THE DEVELOPMENT OF A BUILDING ACOUSTICS ANALYSER

Prepared by:

MAON CATZEL, H.N.D.(Elec.) Engineering, CAPE TOWN.

Submitted to the Cape Technikon in partial fulfilment of the requirements for the Masters Diploma in Technology.

November, 1991.

SYNOPSIS

The purpose of this thesis was to develop a portable, stand-alone building acoustics measuring instrument that would replace the existing equipment used. The existing equipment consisted of a number of instruments making it bulky and time-consuming to operate. The proposed Building Acoustics Analyser (BAA) was to incorporate all the existing equipment into a single computer-controlled instrument. It was anticipated that the completed instrument would increase the speed of measurements and improve the accuracy of the results.

A sound generating section and a sound measuring section of modular design were combined in the BAA. These sections consisted of analog cicuitry that performed all the analog processing and was controlled by digital circuitry. The analog signals were converted to digital format and interfaced to a computer and software package. This enabled measurements to be performed automatically and the data to be stored in memory for further processing.

Tests performed on and with the BAA showed that the instrument conformed to all the applicable standards and specifications. The BAA reduced measurement time by about 75 percent and the results proved to be more accurate than the traditional methods. The BAA, because of its modular design, also offers the ability of being modified to perform other types of sound analysis.

ACKNOWLEDGEMENTS

I would like to thank Mr A.W.D Jongens, Director of the Central Acoustics Laboratory at the University of Cape Town for the enormous help provided in supervising this project.

Mr T. Lind at the Cape Technikon for his supervision of the thesis on behalf of the Cape Technikon.

Professor J.F.Bell of the U.C.T Dept. of Electrical Engineering provided advice on operational amplifier and general circuit design techniques.

Mr S.Schrire of the U.C.T Dept of Electrical Engineering provided advice on the selection of components and also helped with circuit design terchniques.

Mr W. Wenzelburger of the Central Acoustics Laboratory for his advice and helping to get the instrument in its final state of completion.

The secretary of the Department, Mrs R. Perl and all the people of the Central Acoustics Laboratory for their team spirit and assistance throughout the project.

CONTENTS

PAGE

LIST OF ILLUSTRATIONS

NOMENCLATURE

(a) V = volume of room, (meters cubed)
(b) S = surface area of material, (meters squared)
(c) c = speed of sound, (meters per sec)
(d) T1 = empty room reverberation time, (sec)
(e) T2 = room with material reverberation time, (sec)
(f) R.T = reverberation time, (sec)
(g) S.P.L = sound pressure level, (dB)
(h) L1 = source room S.P.L
(i) L2 = receiving room S.P.L
(j) H.P.M = Hewlett Packard Multiprogrammer
(k) B & K = Brüel and Kjaer

1. INTRODUCTION

This thesis was commissioned jointly by Mr. A.W.D Jongens Director of the Central Acoustics Laboratory of the University of Cape Town, and Mr. T. Lind of the Cape Technikon. Mr. Jongens requested that a portable, stand-alone, automated measuring system capable of performing all field and laboratory measurements of classical building acoustic parameters should be developed.

These parameters being:

(1) Reverberation Time
(2) Sound Absorption Coefficients of materials
(3) Sound Insulation of partitions

These measurements complied with the relevant national and international standards as detailed in section 2.2

" Some in-situ measurements are made in accordance with Standards, while others are developed for a particular purpose. In either case there is often a need to increase the speed, improve the accuracy, or increase the versatility of the method."[1]

The thesis objectives were:

1. To use the available equipment, ie. Hewlett Packard Multiprogrammer (H.P.M) and H.P 9836C Computer to digitize the signals from the existing measuring equipment and develop the software to automate and analyze the measurements. This familiarised the author with the present measuring process prior to designing and building the stand-alone system.

---

1 Inst. of Acoustics (1990:vol 58,11)

2. To design and build a stand-alone, portable measuring system that interfaced to an IBM computer and would be controlled by a menu driven software package.

3. To evaluate the completed system and recommend possible modifications or additions to the final system.

The existing manual methods of measurement of classical building acoustic parameters are time-consuming and imprecise, due to the analog interpretation of results. The equipment is also large, bulky and expensive.
The need for a portable, stand-alone instrument that would perform all the functions of the various equipment presently used, is clearly needed.
The instrument will increase speed of measurement process to a large degree and improve accuracy of the results.
The author became familiar with the procedures and practical measurements presently used with the existing equipment.

Sound Absorption and Insulation tests were carried out and various parameters were observed, ie. accuracy of results, total time duration of each test and amount of operator involvement. Various methods of automating the measurement procedures with greater speed and accuracy were investigated.

The instrument was designed, built and tested for correct operation within the specifications.

Finally, conclusions were drawn as to the effectiveness of the system as compared to the old methods, and recommendations made.

2. BACKGROUND TO ACOUSTIC MEASUREMENTS

2.1 Introduction

Architectural acoustics can be defined as the study of the generation and propagation of sound in rooms and buildings. If the principles of acoustics are correctly applied, the quality of life in most situations is improved. Various recommendations and regulations exist to aid us during the design stage of buildings, so that future alterations need not be necessary.

In some cases alterations have to be made to an existing building so as to improve the acoustics. In order to achieve this, various measurements have to be carried out to find the cause of the problems. The measure of **reverberation time** is one of these measurements.

Reverberation is the term applied to the persistence of sound in an enclosure due to the reflections of sound at walls and other structures after the sound source is turned off. Due to the partial absorption of sound at the room walls and contents, whenever sound impinges on and reflects from these surfaces, the sound energy is reduced in the room resulting in a decrease in sound intensity with time. It is of a decaying nature as shown in figure 1 on page 12.

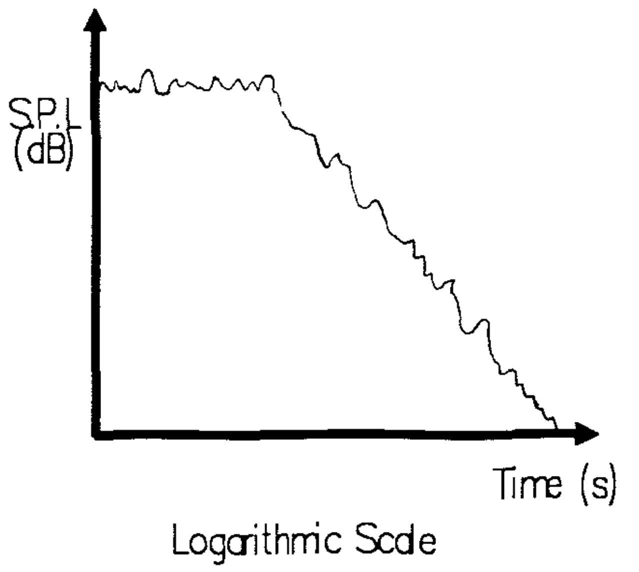


Figure 1: Decay of sound in reverberant room

It is dependant on the size and shape of the enclosure, the sound absorption properties of room surfaces and contents and the frequency spectrum of the sound source. The reverberation time (R.T) at a particular frequency is the time taken for a steady-state sound to decrease by 60 dB after the sound source is turned off.

W.C.Sabine did a considerable amount of research on the acoustics of auditoria and arrived at the following formula:

$$R.T = \frac{0.161 * V}{A}$$

The absorption unit of 1 m^2 sabin represents a surface area capable of absorbing sound at the same rate as 1 m^2 of a perfectly absorbing surface, ie. such as an open window. The absorption coefficient of a material, as defined by Sabine, is the ratio of the sound absorbed by the material to that absorbed by an equivalent area of open window. Therefore, if the various areas and absorption coefficients are known, the reverberation times of an auditorium can be calculated at the design stage. To enable the acoustician to perform these calculations, sets of tables have been derived presenting the absorption coefficients of commonly used building materials as a function of frequency.

2.2 Applicable Standards

The object of international standardization is to set up a set of "rules" to enable a common method of measurement and assessment of products and parameters. It also enables the exchange of goods and services with the participating countries. This enables mutual co-operation and improves the prospect of development. The International Electrotechnical Commission (IEC) and the International Organization for Standardization (ISO) are two bodies that control acoustic related standards. The applicable standards are listed below:

(a) ISO Recommendation R266, 1975 : List of Preferred Centre Frequencies.

This recommendation contains a table of preferred frequencies for acoustic measurements. For bandpass filters the frequencies listed in the table should be in the geometric center frequencies of the band. Refer to Appendix 1 for the table of frequencies.

(b) ISO Recommendation R354, 1985 : Measurement of Absorption Coefficients in a Reverberation Room.

The recommendation concerns sound absorption measurements in specially constructed reverberation rooms. The volume of the room should be close to 200 m^2 and various methods of obtaining a diffuse sound field are suggested, ie. non-parallel boundaries, diffusing elements. The test specimen should cover a single area of 10 m^2. The procedure to be followed is described in section 2.3.

(c) **ISO 140/parts (I-IV), 1980 : Measurement of Sound Insulation in Buildings and of Building Elements.**

Measurement of sound insulation in buildings and of building elements. It consists of the statement of precision requirements, laboratory and field measurements of airborne sound insulation.

(d) **ISO 717/parts (1-3), 1984 : Rating the Sound Insulation in Buildings and of Building Elements.**

Rating the sound insulation in buildings and of building elements. It consists of the methods for rating the airborne and impact sound insulation in buildings and of interior building elements.

(e) **IEC 225, 1989 : Octave Band and Fractional Octave Band Filters**

Tolerance-limit requirements for octave band and fractional-octave band filters.

(f) **IEC 651, 1979 : Sound Level Meters.**

Specifications for sound level meters for the measurement of certain frequency and time weighted sound pressure levels.

2.3 Existing Methods of Measurement

(1) Laboratory tests.

A laboratory test is used for measuring absorption properties of materials or structures and sound transmission loss of partitions in a controlled environment - reverberation rooms.

(2) Field tests.

A field test is used for field measurements of sound transmission loss and reverberation time in an uncontrolled environment - any given room.

Equipment setup for all tests:

The sound source is a white/pink noise generator fed through an amplifier to an omni-directional loudspeaker mounted in the corner[2] of the room. In all rooms, locating the loudspeaker in the corner of the room excites the greatest number of modes of vibration.
A third octave band filter is connected between the noise generator and the amplifier to limit the noise within a selected frequency band, and also to reduce the power to the loudspeaker. A microphone is placed at three random positions in the room. At each position, three measurements at each third octave frequency band are taken. The microphone signal is fed to an amplifier/filter network which displays the reading. The filter is the same as the one used in noise generator circuit and is used to increase the dynamic range of the signal by attenuating background noise outside the frequency band measured.

---

2 Beranek L. (1962:806)

Reverberation Tests:

The output signal from the instrument is fed to a level recorder on which the sound decay slopes are printed on calibrated paper. The decay slopes are interpreted to give the reverberation times by manually using a special protractor. The protractor is calibrated for the paper speed and type of logarithmic potentiometer used by the level recorder. R.T tests may be performed in any environment. A diagram of the equipment layout is shown in figure 2 on page 17.

Absorption Tests:

The reverberation times at third octave frequencies are measured in the empty room, then measured again with a material sample subsequently added. The difference in reverberation times are used in the following formula :

$$\text{Absorption} = \frac{55.3 * V * (1/T2 - 1/T1)}{S * c}$$

This equation provides the per unit absorption coefficients of the material at each third octave frequency. Absorption tests are, in general, conducted in a laboratory environment, because of the requirement for the repeatability of results.

Transmission Loss Tests:

For transmission loss tests, the S.P.L value for each third octave frequency band is visually averaged and recorded manually by the operator. Measurements are performed in the source room, then the receiving room.

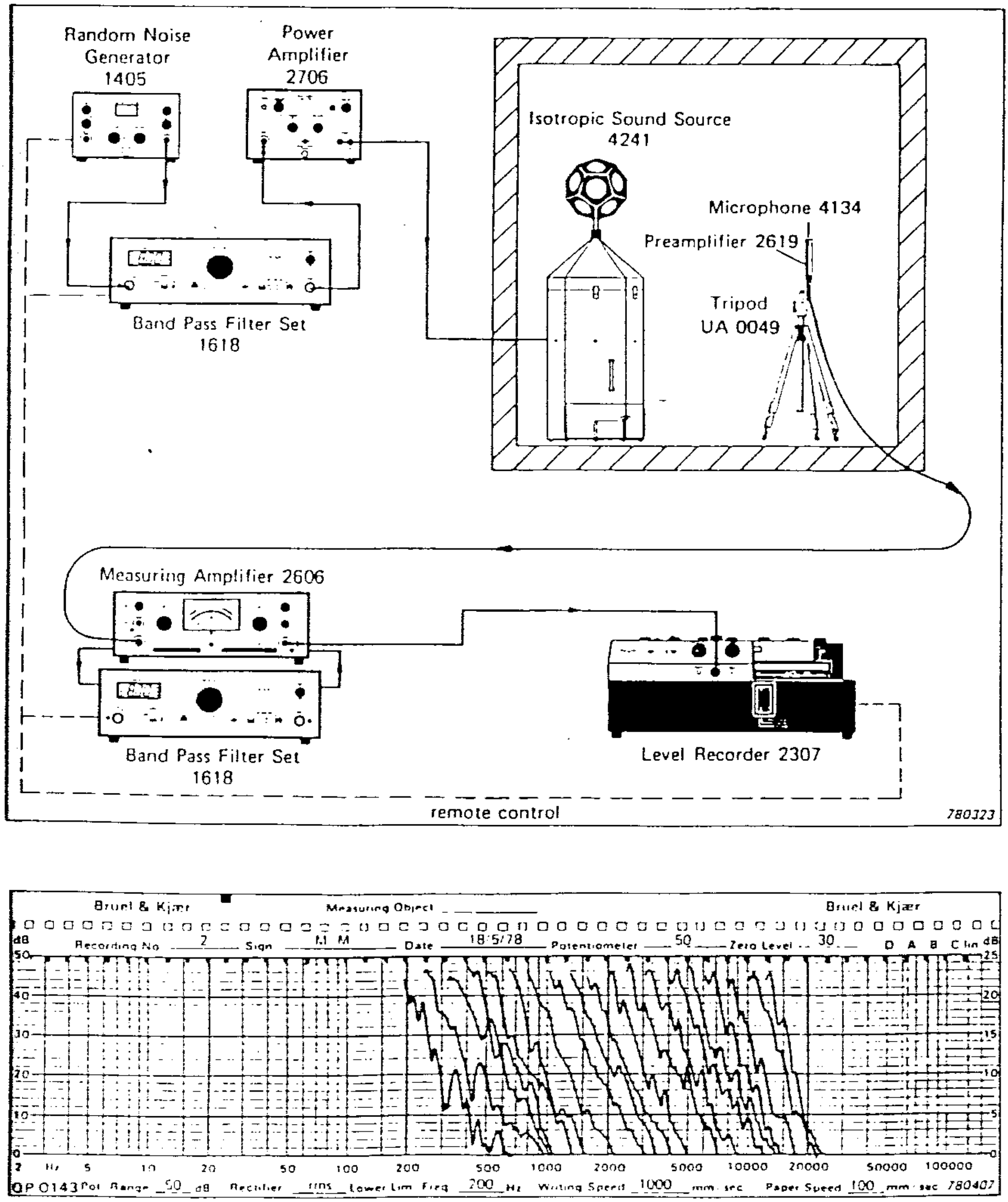


Figure 2: *Automatic arrangement - Paper loop method*

Ref: Architectural Acoustics, Bruel & Kjaer, 1978, K.B.Ginn

In a laboratory environment, the sound reduction index (R) is calculated from:

$$R = L1 - L2 + 10*\log \frac{S}{A} \quad (dB) \quad \text{(Refer to nomenclature)}$$

The correction term of the above equation containing the equivalent absorption area is evaluated from the R.T measured according to ISO R.354 in the receiving room and evaluated using Sabine's formula below:

$$A = \frac{0.163*V}{T}$$

A single number rating, R_w, is derived from the sound reduction values in accordance with ISO 717.

This is accomplished by comparing the reference values of ISO 717, part 1, with values obtained during the measurements. The value of the curve at 500 Hz after shifting it in accordance with the stated method, gives the rating value.
All measurements should give satisfactory repeatability, otherwise they should be repeated.
For field tests, the single number rating D_{nT} is calculated from one of the following formula :

Level difference, $D = L1 - L2$

Standardized level difference, $D_{nT} = D + 10*\log \frac{T}{T_O}$ (dB)

T is the R.T of the receiving room, T_O is the reference R.T equal to 0.5 s for dwellings. The standardizing of the level difference to a R.T of 0.5 s takes into account that in dwellings with furniture the R.T has been found to be equal to 0.5 s. D_{nT} is however, dependant on the direction of the sound transmission if the two rooms have different volumes.

Apparent sound reduction index, $R' = L1 - L2 + 10 \log \frac{S}{A}$

This equation follows the same principle as the equation for laboratory tests.

3. B.A.A PERFORMANCE SPECIFICATION

Noise Generator:

The noise consists of Gaussian white or pink noise spectrum in the range 20 Hz to 20 kHz.
The peak output level is 2.0 volts rms and is level within ±0.5 dB in the range. The between burst level is 0 volt.
The fall off slope above 20kHz is > 18 dB/oct. The pink noise is derived from a -3dB/oct filter added to the white noise output.
Output load impedance is 100 Ω.
Signal to hum ratio is > 60 dB for both outputs.
The generator start/stop function is controlled by the software package.

Filter Set Type:

Frequency range from 89.09 Hz to 11.23 kHz (third octave).
Bandpass filters consisting of 20 active 8^{th} order Butterworth third octave filters, in accordance with IEC 225-1989, Class 2 and ANSI S1.11-1966, Class II & III standards.
Centre Frequencies from 100 Hz to 8000 Hz in third octave steps.
Attenuation at Centre Frequencies is 0 dB ±0.5 dB.
Peak to valley ripple is < 0.5 dB.
Attenuation outside Pass Band > 60 dB at 4 f_o and 1/4 f_o.
Filter shift controlled by software package.
Input impedance of 10 kΩ in series with 10uF.
Maximum input voltage of 3.2 V_{rms} sinus.
Output impedance of < 5 Ω in series with 10uF.
Inherent noise floor of ± 1 mV.

Measuring Inputs/Ranges:

Six BNC type input connectors each with range control switches. Four switches with ranges: 1V, 3V, 10V, 30V in clockwise direction correspond to steps of attenuation of 10 dB. Must be matched to AC output of sound level meter or measuring device connected to BAA instrument.

Overload Indicators:

Overload occurs when the output of the input amplifiers exceeds 3.4 volts rms sinus, on all input settings. Front panel red LED lights during overload condition.

Frequency Weighting:

A, C weighting in accordance with IEC 651, Type 1 meter. Linear response from 20 Hz to 20 kHz.

Detector:

True RMS to DC conversion characteristic.
Linearity range of 60 dB.
Crest factor capability of 7 for one percent error.
Maximum output of 2 V_{rms}.

Time Weighting Characteristic:

"F": Fast response conforming to IEC 651, Type 1 meter.

Analog To Digital Converter:

12-bit resolution and linearity.
10us (100 kHz) conversion time capability.
Very stable on-chip voltage reference.

Personal Computer Interface:

Utilises 8255 Programmable Peripheral Interface chip on PC-based Interface Card.
Contains three 8-bit I/O Ports for interfacing by two 26-way D-Type connectors and ribbon connector cable.

4. THE H.P. MULTIPROGRAMMER SYSTEM

4.1 Introduction

The Multiprogrammer formed part of the first method of using the available equipment to familiarise the author with the measurement process. It formed the last unit in the measurement system and was used for semi-automating and recording the data from the measurements.
The Multiprogrammer offered a flexible mainframe of In/Out cards for use in a broad spectrum of applications, such as data acquisition and conversion. Each In/Out card had a variety of functions and modes to suit the user's application, limited only by memory space and sampling rates.
Various hardware parameters such as positive or negative slope triggers, resolution of the A/D converter, etc were able to be configured when setting up the system. These were constants which do not change throughout the measuring process.

4.2 System Overview

The analog input signal is sampled and converted by an Analog to Digital Converter card which has a maximum conversion time of 30 micro-seconds. It has a 12-bit digital output, which represents a dynamic range of 72 dB. This is adequate for room acoustical evaluations and R.T analysis which requires a 60 dB range[3].
The A/D output is connected to the data input of a pair of Memory cards - this forms the basic Buffered A/D system. Memory card 1 contains a 4K word by 16-bit block of RAM memory and handshake control circuits. Memory card 2 contains a sequencer and counter/registers which control addressing and read/write access to the RAM memory.

---

3 ISO R354 (1985:6)

The A/D card has an **End of Conversion** output which goes logic low indicating when data conversion has been completed. This logic low signal is fed into the **Input Data Available** line telling the memory that data is available and must be stored. The Memory card can operate in the First-In/ First-Out (FIFO) or Recirculate mode. The FIFO mode allows the memory to be used as a buffer, storing 4096 bits of data until it is full, after which all data sent to the card is lost. The Recirculate mode is mostly used in waveform digitization where data inputted to the card is continuously updated, ie. the old data is overwritten. We make use of the Read and Write pointers to determine the position from where data must be read by the software. The Read pointer stores the address of the next memory location from which data can be read into the controller. The Write pointer stores the address of the next memory location in which data can be stored. The Read pointer is set to the beginning of the required data block by subtracting the number of readings to be retrieved from the maximum size of the memory card. This value is then added to the value of the Write pointer as shown : Readpointer = Writepointer + (4096 - Total reads). *Figure 3 shows the system layout.*

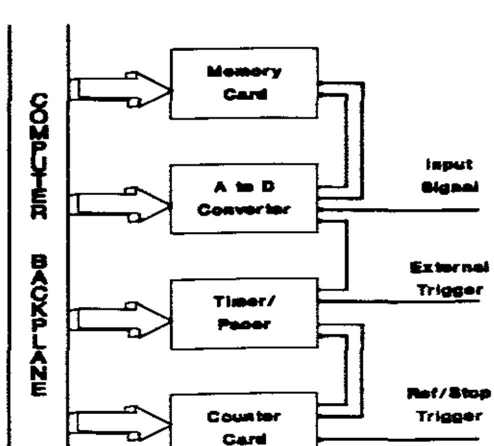


Figure 3: Basic system diagram

The system is controlled by a Timebase unit consisting of a Timer card and Counter card. The A/D is triggered externally by a stream of programmable period pulses from the Timer card - which is cycled by the software. The Timer has two outputs, **OPT** and 'not' **OPT**.

The **OPT** triggers the A/D process while the 'not' **OPT** is connected to the Counter's Up-count input. In this way the Counter is able to keep track of the number of pulses generated by the Timer card. The Most Significant Bit (MSB) of the Counter is connected to the External Enable input of the Timer and must be set to enable the Timer. To set the MSB, the Counter must be preset to a value between 32768 and 65535. This value determines how many pulses will be counted before the Counter rolls over to zero, causing the Timer to disable.

If the preset value is 32768, the card will count 32767 pulses before it rolls over. If preset to 65535, the Counter will roll over as soon as it is triggered. The Counter is triggered by a Stop/Reference trigger value, which is either a signal value from the event being digitized or in some way related to the event.

The 4K block of readings can be moved in relation to the Stop/Reference trigger by varying the preset value of the Counter. This process allows pre-event and post-event triggering.

If the trigger occurs before the event, the 4K block of readings can be delayed to record the event - using preset values from 32768 to 61539. If the trigger occurs during or after the event, then the readings in memory contain data of the event - using preset values from 61538 to 65535. Refer to figure 4 on page 26.

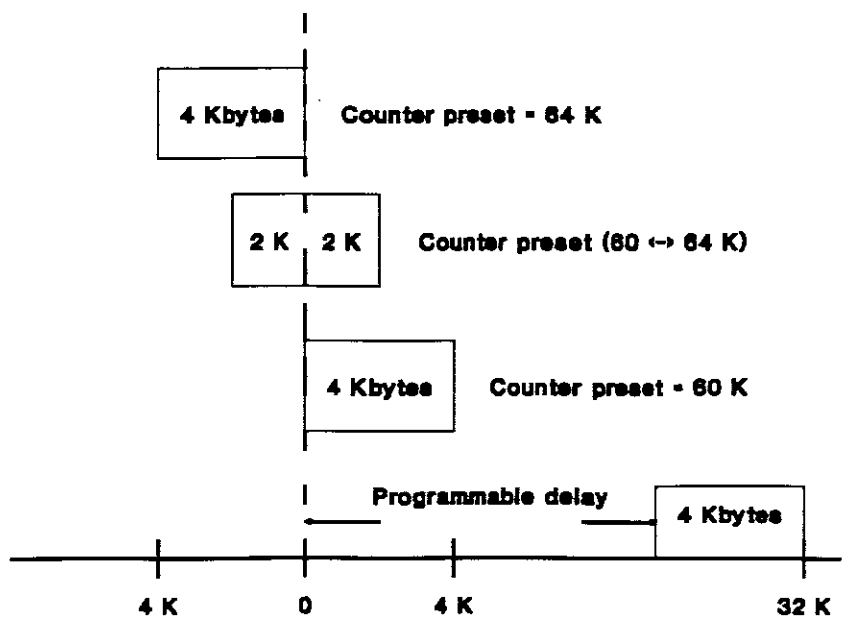


Figure 4: Memory allocation for various triggers

The Multiprogrammer cards found to be useful were the 33kHz A/D, 4K Memory, Timer/Pacer, Counter, and D/A cards. Only one Counter card was available limiting the number of Timebase counts to a maximum of 32K (32768) before the system disabled. This only limits the number of readings taken after the Stop/Ref trigger is received, but not the total number of readings. The total readings are limited by the size of the memory banks.
If the Counter is preset to any value below 32768, the MSB is automatically set to zero causing the system to disable immediately, ie. no data is captured.

By using a Voltage D/A card, a programmable reference level for the trigger can be set. This value is fed into the non-inverting input of the up-enable comparator on the Counter's circuitry. The analog input signal is fed into the inverting input of the comparator, allowing voltage level triggering to occur.

When the signal reaches the predetermined voltage level, the Counter enables via its up-enable comparator circuitry, as explained previously. The Counter starts counting from the preset count value, between 32K and 64K, depending which window of time with respect to the event is required. Both pre- and post- trigger data is required, therefore the count value is set to (65535 - 4096/2), if 4K of memory is used. This produces half pre-trigger and half post-trigger data. This method is flexible in that we can capture any ratio of pre- and post trigger data.

4.3 Operation of Multiprogrammer System

The H.P.M system was set up for the event triggered mode using the cards described in section 4.2.
The A/D and Memory cards were used in the First-In First-Out (FIFO) mode allowing 4K Bytes of data to be stored in the memory. The Timer card was used to trigger the A/D converter via the software program and started the measurement process using a sampling rate of 1.5 kHz which enabled enough data points to be captured for further analysis.

Measurement procedure:

A random pink/white noise generator (B & K type 1405) was activated remotely via an NPN transistor (BC107) by the Digital Output card of the H.P.M. This was controlled by the software program. The noise signal was filtered through a manually operated one octave bandpass filter (B & K type 1613) to a power amplifier to a loudspeaker (type Tannoy) in the reverberation room. The system setup is shown in figure 5 on the page 28.

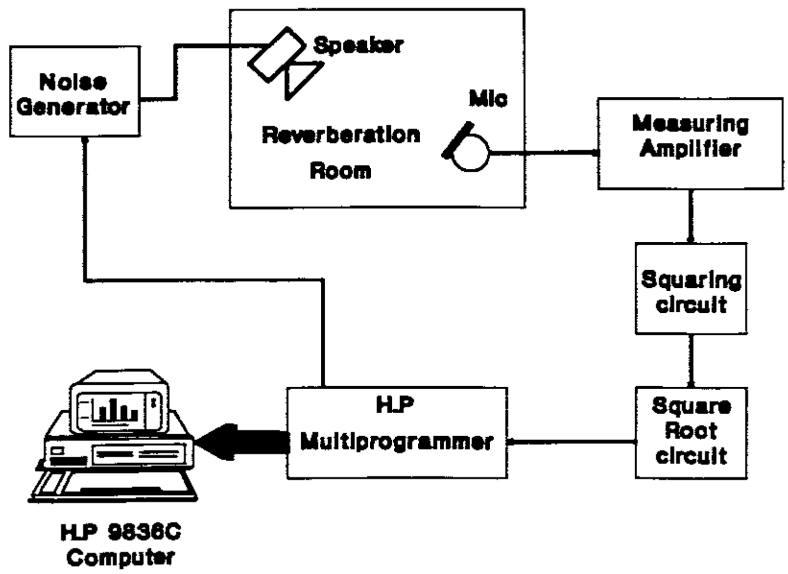


Figure 5: Multiprogrammer system setup

The microphone (B & K type 4134) and pre-amplifier (B & K type 2619) in the reverberation room fed the signal to a measuring amplifier (B & K type 2607) and manually operated third octave bandpass filter (B & K type 1614). A calibrated signal was then fed through a squaring circuit to produce a signal which is always positive, then fed through a square root circuit to obtain the original magnitude of the signal, so that further processing could be correctly done.
The signal was then fed to the A/D converter and memory of the H.P.M where a snapshot of the decay slope was stored.

In the Recirculate mode the memory buffer holds 4096 data bits, receives the next bit as being the trigger bit, then rewrites over the next 2047 data bits before it disables. It disables due to the Counter rolling over to zero, causing the Timer to disable. The memory therefore holds 2048 bits of pre- and post-trigger data each. This window of events

was adjusted via the software program so that an optimum window of the decay slope was available.
The data from the memory buffer was downloaded to the Controller (HP 9836C) memory into the array results(*). This array is structured of 100 rows by 41 columns and the function **RSUM** was used to average each row of 100 data points. The data was averaged to reduce fluctuations in the decay curve which would cause inaccuracy of the results. These values were then used in a plotting routine which produces a smoother graph from which the decay rate and reverberation times were calculated as shown in figure 6.

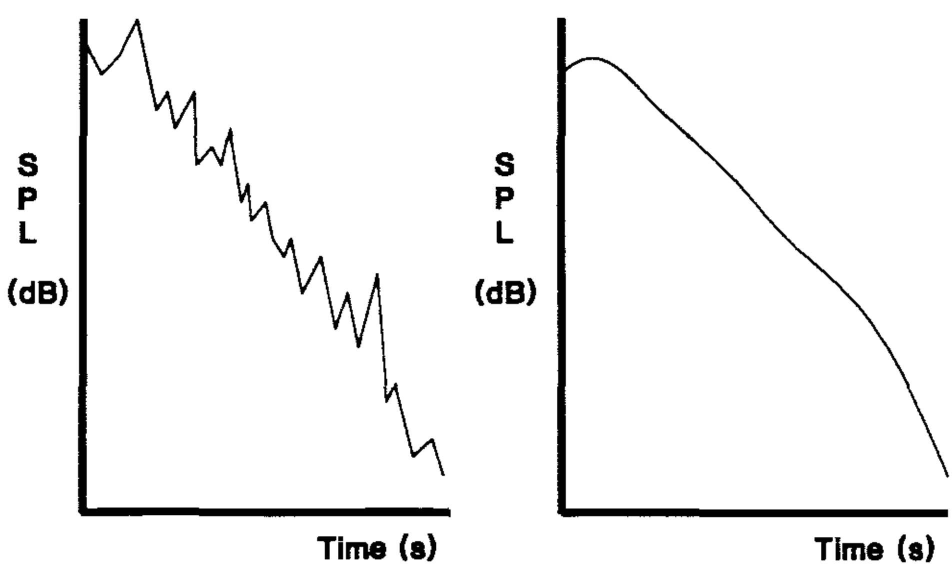


Figure 6: Comparison of decay curves

The software program controlled the measurement start time, the stopping of the noise generator and the measurement stop time. The exact start location of the decay slope corresponded to the noise generator stop time.
The record of the decay rate given by the curve was approximated by a straight line in the region from 5 dB to

about 35 dB below the stationary reference level. The slope of this straight line represents the decay rate in dB/sec and determines the reverberation time[4].
The least squares method using the best straight line fit was applied to the data to give the gradient of the line which is proportional to the decay rate of the slope. The coefficient of determination is also calculated and should be close to unity to prove accuracy of the results.
The R.T is the time taken for the measured S.P.L to decrease by 60 dB and is given by:

$$T = \frac{60}{d} \qquad d = \text{decay rate}$$

These times are stored in records for each third octave frequency band and microphone position.
The early decay time (EDT) which, using the same numerical scale as R.T, defines the decay over the first 10 dB - (and then as if the decay continued over 60 dB). The EDT was found to be more closely correlated with subjective criteria than the R.T.[13] This data was recorded for observation only.
The option of determining the R.T over 20, 30 and 40 dB regions was allowed in cases where the difference between maximum S.P.L and minimum S.P.L was less than 60 dB. If no slope was detected an error message was displayed and recorded.

4 ISO R354 (1985:8)
13 Jordan V.L (1981:253)

4.4 Conclusion

The first method using the available equipment proved to be quite successful producing reasonably accurate results and reducing measurement time considerably once the equipment was set up. A major drawback was that the system could only be used in the laboratory (not mobile) and all the existing bulky equipment had to be used in addition to the H.P.M unit.

The software controlling program designed for the H.P computer/ H.P.M formed a good framework on which the final software package was based. The software was required to be converted to an IBM compatible language viz. Turbo Pascal.

5. THE BUILDING ACOUSTICS ANALYSER CIRCUITRY

Introduction

The BAA automates the measurement and data calculation processes, thereby simplifying the traditional methods of measurement in building acoustics.
This was accomplished by replacing the bulky and unwieldy equipment traditionally used, with a modern, computer - controlled instrument that performed the measurements quickly and accurately.

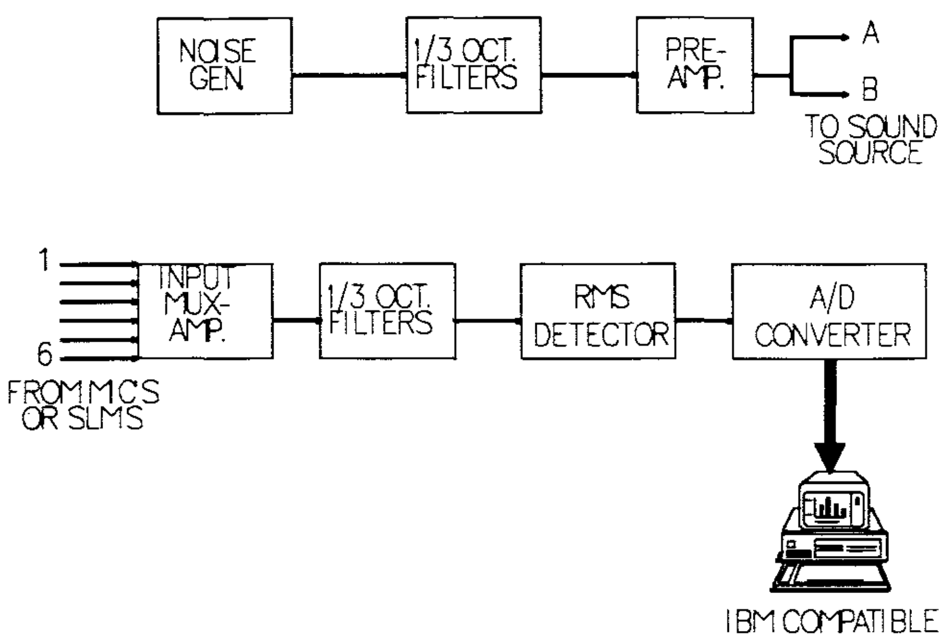


Figure 7: Simplified overall block diagram.

The instrument consists of a noise generating section and a sound measuring section of modular design, as shown in figure 7. The theory and circuit descriptions including diagrams for each module are explained in the following chapters.

THE SOUND GENERATING SECTION

5.1 The Noise Generator

Introduction

The noise generator is used for the generation of wide-band noise for the airborne excitation of sound in rooms and auditoria. There are three possible ways of generating this noise source :

1) Pistol shot
2) Wide band noise
3) Narrow band noise

The first method is the simplest, but is not often used because of its lack of reproducibility and lack of energy content at the lower frequency bands.
The more favourable method is to electronically generate the noise using a wide band of frequencies. The noise is delivered to the room using a powerful amplifier and loudspeaker.
An extension of this is to filter the noise into narrow band noise for sound measurements, which allows a greater S.P.L level in the frequency band of interest to be generated without overloading the loudspeaker. The International Standards dictate that octave or third octave bands should be used. A band of noise is necessary, as opposed to single frequencies, to overcome the problem of room resonances when doing reverberation measurements.
Two types of noise are used: **white** and **pink** noise.

White noise has equal noise energy in equal bandwidths over the total frequency range. Therefore, in the band 100 to 200 Hz there is the same amount of energy as from 5000 to 5100 Hz. When white noise is filtered by adding a -3 dB/oct

filter, pink noise is obtained.
Pink noise has equal energy per percentage change in bandwidth. Therefore, in the band 100 to 200 Hz there is the same amount of energy as from 5000 to 10000 Hz. Pink noise therefore appears to have more bass content than white noise and a more uniform output level in audio testing, which conventionally uses constant percentage bandwidths, ie: octave or third octave bands.

White/ Pink Noise Generator

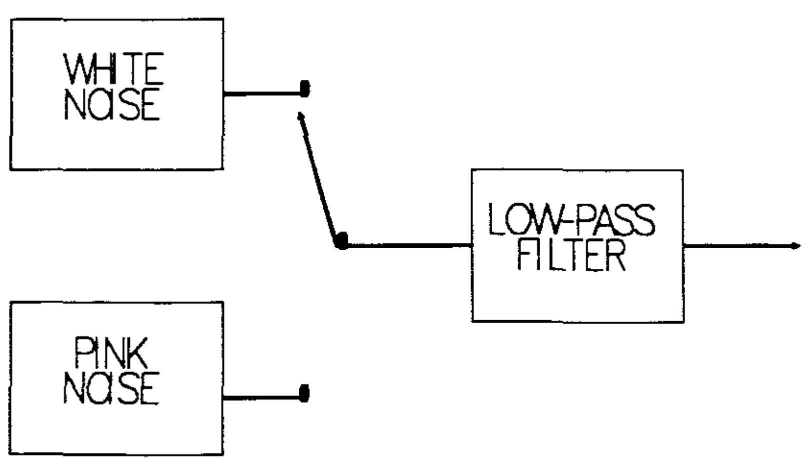


Figure 8: Block diagram of noise generator

One method of generating the noise is to use a 23-bit pseudorandom white noise generator chip, National's MM5437[6], which would be used as the main component of the noise generator circuit. It generates very accurate (±0.25 dB)

6 Horowitz & Hill (1989:452,655)

random Gaussian noise, but was unfortunately not available.

The second and most common method is to use a zener diode as a noise generator. The zener diode provides a flat spectral density from 20 Hz to ±50 kHz. Refer to circuit diagram in figure 9 on page 36.

Transistor Q1 is used as a zener diode by reverse biasing the base-emitter junction which goes into zener breakdown at about 7 to 8 volts. The zener noise current from Q1 flows into the base of Q2, which acts as an amplifier, such that an output of about 150 millivolts of white noise is available. The 'zener' also biases Q2 correctly, and the noise output of Q2 is fed directly to the White Noise output. Capacitor C2 removes the dc component.

To convert the white noise to pink noise a special filter is required which provides a -3 dB attenuation per octave as the frequency increases. A single RC network provides an attenuation of -6 dB/oct, therefore a special network of R's and C's is required to approximate the -3 dB/oct slope.
The problem is that such a network attenuates the noise output, therefore an amplifier is used to restore the output level to that of the white noise output level.
Transistor Q3 is used as the amplifier and the pink noise filter is connected as a feedback network between collector and base. This is to obtain the required characteristic by controlling the gain vs. frequency of the transistor. The pink noise output is obtained at the collector of Q3.
Both noise outputs are accurate to within ±0.5 dB ripple in the frequency band from 20 Hz to about 50 kHz.

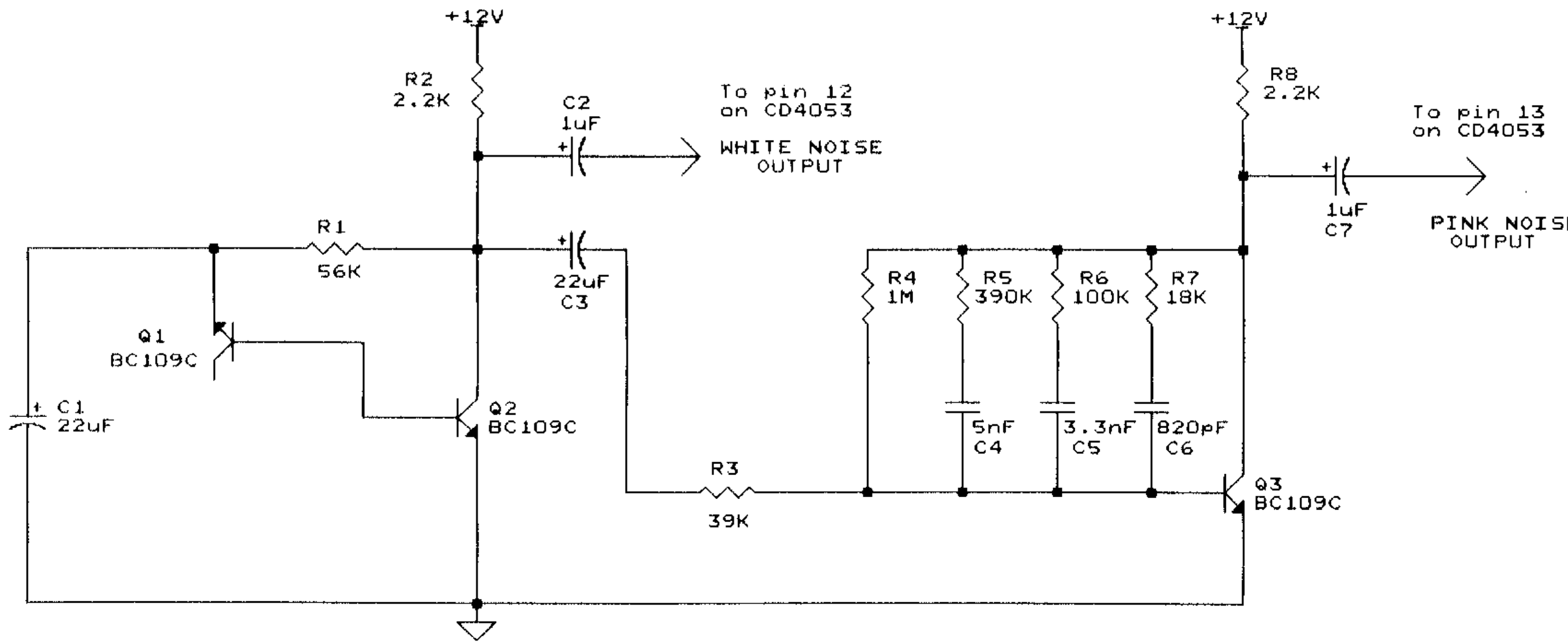


Figure 9: Circuit diagram of Noise Generator

Low-Pass V.C.V.S Filter

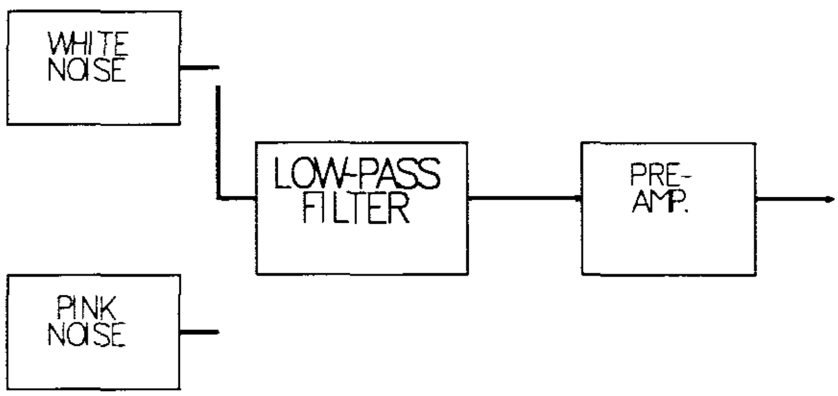


Figure 10: Block diagram of low-pass filter

The low-pass filter is necessary to limit the upper frequency of the noise outputs to 20 kHz, because we are only interested in the audio range of frequencies. Refer to the circuit diagram in figure 11 on page 38.
The voltage-controlled voltage-source (V.C.V.S) filter is a variation of the Sallen and Key filter. It replaces the unity-gain follower with a non-inverting amplifier of gain greater than 1. The number of components needed are minimised (2 poles/op-amp) and it offers non-inverting gain, low output impedance and ease of gain adjustment.
A quad op-amplifier (LF347N) is used for in/out buffering and for the actual filtering. The input buffer isolates the filter from changes in input impedance which may affect the response of the filter.

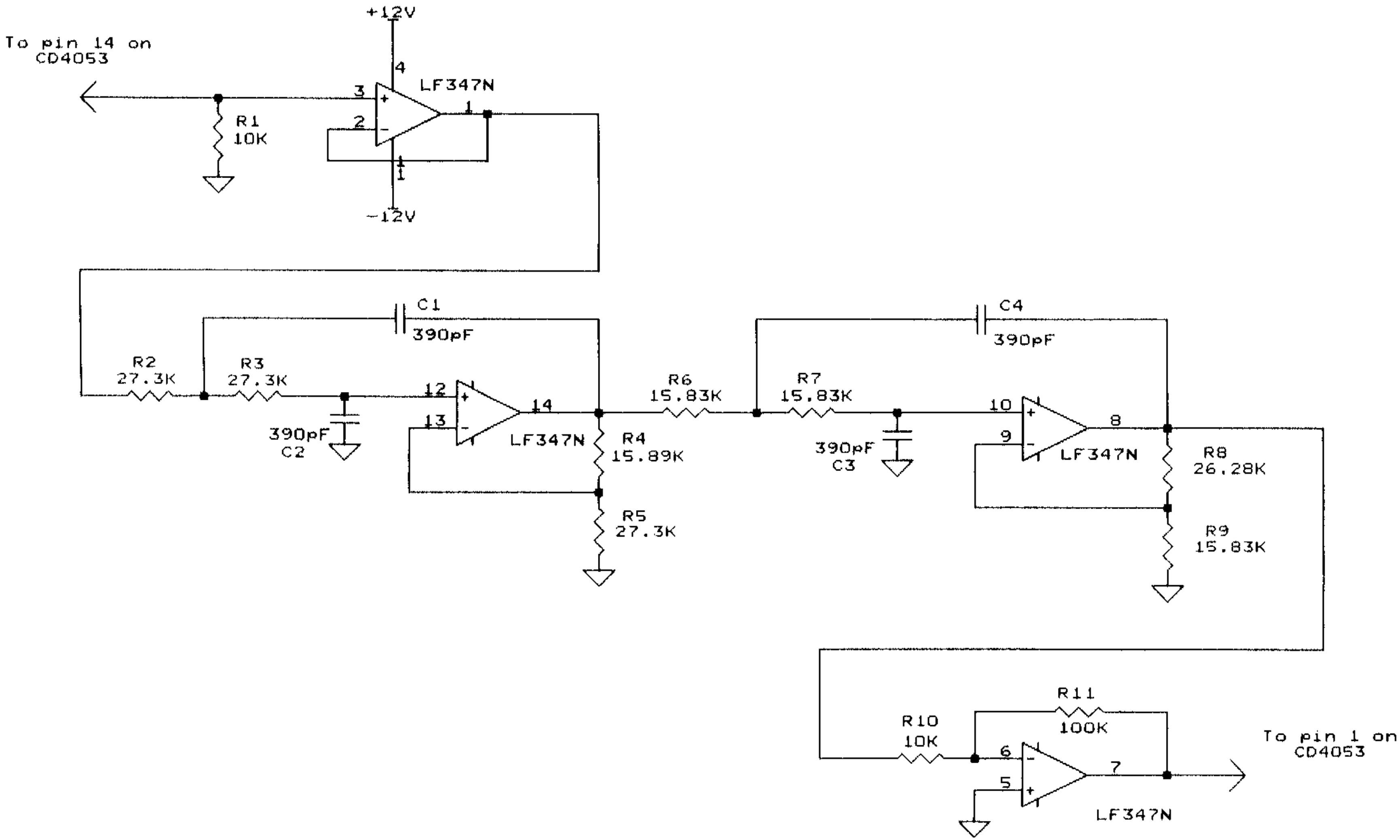


Figure 11: Circuit diagram of Low-pass Filter

The specified rolloff is > 18 dB/oct, therefore a 3-pole minimum filter is required (1-pole => 6 dB/oct). Each VCVS section provides a 2-pole (12 dB/oct) rolloff, therefore two sections are used providing a rolloff of about 24 dB/oct. When 2-pole sections are cascaded together, the individual filter sections are not identical. Each section represents a quadratic polynomial factor of the n_{th} order polynomial describing the whole filter.

The design was based on the method and table of coefficients (f_n and K) used in Horowitz & Hill, page 274. A Chebyshev filter was chosen because of its steep skirt characteristics from passband to stopband. The ripple was 0.5 dB.

The component values for section 1 were chosen as follows:
Choose C1 = C2 = 390 pF.
R in the range 10K to 100K.
The RC products for each section are different and must be scaled by the the normalizing factor f_n : $RC = 1/2\pi f_n f_c$, where f_c = 20 kHz.
Therefore, R = 27.3K
R = R2 = R3 = R5 and R4 = (K - 1)R = 15.89K
For section 2 the values were:
R = 15.83K = R6 = R7 = R9 and R8 = (K - 1)R = 26.28K

The frequency response of the 20 kHz low-pass filter was obtained from a swept sine wave test in the region from 20 Hz to 20 kHz. The data is shown in tabular form in Appendix 2 and the graph is shown on page 40.

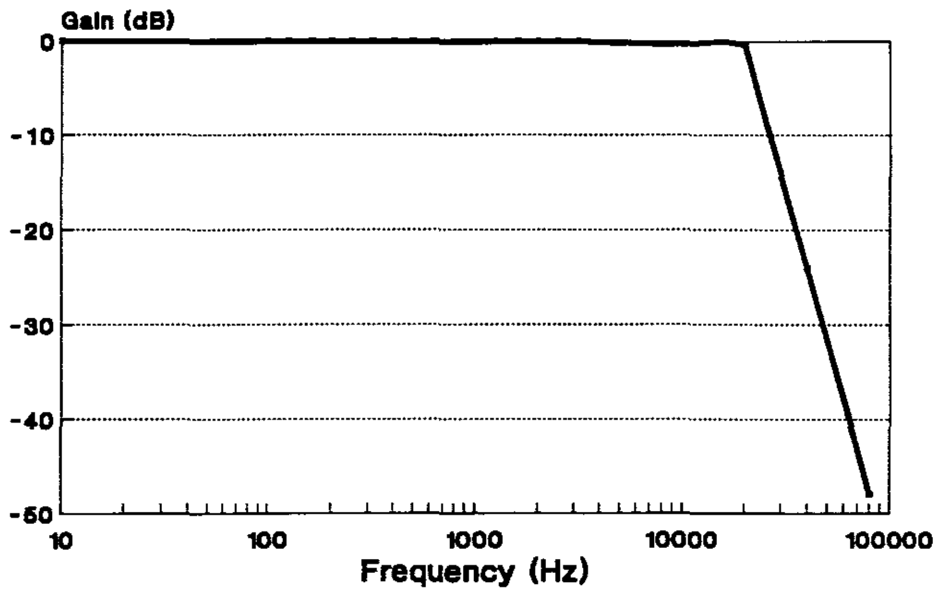


Figure 12: Frequency response of low-pass filter

It can be seen that the frequency characteristic is flat to within ±0.5 dB up to 20 kHz. The attenuation above 20 kHz is greater than 18 dB/oct and conforms to the IEC Recommendation 225 specifications.

5.2 The Bandpass Filter

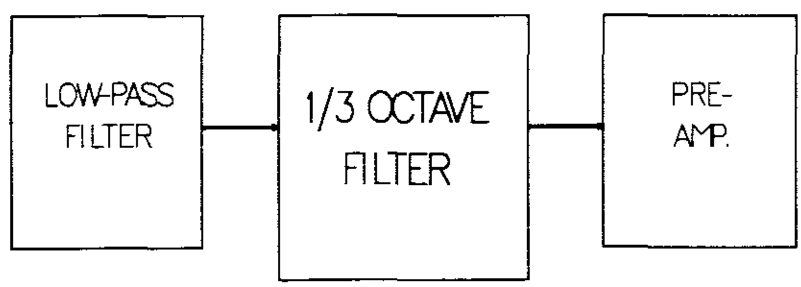


Figure 13: Block Diagram of Bandpass Filter

Filter Specifications

The noise may optionally be filtered through an octave or third octave bandpass filter network. This limits the noise bandwidth to within a selected frequency band and hence reduces the power bandwidth to the loudspeaker.

The filter responses and specifications are referenced to the IEC 225 standard for octave and fractional octave band filters. The aim of these requirements is to reduce any difference in equivalent measurements taken with different filters. The filters designed for this project conform to the Class 2 type filter.
A reference frequency of 1000 Hz is used.
The nominal frequencies are the preferred frequencies from 100 Hz to 8000 Hz in third octave steps as listed in Table

1, IEC 225, in the Appendix 1.
The reference attenuation in the passband is zero decibels for all center frequencies.
The ripple in the passband region must not vary by more than 0.5 dB.
The attenuation tolerance limits set out in IEC 225 are listed in Table 2 in Appendix 3. A roll-off of 46 dB/oct is specified for the Class 2 type filter.

__Filter Type Selection__

Three options are required for designing the bandpass filters.

1. Digital filtering
2. Analog filtering
3. Sampled-data filter

Digital filtering offers sharper roll-off (> 48dB/oct) at the passband edges for accurate third octave bandwidth analysis. Small passband and stopband ripple (0.001dB) is easily obtainable with these steep skirts using **finite impulse response** (FIR) filters. The filters have better stability over time, ie. no trimming of components required to maintain correct values and response curves. There is less drift and phase distortion over time which produces an accurate output.

Digital filtering offers more flexibility for prototype and future changes which can be implemented by the software only. It offers easy interfacing to the I/O port of the computer for automation of filter response and centre frequency changes. Software simulation of the filter response can reflect exact performance. However, FIR coefficient accuracy requires at least 12 to 16 bits. The basic filter components are shown in figure 14 on page 43.

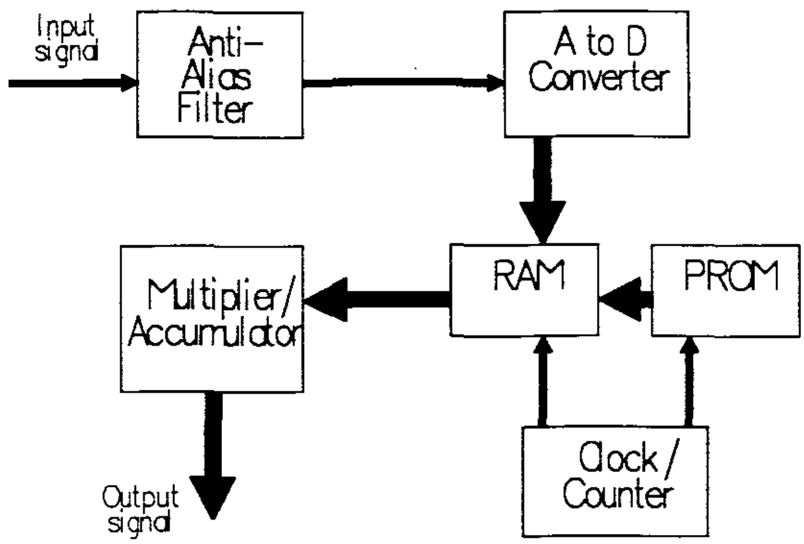


Figure 14: Digital filter configuration

The input signal is fed through a 25 kHz low-pass anti-alias filter to prevent sampling of high frequency noise components outside the required bandwidth. The filter requires good attenuation of the noise frequencies and a roll-off of about 24 dB/oct.

The A/D Converter samples and converts the input signal to digital form. The sampling rate must be about three times the highest frequency component of interest.

The converted samples enter the RAM for temporary storage. The size of RAM equals the number of filter taps multiplied by 12 bits/word. For a 4^{th} order filter, four by 12-bit locations are required.

The PROM stores the required filter coefficients and the clock/counter steps through the RAM and PROM presenting the coefficients and input samples to the multiplier / accumulator (MAC). The filter output is taken from the output of the MAC.

The main disadvantages of digital filtering are the high cost of system components and increased complexity of the final design. A complete study into digital filter design and techniques is required for correct configuration of sections. The accuracy and steep roll-off capabilities are an overkill for the project's application.

Analog Filters

Analog filters are implemented using op-amps and resistor/capacitor networks. Because twenty third-octave filters are required, too many component changes are necessary to obtain the correct centre frequencies and bandwidths for each third octave band analysis.
The centre frequency of the filter module must be stepped automatically under the control of the software program. This would entail switching about 250 resistor values with a multiplexer network using the common biquad type filter which would introduce noise to the system and not be a very practical solution.

Sampled-Data Filters

Introduction

One method of implementing the integrators that are needed in the state variable and biquad analog filters, is to use the switched capacitor network (SCN) technique. It is easy to implement capacitors, switches and op-amps, but difficult to construct resistors with the required accuracy. The recognition that a resistor could be approximated with two MOS switches and one capacitor was the key to solving this problem.
Two MOS analog switches are clocked from an externally applied square wave at some high frequency (typically 100 times faster than the analog signal of interest). The square wave applied to the second switch is fed through an inverter

so that the switches are closed on opposite halves of the square wave, as shown in figure 15 below:

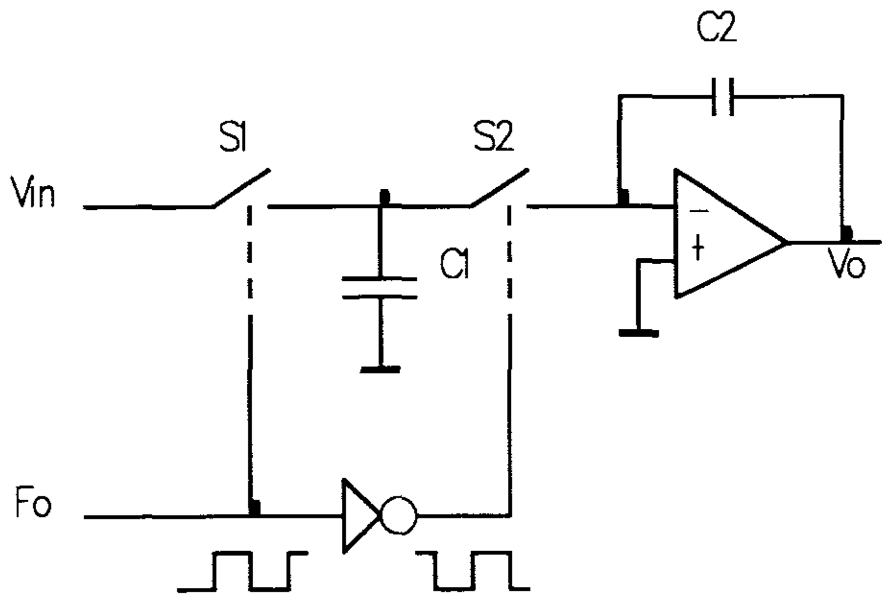


Figure 15: Switched Capacitor Integrator

When s1 is closed, s2 is open, C1 charges to Vin thereby holding the charge Q = C1 * Vin. On the other half cycle, s1 is open, s2 is closed, C1 discharges into the virtual ground, transferring its charge to C2. The voltage across C2 then changes by an amount **$\delta V = \delta Q/C2$ = Vin * C1/C2.**
The output voltage change during each cycle of the square wave is proportional to Vin (small amount), therefore the circuit is an integrator.
The current ≈ **$i(t) = \delta Q/\delta t$ = C1*(Vin-Vo)/Tc,**
where Tc = time for charge to transfer.

Therefore, the size of an equivalent resistor to give the same value of current is :

Rc = (Vin-Vo)/i = Tc/C1 = 1/(Fc*C1) ----> Eq.1

For this approximation to be valid, it is necessary for the switching frequency to be much larger than the signal frequency of interest.
The output voltage change during each cycle of the fast square wave is proportional to Vin, ie. the circuit forms an integrator.
If we replace the SCN with resistor R1, the transfer function is:

Vo/Vin = -1/(R1*C2*s)

The corresponding transfer function of the SCN is found by substituting R1 = Rc from Eq.1

We have **V2/V1 = - F_C*C1/(C2*s)**

We observe the ratio of C1/C2 which may be varied by either changing the capacitor values or the clock frequency (F_C).
Applying this principle to SCN filters, we look at a first order lowpass filter in figure 16 below:

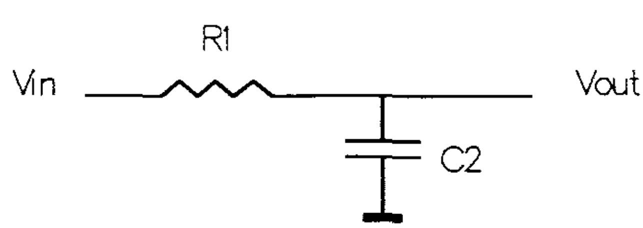


Figure 16: RC Low-Pass Filter

The transfer function is **T(s) = (1/R1*C2)/(s+1/R1*C2)**

The SCN equivalent is found by substituting **R1** = $1/F_C$*C1 from Eq.1 into this function :

$$T(s) = (F_C * C1/C2)/(s+F_C * C1/C2)$$

We see that the half-power frequency of the response is :

$$w = F_C * C1/C2$$

Therefore the filter's frequency can be tuned by either the clock frequency or the ratio of the two capacitors. Obviously, it is easier to vary the clock frequency than to obtain non-standard values of capacitors.

<u>Description of SCN Application</u>

The **MAXIM** company produces various types of filter systems in chip form. The **MF10** type filter requires changing approximately 80 resistors to step through each third octave frequency and presents the same problem as analog filters. The **MAX262** SCN type filter is a dual second-order, universal, microprocessor programmable active filter with parallel port interface for changing filter parameters under software control. Each chip contains two second-order filter sections which may be cascaded to form higher order filters. The only external requirements is a clock source, using an RC network to set the internal clock rate or an external clock source.

The **MAX262** uses a lower range of sampling (f_{clk}/f_o) ratios than the other **MAXIM** series filters to allow higher operating f_o frequencies and signal bandwidths. On-chip MOS switches and capacitors provide feedback to control each filter section's Q and centre frequency.
Internal capacitor ratios are primarily responsible for the accuracy of these parameters.
The filter's internal sample rate is actually one half the input clock rate due to an internal division by two to generate the sampling clock for both filter sections.

The ratio of the clock to filter centre frequency is kept large so that the ideal second-order state-variable response is maintained. Wider bandwidths are required for octave filtering, (70.7 percentage bandwidth), while third octave filtering requires a much narrower band (23.1 percentage bandwidth).

By changing the input clock frequency of the filter chip, the centre frequency of the bandpass filter is changed, thereby stepping through each third/one octave band. The f_{clk}/f_o frequency ratio, Q value and mode selection codes are stored in an internal program memory. Data is stored in the selected address on the rising edge of the WR strobe pulse. When all zeroes are written to the Q addresses of the first filter, the filter enters a shutdown/standby mode.

There are several ways in which the summing amplifier and integrators in each filter section can be configured. These modes use no external components and are selected by writing to input registers **M0** and **M1**.
Typical wideband noise is $0.5mV_{pp}$ from DC to 100kHz and is *independant of clock frequency. The output waveform of the* filter appears as a sampled signal with "staircasing" occuring at the internal sampling rate which is removed by adding a low-pass filter at the output.

MAXIM *provides a design program to simplify the filter* design process and generate programming coefficients for the chip. This program is readily available from electronic component stockists. By typing in the desired parameters such as centre frequency, pass bandwidth, stop bandwidth, minimum passband ripple, stopband attenuation, an 8th order bandpass filter was found to be required. The ripple chosen was 0.5dB using a Butterworth response filter. An example of this design procedure is given in Appendix 4 using a center frequency of 1000 Hz.

All the codes used for octave and third octave bandpass filters is supplied in the Turbo Pascal listing of unit **'FILTER'**. Tuning of the filters by adjusting these codes is described later in this chapter.

The Clock Generator

Various methods of applying the clock generator pulses were investigated.
The simplest configuration of the clock circuit is to use the internal clock circuitry of the filter chip and provide the correct resistor and capacitor values (RC oscillator network). A single capacitor of 10nF was used and the resistor values changed to give the required clock frequency for obtaining the centre frequencies of each octave and third octave band. A digital potentiometer was convenient to use, because of the ease of changing its resistance via software control.

Two 10 kΩ digital potentiometers connected in parallel provided the correct resistance values. Each 10 kΩ pot has 256 possible wiper positions providing approximately 39 ohm step changes. In parallel, the minimum value is the wiper's resistance, about 150 ohms for pot one and about 350 ohms for pot two. The step change decreases to a few ohms by changing one pot only. A serial port is provided for setting and reading the pot value via an 8-bit register that controls which tap point is connected to the wiper output.

However, when the potentiometer was operated in circuit, the internal clock frequency was found to be inaccurate and unstable. This was probably due to the interaction of the voltages present on the potentiometer chip pins and the filter chip pins. A dc offset voltage of 1.5 volts internally generated by the filter chip clock input possibly caused the resistance value of the pot to change slightly.

The clock frequency is critically dependant on the combination and stability of the resistance and capacitance values, therefore any slight change in resistance will cause a clock frequency error. This is unacceptable, because the filter centre frequency is dependant on the clock frequency.

The alternative to the digital pot is to use individual resistors and multiplex them into circuit when required. The major problem here is to find exact values of resistance that are commercially available. The same problem exists as was found in the analog filter section.

Another more complicated method of clock generation (the one used in the project) is to provide an external clock by *means of a voltage-controlled oscillator (VCO) chip* controlled by a digital-to-analog converter (DAC). Refer to figure 17 on page 51.

