UNIVERSITY OF GLASGOW

DEPARTMENT OF PHYSICS AND ASTRONOMY

Digital Signal Processing Audiometer

MSc Dissertation

David M Canning

June 1998

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

ABSTRACT

A prototype, air conduction, screening audiometer has been developed utilising a 16-bit fixed point digital signal processor. Circuit functions such as sine wave generation, envelope shaping and attenuation are performed in the digital domain. The analog signal, generated by a sixteen-bit digital to analog converter, is passed through a filter and amplifier stage before being connected to the right or left earphone via electronic relays. The instrument is capable of operating in three basic modes. The manual mode allows test tone parameters to be controlled by simple push buttons with an LED display indicating the instrument settings. The automatic mode carries out a Hughson-Westlake type of test under the control of the patient with the test results available as a tabulated printout. The third mode allows remote control of the instrument via an RS232 interface. No manual setting up of circuitry is required and calibration is enabled by a single switch. Calibration offsets and various programmable parameters of the audiometer are stored in a non-volatile memory. Comprehensive measurements were carried out on the prototype audiometer by a specialist audiometer calibration laboratory.

TABLE OF CONTENTS

TABLE OF CONTENTS CONTINUED

TABLE OF CONTENTS CONTINUED

1. INTRODUCTION

1.1 BACKGROUND

The initial development of a computer based audiometer was carried out in the Microprocessor Laboratory of the Physics and Astronomy Department of Glasgow University under the supervision of Dr A M MacLeod (see Bolster 1991). At this time (the mid 1980s) the author was employed by the Royal National Institute for Deaf People (RNID) as an engineer at the Institutes Audiometer Calibration Laboratory in Glasgow. Dr MacLeod approached the RNID with a request to evaluate an initial prototype of the Automatic Self Recording Audiometer (ASRA) with a view to possible commercial manufacture of the instrument. The request was accepted by the RNID and from that point onwards the author was closely involved in testing the performance of the evolving ASRA instrument. When the initial series of ASRA audiometers were commercially manufactured the RNID provided the initial calibration of each instrument sold.

The author's work in the Audiometer Calibration Service and involvement with the ASRA project led to a number of thoughts regarding a possible design for an audiometer utilising current digital signal processing techniques. This culminated in 1992 with the construction of a experimental circuit board utilising a digital signal processing (DSP) chip. This board was used to test the practicality of using the DSP to implement a number of basic audiometer functions. Although at this time there had been a number of developments to the ASRA hardware and software the basic signal generation circuitry currently in use remained the same as the original prototype. The author approached Dr MacLeod with the idea of carrying out a part-time research MSc to develop a DSP based audiometer which could become the basis for a new generation of ASRA audiometers. Dr MacLeod reacted very favourably to the research idea but at this point internal re-organisation within the RNID prevented the MSc from going ahead. During the review of the RNID services carried out at this time it was decided by the Chief Executive that the Audiometer Calibration Service did not fit the planned future role of the RNID in Glasgow. It was therefore decided to transfer the service to an organisation more compatible with the calibration work. A local laboratory, specialising in calibration, was found which met the necessary requirements and in April 1993 the Audiometer Calibration Service transferred to Electrical Mechanical Instrument Services (UK) Ltd (EMIS). The author transferred employment to EMIS and as part of the transfer package the RNID agreed to partly fund the part-time MSc course.

The author started work on the MSc six months after the transfer to EMIS in October 1993 and completed the practical work at the end of 1995.

1.2 AIMS OF PROJECT

The main aims of the project can be summarised as follows:

1. To implement as many audiometer functions through DSP processing as possible and reduce the analog circuitry to a minimum.

2. To produce a high quality signal that easily meets the requirements laid down in BS EN 60645-1, "Specification for audiometers".

