EE 318 EDL Report, Electrical Engineering Dept. IIT Bombay, submitted date 4/11/2004

Random Noise Generator

Group No.: D1
Sumit Punglia (01D07032), Rajat Bansal (01D07010), Bikramjit Walia (01D07016)
Supervisor: Prof. P.C. Pandey

ABSTRACT

This EDP aims at designing a random noise generator with digitally selectable bandwidth, spectrum and output level. The bandwidth range of the output spectrum is from 5 Hz to around 50 kHz. It has five different output levels and generates spectrum of pink noise, white noise and ⅓ octave noise. A pseudo random digital sequence is generated using dual 4-bit shift registers and then the digital sequence is converted into three different types of analog noises: white noise, pink noise and ⅓ octave noise by filtering the digital noise. Filtering is done by R-weighted network for realizing a discrete time FIR filter from the shift register taps. All the filter weights are normalized with respect to the clock frequency used for pseudo random generator. User interface is done through the de-bounce keys and serial port. An LCD display serves as the output.

CONTENTS

FIGURES

1. INTRODUCTION

Noise sources are basic building blocks for instrumentation and communication systems, as well as for testing. White noise is a random waveform with a flat frequency spectrum. For white noise, there is equal energy in a given bandwidth at any centre frequency. A closely related concept is pink noise which a sound that is perceptually white noise. Pink noise has the same distribution of power for all frequencies and is often referred to as 1/f noise for reasons which shall be explained later. Another noise is the ⅓ octave noise which is a band pass noise.

Conventional approaches to generate random signals use physical sources of analog noise. Since this approach obtains very low-amplitude noise, their practical use requires amplification, which produces significant spurious correlations. However our EDP generates the pseudo random bit sequence using dual 4-bit shift registers. Besides its obvious application as analog noise source, the PRBS generator can also be used for encipherment of data. They are used extensively in error correcting codes and are ideal for radar ranging codes. Once the noise generation is done, the noise shaping can be performed to generate analog white noise, pink noise and ⅓ octave noise.

It is remarkably easy to generate sequences of bits (or words) that have good randomness properties. These sequences are generated by standard deterministic logic elements (shift registers). They are in fact predictable and repeatable, although any portion of such a sequence looks like a random string of 0's and 1's. Here we aim to use dual 4 bit shift registers with feedback to generate digital noise. Simple low-pass filtering of the digital output of a pseudo random bit sequence can generate band limited white Gaussian noise.

We aim to achieve this using different resistance proportional to the filter weights and difference amplifier to sum up the analog signals. The design should also have a very good digital interface which determines the bandwidth, the output levels and the spectrum to be generated. Our aim is to design a product that can develop various types of noise which are amplitude selectable as well as frequency selectable and are interfaced to computer through serial port and has a LCD output display. Thus, making the product complete in itself.

2. DESIGN

2.1 Noise Generation

The design of Random noise shift generator consists of following stages:

2.1 .1 Dual 4-bit shift register

The IC chip we are using to generate random sequence is CD4015B. It is a dual CMOS four stage shift register. We are generating 31-bit long random sequence therefore we will be using 4 CD4015B chips. The clock given to the shift register is generated by the microcontroller interfaced to the shift register. We first generated a 4-bit shift sequence using one 4015 chip and we applied a tap on the 4-th bit so that the random sequence is continuous and is repeated after some time. Then a 7 bit sequence was generated by using both the shift registers in CD4015. Finally, four such chips were used to generate the 31 bit pseudo random sequence.

In order to minimize our hardware we used taps on 31 and 18. The sequence generated was pseudo random and it was seen that the sequence repeats itself after 2^31-1 bits.

Block Diagram

Also shown below is the block diagram (*fig* 2.1) for the dual 4 bit shift registers. The feedback has been taken using taps on 18th and the 31st bits. The feedback is provided via an exclusive NOR .The purpose of the XOR gate to scramble the feedback bits so that the data input to the shift register maybe considered to be random.

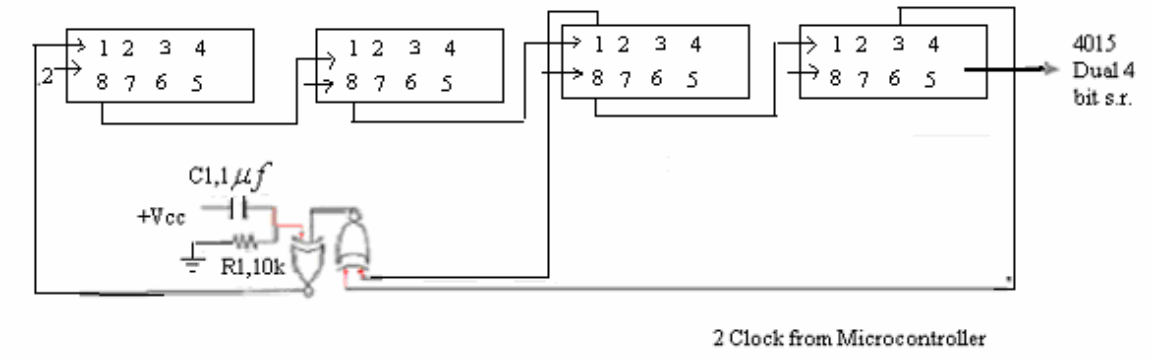


Fig 2.1 PBRS sequence

2.1.2 Noise Filters

The 32 bit output of the shift registers is fed into the summing opamps, with the resistor

weights corresponding to weights of the filter coefficients. The positive weighted inputs and the negative weighted inputs are summed separately. All the positive coefficient bits are summed in summing amplifier U1 and the negative bits summed in U2. Each summing amplifier has an opamp in the inverting adder configuration. The outputs of U1 and U2 are fed to the difference amplifier U3 (*fig* 2.2)

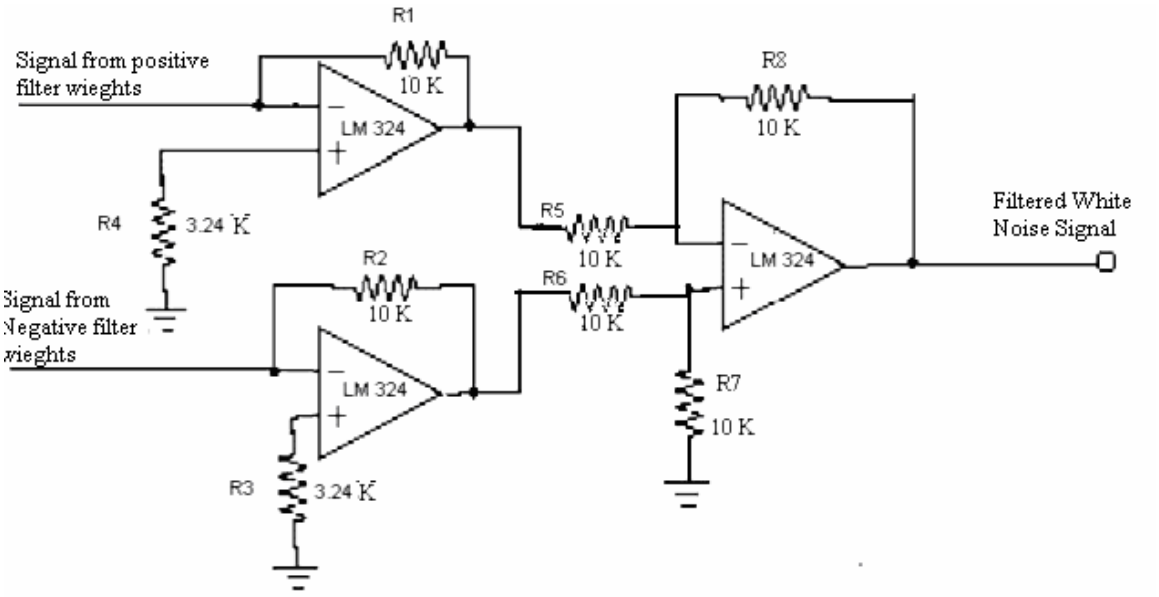


Fig 2.2 - Circuit for digital to analog conversion

2.1.3 Using the microcontroller to generate random sequence

We tried to generate the pseudo random bit sequence using a microcontroller. Thereby we wrote a code for the microcontroller so that the microcontroller gives us a PBRS sequence. The detailed code for the same is attached in Appendix A.

However our microcontroller was able to generate around 10 kHz frequencies which meant that the cutoff of the white noise generated was at maximum around 500 Hz. This prompted us to follow our initial approach of using the dual 4 bit shift registers as they were able to generate much higher frequencies(with cutoffs of around 50 kHz).

2.1.4 Interfacing Micro-controller for amplitude control

The microcontroller is interfaced so as to provide for amplitude control and spectrum selector. It drives a multiplexer which has on the output lines resistances connected in the form of a voltage divider circuit. Thus we are able to observe five different amplitude levels which are 5 V p-p, 4 V p-p, 3 V p-p, 2 V p-p and 1 V p-p.

The microcontroller is also a spectrum selector as it chooses either the white noise or the pink noise depending upon the user's choice.

As discussed above the microcontroller does the all essential job of providing a clock pulse to the shift registers. For this the microcontroller is programmed in the clock out mode.

2.1.5 **Power supply**

The power supply to the circuit is provided with a +/- 18 V transformer. A 7808 regulator and a 7809 regulator bring this voltage down to +/- 9 V, required to drive the opamps. A 7805 regulator brings down the voltage to 5V, which is required for the digital components (*Ref 2)*.

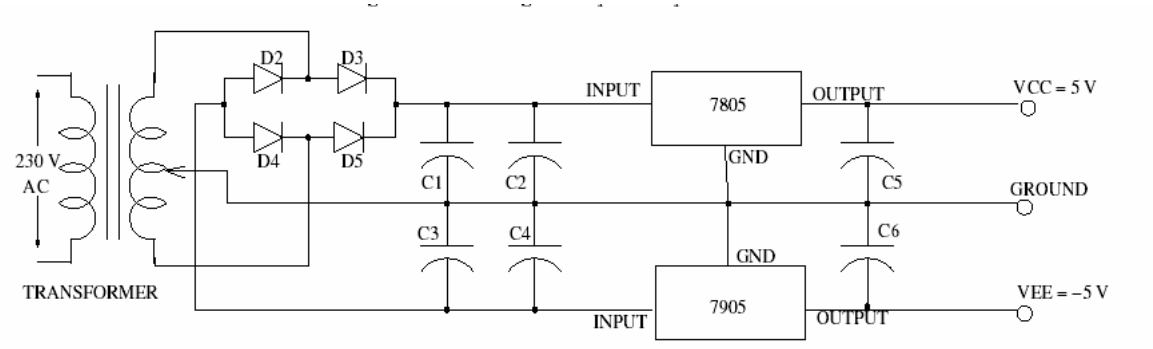


Fig 2.3 Circuit Diagram for Power Supply

2.1.6 **Multiplexer for control**

Maximum amplitude of the output was observed to be 5 V p-p. This was then distributed into different levels with the help of voltage dividers and the multiplexer powered by the microcontroller. The multiplexer depending upon the input from the microcontroller selects one of the voltage dividers so that the amplitude is varied accordingly. The spectra were observed.

