PDF-XCHANGE
Click to buy NOW!
www.docu-track.com

İstanbul Üniversitesi Elektronik Mühendisliği

Graduation Project

Microcontroller Controlled Spectrum Analyzer

Prepered By Adem Burak ÇAKMAK

Supervisor:Doç. Dr. Aydın AKAN

June 2005

PDF-XCHANGE Click to buy NOW! www.docu-track.com

PREFACE

In this study desing of a Spectrum Analyzer is given as a graduation project.This project desires to be helpful for the electronic engineering students who wants to design a spectrum analyzer from the begining or who wants to make fourier analysis from the screen of a fabricated analyzer.All of the design phase of the analyzer is presented.

I would like to thank my lecturer Associated Professor Dr. Aydın AKAN for his helps in the theoritical part of the project. Also I would like to thank my research asistant Erol Öner for his helps in the design of the project.

CONTENTS

PDF-XCHANGE Click to buy NOW! www.docu-track.com

SUMMARY

This study is prepared as a graduation project.In the first part it contains general detailed information about DFT,FFT and analysis of signals in the frequency domain,general information about hardware that are used in the project.In the second part the progress phase of consepting a new Spectrum Analyzer. Chapters and their subtitles are:Theoritical part cotains introduction,information about DFT, FFT,spectrum analyzing in the frequency domain information about PIC microcontroller,general information about spectrum analyzers, and information about LCD that is.Application part contains hardware of the project,software of the project.

LIST OF FIGURES

Page

LIST OF TABLES

1.INTRODUCTION

1.1 Information about DFT

1.1.1 Introduction To DFT

This document describes the Discrete Fourier Transform (DFT), that is, a Fourier Transform as applied to a discrete complex valued series.

Theory

Continuous

For a continuous function of one variable f(t), the Fourier Transform F(f) will be defined as:

$$F(f) = \int_{-\infty}^{\infty} f(t)\ e^{-j2\pi ft}\ dt$$

and the inverse transform as

$$f(t) = \int_{-\infty}^{\infty} F(f)\ e^{j2\pi ft}\ df$$

where j is the square root of -1 and e denotes the natural exponent

$$e^{j\varnothing} = \cos(\varnothing) + j\ \sin(\varnothing).$$

PDF-XCHANGE Click to buy NOW! www.docu-track.com

Discrete

Consider a complex series x(k) with N samples of the form

$$x_0, x_1, x_2, x_3 \ldots x_k \ldots x_{N-1}$$

where x is a complex number

$$x_i = x_{real} + j\, x_{imag}$$

Further, assume that that the series outside the range 0, N-1 is extended N-periodic, that is, $x_k = x_{k+N}$ for all k. The FT of this series will be denoted X(k), it will also have N samples. The forward transform will be defined as

$$X(n) = \frac{1}{N} \sum_{k=0}^{N-1} x(k)\ e^{-jk2\pi n/N} \quad \text{for } n=0..N-1$$

The inverse transform will be defined as

$$x(n) = \sum_{k=0}^{N-1} X(k)\ e^{jk2\pi n/N} \quad \text{for } n=0..N-1$$

Of course although the functions here are described as complex series, real valued series can be represented by setting the imaginary part to 0. In general, the transform into the frequency domain will be a complex valued function, that is, with magnitude and phase.

$$\text{magnitude} = ||X(n)|| = (\ x_{real}*x_{real} + x_{imag}*x_{imag}\ )^{0.5}$$
$$\text{phase} = \tan^{-1}\left(\frac{x_{imag}}{x_{real}}\right)$$

The following diagrams show the relationship between the series index and the frequency domain sample index. Note the functions here are only diagramatic, in general they are both complex valued series.

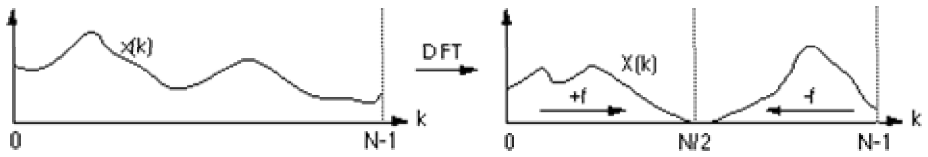


Figure 1.1

For example if the series represents a time sequence of length T then the following illustrates the values in the frequency domain.

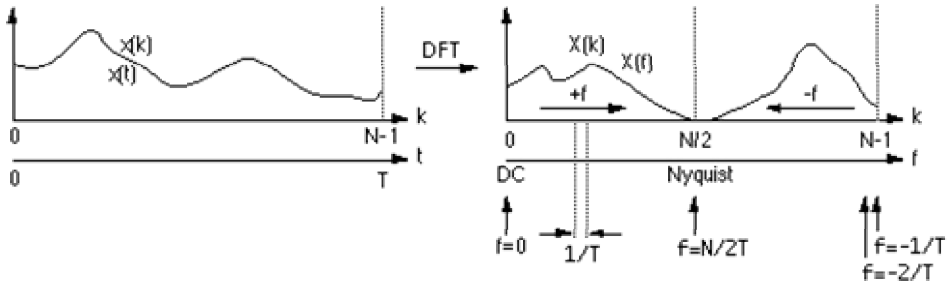


Figure 1.2

Notes

- The first sample X(0) of the transformed series is the DC component, more commonly known as the average of the input series.
- The DFT of a real series, ie: imaginary part of x(k) = 0, results in a symmetric series about the Nyquist frequency. The negative frequency samples are also the inverse of the positive frequency samples.
- The highest positive (or negative) frequency sample is called the Nyquist frequency. This is the highest frequency component that should exist in the input series for the DFT to yield "uncorrupted" results. More specifically if there are no frequencies above Nyquist the original signal can be **exactly** reconstructed from the samples.
- The relationship between the harmonics returns by the DFT and the periodic component in the time domain is illustrated below.

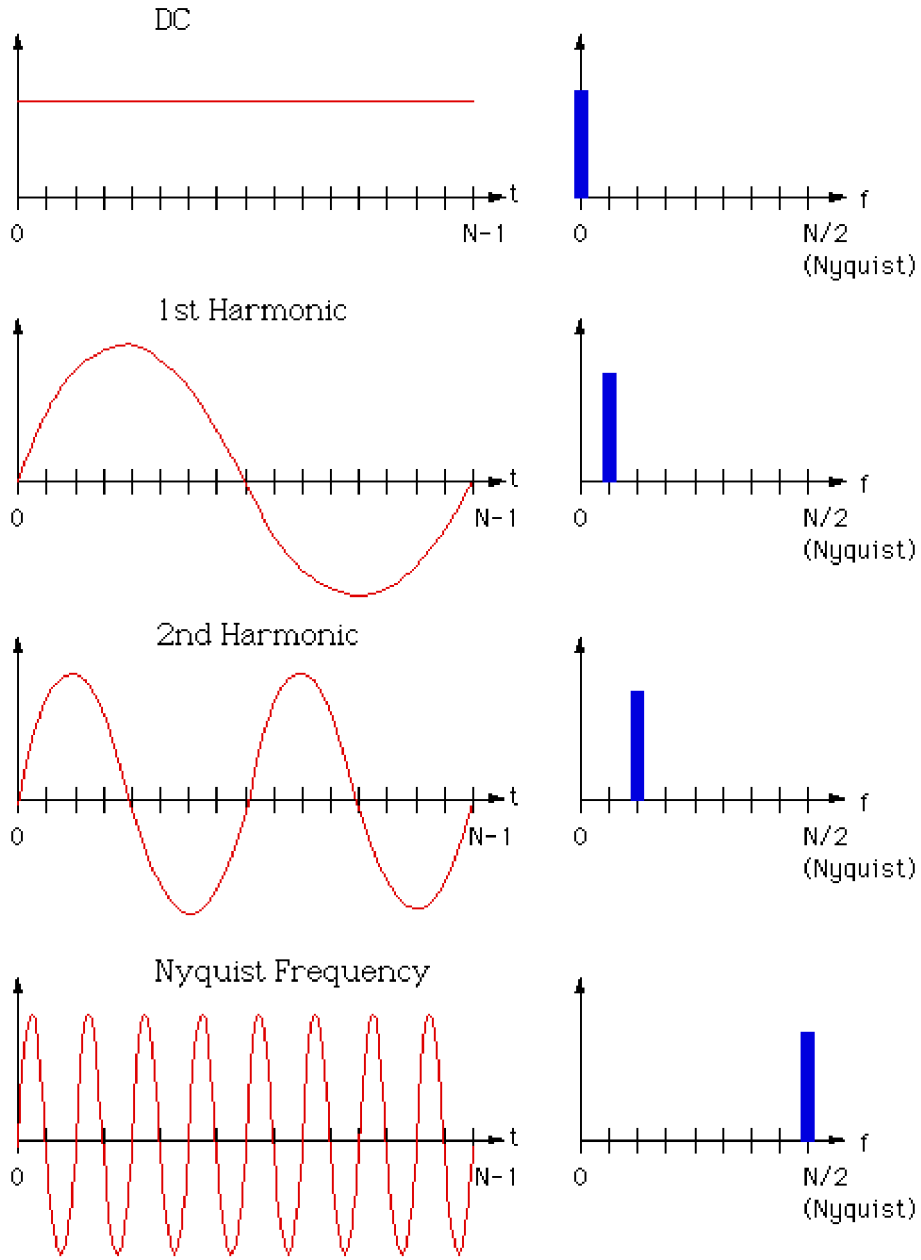


Figure1.3

1.1.2 DFT and FFT algorithm.

While the DFT transform above can be applied to any complex valued series, in practice for large series it can take considerable time to compute, the time taken being proportional to the square of the number on points in the series. A much faster algorithm has been developed by Cooley and Tukey around 1965 called the FFT (Fast Fourier Transform). The only requirement of the the most popular

PDF-XCHANGE Click to buy NOW! www.docu-track.com

implementation of this algorithm (Radix-2 Cooley-Tukey) is that the number of points in the series be a power of 2. The computing time for the radix-2 FFT is proportional to

$$N \log_2(N)$$

So for example a transform on 1024 points using the DFT takes 10 times longer than using the FFT, a significant speed increase. Note that in reality comparing speeds of various FFT routines is problematic, many of the reported timings have more to do with specific coding methods and their relationship to the hardware and operating system.

1.1.3 Sample transform pairs and relationships

- The Fourier transform is linear, that is

$$a\ f(t) + b\ g(t) \text{ ---> } a\ F(f) + b\ G(f)$$

$$a\ x_k + b\ y_k \text{ ---> } a\ X_k + b\ Y_k$$

- Scaling relationship

$$f(t / a) \text{ ---> } a\ F(a\ f)$$

$$f(a\ t) \text{ ---> } F(f / a) / a$$

- Shifting

$$f(t + a) \text{ ---> } F(f)\ e^{-j\ 2\ pi\ a\ f}$$

- Modulation

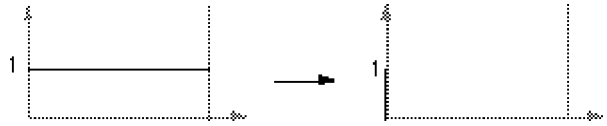
$$f(t)\ e^{j\ 2\ pi\ a\ t} \text{ ---> } F(t - a)$$

- Duality

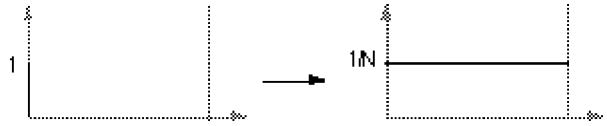
$$X_k \text{ ---> } (1/N)\ x_{N-k}$$

Applying the DFT twice results in a scaled, time reversed version of the original series.

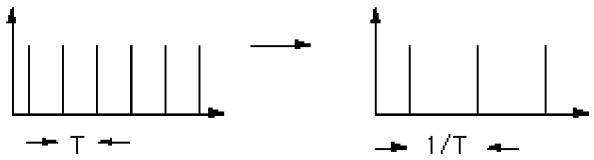
- The transform of a constant function is a DC value only.



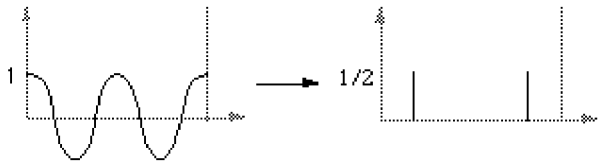
- The transform of a delta function is a constant



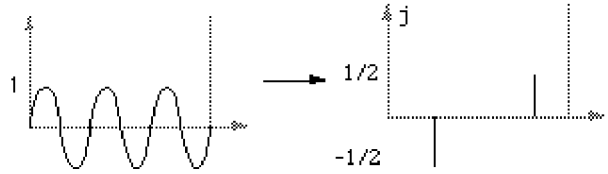
- The transform of an infinite train of delta functions spaced by T is an infinite train of delta functions spaced by 1/T.



