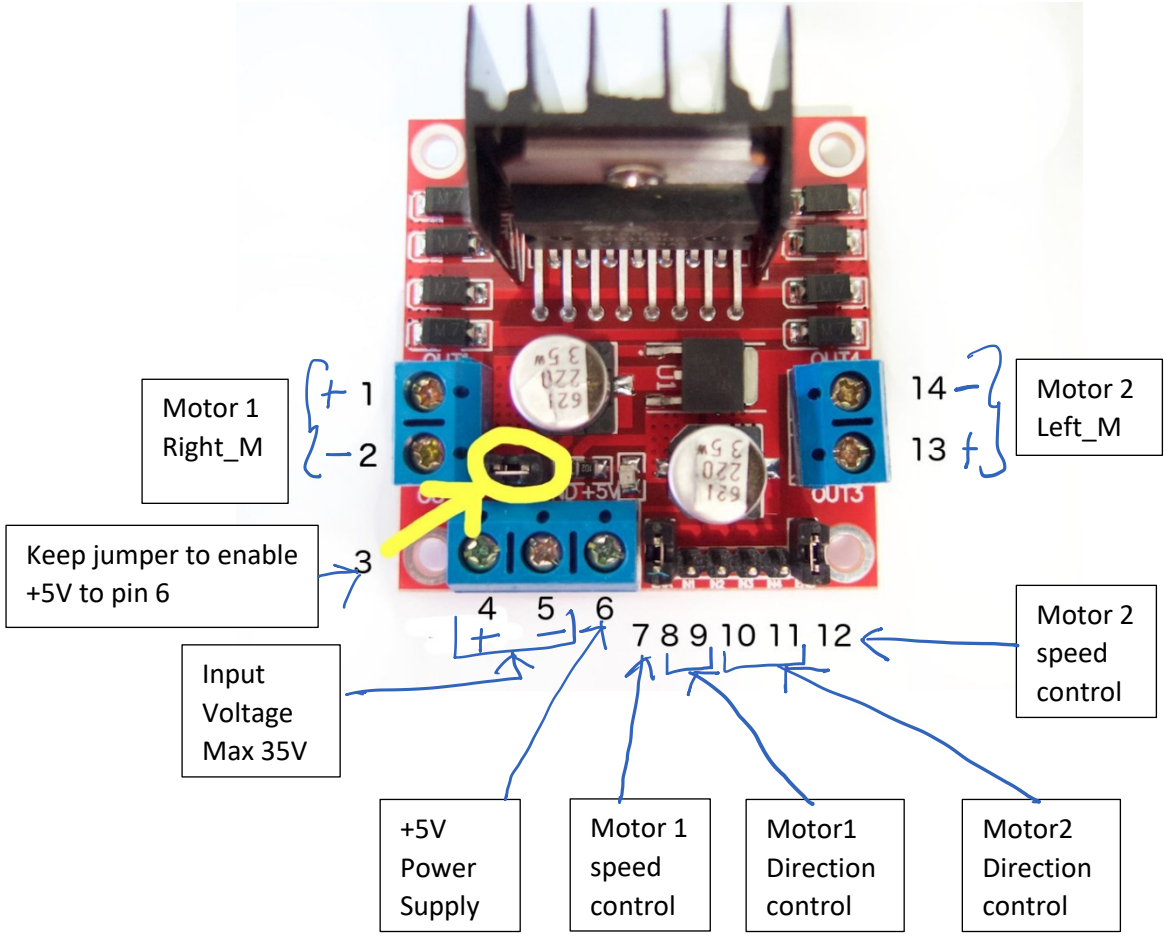


Figure 5: Motor Driver Module Connections

6.1 Motor Direction Control

#	MCU	Motor Driver	Purpose
1	PD5 (OC1B)	7	Control Speed of Motor 1 (Right Motor)
2	PD4 (OC1A)	12	Control Speed of Motor 2 (Left Motor)
3	PA4	11	(LeftM) Motor 2 direction
4	PA5	10	(LeftM) Motor 2 direction
5	PA6	9	(RightM) Motor 1 direction
6	PA7	8	(RightM) Motor 1 direction

Table 1: Microcontroller connection to Motor Driver

#	PA4	PA5	PA6	PA7
Motor 2 (Left Motor) Clockwise	0	1		
Motor 2 (Left Motor) Counter-Clockwise	1	0		
Motor 1 (Right Motor) Clockwise			1	0
Motor 1 (Right Motor) Counter-Clockwise			0	1
Motor Stopped	0	0	0	0

Table 2: Left and Right Motor Direction Control

6.2 Motor Speed Control

PWM signals on OC1A and OC1B pins are used to control the speed of the right and left motor respectively. A duty cycle of 100% will turn the motor at maximum speed, while a duty cycle of 0% will stop the motor.

