Jonathan W. Valvano

First Name: _____ Last Name:_____

April 5, 2013, 10:00 to 10:50am

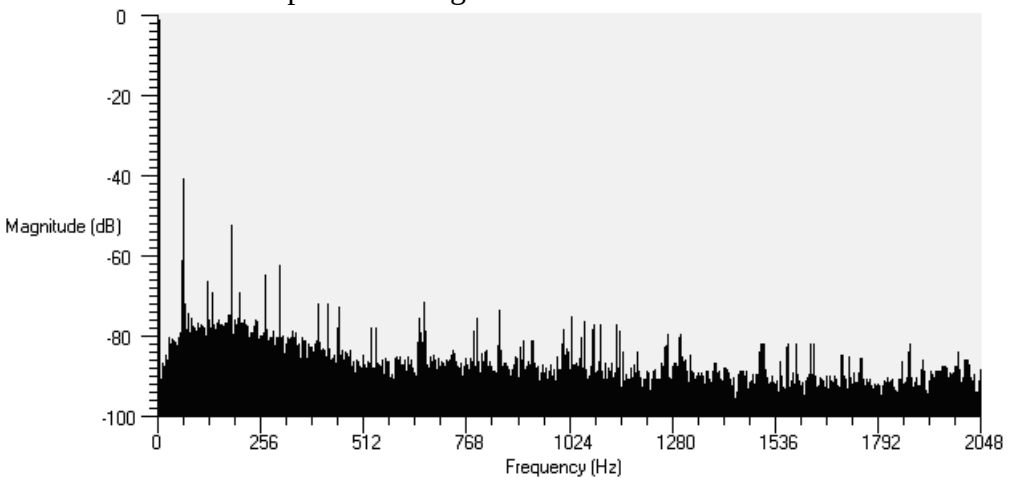
Open book, open notes, calculator (no laptops, phones, devices with screens larger than a TI-89 calculator, devices with wireless communication). Please don't turn in any extra sheets.

(10) Question 1. We defined time-jitter, δt, as the difference between when a periodic task is supposed to be run, and when it is actually run. The goal of a real-time DAS is to start the ADC at a periodic rate, Δt. Let t_n be the nth time the ADC is started. The goal is to make $t_n - t_{n-1} = \Delta t$. The jitter is defined as the constant, δt, such that

$\Delta t - \delta t < t_i - t_{i-1} < \Delta t + \delta t$ for all i.

Assume the input to the ADC can be described as $V(t) = A + B\cos(2\pi ft)$, where A, B, f are constants. Assume the input is not clipped by the finite range of the ADC ($0 < V(t) < 3$V). Derive an estimate of the maximum voltage error, δV_{max}, caused by time-jitter. Basically, solve for δV_{max} as a function of δt, A, B, and f.

(25) Question 2. A 10-bit ADC is sampled at 4096 Hz, and 4096 data points are converted to the frequency domain by calculating the FFT. The magnitude scale is calculated as dB full scale. The data has been windowed to remove spectral leakage.



Choices:

A- The magnitude of the data would increase, meaning data would shift up.
B- The magnitude axis would change from a range 0 through -60 dB to a range of 0 to -80 dB.
C- The magnitude axis would change from a range 0 through -60 dB to a range of 0 to -72 dB.
D- The magnitude axis would change from a range 0 through -100 dB to a range of 0 to -112 dB.
E- No change would occur
F- The frequency axis would change from a range of 0 to 2048 Hz to a range of 0 to 1024 Hz.
G- The frequency axis would change from a range of 0 to 2048 Hz to a range of 0 to 4096 Hz.
H- Every other point along the frequency axis would be removed.
I- There would be twice as many data points, placing a new value in between each of the existing.

Part a) Choose the best answer (A-I) that explains how this FFT spectrum would be different if we changed the number of ADC bits from 10 to 12, keeping all other parameters fixed.