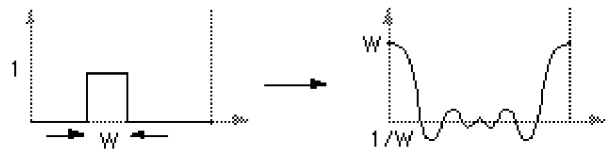
- The transform of a cos function is a positive delta at the appropriate positive and negative frequency.



- The transform of a sin function is a negative complex delta function at the appropriate positive frequency and a negative complex delta at the appropriate negative frequency.



- The transform of a square pulse is a sinc function



More precisely, if f(t) = 1 for |t| < 0.5, and f(t) = 0 otherwise then F(f) = sin(pi f) / (pi f)

- Convolution

$$f(t) \mathbf{x} g(t) \text{ ---> } F(f)\, G(f)$$

$$F(f) \mathbf{x} G(f) \text{ ---> } f(t)\, g(t)$$

$$x_k \mathbf{x} y_k \text{ ---> } N\, X_k\, Y_k$$

$$x_k\, y_k \text{ ---> } (1/N)\, X_k \mathbf{x} Y_k$$

Multiplication in one domain is equivalent to convolution in the other domain and visa versa. For example the transform of a truncated sin function are two delta functions convolved with a sinc function, a truncated sin function is a sin function multiplied by a square pulse.

- The transform of a triangular pulse is a sinc^2 function. This can be derived from first principles but is more easily derived by describing the triangular pulse as the convolution of two square pulses and using the convolution-multiplication relationship of the Fourier Transform.

1.1.4 Sampling theorem

The sampling theorem (often called "Shannons Sampling Theorem") states that a continuous signal must be discretely sampled at least twice the frequency of the highest frequency in the signal.

More precisely, a continuous function f(t) is completely defined by samples every 1/fs (fs is the sample frequency) if the frequency spectrum F(f) is zero for f > fs/2. fs/2 is called the Nyquist frequency and places the limit on the minimum sampling frequency when digitising a continuous sugnal.

If x(k) are the samples of f(t) every 1/fs then f(t) can be **exactly** reconstructed from these samples, if the sampling theorem has been satisfied, by

$$f(t) = \sum_{k=-\infty}^{k=\infty} x(k)\ \text{sinc}(t\ fs\ -\ k)$$

where

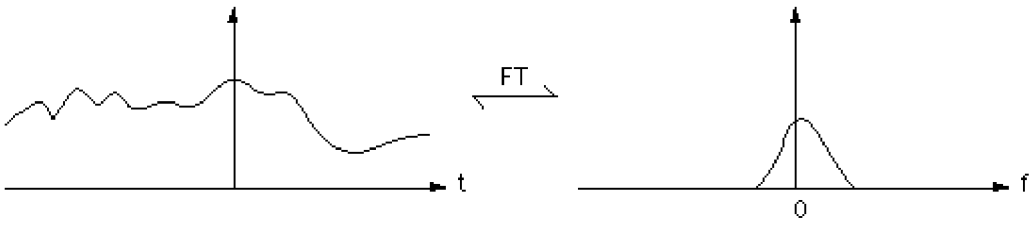
PDF-XCHANGE Click to buy NOW! www.docu-track.com

$$\text{sinc}(x) = \frac{\sin(\pi x)}{\pi x}$$

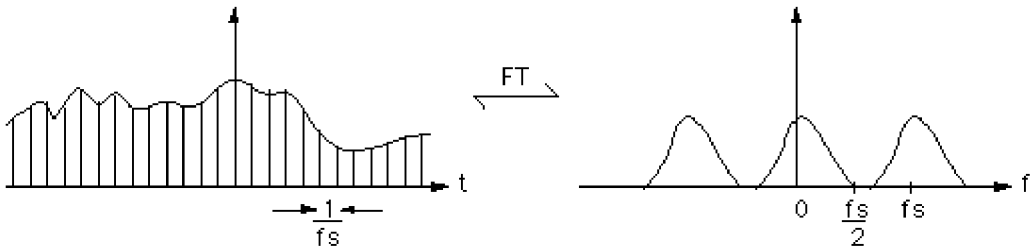
Normally the signal to be digitised would be appropriately filtered before sampling to remove higher frequency components. If the sampling frequency is not high enough the high frequency components will wrap around and appear in other locations in the discrete spectrum, thus corrupting it.

The key features and consequences of sampling a continuous signal can be shown graphically as follows.

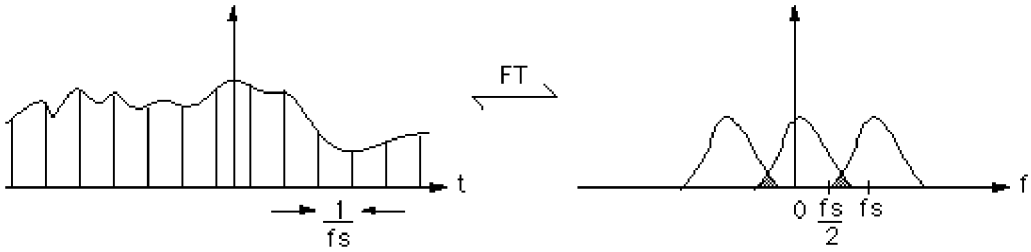
Consider a continuous signal in the time and frequency domain.



Sample this signal with a sampling frequency f_s, time between samples is $1/f_s$. This is equivalent to convolving in the frequency domain by delta function train with a spacing of f_s.



If the sampling frequency is too low the frequency spectrum overlaps, and become corrupted.



Another way to look at this is to consider a sine function sampled twice per period (Nyquist rate). There are other sinusoid functions of higher frequencies that would give exactly the same samples and thus can't be distinguished from the frequency of the original sinusoid.

2.THEORITICAL PART

2.1 Information About Fourier

In complex notation, the time and frequency domains each contain *one signal* made up of *N complex points*. Each of these complex points is composed of two numbers, the real part and the imaginary part. For example, when we talk about complex sample , it refers to the combination of

and *X*[42] *ReX*[42]. In other words, each complex variable holds two numbers. When *ImX*[42]two complex variables are multiplied, the four individual components must be combined to form the two components of the product. The following discussion on *"How the FFT works"* uses this jargon of complex notation. That is, the singular terms: *signal, point, sample*, and *value*, refer to the *combination* of the real part and the imaginary part.

2.1.1 FFT

The FFT operates by decomposing an *N* point time domain signal into *N* time domain signals each composed of a single point. The second step is to calculate the *N* frequency spectra corresponding to these *N* time domain

signals. Lastly, the *N* spectra are synthesized into a single frequency spectrum. Figure 2.1 shows an example of the time domain decomposition used in the FFT. In this example, a 16 point signal is decomposed through four separate stages.

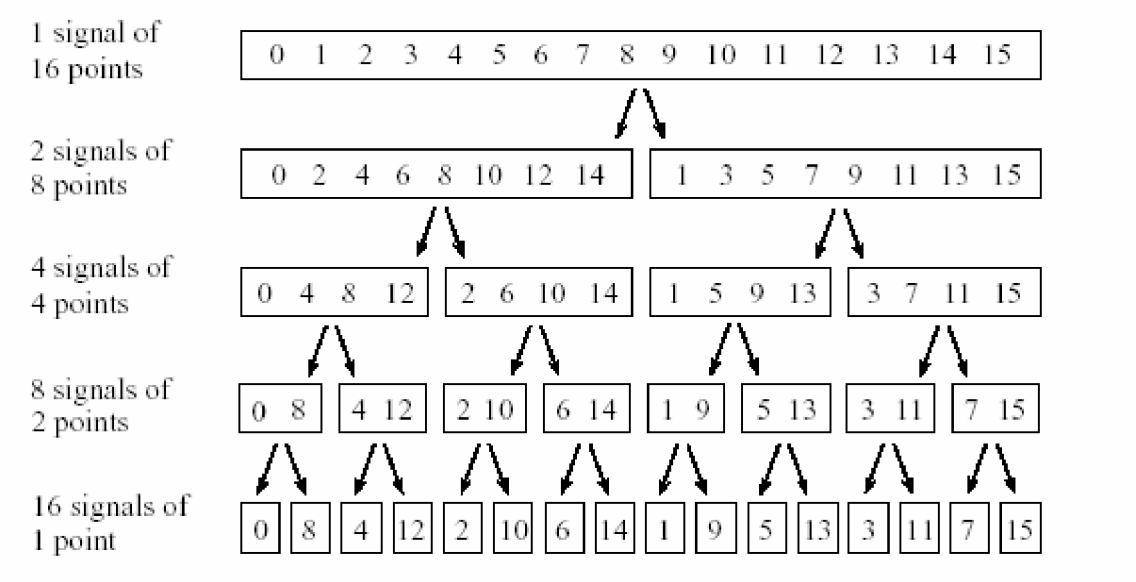


Figure 2.1

Sample numbers in normal order

Decimal	*Binary*
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101

Sample numbers after bit reversal

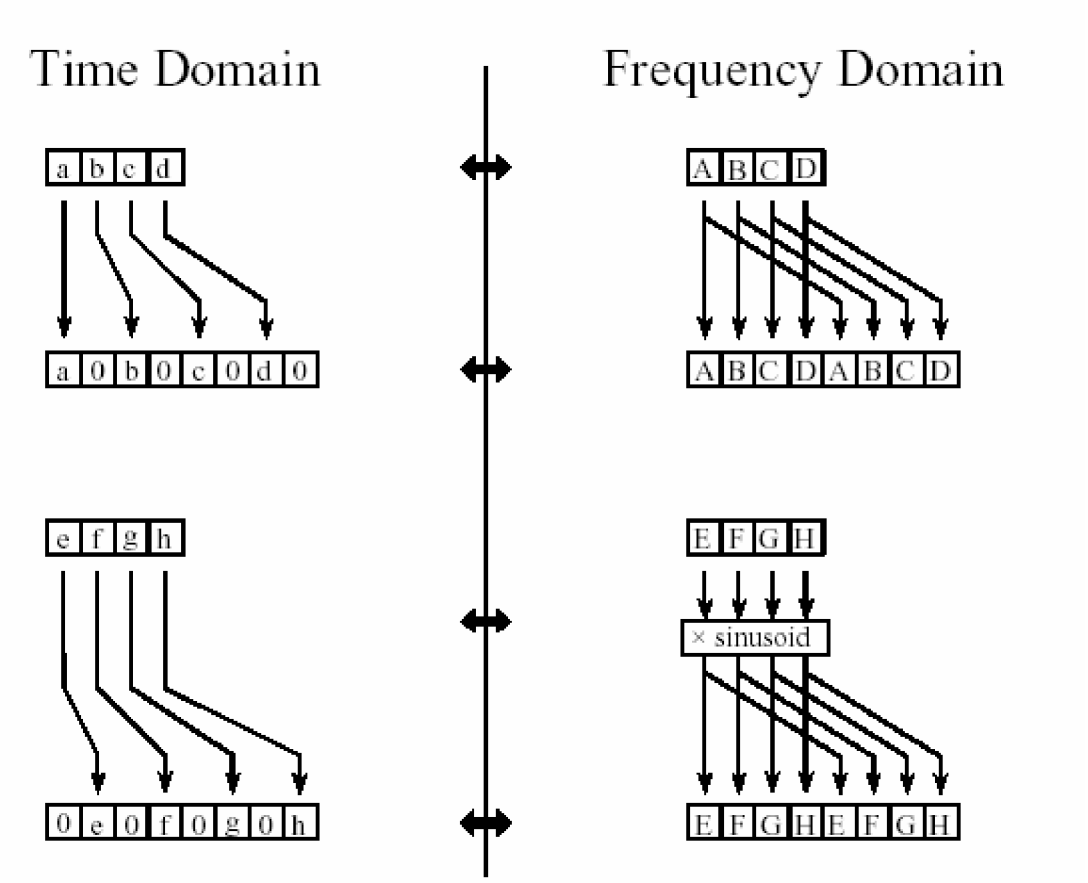
Decimal	*Binary*
0	0000
8	1000
4	0100
12	1100
2	0010
10	1010

6	0110	6	0100
7	0111	14	1110
8	1000	1	0001
9	1001	9	1001
10	1010	5	0101
11	1011	13	1101
12	1100	3	0011
13	1101	11	1011
14	1110	7	0111
15	1111	15	1111

The FFT bit reversal sorting. The FFT time domain decomposition can be implemented by sorting the samples according to bit reversed order separate stages. The first stage breaks the 16 point signal into two signals each consisting of 8 points. The second stage decomposes the data into four signals of 4

points. This pattern continues until there are *N* signals composed of a single point. An **interlaced decomposition** is used each time a signal is broken in two, that is, the signal is separated into its even and odd numbered samples. There are stages required in this decomposition, i.e., *Log*2*N* a 16 point signal (24) requires 4 stages, a 512 point signal (27) requires 7 stages, a 4096 point signal (212) requires 12 stages, etc. Remember this value, ; it will be referenced many times in this chapter. *Log*2*N*. The decomposition is nothing more than a *reordering* of the samples in the signal. Figure 2.2 shows the rearrangement pattern required..On the left, the sample numbers of the original signal are listed along with their binary equivalents. On the right, the rearranged sample numbers are listed, also along with their binary equivalents. The important idea is that the binary numbers are the *reversals* of each other. For example, sample 3 (0011) is exchanged with sample number 12 (1100). Likewise, sample number 14 (1110) is swapped with sample number 7 (0111), and so forth. The FFT time domain decomposition is usually carried out by a **bit reversal sorting** algorithm. This involves rearranging the order of the *N* time domain samples by counting in binary with the bits flipped left-for-right .The next step in the FFT algorithm is to find the frequency spectra of the1 point time domain signals. Nothing could be easier; the frequency spectrum of a 1 point signal is equal to *itself*. This means that *nothing* is required to do this step. Although there is no work involved, don't forget that each of the 1 point signals is now a frequency spectrum, and not a time domain signal.