Review PWM waveform generation on OC1A and OC1B using timer1 in the appendix, then implement the code below.

Your task is to apply the following sequence to the robot on power up:

1. Move the robot forward for two seconds at maximum speed then stop.
2. Move the robot backward for two seconds a quarter of the maximum speed then stop.

NB: Use a PWM frequency of 61 Hz.

6.3 Submission

Modify the code *robot.c* file with your solution and submit for evaluation.

APPENDIX

7 Non-PWM Waveform Generation on the ATMega32 Microcontroller

Square waves of different frequencies and duty cycles can be generated on pins OC0, OC2, OC1A and OC1B of the ATMega32 microcontroller. In general the OC (output compare) pins on the families of AVR microcontroller can be used for waveform generation.

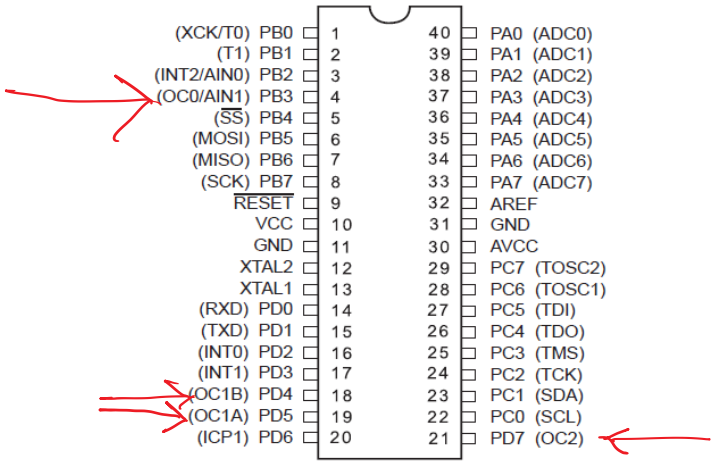


Figure A1: Pinout of the ATMega32 Microcontroller showing the waveform generation pins

7.1 Waveform Generation Using Timer 0 (Non PWM)

The configuration of waveform generation is done using the TCCR0 register. See the details in the ATmega32 datasheet. The CTC Timer/Counter mode and Toggle OC0 on compare match will be used to generate a 50% duty cycle wave. The frequency of the wave depends on Fosc, the prescaler and value in OCR0 register. Recall the CSxx bits in TCCR0 register determine the prescaler value.

7	6	5	4	3	2	1	0	
FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0

Figure A2: TCCR0 Register

Mode	WGM01 (CTC0)	WGM00 (PWM0)	Timer/Counter Mode of Operation
0	0	0	Normal
1	0	1	PWM, Phase Correct
2	1	0	CTC
3	1	1	Fast PWM

Figure A3: Waveform Generation Mode Bits

COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Toggle OC0 on compare match
1	0	Clear OC0 on compare match
1	1	Set OC0 on compare match

Figure A4: Compare Output Mode, Non-PWM Mode

NB: The CTC Timer/Counter mode and Toggle OC0 on compare match will be used to generate a 50% duty cycle wave. The frequency of the wave is determined by Fosc, the prescaler and value in OCR0 register.

Therefore TCCR0 = 0b00011XXX

OCR0 = 0bXXXXXXXX

The frequency of the square wave on OC0 is given by the following formula,

$$F_{OC0} = \frac{F_{osc}}{2N(OCR0 + 1)}$$

Where,

Fosc – Microcontroller oscillator frequency

N – Prescaler (1, 8, 64, 256, 1024)

Example:

Determine the values needed for TCCR0 and OCR0 to generate a 1 kHz wave on OC0 pin of the ATmega32 if it is clocked by a 4MHz crystal. Write a program to generate the square wave.

