Jonathan W. Valvano First:_____ Last:_____

(2) Question 1. Give an important mechanism **Bluetooth low energy (BLE)** uses to achieve low energy. It goes into low power sleep mode for most of the time, lower bandwidth, shorter range.

(2) Question 2. Explain why communication channels like RS485, Ethernet, CAN and USB use **differential drive outputs**? To remove the effect of EMI noise, improve CMRR. For differential drive, added noise becomes common mode and the CMRR removes it.

(2) Question 3. Briefly explain how a **boost convertor** uses an inductor to operate. DC->AC->DC using an LC oscillator, using a charge pump. The basic idea is the LC circuit increases voltage.

(2) Question 4. Why do we **not** use tantalum capacitors for analog high pass filters? I.e., which capacitor parameter determines why we use ceramic rather than tantalum capacitors? Resistance leakage (ESR)

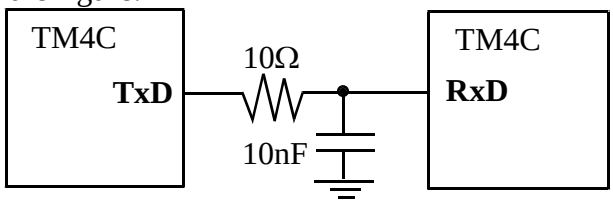
(2) Question 5. You are CEO of a company. Why would you have employees sign a "**non-compete clause**" (NCC) as a requirement for employment? So if employees quit or are fired, they will not use your technology at the competition. This is not really a conflict of interest.

(2) Question 6. Why does a state of the art cell phone employ **modular** design? Modular design allows for **complexity.** Think about how a cell phone fundamentally differs from an EE445L lab.

(2) Question 7. Briefly define DAC **monotonicity**. Increase in digital input always causes an increase in analog output

(5) Question 8. You are asked to design a linear analog amplifier using rail-to-rail components for an embedded system powered with a single 3.3V power. You want it to be both effective (implement the desired transfer function) and to be low cost. What criteria in the problem statement would lead you to choose an instrumentation amp like the INA122 ($6.32) over an op amp like the OPA2350 ($4.23)?
It must have a **differential input** to use an instrumentation amp. Then, we also have one specification of good CMRR, high gain, or low noise

(5) Question 9. This problem addresses the issue of using a 100-foot cable to implement UART serial communication between two microcontrollers. To implement simplex communication, the cable has two wires: signal and ground. The transmit UART pin of one TM4C123 is connected via the 100-foot cable to the receive UART pin of a second TM4C123. As the cable increases in length so will the effective resistance and capacitance of the cable. This low quality cable has a capacitance about 100 pF/foot; so we will assume this 100-ft cable has an effective 10-nF capacitance between signal and ground. 30-gauge wire has about 0.1Ω/foot; so we will assume this 100-ft cable has an effective 10-Ω resistance between output and input, as shown in the figure.



Estimate the fastest baud rate that can be implemented reliability. Show both your work and your estimate of maximum baud rate in bits/sec. time constant is 10Ω *10nF = 100ns, we pick a baud rate about 10 times slower than time constant. So the baud rate should be 1 Mbps or slower.

(8) Problem 10. Consider the following SysTick interrupting system with its corresponding assembly code. You may assume SysTick interrupts occur slowly enough that SysTick will not attempt to interrupt itself. The listing includes absolute addresses. ROM exists from 0x00000000 to 0x0003FFFF. **`Count`** is a 32-bit variable in RAM. There are one or more critical sections.

