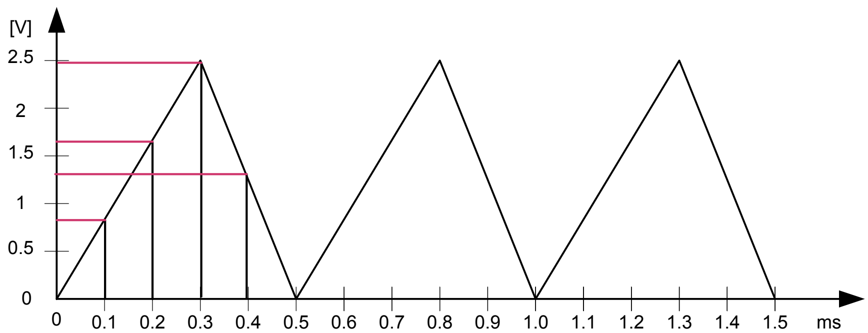
CPE 323 Introduction to Embedded Computer Systems
Homework VI

1 (30)	2 (20)	3 (20)		Total

Problem #1. (30 points) ADC.
Your task is to write a program that samples an analog signal *a0* as illustrated below using the MSP430's ADC12 device. The samples are then forwarded to the MSP430's DAC12 device. Answer the following questions.



A. (2 points) What is the maximum and minimum input voltage of the input signal *a0*?

$V_{min} = 0v$ $\quad V_{max} = 2.5v$

B. (2 points) What is duration of one period of the input signal *a0* in milliseconds? What is the frequency of *a0*?

$T = 0.5\,mS. \quad = 2{,}000\,Hz$

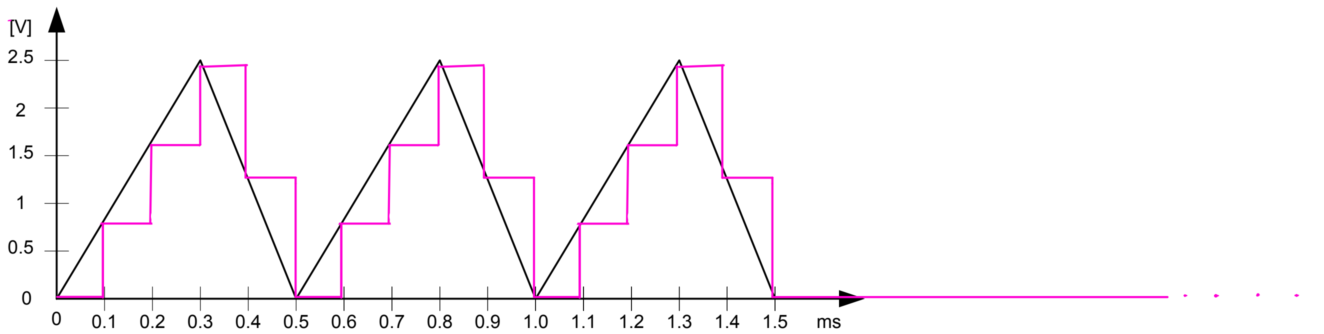
C. (10 points) Let us assume that we configure the MSP430's device to sample analog input *a0* with the sampling frequency f_{sample} = 10 KHz (10,000 samples are taken every second). How many samples do we have per one period of the input analog signal? Fill in the following table (please note that the number of rows does not reflect the number of samples per one period). Assume that our sampling is synchronized with *a0* (i.e., the first sample is taken at the very beginning of an *a0* period at t=0 s). Assume reference voltages V_{R+} = 2.5 V and V_{R-} = 0V.

$f_{sample} = 10kHz$

Sample number	*a0* [V]	Sample value [decimal]
0	V = 0 ✓	0
0.1ms	V = 0.833 ✓	1364
0.2ms	V = 1.66 ✓	2719
0.3 ms	V = 2.50 ✓	4096
0.4ms	V = 1.35 ✓	2211
0	V = 0 ✓	0

12 bits: 2^{12}
2.5v max
Smallest: $\frac{2.5}{2^{12}}$
$= 0.6mV$

D. (4 points) Assume the ADC12 samples are immediately sent to MSP430 DAC12 device. Sketch the output analog signal created by the MSP430's DAC12 device (as you would see it on the oscilloscope).



E. (7 points) Give a short description of your program that performs the ADC and DAC conversions. We assume that clocks are initialized as follows: $f_{MCLK}=f_{SMCLK}=4$ MHz. What should be done to initialize devices, what is done in the main program loop, and what is done in corresponding interrupt service routines?