From the formula above, $OCR0 + 1 = \frac{F_{osc}}{2N * F_{oc0}}$ = 4M/(2*64*1k) = 31.25 ~ 31

Therefore OCR0 = 30, TCCR0 = 0b00011011

Program:

```
#include <avr/io.h>
int main()
{
        DDRB |= (1<< PB3); // Set PB0 (OC0) to an output pin
        OCR0 = 30;
        TCCR0 = 0b00011011;
        while(1);
        return 0;
}
```

8 PWM Waveform Generation on the ATMega32 Microcontroller

The duty cycle of a square wave is defined in the formula below:

$$Duty\ Cycle = \frac{T_{high}}{Period}$$

The figure below shows square waves of different duty cycles.

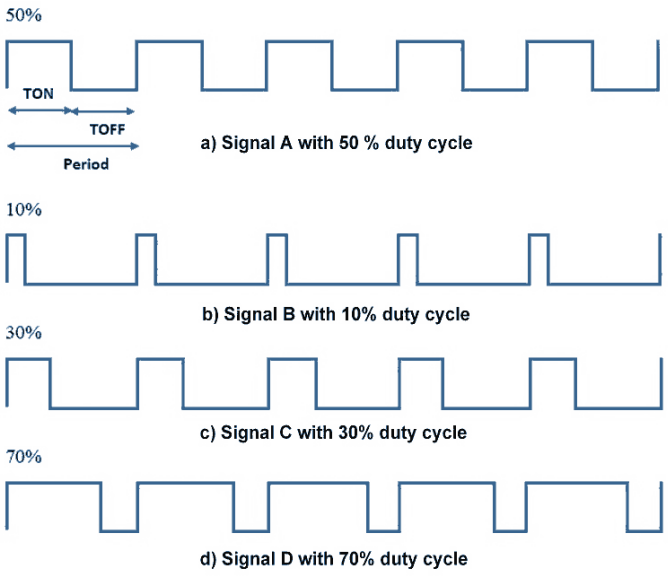


Figure A5: Square waves of Different Duty Cycle

Varying the duty cycle of a square wave actually varies the average voltage applied to the load.

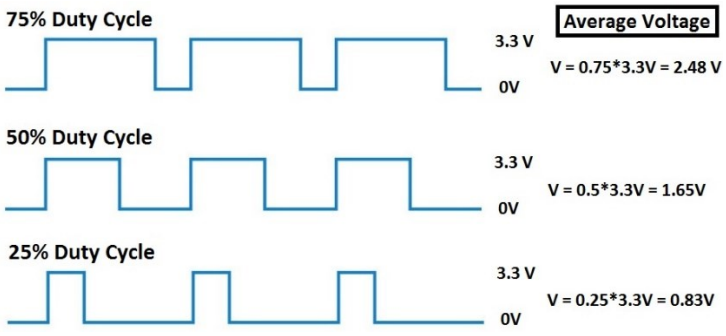


Figure A6: Duty Cycle of a Square Wave varies the Average Voltage to the Load

Varying the duty cycle of a square wave is called pulse with modulation (PWM). During the generation of the PWM wave its frequency remains fixed.

PWM waves can be generated on the OC0, OC2, OC1A and OC1B pins of the ATMega32 microcontroller using Timer0, Timer2 and Timer1 respectively. For this laboratory, we will focus on pins OC1A, OC1B and OC2.

8.1 PWM Generation on OC2 Using Timer2

The following registers are needed to configure the PWM on OC1A and OC1B (See Timer 1 in the AtMega32 datasheet):

1. TCCR2
2. OCR2

NB: Non-Inverted Phase correct PWM waves are used in this laboratory.

7	6	5	4	3	2	1	0
FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20

Figure A7: TCCR2 Register

Mode	WGM21 (CTC2)	WGM20 (PWM2)	Timer/Counter Mode of Operation
0	0	0	Normal
1	0	1	PWM, Phase Correct
2	1	0	CTC
3	1	1	Fast PWM

Figure A8: Waveform Generation Mode Bit Description

COM21	COM20	Description
0	0	Normal port operation, OC2 disconnected.
0	1	Reserved
1	0	Clear OC2 on compare match when up-counting. Set OC2 on compare match when downcounting.
1	1	Set OC2 on compare match when up-counting. Clear OC2 on compare match when downcounting.

