

2-10-1995

Design of a Digital Compensation Filter

Nader Fakhry
Portland State University

Follow this and additional works at: https://pdxscholar.library.pdx.edu/open_access_etds



Part of the [Electrical and Computer Engineering Commons](#)

Let us know how access to this document benefits you.

Recommended Citation

Fakhry, Nader, "Design of a Digital Compensation Filter" (1995). *Dissertations and Theses*. Paper 4961.
<https://doi.org/10.15760/etd.6837>


This Thesis is brought to you for free and open access. It has been accepted for inclusion in Dissertations and Theses by an authorized administrator of PDXScholar. Please contact us if we can make this document more accessible: pdxscholar@pdx.edu.


THESIS APPROVAL

The abstract and thesis of Nader Fakhry for the Master of Science in Electrical and Computer Engineering were presented February 10, 1995, and accepted by the thesis committee and the department.


COMMITTEE APPROVALS:


Yih-Chyün JENQ, CHAIR



W. Robert DAASCH


R. CRITTENDEN
Representative of the Office of
Graduate Studies

DEPARTMENT APPROVAL:


R. SCHAUMANN, CHAIR
Department Of Engineering

ACCEPTED FOR PORTLAND STATE UNIVERSITY BY THE LIBRARY

by  on 7 May 1995

ABSTRACT

An abstract of the thesis of Nader Fakhry for the Master of Science in Electrical and Computer Engineering presented February 10, 1995.

Title: Design Of A Digital Compensation Filter

The 24-bit Motorola DSP56001 processor will be used in combination with the DSP56ADC16 and the PCM-56 to design a good FIR compensation filter. Our objective is to digitize the input analog signal, and to compensate for the attenuation in the magnitude response of the digital sine wave.

Two different experiments will be conducted, a hands on approach, and a simulation program.

The first one will be realized directly, using the DSP system. We will determine the magnitude response of the system, and then deduce the coefficients of the FIR $\sin(x)/x$ filter. A look up table will store those values which will be fetched by the DSP program. With a minimum set of instructions we will generate a new digital output sequence after a N-point circular convolution is performed. The output signal is a good reconstruction of the input signal at frequencies below 22 Khz.

However, a second experiment will be needed to improve

this FIR $\sin(x)/x$ compensation filter, because we are not able to go beyond a 300-point impulse sequence. After that value (300-point), the time that each value is read and is ready to be processed by the DSP56001 becomes smaller than the time each instruction in the DSP program is executed and written to the PCM-56 via the SSI register. To be able to expand our experiment, we need to write a simulation program.

A simulation program of the previous experiment, which take as input the measured magnitude response of the system. The challenge will be to find ways to map the frequency domain, by using the maximum value of each linear convolution sequence, with a finite input sequence.

A step by step approach will be drawn until our final objective is reached. Our final step will be, to increase the number of sampling point in the frequency domain and will be to demonstrate that the result of the simulated program value will coincide with our objective, which is to compensate for the attenuation of the magnitude response of the system. By increasing the sampling frequency we will eventually obtain a good compensation filter.

DESIGN OF A DIGITAL COMPENSATION FILTER

by

NADER FAKHRY

*A thesis submitted in partial fulfillment of the
requirements for the degree of*

**MASTER OF SCIENCE
IN
ELECTRICAL AND COMPUTER ENGINEERING**

Portland State University
1995

ACKNOWLEDGEMENT

I would like to thank my dear parents, Mr. Kamel and Mrs. Aida FAKHRY for their encouragement and their kindness toward me. I believe, I would have never been able to finish my education without their overwhelming support.

Also, I want to give a special thank to my advisor, Dr. Yih-Chyun JENQ, who has continually guided me and shown me new insight into Digital Signal Processing. Dr. Yih-Chyun JENQ has always assisted me during my research work, by suggesting ways to solve the problems that I was faced with.

In addition, I wish to thank Dr. Robert DAASCH and Dr. Richard CRITTENDEN for their help and for their time in reviewing my thesis. I like to acknowledge Ms. Shirley CLARK for her assistance.

Nader FAKHRY

TABLE OF CONTENTS

	PAGE
ACKNOWLEDGEMENTS	ii
LIST OF TABLES	vi
LIST OF FIGURES	vii
 CHAPTER	
I INTRODUCTION	1
II FUNDAMENTAL OF DIGITAL SIGNAL PROCESSING	3
2.a Digitization	3
2.a.1 Ideal Sampling	6
2.a.2 Sampling Rate Selection	7
2.a.3 Uniform Sampling Theorem	8
2.a.4 Discrete-Time Signals	10
2.a.5 The DTFT	12
2.b Filter Fundamentals	13
2.b.1 Low Pass Filters	13
III INTRODUCTION TO THE DSP56001 AND THE ADS	18
3.a DSP56001 Processors	18
3.a.1 Architecture	19
3.a.2 Speed	21
3.a.3 Precision	21

3.a.4	Instruction	23
3.b	General Description Of The ADS	25
3.c	Implementing And Design Of FIR	25
3.c.1	FIR Frequency Response	26
3.d	Implementing An FIR Filter	27
3.d.1	FIR Filter On DSP56001	27
3.d.2	DSP56001 Instructions	30
IV	DIGITAL SIGNAL PROCESSING SYSTEMS	39
4.a	DSP System	39
4.b	DSP56ADC16EVB	41
4.c	DSP56001 SSI Port Pins	42
4.d	Setting Up The SSI Port	43
4.d.1	SSI Port Selection	44
4.d.2	SSI CRA Register	44
4.d.3	SSI CRB Register	45
4.d.4	SSI Status Register	47
4.d.5	SSI Receive Register	47
4.d.6	SSI Transmit Register	47
4.d.7	RX And TX Registers	48
V	RECONSTRUCTION OF ANALOG SIGNAL	50
5.a	Process To Design The Compensation Filter	50
5.a.1	Overview Of The DSP Program	51
5.a.2	Algorithm	51
5.b	The Compensation Filter	53
5.b.1	Measured Values	53

5.b.2	Design Of The Compensated Filter	53
VI	SIMULATION PROGRAM	59
6.a	Overview Of The Program.	59
6.a.1	Digitization And Shift	61
6.a.2	Convolution Algorithm	62
6.a.3	Inverse DFT	63
6.a.4	Main Program	64
6.b	Choice Of The Input Data	66
6.b.1	Introduction	66
6.b.2	Program Constraints	67
6.c	The Compensation Filter	79
6.c.1	Discontinuity At High Frequency.	79
VII	CONCLUSION	84
	REFERENCES.90

LIST OF TABLES

TABLE	PAGE
1. Measured magnitude response of the digital sine wave	54
2. Measured magnitude response of the digital sine after it has been filtered	56
3. Magnitude of the, inverted values and DFT of sin(x)/x filter	81
4. Maximum peaks	82

LIST OF FIGURES

FIGURE	PAGE
1. An analog signal and 3 different type of sampling .	4
2. Quantized signal	5
3. Ideal sampling	6
4. Spectrum of an ideally sampled signal	7
5. Aliasing due to overlap of spectral images	8
6. Spectrum of ideal sampled signal	9
7. Sampling with Dirac and Kronecker impulses	11
8. Ideal filter responses	14
9. Monotonic magnitude response	15
10. Magnitude response with ripples in the passband .	16
11. Magnitude response with ripples in the stopband .	16
12. Magnitude response with ripples in the passband and stopband	17
13. DSP56001 block diagram	23
14. Application development system	24
15. Assembler instructions to implement an FIR filter	32
16. Memory map at the beginning of the n iteration .	33
17. Memory map after the CLR instruction	34
18. Memory map after the execution of the MAC instruction	35

19. Memory map after the execution of the 2 nd MAC instruction	36
20. Memory map after the execution of the last MAC instruction	37
21. Memory map after the execution of the MACR instruction	38
22. DSP system	39
23. EVB block diagram	40
24. SSI pins	43
25. Development process to generate .lod file	52
26. A/D D/A magnitude response	56
27. 124-point $\sin(x)/x$ impulse response	57
28. Compensated magnitude response	57
29. Magnitude responses	67
30. Magnitude response with no phase shift	69
31. Magnitude response with a phase shift of 15 degree	69
32. Magnitude response with a phase shift of 30 degree	70
33. Magnitude response with a phase shift of 45 degree	70
34. Magnitude response with a phase shift of 60 degree	71
35. Magnitude response with a phase shift of 75 degree	71

36. Magnitude response with a phase shift of 90 degree	72
37. Magnitude response of collected maximum	72
38. Sample points in the unit circle	74
39. Magnitude response of the average values	75
40. Convolution sequence with Gibbs' Phenomenon	76
41. Discrepancies for particular phase shift	77
42. Successful correction of the magnitude response	78
43. 496-point $\sin(x)/x$ impulse response	79
44. Inverted magnitude values	80
45. Magnitude response of the compensated values	83

CHAPTER I

INTRODUCTION

In a typical Digital Signal Processing System (DSP), the analog signal is sampled and digitized to produce a digital sample. The digitized signal is processed through a DSP program. Finally, the digital signal is converted back to an analog signal by an D/A converter and a reconstruction filter. In the process of digitization (A/D) and of reconstruction (D/A) of the input signal some discrepancies occur, which in turn introduces some distortion in the signal. The purpose of this digital compensation filter is to compensate for the drop off in the magnitude response of the A/D and D/A converters.

This thesis studies an implementation of a digital compensation filter. It is organized as follows:

- 1) In Chapter II, we introduce some fundamental concept of digital signal processing such as Digitization, Aliasing, and Discrete Time Fourier Transform.
- 2) In Chapter III, the Motorola DSP56001 and the Application Development System (ADS) is described.
- 3) In Chapter IV, the digital signal processing system which consists of an analog-to-digital, a digital-to-analog converters, a DSP56001 processor, and a DSP56000ADS Application Development System is

introduced.

- 4) In Chapter V, the reconstruction of the analog signal and the compensation of the magnitude response up to 22 Khz is shown.
- 5) In Chapter VI, the design criterion for the algorithm are considered as well as the implementation of the algorithm, are shown.
- 6) In Chapter VII, the conclusion of the thesis is given.

CHAPTER II

FUNDAMENTAL OF DIGITAL SIGNAL PROCESSING

Digital signal processing (DSP) is based on the fact that an analog signal can be digitized and input to a general-purpose digital computer or special-purpose digital processor. Once this is accomplished, we are free to perform all sorts of mathematical operations on the sequence of digital data samples inside the processor. Some of these operations are simply digital versions of classical analog techniques, while others have no counterpart in analog circuit devices or processing methods.

2.a DIGITIZATION

Digitization is the process of converting an analog signal such a time-varying voltage or current into a sequence of digital values. Digitization actually involves two distinct parts -sampling and quantization- which are usually analyzed separately for the sake of convenience and simplicity. Three basic types of sampling, shown in Figure 1, are ideal, instantaneous, and natural. From the illustration we can see that the sampling process converts a signal that is defined over a continuous time interval into a signal that has non zero amplitude values only at discrete instants of time (as in ideal sampling) or over a number of discretely

separate by internally continuous subintervals of time (as in instantaneous and natural sampling). The signal that results from a sampling process is called a sampled-data signal. The signals resulting from ideal sampling are also referred to as discrete-time signals.

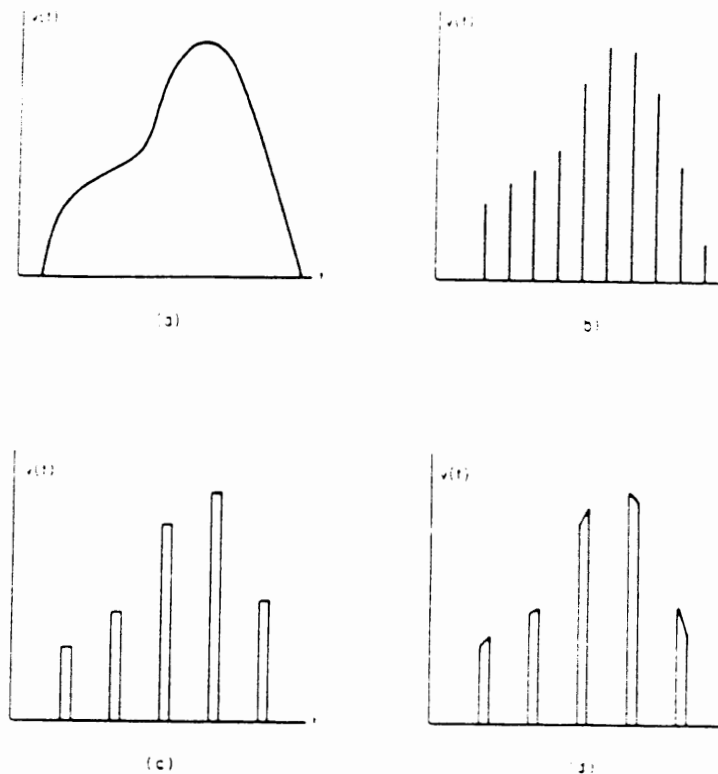


Figure 1. An analog signal (a) and three different types of sampling: (b) ideal, (c) instantaneous, and (d) natural.

Each of the three basic sampling types occurs at different places within a DSP system. The output from a Sample and Hold amplifier or digital-to-analog converter (DAC) is an instantaneously sampled signal. In the output of a practical analog-to-digital converter (ADC) used to sample

a signal, each sample will of course exist for some nonzero interval of time. However, within the software of the digital processor, these values can still be interpreted as the amplitudes for a sequence of ideal samples. In fact, this is almost always the best approach since the ideal sampling model results in the simplest processing for most applications. Natural sampling is encountered in the analysis of the analog multiplexing that is often performed prior to A/D conversion in multiple-signal systems. In all three of the sampling approaches presented, the sample values are free to assume any appropriate value from the continuum of possible analog signal values.

Quantization is the part of digitization that is concerned with converting the amplitudes of an analog signal into values that can be represented by binary numbers having some finite number of bits. A quantized, or discrete-valued signal, signal is shown in Figure 2. The sampling and quantization process will introduce some significant changes

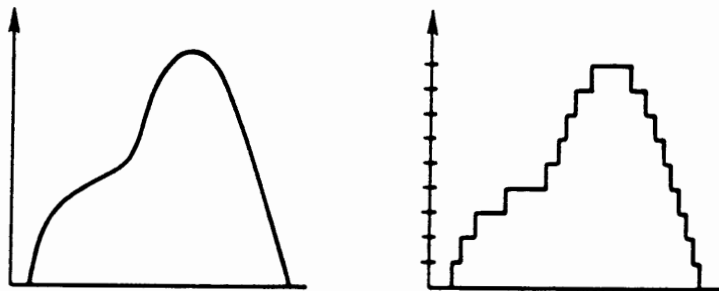


Figure 2. An analog signal (a) and the corresponding quantized signal (b).

in the spectrum of a digital signal. The details of the changes will depend upon both the precision of the quantization operation and the particular sampling model that most aptly fits the actual situation.

2.a.1 IDEAL SAMPLING

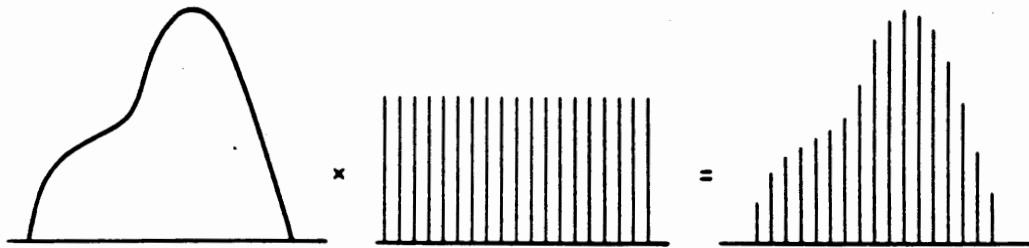


Figure 3. *Ideal sampling*

In ideal sampling, the sampled-data signal, as shown in Figure 3 comprises a sequence of uniformly spaced impulses, with the weight of each impulse equal to the amplitude of the analog signal at the corresponding instant in time. Although not mathematically rigorous, it is convenient to think of the sampled-data signal as the result of multiplying the analog signal $x(t)$ by the periodic train of unit impulses:

$$x_s(.) = x(t) \sum_{n=-\infty}^{n=\infty} \delta(t-nT) \quad (1)$$

The spectrum of the sampled-data signal could be obtained by convoluting the spectrum of the analog signal with the

spectrum of the impulse train:

$$\mathcal{F}[x(t) \sum_{n=-\infty}^{\infty} \delta(t-nT)] = X(f) * [f_s \sum_{m=-\infty}^{\infty} \delta(f-mf_s)] \quad (2)$$

As illustrated in Figure 4, this convolution produces copies, or image, of the original spectrum that are periodically repeated along the frequency axis. Each of the images is an exact (to within a scaling factor) copy of the original spectrum. The center-to-center spacing of the images is equal to the sampling rate f_s , and the edge-to-edge spacing is equal to $f_s - 2f_H$. As long as f_s is greater than 2 times f_H , the original signal can be recovered by a lowpass filtering operation that removes the extra images introduced by the sampling.

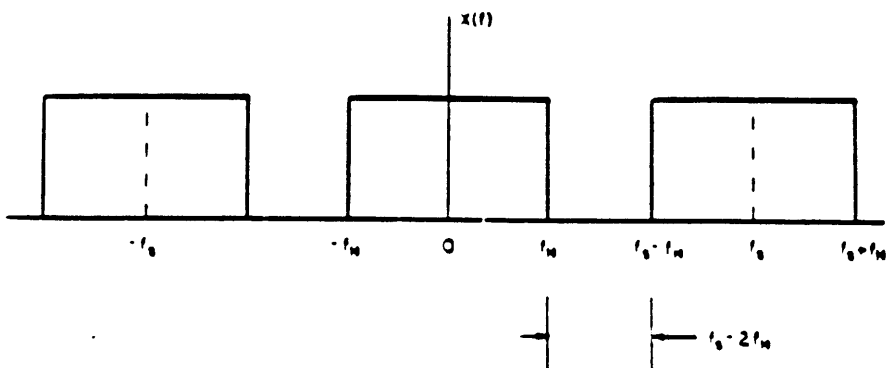


Figure 4. Spectrum of an ideally sampled signal

2.a.2 SAMPLING RATE SELECTION

If f_s is less than $2f_H$, the images will overlap, or alias, as shown in Figure 5, and recovery of the original signal will not be possible. The minimum alias-free sampling rate of $2f_H$ is called the Nyquist rate. A signal sampled exactly at its Nyquist rate is said to be critically sampled.

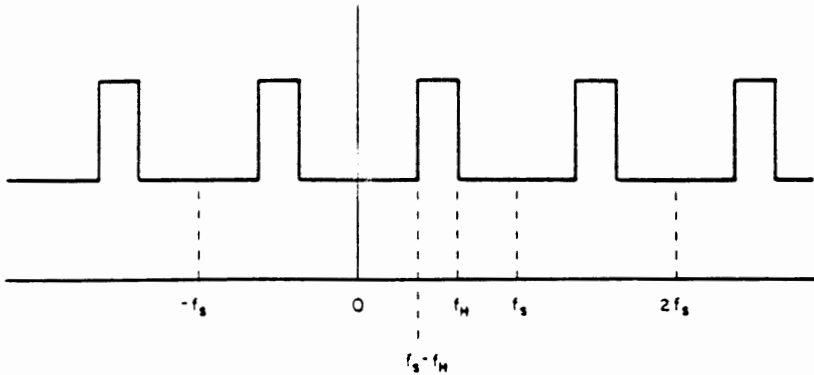


Figure 5. Aliasing due to overlap of spectral images

2.a.3 UNIFORM SAMPLING THEOREM

If the spectrum $X(f)$ of a function $x(t)$ vanishes beyond an upper frequency of f_H Hz or ω_H rad/s, then $x(t)$ can be completely determined by its values at uniform intervals of less than $1/(2f_H)$ or π/ω . If sampled within these constraints, the original function $x(t)$ can be reconstructed from the samples as follow:

$$x(t) = \sum_{n=-\infty}^{\infty} x(nT) \frac{\sin(2f_s(t-nT))}{2f_s(t-nT)} \quad (3)$$

Where T is the sampling interval.

