P13038: Hearing Aid Redesign

Software Summary

Overview:

The code for this project can be found on the EDGE server in a zip file located in the /web/public/FinalDocuments/Software folder. This project can be opened in codewarrior which can be downloaded for free from Freescale's website specifically for the MC56F8006 DSC. There is generated code in this project that can be modified using Processor Expert. There are two main files where code was written to perform the sampling, amplification, and output of signals that are captured by the ADC. These files are called hearing_aid.c and Events.c along with their respective header files. The functionality that is present in the current version of the code allows for samples to be taken from the ADC in a timer interrupt method, these samples are then multiplied by a gain, and that result is output to the DAC over a SPI connection.

File Descriptions:

hearing_aid.c - This file contains the main method for the program. It initializes all of the generated code, as well as any variables that are need for operation. It contains a for(;;) loop which monitors the button inputs to detect user input to the system. The actually functionality for each switch is not yet written, but can be easily added into the if statements for each switch. These if statements only evaluate to true if the pin that each switch is connected to goes low. The pin assignments for each switch can be modified using Processor Expert for each button.

Events.c - This file is where all of the interrupt service routines are located. When a new interrupt is generated in Processor Expert it places a method stub into this file, and the code that should be run when the interrupt is tripped can be written inside of these methods. The important ISR in this file is the TI1_OnInterrupt() method. This interrupt occurs at an interval set in Processor Expert. When the interrupt occurs a reading from the ADC is taken, this reading is then multiplied by a gain value, shifted right 2 to attempt to make up for the resolution difference between the ADC and the DAC. Finally the result is split into two separate 8 bit values to be sent over the SPI connection to the DAC. It needs to be split because the send function only send 8 bits at a time. Before sending the sync line must be driven low and when all of the data is sent it is driven high. The DAC expects data to be sent MSB first which was configured using Processor Expert.

Processor Expert was also used to generate signal processing algorithms such as a FIR filter and FFT and IFFT functions. The FIR filter was planned to be used to filter the microphone signals input to the system. Coefficients for the FIR filter were never chosen because the microphones were not successfully implemented in this phase of the project. To utilize the FIR filter coefficients will need to be found using Matlab, and then stored in a variable. This coefficient variable will be used with the FIR filter and then the samples from the ADC need to be run

through the filter. The filtered samples then should be amplified and output to the DAC. A possible timing issue could arise depending on how long it takes to filter the samples. The code might need to be restructured slightly to accommodate for this.

Testing Results:

The Output of the DAC was briefly analyzed, but more in depth testing should be conducted to ensure that the output is correct. The amplitude of the amplified signal appeared correct based on the gain value used. The frequency of the output was not confirmed to be totally correct as the Oscilloscope was not reporting a correct frequency. This is believed to be due to the quantization noise present in the output of the DAC which was not filtered out before being measured by the oscilloscope.

An issue was seen when using the USB Tap JTAG programmer for downloading code to the MC56F8006. Sometimes the hearing aid project code would not download to the processor without first flashing one of the provided demo programs from Freescale. This issue was seen on both the evaluation board, as well as the PCB's that we had manufactured. No solution was found for this on the Freescale forums. Also pausing the code while executing was possible on the evaluation board but did not seem to work correctly on the PCB, this was not tested extensively due to time constraints, and other issues arising with hardware, but did not seem to have an effect on being able to run and test the code.