**Name: LƯƠNG THÀNH VỸ (2151280)**

**TRẦN TRUNG TOÀN (2151265)**

# BJECTIVE:

* Understanding the operating modes of a timer:
* Understanding how to use a timer for creating delays and generating pulses.

# REFERENCES:

* Lab manual Chapter 4, 5
* Atmel-2505-Setup-and-Use-of-AVR-Timers\_ApplicationNote\_AVR130.pdf

# EXPERIMENT 1:

1. Write a delay subroutine of 1 ms using Timer 0. Use this subroutine to generate a 500 Hz pulse on pin PA0.
2. Simulate and modify the program to achieve accurate pulse generation.
3. Connect pin PA0 to an oscilloscope to verify.

# EXPERIMENT 2:

1. Write a program to generate a 64 us square wave using Timer 0 in Normal mode. Use pin OC0 as the output.
2. Write a program to generate a 64 us square wave using Timer 1 in CTC mode. Use pin OC0 as the output.
3. Connect pin OC0 to an oscilloscope and observe.

# EXPERIMENT 3:

1. Given the program for generating two PWM pulses on OC0A and OC0B, connect pins OC0A and OC0B to two oscilloscope channels, measure and record the waveform, and explain the obtained waveform.

|  |
| --- |
| .org 00  call initTimer0  start:  rjmp start  initTimer0:  // Set OC0A (PB3) and OC0B (PB4) pins as outputs  ldi r16, (1 << PB3) | (1 << PB4);  out DDRB,r16  ldi r16, (1 << COM0B1)|(1 << COM0A1) | (1 << WGM00)|(1 << WGM01)  out TCCR0A,r16 // setup TCCR0A  ldi r16, (1 << CS01)  out TCCR0B,r16 // setup TCCR0B  ldi r16, 100  out OCR0A,r16 //OCRA = 100  ldi r16, 75  out OCR0B,r16 //OCRB = 75  ret |

# EXPERIMENT 4:

1. Modify the program for different combinations of TCCR0A and TCCR0B registers as described in the table:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | TCCR0A | | | | | | | | TCCR0B | | | | | | | |
|  | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 1 | COM0A1 | COM0A0 | COM0B1 | COM0B0 |  |  | WGM01 | WGM00 | FOC0A | FOC0B |  |  | WGM02 | CS02 | CS01 | CS00 |
| 2 | 1 | 0 | 1 | 0 |  |  | 1 | 1 |  |  |  |  | 0 | 0 | 1 | 0 |
| 3 | 1 | 0 | 1 | 0 |  |  | 1 | 1 |  |  |  |  | 1 | 0 | 1 | 0 |
| 4 | 1 | 0 | 1 | 0 |  |  | 0 | 1 |  |  |  |  | 0 | 0 | 1 | 0 |

1. Connect pins OC0A and OC0B to two oscilloscope channels, measure and record the waveforms, and explain the obtained results.

# EXPERIMENT 5:

1. Write a program to generate a 1 kHz square wave with a duty cycle of 25% on pin OC0B.
2. Connect to an oscilloscope and measure the output waveform.
3. Connect OC0B to the R channel of an RGB LED. Write a program to increase the duty cycle on OC0B from 0% to 100% and then decrease it back to 0% over 10 ms, with a 1% increment.

# EXPERIMENT 1:

1. Answer the following questions:
   1. What is the maximum delay achievable using Timer 0 with an 8 MHz frequency? Explain the calculation.

**ANSWER:**

Timer 0 is an 8-bit timer, meaning it can count up to 255. With the highest prescaler setting of 1024, the maximum number of cycles the timer can achieve before overflowing is

At an 8 MHz clock speed, each cycle lasts 0.125 microseconds. Consequently, the timer's maximum delay is calculated as:

* 1. What is the maximum delay achievable using Timer 1 with an 8 MHz frequency? Explain the calculation.

**ANSWER:**

Timer 1 is an 16-bit timer, meaning it can count up to 65535. With the highest prescaler setting of 1024, the maximum number of cycles the timer can achieve before overflowing is

At an 8 MHz clock speed, each cycle lasts 0.125 microseconds. Consequently, the timer's maximum delay is calculated as:

* 1. Explain how to calculate the prescaler values and the values loaded into Timer0 registers for this experiment.

