Jonathan W. Valvano First:_____ Last:_____

This is the closed book section. Calculator is allowed (no laptops, phones, devices with wireless communication). You must put your answers in the boxes. Do not write on the back of the page. When you are done, you turn in the closed-book part and can start the open book part.

(4) Question 1a. State two differences between ceramic and tantalum capacitors.

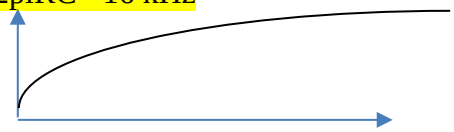
Tantalum are polarized, ceramic are not	Tantalum have more leakage than ceramic.

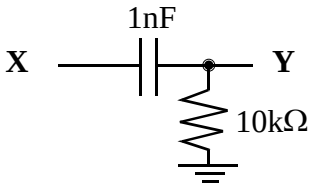
(4) Question 1b. State two differences between TCP and UDP communication.

TCP is more reliable TCP show up in order	UDP is faster

(4) Question 1c. Sketch the gain versus frequency response of this circuit. 1nF*10kΩ is 10 μsec.

fc = 1/2piRC =16 kHz





(4) Question 1d. Explain why this software is classified as an I controller?

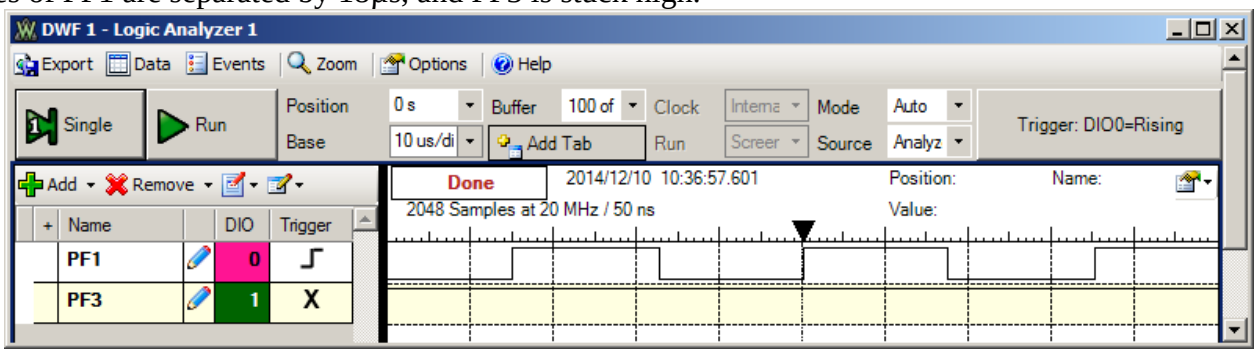
U = U+(3*E)/64 implements a discrete integral
$U(t) = \int K_I * E(t)\, dt$

```
uint32_t Period;
uint32_t Speed;
int32_t E;
int32_t U;
void Timer2A_Handler(void){
  TIMER2_ICR_R = 0x01;
  Speed = 800000000/Period;
  E = 250-Speed;
  U = U+(3*E)/64;
  if(U < 40) U=40;
  if(U>39960) U=39960;
  PWM0_Duty(U);
}
```

(4) Question 1e. List four of the ten points of the IEEE Code of Ethics?

1) Accept responsibility 2) Avoid conflict of interest 3) Be honest	6) maintain competence 7) seek, accept, offer honest criticism 8) treat all persons fairly
4) Reject bribery 5) Improve understanding of technology	9) avoid injury to others 10) assist and support others in development and following ethics

(10) Question 2. The goal of this system is to test a new DAC, by creating a sine wave with a period of 2.56ms using a Timer2A periodic interrupt. The DAC uses SPI running at with a 4 MHz clock. The **`SineWave`** buffer has 256 entries per cycle. **`TIMER2_TAILR_R`** is 799, making the interrupt rate 100 kHz (10µS). Each execution of the ISR outputs one point of the **`SineWave`** buffer to the DAC. The debugging profiles for PF3 and PF1 are shown in the following plot. The PLL is active (80MHz). The edges of PF1 are separated by 18µs, and PF3 is stuck high.




```
void UserTask(void){ uint32_t static I=0;
  PF1 ^= 0x02;           // toggle PF1, profiling
  DAC_Out(SineWave[I]);  // Output one point using the DAC
  I = (I+1)&0xFF;}
int main(void){
  PLL_Init(Bus80MHz);    // bus clock at 80 MHz
  PortF_Init();          // make PF1, PF3 outputs
  Timer2A_Init(&UserTask, 800);  // initialize timer2A (100kHz)
  EnableInterrupts();
  while(1){
    PF3 ^= 0x08;}}
```

(5) Part a) Which of the following changes would you make to fix the error in this DAC test? The period of the sine wave output must remain at 2.56ms. If there is more than one answer, choose just one answer.