The last step in the FFT is to combine the *N* frequency spectra in the exact reverse order that the time domain decomposition took place. This is where the algorithm gets messy. Unfortunately, the bit reversal shortcut is not applicable, and we must go back one stage at a time. In the first stage, 16 frequency spectra (1 point each) are synthesized into 8 frequency spectra (2 points each). In the second stage, the 8 frequency spectra (2 points each) are synthesized into 4 frequency spectra (4 points each), and so on. The last stage results in the output of the FFT, a 16 point frequency spectrum. Figure 2.2 shows how two frequency spectra, each composed of 4 points, are combined into a single frequency spectrum of 8 points. This synthesis must *undo* the interlaced decomposition done in the time domain. In other words, the frequency domain operation must correspond to the time domain procedure of *combining* two 4 point signals by interlacing. Consider two time domain signals, *abcd* and *efgh*. An 8 point time domain signal can be formed by two steps: dilute each 4 point signal with zeros to make it an 8 point signal and than add the signals together.



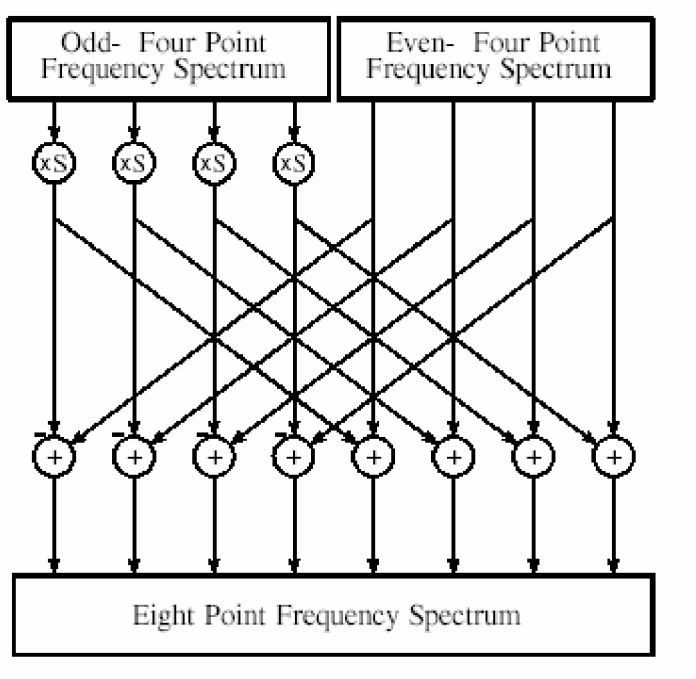


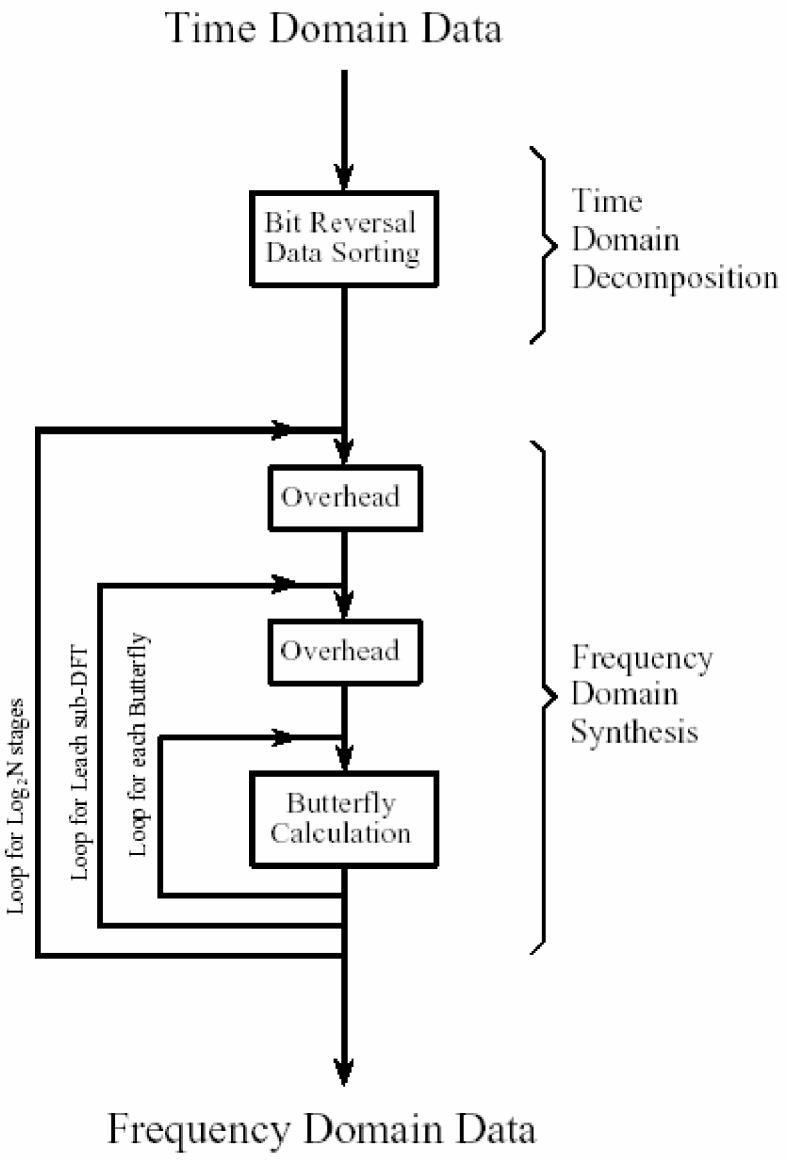
Figure 2.2

That is, *abcd* becomes *a0b0c0d0*, and *efgh* becomes *0e0f0g0h*. Adding these two 8 point signals produces *aebfcgdh*. Diluting the time domain with zeros corresponds to a *duplication* of the frequency spectrum. Therefore, the frequency spectra are combined in the FFT by duplicating them, and then adding the duplicated spectra together. In order to match up when added, the two time domain signals are diluted with zeros in a slightly different way. In one signal, the *odd points* are zero, while in the other signal, the *even points* are zero. In other words, one of the time domain signals is shifted to the right by one sample. This time domain shift corresponds to multiplying the spectrum by a *sinusoid*. To see this, recall that a shift in the time domain is equivalent to convolving the signal with a shifted delta function. This multiplies the signal's spectrum with the spectrum of the shifted delta function. The spectrum of a shifted delta function is a sinusoid.

Figure 2.2 shows a flow diagram for combining two 4 point spectra into a single 8 point spectrum. This simple flow diagram is called a **butterfly** due to its winged appearance. The butterfly is the basic computational element of the FFT, transforming two complex points into two other complex points.

Figure 2.3 shows the structure of the entire FFT. The time domain decomposition is accomplished with a bit reversal sorting algorithm. Transforming the decomposed data into the frequency domain involves *nothing* and therefore does not appear in the figure. The frequency domain synthesis requires three loops. The outer loop runs through the stages . The middle loop moves through each of the individual frequency spectra in the stage being worked on (i.e., each of the boxes on any one level in Fig. 2.2). The innermost loop uses the butterfly to calculate the points in each frequency

spectra (i.e., looping through the samples inside any one box in Fig. 2.2). The overhead boxes in Fig. 2.3 determine the beginning and ending indexes for the loops, as well as calculating the sinusoids needed in the butterflies.



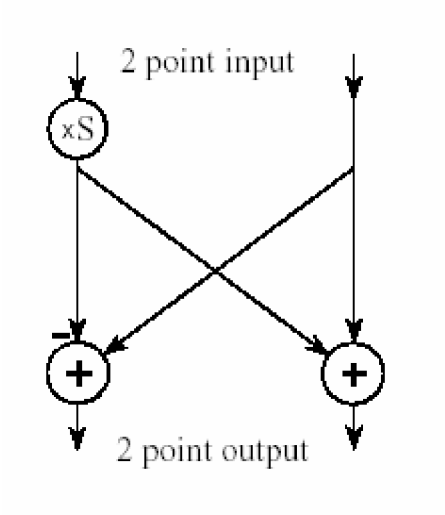
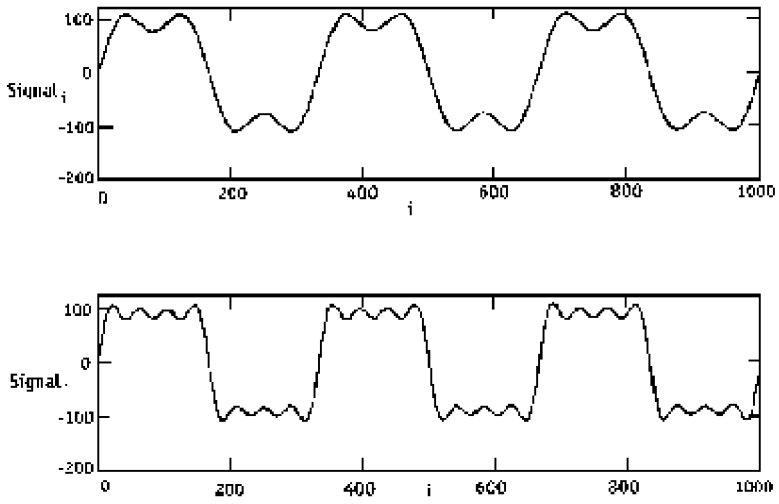


Figure 2.3

2.1.2 Fourier Analysis

Fourier Analysis is based on the concept that real world signals can be approximated by a sum of sinusoids, each at a different frequency. The more sinusoids included in the sum, the better the approximation.



The first trace in the above figure is the sum of 2 sine waves with amplitudes chosen to approximate a 3 Hz square wave (time base is msec). One sine wave has a frequency of 3 Hz and the other has a frequency of 9 Hz. The second trace starts with the first but adds a 15 Hz sine wave and a 21 Hz sine wave. It is clearly a better approximation.

PDF-XCHANGE Click to buy NOW! www.docu-track.com

Such a sum of sinusoids is called a Trigonometric Fourier Series. The terms of the Fourier series for simple waveforms can be found using calculus and many have been published in standard textbooks.

The frequency of each sinusoid in the series is an integer multiple of the frequency of the signal being approximated. These are referred to as the harmonics of the original waveform. In the example above, the basic frequency is 3 Hz so we would expect harmonics at 3 Hz, 6 Hz, 9 Hz, 12 Hz, 15 Hz, etc. It turns out that for this particular waveform, all the even harmonics have amplitudes of zero. This is not true for all waveforms.

The Frequency Spectrum

Each of the harmonic frequencies is defined by a magnitude (amplitude) and a phase. The phase indicates how to shift the harmonic before adding it to the sum. The phase information can be difficult to interpret and its use is restricted to a few specialized applications. The rest of this document is concerned only with harmonic magnitudes.

Plotting the harmonic magnitudes on the y-axis and the frequency of the harmonic on the x-axis generates a Frequency Spectrum. The spectrum is a set of vertical lines or bars because harmonics can only have frequencies that are integer multiples of the original signal frequency. In theory, the spectrum includes frequencies up to infinity but in practice the magnitude of very high frequency harmonics are usually insignificant.

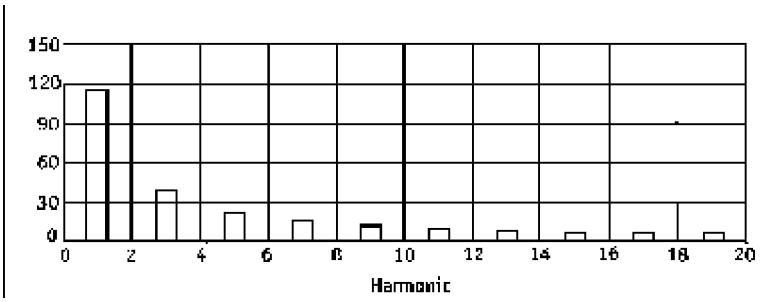


Figure 2.4

The plot shown here includes the first 20 harmonics of a square wave. The x-axis shows the harmonic number. In order to convert harmonic number to Hz, simply multiply by the frequency of the original signal. For example, if the original signal had a frequency of 2 kHz, the third harmonic would be at 6 kHz.

