Adaptive Audio Room Equalizer

(A^2REq)

EECS 452 - Digital Signal Processing Laboratory

Major Design Project

University of Michigan

April 23, 2003

John Bemesderfer

James Glettler

Eric Hatty

Oz Pearlman

Bill Stewart

Landry Tientcheu

E-Mail: 452eq /at/ umich /dot/ edu

Table of Contents

Table of Figures 3

Foreword 4

Objectives 5

Resources 5

Procedure 6

- Measurement Subsystem 6
- Serial Communication & MATLAB software 11
- Equalization Filtering 13
- Summary of Accomplishments 14

Results 15

Future Work 21

Appendix I – Assembly Code for DSP56307 22

- `aareq1.asm` Source Code 22
- `mlsgen.asm` Source Code 25
- `normstor.asm` Source Code 26
- `filtr.asm` Source Code 26
- `ada_equ.asm` – Source Code 28
- `ada_ini.asm` – Source Code 29
- `intequ.asm` – Source Code 35
- `ioequ.asm` – Source Code 36
- `v_init.asm` – Source Code 42

Appendix II – MATLAB Code 44

- `matser.m` – Source Code 44

Appendix III – References 46

Appendix IV – Executive Summary 48

Table of Figures

Figure 1 - Ideal Block Diagram 4
Figure 2 - Overall System Flow Diagram 6
Figure 3 - Preliminary Block Diagram using FFT for measurement 7
Figure 4 - MLS of order 4 8
Figure 5 - Modular Shift Register Generator of Order 4 8
Figure 6 - Circular Cross Correlation of MLS of 4th Order 9
Figure 7 - Using MLS measurement to find Impulse Response 9
Figure 8 - Final Implementation of Impulse Measurement Function 11
Figure 9 - Example of Ideal and Actual Inverse Filter Frequency Response 12
Figure 10 - Final Implementation of Equalizing Filter Function 13
Figure 11 - Anechoic Chamber - Full Impulse Response 15
Figure 12 - Anechoic Chamber - Zoomed Impulse Response 16
Figure 13 - Anechoic Chamber - Front End of Impulse Response 16
Figure 14 - Anechoic Chamber - Non-Normalized Frequency Response 17
Figure 15 - 1001 EECS - Impulse Response – Zoomed 18
Figure 16 - 1001 EECS - Frequency Response 18
Figure 17 - A^2REq - Frequency Response 19
Figure 18 - SigLab - Frequency Response 19
Figure 19 - A^2REq - Frequency Response (Zoom) 20
Figure 20 - SigLab - Frequency Response (Zoom) 20

Foreword

In sound reproduction systems, such as home stereo, what we hear is colored by the individual responses of every component in the system along with the response of the room. Audiophiles and home stereo enthusiasts spend countless hours attempting to tune and tweak their setup to achieve that perfect sound. Unfortunately, the scientific rigor of such activities is almost always lacking making these adjustments difficult at best. Therefore, a system that is capable of making detailed response measurements of the audio system and the acoustic environment is needed. This system must also provide a method for correcting the system response.

This project serves as the major design experience (MDE) for EECS 452 – Digital Signal Processing Laboratory. The scope of this project is to implement a proof-of-concept single channel measurement and correction system. Because of the pressing need for such a system, and the challenges of the design, it was felt that this project meets the requirements and the spirit of the MDE. There are a number of possible uses for this project ranging from consumer to professional to academic use. Similar products exist on the market but are very costly, hard to use and are not expandable.

The block diagram in Figure 1 below shows the basic operation of the system. A microphone is used to record and measure the response of the audio system and environment at the listening position. This can be used by the digital signal processor to pre-correct the program material through equalization to make up for errors in the audio system. The system could also display the measured response information directly for external analysis.

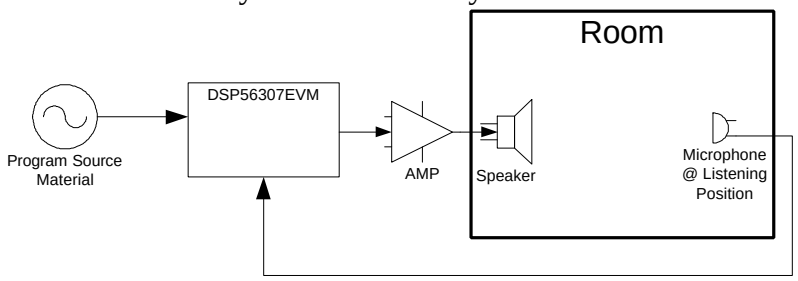


Figure 1 - Ideal Block Diagram

As this project progressed, the scope was modified to support new research and ideas, as the specifications were refined. There are a number of methods available for carrying out each sub-task each with its own benefits and disadvantages. A large portion of this project was extensive research and careful selection of design choices.

Objectives

The objectives of this project went through an evolution as the design process and implementation were underway. Certain aspects became non-essential, such as real time filter updates, in response to the shift from live-performance to home-theater applications. The major objectives that remained constant throughout the design were:

- Implement a system impulse and/or frequency response measurement prototype on a DSP.
- Implement an on-chip real-time filter for equalizing program material to correct the frequency and/or impulse response.
- Use RS-232 serial communication between the DSP and a computer for transferring impulse response and filter coefficient data and displaying results.
- Generate inverse filter coefficients using functions in MATLAB.

Resources

Digital Signal Processor

The initial decision we faced was choosing the DSP that would most adequately suit our needs. From the DSP evaluation modules we had to choose from, all are integer-processing units, so our project will be guided by this constraint. Because we wished to use MATLAB to display response plots, store data and generate inverse filter coefficients, an interface between the DSP and a computer was needed. The Motorola DSP56307EVM has on-board RS-232 serial port that allows for serial communication between the PC and DSP. The Texas Instrument's TMS320C5402 DSK also has on board RS-232 serial, however it lacks a high quality audio codec. Furthermore, the 24-bit word size of the 56303 and 56307 allows for greater precision. The 56307 has significantly more memory than the 56303 and a faster processor. This is important for the making system response measurements where large memory arrays are needed. Another advantage to the 56307 is its Enhanced Filter Co-processor (EFCOP), which allows for automated parallel computation of a filter with main program code.

Support Hardware

The design (see Figure 1) required a few other pieces of hardware. These included a loudspeaker, an audio power amplifier, a computer with two RS-232 ports, and a microphone. These items were mostly available, either through the lab resources or personal items owned by members of the team. The one piece of hardware that could most improve the system would be the microphone. We used a cardioid pattern dynamic microphone from Radio Shack, but a high quality studio microphone with an omni-directional pattern would significantly improve the results without any other direct changes to the design. Because of the low quality of the microphone, the measurement is only useful over a limited range. The cable used for the RS-232 serial interface required some work to function properly. It was necessary to solder the cable to three pins, one each for transmit, receive, and ground.

Support Software

The project required a number of software applications. MATLAB, Debug-56K, and a text editor were the programs used in project development. These were already installed on the computer used in lab, which proved very convenient. MATLAB was also used in the planning and design stages to simulate different design choices.

Procedure

The initial group meetings were brainstorming sessions to establish ideas for procedures. These ideas were not tested until later in the term, and, at first, only in MATLAB simulations. Meeting minutes were kept for almost every group session involving more than half the members.

Our ideal project goal was to be able to make a system response measurement while listening to program material without significantly interrupting or degrading the programmatic material. However, it was obvious that ensuring we had a valid response measurement system was more important than playing program material and making measurements at the same time. A few weeks into the project, when we shifted our market focus to home theater applications, it was decided that concurrent listening and measurement was not needed and could even be met with resistance or rejection.

We broke the project into four segments, as seen below in Figure 2. The labels show details of our final implementation, but the four segments remained the same: response measurement, display & storage of response, computation of filter coefficients, and real time equalization filtering.

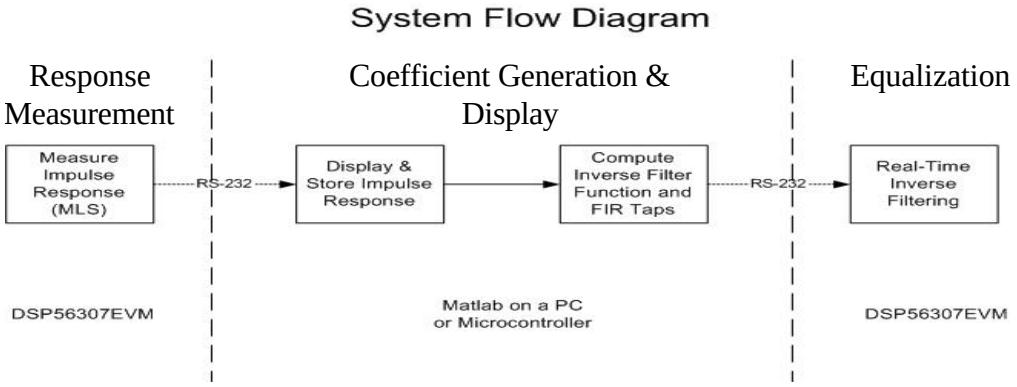


Figure 2 - Overall System Flow Diagram

Measurement Subsystem

Initial System Response Measurement Methods

We began our search for a system response measurement function based on a magnitude frequency response approach. At first, only concerned about the frequency response, our methods focused on directly measuring frequencies. To reduce measurement time and the need to listen to swept sine waves, as used by other measurement systems like SigLab, we tried to use Gaussian white noise to

make our measurement by using a Fast Fourier Transform (FFT) on the recorded output. Our initial block diagram used in our proposal is shown below in Figure 3. However, when we ran MATLAB simulations, we found that because white noise is by definition non-deterministic it was unreasonable to achieve flat frequency content of the noise over a limited time scale and therefore making measurements with white noise would not work. Additionally, we needed an enormously long FFT to get better than 1/3 octave resolution in the lower octaves. It is unfortunate we did not realize how close we were to our final solution and we wasted a lot of time simulating in MATLAB what could not work as we wanted it to. It appeared that a drastic change was necessary in the room response measurements.

