Jonathan W. Valvano First:_____ Last:_____
November 21, 2014, 10:00-10:50am. Open book, open notes, calculator (no laptops, phones, devices with screens larger than a TI-89 calculator, devices with wireless communication). You have 50 minutes, so please allocate your time accordingly. ***Please read the entire quiz before starting***.

(5) Question 1. What is the difference between a buck-boost and a linear regulator? Pick the answer that best differentiates the two regulator types. Put your answer in the box.

A) A linear regular needs capacitors on both input and output, but the buck-boost does not need capacitors.

B) The linear regulator only creates an output voltage that is less than the input voltage, and the buck-boost only creates an output voltage that is greater than or equal to the input voltage.

C) A linear regulator can be used to create a power voltage, whereas a buck-boost is used to create a low-noise analog reference voltage for analog circuits.

D) A linear regulator does not exhibit back EMF, but a buck-boost requires a snubber diode because of the inductor in the circuit; the dI/dt in the inductor will cause a large back EMF voltage.

E) Assume the current is 1 A, the input voltage is 9 V, and the output voltage is 3.3 V. A linear regulator will get hot and a buck-boost will not get hot.

F) The linear regulator is only used for currents less than 1 A, while the buck-boost is only used for currents above 1 A.

G) A linear regulator is better for a battery powered application because the large dropout voltage allows the battery to discharge for longer before the battery voltage finally drops out of range.

E

(10) Question 2. For each application choose busy-wait synchronization or interrupt synchronization. For Specify "**BW**" for busy-wait, and specify "**Int**" for interrupts. Place your answers in the boxes.

A) With a UART transmission such that packets of 17 or more frames are to be sent at a time at unknown intervals. The baud rate is 115200 bits/sec. The protocol is 1 start bit, 8 data bits, even parity, and one stop bit.

Int

B) With an SSI interface where the microcontroller is the slave and the time between the arrival of frames is variable. The SSI clock is 10 MHz and frame size is 8 bits.

Int

C) With timer-start ADC sampling, 1 MHz ADC mode, and 64-element hardware averaging. The sampling rate is periodic (a regular rate). The system has many real-time tasks.

Int

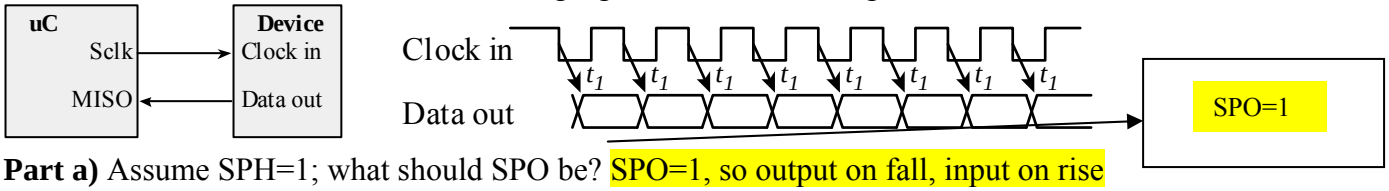
D) An interface of 24 independent input signals, where a corresponding software task is to be executed on the rising edge of each input.

Int

E) The goal is to spin six stepper motors all at a constant speed, but the speed of each motor must be independently controlled.

Int

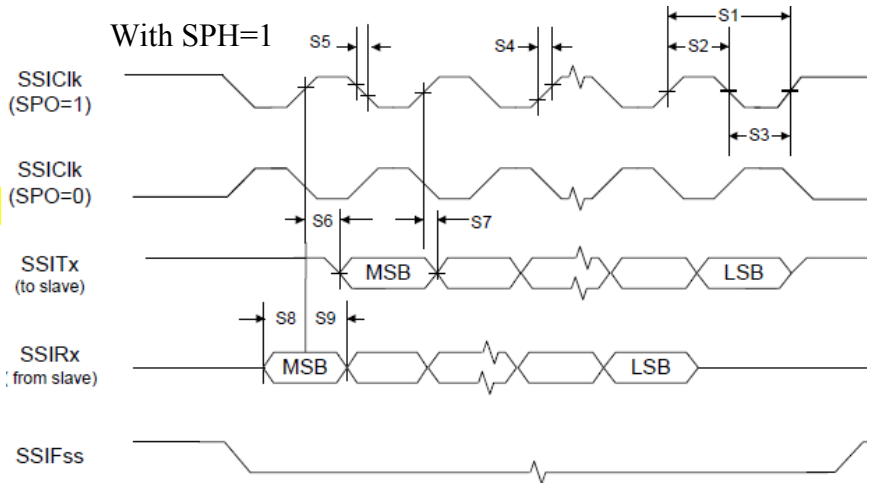
(15) Question 3. The goal is to transmit synchronous serial data as fast as possible using SSI. The external device sends data from the outside world into the microcontroller. The microcontroller is the master, and the external device is a slave. The following figure shows the timing of the external device.