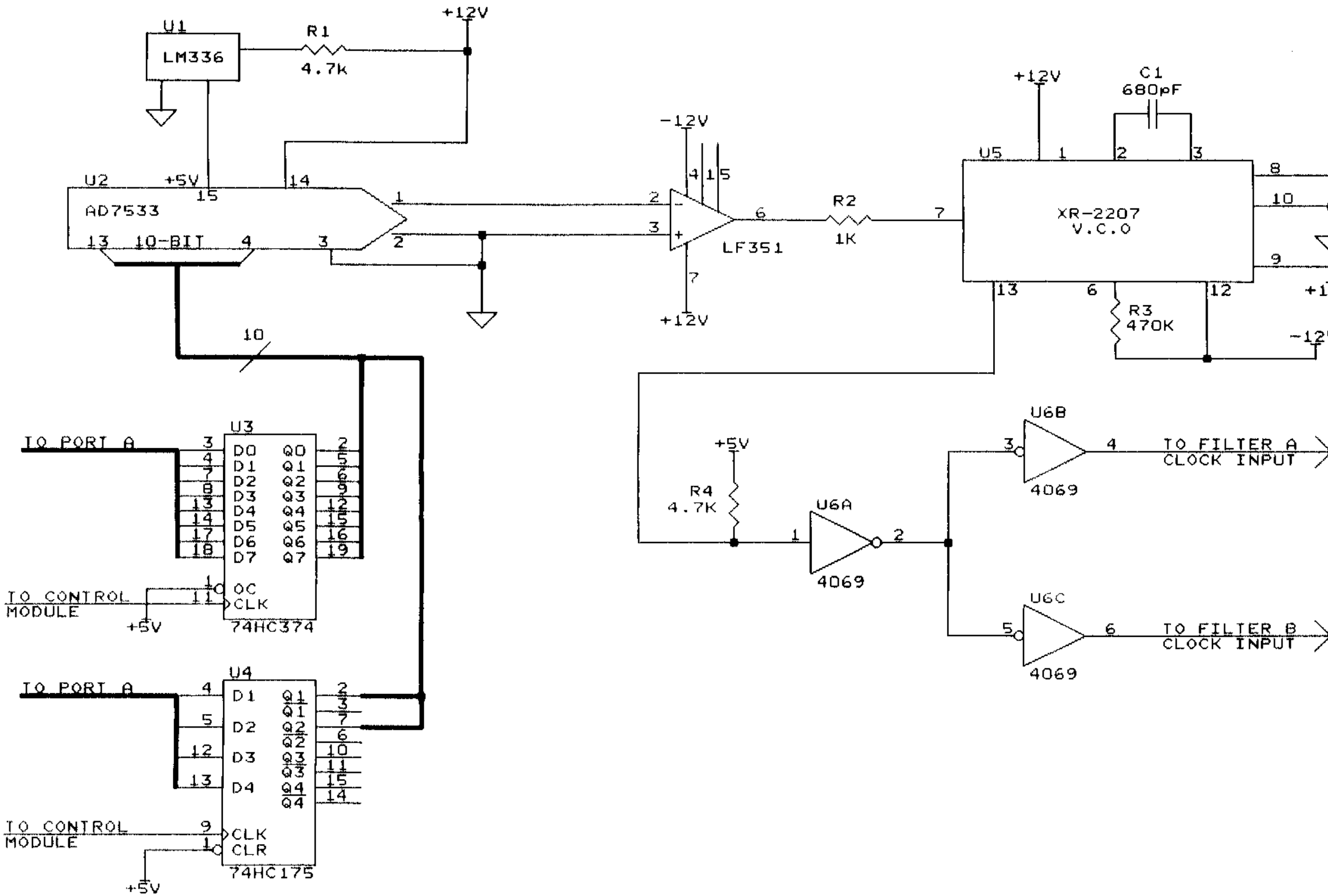


Figure 17: Circuit diagram of Filter Clock Generator

The Digital to Analog Converter

The goal of the DAC is to convert a binary number to a voltage or current proportional to the value of the digital input. The technique used is provided by the R-2R ladder network which generates binary scaled currents. These are fed into an op-amplifier (LF351) which converts the currents into an output voltage. Although a voltage is most convenient, the op-amp tends to be the slowest part of the converter circuit. However, for our application, speed is not a major concern.

The output voltage is a pecentage of an externally applied reference voltage, depending on the number of bits used for the binary input value. For a 10 bit DAC, such as the **AD 7533** (U2), the resolution is 1 part in 1024. Therefore, with a 5 volt reference voltage such as the LM336 (U1), the maximum output voltage of the DAC is 1023/1024 * 5 = 4.995 volt. The minimum output will be 1/1024 * 5 = 4.8825 mV.

The required input control voltages to the VCO to generate the specified clock frequencies of the filter chips are found by the equation :

$$V_C = R2 * V- * \frac{(1 - f*R3*C1)}{R3}$$

The digital equivalent 10-bit code is then found by inserting the voltages into a short software program that calculates the code. This code is applied to the DAC by data latches (U3, U4) when running the main software package and generates the required control voltage to the VCO.

The Voltage Controlled Oscillator

The **XR-2207** VCO chip was chosen because of its good linearity, stability and wide frequency tuning range

variable over a 1000:1 with an external control voltage. The maximum output clock frequency is 1 MHz and the linearity is specified at 1 percent.
The minimum frequency required was calculated at 8500 Hz and the maximum at 720 500 Hz, requiring approximately 85:1 tuning range. This range of frequency values was found using the filter design software that was available with the filter chips. Two chips were used to obtain the 8^{th} order bandpass, each chip having two programmable 2nd order sections. Therefore, four resonant frequencies were required to be programmed using the available f_{clk}/f_o[7] range of 40 to 140. The clock frequency was chosen to fall within these limits.

The frequency of operation is controlled by varying the total timing current drawn from the activated timing pins. This timing current can be modulated by applying a negative control voltage ($-V_c$) to an activated timing pin (pin 7) through a series resistor (R_2). The frequency of operation is proportional to the control voltage and determined using Eq.1 and the circuit in figure 17.

$$f = \frac{1}{R3*C1} * \left[1 - \frac{V_c*R3}{R2*V-}\right] \quad \text{Hz} \quad \text{------>} \quad \text{Eq.1}$$

The minimum frequency can be seen to equal 1/R3*C1 when V_c equals zero.
The output of the VCO drives a CD4069 (U6) buffer network that allows the oscillator to supply a larger load current, because of its limited drive capacity. Both filter modules are driven by this network so that the same clock frequency is available to both chips.

7 See MAX262 specification sheet

Method of Smoothing Filter Output

As mentioned in the introducion to sampled-data filters, the output waveform needs to be low-pass filtered. This is done at the output of each filter chip to reduce the clock related feedthrough, which is about 8 mV. A single pole RC filter successfully removes the staircasing effect of the output waveform. Refer to figures 18 & 19 on pages 55 & 56.

An op-amplifier with parallel RC feedback was used as the filter network. Three op-amplifiers were used, one on the input, one inbetween the filter chips and one on the output. An LF347N (U1) quad op-amplifier was used. These served the dual purpose of filtering and controlling the amplitude of the input signal to the next stage. The feedback resistance was adjusted for the correct signal amplitude to prevent clipping of the output signal. A CD4053 (U2) multiplexer switched in different values of feedback resistance for selecting octave or third octave filtering. Signal diodes (D1, D2, D3, D4) were used at the filter chip inputs to protect the chips from overload and latch-up. This was necessary because the op-amplifiers operated from ±12V supplies while the filter chips used ±5V supplies as per specification.

To be effective, the filter must start cutting off at about 3 times the center frequency of interest. The feedback resistance could not be changed, so the capacitance had to be varied to obtain the correct cutoffs. The capacitors were calculated using the formula:

$$C = 1/(2\pi * R * f) \qquad \text{where } f = 3 * F_C$$

The output signals of the each bandpass filter were tested in groups for minimum noise floor levels (± 1 mV) to enable the same RC feedback network to be used for more than one center frequency.

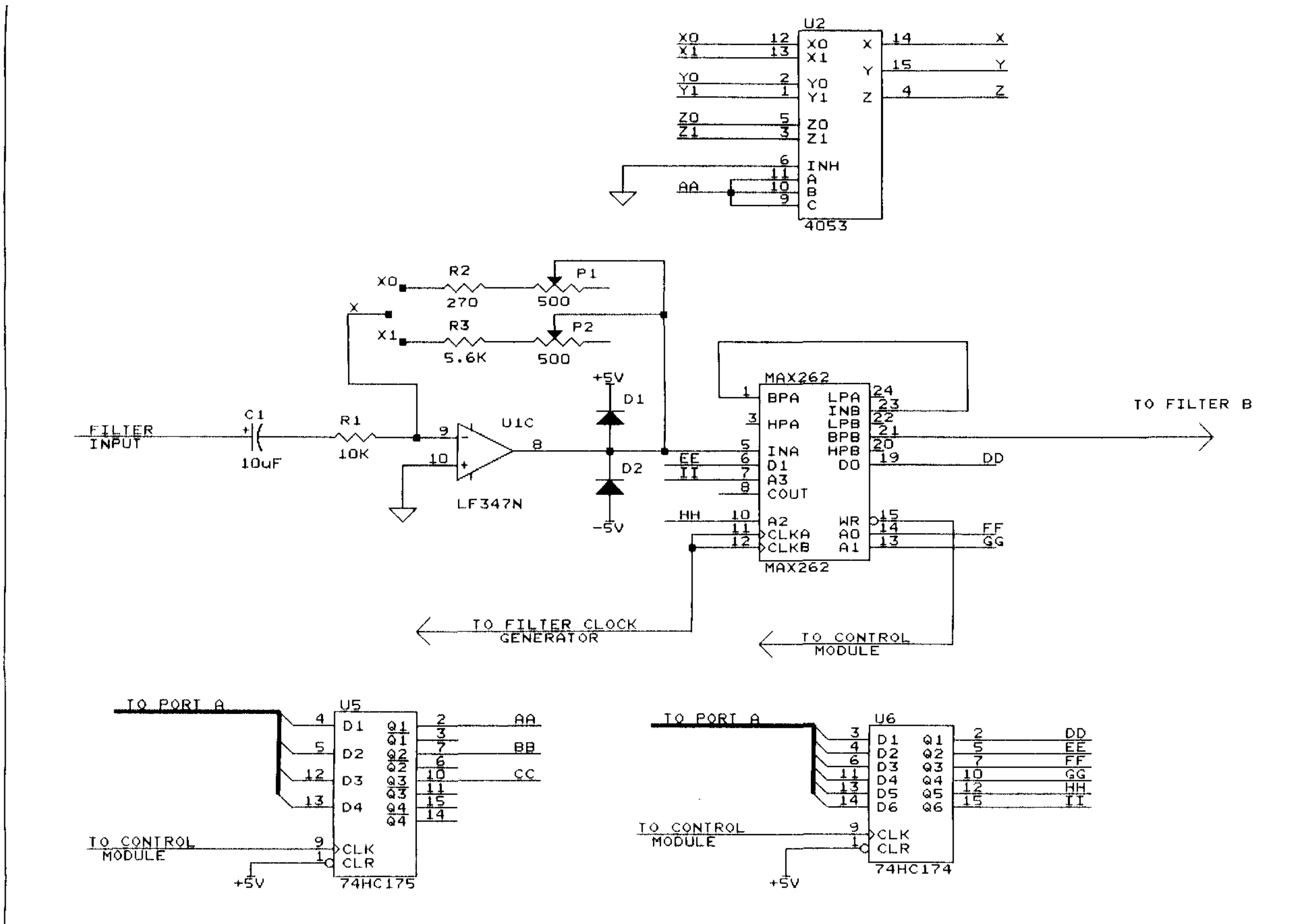


Figure 18: Circuit diagram of Bandpass Filter

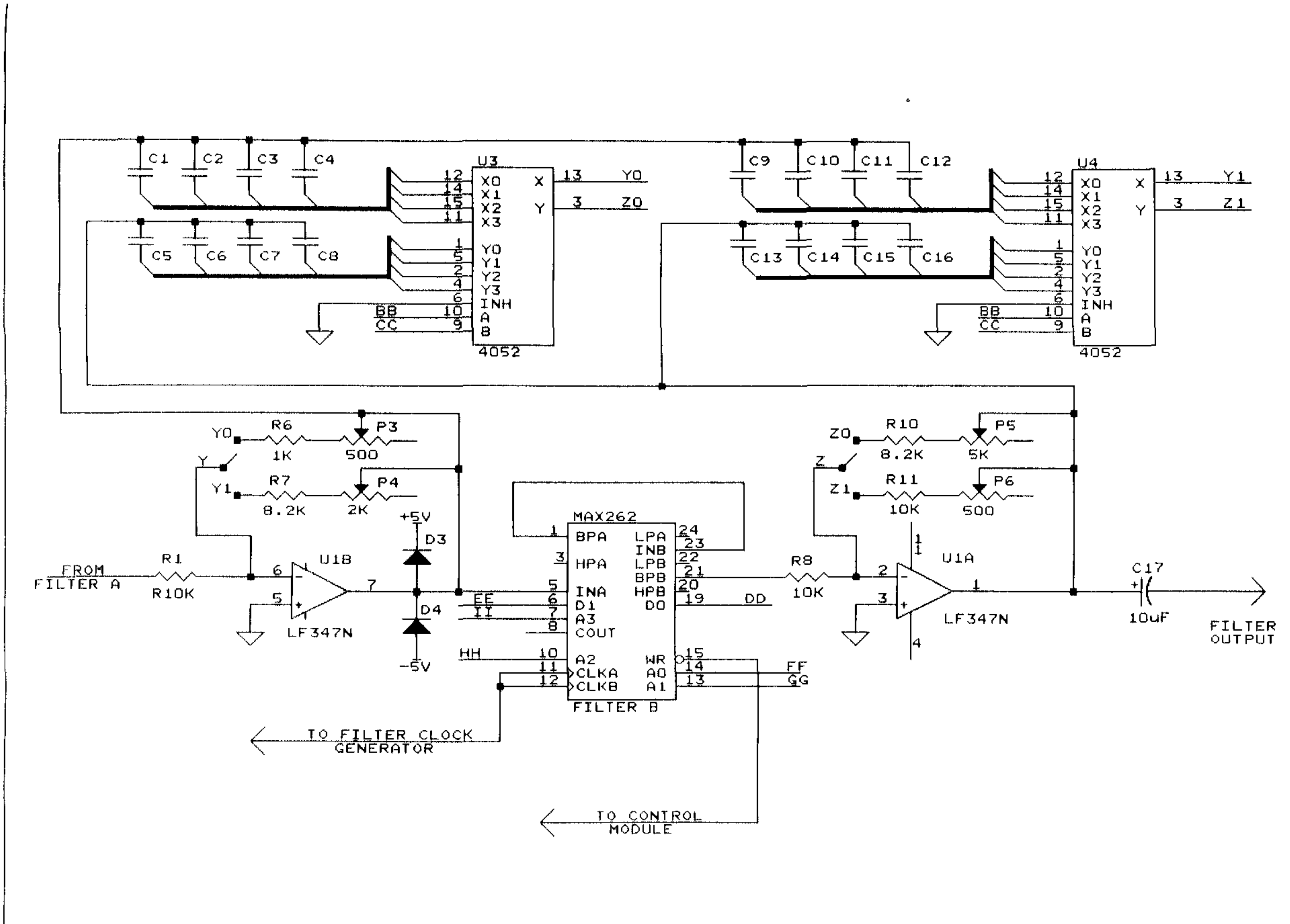


Figure 19: Circuit diagram of Bandpass Filter

Four groups were found to be adequate allowing the noise floor to be kept below 1 mV for each center frequency. The various capacitors were switched in and out of the feedback circuits of the op-amplifiers using two CD4052 (U3, U4) multiplexers.
The various codes were worked out for selecting the various bandpass center frequencies. These codes were applied to the circuit using data latches (U5, U6) when operating the main software package.

Tuning the Filters

After calculating all the necessary programming codes, for the filter chips using the supplied software program, the filters were tested for accurate performance. These codes appear in the unit **'FILTER'**. A listing of the procedures and codes for a 1 kHz filter is shown in Appendix 4.
The H.P 3561A Dynamic Spectrum Analyser formed an important part of the tuning process, allowing close inspection of the passband cutoff limits, attenuation characteristics and shape factor of the filter. An oscilloscope was also useful for checking the low-pass filtered output of the filter for staircasing effects and clipping.

When operating correctly, the filter displayed a pure sinusoid at the output (C17) when a sine wave generator signal was applied to the input (C1).
The attenuation characteristic of each filter was measured as described in the IEC 225 Standard: A signal from the H.P 3325B Frequency Synthesizer was applied to the filter input. The input voltage V1 and the output voltage V2 were measured using a Fluke 8920A True RMS Voltmeters at the specified frequencies throughout the frequency range to demonstrate that the filters comply with the specifications of the standard. The specified test frequencies are those calculated from Table 2 in the Standard.

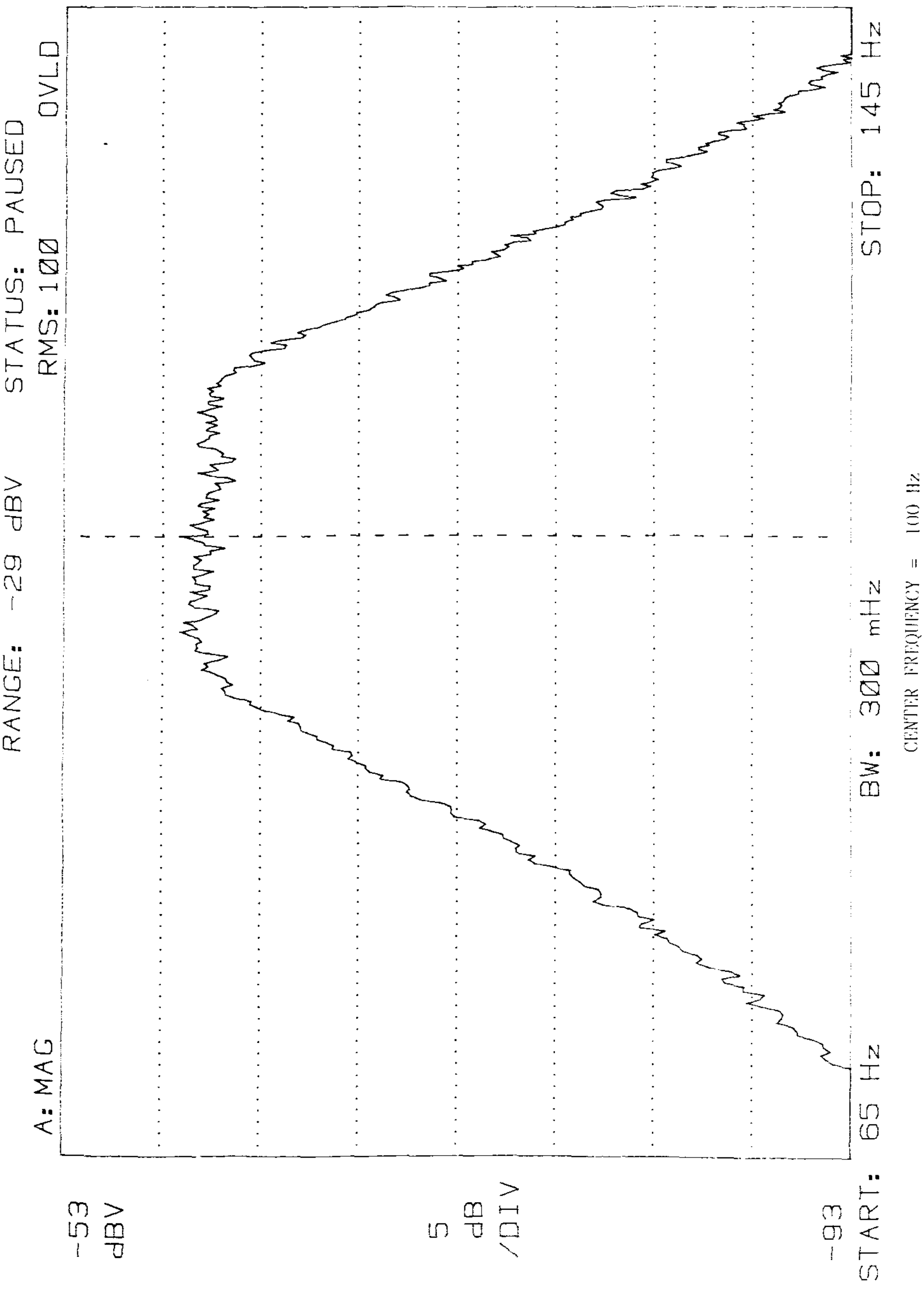
If the filters did not comply with the specifications, the programming codes were altered slightly to adjust for the correct shape factor.
The relative filter attenuation, $\delta A(f)$, at any frequency is given by :

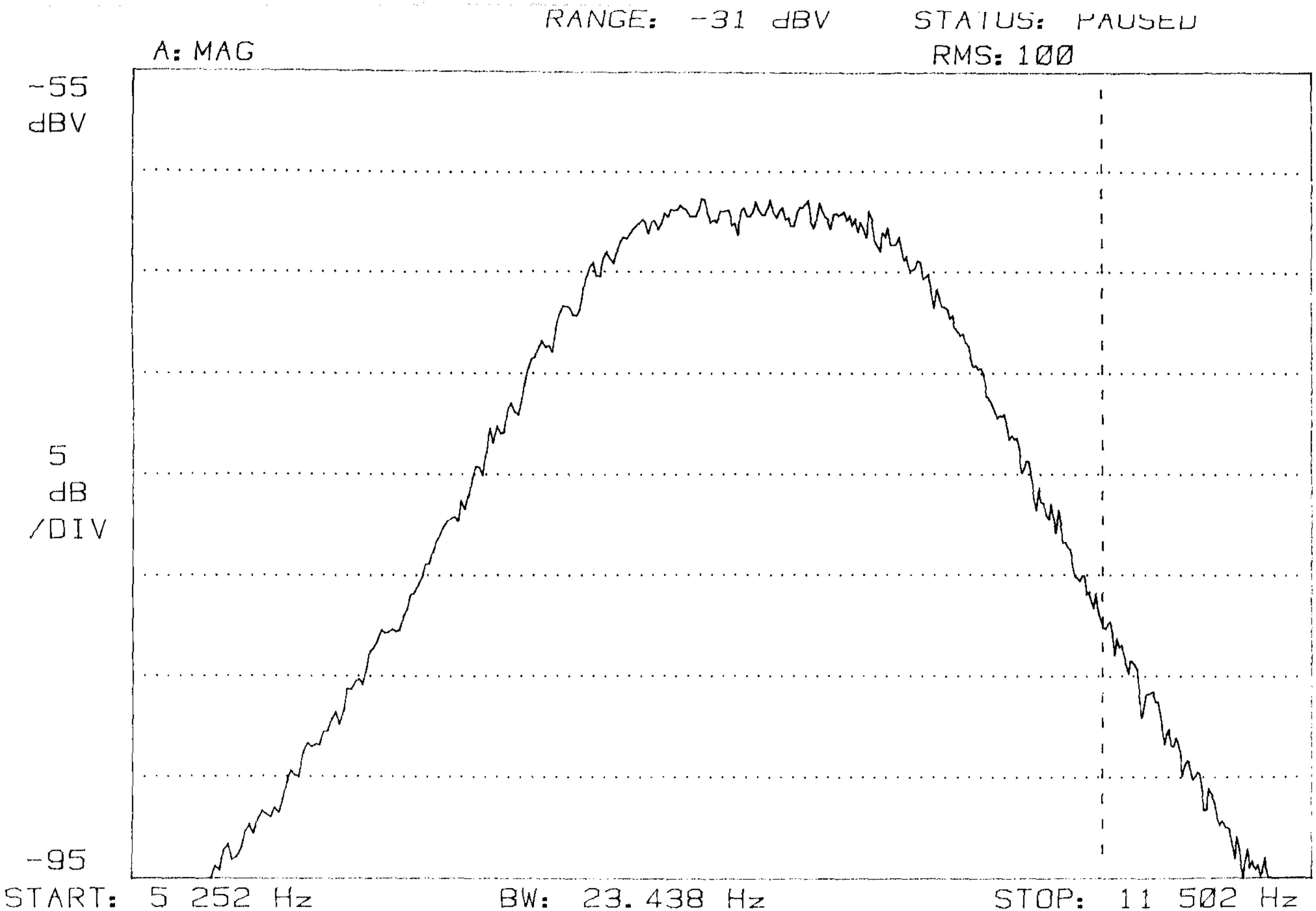
$$\delta A(f) = 20 * \log (V1/V2) - A_{ref} \quad (dB)$$

where A_{ref} = 0 dB for this design

The filter responses, to a random noise input signal, at the lowest (100 Hz) and highest (8000 Hz) center frequencies printed from the H.P 3561A Dynamic Spectrum Analyser were chosen as a representation of the overall characteristic of each filter. These responses are shown in figures 20 & 21 on pages 59 & 60.

Figure 20: Bandpass Filter response at 100 Hz





CENTER FREQUENCY = 8 000 Hz

Figure 21: Bandpass Filter response at 8 000 Hz

A comparison of the B.A.A filter response, the B&K Type 1625 filter response and the tolerance limits are shown in figure 22 below:

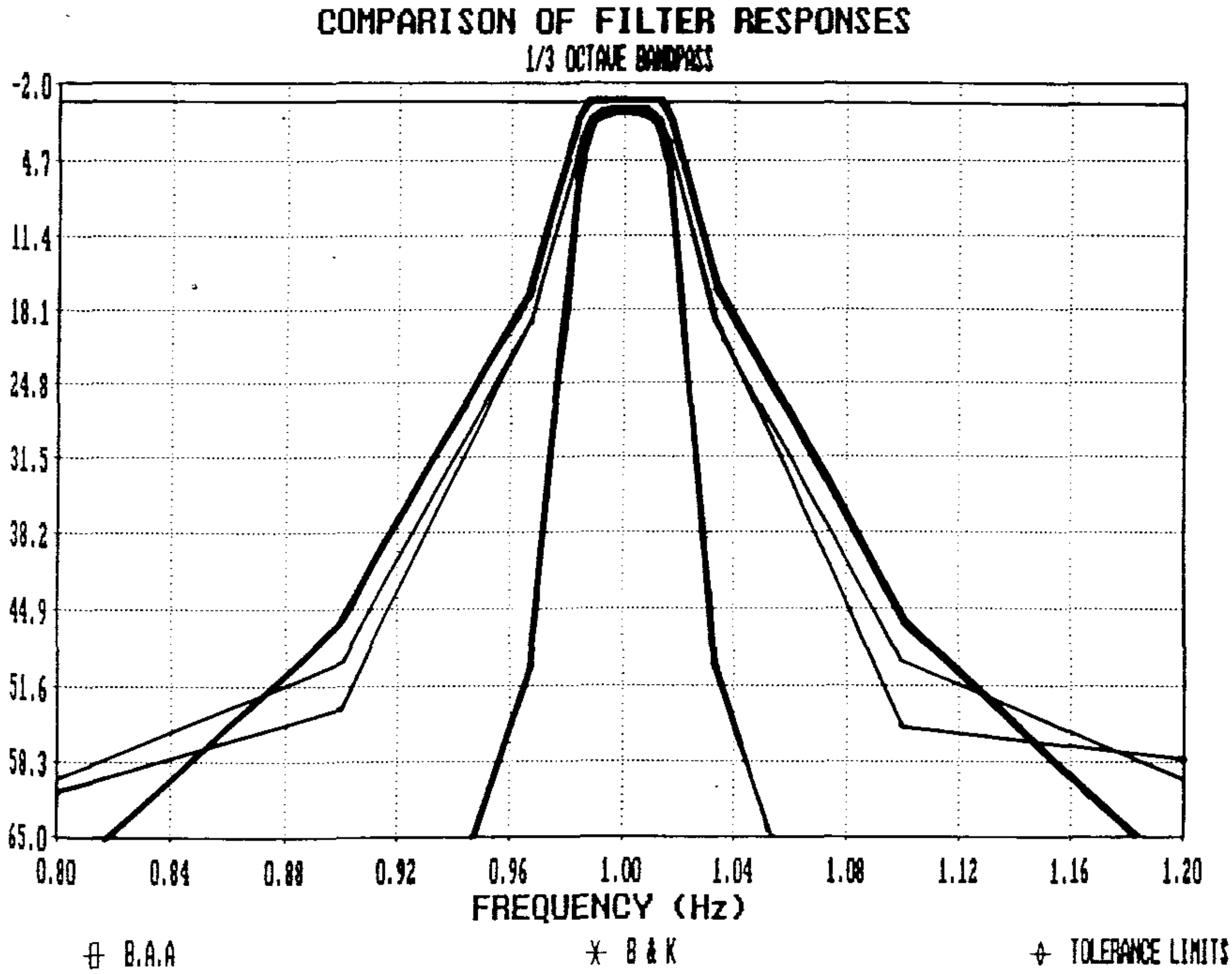


Figure 22: Comparison of Bandpass Filter Responses

A comparison test was performed using the **B&K Type 1625** Third Octave filter set, which complies with the Class 0 type filter attenuation requirements of the IEC 225 Standard. The same method for measuring the attenuation characteristics was used as desribed on page 57. The results of these tests at the lowest (100 Hz) and highest (8000 Hz) center frequencies are shown in Appendix 5.
The results show that the BAA filters comply fully with the IEC 225 specifications. It can further be seen that they exceed the performance of the B & K Type 1625 filters at certain points in the responses.
This proves the good accuracy and shape factor of the BAA filters, because the B & K Type 1625 filters conform to the IEC 225 Type 0 specification which is the most demanding one. The specification sheet appears in Appendix 8.

5.3 The Output Section

The output section forms part of the digital circuitry and module consisting of latches/ switches/ multiplexes, etc. Refer to figure 23 on page 63.

The noise signal was time-burst gated through a CD4053 multiplexer (U1 - pin 3,4,5) which was controlled by the software package. The other in/outputs of U1 were used for noise type and bandpass filter selection.
Two channels, A and B, provide a white/pink noise output level of about 2 V_{rms}. One of, or both outputs were selected by multiplexer (U2 - pin 1,2,15 and pin 12,13,14). The outputs are buffered and fed to a power amplifier and loudspeaker. The output impedance (R1, R2) was chosen to be 100 Ω to reduce the output signal distortion and feedback instabilities caused by the capacitance of long connecting cables. The dual channel output facility enables fully automated sound measurements to be carried out where two loudspeakers are available.

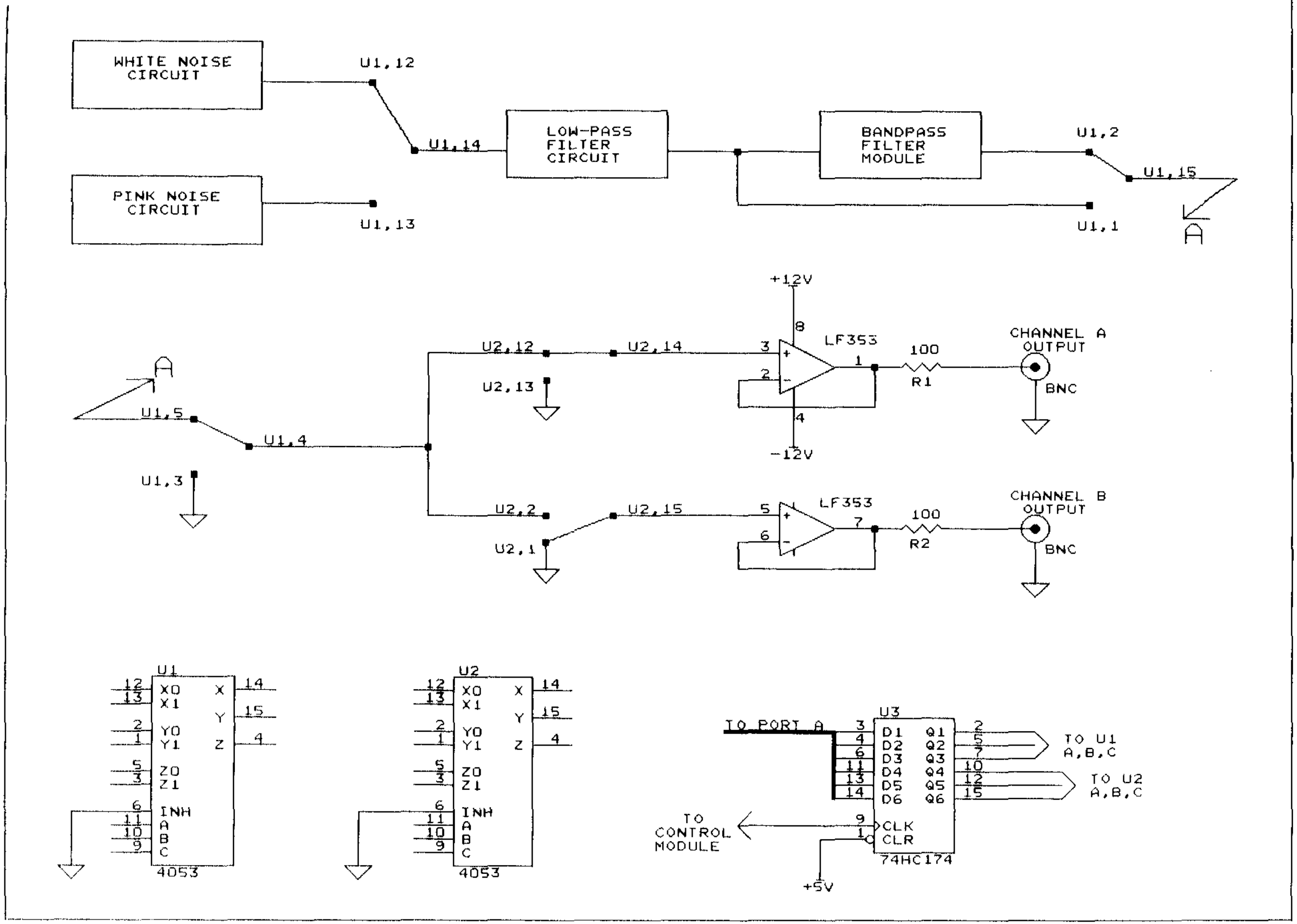


Figure 23: Circuit diagram of Noise Selection Control

The burst for transmission loss measurements is 5 seconds conforming to ISO 140 specifications. For reverberation measurements, a burst of 5 seconds is used, however, a longer burst should be used for auditoria with large volumes to enable the SPL to build up to a steady-state level before a measurement is taken.

This concludes the noise generating section consisting of the white/pink noise module, the low-pass filter module and the bandpass filter module.

THE SOUND MEASURING SECTION

5.4 The Input Amplifiers/Attenuators

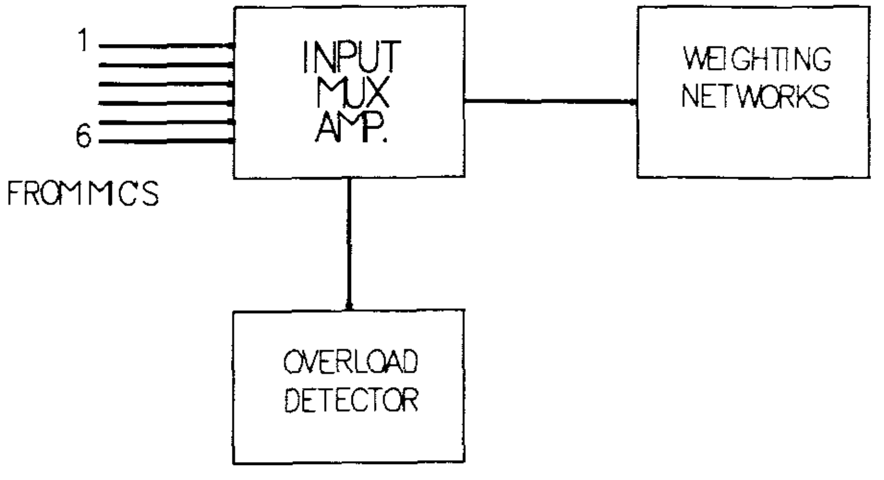


Figure 24: Block diagram of input section

In the measuring section shown in figure 24, the input amplifiers serve as level ranging circuits. At this point the signal is adjusted to the proper level for further processing. An overload detector warns of overloading signals at the instrument inputs so that the operator may then select the next range. The internationally standardized weighting networks, A and C, may be selected to obtain a subjective indication of the loudness of measured sounds. The third-octave filter bank is identical to the one found in the noise generator section. During a measurement, the filters in the two sections are swept synchronously.
The RMS detector determines the true RMS value of the signal

over a 60 dB dynamic range. The sample/hold amplifier operates in conjunction with the A/D converter and freezes the signal during a conversion process. The A/D converter is 12-bit and computes the digital equivalent of the analog input signal. It then sends the resulting codes to the PC for software processing. The software package computes and ouputs the various building acoustics parameters based on the measurements performed. The sofware controls the measurement procedures by selecting the proper input/ outputs, stepping the filters, computing the individual results and "quality control" of the measurements. If a failure occurs the software initiates a second attempt before an error code is stored.

For a fully automatic measuring facility, six input channels are required for transmission loss measurements as stated in ISO 140. The standard requires that 3 microphone positions are needed in the source room and three in the receiving room. Refer to figure 25 on page 67.

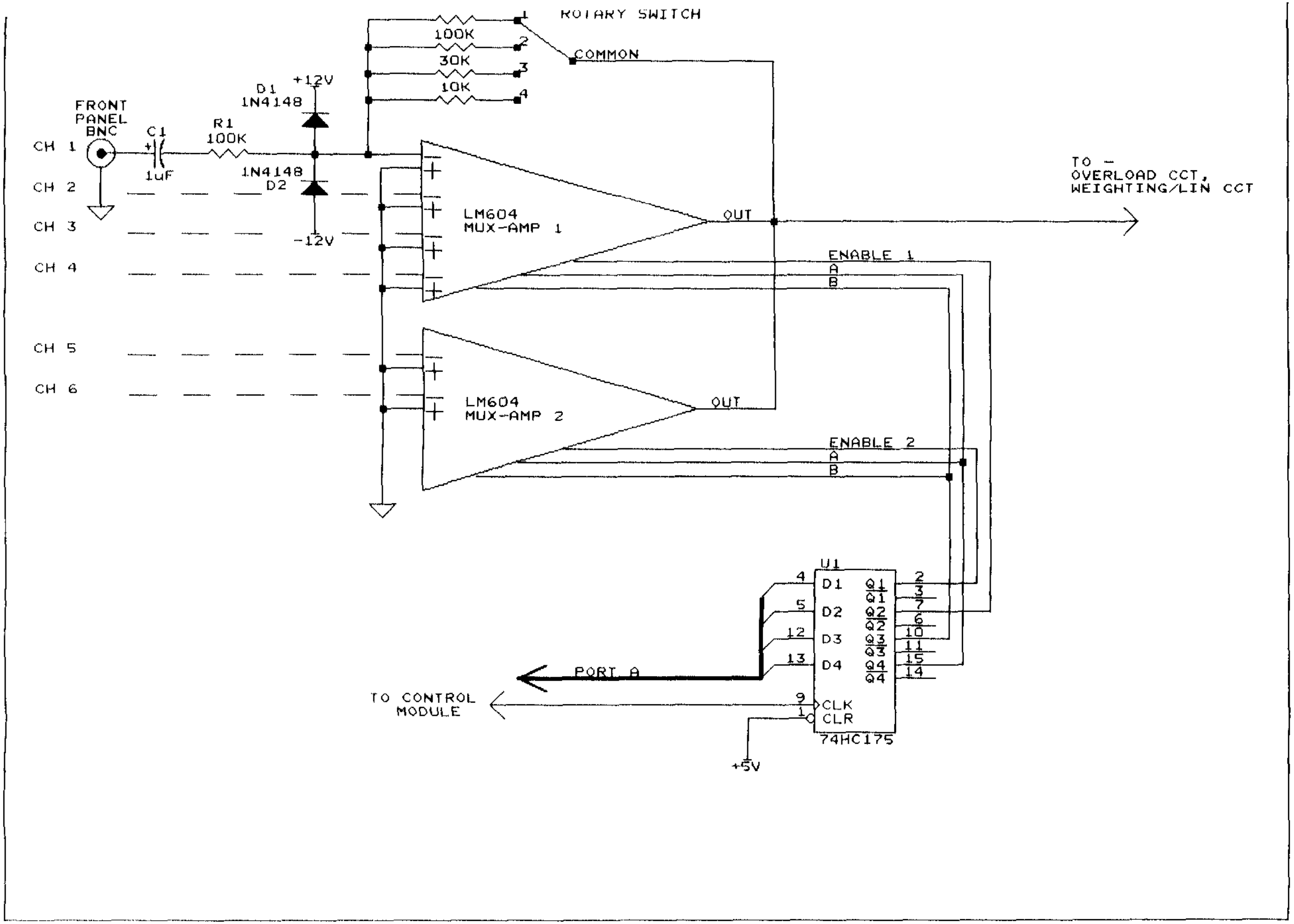


Figure 25: Circuit diagram of Input Amp./Channel selection

Two LM604 4 channel mux-amplifiers were used for the input section. Six channels were used with complete diode protection on all the inputs. The LM604 has a Bi-State output facility which allows more than one mux-amplifier to be connected together at their outputs. When the Bi-State is disabled, the chip becomes a high impedance load that can be driven by another output stage, such as the second mux-amplifier. The channel selection and Bi-State output are controlled by internal logic that interfaces directly to digital I/O lines or data latches (U1).

Each input channel is used as a single op-amplifier with its own feedback loop. The op-amplifiers are used in the inverting configuration with a choice of four range positions of gain/attenuation each, in steps of 10 dB. Each range position is designed to provide a maximum output voltage of 3 volts to the next stage.

The feedback loop of the selected channel determines the gain of the circuit, but also provides feedthrough paths from the inputs of the off channels to the output. This occurs because the feedback resistors and the mux-amplifier output impedance form a voltage divider.

This allows a portion of the off channel's input signal to appear at the output. The amount of signal that feeds through depends on the ratio of output impedance and feedback loop resistance as shown in figure 26 on page 69.

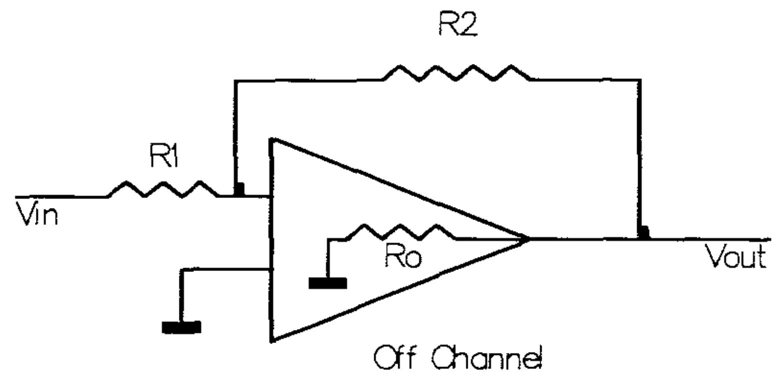


Figure 26: Off Channel Feedthrough

Output impedance varies according to the selected channel gain and the frequency of the feedthrough signal. The maximum gain used is 3, which gives an output impedance (Ro) of about 5 Ω up to 20 kHz. For gains of less than 3, the impedance is closer to 1 Ω. This data is found using the plot of Ro vs. Frequency under the Device Characteristics specification sheet. This value is substituted in the equation:

$$\mathbf{Vout = Vin * \left(\frac{Ro}{R1 + R2 + Ro}\right)}$$

The gain and attenuation factors used were 3, 1, 1/3 and 1/10. The input impedance, R1, was chosen as 100 kΩ for each channel and the maximum allowable input signal amplitude was 3.2 volt. Therefore, using 300 kΩ for a gain of 3, the off channel voltage feedthrough was about 40 microvolts. Using 10 kΩ for an attenuation of 1/10, the off channel voltage feedthrough was about 145 microvolts. These very low voltage leakage levels are insignificant compared to the ON channel signal levels, especially considering a noise floor of ± 1mV in the main circuitry of the instrument.

The various logic codes were worked out for selecting the correct ON channels when running the main software package. All resistors were one percent accuracy, metal film, for accurate amplification or attenuation of the input signal. The range control switching consists of a front panel rotary type switch for each channel.

5.5 The Overload Detector

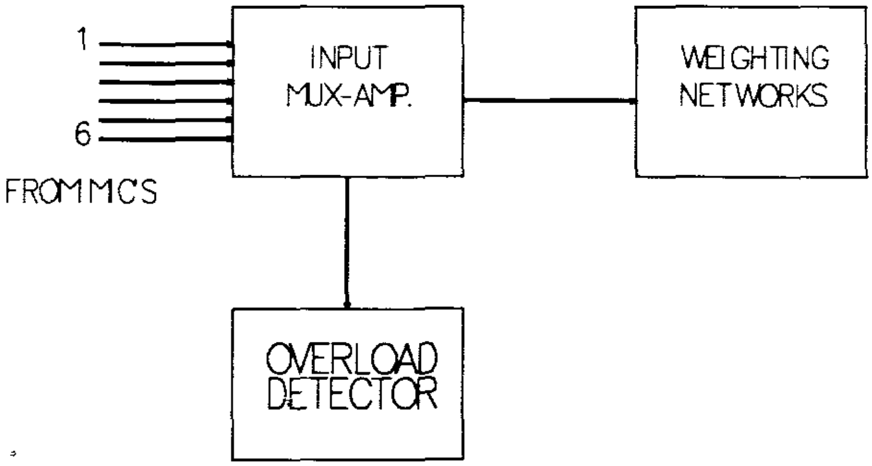


Figure 27: Block diagram of overload detector

On any measuring instrument, an input overload protection system is necessary to protect the input circuitry from damage. This allows a maximum input voltage equal to the rating of the overload diodes without damage. The input series resistor R1, (100kΩ), limits the input current to a harmless low value. Only huge input overloads would cause some form of front-end damage.

Two types of overload protection are provided for the B.A.A instrument:
The first type consists of diodes on all the input channel lines connected to the positive and negative rails. The diodes are connected right at the op-amplifier inverting input pins to reduce noise pickup due to the high input impedance at this point.

The second type is an error detection circuit that displays an overload condition on a front panel LED. Because only one channel is selected at a time by the logic, only one error detection circuit is required. The LED is activated when the input signal exceeds five percent of the maximum allowable input signal amplitude. This five percent margin mainly prevents the filter circuitry from clipping and entering latch-up. The maximum allowable signal is 3.2 volts, therefore overload occurs at about 3.36 volts.

Refer to figure 28 on page 73. The first op-amplifier (LF347N) acts as a full-wave negative rectifier using D1 and D2 for rectification. It has unity gain (R1 = R2) and the output is smoothed by C1 and R3 to a DC level. The second op-amplifier acts as the error comparator, comparing the input signal to a reference level formed by R4, D3 and R5 (voltage divider) on the non-invert pin.

The diode compensates for the diode voltage drop in the rectifier circuit. The output of the op-amplifier drives the front panel error LED via R6. This output signal may also be used to control the software and alert the operator to an overload condition.

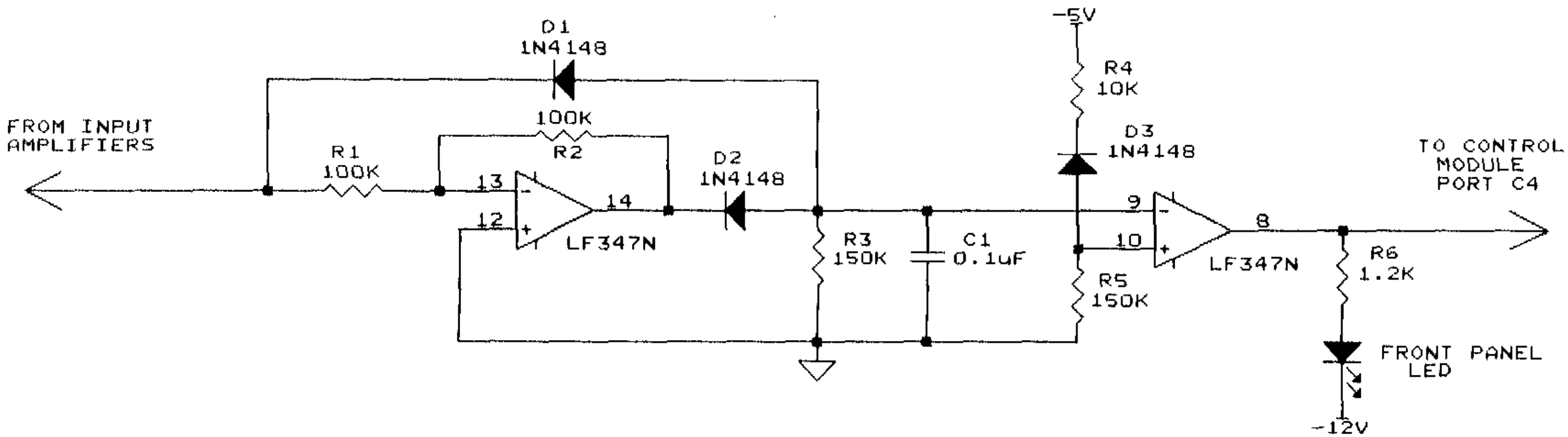


Figure 28: Circuit diagram of Overload Detector

5.6 The Weighting Networks

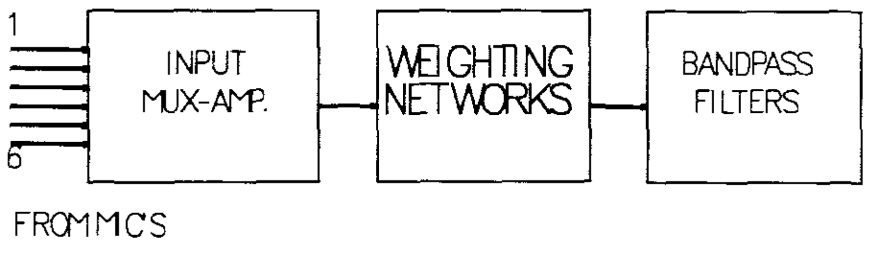


Figure 29: Block diagram of weighting networks

The object of using weighting networks is to perform measurements of a signal spectrum as it corresponds to the frequency response of the human ear, thereby obtaining a single value of the sound pressure level. Historically, it was felt necessary that the various networks should be switched in depending on the loudness of the noise. However, the A-weighting network is now specified for rating sounds irrespective of level and is no longer restricted to low level sounds.

The weighting networks used are defined as:
A-weighting : used for loudness levels below 55 phons.
C-weighting : used for loudness levels over 85 phons.

The frequency response of the relevant weighting networks is shown in figure 30 below. The circuit diagram of the networks is shown in fig 31 on page 76.

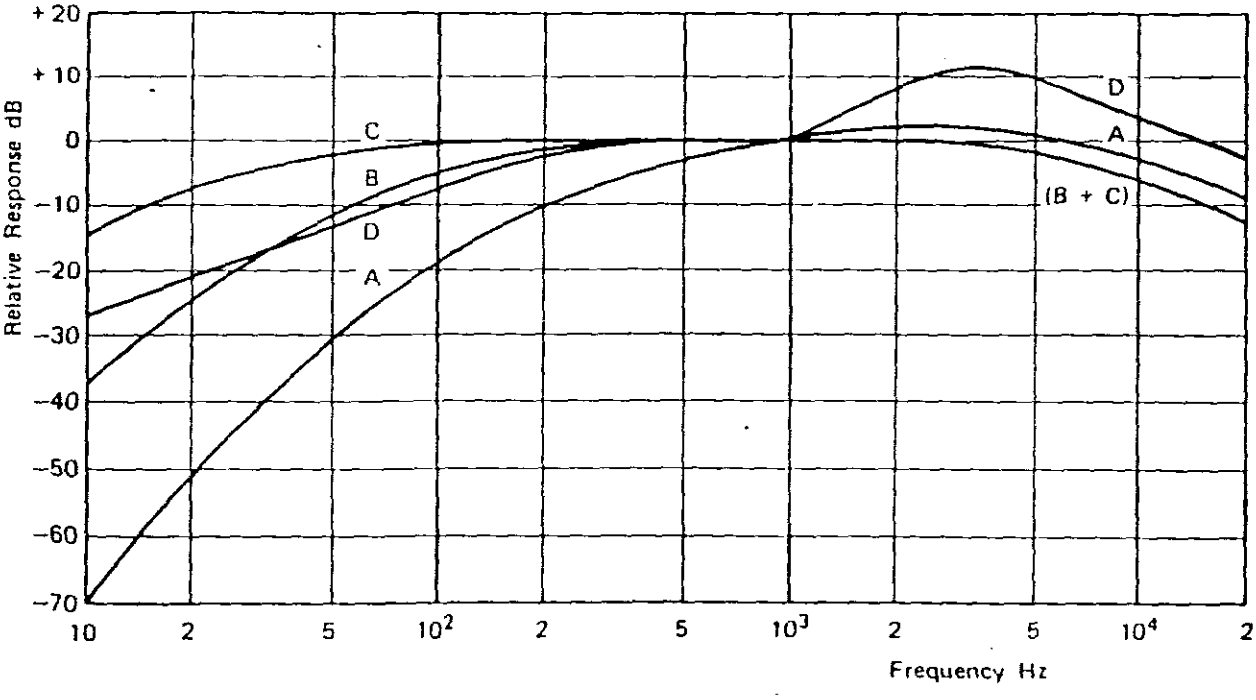


Figure 31: *Frequency response of the weighting networks*

The C-weighting network is achieved by using a Butterworth, V.C.V.S, 2-pole (12 dB/oct) high-pass filter at 20.6 Hz and the same type 2-pole (12 dB/oct) low-pass filter at 12.2 kHz. This is accomplished using an op-amplifier (3/4 of quad LF347N) and connecting the high-pass filter, made up from C1, C2, R1, R2, directly to the low-pass filter, made up from R3, R4, C3, C4. The linearity gain is adjusted using the 500 Ω potentiometer (R6) in the feedback circuit.

The A-weighting network is achieved by adding a 1-pole RC high-pass filter at 107.7 Hz and a 1-pole RC low-pass filter at 737.9 Hz to the C-weight filter. This is achieved using two RC filters (C5, R7 and C6, R8) buffered by op-amplifiers (1/4 and 3/4 of LF347N) configured as followers. The linearity gain is adjusted using the 10K potentiometer (R10) in the second op-amplifier.

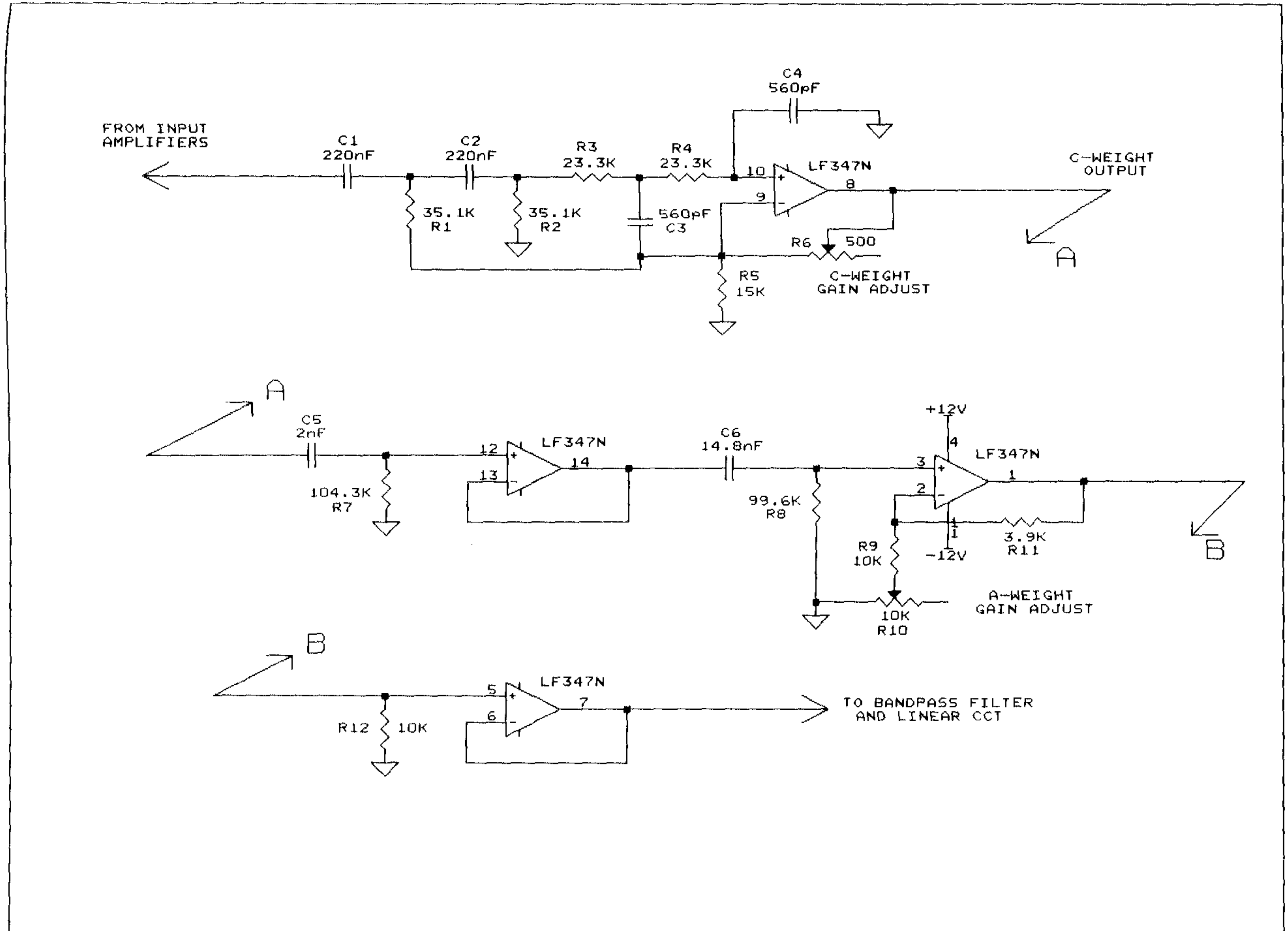


Figure 31: Circuit diagram of Weighting Networks

The linear output is obtained from the buffer op-amplifier (2/4 of LF347N) and passes the signal to the next stage. Both networks are designed in accordance with the tolerances in the IEC 651 standard, pages 17 to 19.

The response of the weighting networks designed in this project are shown below.

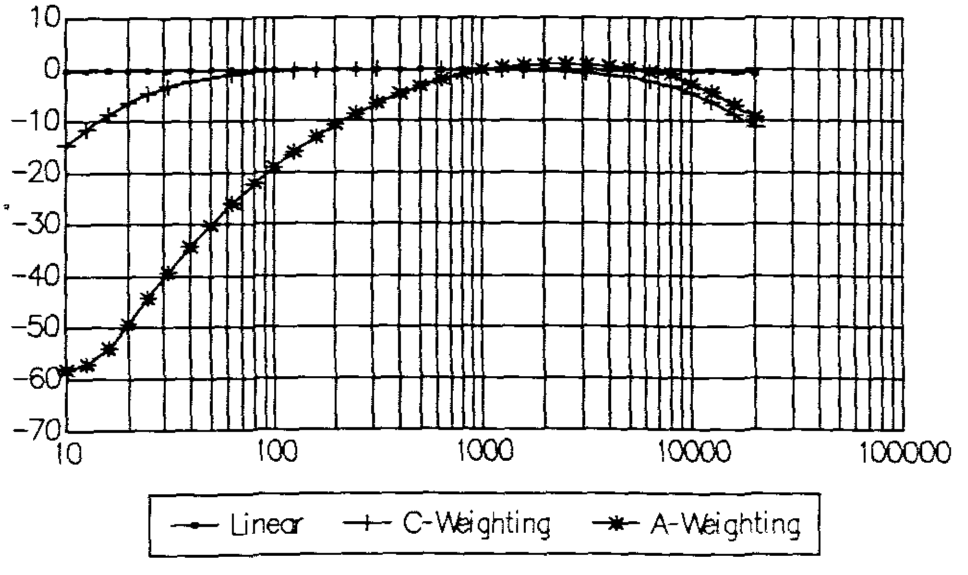


Figure 32: Response of B.A.A weighting networks

The frequency band in which the B.A.A is within tolerance is from 16 Hz to 20 kHz.
A comparison of the B.A.A weighting responses with the reference values and tolerances is shown in Appendix 6.

<u>Selection of Weighting/Filter Networks:</u>

The various options such as the weighting and filtering networks were selected by using two DG201A quad analog switches. Refer to the circuit in figure 33 on page 79.

The DG201A switches were used because of their low on resistance (± 120 ohms) which is constantly linear over the entire signal range. Other factors prevalent were wide signal range (supply rail to rail) and good OFF Channel isolation (70dB).
U1 was used for selecting the linear or weighting networks and U2 for selecting the linear or bandpass filter networks. The digital codes were latched onto the switches using a 74HC174 data latch controlled by the software program.

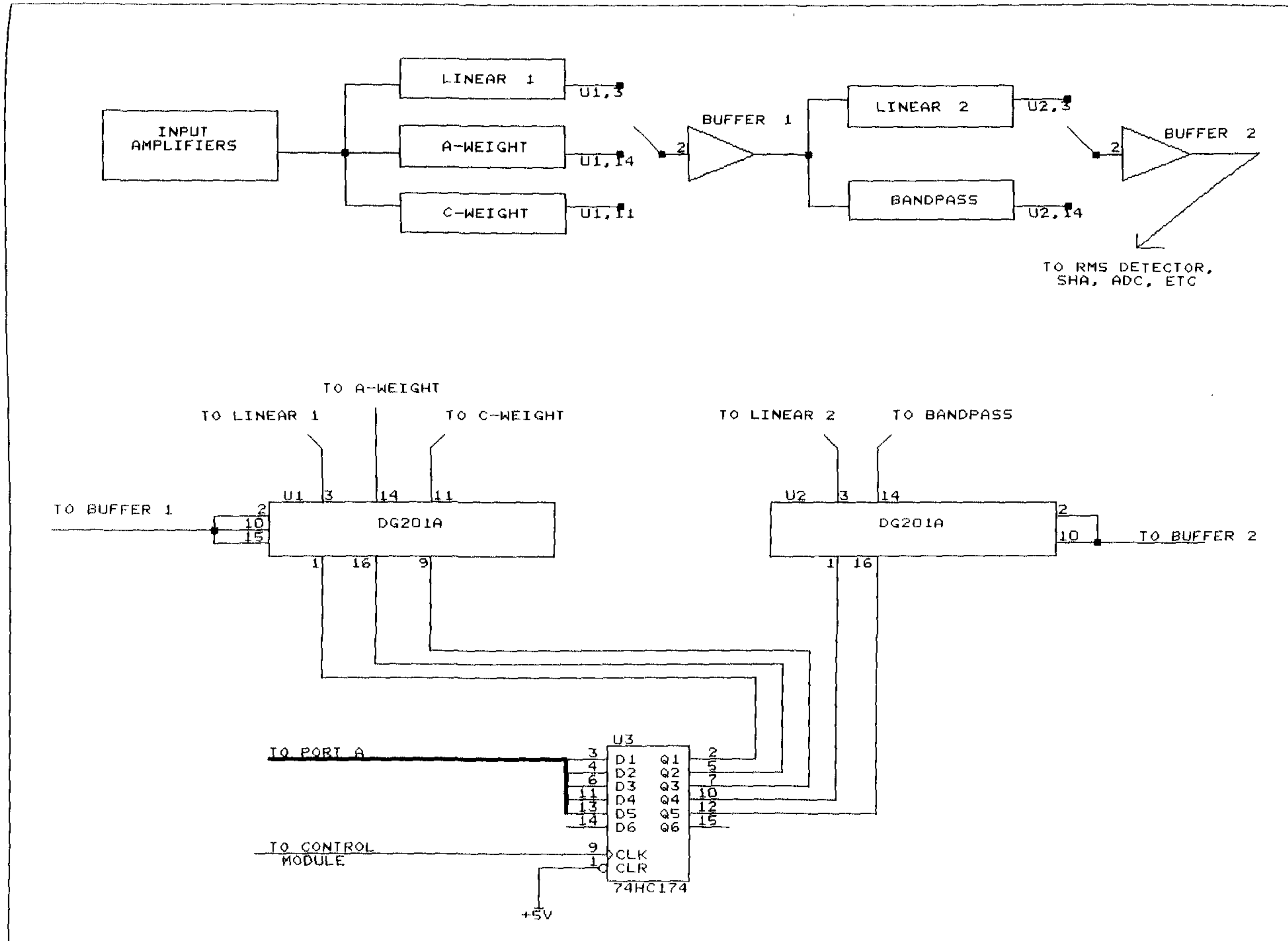


Figure 33: Circuit diagram of Weight/Filter selection

5.7 The RMS Detector

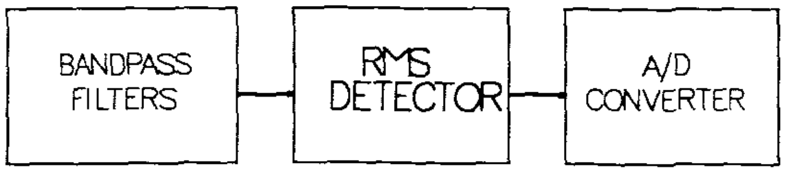


Figure 34: Block diagram of RMS detector

The root mean square (RMS) value of any signal is proportional to its energy content and is therefore one of the most important and often used measures of signal amplitude. For Gaussian noise, the crest factor is theoretically unlimited and true RMS measurement is the only technique to accurately measure noise[8].
The RMS value of a signal is defined as:

$$A_{rms} = \text{sqrt}[\ 1/T * \int_0^T a^2\ (t)dt\ ]$$

a = amplitude of signal
T = period of the signal wave

The RMS detector consists of an RMS to DC converter with an averaging time constant of 125 ms (fast response), in accordance with the IEC 651 specification for sound level meters. The internal circuit is shown in figure 35 on page

8 Analog Devices Linear Design Seminar (1984:III,16)

82. The complete circuit is shown in figure 36 on page 84. The AD536A True RMS to DC converter chip was chosen because of its excellent performance characteristics. The actual computation follows the equation:

$$V_{rms} = Avg * [\frac{Vin^2}{V_{rms}}]$$

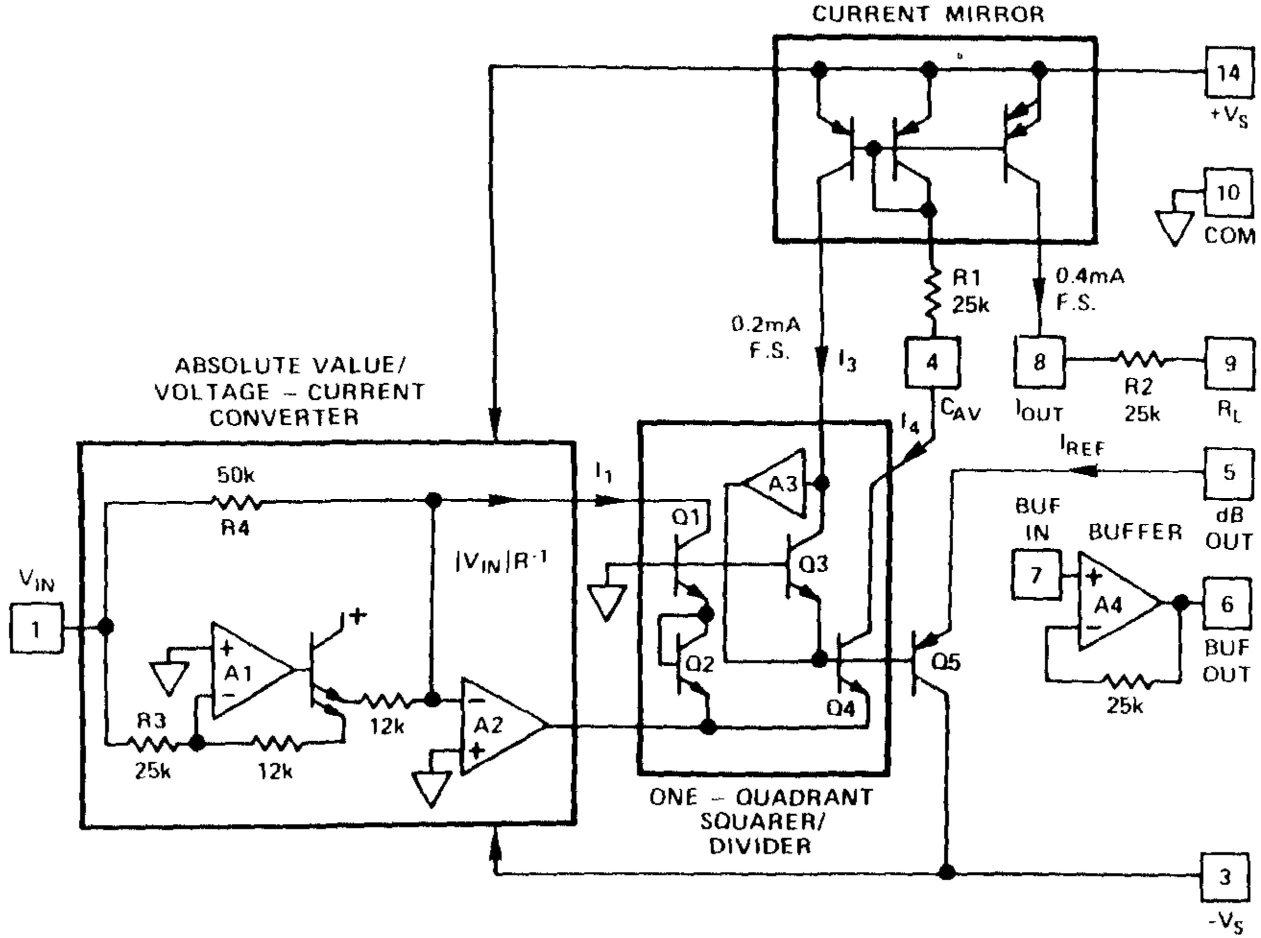
The total error with a DC or sinewave input is ± 2mV which relates to approximately 0.16 percent of the maximum signal level. A useful feature of this chip is the logarithmic or decibel output. The internal circuit which computes dB is very accurate and works well over a 60 dB range. This feature was not used, because the various measuring systems connected to this instrument all had their own dB references and output voltage levels. This made it extremely difficult to correlate the display readings on the B.A.A instrument and the external measuring instruments.
Therefore, only the linear RMS output was used.