Since practical signals cannot be strictly band limited,

sampling of a real-word signal must be performed at a rate greater than $2f_H$ where the signal is known to have negligible

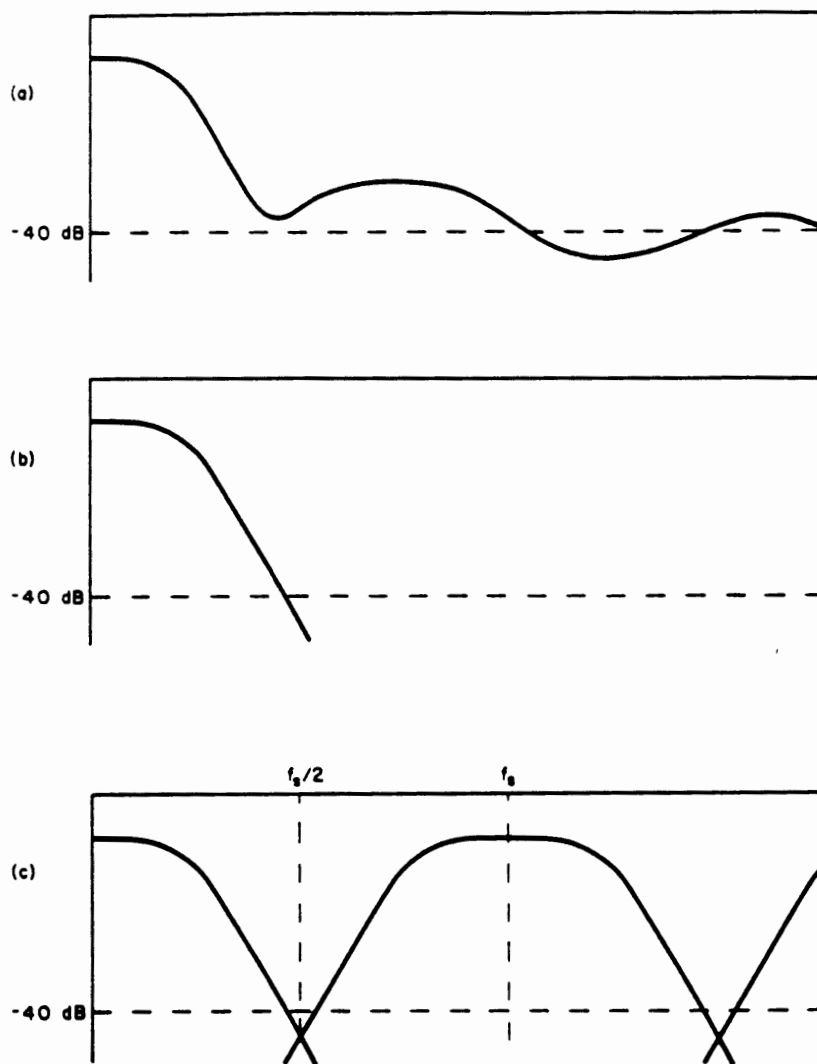


Figure 6. Spectrum of an ideally sampled practical signal: (a) spectrum of raw analog signal, (b) spectrum after lowpass filtering, and (c) spectrum after sampling.

(that is, typically less than 1 percent) spectral energy

above the frequency f_H . When designing a signal processing system, we will rarely, if ever have reliable information concerning the exact spectral occupancy of the noisy real-world signals that our system will eventually face. Consequently, in most practical design situation, a value is selected for f_H based upon the requirements of the particular application, and then the signal is low-pass-filtered prior to sampling. Filters used for this purpose are called antialiasing filters or guard filters. The sample-rate selection and guard filter design are coordinated so that the filter provides alternation of 40dB or more for all frequencies above $f_s/2$. The spectrum of an ideally sampled practical signal is shown in Figure 6. Although some aliasing does occur, the aliased components are suppressed at least 40dB below the desired components. Antialias filtering must be performed prior to sampling. In general, there is no way to eliminate aliasing once a signal has been improperly sampled. The particular type (Butterworth, Chebyshev, Bessel, Cauer, and so on) and order of the filter should be chosen to provide the necessary stop band attenuation while preserving the pass-band characteristic most important to the intended application.

2.a.4 DISCRETE-TIME SIGNALS

Once we are operating strictly within the digital or discrete-time realms, we can dispense with the Dirac delta

impulse and adopt in its place the unit sample function, which is much easier to work with. The unit sample function is also referred to as Kronecker delta impulse.

Figure 7 shows, both the Dirac delta and Kronecker delta representations for a typical signal. In the function sampled using a Dirac impulse train, the variable

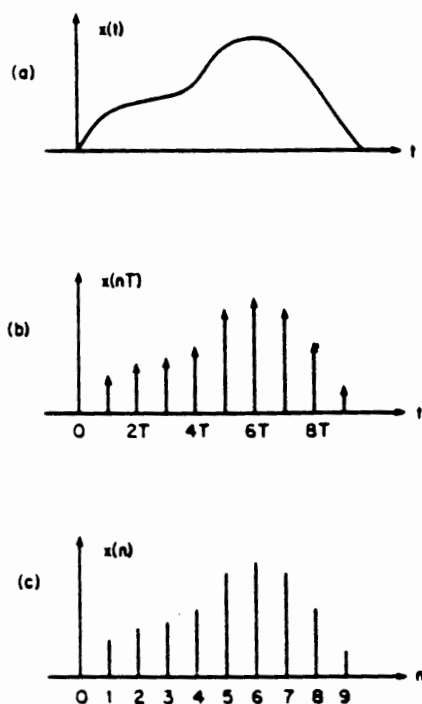


Figure 7. Sampling with Dirac and Kronecker impulses: (a) continuous signal, (b) sampling with Dirac impulses, and (c) sampling with Kronecker impulses.

is the continuous time t , and integer multiples of the sampling interval T are used to explicitly define the discrete sampling instants. On the other hand the Kronecker

delta notation assumes uniform sampling with an explicitly defined sampling interval. The independent variable is the integer-valued index n whose values correspond to the discrete instants at which interval is dispensed with completely by treating all the discrete-time functions as though they have been normalized by setting $T=1$.

2.a.5 DISCRETE-TIME FOURIER TRANSFORM

The Fourier series is given by equation 4 :

$$x(t) = \sum_{n=-\infty}^{\infty} X[n] e^{j2\pi n F t} \quad (4)$$

Where $F = 1/t_0 =$ Sample spacing in the frequency domain

$t_0 =$ period of $x(t)$

Likewise

$$X[n] = \frac{1}{t_0} \int_{t_0} x(t) e^{-jn2\pi F t} dt \quad (5)$$

The fact that the signal $x(t)$ and sequence $X[n]$ form a Fourier series pair with a frequency domain sampling interval of F can be indicated as:

$$x(t) \text{ <---}\mathcal{F}\text{---> } X[n]$$

Once we have defined a discrete-time sequence $x[n]$, the Discrete-Time Fourier Transform (DTFT) can be used to obtain the corresponding spectrum directly from the sequence without having to resort to impulses and continuous-time Fourier analysis.

The Discrete-Time Fourier Transform, which links the discrete-time and the continuous-frequency domain is defined by :

$$X(e^{j\omega T}) = \sum_{n=-\infty}^{\infty} x[n] e^{-j\omega nT} \quad (6)$$

and the corresponding inverse is given by :

$$x[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\omega}) e^{j\omega nT} d\omega \quad (7)$$

2.b FILTER FUNDAMENTALS

Ideal filters would have rectangular magnitude responses as shown in Figure 8. The desired frequencies are passed with no attenuation while the undesired frequencies are completely blocked. If such filters could be implemented, they would enjoy widespread use. Unfortunately, ideal filters are non causal and therefore not realizable. However, there are practical filter designs that approximate the ideal filter characteristics and which are realizable. Each of the major types Butterworth, Chebyshev, and Bessel optimizes a different aspect of the approximation.

2.b.1 MAGNITUDE RESPONSE FEATURES OF LOWPASS FILTERS

The magnitude response of a practical lowpass filter will usually have one of the four general shapes shown in Figure 9 through Figure 12. In all four cases the filter characteristics divide the spectrum into three general

regions as shown. The passband extends from direct current up to the cutoff frequency $2\omega_c$. The transition band extends from ω_c up to the stop band at ω_1 , and the stop band extends upward from ω_1 to infinity. The cutoff frequency ω_c is the frequency at which the amplitude response falls to a specified fraction (usually -3 dB, sometimes -1 dB)

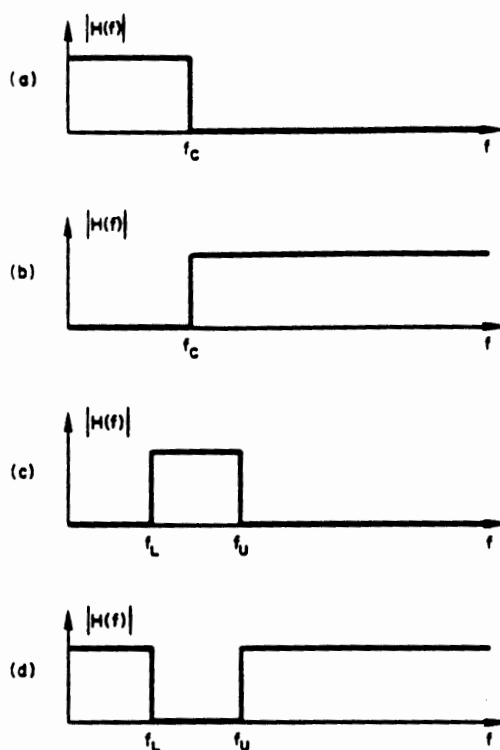


Figure 8. Ideal filter responses: (a) lowpass, (b) highpass, (c) bandpass, and (d) bandstop.

of the peak passband values. Defining the frequency ω_1 which marks the beginning of the stop band is not quite so straightforward. In Figure 9 or Figure 10 there really isn't any particular feature that indicates just where ω_1 should be

located. The usual approach involves specifying a minimum stop band loss α_2 (or conversely a maximum stop band amplitude A_2)

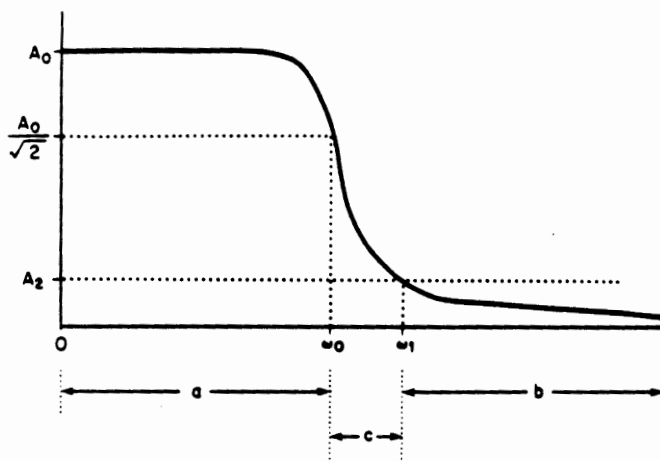


Figure 9. Monotonic magnitude response of a practical lowpass filter: (a) pass band, (b) stop band, and (c) transition band.

and then defining ω_1 as the lowest frequency at which the loss exceeds and subsequently continues to exceed α_2 . The width ω_T of the transition band is equal to $(\omega_c - \omega_1)$. The quantity ω_T/ω_c is sometimes called the normalized transition width. In the case of response shapes like those shown in

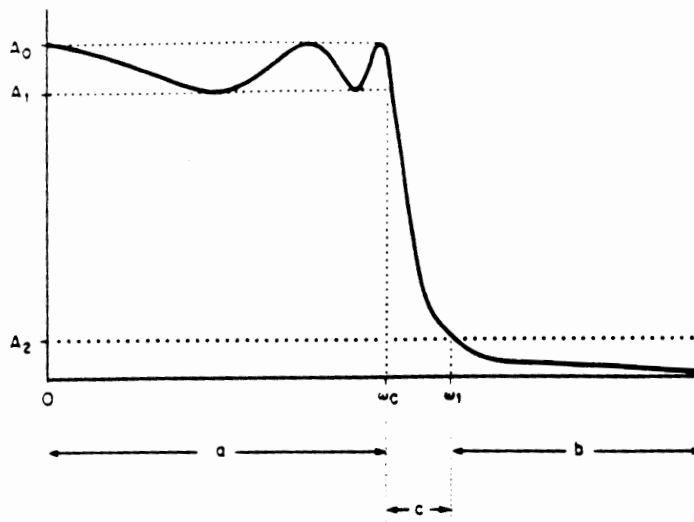


Figure 10. Magnitude response of a practical lowpass filter with ripples in the passband: (a) pass (b) stop (c) transition.

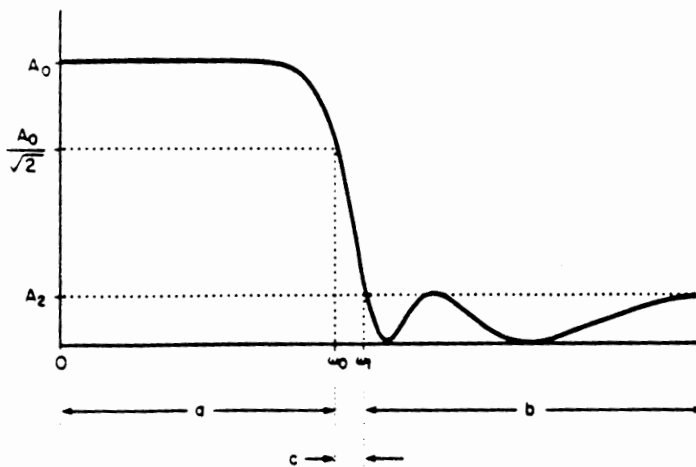


Figure 11. Magnitude response of a practical lowpass filter with ripples in the stop band: (a) pass band (b) stop band (c) transition band.

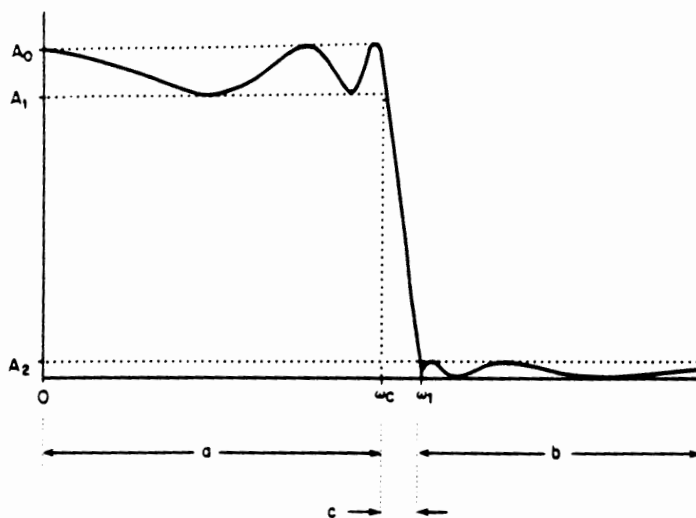


Figure 12. Magnitude response of a practical lowpass filter with ripples in the pass band and stop band: (a) pass band (b) stop band (c) transition band.

Figure 11 and 12, the minimum stop band loss is clearly defined by the peaks of the stop band ripples.

CHAPTER III

INTRODUCTION TO DSP56001 DIGITAL SIGNAL PROCESSORS AND THE DSP56001 ADS APPLICATION DEVELOPMENT SYSTEM

The DSP56K processor is design to be used with the ADS (Application Development System) for real time DSP systems. It is a general purpose and single chip Digital Signal Processors (DSP). In this chapter we will describe the DSP56K.

3.a DSP56001 PROCESSORS GENERAL DESCRIPTION

The DSP56001 processor is designed to be fast, and to be a general purpose DSP implemented with high density, low power, 5-volt HCMOS technology. The DSP56001 processor feature an on-chip program memory and two independent on-chip data memories which are part of a dual Harvard architecture. They are all externally expandable to 64K words (192 Kwords total) and accessible with zero wait states. This processor has 512 words of on-chip program RAM supported by an on-chip Bootstrap Loader.

The DSP56001 processor was not designed for a particular application but was designed to execute commonly used DSP benchmarks (such as Digital Filtering, Signal Processing, Data Processing, Modulation, Numeric Processing, Spectral Analysis) in a minimum time for a single-multiplier

architecture. Useful application combines several of those useful functions with other functions. DSP can duplicate almost any analog electronic circuitry. The advantages in doing so are becoming more and more wide spread since a DSP is faster and more cost effective.

3.a.1 HIGH PERFORMANCE ARCHITECTURE

The DSP56001 has been designed to maximize throughput in data intensive digital signal processor applications. The dual nature of the architecture facilitates writing software for DSP applications. For example, data is naturally partitioned into X and Y spaces for graphics and image-processing applications, into coefficient and data spaces, for filtering applications and into real and imaginary, for performing complex arithmetic. The DSP56001 processors use, a non-obtrusive three-stage instruction fetch/decode/execute pipeline, combined with three independent execution units connected by seven independent buses to three independent on-chip memory blocks, to provides the parallelism needed for high performance digital signal processing.

The major architectural components of the DSP56001 processors is shown in Figure 13 and include:

a) Three independent Execution Units:

- 1) The Data ALU

2) The Address Generation Unit

3) The Program Controller

b) Six on chip Memories:

DSP56001: One 512 word program RAM

One 256 word X-data RAM

One 256 word Y-Data RAM

Two 256 word programmed ROMs

One 32 word Bootstrap ROM

c) Four independent 24-bit Data buses:

1) The XD Data Bus,

2) The YD Data Bus,

3) The PD program Data Bus,

4) The GD Global Data Bus

e) One memory expansion Port A with:

1) 24 package pins assigned for 24-bit data words;

2) 16 package pins assigned for 16-bit, memory block linear addressing to 64 Kword;

3) 3 package pins assigned for segmented selected of a 64 Kword program memory block, a 64 Kword X-Data memory block, and or a 64 Kword Y-Data memory block.

4) 4 package pins for handshaking functions.

f) Three multimode on-chip peripherals:

1) One serial communication Interface or general purpose I/O port C.

2) One Synchronous Serial Interface or general purpose

I/O port C.

- 3) One parallel host interface or general purpose I/O port B.

g) One clock generator.

h) Input/Output:

- 1) Memory Expansion (Port A)
- 2) General-Purpose I/O (Ports B and C)
- 3) Host Interface
- 4) Serial Communication Interface (SCI)
- 5) Synchronous Serial Interface (SSI)

3.a.2 HIGH PERFORMANCE SPEED

At a clock rate of 20.48 MHz, the DSP56001 processor can execute 10.24 million instructions per second (MIPS) including up to 30.72 million concurrent arithmetic and dual data move operations per second (MOPS). At a clock rate of 27Mhz, this performance becomes 13.5 MIPS and 40.5 MOPS respectively. For example, at a 20.48 MHz clock rate the DSP56001 can execute a 1024 point complex fast fourier transform (FFT) with bit reversed data addressing in 3.39 milliseconds using 24-bit fixed point arithmetic; at 27Mhz this same FFT can be executed in 2.57 milliseconds.

3.a.3 HIGH PERFORMANCE PRECISION

The DSP56001 is organized around the registers of a central processor composed of three independent execution

units. The buses move data and instructions while instructions are being executed inside the execution units. Data movement on the chip occurs over four, bidirectional, 24-bit buses: the X data bus (XDB), the Y data bus (YDB), the program data bus (PDB), and the global data bus (GDB). The X and Y data buses can be a one 48-bit data bus by concatenation of XDB and YDB.

