DATA ACQUISITION SYSTEM DESIGN FOR ACOUSTIC MICROSCOPY

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Nadir Zafer Bağdatlıoğlu

August 1996

DATA ACQUISITION SYSTEM DESIGN FOR ACOUSTIC MICROSCOPY

A THESIS
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING
AND THE INSTITUTE OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

By
Nadir Zafer Baştürkmen
August 1996

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Hayrettin Köymen (Principal Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Enis Çetin

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Orhan Aytür

Approved for the Institute of Engineering and Science:

Prof. Dr. Mehmet Baray, Director of Institute of Engineering and Science

ABSTRACT

DATA ACQUISITION SYSTEM DESIGN FOR ACOUSTIC MICROSCOPY

Nadir Zafer Baştürkmen
M.S. in Electrical and Electronics Engineering
Supervisor: Prof. Dr. Hayrettin Köymen
August 1996

Conventional acoustic microscopes suffer from the complexity and low speed of their scanning mechanisms. The frame rates of these instruments are low, while their cost are high. In this work, a data acquisition system for acquiring raster image data at precise positions was designed for an acoustic microscope which uses a high speed, simple, and hence low cost scanning mechanism. The design is based on a data acquisition integrated circuit which is specifically designed to be used in this system. The implementation was done in sucha way that the system can be mounted to a standard personal computer. It is possible to obtain acoustical images on the monitor of the personal computer at rates as high as 50 lines per second, using this system.

Keywords : Acoustic microscopy, acousting imaging, X-Y scanner, image data acquisition.

ÖZET

AKUSTİK MİKROSKOPİ İÇİN VERİ EDİNME SİSTEMİ TASARIMI

Nadir Zafer Baştürkmen
Elektrik ve Elektronik Mühendisliği Bölümü Yüksek Lisans
Tez Yöneticisi: Prof. Dr. Hayrettin Köymen
Ağustos 1996

Geleneksel akustik mikroskopların tarama mekanizmalarının karmaşıklık ve düşük hız sorunları vardır. Bu aletlerin fiyatları yüksekken, çerçeve hızları düşüktür. Bu çalışmada, yüksek hızlı, basit, dolayısıyla ucuz bir tarama mekanizmasına sahip bir akustik mikroskop için kesin konumlarda satır imge verisi edinen bir veri edinme sistemi tasarlanmıştır. Tasarım, bu sistemde kullanılmak üzere tasarlanmış bir veri edinme tümdevresi üzerine kurulmuştur. Gerçekleştirme, sistemin standart bir kişisel bilgisayara monte edilmesine olanak sağlayacak biçimde yapılmıştır. Bu yolla bilgisayar ekranında saniyede 50 satır hızla akustik imge elde etmek mümkündür.

Anahtar Kelimeler : Akustik mikroskopi, akustik imgeleme, X-Y tarayıcısı, imge veri edinme.

To my family

ACKNOWLEDGEMENT

I would like to express my deep gratitude to my supervisor, Dr. Hayrettin Köymen for his guidance, suggestions, and encouragement throughout the development of this thesis.

I would like to thank to Dr. Enis Çetin and Dr. Orhan Aytür for reading and commenting on the thesis.

Special thanks to Ersin Başar for mounting the printed circuit board, and also to all Graduate Students in this department for their continuous support.

TABLE OF CONTENTS

1 INTRODUCTION 1

2 GENERAL DESCRIPTION OF THE SYSTEM 3

3 DATA ACQUISITION CHIP 7

3.1 DAIC Structure 8

3.1.1 Internal Structure 8

3.1.2 Pin Definitions 11

3.2 Running DAIC 13

3.3 Testing DAIC 14

4 DATA ACQUISITION CARD 19

4.1 Analog to Digital Converter Systems 19

4.1.1 Sample/Hold Amplifier 19

4.1.2 Analog To Digital Converter 22

4.1.3 Configuring the A/D Converter System 23

4.2 Chip Load, Read, Reset Logic and other Hardware 28
4.2.1 Hard Reset and Loading DAIC 28
4.2.2 Reading Data from DAIC 30
4.2.3 Resetting DAIC 31
4.2.4 Clock Signal and Power 33
4.3 Software for Data Acquisition Card 34
4.3.1 Software for Loading DAIC 34
4.3.2 Software for Reading Data from DAIC 35

5 MEASUREMENTS AND RESULTS 41
5.1 Timing Measurements 41
5.2 Images 46

6 CONCLUSION 52

APPENDIX 54

A LAYOUT OF DAIC 54

B LAYOUTS OF DATA ACQUISITION SYSTEM 56

C PROGRAM CODES 63

LIST OF FIGURES

2.1 Acoustical Imaging System 3

2.2 Scan Area 4

2.3 Top View of the Scanning Mechanism 5

2.4 Data Acquisition System Block Diagram 6

3.1 Position Signal 7

3.2 Internal Block Diagram of DAIC 10

3.3 Pin Definitions of DAIC 11

3.4 Chip Load Timing Diagram 13

3.5 Configuration to Observe CVT_POS Signal 15

3.6 Configuration to Observe Data Read from DAIC 17

3.7 Part of the Position Signal that DAIC Runs Correctly 18

4.1 General block diagram of the Data Acquisition Card 20

4.2 Pin Assignments of SHC803BM 21

4.3 Sample and Hold Modes of SHC804BM 21

4.4 Pin Assignments of ADC601 . . . . . 22'

4.5 Convert Command and Status Timings 22

4.6 Gain Adjustment for Unipolar Operation 23

4.7 S/H Amplifier and A/D Converter Connected Together 24

4.8 Circuitry of the A/D Converter System 24

4.9 Modification of CVT_POS Signal 26

4.10 Buffer for Analog Position Signal 26

4.11 Clock Pulse for Data Latches, Trigger and Capture of EOC_POS . . 27

4.12 Digital Position Data Latches and Tri-State Buffers 27

4.13 RR Signal Applied to DAIC and External Flip-Flops 28

4.14 Address Decoder and Chip Load Logic 29

4.15 Connections to DAIC 30

4.16 Chip Reset Circuitry 31

4.17 S/H Input and Output Voltages 32

4.18 Oscillator Circuitry for CLK and CLK_BAR Signals 33

4.19 DC-DC Converter Circuitry 34

4.20 Bit Order to be Outed from AL During Chip Load Operation 35

4.21 Interrupt Vector Table 36

4.22 OCW's of PIC 8259A 38

5.1 Clock Signal at 10MHz. 42

5.2 CVT_POS Pulses 42
5.3 A Single CVT_POS Pulse 43
5.4 CVT_POS2 Pulse 43
5.5 Hold Input of SHC804 43
5.6 Convert Signal to ADC601 44
5.7 Status Output From ADC601 44
5.8 CP_POS Signal 44
5.9 EOC_POS Signal Before Synchronization 45
5.10 EOC_POS Signal After Synchronization 45
5.11 CLK_BAR Signal in Synchronization Interval 45
5.12 Image from the card, f=50Hz. 47
5.13 Simulated Image, f=50Hz. 47
5.14 Image from the card, f=100Hz. 48
5.15 Simulated Image, f=100Hz. 48
5.16 Image from the card, f=200Hz. 49
5.17 Simulated Image, f=200Hz. 49
5.18 Image from the card, f=1KHz. 50
5.19 Simulated Image, f=1KHz. 50

A.1 Data Acquisition Integrated Circuit Layout 55

B.1 Data Acquisition System PCAD Scahematic: Part1 57

B.2 Data Acquisition System PCAD Scahematic: Part2 58

B.3 Data Acquisition System PCB Layout: Component Side 59

B.4 Data Acquisition System PCB Layout: Solder Side 60

B.5 Data Acquisition System PCB Layout: Silk Screen 61

LIST OF TABLES

3.1 Explanation of the Pin Functions 12

3.2 Input Combinations to Load DAIC 13

Chapter 1

INTRODUCTION

Acoustic microscopy has a wide range of applications [1, 2, 3]. Its imaging mechanism depends on the elastic response of the materials to acoustic waves, therefore provides information on local changes in elastic properties [4]. Thus, acoustic microscopy is particularly useful for the investigation of elastic anisotropy of the materials such as crystals and metal alloys, determination of cracks in integrated circuits, obtaining images of biological cells with different elastic properties. It is also a very powerful tool for the subsurface examination of opaque materials. This capability is used for determination of subsurface defects and observation of the adhesion properties of layered media [1, 2, 3].

Although the acoustic microscope is a very powerful tool, its use in industry and research is limited because of high cost [5]. This is mainly due to the following factors: (1) The existing generation of commercially available acoustic microscopes does not have up-to-date designs, particularly at high frequency electronics front-end; (2) their scanning mechanism are based on step-motor controlled precision $X-Y$ stages and are very costly. Furthermore, the acoustic lens designs require improvement for a wide range of applications [6]. The high cost of these instruments impeded the extensive use and knowledge generation particularly in the field of medicine.

These observations led to a project entitled EUREKA-525, which aims to develop a low cost acoustic microscope, that uses new electronics and computing technology. The design objectives of this project are as follows:

1) Updating, electronic circuitry through new designs, particularly at the high frequency end.

2) Using systems based on standard PC's rather than specific hardware, to control the instrument and measure and display acoustic data, using special software developed for this hardware.

3) To reduce the cost of $X - Y$ stage through a simpler mechanical scanner. During the project of EUREKA-525, a new type of scanner was designed, which is basically a spring rectilinear scanner.

In this work, a hardware and associated software for acquiring the acoustic data, controlled by scanning position information is designed as a card, which can be installed to a PC to obtain acoustic images on the monitor. Apart from being portable, this system can be used in acoustical imaging systems with fast scan rates, which are typically operating at 30-50 lines per second. In this system, it is possible to have acoustical images in seconds rather than minutes compared to older systems.

In Chapter 2, the proposed acoustic microscopy system is described. In Chapter 3, requirements for data acquisition system are discussed and a special integrated circuit (IC) designed for this system is introduced. In Chapter 4 the design of data acquisition system and related software is explained in detail. In Chapter 5, measurements and results are presented. Conclusions and discussions are presented in Chapter 6. Layout of data acquisition integrated circuit is given in Appendix A. Complete schematics of the whole data acquisition system, the printed circuit board (PCB) layout, which is obtained using PCAD software are given in Appendix B. In Appendix C, the assembly language codes of necessary software to run the system are presented.

Chapter 2

GENERAL DESCRIPTION OF THE SYSTEM

General block diagram of an acoustical imaging system is shown in Figure 2.1. This system can be divided into three main parts. RF circuitry and lens, positioning system, data acquisition system and PC.

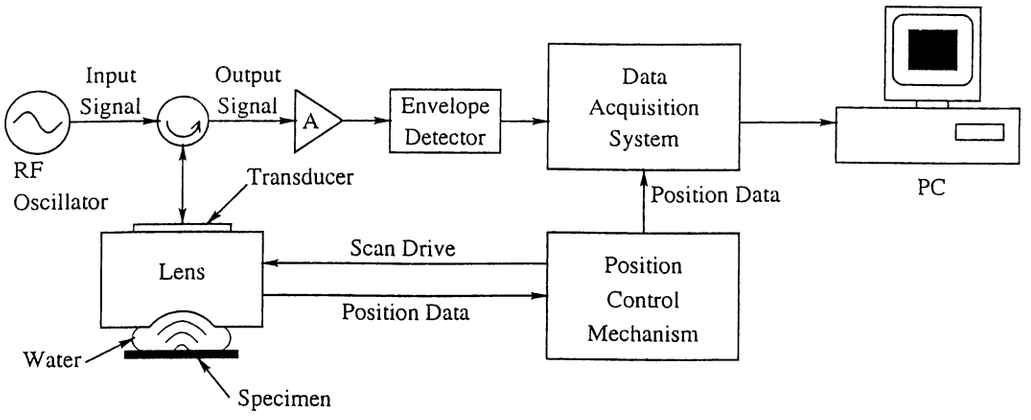


Figure 2.1: Acoustical Imaging System

The first step in image-making process is to convert an electrical signal produced by RF circuitry into an acoustic one by means of a piezoelectric transducer, placed onto the back surface of the aluminum or sapphire lens element[1]. When an electric

field is applied to a piezoelectric material, it changes its mechanical dimensions. Conversely an electrical field is generated in such a material when it is strained[7].

The lens has a cavity which is immersed in a fluid that makes contact with the specimen. The acoustical waves created by piezoelectric material are directed onto the specimen by means of this cavity [1, 2]. The reflected waves from the scanned surface are recollected by the lens and transmitted to piezoelectric material where they are converted back to electrical signals. These reflected signals carry the information about the scanned surface. To obtain acoustical image of the scanned surface, this information should be displayed using the appropriate position data. Position data is provided by the position control mechanism.

Position control mechanism consists of the scanning mechanisms, the electronic circuitry to drive this mechanism and the circuitry for determination of lens position.

Scanning in an acoustic microscope system can easily be achieved by using two step motors which enable lens movement both in X and Y directions, similar to raster scan used in TV's. Figure 2.2 shows an are to be scanned by the acoustic microscope. The lens first moves along X direction. After one row is scanned in this direction, positioning system increments Y position by ΔY and this new row is scanned. Although this method of scanning is quite simple, it suffers from the low

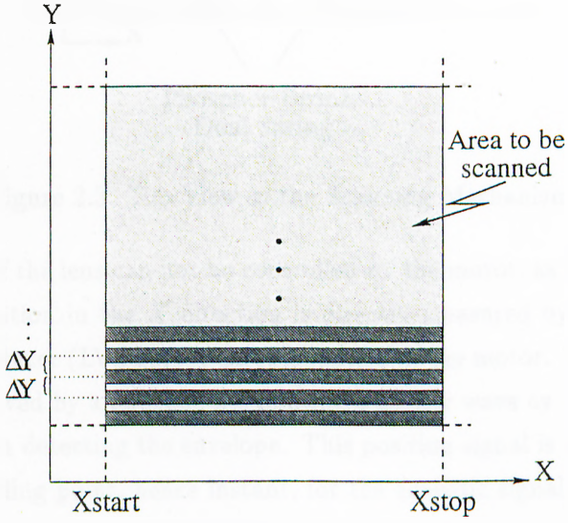


Figure 2.2: Scan Area

speed of motors, especially in X direction. It is apparent that if X-scan speed is increased, overall scan time can be decreased considerably.

One solution to this problem is vibrating the lens, independent from the other parts of the microscope, and collecting data at sample points, which are determined by another circuitry. The X direction movement of the lens can be achieved by connecting the lens to a rod whose other end is connected to a linear motor drive. This system is shown in Figure 2.3. These vibrations are most effectively obtained at the mechanical resonance frequency of the lens-spring system which is arround 50Hz and may vary according to the weight of the lens. Linear motor drives this system at frequencies not exceeding 50Hz. The amplitude of these vibrations are upto 5mm in X direction [5].

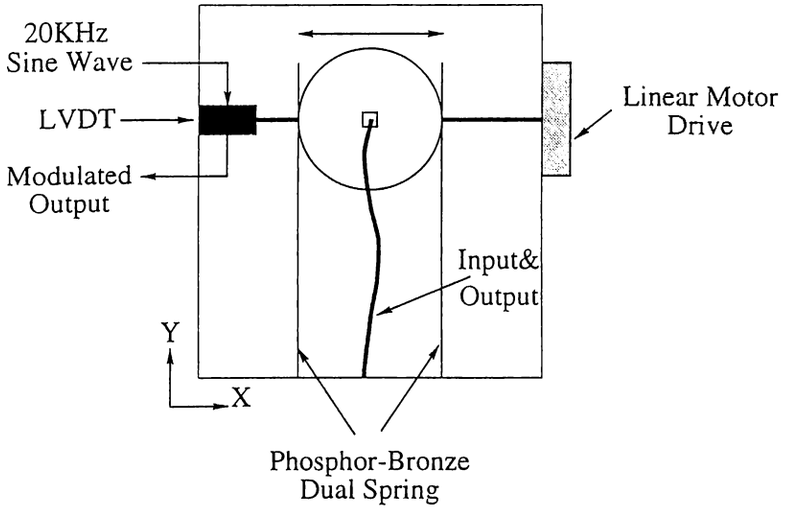


Figure 2.3: Top View of the Scanning Mechanism

The position of the lens can not be controlled by the motor, as in the case of step motors, hence position in the X direction is directly measured by a Liner Variable Differential Transducer (LVDT), mounted opposite to the motor. The measurement of position is achieved by amplitude modulating a carrier wave by the X position information and then detecting the envelope. This position signal is used to determine the accurate sampling point, hence instant, for the acoustic signal.

Acoustic data from lens and position data from position control mechanism must be combined to obtain acoustical images on the monitor of the PC. This job is

carried out by the data acquisition system. A block diagram of this system is given in Figure 2.4. Data acquisition system consists of two Analog to Digital Conversion

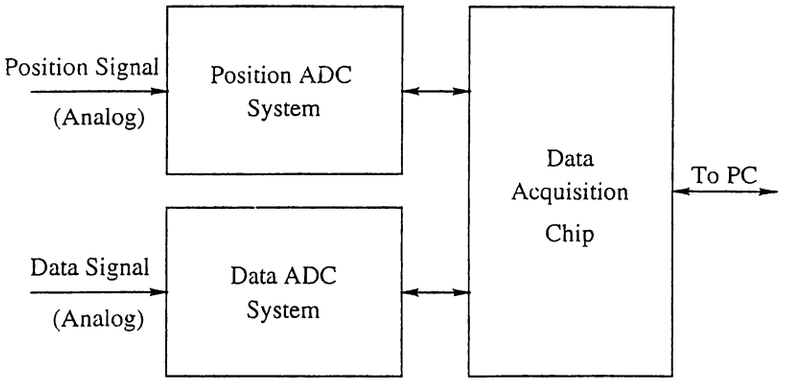


Figure 2.4: Data Acquisition System Block Diagram

(ADC) systems, one for position signal and the other for acoustic data signal. These systems digitize analog position and acoustic data signals to be processed in the computer environment. Another important part of the data acquisition system is a data acquisition chip shown in the above diagram which was designed at Bilkent for this project. This chip performs the collection of acoustic data and interfacing with computer. The data is collected and sent to the computer row by row. The acoustic data vs. position is displayed on the computer screen and acoustic images are obtained.

Chapter 3

DATA ACQUISITION CHIP

The data acquisition chip was designed as the project work of Introduction CMOS VLSI Design course, by a group of 6 people, using CADENCE design system ES2 0.7μ technology, and was fabricated in France. The layout of the chip is presented in Appendix A. The chip will be referred as Data Acquisition Integrated Circuit (DAIC) from now on. The design purpose of DAIC was getting acoustic data which is collected by an acoustic lens. The requirement from this system is to collect acoustic data at equidistant points with reference to position signal driving the scanning mechanism. Since the position signal is a sine wave, time intervals between sampling points do not change linearly. This situation is shown in Figure 3.1. Acoustic data

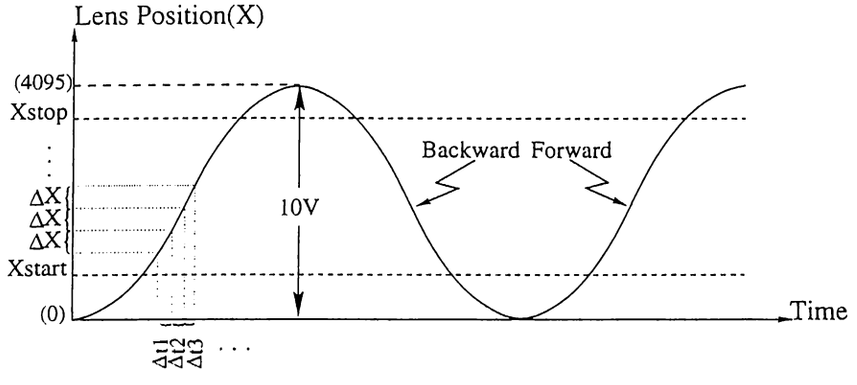


Figure 3.1: Position Signal

should be collected between a start point and an end point, namely Xstart and

Xstop points, in the near-linear region of the position signal. It can be collected in both directions. DAIC has a RAM inside and can store up to 1024 data values. As soon as Xstop position is reached, DAIC interrupts computer and data is read. This completes collecting one row of acoustic data, and next reading cycle is waited. DAIC is interested only in the X position value while acquiring acoustic data and it is assumed by the data acquisition system that a new row is being scanned when the next position cycle arrived.

