EEL 4742C: Embedded Systems
Homework 3

QUESTION 1. **(10 points)**

Part a) Complete the table below for the listed interrupt events.

Interrupt event	Enable bit	Flag bit	Register (containing the enable/flag bits)	Vector
TAR rollback to zero	TAIE	TAIFG	TACTL	TIMERA1-Vector
Channel 0 compare event (TAR = TACCR0)	CCIE	CCIFG	TACCTL0	TIMERA0-Vector
Channel 1 compare event (TAR = TACCR1)	CCIE	CCIFG	TACCTL1	TIMERA1-Vector
Channel 2 compare event (TAR = TACCR2)	CCIE	CCIFG	TACCTL2	TIMERA1-Vector
Port 1 input change (8 events)	P1IE	P1IFG	P1IE/P1IFG	PORT1_Vector

Question 1

b. The three eventsare : GIE = 1 //global interrupt enabled
xIE = 1 //enable interrupts
IFG = 1 //flag has been raised

To avoid persistent interrupts, you disable the interrupt (xIE = 0) or clear the flag (IFG &= ~BIT7)
↓
or related
peripheral.

Question 2

enable global interrupt

a. _enable_interrupt;

b. TIMer_A enabled to raise interrupt in Continuous

```
        TA0CTL= TASSEL_1 | MC_2 | TACLR | TAIE;
        TA0CTL|=TAIE;
```

c.
```
~~[illegible]~~ TA0CCTL0 |= CCIE;
```

d.
```
------> TA0CCTL1 |= CCIE;
```

e.
```
   #pragma vector = TIMER_A0
   __Interrupt void TA0_ISR {
       P1OUT ^= BIT4;
       // HW clears flag }
```

f.
```
# pragma vector = TIMER_A1_Vector
   __interrupt void TA1_ISR {
          P1OUT ^= BIT1; //Toggle 1.1 Led
       if((TA0CCTL2 & CCIFG) !=0 ) {
          P1OUT ^= BIT4; }
TA0CTL ~~TA0CCTL~~ &= ~TAIFG;
    }
```

g. (Next PAGE) →

g.
```
# Pragma vector = PORT1_VECTOR
    __interrupt void PORT1_ISR(){
        if((P1IFG & BIT7) != 0){
            P1OUT ^= BIT5;
            P1IFG &= ~BIT7;
        }
}
```

Question 3

Choose LPM: A - C

a. Smclk w/ interrupt: LPM0

b. Aclk w/ interrupt: LPM3

c. button w/ no timer: LPM4

No LPM override: d, e

d. Smclk in LPM3?

The smclk doesn't respond and remains off

e. ACLK in LPM4?

The ACLK doesn't respond and remains off

LPM override supported: f, g

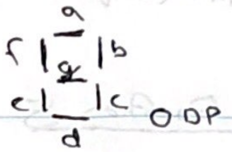
f. Smclk in LPM3?

Peripheral request on override to LPM and clock turns on

g. ~~Smclk~~ ACLK in LPM4?

Peripheral request on override to LPM and clock turns on.

Question 4.

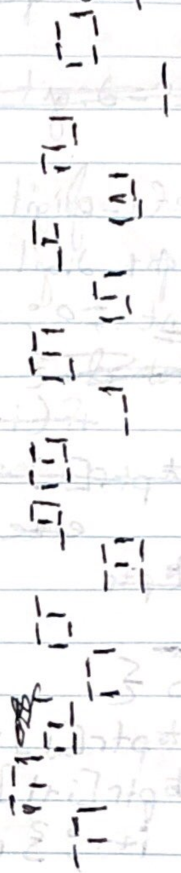


PartA. 0~15 on LCD where LCDM1 MSB[DP G F E D C B A]

unsigned char LCD HexChar[16] =
{

//index	Binary	Decimal
index 0:	0011 1111	0x3F
index 1	0000 0110	0x06
index 2	0101 1011	0x5B
index 3	0100 1111	0x4F
index 4	0110 0110	0x66
index 5	0110 1101	0x6D
index 6	0111 1101	0x7D
index 7	0000 0111	0x 07
index 8	0111 1111	0x7F
index 9	0110 0111	0x67
index 10	0111 0111	0x77
index 11	0111 1100	0x7C
index 12	0011 1001	0x39
index 13	0101 1110	0x5E
index 14	0111 1001	0x79
index 15	0111 0001	0x71