The circuitry of the AD536A is divided into four major sections: absolute value circuit, squarer/divider, current mirror and buffer amplifier.
The AC input voltage, Vin, is converted to a unipolar current, I_1, by the absolute value circuit of A_1 and A_2. I_1 drives one input of the squarer/divider which has the transfer function: $I_4 = I_1^2 / I_3$
The output current, I_4, drives the current mirror through a low-pass filter formed by R1 and the external capacitor, C_{AV}
The current mirror returns a current, I_3, which equals $Avg*I_4$, back to the squarer/divider to complete the RMS equation.

$$I_4 = Avg * [I_1^2/I_4] = I_1 \text{ rms}$$

The current mirror produces the output current, I_{OUT}, which equals $2*I_4*I_{OUT}$ and can be used directly. It can also be converted to a voltage with R_2 and buffered by A_4 to provide

The simplified schematic shown below may be used to illustrate the operation of the AD536A.



AD536A SIMPLIFIED SCHEMATIC

Figure 35: Internal circuit diagram of RMS Detector

a low impedance voltage ouput. The transfer function is then:

$$V_{OUT} = 2*R_2*I_{rms} = V_{IN}\ rms$$

The output of the AD536A differs from the ideal output by a DC error and some amount of AC ripple. This error is dependant on the input signal frequency and the value of C_{AV}. Tables are provided in the specification sheet to select a suitable value of C_{AV} to give a minimum DC error.

To reduce the ripple, a large value of C_{AV} can be used, since the ripple is inversely proportional to capacitance. The only disadvantage with this is that the settling time for a step change in input level is increased by the same factor. To overcome this problem, a 2-pole post filter is used.

A graph is provided for selecting suitable values of C_{AV}, C2 and C3, which form part of the filter. The filter allows these values to be reduced as well as provide fast settling times for a small constant ripple. The peak to peak ripple formed less than 0.1 percent of the output reading at 10 Hz with the filter values chosen.

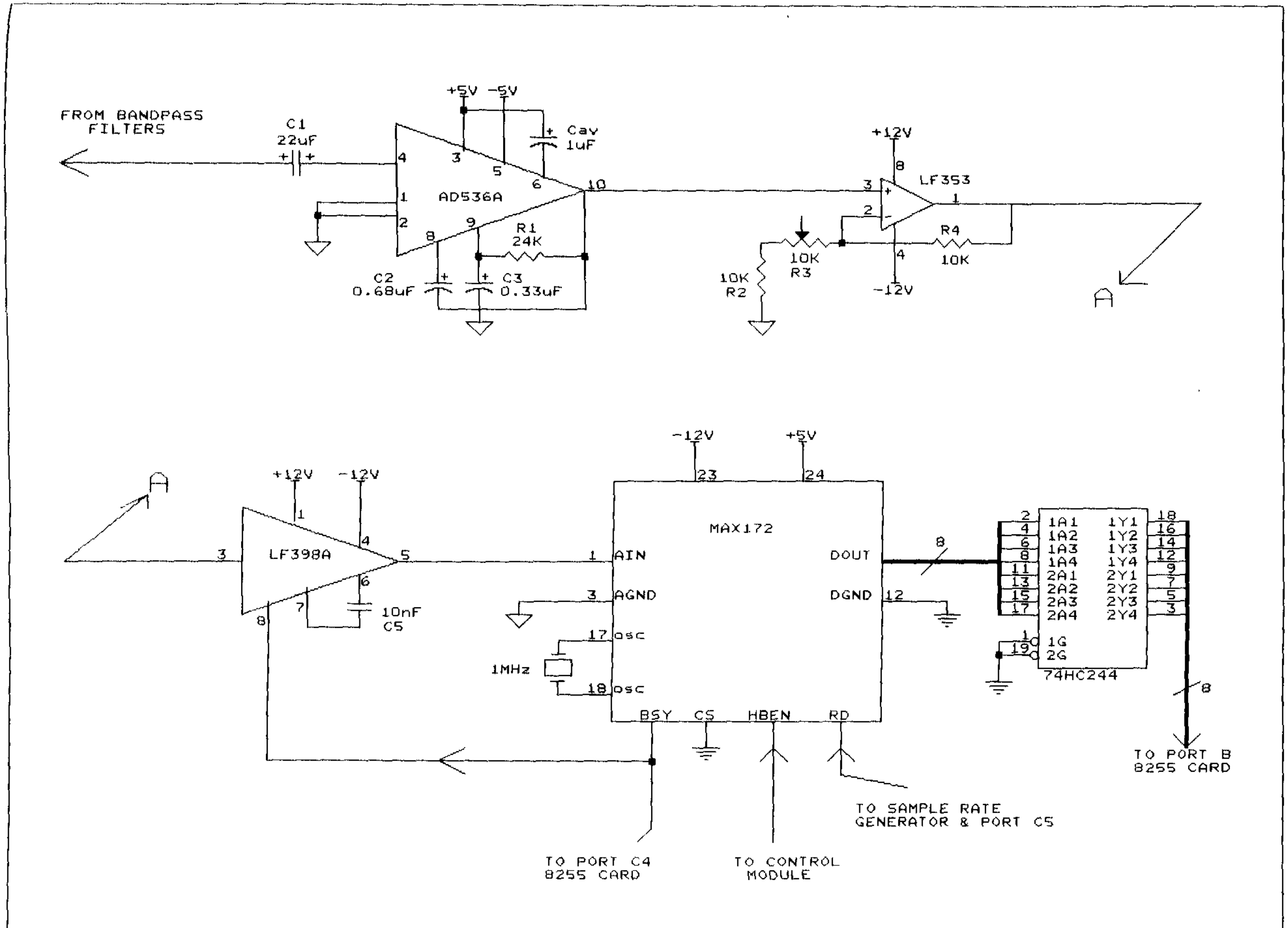


Figure 36: Circuit diagram of RMS and Conversion chips

5.8 The Data Conversion and Interface

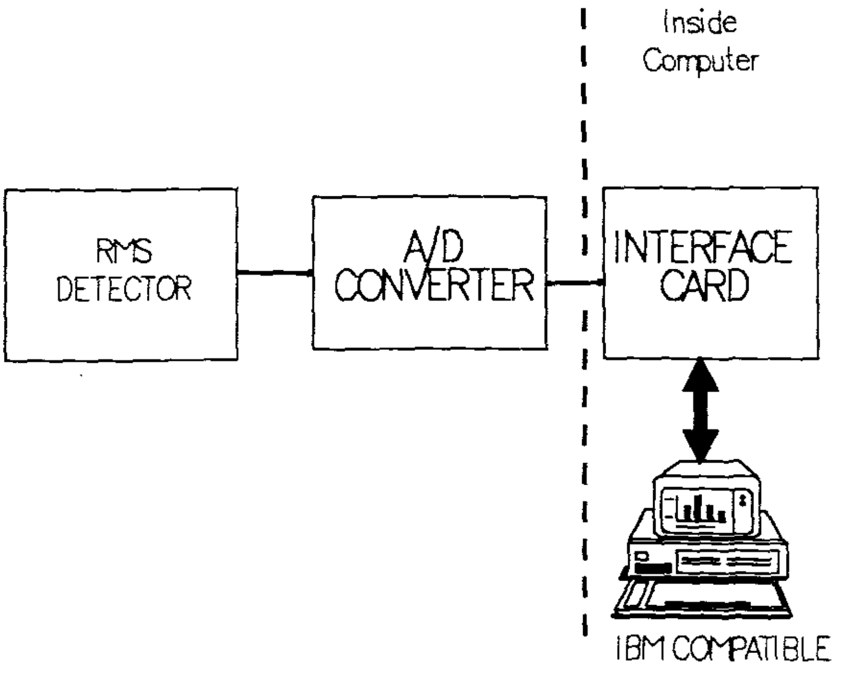


Figure 37: Block Diagram of ADC and Interface

This section concerns the conversion of the analog signal to its digital equivalent form for the purpose of software manipulation. A sample/hold amplifier (SHA) is used in conjunction with an analog to digital converter (ADC) and timing generator circuits. The circuit is shown in figure 36 on page 84.

An op-amplifier (LF353) with a gain of 1.67 was used before the SHA to scale the 3 volt signal to 5 volts so that the input voltage to the ADC covers the full dynamic range (0 - 5v). The gain control was achieved using R2, R3 and R4 in the feedback loop.

The Sample/Hold Amplifier:

The analog input voltage to the ADC must remain fixed during the actual conversion. This requires the use of a sample-hold-amplifier (SHA) which freezes the analog input waveform or takes a "snapshot" of it. It has a **track and hold** mode controlled by the ADC chip (pin 8) and buffers the signal for the noise sensitive, high-impedance ADC. It is an essential component for systems requiring conversion of rapidly changing signals such as Gaussian noise, with high accuracy.

The LF398A was chosen because of its excellent performance characteristics. It has an acquisition time of ten microseconds, gain accuracy of 0.002 percent and low droop rate. The droop rate is defined as the drop in output level over time of the frozen signal when in the hold mode. The overall design guarantees no feedthrough from input to output in the hold mode. The hold capacitor value was carefully selected using the graphs provided in the specification sheet.

A significant source of error in the SHA is the dielectric absorption of the hold capacitor. Only capacitors with a low hysteresis should be used to enable the SHA to accurately take snapshots of the input waveform. A 0.01 μF mylar capacitor (C5) with a low hysteresis ($< 1\%$) was used.

The A/D Converter:

In order to select the most suitable ADC, various factors need to be considered. These are: resolution, accuracy, speed, voltage reference, interfacing, cost and availability.

The resolution determines the number of recognisable quantisation intervals in the ADC transfer function. All ADC's have a quantisation uncertainty of ± 1/2 LSB, therefore an ADC must be selected with enough resolution to reduce this digitising noise to a low enough level. Each bit reduces this noise level by 6 dB[9]. A dynamic range of between 60 to 70 dB was required, therefore a 12 bit ADC with a 72.2 dB range was selected.

Missed codes in an ADC causes the resolution and accuracy to decrease and should be avoided. For example, if a 10-bit ADC misses 10 codes in its transfer function, it is reduced to a LOG_2 (1014) or 9.98 bit converter.

The choice of speed was conservatively chosen at 10 µs (100 kHz), to allow for future development of the instrument. For reverberation and transmission loss measurements, a sampling rate of 250 Hz was found to gather enough data for software analysis. If all six channels are used simultaneously, as in the case of fully automated measurements, a maximum sampling rate of 1500 Hz would be required.

For possible future **impulse response** measurements using two channels, each channel must measure a minimum bandwidth of 10 kHz, requiring a sampling rate of about 30 kHz per channel. To sample both channels together requires a sample rate of about 60 kHz.
A built-in voltage reference is preferred, because it reduces component count and the total cost budget.

The ADC chosen was the **MAXIM 172**, 10 µs, 12-bit, **successive approximation ADC** type designed for moderate conversion speed and resolution. It is functionally equivalent to the more expensive Analog Devices **AD7572 ADC**, but at half the

---

9 Analog Devices Linear Design Seminar (1984:II,6)

price. An on-chip buried zener diode provides a stable 5 volt reference voltage with a low temperature coefficient (25 ppm/°C) to give low drift performance over the full temperature range. It also features a high speed easy to use digital interface with tri-state data outputs.

Conversion is controlled by the **CS**, **RD** and **HBEN** inputs. A logic '0' is required on all 3 inputs to initiate a conversion and cannot be restarted until conversion is complete. Conversion status is indicated by the **BUSY** output, which is low while conversion takes place.
A 1 MHz crystal is used to provide a clock oscillator for the ADC timing.

The output data format was used in the two-byte read for 8-bit data busses, using only data outputs D0-D7. Byte selection is controlled by the **HBEN** input which controls an internal multiplexer. This multiplexes the 12-bits of conversion data onto the lower D0-D7 outputs, (4 MSB's or 8 LSB's). At the end of conversion the low data byte D0-D7 is read from the ADC. A second READ operation with **HBEN** high, places the high byte on data outputs D0-D3 and disables conversion start. Refer to the specifications sheet in Appendix 7 for timing, control and interfacing diagrams. The output data was buffered through an octal latch (74HC244) to prevent any damage to the ADC.

The Assembler Program:

The ADC controlling software had to be extremely fast to read all the data samples accurately, therefore an assembler program included as Turbo Pascal INLINE statements was written to control the ADC read and conversion process. The program listing appears in Appendix 9.

A 'hardware present' check is performed using a time-out facility before a conversion is initiated. The **RD** and **BUSY** lines are monitored for two seconds to make sure they are both low. If they change logic levels during the time-out, this shows that the instrument is not connected or ON and only noise is present on the lines. An error condition is then passed onto the main program and an error window is displayed. If no error is encountered, the conversion sequence is initiated according to the timing diagrams and the required number of samples are captured.

The ADC Sample Rate Generator:

To initiate conversions at a particular rate on the ADC, a sample rate generator was implemented using a Divide by N Binary Down Counter. Refer to figure 38 on page 90.

The 1 MHz clock output function of the ADC was used as the clock input to the counter. Three **MC14029B** 4-bit counters (U1, U2, U3) and a NOR gate (U6) were used to allow 2^{12} (4096) possible sampling rate frequencies to be generated. The codes were latched onto the counters using U4 and U5 by the software program. The output clock pulse of the circuit was very short and it was found that on a faster computer such as the 25MHz 80386, the ADC controlling circuitry and software performed differently than on an XT or AT computer. Instead of doing a two-byte read cycle between ADC conversions, every byte was being read. This was incorrect, as old data was being read on every second read cycle, as shown on the timing diagrams.

The solution was to use a **74HC221** monostable multivibrator (U7) that allowed the **RD** pulse width from the sample rate generator to be lengthened by a factor R1 * C1. The output pulse width, **T = 1.1*R*C**, was adjusted to about 18 µs for the correct operation of the circuitry on all computers.

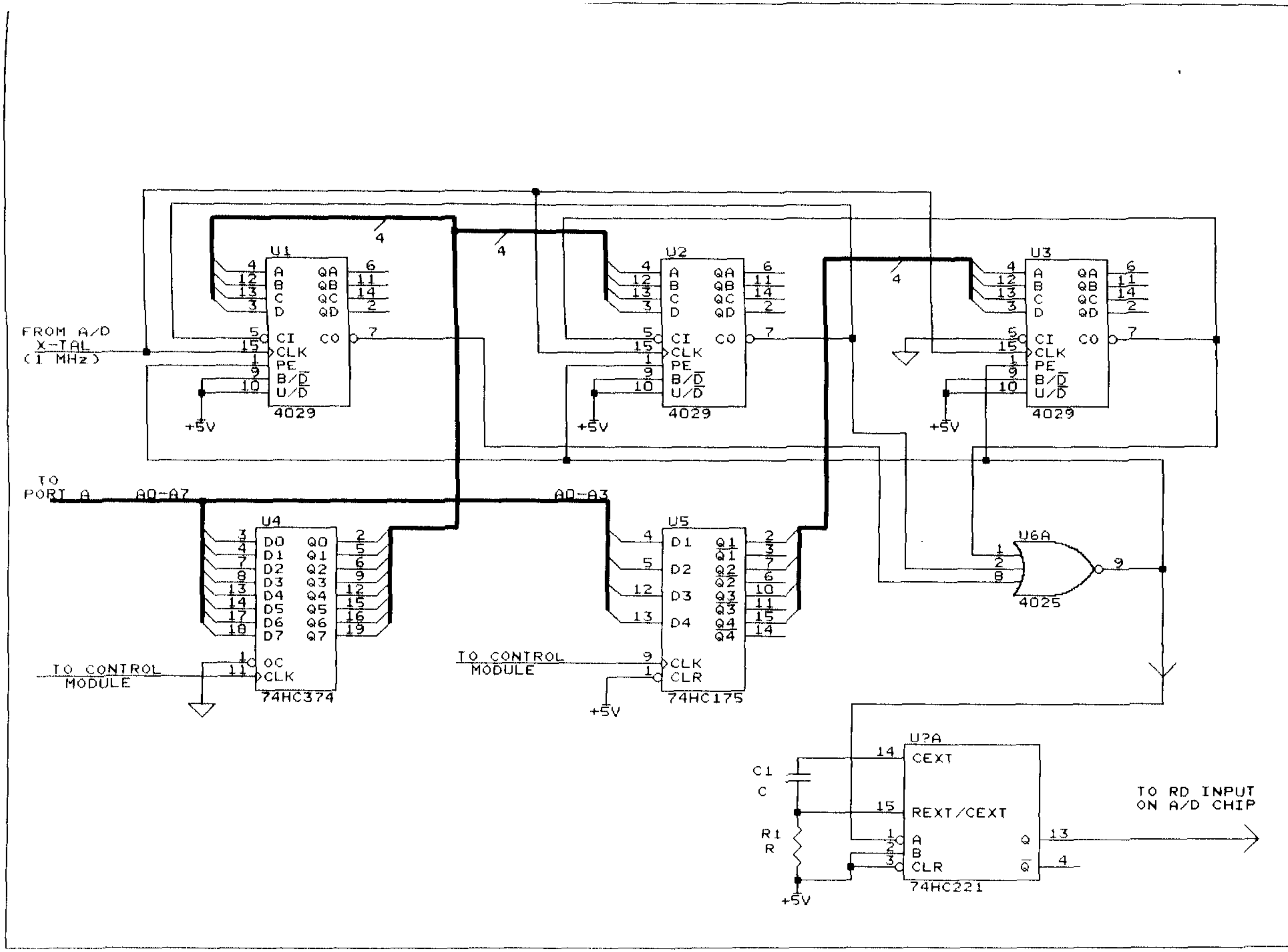


Figure 38: Circuit diagram of Sample Rate Generator

The Digital Interface and Controller

The **8255 Programmable Peripheral Interface** (PPI) is a general purpose I/O component for interfacing peripheral equipment to the microcomputer system bus. The functional configuration of the PPI is programmed by the system software and is used for controlling the circuitry of the instrument and reading in the data from the ADC. An existing general purpose interface card made at the University of Cape Town consisting of the PPI and other addressing chips was used and plugged into a computer expansion slot. Refer to figure 39 on page 92.

The PPI has 24 I/O lines which can be individually programmed in 2 groups of 12 and used in 3 modes of operation. The lines can be used for transmitting or receiving data and for handshaking applications. The 24 lines are divided into 3 ports: port A, B and C. The mode is changed by writing a control word to the PPI for various configurations of the ports. Mode 0 (basic Input/Output) was used with the following setup:

1) Port A as an 8-bit output data bus.
2) Port B as an 8-bit input data bus for the ADC.
3) Port C was split in half. The lower C0 - C3 bits were used as outputs for all addressing and write strobe pulses. The higher C4 - C7 bits were individually used as inputs: C4, C5 for monitoring BUSY and RD signals; C6 for monitoring the input overload error status.

The port C lower bits drive a **74HC154** (U1) 4 to 16 line decoder which drives two **74HC04** (U2,U3) hex inverters. The inverters are for inverting the normally high state of the unselected pins of the decoder to a normally low state. This is necessary, because all the clock strobes of the data latches only operate from a low to high transition.

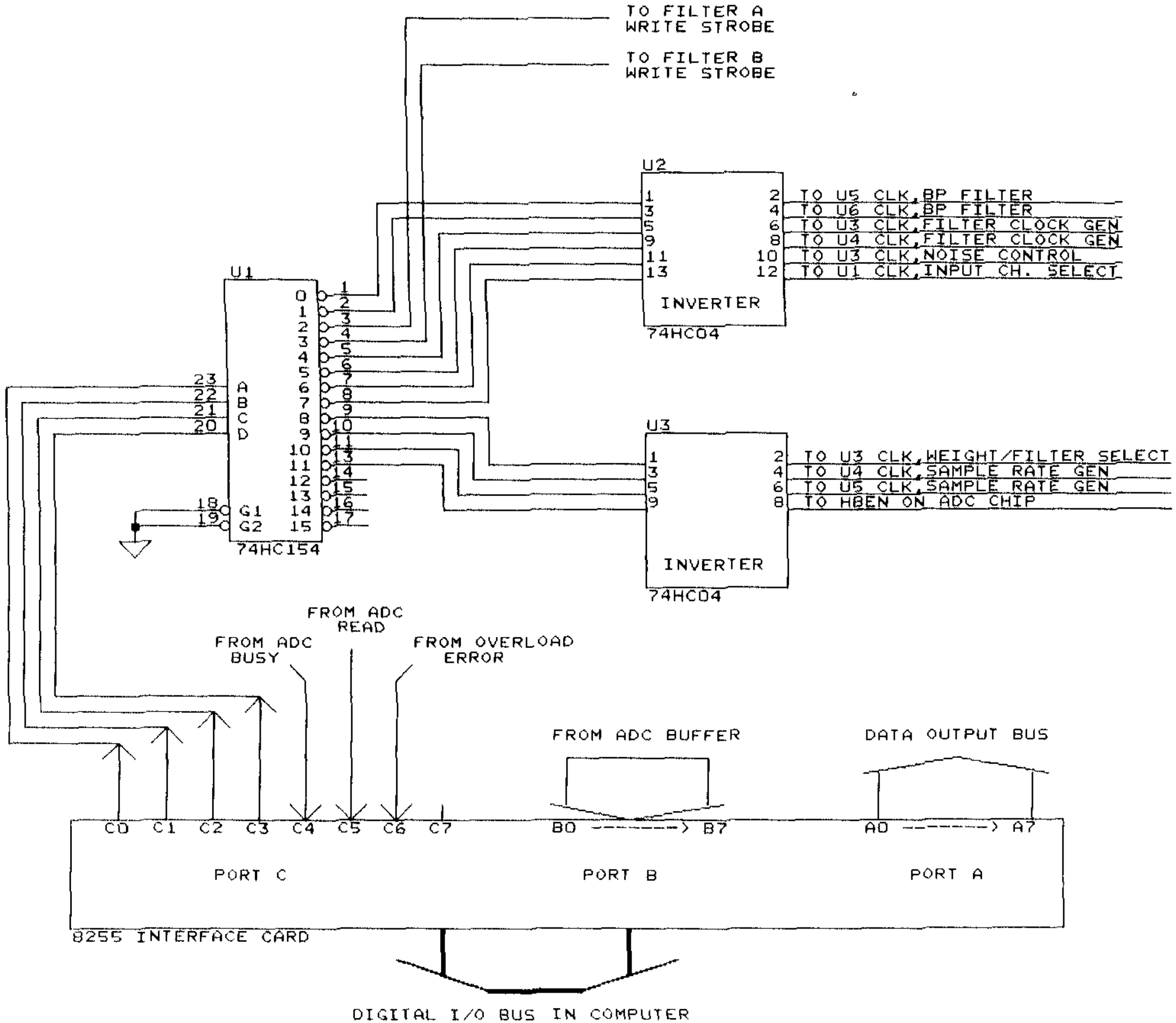


Figure 39: Circuit diagram of Interface/Controller

When a circuit parameter needed to be changed, the 8-bit code was placed on the port A bus, then the specified data latch or address was selected and the data clocked through to the circuit concerned.

5.9 The Software Package

Approximately 3 months was spent becoming familiar with **Turbo Pascal** (version 5.0) and converting the H.P Basic package to run on this system. **Turbo Pascal** is a very fast, structured, high level language designed by Borland International. The reference books used were the **User's Guide, Reference Guide, Turbo Professional** and **Using Turbo Pascal** which guided the author through the arduous task of converting the software package.

The pull-down and pop-up menu system was designed using the **Turbo Professional** package which consists of about 300 state-of-the-art library routines optimized for **Turbo Pascal 4.0** and higher. Developing such a library of routines can take a lot of time and effort which was not available to the author. It also allowed the author to concentrate on developing the controlling software package for the instrument, which is quite large and complicated.

The software package was designed using a main program '**ACOUSTIC**' and fieve units: '**CCTCONT**', '**FILTER**', '**GRAPHICS**' and '**MEASURE**'. The main programme '**ACOUSTIC**' operates with the units to control the menu system and B.A.A instrument. The '**CCTCONT**' unit contols all circuit operations by programming the digital circuit interfaces that control the analog circuits.
The '**FILTER**' unit selects one or third octave filtering and changes the bandpass filter center frequencies.

The '**GRAPHICS**' unit handles all graphics routines such as plotting graphs and tables of results.
The '**MEASURE**' unit performs all measurement and calculation routines for measuring sound pressure level, reverberation time, absorption coefficients and transmission loss data.

A listing and explanation of some of the codes used by the units is shown below:

CCTCONT UNIT

The following codes are used for the B.A.A status. A string code appears on the left - used for displaying the B.A.A status, while the equivalent integer code appears on the right for use in program operations.

noise_type = type of noise (white/pink) = n_t
out_filt = noise output filtering (lin/filtered) = o_f
noise_burst = noise burst control (auto/maual) = n_b
out_chan = noise output channel (A or B) = o_c
start_freq = filter start frequency (100 - 8000) = start_f
stop_freq = filter stop frequency (100 - 8000) = stop_f
in_chan = mic input channels (1 - 6) = i_c
weight = input weighting (none/linear) = w_t
in_filt = input filtering (none/active) = i_f
no_of_samp = no. of measures at each mic position = n_o_s
rt_level = R.T determination level (20,30,40dB) = rt_l
range_pos = range position of S.L.M. = range_p

These variables are used for calculations:
CFactor = calibration factor added/subtracted from value measured by the B.A.A
Volt_Array = an array that stores voltage data read from the A/D converter.
SPL_Array = an array that stores SPL data calculated from the Volt_Array data.

StatusFile = stores all B.A.A status data parameters in a record type file.
Ad_Error = stores the error status of A/D operations.

FILTER UNIT

One_Octave = an array that contains the one octave filter frequencies used by the Standards.
Third_Octave = an array that contains the third octave filter frequencies used by the Standards.

GRAPHICS UNIT

AbsValue = an array that stores the absorption coefficients at each frequency.
Freq1 = refers to the number of frequencies used in a measurement process (from 100 to 8000 Hz).

MENUS UNIT

main = holds information for displaying the TPROF main menu system.
Config = holds information for displaying the TPROF B.A.A status configure menu system.
V,Win,TempWin,MStack = TPROF parameters used as pointers to data structures and the stack/memory for displaying windows and menu systems.
key = key code returned when menu item selected.

MEASURE UNIT

dev1,dev2 = arrays of deviation factors calculated from 3 sets of data values measured at 1 microphone position.
spl1,spl2 = arrays of SPL data.
TempArray = array for storing temporary data.
Pos1,Pos2,Pos3 = temporary arrays for storing transmission loss data at each position before storing data on disk.

ED_T = array of average E.D.T's at each frequency for storing on disk.
R_T = array of average R.T's at each frequency for storing on disk.
DecayTime = array of R.T's for 3 microphone positions.
EDT = array of E.D.T's for 3 microphone positions.
LastPos = holds the value of the last position of the microphone when measuring.
MicPos = holds the value of the microphone position no.
flag = used as an error status flag.

A listing of the program and *units* appears in Appendix 10.

Steps to modify the software

For future modifications to the software the following packages are required: **Turbo Pascal** (version 5.0 or higher), **Turbo Professional** (version 4.0 or higher) and the **B.A.A Software.**
If a different version of **Turbo Pascal** is used, the **Turbo Professional** units will have to be recompiled. For this, the Pascal source and .OBJ files will be required. Refer to the **Turbo Professional Manual** for these procedures. The B.A.A software uses the following **TPROF** units: **TPCrt, TPMenu, TPWindow, TPString** and **TPEdit.**

Copy all Turbo Pascal files with extension .PAS to Turbo directory. These files are supplied with the thesis on the floppy disk labelled " B.A.A Software ". Edit files as required, then recompile them to disk to form unit files with extension .TPU. The 'ACOUSTIC' file will form an .EXE type file which may be run directly. Make sure the TPROF recompiled units are present as well.

The main menu is displayed in figure 40 below:

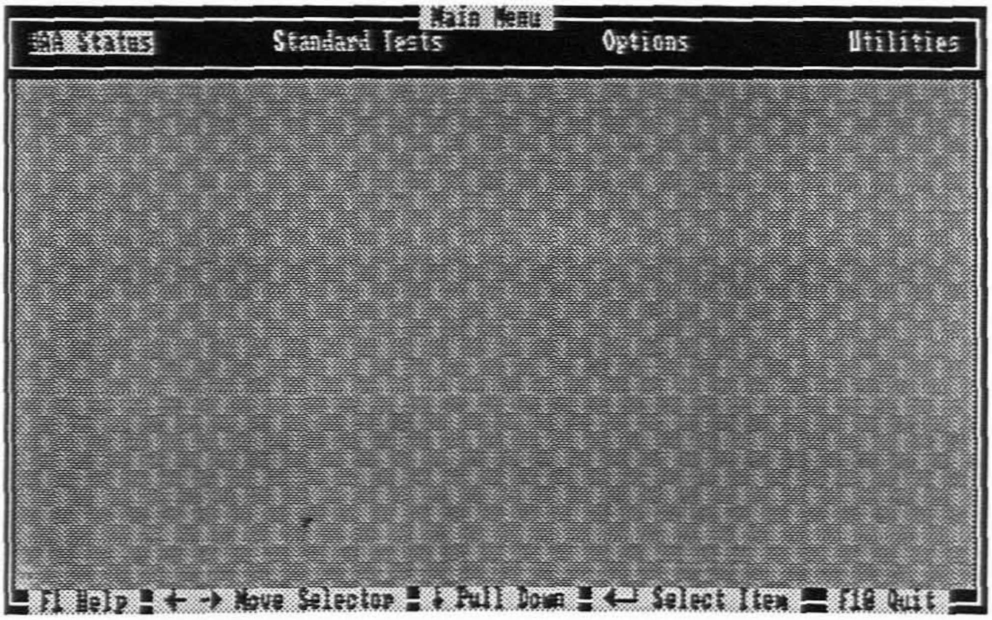


Figure 40: The Main Menu

The options displayed are: **BAA Status, Standard Tests, Options and Utilities.**
The **BAA Status** offers the options of displaying the status of the instrument and/or changing the status on the next screen, as shown in figure 41 below.

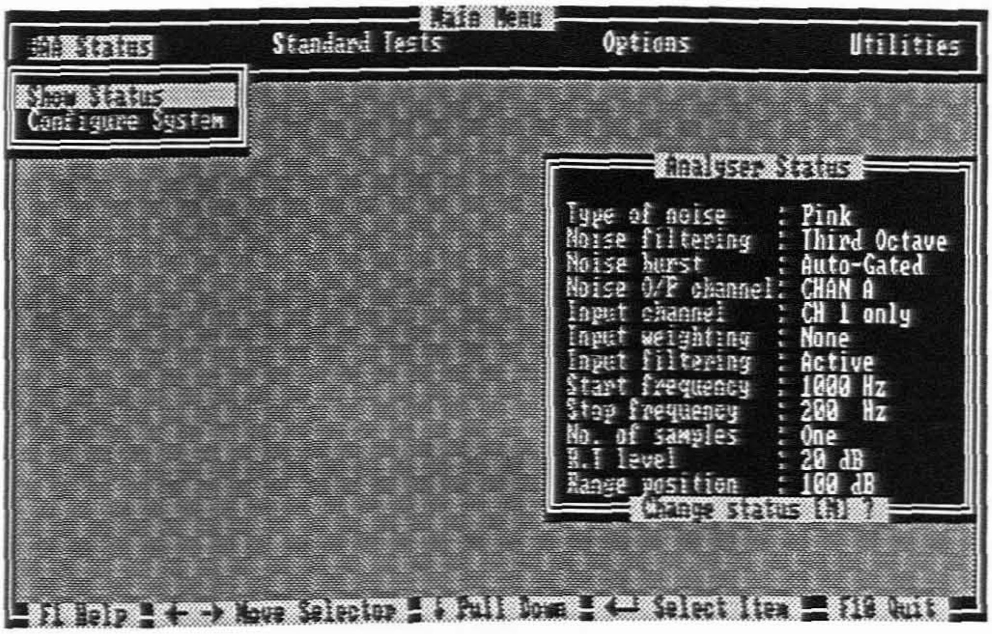


Figure 41: Status Option Menu

<u>BAA Status Menu:</u>

The **Type of Noise** options are white or pink noise. This selection depends on the type of measurement to be carried out. For low frequency noise enhancement and in large auditoria, pink noise would be used.

Noise Filtering concerns the type of filtering of the noise generator output. The options are **Linear, Third Octave** and **One Octave.** The **Linear** selection allows unfiltered white/pink noise to be emitted. The **Third Octave** selection performs third octave filtering of the white/pink noise in the region from 100 Hz to 8000 Hz. The **One Octave** selection performs one octave filtering of the noise, as per the third octave selection.

The **Noise Time Burst** offers the choice of **Auto-Gated** or **Continuous** and determines the state of the noise output. In the **Auto-Gated** mode, the noise burst is controlled by the software program. The duration of the burst is determined by the type of measurement being carried out. In the **Continuous** mode, the noise is continuously available at the output.

The **Noise Output Channel** determines whether the noise will be available at channel A **or** channel A and B. If two loudspeakers are available, both channels may be used for automated measurements.

Input Channel concerns the number of microphones available for the measurement. With only one microphone, channel 1 is used for all measurements. With three microphones, channels 1 to 3 are used for semi or fully automated measurements. With six microphones, channels 1 to 6 are used for fully automated measurements.

Input Weighting offers the choice of **Linear, A-Weighted** or **C-Weighted** responses. **Linear** allows the signal to pass through without any weighting effects. The **A-Weighted** selection adds an A-weighting response to the signal, while the **C-Weighted** selection adds a C-weighting response to the signal.

Input Filtering determines if any filtering is performed on the input signal. The **Linear** selection has a linear response, while **Active** performs octave or third octave filtering using the same type of filter bank as in the noise generator. The type of filtering depends on the type of **Noise Filtering** selected previously.

The **Start Frequency** selects the starting frequency of the measurement to be performed. The range of frequencies are from 100 Hz to 8000 Hz in third octave steps.

The **Stop Frequency** selects the last frequency of a measurement sequence. It has the same range as the **Start Frequency** above. The stop frequency must be greater than the start frequency otherwise no measurement is performed.
If the stop frequency equals the start frequency, only one measurement is performed at that frequency.

The **No. of Samples** determines if one or three microphone positions are to be used in the room. This information tells the software what level of automation is required. For eg:
If one microphone is available and three postions are required, the software must inform the operator to manually move the microphone to each position.

R.T Level concerns reverberation time measurements. The **20, 30** and **40 dB** R.T determination levels are 3 options for calculating the R.T from the decay slopes. All slope calculations start at 5 dB below the steady-state level[10], while the choice of the calculation stop level is up to the operator. Obviously, a maximum level of 40 dB would provide the most acurate results. However, this requires a range of about 55 dB between the steady-state level and the background noise level. It is recommended to be about 10 dB above the background noise level for accurate measurements[11].

The **Range Position** is entered by the operator and corresponds to the full scale deflection value (dB) on the display of the sound level meter or measuring instrument used. The operator may change the configuration of any of the above options by selecting the **Configure System** menu displayed in figure 42 below:

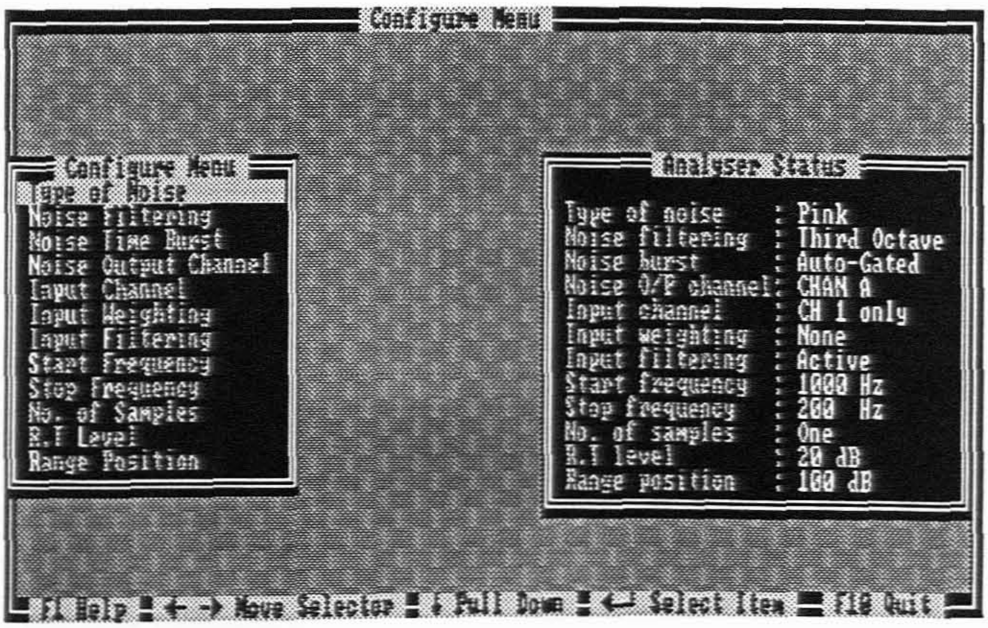


Figure 42: Configure System Menu

When the operator selects a parameter to change on the left side of the screen, a submenu is popped up displaying the various options. The status and circuitry are updated and

---

10 ISO R354 (1985:8)
11 ISO 140 (1978:part 3)

displayed on the right side of the screen. Pressing **F10** quits the configure menu and returns to the main menu.

Standard Tests Menu:

The **Standard Tests** menu is selected after the system has been suitably configured for the type of measurement to be performed. All tests are run from this menu as displayed in figure 43 below:

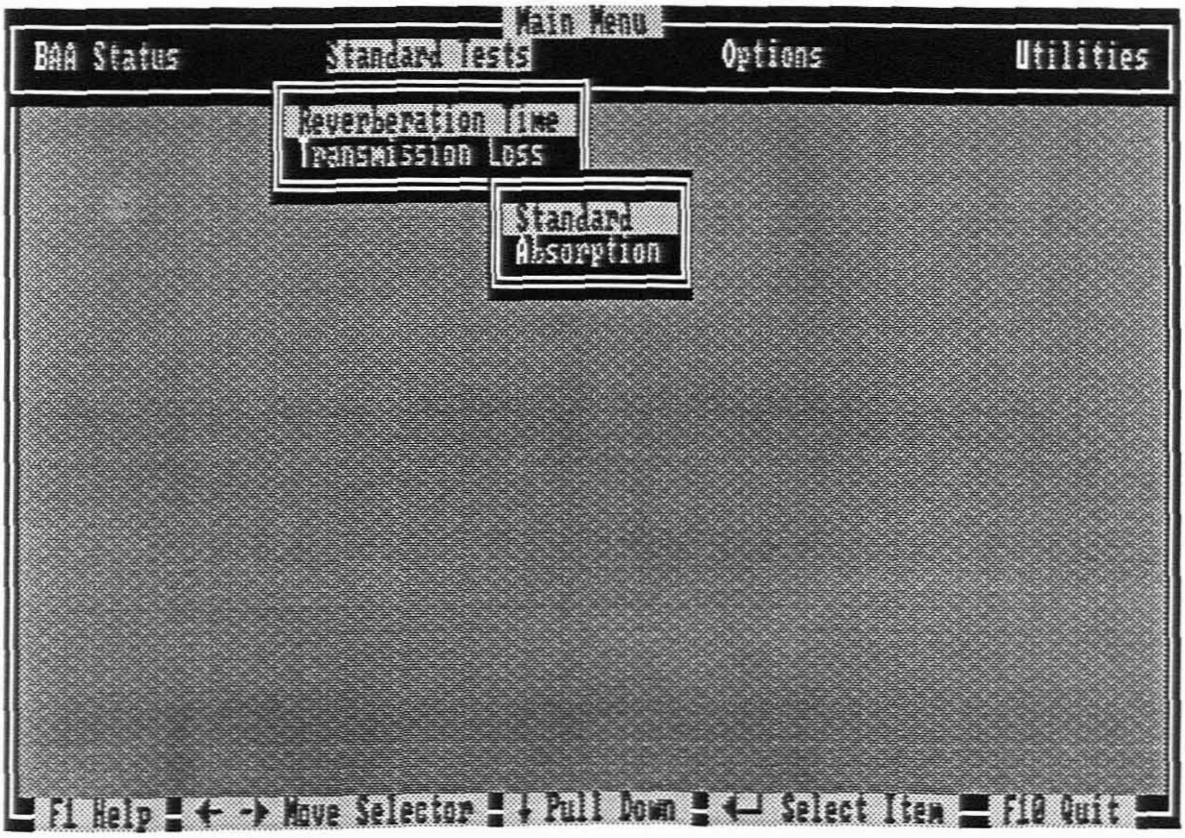


Figure 43: Standard Tests Menu

Reverberation Time offers two options of R.T tests: **Standard** and **Absorption.** A single R.T test in a room may be performed under the **Standard** option, measuring the R.T's at the selected frequencies and storing the results in file named by the operator. An absorption test on a material sample may be performed under the **Absorption** option and the absorption coefficients stored in a file named by the operator.

Transmission loss measurements may be performed according to various calculations, as shown in figure 44 below.

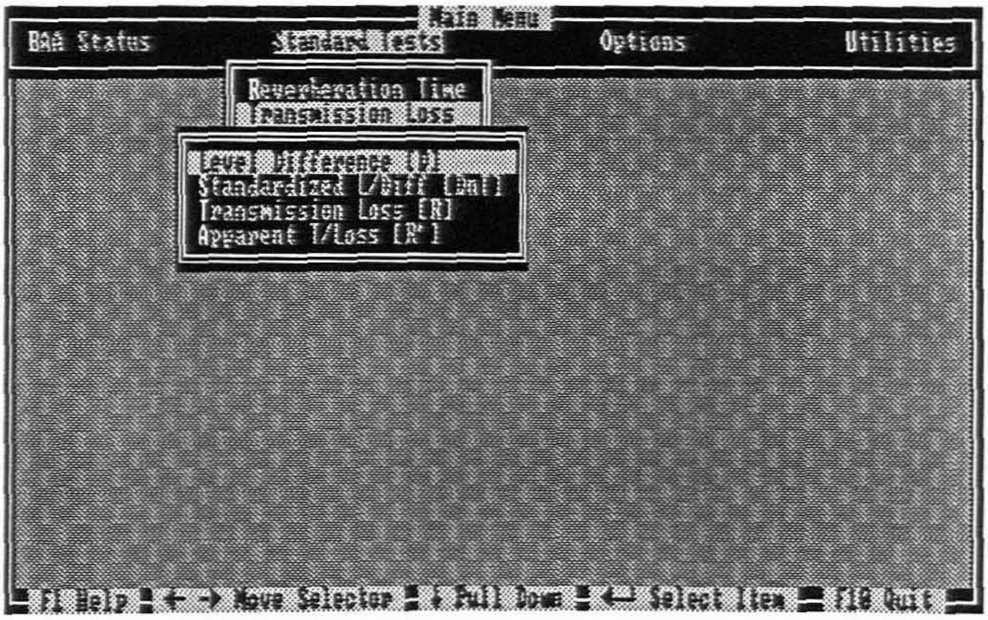


Figure 44: Transmission Loss Menu

These calculations are in accordance with the ISO 140 standard and are calculated from the following equations:

1) **Level Difference:** $D = L1 - L2$
2) **Standardised L/Diff:** $D_{nT} = D + 10*\log(T/0.5)$
3) **Transmission Loss:** $R = D + 10*\log(S/A)$
4) **Apparent T/Loss:** **same as (3) above**

The type of test depends on the calculation selected. When measuring the level difference (D), the R.T is not required and the measurement time is reduced. For all measurements, the receiving room background noise SPL is first measured before the receiving room SPL. A correction is applied if the difference in SPL's does not conform to ISO 140, part 4. If the signal level is less than 3dB above the noise level an error symbol is printed in the output data table at the problem frequency.

The user is prompted for the applicable parameters, such as:

volume, surface area and the file name for calculating and storing the results.

Options Menu:

The **Options** menu offers the following choices: **Calculations**, **RT Table**, **Absorb Table**, **Decay Plot** and **Hardcopy**. These provide most of the displaying options as shown in figure 45 below:

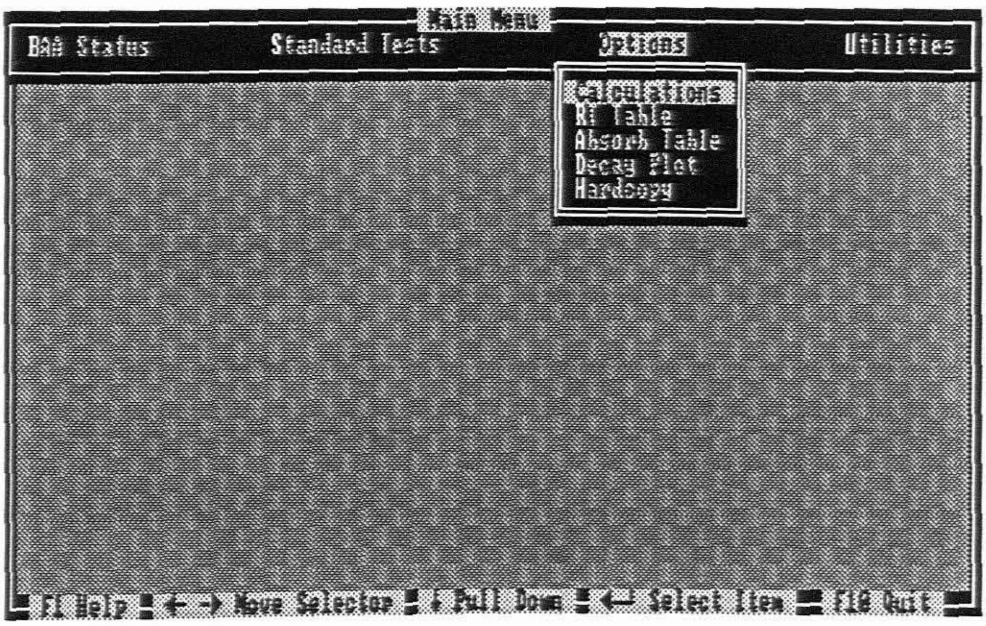


Figure 45: Options Menu

The **Calculations** option performs the same calculations that were selected for the transmission loss tests and then displays the results in tabular form.

R.T Table displays the results of reverberation measurements in tabular form. The early decay time (EDT) is also displayed.

Absorb Table displays the results of an absorption test that was performed on a material sample under the standard tests menu.

The **Decay Plot** facility allows a visual display of a reverberation decay plot at a selected frequency. This is for the purpose of observing an abnormal room response at a certain frequency, causing an R.T result error.
Hardcopy is used for printing the table or graph that is displayed in the current window.

<u>**Utilities Menu:**</u>

The **Utilities** menu offers the following options: **Real Time Display, Delete Files** and **Suspend To Dos.** These functions are shown in figure 46 below:

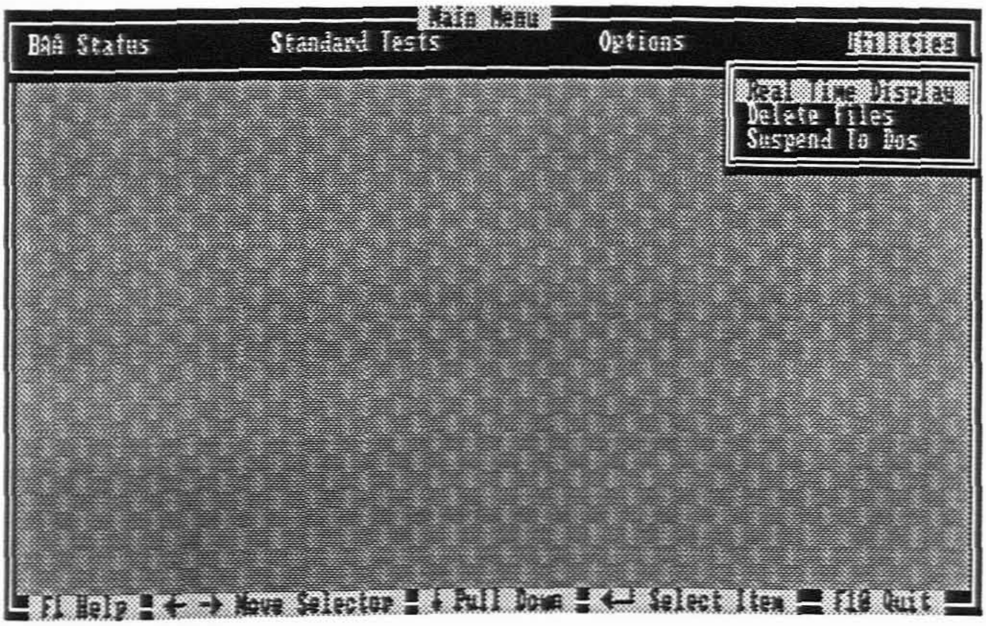


Figure 46: Utilities Menu

The **Real Time Display** offers the facility to display the SPL in decibels of the sound level meter or measuring equipment connected to the analyser instrument. The user selects the instrument configuration under the **BAA Status** menu and then switches to this utility to view the SPL in that mode. There is an option to perform a software calibration before displaying the reading.

The **Delete Files** utility allows the user to delete any existing files that are obselete or for increasing the memory space available.

The **Suspend To Dos** utility temporarily suspends the main program by returning to the operating system. This allows the user to perform any file or directory operations without actually quitting the main program. If enough memory is available, other programs may be run within this shell. To re-enter the main program, the user types **exit** from the command line.

5.10 A Measurement Example

This section describes the actual sequence of steps in a typical measurement procedure. The procedure described is an absorption test on a material sample performed in the Central Acoustics Laboratory at the University of Cape Town in accordance with the ISO R354 standard. The equipment required is listed below:

1) IBM compatible computer (speed preferably > 10 MHz)
2) The Building Acoustics Analyser plus software
3) A sound level meter or mic with appropriate pre-amplifier
4) Power Amplifier (capable of ± 100 dB's re 1pW sound power level)
5) Omni-directional loudspeaker
6) All interconnecting leads

Setup and Calibration:

Once the equipment has been set up as shown in figure 47, the software is installed as described on page 107.

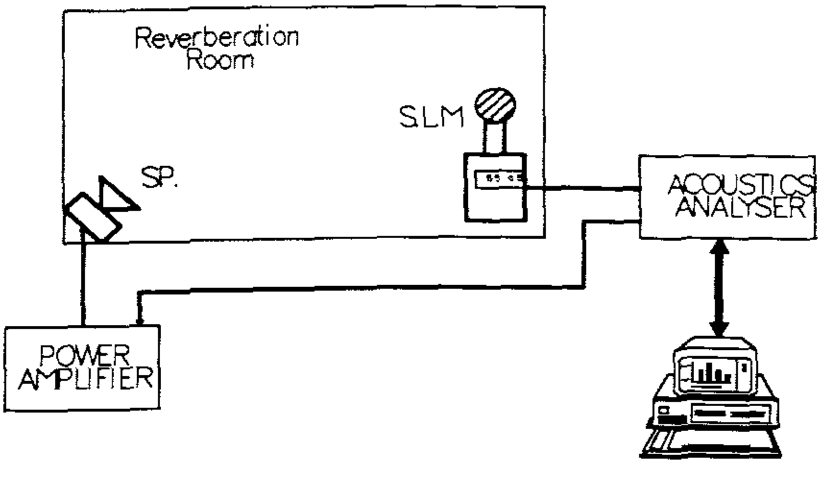


Figure 47: Equipment set-up

1) Install B.A.A interface card into computer and plug ribbon connector into card and rear of B.A.A instrument.
2) Make sure volume on power amplifier is at the minimum.
2) Switch on all equipment including B.A.A and computer.
3) Insert floppy disk labelled " B.A.A Software " into drive A.
2) Type " a: acoustic " the press enter.
3) Wait for main menu system to appear on screen.
4) Increase volume on power amplifier to required setting.

NOTE: Use only the **AC** output of the SLM or measuring instrument.

The user now selects the **Show Status** option under **BAA Status** to check the status of the instrument. To change the options, the **Configure System** menu is selected and the following options are selected:

- Pink noise
- Third octave filtering
- Auto-gated time burst
- Channel A noise ouput
- Input channel 1
- No input weighting
- Active input filtering
- Start frequency of 100 Hz
- Stop frequency of 4000 Hz
- No. of samples equals one
- R.T level of 40 dB
- Range position of 100 dB

Before a real measurement is started the equipment should be calibrated. With the B.A.A, the procedure is simple and the adjustment is automatically taken care of by the instrument by using software auto-calibration.

Calibration is performed with either the B&K Sound Level Calibrator Type 4230, which produces a reference output of 94 dB, or the B&K Pistonphone Type 4220, which produces an output of 124 dB.

The SLM or measuring instrument must first be calibrated according to the applicable manual to display the correct reading before the B.A.A is calibrated. All calibration can be carried out using the same setup. The **Input Weighting** and **Input Filtering** must be set to **None**. The **Range Position** must be set according to the full scale deflection value of the SLM or measuring instrument connected.

The user must then select the **Real Time Display** under the **Utilities** menu. The calibration is then performed by first entering the reference level (eg. 94 dB) when prompted for it, then activating the reference source. The B.A.A takes an average of the SPL over a five second period and stores a correction factor, which is the difference between the typed-in value and the reference level. This correction factor is then applied to further measurements.

Running the Measurement:

The user now selects the **Reverberation Time** option under the **Standard Tests** menu, then selects the **Absorption** option. The user is prompted for the filename for the test and a description of the room or material sample. Once this information is entered, the test is automatically carried out in the empty room with no sample present, then with the sample subsequently added.

In order to arrive at a good estimate of the reverberation times, it should be measured at more than one position in the room and also three times at each position[12]. The user will be informed about where and when to position the microphone(s) if only one microphone is used. Usually, **position 1** is on the one side of the room, **position 2** around the middle, and **position 3** on the opposite side of the first position.

The test is run starting at the selected **Start Frequency** of 100 Hz, after which it automatically shifts to the next center frequency and all the way up to the selected **Stop Frequency.** After completion of the measurements in the various frequency bands at each position, the R.T values are stored in the analyser.

The results may be viewed by selecting the **Absorb Table** under the **Options** menu. The user is prompted for the filename of the test, the dry room temperature in degrees celcius and the surface area of the material sample (metric) The results are displayed in graphical and tabular form and may be printed out using the **Hardcopy** facility. An example of a typical printout of results is shown on pages 110, 111, and 112.

---

12 ISO R354 (1985:8)

Figure 48: Print-out from Absorption test

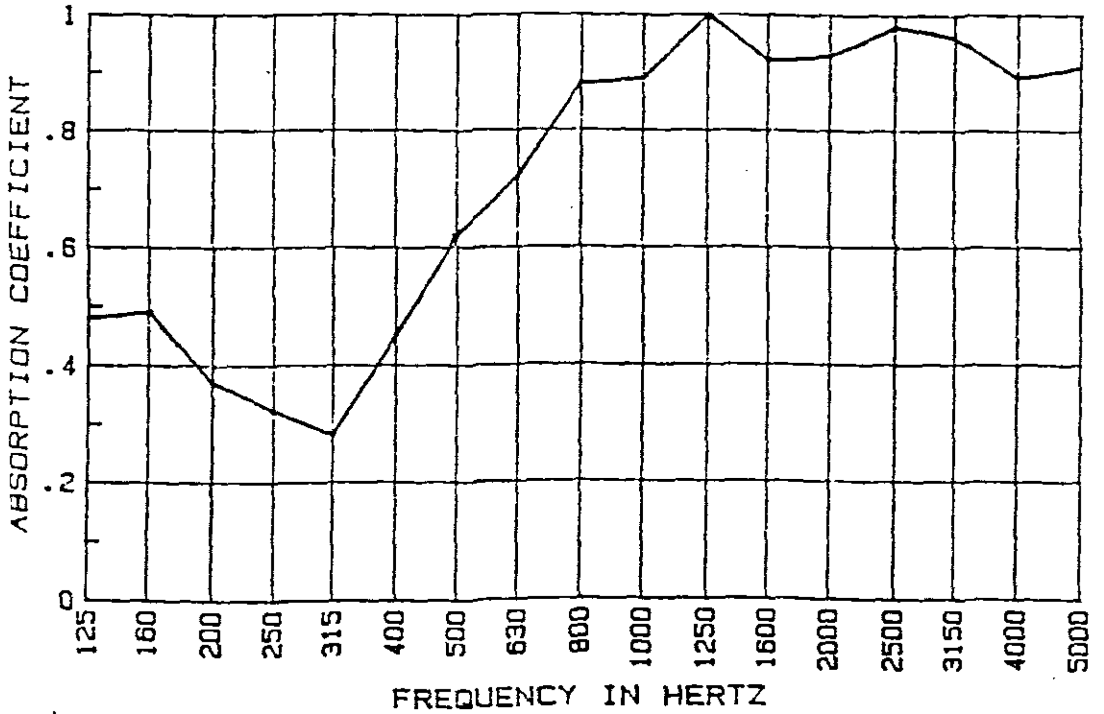
NIVERSITY OF CAPE TOWN

Central Acoustics Laboratory

REPORT NO. 89-4-14

Private Bag · Rondebosch 7700 · Cape · Republic of South Africa
Telephone: 6502792

SOUND ABSORPTION TEST
ON
SKANDIA OFFICE SCREENS



Frequency Hz	Absorption coefficient	Frequency Hz	Absorption coefficient
125	0.48	1000	0.89
160	0.49	1250	1.02
200	0.37	1600	0.92
250	0.32	2000	0.93
315	0.28	2500	0.98
400	0.45	3150	0.96
500	0.62	4000	0.89
630	0.72	5000	0.91
800	0.88		

NRC = 0.75

Figure 49: Print-out from Insulation test

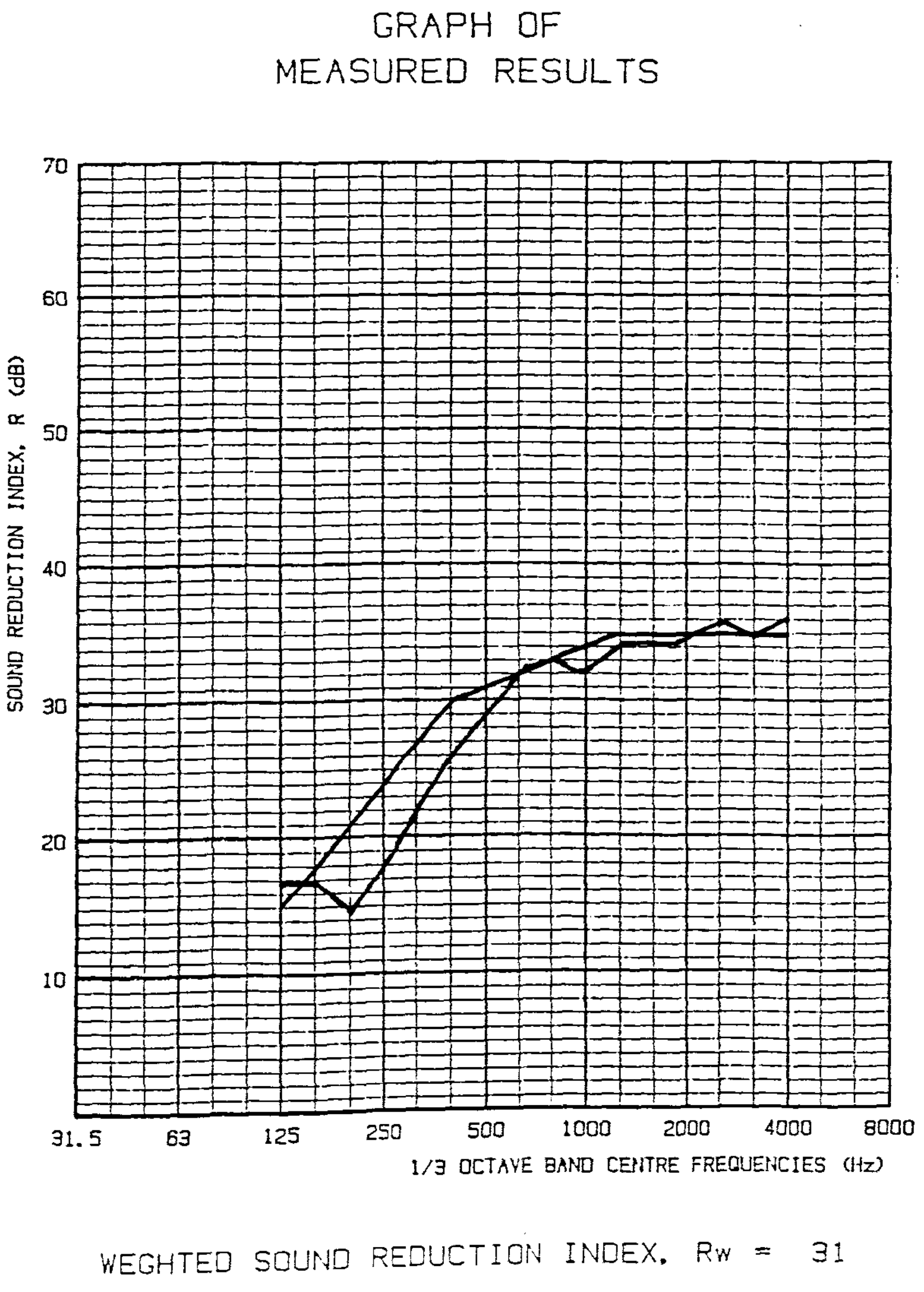
REPORT NO. 89-5-3

TABULATED RESULTS OF INSULATION TESTS

1/3 Octave band centre frequency (Hz)	Sound level difference (dB)	Receiving room Reverb. time (sec)	Receiving room Absorption (metric)	Correction 10 Log S/A (dB)	Reduction Index, R (dB)
125	15.67	1.58	7.99	1.05	16.72
160	16.67	1.30	9.71	.21	16.88
200	14.83	1.12	11.27	-.44	14.39
250	17.84	1.26	10.02	.07	17.91
315	22.50	1.12	11.27	-.44	22.06
400	26.17	1.22	10.34	-.07	26.10
500	29.66	1.16	10.88	-.29	29.37
630	33.00	1.04	12.13	-.76	32.24
800	34.33	.96	13.15	-1.11	33.22
1000	33.00	1.00	12.62	-.93	32.07
1250	35.00	1.08	11.68	-.60	34.40
1600	34.50	1.14	11.07	-.36	34.14
2000	35.17	1.08	11.68	-.60	34.57
2500	36.17	1.16	10.88	-.29	35.88
3150	35.50	1.08	11.68	-.60	34.90
4000	37.67	.88	14.34	-1.49	36.18

Figure 50: Print-out from Insulation test

REPORT NO. 89-5-3



6. COST OF THE SYSTEM

An important part of the project concerns the total cost of the system and availability of the circuit components. Certain components, such as the conversion chips, were available from various manufacturers at different costs. The **MAXIM** components were about half the cost of the **ANALOG DEVICES** components and offered basically the same specifications. The **MAXIM** components were locally available, while the **ANALOG DEVICES** had to be obtained from overseas suppliers. The availability was an important factor in the eventuality of a component being destroyed during design and testing.

The condensed list of component and material costs incurred were :

Complete Outer Box Unit	R 280
Power Supply	R 60
Connectors/P.C.B's	R 315
Logic components	R 130
Analog components	R 126

Specialised Components:

4 x Filter Chips (MAX262)		R 150
1 x DAC (AD7533)		R 30
1 x ADC (MAX172)		R 50
1 x RMS/DC Converter (AD536)		R 40
1 x Sample/Hold (LF398)		R 11
2 x Input Mux-Amplifier (LM604)		R 33
1 x V.C.O (XR2207)		R 15
8255 Interface Card		R 120
	Total	R 1 360

7. RESULTS

The individual results and comparisons of each module have already been discussed in chapter 5. The results and findings of the tests performed with the BAA, as a complete instrument, in comparison to the existing equipment are described in this chapter.

The Brüel & Kjaer type instruments were used as references for all performance comparisons of the BAA for the following reasons:

1. They comply with the strictest international specifications.
2. The instruments were traditionally used for the measurements now performed by the BAA.

The most basic test was the comparison of the measurement of sound pressure level using the B & K type 2230 Precision Integrating Sound Level Meter and the BAA. The tests were performed in the large reverberation room at CAL. The SPL was generated using the third octave filtered noise output of the BAA. The signal was then amplified through a Crown amplifier to the Tannoy loudspeaker in the room.

The BAA was configured by the software to generate and measure the filtered noise at third octave intervals from 100 Hz to 8000 Hz. The noise burst interval was five seconds allowing the author enough time to record the SPL's shown on the 2230 display. The AC output of the 2230 was used as the input to the BAA so that both instruments used the same microphone and pre-amplifier combination. No signal processing was done by the 2230 at the AC output, therefore a true comparison of the displayed SPL's could be achieved because each instrument used its own processing methods.

The results of the SPL comparison test are shown graphically in figure 51 on page 116.
A close correlation between the data can be seen which proves the accurate measurement capability of the BAA. The values agreed to within ± 1.3 dB of each other.

The reverberation test formed the second comparison. The test method was performed as described in chapter 5. The BAA controlled the test procedure, emitting the required filtered noise, measuring and storing the data. The author then used the old equipment to record the data and was present in the reverberation room to synchronize each measurement.

The time taken to complete the test with the old equipment was ± 1.5 hours, while with the BAA this time was reduced to ± 15 minutes, using an 80386 type computer. This proved the amount of time saving achieved when using the BAA. The results of the reverberation comparison test are shown graphically in figure 52 on page 116.

The slight differences could be attributed to the following:

1) The author was present in the reverberation room when recording the data on the old equipment.
2) The BAA uses the least squares technique of curve-fitting to determine the decay rate (R.T), which could result in slightly different values to the subjective evaluation of the old method.

The other comparison tests performed were the Level Difference (with background noise level) and Absorption test. The measurement times taken for these were ± 45 minutes (old method) reduced to ± 8 minutes (BAA method) for transmission loss, and 3 to 4 hours (old method) reduced to ± 30 minutes (BAA method) for absorption test.

