Jonathan W. Valvano Solution

This is the closed book section. You must put your answers in the boxes. When you are done, you turn in the closed-book part and can start the open book part.

(5) Question 1. You are sampling an analog signal with an ADC. Give the equation that relates sampling jitter to measurement error.

Error = slew rate*jitter

(5) Question 2. Give the equation that capacity of a communication channel given the carrier frequency, W, and the signal to noise ratio, SNR. The units of W are Hz, and the units of SNR is watts/watts.

Shannon Channel Capacity $C = W \log_2 (1 + SNR)$ in bits/sec

(5) Question 3. Consider the differences between a buck-boost and a linear regulator. Pick the answer that best differentiates the two regulator types. Put your answer in the boxes on page 1. Place a **BB** for buck-boost and an **L** for linear.

A) Which regulator has lower noise?	L
B) Which regulator can have an output voltage larger than its input voltage?	BB
C) Which regulator type is in a cell phone?	BB
D) Which regulator type is more efficient if V_{in} is much bigger than V_{out}?	BB
E) Which regulator type did you use in your project?	L

(5) Question 4. Assume you are a practicing engineer building products for a company. Which of the following must you do before incorporating software written by others into one of your products?

A) Get permission from the author. You must give appropriate credit, provide a link to the author, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the author endorses you or your use. You are responsible for testing the software.

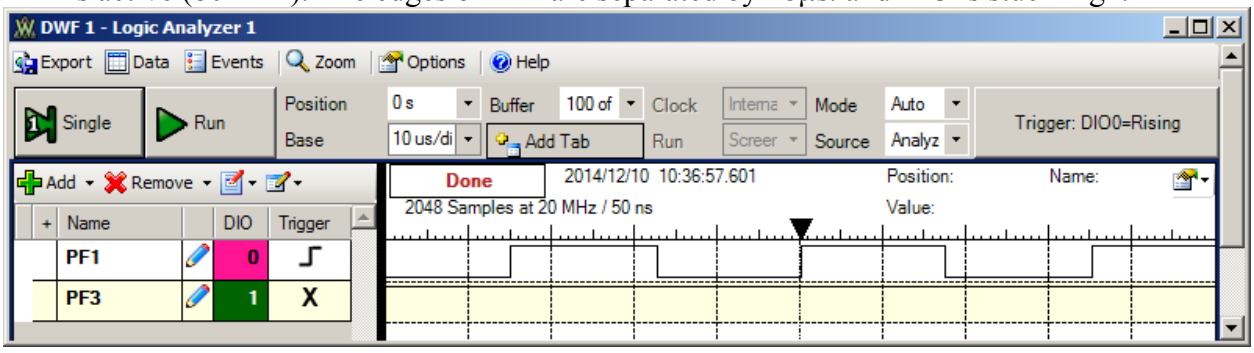
B) You can search for patent protection. If the theory is not protected by IP, then you could create your own implementation based on the theory. You are responsible for testing the software.

C) If the theory is protected by IP, then you could purchase rights to the theory and then implement your own version based on the theory. You are responsible for testing the software.

D) All of the above are ethically sound.

D

(10) Question 5. The goal of this system is to test a new DAC, which I invented, by creating a sine wave with a period of 2.56ms using a SysTick periodic interrupt. The **`SineWave`** buffer has 256 entries per cycle. **`NVIC_ST_RELOAD_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. This is actual data measured on an actual TM4C123, running this program. The PLL is active (80MHz). The edges of PF1 are separated by 18µs. and PF3 is stuck high.




```
void SysTick_Handler(void){ uint32_t static I=0;
  PF1 ^= 0x02;            // toggle PF1, profiling
  DAC_Out(SineWave[I]);   // Output one point using my new DAC
  I = (I+1)&0xFF; }
int main(void){
  PLL_Init();             // bus clock at 80 MHz
  PortF_Init();           // PF3 and PF1 are outputs
  SysTick_Init();         // 10us interrupt
  EnableInterrupts();
  while(1){
    PF3 ^= 0x08;}}        // toggle PF3, profiling
```

(5) Part a) Which of the following changes would you make to fix the error in this DAC test?

A) Increase **`NVIC_ST_RELOAD_R`** so the SysTick interrupts occur less frequently.

B) Decrease **`NVIC_ST_RELOAD_R`** so the SysTick 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) Add a software acknowledgement to the **`SysTick_Handler`**

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 **`NVIC_ST_RELOAD_R`** to 1599, and change **`0xFF`** to **`0x7F`**.

H) This DAC is really bad, throw it away and invent a faster DAC!

