Effect of Finite Word Length on FIR Filter Implemented on 8052 Microcontroller

A Thesis

submitted in the partial fulfillment of requirements for the award of the degree of

Master of Engineering

in

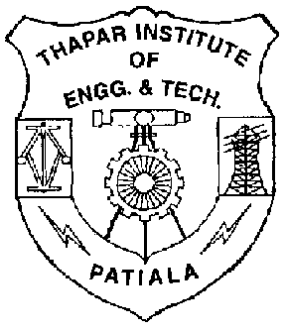
Electronics Instrumentation and Control Engineering

to

THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY

(DEEMED UNIVERSITY)

PATIALA (PUNJAB)-147004



Submitted by
Aman
ME 8044202

Under the Guidance of
Mr. Sunil Kumar Singla
Mr. Nirbhow Jap Singh

Department of Electrical and Instrumentation Engineering

THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY

(DEEMED UNIVERSITY)

PATIALA (PUNJAB) – 147004

JUNE 2006

Acknowledgement

Words are often less to revels one's deep regards. An understanding of work like this is never the outcome of the efforts of a single person. I take this opportunity to express my profound sense of gratitude and respect to all those who helped me through the duration of this thesis.

First off all I would like to thank the Supreme Power, one who has always guided me to work on the right path of the life. Without his grace this would never come to be today's realty.

I am highly indebted to **Madam Manbir Kaur**, Head of Department for providing me requisite environment and being content source of inspiration.

This work would not have been possible without the encouragement and able guidance of my supervisor **Mr. Sunil Kumar Singla**, his enthusiasm and optimism made this experience both rewarding and enjoyable. Most of the novel ideas and solutions found in this thesis are the result of our numerous stimulating discussions. His feedback and editorial comments were also invaluable for the writing of this thesis.

I express my sincerest regards and gratitude to my co-supervisor **Mr. Nirbhow Jap Singh** for his able guidance, valuable advice and helpful suggestions at times of difficulties which I faced pursuing this project. He has guided me and given me full time to understand the minute details and all the basic concepts necessary for the successful completion of the project.

My humble gratitude is also reserved for **Mr. Manmohan Singh** for his judicious help and sustained encouragement, which were the constant source of inspiration to carry out the thesis.

No words of thanks for my dear parents whose support, whose care makes me stay on earth. Thanks to be with me.

(Aman)

ABSTRACT

Digital filters have found important applications in an increasing number of fields in science and engineering, and design techniques have been developed to achieve desired filter characteristics. A digital filter operates on an input-sampled signal to produce an output-sampled signal by means of a computational algorithm. It can be simulated on a general purpose computer or can be constructed with special-purpose digital hardware. Despite the many advantages offered by digital filters, there is an inherent limitation on the accuracy of these filters due to the fact that all digital networks operate with only a finite number of bits. The three common sources of error due to finite word length are the quantization of the input signal into a set of discrete levels, the representation of the filter coefficients a_k and b_k by a finite number of bits, the accumulation of round-off errors committed at arithmetic operations.

This thesis work, Effect of Finite Word Length on FIR Filter implemented on 8052 Microcontroller, includes the design and development of the hardware as well as software of the low pass Finite Impulse Response (FIR) digital filter. To find out the error introduced by finite word length on FIR filter, a low pass FIR filter is designed with the help of Matrix Laboratory (MATLAB) to compare with the FIR filter implemented on 8052 microcontroller. Blackman window, Hanning window and Rectangular window are used. 8051 is an 8-bit microcontroller so the word length of registers is limited to 8-bits. The filter coefficients and the input sequence are either truncated or rounded-off to 8-bit word length. The filter coefficients are convolved with the input sequence by using microcontroller. Results of the convolution are compared with results of convolution done with MATLAB.

TABLE OF CONTENTS

CONTENTS **Page No.**

LIST OF FIGURES

Figure No. Title Page No

LIST OF TABLES

CHAPTER -1

INTRODUCTION

A digital filter is a discrete-time, discrete-amplitude convolver. Basic Fourier transform theory states that the linear convolution of two sequences in the time domain is the same as multiplication of two corresponding spectral sequences in the frequency domain. Filtering is in essence the multiplication of the signal spectrum by the frequency domain impulse response of the filter (shown in Figure 1.0). For an ideal low pass filter the pass band part of the signal spectrum is multiplied by one and the stop band part of the signal by zero [1].

It is important to note that distortion and noise can be introduced into digital filters simply by the conversion of analog signals into digital data, also by the digital filtering process itself and lastly by conversion of processed data back into analog. When fixed-point processing is used, additional noise and distortion may be added during the filtering process because the filter consists of large numbers of multiplications and additions, which produce errors, creating truncation noise. Increasing the bit resolution beyond 16-bits will reduce this filter noise. For most applications, as long as the analog to digital (ADC) and digital to analog (DAC) converters have high enough bit resolution, distortions introduced by the conversions are less of a problem [2].

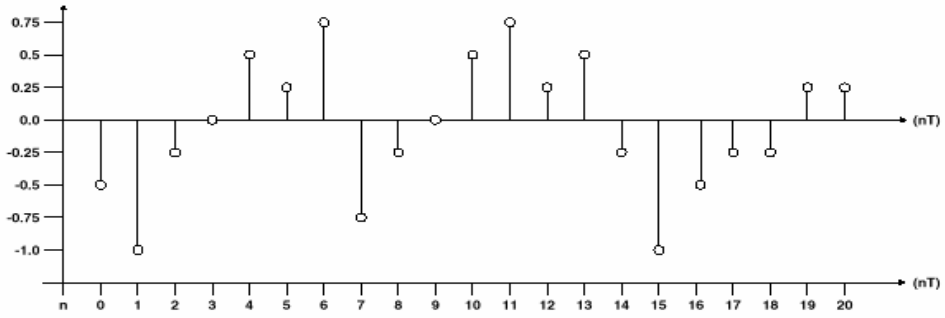


Figure 1.0(a) Signal Samples

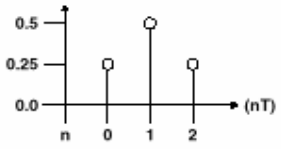


Figure 1.0(b) Filter Impulse Response Sequence

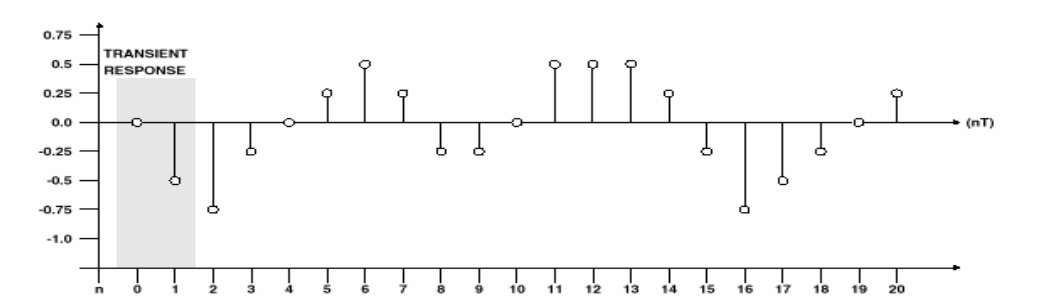


Figure 1.0(c) Output Samples

Figure 1.0 The Discrete Time and Discrete Amplitude Convolution of Signal Samples with FIR sequence

When digital systems are implemented either in hardware or in software, the filter coefficients are stored in binary registers. These registers can accommodate only a finite number of bits and hence, the filter coefficients have to be truncated or rounded off in order to fit into these registers. Truncation and rounding-off the data results in degradation of system performance. Also in digital processing systems, a continuous time input signal is sampled and quantized in order to get the digital systems [2]. There are four ways in which finite word length effects the performance of FIR digital filter:

1) Analog-to-digital (ADC) quantization noise results when the filter input is derived from the analog signals. ADC noise limits the signal-to-noise ratio obtainable.

2) Coefficient quantization errors result from representing filter coefficients with limited number of bits. This has the adverse effect of modifying the desired frequency response. In the stop band of a filter it limits the maximum attenuation possible, thus allowing additional signal transmission.

3) Rounding-off errors from quantization results of arithmetic operations can occur by discarding the lower-order bits before storing the results of the multiplication this is normally forced by the word length of the processor used. This error reduces the signal-to-noise (SNR).Extend of the errors introduced depends on the type of arithmetic used and the filter structure

4) Arithmetic overflow occurs when partial sums or filter outputs exceeds the permissible word length of the system. Essentially, when an overflow occurs, the output sample will be wrong [3].

Problem Definition

The present work has been, undertaken as design and development of digital FIR filter. The assignment is divided in two parts: (i) development of hardware (ii) development software which is further subdivided into:

1) Development of the software for Low pass FIR Digital Filter using windowing technique in MATLAB.

2) Design and development of the hardware and the software for Low pass FIR Digital Filter using Windowing technique in 8-bit Microcontroller.

3) Comparison of both the responses to find out the quantization.

CHAPTER-2

LITERATURE REVIEW

This chapter describes about the digital filters, types of a digital filters then selection between the Finite Impulse Response (FIR) filter and Infinite Impulse Response (IIR) filter, concepts of windowing technique and convolution. In the later section of the chapter 8052 microcontroller, introduction to Matrix Laboratory (MATLAB) and Universal Microprocessor Program Simulator (UMPS) is discussed.

2.1 Digital Filter

A digital filter is a mathematical algorithm implemented in hardware and/or software that operates on a digital input signal to improve output signal for the purpose of achieving a filter objective. The term digital filter refers to the specific hardware and software routine that performs the filtering algorithm. Digital filter often operate on digitized analog signals or just numbers, representing some variable, stored in a computer memory. A simplified block diagram of a real-time digital filter, with analog input and output signals, is given in Figure 2.1. The band limited analog signals is sampled periodically and converted into series of digital samples x(n),n = 0,1,…… the digital processor implements the filtering operation, mapping the input sequence x(n) into the output sequence y(n) in accordance with a computational algorithm for the filter. The DAC converts the digitally filtered output into analog values [3].

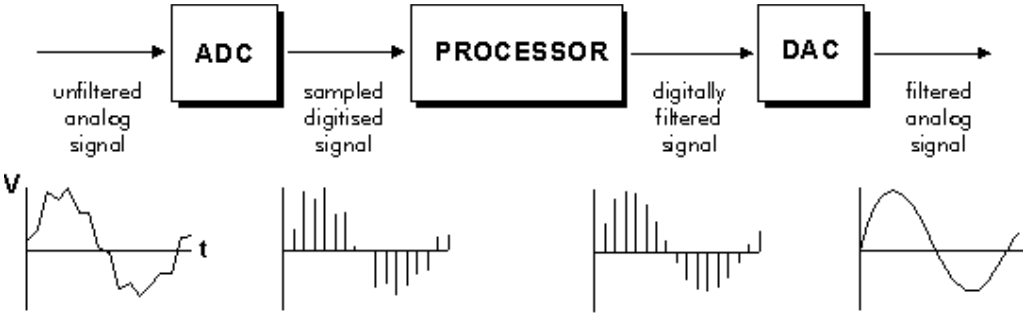


Figure 2.1 A Real-Time Digital Filter

2.2 Types of Digital Filters

Many digital systems use signal filtering to remove unwanted noise, to provide

spectral shaping, or to perform signal detection or analysis. Two types of filters provide these functions are finite impulse response (FIR) filters and infinite impulse response (IIR) filters. Typical filter applications include signal preconditioning, band selection, and low pass filtering.

2.2.1 Finite Impulse Response (FIR) Filter

A Finite Impulse Response (FIR) digital filter is one whose impulse response is of finite duration. The impulse response is "finite" because there is no feedback in the filter if we put in an impulse (that is, a single "1" sample followed by many "0" samples), zeroes will eventually come out after the "1" sample has made its way in the delay line past all the coefficients.

The structure of these algorithms uses a repetitive delay-and-add format that can be represented as "DIRECT FORM-I STRUCTURE".

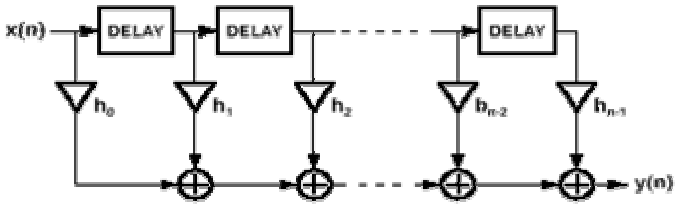


Figure 2.2 Direct Form-I Structure

FIR (Finite Impulse Response) filters are implemented using a finite number "n" delay taps on a delay line and "n" computation coefficients to compute the algorithm (filter) function. The above structure is non-recursive, a repetitive delay-and-add format, and is most often used to produce FIR filters. This structure depends upon each sample of new and present value data. The number of taps (delays) and values of the computation coefficients (h_0,h_1,…..h_n) are selected to "weight" the data being shifted down the delay line to create the desired amplitude response of the filter. In this configuration there are no feedback paths to cause instability. The calculation of coefficients is not constrained to particular values and can be used to implement filter functions that do not have a linear system equivalent. More taps increase the steepness of the filter roll-off while increasing calculation time (delay) and for high order filters, limiting bandwidth.

This can be stated mathematically as:

$$y(n) = \sum_{0}^{N-1} h(k)x(n-k) \tag{2.1}$$

y (n) = Response of Linear Time Invariant (LTI) system

x(k) = Input signal

h(k) = Unit sample response

N = No. of signal samples

FIR filters are simple to design and they are guaranteed to be bounded input-bounded output (BIBO) stable. By designing the filter taps to be symmetrical about the center tap position, a FIR filter can be guaranteed to have linear phase response. This is a desirable property for many applications such as music and video processing. They are simple to implement in hardware. They have desirable numeric properties. In practice, all Digital Signal Processing (DSP) filters must be implemented using "finite-precision" arithmetic, that is, a limited number of bits. The use of finite-precision arithmetic in IIR filters can cause significant problems due to the use of feedback, but FIR filters have no feedback, so they can usually be implemented using fewer bits, and the designer has fewer practical problems to solve related to non-ideal arithmetic. FIR filters also have a low sensitivity to filter coefficient quantization errors. This is an important property to have when implementing a filter on a DSP processor or on an integrated circuit. FIR filter are high order FIR filters have longer delays. More side lobes in stop band than the IIR filter. FIR filters are a higher order than IIR filters, making FIR filters more computationally expensive [3].

2.2.2 Infinite Impulse Response (IIR) Filter

IIR filter is one whose impulse response is infinite. Impulse response is infinite because there is feedback in the filter. This permits the approximation of many waveforms or transfer functions that can be expressed as an infinite recursive series. These implementations are referred to as Infinite Impulse Response (IIR) filters. The functions are infinite recursive because they use previously calculated values in future calculations to feedback in hardware systems [3].

IIR filters can be mathematically represented as:

$$y(n) = -\sum_{0}^{M-1} a_k y(n-k) + \sum_{0}^{N-1} h_k x(n-k) \tag{2.2}$$

Where a_k is the K^{th} feedback tap. M is the number of feed-back taps in the IIR filter and N is the number of feed forward taps.