3.1 DAIC Structure

Basic function of DAIC, as explained above, is to check position signal and get acoustic data at appropriate positions. Since it is a completely digital chip, it needs the analog position signal and analog acoustic data to be converted into digital form by using Analog to Digital Converters(ADC's). The details of the rest of the system will be explained in the next chapter. Now it is sufficient to assume that the digital information is ready at the inputs of DAIC whenever it is needed. Digital data is represented in 12 bit scale. Maximum value of the position or data signal corresponds to 4095 while minimum is 0.

3.1.1 Internal Structure

Internal structure of DAIC consists of the following basic parts:

Registers:

The registers inside DAIC are 12-bit registers since the A/D conversion is done in 12 bits. DAIC has the following three important and externally writable registers:

•**Xstart Register:** Start position for collecting acoustic data is written into this register. DAIC compares the current position with the content of this register before starting to collect data. As soon as the position is equal to or greater than Xstart value, collection operation begins.

•**Xstop Register:** Stop position for collecting acoustic data is written into this

register. DAIC compares the current position with the content of this register before stopping to collect data. As soon as the position is greater than Xstop value, collection operation stops.

•ΔX **Register:** Distance between two consecutive data points is written into this register. Immediately after getting a data at a certain position, DAIC adds the value in this register to that position value, and starts to wait for this new position value to come.

In addition to these three registers, DAIC has two internal registers:

•X **and** X**prev Registers:** This registers are used to keep previous, current or waited position values for the comparison and addition operations inside DAIC.

A block diagram of the internal structure of DAIC is given in Figure 3.2.

Arithmetic Logic Unit (ALU):

This unit consists of 12 bit adders and comparators. It compares the current position with Xstart, Xstop registers, adds ΔX to last data collection position value to determine the waited data collection position. This part also have some dummy registers to exchange data between registers and to manage arithmetic operations on the necessary registers.

Read Only Memory (ROM):

DAIC has a program which starts to run after loading the registers. This program is kept in a ROM inside DAIC. Program contains 48 lines of 5 bit instructions. Each instruction corresponds to an operation such as addition or comparison of the contents of two registers, collecting data etc. The program instructs DAIC how to manage the task of collecting acoustic data.

Program Counter:

This counter points to the current line of program inside the ROM to be executed at each step of the program. Counter does not index the ROM from first line to last line consecutively, rather it can jump from one line to another when a branching instruction is executed. This provides the program to do the tasks that

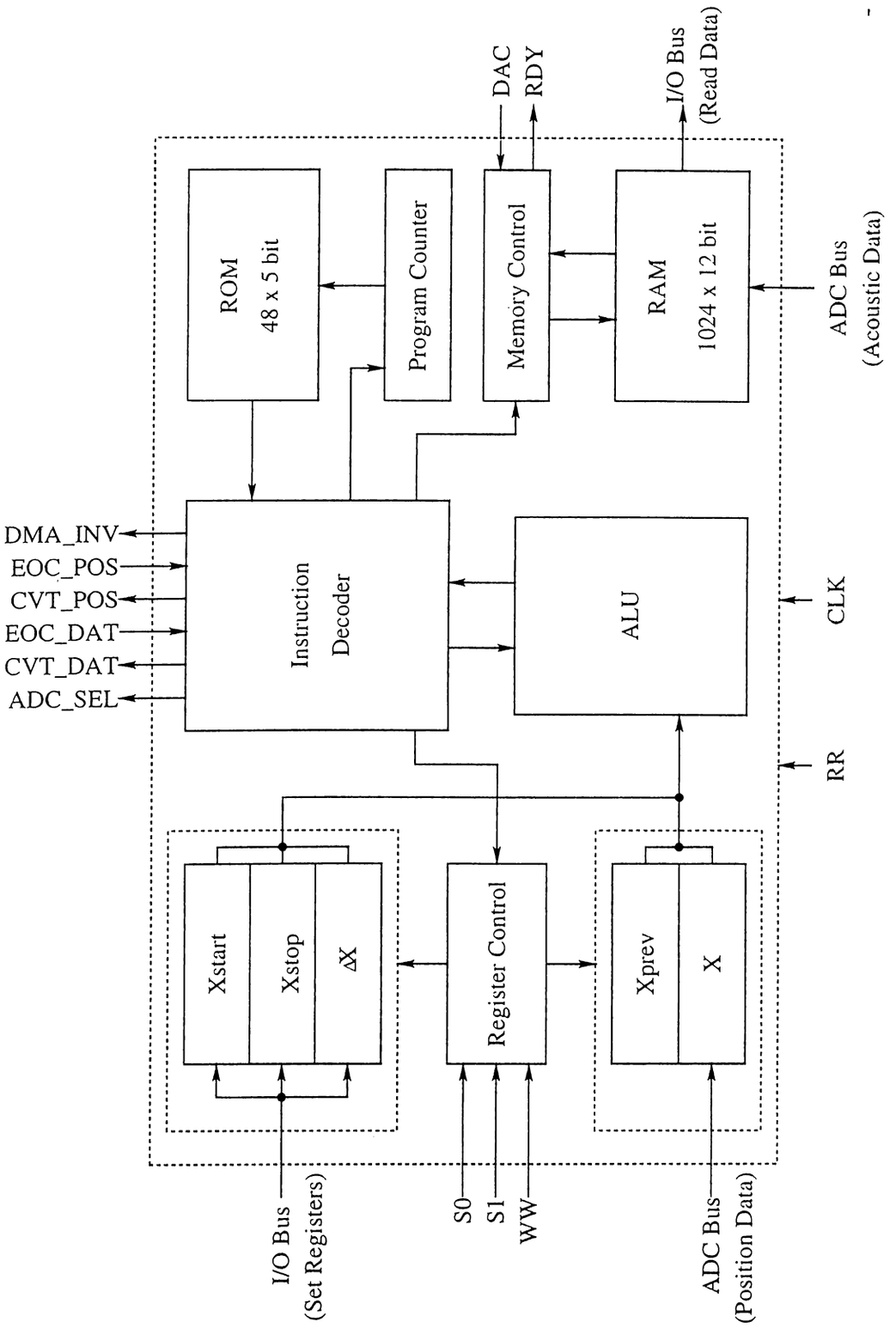


Figure 3.2: Internal Block Diagram of DAIC

need conditional jumps and looping.

Instruction Decode Logic:

This is the part where the program written inside the ROM is executed. At each step, 5 bit opcode of the program line that is pointed by program counter, is entered as the input of the instruction decode logic. This logic circuitry evaluates the corresponding output signals for that opcode. Output signals provide operations such as loading of registers, additions, comparisons etc.

Random Access Memory (RAM):

This is a 1024 $\times$ 12 bit RAM and keeps one row of acoustic data, collected during one cycle. Contents of this RAM is read by computer after one scan cycle is completed.

3.1.2 Pin Definitions

Pin names and numbers of DAIC are shown in Figure 3.3. The functions of these pins are presented in Table 3.1.

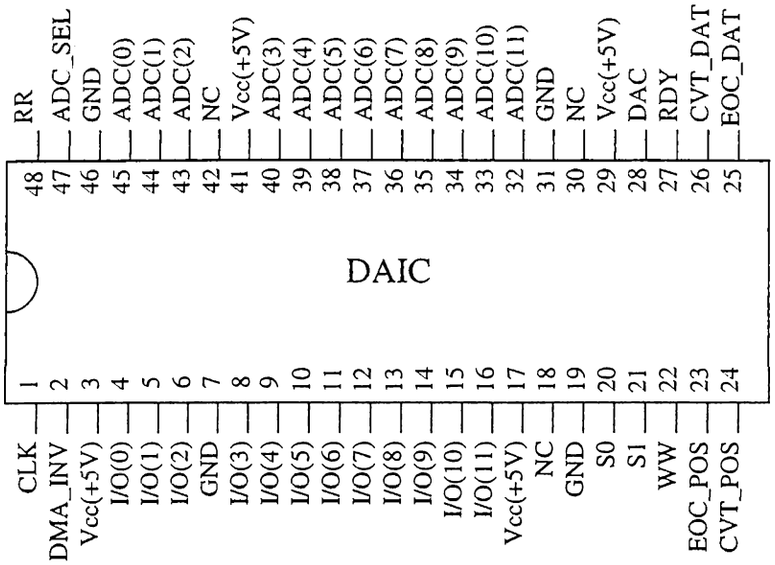


Figure 3.3: Pin Definitions of DAIC

Pin #	*Name*	*Type*	*Function*
1	CLK	Input	An external square wave clock signal is applied to this pin to run the chip.
2	DMA_INV	Output	This pin makes a transition from low to high when Xstop position is reached. It is used to invoke computer to read data at the end of each scan cycle.
3,17,29,41	Vcc	Analog	Power pins. Connected to +5V.
4-6,8-16	I/O [0:11]	Input/ Output	12 bit bus connected to the data bus of computer. Chip is loaded or data from chip is read via this bus. Pin4:LSB, Pin16:MSB.
7,19,31,46	GND	Analog	Ground pins.
18,30,42	NC		Not connected.
20	S0	Input	It is used to write internal registers of the chip together with pins S1 and WW.
21	S1	Input	See S0.
22	WW	Input	See S0.
23	EOC_POS	Input	Acknowledge signal from position ADC system, indicating that position conversion is complete and position value is ready in position buffers. It should stay high more than one clock period.
24	CVT_POS	Output	A one clock cycle duration pulse is sent to position ADC system to start conversion of position, whenever a position data is required.
25	EOC_DAT	Input	Acknowledge signal from data ADC system, indicating that data conversion is complete and that value is ready in data buffers. It should stay high more than one clock period.
26	CVT_DAT	Output	A one clock cycle duration pulse is sent to data ADC system to start conversion of data, whenever an acoustic data is required.
27	RDY	Output	Acknowledge signal to computer, to indicate that, data that is being read from the RAM inside the chip is ready on the I/O Bus of the chip. It stays low until data is ready.
28	DAC	Input	Read signal from computer. When it is made low, chip puts the next data from RAM onto the I/O Bus. I/O Bus is made Hi-Z upon this signal goes high again, concluding that current data was read.
32-40,43-45	ADC [11:0]	Input	This is a multiplexed 12 bit bus connected to outputs of position and data ADC buffers. Each of these buffers are selected in turn by ADC_SEL signal, and values read are written into internal registers accordingly. Pin32:MSB, Pin45:LSB.
47	ADC_SEL	Output	ADC select signal is a square wave of half the frequency of the clock signal. It chooses position and data ADC tri-state buffers in turn, and those data is written into corresponding internal registers. Inverse of this signal should be entered into position ADC buffers (both buffer enables are active low).
48	RR	Input	A low to high transition must occur on this pin before doing any operation on the chip. This is required for resetting some internal registers.

Table 3.1: Explanation of the Pin Functions

3.2 Running DAIC

To make DAIC function properly, a reset signal should be sent to the RR input before doing any operation. This operation sets some registers in DAIC to appropriate initial states. Reset signal should be a low-to-high transition on Pin48 and should stay high during the operation of DAIC.

The next step is the loading of the Xstart, Xstop and ΔX registers. For this operation three pins, S0, S1, and WW are used. To write into a specific register, appropriate combination of these input signals should be applied to S0 and S1 inputs. When WW input is made high, whatever exists on the I/O Bus of DAIC is written into that register. The input combinations of these pins are given in Table 3.2 (Convention: High Voltage = 1).

S0	*S1*	*WW*	*Operation*
1	0	1	Write to Xstart register
0	1	1	Write to Xstop register
1	1	1	Write to ΔX register
0	0	1	Reset chip
0	0	0	Run program

Table 3.2: Input Combinations to Load DAIC

After the registers are loaded, a reset signal is required to initialize the program. This reset signal is applied through S0, S1 and WW pins.

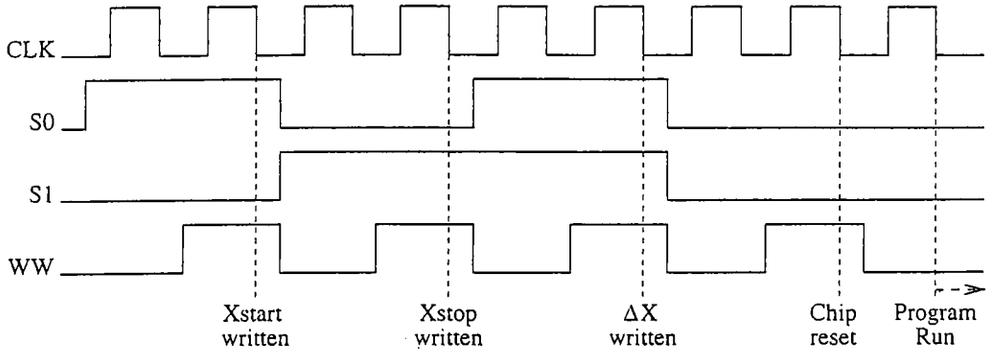


Figure 3.4: Chip Load Timing Diagram

This combination results in an internal reset pulse, that causes the initialization of the program which will be executed by DAIC. Finally setting all these three inputs

to low value causes the program of DAIC to run. The timing diagram for all of these operations are presented in Figure 3.4.

The program inside DAIC collects data using the following algorithm:

Step 1: Wait for position value to decrease
Step 2: Wait for position value to increase
Step 3: Set waited position to Xstart
Step 4: While position value is smaller than Xstop do
  Step 4.1: Get current position
  Step 4.2: If current position is equal to or greater than waited position
    Step 4.2.1: Get acoustic data and write it into RAM
    Step 4.2.2: Increment waited position by ΔX
Step 5: Call computer to read data stored in DAIC
Step 6: Return to Step 2

If the first position values read by DAIC are somewhere between Xstart and Xstop, while position is increasing, acoustic data would be collected by DAIC. The first step of this algorithm prevents DAIC to collect data in this case and hence, prevents collecting the first row of acoustic data starting from the middle of that row. Waiting for a position decrease first, insures that Xstart will be caught while position is increasing. The rest of this algorithm is easy to trace, data is collected between Xstart and Xstop positions in steps of ΔX while position is increasing. After one row of data is collected, i.e. Xstop position is reached, an invoke signal to computer is sent and program returns Step 2, where it waits for Xstart.

3.3 Testing DAIC

Test was done on a breadboard, by hand tracing, i.e. applying necessary signals to appropriate inputs, by hardwiring. For example, after resetting DAIC from Pin48, Xstart value is set on I/O Bus bit by bit as high or low voltage values, S0 is made high, S1 is made low and WW is first made high and then low while the Xstart value is stable on I/O Bus. After other loadings are done and chip reset signal is sent, DAIC starts to run upon setting S0, S1, and WW inputs to low voltage.

When the program starts to run, a CVT_POS signal, duration of one clock cycle, is sent from pin 24 and an EOC_POS signal is waited by DAIC. After setting an appropriate position value, that is a value smaller than the waited position, on ADC Bus, CVT_POS signal can be observed on an oscilloscope screen, as soon as EOC_POS signal is kept high. That is because the position value on ADC Bus does not change, and chip sends sequential CVT_POS signals until the waited position value is reached. Upon sampling EOC_POS signal high, DAIC gets the current position from ADC Bus and immediately sends another CVT_POS signal to get the next position value. For a specific case where,

$X\text{start} = 10$, $X\text{stop} = 20$, $\Delta X = 2$, following position values were applied:

$X = 8, 6, 2, 6, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21$

Before Xstart, DAIC sent CVT_POS signals and got position values when EOC_POS signal was sent. When position value was made 10 (Xstart value) and EOC_POS was made high, a CVT_DAT signal, duration of one clock cycle, was sent by DAIC. Since this signal is a pulse of very short duration, sent once per appropriate position value, It is very difficult to observe it on an oscilloscope screen.

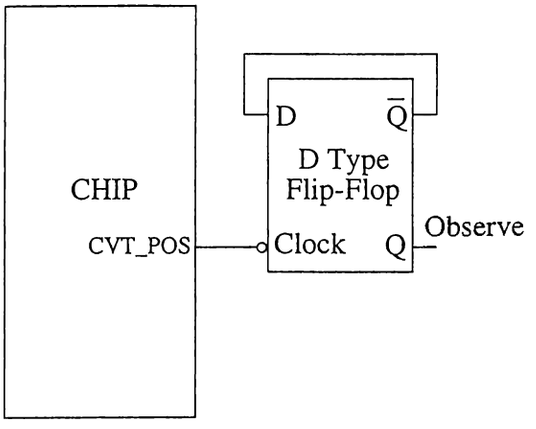


Figure 3.5: Configuration to Observe CVT_POS Signal

To be able to observe it, this pin is connected to the clock input of a D type flip-flop, whose $\bar{Q}$ is connected to its input, as shown in Figure 3.5. The change in the state of the flip-flop indicates that a CVT_DAT was sent. Placing data value on ADC Bus, EOC_DAT signal is made high, to acknowledge DAIC that the data is ready. Then,

trace was continued with subsequent position values. A CVT_DAT signal was sent by DAIC at position values:

X = 10, 12, 14, 16, 18, 20

For those positions, corresponding acoustic data values put onto ADC Bus were:

Data Written = 1, 2, 3, 4, 5, 6

When position value reached Xstop - ΔX, an invoke signal to computer via DMA_INV pin was sent, by a low to high transition on this pin.

After DMA_INV signal is sent, DAIC waits for data inside the RAM to be read by the computer. Read signal from computer is DAC. This is an active low signal, and when there is a high to low transition on this pin, DAIC puts the next data available from RAM. Here, there is a hand-shake procedure between DAIC and computer. As soon as DAC is made low by the computer, DAIC sets its RDY pin, which should be connected to -IO_CH_RDY pin of the computer, to low, and keeps it low until the data is ready on I/O Bus. After this signal is high again, computer understands that data is ready, and reads it. Upon reading data, computer also sets DAC input of DAIC to high value again, which was kept low during the read operation. Then DAIC concludes that computer has read the data and makes I/O Bus Hi-Z, whose initial state was also Hi-Z. When the next DAC signal is received, this procedure is repeated.

During the hand-trace of DAIC, to be able to catch data written on I/O Bus, 74273 latches were used. Bus is connected to the inputs of the latches. RDY signal of DAIC is connected to the clock input of the latches. During low to high transition of this pin, latches captured the data on I/O Bus. Applying consecutive pulses to DAC pin, content of the RAM was read.

The values read from DAIC were:

Data Read = 2, 3, 4, 5, 5, Random, Random, ...

This shows that the first data was lost, and the last two data are the same. If the internal registers of DAIC is initially loaded by Xstart - ΔX and Xstop + ΔX

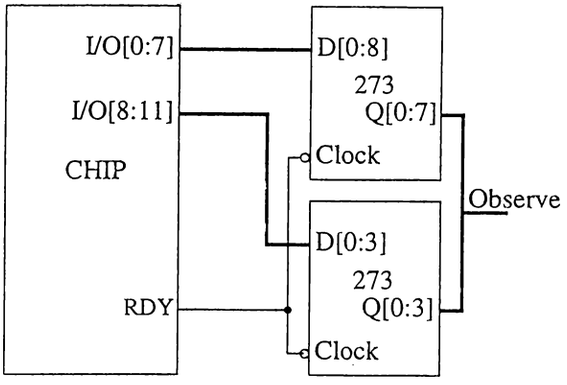


Figure 3.6: Configuration to Observe Data Read from DAIC

both of these problems are eliminated.

During this trace, two important bugs of DAIC are observed. First one is that, if the first position value read by DAIC is somewhere between Xstart and Xstop positions, 2 samples of the DAIC out of 10 sample chips produced, send CVT_POS and DMA_INV signals immediately, which is not the case in simulations. 7 DAIC samples wait in this situation, but they send DMA_INV signal immediately after the position values start to decrease. 1 DAIC sample does not work at all. However remaining 9 samples collect data correctly, if initially position is decreasing and its value is less than Xstart value.

Second bug of DAIC is that, the program does not return to step 2 after one read cycle is complete.