Figure 51: **COMPARISON OF S.P.L LEVELS**

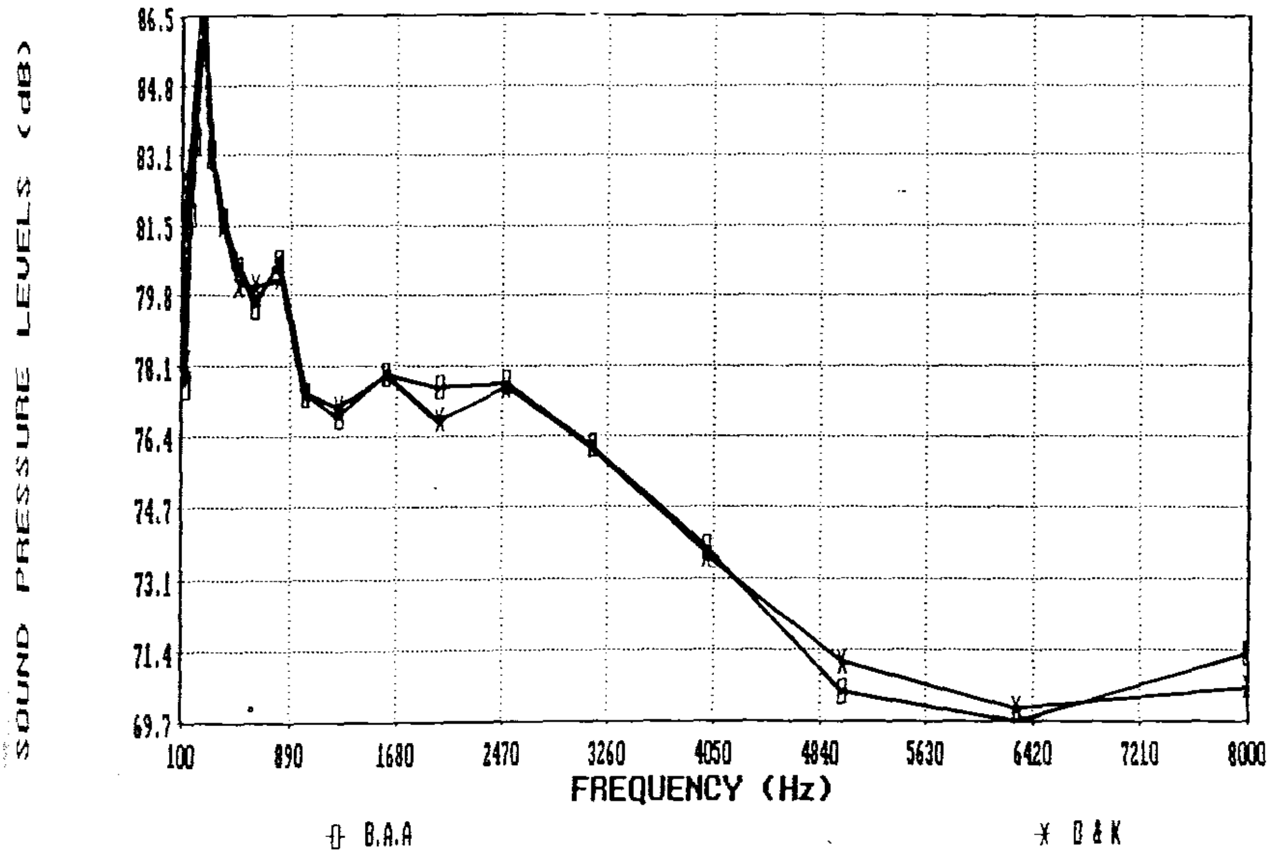
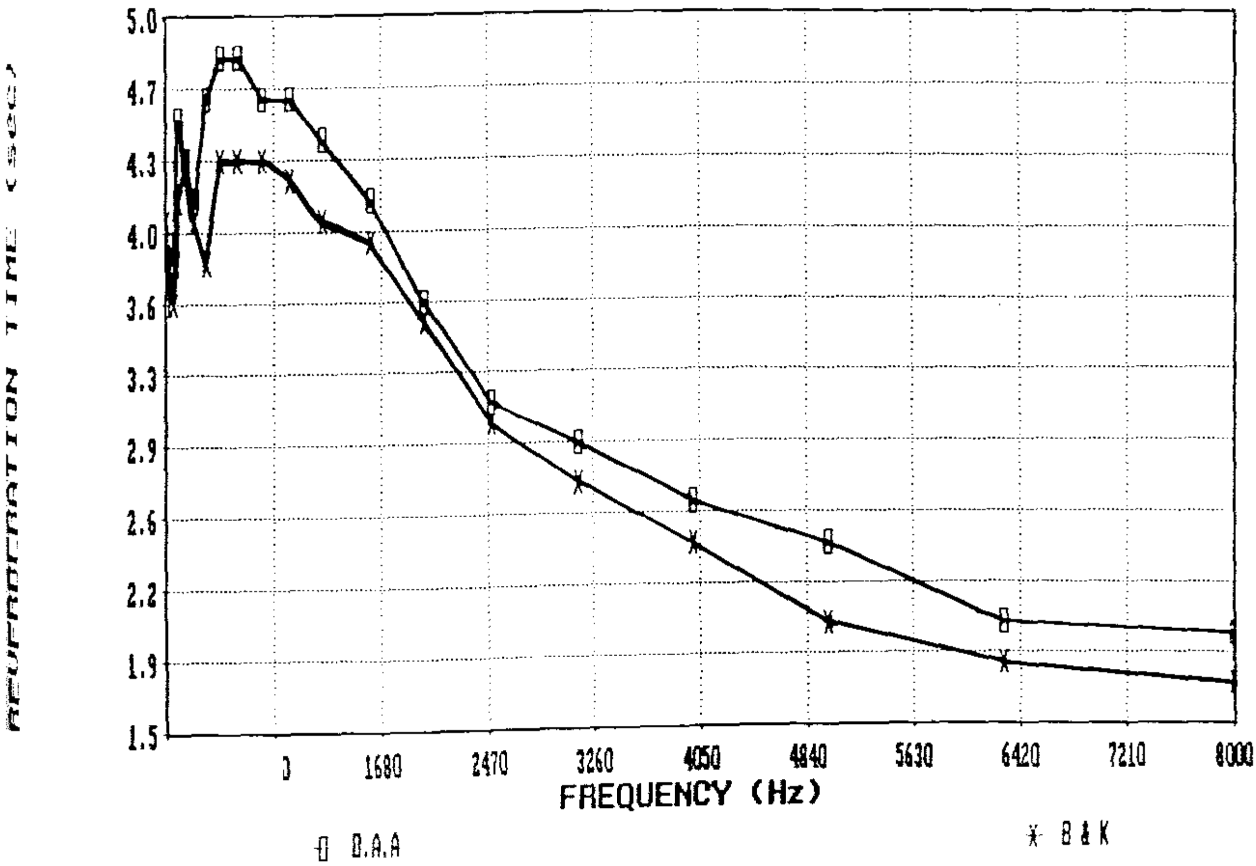


Figure 52: **COMPARISON OF R.T TIMES**



8. CONCLUSIONS

The most significant outcome of designing and building the BAA was the time-saving factor. The measurement times were reduced to approximately one quarter of the total time that the old equipment used. This relates to a 75 percent time saving which would be extremely valuable to acoustic consultants and also other users.

Another important factor concerns the accuracy of the data. The traditional methods of obtaining the data allowed for human error because of the subjective interpretation of the results. The BAA is a computer-based instrument, thus it is able to perform the measurements automatically and with precision. This is achieved because the traditional analog system is replaced by a combined analog and digital system.

A feature of the BAA is its portability and compactness in comparison with the old equipment which is quite bulky and unwieldy. This is especially convenient for field tests and where repetitive measurements are performed at various locations. However, to take full advantage of this feature, a Laptop type computer is required.

9. RECOMMENDATIONS

The following improvements are recommended for any further development of this system, based on the findings and conclusions of this thesis project.

1) The noise generator circuit should incorporate the digital noise source chip, the **MM5437,** to replace the discrete component white/pink noise source. Digitally generated noise is linear to within ± 0.25 dB or less and the output amplitude can be precisely controlled.

2) It would be useful to have a one octave facility in the bandpass filter section. This would allow non-standard sound measurements to be performed in a much shorter time span. The bandpass filters of the B.A.A have the facility for this modification, but due to lack of time, was not completed. The correct components for the filter circuit need to be worked out, while the existing programming codes need only be adjusted or tuned for the correct filter characteristics. The software would require some modifications to enable the octave filter to be selected.

3) The bandpass filters should preferably be designed and built using discrete components. The clock generator circuit would then not be required, removing the ± 2mV noise floor caused by clock signal breakthrough and increase the dynamic range of the B.A.A. The other option would be to have a computer programmable filter module constructed, at a cost of ± R1 500.

4) The portability would be improved by incorporating a rechargeable battery power source. This would allow the user to use the instrument where mains power is not available.

BIBLIOGRAPHY

Borland International Inc., 1989
Turbo Pascal User and Reference Manuals - Version 5.5

Brüel & Kjaer
Frequency Analysis.
Brüel & Kjaer Equipment Application.

Burr-Brown Corporation, 1988
The Handbook of Personal Computer Instrumentation.
Arizona, U.S.A.

Coppens, A., Frey, A., Kinsler, L., Sanders, J., 1982
Fundamentals of Acoustics.
3rd Edition.
J. Wiley & Sons.

Cowell, J.R., Humphreys, H., Parkin, P., 1979
Acoustics, Noise and Buildings.
Faber and Faber Ltd., London.

Ginn, K.V., 1978
Architectural Acoustics.
2nd Edition.
Brüel & Kjaer, Denmark.

Graf, E.R., Irwin, J.D., 1979
Industrial Noise and Vibration Control.

Hewlett Packard
H.P. Multiprogrammer Reference Manuals.

International Electrotechnical Commission, 1989
Octave Band and Fractional Octave Band Filters - IEC 225.

International Electrotechnical Commission, 1979
Sound Level Meters - IEC 651.

International Standards Organisation, 1980
Measurement of Sound Insulation in Buildings and of Building Elements - ISO 140, parts 1, 2 & 4.

International Standards Organisation, 1984
Rating the Sound Insulation in Buildings and of Building Elements - ISO 717, parts 1, 2 & 3.

Lord, P., Templeton, D., 1986
Architecture of Sound.
Architectural Press of Oxford.

Owen, S., Peters, R.J., Smith, B.J., 1982
Acoustics and Noise Control.

Rettinger, M., 1977
Acoustical Design and Noise Control.
Vol 1 - Acoustical Design.

Rettinger, M., 1977
Acoustical Design and Noise Control.
Vol 2 - Noise Control.

Reynolds, D.D., 1981
Engineering Principles of Acoustics.
Allyn and Bacon Book.

Tucker, B., 1984
A Microcomputer Based R.T Measurement Facility.
C.A.L. Technical Memo.

White, M.C., 1990
Undergraduate Thesis of the Cape Town City Hall.

Yester, M., 1989
Using Turbo Pascal.
Que Corporation, U.S.A.

LIST OF REFERENCES

Analog Devices Inc, 1984

Analog Devices Linear Design Seminar.

Analog Devices, England.

Beranek, L, 1962

Acoustic Measurements.

John Wiley and Sons Inc., New York.

Horowitz & Hill, 1989

The Art of Electronics.

2nd Edition.

Cambridge University Press, Cambridge.

Institute of Acoustics, 1990

In-Situ Measurements.

Newsletter of the International Institute of Noise Control Engineering, Vol 58, pg 11.

International Standards Organisation, 1980

Measurement of Sound Insulation in Buildings and of Building Elements - ISO 140, part 3.

International Standards Organisation, 1985

Measurement of Absorption Coefficients in a Reverberation Room - ISO R354.

Jordan, V.L. , 1981

Acoustical Design of Concert Halls and Theatres.

Applied Science Publishers, New York.

Maxim Integrated Products

Microprocessor Programmable Active Filters.

Filter types MAX260/261/262

LIST OF APPENDICES

APPENDIX 1

-1/2 Oct. (Hz)	-1/6 Oct. (Hz)	Center Freq. (HZ)	+1/6 Oct. (Hz)	+1/2 Oct. (Hz)
70.77	89.09	**100.00**	112.3	141.3
89.09	112.3	**125.89**	141.3	177.8
112.3	141.3	**158.49**	177.8	233.9
141.3	177.8	**199.53**	233.9	281.8
177.8	223.9	**251.16**	281.8	354.5
233.9	281.8	**316.23**	354.5	446.6
281.8	354.5	**398.1**	446.6	562.1
354.5	446.6	**501.2**	562.1	707.7
446.6	562.1	**630.96**	707.7	890.9
562.1	707.7	**794.33**	890.9	1 123
707.7	890.9	**1 000**	1 123	1 413
890.9	1 123	**1 258.9**	1 413	1 778
1 123	1 413	**1 584.9**	1 778	2 239
1 413	1 770	**1 995.3**	2 239	2 818
1 778	2 239	**2 511.6**	2 818	3 545
2 239	2 818	**3 162.3**	3 545	4 466
2 818	3 545	**3 981.1**	4 466	5 621
3 545	4 466	**5 011.8**	5 621	7 077
4 466	5 621	**6 309.6**	7 077	8 909
5 621	7 077	**7 943.3**	8 909	11 230

Table of exact frequencies calculated from IEC 225 specifications.

APPENDIX 2

Frequency (Hz)	Attenuation (dB)	Frequency (Hz)	Attenuation (dB)
20	0.21	800	0.0
31.5	0.15	1 000	0.0
40	0.11	1 250	0.0
50	0.07	1 600	0.0
63	0.03	2 000	0.01
80	0.002	2 500	0.01
100	0.0	3 150	0.06
125	0.0	4 000	0.1
160	0.0	5 000	0.16
200	0.0	6 300	0.23
250	0.0	8 000	0.29
315	0.0	10 000	0.3
400	0.0	12 500	0.17
500	0.0	16 000	0.05
630	0.0	20 000	0.32

Table of low-pass filter response values obtained from linearity test on the filter.

APPENDIX 3

Table 2 - Attenuation tolerance limits for octave band and one-third octave band filters

Relative frequency ratio f/f_m or f_m/f				Tolerance limits, in decibels, for the relative filter attenuation $\Delta A(f)$		
Octave band filters		One-third ocatve band filters				
Base 10	Base 2	Base 10	Base 2	Class 0	Class 1	Class 2
1	1	1	1	-0,15 +0,15	-0,3 +0,3	-0,5 +0,5
$10^{3/80}$	$2^{1/8}$	$10^{1/80}$	$2^{1/24}$	-0,15 +0,2	-0,3 +0,4	-0,5 +0,6
$10^{3/40}$	$2^{1/4}$	$10^{1/40}$	$2^{1/12}$	-0,15 +0,4	-0,3 +0,6	-0,5 +0,8
$10^{9/80}$	$2^{3/8}$	$10^{3/80}$	$2^{3/24}$	-0,15 +1,1	-0,3 +1,3	-0,5 +1,6
$10^{3/20}$	$2^{1/2}$	$10^{1/20}$	$2^{1/6}$	+2,3 +4,5	+2,0 +5,0	+1,6 +5.5
$10^{3/10}$	2^{1}	$10^{1/10}$	$2^{1/3}$	+18,0 +50	+17,5 +50	+16,5 +50
-	-	$10^{3/10}$	2^{1}	+47,5 $+\infty$	+47 $+\infty$	+46 $+\infty$
-	-	$10^{6/10}$	2^{2}	+71 $+\infty$	+70 $+\infty$	+69 $+\infty$
$10^{6/10}$	2^{2}	-	-	+42,5 $+\infty$	+42 $+\infty$	+41 $+\infty$
$10^{9/10}$	2^{3}	-	-	+62 $+\infty$	+61 $+\infty$	+60 $+\infty$
*$10^{12/10}$	2^{4}	-	-	+80 $+\infty$	+79 $+\infty$	+78 $+\infty$
*$10^{15/10}$	2^{5}	-	-	+97 $+\infty$	+95 $+\infty$	+78 $+\infty$

*For the computation of tolerance limits for fractional-octave band filters only

APPENDIX 4

third_octave
1/1/80 0:13 1000 PZ<1.0>

FILTER TYPE: BUTTERWORTH BANDPASS

Specified Parameters:
Pass Band Low Edge is 890.9000 Pass Band High Edge is 1123.0000
Stop Band Low Edge is 500.1923 Stop Band High Edge is 2000.1923
Order= 8 N= 4 Amax= 0.5000 db

Calculated Parameters:
Center Frequency is 1000.2403 Pass Bandwidth is 232.1000
Stop Bandwidth is 1500.0000
Amin=55.70 db

Data for Lowpass Prototype, Normalized at 1 rad/sec:

FC= 1.0000 FS= 6.4627

Pole	Q
1.3000	0.5412
1.3000	1.3066

Data for final design of BUTTERWORTH BANDPASS Filter

Quadratic Coefficients for Denormalized Gain Function in Rad/Sec

Numerator			Denominator		
s^2	s^1	s^0	s^2	s^1	s^0
0.0	1.650E+03	0.000E+00	1.0	1.650E+03	3.515E+07
0.0	1.853E+03	0.000E+00	1.0	1.853E+03	4.438E+07
0.0	6.252E+02	0.000E+00	1.0	6.252E+02	2.990E+07
0.0	8.258E+02	0.000E+00	1.0	8.258E+02	5.217E+07

Resonant Frequencies in Hertz:

Pole	Q
943.6381	3.5943
1060.2377	3.5943
870.3282	8.7466
1149.5442	8.7466

Special note on gain of bandpass :
Gain adjustment factor for bandpass = GAF = 8.184
For bandpass filters, the gain of the system at center frequency is given in terms of the gains of the individual stages by the expression:
Bandpass gain at center frequency= HO_BP_1*HO_BP_2* . . . *HO_BP_N / GAF
The above individual stage gains are defined at the center frequency of each stage, which may be different from the center frequency of the overal system. The gain adjustment factor is used to arrive at the correct bandpass gain when defined at the overall system center frequency.

third_octave
1/1/80 0:16 1000 MPP<1.0>
Maxim Program MPP: Calculations for Mode 1

DATA INPUT FOR SECTION #1
Specified center frequency, f0, is 943.6381
Specified Q is 3.5943
Specified clock frequency is 90000.0000

FO CODE REQUIRED FOR SECTION #1
Code for clock ratio: N = 35 = 1 0 0 0 1 1

Q CODE REQUIRED FOR SECTION #1
Code for Q selection: N = 110 = 1 1 0 1 1 1 0

COMMENT ON SECTION #1
Above code applies to parts with a SIX BIT clock/center freq ratio code
(code for parts with ratio range of 40-141, except mode 2 with 28-99)
Value of center frequency 'targeted' was 939.2751
Actual center frequency delivered by the circuit will be 944.5727
Clock ratio dialed in is 95.8186
The Q 'targeted' was 3.6098
Value of Q delivered by the circuit is 3.5398
Following are gains at filter terminals:
HO_LP = -1.0, HO_N1 = -1.0 (freq = 0 Hz), HO_N2 = -1.0 (freq = clk/2)
HO_BP = -Q = -3.5398

DATA INPUT FOR SECTION #2
Specified center frequency, f0, is 1060.2377
Specified Q is 3.5943
Specified clock frequency is 90000.0000

FO CODE REQUIRED FOR SECTION #2
Code for clock ratio: N = 28 = 0 1 1 1 0 0

Q CODE REQUIRED FOR SECTION #2
Code for Q selection: N = 110 = 1 1 0 1 1 1 0

COMMENT ON SECTION #2
Above code applies to parts with a SIX BIT clock/center freq ratio code
(code for parts with ratio range of 40-141, except mode 2 with 28-99)
Value of center frequency 'targeted' was 1061.0330
Actual center frequency delivered by the circuit will be 1067.6088
Clock ratio dialed in is 84.8230
The Q 'targeted' was 3.6150
Value of Q delivered by the circuit is 3.5347
Following are gains at filter terminals:
HO_LP = -1.0, HO_N1 = -1.0 (freq = 0 Hz), HO_N2 = -1.0 (freq = clk/2)
HO_BP = -Q = -3.5347

```
------------------------------------------------------------------------------------
ıird_octave
/1/80 0:17     1000                                                      MPP<1.0>
axim Program MPP: Calculations for Mode 1
------------------------------------------------------------------------------------
                      DATA INPUT FOR SECTION #1
pecified center frequency, f0, is        870.3282
pecified Q is              8.7466
pecified clock frequency is      90000.0000
------------------------------------------------------------------------------------
                    FO CODE REQUIRED FOR SECTION #1
ode for clock ratio: N =  40 = 1 0 1 0 0 0
------------------------------------------------------------------------------------
                    Q CODE REQUIRED FOR SECTION #1
ode for Q selection: N =  121 = 1 1 1 1 0 0 1
------------------------------------------------------------------------------------
                           COMMENT ON SECTION #1
bove code applies to parts with a SIX BIT clock/center freq ratio code
 code for parts with ratio range of 40-141, except mode 2 with 28-99 )
alue of center frequency 'targeted' was          868.1179
ctual center frequency delivered by the circuit will be        869.5878
lock ratio dialed in is           103.6726
he Q 'targeted' was            8.7595
alue of Q delivered by the circuit is          9.1296
ollowing are gains at filter terminals:
0_LP = -1.0, HO_N1 = -1.0 ( freq = 0 Hz ), HO_N2 = -1.0 (freq = clk/2 )
0_BP = -Q =  -9.1296
------------------------------------------------------------------------------------
                      DATA INPUT FOR SECTION #2
pecified center frequency, f0, is       1149.5442
pecified Q is              8.7466
pecified clock frequency is      90000.0000
------------------------------------------------------------------------------------
                    FO CODE REQUIRED FOR SECTION #2
ode for clock ratio: N =  24 = 0 1 1 0 0 0
------------------------------------------------------------------------------------
                    Q CODE REQUIRED FOR SECTION #2
ode for Q selection: N =  121 = 1 1 1 1 0 0 1
------------------------------------------------------------------------------------
                           COMMENT ON SECTION #2
bove code applies to parts with a SIX BIT clock/center freq ratio code
 code for parts with ratio range of 40-141, except mode 2 with 28-99 )
alue of center frequency 'targeted' was         1145.9156
ctual center frequency delivered by the circuit will be       1148.3302
lock ratio dialed in is            78.5398
he Q 'targeted' was            8.7811
alue of Q delivered by the circuit is          9.1071
ollowing are gains at filter terminals:
0_LP = -1.0, HO_N1 = -1.0 ( freq = 0 Hz ), HO_N2 = -1.0 (freq = clk/2 )
0_BP = -Q =  -9.1071
```

third_octave
1/1/80 0:17 1000 MPP<1.0>
Maxim Program MPP: Calculations for Mode 1

DATA INPUT FOR SECTION #1
Specified center frequency, f0, is 870.3282
Specified Q is 8.7466
Specified clock frequency is 90000.0000

F0 CODE REQUIRED FOR SECTION #1
Code for clock ratio: N = 40 = 1 0 1 0 0 0

Q CODE REQUIRED FOR SECTION #1
Code for Q selection: N = 121 = 1 1 1 1 0 0 1

COMMENT ON SECTION #1
Above code applies to parts with a SIX BIT clock/center freq ratio code
(code for parts with ratio range of 40-141, except mode 2 with 28-99)
Value of center frequency 'targeted' was 868.1179
Actual center frequency delivered by the circuit will be 869.5878
Clock ratio dialed in is 103.6726
The Q 'targeted' was 8.7595
Value of Q delivered by the circuit is 9.1296
Following are gains at filter terminals:
HO_LP = -1.0, HO_N1 = -1.0 (freq = 0 Hz), HO_N2 = -1.0 (freq = clk/2)
HO_BP = -Q = -9.1296

DATA INPUT FOR SECTION #2
Specified center frequency, f0, is 1149.5442
Specified Q is 8.7466
Specified clock frequency is 90000.0000

F0 CODE REQUIRED FOR SECTION #2
ode for clock ratio: N = 24 = 0 1 1 0 0 0

Q CODE REQUIRED FOR SECTION #2
ode for Q selection: N = 121 = 1 1 1 1 0 0 1

COMMENT ON SECTION #2
bove code applies to parts with a SIX BIT clock/center freq ratio code
code for parts with ratio range of 40-141, except mode 2 with 28-99)
alue of center frequency 'targeted' was 1145.9156
ctual center frequency delivered by the circuit will be 1148.3302
lock ratio dialed in is 78.5398
he Q 'targeted' was 8.7811
alue of Q delivered by the circuit is 9.1071
ollowing are gains at filter terminals:
O_LP = -1.0, HO_N1 = -1.0 (freq = 0 Hz), HO_N2 = -1.0 (freq = clk/2)
O_BP = -Q = -9.1071

APPENDIX 5

Frequency (Hz)	B A.A (dB)	B & K 1625 (dB)	Tolerance (dB)
100.00	0.08	0.04	-0.5, 0.5
97.15	0.28	0.04	-0.5, 0.6
94.4	0.52	0.14	-0.5, 0.8
91.7	-0.16	1.16	-0.5, 1.6
89.1	2.26	4.12	1.6, 5.5
79.37	17.9	19.8	16.5, 50
50.0	53.0	49.6	46.0, ∞
25.0	60.0	59.8	69.0, ∞
102.93	0.25	0.07	-0.5, 0.6
105.95	0.11	0.13	-0.5, 0.8
109.1	-0.23	0.86	-0.5, 1.6
112.2	1.62	3.32	1.6, 5.5
126.0	19.06	19.16	16.5, 50
200.0	56.3	49.3	46.0, ∞
400.0	59.8	59.8	69.0, ∞

Comparison of filter responses with the tolerances from the IEC 225 Standard. The frequency in bold (100 Hz) is the lowest center frequency of the bandpass filter module.

The frequency in bold (8000 Hz) is the highest center frequency of the bandpass filter module.

Frequency (Hz)	B.A.A (dB)	B & K 1625 (dB)	Tolerance (dB)
8000.0	-0.28	0.09	-0.5, 0.5
7772.0	-0.38	0.1	-0.5, 0.6
7552.0	-0.33	0.14	-0.5, 0.8
7336.0	0.57	0.57	-0.5, 1.6
7128.0	2.65	2.57	1.6, 5.5
6349.6	18.04	18.4	16.5, 50
4000.0	49.44	49.2	46.0, ∞
2000.0	60.0	59.8	69.0, ∞
8234.4	0.25	0.09	-0.5, 0.6
8476.0	0.11	0.31	-0.5, 0.8
8728.0	-0.23	1.77	-0.5, 1.6
8976.0	1.62	5.09	1.6, 5.5
10080	19.06	20.7	16.5, 50
16000	56.3	49.7	46.0, ∞
32000	59.8	59.7	69.0, ∞

APPENDIX 6

Nominal Freq (Hz)	C-Weighting (dB)	A-Weighting (dB)	Tolerance (dB)
10	-14.6(-14.3)	-58.1(-70.4)	+3,- ∞
12.5	-11.6(-11.2)	-57.1(-63.4)	+3,- ∞
16	- 8.7(-8.5)	-54.0(-56.7)	+3,- ∞
20	- 6.41(-6.2)	-49.3(-50.5)	± 3
25	- 4.62(-4.4)	-44.3(-44.7)	± 2
31.5	- 3.18(-3)	-39.25(-39.4)	± 1.5
40	- 2.14(-2)	-34.3(-34.6)	± 1.5
50	- 1.5(-1.3)	-30.2(-30.2)	± 1.5
63	- 0.95(-0.8)	-26.0(-26.2)	± 1.5
80	- 0.62(-0.5)	-22.2(-22.5)	± 1.5
100	- 0.42(-0.3)	-19.0(-19.1)	± 1
125	- 0.28(-0.2)	-16.04(-16.1)	± 1
160	- 0.19(-0.1)	-13.1(-13.4)	± 1
200	- 0.14(0)	-10.7(-10.9)	± 1
250	- 0.11(0)	- 8.6(-8.6)	± 1
315	- 0.08(0)	- 6.6(-6.6)	± 1
400	- 0.08(0)	- 4.7(-4.8)	± 1
500	- 0.08(0)	- 3.25(-3.2)	± 1
630	- 0.08(0)	- 1.94(-1.9)	± 1
800	- 0.11(0)	- 0.89(-0.8)	± 1
1000	- 0.14(0)	- 0.14(0)	± 1
1250	- 0.19(0)	0.37(0.6)	± 1
1600	- 0.28(-0.1)	0.75(1.0)	± 1
2000	- 0.4(-0.2)	0.93(1.2)	± 1
2500	- 0.53(-0.3)	0.95(1.3)	± 1
3150	- 0.77(-0.5)	0.85(1.2)	± 1
4000	- 1.13(-0.8)	0.6(1.0)	± 1
5000	- 1.61(-1.3)	0.16(0.5)	± 1.5
6300	- 2.32(-2)	- 0.5(-0.1)	± 1.5,-2
8000	- 3.33(-3)	- 0.95(-1.1)	+1.5,-3
10000	- 4.67(-4.4)	- 2.81(-2.5)	+2,-4
12500	- 6.35(-6.2)	- 4.5(-4.3)	+3,-6
16000	- 8.67(-8.5)	- 6.82(-6.6)	+3,- ∞
20000	-11.17(-11.2)	- 9.3(-9.3)	+3,- ∞

References values in bold.

APPENDIX 7

Specification sheets of A/D Converter

ANALOG DEVICES

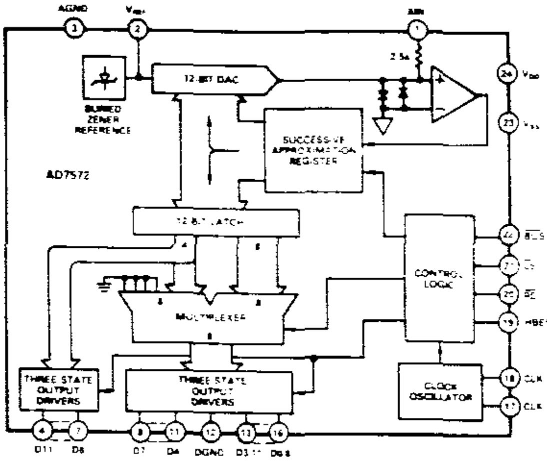
LC²MOS Complete, High-Speed 12-Bit ADC

AD7572

FEATURES

12-Bit Resolution and Accuracy
Fast Conversion Time
AD7572XX05: 5μs
AD7572XX12: 12.5μs
Complete with On-Chip Reference
Fast Bus Access Time: 90ns
Low Power: 135mW
Small, 0.3", 24-pin Package

AD7572 FUNCTIONAL BLOCK DIAGRAM



GENERAL DESCRIPTION

The AD7572 is a complete, 12-bit ADC that offers high-speed performance combined with low CMOS power levels. The AD7572 uses an accurate, high-speed DAC and comparator in a successive-approximation loop to achieve a fast conversion time. An on-chip, buried zener diode provides a stable reference voltage to give low drift performance over the full temperature range and the specified accuracy is achieved without any user trims. An on-chip clock circuit is provided, which may be used with a crystal for stand-alone operation, or the clock input may be driven from an external clock source such as a divided-down microprocessor clock. The only other external components required for basic operation of the AD7572 are decoupling capacitors for the supply voltages and reference output.

The AD7572 has a high-speed digital interface with three-state data outputs and can operate under the control of standard microprocessor Read ($\overline{RD}$) and decoded address ($\overline{CS}$) signals. Interface timing is sufficiently fast to allow the AD7572 to operate with most popular microprocessors, with three-state enable times of only 90ns and bus relinquish times of 75ns.

The AD7572 is fabricated in Analog Devices Linear Compatible CMOS process (LC²MOS), an advanced, all ion-implanted process that combines fast CMOS logic and linear, bipolar circuits on a single chip, thus achieving excellent linear performance while still retaining low CMOS power levels.

The AD7572 is available in both 0.3" wide, 24-pin DIPs and in 28-terminal surface mount packages.

PRODUCT HIGHLIGHTS

1. Fast, 5μs and 12.5μs conversion times make the AD7572 ideal for a wide range of applications in telecommunications, sonar and radar signal processing or any wideband data acquisition system.
2. On-chip buried-zener reference has temperature coefficient as low as 25ppm/°C, giving low full-scale drift over the operating temperature range.
3. Stable DAC and comparator give excellent linearity and low zero error over the full temperature range.
4. Fast, easy-to-use digital interface has three-state bus access times of 90ns and bus relinquish times of 75ns, allowing the AD7572 to interface to most popular microprocessors.
5. LC²MOS circuitry gives low power drain (135mW) from +5, −15 volt supplies.
6. 24-pin 0.3" package offers space saving over parts in 28-pin 0.6" DIP.

SPECIFICATIONS

(V_{DD} = 5V ±5%, V_{SS} = −15V ±5%, f_{CLK} 2.5MHz for AD7572XX05, 1MHz for AD7572XX12. All Specifications T_{min} to T_{max} unless otherwise noted. Specifications apply to Slow Memory Mode).

Parameter	J, A, S Versions[1]	K, B, T Versions	L, U Versions	U Version	Units	Test Conditions/Comments
ACCURACY						
Resolution	12	12	12	12	Bits	
Integral Nonlinearity @ +25°C	±1	±1	±1/2	±1/2	LSB max	
T_{min} to T_{max}	±1	±1	±1/2	±3/4	LSB max	
Differential Nonlinearity	±1	±1	±1	±1	LSB max	
Minimum Resolution for which no Missing Codes are Guaranteed	12	12	12	12	Bits	
Offset Error @ +25°C	±4	±3	±3	±3	LSB max	
T_{min} to T_{max}	±6	±5	±4	±4	LSB max	Typical Change over Temp is ±1LSB
Full Scale (FS) Error[2] @ +25°C	±15	±10	±10	±10	LSB max	V_{DD} = 5V; V_{SS} = −15V; FS = 5V
Full Scale TC[3,4]	45	25	25	25	ppm/°C max	Ideal Last Code Transition = FS − 3/2LSBs
ANALOG INPUT						
Input Voltage Range	0 to +5	0 to +5	0 to +5	0 to +5	Volts	For Bipolar Operation See
Input Current	3.5	3.5	3.5	3.5	mA max	Figures 10 & 12
INTERNAL REFERENCE VOLTAGE						
V_{REF} Output @ +25°C	−5.2/−5.3	−5.2/−5.3	−5.2/−5.3	−5.2/−5.3	V min/V max	−5.25V ±1%
V_{REF} Output TC	40	20	20	20	ppm/°C typ	
Output Current Sink Capability	550	550	550	550	μA max	External Load Should Not Change During Conversion
POWER SUPPLY REJECTION						
V_{DD} Only [*]	±1/2	±1/2	±1/2	±1/2	LSB typ	FS Change, V_{SS} = −15V, V_{DD} = +4.75V to +5.25V
V_{SS} Only	±1/2	±1/2	±1/2	±1/2	LSB typ	FS Change, V_{DD} = 5V, V_{SS} = −14.25V to −15.75V
LOGIC INPUTS						
$\overline{CS}$, $\overline{RD}$, HBEN, CLK IN						
V_{INL}, Input Low Voltage	+0.8	+0.8	+0.8	+0.8	V max	V_{DD} = 5V ±5%
V_{INH}, Input High Voltage	+2.4	+2.4	+2.4	+2.4	V min	
C_{IN},[5] Input Capacitance	10	10	10	10	pF max	
$\overline{CS}$, $\overline{RD}$, HBEN						
I_{IN}, Input Current	±10	±10	±10	±10	μA max	V_{IN} = 0 to V_{DD}
CLK IN						
I_{IN}, Input Current	±20	±20	±20	±20	μA max	V_{IN} = 0 to V_{DD}
LOGIC OUTPUTS						
D11–D0/8, $\overline{BUSY}$, CLK OUT						
V_{OL}, Output Low Voltage	+0.4	+0.4	+0.4	+0.4	V max	I_{SINK} = 1.6mA
V_{OH}, Output High Voltage	+4.0	+4.0	+4.0	+4.0	V min	I_{SOURCE} = 200μA
D11–D0/8						
Floating State Leakage Current	±10	±10	±10	±10	μA max	
Floating State Output Capacitance[5]	15	15	15	15	pF max	
CONVERSION TIME						
AD7572XX05						
Synchronous Clock	5	5	5	5	μs max	f_{CLK} = 2.5MHz. See Under
Asynchronous Clock	4.8/5.2	4.8/5.2	4.8/5.2	4.8/5.2	μs min/max	Control Inputs Synchronization
AD7572XX12						
Synchronous Clock	12.5	12.5	12.5	12.5	μs max	f_{CLK} = 1MHz
Asynchronous Clock	12/13	12/13	12/13	12/13	μs min/μs max	
POWER REQUIREMENTS						
V_{DD}	+5	+5	+5	+5	V NOM	±5% for Specified Performance
V_{SS}	−15	−15	−15	−15	V NOM	±5% for Specified Performance
I_{DD}[6]	7	–	7	7	mA max	$\overline{CS}$ = $\overline{RD}$ = V_{DD}, AIN = 5V
I_{SS}[6]	12	12	12	12	mA max	$\overline{CS}$ = $\overline{RD}$ = V_{DD}, AIN = 5V
Power Dissipation	135	135	135	135	mW typ	
	215	215	215	215	mW max	

NOTES

[1] Temperature range as follows: J, K, L Versions, 0 to +70°C.
A, B, C Versions; −25°C to +85°C.
S, T, U Versions; −55°C to +125°C.

[2] Includes internal voltage reference error.

[3] Full-Scale TC = ΔFS/ΔT, where ΔFS is Full-Scale change from T_A = +25°C to T_{min} or T_{max}.

[4] Includes internal voltage reference drift.

[5] Sample tested to ensure compliance.

[6] Power supply current is measured when AD7572 is inactive, i.e., $\overline{CS}$ = $\overline{RD}$ = $\overline{BUSY}$ = HIGH.

Specifications subject to change without notice.

IMING CHARACTERISTICS[1] (V_{DD} = 5V, V_{SS} – 15V)

ırameter	Limit at +25°C (All Grades)	Limit at T_{min}, T_{max} (J, K, L, A, B, C Grades)	Limit at T_{min}, T_{max} (S, T, U Grades)	Units	Conditions/Comments
	0	0	0	ns min	$\overline{CS}$ to $\overline{RD}$ Setup Time
	190	230	270	ns max	$\overline{RD}$ to $\overline{BUSY}$ Propagation Delay
2	90	110	120	ns max	Data Access Time after $\overline{RD}$, C_L = 20pF
	125	150	170	ns max	Data Access Time after $\overline{RD}$, C_L = 100pF
	t_3	t_3	t_3	ns min	$\overline{RD}$ Pulse Width
	0	0	0	ns min	$\overline{CS}$ to $\overline{RD}$ Hold Time
2	70	90	100	ns max	Data Setup Time after $\overline{BUSY}$
3	20	20	20	ns min	Bus Relinquish Time
	75	85	90	ns max	
	0	0	0	ns min	HBEN to $\overline{RD}$ Setup Time
	0	0	0	ns min	HBEN to $\overline{RD}$ Hold Time
0	200	200	200	ns min	Delay Between Successive Read Operations

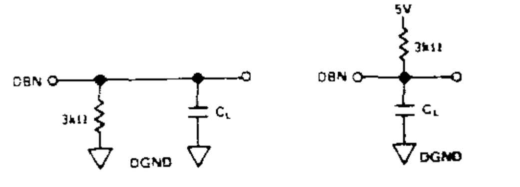
OTES

Timing Specifications are sample tested at +25°C to ensure compliance. All input control signals are specified with r = tf = 5ns (10% to 90% of +5V) and timed from a voltage level of 1.6V.

$_3$ and t_4 are measured with the load circuits of Figure 1 and defined as the time required for an output to cross 0.8V or 2.4V.

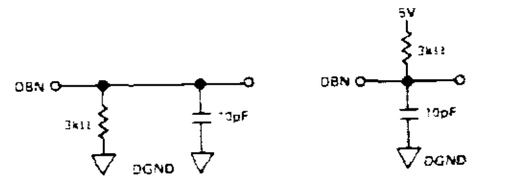
$_7$ is defined as the time required for the data lines to change 0.5V when loaded with the circuits of Figure 2.

pecifications subject to change without notice.



a. High-Z to V_{OH} (t_3) and V_{OL} to V_{OH} (t_4)   b. High-Z to V_{OL} (t_3) and V_{OH} to V_{OL} (t_4)

Figure 1. Load Circuits for Access Time



a. V_{OH} to High-Z   b. V_{OL} to High-Z

Figure 2. Load Circuits for Output Float Delay

ABSOLUTE MAXIMUM RATINGS*

(T_A = +25°C unless otherwise noted)

V_{DD} to DGND . . . . –0.3V to +7V
V_{SS} to DGND . . . . +0.3V to –17V
AGND to DGND . . . . –0.3V, V_{DD} + 0.3V
AIN to AGND . . . . –15V to +15V
Digital Input Voltage to DGND . . . . –0.3V, V_{DD} +0.3V
Digital Output Voltage to DGND . . . –0.3V, V_{DD} +0.3V
Operating Temperature Range
Commercial (J, K, L Versions) . . . . 0 to +70°C
Industrial (A, B, C Versions) . . . . –25°C to +85°C
Extended (S, T, U Versions) . . . . –55°C to +125°C
Storage Temperature . . . . –65°C to +150°C
Lead Temperature (Soldering, 10secs) . . . . +300°C
Power Dissipation (Any Package) to +75°C 1,000mW
Derates above +75°C by . . . . 10mW/°C

*Stress above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

CAUTION:

ESD (Electro-Static-Discharge) sensitive device. The digital control inputs are diode protected; however, permanent damage may occur on unconnected devices subjected to high energy electrostatic fields. Unused devices *must* be stored in conductive foam or shunts. The foam should be discharged to the destination socket before devices are removed.

WARNING!
ESD SENSITIVE DEVICE

ORDERING INFORMATION[1]

CONVERSION TIME = 5μs

Accuracy Grade	Temperature Range and Package Options[2]		
	0 to +70°C	−25°C to +85°C	−55°C to +125°C
	Plastic DIP (N-24)	Hermetic[3] (Q-24)	Hermetic[3] (Q-24)
±1LSB	AD7572JN05	AD7572AQ05	AD7572SQ05
±1LSB	AD7572KN05	AD7572BQ05	AD7572TQ05
±1/2LSB	AD7572LN05	AD7572CQ05	AD7572UQ05
	PLCC[4] (P-28A)		LCCC[5] (E-28A)
±1LSB	AD7572JP05		AD7572SE05
±1LSB	AD7572KP05		AD7572TE05
±1/2LSB	AD7572LP05		AD7572UE05

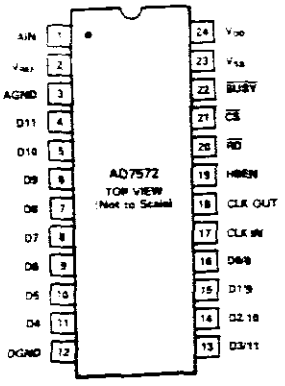
CONVERSION TIME = 12μs

Accuracy Grade	Temperature Range and Package Options[2]		
	0 to +70°C	−25°C to +85°C	−55°C to +125°C
	Plastic DIP (N-24)	Hermetic[3] (Q-24)	Hermetic[3] (Q-24)
±1LSB	AD7572JN12	AD7572AQ12	AD7572SQ12
±1LSB	AD7572KN12	AD7572BQ12	AD7572TQ12
±1/2LSB	AD7572LN12	AD7572CQ12	AD7572UQ12
	PLCC[4] (P-28A)		LCCC[5] (E-28A)
±1LSB	AD7572JP12		AD7572SE12
±1LSB	AD7572KP12		AD7572TE12
±1/2LSB	AD7572LP12		AD7572UE12

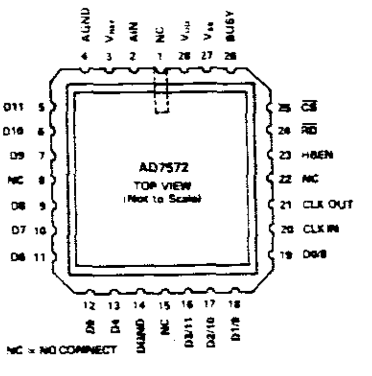
NOTES
[1]To order MIL-STD-883, Class B processed parts, add 883B to part number. Contact your local sales office for military data sheet.
[2]See Section 13 for package outline information.
[3]Analog Devices reserves the right to ship either ceramic (package outline D-24A) or cerdip hermetic (package outline Q-24) packages.
[4]PLCC: Plastic Leaded Chip Carrier.
[5]LCCC: Leadless Ceramic Chip Carrier.

PIN CONFIGURATIONS

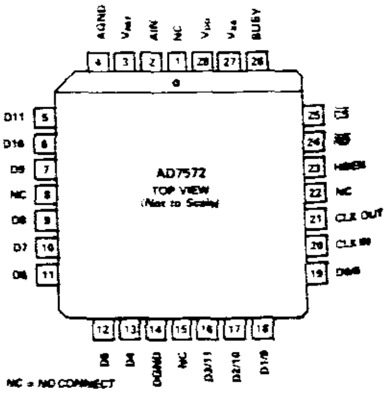
DIP



LCCC



PLCC



PIN FUNCTION DESCRIPTION (DIP PACKAGE)

PIN	MNEMONIC	DESCRIPTION
1	AIN	Analog Input.
2	V_{REF}	Voltage Reference Output. The AD7572 has its own internal −5.25V reference.
3	AGND	Analog Ground.
4 . . . 11	D11 . . . D4	Three State data outputs. They become active when $\overline{CS}$ and $\overline{RD}$ are brought low.
13 . . . 16	D3/11 . . . D0/8	Individual pin function is dependent upon High Byte Enable (HBEN) Input.

DATA BUS OUTPUT, $\overline{CS}$ & $\overline{RD}$ = LOW

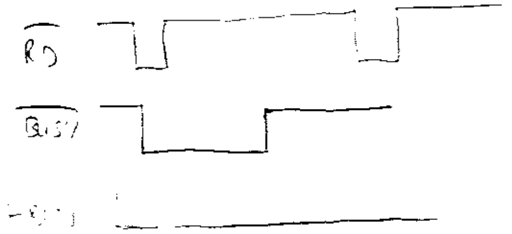
	Pin 4	Pin 5	Pin 6	Pin 7	Pin 8	Pin 9	Pin 10	Pin 11	Pin 13	Pin 14	Pin 15	Pin 16
MNEMONIC*	D11	D10	D9	D8	D7	D6	D5	D4	D3/11	D2/10	D1/9	D0/8
HBEN = LOW	DB11	DB10	DB9	DB8	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
HBEN = HIGH	DB11	DB10	DB9	DB8	LOW	LOW	LOW	LOW	DB11	DB10	DB9	DB8

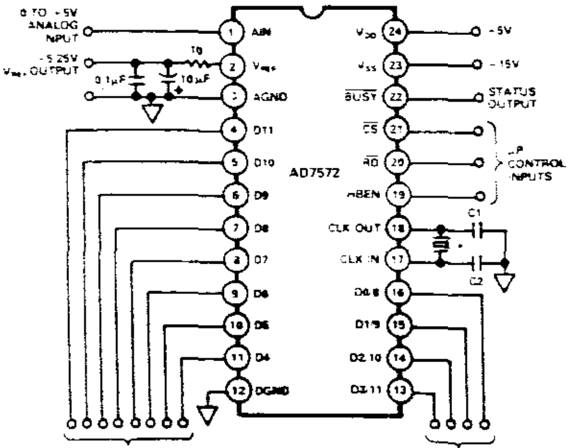
NOTE
*D11 . . . D0/8 are the ADC data output pins.
DB11 . . . DB0 are the 12-bit conversion results, DB11 is the MSB.

PIN	MNEMONIC	DESCRIPTION
12	DGND	Digital Ground.
17	CLK IN	Clock Input pin. An external TTL compatible clock may be applied to this pin. Alternatively a crystal or ceramic resonator may be connected between CLK IN (Pin 17) and CLK OUT (Pin 18).
18	CLK OUT	Clock Output Pin. An inverted CLK IN signal appears at CLK OUT when an external clock is used. See CLK IN (Pin 17) description for crystal (resonator).
19	HBEN	High Byte Enable input. Its primary function is to multiplex the 12-bits of conversion data onto the lower D7 . . . D0/8 outputs (4MSBs or 8 LSBs). See Pin description 4 . . . 11 and 13 . . . 16. *It also disables conversion start when HBEN is high.*
20	$\overline{RD}$	READ input. This active LOW signal, in conjunction with $\overline{CS}$ is used to enable the output data three state drivers and initiate a conversion if $\overline{CS}$ and HBEN are low.
21	$\overline{CS}$	CHIP SELECT Input. This active LOW signal, in conjunction with $\overline{RD}$ is used to enable the output data three state drivers and initiate a conversion if $\overline{RD}$ and HBEN are low.
22	$\overline{BUSY}$	$\overline{BUSY}$ output indicates converter status. $\overline{BUSY}$ is LOW during conversion.
23	V_{SS}	Negative Supply, −15V.
24	V_{DD}	Positive Supply, +5V.

OPERATIONAL DIAGRAM

An operational diagram for the AD7572 is shown in Figure 3. The AD7572 is a 12-bit successive approximation A/D converter. The addition of just a crystal/ceramic resonator and a few capacitors enables the device to perform the analog-to-digital function.





NOTES
AD7572XX05 - 2.5MHz CRYSTAL CERAMIC RESONATOR
AD7572XX12 - 1.0MHz CRYSTAL CERAMIC RESONATOR
C1 and C2 CAPACITANCE VALUES DEPEND ON CRYSTAL CERAMIC RESONATOR MANUFACTURER. TYPICAL VALUES ARE FROM 30 to 100pF

Figure 3. AD7572 Operational Diagram

CONVERTER DETAILS

Conversion start is controlled by the $\overline{CS}$, $\overline{RD}$ and HBEN inputs. At the start of conversion the successive approximation register (SAR) is reset and the three-state data outputs are enabled. Once a conversion cycle has begun it cannot be restarted.

During conversion, the internal 12-bit voltage mode DAC output is sequenced by the SAR from the most significant bit (MSB) to the least significant bit (LSB). Referring to Figure 4, the AIN input connects to the comparator input via 2.5kΩ. The DAC which has a similar 2.5kΩ output impedance connects to the same comparator input. Bit decisions are made by the comparator (zero crossing detector) which checks the addition of each successive weighted bit from the DAC output. The MSB decision is made 80ns (typically) after the second falling edge of CLK IN following a conversion start. Similarly, the succeeding bit decisions are made approximately 80ns after a CLK IN edge until conversion is finished. At the end of conversion, the DAC output current balances the AIN input current. The SAR contents (12-bit data word) which represent the AIN input signal is loaded into a 12-bit latch.

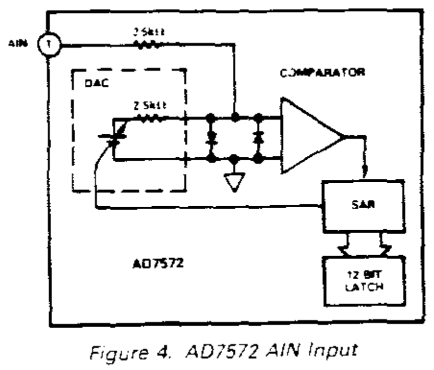


Figure 4. AD7572 AIN Input

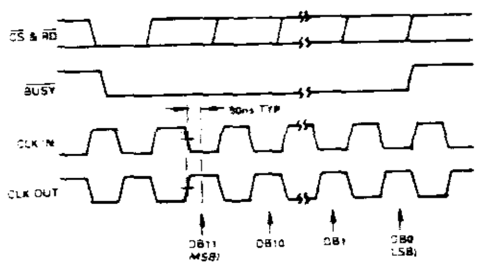


Figure 5. Operating Waveforms Using an External Clock Source for CLK IN

CONTROL INPUTS SYNCHRONIZATION

In applications where the $\overline{RD}$ control input is not synchronized with the ADC clock then conversion time can vary from 12 to 13 CLK IN periods. This is because the ADC waits for the first falling CLK IN edge after conversion start before the conversion procedure begins. Without synchronization, this delay can vary from zero to an entire clock period. If a constant conversion time is required, then the following approach ensures a fixed 5μs conversion time for the AD7572XX05 and 12.5μs for the AD7572XX12: when initiating a conversion, $\overline{RD}$ must go low on either the rising edge of CLK IN or the falling edge of CLK OUT.

DRIVING THE ANALOG INPUT

During conversion, the AIN input current is modulated by the DAC output current at a rate equal to the CLK IN frequency (i.e., 2.5MHz when CLK IN = 2.5MHz). The analog input voltage must remain fixed during this period and as a result must be driven from an op amp or sample hold with a low output impedance. The output impedance of an op amp is equal to the open loop output impedance divided by the loop gain at the frequency of interest.

Suitable devices capable of driving the AD7572 AIN input are the AD OP-27 and AD711 op amps or the AD585 sample hold.

INTERNAL CLOCK OSCILLATOR

Figure 6 shows the AD7572 internal clock circuit. A crystal or ceramic resonator may be connected between CLK IN (Pin 17) and CLK OUT (Pin 18) to provide a clock oscillator for the ADC timing. Alternatively the crystal/resonator may be omitted and an external clock source may be connected to CLK IN. For an external clock the mark/space ratio must be 50/50. An inverted CLK IN signal will appear at the CLK OUT pin as shown in the operating waveforms of Figure 5.

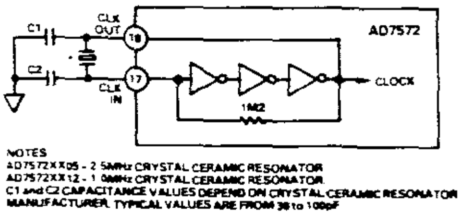


Figure 6. AD7572 Internal Clock Circuit

INTERNAL REFERENCE

The AD7572 has an on-chip, buffered, temperature-compensated, buried zener reference, which is factory trimmed to −5.25V ±1%. It is internally connected to the DAC and is also available at Pin 2 to provide up to 550μA current to an external load.

For minimum code transition noise the reference output should be decoupled with a capacitor to filter out wideband noise from the reference diode (10μF of tantalum in parallel with 100nF ceramic). However, large values of decoupling capacitor can affect the dynamic response and stability of the reference amplifier. A 10Ω resistor in series with the decoupling capacitors will eliminate this problem without adversely affecting the filtering effect of the capacitors. A simplified schematic of the reference with its recommended decoupling components is shown in Figure 7.

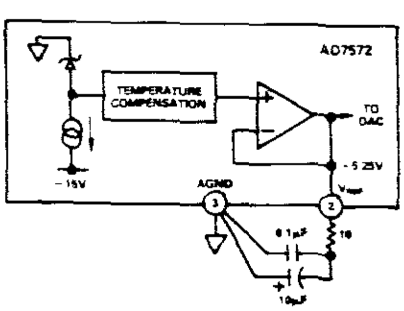


Figure 7. AD7572 Internal −5.25V Reference

UNIPOLAR OPERATION

Figure 8 shows the ideal input/output characteristic for the 0 to 5 volt input range of the AD7572. The designed code transitions occur midway between successive integer LSB values (i.e., 1/2LSB, 3/2LSBs, 5/2LSBs . . . FS-3/2LSBs). The output code is natural binary with 1LSB = FS/4096 = (5/4096)V = 1.22mV.

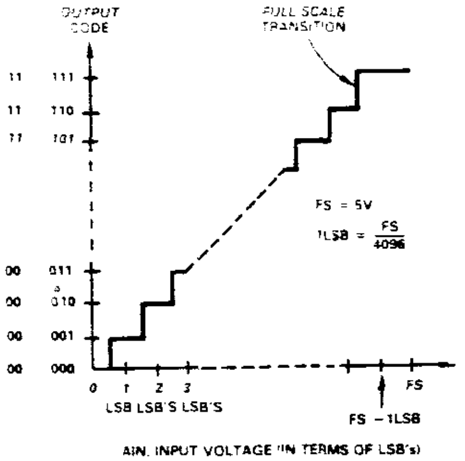


Figure 8. AD7572 Ideal Input/Output Transfer Characteristic

UNIPOLAR OFFSET AND FULL-SCALE ERROR ADJUSTMENT

In applications where absolute accuracy is important then offset and full-scale error can be adjusted to zero. Offset error must be adjusted before full-scale error. Figure 9 shows the extra components required for full-scale error adjustment. Zero offset is achieved by adjusting the offset of the op amp driving AIN (i.e., A1 in Figure 9). For zero offset error apply 0.61mV (i.e., 1/2LSB) at V_{IN} and adjust the op amp offset voltage until the ADC output code flickers between 0000 0000 0000 and 0000 0000 0001.

For zero full-scale error apply an analog input of 4.99817V (i.e., FS-3/2LSBs or last code transition) at V_{IN} and adjust R1 until the ADC output code flickers between 1111 1111 1110 and 1111 1111 1111.

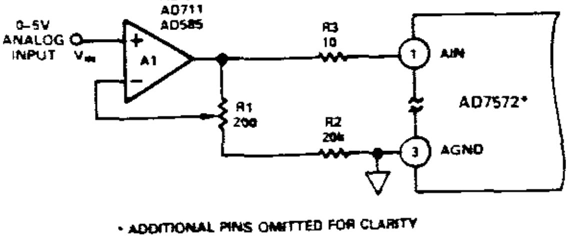


Figure 9. Unipolar 0 to +5V Operation with Gain Error Adjust

BIPOLAR OPERATION

Figures 10 and 12 show how bipolar operation can be achieved with the AD7572. Both circuits use an op-amp to offset the analog signal (V_{IN}) by 2.5V. Alternatively, the op amp (A1) can be replaced by a sample hold as shown in Figure 24. The op amp transfer functions are given below:

Figure 10: AIN = (V_{IN} + 2.5) volts
Figure 12: AIN = (– V_{IN} + 2.5) volts

Both circuits have an analog input range of ±2.5V and an LSB size of 1.22mV. The output codes are offset binary for Figure 10 and complementry offset binary for Figure 12. Their ideal input/output transfer characteristics after offset and full scale adjustment are shown in Figures 11 and 13.

Signal ranges other than ±2.5V are easily accommodated using different values of R3 and R4 for Figure 10, and a different R2 value for Figure 12. These resistors should be chosen such that the voltage range at AIN covers the full dynamic range (i.e., 0V to 5V) of the ADC. All resistors should be the same type and from the same manufacturer so that their temperature coefficients match.

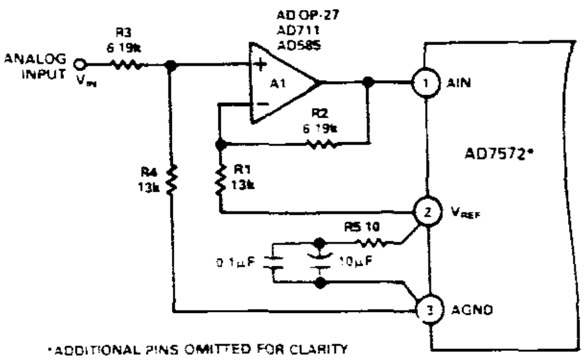


Figure 10. AD7572 Bipolar Operation – Output Code is Offset Binary

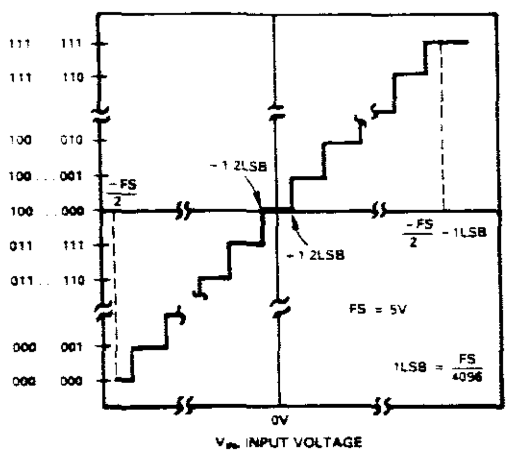
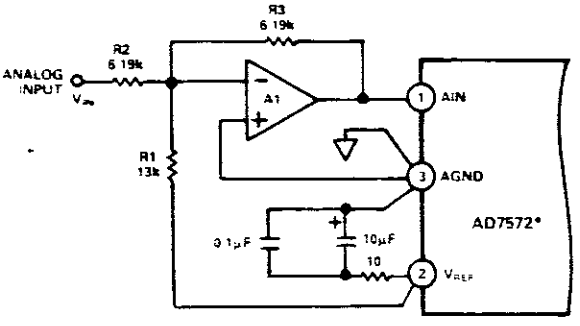


Figure 11. Ideal Input/Output Transfer Characteristic for the Bipolar Circuit of Figure 10



*ADDITIONAL PINS OMITTED FOR CLARITY

Figure 12. AD7572 Bipolar Operation – Output Code is Complementary Offset Binary

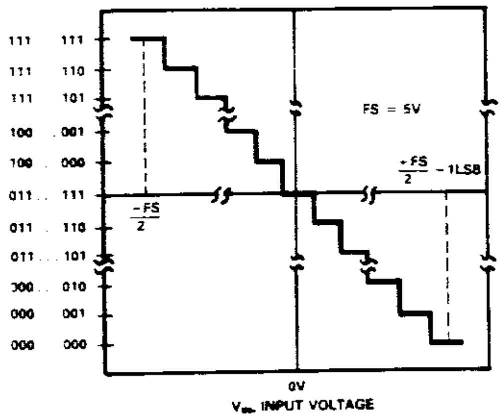
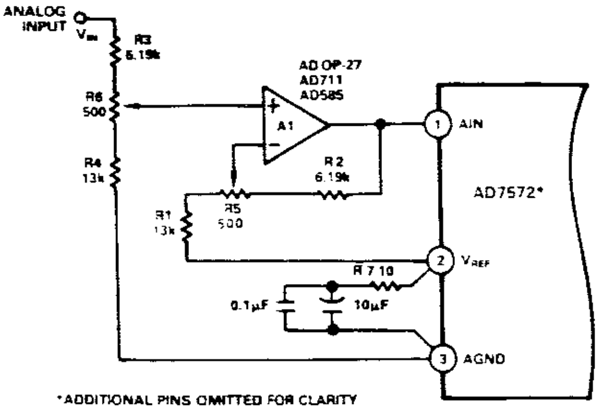


Figure 13. Ideal Input Output Transfer Characteristic for the Bipolar Circuit of Figure 12

OFFSET AND FULL-SCALE ERROR

In most Digital Signal Processing (DSP) applications offset and full-scale error have little or no effect on system performance. A typical example is a digital filter, where an analog signal is quantized, digitally processed and recreated using a DAC. In these type of applications the offset error can be eliminated by ac coupling the recreated signal. Full-scale error effect is linear and does not cause problems as long as the input signal is within the full dynamic range of the ADC. An important parameter in DSP applications is Differential Nonlinearity and this is not affected by either offset or full-scale error.

In measurement applications where absolute accuracy is required, offset and full-scale error can be adjusted to zero as in Figure 14.



*ADDITIONAL PINS OMITTED FOR CLARITY

Figure 14. AD7572 Bipolar Operation with Offset and Gain Error Adjust

BIPOLAR OFFSET AND FULL-SCALE ERROR ADJUSTMENT

The bipolar circuit of Figure 10 can be adjusted for offset and full-scale errors, by including two potentiometers R5 and R6, as shown in Figure 14. Offset must be adjusted before full-scale error. This is achieved by applying an analog input of 0.61mV (1/2LSB) at V_{IN} and adjusting R5 until the ADC output code flickers between 1000 0000 0000 and 1000 0000 0001.

For full-scale error adjustment, the analog input must be at 2.49817 volts (i.e., FS/2 – 3/2LSBs or last transition point). Then R6 is adjusted until the ADC output code flickers between 1111 1111 1110 and 1111 1111 1111.

A similar offset and full-scale error adjustment procedure may be employed for Figure 12 by making R1 and R2 variable. Offset must again be adjusted before full scale error. This is achieved by applying an analog input of 0.61mV at V_{IN} and adjusting R1 until the ADC output code flickers between 0111 1111 1110 and 0111 1111 1111.

For full-scale error adjust, apply a signal source of 2.49817V at V_{IN} and adjust R2 until the ADC output code flickers between 0000 0000 0000 and 0000 0000 0001.

APPLICATION HINTS

Wire wrap boards are not recommended for high resolution or high-speed A/D converters. To obtain the best performance from the AD7572 a printed circuit board is required. Layout for the printed circuit board should ensure that digital and analog signal lines are separated as much as possible. In particular, care should be taken not to run any digital track alongside an analog signal track or underneath the AD7572. The analog input should be screened by AGND.

A single point analog ground (STAR ground) separate from the logic system ground should be established at Pin 3 (AGND) or as close as possible to the AD7572 as shown in Figure 15. Pin 12 (AD7572 DGND) and all other analog grounds should be connected to this single analog ground point. No other digital grounds should be connected to this analog ground point. Low impedance analog and digital power supply common returns are essential to low noise operation of the ADC and the foil width for these tracks should be as wide as possible.

Noise: Input signal leads to AIN and signal return leads from AGND (Pin 3) should be kept as short as possible to minimize input noise coupling. In applications where this is not possible, a shielded cable between source and ADC is recommended. Also, since any potential difference in grounds between the signal source and ADC appears as an error voltage in series with the input signal, attention should be paid to reducing the ground circuit impedances as much as possible.

In applications where the AD7572 data outputs and control signals are connected to a continuously active microprocessor bus, it is possible to get LSB errors in conversion results. These errors are due to feedthrough from the microprocessor to the successive approximation comparator. The problem can be eliminated by forcing the microprocessor into a WAIT state during conversion (see Slow Memory Mode interfacing), or by using three-state buffers to isolate the AD7572 data bus.

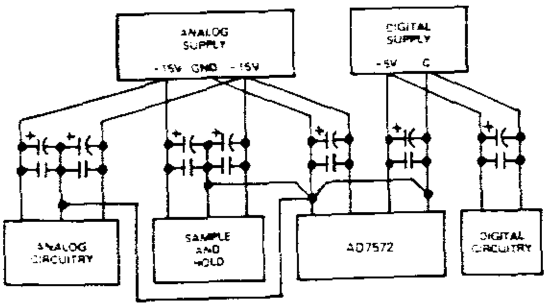


Figure 15. Power Supply Grounding Practice

TIMING AND CONTROL

Conversion start and data read operations are controlled by three AD7572 digital inputs; HBEN, $\overline{CS}$ and $\overline{RD}$. Figure 16 shows the logic structure associated with these inputs. The three signals are internally gated so that a logic "0" is required on all three inputs to initiate a conversion. Once initiated it cannot be re-started until conversion is complete. Converter status is indicated by the $\overline{BUSY}$ output, and this is low while conversion is in progress.

There are two modes of operation as outlined by the timing diagrams of Figures 17 to 20. Slow Memory Mode is designed for microprocessors which can be driven into a WAIT state, a READ operation brings $\overline{CS}$ and $\overline{RD}$ low which initiates a conversion and data is read when conversion is complete.
The second is the ROM Mode which does not require microprocessor WAIT states, a READ operation brings $\overline{CS}$ and $\overline{RD}$ low which initiates a conversion and reads the previous conversion result.

DATA FORMAT

The output data format can either be a complete parallel load (DB11..DB0) for 16-bit microprocessors or a two byte load for 8-bit microprocessors. Data is always right justified (i.e., LSB is the most right-hand bit in a 16-bit word. For a two byte read, only data outputs D7 . . . D0/8 are used. Byte selection is governed by the HBEN input which controls an internal digital multiplexer. This multiplexes the 12-bits of conversion data onto the lower D7 . . . D0/8 outputs (4 MSBs or 8 LSBs) where it can be read in two read cycles. The 4 MSB's always appear on D11 . . . D8 whenever the three-state output drives are turned on.