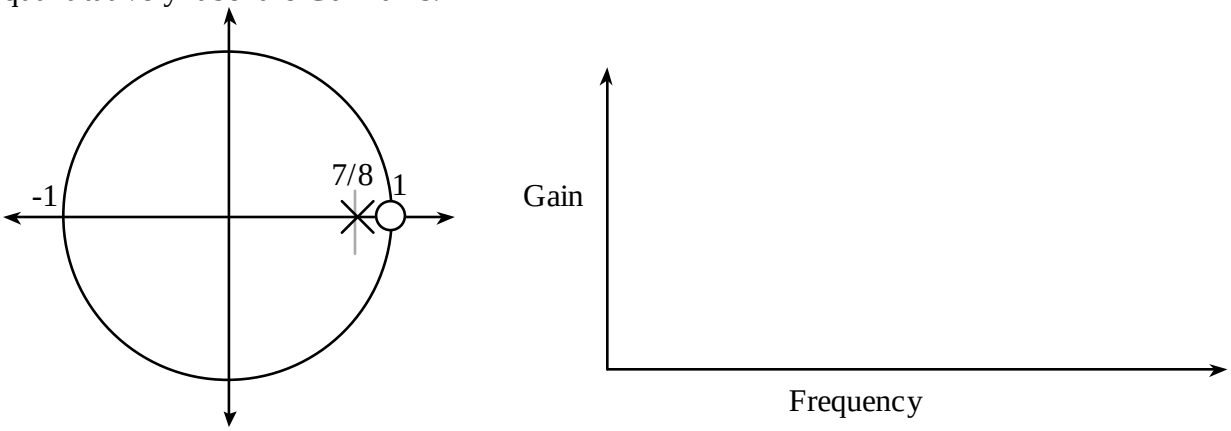
Part b) Choose the best answer (A-I) that explains how this FFT spectrum would be different if we changed the number of points in the FFT from 4096 to 8192, keeping all other parameters fixed (f_s=4096Hz).

Part c) Choose the best answer (A-I) that explains how this FFT spectrum would be different if we changed the sampling rate from 4096 to 8192 Hz, keeping all other parameters fixed (N=4096).

Part d) For this data, the sensor on a data acquisition system is removed, and all inputs are grounded. What type of noise is this?

(30) Question 3. The sampling rate of a real-time data acquisition system is 1000 Hz.

Part a) Make a rough sketch of the gain versus frequency response of a digital filter with this pole-zero plot. You should label the **Frequency** axis with specific values like 250, 500, but you need not quantitatively label the **Gain** axis.

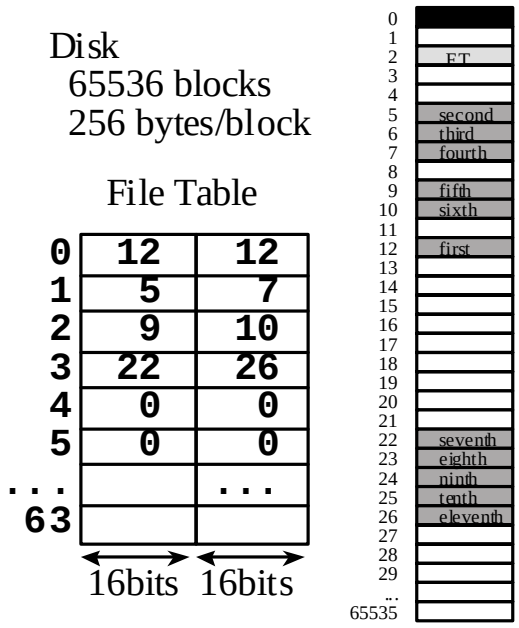


Part b) Derive the equation for the digital filter.