Shape



10	A
11	b
12	c
13	d
14	e
15	f

PartB. Using Ptr address to display all segments.

```
                                            0x7F //
const unsigned char allSegments = ~~0x3F~~   // code for all segments on
unsigned char *ptr[8]={&LCDM1, &LCDM2, &LCDM3, &LCDM4
//Array of addresses     &LCDM5, &LCDM6, &LCDM7, &LCDM8}
   int i=0;
do{
   *ptr[i] = allSegments; //sets each address to
   i++
   } while(i<9);
```

0x64 0x0F 0xF0

right = (0x64&0xF& 0110 0100 0110 0100
0000 1111 1111 0000
0000 0100

Question 5

a.
```
unsigned char n = 0x7D  // 7 d displayed
unsigned char left_digit = 0xF0
unsigned char right_digit = 0x0F

~~left_digit &= 0xF0~~   ~~left_digit = ( n &~~

left_digit &= n;
right_digit &= n;
int i=0;
~~do {~~
    ~~if(i<1)~~                     ~~if(i==1)~~
~~*ptr[i] = right_digit;~~          ~~*ptr[i] = left_digit;~~
    ~~else~~                        ~~else if(i==0)~~
~~*ptr~~                            ~~*ptr[i] = right_digit;~~
do {
  *ptr[i] = right_digit;
  *ptr[i+1] = left_digit;
  i++; } while ((i+1) < 1 )
```

Ignore

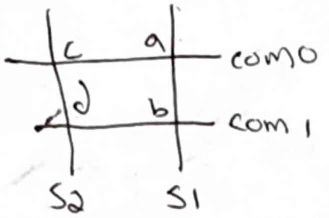
~~old~~ ~~unsigned~~

b.

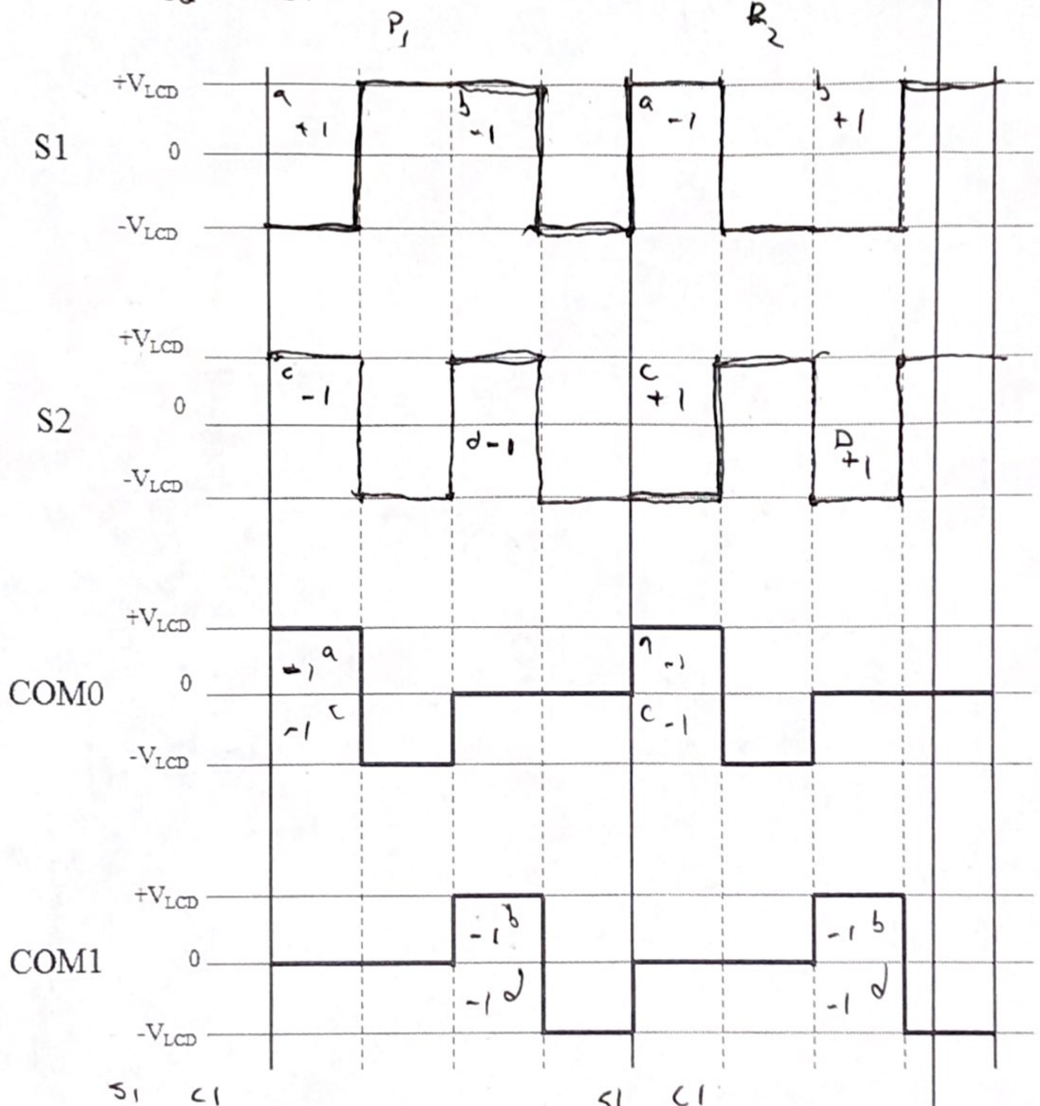
```
unsigned char x = 68 ;
 unsighed char left_digit;
 unsigned char right_digit;

right_digit = (int)x % 10;
left -digit=(int)x /10;
int i=0;
do {
 if(i==0) {
 * ptr[i] = LCD HexChar [right_digit];
 * ptr[i+1] = LCD HexChar[left_digit]; }
  i++
 } while( x!=0 ); //Clears rest to zero
 while(i<=8)
     {
       * ptr[i]=0;
         i++;
     }
```