3. To produce a "stand-alone" audiometer capable of performing an automatic hearing test.

4. To provide the facility of remote control via a PC.

5. To minimise component count and hence reduce manufacturing cost.

6. To eliminate manual adjustment or setting up of circuitry - again reducing manufacturing costs.

7. To allow easy set-up and calibration of the instrument with the facility of automatic control of these parameters.

1.3 RESOURCES

A number of resources were utilised during the course of the project. As stated in paragraph 1.1, above, the RNID funded the two year course fees. EMIS also provided funding in kind by sanctioning a day per week leave from work for attendance at the University. The author also found it necessary to devote a considerable amount of personal time to the project during the two years of practical work leading to writing up being carried out entirely in the authors own time.

The audiometer circuitry was constructed by the author as the instrument developed. The CAD package "Easy PC Professional" was utilised to draft the detailed circuit diagrams (see Appendix 1) and produce PCB track layouts.

Analog Devices "2100 Family Development Software" was used to develop the DSP code. This package contains various modules such as assembler, linker and simulator. Section 4 provides further information regarding the practical use of this software.

The facilities of the Audiometer Calibration Laboratory were used extensively for measurements during the development of the audiometer both at the RNID and EMIS.

The Windows word processor "Word 6.0" and spreadsheet "Excel 5.0" were used to prepare this document.

The facilities of the microprocessor department (including Technician Alex Blackley) were utilised together with guidance and encouragement from Dr MacLeod.

2. AUDIOMETRY AND AUDIOMETER ARCHITECTURES

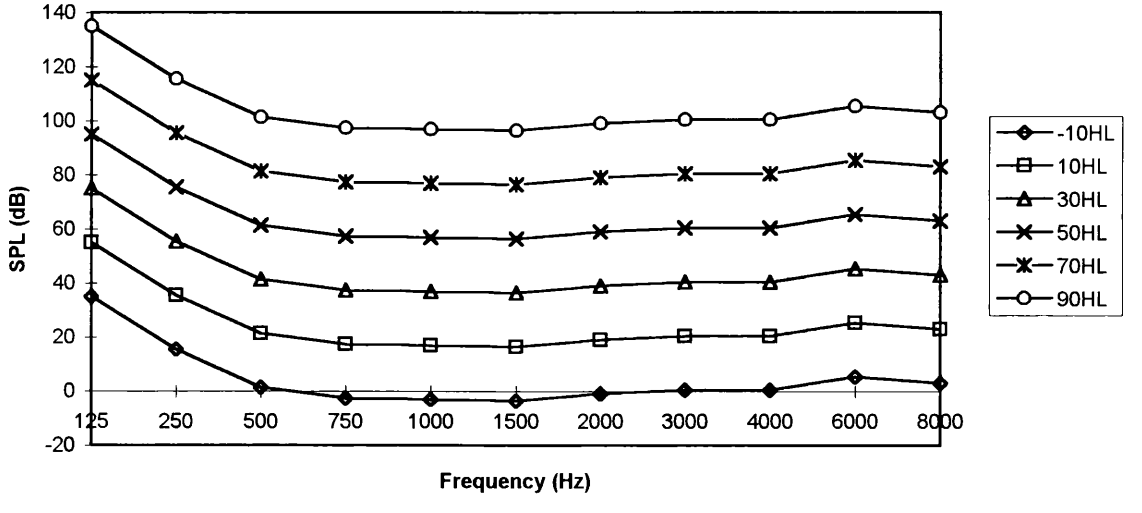
2.1 INTRODUCTION

Audiometry can be defined as the science and practice of hearing evaluation. In the sections which follow some background information is provided about hearing testing and the instrument which is used to carry out the tests - the audiometer. Section 2.2 provides an introduction to audiometry and provides some explanation of the terminology which will be used throughout this document. The remaining sections provide a brief overview of the different types of circuit configurations that have been developed to implement practical audiometers. The final section details the results of a review carried out at the start of the project to determine if an audiometer utilising DSP technology was available on the market.

