# HW #7 : ECE 444/644
# DUE: Fri. Dec. 4, 2020

Humans are weird – especially in preferences for how they want to hear their music. Graphic equalizers are popular for this very reason – it lets folks adjust enhance or suppress frequencies based on personal preference. In this HW, you will design and implement your own (every student should be unique!) audio equalizer. The shape of the frequency response is entirely up to you – but it must be "smooth" ($C^{(0)}$-continuous) over the full range of (non-aliased) frequencies, and it cannot be boring (flat). More specifically, you need to design and implement an FIR filter using fixed-point arithmetic (no floating point numbers!) that implements your equalizer response.

This is a rather open-ended design problem. You need to specify the shape of your equalizer response, and you need to choose an appropriate order for your FIR filter (filter order must be at least 20). You need to design your FIR filter using FWLS. Be sure to impose a linear phase response during the FWLS design process. Once your filter is designed, be sure to accurately plot the resulting frequency response of the filter and compare it to your desired response. Revise as necessary to obtain a suitable response. Remember: your filter should make audio sound better, not worse!

Once you have a satisfactory design, you need to implement the filter on the K22F. Design requirements include:

- Your filter should be designed so that there is no overflow in the output.
- Utilize one of the buttons to switch your filter on and off. When the filter is off, the output should directly follow the input (a digital wire).
- Utilize fixed-point arithmetic. Represent coefficients and input/output data using 16 bits. Intermediate results can, if desired, be more bits (e.g., 32) but still must use a fixed-point format. No floating point data types are allowed in your filter realization.
- Use whatever realization structure your desire (single section or cascade of sections; DFI, DFII, TDFI, TDFII or combination; etc). The important thing is to obtain good performance.
- Use a fixed 20 kHz sampling rate. NOTICE THE SAMPLE RATE CHANGE!
- Use efficient programming techniques.

Devise a method to test your filter. Your testing process should include quantitative tests (e.g., measuring the frequency response) as well as qualitative tests (e.g., does the filter make audio sound better).

**Deliverables:**

1. Demonstrate correct operation to the course instructor.
2. E-mail (*Roger.Green@ndsu.edu*) a *zipped* MATLAB mat-file that contains the impulse-response coefficients of your filter. Name your file using your initials (e.g., `rag.mat`). Your file should include a single variable `h` that is the impulse response of your filter (and also your FIR filter coefficients).
3. You should prepare a report summarizing your design and implementation. As with the previous filter design, it is incumbent upon you, the designer, to provide results in a clear and readable format. I will not waste time trying to make sense of a poor report, even if the underlying design is the best. Attach code and other bulky supporting material as appendices.

To help prepare you for the final exam, you should (at minimum) do the following drill exercises: 8.1, 8.4, 8.7, 8.8, 8.11, 9.2, 9.4, 9.6, 9.9, 9.13, 9.16, and 9.17. Since complete answers to these problems are in the appendix of the text, you DO NOT need to turn in your solutions to these drill problems.