**ANSWER:**

To calculate the prescaler and Timer0 register values, first determine the delay needed in terms of clock cycles, noting that the maximum allowable delay is 261,120 cycles (or 32.64 ms). Divide this cycle count by available prescaler options (1, 8, 64, 256, 1024) to find a value within Timer0's 8-bit range of 0 to 255. The prescaler that brings the result into this range is selected, and the result of the division above is the the number of counts Timer0 must achieve. This count can then be loaded into the Output Compare Register for CTC mode, or subtracted from 255 to initialize the Timer/Counter Register (TCNT) at overflow in Normal mode.

* 1. Source code for the program with comments.

**ANSWER:**

|  |
| --- |
| ; Internal clk = 8 MHz  ; f = 500 Hz ==> T = 2 ms (Period between each pulse)  ; 2 ms = 16 000 cycles => Prescale = 64 , OCR\_val = 249 +22    .EQU OCR0\_val = 249 ; 0 to 255  .EQU TCCR0B\_mode = 0b00000011  start:  SBI DDRA, 0 ; Set PA0 as Output  CBI PORTA, 0 ; Clear PA0    PULSE\_500Hz: SBI PORTA, 0  CBI PORTA, 0  RCALL TIMER0\_DELAY  RJMP PULSE\_500Hz  TIMER0\_DELAY: ; (+3) For CALL to here  LDI R16, 0 ; (+1) Clear Timer  OUT TCNT0, R16 ; (+1)  LDI R16, OCR0\_val ; (+1) Set OCR  OUT OCR0A, R16 ; (+1)  LDI R16, 0B00000010 ; (+1) Choose CTC mode  OUT TCCR0A, R16 ; (+1)  LDI R16, TCCR0B\_mode ; (+1) Choose mode and start Timer  OUT TCCR0B, R16 ; (+1)  AGAIN: SBIS TIFR0, OCF0A ; } (Number of cycles = Prescale \* OCR)  RJMP AGAIN ; }  ; (+2) when Skipped  LDI R16, 0 ; (+1) Stop Timer  OUT TCCR0B, R16 ; (+1)  LDI R16, (1 << OCF0A) ; (+1) Clear TOV flag  OUT TIFR0, R16 ; (+1)  RET ; (+4)  ; ===> Total of cycles = Prescale \* (OCR + 1) + 21 |

**SIMULATTION RESULT:**

A computer screen shot of a computer

Description automatically generated

# EXPERIMENT 2:

1. Answer the following questions:
   1. In Normal mode, when is the TOVx bit set to 1?

**ANSWER:**

In Normal mode, the TOVx bit is set to 1 after the timer reaches its maximum value and rolls over to 0. For an 8-bit timer, this maximum value is 0xFF (255), and for a 16-bit timer, it is 0xFFFF (65535). The TOVx bit is set after the timer reaches these values and increments to roll over (when the Timer increments past its maximum value, not upon reaching the maximum value).

* 1. In CTC mode, when is the OCFx bit set to 1?

**ANSWER:**

In CTC mode, the OCFx bit is set to 1 when the Timer matches the value in the Output Compare Register (OCRx), OCFx is set upon reaching the OCRx value. After the OCRx is each the Timer reset to 0 and OCFx flag will not be cleared automatically.

* 1. Provide the register configurations for Timer 0 for both cases.

**ANSWER:**

A screenshot of a video game

Description automatically generated

A close-up of a list

Description automatically generated

To select Normal mode for Timer 0, set the Waveform Generation Mode (WGM) bits (WGM02, WGM01, and WGM00) to 000. This is achieved by clearing the TCCR0A register entirely (no need to configure COM0A or COM0B) and clearing bit 3 of the TCCR0B register.

For CTC mode, set the WGM bits to 010. This requires setting the second bit of TCCR0A and clearing bit 3 of TCCR0B.

* 1. Source code for the programs in both cases.