IIR Filters are useful for high-speed designs because they typically require a lower number of multiply compared to FIR filters. IIR filters have lower side lobes in stop band as compared to FIR filters. Unfortunately, IIR filters do not have linear phase and they can be unstable if not designed properly. IIR filters are very sensitive to filter coefficient quantization errors that occur due to using a finite number of bits to represent the filter coefficients. One way to reduce this sensitivity is to use a cascaded design. That is, the IIR filter is implemented as a series of lower-order IIR filters as opposed to one high-order.

2.3 Selection between FIR and IIR filter

The choice between the FIR and IIR filter depends largely on the relative advantages of the two filter types:

1) FIR filters can have an exactly linear phase response. The implication of this is that no phase distortion is introduced into the signal by the filter. This is an important requirement in many applications, for example data transmission, biomedicine, digital audio, and image processing. The phase response of IIR filter is nonlinear, especially at the band edges.
2) The effects of using a limited number of bits to implement filters such as round off noise and coefficient quantization errors are much less severe in FIR than IIR.
3) FIR requires more coefficients for sharp cutoff filters than IIR. Thus for a given amplitude response specification, more processing time and storage will be required for FIR implementation. However, one can readily take advantage of the computational speed of the Fast Fourier Transform (FFT) and multirate technique to improve significantly the efficiency of FIR implementations.
4) Analog filters can be readily transformed into equivalent IIR digital filters meeting similar specification. This is not possible with FIR filters as they have no analog converter counterpart. However, with FIR it is easier to synthesize filters of arbitrary frequency response [2].

2.4 Specification of a Filter

Required specifications include specifying the:

1) Signal characteristics (types of signal source and sink, input/output interface, data rates and width and highest frequency of interest).
2) The characteristics of the filter (the desired amplitude and/or phase response and their tolerances, the speed of operation and modes of filtering (real time or batch).

3) The manner of implementation (as a high level language routine in a computer or as a Digital Signal processor-based systems, choice of signal processor), and
4) Other design constraints (the cost of filter) [2].

The designer may not have enough information to specify the filter completely at the outset, but as many of the filter requirements as possible should be specified to simplify the deign process. The characteristics of digital filter are often specified in the frequency domain. For frequency selective filters, such as low pass and band pass filters, the specifications are often in the form of tolerance scheme. Figure 2.3 depicts such a scheme for low pass filter. The shaded horizontal lines indicate the tolerances limits. In the pass band, the magnitude response has a peak deviation of δ_p and in the stop band it has a maximum deviation of δ_s.

The width of the transition band determines the sharpness of the filter. The magnitude response decreases monotonically from pass band to the stop band in this region. The following are key parameters of interest:

δp	Pass band Deviation
δs	Stop band Deviation
fp	Pass edge Frequency
fs	Stop band Edge Frequency

The edge frequencies are often given in the normalized form that is as a fraction of the sampling frequency ($f \nearrow F_s$), whereas f is input frequency and F_s is the sampling frequency but specifications using standard frequency units of hertz or kilohertz are valid and sometimes are more meaningful. Pass band and and stop band deviations may be expressed as ordinary numbers or in decibels (db) when they specify the pass band ripple and minimum stop band attenuation respectively. Thus the minimum stop band attenuation, A_s and the peak band ripple, Ap decibels are given as (FIR filers).

A_s (Stop band attenuation) $= -20 \log_{10} D$

Ap (Pass band ripple) $= 20 \log_{10} (1+ \delta_p)$

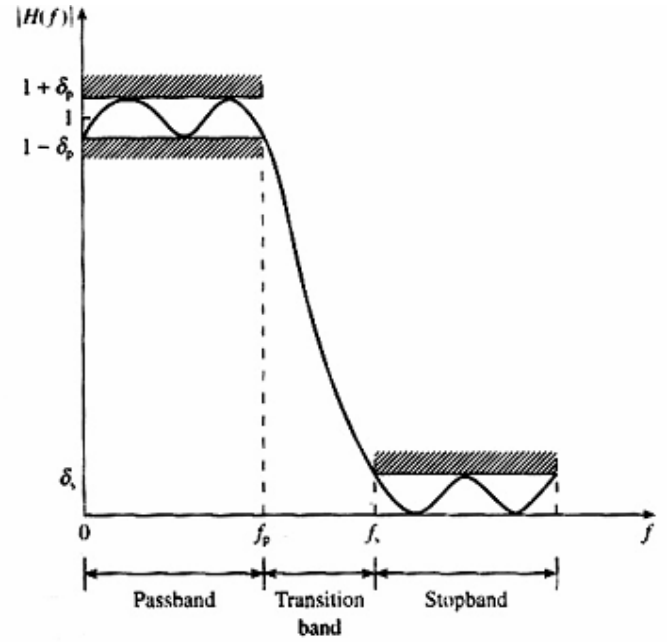


Figure 2.3 Tolerance Scheme for a Low pass Filter

2.5 Coefficient Calculation

One of a number of approximation methods is selected and the values of the coefficients, h(k) for FIR or a_k and b_k system parameters for IIR are calculated. The two basic methods used are the impulse invariant and bilinear transformation methods. With the impulse invariant method, after digitizing the analog filter, impulse response of the original analog filter is preserved, but not its magnitude frequency response. Because of the inherent aliasing, the method is inappropriate for high pass or band stop filters. The bilinear method on the other hand yields very efficient filters and is well suited to the calculation of the coefficients of the frequency selective filters. It allows the design of digital filters with known classical characteristics such as Butterworth, Chebyshev and Elliptic. Digital filters resulting from bilinear transformation will preserve the magnitude response characteristics of the analog filter but not the time domain properties. Efficient computer programs now exist for calculating filter coefficients, using the bilinear method, by merely specifying filter parameters of interest. The impulse invariant method is good for simulating analog systems, but the bilinear method is best for frequency selective IIR filters. The pole zero placement method offers an alternative approach to calculating the coefficients the of IIR filters. It is an easy way of calculating the coefficients of very simple filters. However, for filters with good amplitude response it is not recommended as it relies on 'trial' and 'error' shuffling of the pole zero positions [2].

As with IIR filters there are several methods for calculating the coefficients of FIR filters. The three methods are the window, frequency sampling, and the optimal. The window method offers a very simple and flexible way of computing FIR filters coefficients, but it does not allow the designer adequate control over the filter parameters. The main attraction of the frequency sampling method is that it allows a recursive realization of FIR filters which can be computationally very efficient. However, it lacks flexibly in specifying or controlling filter parameters. With the availability of an efficient and easy-to-use program, the optimal method is now widely used in industry and for most applications will yield the desired FIR filters. In summary, there are several methods of calculating the coefficients of which the following are the most widely used:

1. Impulse invariant (IIR)
2. Bilinear transformation (IIR)
3. Pole –zero placement (IIR)
4. Windows (FIR)
5. Frequency sampling (FIR)
6. Optimal (FIR)

2.6 Filter Design by Windowing Method

The windowing method of FIR filter design bases the filter impulse response sequence (and consequently, the filter transfer function) on the coefficients of Fourier series expansion of the desired frequency response function. Since the series will be infinite and non casual so it is unrealizable. So the series must be truncated to get finite impulse response. This means of truncation is a technique known as windowing. But this results in undesirable oscillations in the pass band and stop band of digital filter. These undesirable oscillations can be reduced by using set of time limited weighting functions, w(n) referred as window function. A variety of windows are available for use. This method begins with the desired frequency response specification $H_d(\omega)$ and determine the corresponding unit sample response $h_d(n)$. Indeed $h_d(n)$ is related to $H_d(\omega)$ by the Fourier Transform relation

$$H_d(\omega) = \sum_{n=0}^{\infty} h_d(n) e^{-j\omega n} \tag{2.3b}$$

$$h_d(n) = \frac{1}{2\pi} \int_{-\pi}^{+\pi} H_d(\omega) e^{j\omega n} d\omega \tag{2.3c}$$

thus given $H_d(\omega)$ from it the unit sample response $h_d(n)$ can determined by the above integral. The unit sample response obtained is infinite in duration and must be truncated at some point at some point n = N-1, to yield an FIR filter of length N. Truncation of $h_d(n)$ to a length N-1 is equivalent to multiplying $h_d(n)$ by a rectangular window defined as [3].

w(n) = 1, n = 0,1,…….N-1 (2.4a)

w(n) = 0, otherwise (2.4b)

The unit sample response of FIR filter becomes $h(n) = h_d(n)w(n)$

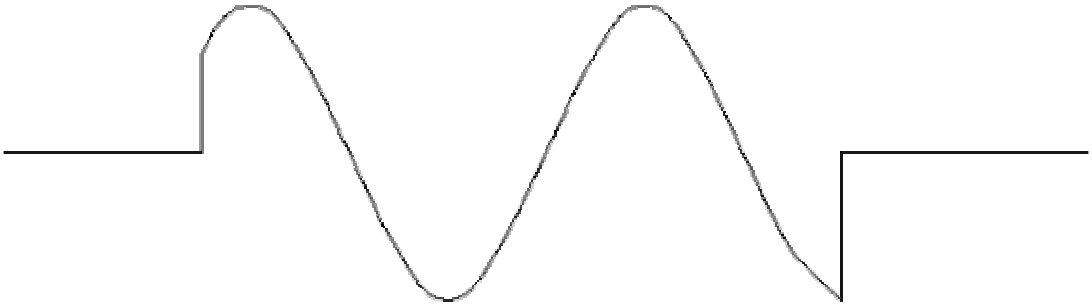


Figure 2.4 Waveform Truncated with Rectangular Window

The multiplication of window function w(n) with $h_d(n)$ is equivalent to convolution of $H_d(\omega)$with $W(\omega)$ where $W(\omega)$ is the frequency representation of window function.

$$w(\omega) = \sum_{n=0}^{N-1} e^{-j\omega n} \tag{2.5a}$$

$$H(\omega) = \frac{1}{2\pi}\int_{-\pi}^{+\pi} H_d(v)W(\omega - v)dv \tag{2.5b}$$

Fourier transform of rectangular window is

$$w(\omega) = \sum_{n=0}^{N-1} e^{-j\omega n} \tag{2.6a}$$

$$w(\omega) = e^{-j\omega(N-1)}\frac{\sin(\omega N/2)}{\sin(\omega/2)} \tag{2.6b}$$

Magnitude response can be calculated by taking modulus of above equation.

2.7 Convolution

Basically convolution gives the response y(n) of the linear time invariant (LTI) systems as a function of the input signal x(n) and the unit sample (impulse) response h(n) .The input x(n) is convolved with impulse response h(n) to yield to the output y(n).

$$y(n) = \sum_{k=-\infty}^{k=+\infty} x(k)h(n-k) \tag{2.7a}$$

The process of computing the convolution between x(k) and h(k) involves the following four steps:

a) Folding: Fold h(k) about k = 0 to obtain h(-k).
b) Shifting: Shift h(-k) by n_0 to right (left) if n_0 is positive (negative), to obtain $h(n_0 - k)$.
c) Multiplication: Multiply x(k) by $h(n_0-k)$ to obtain the product sequence.
d) Summation: Sum all the values of the product sequence to obtain the values of the output at the time $n = n_0$ [3].

The procedure gives y(n) at a single instant say, $n = n_0$. One is interested in evaluating the response of the system over all time instants $-\infty < n < \infty$. Consequently, steps 2 through 4 in summary must be repeated, for all possible time shifts $-\infty < n < \infty$. For the signals $x_1(n)$ and $x_2(n)$, the convolution of these two signals is given by:

$$x_3(n) = x_1(n) * x_2(n) = \sum_{k=-\infty}^{\infty} x_1(n)\, x_2(n) \tag{2.8}$$

2.7.1 Properties of Convolution

Convolution satisfies three main properties:

1) Commutative law: The convolution of two signals x(n) and h(n) is given b

$$y(n) = x(n) * h(n) = \sum_{k=-\infty}^{k=+\infty} x(k)\, h(n-k) \tag{2.9a}$$

$$y(n) = h(n) * x(n) = \sum_{K=-\infty}^{\infty} h(k)\, x(n-k) \tag{2.9b}$$

In the first case the impulse response h(n) is folded and shifted and x(n) is the excitation signal. In the second case the input signal x(n) is folded and shifted. Hence h(n) acts as the excitation signal.

2) Associative Law:

$$\left[x(n) * h\ (n)\right] * h_2(n) = x(n) * \left[h_1(n) * h_2(n)\right] \tag{2.10a}$$

If N linear time invariant systems are in cascade with impulse responses $h_1(n), h_2(n), \ldots, h_N(n)$ then equivalent system impulse response is given by:

$$h(n) = h_1(n) * h_2(n) * .....h_N(n) \tag{2.10b}$$

3 Distributive Law:

$$x(n) * \left[h_1(n) + h_2(n)\right] = x(n) * h_1(n) + x(n) * h_2(n) \tag{2.11a}$$

The distributive law states that if there are N numbers of LTI systems with impulse responses $h_1(n),h_2(n),\ldots,h_N(n)$ excited by the same input x(n), then equivalent system impulse response is given by

$$h(n) = \sum_{l=1}^{N} h_1(n) \qquad (2.11b)$$

This is nothing but the parallel interconnection of individual systems.

2.8 Effect of Finite Word Length in Digital Filters

When digital systems are implemented either in hardware or in software, the filter coefficients are stored in binary registers. These registers can accommodate only a finite number of bits and hence, the filter coefficients have to be truncated or rounded off in order to fit into these registers. Truncation and rounding-off the data results in degradation of system performance. Also in digital processing systems, a continuous time input signal is sampled and quantized in order to get the digital systems.

There are four ways in which finite word length affects the performance of FIR digital filter:

1) Quantization effects in analog-to-digital conversion.
2) Product quantization and coefficient quantization errors in digital filters
3) Limit cycles in IIR filters and
4) Finite word length effects in Fast Fourier transform [1].

2.8.1 Rounding and Truncation Errors

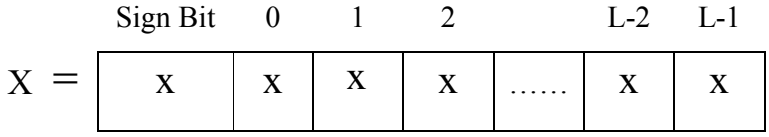
Rounding and truncation introduces an error whose magnitude depends on the number of bits truncated or rounded-off. Also, the characteristics of the error depend on the form of binary number representation. The sign magnitude and two's complement representation of fixed point binary numbers are considered here. Table 2.1 gives the sign magnitude and two's complement representation of fixed-point numbers.

Number	Sign Magnitude	Two's Complement

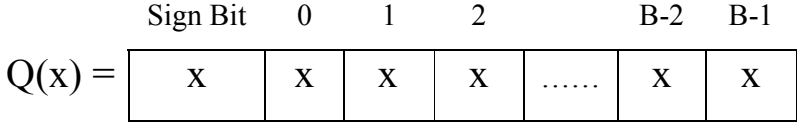
7	0111	0111
6	0110	0110
5	0101	0101
4	0100	0100
3	0011	0011
2	0010	0010
1	0001	0001
0	0000	0000
-0	1000	0000
-1	1001	1111
-2	1010	1110
-3	1011	1101
-4	1100	1100
-5	1101	1011
-6	1110	1010
-7	1111	1001

Table 2.1 Fixed-Point Binary Representation

Consider a number x whose original length is 'L' bits. Let this number be quantized to 'b' bits as shown below. This quantized number is represented by Q(x). Both x and Q(x) are shown below. Note B<L.



and



A Truncation error, ε_T, is introduced in the input signal and thus the quantized signal is

$$Q_t(x) = x + \varepsilon_T \qquad (2.12)$$

The range of values of the error due to truncation of the signal is analyzed here for both sign magnitude and two's complement representation.