Part a) Give the exact ROM locations of the critical section(s). For example, if you were to answer 0x04C4-0x04C8, then you would mean a mistake could happen if the interrupt occurred between the **`PUSH`** and **`MOVS`** instructions, or between the **`MOVS`** and **`BL`** instructions of the SysTick handler. Look for the read modify write to the shared global, so the critical section is 0x03C6 to 0x3CC
You could make the section larger. You could define it as 0x0518 to 0x0522 in main loop

```
volatile int32_t Counts = 0;



void static Add(int32_t n){
  Counts = Counts + n;
}

void SysTick_Handler(void){
  Add(1);
}

int main(void){
  Init(); // includes SysTick_Init
  EnableInterrupts();
  while(1){
    Add(-1);
  }
}
```

```
EnableInterrupts:
0x00000324       CPSIE I
0x00000326       BX    lr

DisableInterrupts:
0x00000328       CPSID I
0x0000032A       BX    lr


Add:
0x000003C4       LDR   r1,=Count
0x000003C6       LDR   r1,[r1,#0x00]
0x000003C8       ADD   r1,r1,r0
0x000003CA       LDR   r2,=Count
0x000003CC       STR   r1,[r2,#0x00]
0x000003CE       BX    lr

SysTick_Handler:
0x000004C4       PUSH  {lr}
0x000004C6       MOV   r0,#0x01
0x000004C8       BL    Add
0x000004CC       POP   {pc}

main:
0x00000510       BL    Init
0x00000514       BL    EnableInterrupts
0x00000518       MOV   r0,#0xFFFFFFFF
0x0000051E       BL    Add
0x00000522       B     0x00000518
```

Part b) How would you solve this critical section? Your solution must maintain overall function of the system, such that **`Count`** will be the difference between ISR executions and main loop executions.

A) Move **`Count`** to be a local variable of the function **Add**
B) Disarm SysTick and rearm SysTick around the critical section(s)
C) Remove the **`volatile`** designation on **`Count`**
D) Remove the **EnableInterrupts** call from **main**
E) Add **DisableInterrupts** at beginning and **EnableInterrupts** at end of **SysTick_Handler**
F) Remove the **`static`** designation from the function **Add**
G) None of the above will remove the critical section

(5) Question 11. Consider three different DAC techniques: resistor string, R-2R ladder and binary weighted. Pick the DAC technique that best answers each question. Place an **RS** for resistor string, an **R-2R** for R-2R ladder, or a **BW** for binary weight. If two answers are ok, list both. If no answer is correct, list **none**. BW is the way we made the DAC in EE319K, 12-bit RS has 4096 resistors, 12-bit R-2R DAC has 12 R resistors and 14 2R resistors

A) Which is best for high-precision DAC (12 bits and over)? ---------- RS or R-2R

B) Which is best for low-precision, low-cost high-frequency sampling? ---------- BW or R2R

C) Which is used in the TLV5616 (Lab 5)? ---------- RS

D) Which has a cost/complexity linearly related to the number of bits? ---------- R-2R, BW

E) Which has a cost/complexity exponentially related to the number of bits? ---------- RS

(8) Question 12. You will use **binary fixed-point** to implement *area* equals *width* times *length*. Assume *width* and *length* are fixed-point numbers with 2^{-8} cm resolution; **`W`** and **`L`** are the integer parts respectively. Assume *area* is a fixed-point number with 2^{-10} cm^2 resolution; **`A`** is the integer part of *area*. Write **C code** that calculates **`A`** as a function of **`W`** and **`L`**. Minimize dropout, ignore overflow.

area = A*2^{-10} cm^2 *width* = W*2^{-8} cm *length* = L*2^{-8} cm

A*2^{-10} cm^2 = W*2^{-8} cm * L*2^{-8} cm

A/1024 = W/256 * L/256

A= W* L*2^{-8}*2^{-8}*2^{10}

```
A = (W*L)/64;
```

(5) Problem 13. The SysTick background thread run at 22 kHz. The ADC is set to 125 kHz. The FIFO is created with the macro in Program 3.10 from the book.

How would you characterize the system implemented above? List all that apply

A) The FIFO fills up and ADC will be lost

B) The use of the FIFO represents nonreentrant code (no code is reentered)

C) The FIFO will become empty and some DAC outputs will be skipped

D) The system is an example of Round Robin execution

E) ADC input is real time (not really real time because of the interrupt execution)

