Class:	CPE301L Digital Systems Architecture and Design 1001		Semester:	Spring 2025
Points		Document author:	Narek Kalikian	
		Author's email:	kalikn1@unlv.nevada.edu	
		Document topic:	Postlab 4	
Instructor's comments:				

1. Introduction / Theory of Operation

In this lab, we will be tasked with understanding and using the AVR microcontroller's internal timers. The Atmega328P microcontroller board that we've been using has three 8-bit and one 16-bit timer. Each of these timers has its own unique set of registers and pins to be used in different modes of operation. We will need to learn about the different modes for each of the timers, what they do, and when to use them. This is important for any embedded system application that requires timing.

2. Prelab Content

1. Accessing the TOV0 Flag: This flag is used to show overflow in timer0/counter0. When overflow occurs in timer0/counter0, the TOV0 flag is set high (1) to indicate this. To access the TOV0 flag, you must look in the TIFR (Timer/Counter Interrupt Flag Register), this is where the TOV0 overflow flag will be located. To clear the TOV0 flag, you must write a logic high (1) to the flag.

2. Normal Mode Assembly Code:

Clock Prescaler = 1024 → Time Period = 1.024 milliseconds

1 Second Delay = 1000 milliseconds → 1000 ms/1.024 ms = 976.56

Timer1 is a 16-bit timer → $2^{15}+2^{14}+2^{13}+2^{12}\ldots+2^{0}$ = 65535, overflow occurs at 65536

Overflow Range - Delay Count → 65536 - 976.56 ≈ 64559

So, 64559 needs to be loaded into Timer1. 64559 in hexadecimal is FC2F

Changes to Code:

```
LDI R20, 0xFC // Load R20 register with the upper byte (8 bits)
STS TCNT1H, R20 // Store the upper 8 bits in TCNT1H, H for high byte
LDI R20, 0x2F // Load R20 register with the lower byte (8 bits)
STS TCNT1L, R20 // Store the lower 8 bits in TCNT1L, L for lower byte
```

3. Assembly Instruction:

- LDI - Load Immediate: Immediately load a value into a register
- OUT - Store Register to I/O Location: Stores data from source register (Rr) in the register file to I/O space (enables outputs)
- STS - Store Direct to Data Space: Stores one byte from a register directly to data space
- RCALL - Relative Call: Calls to an address within PC - 2K + 1 and PC + 2K words
- EOR - Exclusive OR: Performs logical XOR operation between the contents of the destination register (Rd) and source register (Rr) and places the result in Rd
- RJMP - Relative Jump: Uses relative offsets to jump to an address within PC - 2K + 1 and PC + 2K words
- LDS - Load Direct from Data Space: Loads one byte from data space (such as 8 bits of counter) to a register
- CPI - Compare with Immediate: Performs a compare between destination register (Rd) and a constant
- BRSH - Branch if Same or Higher: Unsigned conditional relative branch. Compares the values of 2 registers by checking the Carry flag and branches relatively to PC if C is set
- BRLT - Branch if Less Than: Signed conditional relative branch. Compares the values of 2 registers by checking the Sign flag and branches relatively to the PC if S is set
- RET - Return from Subroutine: Returns from the subroutine in the code, return address is loaded from the stack

3. Description of Experiments

Experiment 1 - C Code w/ Comments:

Lab4_Exp1 main.cpp ASF Wizard

main.cpp C:\Users\kalikn1\Documents\Atmel Studio\7.0\Lab4_Exp1\Lab4_Exp1\main.cpp

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#define F_CPU 16000000;

ISR(TIMER1_COMPA_vect) {
    cli();
    PORTB ^= (1 << PINB0); // EORs PINB0 to blink them
    sei();
}