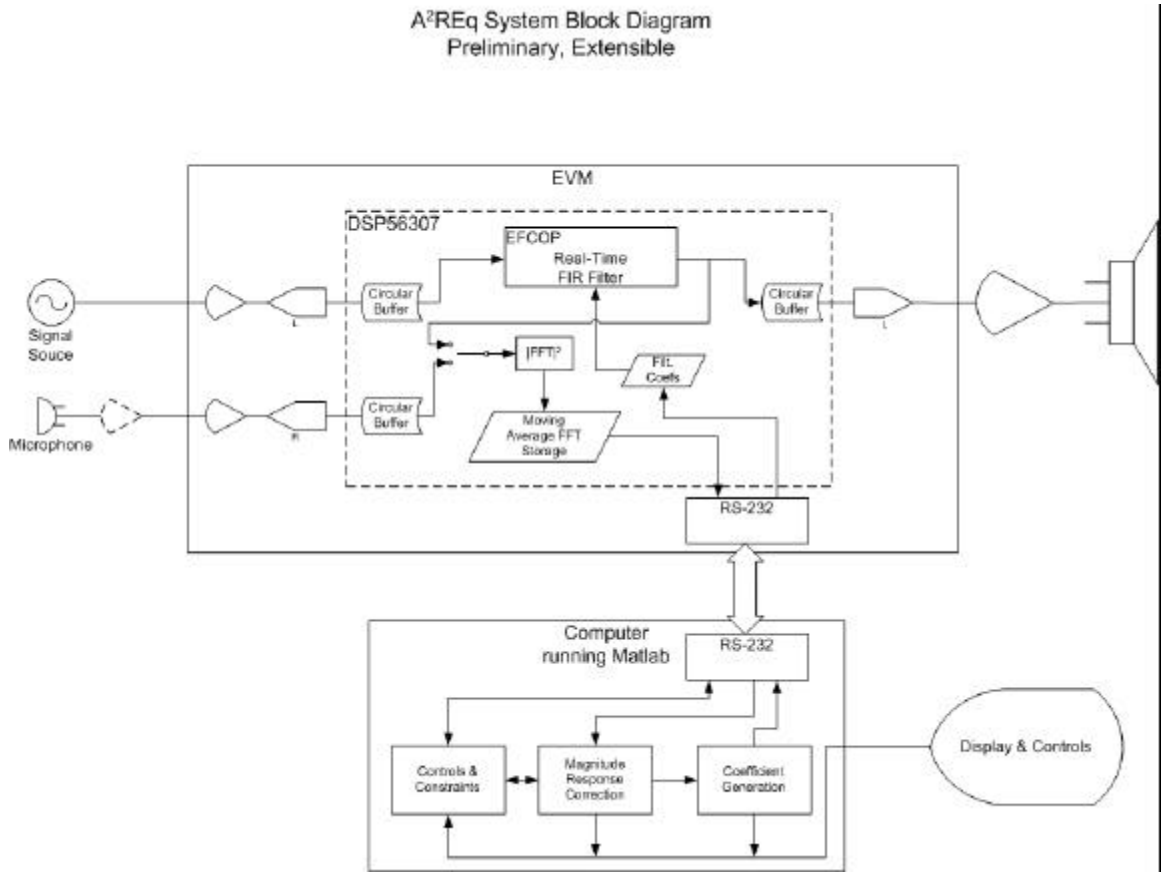


Figure 3 - Preliminary Block Diagram using FFT for measurement

As a result, we started looking back into swept-sine waves or possibly groups of sine waves. We could generate these on chip easily. The difficulty was in making accurate measurements, again due to noise or FFT size limitations. If we didn't use the FFT, we would need a swept matched filter and power detector. This was further complicated by the desire to only focus on 1/3rd octave frequency bands, which made using the direct FFT overly 'brute force' with still limited low frequency resolution.

We were also limited bed these methods because in measuring the frequency response directly, only the steady-state response of the system will be known. No transient information or time information can be ascertained from the results and therefore cannot be corrected for. In seeking

help with Professor Metzger, we were re-introduced to some fundamental concepts and given a new direction.

Impulse Response and M-sequences

The ideal way to characterize an audio system is measure its impulse (or time) response. In fact, almost all professional audio equipment and software make use of the impulse response. With the impulse response, it is possible to actually see the effects of reflections of sound from walls, or time differences in signals from different drivers or speakers. However, the difficulty of measuring an impulse response directly is in the energy content of the signal. Given a true impulse is an infinitely short pulse with infinite amplitude, any attempt to create an impulse in a real system results in miniscule signal energy. This means the impulse measurement would be overpowered by background noise. For example, crude impulse response measurements of acoustic spaces were made with gunshots in the past.

What is needed is a way to time-spread the impulse, play it through the system under test and then compress that signal back together in post-processing. M-Sequences, also known as Maximal-Length Sequences (MLS) or Pseudo-Noise (PN) are a simple to generate time-spread signal that can be used to measure impulse response. An MLS is a binary sequence with values 1 and –1 that has significant similarities to white noise, with somewhat equal frequency content over the band. The difference is that an MLS is generated with a Shift Register Generator (SRG) and is completely deterministic. This means that we always know what the sequence is at all times and therefore we always know the frequency content of the sequence. For a SRG with N registers (order N) an MLS sequence has length 2^N-1. A 4^{th} order MLS is shown in Figure 4 below with length 2^4-1 = 15.

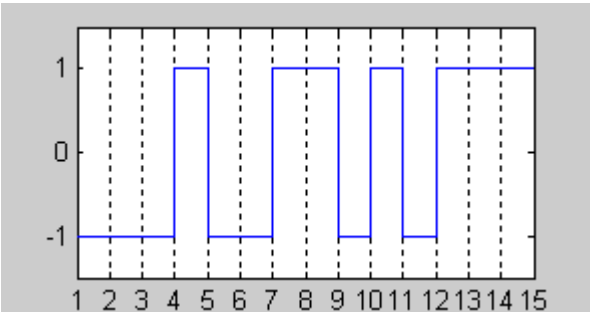


Figure 4 - MLS of order 4

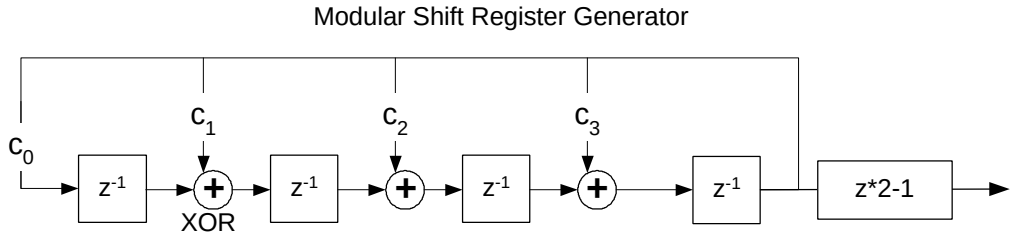


Figure 5 - Modular Shift Register Generator of Order 4

An MLS has an impulse-like two-valued circular cross-correlation function, as shown in Figure 6. Putting an MLS m(t) through a system results in an output signal s(t). The theory, as described by

Rife and Vanderkooy, shows that the system impulse response h(t) can be de-convolved from the system response by circular cross-correlation of m(t) and s(t). A diagram of this process is shown below in Figure 7 provided by Vorlander and Kob. There are alternate mathematical algorithms to perform the de-convolution. One is to take the DFT of s(t) and remove the DFT of m(t) from it and use the IDFT to get back to the system impulse response. Despite the relative ease of taking the FFT, the operations to go back and forth use a lot of memory and time.

