Jonathan W. Valvano First:_____ Last:_____

Open book, open notes, open computer, calculator (wireless devices must be in airplane mode). Screens must not be visible to other students.

**(10) Question 1.** These two initializations are critical to each other, because they have read-modify-write nonatomic sequences with shared information. These are the definitions of the ports

```
#define SYSCTL_RCGCGPIO_R  (*((volatile uint32_t *)0x400FE608))
#define GPIO_PORTE_DIR_R   (*((volatile uint32_t *)0x40024400))
#define GPIO_PORTE_DEN_R   (*((volatile uint32_t *)0x4002451C))
```

```
void InitPE2(void){
volatile uint32_t delay;
  SYSCTL_RCGCGPIO_R |= 0x10;
  delay = SYSCTL_RCGCGPIO_R;
  GPIO_PORTE_DIR_R &= ~0x04; // PE2 input
  GPIO_PORTE_DEN_R |= 0x04;
}
```

```
void InitPE3(void){
volatile uint32_t delay;
  SYSCTL_RCGCGPIO_R |= 0x10;
  delay = SYSCTL_RCGCGPIO_R;
  GPIO_PORTE_DIR_R &= ~0x08; // PE3 input
  GPIO_PORTE_DEN_R |= 0x08;
}
```

**(5) Part a)** What is the basic idea of how to remove the critical section without disabling interrupts or moving the input to another port?

Use bit banding to create atomic access and remove sharing

**(5) Part b)** Rewrite the two functions (show C code), removing the critical section

If the I/O address is 0x4000.0000+*n*, and *b* = bit 0 to 7. The aliased address for this bit is 0x4200.0000 + 32**n* + 4**b*
For PE2 DIR n=0x24400, b=2 address = 0x4200000+0x488000+0x08 = 0x42488008
For PE3 DIR n=0x24400, b=3 address = 0x4200000+0x488000+0x0C = 0x4248800C
For PE2 DEN n=0x2451C, b=2 address = 0x4200000+0x48A380+0x08 = 0x4248A388
For PE3 DEN n=0x2451C, b=3 address = 0x4200000+0x48A380+0x0C = 0x4248A38C

```
#define PE2_DIR    (*((volatile uint32_t *)0x42488008))
#define PE3_DIR    (*((volatile uint32_t *)0x4248800C))
#define PE2_DEN    (*((volatile uint32_t *)0x4248A388))
#define PE3_DEN    (*((volatile uint32_t *)0x4248A38C))
void InitPE2(void){
volatile uint32_t delay;
  SYSCTL_RCGCGPIO_R |= 0x10;
  delay = SYSCTL_RCGCGPIO_R;
  PE2_DIR = 0; // PE2 input
  PE2_DEN = 1;
}
void InitPE3(void){
volatile uint32_t delay;
  SYSCTL_RCGCGPIO_R |= 0x10;
  delay = SYSCTL_RCGCGPIO_R;
  PE3_DIR = 0; // PE3 input
  PE3_DEN = 1;
}
void InitPE23(void){
volatile uint32_t delay;
  SYSCTL_RCGCGPIO_R |= 0x10;
  delay = SYSCTL_RCGCGPIO_R;
  GPIO_PORTE_DIR_R &= ~0x0C; // PE2, PE3 input
  GPIO_PORTE_DEN_R |= 0x0C;
}
```

**(10) Question 2.** The following ADC sampling occurs in this high priority 10kHz periodic ISR. The desired sampling rate is 10 kHz. The ADC is set at maximum speed, so the while loop requires about 1us to execute with SAC=0 (one sample). With SAC=0, the FIFO queue never gets full.

```
void Timer2A_Handler(void){
  TIMER2_ICR_R = 0x01;
  ADC0_PSSI_R = 0x0008;
  while((ADC0_RIS_R&0x08)==0){};
  FIFO_Put(ADC0_SSFIFO3_R);
  ADC0_ISC_R = 0x0008;
}
```

**(5) Part a)** To improve SNR, SAC was increased to 6 (64-sample averaging). This one change caused the FIFO to get full. Why did it fail?