The 24-bit data buses provides 144 dB of data dynamic range, the 48-bit concatenated data buses provides 288 dB of data dynamic range. This data dynamic range is 50% greater than that provided by 16-bit DSP. This is sufficient for most real-world applications since the majority of data converters are 16 bits or less, and not greater than 24 bits. The data ALU registers may be read or written over the XDB and the YDB as 24- or 48-bit operand. The source operands for the data ALU, which may be 24, 48, or 56 bits, always originate from data ALU operations are stored in an accumulator. The 56-bit accumulator internal to the data ALU provides 336 dB of internal dynamic range so that no loss of precision will occur due to intermediate processing. This arrangement allows, for example, up to 256 multiply instructions to be executed without any loss of precision due to truncation or rounding. This capability is directly applicable, for example, to high precision and efficient implementation of Finite Impulse Response (FIR).

Data dynamic range is calculated as:

$$\text{Data dynamic range in dB} = 20 \log_{10}(2^{\text{number of bits}})$$

$$144 \text{ dB} = 20 \log_{10}(2^{24})$$

$$288 \text{ dB} = 20 \log_{10}(2^{48})$$

$$96 \text{ dB} = 20 \log_{10}(2^{16})$$

filters. The two 56-bit accumulators each provide 336 dB of data dynamic range.

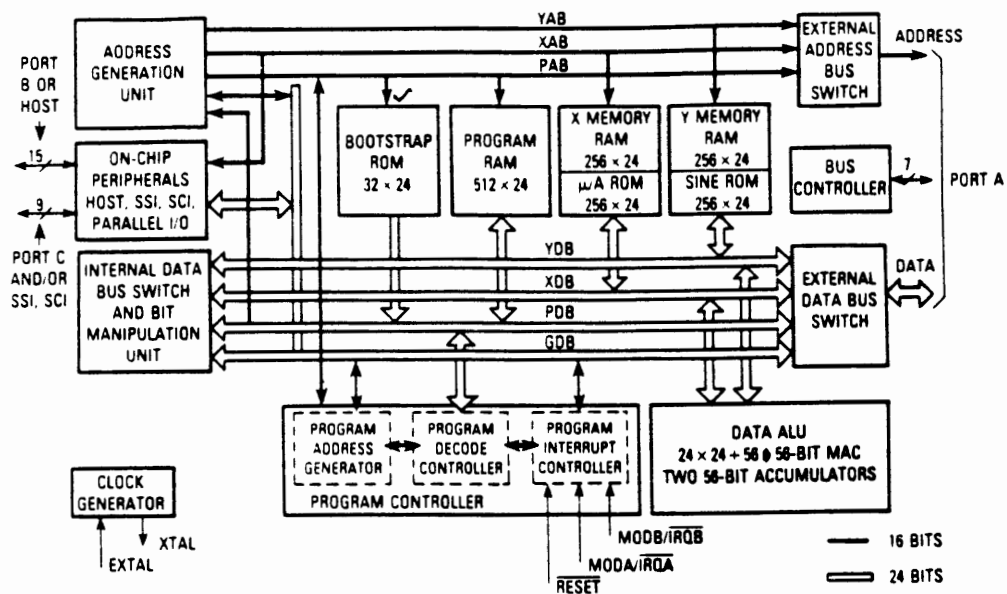


Figure 13. DSP56001 Block Diagram

3.a.4 HIGH PERFORMANCE INSTRUCTION SET

The objective of the instruction set is to provide capability to keep the ALU (Arithmetic Logic Unit) and the AGU (Address Generation Unit) busy with each instruction cycle, achieving maximum speed and minimum program size.

The DSP56001 instructions consist of -one or two 24-bit words- an operation word and an optional effective address

extension word. Four columns are used for each instruction syntax : opcode, operands, and two parallel-move fields. An operand sizes are defined as follows: a byte is 8 bits long, a short word is 16 bits long, a word is 24 bits long, along word is 48 bits long, and an accumulator is 56 bits long.

The instruction set includes 62 microprocessor - like mnemonics which make the transition from programming microprocessors to programming the DSP56001 processor easy.

The instruction set offers features which take full advantage of the duality of the architecture and at the same time yields a compact code. For example, a Repeat Next Instruction (REP) instruction is repeatable n-times. A single REP(n) instruction followed by a single multiply-accumulate (MACR) instruction (with two concurrent data move operations) allows an n-tap FIR filter algorithm to be compactly coded with only two instructions and executed in only $2(n+1)$ clock cycles.

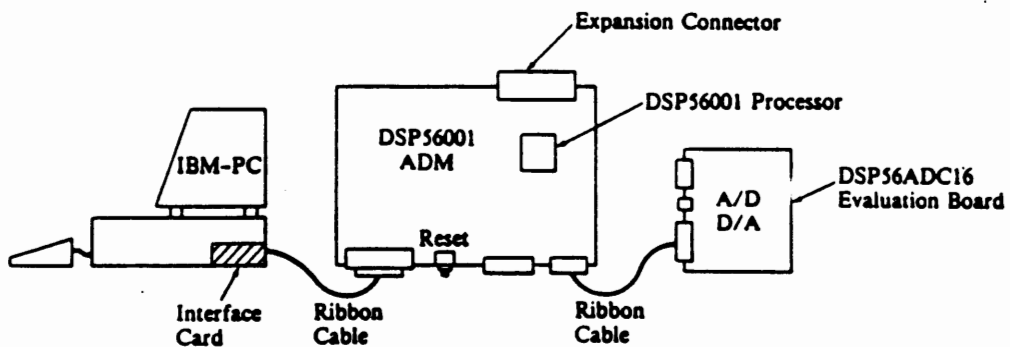


Figure 14. *Application Development System Component*

3.b GENERAL DESCRIPTION OF THE ADS

To design, debug and evaluate hardware tool such as the DSP56000 based systems, a DSP56001 ADS application development System (ADS) is used.

As shown in Figure 14, the ADS contains three components:

- 1) An ADS56000 software program which runs on the host platform. It interfaces with the user and controls the ADM.
- 2) A host interface board which is located into the back plane of the host platform and is hooked up to the ADM via a ribbon cable;
- 3) An application development module (ADM) board which consists of a DSP56001 processor, off-chip expansion memory, an interface and control circuitry, and several connectors for hook-up to application specific boards; The host platform between the user and the ADM uses the IBM-PC. Therefore, it is required that a host interface board which is compatible with the IBM PC's backplane and ADS56000 User Interface Software Program, be used.

3.c DESIGNING FIR FILTERS AND IMPLEMENTING THEM ON THE DSP56001 PROCESSOR

The FIR filter has three distinct properties it is stable, realizable, and can always be designed to have a linear phase response.

3.c.1 FIR FILTER FREQUENCY RESPONSE

The digital output sequence $y(n)$ is related to the FIR filter's digital input sequence $x(n)$ by the following equation:

$$y(n) = \sum_{i=0}^{N-1} b_i x(n-i) \quad (8)$$

Where b_i are filter coefficients and N is the number of those coefficients. The filter coefficients b_i provides the characteristics of the filter through which $x(n)$ is inputted to create the desired output sequence $y(n)$. The output sequence $y(n)$ of this nonrecursive filter (FIR filter) is a function of its current and past inputs. The z-transform transfer function $H(z)$ corresponding to equation 8 can be expressed as:

$$H(z) = \frac{Y(z)}{X(z)} = \sum_{i=0}^{N-1} b_i z^{-i} = \sum_{i=0}^{N-1} h(i) z^{-i} \quad (9)$$

Where $h(i)$ the impulse response of the FIR filter, is composed of b_i , the coefficients of the FIR filter.

To determine the frequency response of the FIR filter, the Z-transform variable z is evaluated on the unit circle as:

$$z = e^{j2\pi f} \quad (10)$$

where $f = \text{normalized frequency}$ (actual frequency divided by the sampling frequency)

By substituting equation (10) into (9) the frequency response of the FIR filter can be written as:

$$H(e^{j2\pi f}) = \sum_{i=0}^{N-1} h(i) e^{-j2\pi fi} \quad (11)$$

3.d IMPLEMENTING AN FIR FILTER

Equation 8 is used to implement an FIR filter. The following algorithm is performed to perform the convolution of the input sequence $x(n)$ and the impulse sequence b_n :

1. The initial value of the input sample $x(n)$ is stored.
2. The input sample $x(n)$ is multiplied by b_0 then, accumulated, the resulting products of the filter states $x(n-i)$ and the coefficients b_i to finally yield the output $y(n)$.
3. Shift the filter states to obtain the next output sample $y(n+1)$.

3.d.1 IMPLEMENTING AN FIR FILTER ON THE DSP56001 PROCESSOR

Step 1-3 of the previous section can be efficiently implemented on the DSP56001 processor by using its:

- a) Address pointers to mimic FIFO-like shifting of RAM data.
- b) Modulo modifier addressing capability ($Mn = \text{Modulus}-1$) to provide wrap-around data buffers.

- c) A signed Multiply-Accumulate (MAC) instruction which multiply two operands and adds the product to a third operand in a single instruction cycle.
- d) A parallel data move with the MAC instruction, to keep the multiplier running at full capability.
- e) Repeat Next Instruction (REP) instruction.

The DSP56001 processor's Address Generator Unit is composed of 8 Address Registers (Rn), 8 associated Offset Registers (Nn), and 8 associated Modulo Registers (Mn). The DSP56001 processor's capability to perform modulo addressing allows an Addressing Register (Rn) value to be incremented (or decremented) and yet remain within an address range of size L, where L is defined by a lower and an upper address boundary.

For the FIR filter, L is equal to the number of coefficients (taps). The value L-1 is stored in the DSP56001 processor's Modifier Register (Mn). Because of the manner in which the DSP56001 processor's modulo addressing mechanism is implemented, the lower address boundary of L (base address) must have zeros in its K LSBs, where $2^k \geq L$, e.g, the base address boundary is the sum of the lower address boundary (base address) plus the modulo size minus one, e.g, the base address value plus L-1. Note, the upper address boundary is not stored in a register.

When modulo addressing is used, the Address Register (Rn) points to a modulo (circular) data buffer located in X-memory and Y-memory. The address pointer (Rn) is not required to point at the lower address range L. If the address pointer increment post the upper address boundary (base address plus L-1 plus 1) it will wrap around to the base address.

3.d.1.a CIRCULAR CONVOLUTION

$$y[n] = \sum_{m=0}^{Ntaps-1} b[m] * x[(n-m)_N] \quad 0 \leq n \leq Ntaps-1 \quad (12)$$

In linear convolution, the computation of the sequence value $y[n]$ involves multiplying one sequence by a time-reversed and linearly shifted version of the other and then summing the values of the product $b_0 * x[n-m]$ over all m . To obtain successive values of the sequence representing the convolution, the two sequences are successively shifted relative to each other. In contrast, for the convolution as given by equation (12), the second sequence is circularly time reversed and circularly shifted with respect to the first. For this reason, the operation of combining two finite-length sequences according to equation (12) is called circular convolution. If both sequences are of the same length, then we can call it a N-point circular convolution.

To illustrate the circular convolution we can take two Ntaps point periodic sequences. To evaluate $y[n]$ for $n=3$, we

have to multiply b_m and $x[3-m]$ and then sum the product terms $b_m * x[3-m]$ for $0 \leq m \leq Ntaps - 1$ to obtain $y[n]$. As n changes, the sequence $x[n-m]$ shifts appropriately, and is evaluated for each value along that interval. This circular convolution of $Ntaps$ points will be used and illustrated in the next section.

3.d.2 DSP56001 INSTRUCTIONS FOR IMPLEMENTING AN FIR FILTER

To implement the $\sin(x)/x$ FIR filter, we take two sequences of same length, $Ntaps$. The input sequence $x[n]$ will be, the digitized sine wave, and b_n will be the coefficient of this filter. Since both have the same length we use an $Ntaps$ -point circular convolution. We have already seen that to convolute this two sequences we need to:

1. Multiply/Add the sequences
2. Rotate any one (since this operation is commutative) of the sequence.

The DSP56001 instructions shown in Figure 15 are used to implement the FIR filter algorithm where:

1. Modulo Register M0 will get the value of $Ntaps - 1$ (modulo NTAPS). Address Register R0 is set to point to the filter data shift register buffer located in X-memory. We will do the same, with Modulo Register M4 which get the value $Ntaps - 1$ and R4 point to the filter coefficients buffer located in Y-memory. We assume that

the program has been executing for some time, and is ready to process the input sample $x(n)$ in the Data ALU Input Register X0, the address in R4 is the base address (lower boundary) of the coefficient buffer. The address in R0 is M, where M is greater than or equal to the upper-boundary X-memory address. The contents of the DSP56001 processor's A-accumulator and Data ALU Input Registers X0 and Y0 are as shown in Figure 16.

2. The CLR instruction not only clear the A-Accumulator, it also moves the input sample $x(n)$ from the Data ALU's Input Register X0 to the X-memory location pointed to by Address Register R0 and transfers the first coefficient from the y-memory location pointed to be Address Register R4 to the Data ALU's Input Register Y0. R0 and R4 are also incremented at the end of this instruction, see Figure 17.
3. The REP loop instruction will executes the following MAC instruction $N_{\text{taps}}-1$ times:

$$\text{mac } x0, y0, a \quad x: (r0)+, x0 \quad y: (r4)+, y0$$
4. A circular convolution will be implemented using the MAC instruction by multiplying the filter state variable in X0 by the coefficient in Y0, and by adding the product to the A-Accumulator, and simultaneously do the same things The X-memory map for the filter states, the Y-memory map for the coefficient and the contents of the

first, second and last MAC instruction are as shown in Figure 18, 19 and 20 respectively.

5. The MACR instruction calculates the final tap of the filter algorithm, performs convergent rounding of the result, and simultaneously decrements the address. This FIFO like shifting of the filter state variables is accomplished by simply adjusting the R0 address pointer. The X-memory map for the filter states, the Y-memory map

```

;   FIR filter
;
;   x0   = input sample
;   a    = output sample
;   Ntaps = number of coefficient taps in the filter

      move #states ,r0      ; point to filter states
      move #ntaps-1,m0     ; mod(ntaps)
      move #coef   ,r4      ; point to filter
                           coefficients
      move #ntaps-1,m4     ; mod(ntaps)

clr   a          x0,x:(r0)+   y:(r4)+,y0
rep   #ntaps-1
mac   x0,y0,a    x:(r0)+,x0   y:(r4)+,y0
macr  x0,y0,a    (r0)-

```

Figure 15. *DSP56001 Assembler Instructions to implement the FIR filter Algorithm*

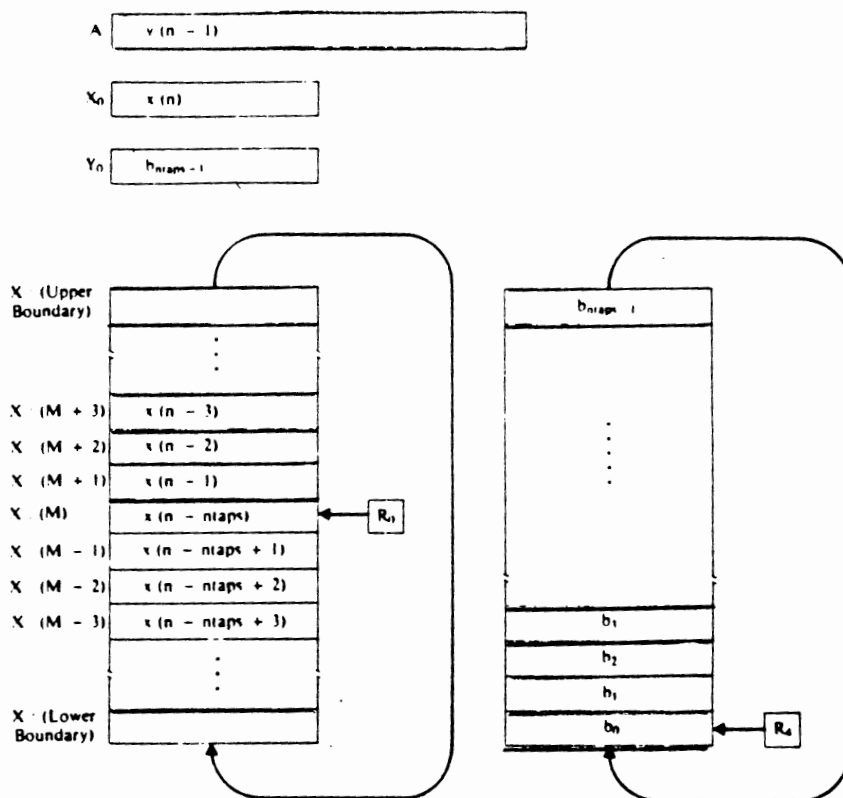


Figure 16. Memory Map and Data Registers at the Beginning of the n iteration.

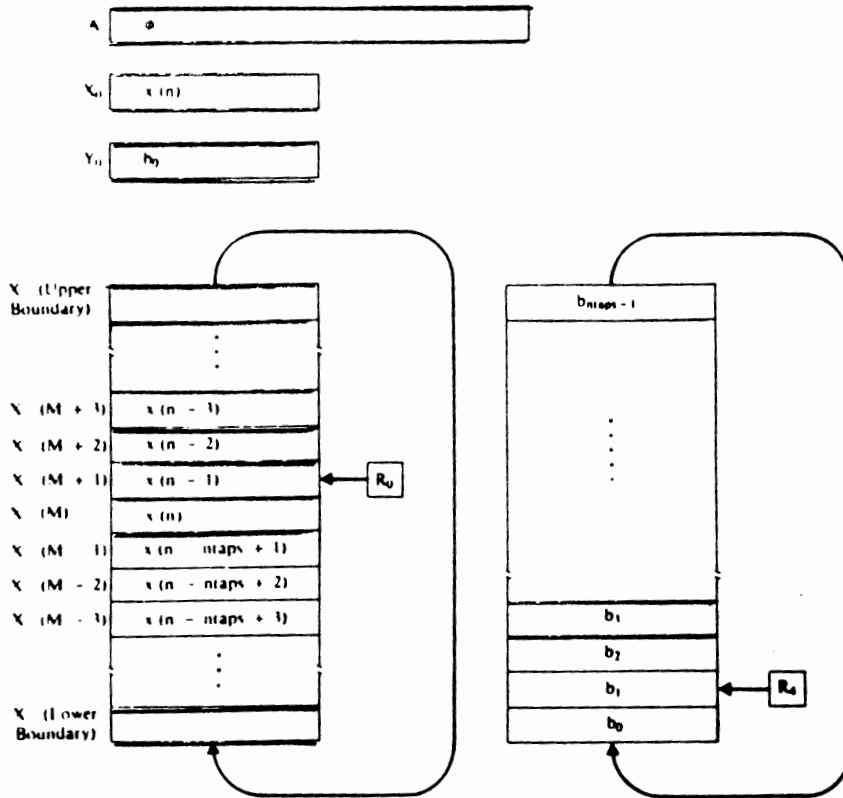


Figure 17. Memory Map and Data Registers after the CLR Instruction.

for the coefficients, and the contents of the A-Accumulator and Data ALU Input Registers X0 and Y0 after execution of the MACR instruction are as shown in Figure 21.