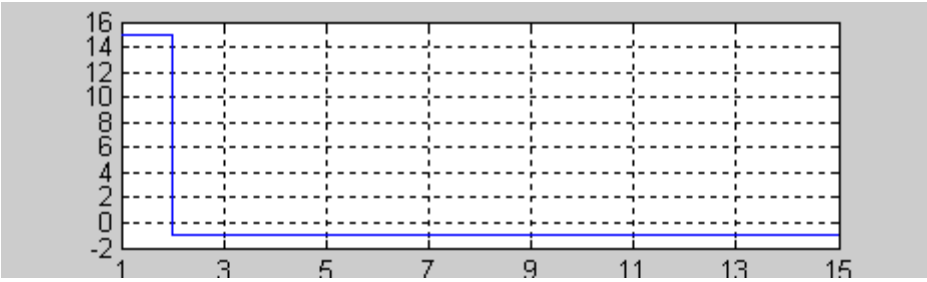
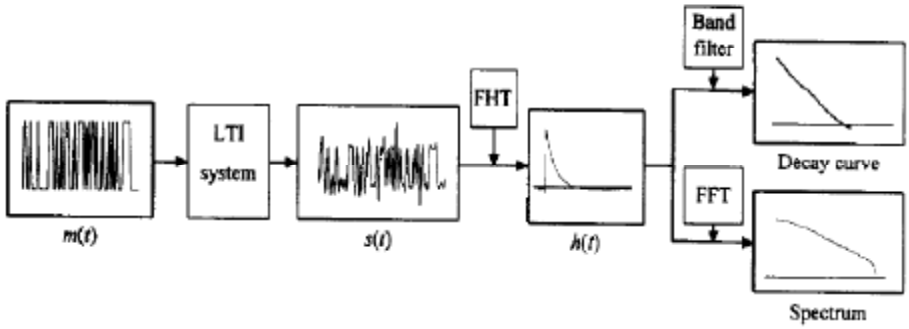


Figure 6 - Circular Cross Correlation of MLS of 4th Order



Schematic signal flow of an MLS measurement.
Vorlander & Kob 1997

Figure 7 - Using MLS measurement to find Impulse Response

The solution to the long computations required by direct correlation and complex math required by the frequency domain method is described by Cohn and Lempel as the Fast M-Sequence Transform. Because of the nature of the M-sequence, the circular cross-correlation can be simplified to a matrix multiplication between an M-sequence matrix. Of course, the matrix has a size that is a square of the length of the sequence and would be too large to implement on chip. The Fast M-Sequence Transform (FMT) relies on the inherent determinism of the M-sequence matrix.

Instead of the matrix multiplication, the FMT uses three steps that can all be computed with minimal memory overhead and computation time. First the data to be transformed is reordered or scrambled according to the sequence generator. Then the Fast-Walsh Hadamard Transform is computed in-place. This transform is nearly identical to the FFT except only additions and subtractions are used and no exponential or multiplications are computed. Finally the data is reordered or unscrambled according to a converse sequence generator. Using the FMT is by far the most efficient and most elegant method for de-convolving the impulse response from an MLS excited audio system.

Final Implementation of Measurement Subsystem

As it turns out, we ran into some difficulty generating the permutation vectors needed to scramble and unscramble the data for using the Fast M-Sequence Transform. Therefore, we settled on using a circular cross-correlation algorithm. Although this takes much longer in terms of computation, it is still robust and only requires twice as much memory as the length of one sequence. Furthermore, we can generate the sequence 'on-the-fly' instead of storing it. In this sense we have traded computation time for memory. On a different DSP with more memory we might have gone in the other direction.

One important task that needed to be dealt with was normalization of the data. As each data point in the output of the cross-correlator is computed by multiply-and-accumulate instructions into the full 56-bit accumulator, when storing the final value into a 24-bit word, some bits in the accumulator must be discarded. Because we cannot know the peak value of the output of the impulse, it is important to only throw away (or truncate) data when absolutely necessary or else the results will be to noisy to be useful. Therefore, we use a memory mapped register to store the amount of normalization, or shifts, used on each store. By using a comparison operator, if bits must be thrown away the cross correlator stops and renormalizes all previously computed values. This gives us maximum dynamic range regardless of the input levels or system response. Alternately, we could not use this method using the Fast Walsh Hadamard Transform because the computation is carried out in-place.

The largest block of memory on the DSP56307 EVM is 64kwords of off chip RAM. Furthermore the maximum size of a circular array pointer is 32768. Therefore, we set the sequence length at 32767, with an SRG order of 15. This results in a sequence length of 0.667 seconds at a sampling rate of 48kHz. In most average size rooms without long-term reverberations, this is more than adequate. It gives a large enough number of points to generate accurate frequency and time domain information. To save on memory and computation time, a Modular Shift Register Generator (MSRG) was used as shown above in Figure 5 using one word and a right shift function.

Because the analog to digital converter (ADC) has 16 bits of resolution and the word size in the DSP56307 EVM is 24 bits, we have 8 bits of headroom to play with. A further advantage of the MLS is that because it is deterministic and periodic with length 2^N-1, it can be coherently averaged over time to increase the signal to noise ratio. In our case, we pre-shift the ADC samples down by 8 bits and then sum 256 series to ensure we loose no information on the recording. This gives a 3dB increase in the signal to noise ratio fro every doubling of averages. It allows for measurements in non-ideal conditions. Some researches have even used such techniques of coherent averaging to make system response measurements during an orchestral concert without disrupting anyone listening.

Measurement Subsystem Operation

The measurement operation was written in assembly for the Motorola DSP56307 EVM. The operational block diagram is shown below in Figure 8. On run, the MSRG is initialized and the circular record buffer is cleared. One full sequence length of an MLS is played out to the speaker, generated one sample at a time in order to excite the system into a "steady-state" and eliminate end-effects. Then 256 lengths of the sequence are played and recorded by the ADC and coherently averaged using sample at a time methods for simplicity and coherency. The interrupts are switched

off for processing and the circular cross-correlation is performed on the recorded data. When completed, interrupts are enabled and the DSP sends the resultant impulse response time reversed over RS-232 serial to MATLAB as discussed below.

Given the linearity of this code, it would be relatively simple to increase the sequence length or number of averages to increase noise immunity. The one major caveat of using M-Sequences for impulse response measurement is that the system cannot tolerate non-linearity. Any significant non-linear distortion in the system will manifest as a complete loss of measurement capability. Because most audio reproduction systems are meant to behave linearly if levels are kept low enough there shouldn't be a problem under normal conditions.

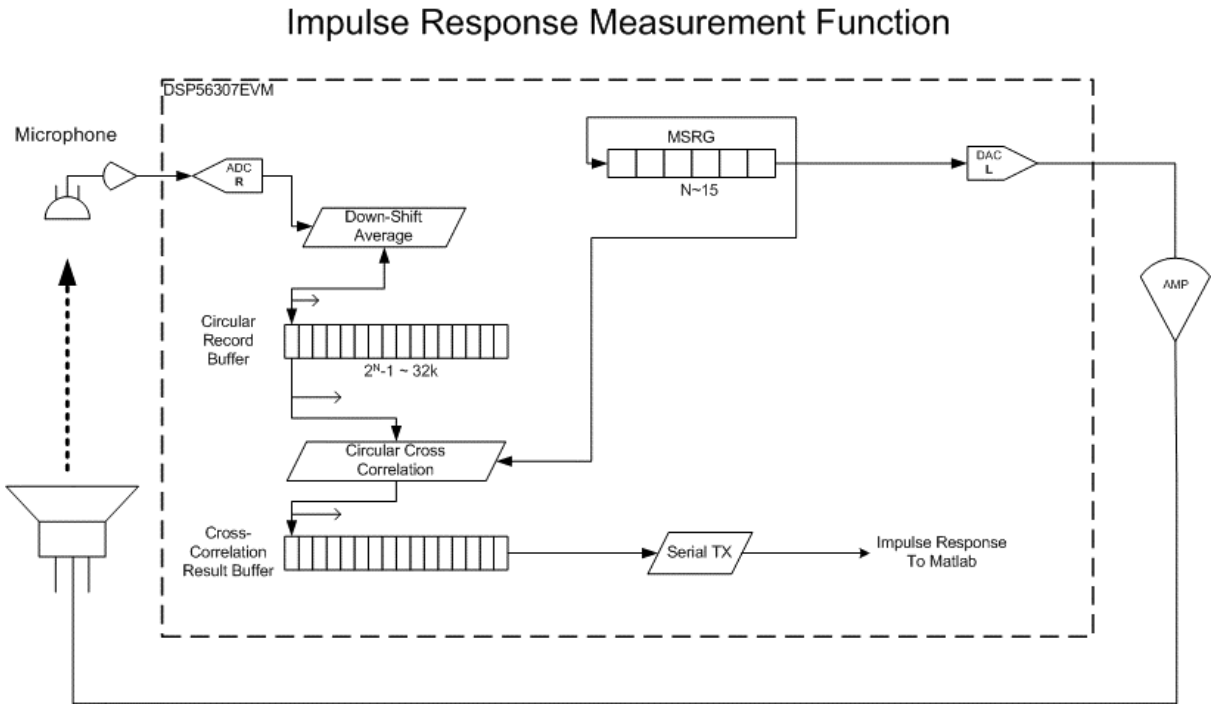


Figure 8 - Final Implementation of Impulse Measurement Function

Serial Communication & MATLAB Software

Sending Impulse Response to MATLAB

The next major obstacle to overcome was the RS-232 serial communication. These represented real world issues of interfacing between different types of hardware. The RS-232 serial communication was also much more intricate than we initially thought. Motorola has multiple tutorials on this subject. However, they are not the easiest manuals to understand because of their lack of examples. We learned by trial and error mostly – inputting the code they gave in the manuals and seeing what type of response it generated. We experimented with using Terra Term Pro, a terminal emulator, to receive I/O via the RS-232, instead of MATLAB. We transitioned to MATLAB from Terra Term Pro early on, as MATLAB was part of our actual project implementation. Progress came in increments. First, any sign that data had been transmitted or received was great news. Then, we attempted to send ASCII text, followed by strings of digits, and finally followed by long arrays of data, as would be necessary in the actual implementation.