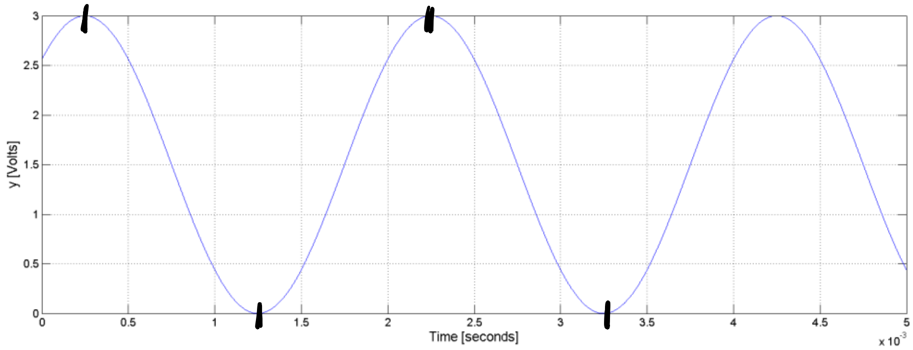
- Clock Subsystem (MCLK = 4MHz, SMCLK = 4MHz, ACLK = 32KHz)
- Configure corresponding ports for special functions (ADC12 in, DAC12 out.)
- Timer A to control sampling; fsample = 5 x 2000 = 10,000 Hz
  ↳ CCR0 up mode. TACCR0 = 4MHz/10kHz = 10.
- Configure ADC12 (trigger timer A CCR0) V_R+ = 2.5V, V_R- = 0V
- Configure DAC12; 12-Bit, scale = 1, V_R = 2.5V ADC12-ISR
  Read sample from A0. DAC12_VDAT = Sample

F. (5 points) If you know that less than 80 clock cycles is spent for processing one sample (read the sample for the ADC12 and write the digital value of the sample to the DAC12 data register), what would be the maximum sampling frequency we could have without oversubscribing our processor time? Elaborate your answer.

5 samples x 80 cc = 400 cc for .5ms
Processor ticks in .5ms is (.5E-3) / (.25E6) = 2000 clock cycles.
Now increase the number of samples to 2000/80 25 samples per period which is 2000*25 = 50,000Hz

Problem #2. (20 points) ADC, DAC.

To drive an external actuator we need to provide an analog periodic signal y, defined as follows, y=1.5*(1+sin(2*pi*f*t+pi/4)), f=500 Hz. Your task is to generate this signal using a **16-bit digital-to-analog** converter peripheral (DAC16). Answer the following questions.



A. (2 points) What is the maximum and minimum output voltage at the analog output? Fill in the table below by specifying min/max values and times when those values are achieved.

Min/Max	Value [Volts]	Time [milliseconds]
Min	0V	t= 1.25ms t= 3.25ms
Max	3V	t= 0.25ms t= 2.25ms t= 4.25ms

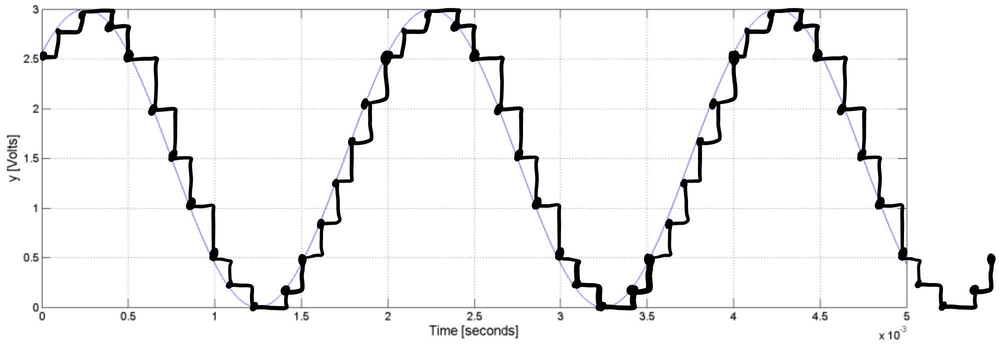
B. (2 points) What is duration of one period of the signal y in milliseconds?

T= 2.0ms

C. (8 points) Let us assume that we provide a lookup table with only 16 samples for a single period of the signal y? Fill in the following table with the values for the first 4 samples (assume the first sample starts at t=0). Assume reference voltages V_{R+} = 3.0 V and V_{R-} = 0V

Sample	*t=? [ms]*	*sin(2*pi*f*t+pi/4)*	*y=1.5*(1+ sin(2*pi*f*t+pi/4))*	*Lookup table [16-bit unsigned value in decimal]*
1	0	0.0137	1.520	33204
2	.125	0.9238	2.885	63022
3	.250	0.1	3	65535
4	.375	0.9238	2.885	63022

E. (4 points) Sketch the output analog signal as you would see it on the oscilloscope. Use the lookup table from part C.



F. (4 points) If we use a TimerB ISR to control sending the next value to the DAC, how many interrupts per second TimerB should generate?