2.1.7 **Audio amplifier**

The amplifier circuit used is designed using LM324. This stage is used to give a unity voltage gain to the output of the Amplifier and boost the current output through the push-pull transistor arrangement as shown in figure. The op amp is in the buffer configuration feeding the push-pull configuration (*fig* 2.4).

The diode D1 helps to reduce the cross over distortion which would be present otherwise if we had used only resistors to bias the transistors. The diode compensates the 0.7 v drop of the push-pull transistor and hence no cross over distortion occurs. The output of circuit is fed back to LM324, which was used at the starting of the circuit, in order to minimize the distortion in output.

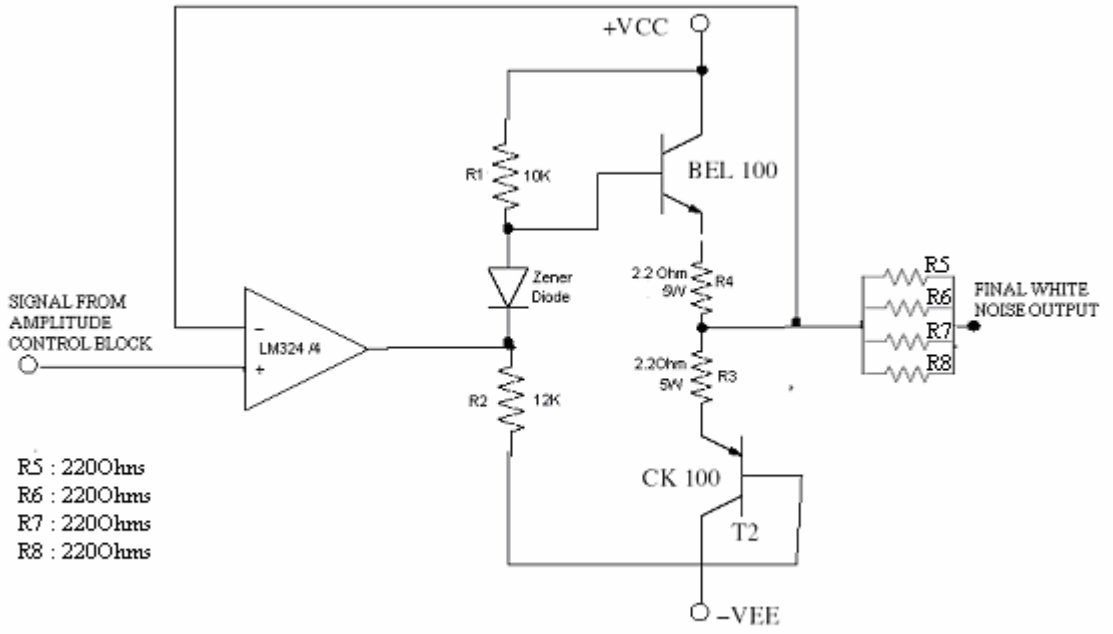


Fig 2.4 –Audio Power amplifier

2.2 FILTER DESIGN & NOISE SHAPING

2.2.1 Digital Filtering

The digital filtering is performed by taking an analog weighted sum of successive output bits (non recursive digital filtering). In this way the effective filter cutoff frequency changes to match changes in the clock frequency. In addition, digital filtering go to extremely low cutoff frequencies where analog filter become awkward.

In order to perform a weighted sum of successive output bits simultaneously, we simply look at the various parallel outputs of the successive shift register bits using resistors of various values into an op-amp summing junction. For a low-pass filter the weights are proportional to the (sin x / x). Note that some levels will have to be inverted, since the weights are of both signs (*fig* 2.6). Since no capacitors are used in the scheme, the output waveform consists of a set of discrete output voltages.

In this case a 32-bit shift register is connected with feedback from stages 31 and 18, generating a maximal length sequence with 2 billion states (at the maximum clock frequency the register completes one cycle in a 1/2 hour). The ratio of resistance should be of order of Sin(x)/x. We wrote a simple matlab program to determine the values of the resistances (*Ref 2)*.

The resistances were determined for white noise using the FIR filter design

theory. A detailed analysis of the same shall be provided in the subsequent sections. Then we plotted a graph (*fig* 2.5) using those resistances and obtained the Fourier transform of our digital low pass filter as follows.

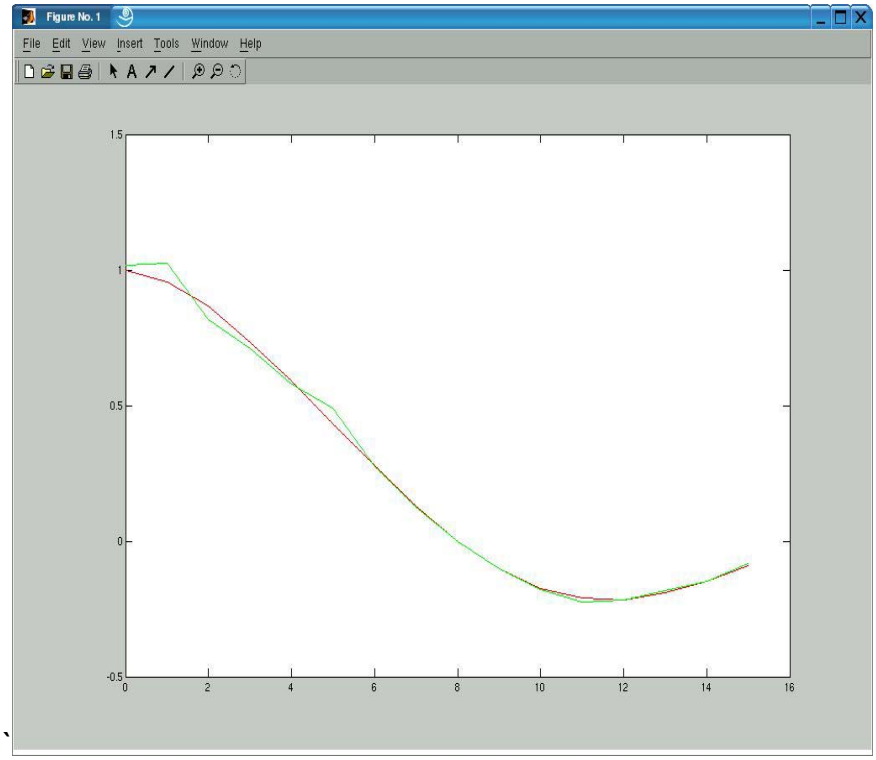


Fig 2.5 – Light curve denotes the actual values of resistances used and the dark curve denotes the calculated resistance values.

Similar analysis yields the filter weights for pink noise (section 2.2.5) and 1/3 dB octave noise (section 2.2.6).

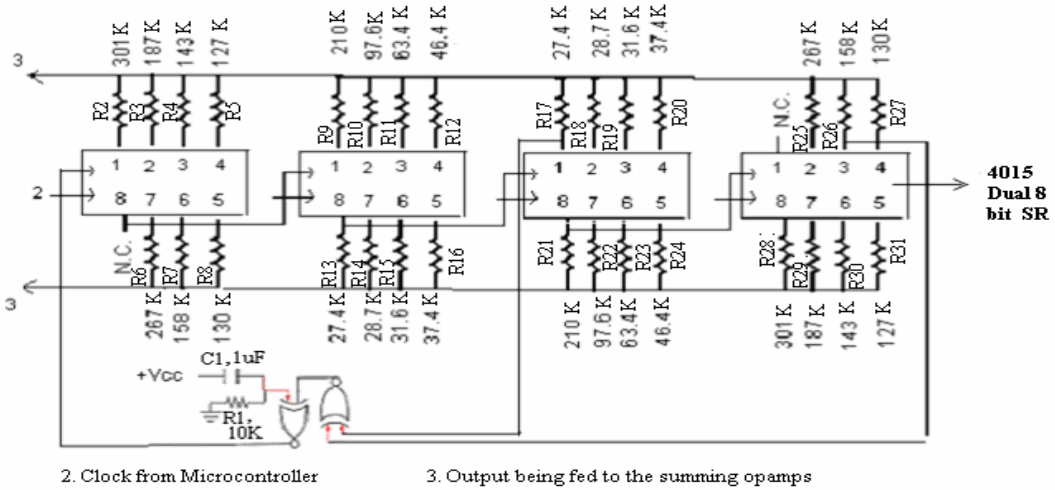


Fig 2.6 Digital filtering of PBRS sequence

2.2.2 Basic Properties of FIR Digital Filters

Digital FIR (finite impulse response) filters cannot be derived from analog filters, since rational analog filter cannot have a finite impulse response. In many digital signal processing applications, FIR filters are preferred over their IIR counterparts.
The main advantages of the FIR filter designs over their IIR equivalents are the following:
1. FIR filters with exactly linear phase can easily be designed. This simplifies the approximation problem, in many cases, when one is only interested in designing of a filter that approximates an arbitrary magnitude response.
Linear phase filters are important for applications where frequency dispersion due to nonlinear phase is harmful e.g. speech processing and data transmission.
2. There are computationally efficient realizations for implementing FIR filters. These include both non-recursive and recursive realizations.
3. FIR filters realized non-recursively are inherently stable and free of limit cycle oscillations when implemented on a finite-word length digital system.
4. Excellent design methods are available for various kinds of FIR filters with arbitrary specifications.
5. The output noise due to multiplication round off errors in an FIR filter is usually very low and the sensitivity to variations in the filter coefficients is also low.

2.2.3 Frequency Response of Linear Phase FIR Digital Filters

One of the simplest types of filters that we can design is an FIR filter with linear phase. Also, only FIR filters can be designed to have linear phase response. It can be shown also that IIR filters cannot have linear phase.

2.2.4 Uniform Frequency-Sampling Method

The frequency sampling method for FIR filter design is basically the technique we shall be using to generate the filter weights. Here, we specify the desired frequency response at a set of equally spaced frequencies .We solve for the unit sample response *h (n)* of the FIR filter from these equally spaced specification (*ref* [4])

We shall begin with the frequency response of the FIR filter

$$y(n) = \sum_{m=0}^{M-1} h(m)x(n-m)$$

$$H(e^{iw}) = \sum_{n=0}^{M-1} h(n)e^{-iwn}$$

If the frequency response of the filter is specified at the frequencies

$$w_k = 2\pi k / M$$

then we can obtain

$$H(e^{jw_k}) = H(w_k) = H(2\pi k / M) = H(k)$$

and

$$H(k) = \sum_{n=0}^{n=M-1} h(n) e^{-i 2\pi nk / M} \qquad \text{For k=0, 1, 2, 3…….M-1}$$

The above expression of $H(k)$ is nothing but the DFT of $h(n)$. Thus $h(n)$ may be visualized as the inverse DFT of $H(k)$.

Once we obtain the above filter coefficients, then it is easy to design the filter as the resistances that we use in digital filtering are proportional to the inverse of the real part of $h(n)$.