1) Truncation Error for Sign Magnitude Representation: when the input number x is positive, truncation results in reducing the magnitude of the number. Thus the truncation error is negative and range is given by

$$-(2^{-B} - 2^{-L}) \leq \varepsilon_T \leq 0 \qquad (2.13a)$$

The largest error occurs when all the discarded bits are one. When the number x is negative, truncation results in reduction of the magnitude only. However, because of the negative sign, the resulting number will be greater than the original number. Let the number be x = -0.374. That is, in sign magnitude form it is represented as x = 1011 and after truncation of one bit, Q(x) = 101. This is equivalent to -0.25 in decimal. But -0.25 is greater than 0.375. Therefore, the truncation error is positive and its range is

$$0 \leq \varepsilon_T \leq (2^{-B} - 2^{-L}) \qquad (2.13b)$$

The overall range of the truncation error for the sign magnitude representation is

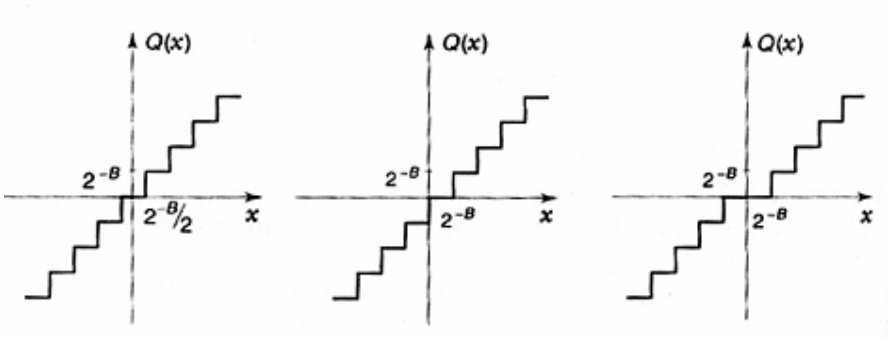
$$-(2^{-B} - 2^{-L}) \leq \varepsilon_T \leq (2^{-B} - 2^{-L}) \qquad (2.13c)$$

2) Truncation Error for Two's Complement Representation: when the input number is positive, truncation results in a smaller number, as in case of sign magnitude numbers. Hence, the truncation error is negative and its range is same as given in Equation (2.13a). If the number is negative, truncation of the number in two's complement form results in a smaller number and error is negative. Thus the complete range of truncation error for two's complement representation is

$$-(2^{-B} - 2^{-L}) \leq \varepsilon_T \leq 0 \qquad (2.14)$$

3) Round-off for Sign Magnitude and Two's Complement Representation: the rounding of a binary number involves only the magnitude of the number and is independent of the type of fixed-point binary representation. The error due to rounding may either positive or negative and the peak value is $(2^{-B} - 2^{-L})/2$.The round off error is symmetric about zero and its range is

$$-(2^{-B} - 2^{-L})/2 \leq \varepsilon_R \leq (2^{-B} - 2^{-L})/2 \qquad (2.15)$$



(a) Rounding (b) Truncation in 2's complement (c) Truncation in sign magnitude

Figure 2.5 Quantization Error in Rounding and Truncation

In most cases infinite precision is assumed, i.e. the length of the unquantized number is assumed to be infinity and as result Equations (2.13c),(2.14),(2.15) can be modified and the range of error for different cases are as follows:

1) Truncation error for sign magnitude representation

$$-2^{-B} \leq \varepsilon_T \leq 2^{-B} \qquad (2.16a)$$

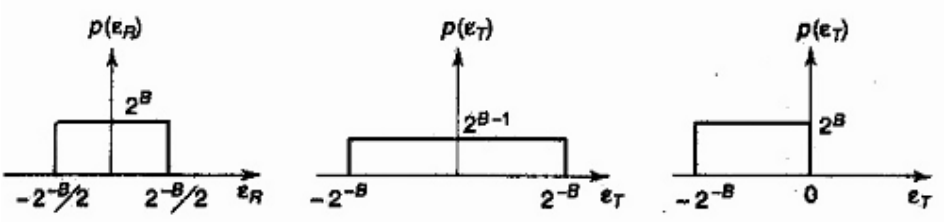
2) Truncation error for sign two's complement representation

$$-2^{-B} \leq \varepsilon_T \leq 0 \qquad (2.16b)$$

3) Round-off error for sign magnitude and two's complement representation

$$-2^{-B}/2 \leq \varepsilon_T \leq 2^{-B}/2 \qquad (2.16c)$$

Figure (2.6) shows quantization error in truncation and rounding-off numbers. In computations involving quantization, a statistical approach is used in characterization of these errors [1].



(a) Rounding- error (b) Sign magnitude Truncation error (c) Two's complement Truncation error

Figure 2.6 Probabilistic Characteristics of Quantization Errors

2.8.2 Round-Off Error in FIR Filter

The difference equation for FIR filter is given by:

$$y(n) = \sum_{m=0}^{N-1} h(m)x(n-m) \qquad (2.17)$$

where each variable is represented by a fixed number of bits. Typically the input and the output samples, x(n-m) and y(n) are each represented by 12-bits and the coefficients by 16-bits in 2's complement format. It is seen from Equation (2.17) that the output of the filter is obtained as the sum of products of h(m) and x(n-m). After each multiplication, the product contains more bits than either h(m) or x(n-m).If a 12-bit input is multiplied by the 16-bit coefficient the result is 28-bit long and will need to be quantized back to 16-bits before it can be stored in a memory or to 12-bits before it can be output to DAC . This quantization leads to errors whose effects are similar to those of ADC noise, but could be more severe. The common way to quantize the result of an arithmetic operation is to truncate the result, that is a) to retain the most significant higher-order bits and to discard the lower-order bits b) to round the result, that is to choose the higher-order bits closest to the unrounded results. This is achieved by adding half an LSB to the result [3].

2.8.3 Overflow Errors

Overflow occurs when the sum of two numbers, usually two large numbers of the same sign, exceeds the permissible word length. Thus in the Equation (2.17) overflow could occur when products of h(0) x(n) and h(1) x(n-1) are added provided that the final output, y(n) is within the permissible word length overflow in partial sum is unimportant. This is a desirable property of the 2's complement arithmetic. However, if the output, y(n) exceeds the permissible limit then clearly the value of the output sample to the DAC will be wrong and steps should be taken to prevent this.

2.9 8052 Microcontroller

The 8052 is an 8-bit microcontroller originally developed by Intel in the late 1970s. It included an instruction set of 255 operation codes (opcodes), 32 input/output lines, three user-controllable timers, an integrated and automatic serial port, and 256 bytes of on-chip Random Access Memory (RAM). The 8051 is a very similar Microprocessor Control Unit (MCU) but it has only two timers and 128 bytes of on-chip RAM. The 8052 was designed such that control of the MCU and all input/output between the MCU and external devices is accomplished via Special Function Registers (SFRs). Each SFR has an address between 128 and 255. Additional functions can be added to new derivative MCUs by adding additional

SFRs while remaining compatible with the original 8052. This allows the developer to use the same software development tools with any MCU that is “8052-compatible”.

2.9.1 Types of Memory

The 8052 has three very general types of memory. The memory types are illustrated in the following graphic. They are: On-Chip Memory, Code Memory, and External RAM.

Code Memory is code (or program) memory that is used to store the actual program. This often resides off-chip in the form of an Erasable Programmable Read Only Memory (EPROM). Many derivative chips allow program storage on the MCU itself (Internal Code Memory) and some modern derivative chips may not even support the concept of having code memory located off-chip. Figure 2.7shows the types of memory in 8052 microcontroller.

External RAM is RAM memory that resides off-chip. This is often in the form of standard static RAM or flash RAM [4].

On-Chip Memory refers to any memory (Code, RAM, or other) that physically exists on the MCU itself.

2.9.1.1 Code Memory

Code memory is the memory that holds the actual 8052 program that is to be run. This memory is conventionally limited to a maximum of 64K and comes in many shapes and sizes: Code memory may be found on-chip, either burned into the microcontroller as ROM or EPROM or loaded into flash program memory. Code may also be stored completely off-chip in an external Read Only Memory (ROM) or, more commonly, an external EPROM. Flash memory is another popular method of storing a program.

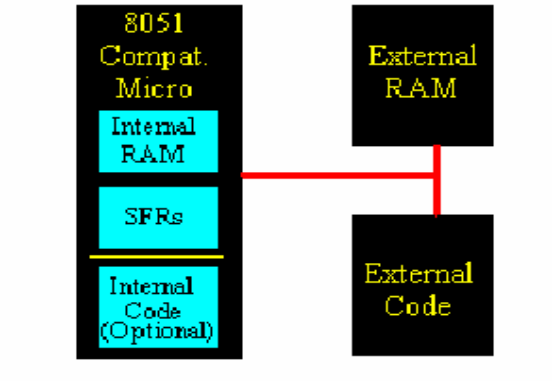


Figure 2.7 Types of Memory

2.9.1.2 External RAM

Although 8052s contain a small amount of on-chip RAM, External RAM is also supported. External RAM is any random access memory that is found off-chip. Since the memory is off-chip the assembly language instructions to access it are slower and less flexible. To increment an Internal RAM location by 1 requires only 1 instruction and 1 instruction cycle. To increment a 1-byte value stored in External RAM requires 4 instructions and 7 instruction cycles. In this case, external memory is 7 times slower and requires 4 times as much program memory. While Internal RAM is normally limited to 256 bytes (128 with 8051s), the 8052 supports External RAM up to 64K.

2.9.1.3 On-Chip Memory

On-chip memory is really one of two types: Internal RAM and Special Function Register (SFR) memory. The layout of the 8052's internal memory is presented in the following memory map (2.8).

8052 has a bank of 256 bytes of Internal RAM. This Internal RAM is found on-chip within the 8052 so it is the fastest RAM available, and it is also the most flexible in terms of reading, writing, and modifying its contents. Internal RAM is volatile so when the 8052 is reset this memory is undefined. The 256 bytes of Internal RAM are

subdivided as shown in the memory map. The first 8 bytes (00H – 07H) are "register

bank 0". By manipulating a certain SFR, a program may choose to use register banks 0, 1, 2, or 3. These alternative register banks are located in internal RAM in addresses 08H through 1FH. All of Internal RAM is byte-wide memory, regardless of whether it is used by register banks or bit memory.

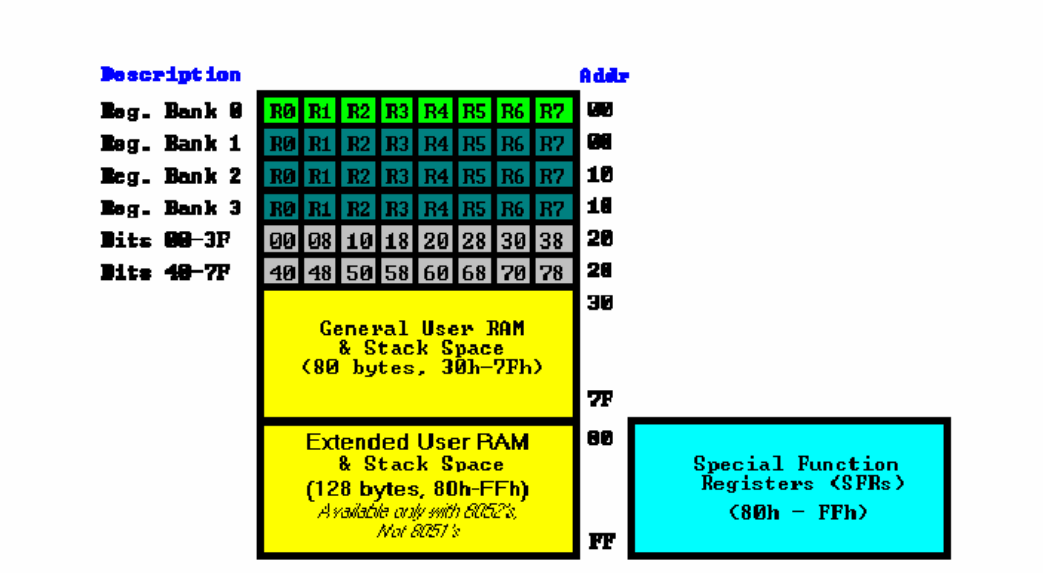


Figure 2.8 Layout of 8052's Internal Memory

The stack is a "last in, first out" (LIFO) storage area that exists in Internal RAM. It is used by the 8052 to store values that the user program manually pushes onto the stack as well as to store the return addresses for CALLs and interrupt service routines. The stack is defined and controlled by a Special Function Register called the Stack Pointer, or SP. SP, as a standard 8-bit SFR, holds a value between 0 and 255 that represents the Internal RAM address of the end of the current stack. If a value is removed from the stack, it will be taken from the Internal RAM address pointed to by Stack Pointer (SP) and SP will subsequently be decremented by 1. If a value is pushed onto the stack, SP will first be incremented and then the value will be inserted in Internal RAM at the address now pointed to by SP. SP is initialized to 07H when an 8052 is first booted. This means the first value to be pushed onto the stack will be placed at Internal RAM address 08h (07h + 1), the second will be placed at 09h, etc.

The 8052 uses 8 "R" registers which are used in many of its instructions. These "R" registers are numbered from 0 through 7 (R0, R1, R2, R3, R4, R5, R6, and R7) and are generally used to assist in manipulating values and moving data from one memory location to another. As the memory map shows, the 8052 has four distinct register banks. When the 8052 is first

booted up register bank 0 (addresses 00H through 07H) is used by default. If program instructs the 8052 to use register bank 1, register R4 will now be synonymous with Internal RAM address 0CH. If register bank 2 is selected R4 is synonymous with 14H, and register bank 3 is selected it is synonymous with address 1CH [4].

The 8052, being a communications and control-oriented microcontroller that often has to deal with "on' and "off" situations, gives the ability to access a number of bit variables directly with simple instructions to set, clear, and compare these bits. These variables may be either 1 or 0. There are 128 bit variables available numbered 00H through 7FH. Bit Memory, like the Register Banks above, is really a part of Internal RAM. In fact, the 128 bit variables occupy the 16 bytes of Internal RAM from 20H through 2Fh. 8052 provides special instructions to access these 16 bytes of memory on a bit-by-bit basis. It is just a subset of Internal RAM and that operations performed on Internal RAM can change the values of the bit variables. While Bit Memory 00H through 7FH are for developer-defined functions in the programs, Bit Memory 80H and above are used to access certain SFRs on a bit-by-bit basis.
Special Function Registers (SFRs) are areas of memory that control specific functionality of the 8052 MCU. Four SFRs permit access to the 8052's 32 input/output lines (8 lines per SFR). Another SFR allows a program to read or write to the 8052's serial port. Other SFRs allow the user to set the serial baud rate, control and access timers, and configure the 8052's interrupt system.