Note that because the original signal was a square wave, the even harmonics are zero. This is not always the case.

PDF-XCHANGE Click to buy NOW! www.docu-track.com

If the input had been a pure tone, we would expect a single bar with an amplitude equal to the amplitude of the input tone.

Fourier Analysis with Computers - FFT

Several techniques have been developed that enable a computer to calculate the Frequency Spectrum of a signal. The first step in all cases is to convert the signal to a set of numbers for the computer to use. This is done by sampling the signal at a regular interval so that a table of values is created. Each sample value is separated from the next by a fixed period of time.

The number of points obtained and the time between samples combine to determine the length of time we look at the signal. The following definitions apply:

fs = sample rate in Hz
dT = 1/fs = interval between samples
N = number of samples taken
T = N x dT = total time period

f1 = 1/T = frequency of the first harmonic in Hz

As an example, if we were interested in the AC power line, we might choose a sample rate (fs) of 6000 Hz and collect 100 samples (n=100). The result would be dT = 0.1167 msec and T = 16.667 msec. The first harmonic would then be f1 = 60 Hz.

The traditional mathematical approach to Fourier analysis was based on approximating continuous waveforms but computer techniques can only deal with a set of samples. This does not change the basic idea of harmonic analysis but now we have to keep in mind the following:

1. The spectrums based on sampled waveforms can generate only N/2 harmonics.

2. If the original signal contains more than N/2 harmonics, the higher frequency harmonics will cause errors in the magnitude spectrum.

This type of error is referred to as Aliasing. Aliasing can be prevented by filtering the input signal before performing the Fourier analysis so that there are no frequency components above f1 and N/2.

The most popular computer algorithm for generating a frequency spectrum is the FFT or Fast Fourier Transform. As the name implies, the FFT is very efficient but it does have one quirk that affects the way it is used. The FFT can only process a sampled waveform where N (number of samples) is a power of 2. Acceptable values of N include 128, 256, 512 and 1024. The full significance of this is discussed later in this note.

PDF-XCHANGE Click to buy NOW! www.docu-track.com

Note: The FFT, like most computer algorithms, generates an Exponential Fourier Series, instead of a Trigonometric Fourier Series. The two series are identical except that the magnitude generated by the exponential series are half the value of the trigonometric series. Most application software automatically compensates for this and presents the magnitude spectrum as a Trigonometric series.

Analyzing More Complicated Waveforms

Up to now we have assumed that the input signal is periodic with a constant frequency. We set up the spectrum so that the first harmonic was the same as the basic input frequency. This straight forward approach is useful when we know something about the signal ahead of time such as in the case when analyzing AC voltage.

In many situations, we are faced with a more complicated signal that may not have an obvious single period. An example is the following:

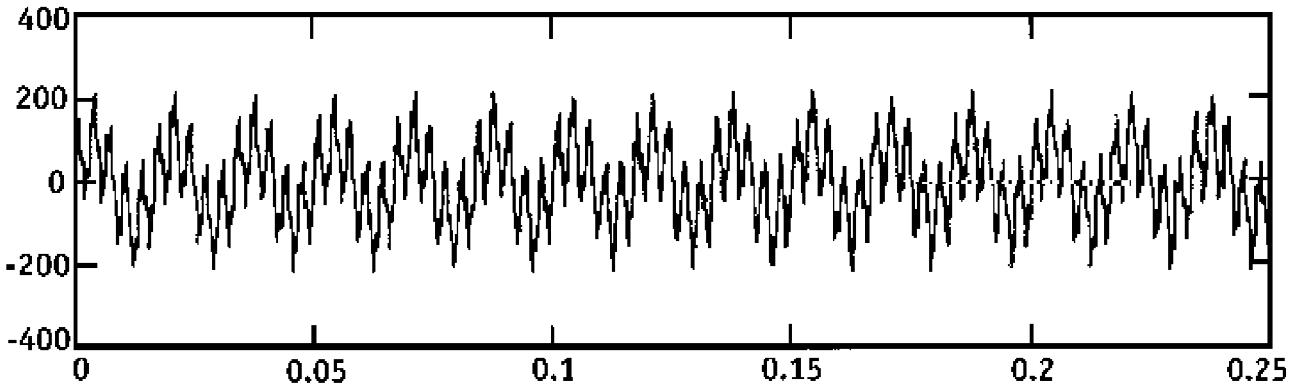


Figure 2.5

The x-axis is time in seconds. If we sample this signal at a rate of 4096 Hz, collect 1024 samples and put them through our FFT we get the following spectrum:

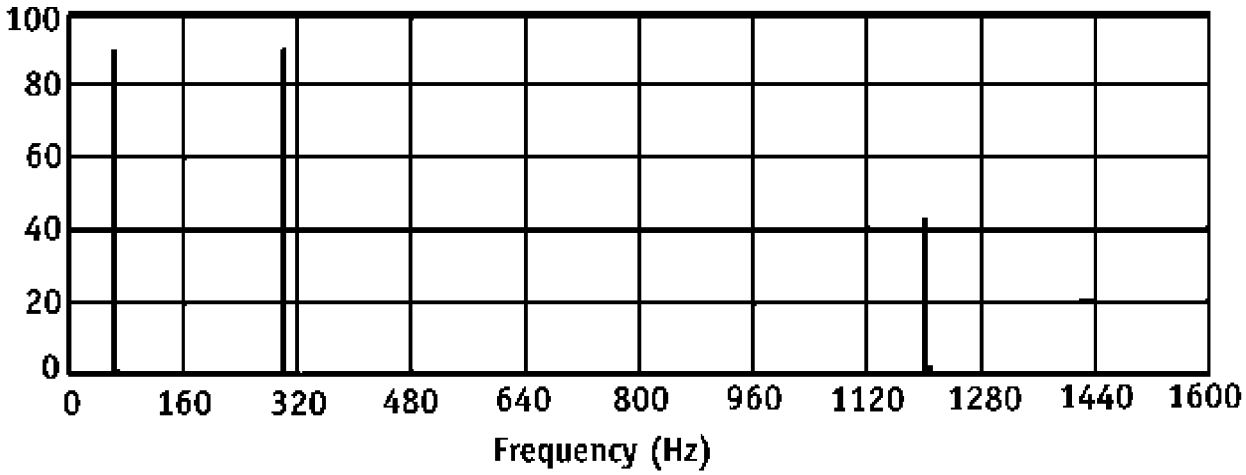


Figure 2.6

The spectrum tells us that the original signal is a composite of 3 pure tones. In fact, these tones are at 60 Hz, 300 Hz and 1200 Hz.

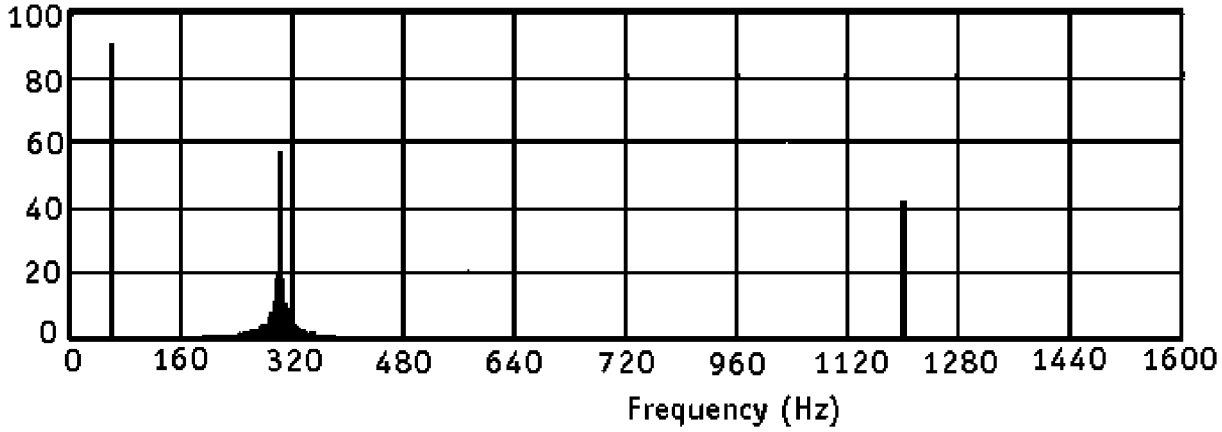
It is useful at this point to make a few observations.

1. Since N=1024, the FFT generated 512 harmonics. To simplify scaling the x-axis, the plot shows only the first 400.
2. The magnitudes are corrected for the trigonometric series and therefore use the same units as the input signal.
3. The length of the sample period is 0.25 seconds, so the first harmonic (f1) is 4 Hz. The spectrum looks nice and clean because all of the tones are multiples of 4 Hz (f1).

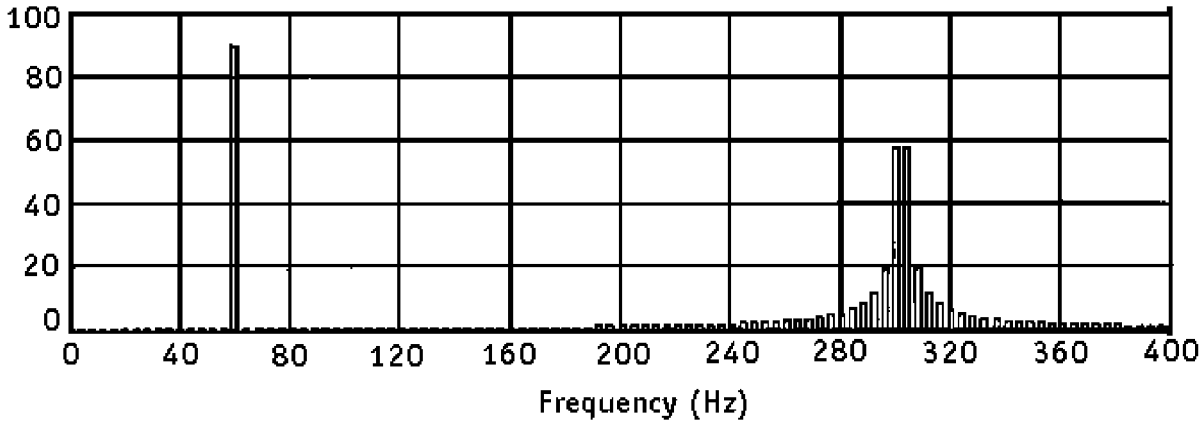
Unfortunately, this is not often the case.

Leakage

The previous example looked great because all the tones making up the original signal were multiples of f1 (1/T). Following is the same arrangement but the 300 Hz tone has moved to 302 Hz.



Although the 302 Hz tone has not actually changed amplitude, the spectrum looks very different. This effect is called Leakage and occurs when a frequency component of the original signal is not an integer multiple of the sampling period (T). The amplitude of the tone appears to leak into other harmonics. The effect can be made more obvious by expanding the x-axis.



This is the same spectrum, but we now show only the first 100 harmonics. The steps of 4 Hz become obvious. Note that the tone amplitude should be 90 (equal to the 60 Hz tone).

Leakage is a direct result of using a limited set of sampled data. If we could collect and process an infinite amount of samples, the frequency increment (x-axis) becomes very small and all possible frequencies would be valid harmonics.

The FFT algorithm does not add anymore leakage than other algorithms but it makes correcting the problem more difficult by limiting our choices for N to powers of 2.

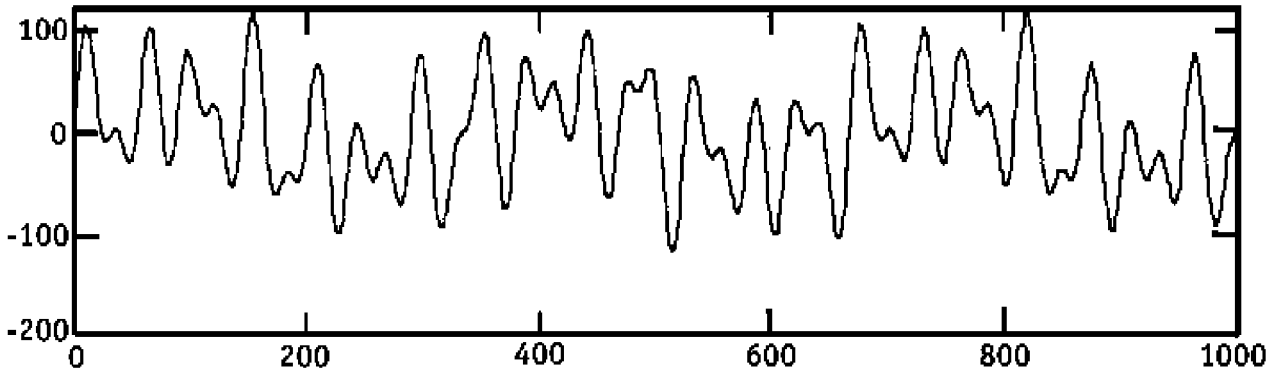
There are two approaches to improving the accuracy of the spectrum when leakage is a problem.