2.2.5 Pink noise

2.2.5.1 Characteristics of pink noise

"Power" can be defined as the average power or energy contained in a signal over a long period of time. White noise has the same distribution of power for all frequencies, so there is the same amount of power between 0 and 500 Hz, 500 and 1,000 Hz or 20,000 and 20,500 Hz. Pink noise has the same distribution of power for each octave, so the power between 0.5 Hz and 1 Hz is the same as between 5,000 Hz and 10,000 Hz.

Since power is proportional to amplitude squared, the energy per Hz will decline at higher frequencies at the rate of about -3dB per octave. To be *absolutely* precise, the roll off should be -10dB/decade, which is about 3 dB/octave. Pink noise is often known as a 1/f noise. It is a type of noise whose power spectra *P (f)* as a function of the frequency *f* behaves like: $P(f) = 1/(f^a)$, where the exponent '*a'* is very close to 1. Thus, pink noise is also known as "1/f noise" (*ref* [9]).

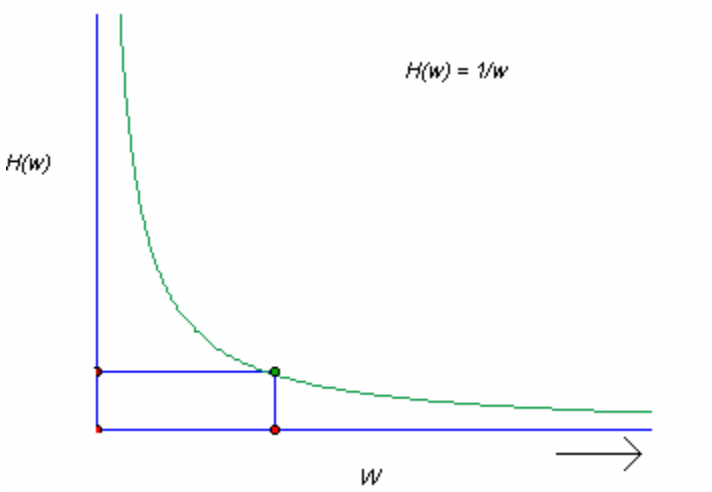


Fig. 2.7 Plot of pink noise spectrum

2.2.5.2 Generation of pink noise

We are using frequency sampling method and FIR filters to generate coefficient for pink noise. We know that pink noise is also known as "1/f" noise. In linear phase FIR filters, we specify the desired frequency response, H(w) at a set of equally spaced frequencies and solve for the unit sample response h(n) of the FIR filter from these equally spaced frequency specifications.

For pink noise, we know that, $H_w = \frac{1}{w}$

Therefore, $w_k = \frac{2\pi k}{M}$ Where M denotes the number of samples taken.

Now, the filter coefficients $h(n)$ can be obtained as the inverse DFT of the frequency samples as

$$h(n) = \frac{1}{M}\sum_{k=0}^{M-1} H(k) e^{j(2\pi/M)nk}$$

where $H(k)$, k=0, 1, 2…….$N-1$; are the samples of the ideal or target frequency response.

It can be shown that for a linear phase filters, with positive symmetrical impulse response, we can write (for N even),

$$h(n) = \frac{1}{N}\sum_{k=0}^{N/2-1} 2\left|H(k)\right|\cos[2\pi k(n-\alpha)/N] + H(0)$$

where $\alpha = (N-1)/2$. For odd values of N, the upper limit in the summation is $(N-1)/2$. The resulting filter will have frequency response that is exactly the same as original response at the sampling instants. To obtain the good approximation to the desired frequency response, clearly we must take sufficient number of frequency samples.

The coefficients for pink noise are calculated using a matlab program. The resistance values that must be used for pink noise generation are the inverse of the calculated coefficients.

2.2.5.3 Uses of pink noise

The main use of pink noise is as an audio signal, to be used directly, to be filtered or to be used to modulate something. Pink noise way below 1 Hz can be used as control signal for simulating randomly fluctuating aspects of music. It might also be desirable to specify an upper bounding frequency, to reduce computational load where high frequencies were not required, here pink noises can be easily used.

2.2.6 ⅓ Octave noise

2.2.6.1 Pass band noise

An octave is a 2:1 ratio of freq. A 1 octave filter requires 1 filter to cover a 2:1 freq range and a ⅓ octave filter requires 3 filters to cover a 2:1 freq range.

The frequency span is the range between the lower corner freq and the upper corner freq. Given a specified center freq where the center freq is the geometric mean of the corner frequency, the lower corner freq of the filter = $0.5^{1/2k}$ times the center freq where k=1 for octave filter and three for a ⅓ octave filter.

This means that the lower corner freq for a ⅓ octave filter = $0.5^{1/6}$=0.8901 times the centre freq and the upper corner freq = $2^{1/6}$ = 1.225 times the center freq.

Similarly, for a 1 octave filter, the lower corner freq= $0.5^{1/2}$ = 0.707 times the center freq and the higher corner freq = $2^{1/2}$ =1.414 times the center freq.

⅓ db is a passband noise whose filter weights are again worked out using the frequency sampling method discussed above. Here the spectrum in the frequency domain would resemble *fig* 2.8

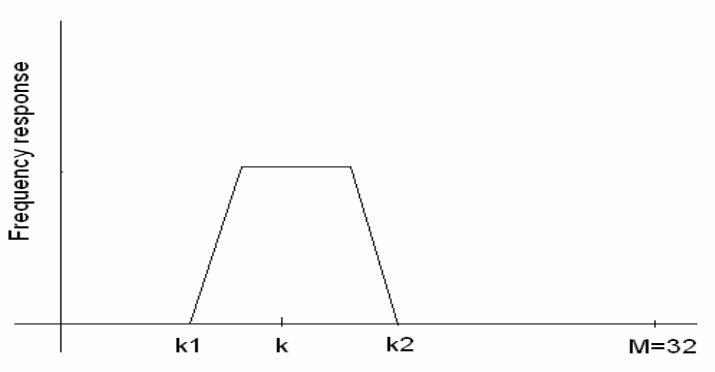


Fig 2.8 Spectrum of ⅓ octave noise

On the x axis M =32 corresponds to Ω of 2π radians.

For ⅓ octave noise, the special property is that $k_2 / k_1 = 2^{1/3}$

Now the value of 'k' has to be chosen in an appropriate manner. 'k' must no be very low as the filter we design then would be equivalent to a low pass filter. 'k' must not be very high as well. A conservative estimate of k can be given by either k=$M/4$ or k=$\sqrt{M/2}$.

In our calculations we took k as k=M /4 i.e. k =8.

Therefore k_1 shall then reduce to $k_1 = \dfrac{M}{4(2^{1/6})}$ and k_2 to $k_2 = \dfrac{M(2^{1/6})}{4}$.

Thus approximating the discrete frequency response, we worked out the filter coefficients $h(n)$ by calculating the inverse DFT and thus designed the filter.

2.3 USER INTERFACE

2.3.1 Interfacing serial port

The RS232 signals are represented by voltage levels with respect to a system common (power / logic ground). The "idle" state (MARK) has the signal level negative with respect to common, and the "active" state (SPACE) has the signal level positive with respect to common. RS232 data is bi-polar; (+3 TO +12 volts) indicating an "ON or 0-state (SPACE) condition" while A (-3 to -12 volts) indicating an "OFF" 1-state (MARK) condition. The microcontroller (AT89C52) has four modes of serial data transmission that enable data communication. Modes are selected by setting the mode bits SM0 and SM1 in SCON. Baud rates are set by using timer 1 and the serial baud rate modify bit (SMOD) in PCON. Here in our device we have established communication at a Baud rate of 2400. Timer 1 is used in timer mode 2 as an auto load 8 bit timer to generate the 8 bit frequency as

$$f_{baud} = \frac{2^{V_1} f_{osc}}{32d(12d)(256d - (V_{TH1}))}$$

where V_1 is the value of the SMOD register and V_{TH1} is the value of Timer 1.

For serial data transmission mode 1, a character uses 10 bits: start, 8 code bits and stop (as shown in *fig* 2.9).

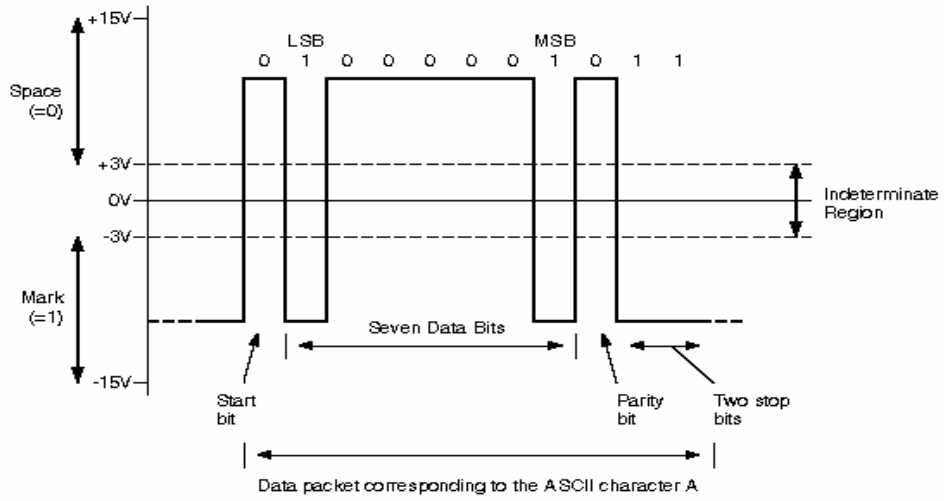


Fig. 2.9 Serial data transmission with start and stop bit

Here while communication with the serial port, the aim is to make to computer think it is talking to a microcontroller. Any data transmitted from the computer must be received by the microcontroller thus TD (Transmitted data) is connected to RD (Received data). The microcontroller must have the same set-up thus RD is connected to TD. Signal Ground (SG) must also be connected so both grounds are common to each computer.

The Data Terminal Ready is looped back to Data Set Ready and Carrier Detect on PC and microcontroller. When the Data Terminal Ready is asserted active, then the Data Set Ready and Carrier Detect immediately become active. At this point the computer thinks the microcontroller to which it is connected is ready and has detected the carrier of the computer (*ref* [10]).

When the computer wishes to send data, it asserts the Request to send (RTS) high .As soon as it receives a Clear to Send (CTS), it transmits the data. The code written for serial port communication meanwhile polls the UART, to see if any new data is available.