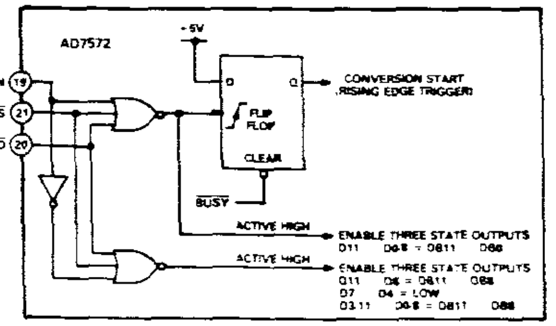


Figure 16. Internal Logic for Control Inputs $\overline{CS}$, $\overline{RD}$ and HBEN

SLOW MEMORY MODE, PARALLEL READ (HBEN = LOW)

Figure 17 and Table I shows the timing diagram and data bus status for Slow Memory Mode, Parallel Read. $\overline{CS}$ and $\overline{RD}$ going low triggers a conversion and the AD7572 acknowledges by taking $\overline{BUSY}$ low. Data from the previous conversion appears on the three state data outputs. $\overline{BUSY}$ returns high at the end of conversion when the output latches have been updated and the conversion result is placed on data outputs D11 . . . D0/8.

SLOW MEMORY MODE, TWO BYTE READ

For a two byte read only 8 data outputs D7 . . . D0/8 are used. Conversion start procedure and data output status for the first read operation is identical to Slow Memory Mode, Parallel Read. See Figure 18 timing diagram and Table II data bus status. At the end of conversion the low data byte (DB7 . . . DB0) is read from the ADC. A second READ operation with HBEN high, places the high byte on data outputs D3/11 . . . D0/8 and disables conversion start. Note the 4MSB's appear on data outputs D11 . . . D8 during the two READ operations above.

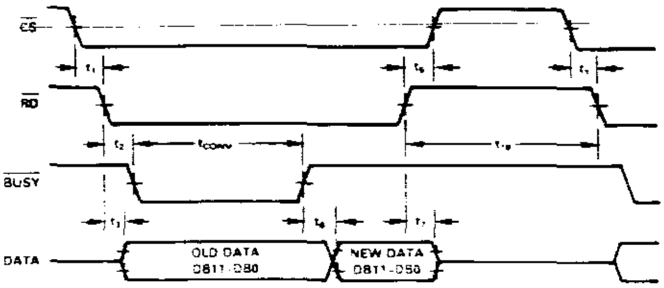


Figure 17. Slow Memory Mode, Parallel Read Timing Diagram

AD7572 Data Outputs	D11	D10	D9	D8	D7	D6	D5	D4	D3/11	D2/10	D1/9	D0/8
Read	DB11	DB10	DB9	DB8	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0

Table I. Slow Memory Mode, Parallel Read Data Bus Status

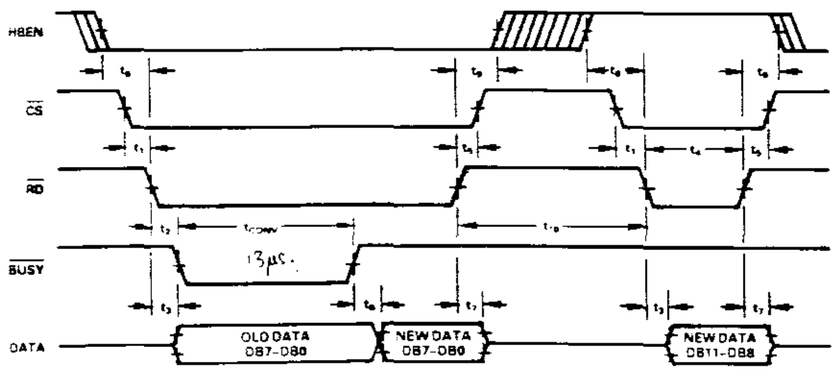


Figure 18. Slow Memory Mode, Two Byte Read Timing Diagram

AD7572 Data Outputs	D7	D6	D5	D4	D3/11	D2/10	D1/9	D0/8
First Read	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Second Read	LOW	LOW	LOW	LOW	DB11	DB10	DB9	DB8

Table II. Slow Memory Mode, Two Byte Read Data Bus Status

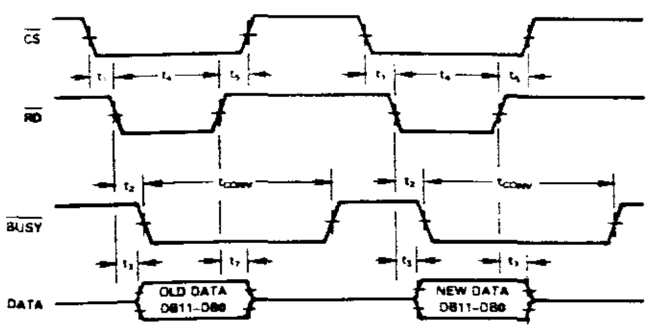


Figure 19. ROM Mode, Parallel Read Timing Diagram

AD7572 Data Outputs	D11	D10	D9	D8	D7	D6	D5	D4	D3/11	D2/10	D1/9	D0/8
First Read (Old Data)	DB11	DB10	DB9	DB8	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Second Read	DB11	DB10	DB9	DB8	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0

Table III. ROM Mode, Parallel Read Data Bus Status

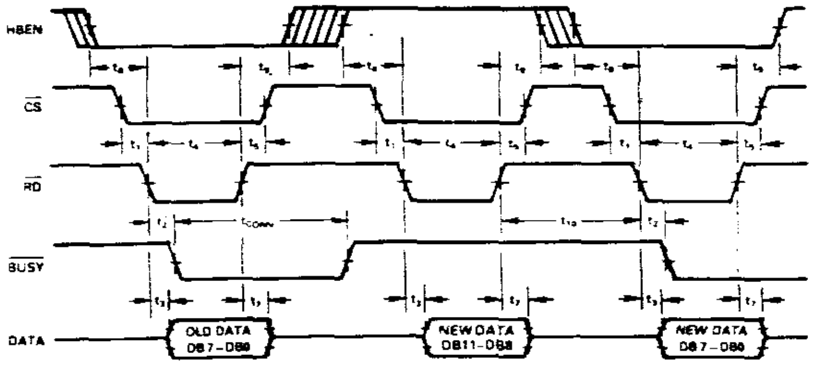


Figure 20. ROM Mode, Two Byte Read Timing Diagram

AD7572 Data Outputs	D7	D6	D5	D4	D3/11	D2/10	D1/9	D0/8
First Read (Old Data)	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Second Read	LOW	LOW	LOW	LOW	DB11	DB10	DB9	DB8
Third Read	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0

Table IV. ROM Mode, Two Byte Read Data Bus Status

ROM MODE, PARALLEL READ (HBEN = LOW)

The ROM Mode avoids placing a microprocessor into a wait state. A conversion is started with a READ operation and the 12-bits of data from the previous conversion is available on data outputs D11 . . . D0/8 (see Figure 19 and Table III). This data may be disregarded if not required. A second READ operation reads the new data (DB11 . . . DB0) and starts another conversion. A delay at least as long as the AD7572 conversion time must be allowed between READ operations.

ROM MODE, TWO BYTE READ

As previously mentioned for a two byte read, only data outputs D7 . . . D0/8 are used. Conversion is started in the normal way with a READ operation and the data output status is the same as the ROM Mode, Parallel Read. See Figure 20 timing diagram and Table IV data bus status. Two more READ operations are required to access the new conversion result. A delay equal to the AD7572 conversion time must be allowed between conversion start and the second data READ operation. The second READ operation, with HBEN high, disables conversion start and places the high byte (4MSBs) on data outputs D3/11 . . . D0/8. A third READ operation accesses the low data byte (DB7 . . . DB0) and starts another conversion. The 4MSB's appear on data outputs D11 . . . D8 during all three read operations above.

MICROPROCESSOR INTERFACING

The AD7572 is designed to interface with microprocessors as a memory mapped device. The $\overline{CS}$ and $\overline{RD}$ control inputs are common to all peripheral memory interfacing. The HBEN input serves as a data byte select for 8-bit processors and is normally connected to the microprocessor address bus.

MC68000 Microprocessor

Figure 21 shows a typical interface for the 68000. The AD7572 is operating in the Slow Memory Mode. Assuming the AD7572 is located at address C000, then the following single 16-bit MOVE instruction both starts a conversion and reads the conversion result.

Move.W $C000,D0

At the beginning of the instruction cycle when the ADC address is selected, $\overline{BUSY}$ and $\overline{CS}$ assert $\overline{DTACK}$, so that the 68000 is forced into a WAIT state. At the end of conversion $\overline{BUSY}$ returns high and the conversion result is placed in the D0 register of the UP.

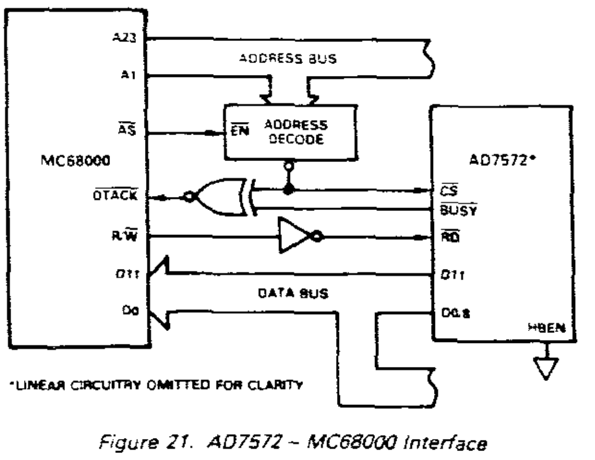


Figure 21. AD7572 – MC68000 Interface

8085A, Z80 MICROPROCESSOR

Figure 22 shows an AD7572 interface for the Z80 and 8085A. The AD7572 is operating in the Slow Memory Mode and a two byte read is required. Not shown in the figure is the 8-bit latch required to demultiplex the 8085A common address/data bus. A0 is used to assert HBEN, so that an even address (HBEN = LOW) to the AD7572 will start a conversion and read the low data byte. An odd address (HBEN = HIGH) will read the high data byte. This is accomplished with the single 16-bit LOAD instruction below.

For the 8085A	LHLD (B000)
For the Z80	LD HL, (B000)

This is a two byte read instruction which loads the ADC data (address B000) into the HL register pair. During the first read operation, $\overline{BUSY}$ forces the microprocessor to WAIT for the AD7572 conversion. No WAIT states are inserted during the second read operation when the microprocessor is reading the high data byte.

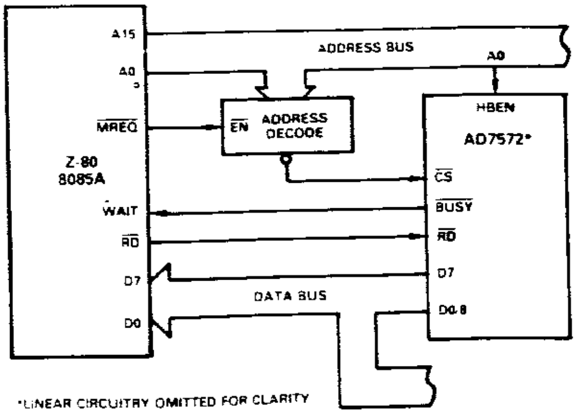


Figure 22. AD7572 – 8085A Z80 Interface

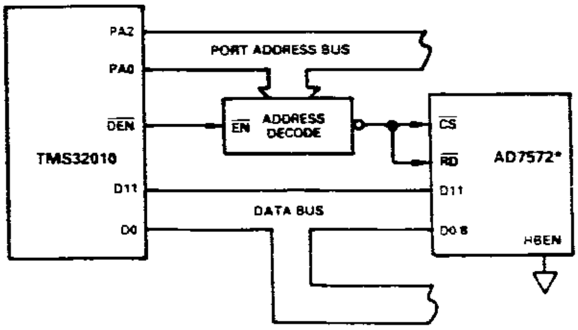
TMS32010 MICROCOMPUTER

Figure 23 shows an AD7572–TMS32010 interface. The AD7572 is operating in the ROM Mode. The interface is designed for a maximum TMS32010 clock frequency of 18MHz but will typically work over the full TMS32010 clock frequency range.

The AD7572 is mapped at a port address. The following I/O instruction starts a conversion and reads the previous conversion result into data memory.

IN A,PA (PA = PORT ADDRESS)

When conversion is complete, a second I/O instruction reads the up-to-date data into the accumulator and starts another conversion. A delay at least as long as the ADC conversion time must be allowed between I/O instructions.



*LINEAR CIRCUITRY OMITTED FOR CLARITY

Figure 23. AD7572 – TMS32010 Interface

AD7572-AD585 SAMPLE-HOLD INTERFACE

Figure 24 shows an AD585 sample-hold amplifier driving the AIN input of the AD7572. The interface contains resistors R1, R2, R3 and R4 to allow a bipolar input signal range of ±2.5 volts. The maximum sampling frequency is 125kHz for the AD7572XX05 (5μs conversion) and 64.5kHz for the AD7572XX12 (12.5μs conversion). This includes the sample-hold amplifier acquisition time (3μs).

When an AD7572 conversion is initiated, the converter $\overline{BUSY}$ output goes low indicating conversion is in progress. The falling edge of this $\overline{BUSY}$ output signal places the sample-hold amplifier into the HOLD mode "freezing" the input signal to the AD7572. When conversion is finished, the $\overline{BUSY}$ output returns HIGH allowing the sample-hold to track the input signal. To achieve the maximum sampling rate, the AD7572 output data must be read within 3μs immediately after conversion while the sample-hold amplifier is acquiring the next sample.

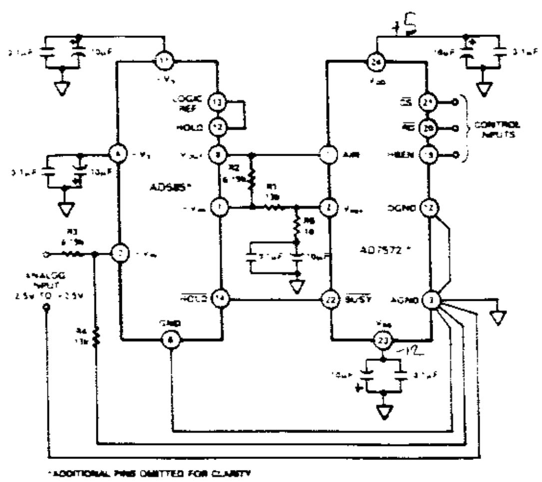


Figure 24. AD7572 – AD585 Sample-and-Hold Interface

APPENDIX 8

Specification sheets of MAX262 filter

MAXIM

Microprocessor Programmable Universal Active Filters

MAX260/261/262

General Description

The MAX260/261/262 CMOS dual second-order universal switched-capacitor active filters allow microprocessor control of precise filter functions. No external components are required for a variety of bandpass, lowpass, highpass, notch and allpass configurations. Each device contains two second-order filter sections which place center frequency, Q, and filter operating mode under programmed control.

An input clock, along with a 6-bit f_0 program input, determine the filter's center or corner frequency without affecting other filter parameters. The filter Q is also programmed independently. Separate clock inputs for each filter section operate with either a crystal, RC network, or external clock generator.

The MAX260 has superior offset and DC specifications than the MAX261 and MAX262 and a center frequency (f_0) range of 7.5kHz. The MAX261 handles center frequencies to 30kHz while the MAX262 extends the center frequency range to 75kHz by employing lower clock-to-f_0 ratios. All devices are available in 24-pin DIP and small outline packages in commercial, extended, and military temperature ranges.

Applications

- μP Tuned Filters
- Anti-Aliasing Filters
- Digital Signal Processing
- Adaptive Filters
- Signal Analysis
- Phase-Locked Loops

Features

- ◆ Filter Design Software Available
- ◆ Microprocessor Interface
- ◆ 64-Step Center Frequency Control
- ◆ 128-Step Q Control
- ◆ Independent Q and f_0 Programming
- ◆ Guaranteed Clock to f_0 Ratio—1% (A grade)
- ◆ 75kHz f_0 Range (MAX262)
- ◆ Single +5V and ±5V Operation

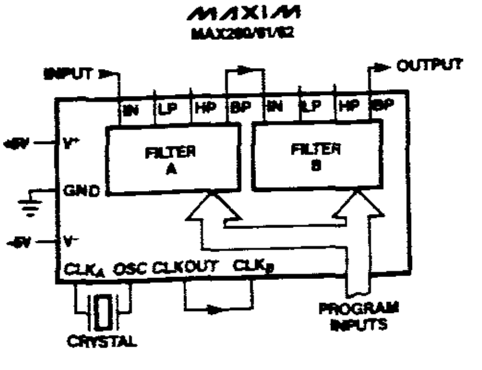
Ordering Information

PART	TEMP. RANGE	PACKAGE*	ACCURACY
MAX260ACNG	0°C to +70°C	Plastic DIP	1%
MAX260BCNG	0°C to +70°C	Plastic DIP	2%
MAX260AENG	-40°C to +85°C	Plastic DIP	1%
MAX260BENG	-40°C to +85°C	Plastic DIP	2%
MAX260ACWG	0°C to +70°C	Wide SO	1%
MAX260BCWG	0°C to +70°C	Wide SO	2%
MAX260AMRG	-55°C to +125°C	CERDIP	1%
MAX260BMRG	-55°C to +125°C	CERDIP	2%
MAX261ACNG	0°C to +70°C	Plastic DIP	1%

** All devices—24-pin packages 0.3" wide packages*

Ordering Information Continued on Last Page

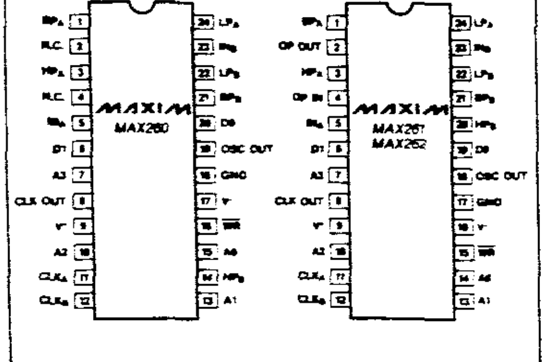
Functional Diagram



Fourth-Order Bandpass Filter

Pin Configuration

Top View



10

Microprocessor Programmable Universal Active Filters

ABSOLUTE MAXIMUM RATINGS

Total Supply Voltage (V^+ to V^-) 15V
Input Voltage, any pin .... V^--0.3V to V^++0.3V
Input Current, any pin .... ±50mA
Power Dissipation
  Plastic DIP (derate 8.33mW/°C above 70°C) ... 660mW
  CERDIP (derate 12.5mW/°C above 70°C) 1000mW
  Wide SO (derate 11.8mW/°C above 70°C) 944mW

Operating Temperature
  MAX260/261/262XCXG .... 0°C to +70°C
  MAX260/261/262XEXG .... -40°C to +85°C
  MAX260/261/262XMXG .... -55°C to +125°C
Storage Temperature .... -65°C to +160°C
Lead Temperature (Soldering, 10 seconds) .... +300°C

Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions above those indicated in the operational sections of the specification is not implied. Exposure to absolute Maximum ratings conditions for extended periods may affect the device reliability.

ELECTRICAL CHARACTERISTICS

(V^+ = +5V, V^- = -5V, CLK_A = CLK_B = ± 5V 350kHz for the MAX260 and 1.5MHz for the MAX261/62, f_{CLK}/f_0 = 199.49 for MAX260/61 and 139.80 for MAX262, Filter Mode 1, T_A = +25°C unless otherwise noted.)

PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
f_0 Center Frequency Range				See Table 1		
Maximum Clock Frequency				See Table 1		
f_{CLK}/f_0 Ratio Error (Note 1)	$T_A = T_{MIN}$ to T_{MAX}	MAX260A MAX260B MAX261/62A MAX261/62B		±0.2 ±0.2 ±0.2 ±0.2	±1.0 ±2.0 ±1.0 ±2.0	%
f_0 Temperature Coefficient				-5		ppm/°C
Q Accuracy (deviation from ideal continuous filter) (Note 2)	$T_A = T_{MIN}$ to T_{MAX} Q = 0.5 to 16 Q = 0.5 to 16 Q = 32 Q = 32 Q = 64 Q = 64 Q = 0.5 to 16 Q = 0.5 to 16 Q = 32 Q = 32 Q = 64 Q = 64	 MAX260A MAX260B MAX260A MAX260B MAX260A MAX260B MAX261/62A MAX261/62B MAX261/62A MAX261/62B MAX261/62A MAX261/62B		 ±1 ±1 ±2 ±2 ±4 ±4 ±1 ±1 ±2 ±2 ±4 ±4	 ±5 ±10 ±10 ±15 ±15 ±22 ±5 ±10 ±10 ±15 ±15 ±22	%
Q Temperature Coefficient				±20		ppm/°C
DC Lowpass Gain Accuracy		MAX260A MAX260B MAX261/62A MAX261/62B		±0.1 ±0.1 ±0.1 ±0.1	±0.2 ±0.3 ±0.25 ±0.5	dB
Gain Temperature Coefficient	Lowpass (at D.C.) Bandpass (at f_0)	MAX260 MAX261/62 MAX260/61/62		-5 -5 +20		ppm/°C
Offset Voltage At Filter Outputs—LP, BP, HP (Note 3)	$T_A = T_{MIN}$ to T_{MAX}, Q = 4 Mode 1	MAX260A MAX260B MAX261A MAX261B MAX262A MAX262B		±0.05 ±0.15 ±0.40 ±0.80 ±0.40 ±0.80	±0.25 ±0.45 ±0.90 ±1.60 ±0.90 ±1.60	V
	Mode 3	MAX260A MAX260B MAX261A MAX261B MAX262A MAX262B		±0.075 ±0.075 ±0.50 ±0.90 ±0.50 ±0.90	±0.30 ±0.50 ±1.00 ±1.60 ±1.00 ±1.60	V
Offset Voltage Temperature Coefficient	f_{CLK}/f_0 = 100.53, Q = 4 $T_A = T_{MIN}$ to T_{MAX}			±0.75		mV/°C

ELECTRICAL CHARACTERISTICS (Continued)

(V^+ = +5V, V^- = -5V, CLK_A = CLK_B = ± 5V 350kHz for the MAX260 and 1.5MHz for the MAX261/62, f_{CLK}/f_0 = 199.49 for MAX260/61 and 139.80 for MAX262, Filter Mode 1, T_A = +25°C unless otherwise noted.)

PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
Clock Feedthrough				±4		mV
Crosstalk				-70		dB
Wideband Noise (Note 4)	Q = 1, 2nd-Order, LP/BP 4th-Order LP (Fig. 26) 4th-Order BP (Fig. 24)			See Typ. Oper. Char. 90 100		μV_{RMS}
Harmonic Distortion at f_0	Q = 4, V_{IN} = 1.5V_{PP}			-57		dB
Supply Voltage Range	T_A = T_{MIN} to T_{MAX}		±2.37	±5	±6.3	V
Power Supply Current (Note 5)	T_A = T_{MIN} to T_{MAX} CMOS Level Logic Inputs	MAX260 MAX261 MAX262		15 16 16	20 20 20	mA
Shutdown Supply Current	$Q0_A$-$Q6_A$ = all 0, CMOS Level Logic Inputs (Note 5)			1.5		mA
INTERNAL AMPLIFIERS						
Output Signal Swing (Note 6)	T_A = T_{MIN} to T_{MAX}, 10kΩ load			±4.75		V
Output Short Circuit Current	Source Sink			50 2		mA
Power Supply Rejection Ratio	0Hz to 10kHz			-70		dB
Gain Bandwidth Product				2.5		MHz
Slew Rate				6		V/µs

ELECTRICAL CHARACTERISTICS (for V± = ±2.5V ±5%)

V^+ = +2.37V, V^- = -2.37V, CLK_A = CLK_B = ±2.5V 250kHz for the MAX260 and 1MHz for the MAX261/62, f_{CLK}/f_0 = 199.49 for MAX260/61 and 139.80 for MAX262, Filter Mode 1, T_A = +25°C unless otherwise noted.

PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
f_0 Center Frequency Range				(Note 7)		
Maximum Clock Frequency				(Note 7)		
f_{CLK}/f_0 Ratio Error (Notes 1, 8)	Q = 8	MAX26XA MAX26XB		±0.1 ±0.1	1 2	%
Q Accuracy (deviation from ideal continuous filter) (Notes 2, 8)	Q = 8 f_{CLK}/f_0 = 199.49 f_{CLK}/f_0 = 199.49 f_{CLK}/f_0 = 139.80	MAX260A MAX260B MAX261A MAX261B MAX262A MAX262B		±2 ±2 ±2 ±2 ±2 ±2	±5 ±10 ±5 ±10 ±5 ±10	%
Output Signal Swing	All Outputs (Note 6)			±2		V
Power Supply Current	CMOS Level Logic Inputs (Note 5)			7		mA
Shutdown Current	CMOS Level Logic Inputs (Note 5)			0.35		mA

Note 1: f_{CLK}/f_0 accuracy is tested at 100.53, 103.67, 106.81, 113.1, 125.66, 150.8, and 199.49 on the MAX260/61, and at 40.84, 43.98, 47.12, 53.41, 65.97, 91.11, and 139.8 on the MAX262.
Note 2: Q accuracy tested at Q = 0.5, 1, 2, 4, 8, 16, 32, and 64. Q of 32 and 64 tested at 1/2 stated clock frequency.
Note 3: The Offset Voltage is specified for the entire filter. Offset is virtually independent of Q and f_{CLK}/f_0 ratio setting. The test clock frequency for Mode 3 is 175kHz for the MAX260 and 750kHz for the MAX261/262.
Note 4: Output noise is measured with an RC output smoothing filter at 4 × f_0 to remove clock feedthrough.
Note 5: TTL logic levels are: HIGH = 2.4V, LOW = 0.8V. CMOS logic levels are: HIGH = 5V, LOW = 0V. Power supply current is typically 4mA higher with TTL logic and clock input levels.
Note 6: On the MAX260 only, the HP output signal swing is typically 0.75V less than the LP or BP outputs.
Note 7: At ±2.5V supplies, the f_0 range and maximum clock frequency are typically 75% of values listed in Table 1.
Note 8: f_{CLK}/f_0 and Q accuracy are a function of the accuracy of internal capacitor ratios. No increase in error is expected at ±2.5V as compared to ±5V however these parameters are only tested to the extent indicated by the MIN or MAX limits.

Microprocessor Programmable Universal Active Filters

INTERFACE SPECIFICATIONS (Note 9)

(V^+ = +5V, V^- = -5V, T_A = +25°C unless otherwise noted.)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
$\overline{WR}$ Pulse Width	t_{WR}		250	150		ns
Address Setup	t_{AS}		25			ns
Address Hold	t_{AH}		0			ns
Data Setup	t_{DS}		100	50		ns
Data Hold	t_{DH}		10	0		ns
Logic Input High	V_{IH}	$\overline{WR}$, D0-D1, A0-A3, CLK_A, CLK_B $T_A = T_{MIN}$ to T_{MAX}	2.4			V
Logic Input Low	V_{IL}	$\overline{WR}$, D0-D1, A0-A3, CLK_A, CLK_B $T_A = T_{MIN}$ to T_{MAX}			0.8	V
Input Leakage Current	I_{IN}	$\overline{WR}$, D0-D1, A0-A3, CLK_B CLK_A $T_A = T_{MIN}$ to T_{MAX}		6	10 60	µA
Input Capacitance	C_{IN}	$\overline{WR}$, D0-D1, A0-A3, CLK_A, CLK_B			15	pF

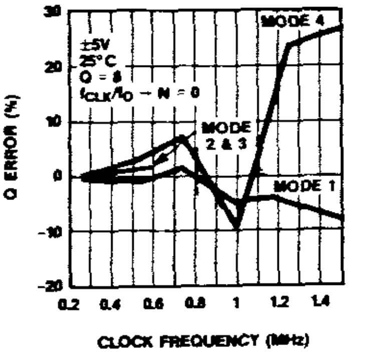
Note 9: Interface timing specifications are guaranteed by design and are not subject to test.

Pin Description

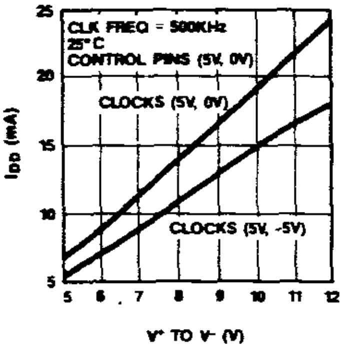
MAX260 PIN #	MAX261/2 PIN #	NAME	FUNCTION
9	9	V^+	Positive supply voltage
17	16	V^-	Negative supply voltage
18	17	GND	Analog Ground. Connect to the system ground for dual supply operation or mid-supply for single supply operation. GND should be well bypassed in single supply applications.
11	11	CLK_A	Input to the oscillator and clock input to section A. This clock is internally divided by 2.
12	12	CLK_B	Clock input to filter B. This clock is internally divided by 2.
8	8	CLK OUT	Clock Output for crystal and R-C oscillator operation
19	18	OSC OUT	Connects to crystal or R-C for self clocked operation
5,23	5,23	IN_A,IN_B	Filter inputs
1,21	1,21	BP_A,BP_B	Bandpass outputs
24,22	24,22	LP_A,LP_B	Lowpass outputs
3,14	3,20	HP_A,HP_B	Highpass/Notch/Allpass outputs
16	15	$\overline{WR}$	Write Enable input
15,13, 10,7	14,13, 10,7	A0,A1 A2,A3	Address inputs for f_0 and Q input data locations
20,6	19,6	D0,D1	Data inputs for f_0 and Q programming
	2	OP OUT	Output of uncommitted op-amp on MAX261/62 only. Pin 2 is a no-connect on the MAX260
	4	OP IN	Inverting input of uncommitted op-amp on MAX261/62 only (Non-inverting input is internally connected to ground). Pin 4 is a no-connect on the MAX260.

Typical Operating Characteristics

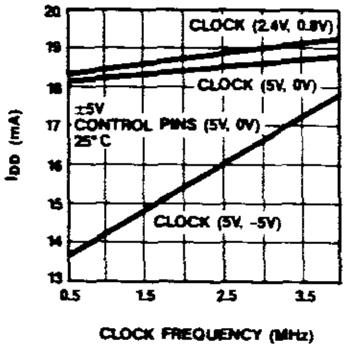
Q ERROR vs CLOCK FREQUENCY MAX260



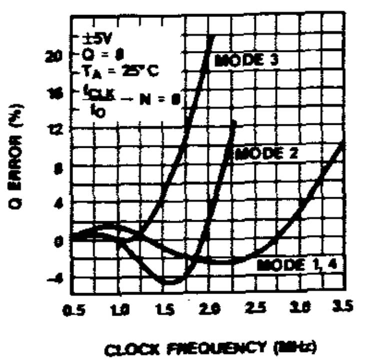
I_{DD} vs POWER SUPPLY VOLTAGE



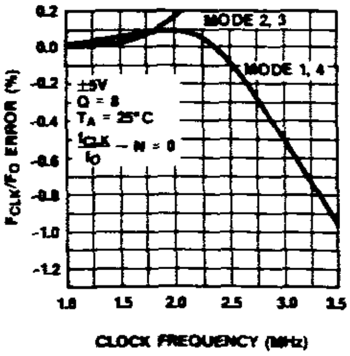
I_{DD} vs CLOCK FREQUENCY



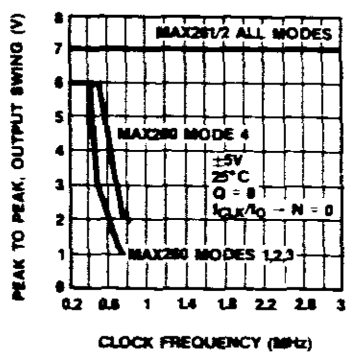
Q ERROR vs CLOCK FREQUENCY MAX261/2



F_{CLK}/F_O ERROR vs CLOCK FREQUENCY MAX261/2



OUTPUT SIGNAL SWING vs CLOCK FREQUENCY



Wideband RMS Noise (db ref. to $2.47V_{RMS}$, $7V_{p-p}$) **±5V Supplies**

	Mode	Q = 1			Q = 8			Q = 64		
		LP	BP	HP/AP/N	LP	BP	HP/AP/N	LP	BP	HP/AP/N
MAX261/2	1	-84	-90	-84	-80	-82	-85	-72	-73	-85
	2	-88	-90	-88	-84	-82	-84	-77	-73	-76
	3	-84	-90	-88	-80	-82	-82	-73	-73	-74
	4	-83	-89	-84	-79	-81	-85	-71	-73	-85
MAX260	1	-87	-89	-86	-81	-81	-86	-73	-73	-86
	2	-89	-88	-85	-83	-80	-82	-75	-72	-74
	3	-87	-88	-85	-80	-82	-80	-71	-72	-72
	4	-87	-88	-86	-81	-80	-86	-71	-72	-86

Notes:
1. f_{CLK} = 1 MHz for MAX261/2, f_{CLK} = 350kHz for MAX260
2. f_{CLK}/f_0 ratio programmed at N = 63 (see Table 2)
3. Clock feedthrough is removed with an RC lowpass at $4f_0$, i.e. R = 3.9kΩ, C = 2000pF for MAX261.

Noise Spectral Distribution (MAX261, f_{CLK} = 1 MHz, dB ref. to $2.47V_{RMS}$, $7V_{p-p}$)

Measurement Bandwidth	Q=1	Q=8	Q=64
Wideband	-84	-80	-72
3 KHz	-87	-87	-86
C Message Weighted	-93	-93	-93

Microprocessor Programmable Universal Active Filters

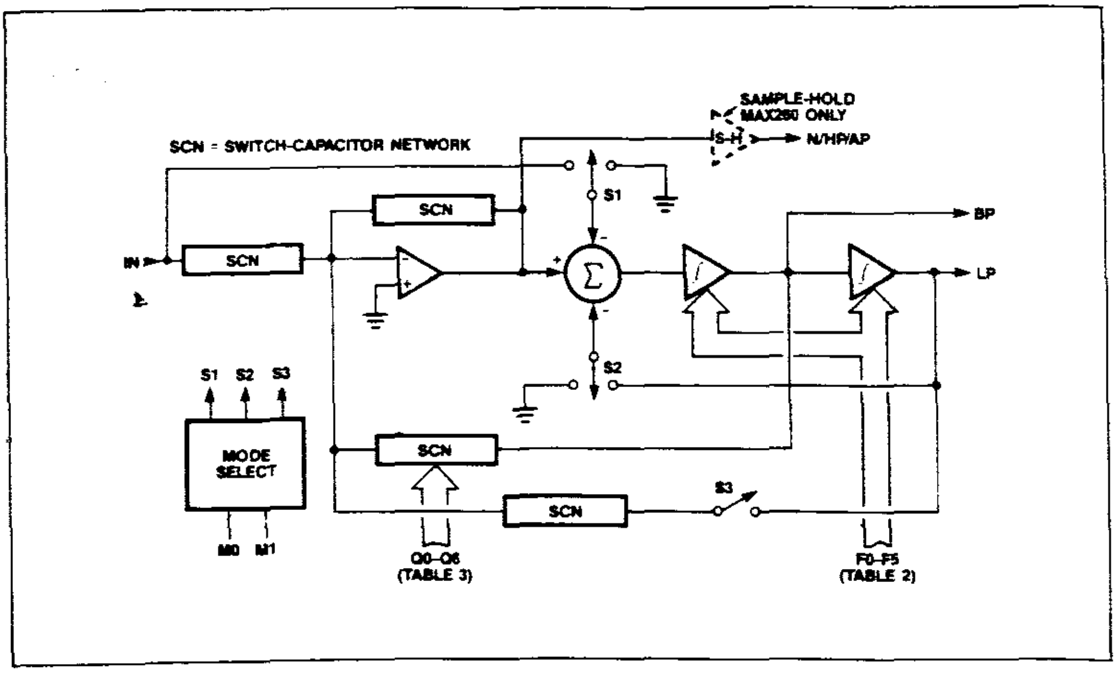


Figure 1. Filter Block Diagram (One Second-Order Section)

Introduction

Each MAX260/61/62 contains two second-order switched-capacitor active filters. Figure 1 shows the filter's state variable topology, employed with two cascaded integrators and one summing amplifier. The MAX261 and MAX262 also contain an uncommitted amplifier. On-chip switches and capacitors provide feedback to control each filter section's f_0 and Q. Internal capacitor ratios are primarily responsible for the accuracy of these parameters. Although these switched-capacitor networks (SCN) are in fact sampled systems, their behavior very closely matches that of continuous filters, such as RC active filters. The ratio of the clock frequency to the filter center frequency (f_{CLK}/f_0) is kept large so that ideal second-order state-variable response is maintained.

The MAX262 uses a lower range of sampling (f_{CLK}/f_0) ratios than the MAX260 or MAX261 to allow higher operating f_0 frequencies and signal bandwidths. These reduced sample rates result in somewhat more deviation from ideal continuous filter parameters than with the MAX260/61. However, these differences can be compensated using Figure 20 (See "Applications Hints") or Maxim's filter design software.

The MAX260 employs auto-zero circuitry not included in the MAX261 or 262. This provides improved DC characteristics, and improved low frequency performance at the expense of high end f_0 and signal bandwidth. The N/HP/AP outputs of the MAX260 are internally sample-and-held, as a result of its auto-zero operation. Signal swing at this output is somewhat reduced as a result (MAX260 only). See Table 1 for bandwidth comparisons of the three filters.

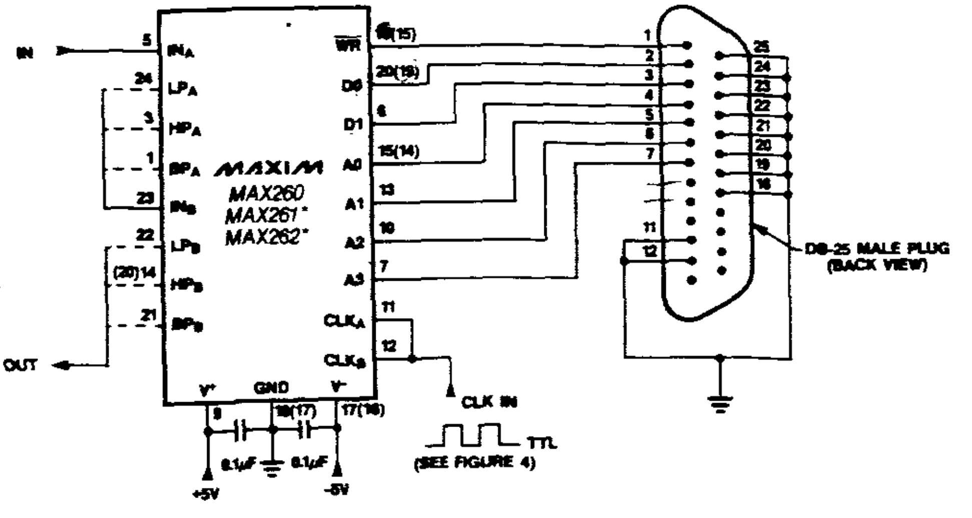
Maxim also provides design programs which aid in converting filter response specifications into the f_0 and Q program codes used by the MAX260 series devices. This software also precompensates f_0 and Q when low sample rates are used.

It is important to note that in all MAX260 series filters, the filter's internal sample rate is one half the input clock rate (CLK_A or CLK_B) due to an internal division by two. All clock related data, tables, and other discussions in this data sheet refer to the frequency at the CLK_A or CLK_B input, i.e. twice the internal sample rate, unless specifically stated otherwise.

Quick Look Design Procedure

The MAX260, MAX261 and MAX262, with Maxim's filter design software, greatly simplifies the design procedures for many active filters. Most designs can be realized using a three step process described in this section. If the design software is not used, or if the filter complexity is beyond the scope of this section, refer to the remainder of this data sheet for more detailed applications and design information.

```
100  AB$="FILTER A " : GOSUB 150 : REM GET DATA FOR SECTION A
110  ADD = 0 : GOSUB 220 : REM WRITE DATA TO THE PRINTER PORT
120  AB$="FILTER B " : GOSUB 150 : REM GET DATA FOR B
130  ADD = 32 : GOSUB 220 : REM WRITE DATA TO PRINTER PORT
140  GOTO 100
150  PRINT "MODE (1 to 4, see Table 5) "; AB$; : INPUT M
160  IF M<1 OR M>4 THEN GOTO 150
170  PRINT "CLOCK RATIO (0 to 63, N of Table 2) "; AB$; : INPUT F
180  IF F<0 OR F>63 THEN GOTO 170
190  PRINT "Q (0 to 127, N of Table 3) "; AB$; : INPUT Q
200  IF Q<0 OR Q>127 THEN GOTO 190 ELSE : PRINT
210  RETURN
220  LPRINT CHR$(ADD+M-1); : ADD = ADD+4
230  FOR I = 1 TO 3
240  X=(ADD + (F - 4*INT(F/4))) : LPRINT CHR$(X);
250  F=INT(F/4) : ADD = ADD + 4
260  NEXT I
270  FOR I = 1 TO 4
280  X=(ADD + (Q - 4*INT(Q/4))) : LPRINT CHR$(X);
290  Q=INT(Q/4) :: ADD = ADD + 4
300  NEXT I
310  RETURN
```



* PIN NUMBERS IN ( ) ARE FOR MAX261/262

Figure 2. Basic Program and Hardware Connections to Parallel Printer Port for "Quick Look" Using a Personal Computer.

Step 1—Filter Design

Start with the program "PZ" to determine what type of filter is needed. This helps determine the type (Butterworth, Chebyshev, etc.) and the number of poles for the optimum choice. The program also plots the frequency response and calculates the pole/zero (f_0) and Q values for each second-order section. Each MAX260/61/62 contains two second-order sections and devices may be cascaded for higher order filters.

Step 2—Generate Programming Coefficients

Starting with the f_0 and Q values obtained in Step 1, use the program "MPP" to generate the digital coefficients which program each second-order section's f_0 and Q. The program displays values for "N" ("N = __ for f_0" and "N = __ for Q"). N is the decimal equivalent of the binary code that sets the filter section's f_0 or Q. These are the same "N"s that are listed in Tables 2 and 3.

Table 1. Typical Clock and Center Frequency Limits

PART	Q	MODE	f_{CLK}	f_0
MAX260	1	1	1Hz-400kHz	0.01Hz-4.0kHz
	1	2	1Hz-425kHz	0.01Hz-6.0kHz
	1	3	1Hz-500kHz	0.01Hz-5.0kHz
	1	4	1Hz-400kHz	0.01Hz-4.0kHz
	8	1	1Hz-500kHz	0.01Hz-5.0kHz
	8	2	1Hz-700kHz	0.01Hz-10.0kHz
	8	3	1Hz-700kHz	0.01Hz-5.0kHz
	8	4	1Hz-600kHz	0.01Hz-4.0kHz
	64	1	1Hz-750kHz	0.01Hz-7.5kHz
	90	2	1Hz-500kHz	0.01Hz-7.0kHz
	64	3	1Hz-400kHz	0.01Hz-4.0kHz
	64	4	1Hz-750kHz	0.01Hz-7.5kHz
MAX261	1	1	40Hz-4.0MHz	0.4Hz-40kHz
	1	2	40Hz-4.0MHz	0.5Hz-57kHz
	1	3	40Hz-4.0MHz	0.4Hz-40kHz
	1	4	40Hz-4.0MHz	0.4Hz-40kHz
	8	1	40Hz-2.7MHz	0.4Hz-27kHz
	8	2	40Hz-2.1MHz	0.5Hz-30kHz

PART	Q	MODE	f_{CLK}	f_0
MAX261	8	3	40Hz-1.7MHz	0.4Hz-17kHz
	8	4	40Hz-2.7MHz	0.4Hz-27kHz
	64	1	40Hz-2.0MHz	0.4Hz-20kHz
	90	2	40Hz-1.2MHz	0.4Hz-18kHz
	64	3	40Hz-1.2MHz	0.4Hz-12kHz
	64	4	40Hz-2.0MHz	0.4Hz-20kHz
MAX262	1	1	40Hz-4.0MHz	1.0Hz-100kHz
	1	2	40Hz-4.0MHz	1.4Hz-140kHz
	1	3	40Hz-4.0MHz	1.0Hz-100kHz
	1	4	40Hz-4.0MHz	1.0Hz-100kHz
	8	1	40Hz-2.5MHz	1.0Hz-60kHz
	8	2	40Hz-1.4MHz	1.4Hz-50kHz
	8	3	40Hz-1.4MHz	1.0Hz-35kHz
	8	4	40Hz-2.5MHz	1.0Hz-60kHz
	64	1	40Hz-1.5MHz	1.0Hz-37kHz
	90	2	40Hz-0.9MHz	1.4Hz-32kHz
	64	3	40Hz-0.9MHz	1.0Hz-22kHz
	64	4	40Hz-1.5MHz	1.0Hz-37kHz

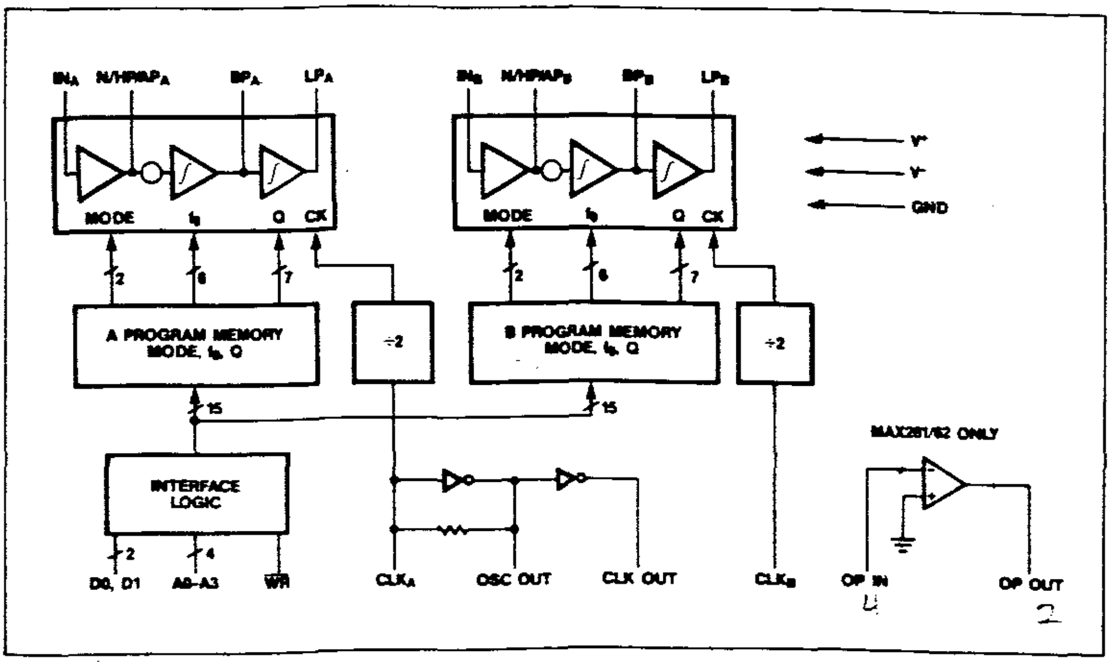


Figure 3. MAX260/61/62 Block Diagram

262: 40 → 140 and 28 → 99
261: 100 → 200 and 71 → 141
(1) 2

Microprocessor Programmable Universal Active Filters

Table 2. f_{CLK}/f_0 Program Selection Table

f_{CLK}/f_0 RATIO				PROGRAM CODE						
MAX260/61		MAX262								
MODE 1,3,4	MODE 2	MODE 1,3,4	MODE 2	N	F5	F4	F3	F2	F1	F0
100.53	71.09	40.84	28.88	0	0	0	0	0	0	0
102.10	72.20	42.41	29.99	1	0	0	0	0	0	1
103.67	73.31	43.98	31.10	2	0	0	0	0	1	0
105.24	74.42	45.55	32.21	3	0	0	0	0	1	1
106.81	75.53	47.12	33.32	4	0	0	0	1	0	0
108.38	76.64	48.69	34.43	5	0	0	0	1	0	1
109.96	77.75	50.27	35.54	6	0	0	0	1	1	0
111.53	78.86	51.84	36.65	7	0	0	0	1	1	1
113.10	79.97	53.41	37.76	8	0	0	1	0	0	0
114.67	81.08	54.98	38.87	9	0	0	1	0	0	1
116.24	82.19	56.55	39.99	10	0	0	1	0	1	0
117.81	83.30	58.12	41.10	11	0	0	1	0	1	1
119.38	84.42	59.69	42.21	12	0	0	1	1	0	0
120.95	85.53	61.26	43.32	13	0	0	1	1	0	1
122.52	86.64	62.83	44.43	14	0	0	1	1	1	0
124.09	87.75	64.40	45.54	15	0	0	1	1	1	1
125.66	88.86	65.97	46.65	16	0	1	0	0	0	0
127.23	89.97	67.54	47.76	17	0	1	0	0	0	1
128.81	91.80	69.12	48.87	18	0	1	0	0	1	0
130.38	92.19	70.69	49.98	19	0	1	0	0	1	1
131.95	93.30	72.26	51.10	20	0	1	0	1	0	0
133.52	94.41	73.83	52.20	21	0	1	0	1	0	1
135.08	95.52	75.40	53.31	22	0	1	0	1	1	0
136.66	96.63	76.97	54.43	23	0	1	0	1	1	1
138.23	97.74	78.53	55.54	24	0	1	1	0	0	0
139.80	98.86	80.11	56.65	25	0	1	1	0	0	1
141.37	99.97	81.68	57.76	26	0	1	1	0	1	0
142.94	101.08	83.25	58.87	27	0	1	1	0	1	1
144.51	102.89	84.82	59.98	28	0	1	1	1	0	0
146.08	103.30	86.39	61.09	29	0	1	1	1	0	1
147.65	104.41	87.96	62.20	30	0	1	1	1	1	0
149.23	105.52	89.54	63.31	31	0	1	1	1	1	1
150.80	106.63	91.11	64.42	32	1	0	0	0	0	0
152.37	107.74	92.68	65.53	33	1	0	0	0	0	1
153.98	108.85	94.25	66.64	34	1	0	0	0	1	0
155.51	109.96	95.82	67.75	35	1	0	0	0	1	1
157.08	111.07	97.39	68.86	36	1	0	0	1	0	0
158.65	112.18	98.96	69.98	37	1	0	0	1	0	1
160.22	113.29	100.53	71.09	38	1	0	0	1	1	0
161.79	114.41	102.10	72.20	39	1	0	0	1	1	1
163.36	115.52	102.67	73.31	40	1	0	1	0	0	0
164.93	116.63	105.24	74.42	41	1	0	1	0	0	1
166.50	117.74	106.81	75.53	42	1	0	1	0	1	0
168.08	118.85	108.38	76.64	43	1	0	1	0	1	1
169.65	119.96	109.96	77.75	44	1	0	1	1	0	0
171.22	121.07	111.53	78.86	45	1	0	1	1	0	1
172.79	122.18	113.10	79.97	46	1	0	1	1	1	0
174.36	123.29	114.66	81.08	47	1	0	1	1	1	1

Table 2. f_{CLK}/f_0 Program Selection Table (Continued)

f_{CLK}/f_0 RATIO				PROGRAM CODE						
MAX260/61		MAX262								
MODE 1,3,4	MODE 2	MODE 1,3,4	MODE 2	N	F5	F4	F3	F2	F1	F0
175.93	124.40	116.24	82.19	48	1	1	0	0	0	0
177.50	125.51	117.81	83.30	49	1	1	0	0	0	1
179.07	126.62	119.38	84.41	50	1	1	0	0	1	0
180.64	127.73	120.95	85.53	51	1	1	0	0	1	1
182.21	128.84	122.52	86.64	52	1	1	0	1	0	0
183.78	129.96	124.09	87.75	53	1	1	0	1	0	1
185.35	131.07	125.66	88.86	54	1	1	0	1	1	0
186.92	132.18	127.23	89.97	55	1	1	0	1	1	1
188.49	133.29	128.81	91.08	56	1	1	1	0	0	0
190.07	134.40	130.38	92.19	57	1	1	1	0	0	1
191.64	135.51	131.95	93.30	58	1	1	1	0	1	0
193.21	136.62	133.52	94.41	59	1	1	1	0	1	1
194.78	137.73	135.09	95.52	60	1	1	1	1	0	0
196.35	138.84	136.66	96.63	61	1	1	1	1	0	1
197.92	139.95	138.23	97.74	62	1	1	1	1	1	0
199.49	141.06	139.80	98.85	63	1	1	1	1	1	1

Notes: 1) For the MAX260/61, $f_{CLK}/f_0 = (64 + N)\pi/2$ in Mode 1, 3, and 4, where N varies from 0 to 63.
2) For the MAX262, $f_{CLK}/f_0 = (26 + N)\pi/2$ in Mode 1, 3, and 4, where N varies 0 to 63.
3) In Mode 2, all f_{CLK}/f_0 ratios are divided by $\sqrt{2}$. The functions are then:
MAX260/61 $f_{CLK}/f_0 = 1.11072\ (64 + N)$, MAX262 $f_{CLK}/f_0 = 1.11072\ (26 + N)$

Table 3. Q Program Selection Table

PROGRAMMED Q		PROGRAM CODE							
MODE 1,3,4	MODE 2	N	Q6	Q5	Q4	Q3	Q2	Q1	Q0
0.500*	0.707*	0*	0	0	0	0	0	0	0
0.504	0.713	1	0	0	0	0	0	0	1
0.508	0.718	2	0	0	0	0	0	1	0
0.512	0.724	3	0	0	0	0	0	1	1
0.516	0.730	4	0	0	0	0	1	0	0
0.520	0.736	5	0	0	0	0	1	0	1
0.525	0.742	6	0	0	0	0	1	1	0
0.529	0.748	7	0	0	0	0	1	1	1
0.533	0.754	8	0	0	0	1	0	0	0
0.538	0.761	9	0	0	0	1	0	0	1
0.542	0.767	10	0	0	0	1	0	1	0
0.547	0.774	11	0	0	0	1	0	1	1
0.552	0.780	12	0	0	0	1	1	0	0
0.556	0.787	13	0	0	0	1	1	0	1
0.561	0.794	14	0	0	0	1	1	1	0
0.566	0.801	15	0	0	0	1	1	1	1
0.571	0.808	16	0	0	1	0	0	0	0
0.577	0.815	17	0	0	1	0	0	0	1
0.582	0.823	18	0	0	1	0	0	1	0
0.587	0.830	19	0	0	1	0	0	1	1
0.593	0.838	20	0	0	1	0	1	0	0
0.598	0.846	21	0	0	1	0	1	0	1
0.604	0.854	22	0	0	1	0	1	1	0
0.609	0.862	23	0	0	1	0	1	1	1

PROGRAMMED Q		PROGRAM CODE							
MODE 1,3,4	MODE 2	N	Q6	Q5	Q4	Q3	Q2	Q1	Q0
0.615	0.870	24	0	0	1	1	0	0	0
0.621	0.879	25	0	0	1	1	0	0	1
0.627	0.887	26	0	0	1	1	0	1	0
0.634	0.896	27	0	0	1	1	0	1	1
0.640	0.905	28	0	0	1	1	1	0	0
0.646	0.914	29	0	0	1	1	1	0	1
0.653	0.924	30	0	0	1	1	1	1	0
0.660	0.933	31	0	0	1	1	1	1	1
0.667	0.943	32	0	1	0	0	0	0	0
0.674	0.953	33	0	1	0	0	0	0	1
0.681	0.963	34	0	1	0	0	0	1	0
0.688	0.973	35	0	1	0	0	0	1	1
0.696	0.984	36	0	1	0	0	1	0	0
0.703	0.995	37	0	1	0	0	1	0	1
0.711	1.01	38	0	1	0	0	1	1	0
0.719	1.02	39	0	1	0	0	1	1	1
0.727	1.03	40	0	1	0	1	0	0	0
0.736	1.04	41	0	1	0	1	0	0	1
0.744	1.05	42	0	1	0	1	0	1	0
0.753	1.06	43	0	1	0	1	0	1	1
0.762	1.08	44	0	1	0	1	1	0	0
0.771	1.09	45	0	1	0	1	1	0	1
0.780	1.10	46	0	1	0	1	1	1	0
0.790	1.12	47	0	1	0	1	1	1	1

Notes: 4) * Writing all 0s into Q0A-Q6A on Filter A activates a low power shutdown mode. BOTH filter sections are deactivated. Therefore this Q value is only achievable in filter B.

Microprocessor Programmable Universal Active Filters

Table 3. Q Program Selection Table (Continued)

PROGRAMMED Q		PROGRAM CODE							
MODE 1,3,4	MODE 2	N	Q6	Q5	Q4	Q3	Q2	Q1	Q0
0.800	1.13	48	0	1	1	0	0	0	0
0.810	1.15	49	0	1	1	0	0	0	1
0.821	1.16	50	0	1	1	0	0	1	0
0.831	1.18	51	0	1	1	0	0	1	1
0.842	1.19	52	0	1	1	0	1	0	0
0.853	1.21	53	0	1	1	0	1	0	1
0.865	1.22	54	0	1	1	0	1	1	0
0.877	1.24	55	0	1	1	0	1	1	1
0.889	1.26	56	0	1	1	1	0	0	0
0.901	1.27	57	0	1	1	1	0	0	1
0.914	1.29	58	0	1	1	1	0	1	0
0.928	1.31	59	0	1	1	1	0	1	1
0.941	1.33	60	0	1	1	1	1	0	0
0.955	1.35	61	0	1	1	1	1	0	1
0.969	1.37	62	0	1	1	1	1	1	0
0.985	1.39	63	0	1	1	1	1	1	1
1.00	1.41	64	1	0	0	0	0	0	0
1.02	1.44	65	1	0	0	0	0	0	1
1.03	1.46	66	1	0	0	0	0	1	0
1.05	1.48	67	1	0	0	0	0	1	1
1.07	1.51	68	1	0	0	0	1	0	0
1.08	1.53	69	1	0	0	0	1	0	1
1.10	1.56	70	1	0	0	0	1	1	0
1.12	1.59	71	1	0	0	0	1	1	1
1.14	1.62	72	1	0	0	1	0	0	0
1.16	1.65	73	1	0	0	1	0	0	1
1.19	1.68	74	1	0	0	1	0	1	0
1.21	1.71	75	1	0	0	1	0	1	1
1.23	1.74	76	1	0	0	1	1	0	0
1.25	1.77	77	1	0	0	1	1	0	1
1.28	1.81	78	1	0	0	1	1	1	0
1.31	1.85	79	1	0	0	1	1	1	1
1.33	1.89	80	1	0	1	0	0	0	0
1.36	1.93	81	1	0	1	0	0	0	1
1.39	1.97	82	1	0	1	0	0	1	0
1.42	2.01	83	1	0	1	0	0	1	1
1.45	2.06	84	1	0	1	0	1	0	0
1.49	2.10	85	1	0	1	0	1	0	1
1.52	2.16	86	1	0	1	0	1	1	0
1.56	2.21	87	1	0	1	0	1	1	1
1.60	2.26	88	1	0	1	1	0	0	0
1.64	2.32	89	1	0	1	1	0	0	1
1.68	2.40	90	1	0	1	1	0	1	0
1.73	2.45	91	1	0	1	1	0	1	1
1.78	2.51	92	1	0	1	1	1	0	0
1.83	2.59	93	1	0	1	1	1	0	1
1.88	2.66	94	1	0	1	1	1	1	0
1.94	2.74	95	1	0	1	1	1	1	1
2.00	2.83	96	1	1	0	0	0	0	0
2.06	2.92	97	1	1	0	0	0	0	1
2.13	3.02	98	1	1	0	0	0	1	0
2.21	3.12	99	1	1	0	0	0	1	1
2.29	3.23	100	1	1	0	0	1	0	0
2.37	3.35	101	1	1	0	0	1	0	1
2.46	3.48	102	1	1	0	0	1	1	0
2.56	3.62	103	1	1	0	0	1	1	1
2.67	3.77	104	1	1	0	1	0	0	0
2.78	3.96	105	1	1	0	1	0	0	1
2.91	4.11	106	1	1	0	1	0	1	0
3.05	4.31	107	1	1	0	1	0	1	1
3.20	4.53	108	1	1	0	1	1	0	0
3.37	4.76	109	1	1	0	1	1	0	1
3.56	5.03	110	1	1	0	1	1	1	0
3.76	5.32	111	1	1	0	1	1	1	1
4.00	5.66	112	1	1	1	0	0	0	0
4.27	6.03	113	1	1	1	0	0	0	1
4.57	6.46	114	1	1	1	0	0	1	0
4.92	6.96	115	1	1	1	0	0	1	1
5.33	7.54	116	1	1	1	0	1	0	0
5.82	8.23	117	1	1	1	0	1	0	1
6.40	9.05	118	1	1	1	0	1	1	0
7.11	10.1	119	1	1	1	0	1	1	1
8.00	11.3	120	1	1	1	1	0	0	0
9.14	12.9	121	1	1	1	1	0	0	1
10.7	15.1	122	1	1	1	1	0	1	0
12.8	18.1	123	1	1	1	1	0	1	1
16.0	22.6	124	1	1	1	1	1	0	0
21.3	30.2	125	1	1	1	1	1	0	1
32.0	45.3	126	1	1	1	1	1	1	0
64.0	90.5	127	1	1	1	1	1	1	1

Notes: 5) In Modes 1, 3, and 4: Q = 64/(128-N)
6) In Mode 2, the listed Q values are those of Mode 1 multiplied by $\sqrt{2}$. Then Q = 90.51/(128-N)

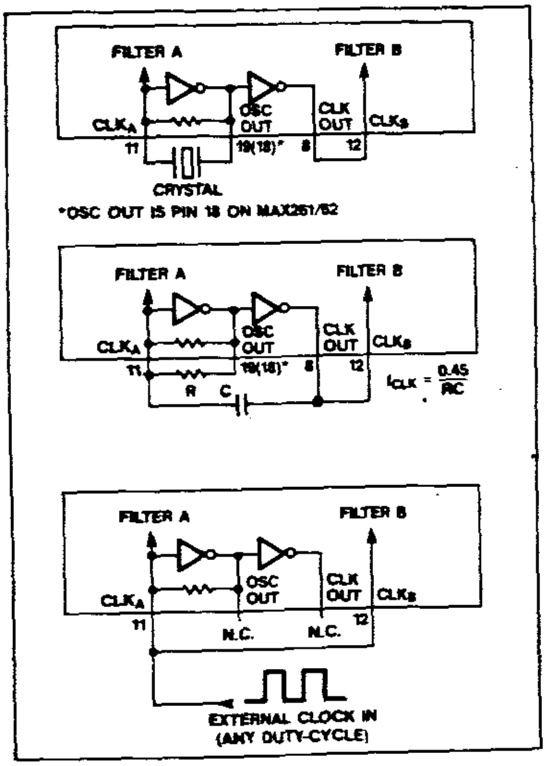


Figure 4. Clock Input Connections

Oscillator and Clock Inputs

The clock circuitry of the MAX260/61/62 can operate with a crystal, resistor-capacitor (RC) network, or an external clock generator as shown in Figure 4. If an RC oscillator is used, the clock rate, f_{CLK}, nominally equals 0.45/RC.

The duty cycle of the clock at CLK_A and CLK_B is unimportant because the input is internally divided by two to generate the sampling clock for each filter section. It is important to note that this internal division also halves the sample rate when considering aliasing and other sampled system phenomenon.

Microprocessor Interface

f_0, Q, and Mode selection data is stored in an internal program memory. The memory contents are updated by writing to addresses selected by A0-A3. D0 and D1 are the data inputs. A map of the memory locations is shown in Table 4. Data is stored in the selected address on the rising edge of $\overline{WR}$. Address and data inputs are TTL and CMOS compatible when the filter is powered from ±5 volts. With other power supply voltages, CMOS logic levels should be used. Interface timing is shown in Figure 5. Note: Clock inputs CLK_A and CLK_B have no relation to the digital interface. They control the switched-capacitor filter sample rate only.

Some noise may be generated on the filter outputs by transitions at the logic inputs. If this is objectionable, the digital lines should be buffered from the device by logic gates as shown in Figure 6.

Table 4. Program Address Locations

DATA BIT		ADDRESS				LOCATION
D0	D1	A3	A2	A1	A0	
FILTER A						
$M0_A$	$M1_A$	0	0	0	0	0
$F0_A$	$F1_A$	0	0	0	1	1
$F2_A$	$F3_A$	0	0	1	0	2
$F4_A$	$F5_A$	0	0	1	1	3
$Q0_A$	$Q1_A$	0	1	0	0	4
$Q2_A$	$Q3_A$	0	1	0	1	5
$Q4_A$	$Q5_A$	0	1	1	0	6
$Q6_A$		0	1	1	1	7
FILTER B						
$M0_B$	$M1_B$	1	0	0	0	8
$F0_B$	$F1_B$	1	0	0	1	9
$F2_B$	$F3_B$	1	0	1	0	10
$F4_B$	$F5_B$	1	0	1	1	11
$Q0_B$	$Q1_B$	1	1	0	0	12
$Q2_B$	$Q3_B$	1	1	0	1	13
$Q4_B$	$Q5_B$	1	1	1	0	14
$Q6_B$		1	1	1	1	15

Note: Writing 0 into $Q0_A$-$Q6_A$ (address locations 4-7) on Filter A activates shutdown mode. BOTH filter sections deactivate.

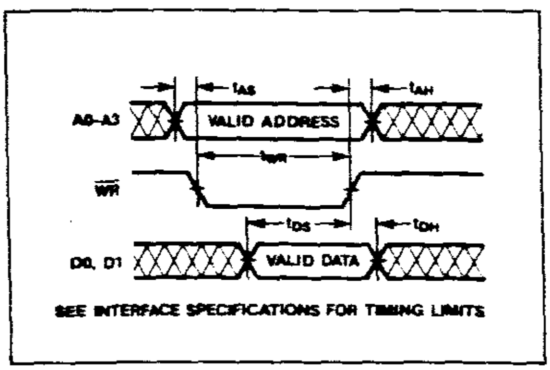


Figure 5. Interface Timing

Microprocessor Programmable Universal Active Filters

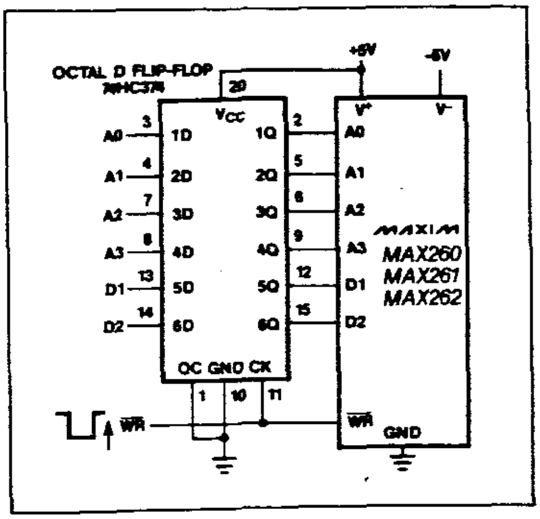


Figure 6. Buffering/Latching Logic Inputs

Shutdown Mode

The MAX260/61/62 enters a shutdown/standby mode when all zeroes are written to the Q addresses of filter A ($Q0_A$-$Q6_A$). When shut down, power consumption with ±5V supplies typically drops to 10mW. When reactivating the filter after shutdown, allow 2ms to return to full operation.

Filter Operating Modes

There are several ways in which the summing amplifier and integrators in each MAX260/61/62 filter section can be configured. The four most versatile interconnections (modes) are selected by writing to inputs M0 and M1 (See Tables 4 and 5). These modes use no external components. A fifth mode, 3A, makes use of an additional op-amp (included in the MAX261 and 262) and external resistors but uses the same internal configuration, and is selected with the same programming code, as Mode 3.