2.10 Interfacing to the External Electrically Erasable PROM:

EEPROMs are electrically-erasable-and-programmable. Internally, they are similar to EPROMs, but the erase operation is accomplished electrically, rather than by exposure to ultraviolet light. Any byte within an EEPROM may be erased and rewritten. Once written, the new data will remain in the device forever or at least until it is electrically erased. The primary trade-off for this improved functionality is higher cost, though write cycles are also significantly longer than writes to a RAM. It wouldn't want to use an EEPROM for main system memory. An EEPROM can be programmed and erased multiple times electrically. It may be erased and reprogrammed only a certain number of times, ranging from 100,000 to 1,000,000, but it can be read an unlimited number of times.
In many systems where the on-chip ROM of the 8052 is not sufficient, the use of external memory is ideal since it allows the program size to be as large as 64K bytes. It is much more

expensive since the ROM containing the program code is connected externally and requires more supporting circuitry. Some of the pins of the 8052 are used in external memory interfacing are reviewed [5].

- **EA Pin**

EA pin is connected to V_{CC} to indicate that program code is stored in the microcontroller's on chip ROM. To indicate that program code is stored in the external ROM this pin must be connected to GND.

- **P0 – P2 Role in Providing Addresses**

Since program counter of the 8052 is 16-bit, it is capable of accessing up to 64K bytes of program code. In the 8052 Port0 and Port2 provide the 16-bit address to access external memory. Of these two ports, P0 provides the lower 8-bit address A0 –A7, and P2 Provides the upper the upper 8-bit address A8 –A15. More importantly, P0 is also used to provide the 8-bit data bus D0-D7. In other words, pins P0.0-P0.7 are used for both the address and data paths. This is called address/data multiplexing in chip design. ALE is an output pin for the 8052 microcontroller. Therefore, when ALE=0 the 8051 uses P0 for the data path and when ALE=1, it is used for the address path. As a result the addresses from the P0 pins we connect to a 74LS373 latch and use the ALE pin to latch the address. This extracting of addresses from P0 is called address/data demultiplexing. Figure 2.9 shows interfacing of microcontroller to the external code memory.

- **PSEN PIN**

To connect the 8031/51 to external ROM containing data RD signal is used to fetch the data. For ROM containing program code PSEN is used to fetch data.

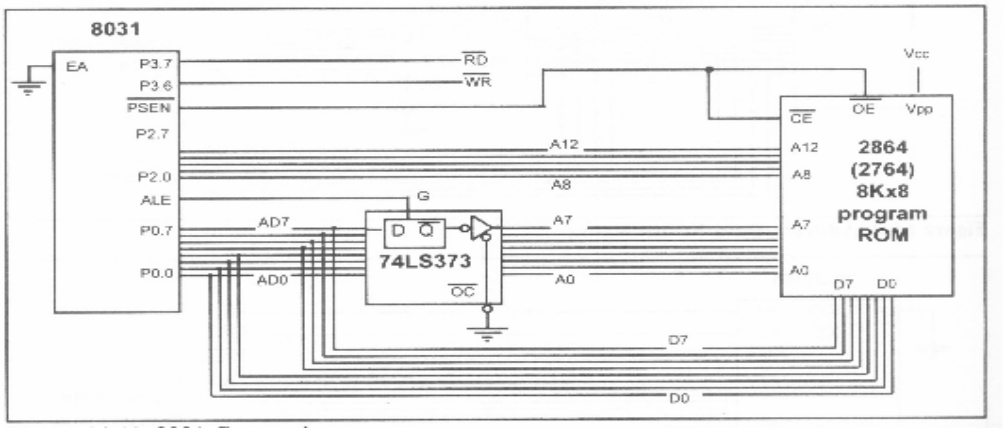


Figure 2.9 Interfacing with External Program ROM

2.11 Interfacing Liquid Crystal Display (LCD) to the 8052

In recent years the LCD is finding widespread use replacing Light Emitting Diodes (LEDs). This is due to following reasons:

1) The declining prices of LCDs.
2) The ability to display numbers, characters and graphics .This is in contract to LEDs, which are limited to numbers and a few characters.
3) Incorporation of a refreshing controller into the LCD, thereby relieving the CPU of the task of refreshing the LCD. In contract, the LED must be refreshed by the CPU to keep displaying the data.
4) Ease of programming for characters and graphics.

2.11.1 LCD Pin Description

The LCD has 14 pins the function of all the pins is given below:

V_{CC} , V_{SS}, and V_{EE}

While Vcc and Vss provide +5V and ground respectively, V_{EE} is used for controlling LCD contract.

RS, Register Select

There are two important registers inside the LCD. The RS pin is used for their selection as follows: If RS = 0, the instruction command code register is selected, allowing to send a command such as clear display, curser at home, etc. If RS =1 the data register is selected, allowing to send data to be displayed on the LCD [5].

R/W, Read/Write

R/W input allows writing information to the LCD or reading information from it. R/W=1 when reading, R/W=0 when writing.

E, Enable

The enable pin is used by the LCD to latch information presented to its data pins. When data is supplied to data pins, a high-to-low pulse must be applied to this pin in order for the LCD to latch in the data present at the data pins. This pulse must be a minimum of 450ns wide.

D0-D7

The 8-bit data pins, D0-D7, are used to send information to the LCD or read the contents of the LCD's internal registers.

To display letters and numbers, ASCII codes are send for the letters A-Z, a-z, and numbers 0-9 to these pins while making RS=1.There are also instructions command codes that can be send to the LCD to clear the display or force the cursor to the home position or blink the

curser. RS=0 is also used to check the busy flag bit to see if LCD is ready to receive information. The busy flag is D7 and can be read when R/W =1 and RS=0, as follows: if R/W =1, RS=0.When D7=1, the LCD is busy taking care of internal operations and will not accept any new information [5].

2.12 Interfacing the Keyboard to 8052

At the lowest level, keyboards are organized in a matrix of row and column .The Central Processing Unit (CPU) accesses both the rows and columns through ports therefore, with two 8-bit ports, an 8*8 matrix of keys can connect to a microcontroller. When a key is pressed, a row and columns make a contact otherwise, there is no connection between rows and columns.

Figure 2.10 shows a 4*4 matrix connected to two ports. The rows are connected to an output port and the columns are connected to an input port. If no key has been pressed, reading the input port will yield 1s or all columns since they are connected to high (Vcc). If all the rows are grounded and a key is pressed, one of all the columns will have 0 since the key pressed provides the path to ground. It is the function of microcontroller to scan the keyboard continually to detect and identify the key pressed.

To detect a pressed key, microcontroller grounds all rows by providing 0 to output latch, then it reads the columns. If data read from the columns is D3-D0 = 1111, no key has been pressed and then the process continues until a key press is detected. However, if one of the column bit has a 0, this means that a key in the D1 columns has been pressed. After a key pressed has been detected, the microcontroller will go through the process of identifying the key. Starting with top row, the microcontroller grounds it by providing a low to row D0 only then it reads the columns. If the data read is all 1s, no key in that row is activated and the process is moved to the next row. It grounds the next row, reads the columns, and checks for any 0. This process continues until the row is identified. After identification of the row in which key has been pressed key belongs to.

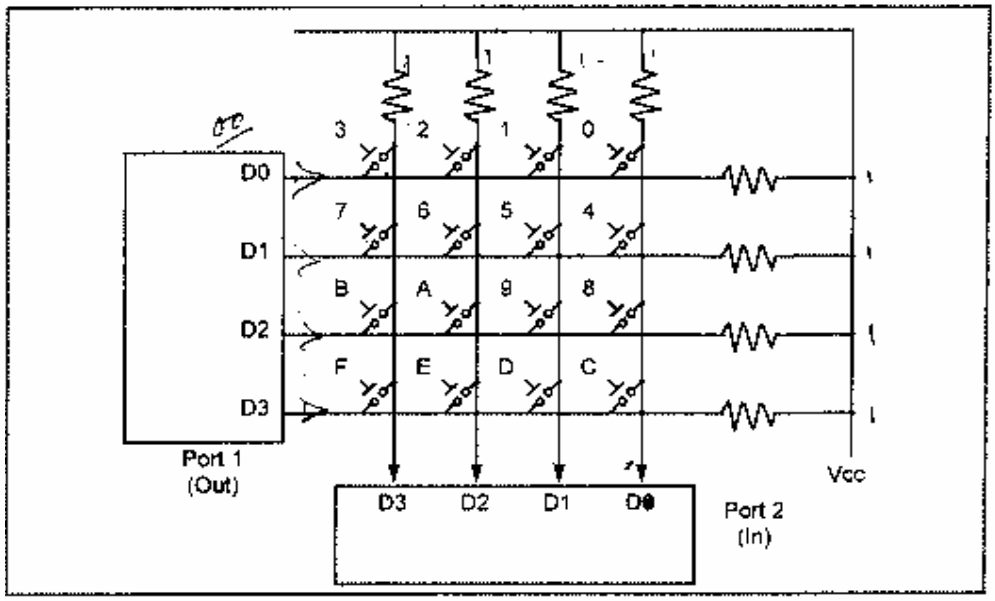


Figure 2.10 Interfacing of Microcontroller to 4*4 Hex Keypad

2.13 Introduction to Matrix Laboratory (MATLAB)

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include:

1) Math and computation.
2) Algorithm development.
3) Data acquisition.
4) Modeling, simulation, and prototyping.
5) Data analysis, exploration, and visualization.
6) Scientific and engineering graphics.
7) Application development, including graphical user interface building

MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. This allows to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar non interactive language such as C or Fortran. The name MATLAB stands for matrix laboratory. MATLAB was originally written to provide easy access to matrix software developed by the LINPACK and EISPACK projects. Today, MATLAB engines incorporate the LAPACK and BLAS libraries, embedding the state of the art in software for matrix computation. MATLAB has evolved over a period of years with input from many users. In

university environments, it is the standard instructional tool for introductory and advanced courses in mathematics, engineering, and science.

In industry, MATLAB is the tool of choice for high-productivity research, development, and analysis. MATLAB features a family of add-on application-specific solutions called toolboxes. Very important to most users of MATLAB, toolboxes allow you to learn and apply specialized technology [6].

2.14 Universal Microprocessor Program Simulator (UMPS)

UMPS runs under the Microsoft Windows, is able to handle the development for a large number of different microcontrollers. Along with an editor and assembler, UMPS contains the ability to simulate devices with external hardware, which really sets UMPS apart from other tools and eliminates the need for stimulus files. This capability of connecting UMPS to virtual devices gives us a real "what if" capability in our application [7]

UMPS runs under Microsoft Windows (3.11, 95, and NT) Personal Computer (PC) operating systems. Along with its own internal assembler, UMPS can initiate the operation of the Microchip PIC Assembler as well as Cosmic Assembler and C Complier and their symbolic output for the UMPS simulator. The most powerful and useful feature of UMPS is its graphical "resources" that can be "wired" to the device. At the time of writing, the virtual hardware for the simulated microcontroller includes:

1) LED
2) Push button
3) Logic functions
4) 7-segment LEDs
5) Square wave generator
6) D/A converter
7) A/D converter
8) A/D slider
9) Serial receivers /transmitters

UMPS currently supports the following devices:

1) Dallas semiconductor; DS87C310, DS87C320
2) Intel : 8031, 8032, 8051
3) Atmel : AT89C1051, AT89C2051
4) Microchip : PIC2C5xx, PIC165C5x, PIC16C84, PIC16F84, PIC16F83

LIU 1971 [8] described the effect of finite word length on the accuracy of digital filter. The three common sources of error due to finite word length are the quantization of the input signal x_n into a set of discrete levels, the representation of the filter coefficients a_k and b_k by a finite number of bits, the accumulation of round-off errors committed at arithmetic operations. The accuracy of a digital filter also depends on two important factors: the form of realization and the type of arithmetic used. Paper discusses the range of errors when fixed-point or floating point numbers are rounded-off. The effects input quantization, co-efficient inaccuracy, effect of rounding-off accumulation in direct form and floating-point realization are discussed in detail.

CROCHIERE 1975 [9] proposed a new statistical approach for estimating the necessary coefficient word length of a digital filter. With this statistical word length definition, an optimization procedure is then proposed for minimizing the statistical word length for a given filter structure and a given set of maximum error constraints. It is shown that by minimizing this statistical word length, the actual coefficient word length upon rounding of the coefficients can generally be reduced as well. Several examples are given and improvements of one to three bits in the actual coefficient word length are observed. The procedure does not necessarily lead to the actual global minimum coefficient word length.

CHAN and RABINER 1975 [10] purposed an analysis of the three possible types of quantization effects in the direct form realization of finite impulse response (FIR) digital filters. These quantization effects include round-off noise, A-D noise, and filter frequency response errors due to coefficient quantization. Since the analysis of round-off noise and A-D noise for the direct form is straightforward, he concentrated on an analysis of the effects of quantized coefficients on the resulting filter frequency response. Based on this analysis, statistical bounds on the error incurred in the frequency response of a filter due to coefficient quantization are developed and verified by extensive experimental data. Using these bounds, a procedure for applying known techniques for FIR filter design to the design of filters with finite word length coefficients is presented. On the whole, the direct form is shown to be a very attractive structure for realizing FIR filters.

OPPENHIEM and WEINSTEIN 1972 [11] explained the effects of finite Register length on in Digital Filtering and Fast Fourier Transform. They discussed that when digital signal processing operations are implemented on a computer or with special-purpose hardware, errors and constraints due to finite word length are unavoidable. The main categories of finite register length effects are errors due to A/D conversion, errors due to round-offs in the arithmetic, constraints on signal levels imposed by the need to prevent overflow, and

quantization of system coefficients. The effects of finite register length on implementations of linear recursive difference equation digital filters, and the fast Fourier transform (FFT), are discussed in some detail. For these algorithms, the differing quantization effects of fixed-point, floating-point, and block floating point arithmetic are examined and compared. The paper is intended primarily as a tutorial review of a subject which has received considerable attention over the past few years. The groundwork is set through a discussion of the relationship between the binary representation of numbers and truncation or rounding, and a formulation of a statistical model for arithmetic round-off. The analyses presented here are intended to illustrate techniques of working with particular models. Results of previous work are discussed and summarized when appropriate. Some examples are presented to indicate how the results developed for simple digital filters and the FFT can be applied to the analysis of more complicated systems which use these algorithms as building blocks.

Kodek 2005 [12] in many practical situations, it is necessary to represent the coefficients of a finite impulse response (FIR) digital filter by a finite number of bits. This not only degrades the filter frequency response but also introduces a theoretical limit on the performance of the filter. Derivation of a lower bound on filter degradation is the purpose of this paper. Consider a general case of a length filter with a discrete set of allowable coefficients. A theorem that gives the lower bound on the increase in minimax approximation error that is caused by the finite word length restriction is presented. Its extension and application to filter design cases is demonstrated. The importance of this bound is not only theoretical. Its practical effectiveness is shown in the algorithm for optimal finite word length FIR filter design where it significantly reduces the amount of computation.