int main(void)
{
    DDRB |= (1 << PINB0); // PB0 is an output
    // Set the Timer Mode to CTC
    TCCR1A |= (0 << WGM11) | (0 << WGM10); // Sets WGM1[1:0]
    // Sets CS1 to 0b101 -> prescalar 1024; finish WGM1 to be 0b0100 for CTC
    TCCR1B |= (0 << WGM13) | (1 << WGM12) | (1 << CS02) | (0 << CS01) | (1 << CS00);
    TIMSK1 |= (1 << OCIE1A); // sets bit to enable Comparator A
    // Set the value that you want to count to
    OCR1A = 15624; //0x7A11 for 2 sec delays
    // initialize counter
    TCNT1 = 0;
    sei();
    while(1);
}
```

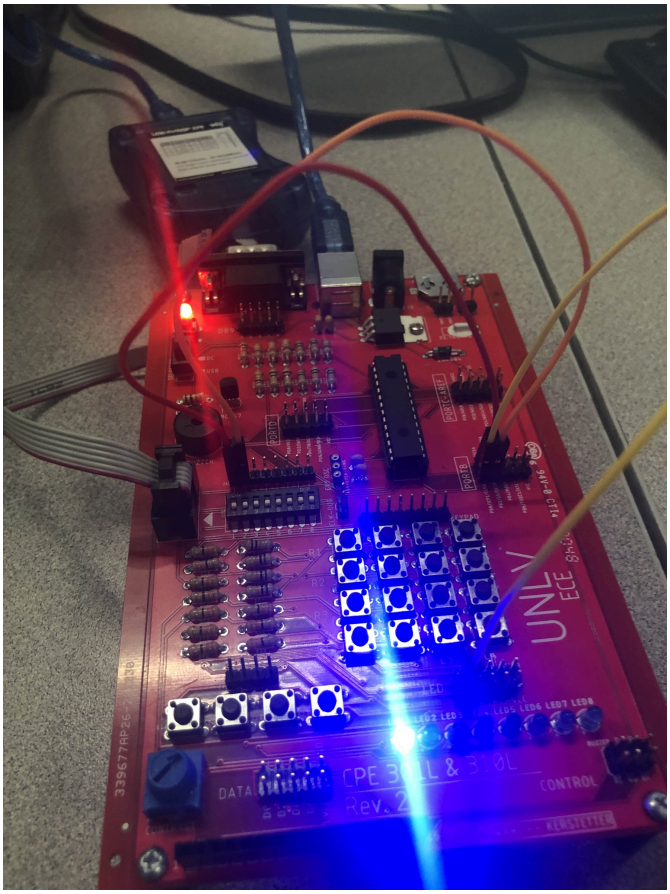
Output

Show output from: Build

```
    Done executing task "RunCompilerTask".
    Task "RunOutputFileVerifyTask"
                Program Memory Usage     :   230 bytes   0.7 % Full
                Data Memory Usage        :   0 bytes   0.0 % Full
                Warning: Memory Usage estimation may not be accurate if there are sections other than .text sections in ELF file
    Done executing task "RunOutputFileVerifyTask".
Done building target "CoreBuild" in project "Lab4_Exp1.cppproj".
Target "PostBuildEvent" skipped, due to false condition; ('$(PostBuildEvent)' != '') was evaluated as ('' != '').
Target "Build" in file "C:\Program Files (x86)\Atmel\Studio\7.0\Vs\Avr.common.targets" from project "C:\Users\kalikn1\Documents\Atmel Studio\7.0\Lab4_Exp1\Lab4_Exp1\Lab4_Exp1.cppproj" (entry point):
Done building target "Build" in project "Lab4_Exp1.cppproj".
Done building project "Lab4_Exp1.cppproj".

Build succeeded.
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
```

Experiment 1 - Board Photo (Functionality Shown During Lab):



Experiment 2 - C Code w/ Comments:

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#define F_CPU 16000000;

ISR(TIMER1_COMPA_vect) {
    cli();
    PORTB ^= (1 << PINB0); // blink LED pin
    int dips = PINB & ((1 << PINB2) | (1 << PINB1)); // read dipswitches
    switch (dips) { // change delay based on dipswitches input
        case 0b00: OCR1A = 0; // 0 sec delay
        break;
        case 0b01: OCR1A = 7812; // 1 sec delay
        break;
        case 0b10: OCR1A = 15624; // 2 sec delay
        break;
        case 0b11: OCR1A = 31249; // 4 sec delay
        break;
        sei();
    }
}
int main(void)
{
    DDRB = (1 << PINB0); // sets pinb0 to output only
    TCCR1A |= (0 << WGM11) | (0 << WGM10); // set bits WGM1[1:0] for CTC
    TCCR1B |= (0 << WGM13) | (1 << WGM12) | (1 << CS02) | (0 << CS01) | ( 1 << CS00); // set WGM[3:2] for CTC & CS[2:0] for prescaler = 1024
    TIMSK1 |= (1 << OCIE1A); // sets bit to enable CTC comparator A
    TCNT1 = 0; // initialize counter to 0
    OCR1A = 0;
    sei();
    while(1);
}
```

```
Output
Show output from: Build
    Done executing task "RunCompilerTask".
    Task "RunOutputFileVerifyTask"
                Program Memory Usage     :   292 bytes   0.9 % Full
                Data Memory Usage        :   0 bytes   0.0 % Full
                Warning: Memory Usage estimation may not be accurate if there are sections other than .text sections in ELF file
    Done executing task "RunOutputFileVerifyTask".
Done building target "CoreBuild" in project "Lab4_Exp1.cppproj".
Target "PostBuildEvent" skipped, due to false condition; ('$(PostBuildEvent)' != '') was evaluated as ('' != '').
Target "Build" in file "C:\Program Files (x86)\Atmel\Studio\7.0\Vs\Avr.common.targets" from project "C:\Users\kalikn1\Documents\Atmel Studio\7.0\Lab4_Exp1\Lab4_Exp1\Lab4_Exp1.cppproj" (entry point):
Done building target "Build" in project "Lab4_Exp1.cppproj".
Done building project "Lab4_Exp1.cppproj".

Build succeeded.
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
```

