IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 9, September 2016

Design and Implementation of Spectrum Analyzer

Monali Chaudhari [1], Vaishali Kulkarni [2]

PG Student, Department of Electronics and Communication Engineering, SKN Collage of Engineering, Pune, India [1]

Professor, Department of Electronics and Communication Engineering, SKN Collage of Engineering, Pune, India [2]

Abstract: Over the last couple of decades, different types of spectrum analyser were developed for numerous applications in various fields, such as, measurement of spectral purity of multiplex signals, percentage of modulation of AM signals, and FM modulation characteristics and pulse-modulated signals. The spectrum analyser is also used to interpret the demonstrated spectra of pulsed RF radiated from a radar transmitter. A spectrum analyser is the primary tool used for studying the spectral anatomy of numerous electrical and optical waveforms. The present day's spectrum analyser is a vital portion of the engineer's toolbox. Which is used for the demonstrations of a power spectrum over a given frequency range, altering the display as the assets of the signal modification. Here, we present the basic idea around how the spectrum analyser works in a different environment, how it is efficient as compared to other type of measuring instruments and all additional characteristics of the spectrum analyser. In the presence of many signals, monitoring is very complicated to determine the performance of a system or device. The major components of spectrum analyser and how it is used during analysing the performance of devices.

Keywords: Signal, Spectrum, Period gram, Fast Fourier Transform (FFT).

I. INTRODUCTION

Spectrum analyser is used for the measurement of an amplitude of a signal with respect to frequency [1]. The inputs for the spectrum analyser are electrical, optical, vibration or acoustic signals generated from different source undergoes analysis for measuring the performance of the respective device. Spectrum analyser generally used for testing and high frequency measurements purpose [2]. Spectrum analyser covers the frequency range of up to 40 GHz and beyond [3]. It is generally used in all wired and wireless applications for development, installation, production and maintenance [4]. In case of microwave receivers, the signal strength is weaker than the noise and of having a lower frequency cannot measure such signal by using spectrum analyser in normal mode. It has a facility to convert a signal having very low or very high strength is converted into decibel (dB). In spectrum analyser, lower frequency signals are modulated to convert it into higher frequency components in the range of the analyser. We can also analyse the performance of the device or system by understanding the characteristics of noise and its types by comparing it with other signals [5].
The section 2 presents the description of the analyser with block diagram followed by Section 3 explains the background essential for the analyser designing. Section 4 presents real time implementation. The conclusion is drawn in section 5.

II. DESCRIPTION OF THE ANALYSER.

The data flow in this paper runs rather linearly. Initially, the audio signal is inputted into the system through the audio jack. This input signal is amplified and filtered to reduce the noise and then it has been sent to the ADC (Analog to Digital Convertor) of the microcontroller for the further processing. Here we have used the 32-bit ARM

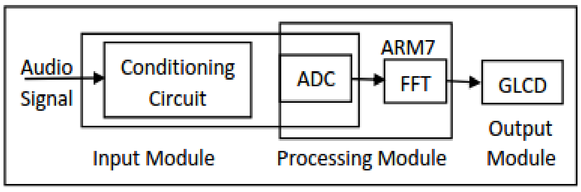

Fig 1. – Block diagram of audio spectrum analyser.

7TDMI RISM LPC 2148 microcontroller for the filtering. The LPC2148 will sample the audio signal at a constant rate and perform a Fast Fourier Transform (FFT) to convert the signal into the frequency domain. This frequency domain data is transmitted to the GLCD, which processes the information into a histogram visualization in real-time.

III.BACKGROUND

The most important and basic question is how the audio signal are sampled and processed into the frequency domain, it essential to understand many fundamental signal processing concepts. It is also important to understand how the sampling rate is determined. We used an ADC channel on the FFT to sample the analog audio input into discrete digital values. As per the Nyquist Sampling Theorem, the sampling rate essential be twice as that of the highest frequency of the sampled signalled to prevent aliasing which will distort the signal. Another

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 9, September 2016

most important factor is to understand how the acoustic signal is converted from its original time domain form to a frequency domain representation. The Fourier transform is a mathematical algorithm that translates a time domain signal into the frequency domain. Various types of Fourier transforms are accessible depending on whether the input is discrete or continuous and whether the output is to be discrete or continuous. Since we are dealing with finite digital systems, we elected to use the Discrete Fourier Transform (DFT) which converts a discrete time domain audio signal of a finite number of points N into a discrete frequency signal of N points, referred to as frequency bins afterword because each point represents a “bin” or range of frequency content. There are multiple algorithms available that are used to calculate the result of the DFT faster than calculating the DFT directly using its defined equation (while preserving the same exact result), recognised as the Fast Fourier Transform or FFT. Many FFT algorithms exist, but almost all of them uses recursive divide-and-conquer techniques that reduce the O(N2) computation time of the DFT O(Nlog2(N)). The large numbers of points N, permits the increase in speed is very significant in reducing calculation time particularly in software applications. Because of its recursive nature, it is usually necessary for N to be a power of 2. Use of a fixed-point number based FFT algorithm adapted for software to convert the discrete digital audio signal into a discrete frequency bins.

