Jonathan W. Valvano First Name: _______________ Last Name:____________________
February 15, 2008, 10:00 to 10:50am

(30) Question 1. Consider a file system that uses a **file translation table** (**FTT**)

(10) Part a) There is no external fragmentation. If there are n free blocks at any locations on the disk, they all can be used to define a file of size n-1 blocks (1 block for the FTT and n-1 blocks for data). Space used for the directory and the FTTs is internal fragmentation.

(5) Part b) Assume a file has **n** data blocks. It takes one *block read* to fetch the **FTT**.

Average = 1 more read

Maximum = 1 more read (*this is a fast method to read when access is random*)

(5) Part c) Consider the linked allocation scheme described in Lab 25.

On average it takes n/2 block reads to access the data

The worst case is the byte is in the last block, requiring n block reads to access the data

(10) Part d) Write a C function that returns a byte from a file at a random location. The important part of this solution is to divide the address into two parts.

```
unsigned char eFile_Read(unsigned char numFTT, unsigned short address){
unsigned char msb,lsb,blocknum;
  eDisk_ReadBlock(FFTbuf,numFTT);
  msb = (unsigned char)address>>8;      // top part of address
  lsb = (unsigned char)address&0x00FF; // bottom part of address
  blocknum = FTTbuf[msb];              // address translation
  eDisk_ReadBlock(Databuf, blocknum);
  return Databuf[lsb];
}
```

(20) Question 2. Choose the largest block size so that the average internal fragmentation is less 5%.
First, approximate by ignoring the 16-byte overhead

Wasted space is half a block $0.5*2^n$

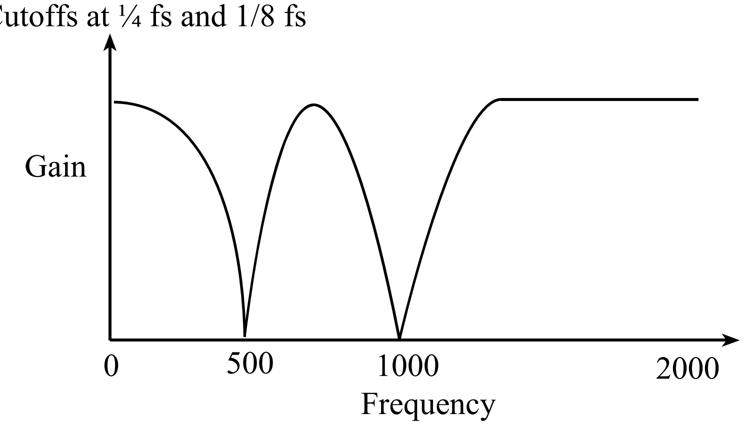
Internal fragmentation $0.5*2^n/100000 < 0.05$, $2^n < 10000$, $2^n = 8192$, n=13

Hard way

Each block can store a total of	(2^n-16) bytes
The number of blocks needed	$m = 100000/(2^n-16)$ (round up to next integer)
Wasted space is	$16*m + 0.5*(2^n-16)$
Internal fragmentation	$(16*(100000/(2^n-16)) + 0.5*(2^n-16))/100000 < 0.05$
Check solution for n=13	$m = 100000/(8192-16) = 13$
	$((16*13) + 0.5*(8192-16))/100000 < 0.05$
	$(208 + 0.5*(8176))/100000 < 0.05$
	$(208 + 4088)/100000 = 0.043 < 0.05$

$2^n = 8192$, n=13

(10) Question 3. Cutoffs at ¼ fs and 1/8 fs



(15) Question 4. The largest component is at 60 Hz (thus, this probably due to EM field pick up), and there is a harmonic at 180 Hz. There are many steps to reduce EM field pick up

Shielded box, and shielded cables, PCB with ground plane
Use twisted pair or shorter cables (reduce area for EM field pickup)
Switch the transducer to differential type and use an instrumentation amp with good CMRR
Reduce the size of the EM source (ground or shield the source), move it farther away
60 Hz notch digital filter (can't use a HPF or LPF because you might remove signal)

(25) Question 5. There are two valid approaches to solving this problem.

Solution 1. Analog ground method. First, design the system assuming a ±V supply. In this first step, analog ground is the same as digital ground. The HPF cutoff is 1 Hz, $f_c = 1/(2\pi R_1 C_1)$. Notice, since the HPF feeds into the +terminal of the op amp, the HPF sees a very high input impedance looking into the amplifier stage. If C_1 is 10 μF, then $R_1 = 1/(2\pi\ 1\ \mu F) = 160\ k\Omega$. Most any $R_1C_1=2\pi$ will be OK (we would like $1k\Omega \le R_1 \le 1M\Omega$). The range goes from 1V to 5V, so we need a gain of 5. In this circuit, the input (-0.5 to +0.5V) is converted to (-2.5 to +2.5V). Thus, we get $V_{out} = 5V_{in}-4V_g$, so $R_4/R_2=5$, $R_4/R_3=4$, choose feedback resistor, R_4= 100 kΩ. (We would like $1k\Omega \le R_2, R_3, R_4 \le 1M\Omega$). The LPF cutoff is 200 Hz. Design the Butterworth LPF using LPF.xls