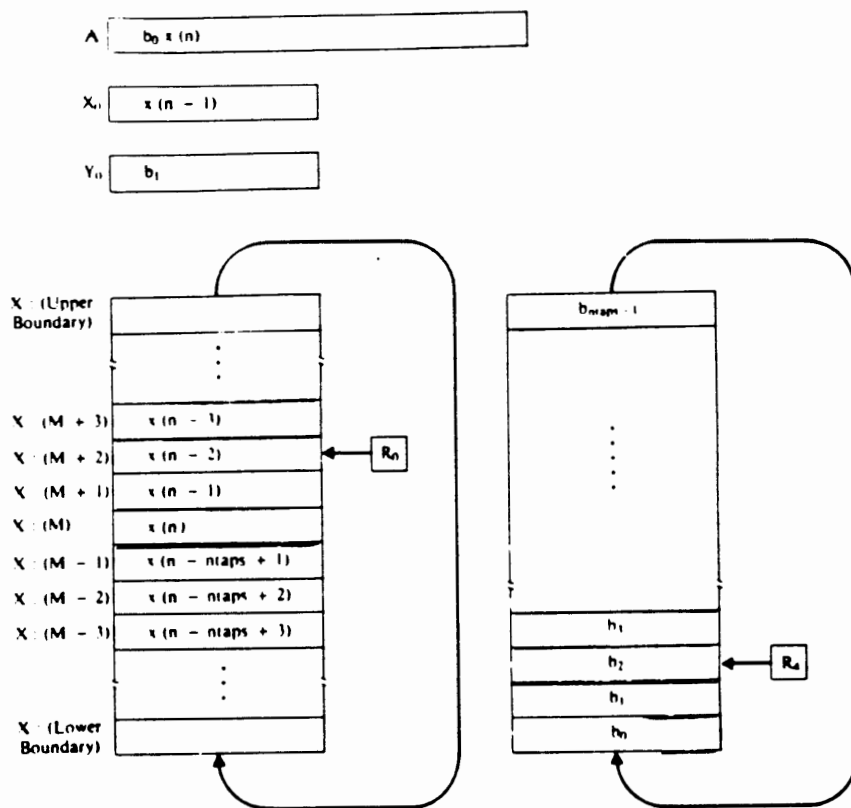


Figure 18. Memory Map and Data Registers after the execution of the First MAC instruction.

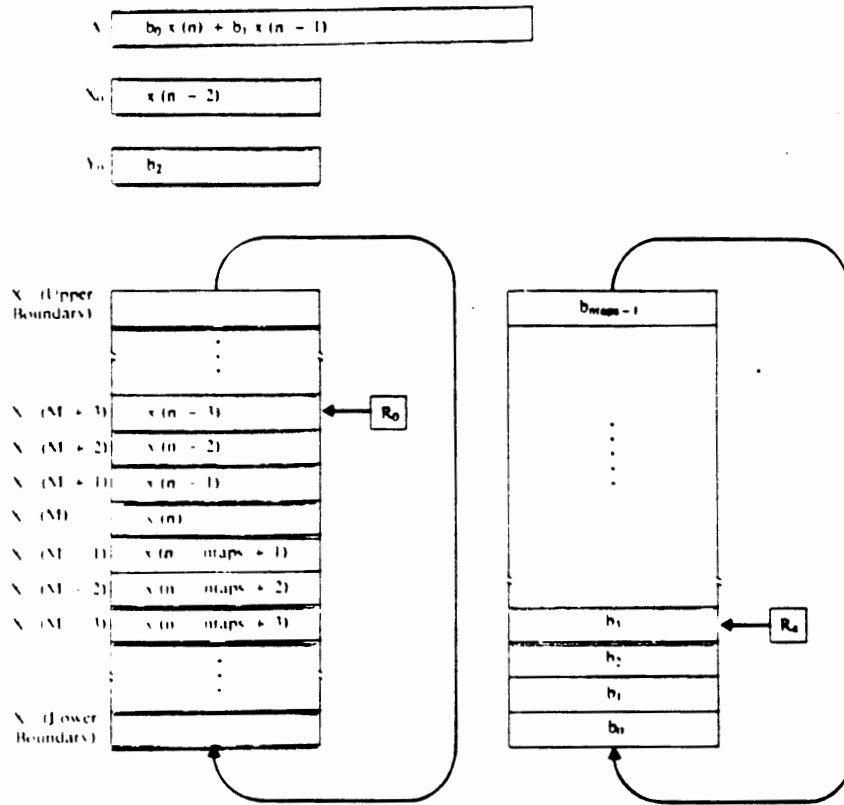


Figure 19. Memory Map and Data Registers after the Execution of the Second MAC Instruction.

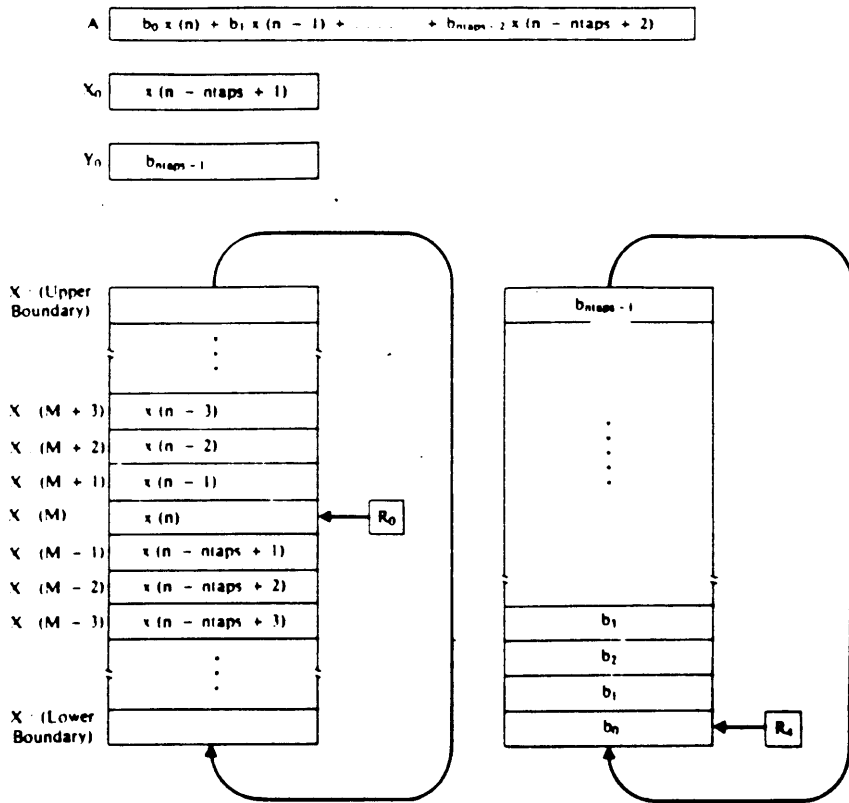


Figure 20. Memory Map and Data Registers after the execution of the Last MAC Instruction.

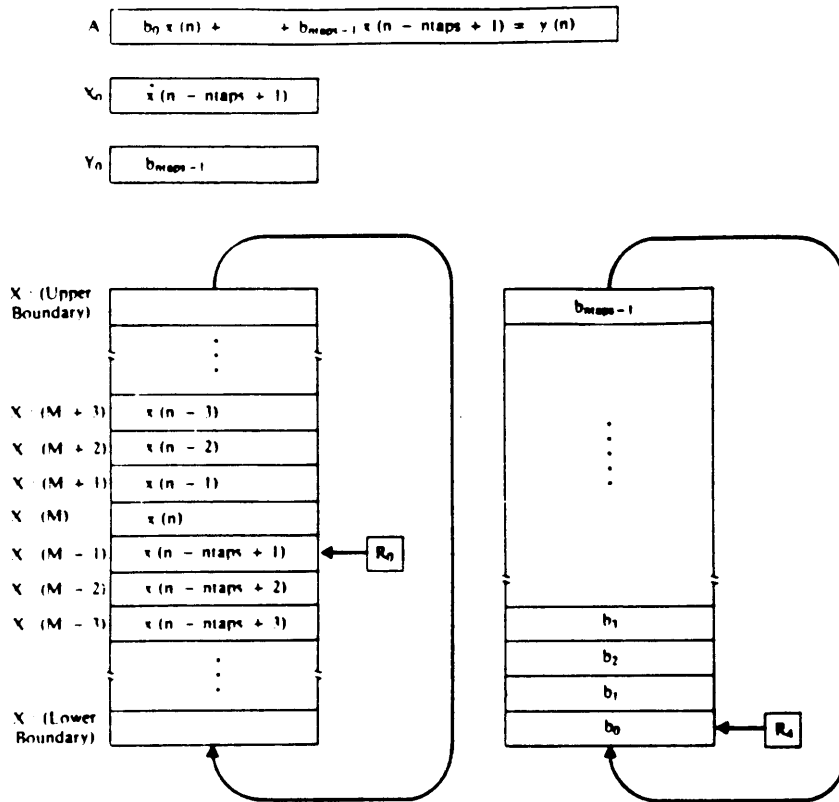


Figure 21. Memory Map and Data Registers after the execution of the MACR instruction.

CHAPTER IV

DIGITAL SIGNAL PROCESSING SYSTEMS

The DSP system consist of an analog-to-digital (A/D) converter, a DSP56001 processor, a digital-to-analog (D/A) converter. A DSP56000ADS Application Development System (ADS) is used to provide and control the DSP56001 processor; a DSP56ADC16EVB Evaluation Board (EVB) is used to provide and control the A/D converter and the D/A converter. The DSP56001 processor's Synchronous Serial Interface (SSI) port is used to accommodate serial data transfers from the D/A converter. In this chapter we will describe this DSP system that monitor the DSP56K.

4.a DSP SYSTEM

In the DSP System defined above and shown in Figure 22 and 23 an analog input signal is digitized by the A/D



Figure 22. DSP system.

converter on the EVB. The digitized signal is the output from

the A/D converter as a 16-bit serial data stream delimited by a Frame Sync Output (FSO) signal via the receive channel of the DSP56001 processor's SSI port, the digitized signal is received by the DSP56001 processor located on the Application Development Module (ADM) of the ADS. The DSP56001 processor

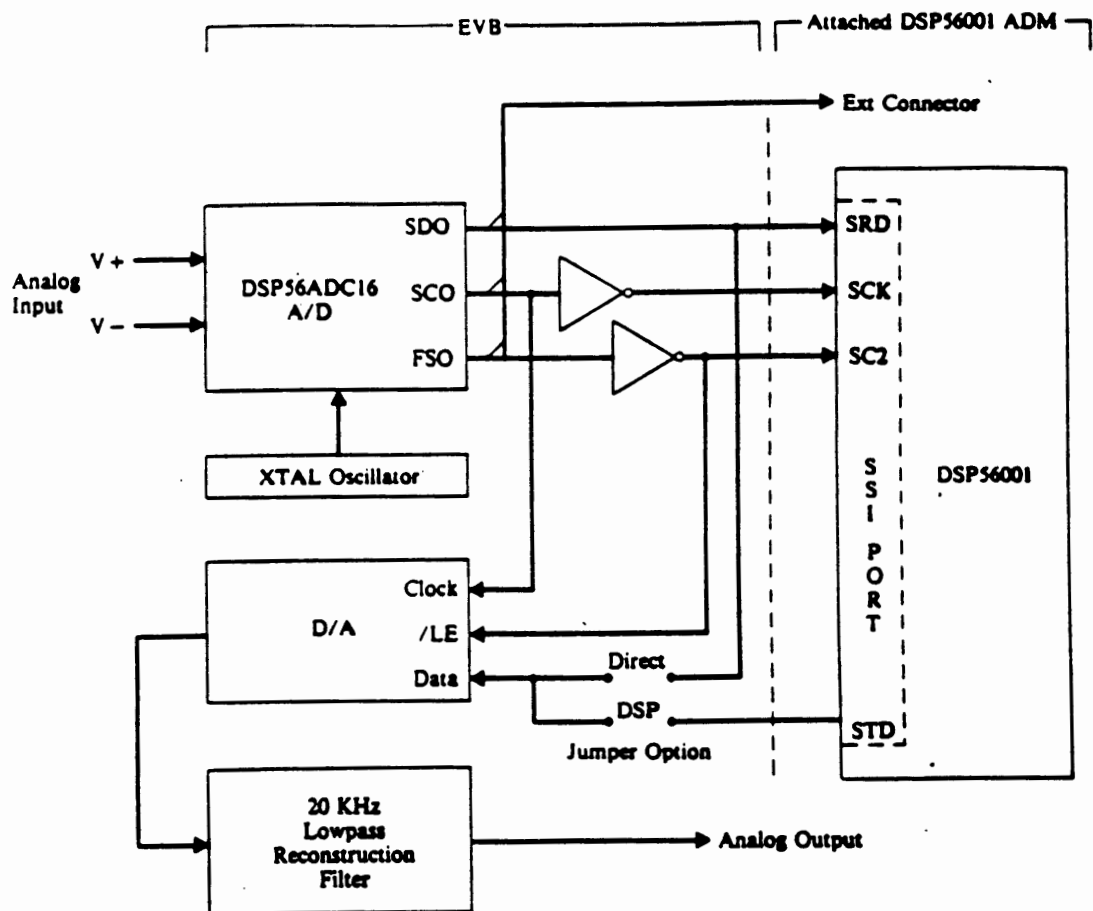


Figure 23. EVB block diagram.

routes the digitized signal from the receive channel of its SSI port to the transmit channel of its SSI port via the transmit channel of the DSP56001 processor's SSI port, the digitized signal is outputted from the DSP56001 processor as a 16-bit serial data stream and routed to the serial input of the D/A converter located on the EVB. The D/A converter constructs an analog output signal from the digitized input signal. A reconstruction filter smooths the analog signal output from the D/A converter. In order for the D/A converter to be able to properly reconstruct the analog signal which was originally input to the A/D converter, the Nyquist Sampling Theorem must first be satisfied. That is, the analog signal input to the A/D converter must be sampled by the A/D converter at a rate greater than twice its highest frequency component (f_h). Sampling at this frequency ($2f_h$) is known as sampling at the Nyquist Rate.

The output signal is a good reconstruction of the input signal at frequencies below 20 KHz. Since the Nyquist theorem is not contradicted ($2*f_h \leq 48 \text{ KHz}$) and the cut off frequency start at 20Khz.

4.b DSP56ADC16EVB EVALUATION BOARD (EVB)

The EVB is an A/D and D/A conversion system that can be used in configuration with the ADS. The major components of the EVB, shown in Figure 23 are:

1. One DSP56ADC16 16-bit, oversampling, Sigma-Delta A/D converter which can serially output 16 bit data samples at group rates up to 100 KHz per 16 bit sample,
2. One 16 bit D/A converter and
3. One low pass reconstruction filter with a cut off frequency of 20 KHz.

4.c DSP56001 PROCESSOR SSI PORT PINS

The SSI port has six dedicated pins as shown in Fig 24

1. Serial Transmit Data (STD)
2. Serial Receive Data (SRD)
3. Serial Clock (SCK)
4. Serial control (SC0)
5. Serial control (SC1)
6. Serial control (SC2)

As shown in Figure 23, the SSI port's STD, SRD, SCK, and SC2 pins are used to interface the DSP56001 processor on the ADM to the A/D converter and D/A converter on the EVB. The STD pin is used to output data from the SSI port's Serial Transmit Shift Register. This data is routed to the input of the D/A converter. The SRD pin is used to receive data into the SSI port's Receive Data Shift Register. This data transferred from the A/D converter. The SCK pin is used by the SSI port to accept an external clock. The source of this

external clock (inverted form) is the Serial Clock Output pin of the A/D converter.

This external clock (non-inverted form) also clocks the D/A converter. The SC2 pin is used by the SSI port to accept external frame sync. This external clock (non-inverted form) also clocks the D/A converter. The SC2 pin is used by the SSI port to accept external frame sync.

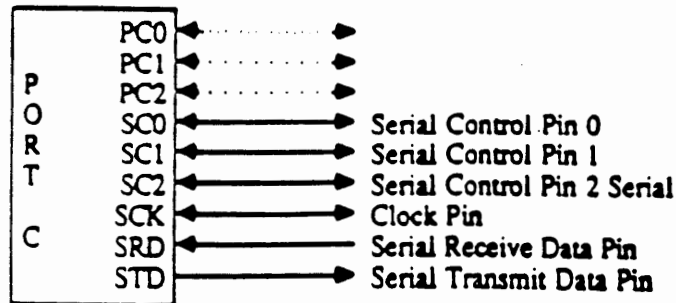


Figure 24. Synchronous Serial Interface Pins.

4.d SETTING UP THE DSP56001 PROCESSOR'S SSI PORT

Each of the Host, SCI, and SSI on chip peripherals have their own control, status, and data registers. The DSP56001 processor accesses each of these registers as if they were memory mapped I/O. These on chip memory mapped registers are accessed at X-Memory locations \$FFC0-\$FFFF.

4.d.1 SSI PORT SELECTION

In the DSP System, the SSI port of the DSP56001 processor which is located on the ADM is used to receive serial data from the A/D converter on the EVB and transmit serial data to the D/A converter on the EVB. The DSP56001 processor's port C Control Register (PCC) has 9 bits which, in a one-on-one fashion, individually select each of the 9 pins of port C to operate either as a general purpose I/O pin or as a SCI and/or as a SSI peripheral function pin. Port C pins can be configured as a general purpose I/O pins by clearing the corresponding PCC register bits; Port C pins can be configured a SCI and/or SSI function pins by setting the corresponding PCC register bits. The PCC register can be accessed at X-Memory location \$FFE1. The following DSP56001 processor instructions select Port C to operate as both a SCI port and a SSI port:

```
pcc      equ      $0001ff
          movep    #pcc,x:$ffe1; write PCC
```

4.d.2 SSI PORT CONTROL REGISTER A (CRA)

The operation of the SSI Port is directed by two 16-bit read/write Control Registers CRA and CRB. The CRA controls the SSI port's:

- a) Internal bit clock generator rate
- b) Internal frame divider rate, and
- c) Data word length

The data word length can be 8, 12, 16, or 24 bits. CRA's word Length Control bits 13 (WL0) and 14 (WL1).

In the DSP System, the SSI port's internal bit clock and frame generators are not used. The DSP56001 instructions below set the CRA to select a 16-bit word length. CRA occupies X-Memory location X:\$FFEC.

```
CRA      equ      $004000
          movep    #CRA,x:$FFEC
```

4.d.3 SSI PORT CONTROL REGISTER B (CRB)

The operation of the SSI Port is directed by two 16 bit read/write Control Registers CRA and CRB. CRB controls the SSI port's:

1. Multifunction pins SC2, SC1, and SC0 ;
2. Serial output flag control bits ;
3. Input versus output direction of pins SCK, SC2, SC1, and SC0 ;
4. Operating modes ;
5. Transmitter and receiver enables ; and
6. Transmitter and receiver interrupt enables.

CRB occupies X-Memory location \$FFED. The DSP56001 instructions shown below set up CRB where:

1. CRB bit 4 is cleared to select the direction of the SC2 pin to be an input (external frame sync) ;
2. CRB bit 5 is cleared to select the direction of

- the SCK pin to be an input (external clock for both the Transmit and Receive Shift Registers) ;
3. CRB bit 6 is cleared to select the Transmit Shift Register to transmit data MSB first and the Receive Shift Register to receive data MSB first;
 4. CRB bit 7 and 8 are cleared to select a word length (16-bit) frame sync ;
 5. CRB bit 9 is set to select both the transmitter and the receiver to operate in the synchronous mode and use a common clock (SCK pin) and a common frame sync (SC2 pin) ;
 6. CRB bit 10 is cleared to select the "continuous" clock operating mode ;
 7. CRB bit 11 is cleared to select the "normal" operating mode where one data word (16-bits) is transmitted or received per frame ;
 8. CRB bits 12 and 13 are both set to enable the transmitter and receiver respectively
CRB bits 14 and 15 are both cleared to disable the transmit and receive interrupts respectively ;
 9. CRB bits 3-0 are not pertinent to this particular SSI port configuration, nevertheless they are cleared.

```
CRB      equ      $003200
         movep    #CRB,x:$FFED
```

4.d.4 SSI PORT STATUS REGISTER

The SSI port's Status Register (SSISR) is an 8-bit read-only register used by the DSP56001 processor to interrogate the status and serial input flags of the SSI port. The Status Register occupies X-memory location \$FFEE.