2.3.2 **Design of the control toolbox:-**

We have also designed the control toolbox for our serial port application. This toolbox has nine selectable frequencies, five amplitude levels and the displays for spectrum selection. This is used to communicate with the microcontroller as it sends the requisite bits (as asked by the user) to the microcontroller. A snapshot of the control toolbox is shown in *fig* 2.10

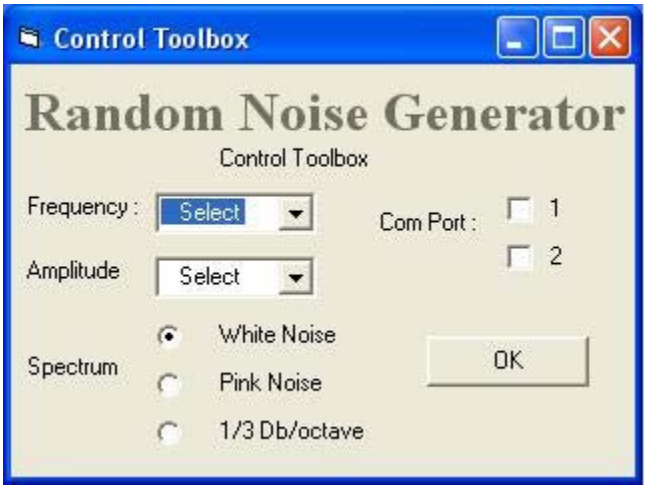


Figure 2.10 snapshot of the control toolbox

2.3.3 **MAX232 for interfacing the serial port**

A standard serial interfacing for PC, RS232C, requires negative logic, i.e., logic '1' is -3V to -12V and logic '0' is +3V to +12V. To convert TTL logic, say, TxD and RxD pins of the uC chips, we need a converter chip. A MAX232 provides 2-channel RS232C port and requires external 10uF capacitors.

The MAX232 is a dual driver/receiver (*fig* 2.10) that includes a capacitive voltage generator to supply EIA-232 voltage levels from a single 5-V supply. Each receiver converts EIA-232 inputs to 5-V TTL/CMOS levels. These receivers can accept ±30-V inputs. Each driver converts TTL/CMOS input levels into EIA-232 levels. The MAX232 requires 5 external 1uF capacitors. These are used by the internal charge pump to create +10 volts and -10 volts.

It also helps protect the processor from possible damage from static that may come from people handling the serial port connectors. The MAX232 includes 2 receivers and 2 transmitters so two serial ports can be used with a single chip.

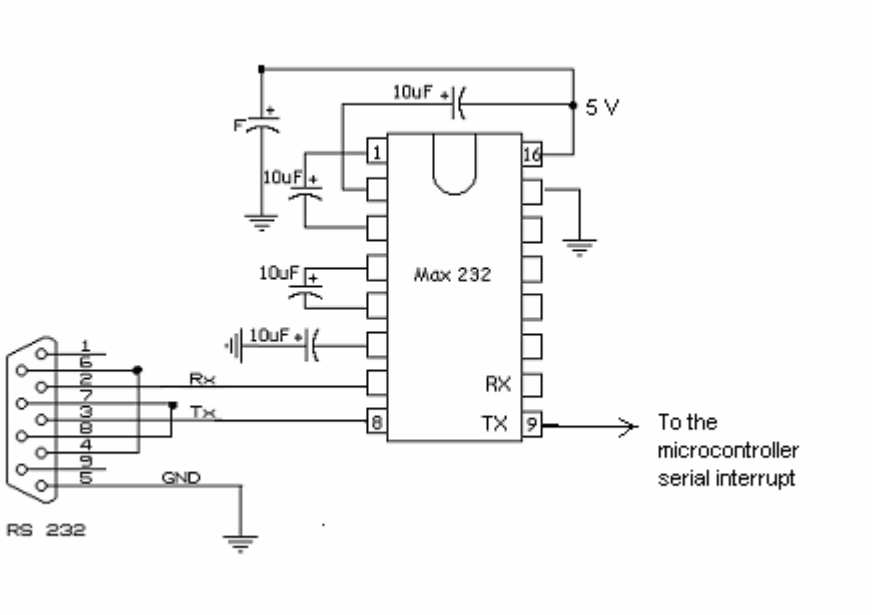


Fig 2.11 A microcontroller interfaced with serial port through MAX232

2.3.4 **Switches and Keys**

There were four keys used i.e. for spectrum control; bandwidth control and amplitude up/down. The keys that were being used for the user interface were not working properly. A time delay of 1 second was provided once a key is pressed. For this time, the same key pressed again would not show any response.

2.3.5 **LCD display**

For displaying the output we have used a LCD display. The LCD display displays the bandwidth, amplitude of the output and the spectrum that is being currently observed. There are three control lines on an LCD (the Register Select, the Read/Write and the Enable). Besides that, the LCD is connected to the microcontroller via its data port. The LCD has been programmed in the 8 character x 2 line format.

To send instructions to the LCD module, the *Register Select* line must be low. After we place a data byte on the data lines, we must then signal to the LCD module to read the data. This is done using the *Enable* line. Data is clocked into the LCD module on the high to low transition. We then wait for a delay, and return the line to a high state ready for the next byte. After we initialize the LCD Module, we want to send text to it. Characters are sent to the LCD's *Data Port.* Then we set up another *for* loop to read a byte from the string and send it to the LCD panel. This is repeated for the length of the string.

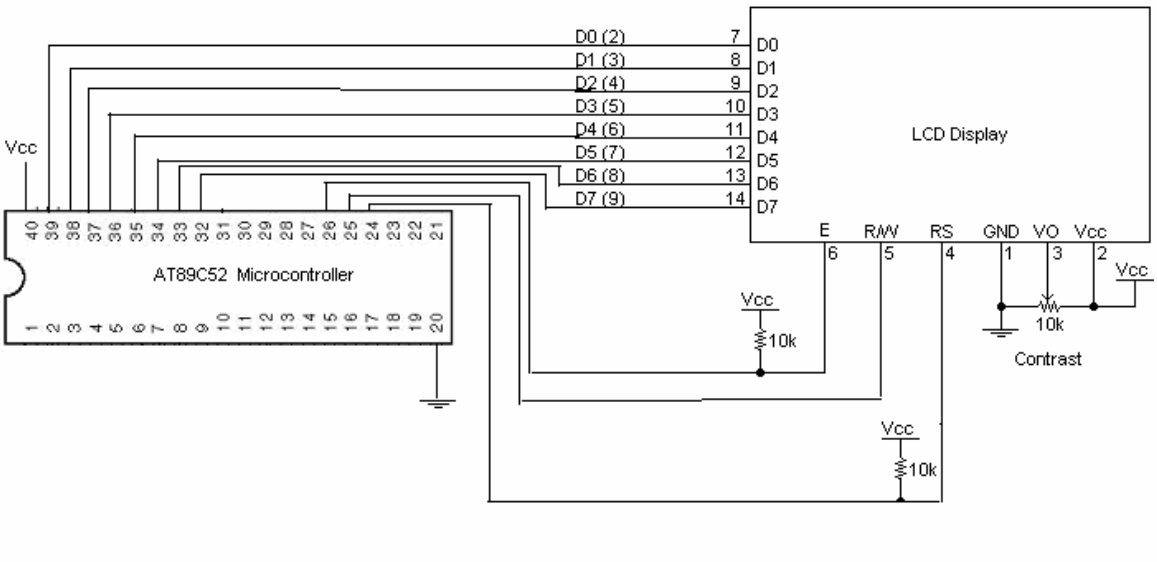


Fig 2.12 Interfacing LCD display with micro-controller

2.4.1 FINAL CIRCUIT SPECIFICATION:-

The final block diagram is as shown below (*fig* 2.13). The final circuit diagram is as shown in *fig* 2.14 and *fig* 2.15.

1) Dual 4 bit shift registers:-

This block generates a pseudo random binary sequence. It has 31 bits (4 shift registers). The length of the sequence is (2^ (31))-1. The clock to these registers is powered by the microcontroller which provides clocks from 0Hz to 50 MHz.

2) Digital filtering:-

The values of the resistances proportional to the filter weights are worked out using the Matlab code. The feedback from the 18^{th} and the 31^{st} pin are sent through the exclusive NOR gate to the shift register's data pin.

3) Noise filters:-

This guarantees white noise of desired frequency of operation. The falloff the spectrum begins at 0.05 times the clock frequency.

4) Interfacing Microcontroller:-

Here we interface the microcontroller for amplitude and spectrum control.

(a) The amplitude is varied in steps of 5v p-p, 4v p-p, 3V p-p, 2V p-and 1Vp-p.
(b) The spectrum can either be white noise or pink noise.

(c) The clock provided by the microcontroller has nine discrete frequency values i.e. 100 Hz , 1kHz , 5 kHz , 10 kHz , 50 kHz , 100 kHz , 200 kHz, 500 kHZ and 1 MHz .

5) Audio amplifier:-

The amplifier circuit used is designed using LM324. The op amp is in the buffer configuration feeding the push-pull configuration. The diode D1 helps to reduce the cross over distortion which would be present otherwise. The output of circuit is fed back to LM324.

6) Power supply:-

The power supply is provided with +/-18 V transformer and regulators are used to bring the voltage down to the requisite levels.

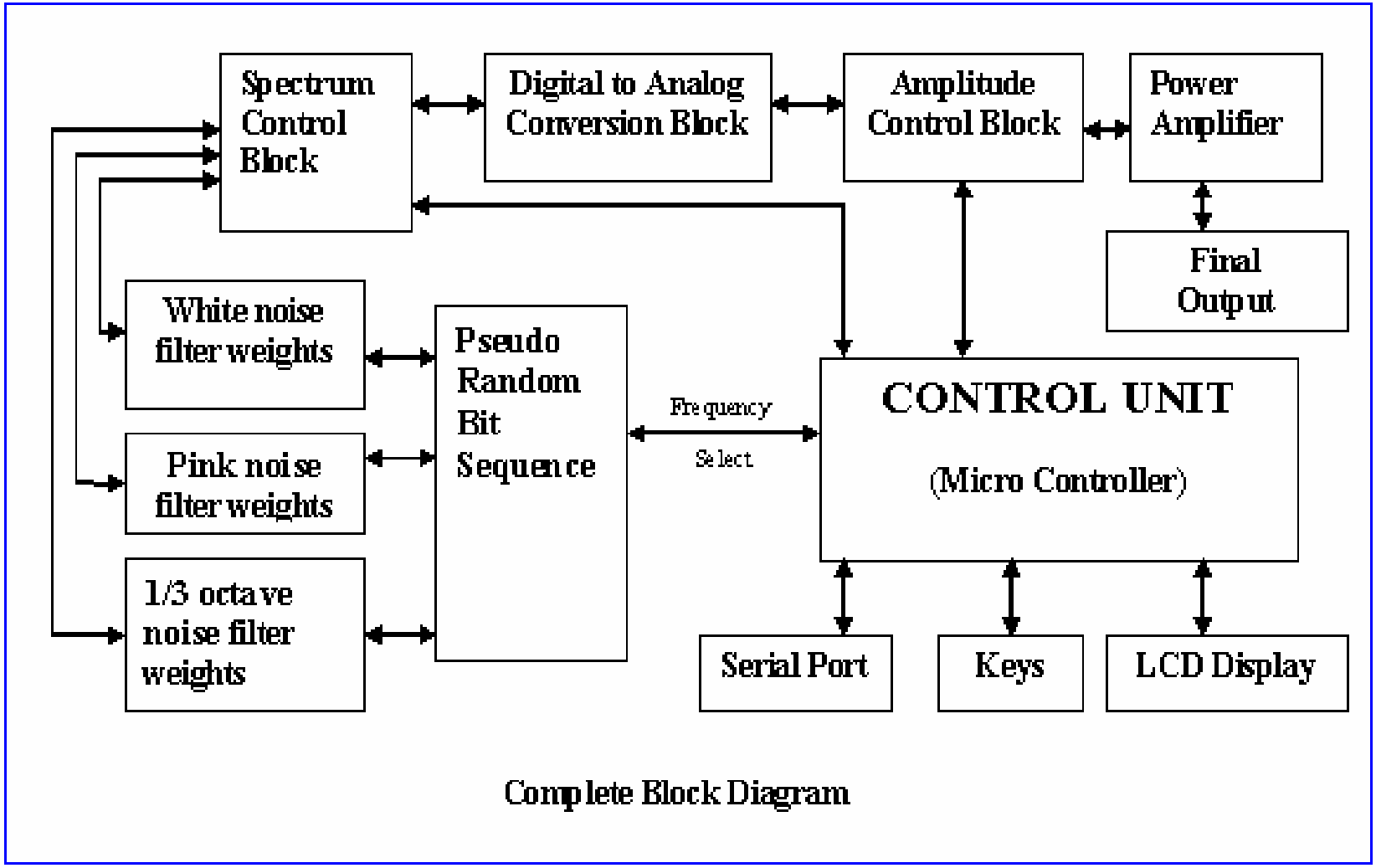
7) The design also has four keys; one for bandwidth adjustment; two for amplitude control and one for spectrum control. They are interfaced with the microcontroller for the proper use of the microcontroller.