Question: 6

Figure



	P1	P2
A	on	off
B	off	on
C	off	on
D	off	on



S1 C1

P_1 a = 1 - (-1) = 2 on

P_1 b = -1 - (-1) = 0 off

P_1 c = -1 - (-1) = 0 off

P_1 d = -1 - (-1) = 0 off

S1 C1

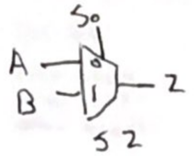
P_2 a = -1 - (-1) = 0

P_2 b = 1 - (-1) = 2

P_2 c = 1 - (-1) = 2

P_2 d = 1 - (-1) = 2

S → segments



$Z = (A\bar{S}) + (BS)$

$\frac{100}{4} = 25 + 4$

S	Z
0	A
1	B

Question 7

a. 4 way multiplex in LCD, Com Lines? 4 com lines

b. 4 way multiplex w/ 100 segments. $\frac{100}{4_{com}} = 25$ S lines

4 S lines for Vcc + gnd etc..

25 + 4 = 29 S lines

c. ~~False~~ True because if the segments change (on/off) the pattern, the S lines will change by pulling up/down ~~the~~ to match the required value of the segment output. ~~It is more~~ c Com lines are clock dependent

d. True because if the segments are set to be turned off and on, the S lines change the value to match the corresponding output by pulling up/down.

e.

	P1	P2
a	on	off
b	off	on
c	off	on
d	off	on

P1

LCDM1 : 0000 1000 → 0x08

P2

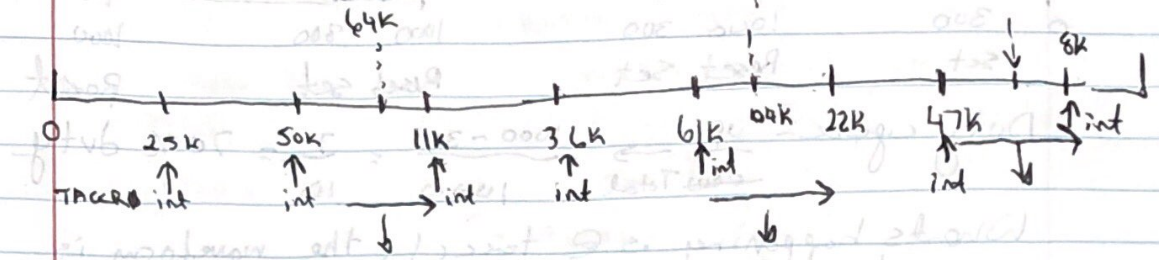
LCDM1 : 0000 0111 → 0x07

f. The multiplexer, The clock signal, the peripheral device and the waveforms.