**ANSWER:**

* **64us delay using Timer 0 in Normal mode:**

**The Period of the square wave is 64 us, we need 32 us delay:**

|  |
| --- |
| ; Internal clk = 8 MHz ==> 1 cycle = 0.125 us  ; Square wave with T = 64 us and f = 15625  ; ==> We need 32 us delay = 256 cycles - 2 cycles (because of SBI and CBI delay)  .EQU TCCR0B\_mode = 0b00000001  .EQU TCNT\_init = 23 ; 0 to 255  TIMER0\_DELAY: ; (+3) For CALL to here  LDI R16, TCNT\_init ; (+1) Init Timer  OUT TCNT0, R16 ; (+1)  LDI R16, 0b00000000 ; (+1) Choose Normal mode  OUT TCCR0A, R16 ; (+1)  LDI R16, TCCR0B\_mode ; (+1) Choose mode and start Timer  OUT TCCR0B, R16 ; (+1)  AGAIN: SBIS TIFR0, TOV0 ; } #Cycles = Prescale \* (256 - TCNT\_init)  RJMP AGAIN ; }  ; (+4) In the last iteration Skipped  LDI R16, 0 ; (+1) Stop Timer  OUT TCCR0B, R16 ; (+1)  LDI R16, (1 << TOV0) ; (+1) Clear TOV flag  OUT TIFR0, R16 ; (+1)  RET ; (+4)  ; ===> Total of cycles = Prescale \* (256 - TCNT\_init) + 21 |

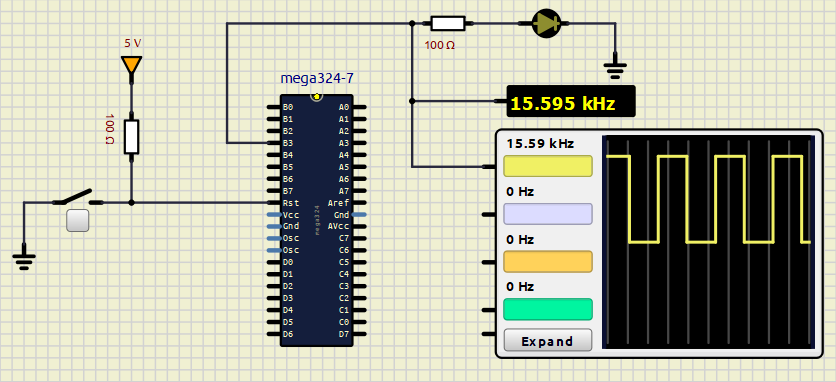
**SIMULATION RESULT:**

The result of 32us delay (half period) of the Delay Subroutine:

A screenshot of a computer program

Description automatically generated

The square wave on Oscilloscope:



For period of 64 us, The frequency is Hz. The result is closely the calculated frequency.

* **64 us delay using Timer 1 in CTC mode:**

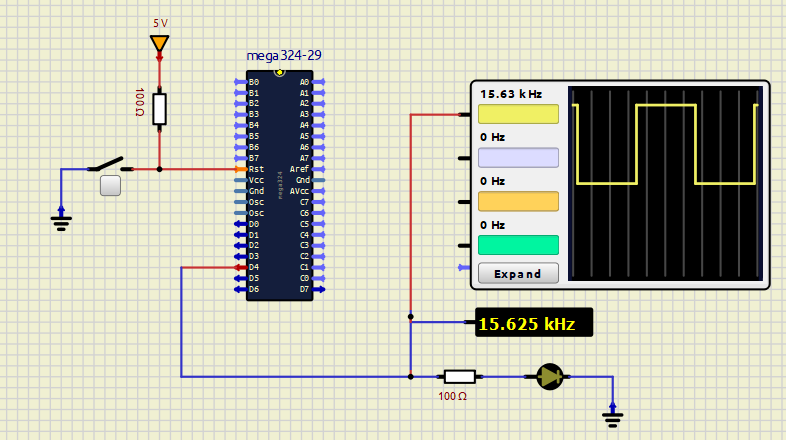
|  |
| --- |
| ; Internal clk = 8 MHz  .EQU OCR1\_val = 220 ; 1 to 65535 -2 cycles due to RJMP LOOP  .EQU TCCR1B\_mode = 0b00001001  TIMER1\_DELAY: ; (+3) For CALL to here  LDI R16, 0 ; (+1) Clear Timer  STS TCNT1H, R16 ; (+2)  STS TCNT1L, R16 ; (+2)  LDI R16, HIGH(OCR1\_val) ; (+1) Set OCR  STS OCR1AH, R16 ; (+2)  LDI R16, LOW(OCR1\_val) ; (+1)  STS OCR1AL, R16 ; (+2)  LDI R16, 0b01000000 ; (+1) Enable OC1A in toggle mode  STS TCCR1A, R16 ; (+2)  LDI R16, TCCR1B\_mode ; (+1) Choose CTC mode  STS TCCR1B, R16 ; (+2) At this point: 20 cycles  AGAIN: SBIS TIFR1, OCF1A ; } (Number of cycles = Prescale \* OCR)  RJMP AGAIN ; }  ; (+4) Last Skip  LDI R16, 0 ; (+1) Stop Timer  STS TCCR1B, R16 ; (+2)  LDI R16, (1 << OCF1A) ; (+1) Clear TOV flag  OUT TIFR1, R16 ; (+1)  RET ; (+4)  ; ===> Total of cycles = Prescale \* (OCR + 1) + 33 |