Both of these bugs can be eliminated, if the program of DAIC is reset through S0, S1 and WW inputs, at every position cycle when position value is decreasing and its value is less than Xstart value. Part of the position signal that DAIC runs correctly is given in Figure 4.7.

A final note about the run of DAIC is that, if the frequency of the CLK signal applied is above 11 MHz, DAIC writes position values into RAM, instead of acoustic data values. This is originated from the strategy used in multiplexing ADC Bus. The DAIC chooses position and data ADC buffers using ADC_SEL signal. Its frequency is half the frequency of CLK signal applied. The data coming from the enabled

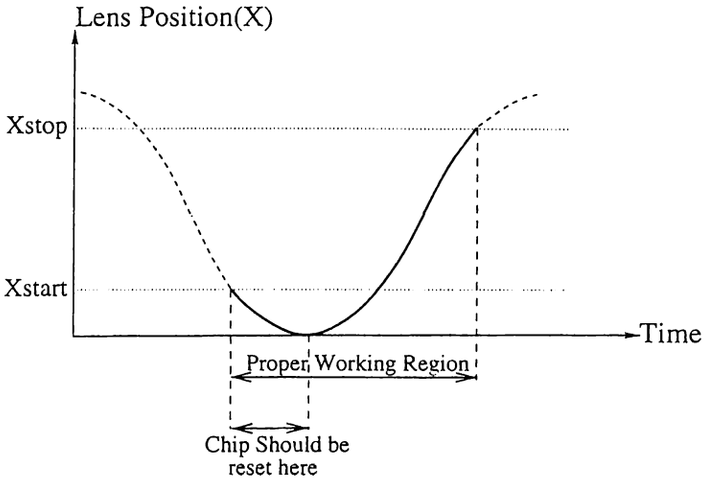


Figure 3.7: Part of the Position Signal that DAIC Runs Correctly

buffers is written into corresponding internal registers. If the implementation of multiplexing were done such that, ADC_SEL pin would switch to data ADC buffers only when acoustic data was being waited, DAIC could be run at a higher frequency, which would providing a higher sampling rate of position signal, which affects the resolution of the system.

Chapter 4

DATA ACQUISITION CARD

In this chapter, the system that uses DAIC will be explained in detail. In the light of the explanations of the last chapter, a general block diagram of this system can be given as in Figure 4.1. In this diagram, there are two Analog to digital (A/D) converter subsystems shown, that communicate with DAIC, and also the logic which is necessary to communicate with the computer. Analog to Digital Converter Systems will be explained first and then the other necessary logic around DAIC will be presented.

4.1 Analog to Digital Converter Systems

This part consists of two chips. First one is a Sample/Hold Amplifier, and the other one is an Analog to Digital Converter.

4.1.1 Sample/Hold Amplifier

Sample/Hold amplifiers are commonly used to hold input voltages to an A/D converter constant during conversion. Digitizing errors result if the analog signal being digitized varies excessively during conversion. To insure the accuracy of the output data, the analog input signal to A/D converter must not change more than 1/2 LSB

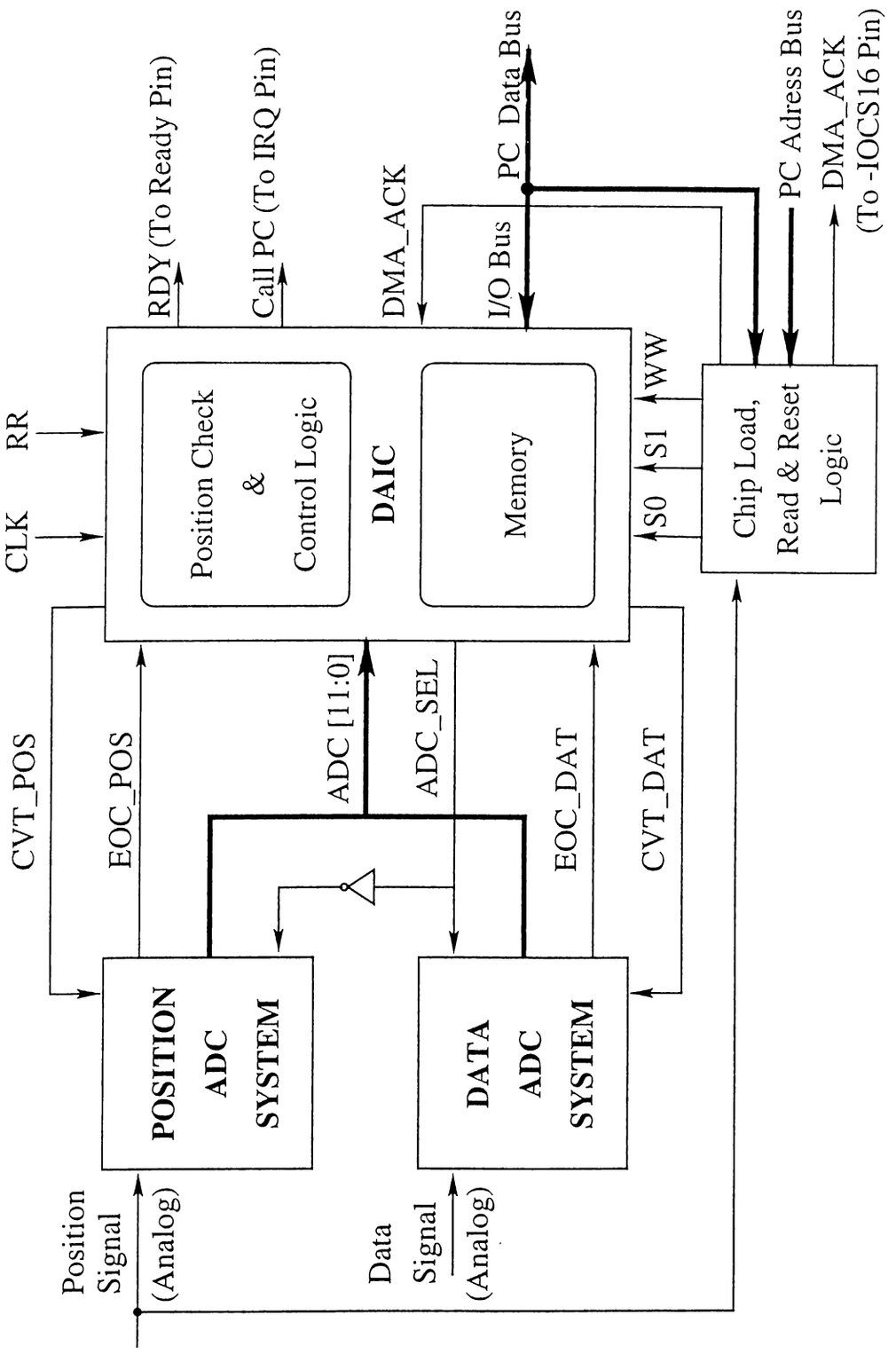


Figure 4.1: General block diagram of the Data Acquisition Card

during the conversion.

Sample/Hold amplifier used in this system is a Burr-Brown SHC804BM. This is a high speed S/H amplifier designed for use in fast 12-bit data acquisition systems and signal processing systems. This chip acquires a 10V signal change in less than 350ns to ±1/2 LSB for 12-bit systems. Pin diagram for this chip is given in Figure 4.2.

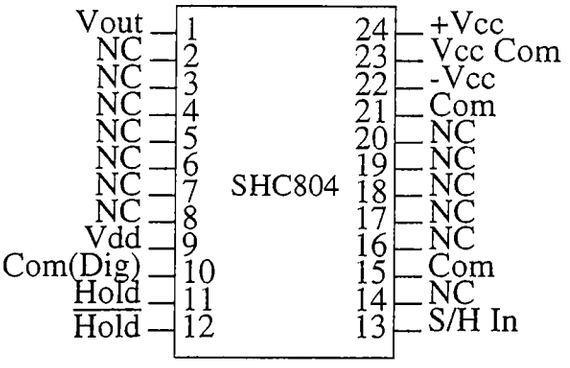


Figure 4.2: Pin Assignments of SHC803BM

A TTL logic "0" at Pin11 or a logic "1" at Pin12 switches the SHC804 into the Sample (track) mode. In this mode the chip acts as a unity-gain inverting amplifier.

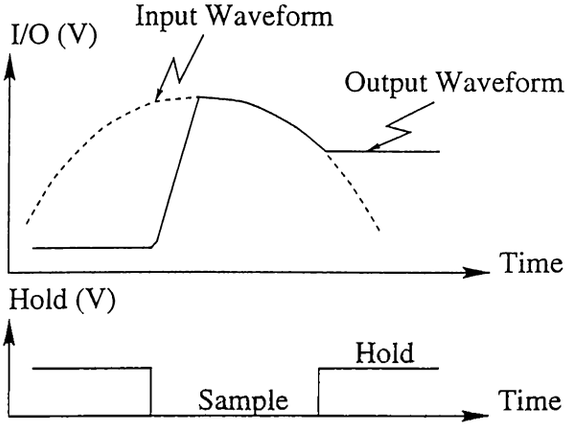


Figure 4.3: Sample and Hold Modes of SHC804BM

A logic "1" at pin 11 or a logic "0" at pin 12 switches the SHC804 into the Hold mode. In this mode, the output voltages will be held constant at the value present when the Hold command is given. Figure 4.3 shows these two modes when logic "0"

and logic "1" values are applied to Pin11.

4.1.2 Analog To Digital Converter

A Burr-Brown ADC601 was used in the system. ADC601 is a high-speed, successive approximation analog-to-digital converter, with internal reference and clock. Conversion time is internally set to 900ns, but can be adjusted down to 700ns by increasing the voltage on Pin19 (Clock Rate Control), in the expense of a higher linearity error. Analog signal input ranges of ±5V, ±10V (bipolar operations) and 0V to -10V (unipolar operation) are possible.

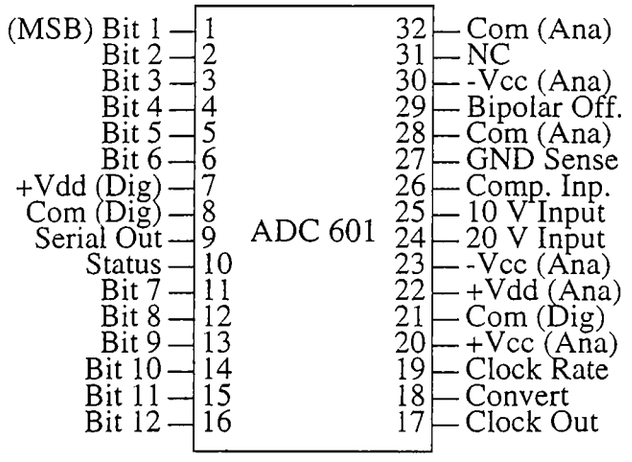


Figure 4.4: Pin Assignments of ADC601

A low-to-high transition on Pin18 (Convert Command) starts conversion. This input should remain high during conversion. If a convert command held at low voltage for a minimum of 50ns, current conversion will be reset and new conversion

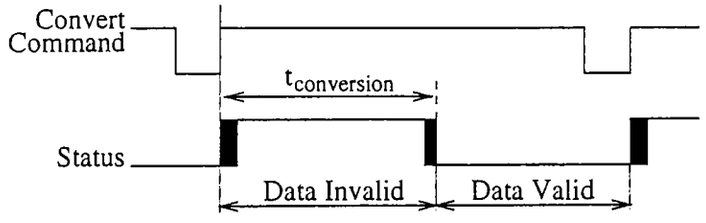


Figure 4.5: Convert Command and Status Timings

will start on the next rising transition, regardless of the state of the converter prior to the convert commands being received. Pin10 is the conversion status strobe and remains high during data conversion. It goes low after all 12 bits are valid. These signal timings are shown in Figure3.5.

Input range scaling is done using pins 26 and 29. For bipolar operation Pin26 and Pin29 should be connected together. For unipolar operation Pin26 is left open and Pin29 is connected to ground. For the second case a series 10Ω resistor should be connected to the analog input. Gain and offset errors may be trimmed to zero using external trim potentiometer for unipolar operation as shown in Figure 4.6.

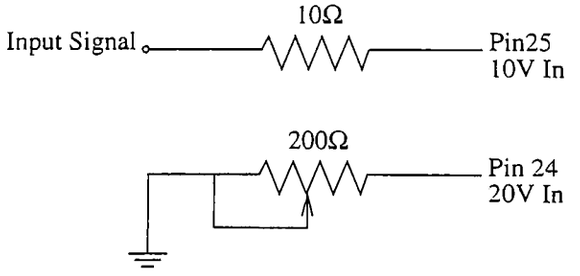


Figure 4.6: Gain Adjustment for Unipolar Operation

Adjustment procedure is the following: The gain potentiometer is adjusted such that the output transition 000...000 to 000...001 occurs at the correct end point transition voltage which is given as -10V + 3/2LSB (1LSB = 2.44mV).

The output code generated is Straight Binary (logic "0" true) for unipolar operation and Bipolar Offset Binary (logic "1" true) for bipolar operations.

4.1.3 Configuring the A/D Converter System

The block diagram in Figure 4.7 shows the S/H amplifier and A/D converter connected together. A convert command, that is applied to convert pin of the A/D converter, causes STATUS pin to go high, and makes the S/H amplifier hold the current voltage value at its output until conversion is done. As soon as conversion is complete, the digital data at the output of A/D converter is latched.

In fact, timing considerations are a little bit more detailed and they will be

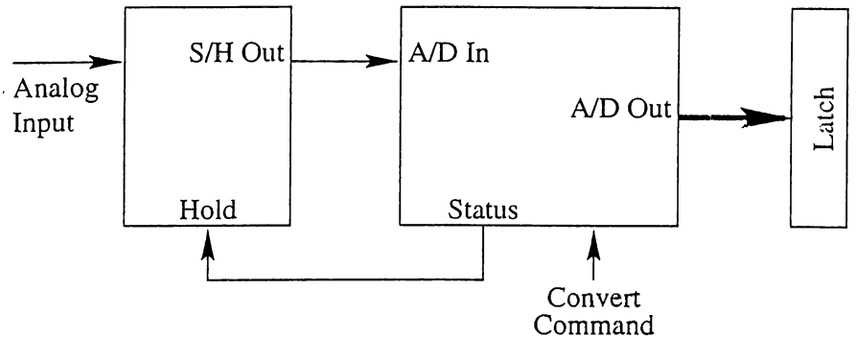
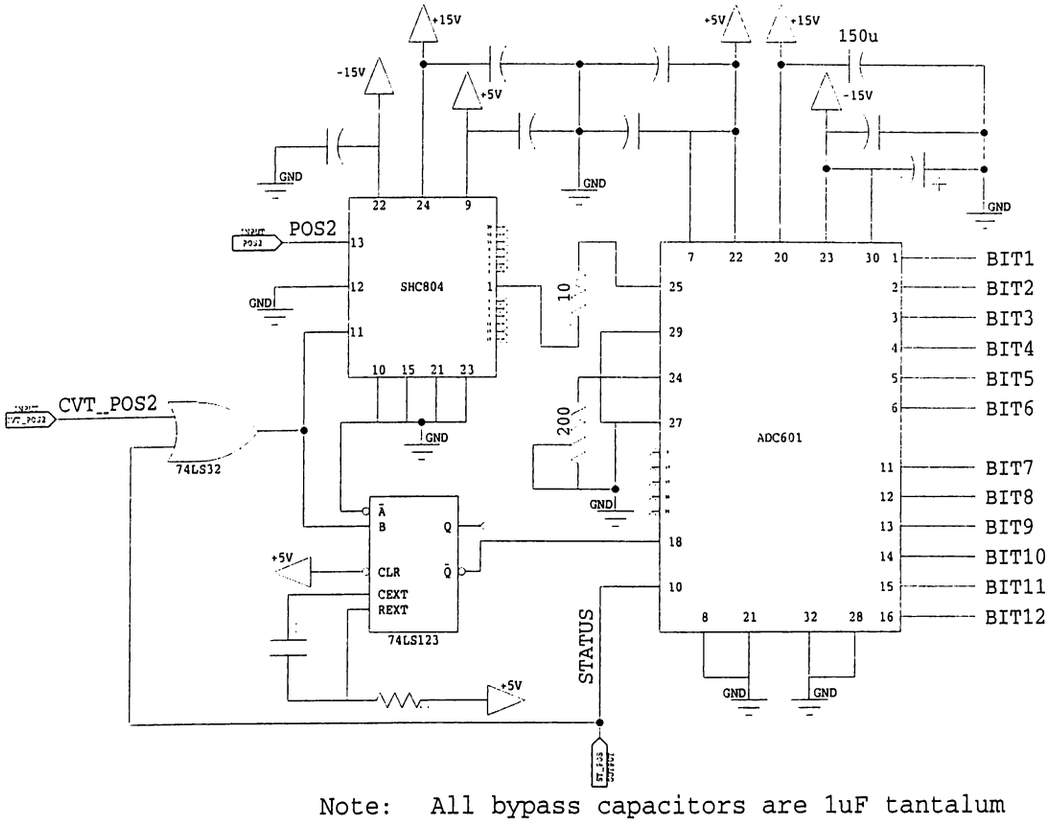


Figure 4.7: S/H Amplifier and A/D Converter Connected Together



Note: All bypass capacitors are 1uF tantalum unless otherwise specified.

Figure 4.8: Circuitry of the A/D Converter System

explained on the real circuitry schematic of the A/D Converter System. The circuit schematic is given for position A/D converter system in Figure 4.8, which is identical to data A/D converter system except the signal names. The arguments here are directly applicable for the second system, provided that the signal names are replaced by the corresponding ones in that system.

In Figure 4.8 a convert command is shown as CVT_POS2, which is different than the convert command of block diagram of Figure 4.7. The convert command shown in block diagram is Pin18 of ADC601 which is connected to $\bar{Q}$ of 74LS123. An external convert command is sent to system to start conversion, which is CVT_POS2 in this figure. Status output of ADC601 is low out of conversion periods, and CVT_POS2 signal should be a positive pulse. Before this signal is applied, output of 74LS32 OR gate is also low. When an external convert command is sent, this output makes a low-to-high transition, that causes SHC804 to hold its output for conversion and triggers the 74LS123 monostable multivibrator, causing a negative pulse at $\bar{Q}$ output, whose duration is determined by the values of R and C connected to this chip. ADC601 requires a 50ns or more duration low pulse at Pin18 before starting a conversion, so this negative pulse should be at least 50ns duration but not more than 400ns as indicated in the data sheet of ADC601. The rising edge of this pulse is the convert command shown in the block diagram. After conversion is started, STATUS output of ADC601 goes high and keeps the output of the OR gate high to make sure that SHC804 is in the hold mode till the end of conversion. An important point here is that, the duration of CVT_POS2 pulse must be adjusted such that STATUS goes high before this pulse returns back to low. Otherwise output of the OR gate makes low-to-high and high-to-low transitions, which retrigger 74LS123 and resets the current conversion. All of these signal timings are demonstrated in the next chapter, with oscilloscope traces taken from the actual circuitry.

CVT_POS2 signal in fact, would be the CVT_POS signal from DAIC, but this signal has a duration of one cycle of the CLK signal (10 MHz), and it does not satisfy the timing requirement explained above. Therefore CVT_POS signal from DAIC is modified by using another monostable multivibrator as shown in Figure 4.9. to obtain CVT_POS2 signal. In this configuration, rising edge of CVT_POS signal triggers 74LS123 and produces a positive pulse at output Q. This signal is used as CVT_POS2 and its duration can be adjusted by R and C connected to

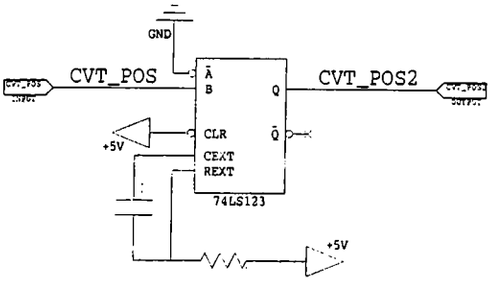


Figure 4.9: Modification of CVT_POS Signal

74LS123, to satisfy the necessary timing requirements.

Another input signal POS2 is shown in Figure 4.8. This is the analog position signal at the output of a unity gain amplifier, LF356, whose input is the POS signal as shown in Figure 4.10. This buffer is used to prevent any loading effect on the position signal of SHC804.