2.2 AUDIOMETRY

The human ear responds to frequencies from 20 Hz to 20 000 Hz with a dynamic range of over a million to one. As the linear range of sound level to which the ear can respond is so large a logarithmic unit of measurement is used to express sound level measurements - the decibel or dB. 0 dB Sound Pressure Level (or SPL) is defined as 20 x 10^{-6} Pa (20μPa) and the threshold of audibility, at 1000 Hz, for an average person is near to this level. The dB scale also reflects the manner in which a difference in loudness is perceived by the ear; a 10 dB difference in SPL whether from 20 to 30 or 50 to 60 dB would be perceived as a similar difference in loudness. However, the ear does not have a perfectly linear response both in terms of frequency and amplitude. The ear is most sensitive in the region of 500 to 4 000 Hz and at frequencies above and below this region the sensitivity decreases. The perception of loudness also varies with frequency although this effect is only readily apparent at low frequencies where equal changes in loudness are compressed (on the dB scale) when compared to higher frequencies.

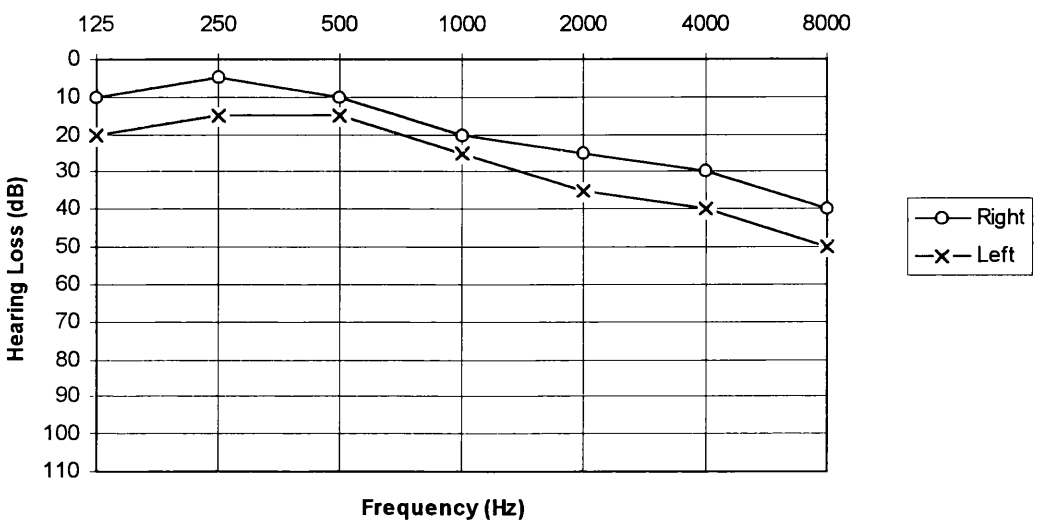
Figure 2.1 - Hearing Level Curves



In audiometry the frequency range of main interest is from 125 to 8 000 Hz (although frequencies above 8 kHz are being increasingly utilised in what is termed High Frequency Audiometry.) as this covers the spectrum of human speech. For the purposes of audiometry a reference threshold of hearing has been defined as the median of a statistically large group of "normal" hearing thresholds. The original threshold determinations were carried out in laboratories throughout the world in the late 1950s using groups of selected, normally hearing, young adults. The resulting standard BS 2497, "Specification for standard reference zero for

the calibration of pure tone air conduction audiometers" contained a reference equivalent SPL for each of the audiometric frequencies. Figure 2.1 plots the reference SPLs at each of the audiometric frequencies in 20 dB intervals. Each curve on this diagram represents what is termed a hearing level and observed hearing measurements are referred to this scale. The basic purpose of an audiometer is therefore to accurately generate the SPLs detailed on this graph at the required frequency and hearing level settings. A sine wave is used as the test signal and the audiometer is set up (or "calibrated") to produce the reference SPL plus the hearing level setting (HLS) at each audiometric frequency.

