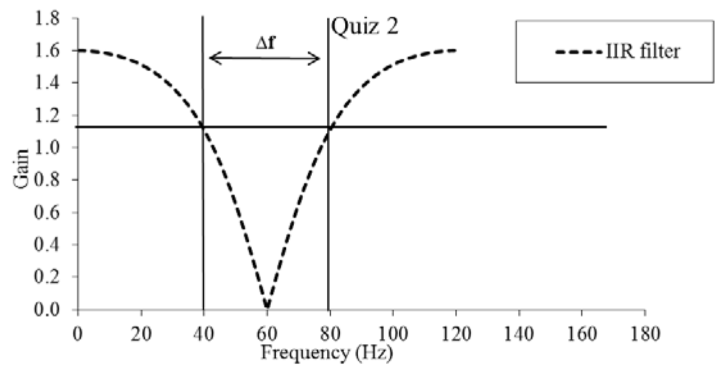
Jonathan W. Valvano

First Name: _______________ Last Name:____________________

April 4, 2014, 10:00 to 10:50am

Open book, open notes, calculator (no laptops, phones, devices with screens larger than a TI-89 calculator, devices with wireless communication). Please don't turn in any extra sheets.

(10) Question 1. Consider the following 60-Hz notch IIR filter. The sampling rate is 240 Hz, and the digital equation is y(n) = x(n) + x(n-2) - 0.25•y(n-2). The gain versus frequency is plotted



(5) Part a) Consider these three similar filters and assume the sampling rate is fixed

Filter A y(n) = x(n) + x(n-2) - 0.25•y(n-2)

Filter B y(n) = (4• (x(n) + x(n-2)) - y(n-2))/4

Filter C y(n) = (5• (x(n) + x(n-2)) - 2•y(n-2))/8

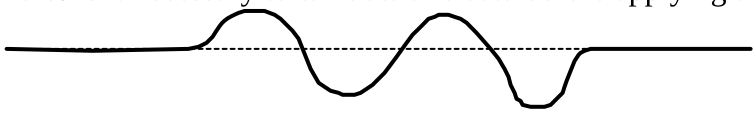
Which filters have the same poles and zeroes as Filter A? Circle one of these four choices

1) Just Filter B
2) Just Filter C
3) Both Filter B and C
4) Neither Filter B or C

(5) Part b) What is the DC gain of this filter? y(n) = (5• (x(n) + x(n-2)) - 2•y(n-2))/8

(15) Question 2. The goal is to study the frequency components of a signal using the FFT. The frequencies of interest are 0 to 50 Hz. The desired frequency resolution is 1 Hz. The SNR of the signal is 40 dB. The sampling rate is 512 Hz.

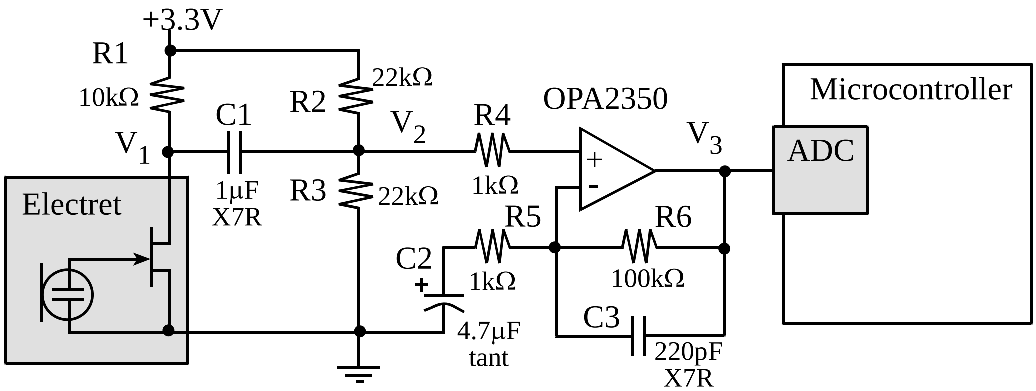
(5) Part a) Here is a typical signal. The sampled data is synchronized, guaranteeing the first 10 and last 10 points are all zero. Is it necessary to window this data before applying the FFT?



(5) Part b) Given the choice of sampling rate in Part a) what is the smallest FFT size would choose, assuming the FFT must have a size that is a power of 2? Why?

(5) Part c) What is the smallest number of ADC bits you could you choose and still capture all the information in the signal? Why?

(20) Question 3. Consider this microphone circuit.