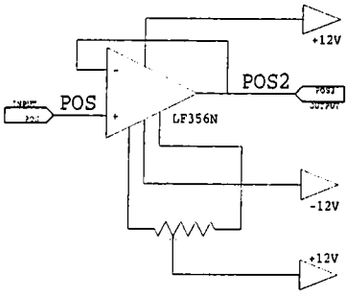


Figure 4.10: Buffer for Analog Position Signal

LF356 is a JFET input operational amplifier, suitable for high impedance buffers. It also has fast settling time (1.5μs to 0.01%) and large gain-bandwidth-product (20 MHz).

Immediately after the conversion is complete, STATUS output of ADC601 becomes low, indicating that the parallel data is ready at outputs BIT1 through BIT12. This data should be latched, before a new convert command changes it. Unfortunately high-to-low transition of status strobe is not suitable to catch the data at the output. A time delay is needed after this transition for proper data value to be written into latches. This delay is provided again using a monostable multivibrator, which is triggered by high-to-low transitions of the status probe of ADC601. This

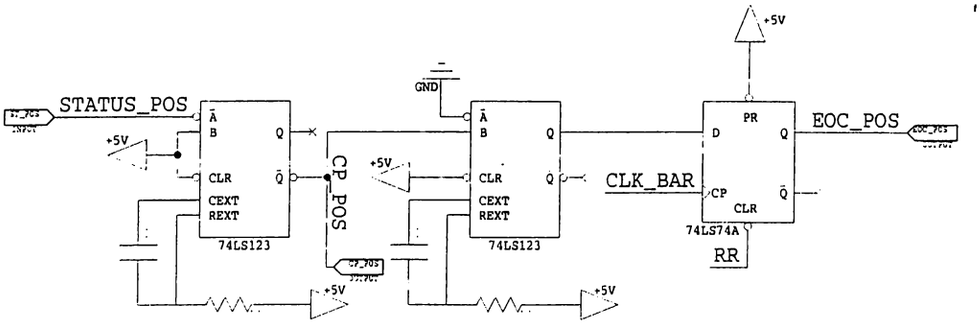


Figure 4.11: Clock Pulse for Data Latches, Trigger and Capture of EOC_POS

is shown in Figure 4.11, STATUS_POS signal, produces a negative pulse at the $\bar{Q}$ of 74LS123, CP_POS, which is used as clock signal of 74LS273 latches as in Figure 4.12. These latches catch digital data when a low-to-high transition occurs, and the delay provided by the negative pulse allows data become completely stable before this operation.

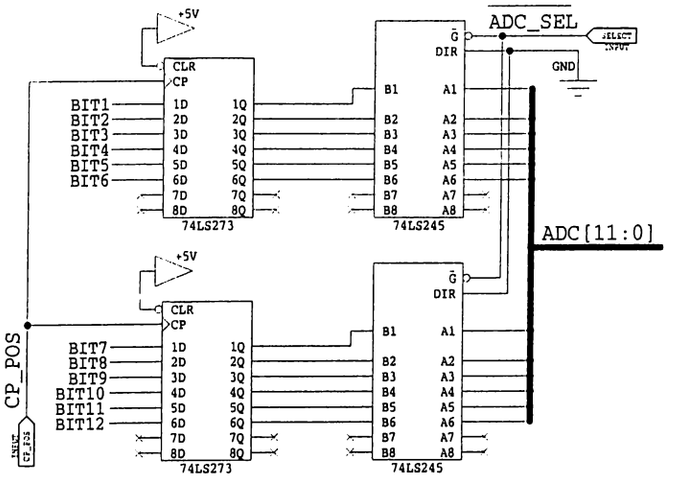


Figure 4.12: Digital Position Data Latches and Tri-State Buffers

The outputs of 74LS273 latches are connected to 74LS245 tri-state buffers. These buffers are needed because of the multiplexed ADC Bus of DAIC. Both data and position buffers are connected to this bus. The ADC_SEL signal determines the buffer pair to be read. This signal should be inverted before it is entered into position buffer enables. In this way, position and acoustic data is written into DAIC one after the other.

As can be observed from Figure 4.11, CP_POS is also used to produce another pulse, at the Q of a 74LS123, as end-of-conversion signal. Since the data is ready at the outputs of the latches, DAIC should be acknowledged about this situation. This end-of-conversion pulse is caught by a D type flip-flop, 74LS74, to synchronize the acknowledge signal with CLK signal driving DAIC. To the clock input of 74LS74, inverted version of CLK signal is applied. The output Q of flip-flop is connected to EOC_POS pin of DAIC. By doing so, end-of-conversion signal is caught by the flip-flop at the falling edge of CLK signal, and observed at the rising edge by DAIC.

4.2 Chip Load, Read, Reset Logic and other Hardware

The explanations until now described the functioning of A/D converter systems. The card also needs some extra logic circuitry for loading and reading data from DAIC as well as for the compensation of the bugs that were explained in the previous chapter.

4.2.1 Hard Reset and Loading DAIC

DAIC must be initialized by a hard reset applied to Pin48. This is achieved by the following simple RC circuit. The capacitor is charged up to 5 volts through resistor, which causes a low-to-high transition on RR pin. This operation resets DAIC. The 74LS74 chips are also reset by this operation to assure that the initial states of EOC_POS and EOC_DAT signals are low.

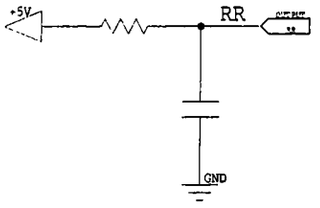


Figure 4.13: RR Signal Applied to DAIC and External Flip-Flops

After this reset operation, internal registers of DAIC should be written according to the procedure given in the previous chapter. This operation is carried out by the computer. The values of S0 and S1 are written into a latch using I/O port address

300h. D0 and D1 lines of computer data bus are entered as inputs of the latch for those signals respectively. Address decoder and other connections to the latch and computer data-bus are shown in Figure 4.14.

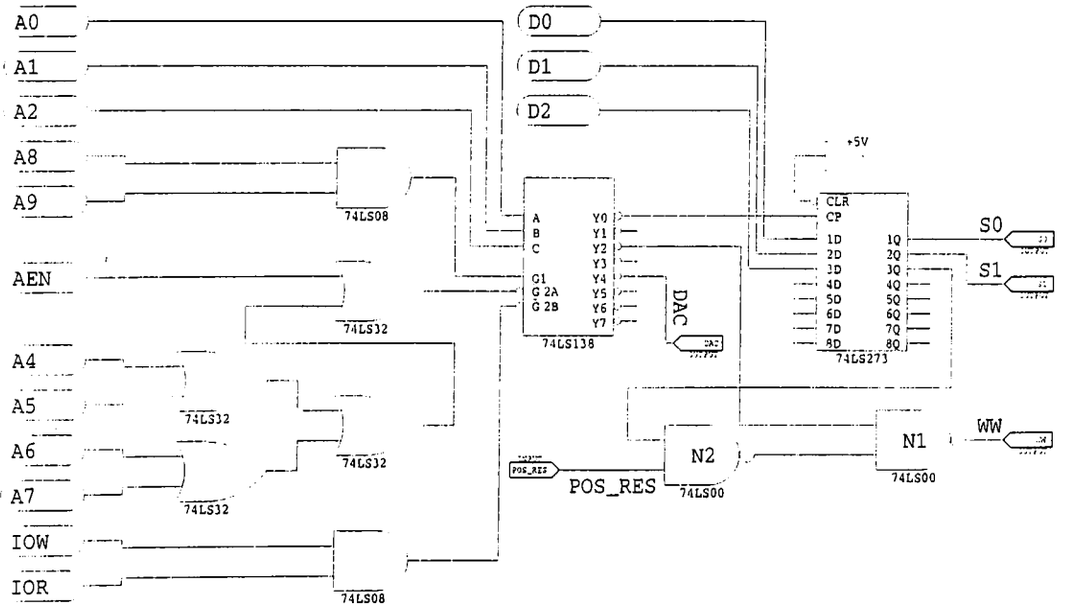


Figure 4.14: Address Decoder and Chip Load Logic

After S0 and S1 signals are set to appropriate values, WW should be made high to write into desired register. Port address 302h was used for this purpose. The value to be written into DAIC is put onto the computer data-bus by an "out" operation. During this out operation, address decoder output Y2 produces a pulse, and it is used as WW signal. When DAIC receives this WW, it captures the value on the data-bus of the computer (which is connected to I/O Bus of DAIC). Since outputs of address decoder 74LS138 are active low, Y2 output of this chip should be inverted before it is connected to WW pin of DAIC. This is done using NAND gate N1 as shown in Figure 4.14. The purpose of the other NAND N2 will become apparent later. The third output of the latch is set to logic "0" via D2 line, when S0 and S1 are loaded, and it disables the input signal POS_RES during the chip load operation. Then the output of NAND gate N2 stays at logic "1", so the NAND gate N1 acts as an inverter for output Y2 of 74LS138.

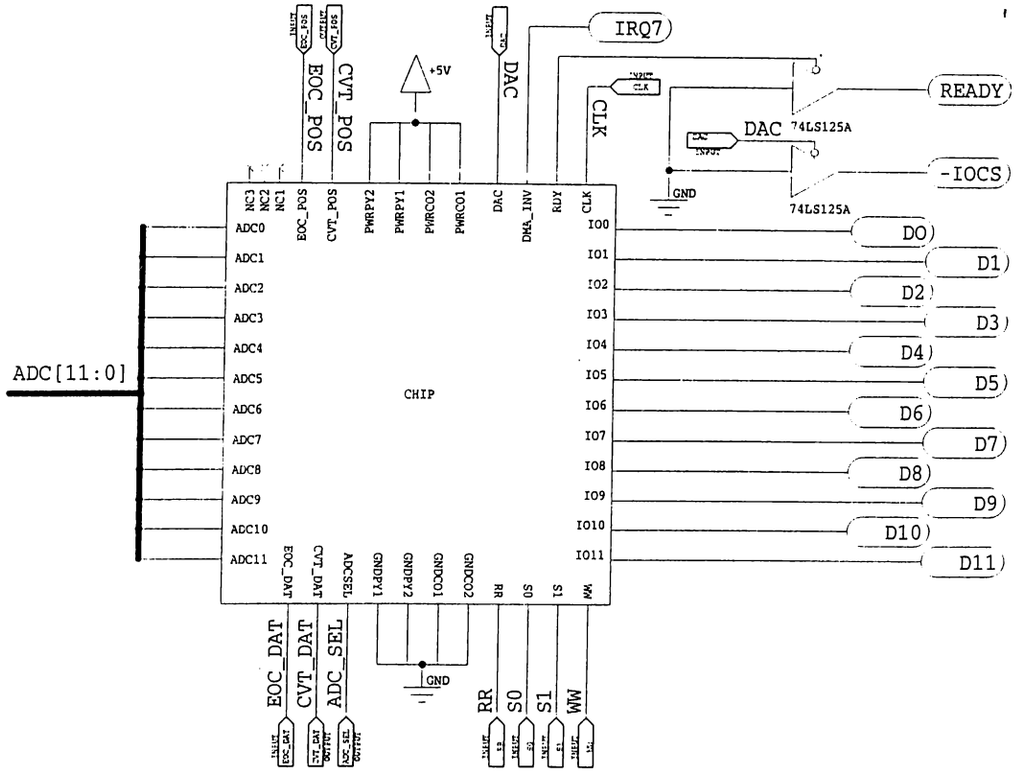


Figure 4.15: Connections to DAIC

4.2.2 Reading Data from DAIC

When the desired position is reached, DAIC sends a DMA_INV signal. This signal is connected to hardware interrupt request pin IRQ7 of the computer. As soon as this signal is received, computer starts to read data from DAIC using I/O port address 304h. The output Y4 of 74LS138 chip is connected to DAC pin of DAIC. The read cycle occurs with the hand-shaking procedure explained before. The RDY signal from DAIC should be connected to -I/O_CH_READY pin of the computer bus. This connection should be done through a tri-state buffer. Since the RDY output of DAIC does not have a Hi-Z state, this signal is used as enable of a tri-state buffer whose input is connected to ground. By doing so, whenever the RDY signal is activated (made low), -IO_CH_READY input of the computer is set to low voltage value, causing the computer to wait for preparation of data on the I/O Bus.

Another important pin on computer bus that should be activated (active low) is

-IOCS16 pin, which provides 16 bit parallel data acquisition from pins D0 through D15 on the data bus of the computer. If this pin is not activated, computer tries to read data from two consecutive port addresses, low byte being from the first port. For this signal again a tri-state buffer is needed, and as in ready signal, its input is connected to ground and enable is connected to DAC signal from the address decoder logic. As soon as a read cycle starts, this pin is set to low voltage, and reading occurs as 16 bits at a time.

4.2.3 Resetting DAIC

Two bugs in chip operations were observed. First one was that if the initial position read by DAIC is greater than Xstart value, a DMA_INV signal is sent immediately and data is not collected, the second one was that the program does not return to beginning after one read cycle is complete. The method offered for the compensation of these bugs was, resetting DAIC once during each cycle of the position signal, when position is decreasing and its value is less than Xstart. A simple comparator circuitry, using operational amplifier LM741, was designed for this purpose. The

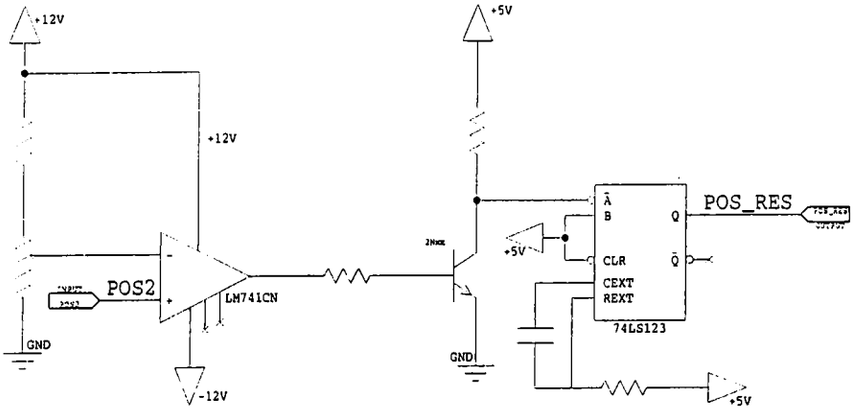


Figure 4.16: Chip Reset Circuitry

S/H amplifier SHC804 is a unity gain inverting amplifier with input voltage range of 0-10V, its output voltage range is -10-0V. The A/D converter quantizes -10V to "0" and 0V to "4095". This is shown in Figure 4.17. The reset region stated as "while position is decreasing and its value is less than Xstart" is depicted in terms of digital position data. As can be observed from the figure, this region corresponds

to a region between threshold voltage of Xstart value and 10V at the input of the' S/H amplifier. Since the comparator uses this original position signal POS2, the

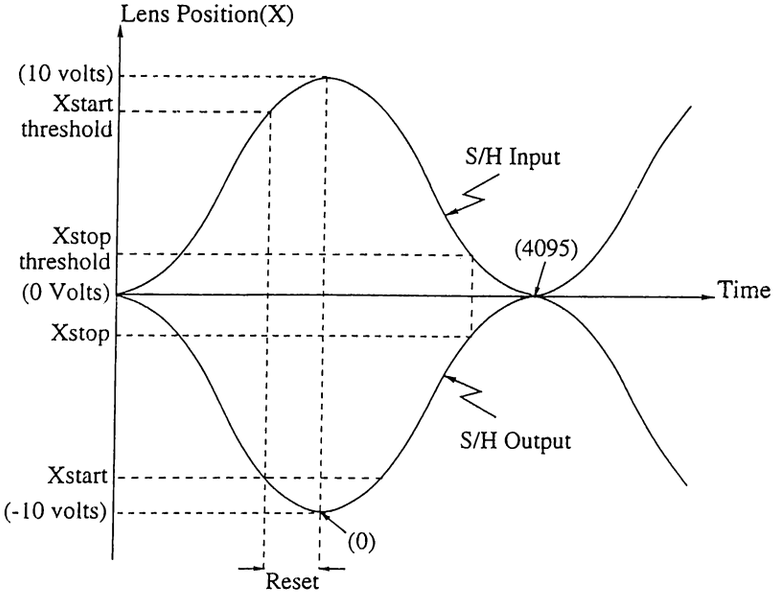


Figure 4.17: S/H Input and Output Voltages

reset operation should occur somewhere between Xstart threshold voltage and 10V. As shown in Figure 4.16, analog position signal is entered into (+) input of LM741 and a reset threshold value, which should be between Xstart threshold and 10V, and can be adjusted by a trimpot, is applied to (-) input. When POS2 is less than reset threshold value, output of the operational amplifier is at the negative saturation value, causing the BJT it is connected to stay in cut-off region. As soon as the POS2 value becomes greater than this threshold value output of the operational amplifier goes to positive saturation value, causing the BJT go into saturation. Collector voltage of BJT makes a high-to-low transition during this change, and hence causes 74LS123 to produce a positive pulse at its output Q. This signal is used as POS_RES, which was mentioned during the chip load operation. During that operation, this signal had been suppressed using the third output of the latch that keeps S0 and S1 values. After the internal registers of DAIC has been loaded, S0 and S1 are set to logic "0". A positive pulse at WW input of DAIC causes a reset of the internal program, and when WW is low again, chip starts to run. This pulse is sent by this chip reset circuitry. To able this signal, the third output of the latch is set to logic

"1" when S0 and S1 are written as "0". Referring to Figure 4.14 again, it is observed that the POS_RES signal appears on WW input of DAIC in this situation. Since Y2 output stays at high voltage when port 302h is not written, two consecutive NAND gates behave as a simple buffer for this signal. This pulse is sent to input WW at each cycle of the position signal and resets the chip program. This reset operation eliminates the bugs as mentioned earlier. DAIC collects acoustic data at each cycle by means of this compensation circuitry.

4.2.4 Clock Signal and Power

Almost all parts of the Data Acquisition Card are explained until now. Two other essential parts are the oscillator circuitry to obtain CLK signal, and circuitry for power supply needs of S/H amplifier and A/D converter chips.

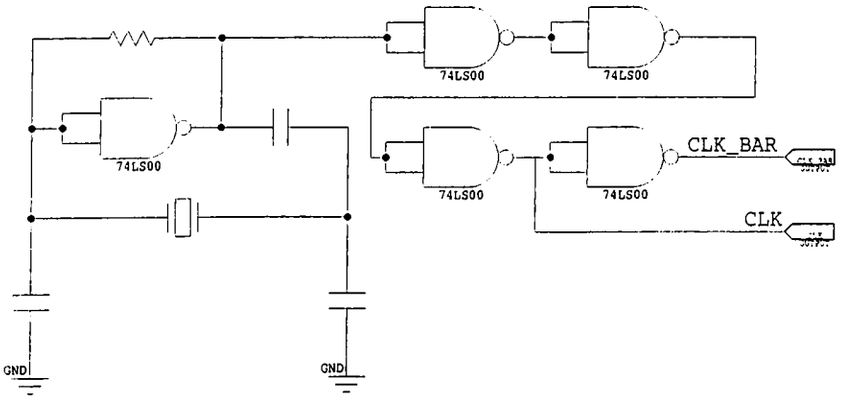


Figure 4.18: Oscillator Circuitry for CLK and CLK_BAR Signals

The crystal oscillator circuitry in Figure 4.18, with 74HC00 NAND gates are used to obtain a square wave clock. Using high speed CMOS logic chips, it is quite feasible to obtain good quality square wave oscillators up to 20MHz at fundamental frequency of the crystal. The frequency of oscillation was set to 10MHz for data acquisition system. The output waveform is inverted a few times to obtain a better square wave.

The power supply requirement of ±15V of A/D converter and S/H amplifier are obtained by using a DC-DC converter, HDA 0515, whose input is fed through the

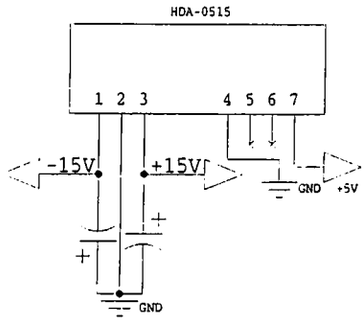


Figure 4.19: DC-DC Converter Circuitry

+5V supply of the computer. The voltage ripple at the output is less than 150mv. It can supply 70mA current for both outputs.