The result of 32us delay (half period) of the Delay Subroutine:

A screenshot of a computer

Description automatically generated

The square wave on Oscilloscope:



For period of 64 us, The frequency is Hz. The result is closely the calculated frequency.

# EXPERIMENT 3:

1. Answer the following questions:
   1. Describe the waveform on the oscilloscope (capture and insert it).

**ANSWER:**

**Waveform captured on Oscilloscope:**

A computer screen shot of a computer

Description automatically generated

Both waveforms are PWM signals with a frequency of 3.906 kHz, but they have different duty cycles. The PWM signal on the OC0B pin has a smaller duty cycle than the signal on the OC0A pin.

* 1. Explain the reasons for the observed waveform (frequency, duty cycle, phase).

**ANWSER:**

The configuration (1 << WGM00) | (1 << WGM01) sets Timer 0 to Fast PWM mode, where the counter increments up to 0xFF and then resets to 0. The bits (1 << COM0A1) and (1 << COM0B1) set OC0A and OC0B to "Clear on Compare Match, set at BOTTOM, meaning OC0A will toggle between high and low, creating a PWM signal where it is high until the counter matches OCR0A (100), then goes low until the counter resets. OC0B similarly toggles based on OCR0B (75).

Given the 8-bit resolution (0-255), setting OCR0A to 100 creates a duty cycle of approximately Similarly, setting OCR0B to 75 creates a duty cycle of approximately

The prescaler (1 << CS01) is set to 8, which means the PWM frequency is the system clock (8 MHz) divided by

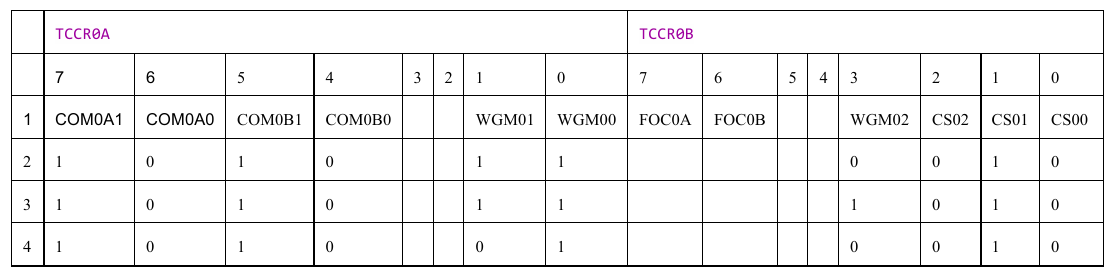
Which is the matched the result observed on Oscilloscope.

# EXPERIMENT 4:

Answer the following questions:

* 1. Identify the working modes of Timer 0 corresponding to the values in the table.

**ANSWER:**

****

COM bits are the same in these 3 modes, it enables the OC0A and OC0B pin , and toggle these pin on compare match.