Figure 2.2 - Sample Audiogram



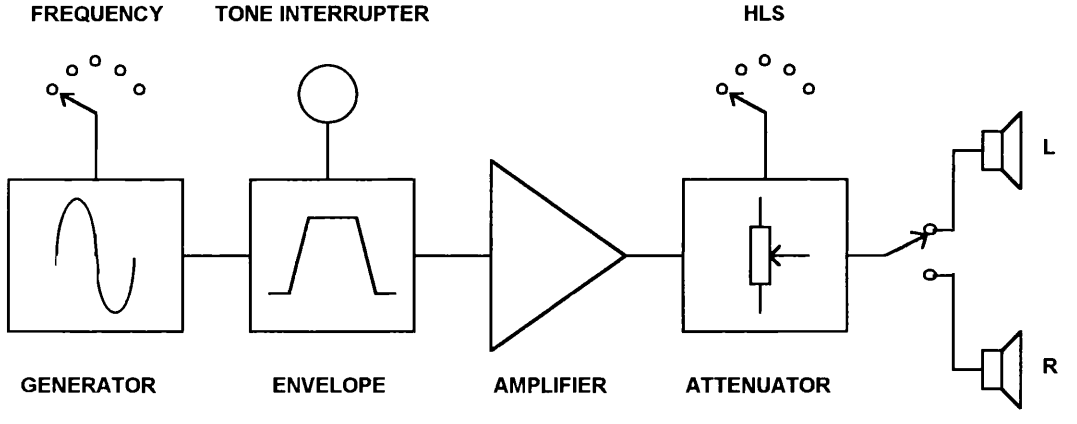
To carry out a hearing test the operator, or "audiologist", utilises a calibrated audiometer and a headset which is worn by the patient. The test tone is usually "presented" to the patient for a period of a half to one second and the patient indicates if the tone has been heard. The response of the patient can simply be the raising of a finger or the depression of a switch ("patient response button") linked to an indicator on the audiometer. The signal level is then increased or decreased until the lowest hearing level at which the patient can detect at least 50% of the test tone presentations. This point, where the test tone can just be heard, is defined as the hearing threshold and the corresponding HLS is recorded on the patient's audiogram. The threshold determination is repeated for each of the remaining test frequencies and the opposite ear until a complete audiogram is obtained. Figure 2.2 is an example of a typical audiogram and it shows the level of hearing loss measured at each test frequency. The vertical axis is scaled in dB

hearing level with the 0 dB point corresponding to the reference threshold of hearing. The right and left ear thresholds are marked with a circle and a cross respectively.

2.3 ANALOG AUDIOMETER ARCHITECTURE

An "analog" audiometer is an instrument in which the circuitry is implemented by analog active devices such as valves, transistors or op-amps. Figure 2.1 outlines such an instrument in block diagram form and shows the four main elements: signal generator, envelope generator, amplifier and attenuator. The signal generator is capable of producing sine waves from 125 Hz to 8 000 Hz. Typically a Wien or state variable oscillator circuit configuration will be utilised with switched resistors to set the frequency of oscillation. The envelope generator switches the sine wave on and off in response to a button press by the operator (this process is termed the "presentation" of the tone to the patient). The tone envelope usually has a rise time of the order of 100ms to ensure that clicks and transients do not interfere with the presentation of the tone. The signal is then passed to an amplifier stage which provides the current gain necessary to

