1. Project Overview

1.1 Project Description

As originally intended, this project was designed to provide a means to digitally equalize a pair of speakers to provide a flat in-room frequency response. This method would be reconfigurable and as such could provide an equalized system for any room. The project is designed around the TMS320C54 DSP chip and performs it function in real time, flattening the frequency response of pink noise in a room until a configurable level of performance is met. The project involves generation of pink noise for output to an amplifier and subsequently speakers, microphones and a mixer to provide input to the DSP to be analyzed, the DSP for analysis and filtering, and an oscilloscope to observe the process.

1.2 Project Goals and Practical Applications:

The goal of the project was to design a real time analyzing (RTA) digital equalizer (DEQ). The proposed DEQ would be able to equalize the frequency response of a loudspeaker system in a given room to a set level of accuracy by generation and analysis of pink noise and then application of filters. Upon completion of the equalization process, a music signal would be filtered by the DEQ and then outputted by the loudspeaker system so that the perceived sound waves in the room would closely match that of the original music signal. The choice of loudspeaker and room is irrelevant, but the placement of the speakers and microphone is relevant. If the listening position is changed or the speaker placement is changed, the room needs to be re-equalized. The DEQ should be

able to equalize any loudspeaker (within the loudspeakers normal operating range) in any room.

In order to accomplish equalization a design procedure had to be developed. The design procedure, which will be described later in much more detail, required accurate generation of pink noise, design and generation of a filter bank, spectral analysis of a signal from a microphone, real time analysis of the power spectrum and on the fly adjustments of the filter bank.

A RTA-DEQ has practical applications in any environment where accurate frequency representation of a sound signal is required. The DEQ could be a dedicated device that remains in-line with all sound equipment, constantly filtering the unamplified music signal or it could be used as test equipment to adjust a cheaper non RTA equalizer. Using the DEQ as test equipment would require visualization of the filtering being applied to the incoming signal which could be displayed on the unit itself or by outputting filter coefficients to a computer. The latter method of visualizing the filters was chosen for this project.

2. Description of Implementation

2.1 Input/Output

The basic input and output of the DSP was accomplished by recording 1024 input samples at a time sampled at 44.1KHz. The output of the DSP was also done 1024 samples at a time at a sampling rate of 44.1KHz.

2.2 High Frequency Analysis

The frequency analysis used to create the real time analyzer consisted of

the collection of 1024 samples of data sampled at a rate of 44.1KHz at a time. This data is run through the assembly based FFT algorithm provided in labs 4 and 5. The magnitude squared of the FFT output provides a view of the power spectrum. This representation is a bit inaccurate to analyze due to the limited sampling of the input data so to give a more sensible estimate the power spectrum output is passed through a 4^{th} order butterworth lowpass IIR filter at 4KHz twice to smooth out the spectrum and provide a better estimate of the average power. The choice of the filter frequency was chosen solely on the fact that this filter was already designed for the decimation algorithm and it provided adequately for what was needed of it in this situation. The frequency bins provided by this method were linearly increasing bins in 43Hz increments spanning from 0 to 22025Hz.

2.3 Low Frequency Analysis

The low frequency analysis was performed very similarly to the high frequency analysis except it involved in total a sampling of sixteen times the input samples, or 16,384 samples. Data was still received in 1024 chunks whether it be from the pink noise generator or input from the microphone, but it was then filtered using a 4^{th} order lowpass IIR filter at 4KHz already mentioned to prevent aliasing and then decimated by 4 which was done by simply keeping only every forth value of the input sequence. This decimated input was stored, and the procedure run another three times. The result of this is 1024 samples with a sampling frequency $1/4^{th}$ of the original sampling frequency of 44.1KHz. The input was further decimated by repeating the exact same process described

for the four times decimation but this time applying it to the already decimated input. So the decimated input of 1024 samples was created four times, and each time it was further decimated by four, stored, and as a result, 1024 samples of a sixteen times decimated signal was created as input. The same 4^{th} order filter was used for each decimation level. The reason for the two stage decimation and not a single sixteen times decimation stems from the fact that the anti-aliasing lowpass filter for a sixteen times decimation would have to have a very low cutoff frequency and as a result the design of a stable filter would require the input to be scaled down significantly, resulting in a large loss in precision. The two stage implementation with filters at 4KHz provided adequate performance. The result of all of this was an input that could be run through the same FFT routine for the high frequency analysis but this time providing 2.7Hz bins which spanned from 0 to what we considered to be a usable, meaning not substantially effected by the 4^{th} order filters, 500Hz.