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

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

A or G or H

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

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

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

E

(10) Question 6. Develop a software module in C to implement a FIFO queue with these specifications:

1) Three 16-bit halfwords are allocated in RAM to store data in the FIFO
2) One 8-bit counter is allocated in RAM to store the number of elements: 0, 1, 2, 3
3) If there is one element in the FIFO, it is stored in **`Buf[0]`**
4) If there are two elements, the oldest is in **`Buf[0]`** and the newest in **`Buf[1]`**
5) If there are three elements, the oldest is in **`Buf[0]`** and the newest is in **`Buf[2]`**

The following code defines the FIFO in RAM. You cannot make changes or additions to the way in which these variables are defined. *You can NOT add more variables or change the prototypes.*

```
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
```

Part a) Write C function to initialize the FIFO.

```
int Fifo_Init(void){
  Size = 0;
  return 0;
}
```

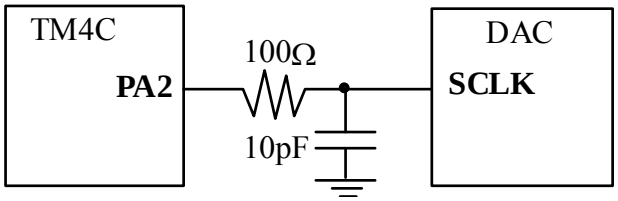
Part b) Write C function that puts one 16-bit element into the FIFO. The input parameter is call by value, and the return parameter is return by value. The return parameter of 0 means success, and 1 means the data was not stored because there were already three elements in the FIFO.

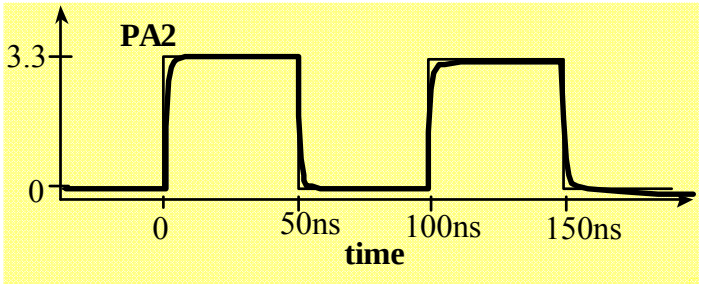
```
int Fifo_Put(uint16_t data){
  if(Size>=3) return 1; // full
  Buf[Size] = data;
  Size++:
  return 0;
}
```

Part c) Write C function that gets one 16-bit element from the FIFO. Use call by reference to return the data retrieved from the FIFO. The other return parameter is returned by value: 0 means success, and 1 means the data was not removed because there were no elements in the FIFO at the time of the call.

```
int Fifo_Get(uint16_t *p){
  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
}
```

(10) Question 7. This problem addresses the issue of capacitive loading on a high-speed serial transmission line like SSI. The SSI port of a TM4C123 is connected via a long cable to a DAC. We will model this cable as a single resistor in series with a capacitor, as shown on the left figure below. Consider a 3.3-V 10 MHz clock from the microcontroller to the DAC. The figure on the right plots the output of the microcontroller, labeled PA2.





(5) Part a) Assume the SCLK has been low while the SSI has been idle and the clock begins to oscillate at time 0, as data is being transferred at 10 MHz. Develop an equation for the SCLK input at the DAC as a function of time for the time-region 0 to 50 ns. Use the equation to make a rough guess (without a calculator) about the voltage of the DAC input at time equals 50ns.

Initially, the C is a short, so $V_{out}(0) = 0$. At infinite time, the C is an open so, $V_{out}(\infty) = 3.3V$.
$I = C\, dV_{out}/dt$, and $I = (3.3-V_{out})/R$
$RC\, dV_{out}/dt + V_{out} = 3.3$
General solution of a linear differential equation is $V_{out}(t) = A + Be^{-t/\tau}$,
Since $V_{out}(\infty) = 3.3V$, $A = 3.3$
Since $V_{out}(0) = 0$, $B = -3.3$
Since $RC(dBe^{-t/\tau}/dt) + Be^{-t/\tau} = 0$, $\tau = RC$
Time constant τ is $100\Omega * 10pF = 1ns$, so the time constant τ is small compared to pulse time.
$V_{out}(t) = 3.3-3.3e^{-t/\tau}$ for 0 to 50 ns

(3) Part b) On the above right figure add a rough sketch of the voltage versus time as seen at the input of the DAC for times 0 to 150 ns.

Because 1ns is much less than 50ns, SCLK will look identical to PA2

(2) Part c) Does this capacitance have any significant effect on operation of this SSI interface at 10 MHz? Answer *No* if there is little effect and the SSI interface should work. Answer *Yes* if there is a large effect and the SSI interface will not work.

No

(5) Question 8. You will use **binary fixed-point** to implement *perimeter* equals 2**width* plus 2**length*. Assume *width* and *length* are fixed-point numbers with 2^{-12} cm resolution; **W** and **L** are the integer parts respectively. Assume *perimeter* is a fixed-point number with 2^{-11} cm resolution; **P** is the integer part of *perimeter*. **Write C code** that calculates **P** as a function of **W** and **L**. Do not worry about overflow.

