Jonathan Valvano

November 21, 2019, 3:30 to 4:45 pm. Open book, open notes, calculator (no laptops, phones, devices with screens larger than a TI-89 calculator, devices with wireless communication). You have 60 minutes for the first 85 points, so please allocate your time accordingly. ***Place your EID at the top of each page. Please read the entire quiz before starting***. We will use **Gradescope**, so you must put your answers in the boxes provided.

(10) Question 1. Consider a buck-boost regulator with 100% efficiency. Let the input voltage be V_{in}, and let the output voltage be V_{out}. Also, let the input current be I_{in}, and let the output current be I_{out}.

(5) Part a) List all the relationships that are possible for the buck-boost regulator?

A) $V_{out} = V_{in}$

B) $V_{out} > V_{in}$ for some values

C) $V_{out} < V_{in}$ for some values

A B and C

(5) Part b) Consider the following possible relationships between input and output. Let R be the equivalent resistance of the system load. Which one equation best describes the approximate power transfer relationship of the regulator.

D) $V_{out} = V_{in}$

E) $I_{out} = I_{in}$

F) $V_{out} * I_{out} = V_{in} * I_{in}$

G) $V_{out} / I_{out} = V_{in} / I_{in}$

H) $V_{out} * I_{in} = V_{in} * I_{out}$

I) $V_{out} * I_{out} = V_{in}^2 / R$

F) $V_{out} * I_{out} = V_{in} * I_{in}$

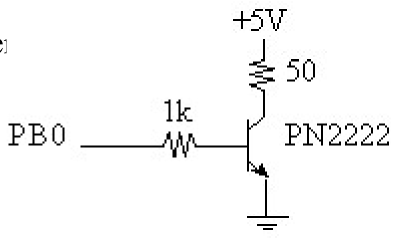
(10) Question 2. Consider this interface between the microcontroller pin PB0 and a 50-ohm speaker. You may assume

$V_{BE(sat)} = 0.8$ V,

$I_{C(max)} = 500$ mA,

$V_{CE(sat)} = 1.0$ V,

$h_{FE(max)} = 100$



Assume the PB0 is an output. In this question, the pin is high with a voltage of 3.2V. In which mode is the BJT? Choose one of these four possibilities:

A) Cutoff (off),

B) Forward active (linear),

C) Saturated (fully on), or

D) Reversed active

C) Saturated (fully on)

What is the current across the 1k resistor?

In saturation V_{BE} is 0.8V

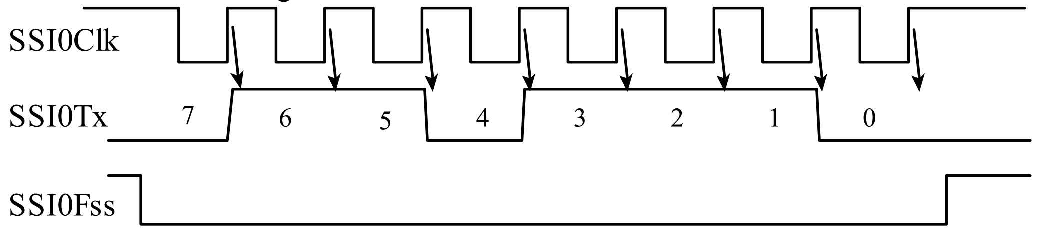
I_B = (3.2-0.8V)/1k = 2.4V/1k = 2.4mA

What is the current across the 50-ohm speaker?

I_C = (5-1.0V)/50ohms = 4V/50ohms = 80mA

Note that I_C is smaller than I_B*h_{FE} (it is in saturation)

(10) Question 3. An output device is interfaced to the microcontroller using SPI. The TM4C123 uses Freescale mode with the TM4C123 as master. The following waveforms were captured with the logic analyzer. Your task is to reverse engineer the SPI mode.



(3) Part a) What value did the software write to **DSS** during initialization?

8 bits (DSS=7)

(2) Part b) What value did the software write to **SPO** during initialization?