Figures 7 through 11 show symbolic representations of the MAX260 filter modes. Only one second-order section is shown in each case. The A and B sections of one MAX260/61/62 can be programmed for different modes if desired. The f_0, f_N (notch), Q, and various output gains in each case are shown in Table 5.

Filter Mode Selection

MODE 1 (Figure 7) is useful when implementing all-pole lowpass and bandpass filters such as Butterworth, Chebyshev, Bessel, etc.. It can also be used for notch filters, but only second-order notches because the relative pole and zero locations are fixed. Higher order notch filters require more latitude in f_0 and f_N, which is why they are more easily implemented with Mode 3A.

Mode 1, along with Mode 4, supports the highest clock frequencies (See Table 1) because the input summing amplifier is outside the filter's resonant loop (Figure 7). The gain of the lowpass and notch outputs

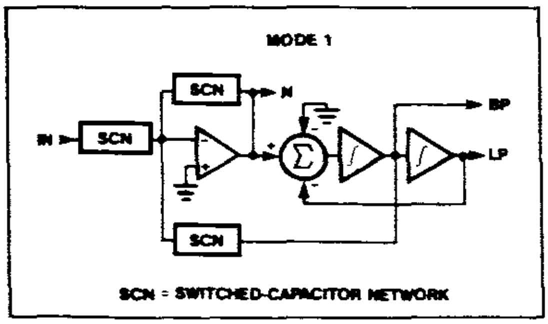


Figure 7. Filter Mode 1: Second-Order Bandpass, Lowpass and Notch

Table 5. Filter Modes for Second-Order Functions

MODE	M1, M0	FILTER FUNCTIONS	f_0	Q	f_N	H_{OLP}	H_{OBP}	H_{ON1} ($f \to 0$)	H_{ON2} ($f \to f_{CLK}/4$)	OTHER
1	0, 0	LP, BP, N	SEE TABLE 2	SEE TABLE 3	f_0	-1	-Q	-1	-1	
2	0, 1	LP, BP, N			$f_0\sqrt{2}$	-0.5	$-Q/\sqrt{2}$	-0.5	-1	
3	1, 0	LP, BP, HP				-1	-Q			$H_{OHP} = -1$
3A	1, 0	LP, BP, HP, N			$f_0\sqrt{\frac{R_H}{R_L}}$	-1	-Q	$+\frac{R_G}{R_L}$	$+\frac{R_G}{R_H}$	$H_{OHP} = -1$
4	1, 1	LP, BP, AP				-2	-2Q			$H_{OAP} = -1$; $f_Z = f_0$, $Q_Z = Q$

Notes:
f_0 = Center Frequency
f_N = Notch Frequency
H_{OLP} = Lowpass Gain at DC
H_{OBP} = Bandpass Gain at f_0
H_{OHP} = Highpass Gain as f approaches $f_{CLK}/4$
H_{ON1} = Notch Gain as f approaches DC
H_{ON2} = Notch Gain as f approaches $f_{CLK}/4$
H_{OAP} = Allpass Gain
f_Z, Q_Z = f and Q of Complex Pole Pair

is 1, while the bandpass gain at the center frequency is Q. For bandpass gains other than Q, the filter input or output can be scaled by a resistive divider or op-amp.

MODE 2 (Figure 8) is used for all-pole lowpass and bandpass filters. Key advantages compared to Mode 1 are higher available Qs (See Table 3) and lower output noise. Mode 2's available f_{CLK}/f_0 ratios are $\sqrt{2}$ less than with Mode 1 (See Table 2) so a wider overall range of f_0s can be selected from a single clock when both modes are used together. This is demonstrated in the Wide Passband Chebyshev Bandpass design example.

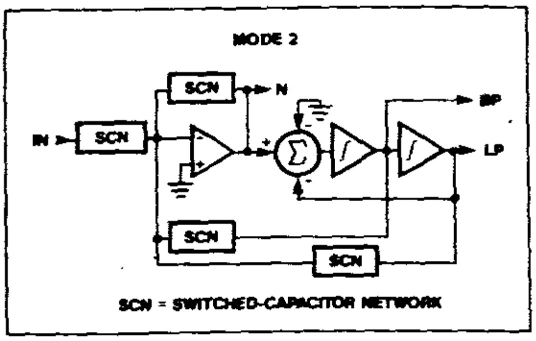


Figure 8. Filter Mode 2: Second-Order Bandpass, Lowpass and Notch

MODE 3 (Figure 9) is the only mode which produces high-pass filters. The maximum clock frequency is somewhat less than with MODE 1 (See Table 1).

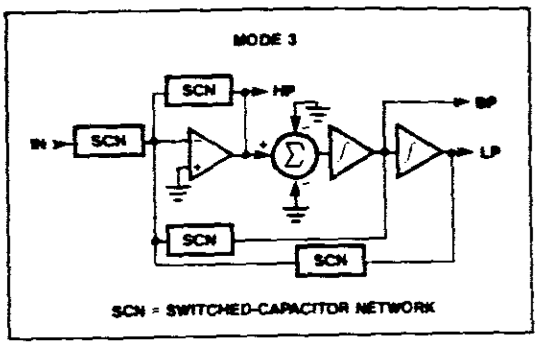


Figure 9. Filter Mode 3: Second-Order Bandpass, Lowpass and Highpass

MODE 3A (Figure 10) uses a separate op-amp to sum the highpass and lowpass outputs of Mode 3, creating a separate notch output. This output allows the notch to be set independently of f_0 by adjusting the op-amp's feedback resistor ratio (R_H, R_L). R_H, R_L, and R_G are external resistors. Because the notch can be independently set, Mode 3A is also useful when designing pole-zero filters such as elliptics.

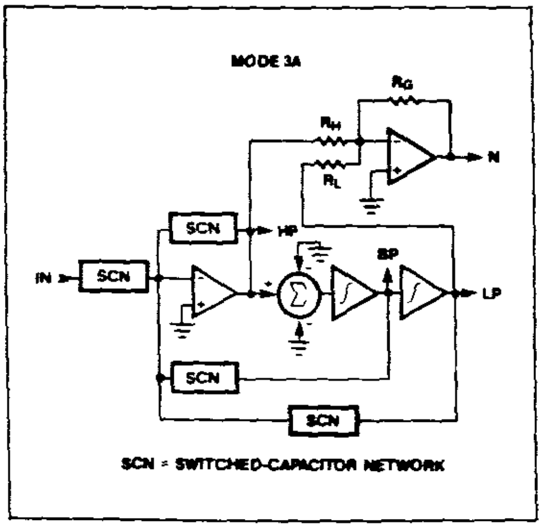


Figure 10. Filter Mode 3A: Second-Order Bandpass, Lowpass, Highpass and Notch. For elliptic LP, BP, HP and Notch, the N output is used

MODE 4 (Figure 11) is the only mode that provides an allpass output. This is useful when implementing group delay equalization. In addition to this, Mode 4 can also be used in all pole lowpass and bandpass filters. Along with Mode 1, it is the fastest operating mode for the filter, although the gains are different than in Mode 1. When the allpass function is used, note that some amplitude peaking occurs (approximately 0.3dB when Q = 8) at f_0. Also note that f_0 and Q sampling errors are highest in Mode 4 (See Figure 20).

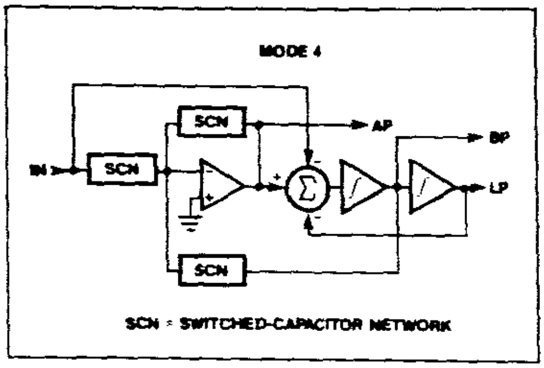


Figure 11. Filter Mode 4: Second-Order Bandpass, Lowpass and Allpass

Microprocessor Programmable Universal Active Filters

Description of Filter Functions

BANDPASS (Figure 12)

For all pole bandpass and lowpass filters (Butterworth, Bessel, Chebyshev) use Mode 1 if possible. If appropriate f_{CLK}/f_0 or Q values are not available in Mode 1, Mode 2 may provide a selection that is closer to the required values. Mode 1 however has the highest bandwidth (See Table 1). For pole-zero filters such as elliptics see Mode 3A.

$$G(s) = H_{OBP} \frac{s(\omega_o/Q)}{s^2 + s(\omega_o/Q) + \omega_o^2}$$

H_{OBP} = Bandpass output gain at $\omega = \omega_o$

f_0 = $\omega_o/2\pi$ = The center frequency of the complex pole pair. Input-output phase shift is -180° at f_0.

Q = The quality factor of the complex pole pair. Also the ratio of f_0 to -3dB bandwidth of the second-order bandpass response.

LOWPASS See Bandpass text. (Figure 13)

$$G(s) = H_{OLP} \frac{\omega_o^2}{s^2 + s(\omega_o/Q) + \omega_o^2}$$

H_{OLP} = Lowpass output gain at DC

f_0 = $\omega_o/2\pi$

HIGHPASS (Figure 14)

Mode 3 is the only mode with a highpass output. It will work for all pole filter types such as Butterworth, Bessel and Chebyshev. Use mode 3A for filters employing both poles and zeros such as elliptics.

$$G(s) = H_{OHP} \frac{s^2}{s^2 + s(\omega_o/Q) + \omega_o^2}$$

H_{OHP} = Highpass output gain as f approaches $f_{CLK}/4$

f_0 = $\omega_o/2\pi$

NOTCH (Figure 15)

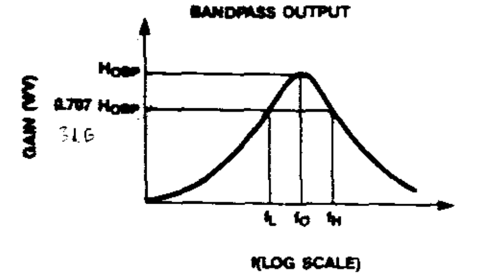
Mode 3A is recommended for multi-pole notch filters. In 2nd order filters, Mode 1 can also be used. The advantages of Mode 1 are higher bandwidth compared to mode 3 (Higher f_N can be implemented) and no need for external components as required in Mode 3A.

$$G(s) = H_{ON2} \frac{s^2 + \omega_n^2}{s^2 + s(\omega_o/Q) + \omega_o^2}$$

H_{ON2} = Notch output gain as f approaches $f_{CLK}/4$

H_{ON1} = Notch output gain as f approaches DC

f_n = $\omega_n/2\pi$

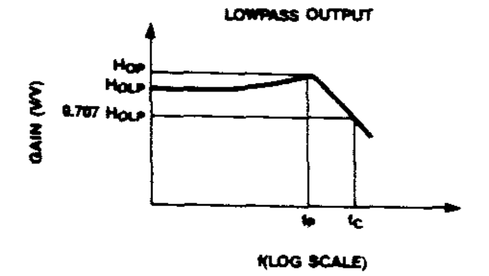


$$Q = \frac{f_0}{f_H - f_L},\ f_0 = \sqrt{f_L f_H}$$

$$f_L = f_0\left[\frac{-1}{2Q} + \sqrt{\left(\frac{1}{2Q}\right)^2 + 1}\right]$$

$$f_H = f_0\left[\frac{1}{2Q} + \sqrt{\left(\frac{1}{2Q}\right)^2 + 1}\right]$$

Figure 12. Second-Order Bandpass Characteristics

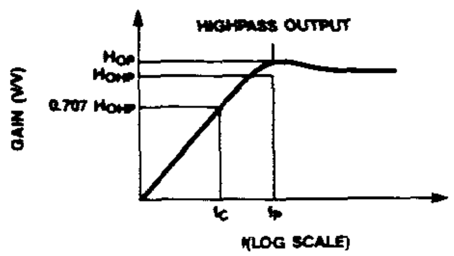


$$f_C = f_0 \times \sqrt{\left(1 - \frac{1}{2Q^2}\right) + \sqrt{\left(1 - \frac{1}{2Q^2}\right)^2 + 1}}$$

$$f_P = f_0\sqrt{1 - \frac{1}{2Q^2}}$$

$$H_{OP} = H_{OLP} \times \frac{1}{\frac{1}{Q}\sqrt{1 - \frac{1}{4Q^2}}}$$

Figure 13. Second-Order Lowpass Characteristics



$$f_C = f_0 \times \left[\sqrt{\left(1 - \frac{1}{2Q^2}\right) + \sqrt{\left(1 - \frac{1}{2Q^2}\right)^2 + 1}}\right]^{-1}$$

$$f_P = f_0 \times \left[\sqrt{1 - \frac{1}{2Q^2}}\right]^{-1}$$

$$H_{OP} = H_{OHP} \times \frac{1}{\frac{1}{Q}\sqrt{1 - \frac{1}{4Q^2}}}$$

Figure 14. Second-Order Highpass Characteristics

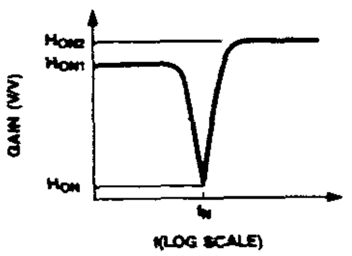


Figure 15. Second-Order Notch Characteristics

ALL PASS

Mode 4 is the only configuration in which an allpass function can be realized.

$$G(s) = H_{OAP}\frac{s^2 - s(\omega_0/Q) + \omega_0^2}{s^2 + s(\omega_0/Q) + \omega_0^2}$$

H_{OAP} = All pass output gain for DC < f < $f_{CLK}/4$

$f_0 = \omega_0/2\pi$

Filter Design Procedure

The procedure for most filter designs is to first convert the required frequency response specifications to f_0s and Qs for the appropriate number of second-order sections that implement the filter. This can be done by using design equations or tables in available literature, or can be conveniently calculated using Maxim's filter design software. Once the f_0 and Qs have been found, the next step is to turn them into the digital program coefficients required by the MAX260/61/62. An operating Mode and clock frequency (or clock/center frequency ratio) must also be selected.

Next, if the sample rate ($f_{CLK}/2$) is low enough to cause significant errors, the selected f_0s and Qs should be corrected to account for sampling effects by using Figure 20 or Maxim's design software. In most cases, the sampling errors are small enough to require no correction, i.e. less than 1%. In any case, with or without correction, the required f_0s and Qs can then be selected from Tables 2 and 3. Maxim's filter design software can also perform this last step. The desired f_0s and Qs are stated, and the appropriate digital coefficients are supplied.

Cascading Filters

In some designs, such as very narrow band filters, several second-order sections with identical center frequency may be cascaded. The total Q of the resultant filter is:

$$\text{Total } Q_T = \frac{Q}{\sqrt{(2^{1/N} - 1)}}$$

Q is the Q of each individual filter section, and N is the number of sections. In Table 6, the total Q and bandwidth are listed for up to five identical second-order sections. B is the bandwidth of each section.

Table 6. Cascading Identical Bandpass Filter Sections

Total Sections	Total B.W.	Total Q
1	1.000 B	1.00 Q
2	0.644 B	1.55 Q
3	0.510 B	1.96 Q
4	0.435 B	2.30 Q
5	0.386 B	2.60 Q

Note: B = individual stage bandwidth, Q = individual stage Q.

Microprocessor Programmable Universal Active Filters

In high order bandpass filters, stages with different f_0s and Qs are also often cascaded. When this happens the overall filter gain at the bandpass center frequency is not simply the product of the individual gains because f_0, the frequency where each section's gain is specified, is different for each second-order section. The gain of each section at the cascaded filter's center frequency must be determined to obtain the total gain.

For all-pole filters the gain, $H(f_0)$, at each second-order section's f_0 is divided by an adjustment factor, G, to obtain that section's gain, $H(f_{0BP})$, at the overall center frequency:

$H_1(f_{0BP}) = H(f_{01})/G_1$ = Section 1's Gain at f_{0BP}

$$G_1 = \frac{Q_1[(F_1^2 - 1)^2 + (F_1/Q_1)^2]^{1/2}}{F_1}$$

where $F_1 = f_{01}/f_{0BP}$

G_1, Q_1, and f_{01} are the gain adjustment factor, Q, and f_0 for the first of the cascaded second-order sections. The gain of the other sections (2, 3 etc.) at f_{0BP} is determined the same way. The overall gain is:

$H(f_{0BP}) = H_1(f_{0BP}) \times H_2(f_{0BP}) \times$ etc.

For cascaded filters with zeros (f_Z) such as elliptics, the gain adjustment factor for each stage is:

$$G_1 = \frac{Q_1[F_{Z1}^2 - F_1^2]\,[(F_1^2 - 1)^2 + (F_1/Q_1)^2]^{1/2}}{F_1^2(F_{Z1}^2 - 1)}$$

where $F_{1Z} = f_{Z1}/f_{0BP}$, and F_1 is the same as above.

Application Hints

Power Supplies

The MAX260/61/62 can be operated with a variety of power supply configurations including +5V to +12V single supply, or ±2.5V to ±5V dual supplies. When a single supply is used, V⁻ is connected to system ground and the filter's GND pin should be biased at V⁺/2. The input signal is then either capacitively coupled to the filter input or biased to V⁺/2. Figure 16 shows circuit connections for single supply operation.

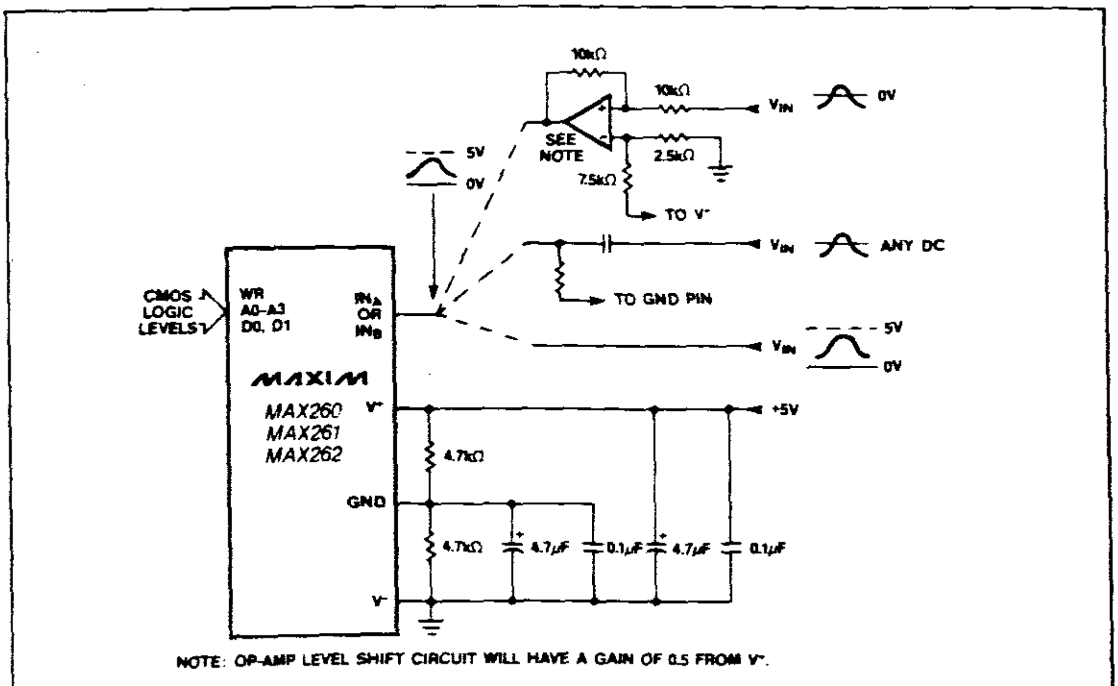


Figure 16. Power Supply and Input Connections for Single Supply Operation

When power supplies other than ±5V are used, CMOS input logic levels (HIGH = V^+, LOW = GND or V^-) are required for $\overline{WR}$, D0-D1, A0-A3, CLK_A and CLK_B. With ±5V supplies, either TTL or CMOS levels can be used. Note however that power consumption at ±5V is reduced if CLK_A and CLK_B are driven with ±5V, rather than TTL or 0 to 5V levels. Operation with +5V or ±2.5V power lowers power consumption but also reduces bandwidth by approximately 25% compared to +12V or ±5V supplies.

Best performance is achieved if V^+ and V^- are bypassed to ground with 4.7μF electrolytic (Tantalum is preferred.) and 0.1μF ceramic capacitors. These should be located as close to the supply pins as possible. The lead length of the bypass capacitors should be shortest at the V^+ and V^- pins. When using a single supply V^+ and GND should be bypassed to V^- as shown in Figure 16.

Output Swing and Clipping

MAX260/61/62 outputs are designed to drive 10kΩ loads. For the MAX261 and MAX262, all filter outputs swing to within 0.15V of each supply rail with a 10kΩ load. In the MAX260 only, an internal sample-hold circuit reduces voltage swing at the N/HP/AP output compared to LP and BP. N/HP/AP therefore swings to within 1V (10kΩ load) of either rail on the MAX260.

To ensure that the outputs are not driven beyond their maximum range (output clipping), the peak amplitude response, individual section gains (H_{OBP}, H_{OLP}, H_{OHP}), input signal level, and filter offset voltages must be carefully considered. It is especially important to check UNUSED outputs for clipping (i.e. the lowpass output in a bandpass hookup) because overload at ANY filter stage severely distorts the overall response. The maximum signal swing with ±4.75V supplies and a 1.0V filter offset is approximately ±3.5V.

For example lets assume a fourth-order lowpass filter is being implemented with a Q of 2 using Mode 1. With a single 5V supply (i.e. ±2.5V with respect to chip GND) the maximum output signal is ±2V (w.r.t. GND). Since in Mode 1 the maximum signal is Q times the input signal, the input should not exceed ±(2/Q)V, or ±1V in this case.

Clock Feedthrough and Noise

Typical wideband noise for MAX260 series devices is $0.5mV_{pp}$ from DC to 100kHz. The noise is virtually independent of clock frequency. In multistage filters, the section with the highest Q should be placed first for lower output noise.

The output waveform of the MAX260 series and other switched capacitor filters appears as a sampled signal with stepping or "staircasing" of the output waveform occurring at the internal sample rate ($f_{CLK}/2$). This stepping, if objectionable, can be removed by adding a single pole RC filter. With no input signal, clock related feedthrough is approximately $8mV_{pp}$. This can also be attenuated with an RC smoothing filter as shown with the MAX261 in Figure 17.

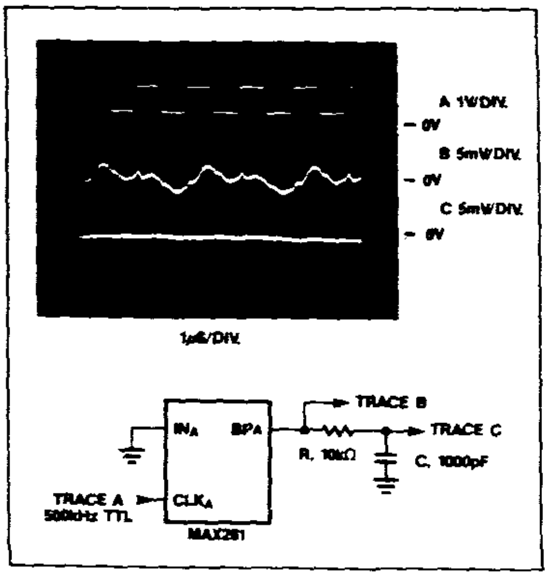


Figure 17. MAX261 Bandpass Output Clock Noise

Some noise also may be generated at the filter outputs by transitions at the logic inputs. If this is objectionable, the digital lines should be buffered from the device by logic gates as shown in Figure 6.

Input Impedance

The input to each filter is the switched capacitor circuit shown in Figure 18. In the MAX260, the input capacitor charges to the input voltage V_{IN} during the first half clock cycle. During the second half-cycle its charge is transferred to the feedback capacitor. The resultant input impedance can be approximated by:

$$R_{IN} = 1/(C_{IN}f_{CLK}/2) = 2/(C_{IN}f_{CLK}).$$

C_{IN} is around 12pF, hence for a clock frequency of 500kHz, R_{IN} = 333kΩ. The input also has about 5pF of fixed capacitance to ground.

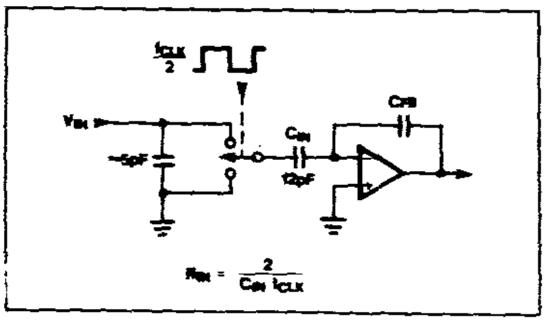


Figure 18. MAX260 Input Model

Microprocessor Programmable Universal Active Filters

The MAX261/262 input structure is shown in Figure 19. Here C_A = 12pF and C_B = 0.016pF and only C_B is switched, so the input resistance is 750 times larger compared to the MAX260 (R_{IN} = 250MΩ). The MAX261/62 has a fixed capacitance of approximately 5pF to ground.

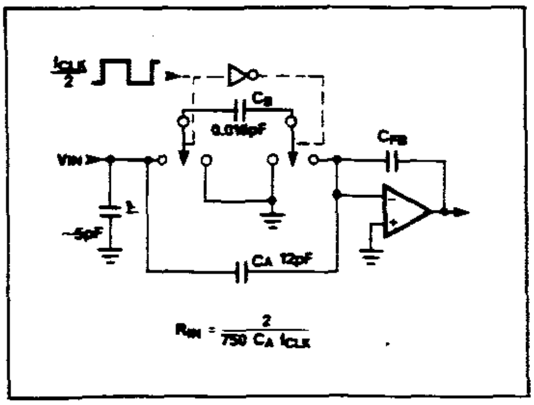


Figure 19. MAX261/262 Input Model

f_0 and Q at Low Sample Rates

When low f_{CLK}/f_0 ratios and low Q settings are selected, deviation from ideal continuous filter response may be noticeable in some designs. This is due to interaction between Q, and f_0 at low f_{CLK}/f_0 ratios and Qs. The data in Figure 20 quantifies these differences. Since the errors are predictable, the graphs can be used to correct the selected f_0 and Q so that the actual realized parameters are on target. These predicted errors are not unique to MAX260 series devices and in fact occur with all types of sampled filters. Consequently, these corrections can be applied to other switched-capacitor filters. In the majority of cases, the errors are not significant, i.e. less than 1%, and correction is not needed. However, the MAX262 does employ a lower range of f_{CLK}/f_0 ratios than the MAX260 or MAX261 and is more prone to sampling errors as the tables show.

Maxim's filter design software applies the previous corrections automatically as a function of desired f_{CLK}/f_0, and Q. Therefore, Figure 20 should NOT be used when Maxim's software determines f_0 and Q. This results in overcompensation of the sampling errors since the correction factors are then counted twice.

The data plotted in Figure 20 applies for Modes 1 and 3. When using Figure 20 for Mode 4, the f_0 error obtained from the graph should be multiplied by 1.5 and the Q error should be multiplied by 3.0. In Mode 2 the value of f_{CLK}/f_0 should be multiplied by $\sqrt{2}$ and the programmed Q should be divided by $\sqrt{2}$ before using the graphs.

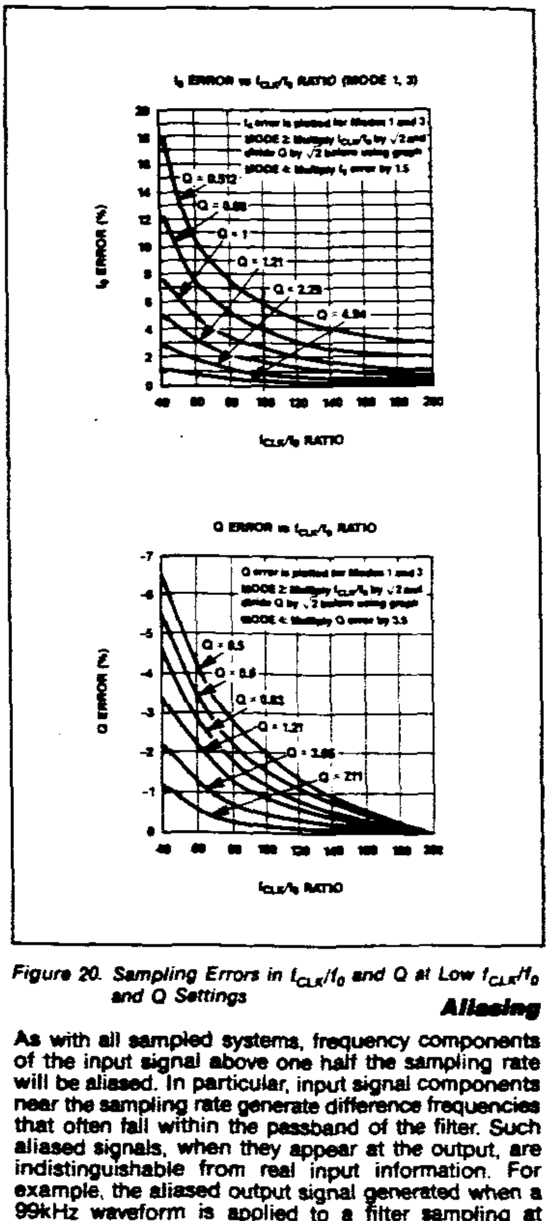


Figure 20. Sampling Errors in f_{CLK}/f_0 and Q at Low f_{CLK}/f_0 and Q Settings

Aliasing

As with all sampled systems, frequency components of the input signal above one half the sampling rate will be aliased. In particular, input signal components near the sampling rate generate difference frequencies that often fall within the passband of the filter. Such aliased signals, when they appear at the output, are indistinguishable from real input information. For example, the aliased output signal generated when a 99kHz waveform is applied to a filter sampling at 100kHz, (f_{CLK} = 200kHz) is 1kHz. This waveform is an attenuated version of the output that would result from a true 1kHz input. Remember that with the MAX260 series filters, the nyquist rate (one half the sample rate) is in fact $f_{CLK}/4$ because f_{CLK} is internally divided by two.

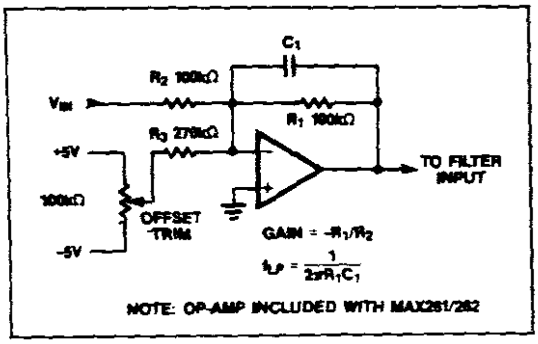


Figure 21. Circuit for DC Offset Adjustment

A simple passive RC lowpass input filter is usually sufficient to remove input frequencies that can cause aliasing. In many cases the input signal itself may be band limited and require no special anti-alias filtering. The wideband MAX262 uses lower f_{CLK}/f_0 ratios than the MAX260/61 and for this reason is more likely to require input filtering than the MAX260 or MAX261.

Trimming DC Offset

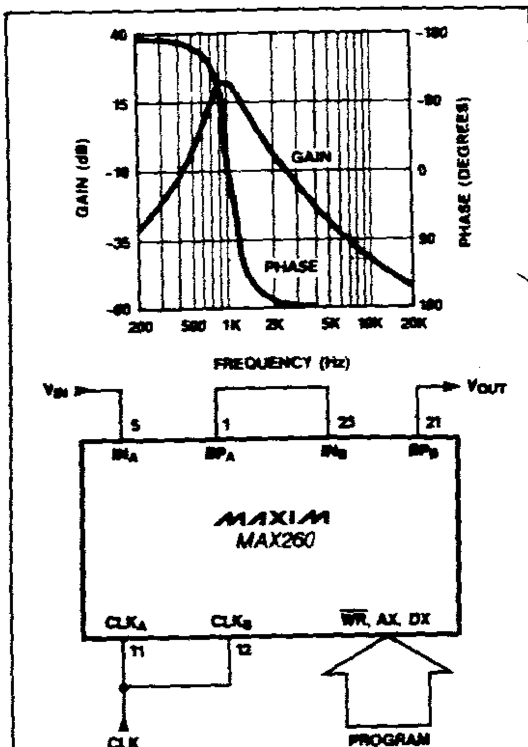
The DC offset voltage at the LP or Notch output can be adjusted with the circuit in Figure 21. This circuit also uses the input op-amp to implement a single pole anti-alias filter. Note that the total offset will generally be less in multistage filters than when only one section is used since each offset is typically negative and each section inverts. When the HP or BP outputs are used, the offset can be removed with capacitor coupling.

Design Examples

Fourth-Order Chebyshev Bandpass Filter

Figure 22 shows both halves of a MAX260 cascaded to form a fourth-order Chebyshev bandpass filter. The desired parameters are:

Center frequency (f_0)	= 1 kHz
Pass bandwidth	= 200 Hz
Stop Bandwidth	= 600 Hz
Max. passband ripple	= 0.5 dB
Min. stopband Attenuation	= 15 dB



$CLK_{A,B}$	MODE	f_{0A}	f_{0B}	Q_A	Q_B
150kHz	1	N = 42	N = 23	N = 119	N = 119

Figure 22. Fourth-Order Chebyshev Bandpass Filter

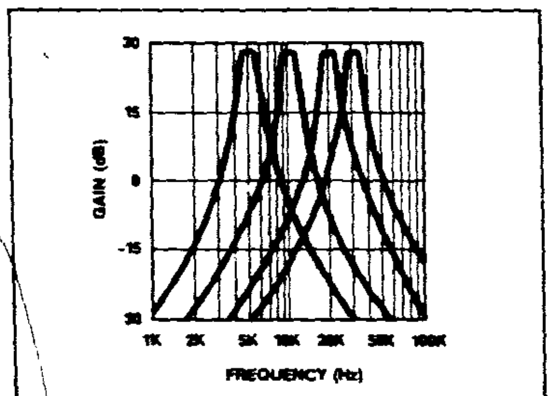


Figure 23. MAX261 Fourth-Order Chebyshev Bandpass Using Coefficients of Figure 22.)

From the above parameters, the order (number of poles), and the f_0 and Q of each section can be determined. Such a derivation is beyond the scope of this data sheet, however there are a number of sources which provide design data for this procedure. These include look-up tables, design texts and computer programs. Design software is available from Maxim to provide comprehensive solutions for most popular filter configurations. The A and B section parameters for the above filter are:

f_{0A} = 904 Hz $\quad$ f_{0B} = 1106 Hz

Q_A = 7.05 $\quad$ Q_B = 7.05

Microprocessor Programmable Universal Active Filters

To implement this filter, both halves operate in Mode 1 and use the same clock. See selection Tables 2 and 3. The programmed parameters are:

$CLK_A = CLK_B = 150$ kHz
$f_{CLK}/f_{0A} = 166.50$ (Mode 1, N=42), actual $f_{0A} = 902.4$Hz
$f_{CLK}/f_{0B} = 136.66$ (Mode 1, N=23), actual $f_{0B} = 1099.7$Hz
$Q_A = Q_B = 7.11$ (Mode 1, N=119)

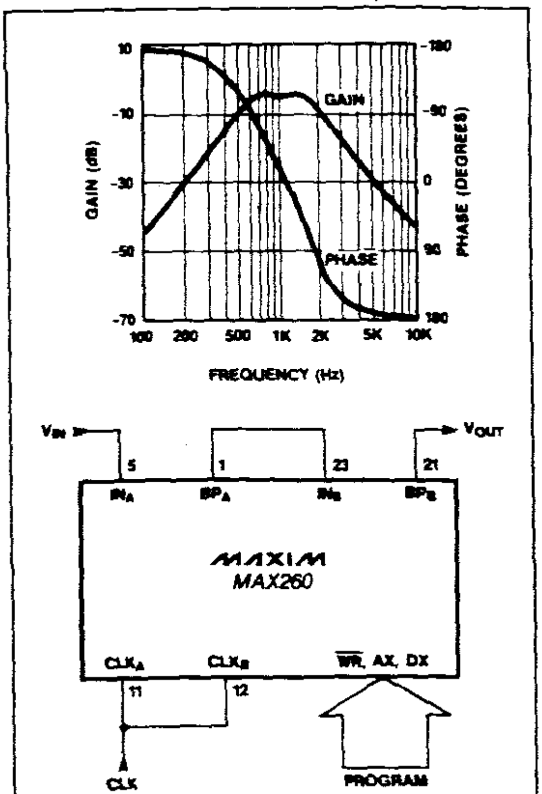
Sampling errors are very small at this f_{CLK}/f_0 ratio so the actual realized Q is very close to 7.05 (See Figure 20 or Filter Program MPP). Often the realized Q will not be exactly the target value at high Qs because programming resolution lowers as Q increases. This doesn't affect most filter designs, since 3-digit Q accuracy is practically never required, and a Q resolution of 1 is provided up to Qs of 10. The overall filter gain at f_0 is 16.4V/V or 24.3dB (See Cascading Filters section). If another gain is required, amplification or attenuation must be added at the input, output, or between stages.

In Figure 23, a series of response curves are shown for the above configuration using a MAX261 with clock frequencies ranging from 750kHz to 4MHz (f_0 from 500Hz to 30kHz). Note that the rightmost curve shows about 2dB of gain peaking compared to the lower frequency curves, indicating the upper limit of usable filter accuracy at this Q (See Table 1)

Wide Passband Chebyshev Bandpass

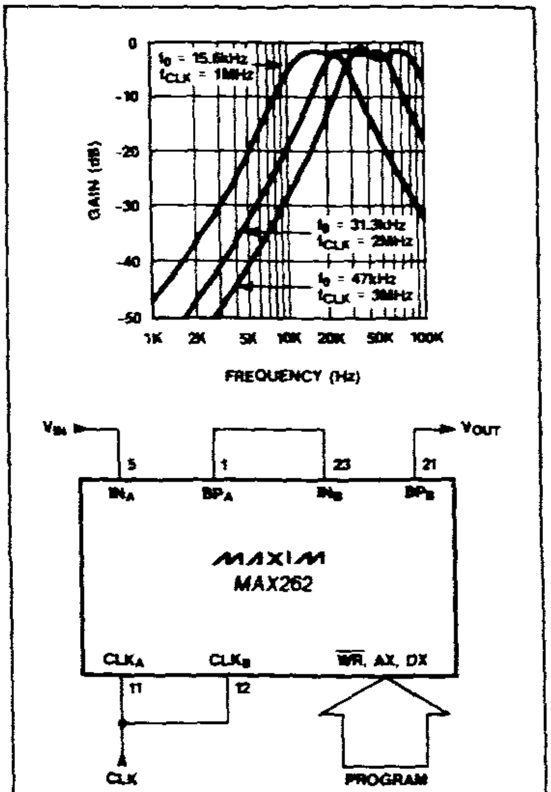
In this example (Figure 24) the desired parameters are:

Center frequency (f_0)	= 1 kHz
Pass bandwidth	= 1 kHz
Stop bandwidth	= 3 kHz
Max passband ripple	= 1 dB
Min stopband Attenuation	= 20 dB



$CLK_{A,B}$	$MODE_A$	$MODE_B$	f_{0A}	f_{0B}	Q_A	Q_B
120kHz	1	2	N=56	N=5	N=96	N=83

Figure 24. Wide Passband Chebyshev Bandpass Filter



$CLK_{A,B}$	MODE	f_{0A}	f_{0B}	Q_A	Q_B
1 to 3MHz	1	N=38	N=0	N=96	N=96

Figure 25. High Frequency Chebyshev Bandpass Filter

APPENDIX 9

Listing of Assembler Program

```
; Programme to Read Maxim 172 A/D Sample
; Intended to become Turbo Pascal INLINE Statement

.radix  16
Code    Segment
assume  cs:code,ds:code

org     100
begin:  jmp     start

time    dw      24                  ; About 2 seconds
tout    dw      0

start:  push    ax
        push    bx
        push    cx
        push    dx
        call    settim              ; Redirect timer interrupt
        mov     ax,cs:[time]
        mov     cs:[tout],ax
        mov     dx,262              ; Port C
        mov     al,0Fh              ; Set HBEN Low
        out     dx,al
htrb:   mov     ax,cs:[tout]        ; This checks if hardware
        cmp     ax,0                ; present
        jz      no_hw
        in      al,dx
        and     al,30
        cmp     al,0
        jnz     htrb                ; Read & busy Not Low

        cli
        mov     bx,0000             ; Pascal Variable (Count)
        mov     cx,[bx]             ; cx = no. of samples
        mov     bx,0000             ; Pascal Variable (Address)
loop1:  call    get_s               ; Get Sample - MAIN LOOP
        mov     [bx],ax             ; Store It
        inc     bx
        inc     bx
        loop    loop1               ; For CX samples
        sti
        mov     ax,0                ; ax = 0, indicates no error
ex_adr: push    ds                  ; Direct Timer Tick back to Original
        push    ax
        cli
        mov     ax,0
        mov     ds,ax
        mov     bx,70
        mov     ax,cs:[fjofs]       ; Offset
        mov     [bx],ax
        inc     bx
        inc     bx
        mov     ax,cs:[fjseg]       ; Segment
        mov     [bx],ax
        sti
        pop     ax
        pop     ds
        mov     bx,0000             ; Pascal Variable (Error)
        mov     [bx],ax             ; If Error = 1, No Hardware Present
        pop     dx                  ; If Error = 0, No Error
        pop     cx
```

```
        pop     bx
        pop     ax
        jmp     pas                 ; Return to pascal program

no_hw:  mov     ax,1                ; ax = 1 means an error
        jmp     ex_adr


get_s:  push    cx                  ; Get 1 sample into ax
        mov     dx,262              ; Port C
        mov     al,0Fh              ; Set HBEN Low
        out     dx,al
rb_nl:  in      al,dx
        and     al,30
        cmp     al,0
        jnz     rb_nl               ; Read & busy Not Low

bush:   in      al,dx
        and     al,30
        cmp     al,10
        jnz     bush                ; Busy High, Read Low

        dec     dx                  ; Port 261  -  Port B
        in      al,dx
        mov     cl,al               ; CL has Low Byte

        inc     dx

rhbh:   in      al,dx
        and     al,10
        cmp     al,10
        jnz     rhbh                ; Busy High

        mov     al,0Bh              ; Set HBEN High
        out     dx,al

rlbh:   in      al,dx
        and     al,30
        cmp     al,10
        jnz     rlbh                ; Busy High, Read Low

        dec     dx                  ; =261 again
        in      al,dx
        and     al,0fh              ; Clear High Nibble (mask it)
        mov     ah,al               ; High byte

        inc     dx                  ; =262 Again

rhbh2:  in      al,dx
        and     al,30
        cmp     al,30
        jnz     rhbh2               ; Busy High, Read High

        mov     al,0fh
        out     dx,al               ; Set HBEN LOW

        mov     al,cl               ; AX now contains Sample
        pop     cx

        ret
```

```
                                    ; Point Timer interrupt (1 int/55 ms) to
settim: push    ds                  ;   'Tick'
        cli                         ; In Case an Interrupt occurs before int
        mov     ax,0                ;  set up.
        mov     ds,ax
        mov     bx,70
        mov     ax,[bx]
        mov     cs:[fjofs],ax       ; Update Our Jump Offset
        lea     ax,tick
        mov     [bx],ax
        inc     bx
        inc     bx
        mov     ax,[bx]
        mov     cs:[fjseg],ax       ; Update Our Jump Segment
        push    cs
        pop     ax
        mov     [bx],ax             ; Update Interrupt Vector
        sti                         ; Re-Start Interrupts
        pop     ds
        ret

tick:   dec     cs:[tout]
        db      0EAh                ; JMP Far instruction
fjofs   dw      0
fjseg   dw      0

pas:    nop

code    ends
end     begin                       ; PROGRAM ENDS !!
```