4.3 Software for Data Acquisition Card

In this section software associated with the data acquisition card will be explained in detail. Since this software directly deals with hardware, Turbo Assembler was preferred as programming language, in order to meet the speed requirement of the program.

This software consists of three programs, one for loading DAIC, one for reading from DAIC and one for storing and displaying the data read.

4.3.1 Software for Loading DAIC

The need for the first program became quite clear within the discussions about the card. As explained earlier, the values of S0, S1 and the control bit for chip reset operation are written into a register on the card using I/O port address 300h. This can be achieved by a simple "out" operation via "AL" register. For example, binary value 010b, for control bit-S1-S0 respectively as shown in Figure 4.20, is moved into this register and "out"ed to that port. The next step is writing data associated with that S0 and S1 combination, which is Xstop value for the example given above. The important point here is that these values are 12-bit, and hence 16-bit out operation

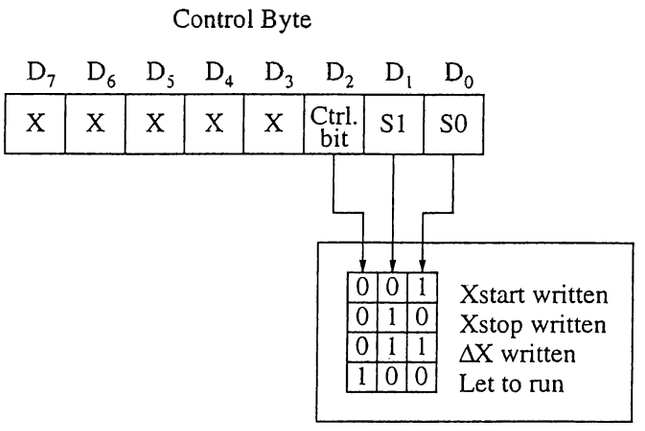


Figure 4.20: Bit Order to be Outed from AL During Chip Load Operation

must occur, since DAIC reads those 12 bits together. To achieve this ".286" compiler directive must be included in the assembly code and out operation must be done via "AX" register. It is safe to use an "even" port address since odd addresses may cause some problems during 16bit operations. This out operation is done using I/O port address 302h. The value of the control bit should be "0" during the load operations of the internal registers of DAIC so that a WW signal is not sent to DAIC by the chip reset circuitry. After loading operations are complete, S0 and S1 values are set to "0" and the control bit is set to "1", allowing the reset signal from chip reset circuitry to reach WW input and enable DAIC to run. In this way, this control bit can be used to turn the card on and off. The full assembly language code of CHLOAD.ASM is given in appendix.

4.3.2 Software for Reading Data from DAIC

Next two programs read data from DAIC. Read operation should be done whenever DMA_INV pin of DAIC is activated. This occurs immediately after Xstop position is reached, and reading task should be completed before the next data collection operation starts. This call signal is connected to IRQ7 pin of the computer bus. Upon receiving this signal, computer executes interrupt service routine (ISR) associated with this interrupt request.

Interrupts are basically separated into two categories: Software interrupts and

hardware interrupts. Software interrupts can be activated by a special assembly language instruction INT. Hardware interrupts are electrical signals on the related interrupt request pin and do not need such an instruction. There are a total of 256 hardware and software interrupts in a PC, numbered through 0h to 0FFh. Each interrupt must have a corresponding ISR if it is being used. ISR's are memory resident programs, i.e. they stay in memory independent of the currently executed program and runs whenever an interrupt request is made for that specific interrupt. The execution of an ISR requires the run of the currently executing program to be interrupted, as its name implies. The state of the processor is saved into stack at the time of interrupt request and reloaded after ISR run is complete, to be able to continue the execution of the original program from the point where it is left.

To be able to reach to ISR's one need to know the exact location of these programs in the computer's memory. The interrupt vector table is used for this purpose. This table specifies the interrupt and the location of its interrupt routine in computer's memory. It occupies the first 1024 bytes of the memory of a PC. In this table, there is a 4 bytes long entry for each interrupt, which is a vector corresponding to the starting address of the interrupt routine in the memory, high two bytes being the segment address and other two bytes being the offset in that segment. Figure

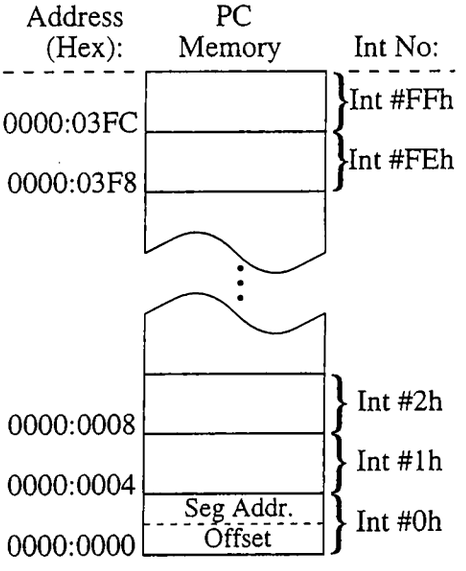


Figure 4.21: Interrupt Vector Table

4.21 demonstrates the organization of first 1024 bytes of PC memory. Since each

vector is 4 bytes long, and interrupt vector for Int #0 starts at memory location 0, interrupt vector start at address, four times the interrupt vector number. When an interrupt is called, the processor automatically retrieves the starting address of the corresponding ISR from this table and begins executing that routine.

Since the data acquisition card uses IRQ7 hardware interrupt, whose interrupt number is 0Fh, memory locations 3Ch through 3Fh in the first segment should be written by the complete address of the ISR that will read data from DAIC. This is done by using DOS interrupt 21h function request 25h, set interrupt vector, and the code is made resident by using function request 31h, terminate and stay resident.

Another essential operation for an external interrupt is the use of the programmable interrupt controller (PIC) chip 8259A inside the computer. External hardware interrupts in a PC are handled by this chip. It adds eight vectored priority encoded interrupts to the microprocessor. This controller can be expanded without additional hardware to accept up to 64 interrupt request inputs. This expansion requires a master 8259A and eight 8259A slaves. There are two 8259A's in a PC (AT or later), one being master and other being slave. These two chips handle 15 hardware interrupts in a PC.

The way in which the 8259A operates is determined how the device is programmed. Two types of command words are provided for this purpose: Initialization command words (ICW's) and operation command words (OCW's). ICW's are used to load the internal control registers of 8259A. There are four such command words ICW1 through ICW4. On the other hand three OCW's permit the computer to initiate variations in the basic operating modes defined by the ICW commands. These are OCW1, OCW2 and OCW3. Since BIOS does the initialization of 8259A, it is not necessary to deal with them in this discussion. The two OCW's that are to be loaded, OCW1 and OCW2 will be explained here.

OCW1 determines which of the 8 interrupt request IR0-IR7 will be active. Since IRQ7 pin of the computer is used, and it is connected to IR7 input of the master 8259A, this interrupt request must be activated. In Figure 4.22.(a), OCW1 is shown. It has 8 mask bits M0 through M7 for IR0 through IR7 respectively. If a mask bit is reset, the corresponding interrupt is made active. So M7 should be reset to make IRQ7 active. This is done by first reading from this register via port address 21h,

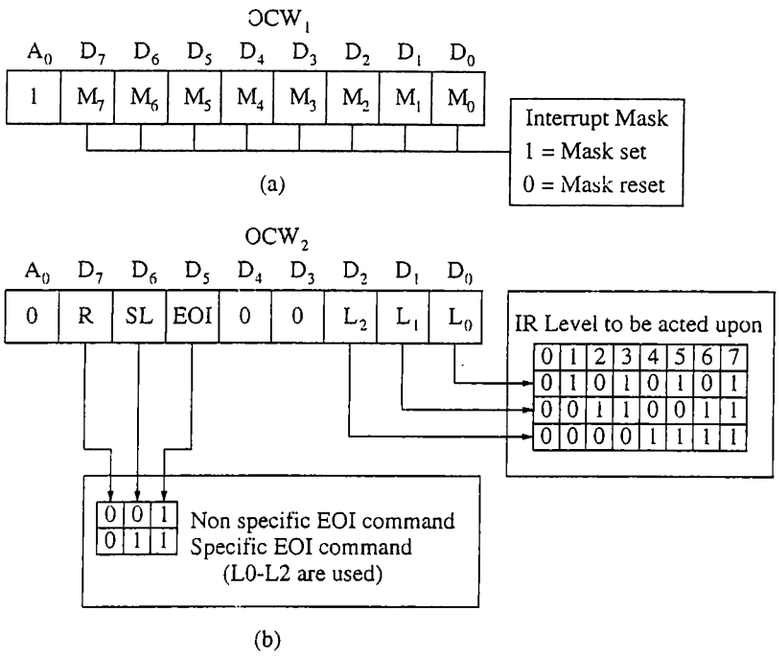


Figure 4.22: OCW's of PIC 8259A

resetting M7 and writing this value back into same register. This operation is carried out only once, when the read program is loaded into memory, and this makes ISR to be activated at every low-to-high transition on IRQ7 pin of the computer.

At each return from a hardware interrupt, an end of interrupt (EOI) command should be sent to 8259A. OCW2 determines the type of EOI. Two of EOI commands, which are obtained from the combinations of bits D5-D7 of OCW2, are shown in Figure 4.22(b). These are "specific end of interrupt" and "non-specific end of interrupt" commands. A specific EOI command is needed in a PC, unless the corresponding initialization command word that is loaded by BIOS is changed by reloading it. Specific EOI command requires the determination of IR level to be acted upon, i.e. the interrupt that is being quited. Then bits D5-D7 of OCW2 should be 1-1-0, while D0-D2 are 1-1-1. These values are outed to port 20h, at the end of each each ISR execution.

It is also possible to determine the priority of IR's in a 8259A using other combinations of D5-D7 bits of OCW2, but it is not done in the data acquisition software and IR7 is left as the lowest priority interrupt of all hardware interrupts in a PC.

Having introduced the necessary information about the handling of hardware interrupt requests in a PC, other two programs can be explained now. These two programs interact with each other, and hence explanations are done together.

One of these programs is a memory resident ISR, as stated before. Other one is the user interface of the ISR. After the ISR is loaded, it is required to pass the number of lines to be scanned and the number of data points at each line, to that routine. The starting address of memory location where the acoustic data read from DAIC will be stored, should also be passed to ISR.

Since after the installation of ISR into memory, it is not possible to reach directly to the variables etc, inside that program, a way of communication should be found between ISR and user interface programs. For this purpose, a pre-determined memory location is used as mail-box between these two programs. This memory location is the last 10 bytes of the interrupt vector table, left for the use of user defined software interrupts. Of course this software interrupts must be currently unused. First two bytes of this space, 3F6 and 3F7 are used as flags. First flag, DoIt, enables and disables the ISR. If it is set, ISR immediately leaves the interrupt without doing any operation. Second flag, Done, acknowledges the user interface program that the data reading operation from DAIC is completed. Following two words starting at 3F8 and 3FA, keeps the number of lines to be scanned and number of data points at each line. Finally the last two words keep the absolute memory address of the starting point of acoustic data array to be written into by ISR.

When the user interface is run, it gets number of lines to be scanned and number of data points for each line from the user. It allocates enough memory for instructed number of data points and puts the necessary information into mail-box. Then it activates ISR via flag DoIt and starts to wait for Done flag to be set by ISR.

After DoIt flag is set by the user interface, ISR reads data from I/O port 304h at each call of IRQ7 by DAIC. For one call, only one line of data is read. After number of lines calls of IRQ7, ISR sets flag Done and deactivates itself by setting DoIt flag. For another read operation, it must be reactivated by the user interface program.

Upon observing Done flag is set, user interface program writes the data stored in the array into a file and quits.

The assembly language codes for interrupt service routine IRQ7SR.ASM and user interface READ.ASM are given in appendix.

Chapter 5

MEASUREMENTS AND RESULTS

This chapter presents the experimental measurements of signal timings and acoustical images obtained by using data acquisition card for different input signals along with the expected computer simulated images.

5.1 Timing Measurements

Timing measurements are done using a HP 54600B digital oscilloscope. A personal computer having IEEE-488 GP-IB card connected to this oscilloscope was used to get the oscilloscope traces. These signals are taken only from position A/D converter system since the signal rates in data A/D converter system are low and difficult to observe. However, the values of the electronic components are the same for both systems, and hence signal timings are identical.

Following figures are obtained with time steps of 0.5ns. Waveforms are triggered with respect to CVT_POS signal, i.e. the rising edge of this signal is set to t=0, except the clock signal in Figure 5.1.

By looking at these oscilloscope plots the following information about the card

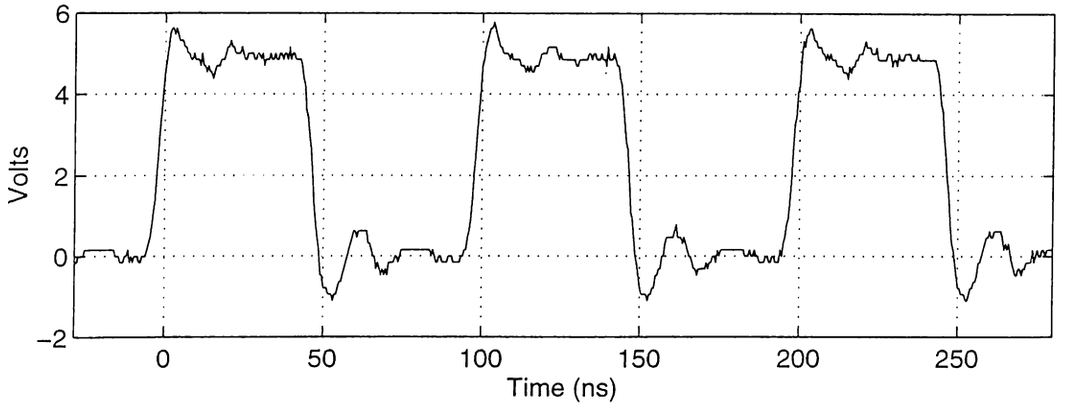


Figure 5.1: Clock Signal at 10MHz.