Idle clock is high

SPO=1

(2) Part c) What value did the software write to **SPH** during initialization?

Master Output changes on rise of clock, slave clocks on fall

SPH=0

(3) Part d) What data value is being transmitted (in hexadecimal)?

Look at data bits on falling edge

0x6E

(10) Question 4. We will store the value -8 cm with the integer -800 and store the value +8 cm with the value +800. Assuming the integer is stored as a 16-bit signed number, what are the minimum, maximum, precision and resolution of this fixed point number system? Give units for each.

Minimum value = -327.68cm	Precision = 16 bits
Maximum value = 327.67cm	Resolution of the system = 0.01cm

(10) Question 5. Assume the goal is to sample the ADC at 1000 Hz (every 1ms).

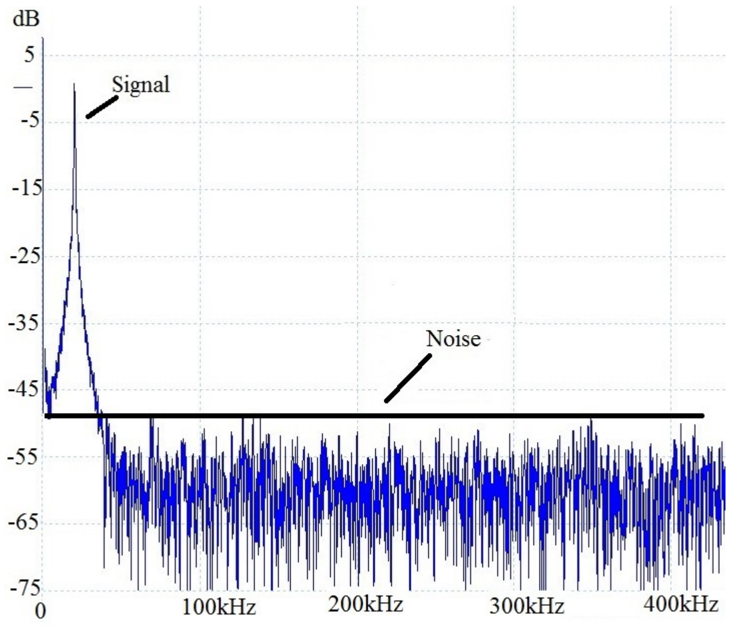
(5) Part a) Define sampling jitter in a real time data acquisition system.

Let Δt be the time between starting1 the ADC (which should be 1ms)
Jitter is the bound, δ such that 1ms- δ ≤ Δt ≤ 1ms+ δ

(5) Part b) Explain why the priority of the ADC interrupt service routine does not affect sampling jitter when using timer-triggered ADC sampling.

Since the ADC is started by the timer every 1ms, there is no jitter. As long as the ADC ISR runs before the next sample is started, there is no jitter. The data is latched into the ADC FIFO representing the conversion at the exact time desired. The timer does not interrupt, it just triggers the ADC and the ADC interrupt occurs when the sample is complete.

(15) Question 6. The system must sample a signal in the presence of noise. We are interested in accuracy so the system must be able to be calibrated. This figure shows a spectrum (dB full scale, dB_{FS} versus frequency) of the signal plus noise. The frequencies of interest in the signal vary from 0 to 50 kHz. The ADC full scale is 3.3V. The noise also exists from 0 to 100 kHz similar to the 100 to 400 kHz noise shown. The peak signal at 20 kHz is 0 dB_{FS} and the noise is below -48 dB_{FS}.



(5) Part a) Choose the sampling rate. In order to save money, we wish to choose the slowest possible sampling rate. We wish to maximize system accuracy, but minimize cost. Justify your answer.

Nyquist ½ f_s >50kHz, so f_s=100.1 kHz

(5) Part b) In order to save money, we wish to choose an ADC precision with the fewest number of bits. We wish to maximize system accuracy, but minimize cost. Justify your answer.