In the final implementation, the DSP sends the impulse response of the room to MATLAB. We recorded 32767 samples, $2^{15} - 1$, and sent these to MATLAB. Serial communication works with 8-bit words. The samples were in the form of 24-bit words on the Motorola DSP56307, and therefore each required 3 transmissions. This ended up totaling 98301 transmissions for the 32767 samples. Sending nearly 100,000 transmissions over serial communication was a major factor in the time of processing in the final implementation.

Generating the Inverse Filter

Upon receiving the impulse response of the room, MATLAB takes the FFT of this data. The FFT is then truncated from 100 Hz to 10 kHz, due to equipment limitations inherent to the microphone we were using. This is easily upgradeable by using a better microphone with a wider frequency response. The FFT data was normalized using a mean value of 0 dB. After the normalization the FFT was inverted and frequencies outside the 100 Hz to 10 kHz range were not affected. Next, MATLAB's `fir2()` function, which uses the Remez exchange algorithm, was used to create the inverse filter. In this manner, MATLAB generates an inverse filter that corrects only for the stead-state magnitude frequency response between 100 Hz and 10kHz. Our research indicated that generating a proper time-domain inverse filter is a project in and of itself, so we used this to demonstrate a proof-of-concept that the overall system would work.

Experimentation was done using 256, 512, and 1024 taps. The number of taps used had a very large impact on the accuracy of the inverse filter. Using 512 taps seemed to create a filter output that looked smooth and sounded the best. This processing did not account for phase or timing of the system as a time-domain generated filter would. It also doesn't account for psycho-acoustic factors. A representation of the desired and actual inverse filters generated by our MATLAB code with 512 taps is shown below in Figure 9.

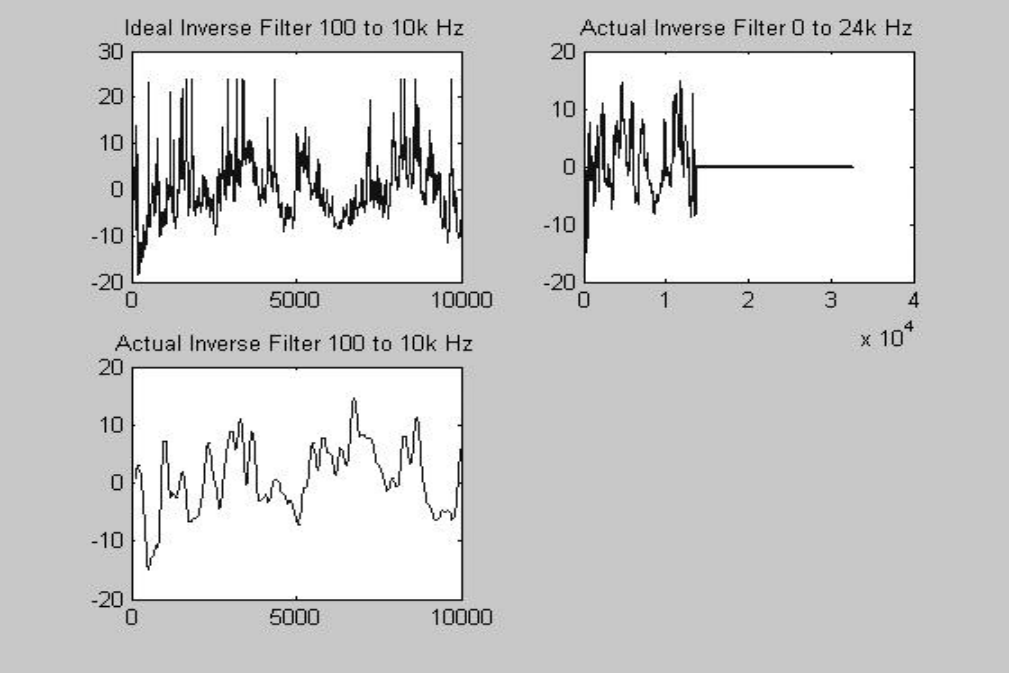


Figure 9 - Example of Ideal and Actual Inverse Filter Frequency Response

Sending Filter Coefficients to the DSP

MATLAB then sends the coefficients through the RS232 to the DSP. As before, each of the coefficients was sent in three, 8-bit segments to the DSP. The DSP then receives the coefficients, converts the 8-bit segments into the 24-bit coefficients, and places them into an on-chip data array.

Equalization Filtering

One of the advanced feature sets of the Motorola DSP56307 is the Enhanced Filter Coprocessor (EFCOP). This was a major reason in our initial choice of the 56307. The idea was to have the EFCOP processing coefficients "behind the scenes", leaving the DSP to process the rest of the program, such as making response measurements. This would reduce our computation time for real time updating of coefficients, possibly moving the coefficient generation algorithm to the DSP.

The scope of the project evolved from a focus on real time updating of coefficients to room response measurement and constant filter implementation. It became apparent during the testing stages of the project that use of the EFCOP was beyond the scope of work to be done on this project due to deadlines and time constraints.

A modified version of `LAB56B.asm` was instead used to create the on-chip inverse-filter. The modifications required for this code included the conversion from the DSP56303 to the DSP56307 and the location from which the coefficients are read. A functional block diagram is shown below in Figure 10.

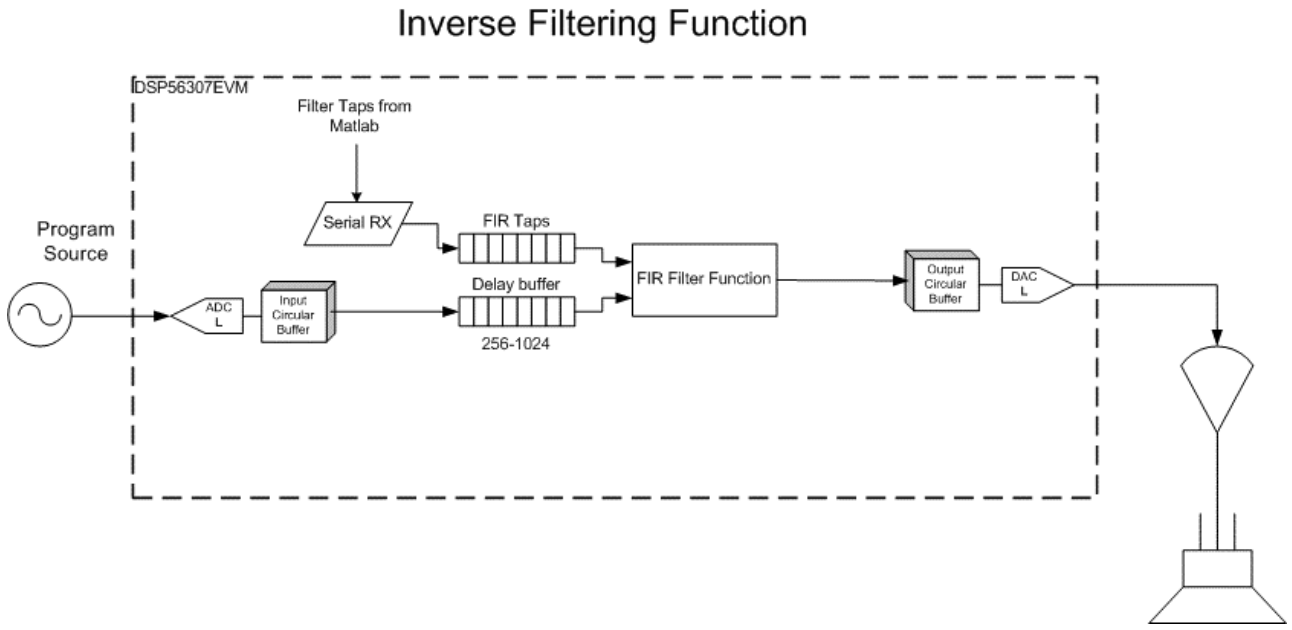


Figure 10 - Final Implementation of Equalizing Filter Function

Summary of Accomplishments

The final implementation used a plethora of techniques to accomplish all major objectives set out during the initial project planning. The chain of events runs as follows:

1. M-sequences are formed using a modular shift register (MSRG) of order 15.
2. One sequence is played to stabilize the system.
3. The Motorola DSP56307 plays 256 M-sequences through the speaker.
4. M-sequences are recorded and averaged to create a 32767 (2^{15} - 1) array of samples.
5. Recorded data is circularly cross-correlated while normalized "on-the-fly" to generate impulse response.
6. This impulse response is transmitted to MATLAB via the RS-232 serial communication with circular buffering.
7. MATLAB is used to take the Fast Fourier Transform (FFT) of this impulse, find an inverse and put the result into the FIR2 code to yield the filter coefficients.
8. These filter coefficients are transmitted back to the DSP via the RS-232 serial communication with circular buffering.
9. The DSP is then used to generate and implement the FIR filter on the board

We were fully successful at implementing all the above processes, which meet our original definition of success. Furthermore, the system is successful at making impulse response measurements, generating inverse filters and real-time filtering of programmatic material.