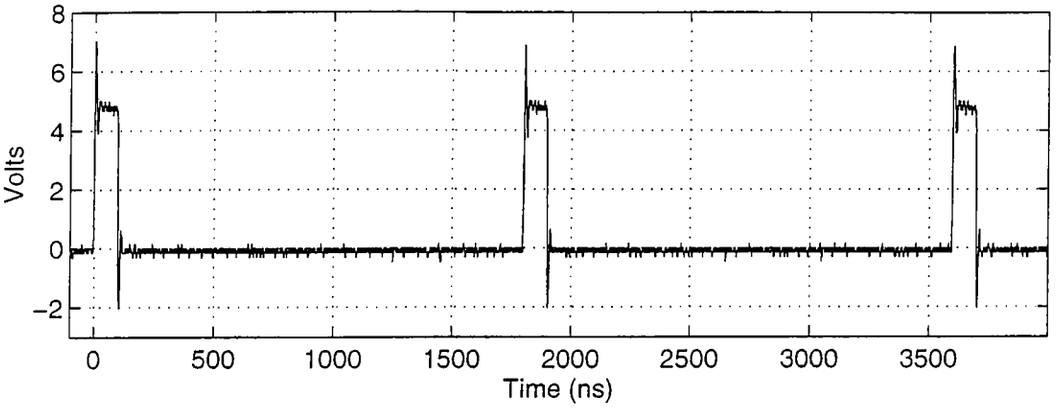


Figure 5.2: CVT_POS Pulses

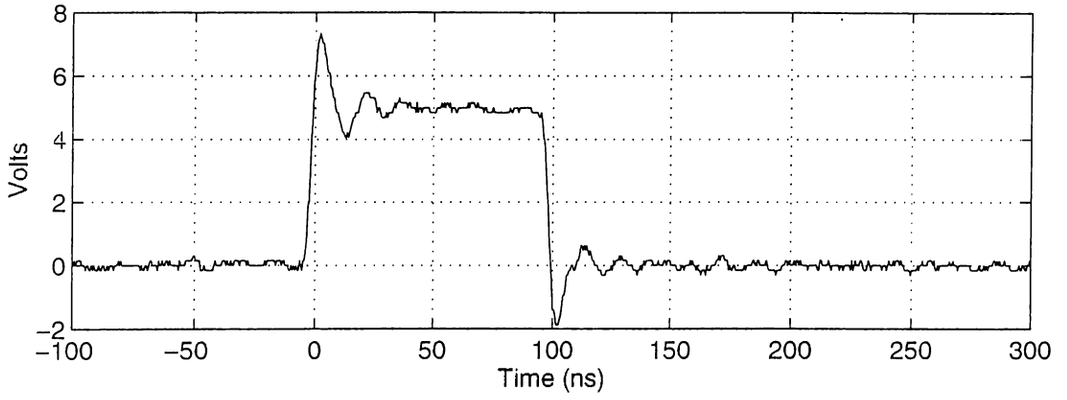


Figure 5.3: A Single CVT_POS Pulse

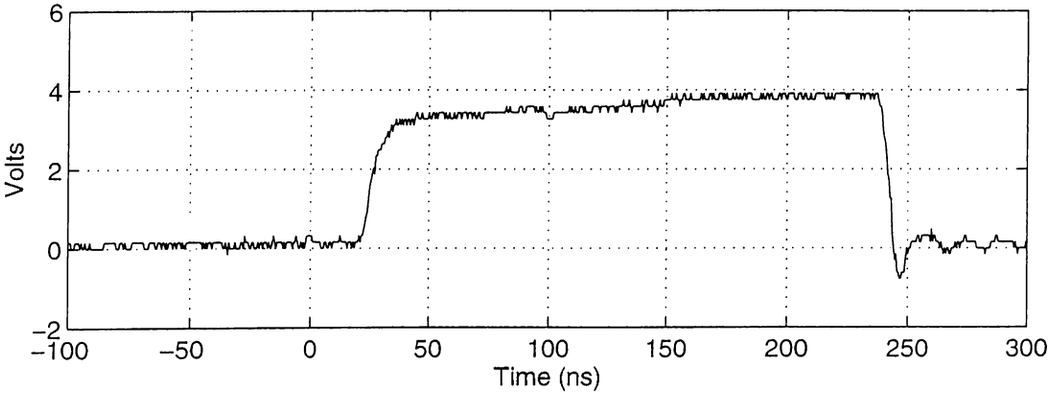


Figure 5.4: CVT_POS2 Pulse

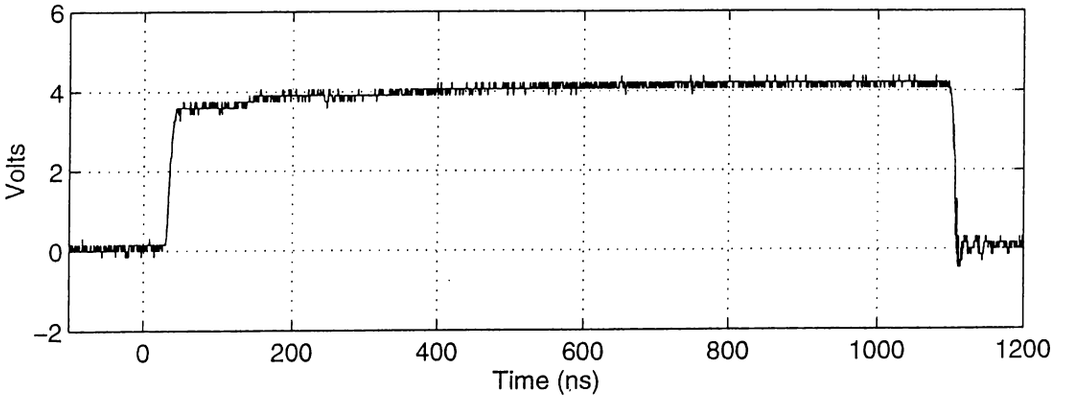


Figure 5.5: Hold Input of SHC804

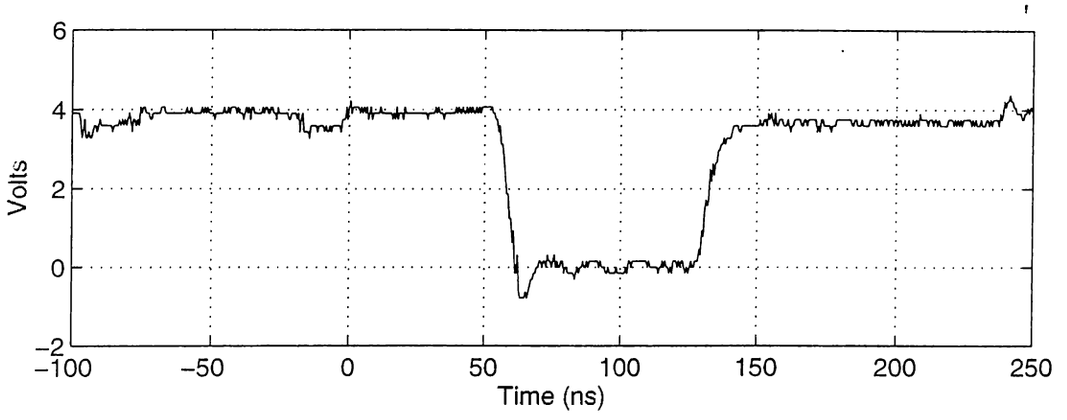


Figure 5.6: Convert Signal to ADC601

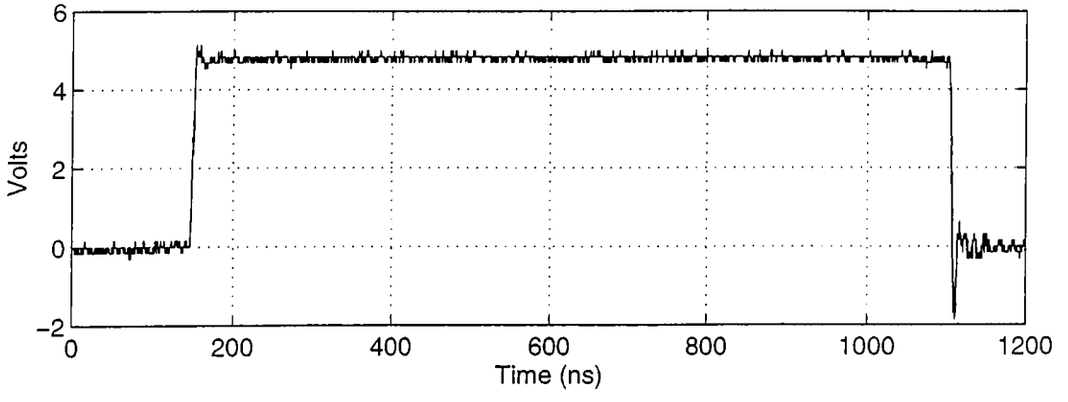


Figure 5.7: Status Output From ADC601

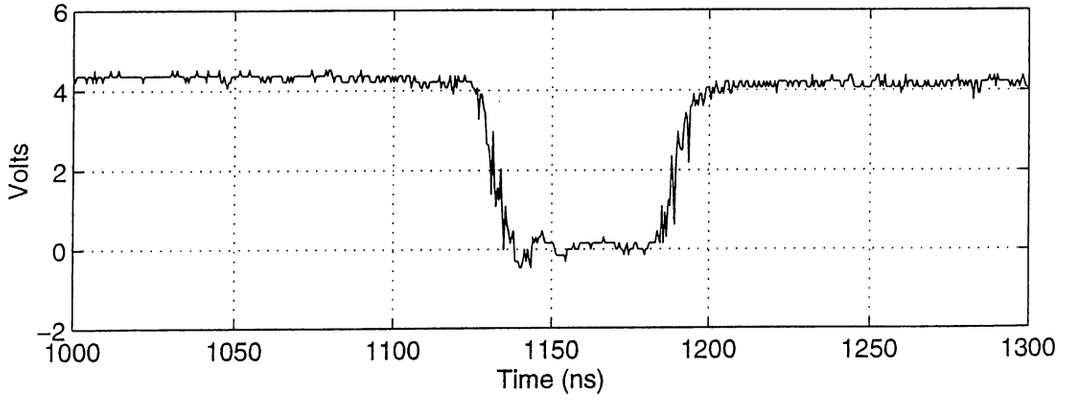


Figure 5.8: CP_POS Signal

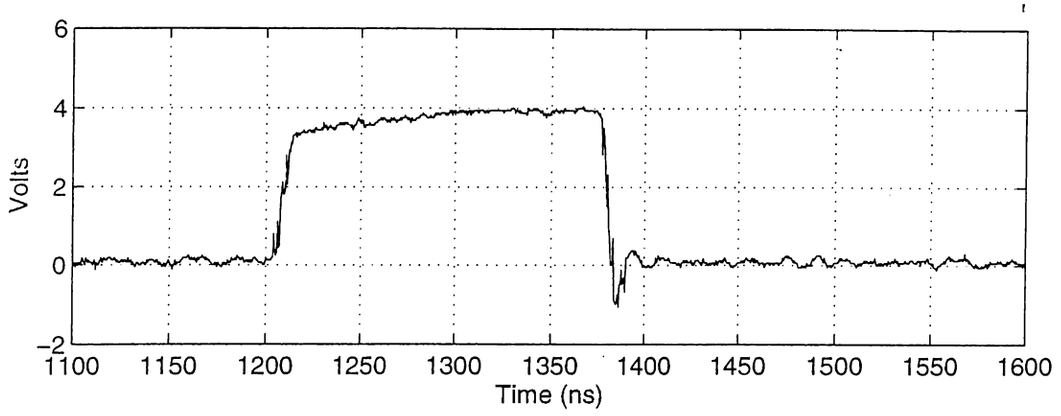


Figure 5.9: EOC_POS Signal Before Synchronization

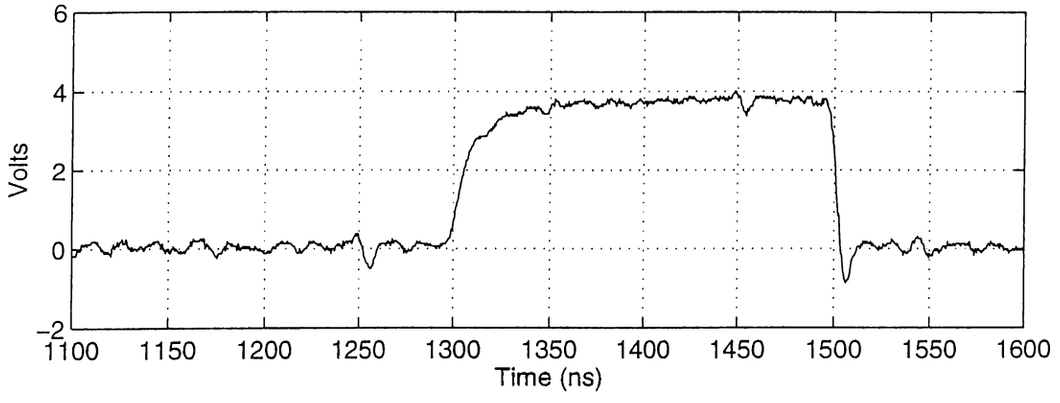


Figure 5.10: EOC_POS Signal After Synchronization

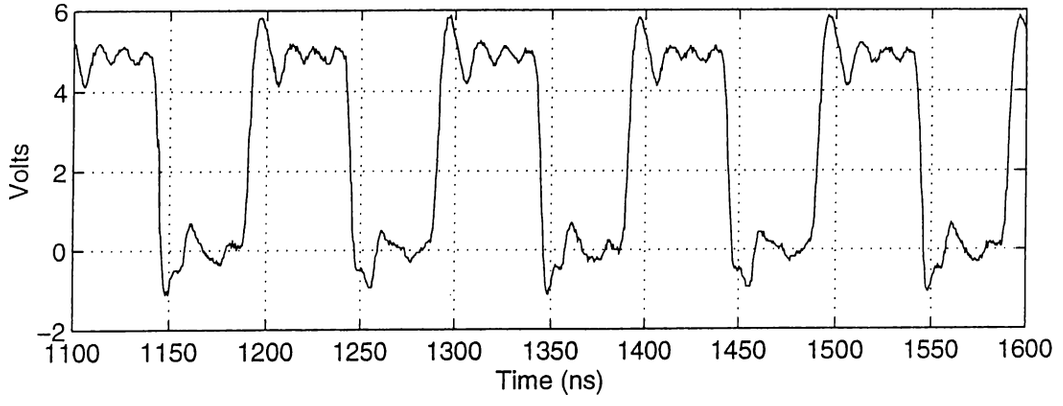


Figure 5.11: CLK_BAR Signal in Synchronization Interval

is observed:

Operating frequency of the clock is 10 MHz. as expected. There are some fluctuations on this signal, but the small distortions on this signal were not observed on an analog oscilloscope screen. They should have been introduced during the acquisition of these traces by the computer, because of the low precision of the measurements.

DAIC samples position signal with 1.8μs intervals which corresponds to a frequency of around 550 KHz. This means that from position 0 to 4095, 5500 position samples can be collected. 2V overshoots were observed during the rise and fall of this signal

From the STATUS output trace it is observed that conversion of position is completed nearly 1100ns after CVT_POS signal is sent. 950 ns of this period is the conversion time of A/D converter.

EOC signal should be captured by DAIC at 1350ns, at the rising edge of the CLK signal. The time interval between this point and the next CVT_POS signal, 450 ns, is the processing time of DAIC.

5.2 Images

Data acquisition card was installed on a personal computer, and the programs were run as explained before. DAIC is loaded as follows:

Xstart = 1500, Xstop=3500, ΔX=10

The test carried out by entering a sinusoidal signal to the acoustic data input of the card. Four different frequency values were applied: 50Hz, 100Hz, 200Hz, and 1KHz. The images of these sinusoidal signals were obtained displaying the data values vs position, using a 256 level gray scale colormap. For a scanning mechanism oscillating with a 5mm amplitude, one side of those images correspond to 2.44mm. The images from the card and MATLAB simulated images for them are given in the following pages.

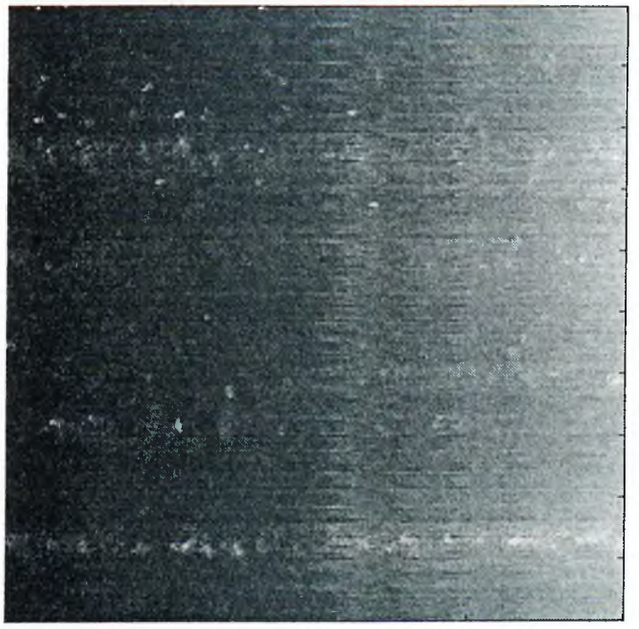


Figure 5.12: Image from the card, f=50Hz.

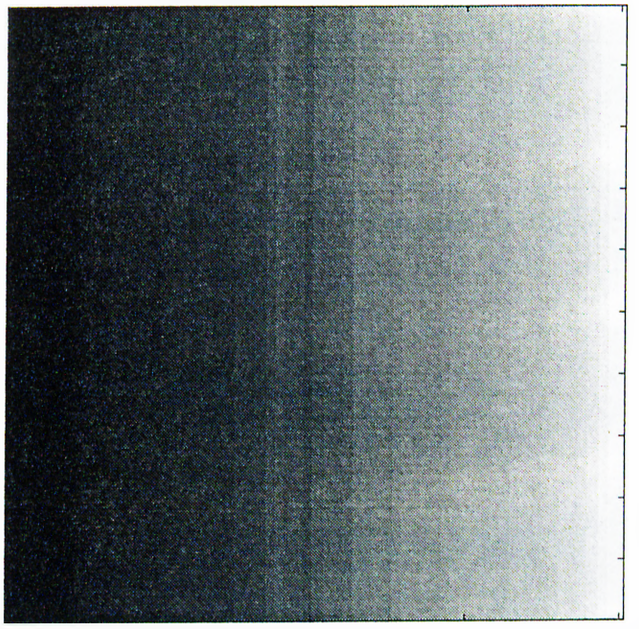


Figure 5.13: Simulated Image, f=50Hz.

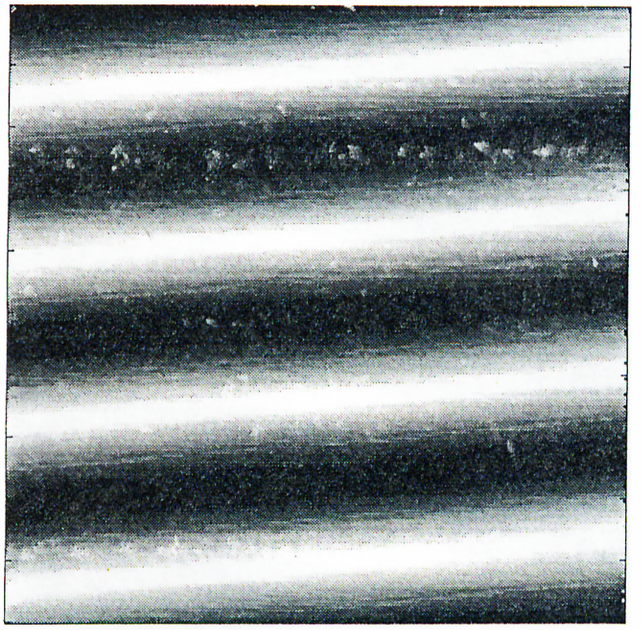


Figure 5.14: Image from the card, f=100Hz.

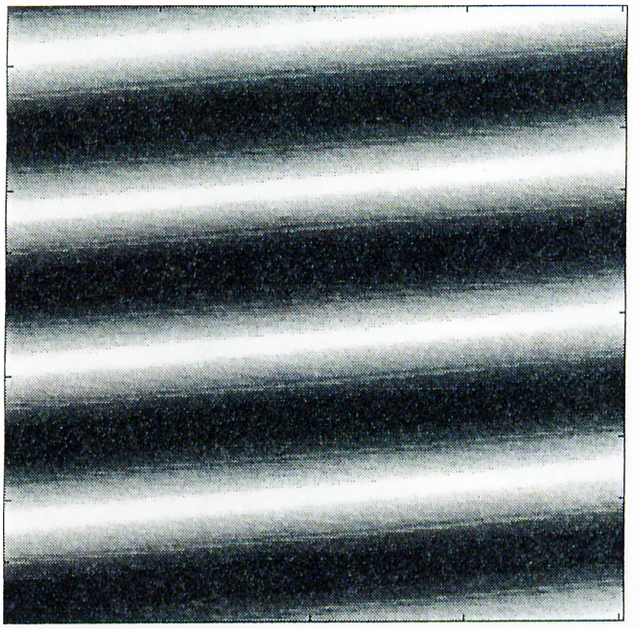


Figure 5.15: Simulated Image, f=100Hz.

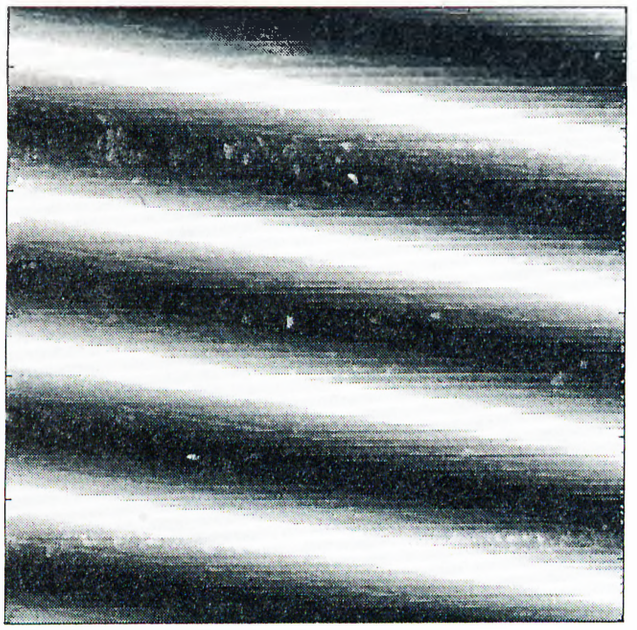


Figure 5.16: Image from the card, f=200Hz.

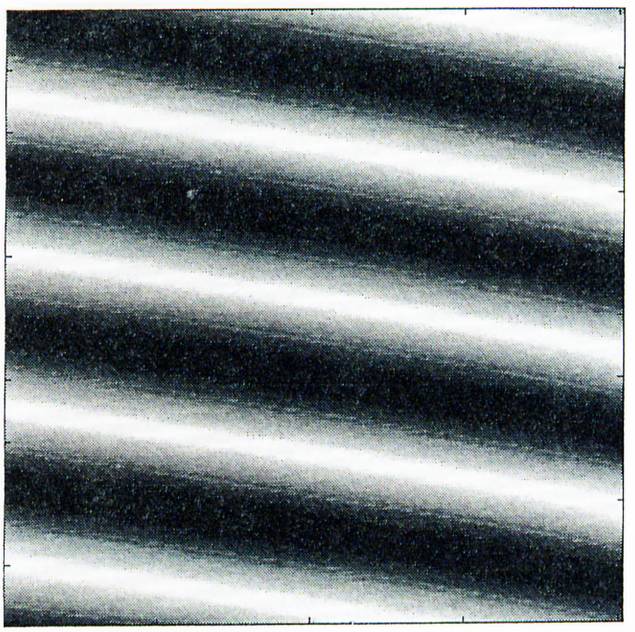


Figure 5.17: Simulated Image, f=200Hz.

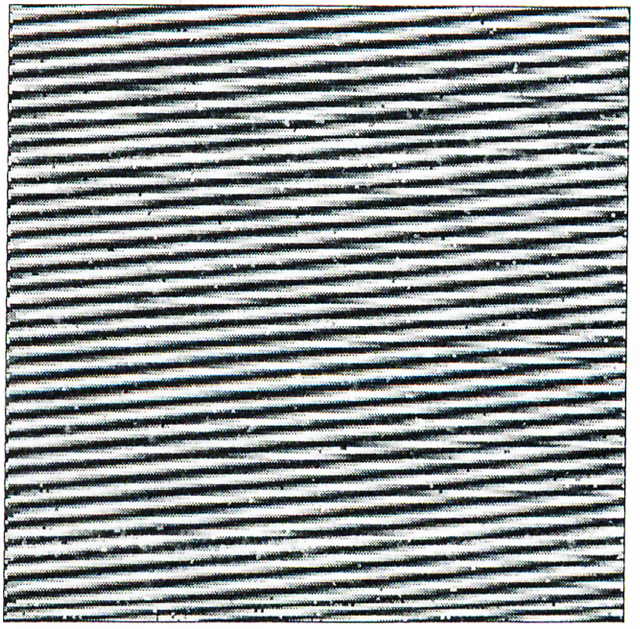


Figure 5.18: Image from the card, f=1KHz.

