Homework 7

1) Write a C program to use Timer0 and Timer2 to generate 1 kHz and 4 kHz square waves on PORTB bit 1 and PORTB bit 2, respectively, using interrupts while at the same time. Assume Fosc = 8 MHz.

2) Given the following C program to generate/output a 1 kHz square wave on PB3 and at the same time transfer data from PORTD to PORTA. Convert the use of Timer 0 for generating the square wave which uses polling (below) to the use of Timer 0 with a Timer 0 overflow (TOV0) interrupt. This will require you to: a) configure Timer 0 overflow (TOV0) as an interrupt source, b) write the Interrupt Service Routine (ISR) for Timer 0 overflow (TOV0).

```
// generate a 1 kHz square wave on PB3 and at the same transferring data from PORTD to
PORTA

// TIMER0 FOR SQUARE WAVE
// PORTD => INPUT, PORTB => OUTPUT
//
// FOSC = 8 MHZ
// THALF = 0.5 X 1000 US = 500 US

// n = 500 us/[(1cc/1MC)(1s/8 10^6 cc)] = 4000 MCs

#include <avr/io.h>



int main(void)
{
	DDRB = 0xff;  // PORTB output
	DDRD = 0x00;  // PORTD INPUT
	PORTD = 0xff; // PULLUP RESISTORS FOR PORTD
	DDRA = 0xff;  // PORTA output

	TCCR0 = 0x04; 0b00000100 for prescale of 256 for clk for ATmega32
	TCNT0 = -16;



	while(1)
	{

		// use TIFR for ATmega32
		while (!(TIFR & (1<<TOV0)))
		{
			PORTA = PIND;
		}

		TCCR0 = 0x00; // turn off timer 0 for reset

		TIFR = 0x01;   // write a 1 to TOV0 to reset overflow flag
```

```
            TCNT0 = -16;    // reset preload
            PORTB ^= 0x08; // 00001000
            TCCR0 = 0x04; // turn timer 0 back on

        }

        return 0;
}
```

3) Assume Timer0 is programmed in normal mode, no prescaler (TCCR0 = 0x01), TCNT0 = 0xF1, I = 1 in SREG, and TOIE0 = 1. Give the steps for how the Timer0 overflow is triggered and how the Timer0 overflow interrupt is serviced (only need to include steps that get to the Timer0 Interrupt Service Routine). What is the last (assembly) instruction of the Timer0 Interrupt Service Routine and what does this instruction do to restore the program to where it was before the Timer0 interrupt?

4) Assume that Timer2 (TOV2) and Timer0 (TOV0)overflow interrupts and External Interrupt 1 are enabled. What happens when TOV2 and TOV0 occur simultaneously? What happens if all three interrupts occur simultaneously? What determines which interrupt is serviced first?

5) Convert the assembly program (below) to **your own** C code (ATmega32).

```
.ORG 0
RJMP MAIN

.ORG 0X0016
RJMP TIMER0_OVR_ISR

.ORG 50
LDI R16, 0X01
IN R17, DDRB
OR R17,R16
OUT DDRB, R17
LDI R16, 0XFF
OUT DDRC, R16
LDI R16, 0X00
OUT DDRD, R16
LDI R16, 0XFF
OUT PORTD, R16

LDI R16, -20
OUT TCNT0, R16
LDI R16, 0X02
OUT TCCR0, R16
LDI R16, 0X01
OUT TIMSK, R16
```

```
SEI
HERE: IN R18, PIND
OUT PORTC, R18
RJMP HERE

.ORG 100
TIMER0_OVR_ISR:  LDI R19, 0X01
IN R20, PORTB
EOR R20,R19
OUT PORTB, R20
LDI R16, -20
OUT TCNT0, R16
RETI
```

6) Chapter 10: 1, 3, 25, 31, 32
Problems are below.