2.4 Peaking Filters

The peaking filters were implemented using 2^{nd} order IIR filters in direct form 1. A bank of filter coefficients for 31 filters with ten different attenuation levels was precomputed in matlab and loaded into data memory on the DSP. Each filter was located 1/3 of an octave apart and the attenuation levels for each filter ranged from -2dB to -20dB. For a more detailed description of the calculation of filter coefficients see appendix B. A power spectrum threshold level was chosen and if the power spectrum of the microphone signal peaked above the threshold at any octave, the IIR filter's attenuation level at that octave

was increased by selecting appropriate coefficients from the list in data memory. The filters were applied in series meaning that the output of one filter was the input of the next filter.

Choosing the correct coefficients involved passing the filters octave number and attenuation level from a C environment to an assembly environment. In order to accomplish this the attenuation level index (0-9) was added to 128*octave number (0-30) and passed from C into assembly through accumulator A. Once in assembly, accumulator A was ANDed with the decimal equivalent of 15 and the result was stored. A repeat block statement was used to move the coefficient pointer to the correct level of attenuation. Next, accumulator A was shifted to the right 7 times (ie dividing by 128) so that the least significan bits now contained the octave index of the filter to be applied. The least significant bits of accumulator A were used to move the coefficient pointer in a repeat block statement so that it now pointed to the filter coefficients at the correct octave AND at the correct attenuation level.

2.5 Pink Noise

Originally the Voss-McCartney algorithm was proposed to generate pink noise. The Voss-McCartney algorithm involves summing multiple (between 8 and 10 for accuracy) white noise sources. Each white noise source is to be updated at one half the rate of the previous source[1]. This algorithm, however, was not used. The accuracy of the pink noise generated, about 1dB, was not high enough for analysis purposes. Instead, a 3rd order IIR filter was used to filter

1 "The spectrum Produced by the Voss-McCartney Pink Noise Generator" http://www.firstpr.com.au/dsp/pink-noise/allan-2/spectrum2.html

white noise into pink noise. A -3dB per octave roll off, plus or minus 0.3dB, was achieved by alternating real poles and zeros. Since all the poles and zeros were real, the 3rd order filter was implemented using three cascaded 1st order IIR filters [2].

*Poles and corresponding zeros of the 3rd order pinking filter

Pole	*Zero*
.99572754	.98443604
.94790649	.83392334
.53567505	.07568359

2.6 Real Time Analysis

The real time analysis implemented on the DSP functions similarly to vu meters selected at different frequencies but does not use multiple bandpass filters. It functions by inputting the power spectrum data, calculating the average power for 1/3 octave bands, and then scales the output according to how many frequency bins were used to create the average. This allows the output bands of the RTA to provide a view of the power per 1/3 octave, which can be observed by the user, and it used by the equalizer when the microphone input is used to calculate which filters need to be turned on and or off and have their gain factors changed.

2.7 Equalizer

The equalizer in the project functions by observing the RTA output and reognizing which frequency bands need alterations. There is a configureable threshold value and when the averaged output of any particular band goes above

2 "DSP Generation of Pink Noise" http://www.firstpr.com.au/dsp/pink-noise/#Allan

this threshold, the equalizer increases the gain applied to the particular filter that controls that band. If the gain is at the maximum value, the equalizer ignores trying to equalize this band. Once there is a change made in the filter bank, new input is gathered and a new average is computed. This whole process repeats a specified number of times to achieve the desired performance and then haults equalization. When this process finishes IO is still active but none of the filters are changed.

2.8 Code Description

The code of the project is a mix of assembly and C. All of the filters, the noise generator, and a few for loops involved in the decimation sequence are in assembly. This allows the code to be written efficiently so that real time performance was realizable. All I/O, the power spectrum calculation, and the internal loop that controls the equalization and decides when the equalization process is finished was put into the main file that is written as C. The main file's name was kept as lab4bmain.c because the I/O from lab4 was basically copied. All other functions such as decimation, spectral analysis, averaging, and band calculations were put into separate C files and #included in the main file to ease the organization of the code. Coefficient files for the 4^{th} order IIR filter were placed in a separate file and copied into the appropriate assembly routines, and the peaking filter coefficients for each and every filter were stored in one large file also included in the appropriate peaking filter assembly code.