Figure A9: Phase Correct PWM Mode

Frequency of generated PWM wave on OC2, $F_{wave} = \frac{F_{osc}}{N \times 510}$

The Duty Cycle (non-Inverted) of the generated wave on OC2, $Duty\ Cycle = \frac{OCR2}{255} \times 100$

Example

Write an AVR C program for the ATMega32 that generates a 75% duty cycle wave with a frequency of 1k Hz on pin OC2. Assume the microcontroller is clocked by an 8 MHz crystal.

The prescaler needed for the 120 Hz signal is, N = Fosc/(Fwave x 510) = 8M/(1k x 510) = 7.8 ~ 8

From figures A7, A8 and A9, and N = 8; TCCR2 = 0b01100010

$Duty\ Cycle = \frac{OCR2}{255} \times 100$, therefore; 75 = OCR2*100/255; => OCR2 = 191.25 ~ 191

Program

```
#include <avr/io.h>
int main()
{
        DDRD |= (1<< PD7); // Set PD7(OC2) to an output pin
        OCR2 = 191;
        TCCR2 = 0b01100010;

        while(1);
        return 0;
}
```

8.2 PWM Generation on OC1A and OC1B Using Timer1

The following registers are needed to configure the PWM on OC1A and OC1B (See Timer 1 in the AtMega32 datasheet):

1. TCCR1A
2. TCCR1B
3. OCR1A
4. OCR1B

The ATmega32 supports Fast PWM and Phase correct PWM, for the laboratory phase correct PWM will be used.

7	6	5	4	3	2	1	0	
COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10	TCCR1A

Figure A10: TCCR1A Register

7	6	5	4	3	2	1	0	
ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B

Figure A11: TCCR1A Register

COM1A1/COM1B1	COM1A0/COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	WGM13:0 = 9 or 14: Toggle OC1A on Compare Match, OC1B disconnected (normal port operation). For all other WGM13:0 settings, normal port operation, OC1A/OC1B disconnected.
1	0	Clear OC1A/OC1B on compare match when up-counting. Set OC1A/OC1B on compare match when downcounting.
1	1	Set OC1A/OC1B on compare match when up-counting. Clear OC1A/OC1B on compare match when downcounting.

Figure A12: Phase Correct PWM Mode

Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation	TOP	Update of OCR1x	TOV1 Flag 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
13	1	1	0	1	Reserved	–	–	–
14	1	1	1	0	Fast PWM	ICR1	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCR1A	BOTTOM	TOP

Figure A13: Timer 1 Modes/Waveform Generation Mode Bit Description

Frequency of generated PWM wave on OC1A or OC1B, $F_{wave} = \frac{F_{osc}}{2 \times N \times Top}$

The Duty Cycle of the generated wave on OC1A or OC1B, $Duty\ Cycle = \frac{Top - OCR1A}{Top} \times 100$

Therefore, if Mode 7 is used, Top = 10 bits, which has a value of 0x3FF = 1024

Example:
Assume XTAL = 8MHz. Using mode 10 write an AVR C program that generates waves with duty cycles of 30% and 60% on the OC1A and OC1B pins, respectively. The frequency of the generated waves should be 125 Hz.

From the equation above, Top = Fosc/(2 x N x Fwave) = 8M /(2 x 64 x 125) = 500. NB: N = 64

Therefore ICR1 = 500

$$Duty\ Cycle = \frac{Top - OCR1A}{Top} \times 100$$

For 30 % Duty cycle, 30 = (500 – OCR1A)*100/500
Therefore OCR1A = 500 - 30*500/100 = 500 – 150 = 350

For 60 % Duty cycle, 60 = (500 – OCR1B)*100/500
Therefore OCR1B = 500 - 60*500/100 = 500 – 300 = 200

TCCR1A = 0b10100010 //Non-inverted PWM, Mode 10
TCCR1B = 0b00010011 //Mode 10 and prescaler = 64

Program:

```
#include <avr/io.h>
int main()
{
        DDRD |= (1<< PD4); // Set PD4(OC1A) to an output pin
        DDRD |= (1<< PD5); // Set PD5(OC1B) to an output pin
        OCR1A = 350;
        OCR1B = 200;
        TCCR1A = 0b10100010;
        TCCR1B = 0b00010011;
        ICR1 = 500;

        while(1);
        return 0;
}
```