CHAPTER 3

SYSTEN DESIGN AND IMPLEMENTATION

This chapter describes about the hardware implementation of Low pass Finite Impulse Response (FIR) digital filter. Interfacing of microcontroller with various peripherals is discussed in detail. In the later section of the chapter software implementation of the filter is discussed.

3.1 Hardware details of the FIR Filter

Figure 3.1 shows the Printed Circuit Board (PCB) schematic of the filter. The main components required for the implementation of hardware are given Table 3.1.

Name of the Components	IC Number
Microcontroller	AT89S52
E2PROM Memory	AT28C64
Latch	SN74LS737
LCD Display	44780
Keypad	4*4 Matrix Hex Keypad
Regulator	LM7805C
Power Supply	12V
Crystal	11.0592 MHz
PCB Push Buttons	4

Table 3.1 Component's Details

In the hardware of the FIR filter 8-bit microcontroller is used. It is a 40-pin dual-in-line package Integrated Circuit (IC). The AT89S52 is a low-power, high-performance CMOS 8-bit microcontroller with 8K bytes of in-system programmable Flash memory. The AT89S52 provides the following standard features: 8K bytes of flash, 256 bytes of RAM, 32 I/O lines, on-chip oscillator and clock circuitry [13].

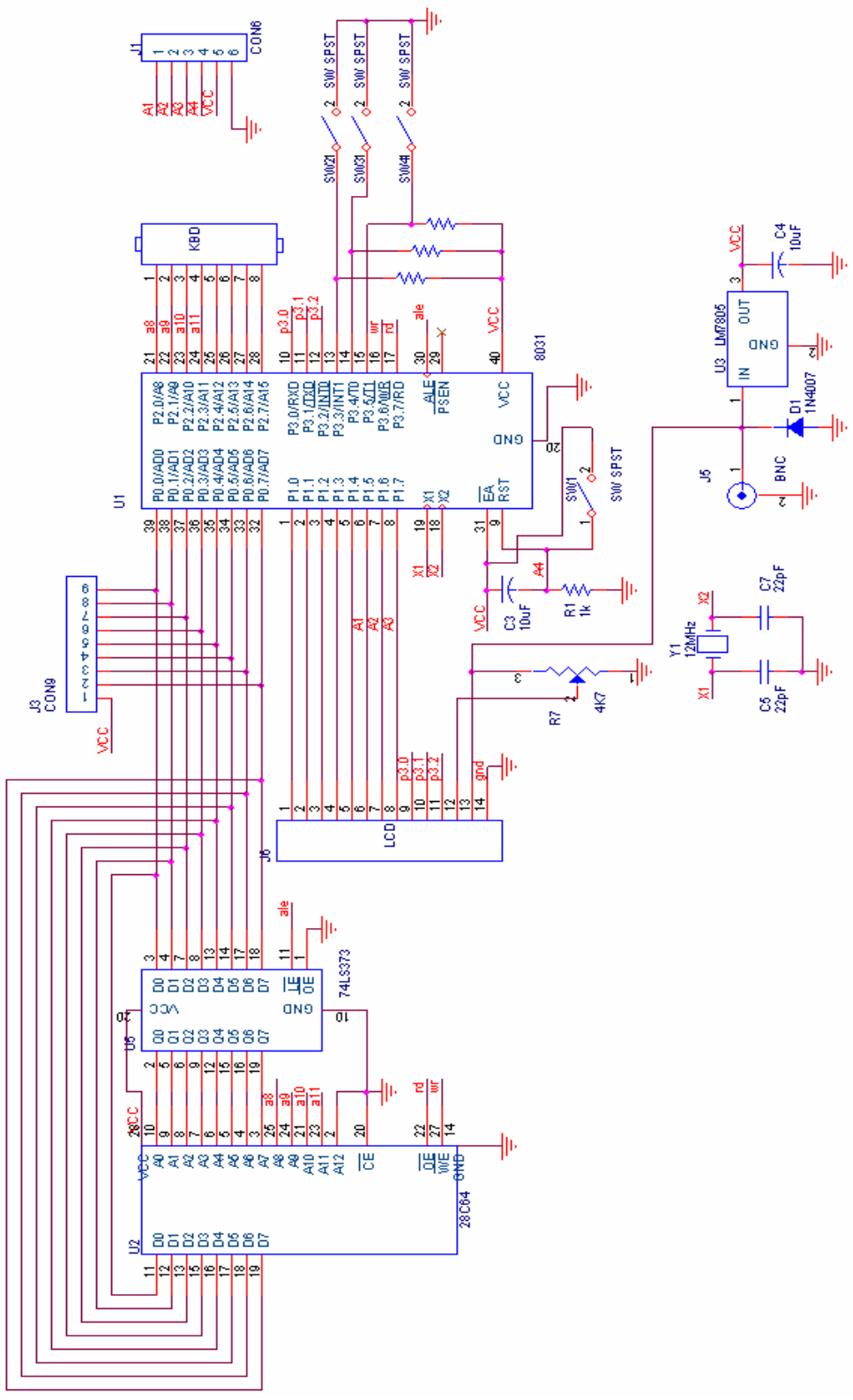


Figure 3.1 PCB Schematic of the Low pass FIR Digital filter

The AT28C64 is a low-power, high-performance 8,192 words by 8-bit nonvolatile electrically erasable and programmable read only memory The AT28C64 is accessed like a Static RAM for the read or write cycles without the need for external components. CMOS

technology offers fast access times of 120 ns at power dissipation. When the chip is deselected the standby current is less than 100 μA. EEPROM is used as data memory in the hardware [14]. It is interfaced to port 0 through a latch. The SN54/74LS373 consists of eight latches with 3-state outputs for bus organized system applications.74LS373 latch is used for the purpose of demultiplexing. SIP is connected on port 0 as pull-up resisters. Pins P0.0 to P0.7 are connected to input pins of the latch. The output pins of the latch are connected to the A0 toA7 address lines of the external memory. Address lines A8 to A11 of latch are connected to P2.0 to P2.3 respectively. A12 is connected to GND to use only 4 kilo bits of memory. CE is an active low pin so it is connected to the GND. OE (Output Enable) pin of the memory is connected to P3.7 (RD) pin of the microcontroller. This control signal is used to read the memory location. Pin 27 of the memory is connected to pin 16(WR) of the microcontroller. This control signal is activated to write data on the external memory. EA pin is connected to V_{cc} because program code is stored in the on-chip ROM of the microcontroller. PSEN pin is left unconnected because external memory is not a code memory. The data pins of the external memory (pin 11 to pin 18) are connected to the data pins of the latch to send or receive data from the microcontroller through latch. ALE pin of the microcontroller is connected to the LE pin of the latch for the purpose of demultiplexing of the address/data bus. When CE and OE are low and WE is high, the data stored at the memory location determined by the address pins is asserted on the outputs. The outputs are put in a high impedance state whenever CE or OE is high. Writing data into the AT28C64 is similar to writing into a Static RAM. A low pulse on the WE or CE input with OE high and CE or WE low (respectively) initiates a byte write. The address location is latched on the falling edge of WE (or CE); the new data is latched on the rising edge [15].

Port1 is connected to the data pins of the LCD. LCDs are used for real time display. It has a 4-bit data bus and operates at +5V power Supply. It has seven data lines plus three control signals. There are three control signals of LCD i.e. Register Select (RS), Read/Write(R/W) and Enable. They are connected to pins P3.0, P3.1 and P3.2 respectively. When RS pin is low, an instruction is being written to the LCD. When it is high, a character is being written to the LCD. When R/W is low, data is written to the LCD. When it is high, data is read from the LCD. When enable is low, LCD is disabled and ignores signals from the R/W and RS. When (E) line is high, the LCD checks the status of the two control lines and responds accordingly. Data pins of the LCD are connected to the 8-bits of the port 1. Pin no.12 of LCD is connected to 4K7 potentiometer. This is used to control the contrast of the LCD. When

voltage is 0V the contrast will be maximum and decreases as voltage increases. Pin no.13 is connected to V_{cc}. Pin no.14 is connected to GND [16].

Figure 3.2 Actual Photograph of Circuit Fabricated

A 4*4 Hex Keypad is interfaced on the 8 pins of port 2 .Out of the 8-pins 4 are used as input and other 4 are used as output pins. P2.0 to P2.3 are act as output port and connected to the columns of hex keypad. Rest of the pins are used as input port and connected to the rows of the keypad.

A PCB switch button is connected across the reset pin and V_{cc} to clear the program counter. In power on reset initially when V_{cc} is applied across the capacitor it acts as a short circuit and whole V_{cc} reaches across the resister and reset the microcontroller. When capacitor is fully charged it acts as an open circuit. Voltage across resistance decreases to zero. A crystal oscillator is connected to pins 18 and 19 of the microcontroller. Three PCB push buttons are connected at P3.3, P3.4 and P3.5. Push button connected at P3.3 is used to see the contents of next memory location. Edit button connected at P3.4 is used to enter data in the memories. Intext switch is used to select one of the memories at a time.

LM7805 regulator is connected to regulate the input power to the microcontroller at 5V. A diode IN4007 is used to protect the circuit from reverse voltage. Capacitor is used filter out the spikes in the output voltage of regulator.

3.2 Software Details of FIR Digital Filter

The present work has been undertaken as, Effect of Finite Word Length on FIR Filter Implemented on 8052 Microcontroller. The software implementation of the FIR filter is divided in two parts:

1) Development of Low pass FIR filter with the help of MATLAB using windowing technique
2) Development of Low pass FIR filter by convolving the rounded-off input sequences and filter coefficients using 8-bit microcontroller.
3) Comparison of the convolution results.

3.2.1 Development of Software using MATLAB

To find out the error introduced by finite word length on FIR filters, a low pass FIR filter is designed with the help of MATLAB to compare with filter implemented by using microcontroller. Blackman window, Hanning window and Rectangular window are used. Figure 3.3 shows the flow chart for development of FIR filter with the help of MATLAB.

Algorithm for Flow chart 3.3 Development of FIR filter with the help of MATLAB

1 Start

2 Take two sine waves of frequency 1400Hz and 1800 Hz respectively.

3 Superimpose the two signals and generate the input sequence by taking samples of input signal at particular instants of time (5000 samples/sec).

4 Convolve the input sequence with the filter coefficients of window.

5 Plot the graph of output result and also plot the result from 8052 to find out the effect of finite word length.

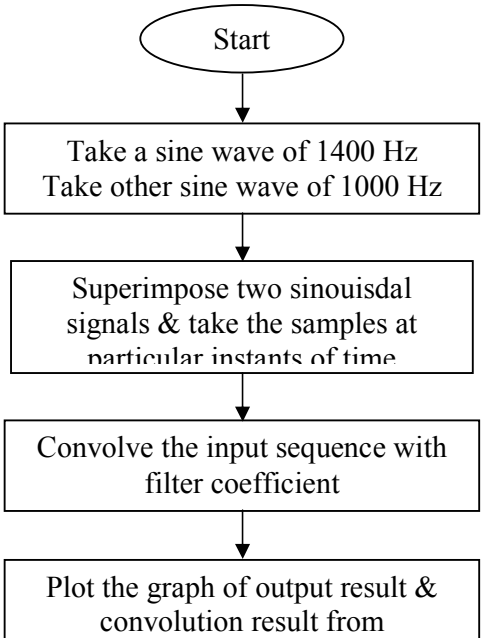
3.2.2 Development of Software using Microcontroller

8052 is an 8-bit microcontroller so the word length of registers is limited to 8-bits.

FIR filter is designed with same specifications as that of above filter. The filter coefficients and the input sequence are either truncated or rounded-off to 8-bit word length. Input sequence is stored in the external memory interfaced to the microcontroller. A hex keypad is used to enter the rounded-off sequence into the memory. Filter coefficients are stored in the internal ROM. The input sequence is convolved with the filter coefficients. The result is stored in the external memory. LCD is used to display the result. Figure 3.4 shows the flow chart of the convolution.

Algorithm for Flow chart 3.4 Convolution

1) Initialize the Data Pointer (DPTR) at 0000H to store the contents of rounded-off input sequence, initialize R0 at 40H to define address of internal RAM and initialize R1 at 8B (length of input sequence).
2) Copy the input sequence from external memory to internal RAM.
3) Initialize R2 at 0AH (length of filter coefficients), Initialize the DPTR at 0100H to store the result at external memory at 0100H.
4) Store the contents of DPTR in Stack Pointer. Initialize DPTR at 00f0H (filter coefficients are stored from this memory location). Move the first input stored at 40H to the R7.
5) Fold the input sequence and multiply it with filter coefficient.
6) If it is negative then subtract it from previous result. Otherwise add it to previous result.
7) If all the filter coefficients are not convolved then increment the go to next filter coefficient and decrement R0 . Repeat from the step 5 again.
8) If all filter coefficients are convolved then check that are all the input sequence is convolved if not store the result in external memory and go to next input and repeat from step 4.