4.d.5 SSI PORT RECEIVE REGISTERS

The SSI port's Receive Shift Register is a 24-bit register that receives incoming data from the Serial Receive Data (SRD) pin. The Receive Data Register (RX) is a 24-bit register that accepts data in parallel from the Receive Shift Register when the Receive Shift Register becomes full. RX occupies X-Memory location \$FFEF. The Receive Data Register Full Flag (RDF) of the SSI port's Status Register (bit 7) is reset when the DSP reads the contents of the Receive Data Register or when the DSP56001 processor is reset.

In the DSP System, the Receive Shift Register accepts serial input data via the Serial Receive Data (SRD) pin when an external frame sync asserts Serial Control Pin 2 (SC2). The Receive Shift Register is externally clocked via Serial Clock (SCK) pin.

4.d.6 SSI PORT TRANSMIT REGISTERS

The SSI port's Transmit Shift Register is a 24-bit register which transmits (outputs) serial data via the Serial Transmit Data (STD) pin. The Transmit Data Register (TX) is

a 24-bit register which supplies data in parallel to the Transmit Shift Register. Data to be transmitted is written into the TX Register and is automatically transferred to the Transmit Shift Register when a frame sync is asserted. TX occupies X-Memory location \$FFEF. The Transmit Data Register Empty Flag (TDE) of the SSI port's Status Register (bit 6) is set when the contents of the TX are transferred to the Transmit Shift Register. TDE is cleared when the DSP56001 processor writes new data to TX or when the DSP56001 processor is reset.

In the DSP System, the Transmit Shift Register transmits serial output data via the Serial Transmit Data (SRD) pin when an external frame sync asserts Serial Control Pin 2 (SC2). The Transmit Shift Register is externally clocked via the Serial Clock (SCK) pin.

4.d.7 READING RX AND WRITING TO TX

For the DSP system, the DSP56001 process reads RX and transfers its contents without modification to TX. The DSP56001 instructions shown below :

1. Polls the RDF in the SSI port's status register (bit 7)
2. Reads RX when RDF becomes set (RX contains data supplied by the DSP56ADC16 A/D converter at a 48 KHz sampling rate)
3. Moves the contents of RX to TX (TX contains data

to be supplied to the D/A converter) and

4. loop back to Step 1

```
poll btst      #7,x:$ffee      ; test for A/D data
jcc           poll            ; loop until RDF bit=1
movep        x:$ffef,a        ; move A/D data to "a"
move         a,x:$ffef        ; send A/D data to D/A
jmp          poll            ; loop indefinitely
```

CHAPTER V
RECONSTRUCTION OF ANALOG SIGNAL UP TO 22 KHz
A/D AND D/A RECONSTRUCTION

An analog input signal is provided from a function generator to the analog input signal of the EVB. The analog input is monitored by one channel of a dual trace oscilloscope. The analog input from the EVB is monitored by the other channel of the oscilloscope.

The DSP56001 processor located on the ADM executes the absolute load file *evb.lod* to route the digitized form of the analog signal. The file *evb.lod* is assembled and linked result of the assembler program.

The analog-to-digital converter (DSP56ADC16) sample the analog signal and send it to the digital-to-analog converter (PCM56) to reconstruct the signal. The output is sent to the oscilloscope via BNC1.

5.a PROCESS TO DESIGN THE COMPENSATION FILTER

The following steps are needed before describing how to get those values :

1. Overview of the DSP program
2. The compensation filter

5.a.1 OVERVIEW OF THE DSP PROGRAM

After editing an assembly source file the ASM56000 Relocatable Macro Cross Assembler is used to translate to the file of assembler source statements for each program or macro into a relocatable file of DSP56001 object code statements. The LNK56000 Linker is then used to process the relocatable object code file generated by the assembler for each program or macro to produce an absolute load file which can be either loaded directly into the ADS for execution and testing or input to the SIM56000 Simulator program for simulated execution and testing. Figure 25 shows the DSP56001 program development process. An alternative way to generate an absolute load file takes advantage of the assembler executable options. A *-b* option specifies that an object file is to be created for assembler output. The type of object file produced depends on the assembler operation mode. If the *-a* option is supplied on the command line, the assembler operates in absolute mode and generates an absolute object file (*filename.cld*).

This file has to undergo another process to be able to be downloaded into the ADS. The command line *cldlod* transform an *.cld* file into an *.lod* which can be used by the DSP56001ADS or the SIM56000.

5.a.2 ALGORITHM

The DSP algorithm which reconstruct an analog signal,

compensate for the $\sin(x)/x$ droop off of the D/A zero order hold effects, works as follow:

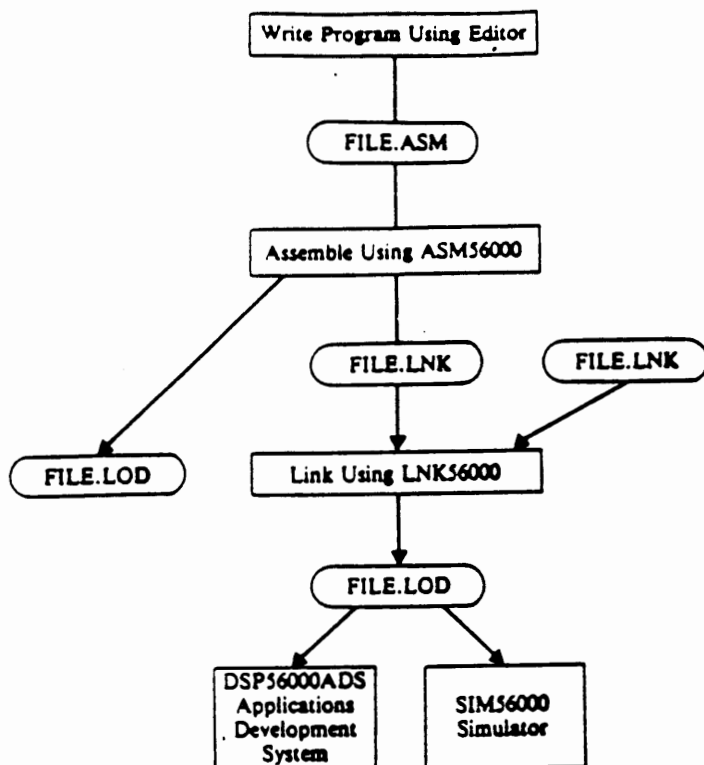


Figure 25. Development process to generate a .lod file.

- A data file which has the FIR filter coefficient is called
- Set up into the X-Memory, starting at location 0, an $ntaps$ storage values for the input sample
- Set up into Y-Memory, starting at

location 0, the FIR coefficients

- Set up the ADS board
- Set up register usage for FIR filter
- Set up port C to function as SSI/SCI, then set the SSI CRA and CRB control register for external continuous clock, synchronous, normal mode.

LOOP: Read the DSP56ADC16 and process samples through a $\sin(x)/x$ compensation filter and write the circular convolution result to the PCM-56 which is a D/A converter.

5.b THE COMPENSATION FILTER

To design a compensation filter we need to sample the signal at an equal frequency interval, get the measured magnitude values, then deduce the impulse response desired to input those values into the compensation filter.

5.b.1 MEASURED VALUES

The frequency interval is chosen to be equal to 387 Hz (Half Sampling frequency / 2 * Ntaps = 24 KHz / 62). All the measured values have been tabulated in Table I.

5.b.2 DESIGN OF THE COMPENSATED FILTER

Using the measured magnitude values in Table I of the

K	FREQ (Hz)	MAG	K	FREQ (Hz)	MAG
123,1	387	1.62	106,18	6968	1.57
122,2	774	1.62	105,19	7355	1.56
121,3	1161	1.615	104,20	7742	1.55
120,4	1548	1.615	103,21	8129	1.545
119,5	1935	1.61	102,22	8516	1.53
118,6	2323	1.605	101,23	8903	1.53
117,7	2710	1.6	100,24	9290	1.515
116,8	3097	1.6	99,25	9677	1.5
115,9	3484	1.595	98,26	10065	1.48
114,10	3871	1.595	97,27	10452	1.475
113,11	4258	1.595	96,28	10839	1.46
112,12	4645	1.59	95,29	11226	1.46
111,13	5032	1.59	94,30	11613	1.44
110,14	5419	1.59	93,31	12000	1.435
109,15	5806	1.585	92,32	12387	1.43
108,16	6194	1.585	91,33	12774	1.42
107,17	6581	1.58	90,34	13161	1.4

(Table continues in the next page)

K	FREQ (Hz)	MAG	K	FREQ (Hz)	MAG
89,35	13548	1.41	75,49	18968	1.2
88,36	13935	1.405	74,50	19355	1.185
87,37	14323	1.385	73,51	19742	1.165
86,38	14710	1.375	72,52	20129	1.115
85,39	15097	1.36	71,53	20516	.84
84,40	15484	1.345	70,54	20903	.49
83,41	15871	1.32	69,55	21290	.24
82,42	16258	1.3	68,56	21677	.122
81,43	16645	1.29	67,57	22064	.0586
80,44	17032	1.28	66,58	22452	.0282
79,45	17419	1.265	65,59	22839	.0146
78,46	17806	1.26	64,60	23226	.00915
77,47	18194	1.24	63,61	23613	.00745
76,48	18581	1.225	62,62	24000	.0002

TABLE I The Analog Signal is set to $2.015 \text{ V} \pm 0.005$ and is processed through a straight A/D D/A converters. The Measured Magnitude Response Of The Digital Sine Wave is recorded Every $24000/62 \text{ Hz}$ and is tabulated.

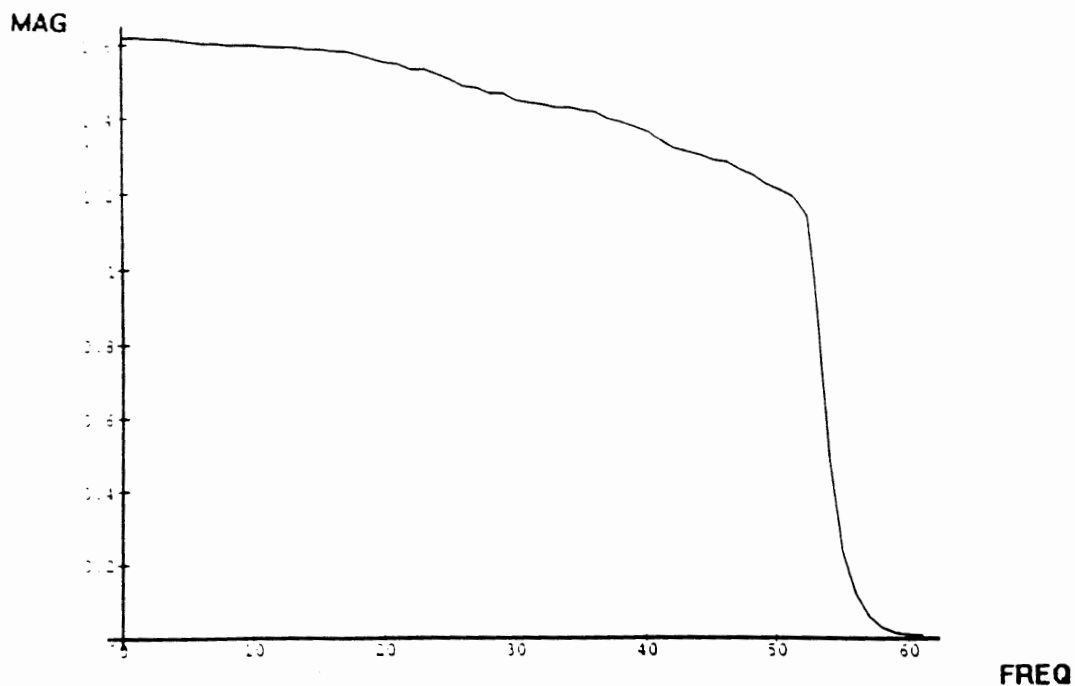


Figure 26. Plot Of The Magnitude Response (as tabulated in Table I) when the analog sine wave is processed through a straight A/D D/A converter. 24000/62 Hz represents a unit in the frequency axis.

FREQ (Hz)	MAG (mV)	FREQ (Hz)	MAG (mV)	FREQ (Hz)	MAG (mV)	FREQ (Hz)	MAG (mV)
19742	17	20903	20	22065	21	23226	13
20129	17	21290	21	22451	21	23612	8
20516	18	21677	21	22839	18	24000	1

Table II Tabulated Result Of The Magnitude Response Of The Digital Signal after it has been filtered.

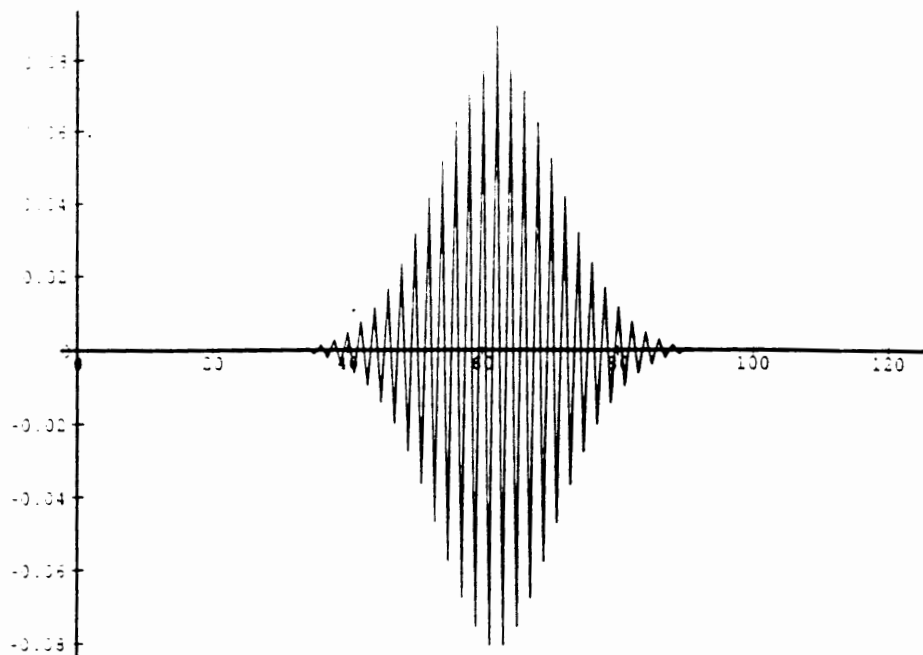


Figure 27. 124-point $\sin(x)/x$ impulse sequence.

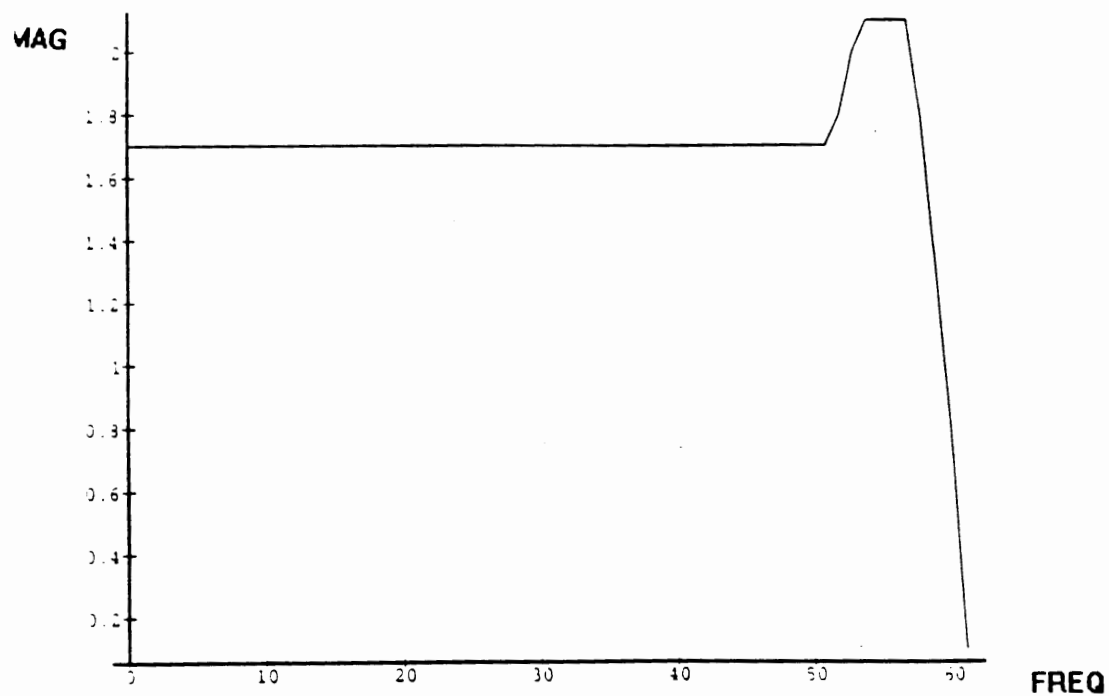


Figure 28. Plot of the magnitude response of the compensated values which were tabulated in Table II. 24000/62 represents a unit in the frequency axis.

straight D/A and A/D converter and computing the IDFT of those values allows us to get an impulse function $h[n]$. The DSP program will then take this impulse function, which can be a real sequence, and will convolute it with the sampled function.

CHAPTER VI
SIMULATION PROGRAM
INTRODUCTION

To show the complexity of the design of the simulation program and the constraint inherent to programming, we will first present a general overview of the algorithm and how to obtain the coefficient of the filter, finally we will discuss the result of our analysis.

We will implement a new program which takes as input a constant unit gain for each input sine wave, and only in the final stage of the program we will let the input sine wave have a gain corresponding to the magnitude response of the system. This stage is necessary to better isolate the compensation process.

6.a AN OVERVIEW OF THE ALGORITHM

To design this simulation program we need to find ways to work around some constraints inherent to programming, while duplicating what we have done using the DSP system. We will first take as input a constant unit gain for each input sine wave, and only at the end we will let the input sine wave have a gain corresponding to the magnitude response of the system. One important limitation is to work with a small number of sine wave as oppose to an unlimited number in the

hands-on approach (using directly the DSP system). To remedy this obstacle we will associate a phase shift for each input sine wave. This phase shift is preferably chosen to be a prime number. After this program digitizes and shifts a sine wave, it will convolute this new sampled sine wave with the designed digital compensation filter. Only the maximum magnitude value resulting from that convolution will be stored, the remaining values will be discarded. For each maximum magnitude value corresponds an increment frequency f_{incr} , up to 24 KHz. This increment frequency is equal to Nyquist frequency divided by the number of point used to design the filter. By collecting all the magnitude value starting at 0 and incrementing by f_{incr} until the Nyquist frequency, we will be able to determine the behavior of this compensation filter up to 24 KHz.

After successfully processing the data (the theoretical values are the same values as the measured values), we will be ready to allow the input gain to be equal to the magnitude response of the system and to monitor the behavior of the magnitude response. The program can be divided into 4 parts:

- 1) Digitization and shift of a sine wave
- 2) Convolution with the design $\sin(x)/x$ filter
- 3) Mathematical tools to design the filter such as the DFT and IDFT

4) The main program

The program shapes the output in Maple V format. This formatted output will allow us to plot the value of a sine wave which is digitized and shifted, or/and the convolution result, or/and the magnitude response of the reconstructed digital signal.

6.a.1. DIGITIZATION AND SHIFT OF THE SINE WAVE

The digitization and shifting procedure, will be used to sample at 48 KHz (theoretical sampling rate of the DSP56ADC16 A/D converter) a sine wave at a given frequency.