A) Increase **`TIMER2_TAILR_R`** so the Timer0A interrupts occur less frequently.

B) Decrease **`TIMER2_TAILR_R`** so the Timer0A interrupts occur more frequently.

C) Change the PLL so the TM4C123 runs slower than 80 MHz.

D) Change **`uint32_t static I=0;`** to **`uint32_t volatile I=0;`**

E) Use a FIFO queue to pass data between threads, decoupling the two execution of the two threads

F) Change **`I=(I+1)&0xFF;`** to **`I=(I+1)%256;`**

G) Reduce the number of points in the **`Wave`** buffer from 256 to 128, set **`TIMER2_TAILR_R`** to 1599, and change **`0xFF`** to **`0x7F`**.

H) Replace the DAC with a version with a faster SPI clock

I) Remove the line **`PF1 ^= 0x02;`**

J) None of the above changes will remove the bug.

G or H

(5) Part b) How would you characterize the debugging profile in the above figure?

A) Nonintrusive **C)** Highly intrusive **E)** Invasive

B) Stabilization **D)** Minimally intrusive **F)** Desk check

D)

(3) Question 3. In order to apply the Central Limit Theorem we assume the noise is random, the noise in each sample is independent from the noise in the other samples, and the noise has zero mean. In order to improve signal to noise ratio in our ADC samples, we can take multiple samples and calculate the average (e.g., ADC Sample Averaging Control, ADC0_SAC_R). List two consequences of the CLT as it applied to this situation.

Distribution becomes Gaussian	Average approaches mean (truth) and also the sigma approaches 0

(3) Question 4. In a communication system, as data is passed from point X to point Y, in what form is the information transmitted? Choose the best answer.

A) Voltage D) Energy
B) Current E) Time
C) Power F) Resistance

D) Energy

(9) Question 5. The following FIFO has one or more critical sections. This FIFO is used in a multithreaded application, such that data is passed from main program (**`Fifo_Put`**) to ISR (**`Fifo_Get`**). Add code to this module to remove the critical section(s).

```
uint16_t Buf[3]; // place for three 16-bit numbers
uint8_t Size;    // 0 means empty, 1 means one, 2 means 2, 3 means full
int Fifo_Init(void){
  Size = 0; // runs during initialization with interrupts disabled
  return 0;
}
int Fifo_Put(uint16_t data){ // store data into FIFO
  if(Size >= 3){
    return 1; // full, fail
  }
  Buf[Size] = data;
  Size++;     // one more
  return 0;   // success
}

int Fifo_Get(uint16_t *p){  // remove data from FIFO
  if(Size == 0){
    return 1; // empty, fail
  }
  *p = Buf[0];          // return data
  Buf[0] = Buf[1];      // shift buffer
  Buf[1] = Buf[2];      // shift buffer
  Size--;               // one less
  return 0;             // success
}
```

DisableInterrupts();

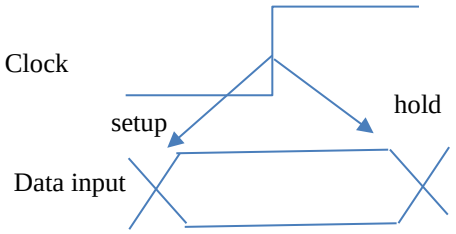
EnableInterrupts();

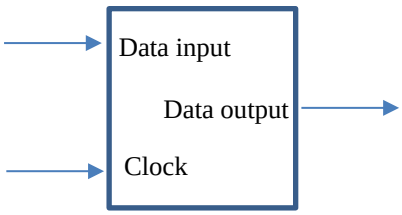
(5) Question 6. Estimate the lifetime of a battery-operated embedded system. The system uses a 3.7-volt LiIon battery with a storage of 740 joules. The regulator has a power efficiency of 50%, creating the 3.3V supply for the system. The average current in the system is 1 mA, because it sleeps most of the time. Determine how many seconds will this battery run the system? Show your work.

Storage = 740 J/3.7V = 200 A-sec
Available storage = 100 A-sec
Time = 100 A-sec/1mA= 100,000 sec

(5) Question 7. Use a flip flop to define setup and hold time. Include a timing diagram.