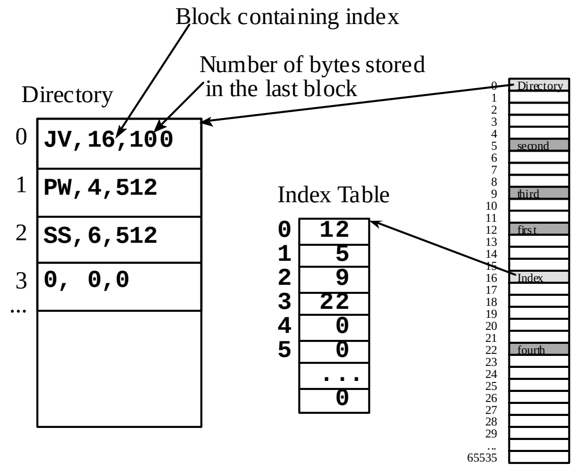
(5) Part a) Use simple op amp rules to derive an equation for voltage on the negative terminal of the op amp as a function of V_1, V_2 or V_3.

(5) Part b) What is the gain (V_3/V_1) for this circuit at DC?

(5) Part c) What is the approximate gain (V_3/V_1) for this circuit at audio signals (around 1 kHz)?

(5) Part d) What is the approximate gain (V_3/V_1) for this circuit at very high frequencies (above 1 MHz)?

(35) Question 4. Consider a file system that manages a solid state disk made with 65536 blocks where each block has 512 bytes, making a total disk size of 32 mebibytes. Block 0 is the directory. Each directory entry has two characters for the name, a 16-bit block number of the index table for that file, and the number of bytes stored in the last block of that file. A 0,0,0 entry in the directory means there is no file at that spot of the directory. The figure shows three files. All blocks except the last block of a file are assumed to be full (512 bytes). Each file has its own **index table (IT)**, and each IT fits in just one block. The file system uses the IT to define the blocks allocated to each file. The IT contains a null-terminated list of the data blocks for that file. The file JV has data at blocks 12, 5, 9, and 22. Data blocks 12, 5, 9 each contain 512 bytes of data. The last block (22) has 100 bytes of data and 412 free bytes in that block. In this system Block 0 is the directory and the other blocks are in one of three states



- 512 bytes of an index table for a file (labeled as Index in the figure)
- 0 to 512 bytes of file data (data for files PW and SS are not shown)
- Completely empty (shown as white in the figure)

(5) Part a) For Part a) assume the white boxes in the figure are free blocks, including all blocks from 23 to 65535. At this point in time, does this file system have external fragmentation? Justify your answer.

(5) Part b) At this point in time, exactly how many bytes of internal fragmentation does this file system have? Justify your answer.

(25) Part c) Assume you are given the following function that reads a 512-byte block from disk

```
int eDisk_ReadBlock(unsigned char *pt, // returned by reference
  unsigned short blockNum);            // which block to read
```

Write a C function that returns a byte from a file at location **address**. Do not worry about error handling (e.g., **eDisk_ReadBlock** error or address too big). The inputs to the function are **numIT** (the block number of the file's **IT**) and **address** (the byte address to read, where 0 is the first byte, 1 means second byte etc.). You can use two RAM buffers.

```
unsigned short IT[256];        // place to store index table
unsigned char Databuf[512];    // place to store data
```

The prototype of the C function you have to write is

```
unsigned char eFile_Read(unsigned short numIT, unsigned long address);
```

For example, assume the IT for the file is in block 16 (file JV), then this call will return the 1000th byte of the file. You must read the IT, determine where the desired data is located, read the appropriate data block(s), and return the desired data.

```
  n= eFile_Read(16, 1000);  // 16 is the block number of the IT
```

(20) Question 5. Write software to implement this moving average low-pass digital filter.

y(n) = (x(n) + x(n-1) + x(n-2) + x(n-3) + x(n-4) + ... + x(n-127))/128

The prototype for the function is

```
long Filter(long data);
```

where the input parameter is the 12-bit ADC sample, and the output parameter is the filter output. An example usage of this filter is

```
void Producer(void){ long data,result;
  data = Fifo_Get();       // sample next ADC from data stream
  result = Filter(data);   // call your function
  LCD_Plot(result);        // display filter output
}
```

The goal is to improve execution speed by reducing memory data bus cycles and by reducing arithmetic operations. Use this implementation, which needs only 3 reads, 2 stores, and 2 additions

Sum = Sum – x(n-128) + x(n)

y(n) = Sum>>7

Show the C function that implements this faster filter using signed 32-bit integer math. Use a data structure that requires a lot less than 128 reads and 128 stores to enter new data. If you need initialization, define a second function that initializes the data structure.