Results

The following plots are the raw outputs from the MATLAB software and are impulse response and frequency response plots of various acoustic environments. The following data sets were made using one ADS Model L620 3-way Loudspeaker (modified), an integrated Technics SA-290 amplifier, and an audio-technica ATR20 cardioid pattern dynamic microphone. A sequence length of 32767 was used.

Anechoic Chamber

The anechoic chamber is a large enclosed space in the Radiation Laboratory in the EECS department. It is a horn style chamber meant for microwave studies and measures approximately 60 feet by 20ft by 20ft. The walls are covered with thick foam absorber. It's not a perfect acoustic chamber but it works well. For these measurements, the loudspeaker was set at the far narrow end of the chamber and the microphone placed on a platform in the center of the wide section, approximately 50 feet away.

In Figure 11, the full impulse response is shown as returned from the DSP. In the chamber, there are minimal reflections so the immediate direct sound is seen. A zoomed in impulse is shown in Figure 12. As one can see, it is nowhere near a perfect impulse or even a nice sinc function. The frequency response is shown in Figure 14.

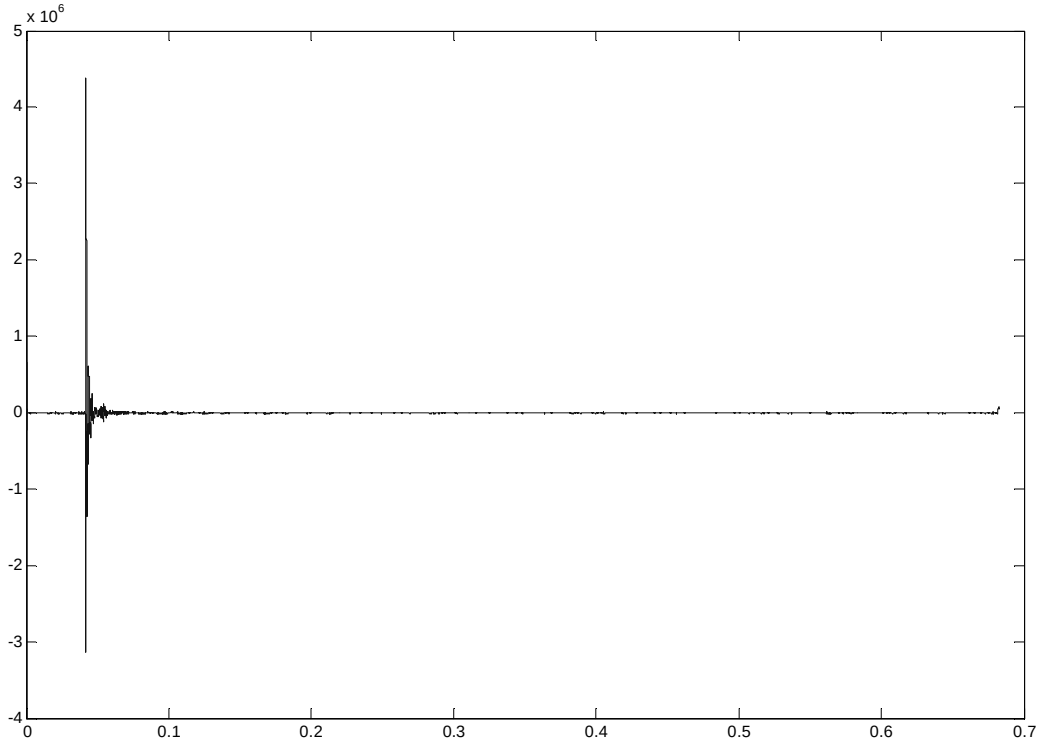


Figure 11 - Anechoic Chamber - Full Impulse Response

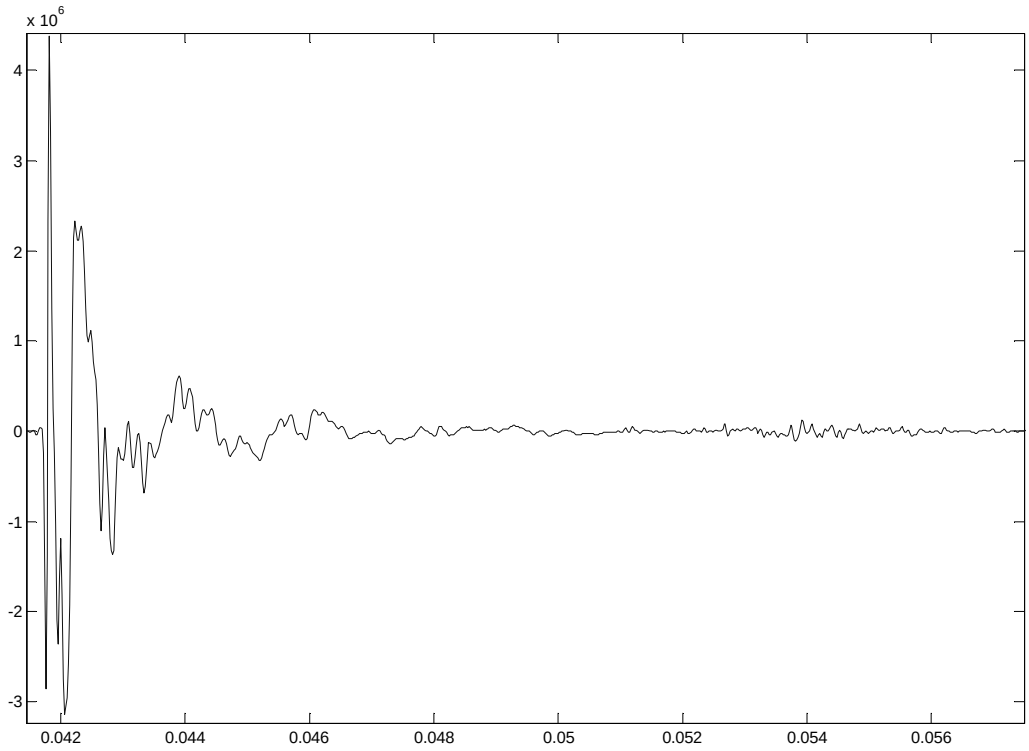


Figure 12 - Anechoic Chamber - Zoomed Impulse Response

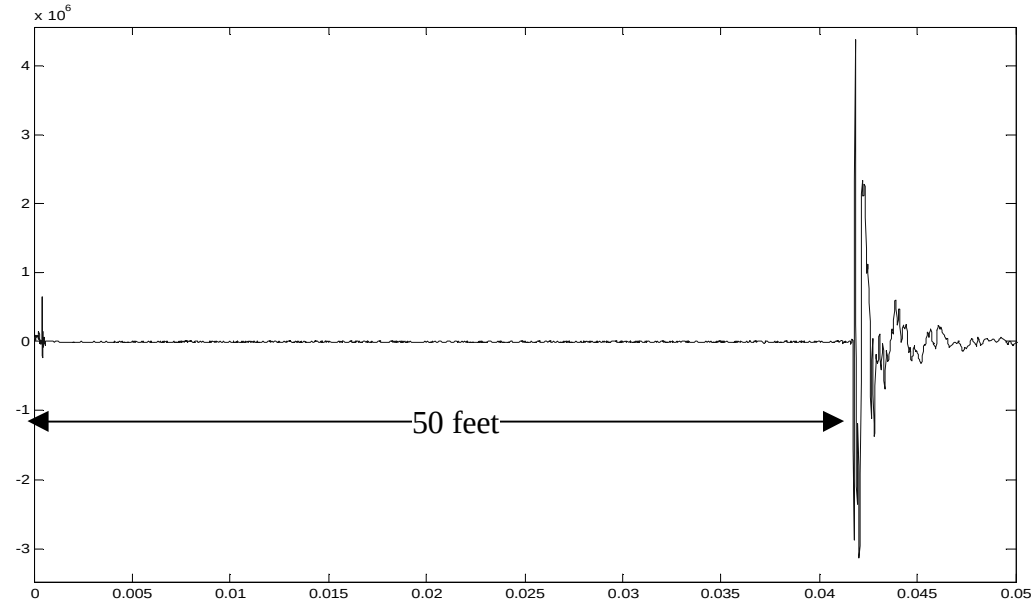


Figure 13 - Anechoic Chamber - Front End of Impulse Response