Part c) Write a C function using integer math to implement this digital filter. Full credit will be given to the solution that executes without multiplication or division (just addition, subtraction, and shifting). The function prototype is

```
short Filter(short input);
// input is the new sampled data (0 to 1023), and
// the return parameter is the output of the filter
```

(40) Question 4. Consider a file system that manages a solid state disk made with 16 mebibytes of EEPROM. The file system uses a **file table** (**FT**) to define the set of blocks allocated to each file. There are 2^{24} bytes on this disk, made up of 65536 blocks, where each block is 256 bytes. Block 0 contains the directory and not available for data. Each file has its own **FT**, and each **FT** fits in just one block. The **FT** is a null-terminated array of pairs of block numbers assigned to the file. For example, the pair {22,26} means consecutive blocks 22,23,24,25,26. The example in the figure shows a file with 11 allocated blocks in this order 12,5,6,7,9,10,22,23,24,25,26, with the first block at 12, and the last block at 26. The directory entry includes the file name, the total number of bytes and the block number of its **FT**. All 256 bytes of each data block can contain data for the file. For example, the figure shows a file with 11*256=2816 bytes of data, stored in 12 blocks (**FT** and 11 data blocks). In this system Block 0 is the directory and the other blocks are in one of three states



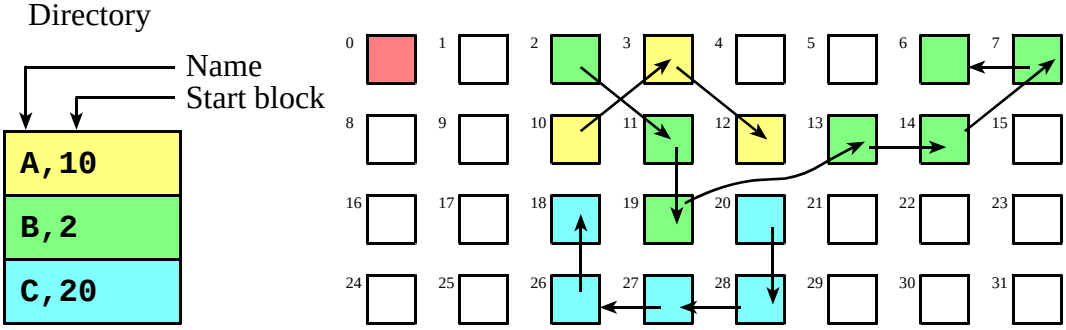
256 bytes of FT (labeled as FT in the figure)
256 bytes of file data (shown as shaded in the figure)
completely empty (shown as white in the figure)

(5) Part a) For Part a) assume the white boxes in the figure are free blocks, including all blocks from 30 to 65534. At this point in time, does this file system have external fragmentation? Justify your answer.

(5) Part b) Is it ever possible for this file system have external fragmentation? Justify your answer.

(5) Part c) Assume a file has *n* data blocks. This question addresses the read time for random access. It takes one **block read** to fetch the directory and a second **block read** to fetch the **FT**. On **average** how many more **block reads** does it take to read a single byte at any random position in the file? What is the **maximum** number of additional **block reads** that it takes to read a single byte in the file (worst case)?

(5) Part d) This question also addresses the read time for random access. Consider the linked allocation scheme described in the following figure. Assume the directory is in memory and the file has **n** data blocks. On **average** how many **block reads** does it take to read any single byte at a random position in the file? What is the **maximum** number of **block reads** that it takes to read a single byte in the file (worst case)?



(20) Part e) Assume you are given the following function that reads a 256-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 a random location. Do not worry about error handling (e.g., **`eDisk_ReadBlock`** error or address too big). The inputs to the function are **`numFT`** (the block number of the file's **FT**) and **`address`** (the byte address, where 0 is the first byte, 1 means second byte etc.). You can use two buffers.

```
unsigned short FTbuf[64][2];   // place to store FT
unsigned char Databuf[256];    // place to store data
```

The prototype of the C function you have to write is

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

For example

```
  n= eFile_Read(2, 1000);  // 2 is the block number of the FT
```

This call will return the 1000th byte of the file as shown in the picture on Page 4.