IV. REAL TIME IMPLEMENTATION OF THE ANALYSER

At the beginning, it is necessary to calculate the input bias voltage offset for the analog audio signal to meet the requirements. Hence, the two most important factor to be consider for the calculations: the ADC reference voltage and the range of the amplifier. The ADC reference voltage is the voltage corresponding to the maximum value of the ADC (1024) and any voltages above this returned the maximum value. The Texas Instruments LM358 op-amp is used as a main amplifier component along the ARM 7TDMI RISM LPC 2148 microcontroller for the digital signal processing as shown in fig 2. A non-inverting amplifier circuit, with the gain of the amplifying circuit is equal to 1+(300kΩ/100kΩ) = 4. Furthermore, a low-pass RC filter has used to remove frequencies above 4 kHz to reduce aliasing. The RC time-constant corresponds to a cut off frequency of about 3.7 kHz, but it has a slow drop off and the RC filter transfer function doesn’t reach values below 0.5 until much higher cut off frequencies, thus we wanted to set the cut-off slightly before our desired value of 4 kHz. The software portion consistsof a code for the FFT MCU. Here the oscilloscope required real-time operation that had periodic Interrupt Service Routines (ISRs) which occurred extremely precisely at equal time intervals. In the software part the FFT MCU code performs the ADC sampling of the audio signal, the FFT frequency conversion and UART transmission of the frequency data.

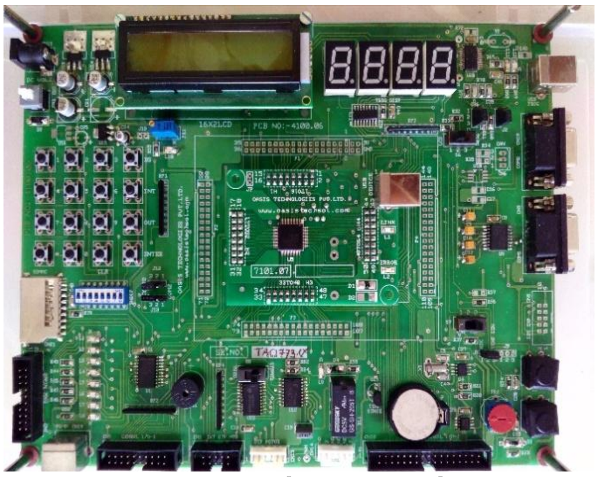
Fig 2 – ARM lpc2148 Board.

Regarding oursoftware setup, we used Keil uVision 4 to build and write our code, and to program processor. We also set our crystal frequency to 12 MHz.

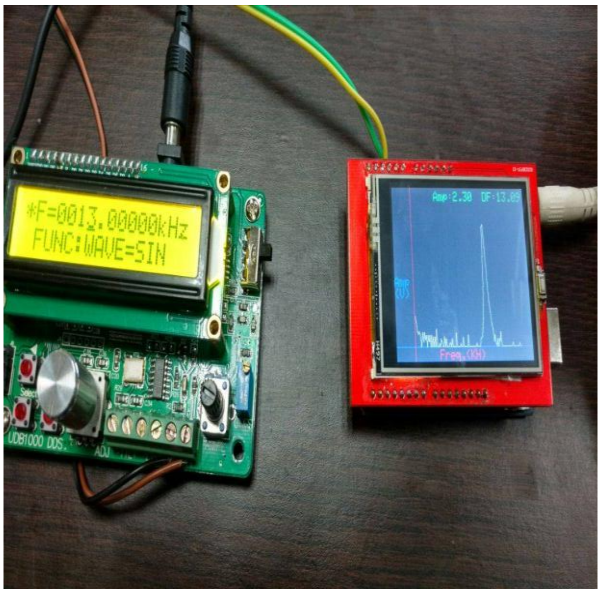

Fig 3 – FFT Spectrum Output display unit.

The FFT MCU does the sampling of the modified signal from the analog circuit whose output was fed into the ADC0. In order to get accurate results, it must ensure that the sample to the ADC port at precisely spaced intervals. Since the ADC is running at 4.5 MHz and it takes 10 ADC cycles having 2.44uS. The ADC set to ‘left adjust result’, so that all 10 bits of the ADC result were stored in the AD0CR register. At a sample rate of 8 kHz, an ADC value will be requested every 125uS, which means sufficient time should be there for a new ADC value to be ready for each requested time interval. The sample rate was set to be cycle accurate at 8 kHz by setting the 12 MHz Timer 0 counter to interrupt at 1500 cycles (12000000/8000=1500), and having the MCU interrupt to sleep (slightly before the main interrupt) to ensure that no other processes would be interfering with the precise execution of the Timer 0 ISR where the ADC is sampled.