first design step is to select the cutoff

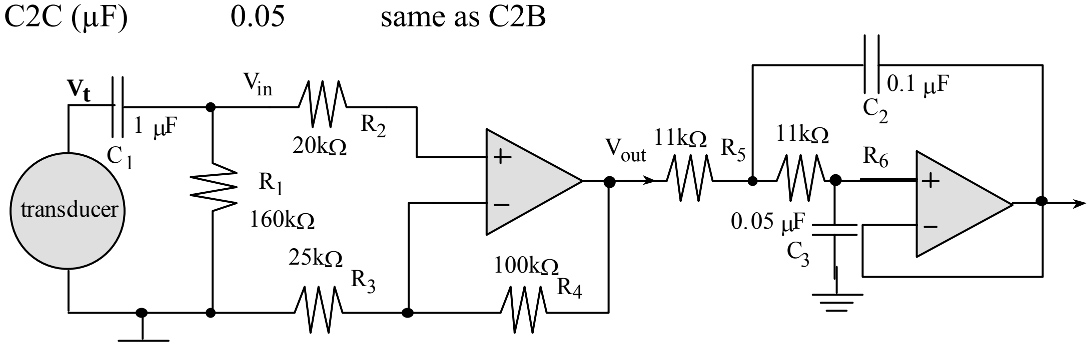
fc (Hz) 200	fill this in	
RA (kohm)	10 same as initial R	
C1A (μF)	0.1125	is 141.4/(2•π•fc)
C2A (μF)	0.0563	is 70.7/(2•π•fc) or 0.5•C1A

second design step is to choose convenient Capacitor values

fc (Hz) 200	same as previous fc	
RB (kohm)	11.252	new value to match exact fc
C1B (μF)	0.1 fill this in	
C2B (μF)	0.05	is 0.5•C1B

third design step is to choose a convenient resistor value

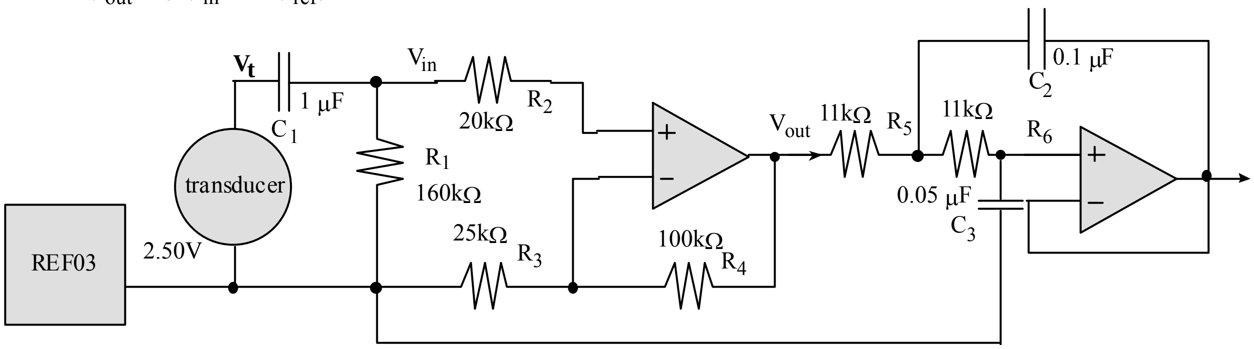
fc (Hz) 204.59	new cutoff based on these convenient values	
RC (kohm)	11.000	fill this value in
C1C (μF)	0.1 same as C1B	
C2C (μF)	0.05	same as C2B



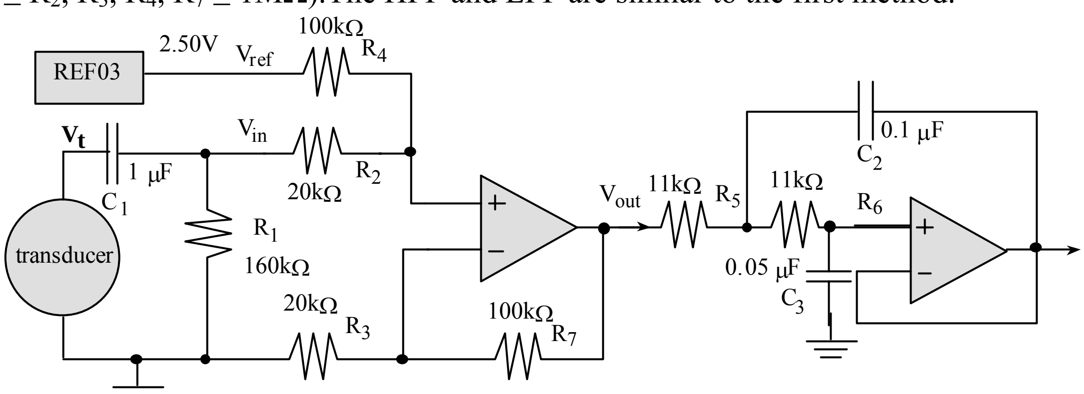
To use rail-to-rail single supply op amps, we set analog ground to 2.50V using the REF03. This way, the transducer output, V_t, has a range of 2 to 3 volts relative to digital ground. The amplifier still has a gain of 5. $V_{out} = 5V_{in}$ (relative to analog ground). Relative to digital ground we get