In practice, using the DSP system we have a large number of periods of input sine waves. To get a replica of this signal and be able to process it in our program we need to shift this sine wave. By shifting this digitized signal we do not change the magnitude of its fourier transform, however we will amplify the magnitude response by $T/2\pi$ and we will normalize the frequency domain by $2\pi/T$.

$$\mathcal{F}(g(t-t_0)) = G(\omega) e^{-j\omega t_0} \quad (13)$$

Now if we introduce a phase shift $\alpha = 2\pi(t-t_0)/T$, the DFT of $g(t)$ becomes:

$$\mathcal{F}[g((2\pi/T)(t-t_0))] = (T/2\pi)G(\omega/(2\pi/T))e^{j\omega t_0} \quad (14)$$

So, shifting a finite number of periods of sine wave is one

solution to recover the original sequence of an unlimited number of periods of the input sine wave and allows us to duplicate the process used by the DSP system. The following is the code used to get the digitized and shifted sine wave

```

BEGIN
for i = 0 to Ntaps-1 do
    digital[i]=gain*sin((2π*freq*i)/S_rate+(shift*phase))
END

```

6.a.2 CONVOLUTION ALGORITHM

There are several techniques to convolute two sequences. It is easier for us to think about two sequences of equal length. If one of them is larger, than the smaller sequence is padded with zero until they are of same length. The following code assumes that the convolution is starting

```

BEGIN
    For j=0 to j<Ntaps do
        Begin
            For i=0 to i≤j do
                temp = First_seq[j-i]*second_seq[i]+temp;
                convolution_seq[j] = temp;
            End
        End
    END

```

Now that the end of the First-seq has been reached the pseudo code will be:

```

BEGIN
  For j=Ntaps to j < 2* Ntaps-1 do
    Begin
      count = count +1
      For i = count to i < Ntaps - do
        temp = first_seq * second_seq[j-i]+temp;
        convolution_seq[j] = temp;
      End
    End
  END

```

6.a.3 INVERSE DISCRETE FAST FOURIER

The Inverse Discrete Fourier Transform will be used to get the $\sin(x)/x$ coefficient filter

```

BEGIN
  For m=0 to m<idft_points do
    For k=0 to k<idft_points do
      Begin
        Angle = 2*m*k/idft-points;
        SumRe=x[k].Re*cos(Angle) -x[k].Im*sin(Angle)+SumRe
        SumIm=x[k].Im*cos(Angle)+x[k].Re*sin(Angle)+SumIm
      End
    End
  x[m].Re = SumRe/idft_points;
  x[m].Im = SumIm/idft_points;
END

```

6.a.4 MAIN PROGRAM

The objective of this program is, to design a good digital compensation filter and to duplicate what we did with the DSP system. This program takes as input the measured magnitude value of the system. Before designing this FIR filter we need to think about two flaws that arise:

- 1) Only a finite sequence of data are available to digitize the continuous sine wave function. To get a good replica of the continuous sine wave we need to shift the sine wave. By adding a phase shift, the magnitude of the discrete fourier transform does not change (see 6.a.1).
- 2) The Gibbs'phenomenon occurs when we are convoluting a finite length data record with the digital filter.

BEGIN

- *Initialization;*
- *Input data;*
- *Create the filter coefficient:*
 - a) *Inverse the gain-array*
 - b) *Inverse discrete fourier transform*
 - c) *Shift impulse sequence*
 - d) *Add zero to the impulse sequence*

While (Frequency less than 24 KHz) do

Begin

For (phase shift=0 till End phase shift angle) do

Begin

- *Digitize and Shift the sine wave*
- *Convolution of the Digitize sine wave with the design FIR filter and discard transient states only the steady state of the convolution is accepted*
- *Determine the maximum magnitude value*

End

Increment frequency

End

- *Compare all the maximum magnitude value after applying different phase shift and pick the maximum of that set.*
- *For each frequency store all the maximum magnitude in an array.*

END

6.b CHOICE OF INPUT DATA

This section will focus on choosing the right input data for which the reconstructed signal will be adequately compensated. After analyzing the behavior of this new signal in the frequency domain we will be able to better understand the output recorded using the DSP system.

6.b.1 INTRODUCTION

From the magnitude of the discrete fourier transform of the impulse sequence, and the measured magnitude values of the system, we are able to compare, the expected theoretical values of the reconstructed digital signal (when both values are multiplied) and, the measured values of the reconstructed signal. Also, in Figure 29, a 50-points DFT $\sin(x)/x$ magnitude response is shown.

From the measured magnitude values of the system, the inverse magnitude values will be computed which in turn will allow us to find the magnitude response of the FIR filter. To layout the complete simulation program we will compare the theoretical magnitude values, derived from the FIR filter and found previously, with the program generated values (let us call them the measured values). The gain is assumed to be 1. later, we will incorporate in the program the corresponding gain for each given frequency.

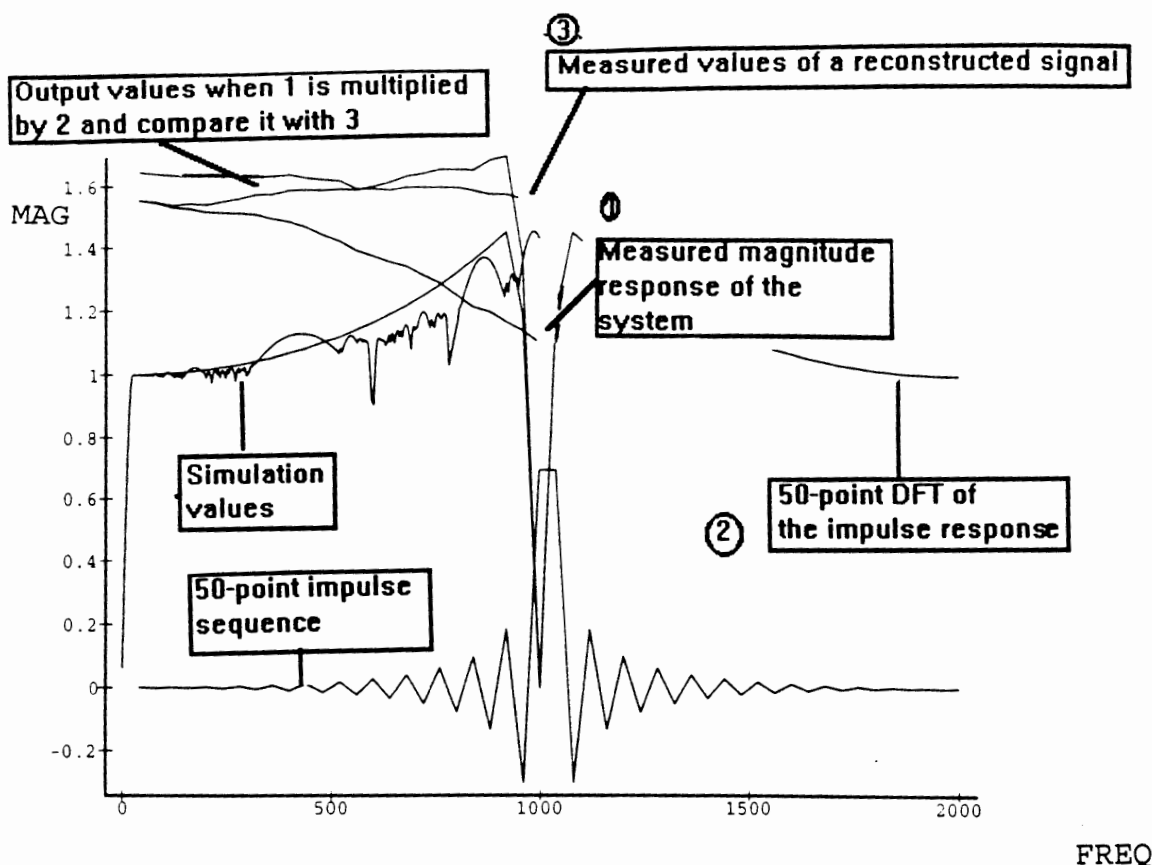


Figure 29. 50-point DFT impulse sequence and magnitude response of the, measured values, programmed values, inverse values, expected reconstructed and straight A/D D/A converters of the digital signal. $24000/1000 = 24$ Hz represents a unit along the frequency axis.

6.b.2 PROGRAM CONSTRAINTS

In this section, we will present and analyze several graphs where discrepancies between the theory and the measurement (programmed value) are observed. A solution to remedy those differences will be suggested.

The Motorola DSP56001 processor in combination with DSP56ADC16 and the PCM-56 converters, can input several continuous sine wave to be sampled and convoluted. The result will be sent directly to the oscilloscope. This process happens so fast that we are not able to see the transient state. Only, the steady state appears on the oscilloscope screen. No limit on the number of periodic sine wave need to be observed. Continuous periodic sine wave versus of finite number of periodic sine wave is one of the program constraint that we will have to deal with.

We can see in Figure 29, that the theoretical and the program magnitude response are different. Let's observe figure 30-36. We notice that at certain points, the measured value peaks and several bumps shapes the curve. Also, it is surprising to see that the measured value is even greater than the theoretical values. Those observations will be analyzed and a step by step approach will be conducted on each graph until we will be able to reconcile the measured and theoretical values.

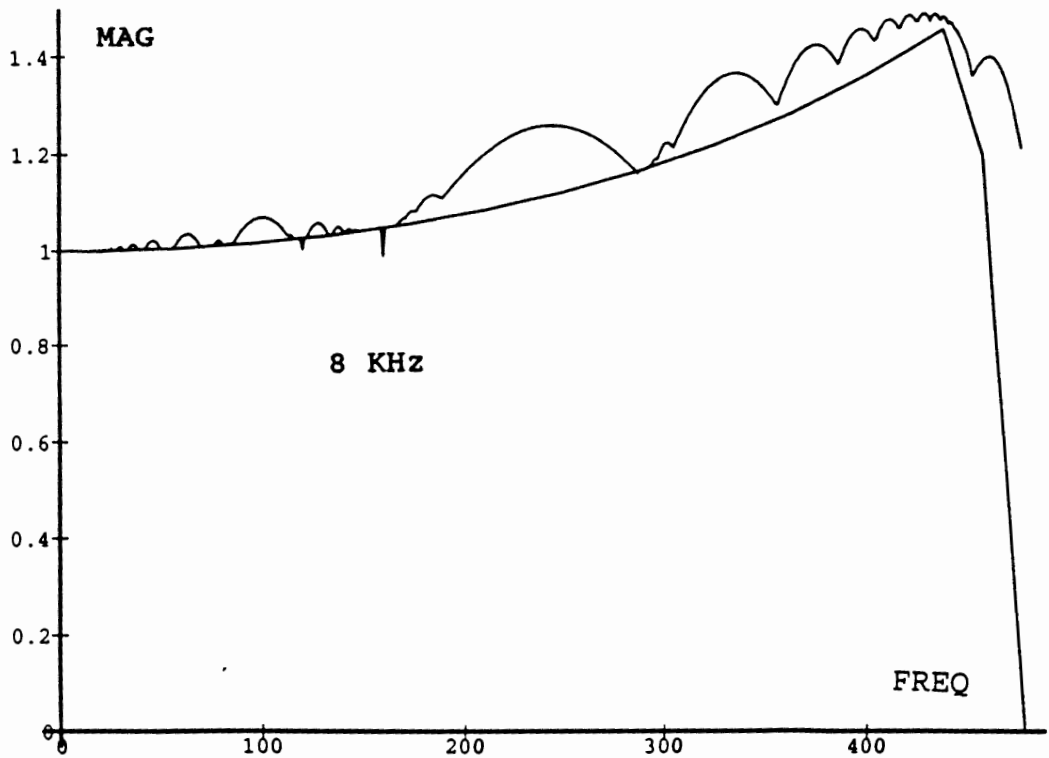
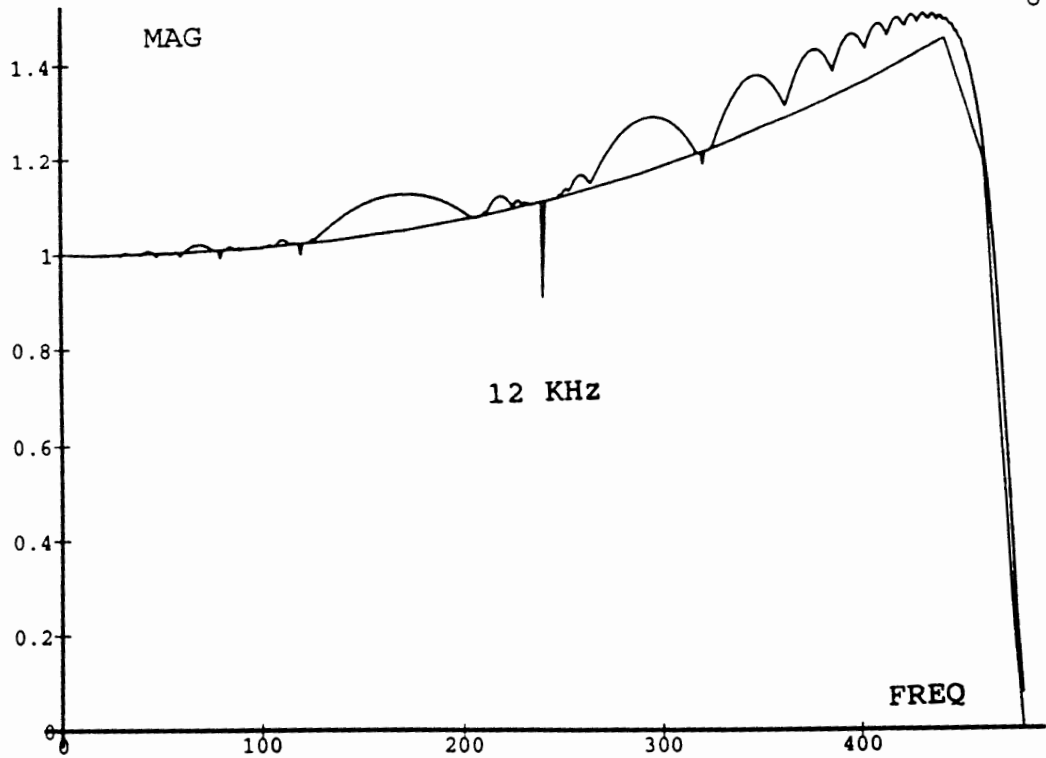


Figure 30 (top) and 31 (bottom). Magnitude Response of the program values and its corresponding 50-point DFT for a phase shift of, respectively 0 and 15 degree. $24000/480 = 50$ Hz represents a unit in the frequency axis.

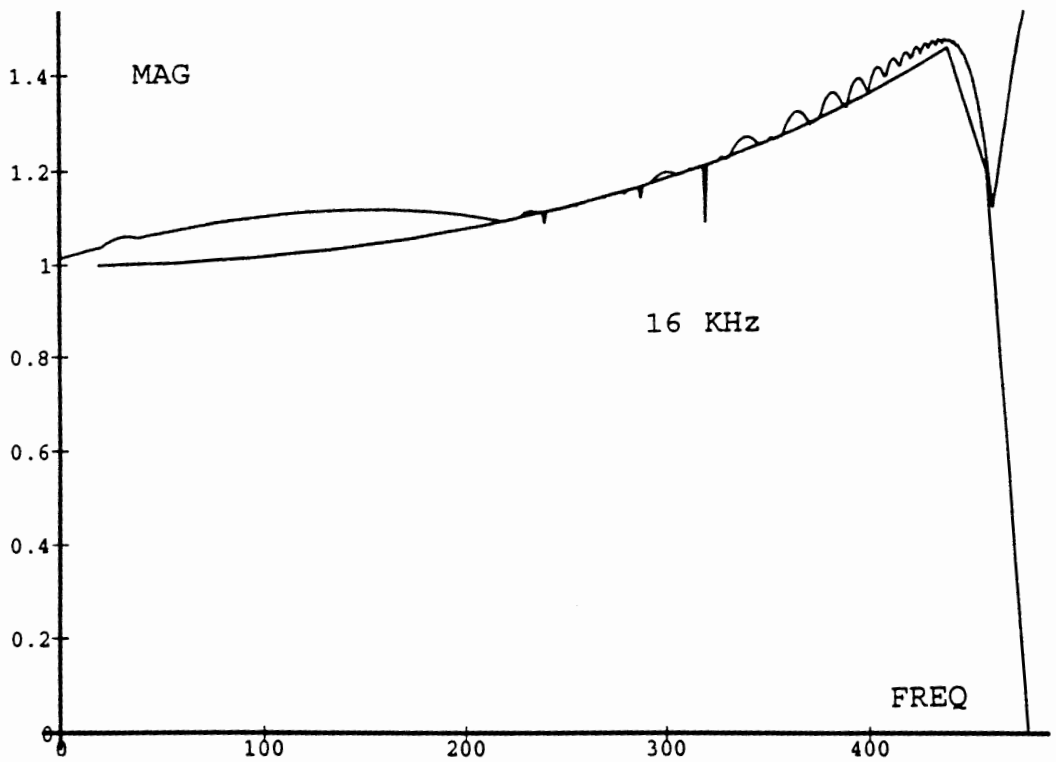
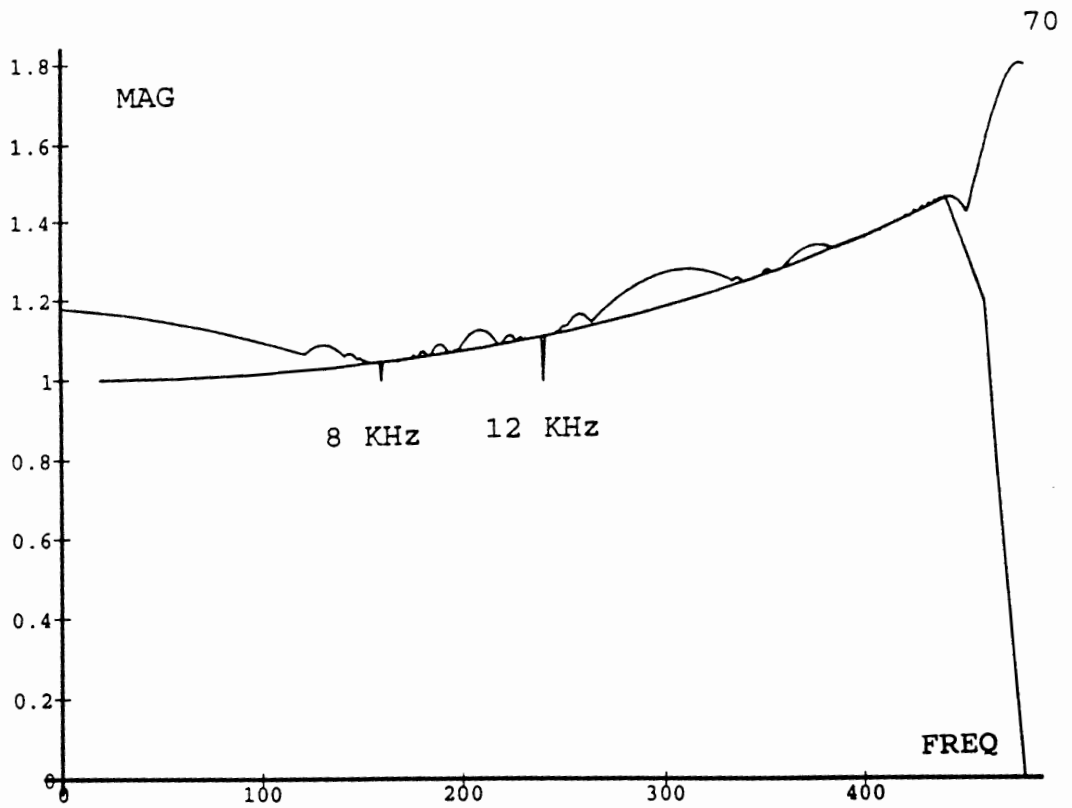


Figure 32 (top) and 33 (bottom). Magnitude Response of the program values and its corresponding 50-point DFT for a phase shift of, respectively 30 and 45 degree. $24000/480 = 50$ Hz represents a unit in the frequency axis.