8) There is a multiplexer used for amplitude control. It is followed by voltage dividers and the microcontroller provides the select lines for the same

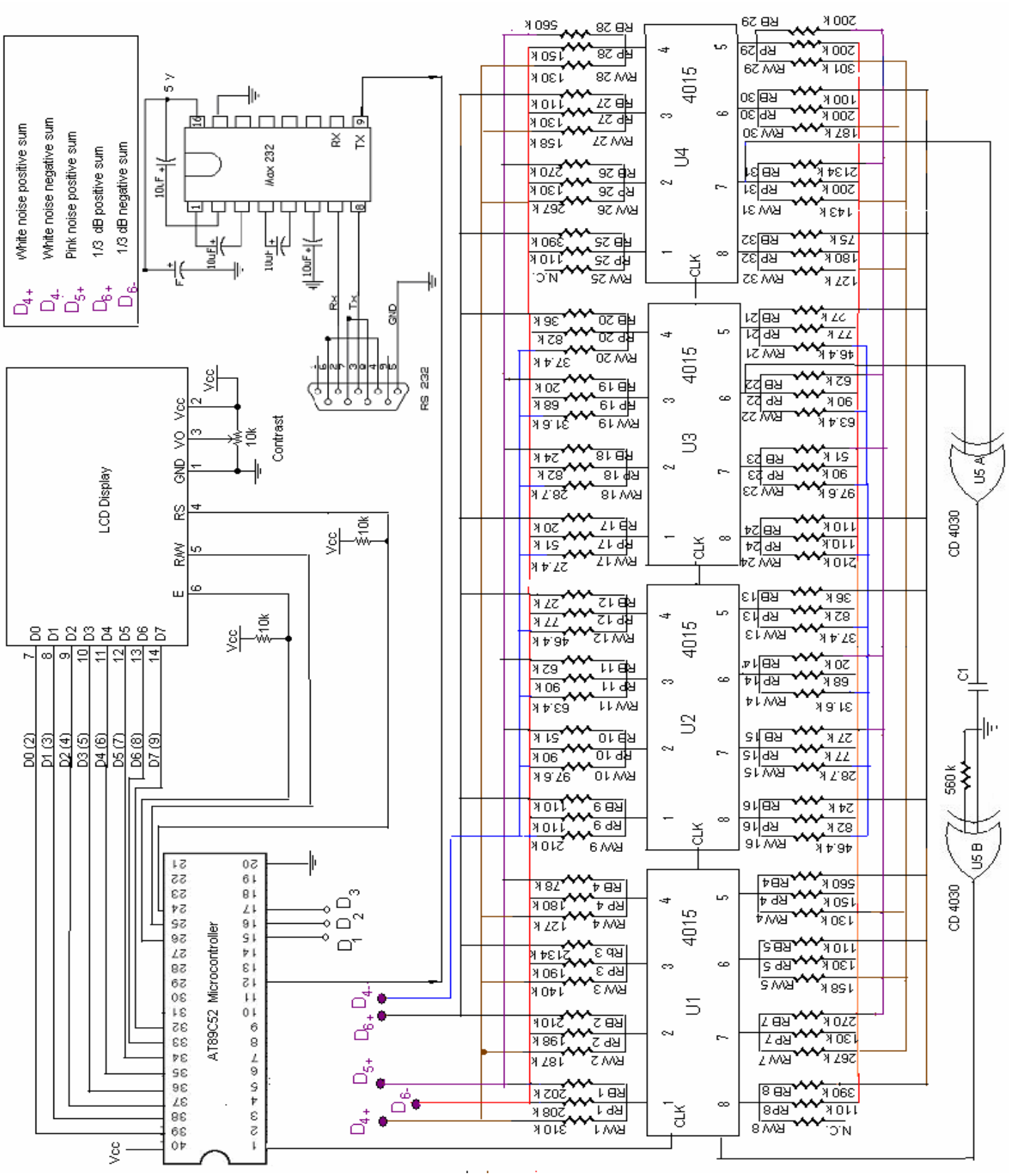
8) The LCD display has been interfaced with the microcontroller.

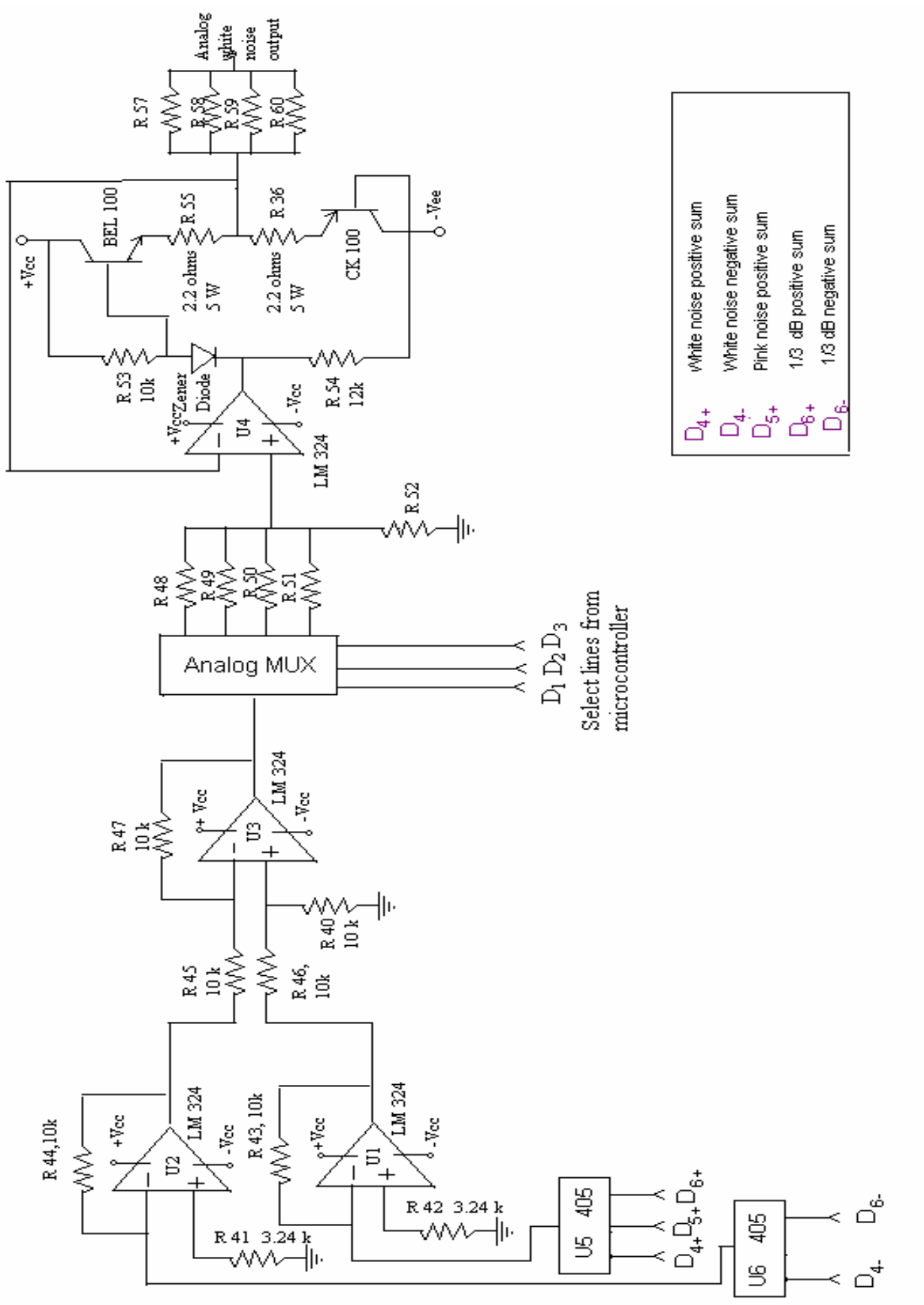
10) A MAX 232 helps interface the random noise generator to a serial port thus enabling serial port communication

11) The 'Pink noise' and '⅓ db noise' filter weights have been implemented. The noise generator is now capable of generating three spectrums of noise i.e White noise , Pink noise and ⅓ db per octave noise.



Complete Block Diagram


White noise positive sum
White noise negative sum
Pink noise positive sum
1/3 dB positive sum
1/3 dB negative sum
D4+
D4-
D5+
D6+
D6-
5 V
Max 232
RS 232
LCD Display
Contrast
AT89C52 Microcontroller
U1
U2
U3
U4
4015
CD 4030
U5 A
U5 B
C1
560 k


Analog white noise output
R 57
R 58
R 59
R 60
BEL 100
R 55
R 36
CK 100
+Vcc
-Vee
2.2 ohms 5 W
2.2 ohms 5 W
R 53 10k
R 54 12k
+Vcc
Zener Diode
-Vcc
U4
LM 324
R 52
R 48
R 49
R 50
R 51
Analog MUX
D1 D2 D3
Select lines from microcontroller
R 47 10 k
+Vcc
-Vcc
U3
LM 324
R 40 10 k
R 45 10 k
R 46, 10k
R 44,10k
R 43, 10k
R 41 3.24 k
R 42 3.24 k
U2
U1
LM 324
U5 405
U6 405
D4+ D5+ D6+
D4- D6-
D4+ White noise positive sum
D4- White noise negative sum
D5+ Pink noise positive sum
D6+ 1/3 dB positive sum
D6- 1/3 dB negative sum

2.4.2 LIST OF COMPONENTS USED

1) IC's :- The IC's used in our project are following :-
    - CD 4015B
    - CD 4030
    - CD 4051
    - CD 4052
    - CD 4081
    - LM324
    - LM 741
    - MAX 232
    - LCD Display Module

2) Resistors
3) Capacitors
4) Transistors
5) Serial port connector and serial port cable.
6) Micro controller :- The micro controller used is AT89C52
7) Push button keys: - four keys are used in our circuit. Following are the functions of the keys:-
    - To select the frequency level
    - To increase the amplitude level
    - To decrease the amplitude level
    - To select the noise filter

8) 11.0592 MHz oscillator
9) Voltage regulators: - We have used three voltage regulators in the circuit. They are 7805, 7809, 7909.
10) Transformer: - A 230/12 volt step down transformer is used to drive the circuit through AC supply.
11) Rectifier: The rectifiers used are IN 4004. They are used in bridged rectifier arrangement to convert +/- 12 volt AC supply to DC supply.
12) BNC connector: Finally the output of our circuit can be used for any applications through the BNC connector attached to the circuit.