Experiment 2 - Assembly Code w/ Comments:

Lab4_Exp2_ASM | main.asm* | ASF Wizard

```
.org 0x0
    jmp setup

setup:
    ldi r23, 0x02 // register used for OCF1A flag
    ldi r16, 0x00 // 8-bits of zeros
    ldi r29, 0x00 // register used for dipswitch inputs
    ldi r17, 0x01 // used for setting PIN0
    ldi r18, 0x01 // Sets PIN0 for LED output
    out DDRB, r17 // Sets PINB0 to output
    ldi r20, 0 << WGM10 // sets WGM1[1:0] for ctc
    ldi r19, 5
    ori r19, 1 << WGM12 // sets WGM1[3:2] for ctc
    sts TCCR1A, r20 // sets CTC bits
    sts TCCR1B, r19 // Sets prescalar = 1024 and CTC bits
    sts TIMSK1, r23 // sets OCIE1A bit to enable CTC comparator A
    // sets the high and low bits of OCR1A for CTC mode
    ldi r22, 0x00 // default 0 sec delay
    ldi r21, 0x00 // default 0 sec delay
main:
    sts OCR1AH, r22 // high byte
    sts OCR1AL, r21 // low byte
    // read input from dipswitches
    in r29, PINB
    andi r29, 0b00000110 // masks for PINB1 and PINB2 (Dip1 and Dip2)
    // switch case statement for dipswitches
    cpi r29,  0b00000000 // mode 0: no blinking
    breq MODE0
    cpi r29,  0b00000010 // mode 1: 1 second blinking
    breq MODE1
    cpi r29,  0b00000100 // mode 2: 2 second blinking
    breq MODE2
    cpi r29,  0b00000110 // mode 3: 4 second blinking
    breq MODE3
continue:
    rcall blink
    rjmp main

blink:
    rcall delay // calling timer to wait for n seconds
    eor r18, r17 // XOR to toggle led
```

Lab4_Exp2_ASM | main.asm | ASF Wizard

```
    eor r18, r17 // XOR to toggle led
    out PORTB, r18
    ret

delay:
    in r28, TIFR1 // load value of TIFR1
    and r28, r23 // mask all bits except OCF1A
    cp r28, r23 // check if OCF1A flag is set
    brne delay
    out TIFR1, r23 // reset OCF1A flag
    ret

mode0:
    // 0000 0000 0000 0000
    // 0 sec delay
    ldi r22, 0x00 // high byte
    ldi r21, 0x00 // low byte
    rjmp continue

mode1:
    // 0001 1110 1000 0100
    // 1 sec delay
    ldi r22, 0x1E // high byte
    ldi r21, 0x84 // low byte
    rjmp continue

mode2:
    // 0011 1101 0000 1000
    // 2 sec delay
    ldi r22, 0x3D // high byte
    ldi r21, 0x08 // low byte
    rjmp continue

mode3:
    // 0111 1010 0001 0001
    // 4 sec delay
    ldi r22, 0x7A // high byte
    ldi r21, 0x11 // low byte
    rjmp continue
```