1. Change the sample set to line up with the harmonic content.

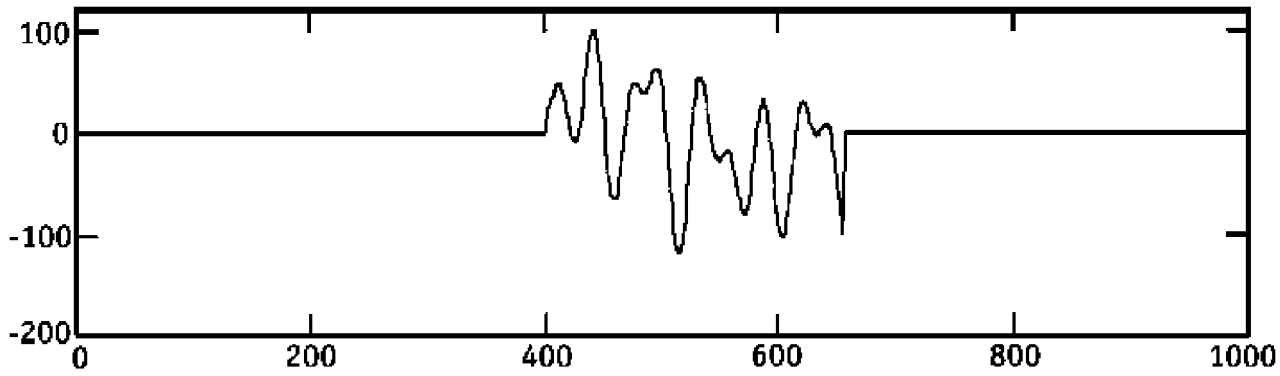
2. Modify the sampled values with a weighted window.

Using Weighted Windows to Minimize Leakage

The term window has a specific meaning when applied to samples used to create a Fourier spectrum. Consider the signal in the first figure. In order to analyze the signal for harmonics we have to sample it for some fixed period (T).



Mathematically, this amount to multiplying one set of values by 1.0 and all other values of the waveform by zero. In effect, we create a set of sampled data by blocking out all other parts of the waveform with a Rectangular Window with an amplitude 1.0. The result is shown in the next



Most computer algorithms, including the FFT, process this set of samples as if it were a continuous loop. If N=256 (samples 0 to 255), then it assumes that the value after 255 is the same as the value at sample 0. This assumption is true when the window (T) includes an integer multiple of a repeating cycle. When it is not true, as in this example, the discontinuity between the value of the first and last samples looks like a step change to the FFT and causes leakage.

PDF-XCHANGE Click to buy NOW! www.docu-track.com

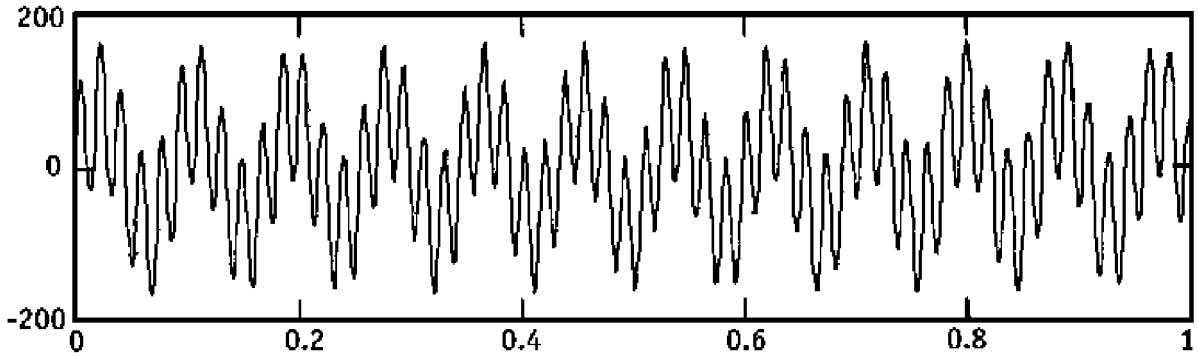
Using a non-rectangular window can reduce leakage. In practice, this is done after the sample set is collected by scaling each sample in the set. The scaled values are than used as the input to the FFT.

Many windows have been developed and each has its own set of costs and benefits.

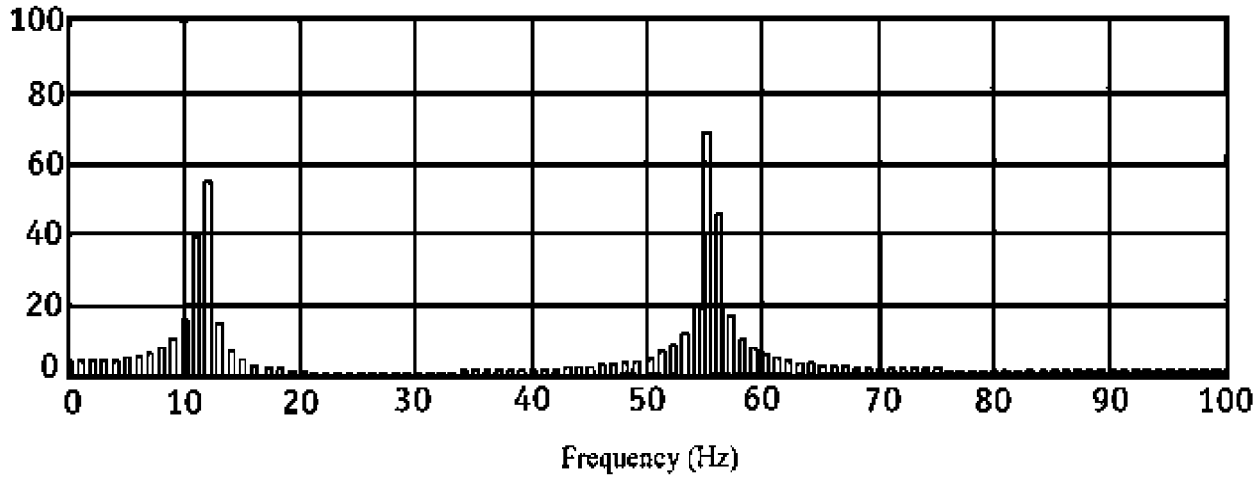
Window Example- Rectangular

The first figure is a plot of 1024 samples that will be used to generate a Fourier magnitude spectrum. The x-axis is time in seconds. The sampling process is defined by the following parameters:

fs = 1024 Hz (sample rate)
N = 1024 (number of samples)
T = 1 second (window length)



The FFT produces 512 harmonics (N/2) from this set of samples. The spacing between harmonics is 1 Hz, equal to the frequency of the first harmonic (f1 = 1/T). The second figure plots magnitudes of the first 100 harmonics. Note that the term harmonic can be confusing because it id derived from the window length T and may have no obvious relationship to the signal.

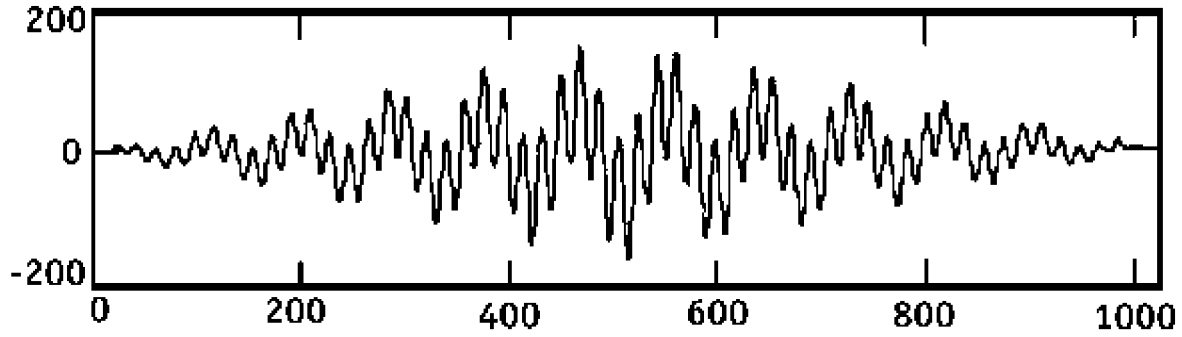


The spectrum indicates that the original signal is composed of 2 basic components but leakage makes it difficult to determine the magnitudes. The original signal was actually composed of 2 tones. One is an 11.6 Hz tone with an amplitude of 75 while the second is 55.4 Hz with an amplitude of 90.

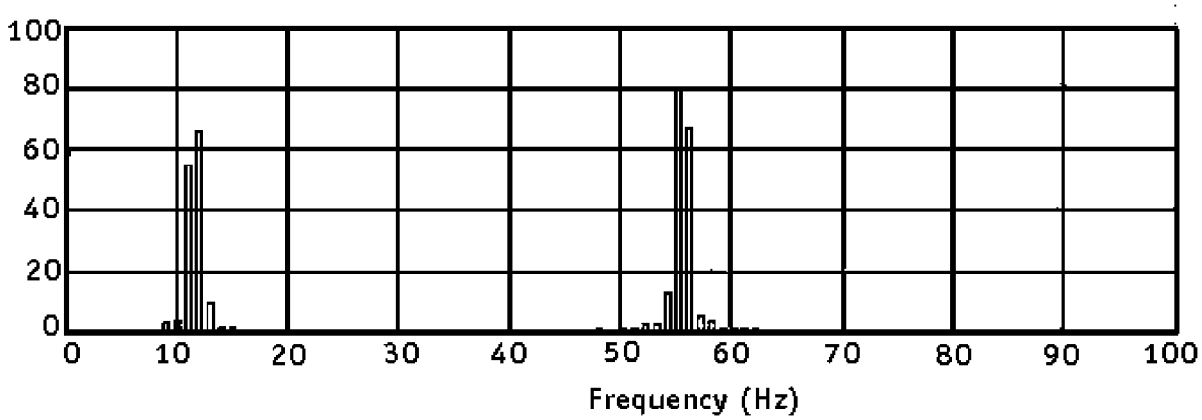
The spectrum shows one peak at 12 Hz with a magnitude of 55.1 and another at 55 Hz with a magnitude of 68.2. The relative magnitude of the 2 components is close but the absolute magnitudes are badly distorted.

Window Example - Triangle and Hamming

The next figure shows the waveform from the previous example modified by a Triangle Window. The original waveform was sampled in exactly the same way as before but the sampled points were scaled by the triangular window. Note how the early and the late sample points are now attenuated.

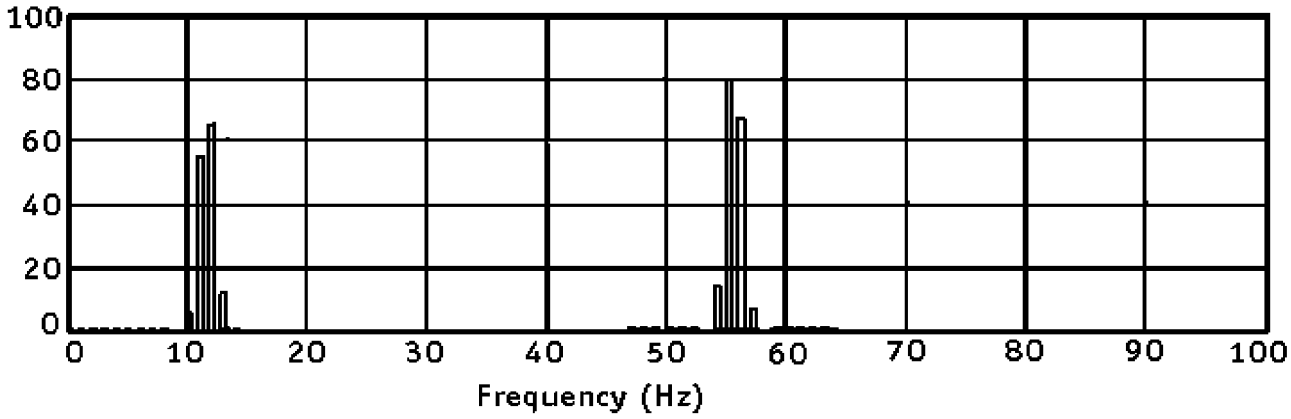


The next plot shows the first 100 harmonics when this modified set of samples is used.



The peaks are at the same frequency but there is clearly less spreading (leakage). The magnitudes are also more accurate (65.7 and 78.8) but still incorrect.

Another popular window is the Hamming Window based on a cosine function. It is also designed to peak in the center of the samples and taper to zero at the ends. Using the Hamming Window on the same set of samples produces the following spectrum. The peaks are located at the same points with amplitudes of 65.7 and 79.1.



Note: The magnitudes have been corrected for a type of amplitude error created by non-rectangular windows. The factor is 0.5 for the Triangle and 0.54 for the Hamming. This is a technical correction based on lobe height.

Notes on Selecting a Window

1. If the input signal is periodic and a multiple of the sampling window, then use a Rectangular window.

2. If the input is a short pulse or burst that starts and ends at the same amplitude, then use a Rectangular window (as long as the sampling window includes the entire transient).