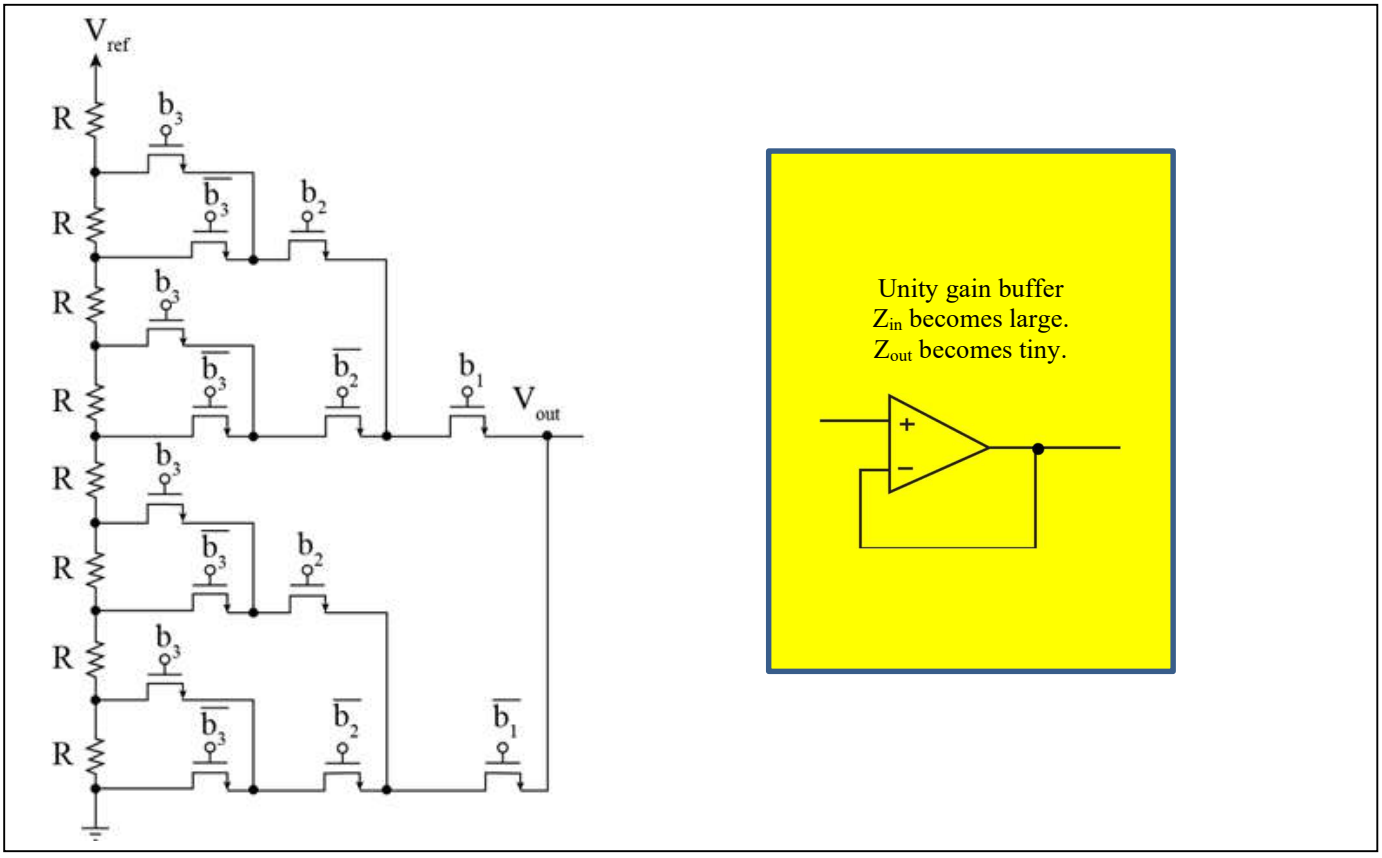
Before the change, the ISR required about 1us/100us = 1% processor time to run. 64*1us = 64us, so now the ISR takes about 64us to run. 64us/100us = 64% processor time. There are a lot fewer bus cycles to run the main program and other interrupts. The FIFO still gets one sample every 100us, but software tasks that get and service the data does not have enough time to complete.
The put rate remains the same at 10k samples/sec

**(5) Part b)** How would you fix it, so the ADC still samples at 10kHz with SAC=6? Explain how you would change the initialization of the ADC and interrupts. When would the ISR run, and what would the ISR do? Show the C code for the new ISR (but no code needed for changes to initialization).