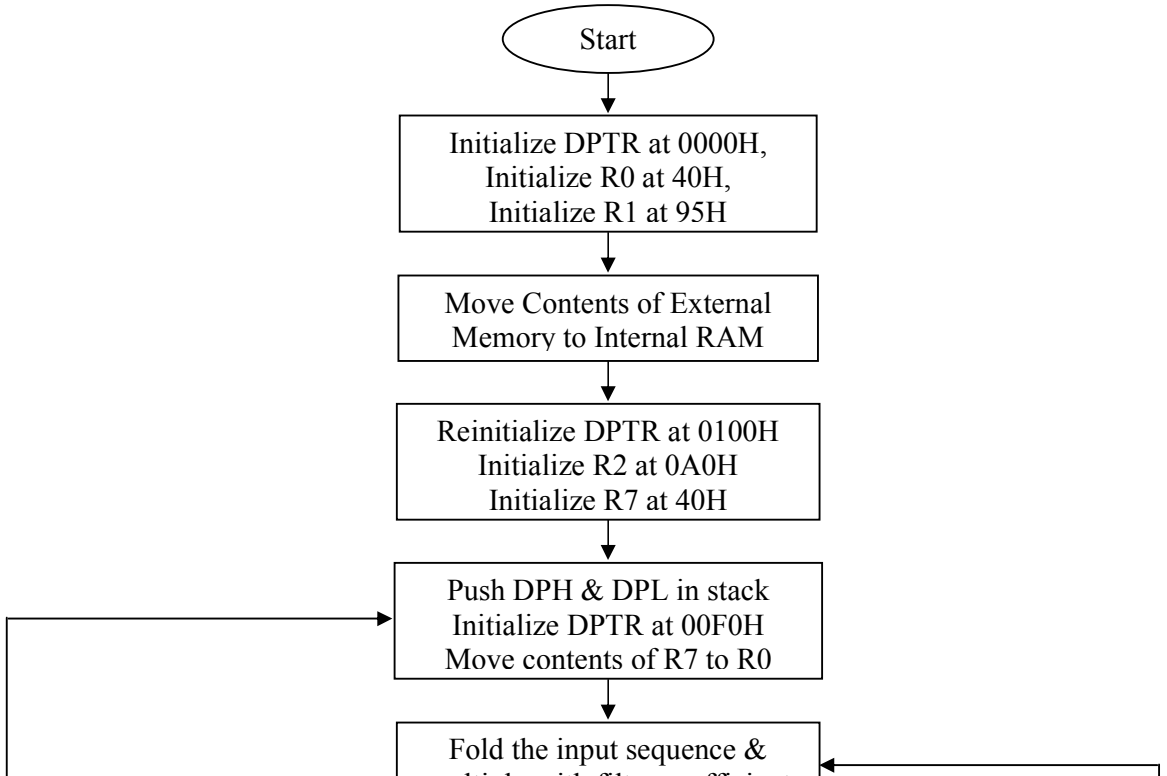


Flow Chart 3.3 Development of FIR Filter with the help of MATLAB

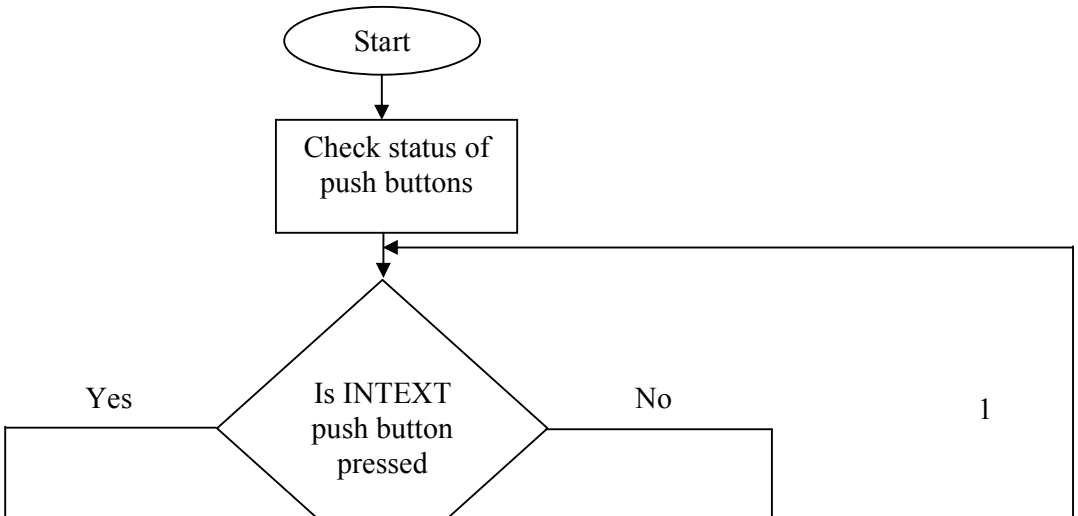
3.2.3 Development of Software for interfacing of 8052 with various Peripherals

Algorithm for Flow Chart 3.5 Interfacing of Microcontroller with External Peripherals

1) Check the status of three push buttons. If Intext button is pressed this mean external memory is selected otherwise internal memory is selected.
2) Check the status of Edit button if it is pressed the contents of the external or internal memory which is selected in step 1 can be edited through 4*4 hex keypad. The edited contents can be seen on the LCD.
3) Check the status of next button if next button is pressed do back to step 1.



Flow chart 3.4 Convolution



Flow Chart 3.5 Interfacing of Microcontroller with External Peripherals

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Designing of Low pass FIR Digital Filter

The input sequence is taken by superimposing the two sinusoidal signals and sampling the resulting signal at particular instants of time (5000 samples/s). One of the sinusoidal signals has a 4000 Hz frequency and other sinusoidal signal has a frequency of 1800Hz. Figures 5.0a and 5.0b show two sinusoidal signals. In figure 5.0c shows the resulting signal after the superimposing of two sin waves [Appendix A].

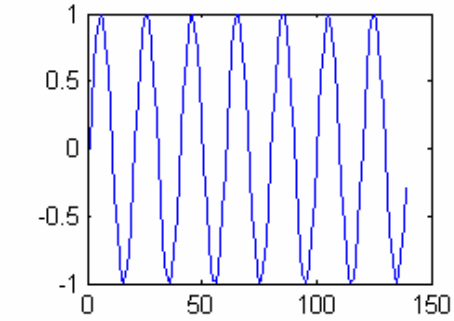


Figure 4.0a Sine Wave

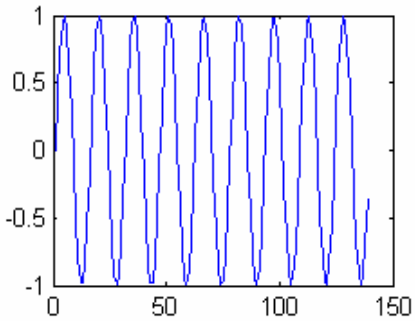


Figure 4.0b Sine Wave

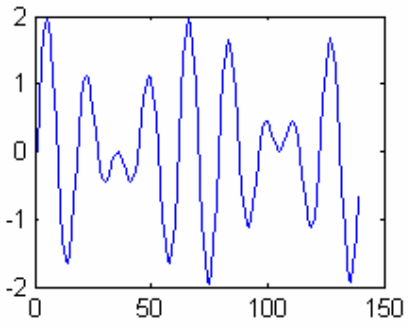


Figure 4.0c Superimposed Wave

4.0 Response of Input Sinusoidal Wave

The input sequence generated with the help of MATLAB is in decimal form. It is converted to hexadecimal form and truncated to 8-bit word length. Decimal point is assumed in between the two nibbles. Response of the rounded-off input sequence in comparison to unrounded-off sequence is shown in Figure 4.1. In case of filter coefficients decimal point is assumed before the 2 nibbles. Rounded-off input sequence is convolved with rounded-off filter coefficients. 3 bytes of result is stored in the external ROM. The decimal point is assumed after the 3 nibbles. The result is displayed on the LCD*.

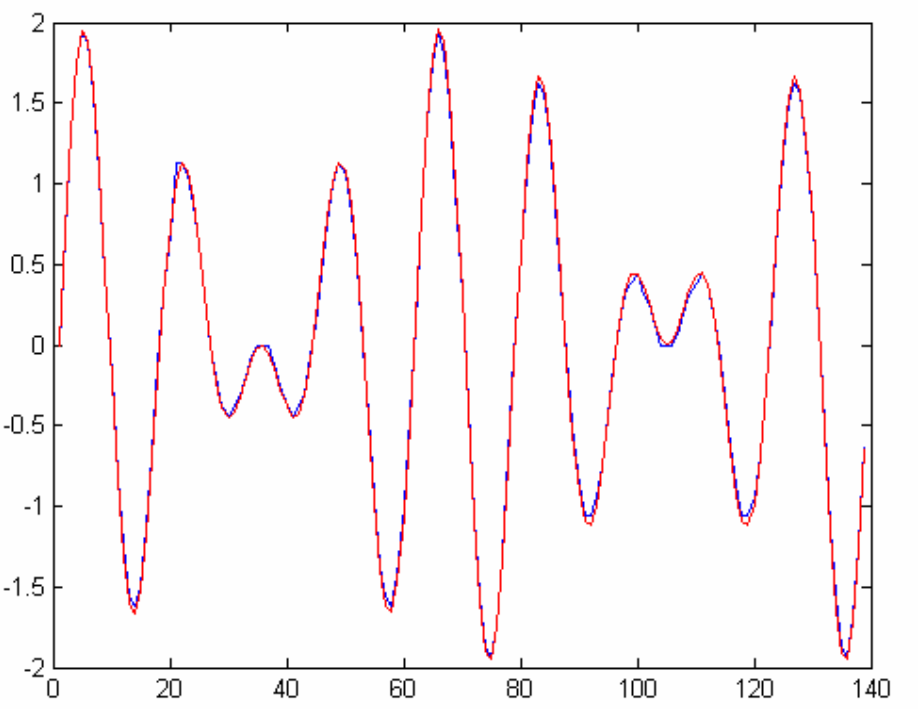


Figure 4.1 Comparison of Rounded-off and Unrounded-off Input Sequence

***** Red colour response is for unrounded-off input sequence.

Blue colour response is for rounded-off input sequence.

4.2 Effect of Rounding-off on FIR filters

4.2.1 Blackman Window

The equation for computing the coefficients of a Blackman Window is:

$$\omega[k+1] = 0.42 - 0.5 \cos \frac{2\pi k}{M-1} + 0.08 \cos \frac{4\pi k}{M-1}, \qquad k= 0,1,2,..,n-1$$

M = length of the filter

The filter coefficients of the Blackman window generated with the help of MATLAB are rounded-off to 8-bits and convolved with the rounded-off input sequence by using 8052 microcontroller. Figure 4.2 shows the comparison of rounded-off and unrounded-off filter coefficients. The convolution result of the microcontroller is compared with result of convolution done with MATLAB is shown in Figure 4.3 [Appendix B].

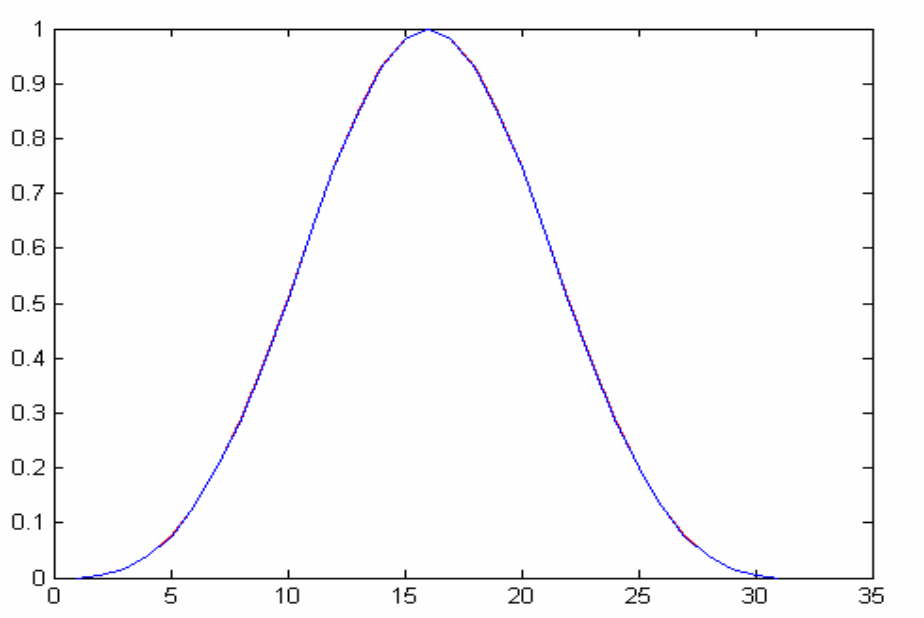


Figure 4.2 Comparison of Rounded-off and Unrounded-off Filter Coefficients

(Blackman Window)

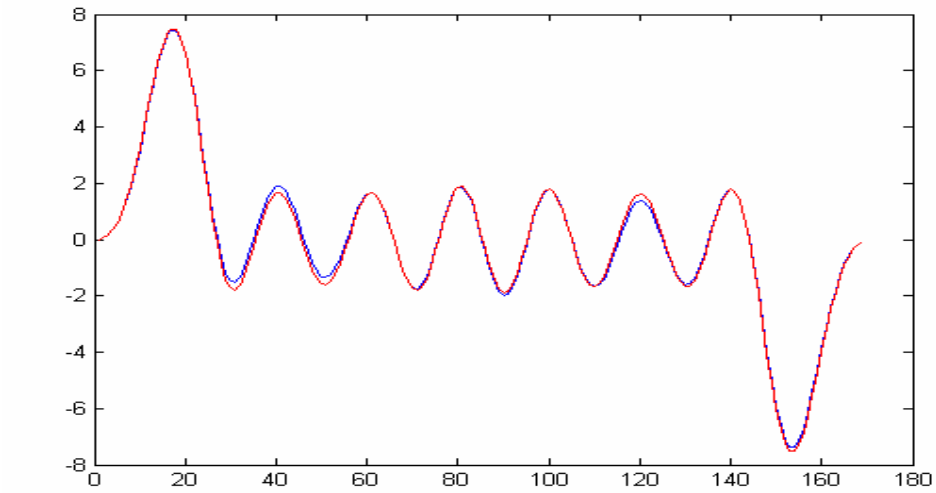


Figure 4.3 Comparison of Convolution results of Microcontroller and MATLAB (Blackman Window)

4.2.2 Hanning Window

The coefficients of a Hanning window are:

$$\omega_{Hann}(n) = 0.5 - 0.5 \cos \frac{2\pi n}{M-1}, \qquad 0 \leq n \leq M-1$$

$$0, \qquad \text{Otherwise}$$

The window function of a non-causal Hanning window is expressed by:

$$\omega_{Hann}(n) = 0.5 + 0.5 \cos \frac{2\pi n}{M-1}, \qquad 0 < |n| < \frac{M-1}{2}$$

$$0, \qquad \text{Otherwise}$$

The filter coefficients of the Hanning window generated with the help of MATLAB are rounded-off to 8-bits and convolved with the rounded-off input sequence by using 8052 microcontroller. Figure 4.4 shows the comparison of rounded-off and unrounded-off filter coefficients. The convolution result of the microcontroller is compared with result of convolution done with MATLAB is shown in Figure 4.5 [Appendix B].

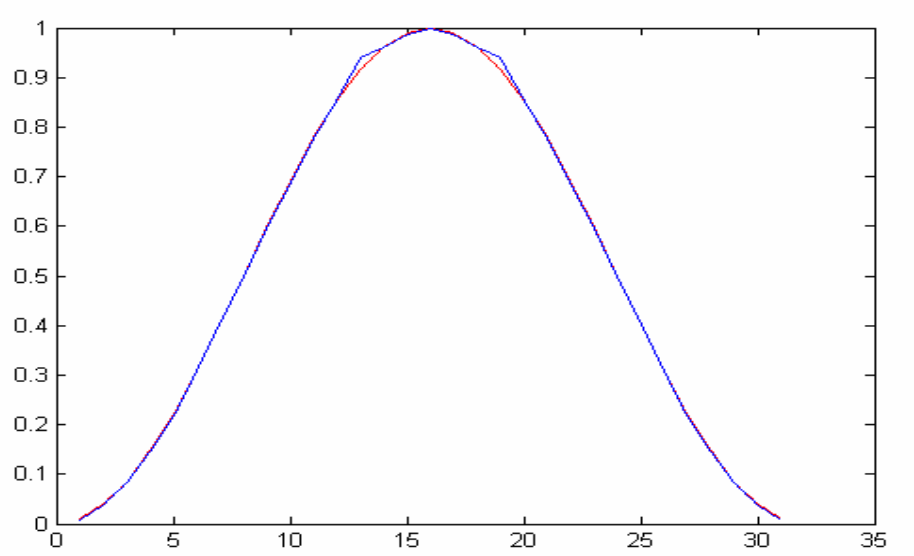


Figure 4.4 Comparison of Rounded-off and Unrounded-off Filter Coefficients of Hanning window

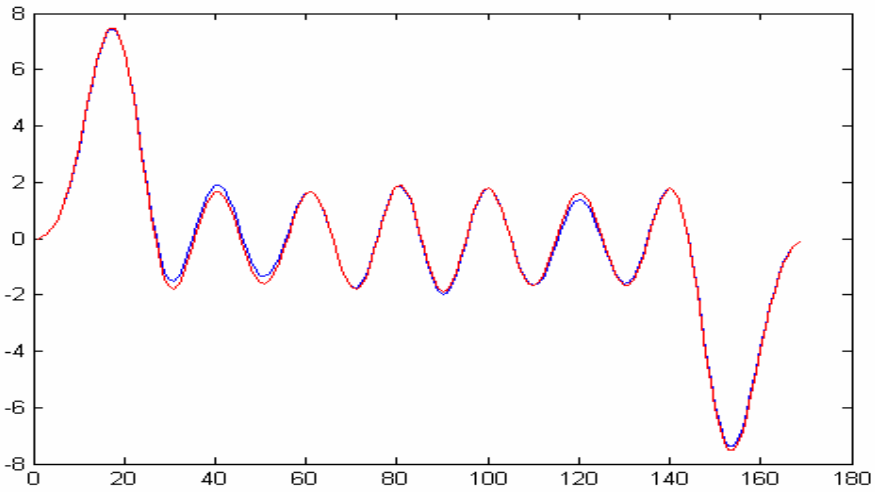


Figure 4.5 Comparison of Convolution results of microcontroller and MATLAB (Hanning Window)

4.2.3 Rectangular window

The weighting function for the rectangular window is given by:

$$\omega_R(n) = \begin{cases} 1, & \text{for } |n| \le \frac{M-1}{2} \\ 0, & \text{otherwise} \end{cases}$$

The filter coefficients of the Rectangular window generated with the help of MATLAB are rounded-off to 8-bits and convolved with the rounded-off input sequence by using 8052 microcontroller. Figure 4.6 shows the comparison of rounded-off and unrounded-off filter coefficients. The convolution result of the microcontroller is compared with result of convolution done with MATLAB is shown in Figure 4.7 [Appendix B].