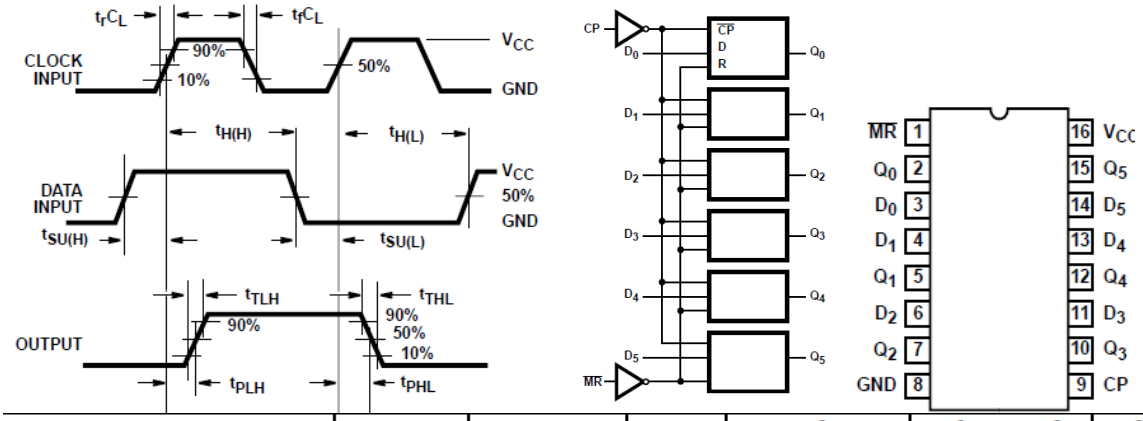
Perimeter = **P***2^{-11} cm; *Width* = **W***2^{-12} cm; *Length* = **L***2^{-12} cm
Perimeter = 2**Width* +2**Length*
P*2^{-11} cm = 2*(**W***2^{-12} cm) +2*(**L***2^{-12} cm)

```
P = W+L;
```

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 on these pages. Please don't turn in any extra sheets.

(10) Question 9. The 74HCT174 is a hex D flip-flop, which stores 6 bits of data on the rising edge of its clock input. Assume we are running at 4.5V V_{cc} power and the temperature is 25 C.



PARAMETER	SYMBOL	TEST CONDITIONS	V_{CC} (V)	25°C MIN	25°C MAX	-40°C TO 85°C MIN	-40°C TO 85°C MAX	-55°C TO 125°C MIN	-55°C TO 125°C MAX	UNITS
Clock Pulse Width	t_w	-	4.5	20	-	25	-	30	-	ns
$\overline{MR}$ Pulse Width	t_w	-	6	25	-	31	-	38	-	ns
Setup Time, Data to Clock	t_{SU}	-	4.5	16	-	20	-	24	-	ns
Hold Time, Data to Clock	t_H	-	6	5	-	5	-	5	-	ns
Removal Time, $\overline{MR}$ to Clock	t_{REM}	-	4.5	12	-	15	-	18	-	ns
Clock Frequency	f_{MAX}	-	6	25	-	20	-	17	-	MHz
Propagation Delay, Clock to Q	t_{PLH}, t_{PHL}	C_L = 50pF	4.5	-	40	50		60		ns
		C_L = 15pF	5	17	-	-		-		ns
Propagation Delay, $\overline{MR}$ to Q	t_{PLH}, t_{PHL}	C_L = 50pF	4.5	-	44	55		66		ns
		C_L = 15pF	5	18	-	-		-		ns
Output Transition Times	t_{TLH}, t_{THL}	C_L = 50pF	4.5	-	15	19		22		ns
Input Capacitance	C_{IN}	-	-	-	10	10		10		pF

Part a) From the perspective of data input (D0) write a timing equation for the **data required** interval? You do not need to fill in numbers, but can leave the equation with variable names from the table.

DR = ($\uparrow$CP-T_{SU}, $\uparrow$CP+T_H)