APPENDIX 10

Listing of main Pascal programs

```
                                                   {$I-}
PROGRAM Acoustic;          {main controlling program body}
Uses
   Dos,Graph,Graphics,Measure,Filter,CctCont,
   Menus,TPCrt,TPMenu,TPWindow,TPString,TPEdit;

VAR
    SaveInt23 : Pointer;
    r : byte;
    IOSave : integer;

{*********************************************************}

PROCEDURE InitVar;        {initialise variables/arrays used in program}
BEGIN
  For r := 1 to 20 do
  begin
    TempArray[r] := 0.0;
    spl1[r] := 0.0;
    spl2[r] := 0.0;
    R_T[r]  := 0.0;
    ED_T[r] := 0.0;
    dev1[r] := 0.0;
    dev2[r] := 0.0;
    Pos1[r] := 0.0;
  end;
END;


Procedure Show_SPL;    {displays SPL from SLM or mic.}
var
   value : string;
begin
    SetSampleRate('spl');        {set sample rate of A/D}
    Calibrate;                   {BAA calibration routine}
    If not MakeWindow(TempWin,26,11,51,16,true,true,true,$0,$0E,$70,
       ' SPL Display ') then ErrorMem;             {popup window}
    If not DisplayWindow(TempWin) then ErrorMem;
    with StatusRec do                      {record of status of BAA}
     repeat
    Ad_Read(15);   {get one sample only every 1/2 second - delay(500) }
    ArrayCon('','1');                    {converts sample values}
    value := Real2Str1(ArrayAve);
    If volt >= 2.99 then begin           {maximum of the range setting}
       clrscr;
       FastWriteWindow(' OVERLOAD ! ',2,7,$8B);
       FastWriteWindow(' Please change RANGE ',3,3,$0B);
       end
    Else begin
     clrscr;
     FastWriteWindow(''+Pad(Real2Str4(volt),6)+'',1,10,$0B);}
     FastWriteWindow(' SPL level on CH '+Long2Str(i_c-9)+'',2,3,$0B);
     FastWriteWindow('  '+value+' dB  ',3,7,$70);
     FastWriteWindow(' Hit SPACEBAR to exit ',5,2,$70);
     Delay(1000);
    End;
    until KeyPressed;                 {display SPL till key pressed}
    DisposeWindow(EraseTopWindow);
end;

Procedure ShowRTplot;                   {displays RT decay plot}
```

```
begin
   Calibrate;
   If not MakeWindow(TempWin,18,11,63,17,true,true,true,$0,$0E,$70,
      ' Reverberation Time Plot ') then ErrorMem;
   If not DisplayWindow(TempWin) then ErrorMem;
   FastWriteWindow('  Getting data ...  ',6,13,$70);
   RT_OnePosition(1);                  {measures decay at selected freq}
   DisposeWindow(EraseTopWindow);
   EraseMenuOntoStack(main,Mstack);   {clear menu sytem from memory}
   InitGraphics;                       {initialise graphics}
   DecayPlot;                          {show decay plot}
   CloseGraph;                         {end graphics routine}
   Display_Main('Main Menu');          {show main menu again}
   DrawMenuFromStack(main,Mstack);
end;


PROCEDURE ShowRTtable;             {show table of reverb times}

Procedure GetRTdata;               {get reverb time data}
var
   c,LineNo,freq : integer;
begin
  If not MakeWindow(TempWin,19,11,59,15,true,true,true,$0,$0E,$70,
     ' R.T Table ') then ErrorMem;
  If not DisplayWindow(TempWin) then ErrorMem;
  ReadString('Enter file name [*.RTD]: ',13,21,12,$0B,$70,$70,Escaped,name);
  clrscr;
  Assign(TempFile,name);
  Reset(TempFile);
  IOSave := IOResult;
  If IOSave <> 0 then                  {error routine if file not found}
     ErrorWindow('File not found !')
  Else begin
     c := 1;
     LineNo := 0;
     while not SeekEof(TempFile) do begin      {read in data}
        Inc(LineNo);                           {start at line 1 in file}
        while not SeekEoln(TempFile) do
         case LineNo of
              1 : read(TempFile,des);       {des=sample/room description}
              2 : read(TempFile,desc);      {desc=file desription}
          3..22 : begin
                    read(TempFile,freq,ED_T[c],dev1[c],R_T[c],dev2[c]);
                    Inc(c);              {read in EDT,RT,deviation factors}
                  end;
         end;  {case LineNo}
     end;   {case SeekEof}
  end;      {IOResult}
  DisposeWindow(EraseTopWindow);
end;

BEGIN
  GetRTdata;                          {get reverb time data}
  If IOSave <> 0 then exit;           {if no data then exit routine}
    EraseMenuOntoStack(main,Mstack);
    InitGraphics;                     {initialise graphics}
    RT_Table;                         {show table of RT values}
    CloseGraph;                       {end graphics}
    Display_Main('Main Menu');        {show main menu again}
    DrawMenuFromStack(main,Mstack);
```

```
END;

PROCEDURE TL_Table;                  {show transmission loss data table}

Procedure GetTLdata;                 {get TL data}
var
   c,LineNo,freq : integer;
begin
  If not MakeWindow(TempWin,19,11,59,15,true,true,true,$0,$0E,$70,
     ' T.L Table ') then ErrorMem;
  If not DisplayWindow(TempWin) then ErrorMem;
  ReadString('Enter file name [*.TLD]: ',13,21,12,$0B,$70,$70,Escaped,name);
  clrscr;
  Assign(TempFile,name);
  Reset(TempFile);
  IOSave := IOResult;
  If IOSave <> 0 then
     ErrorWindow('File not found !')
  Else begin
    while not SeekEoln(TempFile) do
     read(TempFile,des);                       {des=file desription}
     c := 1;
     LineNo := 1;                              {start at line 1 in file}
     while not SeekEof(TempFile) do begin      {read in data}
        Inc(LineNo);
        while not SeekEoln(TempFile) do
         case LineNo of
           2  : read(TempFile,desc1);  {desc1=source}
           23 : read(TempFile,desc2);  {desc2=background}
           44 : read(TempFile,desc3);  {desc3=receiving room}

           3..22 : begin               {source room level}
                     read(TempFile,freq,spl1[c],dev1[c]);
                     Inc(c);             {read frequency,SPL, deviation factor}
                     if c = 21 then c := 1;
                   end;

           24..43,45..64 : begin     {background & rec room level}
                             read(TempFile,freq,spl2[c],dev2[c]);
                             Inc(c);
                           end;
         end;  {case line}
     end;   {case SeekEof}
    end;
  DisposeWindow(EraseTopWindow);
end;

BEGIN
  GetTLdata;                         {get TL data}
  If IOSave <> 0 then exit;          {if no data or file then exit}
    EraseMenuOntoStack(main,Mstack); {clear menu system from memory}
    InitGraphics;
    T_Loss1;                         {show TL data table}
    CloseGraph;
    Display_Main('Main Menu');
    DrawMenuFromStack(main,Mstack);
END;

PROCEDURE ShowAbsorbTable;           {show absorbtion data table}
VAR
  R_T1,R_T2 : RealArrType1;            {R_T1 = empty room RT's}
```

```
  Temp,Surf,SoundSpeed,Delta : real;   {R_T2 = room with sample RT's}

CONST
  CALvolume = 199.7;            { CAL large reverb room }

Procedure GetAbsorbInfo;        {get parameters from user}
begin
  FastWriteWindow('Enter DRY ROOM TEMPERATURE (deg) : ',2,3,$0B);
  ReadReal('',14,35,7,$0B,$70,2,0.0,100.0,Escaped,Temp);
  clrscr;
  FastWriteWindow('Enter SURFACE AREA of material sample ',2,2,$0B);
  ReadReal(' (metric) : ',14,28,7,$0B,$70,2,0.0,0.0,Escaped,Surf);
  SoundSpeed := 331.6 + (0.6*Temp);
  Delta      := (55.3 * CALvolume/SoundSpeed/Surf);
  for r := 1 to 20 do      {calculate absorption coefficients}
      AbsValue[r] := (Delta * (1/R_T2[r] - 1/R_T1[r]));
end;

Procedure GetRTdata;              {get RT data for absorption calc}
var
  c,LineNo,freq : integer;
begin
 If not MakeWindow(TempWin,19,11,59,16,true,true,true,$0,$0E,$70,
    ' Absorption Table ') then ErrorMem;
 If not DisplayWindow(TempWin) then ErrorMem;
 ReadString('Enter file name [*.ATD]: ',13,21,12,$0B,$70,$70,Escaped,name);
 clrscr;
 Assign(TempFile,name);
 Reset(TempFile);
 IOSave := IOResult;
 If IOSave <> 0 then
    ErrorWindow('File not found !')
 Else begin
    c := 1;
    LineNo := 0;
    while not SeekEof(TempFile) do begin      {read in data}
       Inc(LineNo);
       while not SeekEoln(TempFile) do
        case LineNo of
             1 : read(TempFile,des);       {des=sample/room description}
             2 : read(TempFile,desc);      {desc=file desription}
            23 : read(TempFile,desc);      {desc=file desription}
          3..22 : begin
                    read(TempFile,freq,ED_T[c],dev1[c],R_T1[c],dev2[c]);
                    Inc(c);           {get freq,EDT,RT,deviation factors}
                  end;                {for empty room}
         24..43 : begin
                    read(TempFile,freq,ED_T[c-20],dev1[c-20],R_T2[c-20],dev2[c-20]
                    Inc(c);           {get freq,EDT,RT,deviation factors}
                  end;                {for room with sample}
        end;  {case LineNo}
    end;   {case SeekEof}
 end;      {IOResult}
(*  DisposeWindow(EraseTopWindow);*)
end;

BEGIN
 GetRTdata;                         {get RT data for absorbtion table}
 If IOSave <> 0 then Exit;
 GetAbsorbInfo;                     {get parameters/absorb coeff for}
   EraseMenuOntoStack(main,Mstack); {displaying table}
```

```
    InitGraphics;
    AbsTable;                         {show table of absorption coeff}
(*    AbsGraph;*)
    WaitToGo;                         {display till key pressed}
    CloseGraph;
    Display_Main('Main Menu');
    DrawMenuFromStack(main,Mstack);
END;

Procedure DeleteFiles;             {routine to delete files}
begin
  If not MakeWindow(TempWin,19,11,59,15,true,true,true,$0,$0E,$70,
     ' Delete Files ') then ErrorMem;
  If not DisplayWindow(TempWin) then ErrorMem;
  ReadString('Enter file name [*.TXT]: ',13,21,12,$0B,$70,$70,Escaped,name);
  clrscr;
  Assign(TempFile,name);
  Erase(TempFile);                      {delete file}
  If IOResult <> 0 then
     ErrorWindow('File not found !')
  Else begin
     FastWriteWindow(' File deleted ! ',2,11,$0E);
     FastWriteWindow(' Hit ENTER to continue ',4,9,$F0);
     readln;
     DisposeWindow(EraseTopWindow);
  end;
end;

PROCEDURE PrinterStatus;              {routine to check status of printer}

Function PrinterReady(Printer : byte) : boolean;
var
    Regs : registers;
begin
    Regs.AH := $02;
    Regs.DX := Printer;        { 0=LPT1 1=LPT2 2=LPT3 }
    Intr($17,Regs);
    PrinterReady := (Regs.AH and $80 = $80) and   {Test if ready}
                    (Regs.AH and $10 = $10) and   {test if selected}
                    (Regs.AH and $08 = $00);      {test if I/O error}
end;

Function PrinterOutOfPaper(Printer : byte) : boolean;
var
    Regs : registers;
begin                          {check printer-out-of-paper signal}
    Regs.AH := $02;
    Regs.DX := Printer;        { 0=LPT1 1=LPT2 2=LPT3 }
    Intr($17,Regs);
    PrinterOutOfPaper := Regs.AH and $20 = $20;
end;

BEGIN
    If PrinterReady(0) then begin
      if PrinterOutOfPaper(0) then
        ErrorWindow('Printer on line but out of paper !');
      end
     else
        ErrorWindow('Printer not ready !');
END;
```

```
Procedure ConfigHelp;          {future help routines for user}
begin
end;

Procedure TestHelp;
begin
end;

Procedure UtilHelp;
begin
end;

Procedure InitStatFile;           {get previous status of BAA or initialise}
begin                                        {new BAA status file}
  Assign(StatusFile,'StatData.Dat');
  Reset(StatusFile);
  Read(StatusFile,StatusRec);
  IOSave := IOResult;
  UpDateCircuit;                  {configure circuitry to status data}
  If IOSave <> 0 then begin
     Rewrite(StatusFile);         {form new status data file}
     with StatusRec do begin
          noise_type := 'Pink';          n_t := 2;     {type of noise}
          out_filt   := 'Third Octave'; o_f := 4;      {noise O/P filtering}
          noise_burst := 'Auto-Gated';  n_b := 6;      {manual/auto noise gating
          out_chan   := 'CHAN A';        o_c := 8;     {noise O/P channel}
          in_chan    := 'CH 1 only';     i_c := 10;    {mic I/P channel}
          weight     := 'None';          w_t := 16;    {type of weighting}
          in_filt    := 'Active';        i_f := 20;    {input filtering}
          start_freq := '1000';      start_f := 1000; {filter start freq}
          stop_freq  := '200';       stop_f  := 200;  {filter stop freq}
          no_of_samp := 'One';         n_o_s := 23;    {no. of mic positions}
          rt_level   := '20 dB';       rt_l  := 26;    {RT decay level}
          range_pos  := '100';       range_p := 100;  {range position of SLM}
          CFactor    := 0.0;         {SPL correction factor of BAA}
     end;
     UpDateCircuit;              {configure circuitry to the above status data}
     Write(StatusFile,StatusRec);    {write new status data to status file}
     Reset(StatusFile);
  end;
end;

Procedure StartProg;          {start of main program}
begin
  ResetAll; InitVar; HiddenCursor;    {reset parameters/variables and}
  InitStatFile;                       {hide cursor and configure circuitry}
  clrscr; MapColors := True;
   InitMenu(main);                    {initialise main menu}
   Display_Main('Main Menu');         {displays main window}
   Repeat
    key  :=  MenuChoice(main,ch);     {read character from keyboard}
    If ord(ch) = 187 then             {future help utilities}
       case integer(key) of
        100 : ConfigHelp;
        101 : TestHelp;
        102 : UtilHelp;
       end;
    If ord(ch) = 13 then              {enter-key choice}
        case integer(key) of
           30   : Show_Status;        {show status of BAA}
           31   : Edit_Status;        {change status of BAA}
```

```
            32    : Show_SPL;              {show SPL from SLM/mic.}
            34    : DeleteFiles;           {delete files from drives}
            35    : Reverb;                {routine to calculate RT's}
            36    : ShowRTtable;           {displays RT data in table format}
            37    : ShowRTplot;            {displays plot of RT decay curve}
            38    : TL_Table;              {displays TL data in table format}
            60    : ShowAbsorbTable;       {displays absorption data table}
            61    : Absorb;                {routine to calculate absorption coeff}
        50..53    : TransmLoss(key);       {routines to calculate various TL tests}
         end;
   Until ord(ch) = 196;                    {user chooses to quit "F10" }
end;

Procedure CBreak(Flags,CS,IP,AX,BX,CX,DX,SI,DI,DS,ES,BP : word);
Interrupt;                    {routine to handle Ctrl-C break by user}
begin
   Repeat                     {clear windows from screen}
     V := EraseTopWindow;
     DisposeWindow(V);
   Until V = nil;
   clrscr;
   StartProg;                 {start main program again}
end;

{************************** MAIN PROGRAM *********************************}

BEGIN
  GetIntVec($23,SaveInt1B);         { take control of Ctrl-C interrupt }
  SetIntVec($23,@CBreak);
  StartProg;
   Repeat                           {clear windows from screen}
     V := EraseTopWindow;
     DisposeWindow(V);
   Until V = nil;
   clrscr;
   ResetAll;                        {reset all parameters/variables}
   Write(StatusFile,StatusRec);
   Reset(StatusFile);
   Close(StatusFile);
   SetIntVec($23,SaveInt23);        { restore Ctrl-C interrupt }
END.                                { program body }
```

```
Unit Measure;             { all RT, SPL.. data measurements }

Interface
uses TPCrt,TPString,TPMenu,TPWindow,TPEdit,Menus,
     CctCont,Filter;

type
   RealArrType1 = array[1..20] of real;
   RealArrType2 = array[1..40] of real;
   RealArrTyp   = array[1..3] of real;

var
   TempWin  : WindowPtr;
   key      : MenuKey;
   Escaped  : boolean;
   TempFile : Text;
   dev1,spl1,TempArray,Pos1,Pos2,Pos3,ED_T,R_T : RealArrType1;
   spl2,dev2 : RealArrType2;
   DecayTime,EDT : RealArrTyp;
   Ave           : real;
   InFileName    : string;
   LastPos,MicPos,flag : byte;

Const
     name : string = ('lab1');
     desc : string = ('');

Procedure TransmLoss(key : byte);
Procedure ErrorWindow(message : string);
Procedure Calibrate;
Procedure Reverb;
Procedure Absorb;
Procedure RT_OnePosition(e : byte);

Implementation
var
   a : byte;

Procedure ErrorWindow(message : string);  {displays error window message}
begin                                     {when user makes mistake}
  If not MakeWindow(TempWin,19,11,59,16,true,true,true,$0,$0E,$70,
     ' Error Status ') then ErrorMem;
  If not DisplayWindow(TempWin) then ErrorMem;
  writeln(#7);                                          {error bell}
  FastWriteWindow(''+Center(message,39)+'',2,1,$0E);  {write message}
  FastWriteWindow(' Hit ENTER to continue ',5,9,$70); {wait for user}
  readln;
  DisposeWindow(EraseTopWindow);                        {erase error window}
  DisposeWindow(EraseTopWindow);
end;

Procedure Calibrate;     {for calibrating BAA before measurements}
var                      {adds/subtracts calibration factor to data}
   Key : word;
   Ch  : Char absolute Key;
   cal_value : real;           {calib. reference dB value : 94 / 124 dB}
begin
  with StatusRec do begin
   cal_value := 0.0;           {initialise cal. factor to zero}
   If not MakeWindow(TempWin,24,11,58,15,true,true,true,$0,$0E,$70,
```

```
      ' Calibrate System ') then ErrorMem;
   If not DisplayWindow(TempWin) then ErrorMem;  {show calibrate menu}
   repeat
    FastWriteWindow(' Do you want to calibrate [N] ? ',2,2,$0B);
    Key := ReadKeyWord;
   until upcase(Ch) in ['Y','N',^M];              {answer yes/no}
   case upcase(Ch) of
    'Y' : begin    {enter ref. calibration value ie. 94 dB}
          ReadReal(' Enter calibration value : ',13,25,5,$0B,
             $70,2,0,130,Escaped,cal_value);
          FastWriteWindow(' Please wait about 30 secs to  ',2,2,$70);
          FastWriteWindow('    calibrate the circuit.     ',3,2,$70);
          Ad_Read(625);                      {get ave SPL in 5 seconds }
          ArrayCon('','0');                  {convert data to dB format}
          CFactor := cal_value - ArrayAve;   {calculate calib. factor}
          DisposeWindow(EraseTopWindow);
         end;
    'N' : begin DisposeWindow(EraseTopWindow); Exit; end;     {answer = no}
    ^M  : begin DisposeWindow(EraseTopWindow); ch := ^A; Exit; end;  {exit}
  end;
 end;
end;

Function FreqToD(start_f : integer) : byte;
begin
  case start_f of              {converts freq. selected by user to a number}
    100 : FreqToD := 1;        {corresponding to the position of that freq}
    125 : FreqToD := 2;        {in the array of third octave freq's. Used in}
    160 : FreqToD := 3;        {TL_OnePosition and other measurements}
    200 : FreqToD := 4;
    250 : FreqToD := 5;
    315 : FreqToD := 6;
    400 : FreqToD := 7;
    500 : FreqToD := 8;
    630 : FreqToD := 9;
    800 : FreqToD := 10;
    1000 : FreqToD := 11;
    1250 : FreqToD := 12;
    1600 : FreqToD := 13;
    2000 : FreqToD := 14;
    2500 : FreqToD := 15;
    3150 : FreqToD := 16;
    4000 : FreqToD := 17;
    5000 : FreqToD := 18;
    6300 : FreqToD := 19;
    8000 : FreqToD := 20;
  end;
end;

Procedure WriteToDisk;    {writes measurement data to disk/memory}
Var
   c : byte;
begin
   for c := 1 to 20 do    {covers full range of freq's}
   begin
     write(TempFile,ThirdOctave[c]);          {write freq's}
     write(TempFile,'  ');
     write(TempFile,TempArray[c]:4:1);        {write data}
     write(TempFile,'  ');
     writeln(TempFile,dev1[c]:0:1);           {write deviation factors}
   end
```

```
end;

Procedure WriteToD_RT;     {write RT data to disk}
var
   c : byte;
begin
   writeln(TempFile,'Reverb R.T averages');     {label file of data}
   for c := 1 to 20 do
   begin
     write(TempFile,ThirdOctave[c]);     {write freq's}
     write(TempFile,'  ');
     write(TempFile,R_T[c]:0:1);         {write RT times}
     write(TempFile,'  ');
     writeln(TempFile,dev2[c]:0:1);
   end;
end;

Procedure CheckDev(a,flag : byte);  {check deviation between data as}
begin                                {required by standards}
  case ThirdOctave[a] of             {check for repeatability}
    100,125,160,200 : case flag of                       {± 5 dB allowed}
                        0 : if dev1[a] > 5 then flag := 1; {1st error,set flag
                        1 : if dev1[a] > 5 then begin      {2nd error, write}
                              TempArray[a] := 1;           {error code}
                              flag := 0;
                            end;
                      end;
                250 : case flag of                       {± 3 dB allowed}
                        0 : if dev1[a] > 3 then flag := 1; {1st error,set flag
                        1 : if dev1[a] > 3 then begin      {2nd error, write}
                              TempArray[a] := 1;           {error code}
                              flag := 0;
                            end;
                      end;
        315,400,500 : case flag of                        {± 2 dB allowed}
                        0 : if dev1[a] > 2 then flag := 1;  {1st error}
                        1 : if dev1[a] > 2 then begin       {2nd error}
                              TempArray[a] := 1;
                              flag := 0;
                            end;
                      end;
    630,800,1000,1250 : case flag of                        {± 1 dB allowed}
                          0 : if dev1[a] > 1 then flag := 1;  {1st error}
                          1 : if dev1[a] > 1 then begin       {2nd error}
                                TempArray[a] := 1;
                                flag := 0;
                              end;
                        end;
       1600..8000 : case flag of                        {± 2 dB allowed}
                      0 : if dev1[a] > 2 then flag := 1;  {1st error}
                      1 : if dev1[a] > 2 then begin       {2nd error}
                            TempArray[a] := 1;
                            flag := 0;
                          end;
                    end;
    end;
  end;

Procedure TL_OnePosition(mmtype : string);        {measures SPL at 1 position}
Var                                             { 1 or 3 samples/position}
   a,b,flag : byte;
```

```
begin
 with StatusRec do begin
  a := FreqToD(start_f);           {convert start freq to equivalent}
  flag := 0;                       {no. in 1/3 octave array,set error flag}
  repeat                           { freq step loop }
     clrscr;
     FastWriteWindow(' Measuring at : ',3,10,$0B);
     FastWriteWindow(' '+Long2Str(ThirdOctave[a])+' Hz ',3,27,$70);
     SetSampleRate('spl');         {set sample rate of A/D}
     If o_f <> 3 then
        SetFilters(o_f,ThirdOctave[a]);        {if linear input not selected}
     If (mmtype = 'back') or (MicPos in [4..6]) then GenOff
       else GenOn;   {switch noise gen off if doing background measurements}
     Delay(10000);   {allow noise energy to build up}
     Ad_Read(625);   {read data in from A/D}
        If Ad_Error = 1 then begin          {if A/D type error occurs}
           ErrorWindow('ERROR - Analyser not ON !');
           DisposeWindow(EraseTopWindow);
           TransmLoss(key);
        end;                             { Ad_Error }
     GenOff;                             {switch noise gen off}
     Delay(10000);
     ArrayCon('tl','1');                 {convert A/D data to dB format}
     TempArray[a] := ArrayAve;           {store results in temporary array}
     CheckDev(a,flag);                   {check deviation between results}
     Inc(a);
  until stop_f <= ThirdOctave[a-1];  {measure till reach stop freq}
 end;  {StatusRec}
end;   {OnePos}

Procedure OneMic1Sample(key : byte);    {routine to control measurements}
begin                                   {using 1 mic doing 1 sample}
  with StatusRec do begin               {  for transmission loss }
    clrscr;
    SetInChan(10);                      {use input chan 1 only}
    FastWriteWindow(' Put microphone in SOURCE room ',3,7,$0B);
    FastWriteWindow(' Hit ENTER to run test ',6,11,$F0);
    readln;
    FastWriteWindow('  Working...SOURCE ROOM LEVEL   ',6,7,$70);
    TL_OnePosition('source');          {perform source measurement}
    writeln(TempFile,'Source Room');   {label source room data}
    WriteToDisk;                       {write SPL data at each freq}
    clrscr;
    FastWriteWindow(' Put microphone in RECEIVING room ',2,5,$0B);
    FastWriteWindow('  Check that SPL within RANGE of  ',3,5,$0B);
    FastWriteWindow('  SLM and correct if necessary    ',4,5,$0B);
    FastWriteWindow('     Hit ENTER to continue     ',6,7,$F0);
    readln;
    clrscr;
    FastWriteWindow('       Enter Y or N ...         ',6,7,$F0);
    If YesOrNo('Change RANGE in software ? ',14,27,$0B,'N') then begin
        key := 29;
        UpDateStatus(key);             {edit range position in status}
    end;
    clrscr;
    FastWriteWindow('  Working...BACKGROUND LEVEL    ',6,7,$70);
    TL_OnePosition('back');            {perform background measurement}
    writeln(TempFile,'Background Level');  {label background data}
    WriteToDisk; delay(2000);              {write SPL data at each freq}
    clrscr;
    FastWriteWindow(' Working...RECEIVING ROOM LEVEL ',6,7,$70);
```

```
    TL_OnePosition('rec');               {perform receiving room measurement}
    writeln(TempFile,'Receiving Room');   {label receiving room data}
    WriteToDisk;                          {write SPL data at each freq}
    If key in [51..53] then begin         {do RT measurement as well if}
      If o_c = 8 then                     {complying to standard}
        FastWriteWindow(' Place SPEAKER in RECEIVING room ',3,5,$0B)
      Else begin
        FastWriteWindow('  Check that SPL within RANGE of  ',3,5,$0B);
        FastWriteWindow('  SLM and correct if necessary    ',4,5,$0B);
        FastWriteWindow('    Hit ENTER to start noise      ',6,7,$F0);
        readln;
        GenOn;
        clrscr;
        FastWriteWindow('       Enter Y or N ...          ',6,7,$F0);
        If YesOrNo('Change RANGE in software ? ',14,27,$0B,'N') then begin
          key := 29;
          UpDateStatus(key);
        end;
        GenOff;
        clrscr;
        FastWriteWindow('  Working...REC ROOM R.T TIMES  ',6,7,$70);
        RT_OnePosition(0);    {measure RT times at one position}
        WriteToD_RT;          {write RT data to disk}
      end;
    end;
  end;
end;

(*
Procedure TwoMic1Sample(o_c : byte);     {this procedure not used !!}
begin                                    {for 2 mic's only}
     clrscr;
     SetInChan(10);         {channel 1}
     FastWriteWindow(' Measuring in SOURCE room ',3,10,$0B);
     FastWriteWindow(' Hit ENTER to run test ',6,12,$F0);
     readln;
     FastWriteWindow(' Working...SOURCE ROOM LEVEL ',6,8,$70);
     TL_OnePosition('source');
     writeln(TempFile,'Source Room');
     WriteToDisk;                        {SPL at each freq}
     clrscr;
     SetInChan(11);                      {channel 2}
     If o_c = 8 then begin
        FastWriteWindow(' Place speaker in RECEIVING ROOM ',3,6,$0B);
        FastWriteWindow('     Hit ENTER to continue     ',6,7,$F0);
        readln; end;
     clrscr;
     FastWriteWindow(' Measuring in RECEIVING ROOM ',3,7,$0B);
     Delay(2000);
     FastWriteWindow('  Working...BACKGROUND LEVEL    ',6,7,$70);
     TL_OnePosition('back');
     writeln(TempFile,'Background Level');
     WriteToDisk; delay(2000);           {SPL at each freq}
     clrscr;
     FastWriteWindow(' Working...RECEIVING ROOM LEVEL ',6,7,$70);
     TL_OnePosition('rec');
     writeln(TempFile,'Receiving Room');
     WriteToDisk;                        {SPL at each freq}
end;
*)
```

```
Procedure DataAve;  {get ave of 3 positions,must conform to ISO 140,part II}
var
   c : integer;
   X1,X2 : real;
begin
   c := 1;
   repeat                              {calculates ave SPL of 3 mic positions}
     X1 := (sqr(Pos1[c] + Pos2[c] + Pos3[c]))/3;
     X2 := sqr(Pos1[c]) + sqr(Pos2[c]) + sqr(Pos3[c]);
     dev1[c] := sqrt((X2 - X1)/2);
     TempArray[c] := (sqrt(X1*3))/3;
     case c of
       1..4 : if dev1[c] > 5 then TempArray[c] := 1;  {check within limits}
          5 : if dev1[c] > 3 then TempArray[c] := 1;  {store error if not}
       6..8 : if dev1[c] > 2 then TempArray[c] := 1;
      9..12 : if dev1[c] > 1 then TempArray[c] := 1;
     13..20 : if dev1[c] > 2 then TempArray[c] := 1;
     end;  {case c}
     Inc(c);
   until c = 21;
end;

Procedure ThreePositions(key : byte);    {same measurements as above but now}
var                                      {doing 3 positions / room using}
   a : byte;                             {1, 3 or 6 mic's}
begin
 clrscr;
 If key in [51..53] then LastPos := 12   {check if RT measurement required}
 else LastPos := 9;
 with StatusRec do begin  {MicPos refers to mic position in rooms}
  repeat                  {1,2,3=source room  4,5,6=rec room/background noise}
   case i_c of            {7,8,9=rec room level  10,11,12=rec room RT's}

{**************************** 1 mic ***********************************}

10 : case MicPos of         {MicPos refers to mic position in room}
        4 : begin
              FastWriteWindow(' Put microphone in RECEIVING room ',2,5,$0B);
              FastWriteWindow('  Check that SPL within RANGE of  ',3,5,$0B);
              FastWriteWindow('  SLM and correct if necessary    ',4,5,$0B);
              FastWriteWindow('     Hit ENTER to continue      ',6,7,$F0);
              readln;
              clrscr;
              FastWriteWindow('       Enter Y or N ...         ',6,7,$F0);
              ReadCharacter('Change RANGE in software ? ',14,27,$0B,['Y','N'],
              If upcase(ch) = 'Y' then begin
                key := 29;
                UpDateStatus(key);
              end;
              clrscr;
            end;
    1..12 : begin
              FastWriteWindow(' Put microphone in position  ',3,7,$0B);
              FastWriteWindow(' '+Long2Str(MicPos)+' ',3,35,$70);
              If MicPos = 1 then
                 FastWriteWindow(' Hit ENTER to run test ',6,12,$F0)
              Else
                 FastWriteWindow('     Hit ENTER to continue      ',6,7,$F0);
              readln;
            end;
       10 : begin
```

```
            If o_c = 8 then
               FastWriteWindow(' Place SPEAKER in RECEIVING room ',2,5,$0B)
            Else begin
               FastWriteWindow('  Check that SPL within RANGE of  ',3,5,$0B)
               FastWriteWindow('  SLM and correct if necessary    ',4,5,$0B)
               FastWriteWindow('    Hit ENTER to start noise      ',6,7,$F0)
               readln;
               GenOn;
               clrscr;
               FastWriteWindow('       Enter Y or N ...         ',6,7,$F0);
               ReadCharacter('Change RANGE in software ? ',14,27,$0B,['Y','N
               If upcase(ch) = 'Y' then begin
                  key := 29;
                  UpDateStatus(key);
               end;
               GenOff;
               clrscr;
               FastWriteWindow('  Working...REC ROOM R.T TIMES  ',6,7,$70);
               RT_OnePosition(0);
               WriteToD_RT;
            end;
                     end;
end;    {case MicPos}

{*********************** 2 mics = routine not used !! *****************}

(* 11 : case MicPos of
       4 : begin
             FastWriteWindow(' Put microphones in RECEIVING room ',3,5,$0B);
             FastWriteWindow('     Hit ENTER to continue     ',6,7,$F0);
             readln;
           end;
     1,4 : begin
             FastWriteWindow(' Put CH 1 microphone in position  ',3,4,$0B);
             FastWriteWindow(' '+Long2Str(MicPos)+' ',3,37,$70);
             If MicPos = 1 then
                FastWriteWindow(' Hit ENTER to run test ',6,12,$F0)
             Else
                FastWriteWindow('     Hit ENTER to continue     ',6,7,$F0);
             readln;
           end;
   3,6,8 : begin
             FastWriteWindow(' Put CH 2 microphone in position  ',3,4,$0B);
             FastWriteWindow(' '+Long2Str(MicPos)+' ',3,37,$70);
             FastWriteWindow('     Hit ENTER to continue     ',6,7,$F0);
             readln;
           end;
       9 : begin
             FastWriteWindow(' Put CH 2 microphone in position  ',3,4,$0B);
             FastWriteWindow(' '+Long2Str(MicPos)+' ',3,37,$70);
             FastWriteWindow('     Hit ENTER to continue     ',6,7,$F0);
             readln;
           end;
       end;   {case MicPos}
*)
{****************************** 3 mics *********************************}

12 : case MicPos of
       4 : begin
             FastWriteWindow(' Put microphones in RECEIVING room ',2,5,$0B);
             FastWriteWindow('  Check that SPL within RANGE of  ',3,5,$0B);
```

```
            FastWriteWindow('  SLM and correct if necessary    ',4,5,$0B);
            FastWriteWindow('     Hit ENTER to continue      ',6,7,$F0);
            readln;
            clrscr;
            FastWriteWindow('        Enter Y or N ...        ',6,7,$F0);
            ReadCharacter('Change RANGE in software ? ',14,27,$0B,['Y','N'],
            If upcase(ch) = 'Y' then begin
               key := 29;
               UpDateStatus(key);
             end;
             clrscr;
          end;
     1..3 : begin
            FastWriteWindow(' Measuring in SOURCE room ',3,10,$0B);
          end;
       10 : begin
            If o_c = 8 then
               FastWriteWindow(' Place SPEAKER in RECEIVING room ',2,5,$0B)
            Else begin
               FastWriteWindow('  Check that SPL within RANGE of  ',3,5,$0B)
               FastWriteWindow('  SLM and correct if necessary    ',4,5,$0B)
               FastWriteWindow('    Hit ENTER to start noise      ',6,7,$F0)
               readln;
               GenOn;
               clrscr;
               FastWriteWindow('        Enter Y or N ...         ',6,7,$F0);
               ReadCharacter('Change RANGE in software ? ',14,27,$0B,['Y','N
               If upcase(ch) = 'Y' then begin
                   key := 29;
                   UpDateStatus(key);
               end;
               GenOff;
               clrscr;
               FastWriteWindow('  Working...REC ROOM R.T TIMES  ',6,7,$70);
               RT_OnePosition(0);
               WriteToD_RT;
            end;
          end;
    4..12 : begin
            FastWriteWindow(' Measuring in RECEIVING ROOM ',3,7,$0B);
          end;
end;    {case MicPos}

{***************************** 6 mics **********************************}

15 : case MicPos of
     1..3 : begin
            FastWriteWindow(' Measuring in SOURCE room ',3,10,$0B);
          end;
        4 : begin
            FastWriteWindow('  Check that SPL within RANGE of  ',3,5,$0B);
            FastWriteWindow('  SLM and correct if necessary    ',4,5,$0B);
            FastWriteWindow('     Hit ENTER to continue      ',6,7,$F0);
            readln;
            clrscr;
            FastWriteWindow('        Enter Y or N ...        ',6,7,$F0);
            ReadCharacter('Change RANGE in software ? ',14,27,$0B,['Y','N'],
            If upcase(ch) = 'Y' then begin
               key := 29;
               UpDateStatus(key);
            end;
```

```
             clrscr;
           end;
      10 : begin
             If o_c = 8 then
                FastWriteWindow(' Place SPEAKER in RECEIVING room ',2,5,$0B)
             Else begin
                FastWriteWindow('  Check that SPL within RANGE of  ',3,5,$0B)
                FastWriteWindow('  SLM and correct if necessary    ',4,5,$0B)
                FastWriteWindow('    Hit ENTER to start noise      ',6,7,$F0)
                readln;
                GenOn;
                clrscr;
                FastWriteWindow('       Enter Y or N ...         ',6,7,$F0);
                ReadCharacter('Change RANGE in software ? ',14,27,$0B,['Y','N
                If upcase(ch) = 'Y' then begin
                   key := 29;
                   UpDateStatus(key);
                end;
                GenOff;
                clrscr;
                FastWriteWindow('  Working...REC ROOM R.T TIMES  ',6,7,$70);
                RT_OnePosition(0);
                WriteToD_RT;
             end;
           end;
   4..12 : begin
             FastWriteWindow(' Measuring in RECEIVING ROOM ',3,7,$0B);
           end;
    end;    {case MicPos}
end;        {case i_c}

{*************************** label screen *****************************}

   clrscr;
   case MicPos of
     1..3 : FastWriteWindow('  Working...SOURCE ROOM LEVEL    ',6,7,$70);
     4..6 : FastWriteWindow('  Working...BACKGROUND LEVEL     ',6,7,$70);
     7..9 : FastWriteWindow(' Working...RECEIVING ROOM LEVEL ',6,7,$70);
   10..12 : FastWriteWindow('  Working...REC ROOM R.T TIMES  ',6,7,$70);
   end;

{************************** set input channels *************************}

   case i_c of
     10 : SetInChan(10);                    { 1 mic }
(*   11 : case MicPos of                    { 2 mics = not used}
           1,4,7        : SetInChan(10);
           2,3,5,6,8,9 : SetInChan(11);
         end;
*)      12 : case MicPos of                    { 3 mics }
           1,4,7,10 : SetInChan(10);
           2,5,8,11 : SetInChan(11);
           3,6,9,12 : SetInChan(12);
         end;
     15 : case MicPos of                    { 6 mics }
           1 : SetInChan(10);
           2 : SetInChan(11);
           3 : SetInChan(12);
       4,7,10 : SetInchan(13);
       5,8,11 : SetInChan(14);
       6,9,12 : SetInChan(15);
```

```
        end;
   end;  {case i_c}

{*************************** do measurements ***************************}

   TL_OnePosition('');           {get SPL data for 1 position}
   case MicPos of
     1 : for a := 1 to 20 do
             Pos1[a] := TempArray[a];   {store data for mic pos 1}
     2 : for a := 1 to 20 do
             Pos2[a] := TempArray[a];   {store data for mic pos 2}
     3 : begin
           for a := 1 to 20 do             {store data for mic pos 3}
               Pos3[a] := TempArray[a];
           writeln(TempFile,'Source room level ave');
           DataAve;              {get data ave of 3 positions}
           WriteToDisk;          {SPL,st. deviation at each freq}
         end;

     4 : for a := 1 to 20 do
             Pos1[a] := TempArray[a];    {store data for mic pos 4}
     5 : for a := 1 to 20 do
             Pos2[a] := TempArray[a];
     6 : begin
           for a := 1 to 20 do
               Pos3[a] := TempArray[a];
           writeln(TempFile,'Rec room Background level ave');
           DataAve;
           WriteToDisk;          {SPL,st. deviation at each freq}
         end;

     7 : for a := 1 to 20 do
             Pos1[a] := TempArray[a];
     8 : for a := 1 to 20 do
             Pos2[a] := TempArray[a];
     9 : begin
           for a := 1 to 20 do
               Pos3[a] := TempArray[a];
           writeln(TempFile,'Rec room level ave');
           DataAve;
           WriteToDisk;          {SPL,st. deviation at each freq}
         end;

   end;   {case MicPos}
  Inc(MicPos);
  until MicPos > LastPos;     {measure till at last mic position}
 end;   {StatusRec}
end;

Procedure CheckLevels;    {check & update rec room level with background level}
var                       {must conform to ISO 140,part IV}
   c,line,diff,flag       : byte;
   freq                   : integer;
   des,desc1,desc2,desc3 : string;
   dev2,spl2 : array[1..40] of real;
   spl1,dev1 : array[1..20] of real;
egin
    Reset(TempFile);                   {read from disk}
    while not SeekEoln(TempFile) do
          read(TempFile,des);          {read file desription from disk}
    c := 1;
```

```
line := 1;                              {start at line 1}
flag := 0;                              {reset error flag}
while not SeekEof(TempFile) do begin        {read in data}
   Inc(line);
   while not SeekEoln(TempFile) do
    case line of
      2  : read(TempFile,desc1);  {source name}
      23 : read(TempFile,desc2);  {background name}
      44 : read(TempFile,desc3);  {rec room name}

      3..22 : begin               {source room level/data}
                read(TempFile,freq,spl1[c],dev1[c]);
                Inc(c);
                if c = 21 then c := 1;
              end;

      24..43,45..64 : begin     {background & rec room level/data}
                        read(TempFile,freq,spl2[c],dev2[c]);
                        Inc(c);
                        if c = 41 then begin
                               c := 1;
                               repeat
                                 diff := trunc(spl2[c+20] - spl2[c]);
                                 case diff of
  {check difference in levels}     0..3 : begin
  {add corrections if necessary}              spl2[c+20] := 2;
  {as per standard}                           flag := 1;
                                            end;
                                   3 : begin
                                          spl2[c+20] := spl2[c+20] + 3;
                                          flag := 1;
                                       end;
                                   4..5 : begin
                                             spl2[c+20] := spl2[c+20] + 2;
                                             flag := 1;
                                          end;
                                   6..9 : begin
                                             spl2[c+20] := spl2[c+20] + 1;
                                             flag := 1;
                                          end;
                                 end;
                                Inc(c);
                               until c = 21;
                        end; {if c=41}
                      end;
    end;  {case line}
end;   {case SeekEof}
if flag = 1 then begin        {if corrections required then rewrite}
   Rewrite(TempFile);         {data to disk}
   writeln(TempFile,des);
   writeln(TempFile,desc1);
   for c := 1 to 20 do begin
     write(TempFile,ThirdOctave[c]);
     write(TempFile,'  ');
     write(TempFile,spl1[c]:4:1);
     write(TempFile,'  ');
     writeln(TempFile,dev1[c]:0:1);
   end;
   writeln(TempFile,desc2);
   for c := 1 to 40 do begin
       if c = 21 then writeln(TempFile,desc3);
```

```
            if c > 20 then write(TempFile,ThirdOctave[c-20])
            else write(TempFile,ThirdOctave[c]);
                 write(TempFile,'  ');
                 write(TempFile,spl2[c]:4:1);
                 write(TempFile,'  ');
                 writeln(TempFile,dev2[c]:0:1)
        end;
     end;  {flag}
end;

Procedure TransmLoss(key : byte);   {routine to control transmission loss}
begin                               {data measurements}
  Calibrate;                        {calibrate BAA}
  If not MakeWindow(TempWin,18,11,63,17,true,true,true,$0,$0E,$70,
     ' Transmission Loss Test ') then ErrorMem;
  If not DisplayWindow(TempWin) then ErrorMem;
  FastWriteWindow('Enter filename for TRANSMISSION LOSS test,',2,2,$0B);
  ReadString('not more than 8 chars: ',15,20,8,$0B,$70,$70,Escaped,name);
  clrscr;
  FastWriteWindow('Enter description of rooms:',2,10,$0B);
  ReadString('',15,22,40,$0B,$70,$70,Escaped,desc);
  clrscr;
  InFileName := ForceExtension(name,'TLD');   {TLD = Transm Loss Data}
  Assign(TempFile,InFileName);                {make file called ...}
  Rewrite(TempFile);
  writeln(TempFile,desc);
  with StatusRec do begin
   MicPos := 1;                     {set MicPos start value}
   case n_o_s of
     23 : case i_c of                     { 1 position/room}
          10 : OneMic1Sample(key);        {do measurement}
(*        11 : TwoMic1Sample(o_c,key);  *)
         end;  {case i_c}
     24 : ThreePositions(key);         { 3 positions/room}
   end;  {case n_o_s}
   CheckLevels;                        {check levels within limits}
   Close(TempFile);                    {store data / close file}
  end;    {case StatusRec}
  clrscr;
  FastWriteWindow(' Please move to ',2,3,$0B);
  FastWriteWindow(' Options ',2,20,$70);
  FastWriteWindow(' for results ',2,30,$0B);
  FastWriteWindow('  and other Standard calculations. ',3,5,$0B);
  FastWriteWindow('    Hit SPACEBAR to continue   ',6,8,$F0);
  repeat until keypressed;
  DisposeWindow(EraseTopWindow);
end;

Function CheckForDecay(rt_l : byte; var StopPt : integer): boolean;
var
  StartPt : real;                 {check for RT at each freq}
  a : integer;
egin
  StartPt := 0;
  for a := 1 to 10 do                  {take average of 10 samples to find}
    StartPt := StartPt + SPL_Array[a];  {start point of decay}
  StartPt := StartPt/10;
  case rt_l of
    26 : begin                         { 20dB determination level}
          for a := 1 to 2500 do begin
            if (StartPt - SPL_Array[a]) > 20 then begin
```

```
               CheckForDecay := true;            {if max SPL - min SPL}
               StopPt := a;                      { > 20 dB then decay exists}
               exit;
            end
            else
               CheckForDecay := false;           {no decay}
         end;
       end;
     27 : begin                           { 30dB determination level}
          for a := 1 to 2500 do begin
            if (StartPt - SPL_Array[a]) > 30 then begin
               CheckForDecay := true;            {decay exists}
               StopPt := a;
               exit;
            end
            else
               CheckForDecay := false;           {no decay}
          end;
        end;
     28 : begin                           { 40dB determination level}
          for a := 1 to 2500 do begin
            if (StartPt - SPL_Array[a]) > 40 then begin
               CheckForDecay := true;            {decay exists}
               StopPt := a;
               exit;
            end
            else
               CheckForDecay := false;           {no decay}
          end;
        end;
   end;   {case rt_1}
end;

Procedure Find_RT(rt_1,b : byte; StopPt : integer);    {find RT's}
Var
   r1,r2,r3,r4,r5,SumX,SumY,TimePerPt,StartPt   : real;
   TpP,AveX,AveY,Gradient,Yhat,YEq,error  : real;
   a,ref,c,temp,new                        : integer;
Begin
   r1:=0; r2:=0; r3:=0; r4:=0; r5:=0; AveX := 0;
   TpP := 0.002;
   TimePerPt := TpP;   {1/sampling rate -> 1/1000}
   StartPt := 0;

   for a := 1 to 10 do               {take average of 10 samples to}
     AveX := AveX + SPL_Array[a];    {find decay start point}
   AveX := AveX/10;
   a := 0;
   repeat                            {find start point, -5dB down}
     Inc(a);
     if SPL_Array[a] <= (AveX - 5) then
        StartPt := SPL_Array[a];
   until SPL_Array[a] <= (AveX - 5);
   for new := 1 to (2500-a) do                {make new data array from}
      SPL_Array[new] := SPL_Array[new+a-1];   {start point onwards}
   TimePerPt := 0; a := 0;
   repeat                                  {find EDT using least squares}
     Inc(a);
     r1 := r1 + TimePerPt;                 {sum x}
     r2 := r2 + TimePerPt*TimePerPt;       {sum x^}
     r3 := r3 + SPL_Array[a];              {sum y}
```

```
      r4 := r4 + SPL_Array[a]*SPL_Array[a]; {sum y^}
      r5 := r5 +(SPL_Array[a]*TimePerPt);   {sum x*y}
      TimePerPt := TimePerPt + TpP;
    until (StartPt - SPL_Array[a]) > 10;  {early decay => 10 dB region}
    AveX := r1/a;                         {mean x}
    AveY := r3/a;                         {mean y}
    SumX := (r2 - r1*r1/a);                   {sum x^2}
    SumY := (r4 - r3*r3/a);                       {sum y^2}
    Gradient := (r5 - r1*r3/a)/(r2 - r1*r1/a);    {gradient}
    Yhat := AveY - AveX*Gradient;                 {y hat equation}
(*    YEq := Yhat + Gradient*x; *)                {y equation}
    error := sqr(r5 - r3*r1/a)/(SumX*SumY);

    if error >= 0.8 then                  {set min error level -> 80 % }
      EDT[b] := abs(60/Gradient)          {calculate EDT time}
    else
      EDT[b] := 0;                        {store EDT error code}
                                          {end EDT calc}
    case rt_1 of
      26 : ref := 20;    { 20dB determination level}
      27 : ref := 30;    { 30dB determination level}
      28 : ref := 40;    { 40dB determination level}
    end;   {case rt_1}
    r1:=0; r2:=0; r3:=0; r4:=0; r5:=0;
    a := 0;
    TimePerPt := 0;
    repeat                                   {now find RT}
      Inc(a);
      r1 := r1 + TimePerPt;                  {sum x}
      r2 := r2 + TimePerPt*TimePerPt;        {sum x^}
      r3 := r3 + SPL_Array[a];               {sum y}
      r4 := r4 + SPL_Array[a]*SPL_Array[a]; {sum y^}
      r5 := r5 +(SPL_Array[a]*TimePerPt);    {sum x*y}
      TimePerPt := TimePerPt + TpP;
    until (StartPt - SPL_Array[a]) > ref;
    AveX := r1/a;                            {mean x}
    AveY := r3/a;                            {mean y}
    SumX := (r2 - r1*r1/a);                      {sum x^2}
    SumY := (r4 - r3*r3/a);                          {sum y^2}
    Gradient := (r5 - r1*r3/a)/(r2 - r1*r1/a);   {gradient}
    Yhat := AveY - AveX*Gradient;                {y hat equation}
(*    YEq := Y^ + Gradient*x; *)                 {y equation}
    error := sqr(r5 - r3*r1/a)/(SumX*SumY);

    if error >= 0.9 then                     {set min error level -> 90 % }
      DecayTime[b] := abs(60/Gradient)       {calculate R.T time}
    else
      DecayTime[b] := 0;                     {full decay time}
End;

Procedure RT_OnePosition(e : byte);    {measures R.T at 1 position}
Var
    a,b,count : byte;
    StopPt : integer;                  { e is the code for plotting RT decay}
    X1,X2,X3,X4     : real;
    flag : boolean;

Begin
 with StatusRec do begin
  a := FreqToD(start_f);
  b := 1; StopPt := 0;
```

```
  flag := true; count := 0;
  repeat                                    { freq step loop }
     clrscr;
     FastWriteWindow(' Measuring at : ',3,10,$0B);
     FastWriteWindow(' '+Long2Str(ThirdOctave[a])+' Hz ',3,27,$70);
     SetSampleRate('rt');
     If o_f <> 3 then
        SetFilters(o_f,ThirdOctave[a]);
     repeat                          { take average of 3 }
        GenOn;                       {noise on}
        Delay(2000);                 {noise builds up}
        GenOff;                      {noise off}
        Ad_Read(2500);               {start reading when noise switched off}
        if Ad_Error = 1 then begin
           ErrorWindow('ERROR - Analyser not ON !');
           DisposeWindow(EraseTopWindow);
           Reverb;                   {start reverb measure routine again}
        end;
        ArrayCon('','1');
        if e = 1 then exit;
        flag := CheckForDecay(rt_1,StopPt);  {make sure decay exists}
        if flag then
             Find_RT(rt_1,b,StopPt)          {continue measuring}
        else  begin
          case b of                          {set error detection parameters}
            1 : if count = 0 then b := b - 1 else b := b + 2;
            2 : if count = 0 then b := b - 1 else b := b + 1;
            3 : if count = 0 then b := b - 1;
          end;
          Inc(count);
        end;
        Inc(b);
     until b > 3;
     b  := 1;                                 {find ave of EDT/RT data}
     X1 := (sqr(EDT[1] + EDT[2] + EDT[3]))/3;
     X2 := sqr(EDT[1]) + sqr(EDT[2]) + sqr(EDT[3]);
     dev1[a] := sqrt((X2 - X1)/2);
     ED_T[a] := (sqrt(X1*3))/3;
     X3 := (sqr(DecayTime[1] + DecayTime[2] + DecayTime[3]))/3;
     X4 := sqr(DecayTime[1]) + sqr(DecayTime[2]) + sqr(DecayTime[3]);
     dev2[a] := sqrt((X4 - X3)/2);
     R_T[a] := (sqrt(X3*3))/3;               {store results in arrays}
     Inc(a);
  until stop_f <= ThirdOctave[a-1];          {measure till last freq selected}
 end;  {StatusRec}
end;   {OnePos}

Procedure RT_Measure(n_o_s : byte);  {routine to control R.T measurements}
var
   a,mpos : byte;                    { mpos = mic position} .
begin
 clrscr;
 if n_o_s = 23 then                  { 1 mic positions}
    mpos := 1
 else
    mpos := 3;                       { 3 mic positions}
 with StatusRec do begin
  repeat
   case i_c of
     10 : case MicPos of      { 1 mic }
         1..3 : begin
```

```
              FastWriteWindow(' Put microphone in position  ',3,7,$0B);
              FastWriteWindow(' '+Long2Str(MicPos)+' ',3,35,$70);
              If MicPos = 1 then
                 FastWriteWindow(' Hit ENTER to run test ',6,12,$F0)
              Else
                 FastWriteWindow('     Hit ENTER to continue     ',6,7,$F0
                 readln;
            end;
        end;    {case MicPos}
   11 : case MicPos of     { 2 mics }
         1 : begin
               FastWriteWindow(' Put CH 1 microphone in position  ',3,4,$0B)
               FastWriteWindow(' '+Long2Str(MicPos)+' ',3,37,$70);
               FastWriteWindow(' Put CH 2 microphone in position  ',4,4,$0B)
               FastWriteWindow(' '+Long2Str(MicPos+2)+' ',4,37,$70);
               FastWriteWindow(' Hit ENTER to run test ',6,12,$F0);
               readln;
             end;
         3 : begin
                FastWriteWindow(' Put CH 2 microphone in position  ',3,4,$0
                FastWriteWindow(' '+Long2Str(MicPos)+' ',3,37,$70);
                FastWriteWindow('     Hit ENTER to continue     ',6,7,$F0);
                readln;
              end;
        end;   {case MicPos}
   12 : case MicPos of      { 3 mics }
         1..3 : begin
                 FastWriteWindow(' Measuring R.T in REVERB room ',3,10,$0B)
                 Delay(2000);
               end;
        end;    {case MicPos}
end;    {case i_c}
clrscr;
case MicPos of
  1,2,3 : FastWriteWindow('   Working...Reverb Times   ',6,8,$70);
end;
case i_c of
  10 : SetInChan(10);                  { 1 mic input channel}
  11 : case MicPos of                  { 2 mic input channels}
         1   : SetInChan(10);
         2,3 : SetInChan(11);
       end;
  12 : case MicPos of                  { 3 mics }
         1 : SetInChan(10);
         2 : SetInChan(11);
         3 : SetInChan(12);
       end;
end;  {case i_c}
RT_OnePosition(0);                     {measure RT at each position}
case MicPos of
  1 : for a := 1 to 20 do              {store results in arrays}
         Pos1[a] := R_T[a];
  2 : for a := 1 to 20 do
         Pos2[a] := R_T[a];
  3 : begin
        for a := 1 to 20 do
           Pos3[a] := R_T[a];
        for a := 1 to 20 do
           TempArray[a] := (Pos1[a] + Pos2[a] + Pos3[a])/3;
      end;
end;   {case MicPos}
```

```
  Inc(MicPos);
  until MicPos > mpos;
 end;   {StatusRec}
end;

Procedure WriteToDiskRT(code : byte);   {write RT results to disk}
var
   c : byte;
begin
   case code of
        0 : writeln(TempFile,'Reverb Time averages');
        1 : writeln(TempFile,'Empty Room R.T averages');
        2 : writeln(TempFile,'Room+Sample R.T averages');
   end;
   for c := 1 to 20 do
   begin
     write(TempFile,ThirdOctave[c]);          { each freq}
     write(TempFile,'  ');
     write(TempFile,ED_T[c]:4:1);             { EDT times}
     write(TempFile,'  ');
     write(TempFile,dev1[c]:0:1);             { deviations}
     write(TempFile,'  ');
     if MicPos = 2 then
        write(TempFile,Pos1[c]:4:1)           { RT times }
     else
        write(TempFile,TempArray[c]:4:1);     { RT times }
     write(TempFile,'  ');
     writeln(TempFile,dev2[c]:0:1);
   end
end;

Procedure Reverb;           {controls complete RT measurement process}
begin
  Calibrate;
  If not MakeWindow(TempWin,18,11,63,17,true,true,true,$0,$0E,$70,
     ' Reverberation Time Test ') then ErrorMem;
  If not DisplayWindow(TempWin) then ErrorMem;
  FastWriteWindow('Enter filename for R.T test,',2,6,$0B);
  ReadString('not more than 8 chars: ',15,24,8,$0B,$70,$70,Escaped,name);
  clrscr;
  FastWriteWindow('Enter description of room/sample:',2,6,$0B);
  ReadString('',15,21,40,$0B,$70,$70,Escaped,desc);
  clrscr;
  InFileName := ForceExtension(name,'RTD');
  Assign(TempFile,InFileName);                {make data file}
  Rewrite(TempFile);
  writeln(TempFile,desc);

  With StatusRec do begin
   MicPos := 1;                               {start at mic position 1}
   RT_Measure(n_o_s);                         {do measurements}
   WriteToDiskRT(0);                          {store results on disk}
   Close(TempFile);
  end;   {case StatusRec}

  clrscr;
  FastWriteWindow(' Please move to ',2,3,$0B);
  FastWriteWindow(' Options ',2,20,$70);
  FastWriteWindow(' for results ',2,30,$0B);
  FastWriteWindow('  and other Standard calculations. ',3,5,$0B);
  FastWriteWindow('   Hit SPACEBAR to continue   ',6,8,$F0);
```

```
  repeat until keypressed;
  DisposeWindow(EraseTopWindow);
end;

Procedure Absorb;       {controls complete absorption measure process}
begin
  Calibrate;
  If not MakeWindow(TempWin,18,11,63,17,true,true,true,$0,$0E,$70,
     ' Absorption Test ') then ErrorMem;
  If not DisplayWindow(TempWin) then ErrorMem;
  FastWriteWindow('Enter filename for Absorption Test,',2,6,$0B);
  ReadString('not more than 8 chars: ',15,24,8,$0B,$70,$70,Escaped,name);
  clrscr;
  FastWriteWindow('Enter description of room/sample:',2,6,$0B);
  ReadString('',15,21,40,$0B,$70,$70,Escaped,desc);
  clrscr;
  InFileName := ForceExtension(name,'ATD');
  Assign(TempFile,InFileName);
  Rewrite(TempFile);
  writeln(TempFile,desc);

  with StatusRec do begin
   MicPos := 1;                     {start at mic position 1}
   RT_Measure(n_o_s);               {do empty room measurements}
   WriteToDiskRT(1);                {store RT data}
   MicPos := 1;
   RT_Measure(n_o_s);               {do room with sample measurements}
   WriteToDiskRT(2);                {store RT data}
   Close(TempFile);
  end;    {case StatusRec}

  clrscr;
  FastWriteWindow(' Please move to ',2,3,$0B);
  FastWriteWindow(' Options ',2,20,$70);
  FastWriteWindow(' for results ',2,30,$0B);
  FastWriteWindow('  and other Standard calculations. ',3,5,$0B);
  FastWriteWindow('   Hit SPACEBAR to continue   ',6,8,$F0);
  repeat until keypressed;
  DisposeWindow(EraseTopWindow);
end;

END.     {unit measure}
```

```
Unit Menus;             { All menu routines }

Interface
uses TPCrt,TPString,TPMenu,TPWindow,TPEdit,
     CctCont,Filter;
var
   ch                 : char;
   main,config        : MENU;
   V,Win,TempWin      : WindowPtr;
   Escaped,convert    : boolean;
   Mstack             : MenuStackP;
   key                : MenuKey;

Procedure Edit_Status;
Procedure Show_Status;
Procedure ErrorMem;
Procedure InitMenu(var M : Menu);
Procedure InitConfig(var M : Menu);
Procedure Display_Main(str : string);
Procedure UpDateStatus(key : integer);
Procedure WriteStatus;

Implementation

Procedure ErrorMem;         {displays error if pop-up menus/windows}
begin                       {cannot be shown due to not enough memory}
    Window(1,1,80,25);
    NormVideo; Clrscr;
    Writeln('Insufficient memory'); readln;
    Halt(1);
end;

Function check_f(temp_f : integer): boolean;  {checks if freq selected by}
var                                           {user is valid}
   a : byte;
begin
     check_f := false;                      {default : freq incorrect}
     for a:= 1 to 20 do
         if temp_f = ThirdOctave[a] then    {checks freq with array of}
     begin                                  {standard freq's}
         check_f := true; exit;             {sets answer to true if freq}
     end;                                   {is valid}
end;

Function get_start_freq(start_freq : string): string;  {get filter start}
begin                                                  {freq from user}
  with StatusRec do
  repeat
    convert := Str2Int(start_freq,start_f);  {convert string to integer}
    If not MakeWindow(TempWin,10,20,45,23,true,true,true,$0,$0E,$70,
       ' Edit Frequency ') then ErrorMem;
    If not DisplayWindow(TempWin) then ErrorMem;
    ReadInteger('Enter the START FREQUENCY: ',22,13,4,$0B,$70,100,8000,
         Escaped,start_f);
    if check_f(start_f) = false then writeln(#7); {if freq = wrong then beep}
    get_start_freq := Long2Str(start_f);     {convert integer to string}
    DisposeWindow(EraseTopWindow);
  until check_f(start_f);                    {keep trying to get correct freq}
end;

Function get_stop_freq(stop_freq : string): string; {get filter stop freq}
```

```
begin                                                   {from user}
  with StatusRec do
  repeat
    convert := Str2Int(stop_freq,stop_f);
    If not MakeWindow(TempWin,10,20,45,23,true,true,true,$0,$0E,$70,
       ' Edit Frequency ') then ErrorMem;
    If not DisplayWindow(TempWin) then ErrorMem;
    ReadInteger('Enter the STOP FREQUENCY: ',22,13,4,$0B,$70,100,8000,
         Escaped,stop_f);
    if check_f(stop_f) = false then writeln(#7);
    get_stop_freq := Long2Str(stop_f);
    DisposeWindow(EraseTopWindow);
  until check_f(stop_f);
end;

Function get_range(range_pos : string): string;  {get range position of}
begin                                            {SLM from user}
  with StatusRec do begin
    convert := Str2Int(range_pos,range_p);
    If not MakeWindow(TempWin,10,20,45,23,true,true,true,$0,$0E,$70,
       ' Edit Range ') then ErrorMem;
    If not DisplayWindow(TempWin) then ErrorMem;
    ReadInteger('Enter the RANGE POSITION: ',22,13,3,$0B,$70,0,150,
         Escaped,range_p);
    get_range := Long2Str(range_p);
    DisposeWindow(EraseTopWindow);
  end;
end;

Procedure UpDateStatus(key : integer);  {update status menu display}
begin                                   {with new parameters}
 with StatusRec do begin
  case integer(key) of                  {update status menu}
    1 : begin noise_type  := 'White';        n_t := key; end;
    2 : begin noise_type  := 'Pink';         n_t := key; end;
    3 : begin out_filt    := 'Linear';       o_f := key; end;
    4 : begin out_filt    := 'Third Octave'; o_f := key; end;
    5 : begin out_filt    := 'One Octave';   o_f := key; end;
    6 : begin noise_burst := 'Auto-Gated';   n_b := key; end;
    7 : begin noise_burst := 'Continuous';   n_b := key; end;
    8 : begin out_chan    := 'CHAN A';       o_c := 8; end;      {noise O/P on
    9 : begin out_chan    := 'CHAN A + B';   o_c := 8; end;      {noise O/P on
   10 : begin in_chan     := 'CH 1 only';    i_c := key; end;    {using 1 mic}
   11 : begin in_chan     := 'CH 1 -> 2';    i_c := key-1; end;  {using 2 mic's
   12 : begin
          If n_o_s = 23 then begin  {check if correct no. of mics used}
            writeln(#7);            {depending on no. of mic positions}
            FastWriteWindow(' Error ! Can"t have more than 2  ',6,1,$70);
            FastWriteWindow('   mics if no. of samples = 1    ',7,1,$70);
            Delay(4000);
            clrscr;
          end
          Else begin
            in_chan := 'CH 1 -> 3';   {using 3 mic's}
            i_c     := key;
          end;
        end;
   15 : begin
          If n_o_s = 23 then begin
            writeln(#7);
            FastWriteWindow(' Error ! Can"t have more than 2  ',6,1,$70);
```

```
          FastWriteWindow('   mics if no. of samples = 1     ',7,1,$70);
          Delay(4000);
          clrscr;
        end
        Else begin
          in_chan := 'CH 1 -> 6';    {using 6 mic's}
          i_c     := key;
        end;
      end;
   16 : begin weight  := 'None';          w_t := key; end;
   17 : begin weight  := 'A - Weight';    w_t := key; end;
   18 : begin weight  := 'C - Weight';    w_t := key; end;
   19 : begin in_filt := 'None';          i_f := key; end;
   20 : begin in_filt := 'Active';        i_f := key; end;
   21 : start_freq    := get_start_freq(start_freq);
   22 : stop_freq     := get_stop_freq(stop_freq);
   23 : begin
        If (i_c = 12) or (i_c = 15) then begin       {if > 1 mic used then}
          writeln(#7);                             {sound the error bell}
          FastWriteWindow('   Error !   Must first limit     ',6,1,$70);
          FastWriteWindow('   input mics to less than 3      ',7,1,$70);
          Delay(4000);
          clrscr;
        end
        Else begin
          no_of_samp := 'One';  {measure at 1 position}
          n_o_s      := key;
        end;
       end;
   24 : begin no_of_samp := 'Three';  n_o_s := key; end;  {measure at 3 pos.}
   26 : begin rt_level   := '20 dB';  rt_l  := key; end;  {reverb. determ.}
   27 : begin rt_level   := '30 dB';  rt_l  := key; end;       {levels}
   28 : begin rt_level   := '40 dB';  rt_l  := key; end;
   29 : range_pos        := get_range(range_pos);
  end;
    UpDateCircuit;                 {update configuration of circuitry}
    CFactor := CFactor;            {set correction factor}
    Write(StatusFile,StatusRec);   {store new status data to file}
    Reset(StatusFile);
 end;
end;

Procedure WriteStatus;             {write status of BAA to screen display}
var
   R : char;
begin
   with StatusRec do begin
     FastWrite('Type of noise     :',9,47,$07);
        FastWrite(' '+Pad(noise_type,5)+' ',9,65,$0B);
     FastWrite('Noise filtering   :',10,47,$07);
        FastWrite(' '+Pad(out_filt,12)+' ',10,65,$0B);
     FastWrite('Noise burst       :',11,47,$07);
        FastWrite(' '+noise_burst+' ',11,65,$0B);
     FastWrite('Noise O/P channel:',12,47,$07);
        FastWrite(' '+Pad(out_chan,10)+' ',12,65,$0B);
     FastWrite('Input channel     :',13,47,$07);
        FastWrite(' '+in_chan+' ',13,65,$0B);
     FastWrite('Input weighting   :',14,47,$07);
        FastWrite(' '+Pad(weight,10)+' ',14,65,$0B);
     FastWrite('Input filtering   :',15,47,$07);
        FastWrite(' '+Pad(in_filt,6)+' ',15,65,$0B);
```

```
      FastWrite('Start frequency  :',16,47,$07);
         FastWrite(' '+Pad(start_freq,4)+' Hz',16,65,$0B);
      FastWrite('Stop frequency   :',17,47,$07);
         FastWrite(' '+Pad(stop_freq,4)+' Hz',17,65,$0B);
      FastWrite('No. of samples   :',18,47,$07);
         FastWrite(' '+Pad(no_of_samp,5)+' ',18,65,$0B);
      FastWrite('R.T level        :',19,47,$07);
         FastWrite(' '+rt_level+' ',19,65,$0B);
      FastWrite('Range position   :',20,47,$07);
         FastWrite(' '+Pad(range_pos,3)+' dB',20,65,$0B);
   end;
end;

Procedure Display_Main(str : string);     {show main border menu}
var
    work,row,col : byte;
begin
    for row := 1 to 24 do                 {fill screen}
      for col := 1 to 80 do
        Fastwrite(#176,row,col,$15);
    for work := 1 to 80 do begin          {horizontal lines}
        Fastwrite(#196,1,work,$0E);
        Fastwrite(#196,25,work,$0E);
    end;
    for work := 1 to 24 do begin          {vertical lines}
        Fastwrite(#179,work,1,$0E);
        Fastwrite(#179,work,80,$0E);
    end;
    Fastwrite(#218,1,1,$0E);              {show corners of border}
    Fastwrite(#191,1,80,$0E);
    Fastwrite(#192,25,1,$0E);
    Fastwrite(#217,25,80,$0E);
    Fastwrite(Center(str,16),1,30,$70);
    Fastwrite(' F1 Help ',25,3,$70);      {display usable function keys}
    Fastwrite(' '#17#45' '#45#16' Move Selector ',25,13,$70);
    Fastwrite(' '#25' Pull Down ',25,35,$70);
    Fastwrite(' '#17#196#217' Select Item ',25,49,$70);
    Fastwrite(' F10 Quit ',25,68,$70);
end;


Procedure InitMenu(var M : Menu);    {setup and display main menu systems}
const
  Color1 : MenuColorArray = ($0E, $70, $07, $70, $0E, $0E);
  Frame1 : FrameArray = '┌└┐┘─│';
  Frame2 : FrameArray = '╓╙╖╜═║';

begin
  {Customize this call for special exit characters and custom item displays}
  M := NewMenu([#187,#196], nil);

  SubMenu(1,1,0,Horizontal,Frame1,Color1,' Main Menu ');
    MenuItem('BAA Status',2,1,0,'');
    SubMenu(1,3,0,Vertical,Frame2,Color1,'');
      MenuItem('Show Status',1,1,30,'');
      MenuItem('Configure System',2,1,31,'');
      PopSublevel;
    MenuItem('Standard Tests',22,1,0,'');
    SubMenu(19,3,0,Vertical,Frame2,Color1,'');
      MenuItem('Reverberation Time',1,1,0,'');
      SubMenu(34,6,0,Vertical,Frame2,Color1,'');
```

```
        MenuItem('Standard',1,1,35,'');
        MenuItem('Absorption',2,1,61,'');
        PopSublevel;
      MenuItem('Transmission Loss',2,1,33,'');
      SubMenu(15,6,0,Vertical,Frame2,Color1,'');
        MenuItem('Level Difference [D]',1,1,50,'');
        MenuItem('Standardized L/Diff [DnT]',2,1,51,'');
        MenuItem('Transmission Loss [R]',3,1,52,'');
        MenuItem('Apparent T/Loss [R'']',4,1,53,'');
        PopSublevel;
      PopSublevel;
    MenuItem('Options',49,1,0,'');
    SubMenu(46,3,0,Vertical,Frame2,Color1,'');
      MenuItem('Calculations',1,1,0,'');
      SubMenu(40,6,0,Vertical,Frame2,Color1,'');
        MenuItem('Level Difference [D]',1,1,38,'');
        MenuItem('Standardized L/Diff [DnT]',2,1,39,'');
        MenuItem('Transmission Loss [R]',3,1,40,'');
        MenuItem('Apparent T/Loss [R'']',4,1,41,'');
        PopSublevel;
      MenuItem('RT Table',2,1,36,'');
      MenuItem('Absorb Table',3,1,60,'');
      MenuItem('Decay Plot',4,1,37,'');
      MenuItem('Hardcopy',5,1,38,'');
      PopSublevel;
    MenuItem('Utilities',69,1,0,'');
    SubMenu(60,3,0,Vertical,Frame2,Color1,'');
      MenuItem('Real Time Display',1,1,32,'');
      MenuItem('Delete Files',2,1,34,'');
      MenuItem('Suspend To Dos',3,1,0,'');
      PopSublevel;
    PopSublevel;

  ResetMenu(M);
end;

Procedure InitConfig(var M : Menu);   {setup and display configure menu}
const
  Color1 : MenuColorArray = ($0E, $70, $07, $70, $0E, $0E);
  Frame1 : FrameArray = '╔╚╗╝═║';

begin
  {Customize this call for special exit characters and custom item displays}
  M := NewMenu([#187,#196], nil);

  SubMenu(1,7,0,Vertical,Frame1,Color1,' Configure Menu ');
    MenuItem('Type of Noise',1,1,0,'');
    SubMenu(21,19,0,Vertical,Frame1,Color1,'');
      MenuItem('White noise',1,1,1,'');
      MenuItem('Pink noise',2,1,2,'');
      PopSublevel;
    MenuItem('Noise Filtering',2,1,0,'');
    SubMenu(20,18,0,Vertical,Frame1,Color1,'');
      MenuItem('Linear',1,1,3,'');
      MenuItem('Third Octave',2,1,4,'');
      MenuItem('One Octave',3,1,5,'');
      PopSublevel;
    MenuItem('Noise Time Burst',3,2,0,'');
    SubMenu(21,18,0,Vertical,Frame1,Color1,'');
      MenuItem('Auto-Gated',1,1,6,'');
      MenuItem('Continuous',2,1,7,'');
```

```
        PopSublevel;
      MenuItem('Noise Output Channel',4,1,0,'');
      SubMenu(20,18,0,Vertical,Frame1,Color1,'');
        MenuItem('Channel A',1,9,8,'');
        MenuItem('Channel A + B',2,13,9,'');
        PopSublevel;
      MenuItem('Input Channel',5,3,0,'');
      SubMenu(21,17,0,Vertical,Frame1,Color1,'');
        MenuItem('CH 1 only',1,4,10,'');
(*      MenuItem('CH 1 -> 2',2,9,11,'');  *)
        MenuItem('CH 1 -> 3',2,9,12,'');
        MenuItem('CH 1 -> 6',3,9,15,'');
        PopSublevel;
      MenuItem('Input Weighting',6,4,0,'');
      SubMenu(20,18,0,Vertical,Frame1,Color1,'');
        MenuItem('None',1,1,16,'');
        MenuItem('A - Weighting',2,1,17,'');
        MenuItem('C - Weighting',3,1,18,'');
        PopSublevel;
      MenuItem('Input Filtering',7,7,0,'');
      SubMenu(22,19,0,Vertical,Frame1,Color1,'');
        MenuItem('None',1,1,19,'');
        MenuItem('Active',2,1,20,'');
        PopSublevel;
      MenuItem('Start Frequency',8,1,21,'');
      MenuItem('Stop Frequency',9,7,22,'');
      MenuItem('No. of Samples',10,9,0,'');
      SubMenu(22,18,0,Vertical,Frame1,Color1,'');
        MenuItem('One',1,1,23,'');
        MenuItem('Three',2,1,24,'');
        PopSublevel;
      MenuItem('R.T Level',11,5,0,'');
      SubMenu(23,18,0,Vertical,Frame1,Color1,'');
        MenuItem('20 dB',1,1,26,'');
        MenuItem('30 dB',2,1,27,'');
        MenuItem('40 dB',3,1,28,'');
        PopSublevel;
      MenuItem('Range Position',12,1,29,'');
      PopSublevel;

  ResetMenu(M);
end;

Procedure Edit_Status;       {edit status of BAA circuitry}
begin
   Repeat                    {clear windows from screen}
    V := EraseTopWindow;
    DisposeWindow(V);
   Until V = nil;
   EraseMenu(main,False);
   InitConfig(config);                {show configure menu system}
   Display_Main('Configure Menu');
   If not MakeWindow(Win,45,7,79,21,true,true,true,$0,$0E,$70,
      ' Analyser Status ') then ErrorMem;
   If not DisplayWindow(Win) then ErrorMem;
   WriteStatus;                       {write status data to screen}
   repeat
    key := MenuChoice(config,ch);
    If ord(ch) = 13 then              {enter-key choice for editing parameters}
    case integer(key) of
      1..29 : begin UpDateStatus(key); WriteStatus;  end;
```

```
    end;
   until ord(ch) = 196;              {until user quits}
   DisposeWindow(Win);
   Repeat                            {clear windows from screen}
    V := EraseTopWindow;
    DisposeWindow(V);
   Until V = nil;
   EraseMenu(config,False);          {remove configure menu system from screen}
   InitMenu(main); ch := ^A;
end;

Procedure Show_Status;      {display status of BAA on screen}
var
   Key : word;
   Ch  : Char absolute Key;
begin
  If not MakeWindow(TempWin,45,7,79,21,true,true,true,$0,$0E,$70,
     ' Analyser Status ') then ErrorMem;
  If not DisplayWindow(TempWin) then ErrorMem;
  WriteStatus;               {write status data to screen}
  repeat
     FastWriteWindow(' Change status [N] ? ',14,7,$70);
     Key := ReadKeyWord;
 until upcase(Ch) in ['Y','N',^M];     {answer yes/no}
 DisposeWindow(EraseTopWindow);
 case upcase(Ch) of
    'Y' : Edit_Status;
    'N' : Exit;
    ^M  : begin ch := ^A; Exit; end;   {remove status display}
 end;
end;

END.      { unit menus }
```

```
Unit filter;            { one and third octave filter configuration }

Interface
const
   OneOctave   : array [0..6] of integer  = (0,125,250,500,1000,2000,4000);
   ThirdOctave : array [0..20] of integer = (0,100,125,160,200,250,315,400,500,
                                             630,800,1000,1250,1600,2000,2500,
                                             3150,4000,5000,6300,8000);

Procedure SetFilters(o_f,start_f : integer);

Implementation
uses TPCrt;

var
   add,m,f,q    : integer;
   temp                  : boolean;

procedure setdac(A1,A2  : byte);    {programs the DAC O/P for filter clock}
begin
     port[$262] := $f;      { resets all programming clock lines high}
     port[$260] := A1;      {sets data for DAC latch 1}
     port[$262] := $04;     {writes data to latch 1 address}
     port[$262] := $f;      { resets clock lines high again}

     port[$260] := A2;      {sets data for DAC latch 2}
     port[$262] := $05;     {writes data to latch 2 address}
     port[$262] := $f;      { resets clock lines high again}
end;

procedure set_smoothing(A3 : byte);
begin
     port[$262] := $f;
     port[$260] := A3;      {sets data for smoothing latch}
     port[$262] := $0;      {writes data to smoothing latch address}
     port[$262] := $f;
end;

Procedure output1;              {write codes to filter A}
var
  e,z,x : integer;
begin
  temp := true;
  e:= (add+m-1);              {e : calculates mode code for filter A}
  port[$262] := $f;
  port[$260] := e;            {data code for filter A}
  port[$262] := $01;          {sets data for latch for filter A}
  port[$262] := $02;          {writes data to filter A}
  port[$262] := $f;
  add:= add+4;
  for z:= 1 to 3 do
   begin
     x:= (add+(f-4*trunc(f/4)));   {x : calculates filter clock ratio}
     port[$262] := $f;             {for filter A}
     port[$260] := x;
     port[$262] := $01;
     port[$262] := $02;
     port[$262] := $f;
     f:= trunc(f/4);
     add:= add+4;
```

```
    end;
   for z:= 1 to 4 do
    begin
      x:= (add+(q-4*trunc(q/4)));    {x : calculates Q selection code}
      port[$262] := $f;              {for filter A}
      port[$260] := x;
      port[$262] := $01;
      port[$262] := $02;
      port[$262] := $f;
      q:= trunc(q/4);
      add:= add+4;
    end;
 end;

 procedure output2;               {write codes to filter B}
 var
   e,z,x : integer;
 begin
   e:= (add+m-1);                 {e : calculates mode code for filter B}
   port[$262] := $f;
   port[$260] := e;               {data code for filter B}
   port[$262] := $01;             {sets data for latch for filter B}
   port[$262] := $03;             {writes data to filter B}
   port[$262] := $f;
   add:= add+4;
   for z:= 1 to 3 do
    begin
      x:= (add+(f-4*trunc(f/4)));    {x : calculates filter clock ratio}
      port[$262] := $f;              {for filter B}
      port[$260] := x;
      port[$262] := $01;
      port[$262] := $03;
      port[$262] := $f;
      f:= trunc(f/4);
      add:= add+4;
    end;
   for z:= 1 to 4 do
    begin
      x:= (add+(q-4*trunc(q/4)));    {x : calculates Q selection code}
      port[$262] := $f;              {for filter B}
      port[$260] := x;
      port[$262] := $01;
      port[$262] := $03;
      port[$262] := $f;
      q:= trunc(q/4);
      add:= add+4;
    end;
end;

{******************* one octave filter programming ****************}

{ m = filter mode, f = clock ratio code, q = Q selection code}
{ refer to appendix 4 for code calculation method }

{*****************************************************************}

procedure set_o100;       {set_o100 refers to one octave bandpass: 100 Hz}
begin
   add:= 0;  m:= 1; f:= 57;  q:= 107;   {inner left peak of center freq}
   output1;                             {program filter A}
   add:= 32; m:= 1; f:= 10;  q:= 108;   {inner right peak of center freq}
```

```
   output1;                                    {program filter A}
   add:= 0;  m:= 1; f:= 41;  q:= 75;           {outer left peak of center freq}
   output2;                                    {program filter B}
   add:= 32; m:= 1; f:= 20;  q:= 75;           {outer right peak of center freq}
   output2;                                    {program filter B}
   setdac($01,$0);                             {sets filter A & B clock freq}
   set_smoothing($0);                          {configures smoothing circuitry}
end;

procedure set_o125;       {octave, 125 Hz}
begin
   add:= 0;  m:= 1; f:= 47;  q:= 107;
   output1;
   add:= 32; m:= 1; f:=  6;  q:= 108;
   output1;
   add:= 0;  m:= 1; f:= 34;  q:= 75;
   output2;
   add:= 32; m:= 1; f:= 15;  q:= 76;
   output2;
   setdac($02,$0);
   set_smoothing($0);
end;

procedure set_o160;
begin
   add:= 0;  m:= 1; f:= 46;  q:= 107;
   output1;
   add:= 32; m:= 1; f:=  6;  q:= 108;
   output1;
   add:= 0;  m:= 1; f:= 32;  q:= 75;
   output2;
   add:= 32; m:= 1; f:= 14;  q:= 76;
   output2;
   setdac($04,$0);
   set_smoothing($2);
end;

procedure set_o200;
begin
   add:= 0;  m:= 1; f:= 44;  q:= 107;
   output1;
   add:= 32; m:= 1; f:=  5;  q:= 108;
   output1;
   add:= 0;  m:= 1; f:= 31;  q:= 75;
   output2;
   add:= 32; m:= 1; f:= 13;  q:= 76;
   output2;
   setdac($06,$0);
   set_smoothing($2);
end;

procedure set_o250;
begin
   add:= 0;  m:= 1; f:= 45;  q:= 107;
   output1;
   add:= 32; m:= 1; f:=  5;  q:= 108;
   output1;
   add:= 0;  m:= 1; f:= 32;  q:= 75;
   output2;
   add:= 32; m:= 1; f:= 14;  q:= 76;
   output2;
```

```
   setdac($09,$0);
   set_smoothing($2);
end;

procedure set_o315;
begin
   add:= 0;  m:= 1; f:= 44;  q:= 109;
   output1;
   add:= 32; m:= 1; f:=  7;  q:= 109;
   output1;
   add:= 0;  m:= 1; f:= 32;  q:= 79;
   output2;
   add:= 32; m:= 1; f:= 15;  q:= 79;
   output2;
   setdac($0d,$0);
   set_smoothing($4);
end;

procedure set_o400;
begin
   add:= 0;  m:= 1; f:= 46;  q:= 107;
   output1;
   add:= 32; m:= 1; f:=  6;  q:= 108;
   output1;
   add:= 0;  m:= 1; f:= 33;  q:= 75;
   output2;
   add:= 32; m:= 1; f:= 15;  q:= 76;
   output2;
   setdac($11,$0);
   set_smoothing($4);
end;

procedure set_o500;
begin
   add:= 0;  m:= 1; f:= 43;  q:= 107;
   output1;
   add:= 32; m:= 1; f:=  4;  q:= 108;
   output1;
   add:= 0;  m:= 1; f:= 30;  q:= 75;
   output2;
   add:= 32; m:= 1; f:= 13;  q:= 76;
   output2;
   setdac($15,$0);
   set_smoothing($4);
end;

procedure set_o630;
begin
   add:= 0;  m:= 1; f:= 45;  q:= 107;
   output1;
   add:= 32; m:= 1; f:=  5;  q:= 108;
   output1;
   add:= 0;  m:= 1; f:= 32;  q:= 75;
   output2;
   add:= 32; m:= 1; f:= 14;  q:= 76;
   output2;
   setdac($1c,$0);
   set_smoothing($4);
end;

procedure set_o800;
```

```
begin
   add:= 0;  m:= 1; f:= 46;  q:= 107;
   output1;
   add:= 32; m:= 1; f:=  6;  q:= 108;
   output1;
   add:= 0;  m:= 1; f:= 33;  q:= 75;
   output2;
   add:= 32; m:= 1; f:= 14;  q:= 76;
   output2;
   setdac($25,$0);
   set_smoothing($4);
end;

procedure set_o1000;
begin
   add:= 0;  m:= 1; f:= 44;  q:= 100;
   output1;
   add:= 32; m:= 1; f:=  5;  q:= 108;
   output1;
   add:= 0;  m:= 1; f:= 31;  q:=  70;
   output2;
   add:= 32; m:= 1; f:= 13;  q:=  76;
   output2;
   setdac($2e,$0);
   set_smoothing($4);
end;

procedure set_o1250;
begin
   add:= 0;  m:= 1; f:= 43;  q:= 107;
   output1;
   add:= 32; m:= 1; f:=  5;  q:= 108;
   output1;
   add:= 0;  m:= 1; f:= 31;  q:= 75;
   output2;
   add:= 32; m:= 1; f:= 13;  q:= 76;
   output2;
   setdac($3a,$0);
   set_smoothing($6);
end;

procedure set_o1600;
begin
   add:= 0;  m:= 1; f:= 45;  q:= 107;
   output1;
   add:= 32; m:= 1; f:=  5;  q:= 108;
   output1;
   add:= 0;  m:= 1; f:= 31;  q:= 75;
   output2;
   add:= 32; m:= 1; f:= 14;  q:= 76;
   output2;
   setdac($4b,$0);
   set_smoothing($6);
end;

procedure set_o2000;
begin
   add:= 0;  m:= 1; f:= 45;  q:= 107;
   output1;
   add:= 32; m:= 1; f:=  5;  q:= 108;
   output1;
```

```
   add:= 0;  m:= 1; f:= 32;  q:= 75;
   output2;
   add:= 32; m:= 1; f:= 14;  q:= 76;
   output2;
   setdac($60,$0);
   set_smoothing($6);
end;

procedure set_o2500;
begin
   add:= 0;  m:= 1; f:= 43;  q:= 107;
   output1;
   add:= 32; m:= 1; f:=  4;  q:= 108;
   output1;
   add:= 0;  m:= 1; f:= 30;  q:= 75;
   output2;
   add:= 32; m:= 1; f:= 13;  q:= 76;
   output2;
   setdac($77,$0);
   set_smoothing($6);
end;

procedure set_o3150;
begin
   add:= 0;  m:= 1; f:= 45;  q:= 107;
   output1;
   add:= 32; m:= 1; f:=  5;  q:= 108;
   output1;
   add:= 0;  m:= 1; f:= 32;  q:= 75;
   output2;
   add:= 32; m:= 1; f:= 14;  q:= 76;
   output2;
   setdac($9b,$0);
   set_smoothing($6);
end;

procedure set_o4000;
begin
   add:= 0;  m:= 1; f:= 44;  q:= 107;
   output1;
   add:= 32; m:= 1; f:=  5;  q:= 108;
   output1;
   add:= 0;  m:= 1; f:= 31;  q:= 75;
   output2;
   add:= 32; m:= 1; f:= 13;  q:= 76;
   output2;
   setdac($c2,$0);
   set_smoothing($6);
end;

procedure set_o5000;
begin
   add:= 0;  m:= 1; f:= 44;  q:= 107;
   output1;
   add:= 32; m:= 1; f:=  5;  q:= 108;
   output1;
   add:= 0;  m:= 1; f:= 31;  q:= 75;
   output2;
   add:= 32; m:= 1; f:= 13;  q:= 76;
   output2;
   setdac($f9,$0);
```

```
   set_smoothing($6);
end;

procedure set_o6300;
begin
   add:= 0;  m:= 1; f:= 43;  q:= 107;
   output1;
   add:= 32; m:= 1; f:=  5;  q:= 108;
   output1;
   add:= 0;  m:= 1; f:= 30;  q:= 75;
   output2;
   add:= 32; m:= 1; f:= 13;  q:= 76;
   output2;
   setdac($3b,$1);
   set_smoothing($6);
end;

procedure set_o8000;
begin
   add:= 0;  m:= 1; f:= 45;  q:= 107;
   output1;
   add:= 32; m:= 1; f:=  5;  q:= 108;
   output1;
   add:= 0;  m:= 1; f:= 32;  q:= 75;
   output2;
   add:= 32; m:= 1; f:= 14;  q:= 76;
   output2;
   setdac($9e,$1);
   set_smoothing($6);
end;

{******************* third octave filter programming *****************}

procedure set_t100;     {set_t100 refers to third octave bandpass: 100Hz}
begin
   add:= 0;  m:= 1; f:= 39;  q:= 119;
   output1;
   add:= 32; m:= 1; f:= 26;  q:= 122;
   output1;
   add:= 0;  m:= 1; f:= 37;  q:= 111;
   output2;
   add:= 32; m:= 1; f:= 30;  q:= 104;
   output2;
   setdac($02,$0);
   set_smoothing($0);
end;

procedure set_t125;
begin
   add:= 0;  m:= 1; f:= 39;  q:= 119;
   output1;
   add:= 32; m:= 1; f:= 26;  q:= 122;
   output1;
   add:= 0;  m:= 1; f:= 37;  q:= 111;
   output2;
   add:= 32; m:= 1; f:= 30;  q:= 104;
   output2;
   setdac($04,$0);
   set_smoothing($0);
end;
```

```
procedure set_t160;
begin
   add:= 0;  m:= 1; f:= 40;  q:= 120;
   output1;
   add:= 32; m:= 1; f:= 27;  q:= 120;
   output1;
   add:= 0;  m:= 1; f:= 35;  q:= 109;
   output2;
   add:= 32; m:= 1; f:= 29;  q:= 106;
   output2;
   setdac($07,$0);
   set_smoothing($2);
end;

procedure set_t200;
begin
   add:= 0;  m:= 1; f:= 43;  q:= 123;
   output1;
   add:= 32; m:= 1; f:= 27;  q:= 120;
   output1;
   add:= 0;  m:= 1; f:= 36;  q:= 113;
   output2;
   add:= 32; m:= 1; f:= 28;  q:= 107;
   output2;
   setdac($0a,$0);
   set_smoothing($2);
end;

procedure set_t250;
begin
   add:= 0;  m:= 1; f:= 44;  q:= 120;
   output1;
   add:= 32; m:= 1; f:= 28;  q:= 120;
   output1;
   add:= 0;  m:= 1; f:= 39;  q:= 112;
   output2;
   add:= 32; m:= 1; f:= 32;  q:= 111;
   output2;
   setdac($0e,$0);
   set_smoothing($2);
end;

procedure set_t315;
begin
   add:= 0;  m:= 1; f:= 41;  q:= 119;
   output1;
   add:= 32; m:= 1; f:= 28;  q:= 120;
   output1;
   add:= 0;  m:= 1; f:= 38;  q:= 108;
   output2;
   add:= 32; m:= 1; f:= 31;  q:= 107;
   output2;
   setdac($12,$0);
   set_smoothing($4);
end;

procedure set_t400;
begin
   add:= 0;  m:= 1; f:= 38;  q:= 119;
   output1;
   add:= 32; m:= 1; f:= 25;  q:= 120;
```

```
   output1;
   add:= 0;  m:= 1; f:= 35;  q:= 108;
   output2;
   add:= 32; m:= 1; f:= 28;  q:= 108;
   output2;
   setdac($16,$0);
   set_smoothing($4);
end;

procedure set_t500;
begin
   add:= 0;  m:= 1; f:= 39;  q:= 119;
   output1;
   add:= 32; m:= 1; f:= 27;  q:= 119;
   output1;
   add:= 0;  m:= 1; f:= 36;  q:= 108;
   output2;
   add:= 32; m:= 1; f:= 29;  q:= 106;
   output2;
   setdac($1d,$0);
   set_smoothing($4);
end;

procedure set_t630;
begin
   add:= 0;  m:= 1; f:= 42;  q:= 119;
   output1;
   add:= 32; m:= 1; f:= 28;  q:= 119;
   output1;
   add:= 0;  m:= 1; f:= 37;  q:= 108;
   output2;
   add:= 32; m:= 1; f:= 31;  q:= 107;
   output2;
   setdac($27,$0);
   set_smoothing($4);
end;

procedure set_t800;
begin
   add:= 0;  m:= 1; f:= 39;  q:= 119;
   output1;
   add:= 32; m:= 1; f:= 26;  q:= 120;
   output1;
   add:= 0;  m:= 1; f:= 36;  q:= 109;
   output2;
   add:= 32; m:= 1; f:= 29;  q:= 108;
   output2;
   setdac($2f,$0);
   set_smoothing($4);
end;

procedure set_t1000;
begin
   add:= 0;  m:= 1; f:= 38;  q:= 119;
   output1;
   add:= 32; m:= 1; f:= 26;  q:= 119;
   output1;
   add:= 0;  m:= 1; f:= 36;  q:= 108;
   output2;
   add:= 32; m:= 1; f:= 28;  q:= 108;
   output2;
```

```
   setdac($3c,$0);
   set_smoothing($4);
end;

procedure set_t1250;
begin
   add:= 0;  m:= 1; f:= 39;  q:= 119;
   output1;
   add:= 32; m:= 1; f:= 27;  q:= 119;
   output1;
   add:= 0;  m:= 1; f:= 36;  q:= 107;
   output2;
   add:= 32; m:= 1; f:= 29;  q:= 107;
   output2;
   setdac($4d,$0);
   set_smoothing($6);
end;

procedure set_t1600;
begin
   add:= 0;  m:= 1; f:= 39;  q:= 119;
   output1;
   add:= 32; m:= 1; f:= 27;  q:= 120;
   output1;
   add:= 0;  m:= 1; f:= 36;  q:= 106;
   output2;
   add:= 32; m:= 1; f:= 30;  q:= 106;
   output2;
   setdac($63,$0);
   set_smoothing($6);
end;

procedure set_t2000;
begin
   add:= 0;  m:= 1; f:= 38;  q:= 119;
   output1;
   add:= 32; m:= 1; f:= 26;  q:= 119;
   output1;
   add:= 0;  m:= 1; f:= 35;  q:= 107;
   output2;
   add:= 32; m:= 1; f:= 29;  q:= 107;
   output2;
   setdac($7a,$0);
   set_smoothing($6);
end;

procedure set_t2500;
begin
   add:= 0;  m:= 1; f:= 39;  q:= 119;
   output1;
   add:= 32; m:= 1; f:= 27;  q:= 119;
   output1;
   add:= 0;  m:= 1; f:= 36;  q:= 107;
   output2;
   add:= 32; m:= 1; f:= 29;  q:= 107;
   output2;
   setdac($9e,$0);
   set_smoothing($6);
end;

procedure set_t3150;
```

```
begin
   add:= 0;  m:= 1; f:= 38;  q:= 119;
   output1;
   add:= 32; m:= 1; f:= 26;  q:= 119;
   output1;
   add:= 0;  m:= 1; f:= 35;  q:= 107;
   output2;
   add:= 32; m:= 1; f:= 29;  q:= 107;
   output2;
   setdac($c5,$0);
   set_smoothing($6);
end;

procedure set_t4000;
begin
   add:= 0;  m:= 1; f:= 38;  q:= 119;
   output1;
   add:= 32; m:= 1; f:= 25;  q:= 120;
   output1;
   add:= 0;  m:= 1; f:= 35;  q:= 108;
   output2;
   add:= 32; m:= 1; f:= 29;  q:= 108;
   output2;
   setdac($fb,$0);
   set_smoothing($6);
end;

procedure set_t5000;
begin
   add:= 0;  m:= 1; f:= 38;  q:= 119;
   output1;
   add:= 32; m:= 1; f:= 26;  q:= 119;
   output1;
   add:= 0;  m:= 1; f:= 35;  q:= 108;
   output2;
   add:= 32; m:= 1; f:= 28;  q:= 107;
   output2;
   setdac($3d,$1);
   set_smoothing($6);
end;

procedure set_t6300;
begin
   add:= 0;  m:= 1; f:= 39;  q:= 119;
   output1;
   add:= 32; m:= 1; f:= 27;  q:= 119;
   output1;
   add:= 0;  m:= 1; f:= 36;  q:= 107;
   output2;
   add:= 32; m:= 1; f:= 30;  q:= 108;
   output2;
   setdac($a5,$1);
   set_smoothing($6);
end;

procedure set_t8000;
begin
   add:= 0;  m:= 1; f:= 38;  q:= 119;
   output1;
   add:= 32; m:= 1; f:= 26;  q:= 120;
   output1;
```

```
   add:= 0;  m:= 1; f:= 36;  q:= 108;
   output2;
   add:= 32; m:= 1; f:= 29;  q:= 108;
   output2;
   setdac($22,$2);
   set_smoothing($6);
end;

Procedure SetFilters(o_f,start_f : integer);   {sets filters to 1/3 or 1/1}
begin                                           {bandpass filtering}
    case start_f of                                       {start_f = start freq}
        100  : if o_f = 5 then set_o100 else set_t100;   {set_o = 1/1 oct and}
        125  : if o_f = 5 then set_o125 else set_t125;   {set_t = 1/3 oct}
        160  : if o_f = 5 then set_o160 else set_t160;
        200  : if o_f = 5 then set_o200 else set_t200;
        250  : if o_f = 5 then set_o250 else set_t250;
        315  : if o_f = 5 then set_o315 else set_t315;
        400  : if o_f = 5 then set_o400 else set_t400;
        500  : if o_f = 5 then set_o500 else set_t500;
        630  : if o_f = 5 then set_o630 else set_t630;
        800  : if o_f = 5 then set_o800 else set_t800;
        1000 : if o_f = 5 then set_o1000 else set_t1000;
        1250 : if o_f = 5 then set_o1250 else set_t1250;
        1600 : if o_f = 5 then set_o1600 else set_t1600;
        2000 : if o_f = 5 then set_o2000 else set_t2000;
        2500 : if o_f = 5 then set_o2500 else set_t2500;
        3150 : if o_f = 5 then set_o3150 else set_t3150;
        4000 : if o_f = 5 then set_o4000 else set_t4000;
        5000 : if o_f = 5 then set_o5000 else set_t5000;
        6300 : if o_f = 5 then set_o6300 else set_t6300;
               8000 : if o_f = 5 then set_o8000 else set_t8000;
        end;
end;

END.     {unit filter}
```

```
Unit Graphics;        {all graphics display routines : plots/tables..}

Interface
Uses Graph,Measure;

TYPE
   Prompt      = string[80];
   temp_s      = string[100];

VAR
   IOErr          : boolean;
   name           : string;
   AbsValue       : RealArrType1;
   OldExitProc                : Pointer;
   gdriver,gmode,ErrorCode    : integer;
   MaxColor,MaxX,MaxY,H       : word;
   ViewInfo                   : ViewPortType;
   Freq1,n,o                  : integer;
   des,desc,desc1,desc2,desc3    : string;


PROCEDURE MyExitProc;
PROCEDURE CsrOff;
PROCEDURE CsrOn;
PROCEDURE Inverse;
PROCEDURE Normal;
FUNCTION Real2Str1(K : real) : temp_s;
FUNCTION Real2Str2(K : real) : temp_s;
Function Real2Str4(K : real) : temp_s;
FUNCTION Real2Str(K : real) : temp_s;
PROCEDURE InitGraphics;
PROCEDURE RT_Table;
PROCEDURE DecayPlot;
PROCEDURE AbsGraph;
PROCEDURE AbsTable;
PROCEDURE T_Loss1;
PROCEDURE WaitToGo;

Implementation
Uses TPCrt,TPWindow,TPString,Filter,CctCont;

{$F+}
Procedure MyExitProc;         {exits graphics mode if error occurs}
Begin
   ExitProc := OldExitProc;
   CloseGraph;
End;
{$F-}

Procedure InitGraphics;        {initialise graphics mode}
Begin
  OldExitProc := ExitProc;
  ExitProc    := @MyExitProc;
  gdriver     := Detect;
  InitGraph(gdriver,gmode,'');
  ErrorCode   := GraphResult;
  If ErrorCode <> grOK then
     ErrorWindow('E R R O R !!');
  MaxColor := GetMaxColor;
  MaxX := GetmaxX;             {get X co-ords of screen}
```

```
end;

Procedure CsrOff;              {switches cursor off}
Begin
   inline ($b4/$01/$b9/$ffff/$cd/$10);
End;

Procedure CsrOn;               {switches cursor on}
Begin
   inline ($b4/$01/$b9/$0d0e/$cd/$10);
End;

Procedure Inverse;             {inverts text color}
Begin
   textbackground(7); textcolor(0);
End;

Procedure Normal;              {shows text in mormal color}
Begin
   textbackground(0); textcolor(7);
End;

Function Real2Str1(K : real) : temp_s;
VAR
  T : temp_s;
BEGIN                         {converts real no. to string format}
  Str(K:0:1, T);              {using 1 decimal place}
  Real2Str1 := T;
END;

Function Real2Str2(K : real) : temp_s;
VAR
  T : temp_s;
BEGIN
  Str(K:0:2, T);              {using 2 decimal places}
  Real2Str2 := T;
END;

Function Real2Str4(K : real) : temp_s;
VAR
  T : temp_s;
BEGIN
  Str(K:0:4, T);              {using 4 decimal places}
  Real2Str4 := T;
END;

Function Real2Str(K : real) : temp_s;
VAR
  T : temp_s;
BEGIN
  Str(K:0:0, T);              {using no decimal places}
  Real2Str := T;
END;

Procedure DefaultColors;      {set screen color to default}
begin
    SetColor(MaxColor);
end;

Procedure DrawBorder;         {draws border using present co-ords}
VAR
```

```
    Viewport : Viewporttype;
  BEGIN
    DefaultColors;
    Setlinestyle(Solidln,0,Normwidth);
    Getviewsettings(Viewport);
    with Viewport do
      Rectangle(0,0,x2-x1,y2-y1);
  END;

  Procedure DrawThickBorder;          {draws thick line border}
  VAR
    Viewport : Viewporttype;
  BEGIN
    DefaultColors;
    Setlinestyle(Solidln,0,Thickwidth);
    Getviewsettings(Viewport);
    with Viewport do
      Rectangle(0,0,x2-x1,y2-y1);
    Setlinestyle(Solidln,0,Normwidth);
  END;

  Procedure FullPort;                 {sets available display area to max}
  begin
      SetViewPort(0,0,MaxX,MaxY,ClipOn);
  end;

  Procedure MainWindow(Header : temp_s);  {sets up main window for displaying}
  BEGIN                                   {plots/tables..}
    Setcolor(yellow);
    Cleardevice;
    SetTextStyle(Triplexfont,Horizdir,3);
    Settextjustify(Centertext,Toptext);
  {  SetViewPort(0,0,MaxX,MaxY,ClipOn);}
    FullPort;
    OuttextXY(MaxX div 3,2,Header);
    Setviewport(0,Textheight('M')+5,MaxX,MaxY-(Textheight('M')-8),Clipon);
    Drawborder;
    Setviewport(1,Textheight('M')+6,MaxX-1,MaxY-(Textheight('M')+3),Clipon);
    Settextstyle(Defaultfont,Horizdir,1);
    Setcolor(white);
  END;

  Procedure StatusLine(Msg : temp_s);   {sets up status message line}
  BEGIN
    FullPort;
    Settextstyle(Defaultfont,Horizdir,1);
    Settextjustify(Centertext,Toptext);
    Setlinestyle(Solidln,0,Normwidth);
    SetFillStyle(SolidFill,blue);
    Bar(0,MaxY-(Textheight('M')+5),MaxX,MaxY);     {erase old status line}
    Rectangle(0,MaxY-(Textheight('M')+5),MaxX,MaxY);
    Setcolor(white);
    OuttextXY(MaxX div 2,MaxY-(Textheight('M')+2),Msg);
    Setviewport(1,Textheight('M')+5,MaxX-1,MaxY-(Textheight('M')+5),ClipOn);
  END;

  Procedure WaitToGo;      {wait till user presses key to continue}
  BEGIN
    StatusLine('Hit enter to continue . . .');
    Readln;
  END;
```

```
Procedure T_Loss1;       { D = L1 - L2, level diff }
var
   TextCen,height,temp   : integer;  {draws table of transmission}
   c,LineNo,diff,flag    : byte;     {loss data}
   freq : integer;
   DnT : real;
begin
  with StatusRec do begin
   MainWindow('Table of Insulation Test Results');   {draw mainframe}
   Getviewsettings(ViewInfo);
   with ViewInfo do
       Setviewport(x1+155,y1+30,x2-150,y2-8, Clipon);
   Getviewsettings(ViewInfo);
   Setcolor(white);
   with ViewInfo do
   Begin
    Setlinestyle(Solidln,0,Thickwidth);
    Rectangle(0,0,x2-x1,y2-y1);          {draws frame of table}
    Rectangle(0,0,x2,50);
    Setlinestyle(Solidln,0,Normwidth);
    Settextstyle(Smallfont,Horizdir,8);
    SetTextJustify(1,1);
    n := round((x2-x1)/3);
    TextCen := n div 2;
    While n <= x2 do
      begin
        Line(n,0,n,y2);                  {draw vertical lines of table}
        n := n + round((x2-x1)/1);
      end;
    height := TextHeight('E');
    Setcolor(yellow);
    OuttextXY(TextCen,height,'Freq');
    OuttextXY(4*TextCen,height,'Level Diff [D]');
{    OuttextXY(5*TextCen,20,'T[20]');
    OuttextXY(5*TextCen,20,'T[30]');
    OuttextXY(5*TextCen,20,'T[40]');}

    SetViewPort(x1,y1+50,x2,y2,ClipOn);
    Setcolor(white);
    n := (y2-y1) div 22;
    temp := n;
    While n <= y2 do begin     {draw horizontal lines of table}
        Line(0,n,x2,n);
        n := n + temp;
    end;
    n := temp-9;
    o := 1;
    Settextstyle(SmallFont,Horizdir,5);
    Setcolor(yellow);
    repeat
      OuttextXY(TextCen,n,Long2Str(ThirdOctave[o])); {write freq's and data}
      DnT := spl1[o] - spl2[o+20];                    {to screen}
      OuttextXY(4*TextCen,n,Real2Str2(DnT));
(*      OuttextXY(5*TextCen,n,Real2Str2(R_T[o]));   { R.T } *)
      n := n + temp;
      Inc(o);
   until o = 21;        {keep writing till all freq's done}
   End; {ViewInfo}
  end; {StatusRec}
  WaitToGo;
```

```
end;


Procedure RT_Table;    {shows table of RT data}
var
   TextCen,height,temp   : integer;
begin
  with StatusRec do begin
   MainWindow('Table of RT times');     {draws mainframe}
   Getviewsettings(ViewInfo);
   with ViewInfo do
       Setviewport(x1+155,y1+30,x2-150,y2-8, Clipon);
   Getviewsettings(ViewInfo);
   Setcolor(white);
   with ViewInfo do
   Begin
    Setlinestyle(Solidln,0,Thickwidth);
    Rectangle(0,0,x2-x1,y2-y1);           {draws frame of table}
    Rectangle(0,0,x2,50);
    Setlinestyle(Solidln,0,Normwidth);
    Settextstyle(Smallfont,Horizdir,8);
    SetTextJustify(1,1);
    n := round((x2-x1)/3);
    TextCen := n div 2;        {divides table into halves}
    While n <= x2 do
      begin
        Line(n,0,n,y2);              {draws vertical lines of table}
        n := n + round((x2-x1)/3);
      end;
    height := TextHeight('E');
    Setcolor(yellow);
    OuttextXY(TextCen,height,'Freq');         {label table}
    OuttextXY(3*TextCen,height,'E.D.T');
   case rt_l of
        26 : OuttextXY(5*TextCen,20,'T[20]');
        27 : OuttextXY(5*TextCen,20,'T[30]');
        28 : OuttextXY(5*TextCen,20,'T[40]');
    end;
    SetViewPort(x1,y1+50,x2,y2,ClipOn);
    Setcolor(white);
    n := (y2-y1) div 22;
    temp := n;
    While n <= y2 do begin     {draws horizontal lines of table}
        Line(0,n,x2,n);
        n := n + temp;
    end;
    n := temp-9;
    o := 1;
    Settextstyle(SmallFont,Horizdir,5);
    Setcolor(yellow);
    repeat                   {write freq's, EDT, RT data to table}
      OuttextXY(TextCen,n,Long2Str(ThirdOctave[o]));   {frequency}
      OuttextXY(3*TextCen,n,Real2Str2(ED_T[o]));   { EDT }
      OuttextXY(5*TextCen,n,Real2Str2(R_T[o]));   { R.T }
      n := n + temp;
      Inc(o);
    until o = 21;
    End; {ViewInfo}
  end; {StatusRec}
  WaitToGo;
end;
```

```
'ROCEDURE DecayPlot;      {draws reverberation decay plot}
'AR
 MaxdB,Xstep,Ystep,Xscale,Yscale,K,I,TotalTime : real;
 MaxSP,MinSP : real;           {max & min SPL values}
 J,y           : integer;

'rocedure PlotDecayAxis;   {draws graph X,Y axis}
legin
 MaxSP := 0; MinSP := 100;
 MainWindow('Reverberation Decay Plot');
 H := 3*TextHeight('M');
 GetViewSettings(ViewInfo);
 with ViewInfo do
   SetViewPort(x1+50,y1+20,x2-50,y2-20, ClipOff);
 GetViewSettings(ViewInfo);
 with ViewInfo do
 Begin
   Line(H,H,H,(y2-y1)-H);                              {y-axis}
   for n := 1 to 2500 do begin        {find max,min SPL to make plot fit}
       if MaxSP < SPL_Array[n] then MaxSP := SPL_Array[n];
       if MinSP > SPL_Array[n] then MinSP := SPL_Array[n];
   end;
   Yscale := (MaxSP - MinSP)/((y2-y1) - 2*H);     {Y scale ratio}
   Line(H,(y2-y1)-H,(x2-x1)-H,(y2-y1)-H);         {x-axis}
   YStep := (((y2-y1)-2*H) / (MaxSP - MinSP)*5); {Y steps}
   XStep := ((x2-x1)-(2*H)) / 10;                 {X steps}
   J := (y2-y1)-H;
   SetTextJustify(CenterText, CenterText);
   I := MinSP;                            {min SPL value}
 repeat
   Line(H div 2,J,H,J);
   OutTextXY(0,J,Real2Str(I));          {label Y axis}
   I := I + 5;
   J := round(J-Ystep);
 until I > MaxSP;
 SetTextJustify(CenterText, TopText);
 J := H;
 K := 0;
 repeat
    Line(J,(y2-y1)-H,J,(y2-y1-3)-(H div 2));
    OutTextXY(J,(y2-y1)-(H div 2),Real2Str1(K));   {label X axis}
    J := Round(J + Xstep);
    K := K + 0.5;
 until K > TotalTime;
 Setcolor(Yellow);
 Settextstyle(Triplexfont,VertDir,3);
 OuttextXY(-40,70,'Decibels');                    {draw labels}
 Settextstyle(SmallFont,VertDir,6);
 OuttextXY(-36,20,'[dB]');
 Settextstyle(Triplexfont,Horizdir,3);
 OuttextXY(x2-120,y2-50,'Time');
 Settextstyle(SmallFont,Horizdir,6);
 OuttextXY(x2-67,y2-42,'[s]');
 End;
ind;

'rocedure PlotDecay;          {draw the reverb plot}
'ar
  J,K,x,Xave : integer;
  spl : real;
```

```
  PL :array[1..500] of integer;
begin
 for J := 1 to 2500 do                   {scale the SPL data values}
    SPL_Array[J] := (SPL_Array[J] - MinSP) /Yscale-H;
 with ViewInfo do begin
  spl := 0; x := 1;
  Xave := round(2500/ ((x2-x1)-2*H));    {no of averages to be done}
  for J := x to Xave do                  {find ave of about 10 points}
       spl := spl + SPL_Array[J];
  PL[1] := round(spl/Xave);

  MoveTo(1+H,((y2-y1)-2*H)-PL[1]);       {goto 1st point}
  spl := 0; x := 6; y := 5;
  for K := 2 to ((x2-x1)-2*H) do begin
   for J := x to (y+Xave) do             {find ave of about 10 points}
       spl := spl + SPL_Array[J];
   PL[K] := round(spl/Xave);
   LineTo(K+H,((y2-y1)-2*H)-PL[K]);      {draw plot lines}
   x := x + Xave;
   y := y + 5;
   spl := 0;
  end;
 end;
end;

BEGIN
  TotalTime := 5;    {about the maximum reverb plot X axis time span}
  PlotDecayAxis;     {plot axis of graph}
  PlotDecay;         {plot the decay}
  WaitToGo;
END;

Procedure AbsGraph;   {draws graph of absorption vs freq}
Const
  MaxAbs = 1;
  TotalTime = 5.470;
Var
  Xstep,Ystep,I,scaley : real;
  K,J,n,a,b            : integer;
  y,L                  : array[1..20] of integer;
Begin
  Settextstyle(Triplexfont,Horizdir,2);
  MainWindow('Sound Absorption Test on :   ');
  Settextjustify(Lefttext,Toptext);
  OuttextXY(450,-12,'name');              {labels graph with sample name}
  H := 3*TextHeight('M');
  GetViewSettings(ViewInfo);
  With ViewInfo do
    SetViewPort(x1+50,y1+20,x2-50,y2-20, ClipOff);   {set display frame}
  GetViewSettings(ViewInfo);
  With ViewInfo do
  Begin
    scaley := ((y2-y1-H) - (H-2))/ 1;     {find y scaling factor}
    Setlinestyle(Solidln,0,Thickwidth);
    Line(H,H-2,H,(y2-y1)-H);
    Line(H,(y2-y1)-H,(x2-x1)-H,(y2-y1)-H);
    YStep := ((y2-y1)-(2*H)) / 5;         {find Y steps}
    XStep := ((x2-x1)-(2*H)) / Freq1+1;   {find X steps}
    J := (y2-y1)-H;
    SetTextJustify(CenterText, CenterText);
    Setlinestyle(Dottedln,0,Normwidth);
```

```
    I := 0;
  repeat                                {draw y-marks}
    Line((H div 2)+5,J,x2-75,J);
    OutTextXY(0,J,Real2Str2(I));
    I := I + 0.2;
    J := round(J-Ystep);
  until I > MaxAbs;
  SetTextJustify(CenterText, TopText);
  Settextstyle(Smallfont,VertDir,5);
  K := 1;
  L[K] := H; L[1] := 0; y[1] := 0;
  repeat                          {draw x-marks(freq) and decay plot}
     Line(L[K],y1-25,L[K],(y2-y1-6)-(H div 2));
     OutTextXY(L[K],(y2-y1)-(H div 2),Long2Str(ThirdOctave[K-1]));
     y[K] := round((y2-y1-H) - (AbsValue[K-1] * scaley));
     Setlinestyle(Solidln,0,Thickwidth);
     MoveTo(L[K],y[K]);
     If K <> 1 then LineTo(L[K-1],y[K-1]);
     Setlinestyle(Dottedln,0,Normwidth);
     L[K+1] := Round(L[K] + Xstep);
     Inc(K);
  until K = 21;
  Settextstyle(Triplexfont,VertDir,1);      {label graph frame}
  OuttextXY(-40,50,'Absorption Coeff.');
  StatusLine('Press a key for TABLE..');
  Settextstyle(Triplexfont,Horizdir,1);
  Setfillstyle(0,0);
  Bar(MaxX-250,MaxY-30,MaxX-20,MaxY-5);
  Setcolor(15);
  OuttextXY(MaxX-120,MaxY-30,'Frequency in Hz');
  End;
End;

Procedure AbsTable;    {draws table of absorption coeff's vs freq}
Var
  Xstep,Ystep,I,nrc : real;    {nrc = noise reduction coefficient}
  K,J             : integer;
Begin
  MainWindow('Sound Absorption Test on :   ');   {draw frame and label}
  Settextjustify(Lefttext,Toptext);
  name := 'SAMPLE';
  OuttextXY(450,-12,name);
  H := 3*TextHeight('M');
  GetViewSettings(ViewInfo);
  with ViewInfo do
    SetViewPort(x1+100,y1+20,x2-100,y2-100, ClipOff);  {set table area}
    DrawThickBorder;
  GetViewSettings(ViewInfo);
  with ViewInfo do
  Begin
    SetTextJustify(CenterText, CenterText);     {label table divisions}
      Settextstyle(Smallfont,Horizdir,5);
      OuttextXY((x2-100) div 7,10,'Frequency');
      OuttextXY((x2-100) div 7,28,'   (Hz)   ');
      OuttextXY((x2-360),10,'Absorption');
      OuttextXY((x2-360),28,'coefficient');
      OuttextXY((x2-260),10,'Frequency');
      OuttextXY((x2-260),28,'   (Hz)   ');
      OuttextXY((x2-160),10,'Absorption');
      OuttextXY((x2-160),28,'coefficient');
    Setlinestyle(Solidln,0,Thickwidth);
```

```
    Settextstyle(Sansseriffont,Horizdir,2);
    K := 40;
    J := 1;
  repeat                       {draw horiz lines/ labels/ data}
    Line(0,K,x2-100,K);
      K := K + 26;
      OuttextXY((x2-100) div 7,K-15,Long2Str(ThirdOctave[J]));
      OuttextXY((x2-40) div 3,K-15,Real2Str2(AbsValue[J]));
      OuttextXY((x2-260),K-15,Long2Str(ThirdOctave[J+10]));
      OuttextXY((x2-160),K-15,Real2Str2(AbsValue[J+10]));
     Inc(J);
    Setlinestyle(Solidln,0,Normwidth);
  until J = 11;
  SetTextJustify(CenterText, TopText);
  Settextstyle(Smallfont,VertDir,5);
  J := H;
  K := 120;
  repeat                       {draw vertical lines}
    Line(K,0,K,y2-47);
(*    If Freq1 = 18 then*)
      K := K + 105
(*    Else
      K := 391;*)
  until K > 390;
  Line(226,0,226,y2-47);
  nrc := (AbsValue[4] + AbsValue[7] + AbsValue[10] + AbsValue[13])/4;
  Settextstyle(Sansseriffont,Horizdir,2);
  Setcolor(14);
  OuttextXY((x2 div 2)-80,y2-20,'NRC = ');     {show noise reduction coeff}
  OuttextXY((x2 div 2),y2-20,Real2Str2(nrc));
  StatusLine('Press a key..');
  End;
end;

END.   { unit graphics }
```

```
Unit CctCont;            { circuit control }

Interface
type
   VoltArray   = array[1..2500] of word;
   RealArray   = array[1..2500] of real;
   Stat_names  = record
                   noise_type,out_filt,noise_burst,out_chan,start_freq,
                   stop_freq,in_chan,weight,in_filt,no_of_samp,rt_level,
                   range_pos                                : string[20];
                   n_t,o_f,n_b,o_c,start_f,stop_f,i_c,w_t,i_f,
                   n_o_s,rt_l,range_p                       : integer;
                   CFactor                                  : real;
                 end;
var
   Volt_Array  : VoltArray;
   SPL_Array   : RealArray;
   StatusRec   : Stat_names;
   StatusFile  : file of Stat_names;
   volt,dB,ArrayAve     : real;
   Ad_Error             : integer;

Procedure ArrayCon(code1,code2 : string);
Procedure SetInChan(code : byte);
Procedure SetWeightAndFilt;
Procedure UpdateCircuit;
Procedure Ad_Read(samples : integer);
Procedure GenOn;
Procedure GenOff;
Procedure ResetAll;
Procedure SetSampleRate(scode : string);

Implementation
uses TPCrt,Filter;

var
   tempx                 : char;
   NoiseCode,InCode,WeightCode,GenOnCode,GenOffCode : byte;
   numofs                : integer;



Procedure ResetAll;          {reset configuration of BAA circuitry}
begin
    port[$263] := $8a;      {reset 8255 peripheral interface}
    port[$262] := $f;       {set all programming clock lines high}
    port[$260] := $16;      {set data for noise control latches}
    port[$262] := $06;      {reset noise output, white-linear-CH A}
    port[$262] := $f;
    port[$260] := $f6;      {set data for input weight/filtering latches}
    port[$262] := $08;      {reset input weight/filtering - linear input}
    port[$262] := $f;
    port[$260] := $00;      {set data for input channel latches}
    port[$262] := $07;      {reset input channel - CH 1}
    port[$262] := $f;
end;

Procedure SetNoiseFiltChan;  {set noise type,filtering,noise channel}
begin
 with StatusRec do begin          {use status file data}
   case n_t of
```

```
    1 : case o_f of
          3 : begin
                if o_c = 8 then NoiseCode := $16;  {white,linear,ChA}
                if o_c = 9 then NoiseCode := $0E;  {white,linear,ChB}
              end;
        4,5 : begin
                if o_c = 8 then NoiseCode := $14;  {white,filtered,ChA}
                if o_c = 9 then NoiseCode := $0C;  {white,filtered,ChB}
              end;
        end;  {case o_f}
    2 : case o_f of
          3 : begin
                if o_c = 8 then NoiseCode := $17;  {pink,linear,ChA}
                if o_c = 9 then NoiseCode := $0F;  {pink,linear,ChB}
              end;
        4,5 : begin
                if o_c = 8 then NoiseCode := $15;  {pink,filtered,ChA}
                if o_c = 9 then NoiseCode := $0D;  {pink,filtered,ChB}
              end;
        end;  {case o_f}
   end; {case n_t}
    port[$262] := $f;                        {configure noise circuitry}
    if n_b = 7 then
        port[$260] := NoiseCode AND $FB      {continuous noise}
    else
        port[$260] := NoiseCode;        {set data for noise config}
    port[$262] := $06;                  {write data to noise control latch}
    port[$262] := $f;
  end;
end;

Procedure SetInChan(code : byte);   { set input read channel }
begin
    case code of
        10 : InCode := $01;     { CH 1 }
        11 : InCode := $05;     { CH 2 }
        12 : InCode := $09;
        13 : InCode := $0D;
        14 : InCode := $02;
        15 : InCode := $06;     { CH 6 }
    end;
    port[$262] := $f;
    port[$260] := InCode;       {set data for input channel select}
    port[$262] := $07;          {write to input mux latch}
    port[$262] := $f;
end;

Procedure SetWeightAndFilt;     {configure weight/filter circuit}
begin
    with StatusRec do
      case w_t of
        16 : if i_f = 19 then WeightCode := $16  {no weight,no filtering}
                else WeightCode := $0E;          {no weight,filter on}
        17 : if i_f = 19 then WeightCode := $13  {A-weight,no filter}
                else WeightCode := $0B;          {A-weight,filter on}
        18 : if i_f = 19 then WeightCode := $15  {C-weight,no filter}
                else WeightCode := $0D;          {C-weight,filter on}
      end;
    port[$262] := $f;
    port[$260] := WeightCode;     { set up weight & input filtering(on/off) }
    port[$262] := $08;            { write to weight/filter latch }
```

```
    port[$262] := $f;
end;

Procedure UpDateCircuit;            { routines to configure BAA circuit }
begin                               { with status data }
    with StatusRec do begin
       SetNoiseFiltChan;
       SetFilters(o_f,start_f);
       SetInChan(i_c);
       SetWeightAndFilt;
    end;
end;

Procedure GenOn;      {determine code to turn noise gen ON using a mask}
begin
  case NoiseCode of
    $16 : GenOnCode := $16  AND  $FB;     (* $FB is mask to switch noise *)
    $0E : GenOnCode := $0E  AND  $FB;     (* generator on *)
    $14 : GenOnCode := $14  AND  $FB;
    $0C : GenOnCode := $0C  AND  $FB;
    $17 : GenOnCode := $17  AND  $FB;
    $0F : GenOnCode := $0F  AND  $FB;
    $15 : GenOnCode := $15  AND  $FB;
    $0D : GenOnCode := $0D  AND  $FB;
  end;
     port[$262] := $f;
     port[$260] := GenOnCode;       {set data for noise to be on}
     port[$262] := $06;             {write data to noise control latch}
     port[$262] := $f;
end;

Procedure GenOff;      {determine code to turn noise gen OFF using a mask}
begin
  case NoiseCode of
    $16 : GenOffCode := $16  OR  $4;     (* $4 is mask to switch noise *)
    $0E : GenOffCode := $0E  OR  $4;     (* generator OFF *)
    $14 : GenOffCode := $14  OR  $4;
    $0C : GenOffCode := $0C  OR  $4;
    $17 : GenOffCode := $17  OR  $4;
    $0F : GenOffCode := $0F  OR  $4;
    $15 : GenOffCode := $15  OR  $4;
    $0D : GenOffCode := $0D  OR  $4;
  end;
     port[$262] := $f;
     port[$260] := GenOffCode;       {set data for noise to be off }
     port[$262] := $06;              {write data to noise control latch}
     port[$262] := $f;
end;

Procedure SetSampleRate(scode : string);  {set sample rate of A/D}
var                                       { s.r = sample rate }
   dec               : integer;
   msb,lsb           : byte;
begin
(*    dec  := round(1e6/sr);        { separate value into two bytes }
    msb  := trunc(dec/256);
    lsb  := dec - (msb*256); *)
    if scode = 'rt' then begin    {for RT measurements, s.r = 500 Hz, but}
      lsb := $E8;                 {the A/D uses 2 cycles/read so we use}
      msb := $3;                  {2 * s.r = 1000 Hz sampling rate}
    end
```

```
    else begin
      lsb := $A0;               {for SPL measurements, sr = 125 Hz}
      msb := $F;                {therefore we use 250 Hz sampling rate}
    end;
     port[$262] := $f;
     port[$260] := lsb;         {set data for s.r generator clock freq}
     port[$262] := $09;         {write to s.r generator latch 1}
     port[$262] := $f;

     port[$260] := msb;         {set data for s.r generator clock freq}
     port[$262] := $0a;         {write to s.r generator latch 2}
     port[$262] := $f;
end;

Function Log10(x : real) : real;     {calculate the log to base 10}
begin
     Log10 := ln(x)/ln(10);
end;

Procedure ArrayCon(code1,code2 : string);  {converts A/D code to real no.}
var
   a            : integer;
   ave  : real;
begin
 ave := 0;
 with StatusRec do begin
 if code2 = '0' then CFactor := 0;       {no correction factor}

{********************** for transmission loss measurements ***************}

  If code1 = 'tl' then begin
   a := 1;                      { use every tenth sample }
   repeat
     volt := (5.00 * Volt_Array[a]/ 4096.0)*3/5;   {convert code to voltage}
     if volt  = 0 then volt := 3/4096;             {set volt to min}
     dB := 20*(Log10(volt) - Log10(3.0));          {get dB value}
     SPL_Array[a] := (range_p + dB) + CFactor;     {get correct dB value}
     ave := ave + SPL_Array[a];                    {find average of every}
     a := a + 10;                                  {tenth value}
   until a > numofs;   {loop}           {numofs = no. of samples}
   ArrayAve := ave/(numofs/10);         {ArrayAve = ave dB value}
  end    {code = 'tl'}

{******************* for all other measurements **************************}

  Else begin
   for a := 1 to numofs do
   begin
     volt := (5.00 * Volt_Array[a]/ 4096.0)*3/5;  {convert code to voltage}
     if volt  = 0 then volt := 3/4096;
     dB := 20*(Log10(volt) - Log10(3.0));
     SPL_Array[a] := (range_p + dB) + CFactor;
     ave := ave + SPL_Array[a];
   end;   {loop}
   ArrayAve := ave/numofs;
  end    {normal calc}
 end;    {StatusRec}
end;

Procedure Ad_Read(samples : integer);    {read in data from A/D}
begin
```

```
numofs := samples;                            {numofs = no. of data values}
    INLINE (
   $50/                   {  PUSH    AX                                  }
   $53/                   {  PUSH    BX                                  }
   $51/                   {  PUSH    CX                                  }
   $52/                   {  PUSH    DX                                  }
   $FA/                   {  CLI                                         }
   $BB/numofs/            {  MOV     BX,0000                             }
   $8B/$0F/               {  MOV     CX,[BX]                             }
   $BB/Volt_Array/        {  MOV     BX,0000                             }
   $E8/$0E/$00/           {  CALL    011E                                }
   $89/$07/               {  MOV     [BX],AX                             }
   $43/                   {  INC     BX                                  }
   $43/                   {  INC     BX                                  }
   $E2/$F7/               {  LOOP    010D                                }
   $FB/                   {  STI                                         }
   $5A/                   {  POP     DX                                  }
   $59/                   {  POP     CX                                  }
   $5B/                   {  POP     BX                                  }
   $58/                   {  POP     AX                                  }
   $EB/$41/               {  JMP     015E                                }
   $90/                   {  NOP                                         }
   $51/                   {  PUSH    CX                                  }
   $BA/$62/$02/           {  MOV     DX,0262                             }
   $B0/$0F/               {  MOV     AL,0F                               }
   $EE/                   {  OUT     DX,AL                               }
   $EC/                   {  IN      AL,DX                               }
   $24/$30/               {  AND     AL,30                               }
   $3C/$00/               {  CMP     AL,00                               }
   $75/$F9/               {  JNZ     0125                                }
   $EC/                   {  IN      AL,DX                               }
   $24/$30/               {  AND     AL,30                               }
   $3C/$10/               {  CMP     AL,10                               }
   $75/$F9/               {  JNZ     012C                                }
   $4A/                   {  DEC     DX                                  }
   $EC/                   {  IN      AL,DX                               }
   $8A/$C8/               {  MOV     CL,AL                               }
   $42/                   {  INC     DX                                  }
   $EC/                   {  IN      AL,DX                               }
   $24/$10/               {  AND     AL,10                               }
   $3C/$10/               {  CMP     AL,10                               }
   $75/$F9/               {  JNZ     0138                                }
   $B0/$0B/               {  MOV     AL,0B                               }
   $EE/                   {  OUT     DX,AL                               }
   $EC/                   {  IN      AL,DX                               }
   $24/$30/               {  AND     AL,30                               }
   $3C/$10/               {  CMP     AL,10                               }
   $75/$F9/               {  JNZ     0142                                }
   $4A/                   {  DEC     DX                                  }
   $EC/                   {  IN      AL,DX                               }
   $24/$0F/               {  AND     AL,0F                               }
   $8A/$E0/               {  MOV     AH,AL                               }
   $42/                   {  INC     DX                                  }
   $EC/                   {  IN      AL,DX                               }
   $24/$30/               {  AND     AL,30                               }
   $3C/$30/               {  CMP     AL,30                               }
   $75/$F9/               {  JNZ     0150                                }
   $B0/$0F/               {  MOV     AL,0F                               }
   $EE/                   {  OUT     DX,AL                               }
   $8A/$C1/               {  MOV     AL,CL                               }
   $59/                   {  POP     CX                                  }
```