A 4.7 *nf* decoupling capacitor is provided between the Vcc and ground of every chip in the circuit.

3. TEST RESULTS

1) Pseudo random bit sequence :-

Before assembling this block each of the shift registers were tested separately. If individual 4 bit dual registers were working, the whole block was compiled and tested using an external clock from the signal generator. The Microcontroller block came much later and was interfaced with the random bit sequence once it was ready.

One of the problems arose when we initially had decided to use CD4021 which is also an 8 bit shift register. However it cannot be used for this problem because it is serial in and serial out. The requirements for our design are entirely different from that (We want parallel out).

The feedback loop was also tested and was found to be working reliably.
The pseudo random bit sequence gives a digital noise output which if as shown in *fig* 3.1 (a). The power spectrum of the unfiltered digital shift register output signal is as shown in *fig* 3.2(a).
(The envelope is proportional to (sin(x)/x). The half power is obtained at 44% f (clock) and the first zeros at f (clock).)

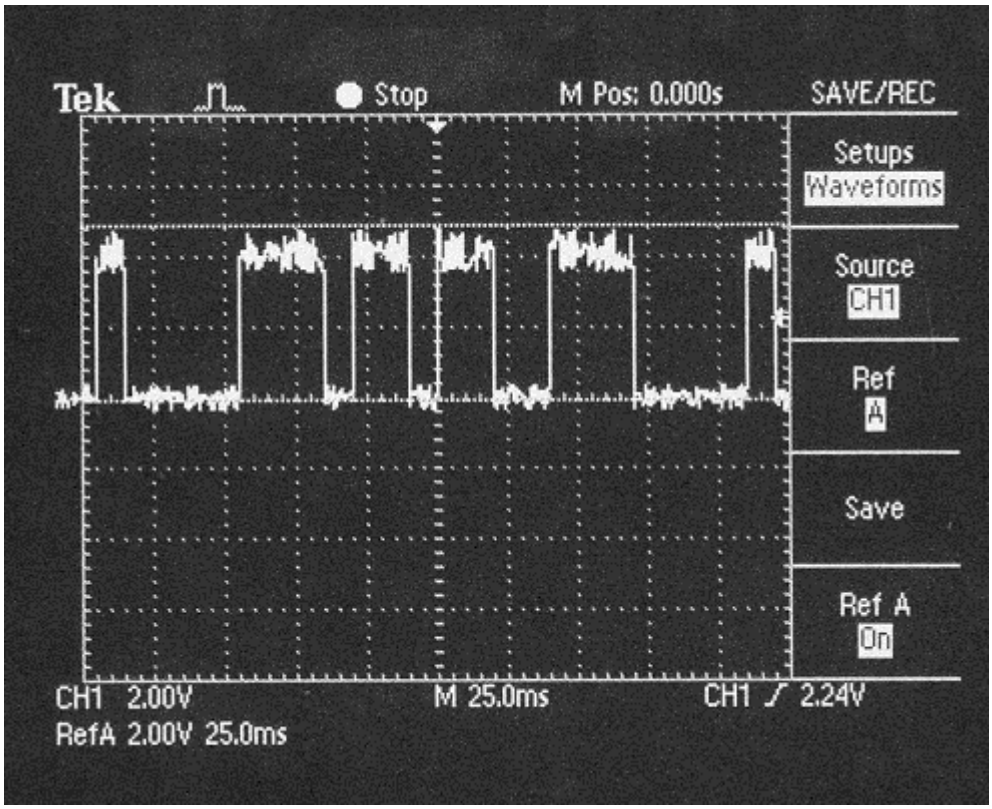


Fig 3.1(a) - A digital PRBS

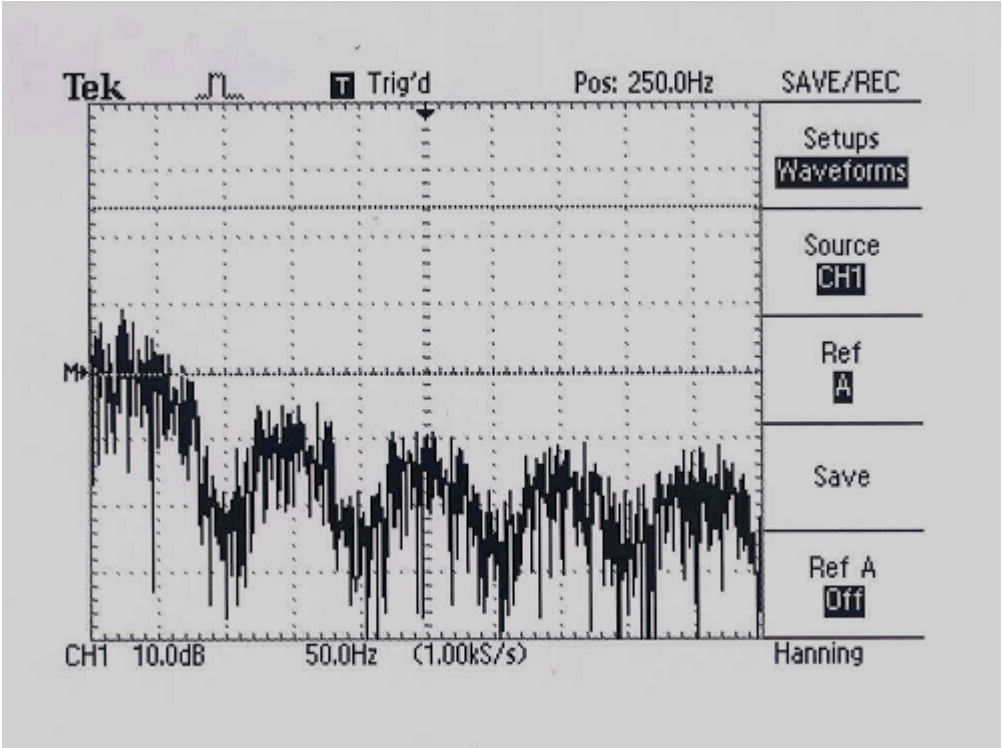


Fig 3.2(b) - Spectrum of digital noise

2) Digital filtering :-

The digital filtering is done using a weighted resistor bank. At the output of the positive and the negative resistor banks, the spectrum was observed.

3) Noise filter :-

The PRBS sequence is interfaced to the noise filter. The difference amplifier interfaced after the opamps summing function had taken place for the positive and negative values separately. The white noise was observed at the output of this whole block and its power spectrum observed. The power spectrum for different frequencies of operation is as observed in *fig* 3.3(a), 3.3(b), 3.3(c) and 3.3(d). The 3db points of these filters were observed.

(The power spectrum remains constant for some while and then begins to fall at 0.05 times the clock frequency)

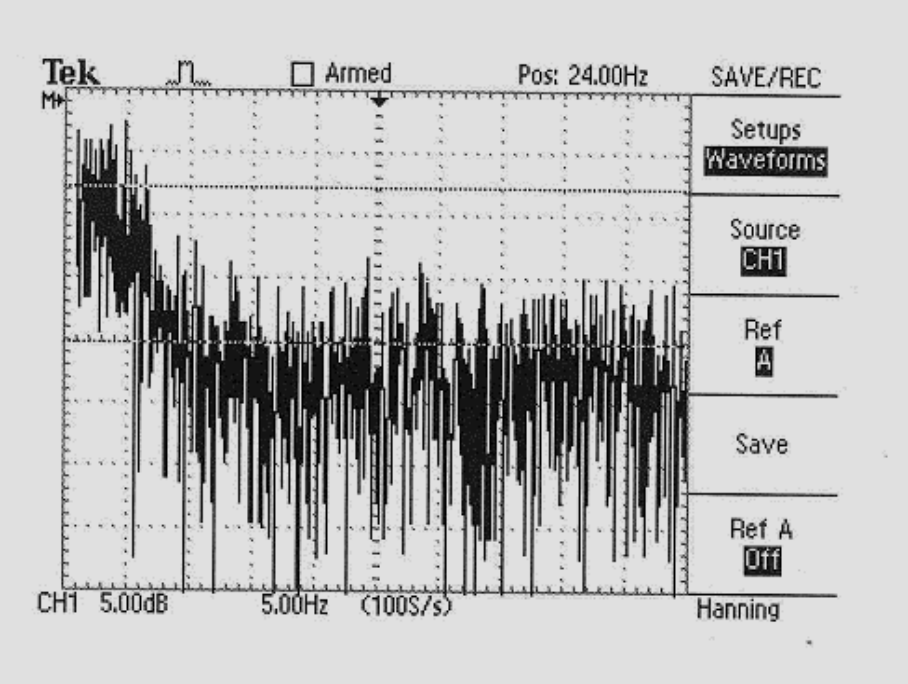


fig 3.3(a) - White noise FFT (frequency of clock = 100Hz)