$V_{out} = 5V_{in} - 10$. $V_{ref} = 2.50V$

$V_{out} = 5V_{in} - 4 V_{ref}$.

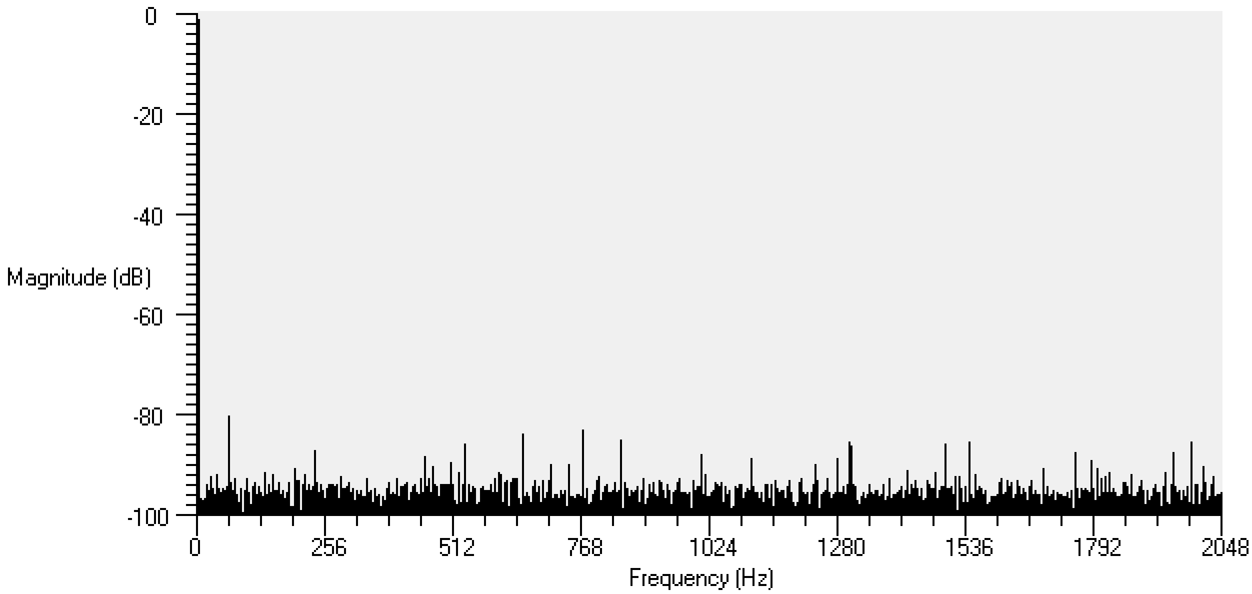


Solution 2. Digital ground method. We will design the system rail-to-rail single supply op amps, but be very careful not to have negative voltages on any of the op amp terminals. To convert the input (-0.5 to +0.5V) to (0 to +5V), we need $V_{out} = 5V_{in} + 2.5$, which is $V_{out} = 5V_{in} + V_{ref}$. We will need a ground gain of -5, so the sum of all gains is equal to 1. We need to make $R_7/R_2=5$ (V_{in} gain), $R_7/R_3=5$ (V_g gain), $R_7/R_4=1$ (V_{ref} gain), choose feedback resistor, $R_5 = 100$ kΩ. (We would like $1k\Omega \le R_2, R_3, R_4, R_7 \le 1M\Omega$).The HPF and LPF are similar to the first method.



The HPF in this case will see the R_2+R_4 impedance to ground, so the analysis of this circuit is hardware than the first one.

FYI here is an FFT with noise much less than the ADC resolution.




```
// we could have checked for illegal block numbers
unsigned char eFile_Read(unsigned char numFTT, unsigned short address){
unsigned char msb,lsb,blocknum;
  if(numFTT == 0) return 0;            // illegal block number
  eDisk_ReadBlock(FFTbuf,numFTT);
  msb = (unsigned char)address>>8;     // top part of address
  lsb = (unsigned char)address&0x00FF; // bottom part of address
  blocknum = FTTbuf[msb];              // address translation
  if(blocknum == 0) return 0;          // illegal block number
  eDisk_ReadBlock(Databuf, blocknum);
  return Databuf[lsb];
}
```