FIGURE 2.1 - "ANALOG" AUDIOMETER ARCHITECTURE



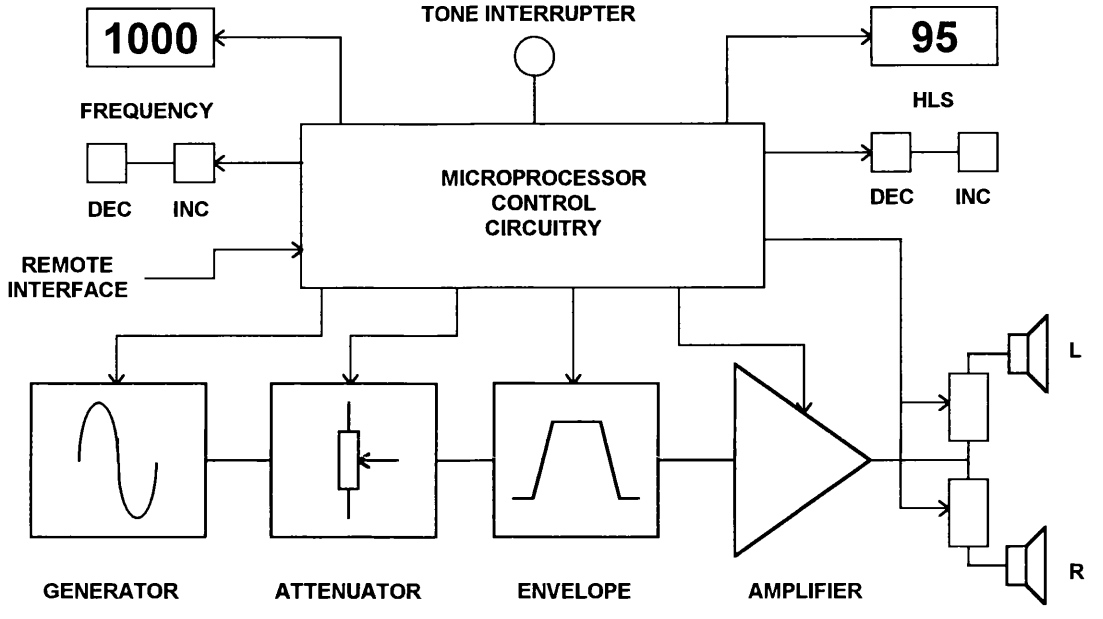
drive the audiometric earphones. The earphones utilised for audiometry have impedances of between 10 and 50 Ω and produce maximum output with input voltages in the range of 2 to 3 V rms. The power output requirement for the amplifier is therefore relatively modest, however it is important that the amplifier has a low noise and distortion performance. The attenuator stage allows the level of the tone to be varied in 5 dB steps over the test range of the instrument. For a clinical instrument this could be up to 140 dB - corresponding to the complete dynamic range of the human ear. Typically the attenuator would be of the constant impedance PI or T network type. The order of these signal blocks is typical for an instrument of this type with a low impedance attenuator which directly matches the headphone impedance. Signal control is carried out by conventional switching arrangements to directly alter circuit

parameters (eg the frequency selector would switch in appropriately valued resistors to alter the frequency of oscillation of the generator circuit). The calibration of an instrument of this type would be carried out by manually adjusting preset potentiometers to set the appropriate parameter.

2.4 MICROPROCESSOR AUDIOMETER ARCHITECTURE

The ready availability of 8 bit microprocessor technology in the late 1970s and 1980s