PDF-XCHANGE Click to buy NOW! www.docu-track.com

3. If the input is a sections of a continuous waveform that is not periodic then use the Hamming or Triangular windows.

Re-Sampling

As illustrated in the previous examples, the use of weighted windowing can improve the quality of the spectrum when frequency components are not integer multiples of the sampling window. Unfortunately, the examples also indicate that even with a weighted window, there is still some distortion.

In situations where major frequency components are multiples of one another, the best approach is to adjust the sampling process so that a full period of the input signal lines up with the sampling window. If this can be done, the best window is Rectangular. This situation is found in applications like power line monitoring where there is a strong primary frequency driving the signal.

The cleanest approach is to adjust the sample rate until N points are obtained in exactly one period of the input signal. This may not be practical when using the FFT because N must be a power of 2. The required sample rate may not be available or the data may have been collected previously and cannot be duplicated.

The solution to both problems is to re-sample a set of data, interpolating where required to produce the desired set of samples.

Decibels

It is common to present the amplitude of the frequency spectrum in decibels (dB). A decibel (1/10 Bel) is a unit originally used to measure the intensity of a sound with reference to a standard sound. In general, it can be used to indicate the intensity or power of any signal with respect to a reference. The user of a logarithmic scale permits a large range of amplitudes to be presented on a single plot.

As it applies to power, the dB is defined as: dB = 10log(Power/Reference Power)

Note that a negative value indicates power less than the reference. The reference varies with the application and this fact can cause some confusion. A letter is sometimes added to indicate the use of a standard reference. For example, a dBm measures signal power with respect to 1 milliwatt. Assuming power is measured in watts, the following equation applies: dBm =10Log(Mag/.001)

It has become common practice to treat the squared value of certain signals as power. In the case of a voltage waveform and a fixed load, this is valid because power is actually V*V/R.

Following this logic, the units of dBm are sometime used when only a voltage signal is available by assuming a standard load of 50 ohms. A sinusoidal voltage with a magnitude of 0.316 V is shown as 0 dBm (1 milliwatt).

The situation gets more confusing because most electrical engineers use the term dB to indicate signal amplitude (not signal power) with respect to a reference. This can be linked mathematically to the standard definition as long as a constant electrical load (R) is assumed because:

P1/P2 = [(V1 x V1)/R]/[V2 x V2)/R] = (V1 x V1)/(V2 x V2)
and
10Log[(V1 x V1)/(V2 x V2)] = 20 Log(V1/V2)

Therefore, when calculating dB from signal amplitude (instead of signal power) user a factor of 20: dB = 20Log(Magnitude/Reference)

The units of dBV are based on a reference of 1V and assume the input signal is in volts. dBV - 10Log(Mag/1V)

The magnitude can also be compared to the peak of the original signal or the RMS of the original signal to derive additional dB scales.

Instruments with a built-in amplifier can use the unit dBVFS. This is obtained by referencing the magnitude to the full scale range setting of the input amplifier (20Log(Mag/FS). This is useful because it does not require that the input signal be in volts or watts.

2.2 INFORMATION ABOUT THE HARDWARE

2.2.1 PIC 16F877

PIC is such a kind of microcontroller that is produced by MICROCHIP.Its integrated structure contains CPU,RAM and I/O Unit all together.This is the main differance of microcontrollers from microprocessors.Because of its integrated achitecture there is no need to set up a databus and address bus between these CPU,RAM and I/O.So they are very useful.In addition we use only 35 assembly codes for programming PICs.This makes very easier to use it.

16F877 is one of the most advanced microcontroller of PIC family.It supports an external crystal up to 20 MHz.We can use 14 different inerrupts with 16F877.It also contains 8K flash memory,5 differant ports(A,B,C,D,E),3 timers and ADC(12 bits).

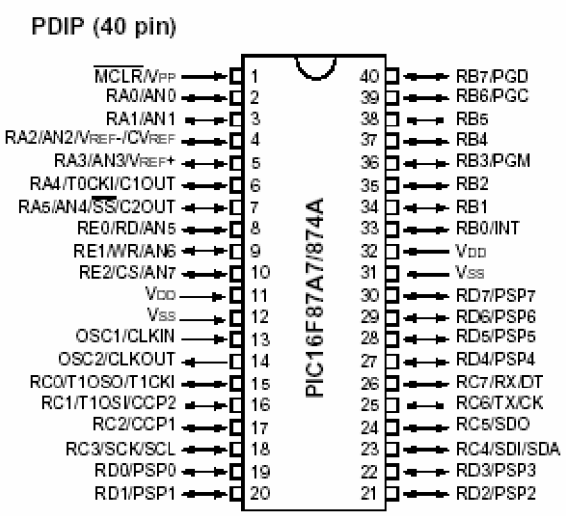


Figure 2.7

But in this Project the RAM of PIC 16F877 was not enough for the vectors that we have to define for the FFT part of the software.So we use PIC 18F452.There is no difference in pin configuration and programming between 16F877 and 18F452.

Key Features	PIC16F873A	PIC16F874A	PIC16F876A	PIC16F877A
Operating Frequency	DC - 20 MHz	DC - 20 MHz	DC - 20 MHz	DC - 20 MHz
RESETS (and Delays)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)
FLASH Program Memory (14-bit words)	4K	4K	8K	8K
Data Memory (bytes)	192	192	368	368
EEPROM Data Memory (bytes)	128	128	256	256
Interrupts	14	15	14	15
I/O Ports	Ports A,B,C	Ports A,B,C,D,E	Ports A,B,C	Ports A,B,C,D,E
Timers	3	3	3	3
Capture/Compare/PWM modules	2	2	2	2
Serial Communications	MSSP, USART	MSSP, USART	MSSP, USART	MSSP, USART
Parallel Communications	—	PSP	—	PSP
10-bit Analog-to-Digital Module	5 input channels	8 input channels	5 input channels	8 input channels
Analog Comparators	2	2	2	2
Instruction Set	35 Instructions	35 Instructions	35 Instructions	35 Instructions
Packages	28-pin PDIP 28-pin SOIC 28-pin SSOP 28-pin MLF	40-pin PDIP 44-pin PLCC 44-pin QFP	28-pin PDIP 28-pin SOIC 28-pin SSOP 28-pin MLF	40-pin PDIP 44-pin PLCC 44-pin QFP

Table 2.1

2.2.2YMS 12864-02 GRAPHIC LCD

This LCD module has a resolution of 128x64.It mean there is 128 columns and 64 lines.This module is controlled by to microprocessors.One of them controlles the left side and the other controlles the right side.You can not reach to the each of the lines on the y axis.Firstly you have to choose the page then the column and finally send the data.For example if you send D=11111111 all of the pixels in the arranged page and column will be black.

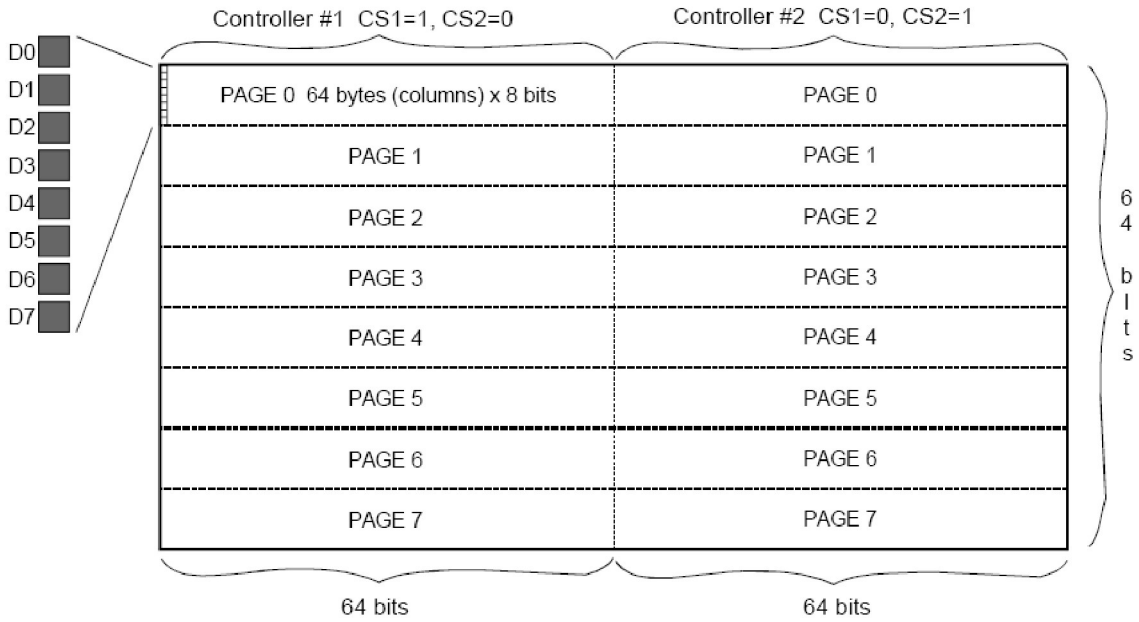


Figure 2.8

Here you can see the connection of LCD to adjust the contrast of it.

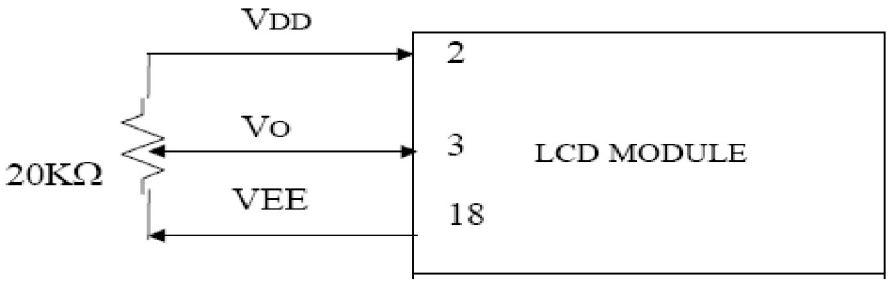


Figure 2.9

In the table below there is the configuration bits for data write yo LCD.If you see the schematic of the project there would be seen that you are not able to read data from LCD.Because there is no need to read data in this project.So The R/W pin is grounded.

Instructions	Code										Functions	
	R/W	D/I	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Display on/off	0	0	0	0	1	1	1	1	1	1/0	Controls display on/off. RAM data and internal status are not affected. (0:OFF,1:ON)	
Display start line	0	0	1	1	Display start line (0-63)						Specifies the RAM line displayed at the top of the screen.	
Set Page (x address)	0	0	1	0	1	1	1	Page (0-7)			Sets the page (X address) of RAM at the page (X address) register.	
Set Y address	0	0	0	1	Y address (0-63)						Sets the Y address in the Y address counter.	
Status read	1	0	Busy	0	ON / OFF	Reset	0	0	0	0	Reads the status. Reset 1: Reset 0: Normal ON/OFF 1: Display off 0: Display on Busy 1: Internal operation 0: Ready	
Write display data	0	1	Write data								Writes data DB0 (LSB) to DB7 (MSB) on the data bus into display RAM.	Has access to the address of the display RAM specified in advance. After the access, Y address is increased by 1.
Read display data	1	1	Read data								Reads data DB0 (LSB) to DB7 (MSB) from the display RAM to the data bus.	

Table 2.2

Note:Be careful that if you see the software part of the project you will see a function called LCD_INIT.If you don't use this function in the main() function LCD would not work.We send the initilization bits in that function.

2.2.3 General Information About Spectrum Analyzers

If the equation for a waveform is known,it can be mathematically processed by Fourier analysis to evaluate its component waves.In a digital spectrum analyzer,the waveform to be analyzed is sampled,and the samples are digitized and fed to computer.The computer is programmed to

PDF-XCHANGE Click to buy NOW! www.docu-track.com

determine the waveform equation from the samples,and to analyze the equation to calculate the component waves,The component waveforms are storred in the memory.

The sampling and conversion tecniques in digital spectrum analyzer are the same as those used in a digital oscilloscop.

The algorithm used in digital spectrum analyzer is known as Fast Fourier Transform(FFT).Therefore This type of instrument are also known as FFT analyzer or Fourier analyzer.

Digital spectrum analyzers measures all frequencies simultaneously,so they are able to investigate changing,or dynamic waveforms.Consequently they are also called dynamic signal analyzers,as compared to the digital type, which display stored signals.

Digital spectrum analyzer are essantially low-frequency instruments.A typical input frequency range is 250 microHZ to 100 kHz ,and a typical input voltage amplitute might be 4 mV to 30 V peak.

Aliasing or generation of waveforms that are not part of the original signal digital spectrum analyzer ,as in a digital storage oscilloscope.This is avoid by the use of filters,and ensuring that the input is sampled at a rate greater than two samples per cycle.

These instruments can be employed for investigating the characteristics of the audio amplifiers,filters and loudspeakers.

3.PRACTICAL PART

3.1 Software

3.1.1 Development Of The Software

In the software part of the project PICC compiler is used for development of the software.C programming language is prefered because it is more easy to program PIC with C according to assembly.