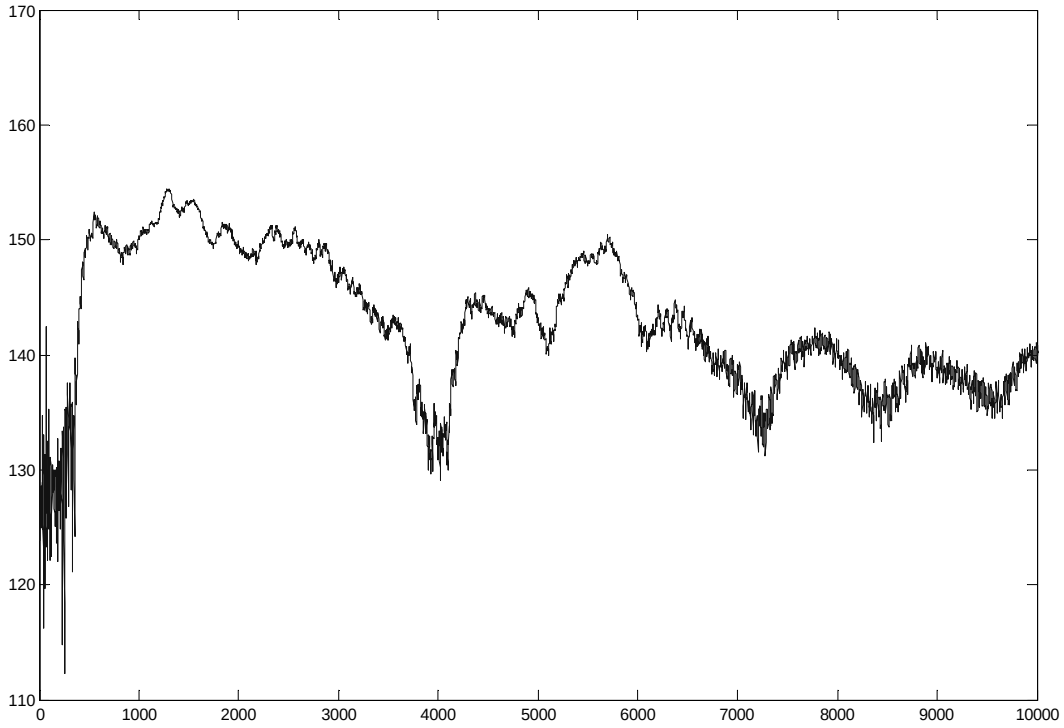


Figure 14 - Anechoic Chamber - Non-Normalized Frequency Response

1001 EECS Classroom

The following measurements were made in the 1001 EECS Classroom. The speaker was set on the front table facing towards the back of the room while the microphone was placed on the table approximately 15 feet away.

In the impulse response (Figure 15) the first reflection is very clear and occurs a few milliseconds after the main impulse with what appears be the same phase. Based on a rough estimation of distances, this looks like the first reflection off the ceiling. The measurement is also much more noisy due to all the other reflections taking place. Consequently, the frequency response is very different and much more noisy. No 1/3rd octave graphic equalizer or real time analyzer could come close to actually measuring and correcting these problems.

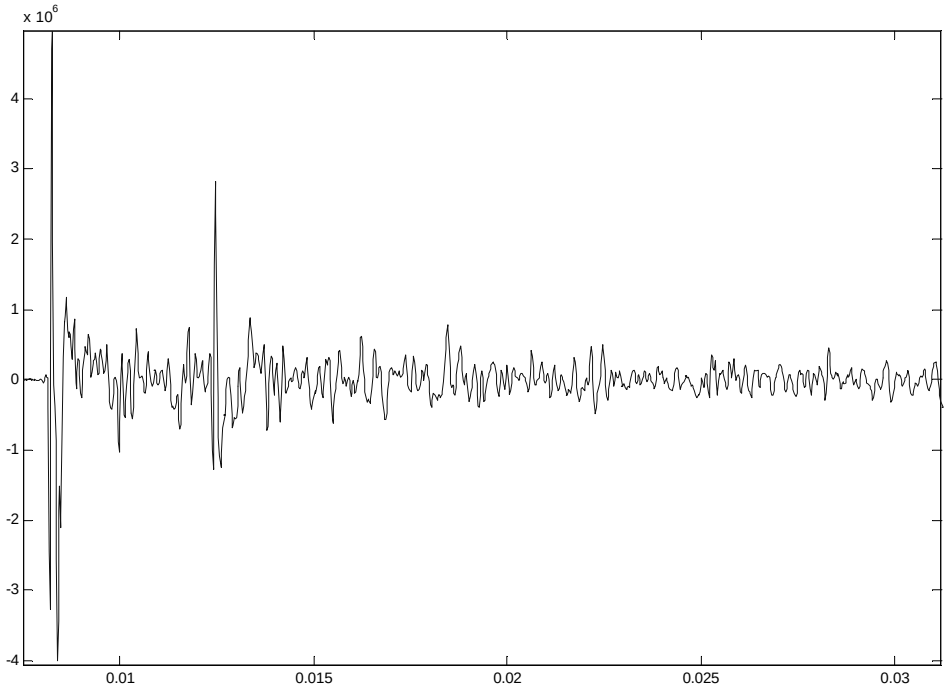


Figure 15 - 1001 EECS - Impulse Response – Zoomed

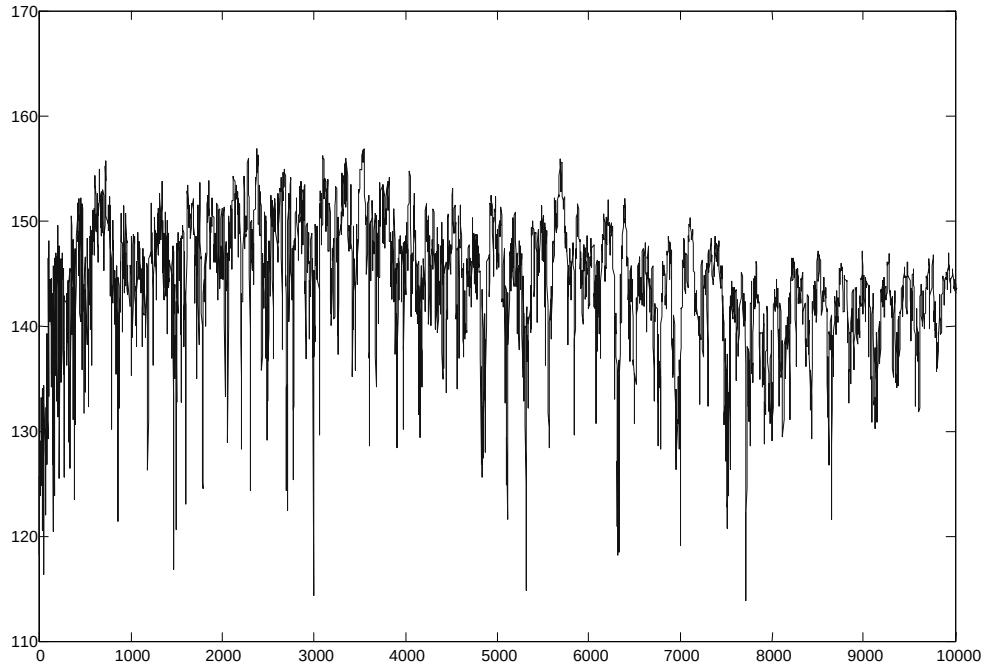


Figure 16 - 1001 EECS - Frequency Response

Experimental Accuracy

The following frequency response plots were made using a small omni-directional electret microphone and a cheap computer speaker placed approximately one foot apart. One measurement was made with the A^2REq system and one was made with a SigLab swept-sine analyzer. Do note that the A^2REq measurement agrees with the SigLab measurement but is much more detailed and excluding processing time, completes much faster with less audible irritation.