Set the ADC to be timer triggered. The ADC will then be triggered at 10kHz without software interaction. Set the ADC to interrupt when done (remove the Timer ISR, add an ADC ISR). Since the interrupt occurs after the 64 samples are collected and averaged, the ISR will have NO while loops, and execute faster than 1us.

```
void ADC0Seq3_Handler(void){
  FIFO_Put(ADC0_SSFIFO3_R);
  ADC0_ISC_R = 0x0008;
}
```

**(5) Question 3.** The following is a broken 3-bit **resistor string** DAC. If there is nothing connected to $V_{out}$, the output is a linear function of the binary input $b_3,b_2,b_1$, with a range of 0 to $V_{ref}$. However, if we connect $V_{out}$ to another circuit, the output voltage drops a lot and is no longer linear. Redesign (fix) this broken circuit so the output can be connected to other circuits.




**(10) Question 4.** You are evaluating possible PID controllers for your design. To compare the various choices, you need a **figure of merit (FOM)**, which is a single number that combines all the performance metrics that matter to you. Create a FOM with at least **3 parts to it**, such that the bigger the value the better the controller, avoid calculations of infinity as FOM. Explain each component and give units for each. Draw figures to explain

Let **A** be average accuracy of full scale at steady state. $x_t$ is desired speed truth, $x_m$ is measured speed.

Average accuracy of full scale $A = \frac{1}{n}\sum_{i=1}^{n}\frac{|x_{ti}-x_{mi}|}{x_{max}}$ (dimensionless)

Calculate **X** = min(0.001,**A**), where 0.001 means my desired performance

Let **T** be the response time (time from change of setpoint to 1% final steady state value), in seconds.

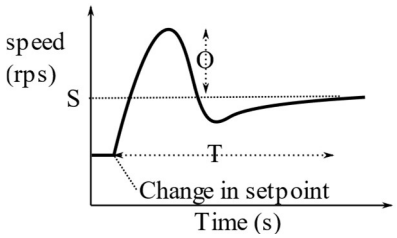
Let **τ** be the time constant of the motor, in seconds

Calculate **Y** = **τ** /T (dimensionless), Y=1 means perfect.

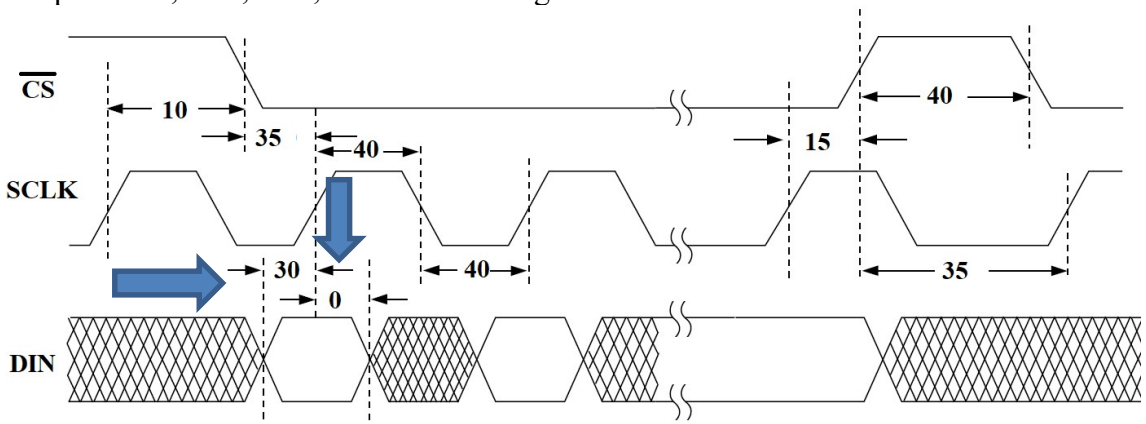
Let Z=min(0.01,**O/S)** be the maximum amount of overshoot occurring when setpoint changed in percent