In the program routin firstly the TIMER1 interrupt is seen.In this interrupt the PIC is getting samples with a maximum period of 5 kHz.And then these samples are refreshed every 500 ms.And also I find it suitable to take the DFT of these signals in this interrupt.It can absolutly be suitable to call the DFT routin in the main function.

PDF-XCHANGE Click to buy NOW! www.docu-track.com

The TIMER1 interrupt is a 16 bits of interrupt.DIV8 means timer will count 0 to 65536 in 52.4 us.It can be seen in timer that timer is waited for 100 times in the timer counter time variable.So we can obtain a 500 ms refresh time.

After that the DFT routin is seen.in this routin I convert the classical formula of DFT into C routin.In this routin 323 point DFT of the sampled signal is taken.

Then the next function is the send().in this function.The samples are taken to the input[32] vector.Then the dft() function takes DFT of the input[] and takes it in the output[32] vector.send() function sends these components to the LCD.

The left part of the LCD is used in this project.when the gonder gets the output components, it looks at the length of this components.These lengths must be between 0 and 64.Then send() arranges the hight of the diracs and sends the output.The send() function is called in the main() function.It can be seen that an infinite loop with for is used.until the TIMER1 interrupt cuts the program,the routin will continue to send the output to the LCD.

The maximum sampling frequency and the refreshment rate can be set up in TIMER1 interrupt.If you change the initial value of the time variable the refreshment rate can be adjusted.If you change the time between two samples the maximum sampling frequency should be adjusted.

Becareful that it is impossible that the two sides of the LCD can not be used simultaneously.After using the one side of it LcdSelectSide() funtion in the header file must be used and then the other side must be selected.

Becareful that for progmamming the microcontroller do not use the PIC UP loader program.Because this programs cache is up to 3FF .It means the greater size of programs can not be loaded to PIC with this program.It means that programs with the size of more than 6 kb are not suitable.When a math.h function is used in a program routin the size of it easily incease more then 6kb.

The software is designed in two parts.The first part is the main file.And the second is the header file .The header file is used not to be seen of the main file more crowded.The functions and variables used in the main file can be defined in the header file.After compleating the header file it is included at the top of the main file.

In PICC compiler at the top of the program the name of the device,the oscillator frequency and the headers used in the program have to be included.

If you try to debug this program for PIC16F877 PICC will give an error message that the RAM of the device is not enough for all variables.But if the program routin is debugged for the device of 18F452 there won't be any error messages.

It is recommended to use the loader program as ICPROG.

PDF-XCHANGE Click to buy NOW! www.docu-track.com

3.1.2 Source Code

```
//This is the source code for analyzer.c
#include "C:\Documents and Settings\adem\Desktop\analyzer.h"
 //show the directory of the header file

#include "math.h"

int hk;
unsigned int output[32];
int i,n,v,fg;
float input[32];
float dur;
int time=100;
int dft();

#int_TIMER1
TIMER1_isr()
{
set_timer1(60000);
time--;
if(time==0)
{
for(i=0;i<32;i++)
{
   input[i]=read_adc()/256;
   delay_us(200);


}
dft();
}



}


int dft()

{

float p=3.1416;
int16 REx[32],IMx[32],y;
int k,x;



for(k=0;k<32;k++)
{

  for(n=0;n<32;n++)
  {


    y=((2*pi*n*k)/4);


    REx[k]+=input[n]*cos(y);
```

```
    IMx[k]+=input[n]*sin(y);



}
}

for(k=0;k<32;k++)
{

  output[k]=abs(sqrt(pow(IMx[k],2)+pow(REx[k],2)));

}


}



int send ()

{

int yatay=0;
int temp;
int k;

       for(i=0;i<32;i++)
       {

       temp=output[i];
        v=temp % 8;
       if(v==0)
         {

     LcdInstructionWrite (X_ADRESS +(temp/8));
     LcdInstructionWrite (Y_ADRESS +yatay);
     LcdDataWrite (0);
        }

        if(v==1)
        {

     LcdInstructionWrite (X_ADRESS +(temp/8));
     LcdInstructionWrite (Y_ADRESS +yatay);
     LcdDataWrite (1);
        }

        if(v==2)
        {

     LcdInstructionWrite (X_ADRESS +(temp/8));
     LcdInstructionWrite (Y_ADRESS +yatay);
     LcdDataWrite (3);
        }

        if(v==3)
        {

     LcdInstructionWrite (X_ADRESS +(temp/8));
```

PDF-XCHANGE Click to buy NOW! www.docu-track.com

```
      LcdInstructionWrite (Y_ADRESS +yatay);
      LcdDataWrite (7);
         }
         if(v==4)
         {

      LcdInstructionWrite (X_ADRESS +(temp/8));
      LcdInstructionWrite (Y_ADRESS +yatay);
      LcdDataWrite (15);
         }
         if(v==5)
         {

      LcdInstructionWrite (X_ADRESS +(temp/8));
      LcdInstructionWrite (Y_ADRESS +yatay);
      LcdDataWrite (31);
         }
         if(v==6)
         {

      LcdInstructionWrite (X_ADRESS +(temp/8));
      LcdInstructionWrite (Y_ADRESS +yatay);
      LcdDataWrite (63);
         }
         if(v==7)
         {

      LcdInstructionWrite (X_ADRESS +(temp/8));
      LcdInstructionWrite (Y_ADRESS +yatay);
      LcdDataWrite (127);
         }

       for(n=0;n<(output[i]/8);n++)
       {


      temp=temp-8;
      LcdInstructionWrite (X_ADRESS +(temp/8));
      LcdInstructionWrite (Y_ADRESS +yatay);
      LcdDataWrite (255);


       }

      yatay=yatay+1;

      }

}


void main() {


   setup_adc_ports(RA0_ANALOG);
   setup_adc(ADC_CLOCK_DIV_2);

   setup_psp(PSP_DISABLED);
   setup_spi(FALSE);
```

PDF-XCHANGE Click to buy NOW! www.docu-track.com

```
   setup_timer_2(T2_DISABLED,0,1);


   setup_timer_1(T1_INTERNAL|T1_DIV_BY_8);
   setup_timer_2(T2_DISABLED,0,1);
   enable_interrupts(INT_TIMER1);
   disable_interrupts(INT_RB);
   enable_interrupts(global);
   set_timer1(60000);


   GLCD_LcdInit();

   LcdSelectSide(LEFT);


      for(;;)
      {
      send();
      }


}
```

```
//This is the source code for analyzer.h Write this routin in a //different
C file
#include <18F452.h>
#device adc=8
#use delay(clock=20000000)
#fuses XT,NOWDT,NOLVP

/* LCD Registers */
#define X_ADRESS  0xB8  /* Adress base for Page 0 */
#define Y_ADRESS  0x40  /* Adress base for Y0     */
#define START_LINE      0xC0  /* Adress base for line 0 */
#define DISPLAY_ON      0x3F  /* Turn display on        */
#define DISPLAY_OFF     0x3E  /* Turn display off       */

/* General use definitions */
#define RIGHT           0
#define LEFT            1
#define BUSY            0x80
```

PDF-XCHANGE Click to buy NOW! www.docu-track.com

```
#define sin_pi_6  0.5



void LcdSelectSide(int LcdSide);
void GLCD_ClearScreen (void);
void LcdInstructionWrite (int Instruction);



void LcdDelay(int Duration)
{
   int Delay;
      for (Delay=0; Delay<Duration;Delay++);
}


void GLCD_LcdInit(void)
{
   output_d(0x00);
      output_low(PIN_C6);

      output_low(PIN_C7);
      output_low(PIN_E0);
      output_low(PIN_E1);

      output_high(PIN_E2);
      LcdDelay(10);
      output_low(PIN_E2);
      LcdDelay(10);
      output_high(PIN_E2);

      LcdSelectSide(LEFT);
      LcdInstructionWrite(DISPLAY_OFF); /* Display OFF */
      LcdInstructionWrite(START_LINE);
      LcdInstructionWrite(X_ADRESS);
      LcdInstructionWrite(Y_ADRESS);
      LcdInstructionWrite(DISPLAY_ON); /* Display ON */

      LcdSelectSide(RIGHT);
      LcdInstructionWrite(DISPLAY_OFF); /* Display OFF */
      LcdInstructionWrite(START_LINE);
      LcdInstructionWrite(X_ADRESS);
      LcdInstructionWrite(Y_ADRESS);
      LcdInstructionWrite(DISPLAY_ON); /* Display ON */

      GLCD_ClearScreen();
}


void LcdSelectSide(int LcdSide)
{
      if(LcdSide == RIGHT)
      {
            /* switch to right */
            output_low(PIN_C7);
            output_low(PIN_C6);

                  output_low(PIN_E0);
                  output_high(PIN_E1);
```

PDF-XCHANGE Click to buy NOW! www.docu-track.com

```
                    LcdInstructionWrite(Y_ADRESS); /* Set column to 0 */

          }
        else
        {
              /* switch to left */
              output_low(PIN_C7);
              output_low(PIN_C6);

                    output_high(PIN_E0);
                    output_low(PIN_E1);

                    LcdInstructionWrite(Y_ADRESS); /* Set column to 0 */

    }
}



void LcdDataWrite (int Data)
{
      Lcddelay(2);
      output_high(PIN_C6);          /* Data mode */


      output_d(Data);   /* outbyte */

      output_high(PIN_C7);          /* Strobe */
      LcdDelay(1);
      output_low(PIN_C7);
}


void LcdInstructionWrite (int Instruction)
{
      LcdDelay(2);            /* wait until LCD not busy */
      output_low(PIN_C6);           /* Instruction mode */


      output_d(Instruction);/* outbyte */

      output_high(PIN_C7);          /* Strobe */
      LcdDelay(1);
      output_low(PIN_C7);
}



void GLCD_ClearScreen (void)
{
      int Page=0;
      int Column=0;

      /* process the 8 pages of the LCD */
      for (Page = 0; Page < 8; Page++)
      {
                    LcdSelectSide(LEFT);                /* Select left side */
                    LcdInstructionWrite(X_ADRESS | Page); /* Set the page
number */
```

PDF-XCHANGE Click to buy NOW! www.docu-track.com

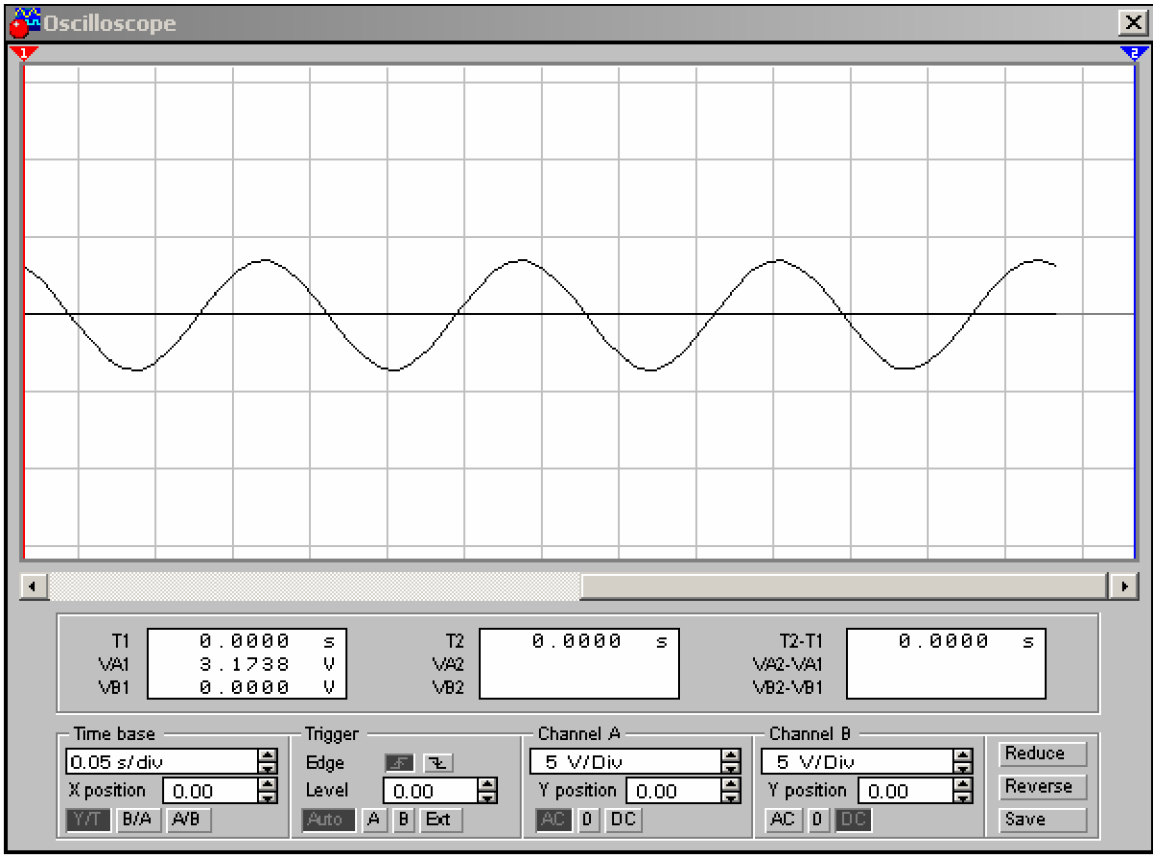
```
                    LcdInstructionWrite(Y_ADRESS);            /* Set column to
0 */

                    /* process a page on both sides */
                    for (Column = 0; Column < 128; Column++)
              {
                    if (Column == 64)
                                {
                                LcdSelectSide(RIGHT);   /* Select right side
*/
                                LcdInstructionWrite(X_ADRESS | Page); /* Set
the page number */
                                LcdInstructionWrite(Y_ADRESS); /* Set column
to 0 */
                                      }
                    LcdDataWrite (0x00);          /* erase a column */
              }
        }
}
```

3.2 Hardware

3.2.1 Development Of The Hardware

In the hardware phase of the project there was a big problem.The ADC of the PIC wasn't able to convert the negative signal because there is not a negative reference input in PIC. The maximum input voltage for PIC is 5V.So firstly with a voltage divider the 5V AC max. input voltage must be decreased to 2.5 V AC.Then by adding 2.5V DC the input will oscillate around 5V to 0V.Now any input isready for execution.