SNR = 0 - -48 dB = 48 dB
Compare noise to ADC resolution
48 dB = 20 $\log_{10} 2^n$, where n is the number of ADC bits
n = 8 bits

(5) Part c) Which type of ADC would you choose? Flash, resistor string, binary weighted, successive approximation, sigma delta, or R-2R? Why?

successive approximation because slow sampling rate, few number of bits, and accurate results needed; sigma delta converters are high precision but not accurate (sigma delta is 12 to 32 bits) ; flash converter is much too fast and expensive for 100 kHz; resistor string, binary weight and R-2R are DACs not ADCs

(15) Question 7. Design an analog circuit with the following transfer function: $V_{out} = 10*V_1 - 2*V_2$. You may assume inputs V_1 and V_2 are such that output range will be 0 to 3.3V. You may use any chips shown in the book or presented in class. Show your work and label all chip numbers and resistor values. You do not have to show pin numbers. You do not need to add a low pass filter. Use E24 values

10	11	12	13	15	16	18	20	22	24	27	30
33	36	39	43	47	51	56	62	68	75	82	91

Table 9.2. E24 Standard resistor and capacitor values for 5% tolerance.

No reference needed, You cannot use INA122; the INA122 is way too expensive to use in a simple problem like this, you only need one rail to rail op amp

$V_{out} = 10*V_1 - 2*V_2$

Add ground gain to get sum of gains to equal +1

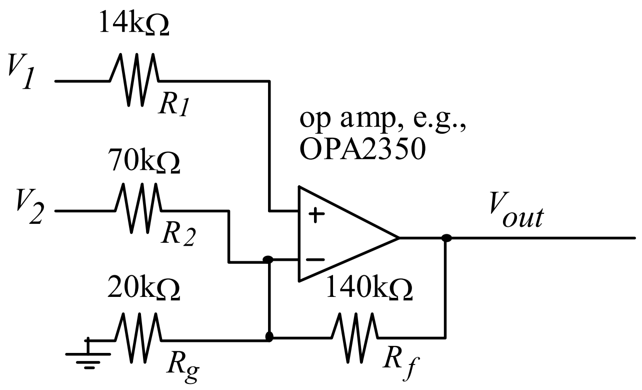
$V_{out} = 10*V_1 - 2*V_2 - 7*V_g$

Build with standard resistors

14k = 1k+13k

70k = 68k+2k

140k = 120k+20k



(5) Question 8. Consider this shunt diode, which is similar to but not identical to the LM4041C. Assume

$V_s = 5V$

$V_{ref} = 1.2V$

$R_s = 1k$

$R1 = 10k$

$R2 = 20k$

What is V_z?