The **min** functions avoid infinity calculations

**FOM = Y/(X*Z)**



**(10) Question 5.** Consider this timing diagram for the MAX549A 8-bit DAC. You are asked to interface this DAC to pins PB7, PB6, PB5, and/or PB4 using SSI2.




**(2) Part a)** What is the setup time for the MAX549A? Give the number in ns.

Setup = 30ns

**(2) Part b)** What is the hold time for the MAX549A? Give the number in ns.

Hold = 0ns

**(2) Part c)** To which TM4C123 pin (PB7, PB6, PB5, or PB4) would you connect CS?

Connect CS to SSI2Fss, PB5

**(2) Part d)** To which TM4C123 pin (PB7, PB6, PB5, or PB4) would you connect SCLK?

Connect SCLK to SSI2Clk, PB4

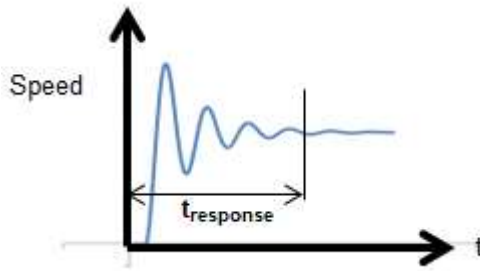
**(2) Part e)** To which TM4C123 pin (PB7, PB6, PB5, or PB4) would you connect DIN?

Connect DIN to SSI2Tx, PB7

**(5) Question 6.** Consider a N-point DFT, which was sampled at $\mathbf{f_s}$. The output of the DFT is an array of complex numbers with index **k** that ranges from 0 to N-1. For **k** < ½ N, give the relation between the index **k** and the equivalent frequency for the array value at index **k**.

$f = k*f_s/N$

**(10) Question 7. rps** is the measured speed. **Xstar** is the desired speed. The PI controller has this response (**rps** versus time) to a change in setpoint, **Xstar**. **Kp** and **Ki** are controller constants. The software runs at a constant rate of 100 Hz.




```
E   = Xstar-rps;
P   = (Kp * E)/4096;
P   = max(300,min(39990,P));
I   = I + (Ki * E)/4096;
I   = max(300,min(39990,I));
U   = P + I;
U   = max(300,min(39990,U));
PWM0A_Duty(U);
```

**(5) Part a)** To reduce overshoot, which would you change **Kp** or **Ki**?

Kp

**(5) Part b)** Would you increase its value, decrease its value, or change its sign?

Decrease Kp to reduce overshoot

**(10) Question 8.** Consider the following frequency measurement performed on PE0 input using edge-triggered and periodic interrupts. The edge-triggered ISR is invoked on the rising edge of PE0. The timer interrupt occurs every 10 ms. You may assume the bus clock is 80 MHz.

```
uint32_t Count,Freq;
void GPIOPortE_Handler(void){
  GPIO_PORTE_ICR_R = 0x01;
  Count++;
}
```

```
void Timer2A_Handler(void){
  TIMER2_ICR_R = 0x01;
  Freq = Count;
  Count = 0;
}
```