Before addition of 2.5V DC

Figure 3.1

By using the circuit below we shifted the level of signal.Becareful that the capacitor here is very important .It is used to prevent the DC signal not to reach to the AC supply.

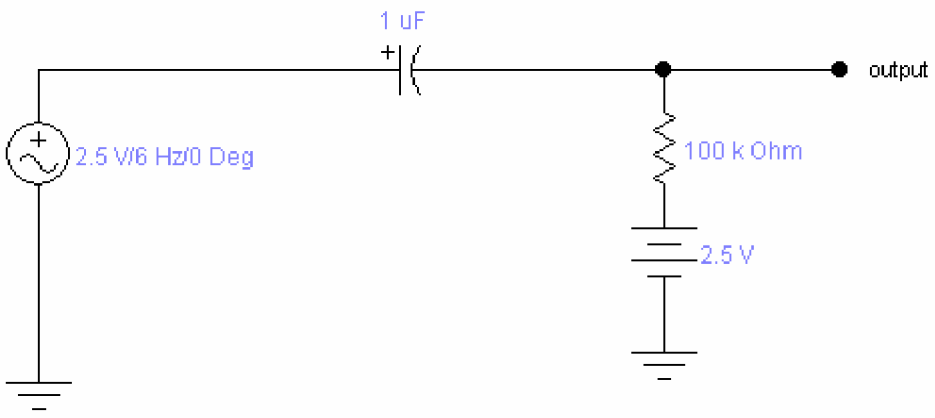
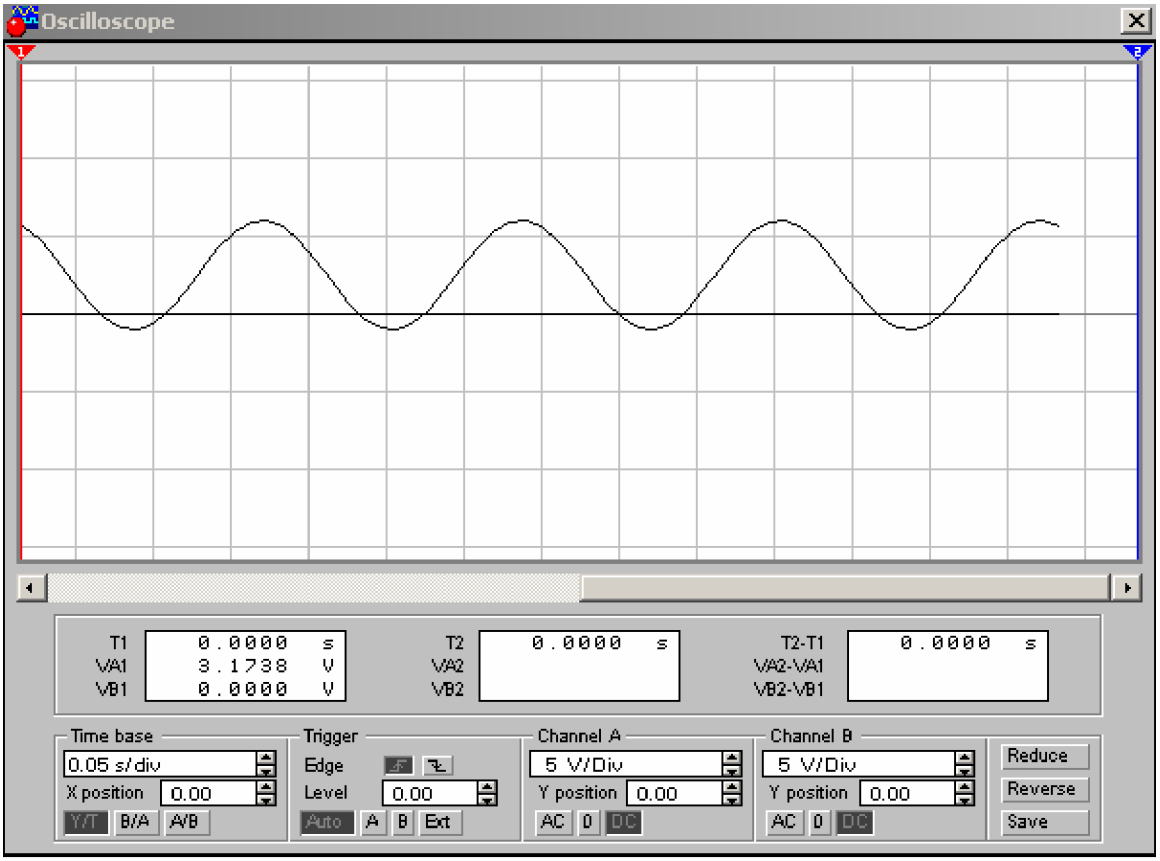


Figure 3.2



After addition of 2.5V DC

Figure 3.3

It can be seen in the schematic of the project that there is 4 100nF capacitprs.These are used randomly where I see important.

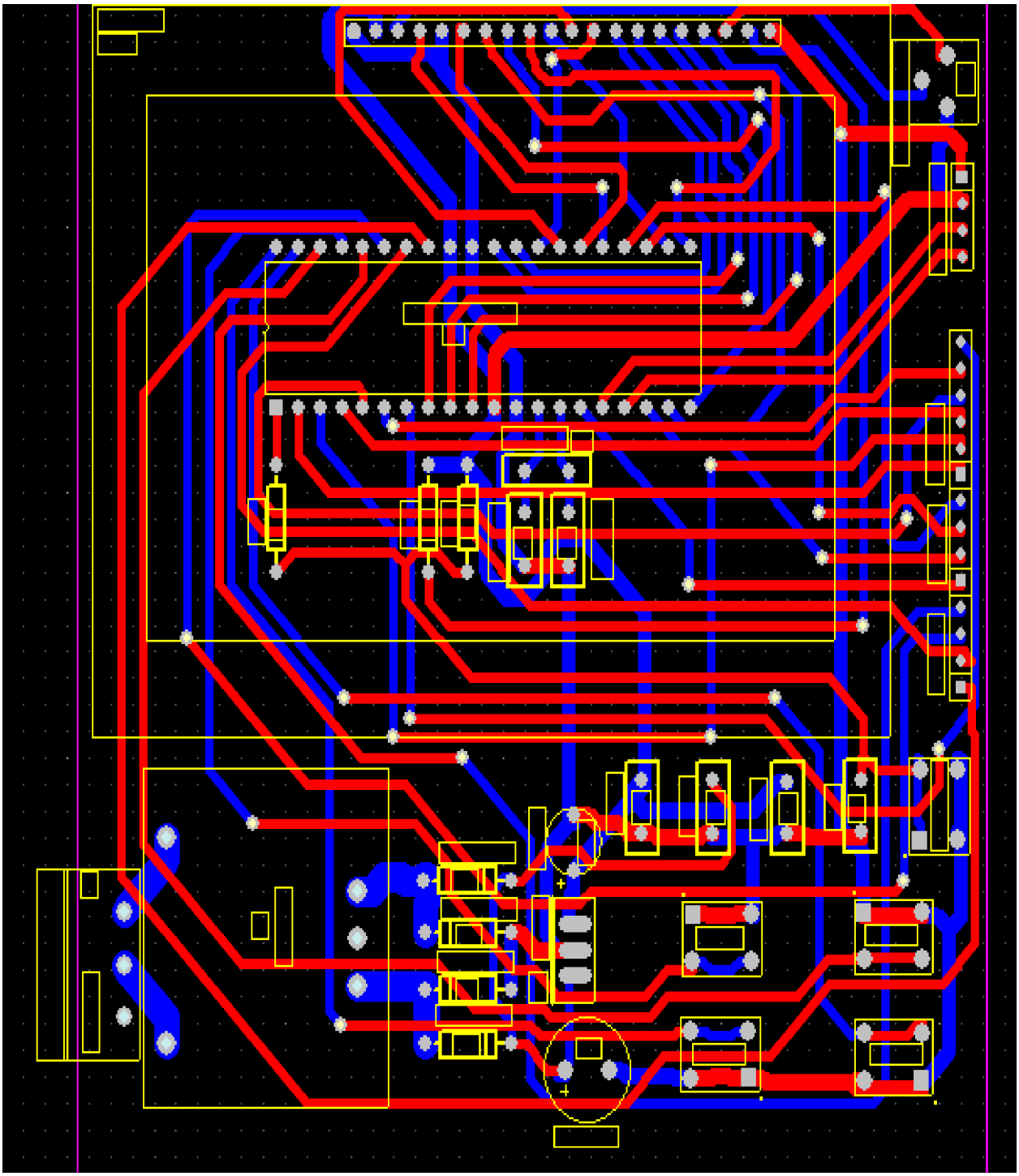
A bridge rectifier is used in the PCB.The transformator decreases 220V AC to 24V AC.

And finally 7805 IC which guaranties 5V DC at the output is used to feed the PIC and LCD.For compatible use of PIC(it has to be uninstalled for programming) a base for PIC is used.So you can install it on to the base and easilly uninstall.Also a 20 pin connector was used for LCD with the same purpose.There are also a variaty of inputs on PCB .Because this PCB is not designed for only spectrum analyzing. It can be used for lots of purposes you want.For example by using the ADC pin inputs it can be converted into a digital oscilloscope.

Also there si a variaty of buttons on PCB.These are connected to B Ports of the PIC.So by using the Port_B interrupt.

PDF-XCHANGE
Click to buy NOW!
www.docu-track.com

3.2.2 PCB Of The Analyzer



PDF-XCHANGE Click to buy NOW! www.docu-track.com

3.2.4 List Of Components

- PIC 18F452 Microcontroller
- YMS12864-02 Graphic LCD
- 1x7805 Regulator IC
- 2x 100uF capacitor
- 4x 100nF capacitor
- 4x 1N4007 general purpose diod
- 220/24 transformator
- 100k potentiometer
- 4 button
- 20 MHz oscillator
- 2x 22nF capacitor
- 10k resistor
- 27k resistor
- 100 ohm resistor
- 1x junction for power input
- Some connectors for general purpose input to the circuit

4.CONCLUSION

In this project it is desired to design a spectrum analyzer with microcontroller and graphic LCD.There had been some problems during the design of the project.The algorithms which were working on Turbo C compiler were not working on PICC compiler.So I have find some different solutions for that.The variables that you define are very important.The length and the type of values must be cosidered.There is no need in Turbo C.

There is no problem in the hardware of this project.This PCB can be used as a digital oscilloscope too.If there is a need for a DFT which requires more points then 32 (up to 128) the required changes should be done on the prototype program.At the end of the design phase of the project the device is working now.

References

PDF-XCHANGE Click to buy NOW! www.docu-track.com

Fast Fourier Transforms

Walker, J.S.

CRC Press. 1996

Fast Fourier Transforms: Algorithms

Elliot, D.F. and Rao, K.R.

Academic Press, New York, 1982

Electronic Instrumentation and Measurements

David A. Bell

Lampton Collage of Applied Arts and

Tecnology.Ontario,1994

Signals and Systems

Hwei P. HSU

Fairleigh Dickinson University

Shaum's Outline Series ,1995

LIST OF ABBREVIATIONS

PCB: Printed Circuit Board

DFT: Dicreate Fourier Transform

FFT: Fast Fourier Transform

LCD: Liquid Crystal Display

AUTOBIOGRAPHY

Adem Burak Çakmak was born in Gaziantep in 1982.He graduate from Devrek Anatolian High School in 2000.In the same year he started to his bachelor degree in Istanbul University on Electronics Engineering.He has designed some projects in the R&D department of INTER Electronics.He is also interested in C#,Visual C++ and studying on software in his free time. He is stil studying in Istanbul University at fourth class.