FFT conversion of the Audio Signal to Frequency Bins, the ADC values are stored in a buffer of length 128 as fixed-point FFT code which is maximized at a 128-point operation, Because, the program may have crash due to memory overflow for any larger FFT. A graphical LCD (GLCD) for displaying the output shown in fig 3. As it is always good have the more number of points possible which results in more frequency bins and a more precise frequency representation of the audio signal. This value of 128 is parameterized as N_WAVE in the code. The code, with a subroutine called FFT fix, took two arrays of length 128, the real and imaginary components, and the base 2 logarithm of 128 for the number of FFT iterations (recursions) to complete. The code performs the forward transformation of the time domain to the frequency domain of the input real and imaginary arrays, and return the result in place in the same arrays that were inputted. The fixed-point 32-bit numbers are used, where the high order 16- bits represent the integer digits and the low order 16-bits represent the decimal digits. The code requires a table of length 128 of one succession of a sine wave to be executed in order to run the FFT and which is initialized in the main method. Since the input array is purely real, the 128 length zero vector has been initialised to represent the imaginary component of the input signal. It also included several fixed-point number operation macros that perform multiplication and convert various data types to fixed-point format. In the main, program continuously waited until the ADC buffer filled to 128 samples, and then it immediately starts the FFT operation. Initially, the imaginary buffers from the previous FFT operation makes zeroed out, and then ADC buffer copied into an array representing the real input. The window of real array with a trapezoidal mask with side-slopes of 32 points are used to remove any sharp cut offs at the end of the ADC buffer which may introduce a high frequency content. Because, ideally ADC input to be infinite and continuous but Fourier transform must be finite in implementation, and should have this is then shifted left by 16 to increase the values of the FFT output.

Therefore, the ADC values copied into bits 17-24 of the 32-bit number. Since all the ADC values are purely integers and the fixed-point FFT does not know whether the input is decimal valued or not, it does not really matter where data sits bitwise in the 32-bit fixed-point number since the end result will be read as an integer and not a fixed-point number. This real input array and a zeroed out imaginary input array are the inputs to FFT. Here we have only taken the magnitude information of the frequency content of the signal in the consideration. Thus, the magnitude of the frequency content by taking the sum of the squares of both the real and imaginary frequency outputs.

Theoretically to get the actual magnitude, it is necessary to take the square root of the sum of squares, but since it is just a scaling factor and too low in magnitude of the frequency content, left it as the sum of squares. In the first FFT code implementation lot of operations has been done with integer operations and not with the fixed-point operations, which implies incorrect frequency magnitude data and caused output to look "noisy". So, after switching to the fixed point operation by using the fixed-point macros, shows the much cleaner output and represents the actual frequency content of the audio signal. It is tested by hard coding a sine wave signal with known frequency and viewing the frequency output to see if there is only one non-zero frequency bin. Since the frequency content are mirrored through middle of the FFT output, when the input is purely real, only the first 64 points of the 128-point output is relevant and stored. These 64 points represents frequency bins. So as to only transmit and display 32 frequency bins, depending on the frequency range the to be display, either paired and combined adjacent bins to form 32 bins of 125 Hz resolution (0-4 kHz range), or only transmit the first 32 bins (0-2kHz range). Once this frequency magnitude content array prepared, the data is ready to transmit to the GLCD. Before, fully implementation of the code, the output of FFT tested by just outputting 32 bin values as a text string by using UART through USB port on PCB to PC by using a serial connection at 9600 baud rate and viewed the output using HyperTerminal. After the data transmission, the ADC index set back to 0, So that program could start to sample ADC values again into ADC buffer, overwriting the old buffer's values.

V. CONCLUSION

With the extremely satisfying final results we are able to meet the majority of our expectations set in the project. We successfully implemented a fully functional audio spectrum analyser which operates in real time mode and accurately displayed the frequency content with a histogram imagining of the input audio signal. By using standard 3.5mm audio jacks or small mice as analog to digital transducer along with very few components we have built a low cost, very economical audio spectrum analyser. Which may not only allow the users to view their favourite music in a fun and interactive fashion, but it also offers information about the music otherwise inaccessible.

REFERENCES

[1] S. M. Metev and V. P. Veiko, Laser Assisted Microtechnology, 2nd ed., R. M. Osgood, Jr., Ed. Berlin, Germany: Springer-Verlag, 1998.
[2] Morris Engelson, "Modern spectrum analyzer: theory and applications", Artech House (1984).
[3] A. A. Abidi, "High-frequency noise measurements on FET's with small dimensions" Electron Devices, IEEE Transactions, Vol. 33, Issue 11 (1986).
[4] Mark Pelusi1, Feng Luan, Trung D. Vo, Michael R. E. Lamont, Steven J. Madden, Douglas A. Bulla, Duk-Yong Choi, Barry Luther-Davies and Benjamin J. Eggleton, "Spectrum analyzer covers frequency range of up to 40 GHz and beyond. It is generally used in all wired and wireless applications for development, installation, production and maintenance, Nature Photonics 3, pp. 139 - 143 (2009).
[5] S. Thomas, N. S. Haider, "A Study on BASICS OF A SPECTRUM ANALYZER", IJAREEIE, Vol. 2, Issue 6 (2013).
[6] Frequency Domain and Fourier Transforms https://www.princeton.edu/~cuff/ele201/kulkarni_text/frequency.pdf