Question 8

Part A) interrupts every 25k (25600)

64k clock



TACCRφ 75k (76800)

1 − 64k (65536)

TACCR1 11k (11264)

86k

− 64k (65536)

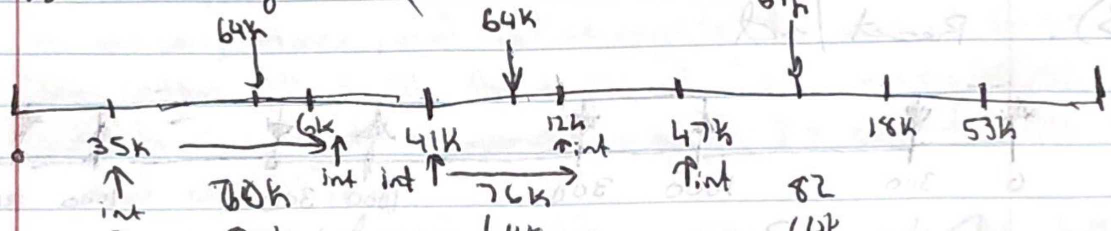
22k (22528)

72k

64k

8k (8192)

Part B every 35k (35840)



70k

− 64k

TACCR2 = 6k (6144)

76k

64k

12k (12288)

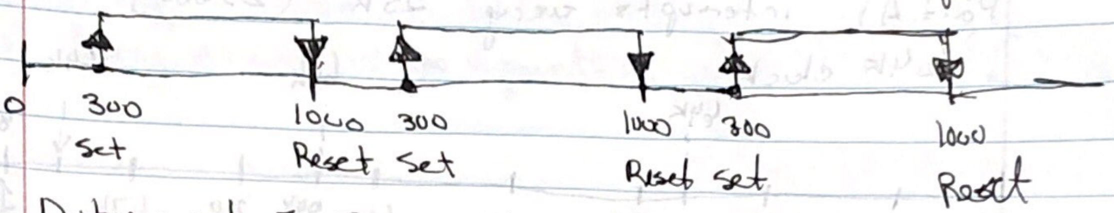
82

64k

18k (18432)

Question 9.

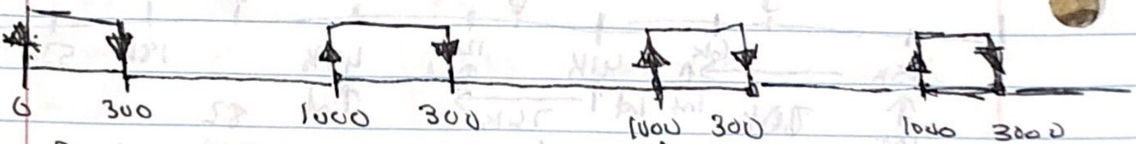
A.) Set/Reset TACCR1=300 w/ 1000 cycles



Duty cycle = $\frac{up}{\cancel{dw}\ Total} \rightarrow \frac{1000-300}{1000} = \frac{7}{10} = 70\%$ duty

Whats happening is @ taccr1, the waveform is set by turning on until it rolls back to zero where it is reset.
This means that the clock runs with 70% of the time running.

B). Reset/set



Duty $\frac{300}{1000} = \frac{3}{10} = 30\%$ duty

Whats happening is @ 300=TACCR1, the cycle is reset to zero until it rolls bau over to zero where its set to positive again.

push button, wait .25 seconds, trigger interrupt

Question 10

Part A) up mode - yes

32khz clk
TACCRO = 8192 // .25 seconds @ 32khz
MC = 1
ID = 0
TASSEL = 1
TAIE
LPM3

- button interrupt
  enable timer interrupt
  disable button
  clear flg
- Timer
  disable timer
  enable ~~timer~~ button ← trigger req interrupt
  clear flg

This is possible bc we need 3 required interrupts; a button, timer, and the interrupt @ the end of timer. The button will enable the timer set to Taccl ~~2~~8192 which represents a quarter second, It will disable button and clear flag.
The timer interrupt will disable timer, enable button and enable the interrupt before clearing the flag.

Part B) continuous - yes

MC = 2
ID = 0
TASSEL = 1
TAIE
LPM3

To do it in continuous mode the only difference is iN the button interrupt, we set TAOCCRO to the TAR + ~~[illegible]~~ (32khz clock)
24576

Which is .75 seconds into the 32khz frequency. When the timer flag is raised after .25 seconds, we can disable the timer @ the end before reenabling it in the button.

c. The up mode is ~~poffe~~ preferable because we can set a limit for the timer and use only that interval as opposed to forcing TAR+9 to be @ a certain point to 'run out the clock'. It also allows for better reusability.

d. Yes, to do so, we enable the timer interrupt in the prior interrupt when its used and disable the timer interrupt.

e. Channels ↓

	TAIFG	CCIFG
0	1	0
1	1	1
2	1	1

//HW Clears flag

f. When we disable the interrupt, that way the flag is completely cleared when we do enable it to avoid any timer error or errant flag raising.