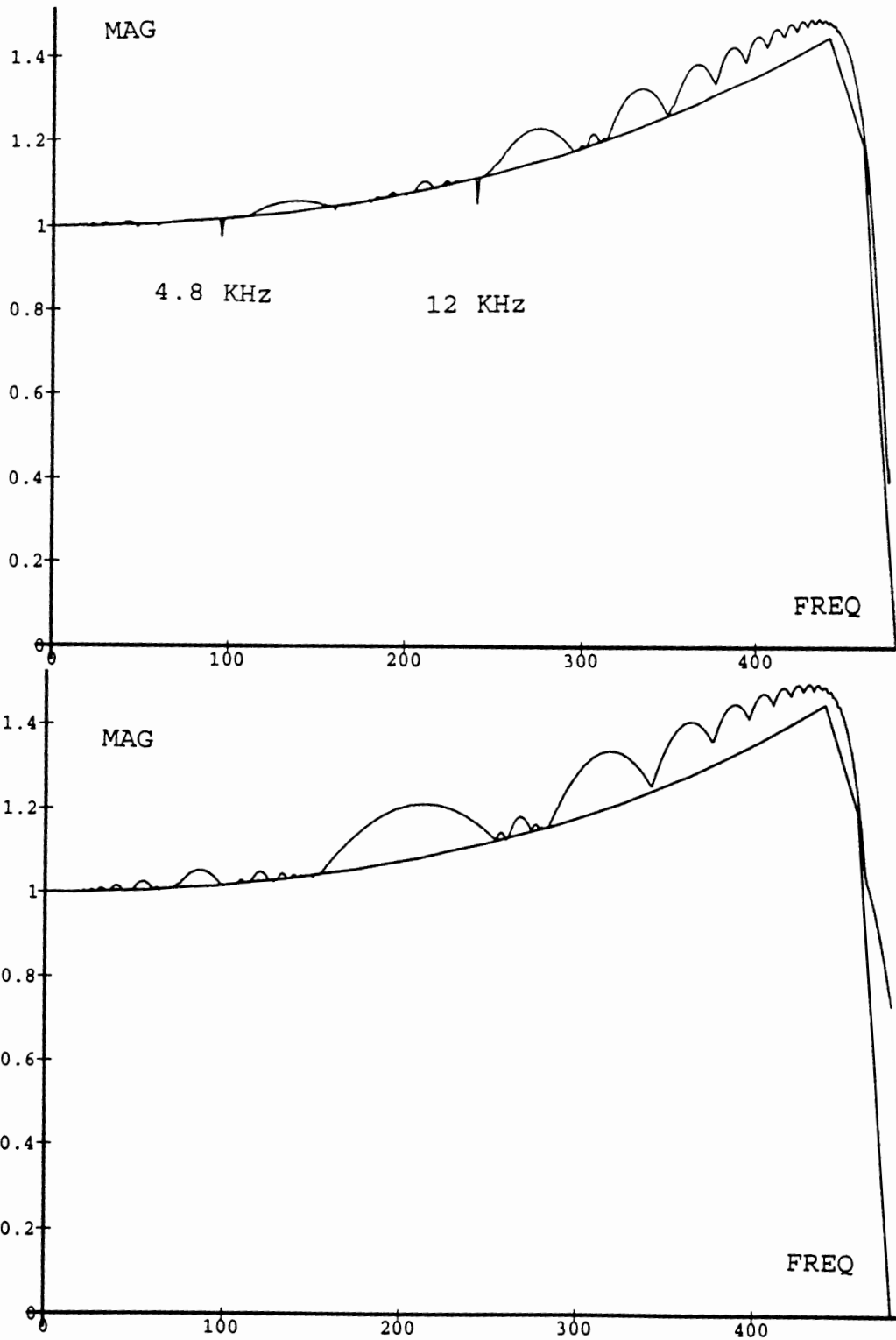


Figure 34 (top) and 35 (bottom). Magnitude Response of the program values and its corresponding 50-point DFT for a phase shift of, respectively 60 and 75 degree. $24000/480 = 50$ Hz represents a unit in the frequency axis.

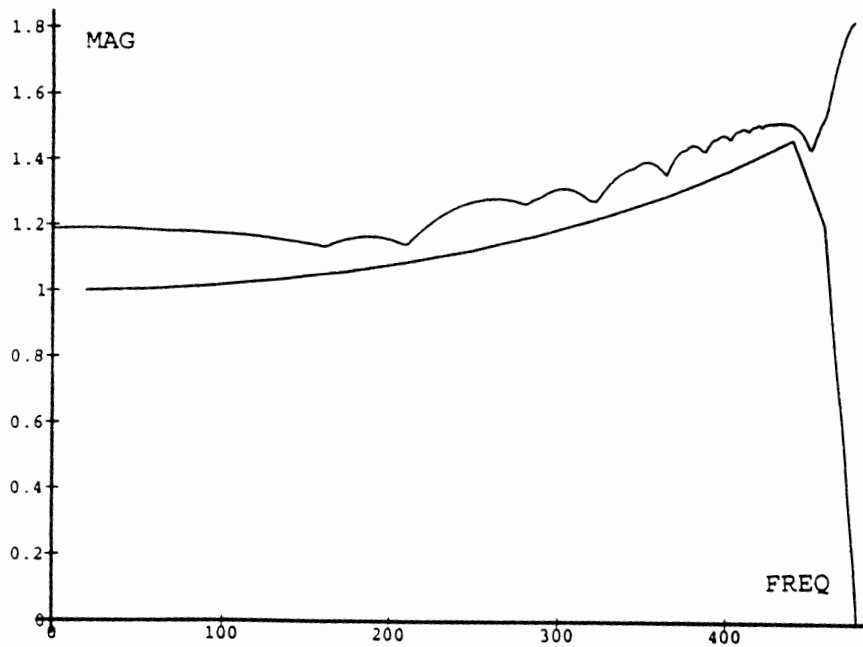
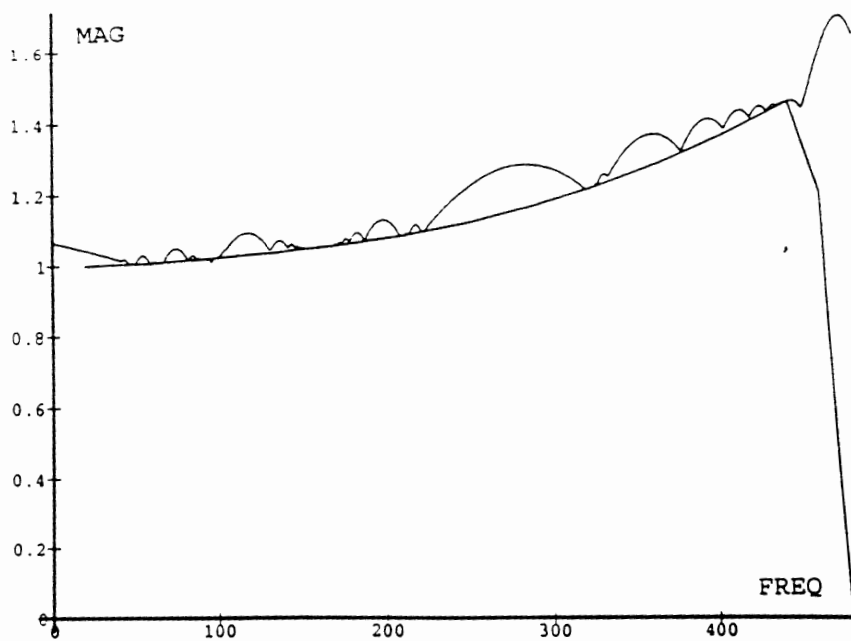


Figure 36 (top) and 37 (bottom). Magnitude Response of the program values and its corresponding 50-point DFT for a phase shift of, 90 degree. Collected maximum of each phase shift is shown in figure 41. 50 Hz represents a unit in the frequency axis.

To get the program values of the magnitude response we take the maximum amplitude value of the convolution sequence at each incremented frequency as seen in Figure 30-36. Our first objective is to explain why the measured values peaks at certain frequencies. From the graph we notice that the points at which the amplitude peaks, are when those values are multiples of 48 KHz (sampling frequency). This observation indicates that the sine wave has not been properly digitized at those particular frequencies see Figure 38.a and 38.b.

As a result of those observations, we have introduces a phase shift for each sine wave. By shifting the sine wave, we will not change the magnitude response of the signal. Each sine wave shifted will have its corresponding maximum value determined after convoluting. The magnitude response of this digital signal will be chosen to be the maximum value of the collected maximum values found previously (see Figure 37).

6.b.2.b GIBBS' PHENOMENON

In Figure 37, most of the minimum peaks have been removed. We need now, to minimize the error that arises between the measured and the theoretical values.

We notice that when averaging all the maximum value of the corresponding phase shift, the magnitude response of the measured value come closer to the expected values. But

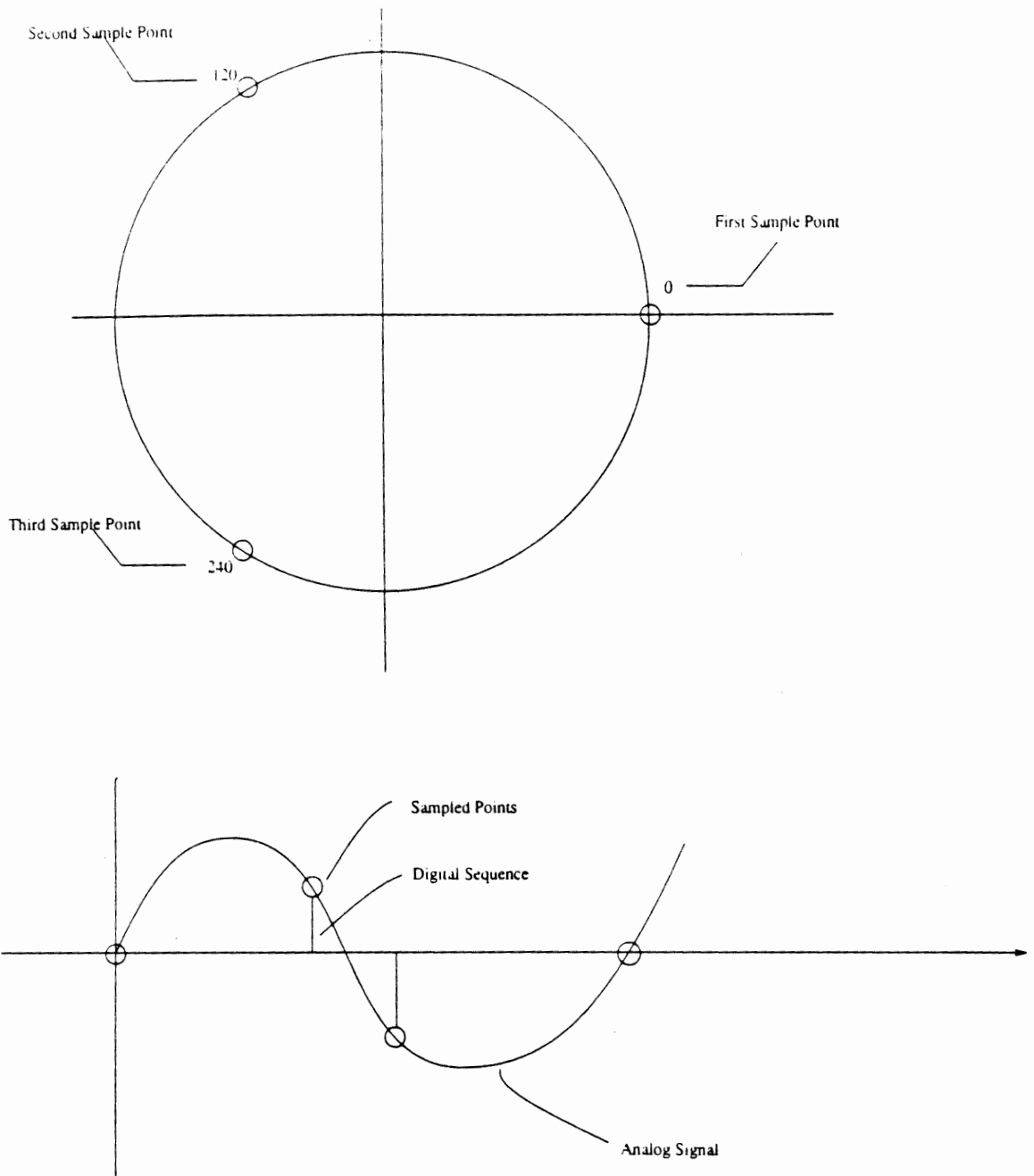


Figure 38. (a) Shows the sample points in the unit circle, when the frequency of the signal is 16 KHz (b) Shows the corresponding point in the time domain.

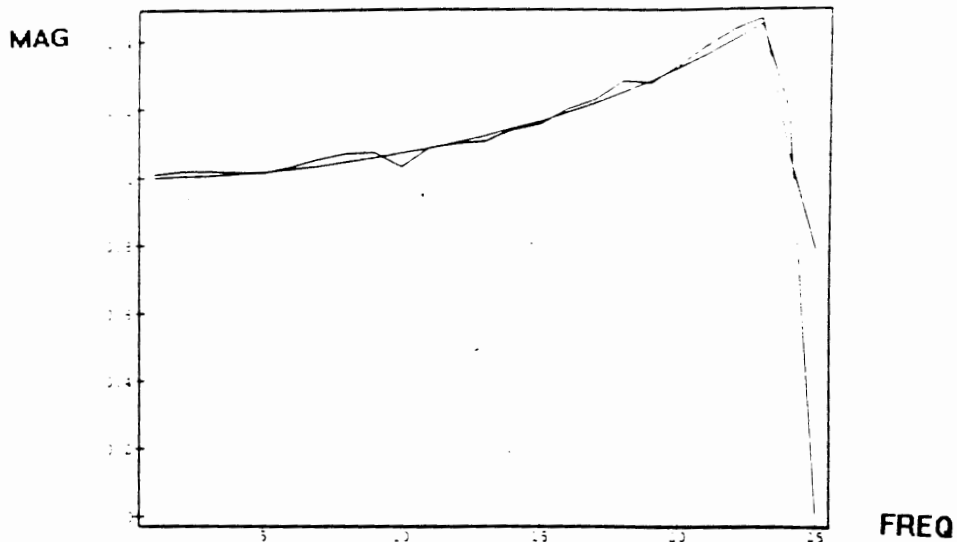


Figure 39. Magnitude response of the average values of each shifted sine wave, shown in figure 30-36. Each unit in the frequency domain represents 960 Hz.

again, the measured values gives us at certain frequencies, a magnitude response greater than the one we want, see Figure 39. We need to examine why the magnitude response is greater than the magnitude values.

In the time domain, we can see that the convolution sequence reconstructs a distorted peak at the beginning (can also happen at the end) of the steady-state see Figure 40. This explains why, an error of up to 14% occurs between the expected and the measured values, in the frequency domain. To avoid taking the wrong maximum values, we have discarded the

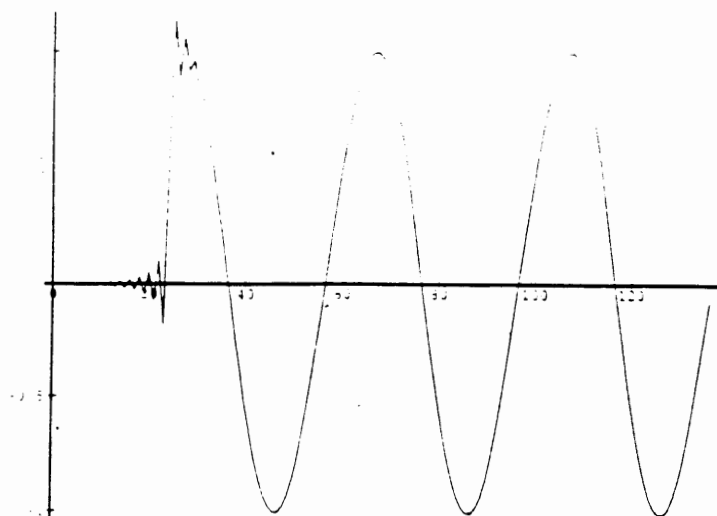


Figure 40. Represents the transient and the steady state of the convolution sequence of, the sine wave at 2880 Hz, with the $\sin(x)/x$ FIR filter. A discrepancy of the sine wave peak in the transient state of the convolution sequence due to the Gibbs' Phenomenon is shown.

beginning and the end sequence of the convolution sequence. Only, the middle of the steady state is processed to obtain the maximum value. The result of this operation is shown in Figure 41.

6.b.2.e PARTICULAR VALUES OF THE PHASE SHIFT

By shifting the sine wave we are able to improve the magnitude response (see figure 30-36 and 37). In figure 41, we can see that some discrepancies still exist at several points. These points are particular since the sine wave frequency divided by the sampling frequency are multiple of

the phase shift, so it is as if no phase shift has been applied. The first point where a discrepancy occurs, is at 2.4 KHz. 20 points will sample this sine wave at every $\pi/10$ radian, and the phase shift happen to be twice this latter value, see Figure 41.

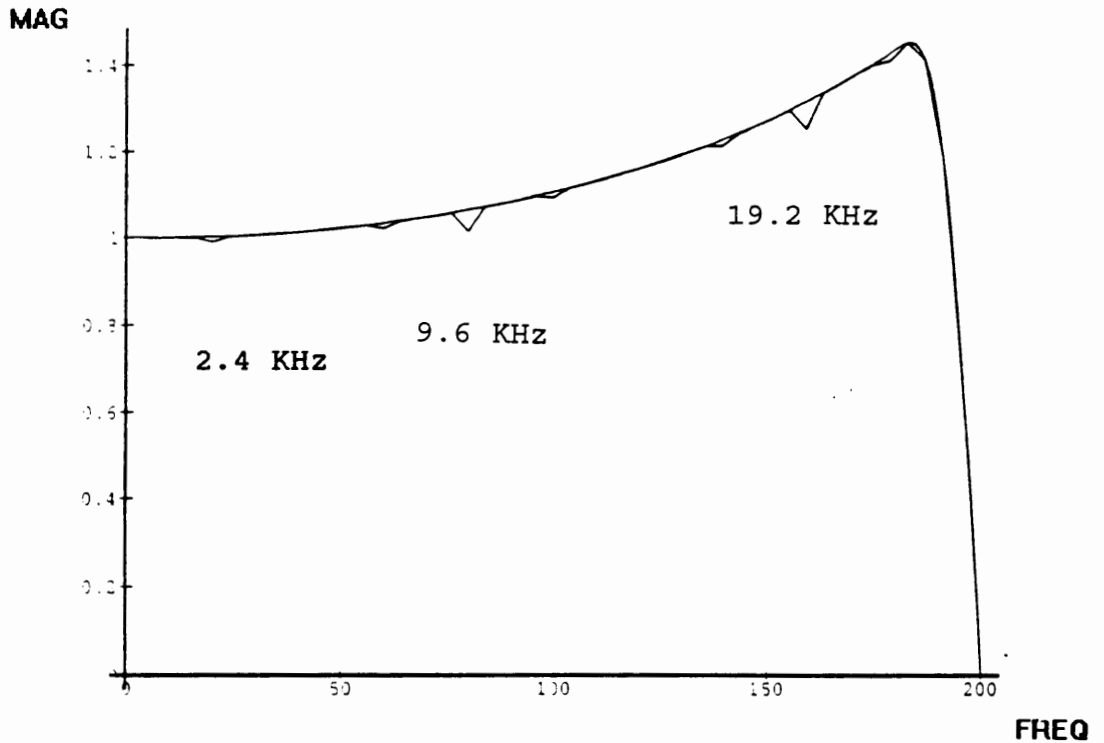


Figure 41. Discrepancies of the magnitude response exist when the frequency (of the signal) divided by the Sampling Frequency are multiple of the phase shift. $2\pi/10$ rad is the phase shift. 120 Hz represents a unit in the frequency domain.

One solution would be to take a prime number as phase shift.

The new magnitude response is shown in figure 42.