**(5) Part a)** What is the **frequency measurement resolution**? Give units

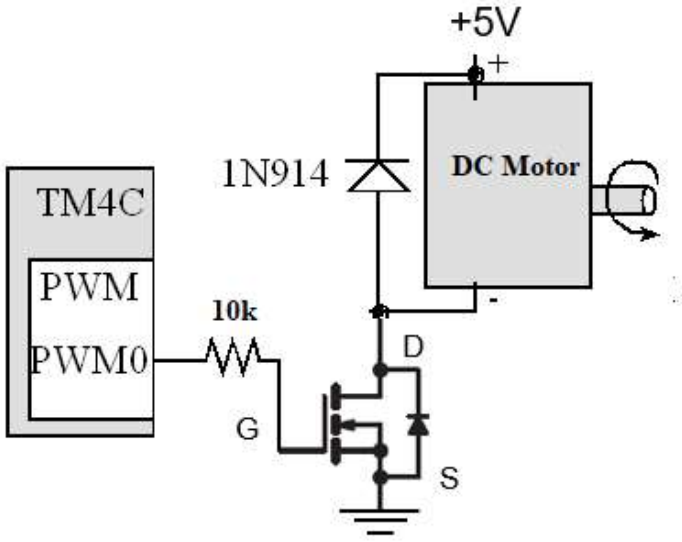
The timer ISR determines the number of rising edges in 10ms. The units of **Freq** will be 100Hz. If **Freq** = 12, the frequency is 1200 Hz. If **Freq** = 13, the frequency is 1300 Hz. The resolution is 100Hz.

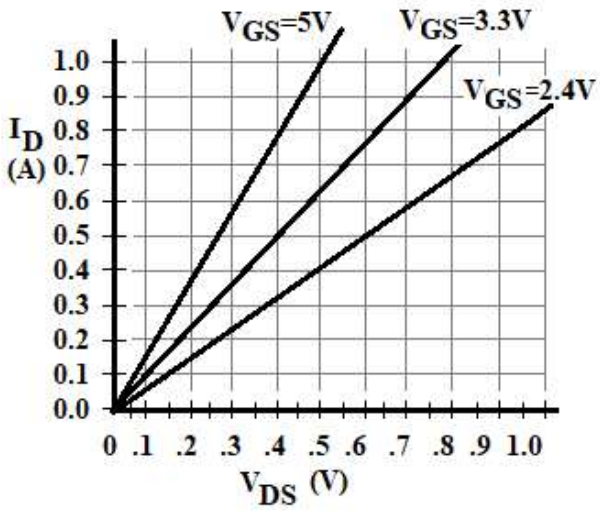
**(5) Part b)** Estimate the **frequency measurement range**. Give minimum and maximum values.

The minimum is 0 Hz (Count will be 0)
The maximum is determined by the execution time of the edge-triggered ISR, which will be about 1us. So the maximum frequency will be about 1000 kHz. If the frequency is above this maximum, the edge-triggered ISR will miss some edges, and consume 100% of the processor time.

Note: checkout the frequency measurement code in the book. It uses hardware counting so the maximum would go up to ½ bus frequency, and just uses the timer ISR.

**(10) Question 9**. Consider motor interface powered by 5V and a MOSFET. When 4 to 6V is applied directly across the motor, the current is about 0.5A. When the MOSFET is disconnected from the microcontroller, the output voltage, $V_{OH}$, is 3.3V. At a maximum $I_{OH}$ of 8mA, the $V_{OH}$ will be 2.4V. Estimate the power (in watts) applied to motor when the PWM duty cycle is 25%. Show your work.



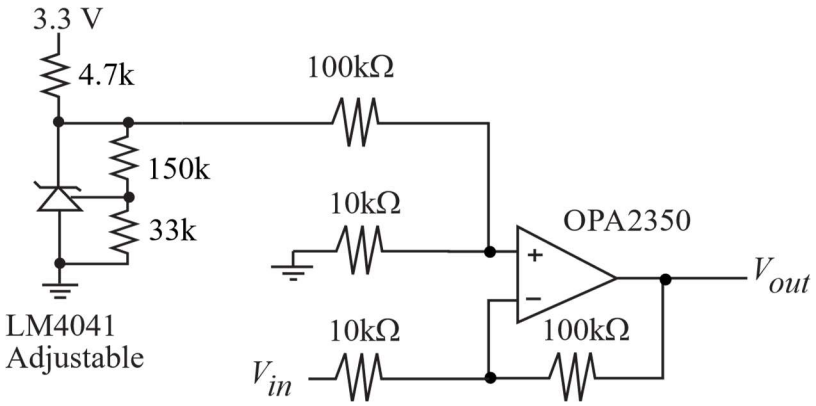




No current flows into the gate, so no current in 10k, so $V_{OH}$ will be 3.3V.
At $I_d$=0.5A, $V_{DS}$ will be 0.4V.
So, the voltage across the motor will be 4.6V
The max power will be 0.5A*4.6V = 2.3W
The power at 25% duty cycle is 2.3W/4 = 0.575W