3. Relevant Theory

3.1 Equalizers

The equalizing function attempts to mimic the efforts of analogue and digital 31 band equalizers. Bands are spaced 1/3 of an octave apart and use a constant Q. The decision to function this way was not baseless. This implementation follows the design of commercially available equalizers. As such it allows one to take the results achieved on the DSP and dial in a 31 band equalizer to the same settings as the DSP resulted in. This allows the DSP to not be required in the circuit once the response is analyzed and corrected.

3.1 Peaking Filters

In order to provide adequate performance and a result that could be reproduced on a simple graphic equalizer a reproduction of the methodology used by many graphic equalizers, analogue or digital, was used. The peaking filter was used as it is the primary type of filter used in real world devices. It allows the gain to be altered by a simple change in coefficients, it has an adjustable bandwidth, and it also functions with a constant constant- Q or quality factor. Without a constant Q, filters would have a great deal of overlap when using low gain settings and a very narrow bandwidth when using high gain settings. This results in unpredictable behavior when combining a large amount of filters and is especially difficult to plan for when devising an algorithm to correctly alter filter coefficients to shape a transfer function. Constant- Q filters such as the peaking filters used however do not suffer from this problem, and as such the Q factor is selected and it remains the same no matter what gain value is selected [3]. The filters were arranged in a series manner meaning one filter's output was the input to the next. In an analogue circuit this would not be

3 “Constant- Q Graphic Equalizer” http://www.rane.com/note101.html

acceptable due to noise buildup, however in the digital world, it is less of a problem and allows for phase relationships to be ignored. A parallel filter bank for example would have to be designed to consider the possible phase relations that would exist between adjacent filters[4].

3.2 Pink Noise

The chosen signal to equalize was pink noise. Pink noise has equal energy per octave which approximates a roll off of -3dB/octave. Sending pink noise through the speakers and then equalizing the frequency response of the speakers so that the microphone input is receiving pink noise (or at least close to it) allows for easier comparison upon breaking the power spectrum of the microphone signal into 1/3 octaves. Any deviation from a flat power spectrum across the octaves should be equalized.

4. Pitfalls

The project as a whole turned out to be difficult to implement on the used DSP. The precision allowed under 16bit integer arithmetic proved to be a large obstacle in the design and quality of the final implementation. A final working product that provided what was originally intended was reached, however its performance was impacted by the aforementioned obstacle.

Most evident was the inability to create working filters at frequencies of 8KHz and 10KHz. The filters at these frequencies had stability issues when implemented on the DSP. The calculation of the coefficients for these two particular filters met a sign change because of a cosine term involved in the filter

4 [music-dsp] “Constant Q EQ” http://aulos.calarts.edu/pipermail/music-dsp/2004-February/026154.html

coefficient calculations and because of the lack of precision on the DSP, neither of filters at these two frequencies would function correctly. An attempt to move the center frequencies of the two filters to 7KHz and 9KHz still resulted in the same stability issue. Also, changing the original filters Q values had no effect on stability. As a result of this problem, the 1/3 octave frequency bands centered around 8KHz and 10KHz were left out of the equalization process.

Additionally, the low frequency filters and analysis originally intended to be in the design were left out for numerous reasons. First, the mixers used to amplify and mix the microphone inputs contained high pass filters at 80Hz. These filters were removable but upon inspection, they could not be removed without permanently altering the PCB and surface mount components on the mixer circuit itself. So processing anything below 80Hz was not possible. Also, the DSP precision came into play again. Filters below 250Hz would not run without stability problems because of the lack of precision offered by the 16bit integer precision available. As such, frequencies below 500Hz were left out of the final implementation to simplify the procedure. Originally from 25Hz to 400Hz, the input from the microphones was decimated 16 times and then analyzed to provide the ability to equalize from the previously mentioned range. When it was discovered that only 250Hz and above could be reliably processed, it was decided that the three frequency bands that could be reliably analyzed and equalized was not worth the processing cost and added algorithm complexity required so this portion of the project was left out all together.

As a final failure of the project, the equalized output of the DSP contained

artifacts of the filtering. It was evident and audible by the ear that the output contained a periodic whooshing sound of sorts. This artifact was not present in the initial pink noise provided to the amplifier but as the number of filters inserted into the signal path increased the noise became more apparent. This problem again stems from the limited precision offered by the DSP platform. The problem was verified to not result from the analysis algorithm. The problem was a result of the filters and was tested by inserting filters one after another in a test program while watching the output as more filters were inserted into the signal path. Limiting or decreasing the input's magnitude helped decrease the noise, but this could not fix the problem because precision was lost upon scaling down the input.