F) DAC output is real time

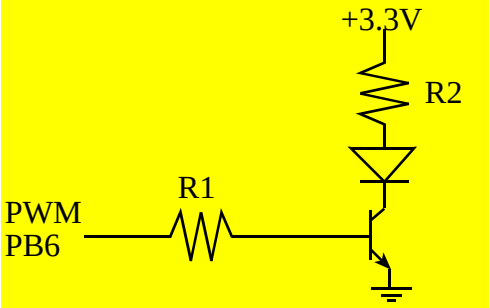
G) System has a critical section

(20) Question 14. The overall goal of this problem is to design a system that sinusoidally oscillates a high-power LED at 1 Hz. In other words, the LED goes bright-dim-bright once a second. You must use the following table to implement 32 different brightness's.

```
const uint32_t Table[32]={98,97,94,90,84,77,68,59,50,41,32,23,16,
  10,6,3,2,3,6,10,16,23,32,41,50,59,68,77,84,90,94,97};
```

(5) Part a) Show the hardware interface between the microcontroller and the LED. Full brightness occurs at 1.5V 500 mA. The 3.3V is the only power. You may use any pin(s) on the TM4C123. Label all parts and give resistor and capacitor values. Please show your work. Choose either PN2222 or TIP120. Assume PN2222 has h_{fe}=100, V_{be}=0.6V, V_{ce}=0.3V. Assume TIP120 has h_{fe}=1000, V_{be}=1V, V_{ce}=0.7V.

Must choose TIP120 in order to get 500mA Ice. Voltage across R2 will be 3.3-0.7-1.5= 1.1V, current is 500 mA, so R2 = 1.1V/0.5A = 2.2Ω. Voh=3.3V, hfe=1000, Ib = Ice/hfe=500mA/1000 = 0.5mA. R1<(Voh-Vbe)/Ib =2.3V/0.5mA = 4.6kΩ. Choose R1 smaller than that, such as 1kΩ.



If you were to try to do this with a DAC, you would need to run the TIP120 in linear region. It would be almost impossible to match the nonlinearity of LED brightness with the nonlinearity of the TIP120 circuit. PWM with saturated TIP120 gives precise and linear control over brightness.

(15) Part b) Show all software required to run the system. The body of the main MUST be a do-nothing while loop. You may call any function listed in the book without showing the body of the function.

```
uint32_t index=0;
void Background(void){      // runs at 32 Hz
  PWM0_Duty(Table[index]);  // Program 6.8
  index = (index+1)&0x1F;   // 0 to 31
}
void main(void){ // runs at 16 MHz
  // programs 6.6 and 6.8
  PWM0_Init(100,98); // PB6 PWM output, 16MHz/2/100 = 80 kHz, 98% duty
  Timer2A_Init(&Background, 16000000/32); // 32 Hz periodic interrupt
  while(1){
  }
}
```

(15) Question 15. Design a system with an analog input and a digital output. The input impedance must be larger than 1 MΩ. The input range is 0 to 3.3V. If the input is less than 1V, the output is digital low (about 0V). If the input is greater than 1 and less than 2V, the output is digital high (about 3.3V). If the input is greater than 2V, the output is digital low (about 0V). If you use a chip not in the book, please describe its behavior. Label all chips and resistors. You may use any software function listed in the book without showing the body of the function. Show your work.

Hardware approach (OP2350 ($4.23)+PN2222($0.21) costs $4.41

1) Create reference voltages for 1V 2V using a resistor divider from 3.3V

2) Use one op amp. Tie input to - and 1V to + input of op amp. So V1 op amp output is high if input is less than 1 volt.

3) Use a second op amp. Tie input to + and 2V to + input of op amp. So V2 op amp output is high if input is greater than 2 volts.

4a) Set circuit output to 0 if either V1 or V2 is high (output to 3.3V if both V1 V2 are low). Use a PN2222 with two 1k resistors into base from V1 V2. Connect PN2222 emitter to ground. Connect 10k resistor from PN2222 collector to 3.3V. Output is PN2222 collector.

4b) Connect V1 and V2 to the inputs of a 2-input NAND gate.