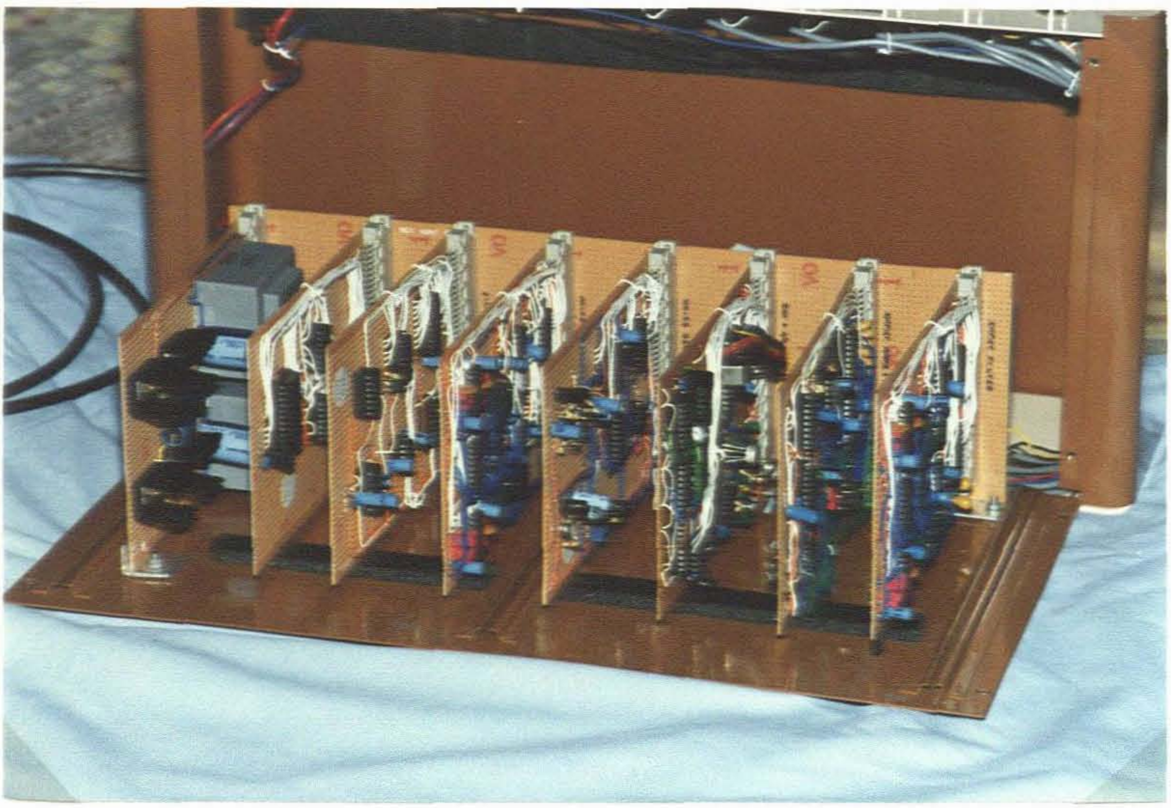
```
      $C3/                    {  RET                                   }
      $90/                    {  NOP                                   }
      $90);

end;
END.     {unit measure}
```

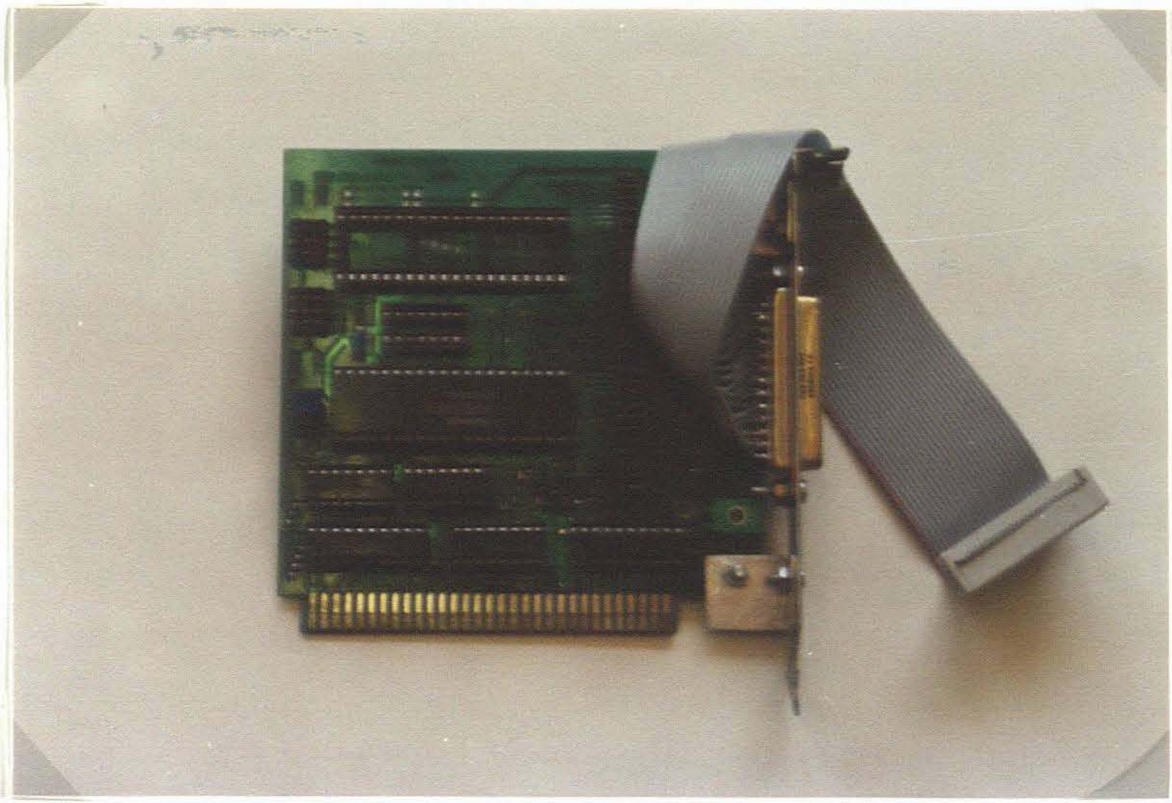
APPENDIX 11

Photographs of B.A.A. instrument

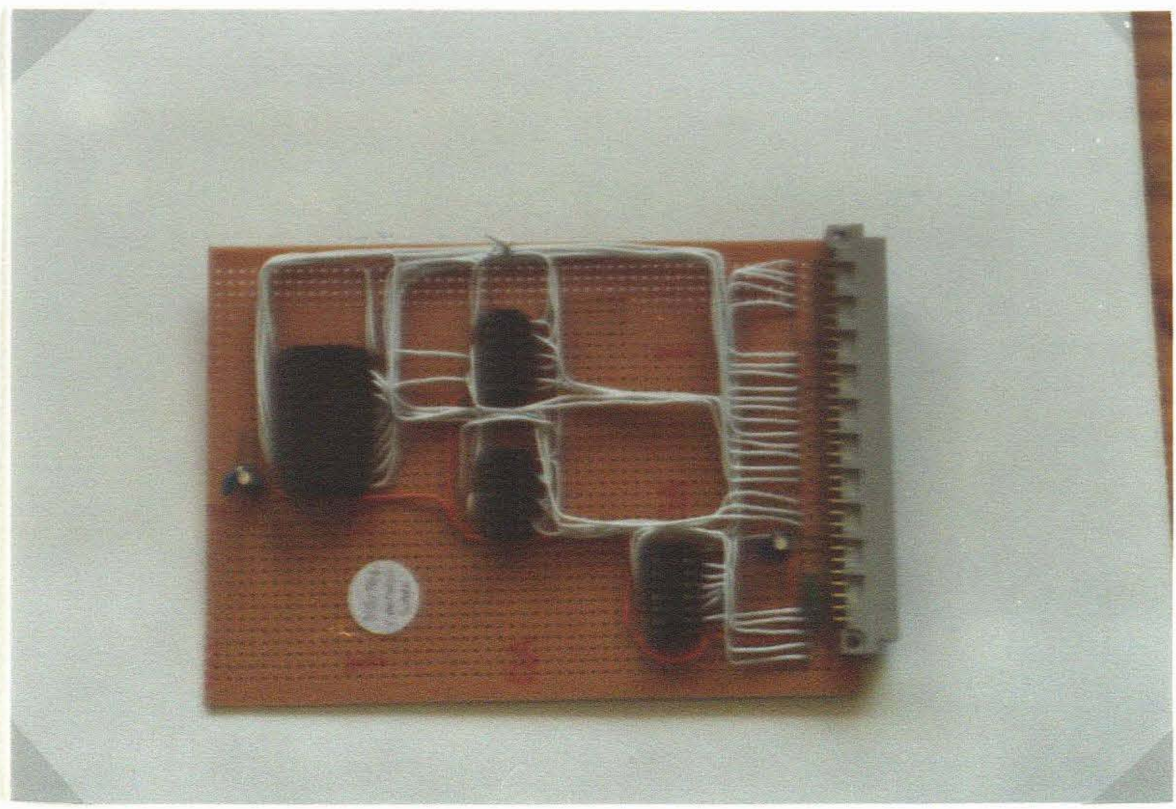
Front panel of B.A.A. instrument



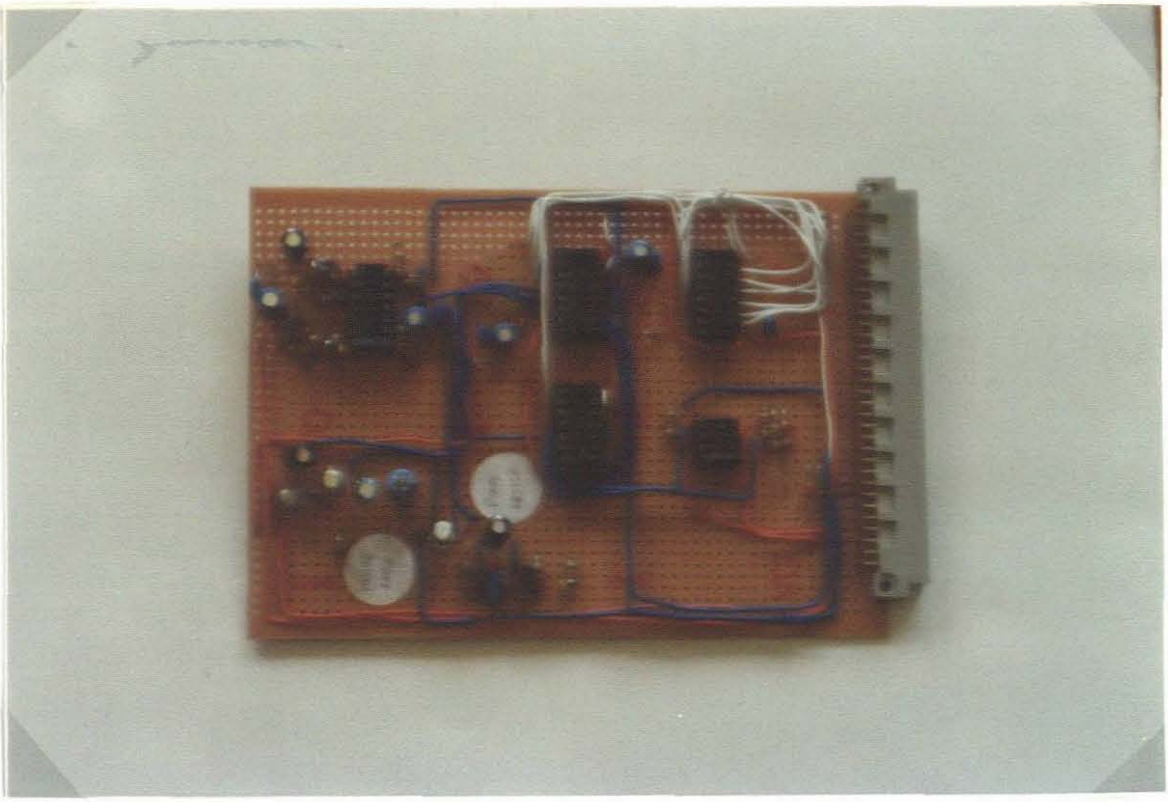
Internal layout of modules



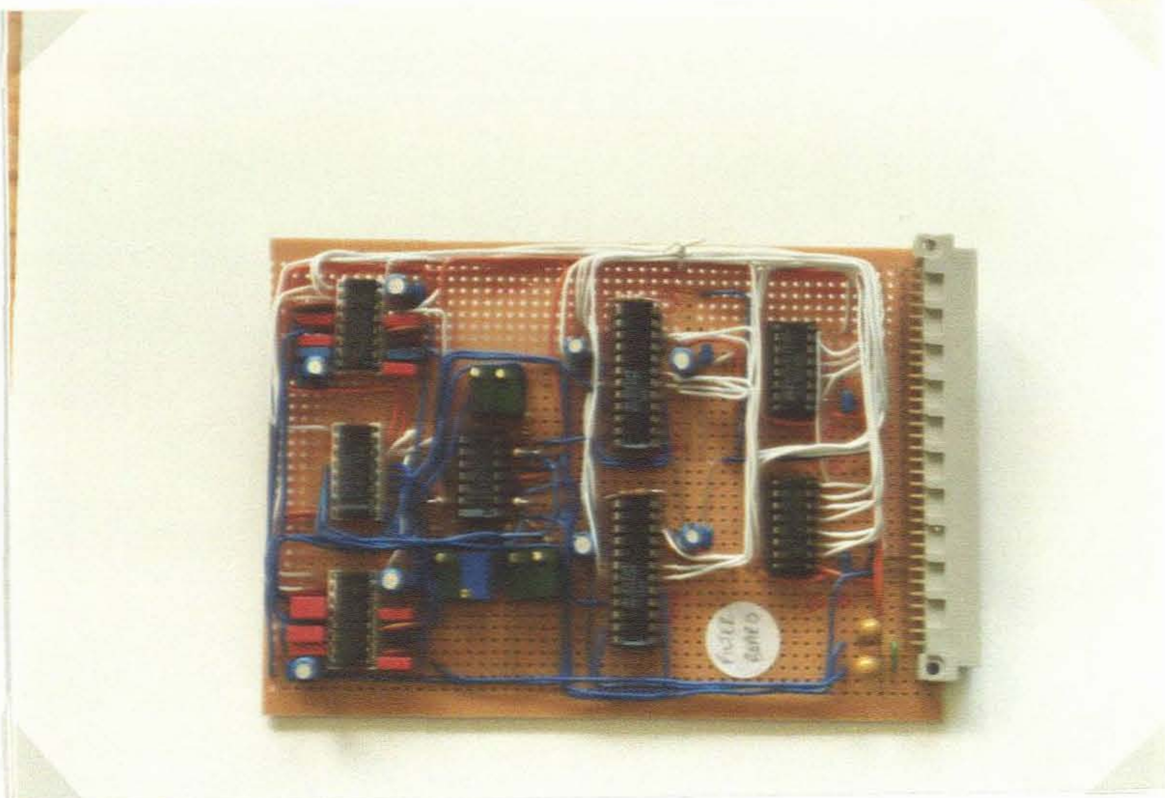
Digital interface card



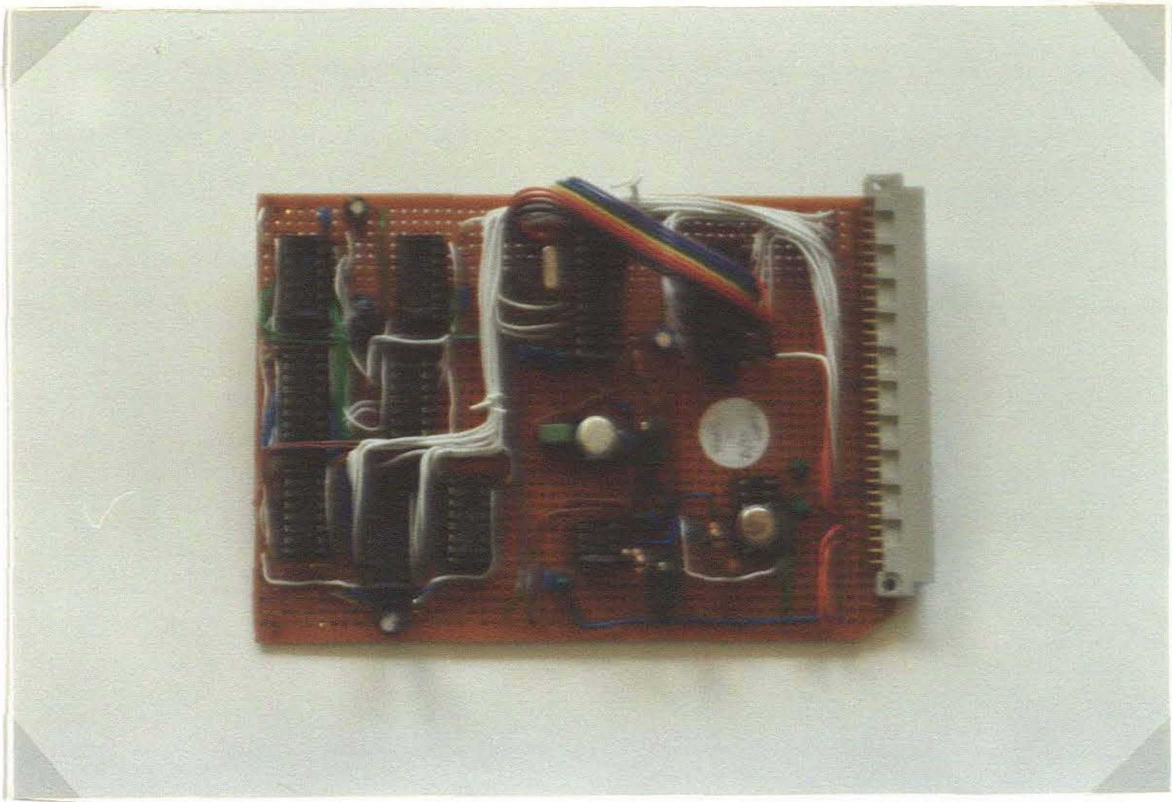
Digital control module



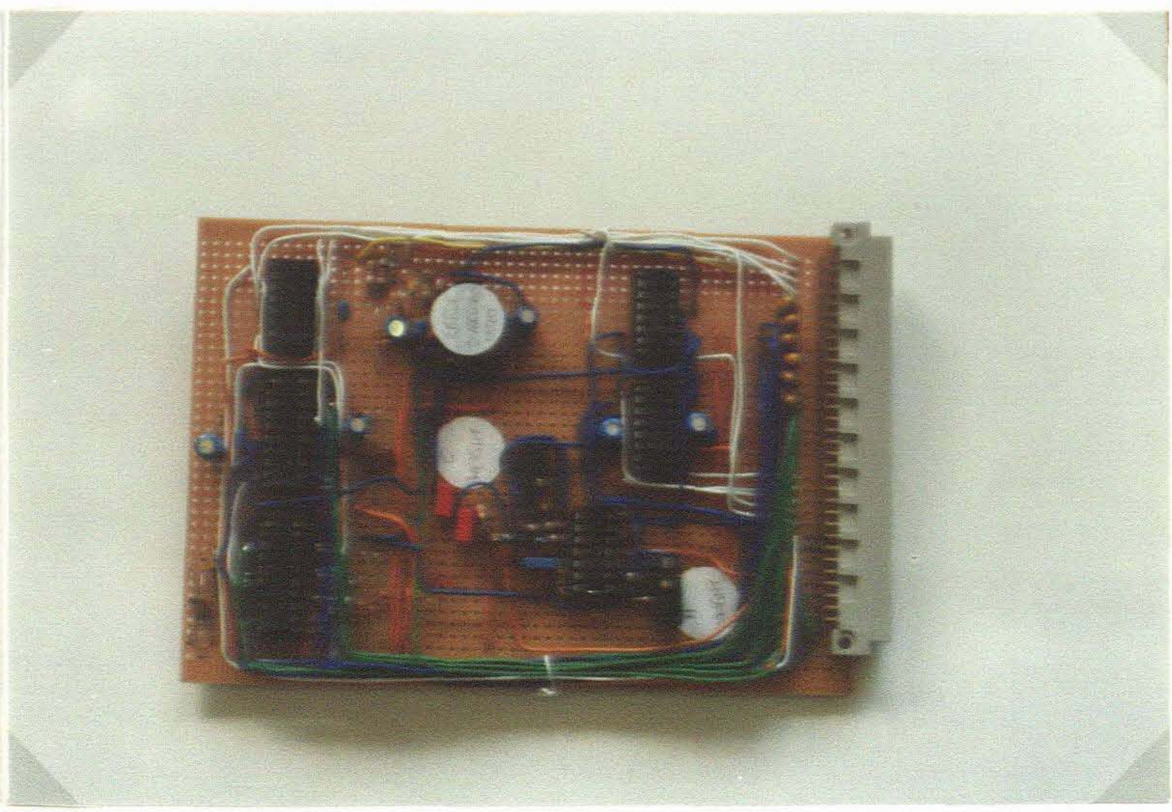
Noise generator module



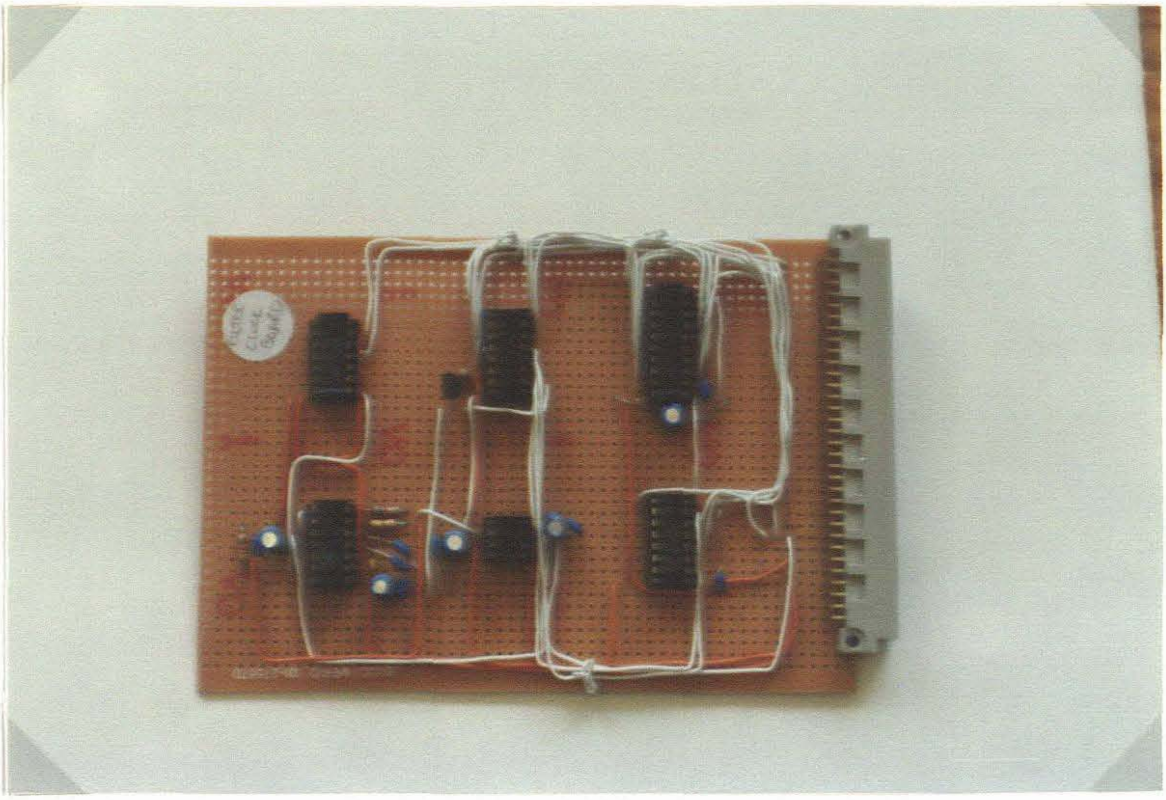
Bandpass filter module



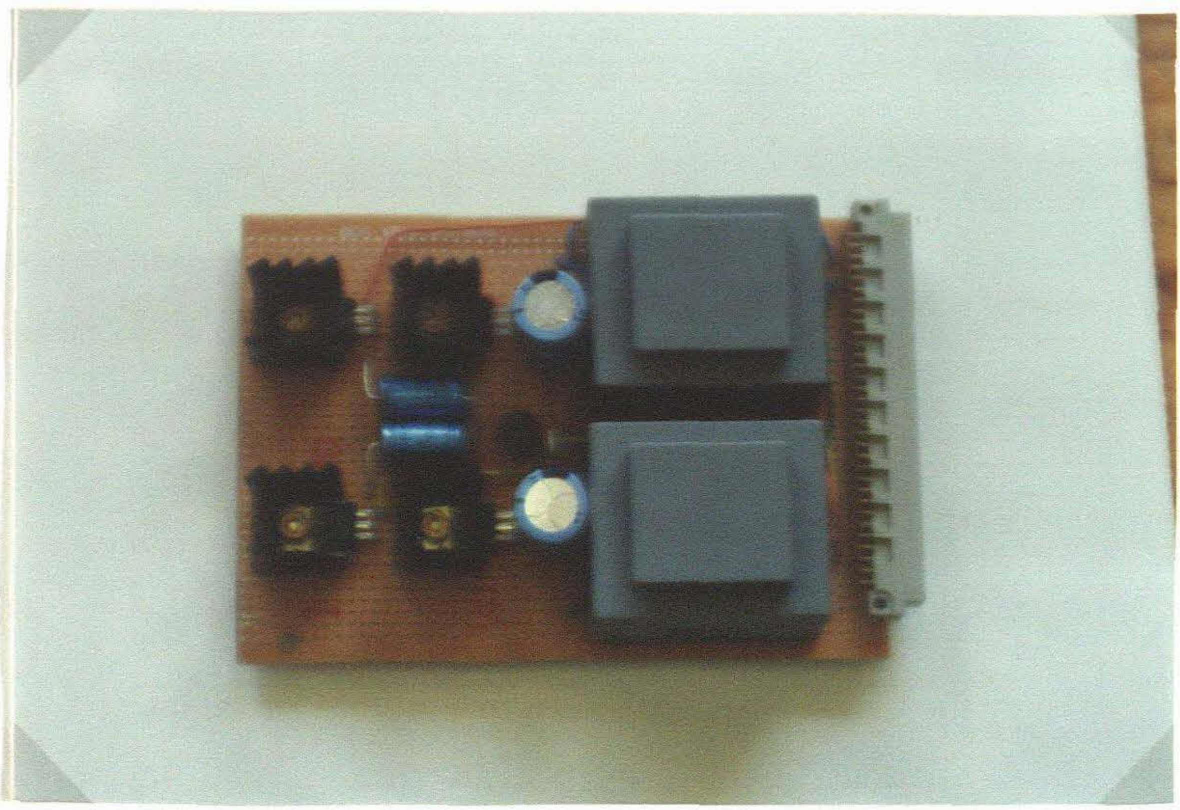
Input amp., overload detect & weighting network module



RMS, A/D and Sample rate generator module



Filter clock generator module



Power supply module