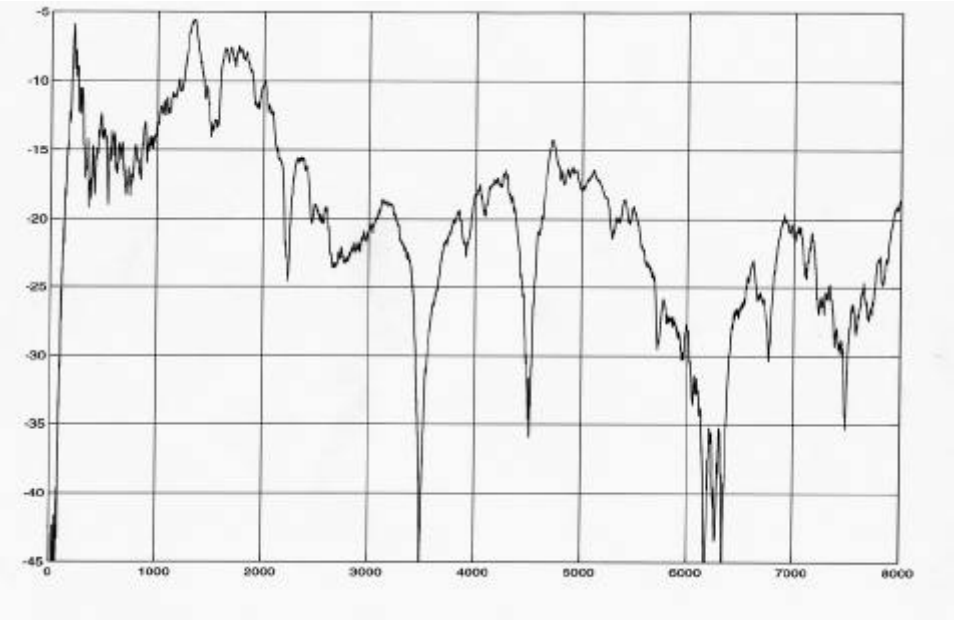


Figure 17 - A^2REq - Frequency Response

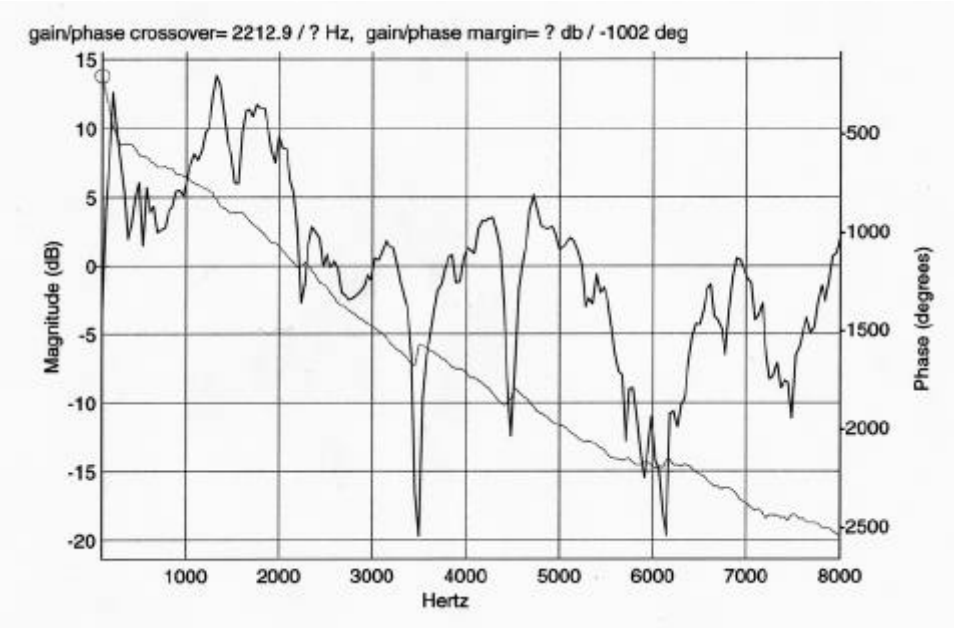


Figure 18 - SigLab - Frequency Response

These plots are a zoom in on a smaller frequency range. The A^2REq measurement just required a change in the plot axis. However to get an adequate frequency response in SigLab, a new measurement was needed with more points.

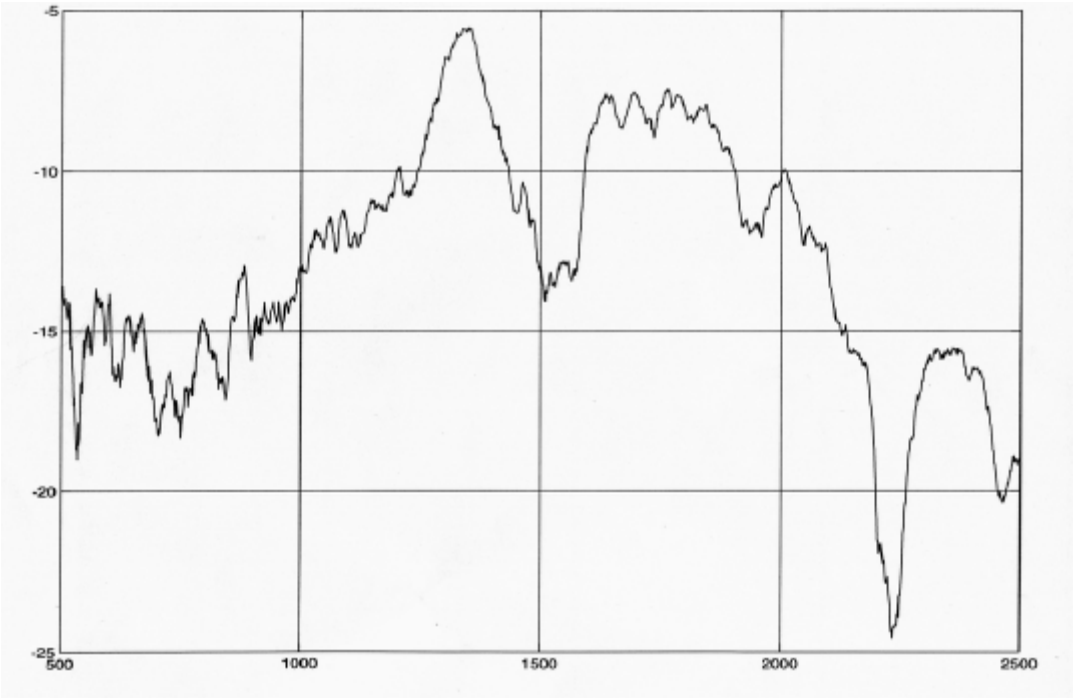


Figure 19 - A^2REq - Frequency Response (Zoom)

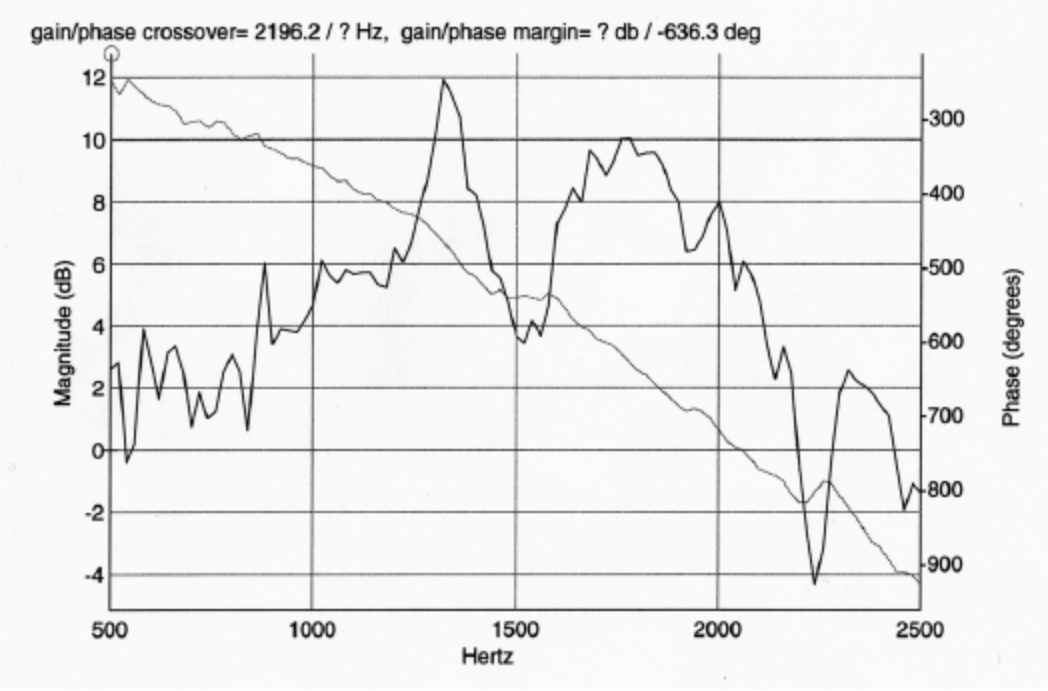


Figure 20 - SigLab - Frequency Response (Zoom)

Future Work

Many facets of the project's scope had to be sacrificed or changed in some way in order to yield a workable prototype by the deadline. The real-time equalization updating of the system could not be achieved without further research and testing. Many ideas that were placed aside could have been utilized, but time constraints and imposing deadlines loomed.