Clocking data into a flip flop, such that data is latched on rising edge of clock. Setup is the time before the rising edge the input data must be stable, and hold is the time after the clock the data must remain stable.





Jonathan W. Valvano First:_____ Last:_____

Open book, open notes, calculator (no laptops, phones, devices with screens larger than a TI-89 calculator, devices with wireless communication). You must put your answers in these boxes. Please don't turn in any extra sheets or write on the back of the pages.

(20) Problem 8. You are given two analog inputs signal connected to the microcontroller (any pins you wish). Your goal is to produce a single analog output, which is the arithmetic average of the two inputs. You must use the ADC, interrupts, and a DAC. The signals of interest are 0 to 100 Hz.

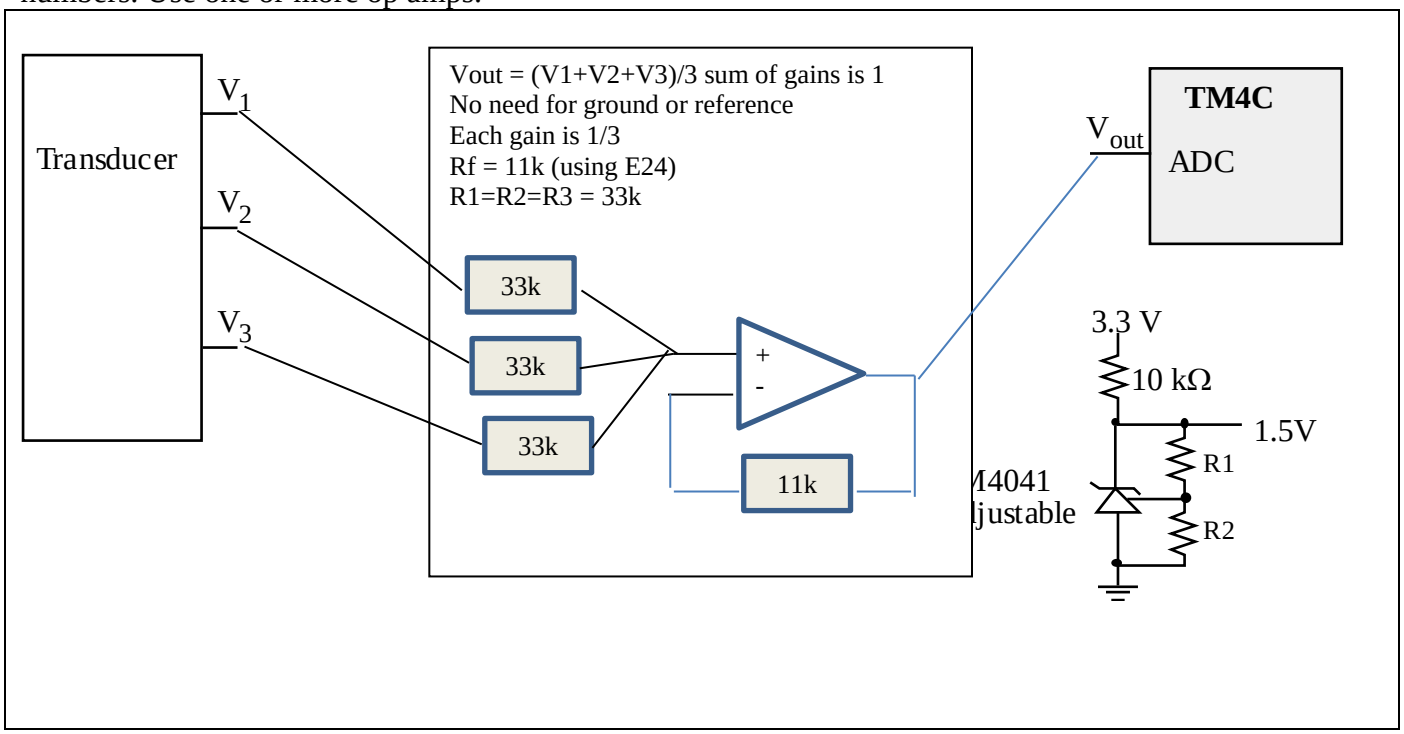
(12) Part a) Show the main program that initializes the ADC, periodic interrupts and the DAC. You may call any software function defined in the book (without showing the book code).

```
void main(void){
  PLL_Init();     // Program 2.9, 80 MHz
  ADC_Init();     // Program 8.9, 1000 Hz two channel sampling
  DAC_Init(2048); // Program 7.2
  EnableInterrupts();
  while(1){
  }
}
```