FIGURE 2.2 - "MICROPROCESSOR" AUDIOMETER ARCHITECTURE

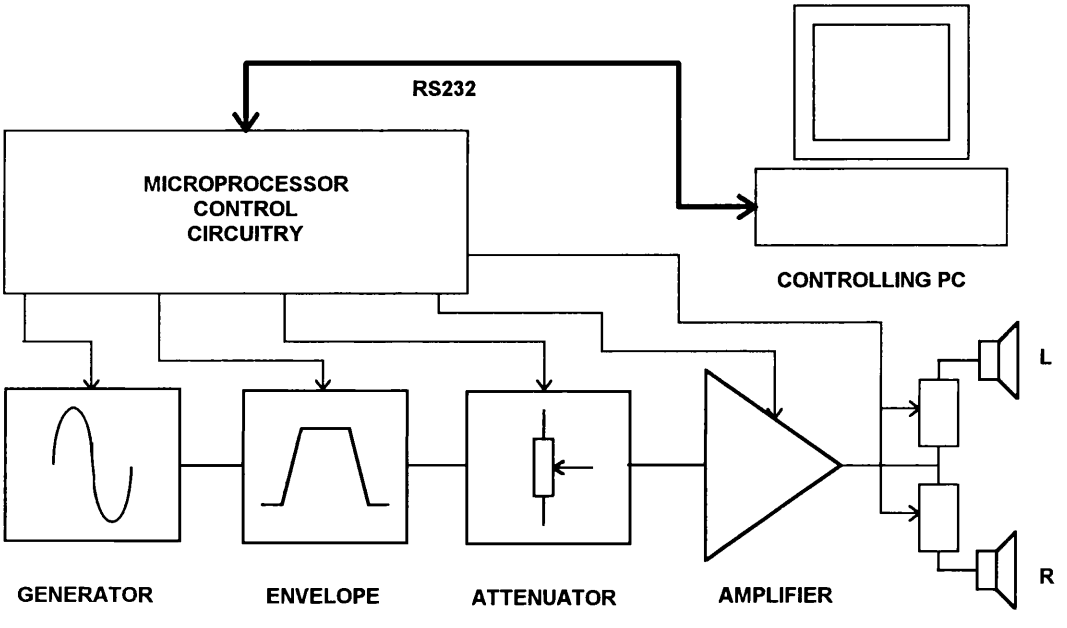


prompted the development of microprocessor controlled instruments. Figure 2.2 is a block diagram showing the typical configuration of this type of audiometer. The same circuit elements as the analog instrument are present and they are implemented using analog techniques. However, in this case the direct control of the circuit parameters is replaced by a microprocessor which interfaces to the various circuits through digitally controlled switches, analog-to-digital converters, digital-to-analog converters etc. The microprocessor also controls the interface to the operator by scanning the audiometer controls and updating the instrument displays (LED, LCD, CRT etc.). The processing capabilities of the microprocessor together with the ability to store data in memory allows this type of instrument to be programmed to perform various audiometric tests automatically. The test results can be stored and then printed or downloaded to a PC via a parallel or serial interface. Most instruments of this type also have the facility of remote control allowing the instrument to be controlled by a PC sending appropriate command strings. Depending on the sophistication of the instrument calibration is either carried out by manual adjustment of presets as in the "Analog" type of audiometer or by storing digital calibration offsets in a non-volatile memory chip.

2.5 ASRA CIRCUIT ARCHITECTURE

Figure 2.3 outlines the circuit configuration of the first generation of ASRA audiometers. A microprocessor (in this case a 68701 or 6801 processor) controls analog signal generation circuitry in a similar manner to the microprocessor audiometer described in section 2.3. However in this case the processor is interfaced directly to a controlling PC which provides the interface to the operator through a comprehensive software program. The audiometer hardware could therefore be described as a "black box" system since it cannot be operated independently of the PC.

FIGURE 2.3 - "ASRA SERIES ONE" AUDIOMETER ARCHITECTURE



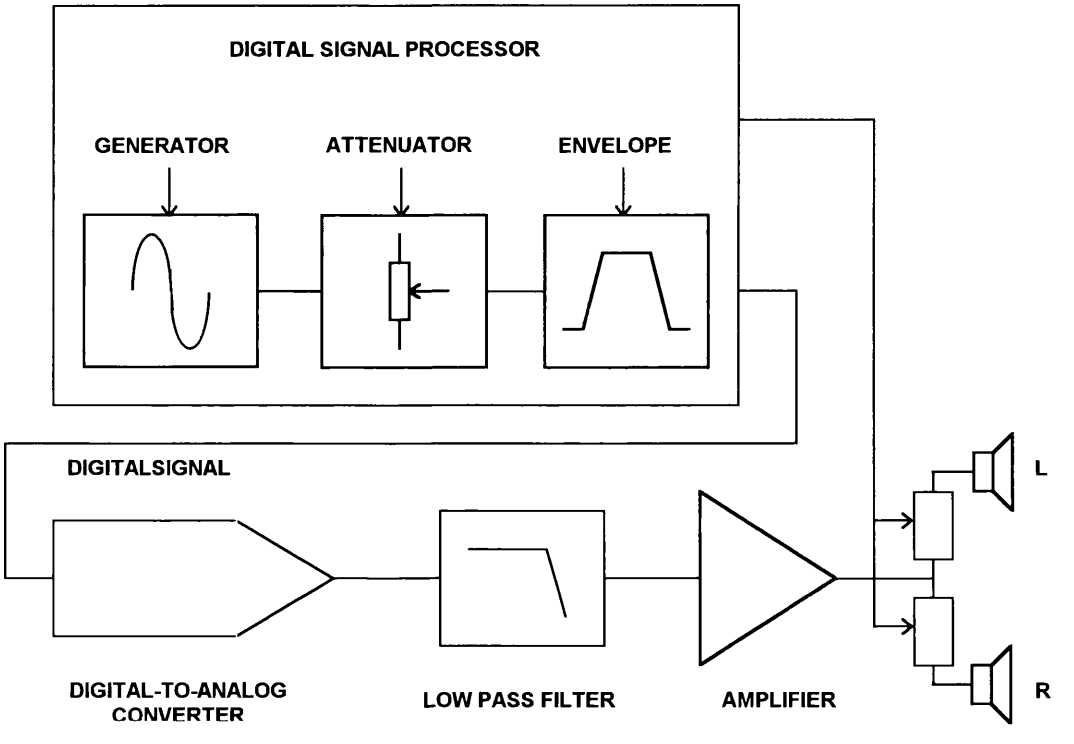
The Series II ASRA audiometers retained the same basic signal generation circuitry as the first generation instruments however the microprocessor was removed from the design. Instead the control signals were derived from the serial and parallel ports of the controlling PC via discrete digital circuitry.