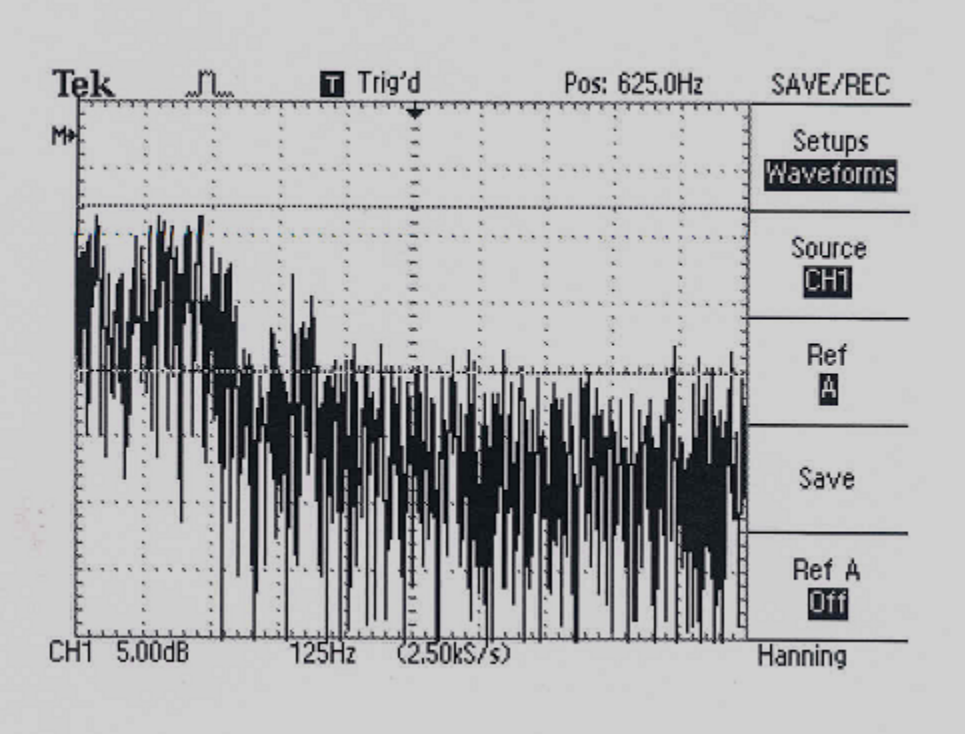


fig 3.3(b) - White noise at frequency of clock= 5 KHz

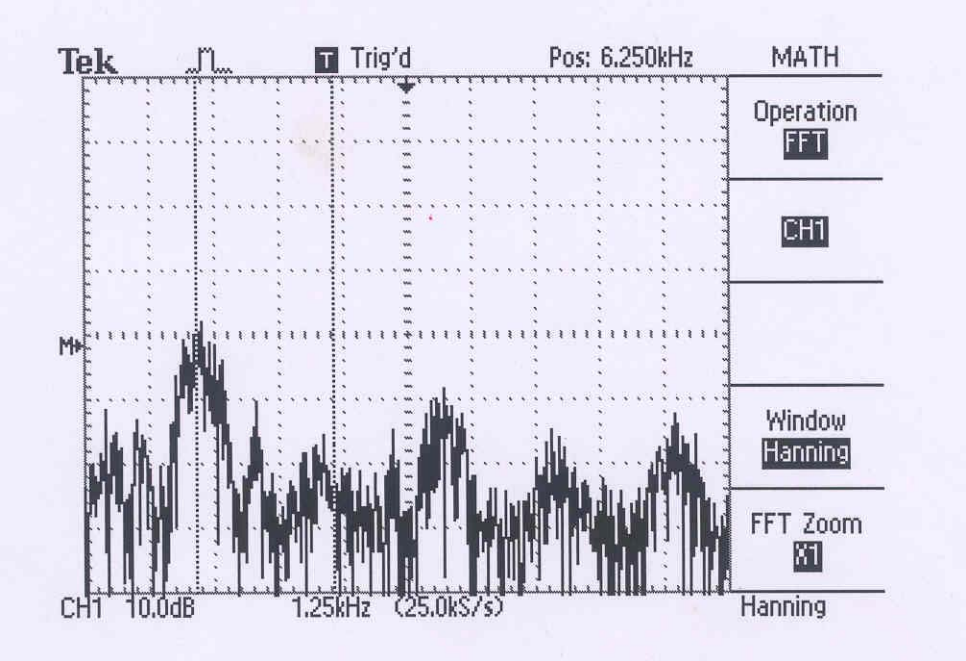


fig 3.3(c) - ⅓ Octave noise , frequency of clock= 5 KHz.

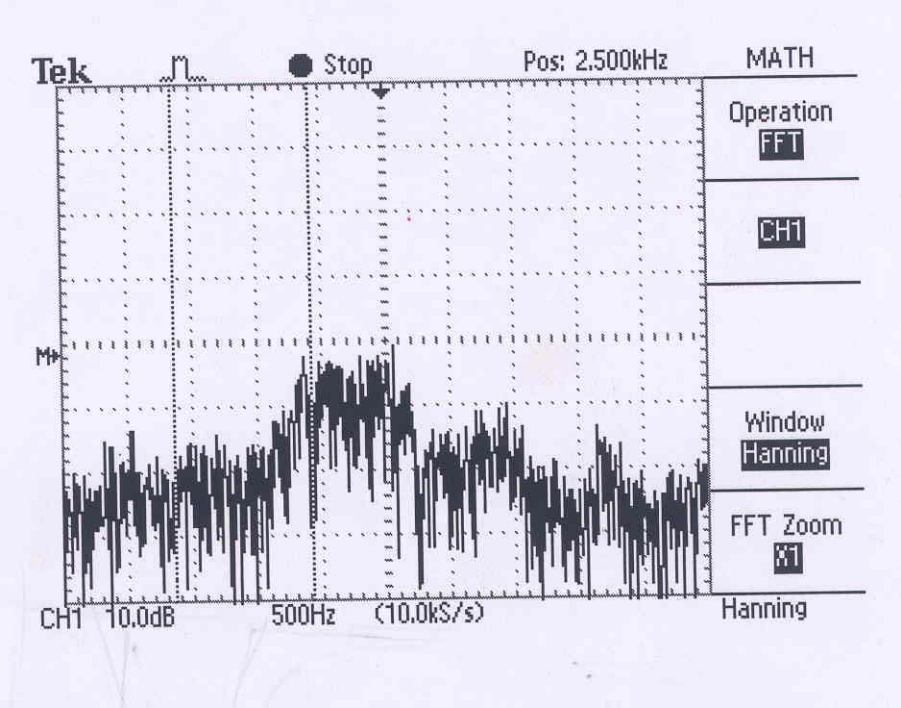


fig 3.3(d) – ⅓ Octave noise at operating at clock frequency of 5 KHz

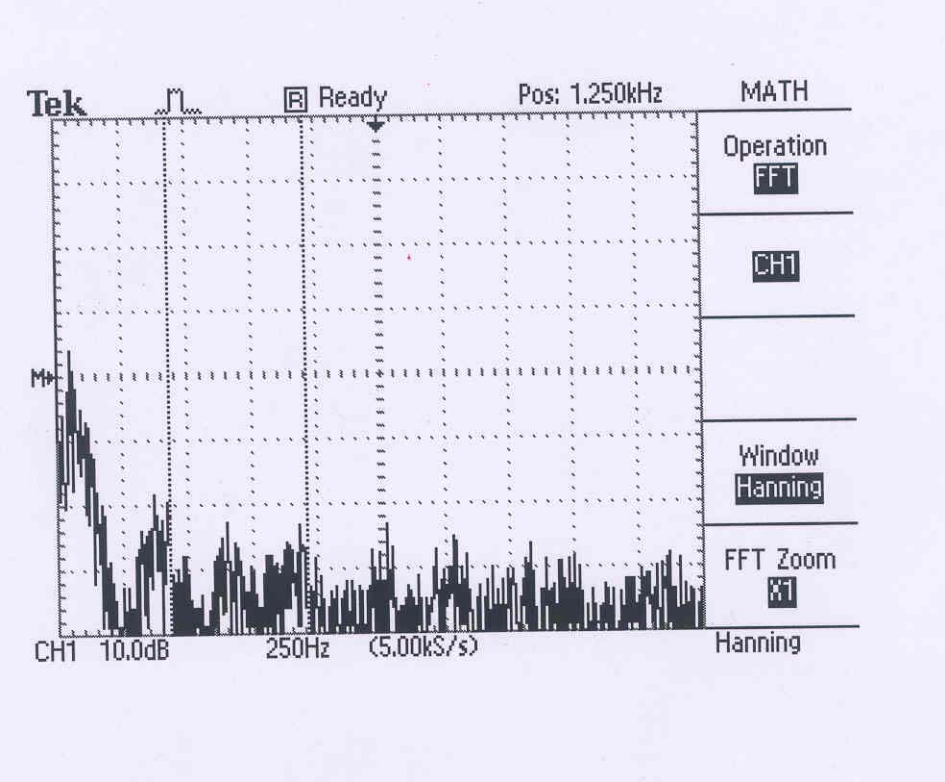


fig 3.3(e) - Pink noise at operating at clock frequency of 5 KHz

4) Interfacing the microcontroller :-

The microcontroller was interfaced with the PRBS sequence once it was ready. It generates clocks of different frequencies ranging from 50 Hz to 1 MHz. The microcontroller (89C52) is used in the programmable clock out mode. The RCAP2H and the RCAP2L registers in the 8952 are provided with different values to generate clock with frequencies which are multiples of the clock frequency. The spectrum for the same is observed in *fig* 4.

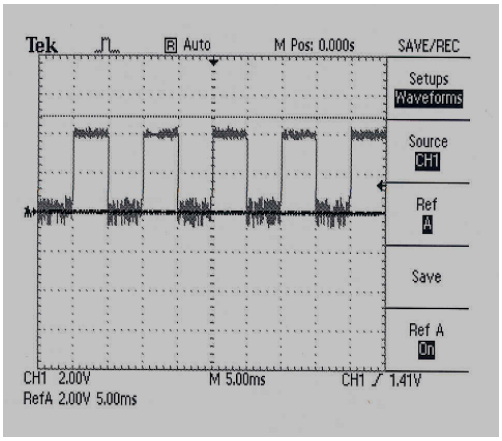


Fig 3.4 - Clock generated by the microcontroller

5) Multiplexer for controls

Maximum amplitude of the output was observed to be 5V p-p. This was then distributed into different levels with the help of voltage dividers and the multiplexer powered by the microcontroller. The multiplexer depending upon the input from the microcontroller selects one of the voltage dividers so that the amplitude is varied accordingly. The spectrums were observed.

6) Switches and keys

There were four keys used i.e. for spectrum control; bandwidth control and amplitude up/down.
The keys that were being used for the user interface were not working properly. A time delay of 1 second was provided once a key is pressed. For this time, the same key pressed again would not show any response.

7) Audio amplifier :-

The amplifier circuit used is designed using LM324. This stage is used to give a unity voltage gain to the output of the Amplifier and boost the current output through the push-pull transistor arrangement as shown in figure. The op amp is in the buffer configuration feeding the push-pull configuration.

The diode D1 helps to reduce the cross over distortion which would be present otherwise if we had used only resistors to bias the transistors.
The diode compensates the 0.7 V drop of the push-pull transistor and hence no cross over distortion occurs.

8) Power supply :-

The power supply was connected to the other part of the circuit and tested for the load regulation and its line regulation.

4. CONCLUSION

In our EDP we have been able to implement random noise generation. We have different amplitude levels for which we have generated white noise. The bandwidth is also digitally selectable. The advantage that we have obtained through digital filtering is the availability of operating at large number of frequencies in our design. White noise which is used for testing RF circuits is available for low frequency testing (~ <10 Hz) as well as for very high frequencies (~ 50 KHz).

The filter weights have been varied so as to generate pink noise as well. For audio testing, pink noise is an invaluable tool. It is essentially a flat frequency response noise source, and will quickly show any anomalies in speaker systems, room acoustics and crossover networks. White noise (the sound you hear when a TV is tuned to a non-existent station) has a frequency characteristic which raises the power level by 3dB with each increasing octave, and is not suitable for response testing. By combining a 3dB / octave filter and a white noise source, we can get a very good approximation to "perfect" pink noise, where the power in the octave (for example) 40 to 80Hz is exactly the same as in the octave 10kHz to 20kHz.