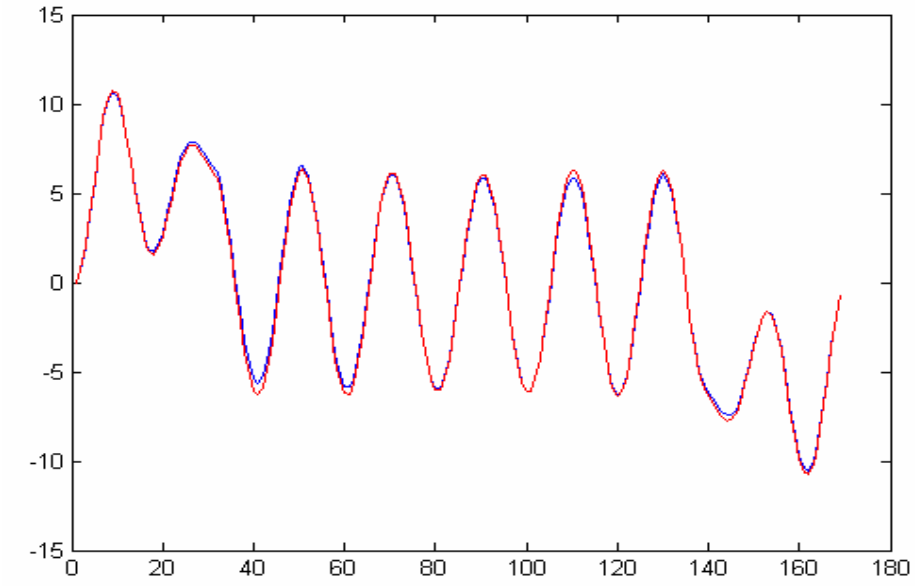


Figure 4.6 Comparison of Convolution results of microcontroller and MATLAB (Rectangular Window)

4.3 Conclusion

In conclusion A FIR filter can be implemented on a microcontroller. Because of the fixed word length of the microcontroller the input sequence and filter coefficients have to be rounded-off which creates deviation in the performance of the FIR filter implemented on microcontroller. When Blackman window is used the error introduced is -11.7%. Deviation from reference response in case of hanning and rectangular windows are

-14.45% and -6.62%. The error is least in case of rectantangular window because only the input sequence is rounded-off.

4.4 Future Scope of the work

Any work, whatsoever precise it may be, has always some scope of improvement. On the same lines, the author envisages that there is a lot of scope of improvement in the present. FIR filter can be implemented on any other microcontroller and by using other windows. The length of filter coefficient can be increased. Real time filter working on particular frequency range i.e. Audio frequency range can be implemented. Instead of convolution Fast Fourier Transform or Discrete Fourier Transform used. Work is done only for fixed-point sequences, the floating-point sequences can be included.

REFERENCES

[1] S. Salivahanan, A. Vallavaraj, C. Gnanaapriya, "Digital Signal Processing", Tata McGraw-Hill, Sixth reprint 2002.

[2] E.C. Ifeachor, B.W.Jervis., "Digital Signal Processing", Pearson Education pte. Ltd.

[3] John G. Poakis, Dimitris G. Manolakis, "Digital Signal Processing (Principals, Algorithms and Application)", Pearson Education, Third Edition.

[4] Craig Steiner, "The 8052 Tutorial & Reference", Vault Information Services LLC, Version 1.00 - 25/Jun/2004, www.8052.com.

[5] Muhammad Ali Mazidi, Janice Gillispie Mazidi, "The 8051 Microcontroller and Embedded Systems", Pearson Education, Thirteenth Indian Reprint 2005, ISBN-81- 7808- 574-7.

[6] www.mathworks.com

[7] Myke Predeko, "Programming and Customizing the 8051 Microcontroller", Tata McGraw-Hill, Edition 1999.

[8] BEDE LIU, "Effect of Finite Word length on the Accuracy of Digital Filter-A Review", IEEE Transactions on Circuit Theory, Vol. cT-18, No. 6, Novamber 1971.

[9] Allan V. Oppenheim and Clifford J. Weinstein, "Effects of finite Register length in Digital Filtering and Fast Fourier Transform", Proceedings of The IEEE, Vol. 60, No. 8, August 1972.

[10] David S. K. Chan and Lawrence R. Rabiner, "Analysis of Quantization Errors in

the Direct Form for Finite Impulse Response Digital Filters", IEEE Transactions on Audio and Electroacoustics, Vol. AU-21, No. 4, August 1973.

[11] R.E.Crochiere, "A New Statistical Approach to the Coefficient Word Length Problem for Digital Filters", IEEE Transaction on Circuits and Systems, Vol. CAS-22, No. 3, March 1975.

[12] Dusan M. Kodek , "Performance Limit of Finite Word length FIR Digital Filters", IEEE Transaction on Signal Processing, Vol. 53, No. 7, July 2005

[13] Data sheet of 8-bit Microcontroller with 8K Bytes In-System Programmable Flash AT89S52 on www.atmel.com

[14] Data Sheet of 64K (8K*8) Parallel EEPROM AT28C64 on www.atmel.com

[15] Data sheet of 74LS373 on www.atmel.com

[16] Data sheet of 44780 LCD.

[17] Data sheet of UM7805 regulator.

[18] Kenneth J. Ayala, "The 8051 Microcontroller Architecture, Programming & Applications" , Penram International, Second Edition ISBN:81-900828-4-1.

[19] Rorabaugh, C. Britton, "Complete Digital Signal Processing", McGraw-Hill Professional, Pub date: 2005-02-01, Copy Right: 2005.

[20] A.V.Opperheim & R.W.Schfer, "Discrete Time Signal Processing", Engle Chiff.,N.J. Prentice –Hall .1989.

.

APPENDIX A

INPUT SEQUENCE

Sr.NO.	INPUT SEQUENCE	QUANTIZED HEX VALUE	Sr.NO.	INPUT SEQUENCE	QUANTIZED HEX VALUE
1	0.7074	0.b	48	1.1255	1.2
2	1.3191	1.5	49	1.0814	1.1
3	1.753	1.C	50	0.8727	0.D
4	1.9524	1.F	51	0.5133	0.8
5	1.8938	1.E	52	0.04	0
6	1.5895	1.9	53	-0.4793	8.7
7	1.086	1.1	54	-0.9791	8.F
8	0.456	0.7	55	-1.3818	9.6
9	-0.2118	8.3	56	-1.6221	9.9
10	-0.8267	8.D	57	-1.6547	9.A
11	-1.308	9.4	58	-1.4626	9.7
12	-1.5969	9.9	59	-1.0612	9
13	-1.6639	9.A	60	-0.4968	8.7
14	-1.5119	9.8	61	0.1584	0.2
15	-1.1743	9.2	62	0.8166	0.D
16	-0.7087	8.B	63	1.3865	1.6
17	-0.1873	8.2	64	1.7868	1.C
18	0.3146	0.5	65	1.9583	1.F
19	0.7298	0.B	66	1.8728	1.D
20	1.008	1.2	67	1.538	1.8
21	1.1261	1.2	68	0.9964	0.F
22	1.0822	1.1	69	0.3202	0.5
23	0.9019	0.E	70	-0.3993	8.6
24	0.628	0.A	71	-1.0648	9.1
25	0.3135	0.5	72	-1.5864	9.9
26	0.0116	0	73	-1.8951	9.E
27	-0.2327	8.3	74	-1.9517	9.F
28	-0.3898	8.6	75	-1.7528	9.C
29	-0.4493	8.7	76	-1.3302	9.5

30	-0.42	8.6	77	-0.7462	8.B
31	-0.326	8.5	78	-0.6838	8.1
32	-0.2047	8.3	79	0.5655	0.9
33	-0.0909	8.1	80	1.115	1.1
34	-0.0169	0	81	1.4949	1.7
35	0.0025	0	82	1.6621	1.A
36	-0.0515	0	83	1.605	1.9
37	-0.151	8.2	84	1.3441	1.5
38	-0.2739	8.4	85	0.927	0.E
39	-0.3845	8.6	86	0.421	0.6
40	-0.446	8.7	87	-0.0986	8.1
41	-0.4282	8.6	88	-0.5596	8.8
42	-0.3144	8.5	89	-0.9038	8.E
43	-0.1063	8.1	90	-1.0945	9.1
44	0.1748	0.2	91	-1.1205	9.1
45	0.49	0.7	92	-0.9965	8.F
46	0.7887	0.C	93	-0.758	8.C
47	1.0167	1	94	-0.896	8.5

Sr. NO.	**INPUT SEQUENCE**	**QUANTIZED HEX VALUE**
95	-0.141	8.2
96	0.1335	0.2
97	0.3319	0.5
98	0.4347	0.6
99	0.4427	0.7
100	0.374	0.5
101	0.2602	0.4
102	0.1382	0.2
103	0.0432	0
104	0.0009	0
105	0.0224	0
106	0.102	0.1
107	0.2186	0.3
108	0.3393	0.5
109	0.4269	0.6
110	0.4474	0.7
111	0.3771	0.6
112	0.2095	0.3
113	0.043	0
114	-0.3487	8.5
115	-0.6615	8.A
116	-0.9275	8.E
117	-1.0945	9.1
118	-1.1212	9.1
119	-0.9855	8.F
120	-0.6896	8.B
121	-0.262	8.4
122	0.2456	0.3
123	0.7646	0.C
124	1.2194	1.3
125	1.5388	1.8
126	1.6676	1.A
127	1.5755	1.9
128	1.2632	1.4

129	0.7637	0.C
130	0.1386	0.2
131	-0.5296	8.8
132	-1.1497	9.2
133	-1.6343	9.9
134	-1.9129	9.E
135	-1.9429	9.F
136	-1.7156	9.B
137	-1.2586	9.4
138	-0.6318	8.A

Table A.1 Input Sequence

APPENDIX B

Sr. NO.	Unqauntized Output Result	Quanztized Output Result	Sr. NO.	Unqauntized Output Result	Quanztized Output Result	Sr. NO.	Unqauntized Output Result	Quanztized Output Result

1	0	0	61	2.9035	2.8833	121	2.1857	1.9097
2	-0.0000	0	62	3.1331	3.0786	122	2.0088	1.8176
3	0.0028	0.0027	63	3.0619	2.9653	112	1.6980	1.5786
4	0.0171	0.0159	64	2.6610	2.5225	124	1.2548	1.1987
5	0.0576	0.0300	65	1.9380	1.7690	125	0.6876	0.6707
6	0.1448	0.0911	66	0.9424	0.7649	126	0.0173	0.0313
7	0.3044	0.2305	67	-0.2365	-0.3997	127	-0.7192	-0.6890
8	0.5652	0.4790	68	-1.4763	-1.5896	128	-1.4682	-1.4297
9	0.9551	0.8655	69	-2.6348	-2.6797	129	-2.1608	-2.1040
10	1.4963	1.4099	70	-3.5672	-3.5198	130	-2.7191	-2.6436
11	2.1990	2.1216	71	-4.1453	-3.9917	131	-3.0653	-2.9631
12	3.0561	2.9920	72	-4.2761	-4.0198	132	-3.1333	-2.9956
13	4.0379	3.9883	73	-3.9167	-3.5747	133	-2.8803	-2.7080
14	5.0905	5.0559	74	-3.0837	-2.6963	134	-2.2969	-2.0967
15	6.1368	6.1126	75	-1.8545	-1.4705	135	-1.4137	-1.1980
16	7.0819	7.0652	76	-0.3605	-0.0281	136	-0.3029	-0.0933
17	7.8223	7.8084	77	1.2274	1.4580	137	0.9266	1.1001
18	8.2584	8.2440	78	2.7194	2.8191	138	2.1394	2.2590
19	8.3079	8.2896	79	3.9311	3.8774	139	3.1895	3.2292
20	7.9193	7.9000	80	4.7075	4.4917	140	3.9391	3.8838
21	7.0823	7.0643	81	4.9441	4.5928	141	4.2769	4.1294
22	5.8338	5.8270	82	4.6019	4.1506	142	4.1328	3.9011
23	4.2579	4.2694	83	3.7137	3.2202	143	3.4870	3.2080
24	2.4790	2.5086	84	2.3808	1.9133	144	2.3745	2.1082
25	0.6492	0.7152	85	0.7615	0.3762	145	0.8830	0.6680
26	-1.0690	-0.9025	86	-0.9497	-1.2083	146	-0.8582	-0.9802
27	-2.5220	-2.2097	87	-2.5464	-2.6409	147	-2.6937	-2.6992
28	-3.5846	-3.1130	88	-3.8367	-3.7593	148	-4.4612	-4.3272
29	-4.1745	-3.5627	89	-4.6681	-4.4329	149	-6.0104	-5.7358
30	-4.2625	-3.5781	90	-4.9463	-4.5874	150	-7.2208	-6.8259
31	-3.8744	-3.1801	91	-4.6465	-4.2305	151	-8.0130	-7.5308
32	-3.0853	-2.4416	92	-3.8155	-3.4028	152	-8.3551	-7.8357
33	-2.0100	-1.4744	93	-2.5643	-2.2209	153	-8.2619	-7.7590
34	-0.7882	-0.3994	94	-1.0522	-0.8333	154	-7.7890	-7.3586
35	0.4344	0.6580	95	0.5345	0.5925	155	-7.0208	-6.7104
36	1.5247	1.5894	96	2.0070	1.8835	156	-6.0584	-5.8987
37	2.3774	2.3054	97	3.1978	2.9077	157	-5.0047	-5.0090
38	2.9249	2.7595	98	3.9809	3.5537	158	-3.9532	-4.1106
39	3.1411	2.9353	99	4.2855	3.7761	159	-2.9791	-3.2654
40	3.0395	2.8499	100	4.1018	3.5786	160	-2.1339	-2.5090
41	2.6661	2.5454	101	3.4790	3.0110	161	-1.4449	-1.8486
42	2.0890	2.0774	102	2.5146	2.1694	162	-0.9173	-1.2927
43	1.3864	1.4998	103	1.3399	1.1555	163	-0.5394	-0.8271
44	0.6356	0.8711	104	0.1000	0.1008	164	-0.2884	-0.4539
45	-0.0959	0.2612	105	-1.0640	-0.8894	165	-0.1358	-0.2175
46	-0.7562	-0.3186	106	-2.0331	-1.7102	166	-0.0534	-0.0852
47	-1.3104	-0.8318	107	-2.7219	-2.2915	167	-0.0156	-0.0286
48	-1.7392	-1.2573	108	-3.0859	-2.6118	168	-0.0025	-0.0146
49	-2.0351	-1.5940	109	-3.1216	-2.6648	169	0.0000	-0.0024
50	-2.1971	-1.8162	110	-2.8617	-2.4868			
51	-2.2265	-1.9260	111	-2.3658	-2.1223			
52	-2.1235	-1.9021	112	-1.7091	-1.6191			
53	-1.8873	-1.7402	113	-0.9707	-1.0425			
54	-1.5171	-1.4280	114	-0.2231	-0.4509			
55	-1.0171	-0.9668	115	0.4741	0.1196			
56	-0.4006	-0.3740	116	1.0787	0.6389			
57	0.3047	0.3210	117	1.5648	1.0852			
58	1.0544	1.0696	118	1.9201	1.4458			
59	1.7875	1.7996	119	2.1415	1.7112			
60	2.4302	2.4287	120	2.2297	1.8694			