$V_{ref} = V_z * R1 /(R1 + R2)$

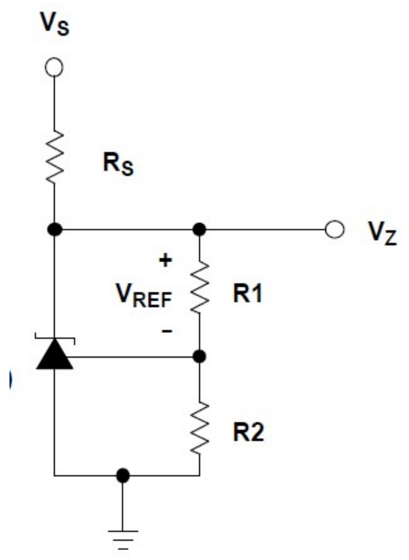
$(R1 + R2)*V_{ref} = V_z * R1$

$(1 + R2/ R1)*V_{ref} = V_z$

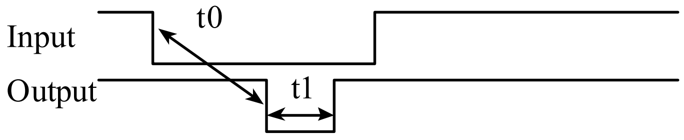
$(1 + 20/ 10)*1.2 = V_z$

$3*1.2 = V_z$

$3.6V = V_z$



(15) Question 9. There is a digital input connected to PB6, and a digital output connected to PB7. There are two global variables: **t0** and **t1**, which represent times in usec. You may assume both times are greater than 100 and less than 10000 usec. The initial values are shown below, but you should expect some other software to modify **t0** and **t1**. You must solve this problem by writing two interrupt service routines and have NO backward jumps. You may assume the PLL is active such that the bus clock period is 12.5 ns. You may use the global **Count** however you wish, but you cannot write to **t0** and **t1**.



```
uint32_t t0=200; // desired time from fall of PB6 and fall of PB7 (us)
uint32_t t1=400; // desired pulse width of PB7 (us)
uint32_t Count;  // you may use this however you wish
```

This initialization is called once. PB6 input capture interrupts on falling edge (IRQ 19). PB7 is a simple GPIO output. Timer 2 is initialized in one-shot interrupt mode, but initially disabled (IRQ 23). You do not need to show the main program. Software must write to GPIO_PORTB_DATA_R to change PB7.

```
void System_Init(void){
  SYSCTL_RCGCGPIO_R |= 0x02;    // activate clock for Port B
  SYSCTL_RCGCTIMER_R |= 0x05;   // activate timer0 and timer2
  Count = 0;
  GPIO_PORTB_DIR_R |= 0x80;     // enable digital out on PB7
  GPIO_PORTB_DEN_R |= 0xC0;     // enable digital on PB7 and 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 |= 0x0004;       // TAEVENT is falling 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
  NVIC_PRI4_R =(NVIC_PRI4_R&0x00FFFFFF)|0x00000000; // Timer0A=pri 0
  NVIC_EN0_R = 0x00080000;      // enable interrupt 19 in NVIC
  TIMER2_CTL_R = 0x00000000;    // disable timer2A during setup
  TIMER2_CFG_R = 0x00000000;    // configure for 32-bit mode
  TIMER2_TAMR_R = 0x00000001;   // configure for one-shot mode
  TIMER2_TAILR_R = 79999;       // reload value, 1 ms
  TIMER2_TAPR_R = 0;            // 12.5 ns bus clock resolution
  TIMER2_ICR_R = 0x00000001;    // clear timer2A timeout flag
  TIMER2_IMR_R = 0x00000001;    // arm timeout interrupt
  NVIC_PRI5_R = (NVIC_PRI5_R&0x00FFFFFF)|0x00000000; // Timer2A=pri 0
  NVIC_DIS0_R = 1<<23;          // disable IRQ 23 in NVIC
  TIMER2_CTL_R = 0x00000001;    // enable timer2A
  EnableInterrupts();
}
```

No backward jumps or time delays are allowed in the ISRs. These are the only priority 0 ISRs.

Show the Timer 0 input capture interrupt service routine.

```
void Timer0A_Handler(void){ // called on fall of PB6
  TIMER2_TAILR_R = 80*t0-25; // first wait time (25 cycle offset)
  Count = 0; // first wait
  TIMER2_ICR_R = 0x00000001; // clear timer2A timeout flag
  NVIC_EN0_R = 1<<23;        // enable IRQ 23 in NVIC
  TIMER0_ICR_R = 0x00000004; // ack timer0A capture flag
}
// 25 cycles to finish this ISR and start the next ISR
```

Show the Timer2 interrupt service routine.

```
void Timer2A_Handler(void){ // called when timer 2A counts to 0
  if(Count == 0){
    GPIO_PORTB_DATA_R &= ~0x80; // PB7 low
    Count = 1;
    TIMER2_TAILR_R = 80*t1-25;  // second wait time (offset)
  }else{
    GPIO_PORTB_DATA_R |= 0x80;  // make PB7 high
    NVIC_DIS0_R = 1<<23;        // disable IRQ 23 in NVIC
  }
  TIMER2_ICR_R = 0x00000001;    // ack timer2A timeout flag
}
// 25 cycles to finish this ISR and start this ISR again
```