Part a) Assume SPH=1; what should SPO be? SPO=1, so output on fall, input on rise

Part b) The time t_1 is [50, 400ns]. What is the shortest SSI clock period that this device can be interfaced? You may assume S4 and S5 are zero, and the clock will be 50% duty cycle. Show your work.

Let T be the SSI period,
DR = (0.5*T - S8 , 0.5*T + S9)
DR = (0.5*T – 17.15 , 0.5*T + 0)
DA = (t_1 , T + t_1),
DA = (400 , T + 50), taking worst case
So 400 ≤ 0.5*T - 17.15,
2*417.17 ≤ 0.5*T,
834.34ns ≤ T,
f_{max} = 1/834.34ns = 1.2 MHz

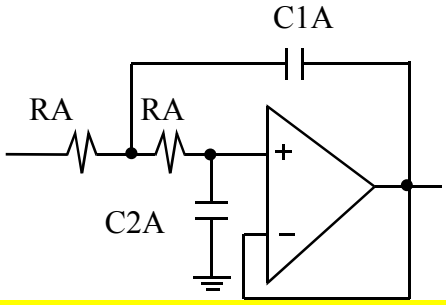


Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
S1	T_{CLK_PER}	SSIClk cycle time, as master[a]	40	-	-	ns
		SSIClk cycle time, as slave[b]	150	-	-	ns
S2	T_{CLK_HIGH}	SSIClk high time, as master	20	-	-	ns
		SSIClk high time, as slave	75	-	-	ns
S3	T_{CLK_LOW}	SSIClk low time, as master	20	-	-	ns
		SSIClk low time, as slave	75	-	-	ns
S4	T_{CLKR}	SSIClk rise time[c]	1.25	-	-	ns
S5	T_{CLKF}	SSIClk fall time[c]	1.25	-	-	ns
S6	T_{TXDMOV}	Master Mode: Master Tx Data Output (to slave) Valid Time from edge of SSIClk	-	-	15.7	ns
S7	T_{TXDMOH}	Master Mode: Master Tx Data Output (to slave) Hold Time from next SSIClk	0.31	-	-	ns
S8	T_{RXDMS}	Master Mode: Master Rx Data In (from slave) setup time	17.15	-	-	ns
S9	T_{RXDMH}	Master Mode: Master Rx Data In (from slave) hold time	0	-	-	ns

(10) Question 4. Assume Ports A, B, C and D are already initialized to interrupt on rising edges of PA7 PB7, PC7 and PD7. Also assume interrupts are armed and enabled. Write C code to set the priority so that PD7 is the highest, PC7 is the next highest, and PA7/PB7 are equal priority. Assume there are priority 3 interrupts that are less important than any of these edge-triggered interrupts,

```
NVIC_PRI0_R = 0x00204040; // D=0, C=1, A=B=2
```

(10) Question 5. Design a two-pole Butterworth low pass filter with a cutoff frequency of 51 Hz. Show your work. Specify RA, C1A, and C2A.



All you need to do is divide both capacitors by (2π51). C1A=0.44 µF, C2A=0.22 µF.

first design step is to select the cutoff		
fc (Hz)	51	fill this in
RA (kohm)	10	same as initial R
C1A (µF)	0.44126	is 141.4/(2•π•fc)
C2A (µF)	0.22063	is 70.7/(2•π•fc) or 0.5•C1A
second design step is to choose convenient Capacitor values		
fc (Hz)	51	same as previous fc
RB (kohm)	10.029	new value to match exact fc
C1B (µF)	0.44	fill this in
C2B (µF)	0.22	is 0.5•C1B
third design step is to choose a convenient resistor value		
fc (Hz)	51.1466	new cutoff based on these convenient values
RC (kohm)	10.000	fill this value in
C1C (µF)	0.44	same as C1B
C2C (µF)	0.22	same as C2B

(10) Question 6. You will use **decimal fixed-point** to implement *area* equals *width* times *length*. Assume *width* and *length* are fixed-point numbers with 0.001 cm resolution; **W** and **L** are the integer parts respectively. Assume *area* is a fixed-point number with 0.001 cm^2 resolution; **A** is the integer part of *area*. **Write C code** that calculates **A** as a function of **W** and **L**.

```
// Area = Width*Length;                          objective
// Area = A/1000; Width=W/1000; Length=L/1000; definitions
// A/1000 = W/1000 * L/1000;                     algebraic substitution
  A  = (W*L)/1000;
```

(15) Question 7. Design an analog circuit with the following transfer function $V_{out} = 2V_{in}+2$. The input is a single voltage (not differential). You may assume the input is bounded such that the output ranges from 0 to 3V ($-1 \le V_{in} \le 0.5$). R1 and R2 are already chosen such that the analog reference is 2.00V. You will use one rail to rail op amp (not an instrumentation amp). Show your work and label all chip numbers and resistor values, including R1 and R2. You do not have to show pin numbers.