During the planning stages of the project, extensive research was done on the Fast M-Sequence Transform (FMT) using the Fast Walsh-Hadamard Transform (FWHT). This idea was placed aside as the scope of the project was reduced due to time constraints, hardware constraints, difficulty in theory and looming deadlines. Implementing the FMT will decrease the computation time of the impulse response significantly as it is analogous to doing the FFT instead of the direct DFT

Another current limitation on our project is that of the equipment. The microphone we used has a rough frequency range of 100 Hz to 10 kHz. By upgrading to a better microphone with a wider frequency range, the measurement system will be more robust. Furthermore the system is currently monaural. This project could come closer to marketability if a stereo measurement system was implemented such as the one proposed by Xiang and Schroeder.

The required hardware to make this project run is quite formidable and costly. Making this into a stand-alone system would be critical for any attempt at marketing this product. This would include transferring from MATLAB, a costly software package not always available, over to something more mainstream, such as C++.

The impulse response room measurement system using M-sequences delivers a tremendous amount of information about the room or listening environment being tested. Not only is information about the frequency response available, but also about the phase response and time alignment. The current inverse filtering system only corrects for frequency response variation, and does nothing along the lines of correcting phase. This would definitely be the next step in the process of developing a real world version of the audio adaptive room equalization system.

This summer, a few of the A^2REq project team plan to continue work on this project. Our first three main tasks will be to: 1) migrate the operation to an alternate DSP platform, possibly a TI C5510 or an Analog Devices Shark-Melody DSP, with preference given to floating-point systems, 2) fully implement the Fast M-sequence Transform on the DSP of choice, and 3) increase the noise immunity of the system by increasing the length of the sequence and the number of coherent averages used, possibly even allowing for a reduced amplitude used for characterization.

Appendix I - Assembly Code for DSP56307 EVM

There were two assembly programs written for the A^{2}REq project that operate separately for the measurement (`aareq1.asm`) and the equalizing filter functions (`filtr.asm`). The measurement program also needs two support macros: `mlsgen.asm` and `normstore.asm`. Finally both programs require a few support files relating to the audio codec and interrupt support: `ada_equ.asm`, `ada_ini.asm`, `ioequ.asm`, `intequ.asm`, and `v_init.asm`. These support files were not significantly modified beyond `ada_ini.asm` to support dual channel input circular buffers and simultaneous sample at a time input/output.

Removed for Publication

Appendix II – MATLAB CODE

A number of MATLAB programs were written for testing and simulation of various aspects of this project in the design phase. However, the final system uses one MATLAB program to receive and send data to the DSP and to compute the required inverse filter coefficients (`matser.m`). Simulation scripts can be provided on request.

Removed for Publication

Appendix III – References

References are in alphabetical order by author and then title.

Atkinson, John. "Measuring Loudspeakers, Part Two." Stereophile (December 1998) republished < http://www.stereophile.com/printarchives.cgi?100 >

Borish, Jeffrey and Angell, James B. "An Efficient Algorithm for Measuring the Impulse Response Using Pseudorandom Noise." J. Audio Engineering Society 31 (July/August 1983): 478-487

Cohn, Martin and Lempel, Abraham. "On Fast M-Sequence Transforms." IEEE Transactions on Information Theory (January 1977):135-137

Cowan. "An Introduction to Adaptive Filters." Loughborough University.

Farina, Angelo. "MLS Impulse Response Measurements for Underwater Bottom Profiling." University of Parma. (17 October 2000) < http://www.ramsete.com/Public/Papers/116-ECUA98.PDF >

Genereux, Ronald "Adaptive Filters for Loudspeakers and Rooms" J. Audio Engineering Society (1992) republished < http://www.realspace.com/SigTech/aes_90sf.html >

Gumas, Charles Constantine. "A century old, the fast Hadamard transform proves useful in digital communications." Personal Engineering & Instrumentation News (November 1997):57-63 republished < www.chipcenter.com/dsp/DSP000517F1.html >

Ludwig, Arthur C. "Part III: Maximum Length Sequences" < http://www.silcom.com/~aludwig/Signal_processing/Maximum_length_sequences.htm >

Metzger, Kurt. SREM.C Computer Software. (1992)

Mommertz and Muller. "Measuring Impulse Response with Digitally Pre-emphasized Pseudorandom Noise Derived From Maximum-Length Sequences." Applied Acoustics 44 (1995): 195-214

Peltonen, Timo. "A Multichannel Measurement System for Room Acoustic Analysis" Helsinki University of Technology. (23 October 2000)

Rife, Doug. "Build a High-end Surround Stereo System Around a Digital Parametric Equalizer." DRA Labs. < http://www.mlssa.com/surround/surround3.htm >

Rife, Douglas D. and Vanderkooy, John. "Transfer-Function Measurement with Maximum-Length Sequences." J. Audio Engineering Society 37 (June 1989): 419-443

Smith, Julias O. III. Introduction to Digital Filters Center for Computer Research in Music and Acoustics < http://ccrma-www.stanford.edu/~jos/filters/ >

Talantizis, Fotios and Ward, Darren B. "Multi-Channel Equalization in an Acoustic Reverberant Environment: Establishment of Robustness Measures." Imperial College of Science, Technology and Medicine. (2002) < http://www.ee.ic.ac.uk/dward/papers/ioa-2002.pdf >

Torger, Anders. NWFIIR Audio Tools: Software Equalizer and Tools for High Resolution Digital Audio < http://www.ludd.luth.se/~torger/filter.html >

Tossavainen, Timo H. "Fast in-place Walsh-Hadamard Transform." < http://www.musicdsp.org/archive.php?classid=2#18 >

Vorlander, Michael and Kob, Malte. "Practical Aspects of MLS Measurements in Building Acoustics." Applied Acoustics 52 (1992):239-258

Xiang, Ning and Schroeder, Manfred R. "Reciprocal maximum-length sequence pairs for acoustical dual source measurements." J. Acoustical Society of America 113 (May 2003): Proof Copy 008305JAS

DSP56300 Family Manual Motorola. (August 1999)

DSP56303: 24-Bit Digital Signal Processor User's Manual Motorola. (1998)

DSP56307 Product Summary Page Motorola < http://e-www.motorola.com/webapp/sps/site/prod_summary.jsp?code=DSP56307&nodeId=0127953350V37G >

"Workshop Signals & Systems: Part C Pseudo Random Binary Signal Correlation Functions." Hogskolan Kalmar University.

Appendix IV - Executive Summary

John Bemesderfer

I am a senior engineering student with one class left for my undergraduate degree in Electrical Engineering, and I also have a degree in English. I was involved with research into and implementation of the impulse measurement system, MATLAB simulations, and extensive debugging.

I hope to continue work on this project to expand its usefulness to the consumer, particularly the audiophile. I am currently accepted and plan to attend the University of Michigan Master's program in Systems Electrical Engineering.

James Glettler

I am a third year electrical engineering undergraduate with one remaining semester. I have been most involved in this project with designing and implementing the impulse response measurement subsystem. I also have helped out with debugging the other various aspects of this project.

This summer I hope to continue working on this project and bring it to commercial readiness. I also will be applying to graduate school. I hope to get a Ph.D. in signal processing and create a high-fidelity sound research and development firm, Elysian Audio (http://www.elysianaudio.com).

Eric Hatty

I am a fourth year electrical engineering student, graduating in April 2003. My involvement in this project includes RS-232 serial communication between MATLAB and the DSP, as well as research into the DSP56307's Enhanced Filter Coprocessor. I was also involved in several stages of debugging and testing of the final design.

I will begin my career with NEC Automotive Electronics this May as a Field Applications Engineer. I hope to continue my education with a master's degree in Business Administration after gaining more work experience.

Oz Pearlman

I am a graduating senior with a B.S. in electrical engineering. I have been most involved in this project in implementing the RS-232 serial communication, creating the presentations, and writing this technical report.

This summer I will be moving to New York City to work for Merrill Lynch in Global Technology Services. I am also planning to continue my magic career. My website is www.watchmagic.com and I am releasing two magic DVD's to the market in the upcoming weeks.

Bill Stewart

I am a graduating senior with a B.S. in electrical engineering. I have been most involved in this project in implementing the RS-232 serial communication, creating the MATLAB inverse filter code, and debugging assembly code.

I am currently looking for a job in the electrical engineering field. I hope to continue my education with either a masters in engineering or business.

Landry Tientcheu

I am a graduating senior student at the University of Michigan (Michigan) with a Bachelor of Science in Electrical Engineering and Mathematics. For my senior design project, I decided to be part of the Adaptive Audio Room Equalizer team to nourish my interest in acoustic music and enhance my knowledge of Audio equalizers.

For this project, I essentially worked on the implementation of circular buffers. I was also involved with the Enhanced Filter Coprocessor FIR filter (EFCOP) and of course the writing and debugging of the working code. I will be going to graduate school next school year to complete a Master's degree in Electrical Engineering and another one in Business Administration. Eventually, I would love to make a career in the music industry.