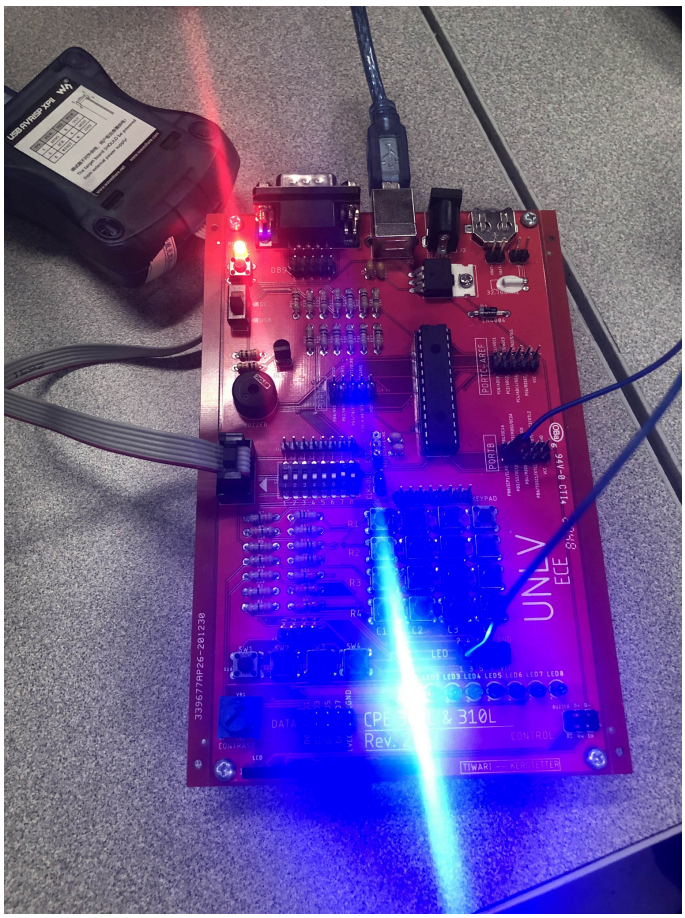
Output

Show output from: Build

```
Done building target "Build" in project "Lab4_Exp2_ASM.asmproj".
Done building project "Lab4_Exp2_ASM.asmproj".

Build succeeded.
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
```

Experiment 2 - Board Photo (Functionality Shown During Lab):



4. Questions and Answers

1. Watchdog Timer: A watchdog timer is a timer used for monitoring programs running on a microcontroller. It is used to detect and recover from events such as the program malfunctioning, getting out of control, or just generally stopping its operation. They can also be used to monitor and limit software execution time on different hardware devices. For example, a watchdog timer can be used when trying to upload/run/use untrusted code to prevent types of denial-of-service attacks.

2. Minimum and Maximum Times Before Overflow (1 MHz Clock):

- Prescalers: 1, 8, 64, 256, 1024
- Timer 0 Resolution: 2^8 = 256
- Timer 1 Resolution: 2^16 = 65536
- Clock Frequency: 1 MHz
- Period: 1 µS

Overflow = ((Prescaler)*(Resolution))/(Frequency)

- Timer 0, prescaler = 0 → ((1)*(256))/(1 * 10^6)) = 2.56 * 10^-4 seconds
- Timer 1, prescaler = 0 → ((1)*(65536))/(1 * 10^6)) = 6.55 * 10^-2 seconds
- Timer 0, prescaler = 1024 → ((1024)*(256))/(1 * 10^6)) = 0.260 seconds

- Timer 1, prescaler = 1024 → ((1024)*(65536))/(1 * 10^6)) = 67.1 seconds

3. Overflow Calculations (8 MHz Clock):

- TCNT = 0xA473 = (0xA4 * 256) + 0x73 → (164)(256)+115 = 42099
- Max Time = 65536 → 65536 - 42099 = 23437 counts until overflow occurs
- LDI R20, 0x05 → Prescaler = 1024
- Tick Time = (Prescaler)/(Frequency) → 1024/8M = 1.28 * 10^-4 seconds
- Overflow = (Counts Remaining)*(Tick Time) → 23437 * 1.28 * 10^-4 = 3 seconds

5. Conclusions

In this lab, we were successfully able to learn about and use the AVR microcontroller's internal timer using both high-level and assembly code implemented on our ATmega328P boards. We tested these programs on the board by checking the timed delays of various LEDs lighting up using timer overflow in our code. The only real issue we had during this lab was hardware-related. Our first experiment wasn't working on the board at all due to a faulty wire. After replacing this wire connection between the input port and the LED port, we were able to get it to work successfully.