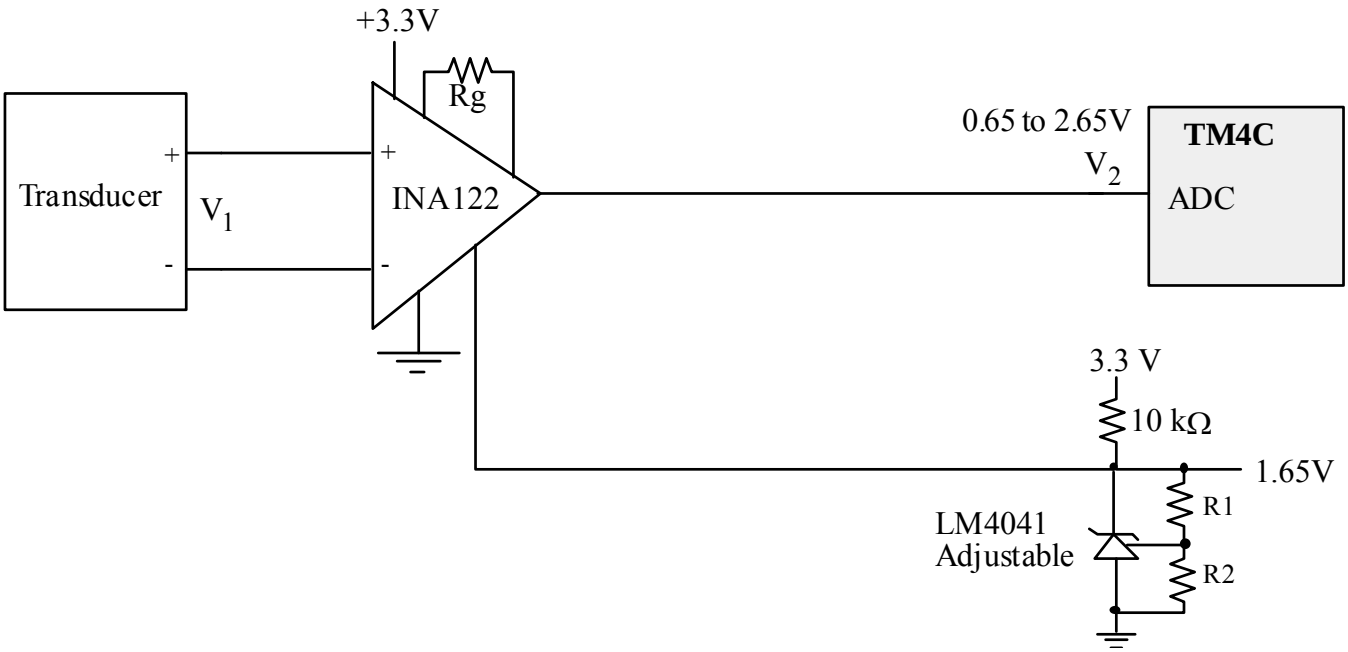
Part b) From the perspective of data output (Q0) write a timing equation for the **data available** interval?

DA = ($\uparrow$CP+T_{pLH}, $\uparrow$CP+ T_{pHL})

(5) Question 10. What is the **diameter** of the Lab 10 configuration? Lab 10 hardware included an unencrypted access point (not connected to the internet), and two TM4C123/CC3100 systems. I don't need accurate numbers, but you should estimate the diameter approximately.
About the size of a house 20 to 30 meters

(10) Question 11. Interface this transducer to the ADC. Each of the voltages out of the transducer is about 1V relative to ground. The information is encoded as the difference between the plus and minus terminals of the transducer. The differential output of this transducer ranges from -0.1 to +0.1V, in other words, $-0.1 \le V_1 \le +0.1$. One of the tricks in creating a linear and high-accuracy system is avoiding the extremes of the analog circuits including the ADC. In this system map the analog signal to 1.65±1V, in other words, make $0.65 \le V_2 \le 2.65$. Low pass filters are usually needed, but in this question you need not add an antialiasing low pass filter. Show all resistors, capacitors, and chip numbers. The available power supply voltage is 3.3V. Assume R1 and R2 are already chosen to achieve a reference of 1.65V.

$V_2 = 10V_1 + 1.65$ so Rg = 200k/(G-5) = 200k/5 = 40k



(10) Question 12. One of the limitations of Lab 10D is three of the four numbers of the IP address were hard coded. For example, "101<What is an interrupt?" meant the IP address was 192.168.0.101. Expand this capability so you could encode/transmit/decode the entire 4-number IP address.

Part a) Design a new protocol that supports the transmitting the full 4-number IP address. Illustrate your protocol by showing the payload when sending these messages. You may assume the message remains an ASCII string, like Lab 10D.

Just send the IP address as the first 4 bytes little endian

192.168.0.101	"Interrupt?"	0x65,0x00,0xA8,0xC0, "Interrupt?",0
192.7.10.1	"Can I have an A?"	0x01,0x07,0x0A,0xC0, "Can I have an A?",0
0.0.0.0	"Ping"	0x00,0x00,0x00,0x00, "Ping",0

Part b) Write a C function that accepts the payload and returns the IP address in 32-bit form. For example if the IP address is 192.168.0.101, your function should return **0xC0A80065.** Be careful and clear about endianness. Assume you already have the UDP payload received in a buffer such as

```
#define BUF_SIZE  256
UINT8 uBuf[BUF_SIZE];  // payload
uint32_t GetIP(void){
  return (uint32_t)BUF[0];
}
```