$V_{out} = 2V_{in}+2$

Create a 2.00 V reference with LM4041

Vz=1.233 (1+R2/R1), (1.233 is the fixed voltage of the zener)

2 = 1.233(1+R2/R1), R2/R1= 0.622, R2=31.6kΩ, and R1=51.1kΩ

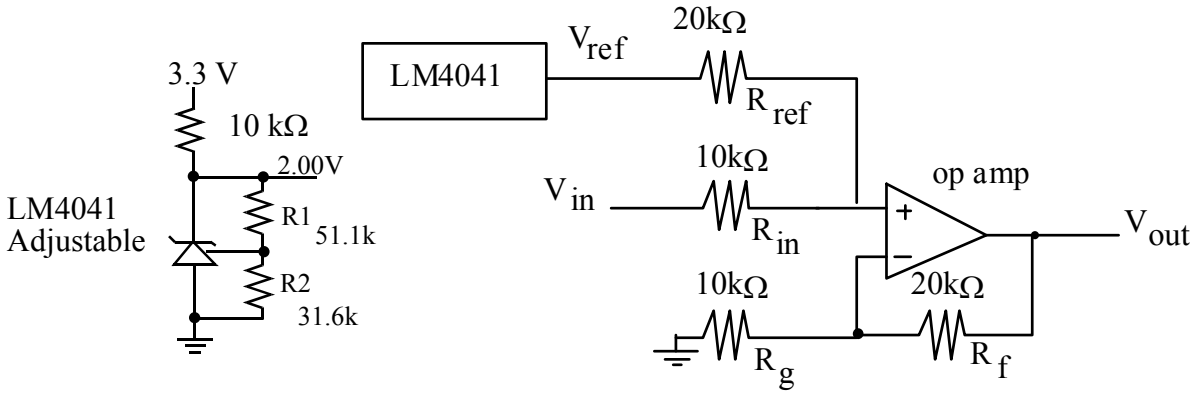
$V_{out} = 2V_{in}+V_{ref}$

Add ground gain of -2 to make all gains sum to 1

$V_{out} = 2V_{in}+V_{ref} - 2V_g$

Choose R_f to be common multiple of 1, 2 $R_f = 20k\Omega$,

Choose other resistors to create needed gains, $R_{in}=10k\Omega$, $R_{ref}=20k\Omega$, $R_g=10k\Omega$



(25) Question 8. The following code uses Timer0A to increment count on the rising edge of PB6. Edit the code so it uses Timer1A to increment count on the rising edge of PB4. You can skip the priority register.

```
volatile uint32_t Count;      // incremented on interrupt

void TimerCapture_Init(void){
  SYSCTL_RCGCTIMER_R |= 0x01;      // activate timer0
  SYSCTL_RCGCGPIO_R |= 0x00000002; // activate port B
  Count = 0;                       // allow time to finish activating
  GPIO_PORTB_DEN_R |= 0x40;        // enable digital I/O on PB6
  GPIO_PORTB_AFSEL_R |= 0x40;      // enable alt funct on PB6
  GPIO_PORTB_PCTL_R = (GPIO_PORTB_PCTL_R&0xF0FFFFFF)+0x07000000;
  TIMER0_CTL_R &= ~0x00000001;     // disable timer0A during setup
  TIMER0_CFG_R = 0x00000004;       // configure for 16-bit timer mode
  TIMER0_TAMR_R = 0x00000007;      // configure for input capture mode
  TIMER0_CTL_R &= ~(0x000C);       // TAEVENT is rising edge
  TIMER0_TAILR_R = 0x0000FFFF;     // start value
  TIMER0_IMR_R |= 0x00000004;      // enable capture match interrupt
  TIMER0_ICR_R = 0x00000004;       // clear timer0A capture flag
  TIMER0_CTL_R |= 0x00000001;      // enable timer0A 16-b, +edge timing
  NVIC_PRI4_R =(NVIC_PRI4_R&0x00FFFFFF)|0x40000000; //Timer0A=priority 2
  NVIC_EN0_R = 0x00080000;         // enable interrupt 19 in NVIC
  EnableInterrupts();
}
void Timer0A_Handler(void){
  TIMER0_ICR_R = 0x00000004;       // acknowledge timer0A capture match
  Count = Count + 1;
}
```

0x02
0x01
0x01
0xFFF0FFFF
0x00070000
TIMER1
PRI5
0xFFFF00FF
0x00004000
TIMER1
0x00200000
21