2.6 DIGITAL SIGNAL PROCESSING

In the early 1980s the first digital signal processing (DSP) chips began to appear on the market. They were based on a 16 bit microprocessor architecture with an instruction set optimised for high speed implementation of digital processing algorithms such as digital filtering, Fast Fourier Transforms etc. Since then the use of DSPs has increased rapidly (see Higgins 1990) and digital processing has replaced analog circuitry in a wide variety of applications such as filtering, telephony, electronic musical instruments, acoustic processing, sound recording etc.

Figure 2.4 shows how a DSP can be utilised to implement an audiometer. In this case (and in contrast to the "analog" and "microprocessor" audiometers detailed above) three of the circuit blocks are implemented in the digital domain using algorithms programmed for the DSP chip. These algorithms perform the sine wave generation, envelope formation and attenuation which

FIGURE 2.4 - "DSP" AUDIOMETER ARCHITECTURE



are performed by discrete circuitry in the other designs. A digital-to-analog converter produces the analog signal which is then fed to an amplifier stage.

The use of the DSP has a number of significant advantages over the conventional analog circuit techniques detailed in sections 2.2 to 2.4 above. The use of 16 bit processing enables the generation of accurate signals (both in terms of frequency and amplitude) with low levels of total harmonic distortion and noise. Digital algorithms are used for signal generation which gives great flexibility to the type of signals that can be generated and allows signal parameters to be easily altered in software without requiring alterations to hardware. The digital implementation of circuit functions reduces the amount of analog circuitry required and this leads to a more compact design with less possibility of grounding problems, noise pickup etc. In a similar way to a microprocessor the DSP can be configured with a variety of different interfaces - serial, parallel, displays, switches, controls etc. The processing power of the DSP also allows the implementation of automatic tests in a similar manner to the microprocessor based audiometer however, as the instruction set of the DSP is optimised for signal processing its data handling capabilities are limited.

2.7 MARKET REVIEW

At the start of the project a review was carried out of the air conduction audiometers available on the market and the circuit configurations employed by different manufacturers. The purpose of this review was to ascertain if DSP technology was already in use by the audiometer manufacturers.

Screening audiometers can be broken down into two basic categories. The first consists of a manual, readily portable instrument designed for community and educational screening use. The second category is an automatic instrument designed for hearing assessment in industry (or similar areas) where an occupational health nurse with minimal audiological training would typically carry out the tests. The circuit configuration of the available manual instruments was found to follow the analog architecture outlined in section 2.2 almost without exception, while the industrial instruments were either microprocessor or computer based as described in sections 2.3 and 2.4.