Besides the above two, ⅓ db noise has also been generated using digital filtering.

The device has a very good user interface and stable characteristics. A microcontroller provides the clock and also does the amplitude and spectrum control. The LCD provides the visual display for the output. The device is also interfaced with the serial port.

5. FUTURE SCOPE

Many more spectrums of noise can be added to the project. Different kinds of noise find a lot of different application. All that has to be changed in the design is the weight of filters in the digital filtering.

The values of amplitude and frequency are discrete right now and that can be improved by making these continuous. We can also interface the current model for testing devices with some minor modifications in our design. One of the significant features of the design is the use of the design for low frequency RF testing.

Our overall circuit can also be implemented on a single dedicated DSP chip which can do the entire analog and digital operations of the circuit since DSPs are optimized for operation like convolution and discrete fourier transform this product will be much more cost effective and efficient if implemented on a DSP.

Appendix

The visual basic code for control toolbox is given in this section

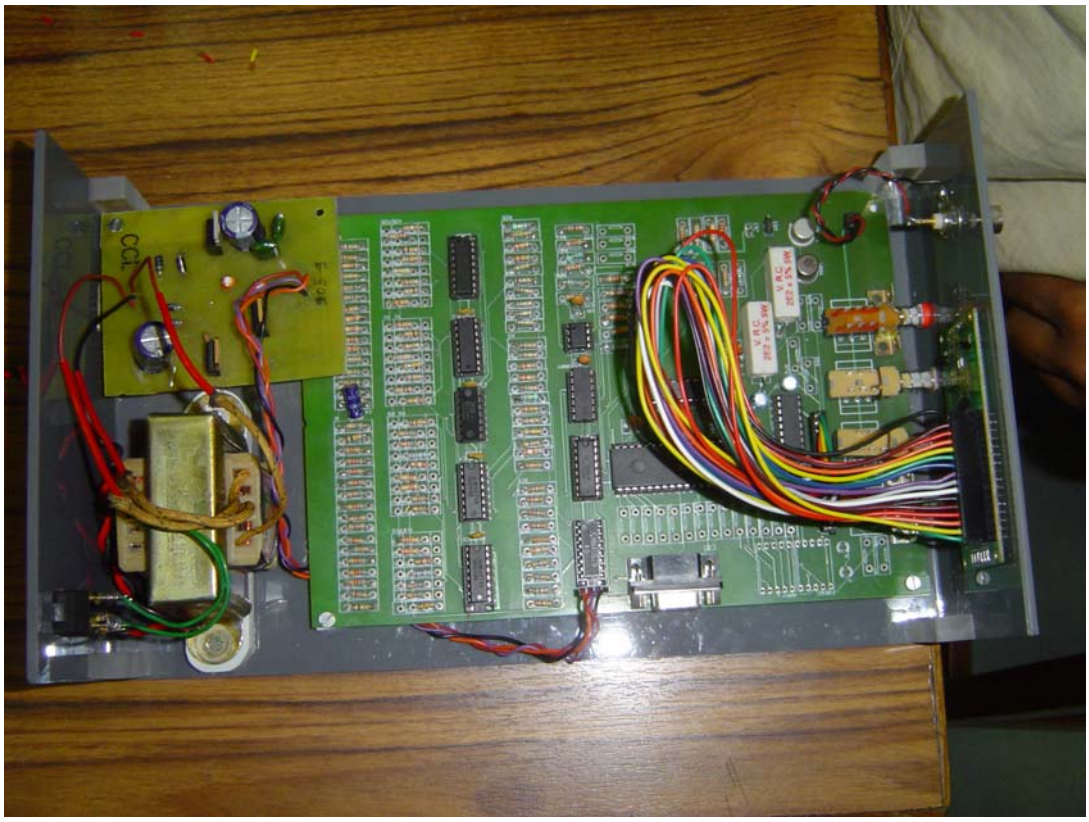
```
VERSION 5.00
Object = "{648A5603-2C6E-101B-82B6-000000000014}#1.1#0"; "MSCOMM32.OCX"
Begin VB.Form Form1
   Caption        =   "Control Toolbox"
   ClientHeight   =   3090
   ClientLeft     =   60
   ClientTop      =   450
   ClientWidth    =   4680
   LinkTopic      =   "Form1"
   ScaleHeight    =   3090
   ScaleWidth     =   4680
   StartUpPosition =   3  'Windows Default
   Begin VB.CheckBox Check2
      Caption        =   " 2 "
      Height         =   255
      Left           =   3720
      TabIndex       =   16
      Top            =   1320
      Width          =   495
   End
   Begin VB.CheckBox Check1
      Caption        =   " 1"
      Height         =   255
      Left           =   3720
      TabIndex       =   15
      Top            =   960
      Width          =   615
   End
   Begin MSCommLib.MSComm MSComm1
      Left           =   4080
      Top            =   2520
      _ExtentX       =   1005
      _ExtentY       =   1005
      _Version       =   393216
      DTREnable      =   -1  'True
   End
   Begin VB.OptionButton Option2
      Caption        =   "Option2"
      Height         =   255
      Left           =   1080
      TabIndex       =   10
      Top            =   2280
      Width          =   255
   End
   Begin VB.CommandButton Command2
      Caption        =   "OK"
      Height         =   375
      Left           =   3120
      TabIndex       =   5
      Top            =   2040
      Width          =   1215
   End
   Begin VB.OptionButton Option3
      Caption        =   "Option3"
      Height         =   255
      Left           =   1080
      TabIndex       =   4
      Top            =   2640
      Width          =   255
   End
   Begin VB.OptionButton Option1
      Caption        =   "Option1"
      Height         =   255
```

```
  Left          =   1080
  TabIndex      =   3
  Top           =   1920
  Width         =   255
End
Begin VB.ComboBox Combo2
  Height        =   315
  Left          =   1080
  TabIndex      =   2
  Text          =   "   Select "
  Top           =   1440
  Width         =   1215
End
Begin VB.ComboBox Combo1
  Height        =   315
  Left          =   1080
  TabIndex      =   1
  Text          =   "   Select "
  Top           =   960
  Width         =   1215
End
Begin VB.Label Port
  Caption       =   "Com Port : "
  Height        =   255
  Left          =   2760
V1  TabIndex    =   14
  Top           =   1080
  Width         =   855
End
Begin VB.Label Control
  Caption       =   "Control Toolbox"
  Height        =   255
  Left          =   1560
  TabIndex      =   13
  Top           =   600
  Width         =   1335
End
Begin VB.Label Octave
  Caption       =   "1/3 Db/octave "
  Height        =   255
  Left          =   1560
  TabIndex      =   12
  Top           =   2640
  Width         =   1215
End
Begin VB.Label Pink
  Caption       =   "Pink Noise"
  Height        =   255
  Left          =   1560
  TabIndex      =   11
  Top           =   2280
  Width         =   975
End
Begin VB.Label White
  Caption       =   "White Noise"
  Height        =   255
  Left          =   1560
  TabIndex      =   9
  Top           =   1920
  Width         =   1215
End
Begin VB.Label Spectrum
  Caption       =   "Spectrum"
  Height        =   255
  Left          =   120
  TabIndex      =   8
  Top           =   2160
  Width         =   855
End
```

```
  Begin VB.Label Amplitude
    Caption        =   "Amplitude"
    Height         =   255
    Left           =   120
    TabIndex       =   7
    Top            =   1440
    Width          =   855
  End
  Begin VB.Label Frequency
    Caption        =   "Frequency : "
    Height         =   255
    Left           =   120
    TabIndex       =   6
    Top            =   960
    Width          =   855
  End
  Begin VB.Label Main
    Caption        =   " Random Noise Generator"
    BeginProperty Font
      Name           =   "Times New Roman"
      Size           =   20.25
      Charset        =   0
      Weight         =   700
      Underline      =   0   'False
      Italic         =   0   'False
      Strikethrough  =   0   'False
    EndProperty
    ForeColor      =   &H80000015&
    Height         =   765
    Left           =   0
    TabIndex       =   0
    Top            =   120
    Width          =   8520
  End
End
Attribute VB_Name = "Form1"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False

Dim noise As Integer
Dim outbuf As String
Dim send As Integer


Private Sub Option1_Click()
noise = 1
End Sub

Private Sub Option2_Click()
noise = 2
End Sub

Private Sub Option3_Click()
noise = 3
End Sub

Private Sub Command2_Click()
send1 = 16 * (Combo1.ListIndex + 1) + 4 * Combo2.ListIndex + noise
MSComm1.Settings = "9600,n,8,1"
MSComm1.RTSEnable = False
MSComm1.DTREnable = True
MSComm1.PortOpen = True
outbuf = Trim$(Chr$(send))
MsgBox (outbuf)
MSComm1.Output = outbuf
MSComm1.PortOpen = False
End Sub
```

```
Private Sub Form_Load()

Combo1.AddItem "  .005 KHz"
Combo1.AddItem "  0.05 KHz"
Combo1.AddItem "  0.25 KHz"
Combo1.AddItem "  0.50 KHz"
Combo1.AddItem "  2.50 KHz"
Combo1.AddItem "  5.00 KHz"
Combo1.AddItem "  10.0 KHz"
Combo1.AddItem "  25.0 KHz"
Combo1.AddItem "  50.0 KHz"

Combo2.AddItem "  5V p-p"
Combo2.AddItem "  -1 dB"
Combo2.AddItem "  -2 dB"
Combo2.AddItem "  -3 dB"

Option1 = True

End Sub
```

Front view of random noise generator



Inner circuit view of random noise generator

REFERENCES

[1] K.J.Ayala, "*8051 Microcontroller: Architecture, Programming and Applications*", New York: McGraw-Hill, 1964, pp. 15–64.

[2] P. Horowitz and W. Hill, "*The Art of Electronics*", Cambridge: Cambridge University Press, 1980

[3] A.V.Oppenhium, A.S.Willsky, and I.T.Young, "*Signals and Systems*", New Jersey: Prentice-Hall, 1989

[4] E.C. Ifeachor,B.W. Jervis, "*Digital Signal Processing"* , New Jersey: Pearson Education Ltd., 2002

[5] J.G. Proakis,D.G. Manolakis, "*Digital Signal Processing",* New Jersey: Prentice-Hall, 1996

[6] R.Jonathan, "*White and Pink noises*", 1998; http://www.home.howstuffworks.com/question47.htm

[7] R.Whittle, *"DSP Generation of Pink Noise"*, 2001; http://www.firstpr.com.au/dsp/pink-noise/

[8] N.J.Elliot, "*White and Pink Noise Generation Circuits*", 2000; http://www.discover circuits.com/N /noise gen.htm

[9] A.Judson, "*A P-spice Simulation of Pink Noise Filter*", 2001;http://www.users.big pond.com /vk3jaj /pinkfilt/

[10] C.Peacock, "*Interfacing the Serial / RS232 Port",* 2001; http://www.beyondlogic.org /serial/serial.htm