These were the extent of the failures of the project. As for the accomplishments and working components, the following parts of the project were successfully implemented. White noise and subsequent pink noise generation was done successfully. Power spectrum analysis via the FFT was accomplished. For higher frequencies, 500Hz – 16KHz, this was done by performing the FFT of the 1024 input samples which resulted in approximately 43Hz frequency bins, then calculating the magnitude squared of the result, low pass filtering this result, and then grouping the filtered output into octave bands, scaled to represent the power per 1/3 octave. For the lower frequencies, the input was decimated by 16 to give approximately 2.7Hz frequency bins and then the same procedure as for the higher frequency analysis was performed. All of the filters needed for equalization were successfully created and verified

via matlab and implemented on the DSP. There functioning was a bit less than ideal though as discussed previously. The algorithm to analyze the power spectrum in the 1/3 octave frequency bands was also created and worked. The ability to insert and change the gain of the designed filters in real time was also implemented and working although only in discrete steps. Finally, to wrap the entire project together into performing what was designed for, an algorithm for analyzing the power of each band and performing the necessary alterations to each filter was implemented and worked correctly.

5. Extensions

Given more time, the project had a few extensions that would have been useful. One such extension is to allow, once the frequency response is analyzed and corrected, input from a music source to be switched automatically to the input of the DSP. This input would be passed through the filter bank as configured, and would provide room corrected equalized music output. Another extension that was not implemented that should be is to output the gain coefficients somehow. Ideally as a stand- alone unit the coefficients could be sent out through the serial port to an LCD controller and output to a display. This would be indicative of what a commercially available product would be expected to do. In lab it would be easier and more feasible however to output the final results to matlab for recording. Either way provides the user with easily viewable output and is a worthwhile extension to the project.

Another beneficial extension to the project would be the ability to calculate peaking filter coefficient on the fly by the DSP. This would allow for

much more precise levels of attenuation. Attempts were made to accomplish this, but converting float data types to integer data types proved difficult and inexact due to what appeared to be rounding issues on the DSP and how the compiler dealt with the conversion. The idea was scrapped due to time constraints.

6. Testing Methodology

6.1 Pink Noise Testing Methodology

In order to test the reliability of the projects pink noise generator a commercial DEQ's pink noise generator (Behringer DEQ2496 [5]) was used for comparison. Comparisons between pink noise signals were done in the time domain through listening tests and also in the frequency domain via spectral and power analysis. The comparisons showed a very strong similarity between the two pink noise sources.

6.2 Peaking Filter Testing Methodology

Each peaking filter was tested at each gain level by inputting a sine wave produced by a function generator to the DSP and sweeping from 10hz to 20khz. Every filter but one was turned off by setting the attenuation level to 0. This way each filter could be tested individually.

6.3 RTA Testing Methodology

Two methods were used for testing the real time analysis of the octave power levels. First a function generator was used to sweep a sine wave from 10hz to 20khz. The power spectrum per band was outputted to an oscilloscope.

5 http://www.behringer.com/DEQ2496/index.cfm?lang=ENG

As the frequency of the sine wave was changed, the power spectrum of the bands displayed on the oscilloscope changed accordingly. Also, a computer was used to play music through a sound system. A spectrum analyzer was run on the computer and the results were compared with the DSP's spectrum analysis being run on the input from the microphone. The results were close enough to conclude that the RTA was functioning correctly.

6.4 Testing of Overall Equalizer Functioning

Once all the RTA-DEQ units were each tested individually and deemed functioning, the system as a whole was tested. This was done by introducing a frequency hump of about +10dB centered at 3.3khz with an equalizer that was built into the audio amplifier being used to drive the loudspeakers being equalized. Refer to Appendix B for a block diagram of the whole system. The DSP was programmed to output pink noise and run a continuous equalizing algorithm. The power spectrum was displayed on an oscilloscope and updated after each successive equalization step. Slowly the hump at 3.3khz was lowered and the power spectrum of the signal being detected by the microphone began to match that of pink noise. There was also a significant audible change in the sound coming from the loudspeaker as the equalization levels changed. Upon completion of the equalization process the data memory of the DSP was inspected and indeed the 2.5khz, 3.2khz and 4khz filters had the highest level attenuation indexes. This means that the power spectrum at those three octaves rose above the threshold level the most and thus were attenuated the most. Filters at other frequencies were also activated, but the filters around 3.3khz had

the highest level of attenuation which was to be expected by injecting a +10dB gain at 3.3khz. Thus it was concluded that the RTA and equalizer were working properly, but not necessarily at their optimum levels of precision.