**Mode 1: WGM02 = 0, WGM01 = 1, WGM00 = 1**

Timer 0 operates in Fast PWM mode, where the timer counts from 0 to 255 and then resets, producing a high-speed PWM waveform.

**Mode 2:** **WGM02 = 1, WGM01 = 1, WGM00 = 1**

The timer 0 operates in Fast PWM mode with OCRA as the TOP value, meaning the counter resets when it reaches the value stored in OCR0A, allowing adjustable frequency control.

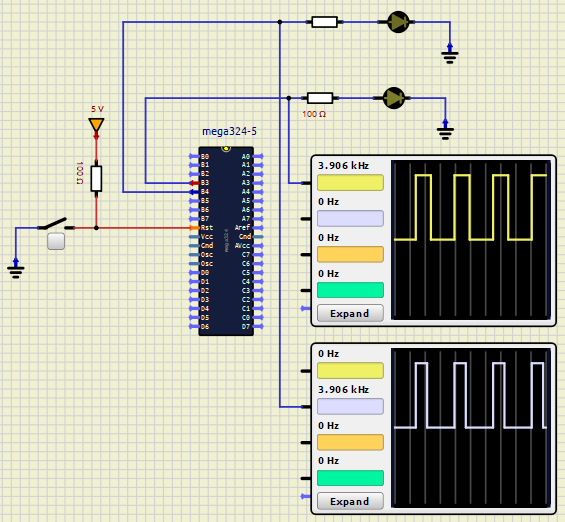
**Mode 3: WGM02 = 0, WGM01 = 1, WGM00 = 0**:

The timer 0 operates in CTC mode, in which the timer resets upon reaching the OCR0A value, generating precise time intervals.

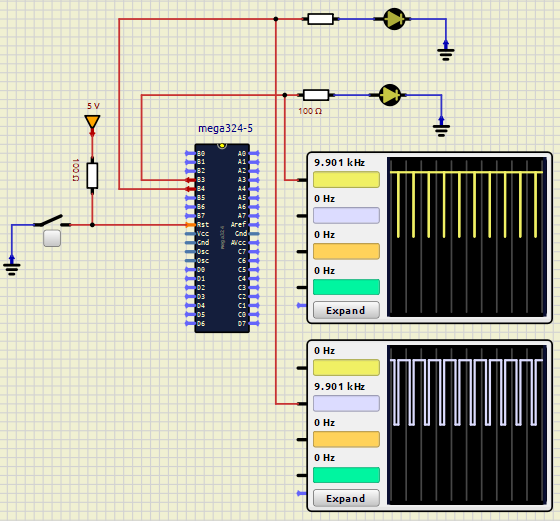
* 1. Capture images of the waveforms corresponding to the different working modes and explain the results.

|  |
| --- |
| .EQU TCCR0A\_org\_ex3 = 0b10100011 ; Same as table 1  .EQU TCCR0B\_org\_ex3 = 0b00000010  .EQU TCCR0A\_table\_1 = 0b10100011 ; {COM0A1, COM0A0} = {COM0B1, COM0B0} = 2'b10  .EQU TCCR0B\_table\_1 = 0b00000010 ; {WGM02, WGM01, WGM 00} = 3'b011  ; {CS02, CS01, CS00} = 3'b010  .EQU TCCR0A\_table\_2 = 0b10100011 ; {COM0A1, COM0A0} = {COM0B1, COM0B0} = 2'b10  .EQU TCCR0B\_table\_2 = 0b00001010 ; {WGM02, WGM01, WGM 00} = 3'b111  ; {CS02, CS01, CS00} = 3'b010  .EQU TCCR0A\_table\_3 = 0b10100001 ; {COM0A1, COM0A0} = {COM0B1, COM0B0} = 2'b10  .EQU TCCR0B\_table\_3 = 0b00000010 ; {WGM02, WGM01, WGM 00} = 3'b001  ; {CS02, CS01, CS00} = 3'b010  .ORG 00  MAIN: CALL INITTIMER0  HALT: RJMP HALT  INITTIMER0:  ; SET OC0A (PB3) AND OC0B (PB4) PINS AS OUTPUTS  LDI R16, (1 << PB3) | (1 << PB4)  OUT DDRB, R16  LDI R16, TCCR0A\_table\_3 ; Adjust for to select Mode in the tables  LDI R17, TCCR0B\_table\_3  OUT TCCR0A, R16 ; SETUP TCCR0A  OUT TCCR0B, R17 ; SETUP TCCR0B  LDI R16, 100  OUT OCR0A, R16 ; OCRA = 100  LDI R16, 75  OUT OCR0B, R16 ; OCRB = 75  RET |