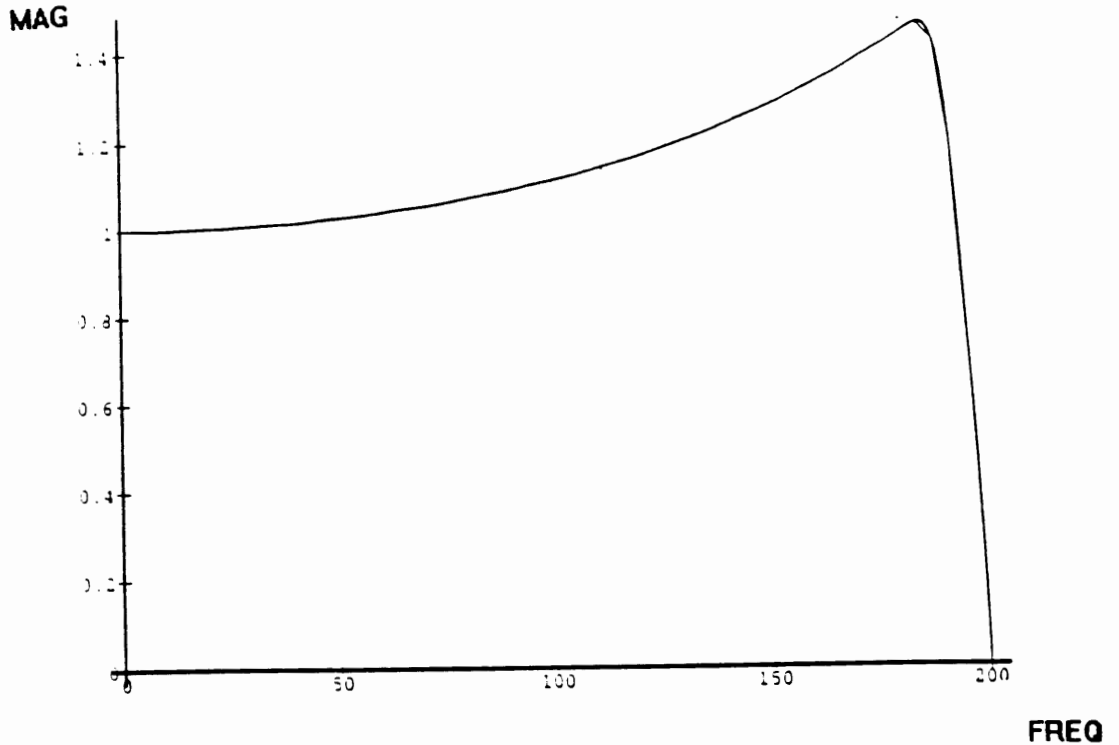


Figure 42. Successful correction of the magnitude response of the program value by taking a prime number as a phase shift (43 degree). 120 Hz represents a unit in the frequency axis.

Now that the errors have been corrected we are ready to design the filter by introducing the corresponding gain of the measured magnitude values (straight A/D and D/A measured values) at each given frequency.

6.c THE COMPENSATION FILTER

After the analog signal has been pass through the A/D and D/A converters, we are able to determine the magnitude response of the digital sine wave. To design the FIR compensation filter we need to invert those values and take the inverse discrete fourier transform.

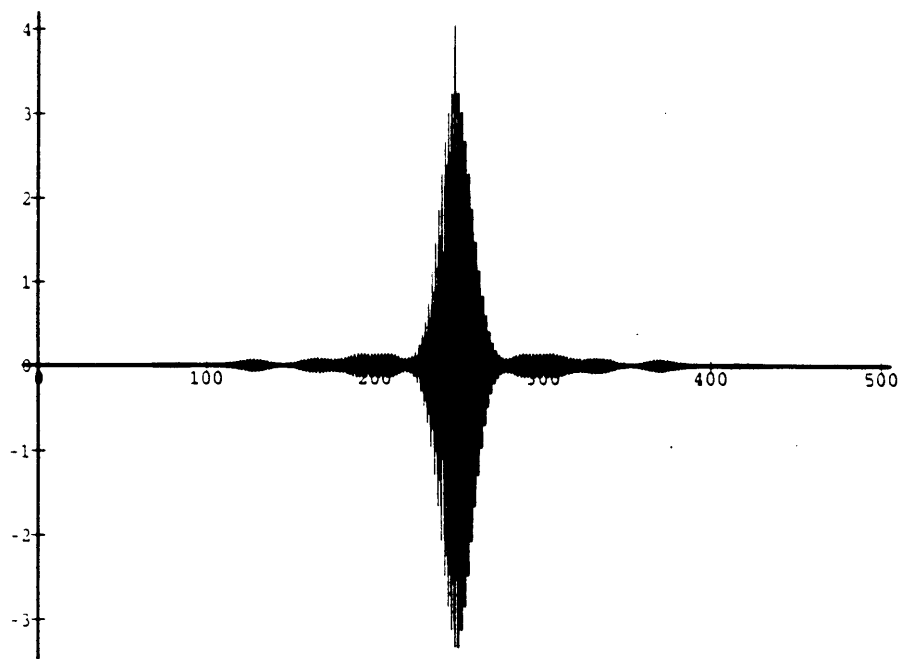


Figure 43. *Impulse response of a 496-point $\sin(x)/x$ FIR filter.*

6.c.1 DISCONTINUITY AT HIGH FREQUENCY

Around 20 Khz (point 50 in Figure 44) the inverted values begins to increase very sharply, while the number of

sampling frequency points are limited. We can see in Figure 43 that at low frequency the rate at which the curve changes is almost 0 while there is a large number of points that map this frequency region. However, around 20 KHz (point 50 in Figure 44) the rate start to change and reaches a very large number while a very small number map this frequency region.

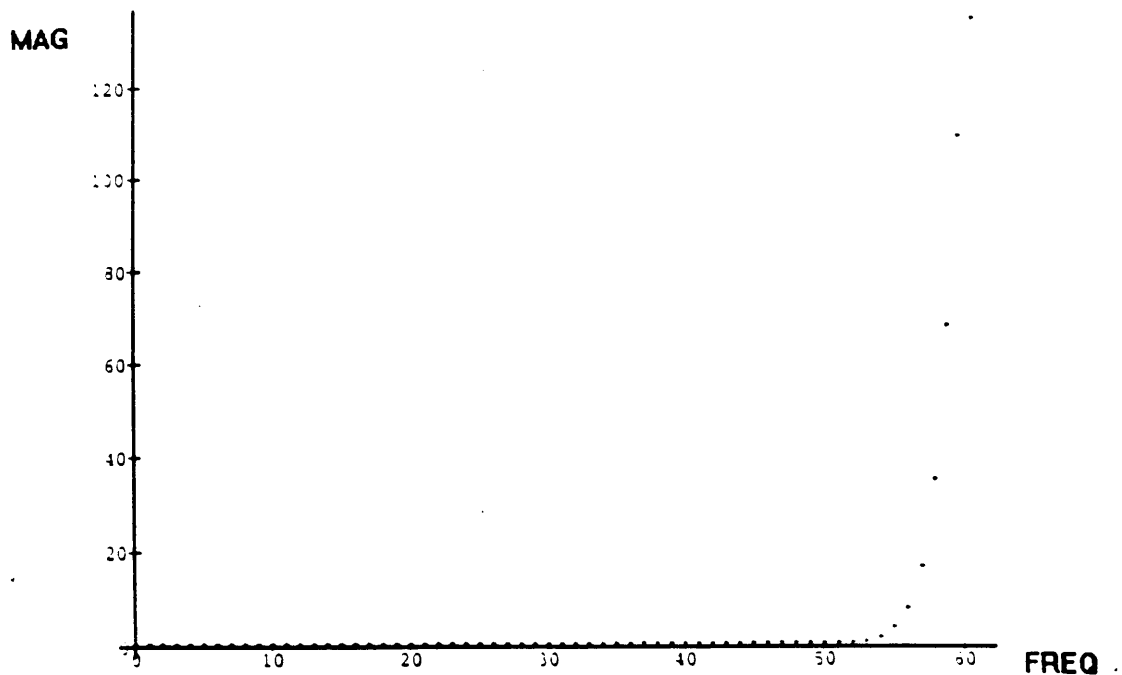


Figure 44. *Inverted magnitude values of the straight A/D D/A converters (See Table I). Only 62 points sample the frequency up to 24 KHz. A unit in the frequency axis, represents 387 Hz.*

From the magnitude response of the system recorded in Table I, we have inverted those values and tabulated them in Table III. Also, we have computed the DFT of the impulse response to compare it with the inverted values. Both values should be the same since the coefficient

FREQ (KHz)	INVERSE	DFT OF COEFF	FREQ (KHz)	INVERSE	DFT OF COEFF
18.43	.8403	.8368	21.53	2.5	2.159
18.62	.8474	.8439	21.73	3.125	2.812
18.82	.8474	.8475	21.92	3.636	3.381
19.01	.8547	.8511	22.12	5.0	4.318
19.2	.8621	.8584	22.31	10.0	7.5
19.4	.625	.7435	22.5	14.29	12.14
19.6	.625	.625	22.7	20.0	17.14
19.8	.8658	.7454	22.9	33.33	26.67
20	.8696	.8677	23.08	40.0	36.67
20.18	.8696	.8696	23.28	52.63	46.32
20.37	.9091	.8893	23.47	58.82	55.73
20.56	1.0	.9545	23.67	66.66	62.75
20.76	1.052	1.026	23.86	76.92	71.79
20.95	1.111	1.082	24	76.92	76.92
21.15	1.333	1.222			
21.34	1.818	1.576			

TABLE III *Magnitude of the, Inverted values and DFT of the coefficient of the filter (same starting frequency)*

of the filter has been derived from the inverted values by doing an Inverse Fourier Transform.

Table III shows that, the magnitude response of the inverted values and of the DFT of the coefficient of the $\sin(x)/x$ filter are not the same values, at high frequency. Therefore it is normal that there will be some discontinuity in the magnitude response as seen in Figure 44. The frequency response of the digital sine wave has been compensated up to 22 Khz. The discontinuities that we see at high frequencies are due to the limited number of point that map this frequency region. So, as we sample the magnitude response of the straight A/D D/A converters with a larger number of point the discrepancies will decrease. Table IV, shows that as we increase this number (*sampling frequency point*) the maximum amplitude peak is reduced, from a magnitude of 1.96 with 62 sampling point to 1.2 for 248 sampling point as shown also on Figure 44.

<i>SAMPLING POINTS</i>	62 POINTS	124 POINTS	248 POINTS
<i>MAXIMUM AMPLITUDE</i>	1.96	1.52	1.2

Table IV Shows that as we increase the number of sampling point in the frequency domain the maximum peaks decreases.

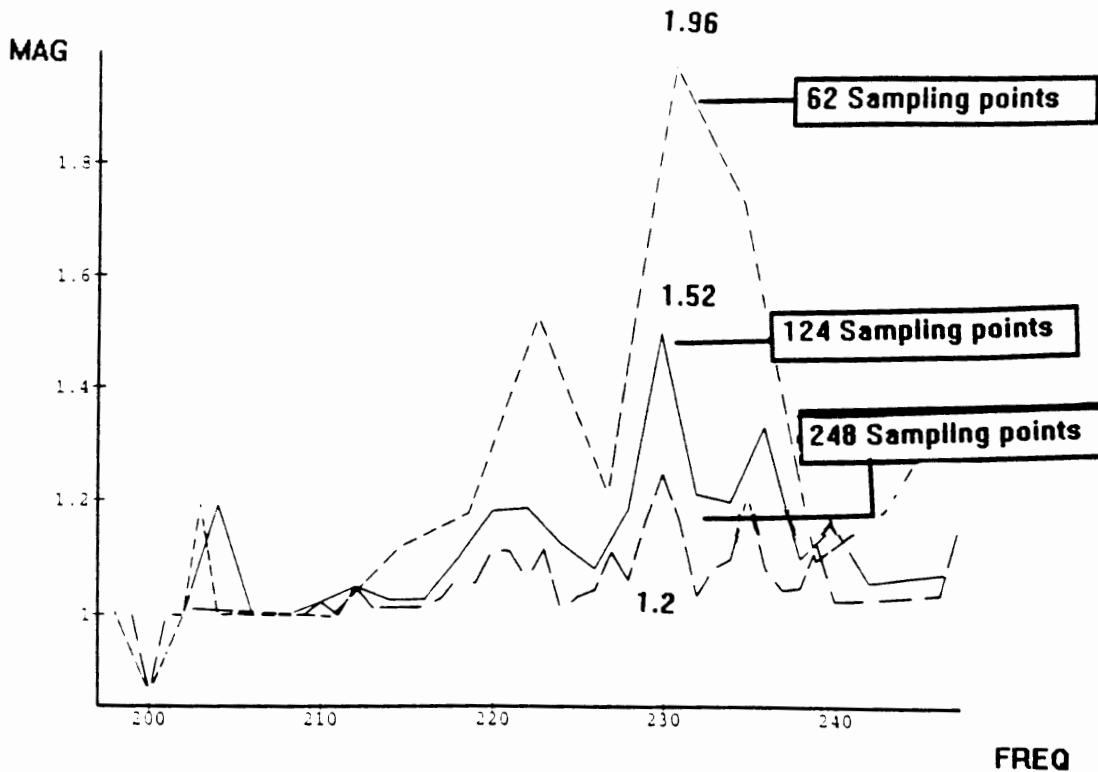


Figure 45. Plot of 3 magnitude response of the compensated program values, which have been determined after the analog sine wave is passed through respectively 124-point, 496-point and 992-point $\sin(x)/x$ FIR compensation filter. Respectively, 62, 124 and 248 points are needed to sample the magnitude response of the A/D D/A converters. A unit in the frequency axis represents 115 Hz. The frequency starts at 19.4 KHz.

CHAPTER VII

CONCLUSION

This thesis focused on the design of a good digital compensation filter. Our objective was to correct for the magnitude attenuation of the input signal when processed by the Motorola DSP56K. Our simulation program which allowed us to monitor the magnitude response of the system has shown that, as we decreased the frequency interval increment f_{incr} , we got closer to an ideal magnitude response.

We have used the 24-bit Motorola DSP56K which is a general purpose, single chip Digital Signal Processor (DSP). It was not designed for a particular application but to execute commonly used Digital Signal Processing benchmarks (such as Digital Filtering, Signal Processing, Data Processing). A DSP can duplicate almost any analog electronic circuitry.

To compensate for the magnitude decrease of the output signal (sine wave), we have inverted the magnitude response of the system and taken the Inverse Discrete Fourier Transform (IDFT) of the inverted response. The magnitude response of the system is determined by finding the magnitude at each frequency interval increment which is equal to the Nyquist frequency (frequency at which no aliasing occurs) divided by the number of taps (the number of point used to

design the filter). The impulse response (coefficient of the filter) obtained is convoluted by the digitized sine wave. Each digitized sine wave has its corresponding gain, which is determined by the system. At the beginning of the experiment, a phase shift is introduced which depends on the time we have sampled the signal. Although we have focused our study on a sine wave input signal, it can be generalized to other periodic signals.

Two experiments have been conducted to compensate for the magnitude decline of the digital signal:

The first experiment used the DSP56K Application Development System, which allowed us to design, debug, and to evaluate a hardware tool such as the DSP56K based system. Also, the 24-bit Motorola DSP56K processor has been used in combination with the DSP56ADC16 (A/D converter) and the PCM-56 (D/A converter), to determine the magnitude response of the system. From those values, we have deduced the coefficients of the FIR filter by taking the IDFT of the inverse magnitude response. A look-up table stored those values which would be fetched later by the DSP program. By taking advantage of the instruction set features offered by DSP56K we were able to generate a compact code. By combining a Repeat Next Instruction (REP) with a single multiply-accumulate (MACR) instruction, which is capable of executing two concurrent data move operations, we were able to design an N-tap FIR filter algorithm. This minimum set of

instructions performed a N-point circular convolution which allowed us to generate a digital output sine wave of constant amplitude at low frequency (less than 20 KHz). However, a second experiment will be needed to improve this digital compensation filter because the Gibbs' phenomenon introduced errors at high frequency (greater than 20 KHz). Also, we were not able to go beyond a 300-point impulse sequence, because the time that each sample value is read and is processed by the DSP56K is shorter than the time each instruction in the DSP program is executed and is written to the PCM-56. To be able to expand our experiment, we needed to conduct a new experiment, to show our premise (as we decrease the frequency interval increment, we would obtain a good filter).

In the second experiment, we wrote a new program which took as its input a constant unit gain for each input sine wave, and at the final stage of the program we let the input sine wave have a gain corresponding to the magnitude response of the system. To do this it is necessary to better isolate the compensation process. This program has enabled us to design the desired filter associated with the DSP56K Application Development System and to observe the compensated magnitude response. To implement this program, a step by step approach has been drawn until our final objective has been reached.

To design this simulation program we needed to find ways to work around some constraints intrinsic to programming,

while duplicating what we did with the DSP system. One important limitation of this program was that it worked with a small number of periods of the input sine wave as opposed to a continuous sine wave (using directly the DSP systems). To remedy this obstacle we have associated a phase shift for each input sine wave. This phase shift is preferably chosen to be a prime number to avoid that the frequency divided by the sampling rate is a multiple of the phase shift because this would be as if we didn't shift the input signal. Also, we noticed that the sine wave resulting from the convolution of the digitized sine wave and the impulse sequence peaked at the beginning and/or at the end of the sequence, and introduced an error of up to 14% compared with the maximum peak in the steady state component of that sequence. To avoid those error values, we have discarded the beginning and the end of the convolution sequence. This method allowed us to get the maximum peak in the steady state. Each shifted sine wave would have its corresponding optimum value, that is the maximum value of the whole convoluted sequence. The magnitude value at a particular frequency f_c was determined by collecting all the optimum values at f_c and by taking the maximum in absolute value of those values. By gathering all the magnitude values starting at 0 and at increments of f_{incr} until the Nyquist frequency, we were able to successfully map the frequency response of the system. Until now, the compensation process hasn't started, (the gain of the input

sequence remains equal to one).

Now we set the input gain to be equal to the magnitude response of the system and designed several digital compensation filters, to compare and to analyze their magnitude response. Two observations have been made after designing a 124-point compensation filter:

First, the inverse magnitude response divided the frequency domain into two regions. The first one is the frequency below 20 KHz which has a good mapping of this frequency region (most of the magnitude values are concentrated here and the magnitude response is a constant), while in the second one, the rate at which the curve changed, became larger as the frequency increased and only a few points (9 points) covered the remaining 4 KHz.

Second, we observed that, at high frequency the DFT of this impulse sequence (b_i) is not the same value as the inverse magnitude response of the system.

From these observations we were able to conclude that, as we increase the sampling frequency, b_i would eventually converge to the inverse magnitude response of the system. However, they would never be equal because of the Gibbs' Phenomenon. Two other digital filters have been designed: a 248-point and 496-point digital filter. The latter is closer to the ideal filter than the former, supporting our conclusion. Our underlying hypothesis from the beginning has been that these filters could be adequately approximated:

such sharp cutoff digital filters can be realized, although they are more costly. However, if we compare them with sharp cutoff analog filters which are designed using active networks and integrated circuits, a digital filter may be the cheaper alternative.

REFERENCES

- [1] Yih-Chyun Jenq, *Sine Interpolation Errors in Finite Data Record Length*, IEEE IMTC/94, Hamamatsu, Japan, May 10-12 1994.
- [2] A.D Evans (Ed.), *Designing with Field-Effect Transistors*, New York: McGraw-Hill, 1981.
- [3] Williams, Arthur B., *Electronic Filter Design Handbook*. New York, NY:McGraw-Hill, 1981.
- [4] Holt, C. A. *Electronic Circuits Digital and Analog*. New York: Wiley, 1978.
- [5] Mohammed El Sharkawy, *Real Time Digital Signal Processing Applications With Motorola's DSP56000 Family*, Prentice-Hall, 1990.
- [6] Alan V. Oppenhenheim and Ronald W. Schafer, *Discrete-Time Signal Processing*, Prentice Hall, 1989.
- [7] Antoniou, A.: *Digital Filters: Analysis and Design*, McGraw-Hill, New York, 1979.