Table B.1 Results of Convolution of Unquantized and Quantized Sequences For Blackman window

Sr. NO.	**Unqauntized Output Result**	**Quanztized Output Result**	**Sr. NO.**	**Unqauntized Output Result**	**Quanztized Output Result**	**Sr. NO.**	**Unqauntized Output Result**	**Quanztized Output Result**

1	0	0	61	1.6711	1.8982	121	1.5541	1.7135
2	0.0566	0.0054	62	1.5811	1.6459	122	1.3588	1.4564
3	0.1692	0.0344	63	1.3350	1.2275	112	1.0355	1.0328
4	0.3438	0.1162	64	0.9534	0.7004	124	0.6132	0.4867
5	0.5908	0.2837	65	0.4707	0.1159	125	0.1300	-0.1234
6	0.9230	0.5664	66	-0.0676	-0.4658	126	-0.3699	-0.7205
7	1.3514	0.9893	67	-0.6089	-0.9808	127	-0.8401	-1.2468
8	1.8820	1.5615	68	-1.0990	-1.3971	128	-1.2359	-1.6459
9	2.5121	2.2756	69	-1.4869	-1.6516	129	-1.5186	-1.8644
10	3.2283	3.1035	70	-1.7308	-1.7389	130	-1.6591	-1.8941
11	4.0053	4.0034	71	-1.8027	-1.6526	131	-1.6410	-1.7226
12	4.8063	4.9231	72	-1.6915	-1.3967	132	-1.4627	-1.3725
13	5.5859	5.7991	73	-1.4050	-1.0186	133	-1.1378	-0.8915
14	6.2928	6.5855	74	-0.9695	-0.5508	134	-0.6950	-0.3193
15	6.8751	7.2137	75	-0.4278	-0.0373	135	-0.1752	0.2759
16	7.2850	7.5842	76	0.1652	0.4696	136	0.3715	0.8356
17	7.4837	7.7008	77	0.7481	0.9174	137	0.8909	1.3103
18	7.4456	7.5192	78	1.2595	1.2785	138	1.3300	1.6370
19	7.1611	7.0992	79	1.6446	1.5076	139	1.6423	1.8061
20	6.6381	6.4792	80	1.8613	1.5959	140	1.7867	1.7918
21	5.9016	5.6838	81	1.8850	1.5431	141	1.6937	1.6020
22	4.9919	4.7635	82	1.7117	1.3439	142	1.3625	1.2380
23	3.9612	3.7707	83	1.3586	1.0293	143	0.8049	0.7066
24	2.8697	2.7582	84	0.8621	0.6132	144	0.0459	0.0211
25	1.7806	1.7823	85	0.2744	0.1432	145	-0.8770	-0.7970
26	0.7559	0.8811	86	-0.3423	-0.3436	146	-1.9150	-1.7251
27	-0.1481	0.0996	87	-0.9228	-0.8082	147	-3.0109	-2.7251
28	-0.8839	-0.5476	88	-1.4057	-1.2029	148	-4.1022	-3.7598
29	-1.4154	-1.0353	89	-1.7404	-1.5014	149	-5.1256	-4.7602
30	-1.7193	-1.3584	90	-1.8922	-1.6703	150	-6.0215	-5.6837
31	-1.7851	-1.5056	91	-1.8464	-1.6967	151	-6.7386	-6.4742
32	-1.6151	-1.4857	92	-1.6095	-1.5762	152	-7.2382	-7.0820
33	-1.2867	-1.2767	93	-1.2080	-1.3107	153	-7.4968	-7.4915
34	-0.8362	-0.9162	94	-0.6858	-0.9242	154	-7.5086	-7.6524
35	-0.3111	-0.4342	95	-0.0986	-0.4401	155	-7.2850	-7.5180
36	0.2353	0.1448	96	0.4916	0.0876	156	-6.8532	-7.1299
37	0.7488	0.7311	97	1.0240	0.6133	157	-6.2533	-6.4920
38	1.1804	1.2982	98	1.4448	1.0876	158	-5.5337	-5.7014
39	1.4901	1.7621	99	1.7128	1.4612	159	-4.7469	-4.8245
40	1.6511	2.0744	100	1.8037	1.7036	160	-3.9434	-3.9089
41	1.6512	2.2031	101	1.7118	1.7784	161	-3.1684	-3.0178
42	1.4938	2.1261	102	1.4502	1.6704	162	-2.4573	-2.2026
43	1.1969	1.8583	103	1.0487	1.3913	163	-1.8343	-1.5039
44	0.7907	1.4116	104	0.5503	0.9615	164	-1.3118	-0.9473
45	0.3150	0.8402	105	0.0066	0.4255	165	-0.8914	-0.5386
46	-0.1854	0.2032	106	-0.5280	-0.1667	166	-0.5667	-0.2673
47	-0.6639	-0.4408	107	-1.0015	-0.7570	167	-0.3263	-0.1084
48	-1.0769	-1.0123	108	-1.3695	-1.2807	168	-0.1576	-0.0317
49	-1.3872	-1.4525	109	-1.5990	-1.6842	169	-0.0505	-0.0049
50	-1.5671	-1.7179	110	-1.6711	-1.9184			
51	-1.6005	-1.7787	111	-1.5824	-1.9644			
52	-1.4845	-1.6246	112	-1.3444	-1.8095			
53	-1.2294	-1.2838	113	-0.9824	-1.4680			
54	-0.8579	-0.7848	114	-0.5322	-0.9827			
55	-0.4034	-0.1937	115	-0.0371	-0.3924			
56	0.0929	0.4249	116	0.4568	0.2194			
57	0.5853	1.0076	117	0.9040	0.8075			
58	1.0277	1.4809	118	1.2640	1.2931			
59	1.3775	1.8163	119	1.5046	1.6239			
60	1.5999	1.9546	120	1.6042	1.7740			

Table B.2 Results of Convolution of Rounded-off and Unrounded-off Sequences for Hanning Window

Sr. NO.	Unqauntized Output Result	Quanztized Output Result	Sr. NO.	Unqauntized Output Result	Quanztized Output Result	Sr. NO.	Unqauntized Output Result	Quanztized Output Result

1	0	0	61	-5.8750	-6.2498	121	-6.0625	3.1310
2	0.6875	0.7074	62	-5.3750	-5.6714	122	-5.2500	4.6797
3	2.0000	2.0265	63	-4.2500	-4.5280	112	-4.0000	5.7652
4	3.7500	3.7795	64	-2.6875	-2.9368	124	-2.3125	6.2759
5	5.6875	5.7320	65	-0.8750	-1.0591	125	-0.3750	6.1578
6	7.5625	7.6257	66	1.0625	0.9161	126	1.5625	5.4200
7	9.1251	9.2152	67	2.8750	2.7915	127	3.3125	4.1345
8	10.1876	10.3012	68	4.3750	4.3810	128	4.7500	2.4294
9	10.6251	10.7573	69	5.4375	5.5284	129	5.6875	0.4757
10	10.4376	10.5454	70	6.0000	6.1226	130	6.0625	-1.5299
11	9.6251	9.7187	71	6.0000	6.1078	131	5.7500	-3.3844
12	8.3751	8.4107	72	5.5625	5.4891	132	4.9375	-4.8999
13	6.8126	6.8138	73	4.1875	4.3308	133	3.5625	-5.9225
14	5.1876	5.1499	74	2.6250	2.7501	134	1.8750	-6.3484
15	3.6876	3.6379	75	0.7500	0.9046	135	0	-6.1342
16	2.5626	2.4636	76	-1.1250	-1.0230	136	-1.9375	-5.3018
17	1.8751	1.7549	77	-2.8750	-2.8432	137	-3.6250	-3.9357
18	1.7501	1.5676	78	-4.3125	-4.3781	138	-4.9375	-2.1746
19	2.0626	1.8822	79	-5.3750	-5.4785	139	-5.7500	-0.1972
20	2.7501	2.6120	80	-5.9375	-6.0385	140	-6.0625	1.7964
21	3.8751	3.6207	81	-5.9375	-6.0050	141	-6.4375	3.6051
22	5.0001	4.7468	82	-5.3125	0.9046	142	-6.8750	5.0471
23	6.0626	5.8290	83	-4.1875	-1.0230	143	-7.2500	5.9784
24	6.9376	6.7309	84	-2.6250	-2.8432	144	-7.4375	6.3073
25	7.5626	7.3590	85	-0.8750	-4.3781	145	-7.4375	6.0033
26	7.8751	7.6725	86	0.9375	-5.4785	146	-7.1250	5.0996
27	7.8751	7.6840	87	2.6875	-6.0385	147	-6.5000	3.6898
28	7.6876	7.4513	88	4.1875	-6.0050	148	-5.6250	1.9173
29	7.3126	7.0615	89	5.3125	0.9046	149	-4.5625	-0.0389
30	6.8751	6.6123	90	5.8750	-1.0230	150	-3.5000	-1.9827
31	6.5001	6.1923	91	5.8125	-2.8432	151	-2.5625	-3.7207
32	6.1876	5.8655	92	5.1875	-4.3781	152	-1.8750	-5.0813
33	5.3126	4.9533	93	4.1250	-5.4785	153	-1.6250	-5.9317
34	3.9376	3.5434	94	2.5625	-6.0385	154	-1.8125	-6.2711
35	2.1876	1.7735	95	0.7500	-6.0050	155	-2.5625	-6.6980
36	0.2501	-0.1815	96	-1.1250	-5.3828	156	-3.7500	-7.1454
37	-1.6249	-2.1268	97	-2.9375	-4.2339	157	-5.2500	-7.5225
38	-3.3125	-3.8673	98	-4.4375	-2.6714	158	-6.8750	-7.7320
39	-4.6250	-5.2272	99	-5.5625	-0.8480	159	-8.4375	-7.6890
40	-5.4375	-6.0678	100	-6.0625	1.0581	160	-9.6875	-7.3402
41	-5.6875	-6.3020	101	-6.0625	2.8610	161	-10.4375	-6.6788
42	-5.2500	-5.9035	102	-5.4375	4.3845	162	-10.5625	-5.7513
43	-4.3125	-4.9098	103	-4.2500	5.4795	163	-10.0625	-4.6568
44	-2.8125	-3.4192	104	-2.6875	6.0384	164	-8.9375	-3.5356
45	-1.0625	-1.5805	105	-0.8125	6.0051	165	-7.3750	-2.5501
46	0.8750	0.4215	106	1.1250	5.3814	166	-5.5000	-1.8605
47	2.7500	2.3845	107	2.9375	4.2266	167	-3.5625	-1.5985
48	4.4375	4.1099	108	4.4375	2.6519	168	-1.8750	-1.8441
49	5.6875	5.4227	109	5.4375	0.8107	169	-0.6250	-2.6086
50	6.4375	6.1895	110	5.8750	-1.1172			
51	6.5625	6.3325	111	5.7500	-2.9420			
52	5.9375	5.8370	112	5.0625	-4.4829			
53	4.8125	4.7534	113	3.8125	-5.5861			
54	3.3125	3.1918	114	2.1875	-6.1399			
55	1.5000	1.3108	115	0.3125	-6.0861			
56	-0.5000	-0.6990	116	-1.6250	-5.4267			
57	-2.3750	-2.6346	117	-3.3750	-4.2237			
58	-4.0000	-4.3009	118	-4.8125	-2.5940			
59	-5.2500	-5.5308	119	-5.8125	-0.6980			
60	-5.8750	-6.2022	120	-6.2500	1.2761			

Table B.3 Results of Convolution of Rounded-off and Unrounded-off Sequences for Rectangular Window

APPENDIX C

Code for Convolution

```
        org 0000h
        mov r0,#40h
        mov dptr,#0000h
        mov r1,#95h
jmp: movx a,@dptr
        mov @r0,a
        inc r0
        inc dptr
        djnz r1,jmp
        mov dptr,#0100h
        mov sp,#0010h
        mov r2,#0a0h
        mov r7,#40h
conv:acall muladd
         mov a,r1
         movx @dptr,a
         mov a,r4
         inc dptr
         movx @dptr,a
         mov a,r3
         inc dptr
         movx @dptr,a
         inc dptr
         inc r7
         djnz r2,conv
         ret
jmp1:mov a,#00h
         mov r6,#00h
         movc a,@a+dptr
         jz next3
         mov b,a
         mov a,@r0
         mov c,acc.7
         mov 00h,c
         clr acc.7
         mul ab
         mov r6,a
         mov c,00h
         jnc next3
         clr c
         mov a,r3
         subb a,r6
         mov r3,a
         mov a,r4
         subb a,b
         mov r4,a
         jnc nc
         dec r1
         sjmp nc
next3:add a,r3
         mov r3,a
```

```
        mov a,b
        addc a,r4
        mov r4,a
        jnc nc
        inc r1
  nc:   inc dptr
        dec r0
        djnz r5,jmp1
        pop dpl
        pop dph
        ret
 wait:   mov r1,#0ffh
  w3:     mov r2,#0ffh
  w2:     djnz r2,w2
          djnz r1,w3
           ret
        org 00f0h
        db 1h,1h,1h,1h,1h,1h,1h,1h,1h,1h,1h,1h,1h,1h,1h,1h,
        db 1h,1h,1h,1h,1h,1h,1h,1h,1h,1h,1h,1h,1h,1h,1h,1h,1h
        end
```