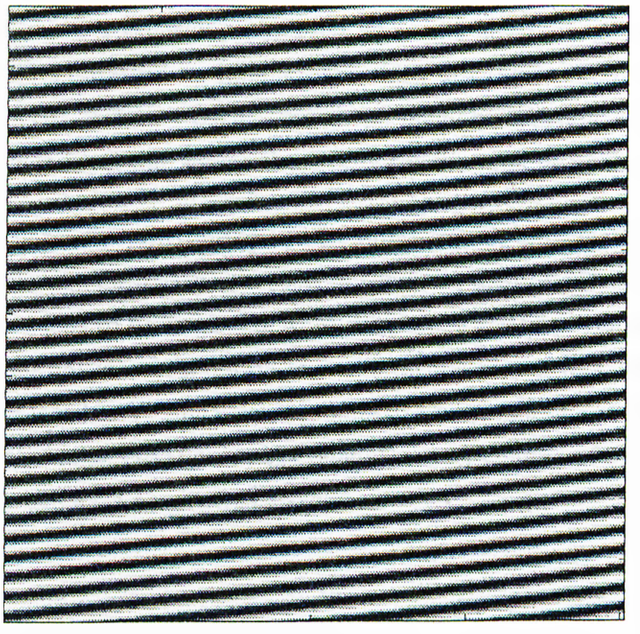


Figure 5.19: Simulated Image, f=1KHz.

From these figures it was observed that, the images obtained from the card are very similar to simulated images. However some distortion was observed in those images. This is especially observed when data signal is the same as the position signal. In this case the rising portion of the sinusoidal signal is observed, in synchronization with the position signal. The expected image is a smooth variation in gray tone, from left to right as in the computer simulation. However in the actual image there are some fluctuations in tone from top to bottom. This is because DAIC unexpectedly writes some data values twice into its internal RAM. This duplications causes shifts within a row. From one row to next, same data values can not be displayed at the same index, and hence image is distorted. The occurrences of these duplications does not have a regular pattern, which makes the detection and elimination of them impossible for an arbitrary input signal.

For input signals other than 50Hz, the appearance of the images differ from the first one. The undulations are not parallel from top to bottom. This is because the fact that the applied data signal is not a perfect harmonic of the position signal. A similar effect is observed on the simulated image when the data signal frequency is shifted slightly (for example 102Hz instead of 100Hz).

Another distortion is especially observed when the input image signal is 1KHz. There are some dots observed on the picture, which are due to the presence of noise in the position ADC system. DAIC leaves the memory location unchanged for a data point that is missed, and hence these dots corresponds to unchanged data.

The noise is partially contributed by the input signal, and partially introduced by the data acquisition hardware. To observe the noise introduced by the hardware, a clean DC source had to be used. For that purpose, a battery was employed. The maximum noise amplitude observed on the oscilloscope screen was 20mV. Since the quantization level is 2.44mV, 20mV corresponds to a variation of 8 at the output. On the other hand, the noise variation at the output was 14 instead of 8. The difference is contributed by the data acquisition hardware.

Chapter 6

CONCLUSION

Acoustic microscopy has been proven to be a very useful tool with its wide range of applications. However the complexity, and hence the cost of an acoustical imaging system limits the progress of its extended use. Development of minimal specific hardware dependent, new technology and low cost acoustic microscopes is necessary.

In this work, a data acquisition system of an acoustic microscope working at high scan rates is implemented. Implementation is done such that, the system can be mounted and run on a personal computer. A data acquisition chip that was designed for this system is used in the design.

The system is tested by using sinusoidal input signals. The images that were obtained by using the system were compared with the computer simulated images. Some deviations from the expected performance were observed, which are mostly caused by the data acquisition chip used in the system.

Some problems caused by the data acquisition chip were eliminated within the design by extra compensation circuits around the chip. However one particular problem was the repetitive storage of some data values into the internal RAM of DAIC. It might be caused by the glitches inside the chip on the "write" signal. Although the simulation tool used, Verilog, considers the signal delays, delays on some critical signal lines such as the clock line, might be responsible for this situation. To eliminate this defect, the length of the fast signal lines must be carefully controlled,

in order to avoid uneven delays. This requires placement of internal components of the chip manually, rather than using auto-placement tools, as was done during the design of present DAIC.

Another problem was the noise in the system. The recommended layout scheme in the data book of A/D converter and S/H amplifier is to use large ground and power planes. This requires a 4 layered production technology. Existing PCB, on the other hand, is produced as a 2 layer card, and only a ground plane of limited area around ADC's and S/H amplifiers could be implemented. Analog and digital ground pins of A/D converter and S/H amplifier are connected directly to this plane. However it is observed that this is not sufficient, especially for a PC card, where digital noise levels are high. Therefore it would be a better practice to separate digital and analog ground lines and connect them together very close to slot pins of the card. Obviously, best approach is to produce a PCB with 4 layer technology and use whole ground and power planes.