SECTION 10.1: AVR INTERRUPTS

1. Which technique, interrupt or polling, avoids tying down the microcontroller?
2. List some of the interrupt sources in the AVR.
3. In the ATmega32 what memory area is assigned to the interrupt vector table?
4. True or false. The AVR programmer cannot change the memory address location assigned to the interrupt vector table.
5. What memory address is assigned to the Timer0 overflow interrupt in the interrupt vector table?
6. What memory address is assigned to the Timer1 overflow interrupt in the interrupt vector table?
7. Do we have a memory address assigned to the Time0 compare match interrupt in the interrupt vector table?
8. Do we have a memory address assigned to the external INT0 interrupt in the interrupt vector table?
9. To which register does the I bit belong?
10. Why do we put a JMP instruction at address 0?
11. What is the state of the I bit upon power-on reset, and what does it mean?
12. Show the instruction to enable the Timer0 compare match interrupt.
13. Show the instruction to enable the Timer1 overflow interrupt.
14. The TOIE0 bit belongs to register_____.
15. True or false. The TIMSK register is not a bit-addressable register.
16. With a single instruction, show how to disable all the interrupts.
17. Show how to disable the INT0 interrupt.
18. True or false. Upon reset, all interrupts are enabled by the AVR.
19. In the AVR, how many bytes of program memory are assigned to the reset?

7)

SECTION 10.2: PROGRAMMING TIMER INTERRUPTS

20. True or false. For each of Timer0 and Timer1, there is a unique address in the interrupt vector table.

21. What address in the interrupt vector table is assigned to Timer2 overflow?
22. Show how to enable the Timer2 overflow interrupt.
23. Which bit of TIMSK belongs to the Timer0 overflow interrupt? Show how it is enabled.
24. Assume that Timer0 is programmed in Normal mode, TCNT0 = $E0, and the TOIE0 bit is enabled. Explain how the interrupt for the timer works.
25. True or false. The last three instructions of the ISR for Timer0 are:

```
LDI   R20,0x01
OUT   TIFR,R20    ;clear TOV0 flag
RETI
```

26. Assume that Timer1 is programmed for CTC mode, TCNT1H = $01, TCNT1L = $00, OCR1AH = $01, OCR1AL = $F5, and the OCIE1A bit is enabled. Explain how the interrupt is activated.
27. Assume that Timer1 is programmed for Normal mode, TCNT1H = $FF, TCNT1L = $E8, and the TOIE1 bit is enabled. Explain how the interrupt is activated.
28. Write a program using the Timer1 interrupt to create a square wave of 1 Hz on pin PB7 while sending data from PORTC to PORTD. Assume XTAL = 8 MHz.
29. Write a program using the Timer0 interrupt to create a square wave of 3 kHz on pin PB7 while sending data from PORTC to PORTD. Assume XTAL = 1 MHz.

8)

SECTION 10.4: INTERRUPT PRIORITY IN THE AVR

47. Explain what happens if both INT1F and INT2F are activated at the same time.
48. Assume that the Timer1 and Timer0 overflow interrupts are both enabled. Explain what happens if both TOV1 and TOV0 are activated at the same time.
49. Explain what happens if an interrupt is activated while the AVR is serving an interrupt.
50. True or false. In the AVR, an interrupt inside an interrupt is not allowed.

9)

SECTION 10.3: PROGRAMMING EXTERNAL HARDWARE INTERRUPTS

30. True or false. An address location is assigned to each of the external hardware interrupts INT0, INT1, and INT2.
31. What address in the interrupt vector table is assigned to INT0, INT1 and INT2? How about the pins?
32. To which register does the INT0 bit belong? Show how it is enabled.
33. To which register does the INT1 bit belong? Show how it is enabled.
34. Show how to enable all three external hardware interrupts.
35. Assume that the INT0 bit for external hardware interrupt is enabled and is negative edge-triggered. When is the interrupt activated? How does this interrupt work when it is activated.
36. True or false. Upon reset, all the external hardware interrupts are negative edge triggered.
37. The INTF0 bit belongs to the _____ register.
38. The INTF1 bit belongs to the _____ register.
39. Explain the role of INTF0 and INT0 in the execution of external interrupt 0.
40. Explain the role of I in the execution of external interrupts.
41. True or false. Upon power-on reset, all of INT0–INT2 are positive edge triggered.
42. Explain the difference between low-level and falling edge-triggered interrupts.
43. Show how to make the external INT0 negative edge triggered.
44. True or false. INT0–INT2 must be configured as an input pin for a hardware interrupt to come in.
45. Assume that the INT0 pin is connected to a switch. Write a program in which, whenever it goes low, the content of PORTC increases by one.