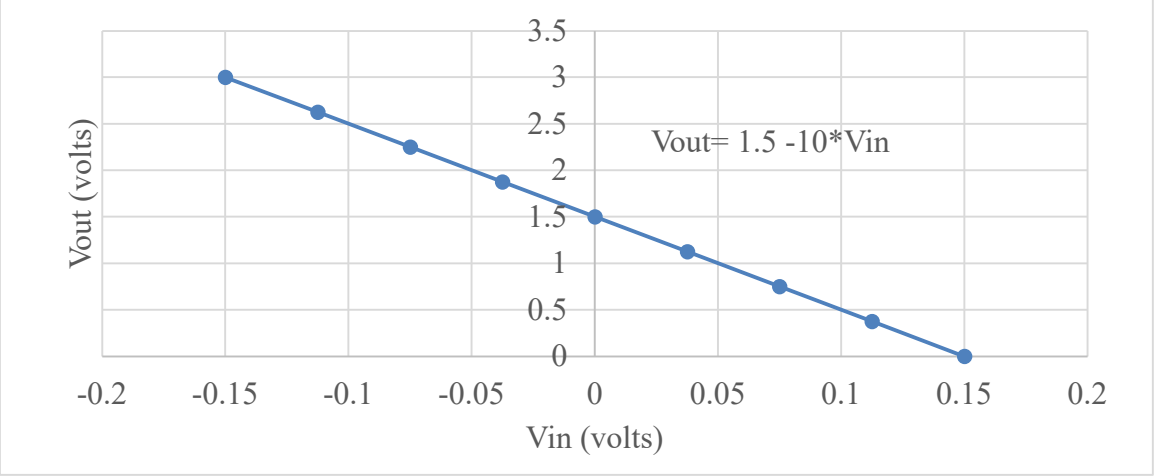
**(5) Question 10.** You have a wired communication channel (UART, CAN, Ethernet, I2C, SPI), which could run at a maximum of **F** bps. The channel allows the sender and receiver to negotiate the actual speed, which might be less than or equal to **F** bps. Give at least three reasons might you have to negotiate a slower speed than the maximum **F**?

To lower the power, save energy
To improve reliability because of poor cable quality
To reduce error rate
Because the capacitive load is causing errors
Because a long cable length causes speed of light delays
Because added EM noise is causing errors
Receiver can't handle max rate
Slower speed increases energy/packet

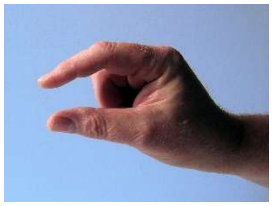
**(10) Question 11.** Derive the equation relating $V_{out}$ to $V_{in}$, show your work. You may assume $V_{out}$ remains 0 to 3.3V, and the OPA2350 is powered by 3.3V.




The reference voltage is $V_{ref}$ = 1.233*(1+33k/150k) = 1.50V
Voltage at + terminal is 1.5V*10k/110k
Voltage at – terminal also is 1.5V*10k/110k
Current through 10k is ($V_{in}$-1.5V*10k/110k)/10k
Current through 100k is (1.5V*10k/110k-$V_{out}$)/100k
Currents are equal ($V_{in}$-1.5V*10k/110k)/10k = (1.5V*10k/110k-$V_{out}$)/100k
10*$V_{in}$ -1.5V*10/11 = 1.5V*1/11 - $V_{out}$
$V_{out}$ =1.5V*1/11 – 10*$V_{in}$ +1.5V*10/11
$V_{out}$ = 1.5V – 10*$V_{in}$




**Bonus Question:** When Valvano does this, to what embedded system concept is he referring?



Time to execute all ISRs should be small and bounded.