APPENDIX A

LAYOUT OF DAIC

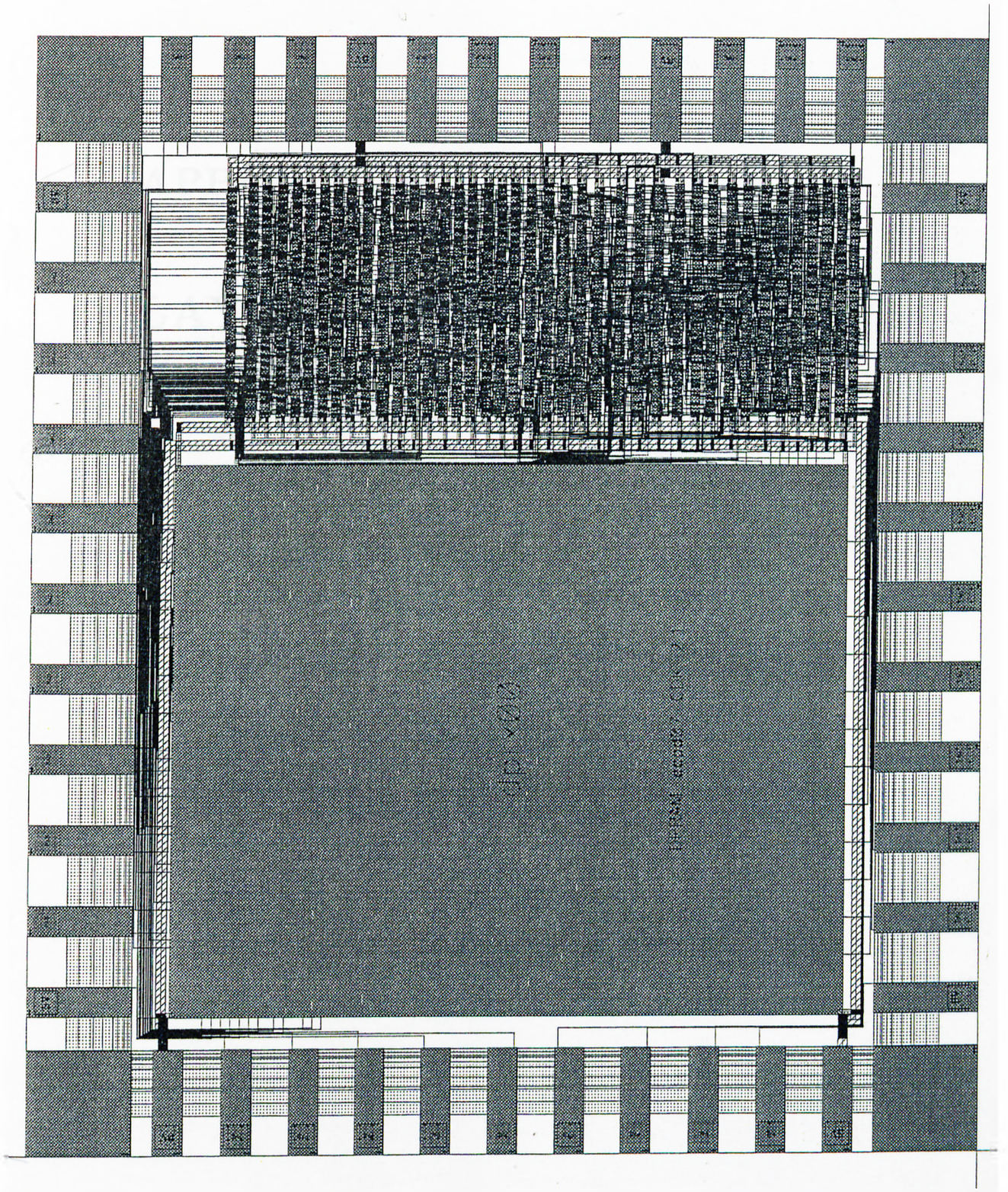


Figure A.1: Data Acquisition Integrated Circuit Layout

APPENDIX B

LAYOUTS OF DATA ACQUISITION SYSTEM

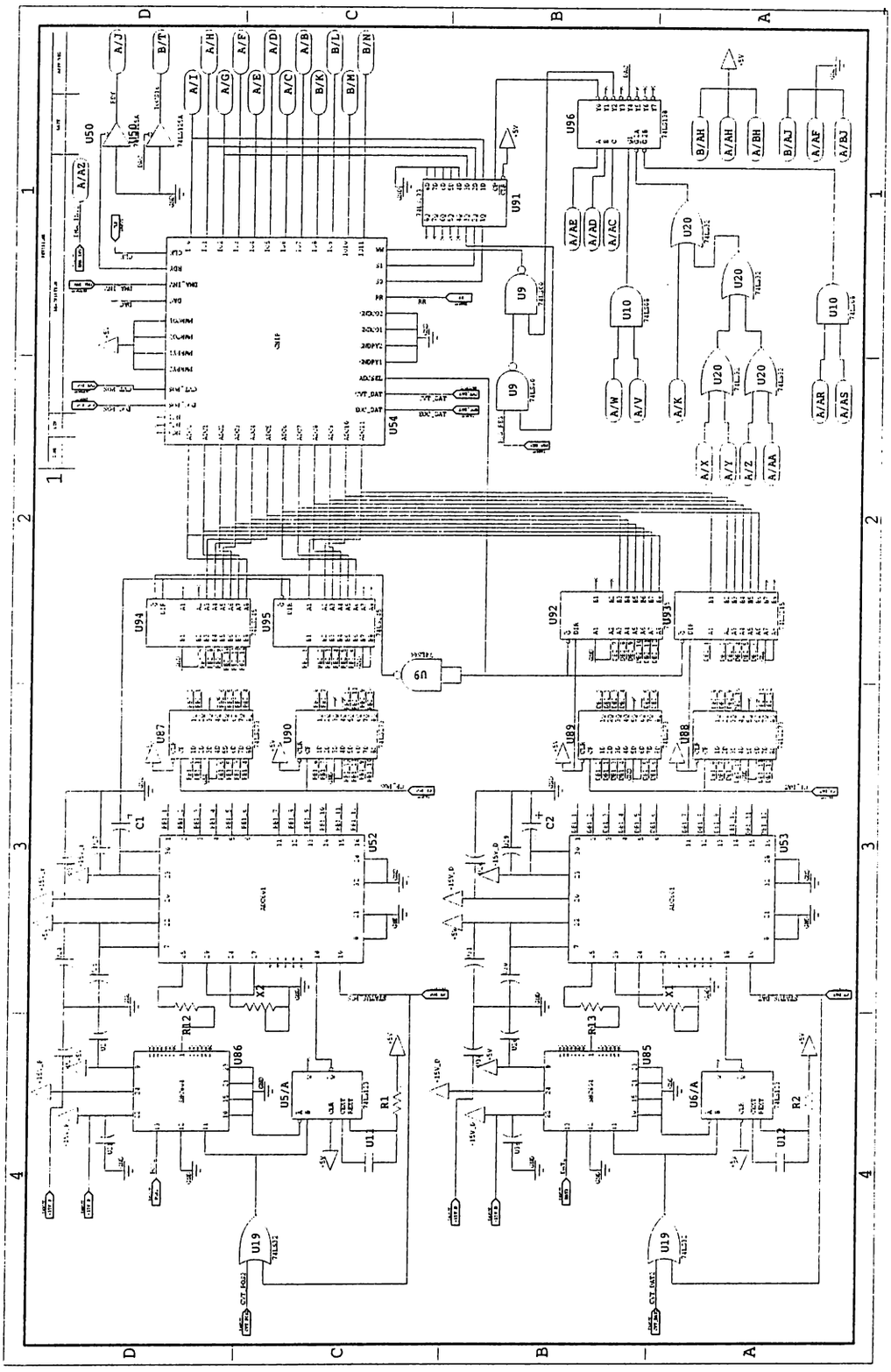


Figure B.1: Data Acquisition System PCAD Scahematic: Part1

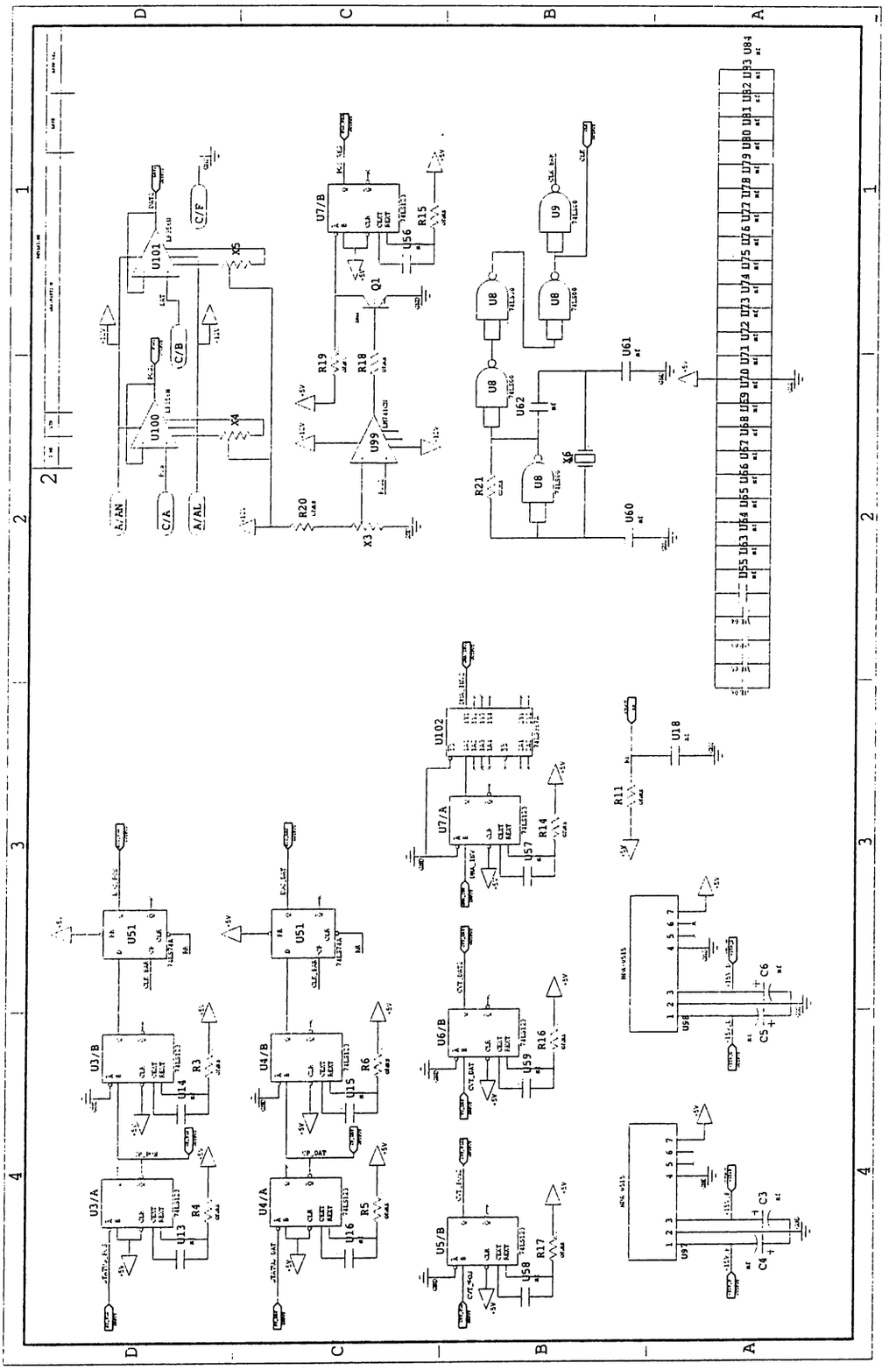


Figure B.2: Data Acquisition System PCAD Scahematic: Part2

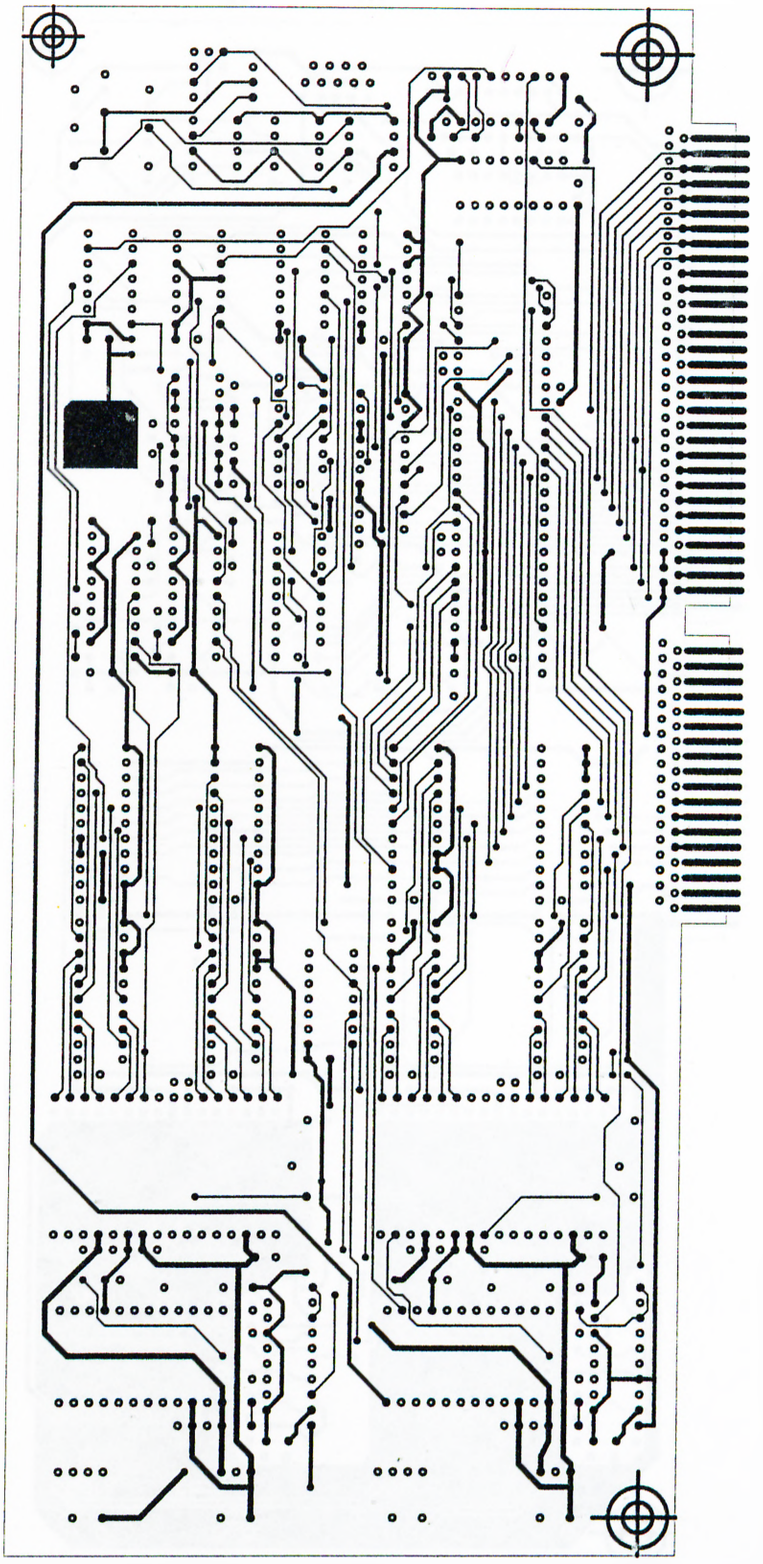


Figure B.3: Data Acquisition System PCB Layout: Component Side

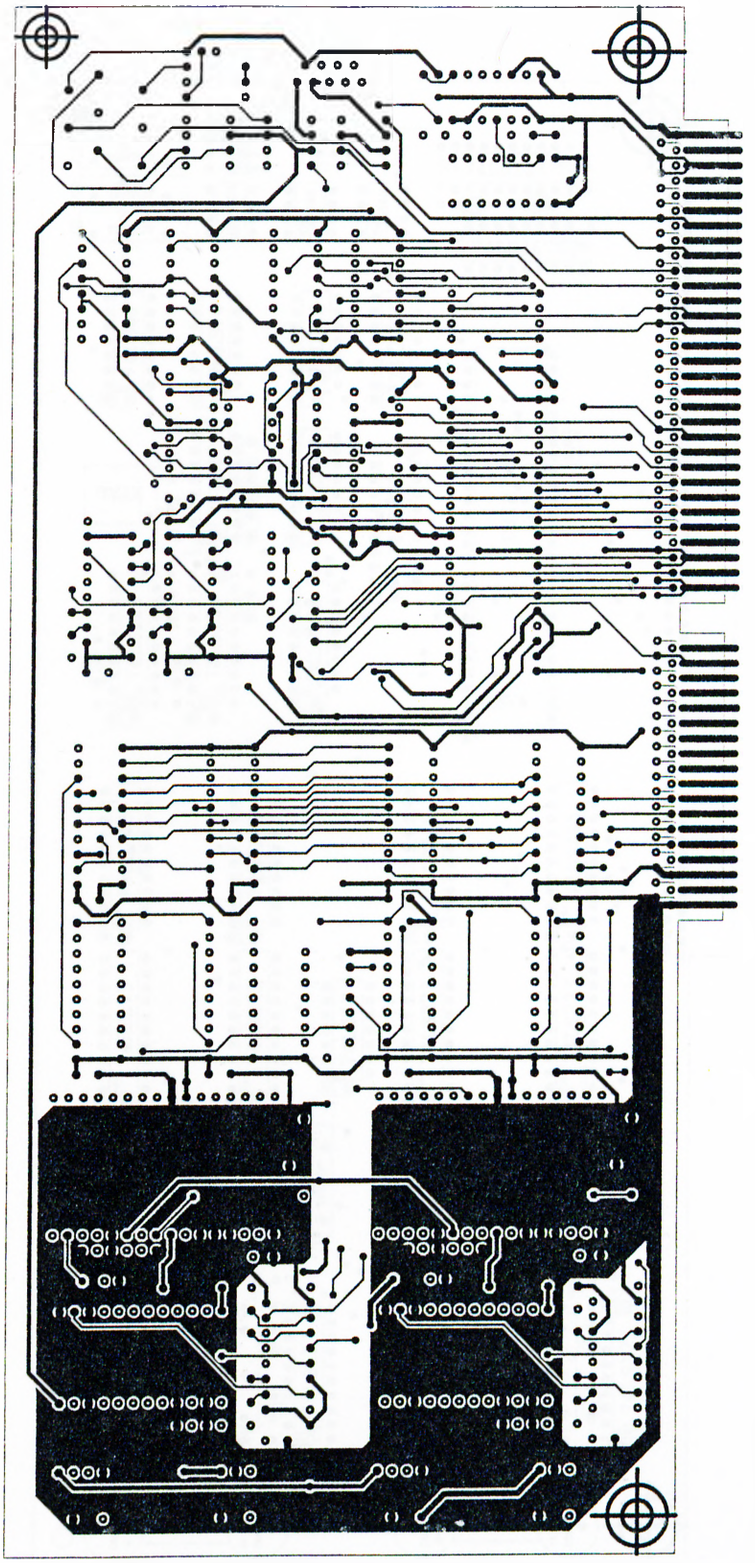


Figure B.4: Data Acquisition System PCB Layout: Solder Side

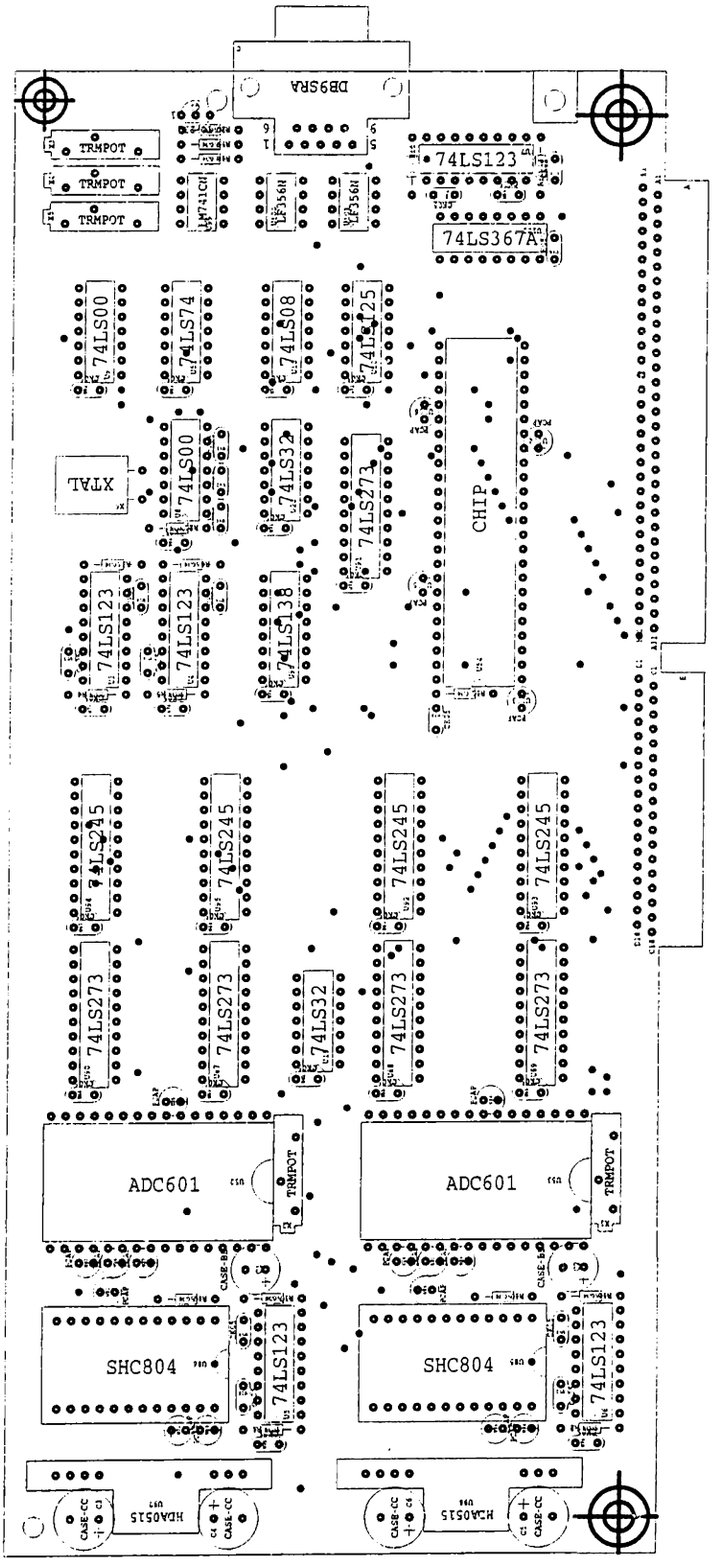


Figure B.5: Data Acquisition System PCB Layout: Silk Screen

COMPONENT VALUES

Resistors:

R1:4.7KΩ R2:4.7KΩ R3:15KΩ R4:3.3KΩ R5:3.3KΩ R6:15KΩ R11:12KΩ

R12:10Ω R13:10Ω R14:15KΩ R15:120KΩ R16:18KΩ R17:18KΩ R18:1KΩ

R19:10KΩ R20:1.8KΩ R21:10M

Trimpots:

X1:200 X2:200 X3:10KΩ X4:20KΩ X5:20KΩ

Capacitors:

U11:2.2pF U12:2.2pF U13:1.5pF U14:10pF U15:10pF U16:1.5pF U56:10pF U57:10pF U58:10pF U59:10pF U60:66pF U61:15pF U62:66pF

U21, U22, U23, U24, U25, U26, U27, U28, U29, U30, U31, U34, U36, U37: 1uF (tantal)

C1, C2:150uF (tantal)

C3, C4, C5, C6:1000uF

All other decoupling capacitors:100pF

APPENDIX C

PROGRAM CODES

CHLOAD.ASM

```
DOSSEG
.MODEL SMALL
xstart equ 1500
xstop equ 3500
deltax equ 10
.STACK 200h
.286
.CODE
start: mov dx,300h
mov al,001b
out dx,al
mov dx,302h
mov ax,xstart
out dx,ax
mov dx,300h
mov al,011b
out dx,al
mov dx,302h
mov ax,deltax
out dx,ax
mov dx,300h
mov al,010b
out dx,al
mov dx,302h
mov ax,xstop
out dx,ax
mov dx,300h
mov al,000b
out dx,al
mov ah,4ch
int 21h
END Start
```

READ.ASM

```
DOSSEG
DoIt equ 3F6h
Done equ 3F7h
num_rows_ptr equ 3F8h
num_dat_pts_ptr equ 3FAh
dat_arr_ptr equ 3FCh
max_num_rows equ 300                    ;maximum number of rows to be scanned
.MODEL SMALL
.STACK 200h
.CODE
program_start:
jmp start2
num_rows dw ?                                   ;number of rows to be scanned
num_dat_pts dw ?                                ;number of data points in each row
arr_of_rows dw max_num_rows DUP (?)
                ;array of rows is the acoustic data array whose entries are pointers
                ;to row arrays. num_rows entry of this array contain valid pointers.
                ;Row arrays are array of words and their size are num_dat_pts. One
                                        ;row array keeps one row of acoustic data.
outhdl dw ?
outfile db 'c:test.out',0
row_msg db 13,10,'Enter # of Rows: ','$'
num_dat_msg db 13,10,'Enter # of Data Points: ','$'
err_msg1 db 13,10,'Memory Allocation Error','$'
;Procedure reads a number from keyboard
;Regs Destroyed :AX,CX
;Returns: read number in BX
ReadNum PROC
xor bx,bx                                               ;mov 0 to bx
mov cx,10
read_loop:
mov ah,01h
int 21h                                                 ;read as character
```

```
cmp al,'.'                                        ;reading ends with character '.'
jz end_read
;rest of the procedure converts the input character set
;into a number in BX
push ax
mov ax,bx
mul cx
mov bx,ax                                                  ;multiply BX by 10
pop ax
cbw
sub ax,30h                                      ;convert last character into a digit
add bx,ax                                       ;add it as the least significant digit
jmp read_loop
end_read:
ret
ReadNum ENDP
;Destroys: AH
Display_String PROC
mov ah,09h
int 21h
ret
Display_String ENDP
start2:
mov ax,cs
mov ds,ax
sub ax,10h              ;AX contains the Program Segment Prefix (PSP) address
mov es,ax
mov bx,offset program_end                       ;BX contains the total code size
mov cl,4
shr bx,cl             ;BX now contains # of 16 byte paragraphs needed for code
add bx,31h                                                 ;extra space for stack
mov ah,4Ah ;execution of this DOS function frees the memory which is not used by
        ;by the program. This space will be used to allocate memory for row arrays.
int 21h
```

```
mov dx,offset row_msg
Call Display_String
Call ReadNum
mov num_rows,bx
mov dx,offset num_dat_msg
Call Display_String
Call ReadNum
mov num_dat_pts,bx
mov di,offset arr_of_rows                ;di points to the first entry in this array
mov bx,num_dat_pts                              ;#of data points in one row
mov cl,3
shr bx,cl                    ;divide by 8(array contains words,so /16x2) to find
inc bx                             ;the # of 16 byte paragraphs to be allocated
mov cx,num_rows                  ;total number of row arrays to be allocated is
;the number of lines to be scanned.
allocate:
push cx
mov ah,48h                              ;DOS function to allocate memory block
int 21h
jnc continue1                                                  ;no errors occured
mov dx,offset err_msg1
Call Display_string
jmp leaves
continue1:
pop cx
mov es,ax
mov [di],ax              ;mov the segment address of row array into the current
                  ;index of acoustic data array. Offset of row array is automatically
                                ;zero since allocation is done in this way by DOS.
inc di                              ;go to the next position in acoustic data array
inc di
loop allocate
xor cx,cx
mov es,cx                                                          ;mov 0 to ES
```

```
mov bx,dat_arr_ptr        ;put absolute address of acoustic data array into mail-box
mov ax,seg arr_of_rows
mov es:[bx],ax
inc bx
inc bx
mov ax,offset arr_of_rows
mov es:[bx],ax
mov bx,num_dat_pts_ptr
mov ax,num_dat_pts
mov es:[bx],ax                ;put number of points in each row into mail-box
mov bx,num_rows_ptr
mov ax,num_rows
mov es:[bx],ax                ;put number of rows to be scanned into mail-box
xor ax,ax
mov bx,DoIt
mov es:[bx],al                                    ;let resident program to operate
mov bx,Done
mov es:[bx],al                                          ;initialize Done to 0
mov dx,300h
mov ax,100b                   ;enable reset signals from position reset circuitry
out dx,ax
mov bx,Done
waits:
cmp byte ptr es:[bx],1                    ;wait for Done flag to be set by ISR
jnz waits
;ISR has read the acoustic data into the allocated memory
;rest of the program writes the acoustic data into a file
push ds
mov ax,seg outfile
mov ds,ax
mov dx,offset outfile
mov cx,0
mov ah,3ch                                              ;open the out file
int 21h
```

```
pop ds
mov outhdl,ax
mov bx,ax                          ;mov outhdl to BX for function req. 40h
push bx
mov cx,num_rows                    ;num_rows row arrays will be saved
mov ax,seg arr_of_rows
mov es,ax
mov si,offset arr_of_rows          ;get first row array index
mov di,num_dat_pts                 ;keep variable in DI since DS changes
push ds
writes:
push cx
mov ax,es:[si]
mov ds,ax
xor dx,dx          ;DS:DX contains the starting address of the memory block
        ;block to be written. CX contains the size of memory block in # of bytes
mov cx,di                                                    ;# of words
shl cx,1                                                     ;# of bytes
mov ah,40h
int 21h
inc si                                                       ;next row array index
inc si
pop cx
loop writes
pop ds
pop bx
mov ah,3Eh                                                   ;close file handle
int 21h
leaves:
mov ah,4ch
int 21h
program_end db ?
END program_start
```

IRQ7SR.ASM

```
DOSSEG
;Following addresses (3F6-3FC) are the offset addresses in the 0'th
;segment. This space is used as a mail-box between this program
;and READ.ASM
DoIt equ 3F6h                    ;flag that controls the activation of the ISR
Done equ 3F7h                    ;flag that indicates whether the read operation
;is complete or not
num_rows_ptr equ 3F8h            ;word containing the # of lines to be scanned
num_dat_pts_ptr equ 3FAh         ;word containing the # of data points in each line
dat_arr_ptr equ 3FCh             ;two words containing the absolute address of the
;acoustic data array. Each entry in acoustic data array is a pointer to an array (row
;array) which contains num_dat_pts words, to store one row of data. Size of acous-
tic data array is determined by READ.ASM and it should be larger than the total
;number of lines to be scanned.
inta0 equ 20h                                    ;port address of OCW2
inta1 equ 21h                                    ;port address of OCW1
EOI equ 67h                                      ;specific EOI for IR7
.MODEL SMALL
.STACK 200h
.286                                             ;enable 286 instructions
.CODE program_start:
cli
in al,inta1                                      ;read OCW1
and al,7Fh                                       ;program 8259A: enable IR7
out inta1,al                                     ;rewrite OCW1
sti
mov ah,25h                                       ;DOS set interrupt vector function
mov al,0Fh                                       ;IRQ7 will be used
mov dx,seg int_rut
mov ds,dx
mov dx,offset int_rut
int 21h
mov ah,31h                       ;DOS terminate and stay resident function
```

```
mov dx,offset e_of_interrupt-offset program_start
inc dx                                  ;amount of memory occupied by the code
int 21h
INT_RUT:
pusha                                                   ;store registers
push es
cli
jmp int_start
rep_num dw 0                                            ;count of ISR calls
num_rows dw ?                                   ;number of rows to be scanned
num_dat_pts dw ?                                ;number of data points in each row
dat_arr_seg dw ?                                ;segment address of acoustic data array
dat_arr_indx dw ?                               ;index within a row of the data array
int_start:
xor ax,ax
mov es,ax                                               ;mov 0 to ES
mov bx,DoIt                             ;mov flag address to BX, ES:[BX] is the flag value
cmp al,es:[bx]                                  ;compare the content of the flag with 0
jz continue                                     ;if it is 0, perform the rest of the operations
jmp leaves                                      ;if it is 1, leave ISR without doing anything
continue:
inc rep_num
cmp rep_num,1                                           ;is it the first call?
jnz skip
                                        ;if yes read the necessary information from mail-box
xor ax,ax
mov es,ax
mov bx,num_rows_ptr
mov ax,es:[bx]
mov num_rows,ax                                 ;store number of lines to be scanned
mov bx,num_dat_pts_ptr
mov ax,es:[bx]
mov num_dat_pts,ax                              ;store number of data points in each line
mov bx,dat_arr_ptr
```

```
mov ax,es:[bx]
mov dat_arr_seg,ax                    ;store segment address of data array
inc bx
inc bx
mov ax,es:[bx]
mov dat_arr_indx,ax                   ;store offset address of data array
skip:
mov es,dat_arr_seg
mov bx,dat_arr_indx
mov ax,es:[bx]
mov es,ax                ;ES contains the segment address of the row array
mov si,0                            ;array is allocated by DOS at index 0
mov dx,304h                                 ;read from port address 304h
mov cx,num_dat_pts ;read num_dat_pts points
loop1:
in ax,dx                              ;get the acoustic data from DAIC
and ah,0Fh                                          ;set bits 12-15 to 0
mov es:[si],ax                                      ;store it in the array
inc si                              ;go to next position in the row array
inc si
loop loop1
inc dat_arr_indx                  ;go to next position in acoustic data array
inc dat_arr_indx
mov ax,num_rows
cmp ax,rep_num                      ;check if all of the rows are scanned
jg leaves                                           ;not scanned yet
xor ax,ax
mov es,ax
mov bx,Done                         ;all rows are scanned, set done flag
mov byte ptr es:[bx],1
mov bx,DoIt                                         ;deactivete IRQ7SR
mov byte ptr es:[bx],1
mov rep_num,ax                                      ;reset rep_num
mov dx,300h
```

```
mov al,000b                    ;disable reset signal from the position reset
out dx,al                      ;circuitry to prevent further IRQ7 calls
leaves:
mov al,EOI                     ;send specific End Of Interrupt command to 8259A
out inta0,al
sti
pop es
popa                                                    ;restore registers
iret
e_of_interrupt:nop
END program_start
```

REFERENCES

[1] Calvin F. Quate "The Acoustic Microscope," *Sci. Am.*, pp. 62–70, October 1979.

[2] Calvin F. Quate "Acoustic Microscopy," *Physics Today*, August 1985.

[3] Martin Hoppe and Walter J. Patzelt "Acoustic microscopy gains resolution, finds new applications," *Research & Development*, April 1984.

[4] Andrew Briggs. *Acoustic Microscopy.* Oxford Science Publications, 1992.

[5] Abdullah Atalar. "TÜBİTAK EUREKA-525 Projesi 1. Gelişme Raporu,". Technical report, Bilkent University, 1991.

[6] Andrew Briggs, ed. *Advances in Acoustic Microscopy*, chapter Lens Geometries for Quantitative Acoustic Microscopy. Plenum Press, 1995.

[7] Gordon S. Kino. *Acoustic Waves: Devices, Imaging & Analog Signal Processing.* Prentice Hall, 1987.

[8] Edmund Strauss. *80386 Technical Reference.* Brady, 1987.

[9] Walter A. Triebel. *The 80386DX Microprocessor: Hardware, Software, and Interfacing.* Prentice Hall, 1992.

[10] John H. Crawford and Patrick P. Gelsinger. *Programming the 80386.* Sybex, 1987.

[11] Steven Armbrust and Ted Forgeron. *Programmer's Reference Manual for IBM Personal Computers.* Dow Jones-Irwin, 1986.

[12] Burr-Brown. *Burr-Brown Integrated Circuits Data Book Supplement Volume 33c*, 1992.

[13] National Semiconductor Corporation. *Linear Databook*, 1982.

[14] National Semiconductor Corporation. *CMOS Logic Databook*, 1988.