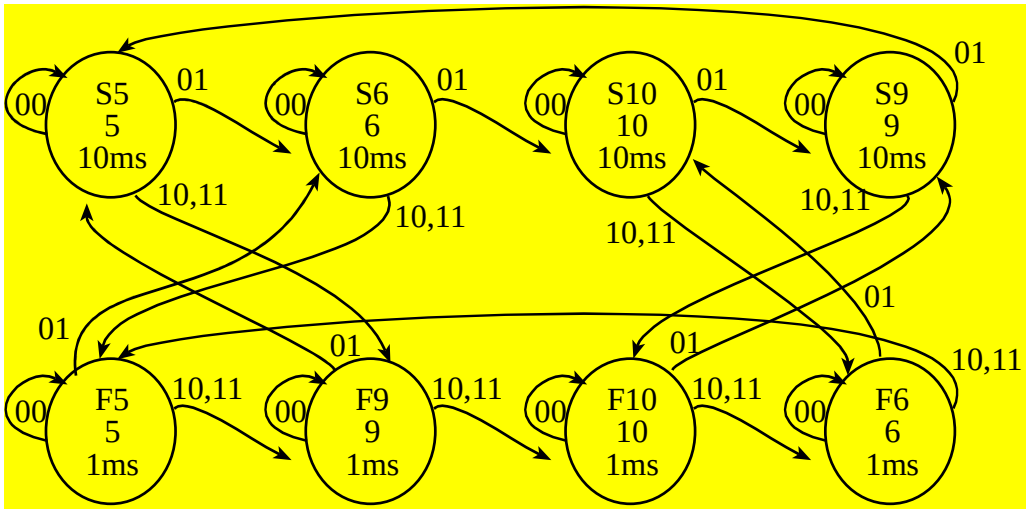
Software approach (a microcontroller like the MSP430 costs less than $2)

1) Hardware: Connect input to PD3 ADC of microcontroller and output to a GPIO PD0

2) Software: initialize ADC and GPIO output. Main loop samples ADC and implements the desired function.

```
#define T1 (40950/33)  // 1V
#define T2 (2*T1)      // 2V
int main(void){
  PLL_Init();   // 80 MHz
  Fifo_Init();
  SYSCTL_RCGCGPIO_R |= 0x08;         // 1) activate clock for Port D
  while((SYSCTL_PRGPIO_R&0x08) == 0){};// ready?
  GPIO_PORTD_DIR_R |= 0x01;          // PD0 is an output
  GPIO_PORTD_AFSEL_R &= ~0x01;       // regular port function
  GPIO_PORTD_AMSEL_R &= ~0x01;       // disable analog on PD0
  GPIO_PORTD_PCTL_R &= ~0x0000000F; // PCTL GPIO on PD0
  GPIO_PORTD_DEN_R |= 0x01;          // PD0 is enabled as a digital port
  ADC0_InitTimer0ATriggerSeq3PD3(80000); // PD3 input, 1000 kHz
  while(1){
   while(Fifo_Get(&data)==0){};      // wait for data
   if(data<T1)||(data>T2){
    GPIO_PORTD_DATA_R &= ~0x01;      // make PD0 low
   }else{
    GPIO_PORTD_DATA_R |= 0x01;       // make PD0 high
   }
}
```

(10) Problem 16. Draw a Moore FSM graph that has two inputs and four outputs. The machine will control a bipolar stepper motor with 100 steps/revolution. The sequence {5, 6, 10, ...} rotates clockwise. If the input is 00, the motor should stop. If the input is 01, the motor should spin clockwise at 1 rps. If the input is 10 or 11 the motor should spin counterclockwise at 10 rps. The controller pattern will be output, input, next. No software is required, just the FSM graph. Use good state names to clarify operation.
1rps is 100 steps per second, or 1 step every 10ms 10rps is 1000 steps per second, or 1 step every 1ms
The trick is to maintain the 5,6,10,9 or 9,10,6,5 sequence as flipping from one loop to the other



(5) Problem 17. Create two MACROs that implement minimally-intrusive debugging instruments. Assume PA7 is already initialized as an output. One MACRO sets PA7 and the other clears it. Show the C code. For full credit your solution must be friendly and have no critical sections. Running at 80 MHz, estimate the intrusiveness of these instruments.

```
#define PA7 (*((volatile uint32_t *)0x40004200))
#define Debug_Set()   (PA7 = 0x80)
#define Debug_Clear() (PA7 = 0x00)
```

The instrument takes 3 instructions and 5 cycles, at 12.5ns/cycle, this means 62.5ns

```
  LDR R0,=0x40004200
  MOV R1,#0x80
  STR R1,[R0]
```