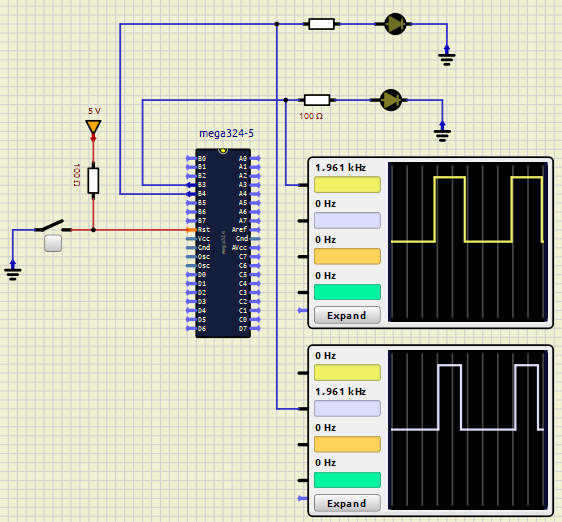
**Waveform of Mode 1:**



**Waveform of Mode 2:**

****

**Waveform of Mode 3:**



# EXPERIMENT 5:

1. Answer the following questions:
   1. In which mode is Timer 0 operating?

**ANSWER:**

In this program, Timer 0 is set to CTC mode, which controls the delay or frequency at which the signal on the OC0B pin toggles, creating a PWM waveform with a specified frequency and duty cycle.

* 1. What values are loaded into Timer 0 registers, and why?

**ANSWER:**

The value loaded into OCR0B for Timer 0 is 15, and a prescaler of 64 is selected. This setup is chosen because, in CTC mode, OCR0A defines the frequency by setting the TOP count, while OCR0B determines the toggle point on the OC0B pin, establishing the duty cycle. With these values, the timer generates a PWM signal with a specific frequency controlled by the OCR0A value and a 25% duty cycle determined by the OCR0B value of 15.

1. Present the source code with comments.

|  |
| --- |
| ; Internal clk = 8 MHz ==> 1 cycle = 0.125 us  ; f = 1 KHz ==> T = 1 ms = 8000 cycles  .EQU TCCR0A\_value = 0b00100001 ; {COM0B1, COM0B0} = 2'b10  .EQU TCCR0B\_value = 0b00001011 ; {WGM02, WGM01, WGM 00} = 3'b101, Prescale = 8  ; {CS02, CS01, CS00} = 3'b011  ; Duty cycle = 100 \* (256 - OCR)/256  .EQU OCR0A\_value = 62  .EQU OCR0B\_value = 15  .ORG 00  MAIN: CALL INIT\_TIMER0  HALT: RJMP HALT  INIT\_TIMER0:  LDI R16, (1 << PB4) ; SET OC0B (PB4) PINS AS OUTPUTS  OUT DDRB, R16  LDI R16, TCCR0A\_value  LDI R17, TCCR0B\_value  OUT TCCR0A, R16 ; SETUP TCCR0A  OUT TCCR0B, R17 ; SETUP TCCR0B  LDI R16, OCR0A\_value ; duty cycle = OCR0A\_value/256  OUT OCR0A, R16  ; frequency of OC0B = OCR0B \* Prescale = 15 \* 64 = 960 = 1 KHz  LDI R16, OCR0B\_value  OUT OCR0B, R16  RET |

**SIMULATION RESULT:**

A computer screen shot of a computer

Description automatically generated

**Comment:**

* The frequency is set accurately at 1 kHz, meeting the specified requirement.
* Observing the PWM on the oscilloscope, it remains HIGH for one time division and LOW for three time divisions, confirming that it is operating with a 25% duty cycle.