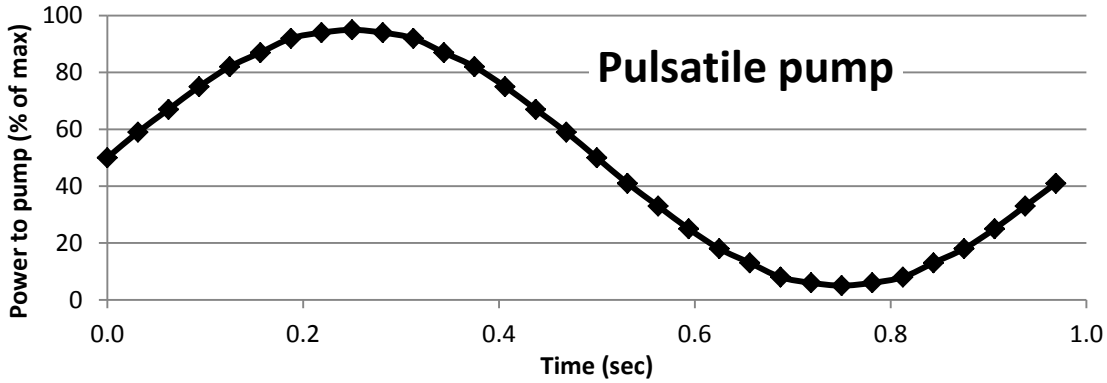
(8) Part b) Show the interrupt service routine that runs in the background, sampling the ADC, calculating the average, and outputting to the DAC. You may call any software function defined in the book (without showing the book code).

```
void ADC0Seq2_Handler(void){ // see program 8.9
uint32_t in1,in2,out;
  ADC0_ISC_R = 0x04;      // acknowledge ADC sequence 2 completion
  in1 = ADC0_SSFIFO2_R;  // PD3, Channel 4 first
  in2 = ADC0_SSFIFO2_R;  // PD3, Channel 5 second
  out = (in1+in2)/2;      // average
  DAC_Out(out);           // Program 7.2
}
```

(10) Question 9. Design an analog circuit that has three inputs and one output, such that the output is the arithmetic average of the three inputs. The input voltages are constrained to 0 to 3.3V. No analog filter is required in this question. The only available power supply voltage is 3.3V. Assume R1 and R2 are already chosen to achieve a reference of 1.5V. Show design steps, and specify all resistors, capacitors and chip numbers. Use one or more op amps.



(15) Question 10. The goal of this problem is to build a controller for a DC motor that drives a pump. Full power to the motor occurs at 12 V 10 A. In order to reduce shear stress on the pipes, you will implement a pulsatile pump controller. Over the course of 1 second, you will vary the applied power according this waveform. The time constant of the motor is about 100ms. Because this 1 second wave is 10 times slower than the time constant of the motor, the motor will respond to this wave.



The points on this waveform are captured in the following ROM-based array.

```
uint8_t const Wave[32] = {50,59,67,75,82,87,92,94,95,94,92,87,82,
    75,67,59,50,41,33,25,18,13,8,6,5,6,8,13,18,25,33,41};
```

Part a) The first task is to interface the DC motor to the microcontroller. Pick which interface circuit would you use from the book. If there is more than one answer, choose the best answer.

A) Figure 6.14 with a 2N2222 BJT
B) Figure 6.14 with a 7406 open collector driver
C) Figure 6.14 with a TIP120 Darlington
D) Figure 6.16 with both a 2N2222 BJT and IRF540 MOSFET
E) Figure 6.17 with a L293 H-bridge

D) MOSFET

Part b) Show the ritual that initializes the system. The main program calls this initialization, enables interrupts, and then performs other unrelated tasks. Assume the bus clock is 80 MHz. You may call any software function defined in the book.

```
uint32_t Time=0;    // units 1/32 sec
// PWM clock is bus cycle/2, 40 MHz, 1ms is 40000 counts
#define PMAX 40000  // PWM is fixed period 1ms, variable duty cycle
void Init(void){
  Time = 0;
  Timer2A_Init(&UserTask,80000000/32); // Program 6.6, 32 Hz interrupt
  PWM0_Init(PMAX,2); // Program 6.8, fixed PWM period at 1ms
  EnableInterrupts();
}
```

Part c) Show the interrupt service routine that produces the desired output to the DC motor.

```
void UserTask(void){
  PWM0_Duty((Wave[Time]*PMAX)/100); // Program 6.8, new duty cycle
  Time = (Time+1)%32;
}
```