ECE 444/644 : Fall 2020 : HW #1 : Due Monday, 9/14/2020

- Review (scan) Ch. 1, carefully read Ch. 2, and read Appendix A (MATLAB tutorial) of the book. Begin reviewing the various technical data sheets available on the class Blackboard site.

This assignment is just a single problem. You will need to demonstrate working hardware to the instructor (likely through Zoom) for full credit (we'll discuss details in class next week). It is anticipated that checkoff will take 5 to 10 minutes per student. No write up is necessary for this assignment—I will have you explain your work during checkoff. Have everything set up and running before your assigned checkoff time.

Implement a "warbling digital wire" on the K22F hardware. That is, you need to acquire a sample using an ADC, multiply it by a "warble" factor (basically a periodic envelope), and then output the modified sample to a DAC – all at a regular (periodic) rate of $F_s = 10{,}000$ Hz. I recommend the following steps:

1. Download and unzip the `K22_Project_Framework_F20.zip` file from Blackboard. This file contains the framework project for this problem and later assignments.
2. Install the Keil Microvision Integrated Development Environment (IDE, I am using μVision V5.31.0.0.) on your personal PC/laptop (ECE 237 machines might also work, but I suspect your personal machine will operate better and more reliably). Consult the file `KeiluVision5_SetupForECE444FreescaleBoard` (availabel on BlackBoard) for guidance on configuring KEIL to work with our FRDM-K22F board.
3. Open the `K22_Project_Framework_F20` project. Acquaint yourself with the various files that make up the project, include `ADC.c`, `DAC.c`, `main.c`, `MCG.c`, `TimerInt.c`, and associate header (`.h`) files. You should be able to identify gross functionality for each file.
4. Connect your K22F board to your PC through a USB cable. Press the "Download" button. Barring any surprises, you should get confirmation that your DSP is erased, programmed, and then verified—thus confirming proper connection and communcation between Keil and the K22F.
5. Open the `TimerInt.c` file. Find the line that loads the number of clock cycles to establish the period of the interrupt. Change the bogus value to an appropriate value to generate a sampling rate of $F_s = 10{,}000$ Hz. You'll need to do some reading in the reference documents to determine this value.
6. For input, we'll be using `ADC0`. In `main.c` in the `PIT0_IRQHandler` routine, you'll need to add code to do two things: initiate a new ADC conversion and read the result of an ADC conversion. Read the result into a (nonstatic, globally-defined) variable of type `uintXX_t`, where `XX` is replace with the appropriate number of bits—8, 16, 32, or some such number. To finish this step, you'll need to again do some reading in the program files and technical documents. I accomplished this task with two lines of code in the interrupt service routine (ISR) and a variable declaration (for the read sample) outside of the ISR.
7. Next, multiply the input sample by the corresponding sample of a periodic signal, thus producing a sample of the "warbled" output. The periodic signal acts as an envelope to the input, should have a repetition frequency somewhere around 1 to 10 Hz, and is smoothly shaped (sine, triangle, etc) so that each period it oscillates from 10% full scale to 100% full scale and then back. This step will require the use of a global variable. Further, you might skip this step until all other steps are complete.
8. For output, we'll be using `DAC0`. In `main.c` in the `PIT0_IRQHandler` routine, you'll need to add code to do one thing: write a (previously read ADC) value to the DAC. To finish this step, you'll need to again do some reading in the various program files and technical documents. I accomplished this task with two lines of code, which include some bit masking and shifting, in the interrupt service routine (ISR).
9. Your "warbling digital wire" program should now be complete. Compile and download your code. Add wires for input (plus ground) and output (plus ground) to the appropriate board headers. Test functionality by applying a DC-offset sine wave as input, and verifying that a warbling sine wave is generated on the output. **IMPORTANT: You must make sure your input $x(t)$ is within the amplitude range $0.0 \leq x(t) \leq 3.0$ (volts). If you apply an inappropriate voltage, you can damage the board! Be sure the function generator output termination is set to "high-z".**
10. In the event you do not get expected results, welcome to the exciting and rewarding life of software debugging! Remember—in most cases, the problem is not the hardware, the problem is most likely you and your software. Blaming the hardware feels good, but usually does little to actually solve the problem. If you have no idea how to debug code, talk to your classmates or course instructor. We'll work on debugging strategies throughout the semester.