Timer B IPS? For DAC

First way

Frequency = 500 Hz

1 period = 2ms

1 period = 16 samples

500 samples persecond

500x16 = 8,000

Second way

16 samples in 2ms

Sample persecond

$$\frac{16}{2E-3} = \frac{x}{1} \qquad \frac{16}{2E-3}$$

= 8000 samples in one second.

If a sample is sent every time an interrupt is called we need 8000 interrupts per second

3. (20 points) I/O Peripherals

A. (20 points) You are designing an embedded application that interfaces a buzzer and a switch using MSP430's general-purpose input/output pins. The button is connected to P4.4 and the buzzer is connected to P4.2. Illustrate how the button and the buzzer connect to the MSP430. Your application should control the buzzer as follows: as long as we keep the button pressed (logic 0 at the input), the buzzer should be on. To activate the buzzer you should provide a square wave with frequency of 8 KHz.

Sketch the microcontroller and its connections to the button and the buzzer. Describe the application design. What needs to be done during initialization, what is done in the main program loop, and what is done in interrupt service routine(s)? You can use C/C++ code or English/pseudo-code to illustrate your design.

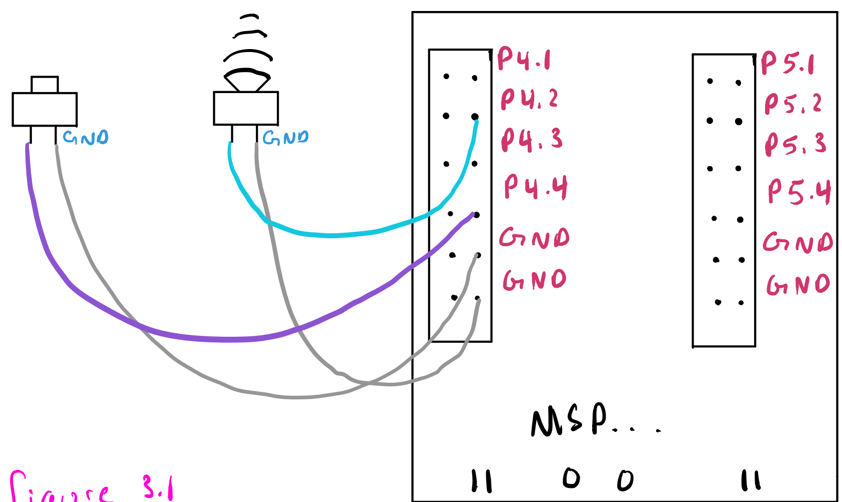


Figure 3.1

Button, Buzzer, Device System.

```c
#include <msp430F5529.h>
#include <stdbool.h>

bool Buzzer_Enabled = false;

void main(void)
{
    WDTCTL = WDTPW + WDTHOLD;   // Stopping the watchdog timer
    __enable_interrupt();       // Enable interrupts globally

    TA0CCTL0 = CCIE;            // Enabled TimerA interrupt
    TA0CCR0 = 4;                // 4 / 32768 Hz = 0.125ms or 8KHZ
    TA0CTL = TASSEL_1 + MC_1;   // ACLK, up mode

    P4DIR |= 0x02;              // P4.2 is output direction for buzzer.
    P4REN |= BIT2;              // Enable the pull-up resistor at P4.2
    P4OUT &= ~0x02;             // Buzzer is off at start.

    P4DIR &= ~BIT4;             // Set P4.4 as input (External Switch)
    P4REN |= BIT4;              // enable pull-up resistor
    P4OUT |= BIT4;

    P4IE  |= BIT4;              // Configuring switch on P4.4 interrupt.
    P4IES |= BIT4;
    P4IFG &= ~BIT4;

    _BIS_SR(LPM0_bits + GIE);   // Enter Low Power Mode 0
}

// External Switch Press.
#pragma vector = PORT4_VECTOR
__interrupt void PORT4_ISR(void)
{
    P4IFG &= ~BIT4;             // Clear interrupt flag.
    Buzzer_Enabled ^= true;     // Set the Buzzer to true.
    P4IES ^= BIT4;              // Toggles External Switch edge select.
}

//TimerA Interrupt Service Routine
#pragma vector = TIMER0_A0_VECTOR
__interrupt void timerA_isr()
{
    if(Buzzer_Enabled)
    {
        P4OUT ^= 0x02;          // Toggle the buzzer at 8KHZ. 4 up toggles it.
    }
}
```

Figure 3.2

Code will toggle Buzzer @ 8KHz while the button on P4.4 is pressed.