The Danish company Interacoustics is one of the main suppliers of audiometric equipment in the world. Its instruments are marketed under the Kamplex name in the UK. The Kamplex AS7 is a manual screening instrument. Its small size and battery power facility make it readily portable. The front panel consists of a rotary frequency and hearing level controls, a right/left earphone selection switch, a +20 dB switch and a tone interrupter touch button. The circuitry follows the outline of section 2.2. A FET stabilised Wien oscillator is utilised as the signal source with the frequency switch connecting pairs of equal value resistors to set the operating frequencies. The tone interrupter is implemented by a voltage controlled amplifier chip which is driven by a touch circuit built from CMOS NAND gates. Calibration is provided by a series of potentiometers before the signal is applied to a four transistor, push-pull amplifier stage. The amplifier feeds a resistive ladder attenuator which is tapped by the hearing level switch at the appropriate attenuation setting. Finally the signal is routed via a switch to either the right or left earphone. The circuit also incorporates an auto power off facility (comprising one CMOS IC and two transistors) which disconnects the battery supply if the tone interrupter button has not been pressed for five minutes or so. The Interacoustics BA20 is an automated screening instrument with a number of features designed for hearing conservation schemes. The instrument functions are controlled from a number of front panel push buttons and an external QWERTY keyboard can be attached to facilitate data input. A four line LCD display indicates the status of the instrument and an in-built printer allows test results to be printed out. Tests can be performed manually or automatically and an internal database allows test results to be stored in a non-volatile memory. The BA20 circuitry is microprocessor based and follows the

configuration outlined in section 2.3. A state variable oscillator circuit is utilised with the oscillation frequency voltage controlled via digital to analog converters driven by the digital circuit. In a similar manner the attenuator is implemented by a voltage controlled gain module which allows calibration offsets to be used (these are stored in non-volatile memory) rather than adjustment presets. Tone presentation is implemented by opto-coupled resistive devices and two amplifiers feed the right and left earphones. Switching between headphone outputs is implemented by CMOS analog switches at the amplifier imputs.

Grason Stadler is the leading audiometric equipment manufacturer in the USA. The GSI17 is a manual screening audiometer with almost identical facilities to the AS7. However, rather than purely analog signal generation circuitry the audiometer incorporates a micro-controller in a circuit configuration of the type detailed in section 2.3. The GSI17 utilises a timer/counter chip to generate a square wave at the required audiometric frequency. A switched capacitor low-pass filter tracks the fundamental frequency of the square wave but attenuates the upper harmonics to produce a sine wave. The remaining circuitry is similar to the BA20 in that a VCA is used to implement the attenuator (controlled by an 8-bit DAC) and separate power amps drive the right and left earphones. Calibration corrections are stored in a non-volatile memory which is integrated into the micro-controller itself. The GSI17 can be converted to an automatic industrial audiometer by the addition of a remote PC interface. This allows the GSI17 to be controlled by a PC running audiometry software in a similar manner to the ASRA audiometer described in section 2.5. The GSI software implements a comprehensive series of automatic tests, audiogram analysis, record handling and printing facilities.

Madsen Electronics are another Danish audiometer manufacturer. Their MM304 screening audiometer is a battery-powered portable instrument with a similar specification to the AS7. However the instrument uses an array of LEDs in the form of an audiogram chart to indicate the frequency and attenuator settings. Again signal generation is carried out by analog circuitry and CMOS digital ICs are used to implement the control circuitry. In this case the attenuator is implemented by a resistor chain with CMOS analog switches selecting the appropriate tap. Calibration is controlled by presets which are selected by a CMOS analog multiplexer. The MTA86 is a micro-processor based industrial screening audiometer with similar features to the BA20. Its facilities however, are less comprehensive (no audiogram storage or QWERTY keyboard for example) and it does not have the facility of a fixed frequency Bekesy test. The signal generation circuitry is almost identical to the MM304 but with the addition of data latches which control the CMOS analog switches. Again presets are used for calibration adjustment.

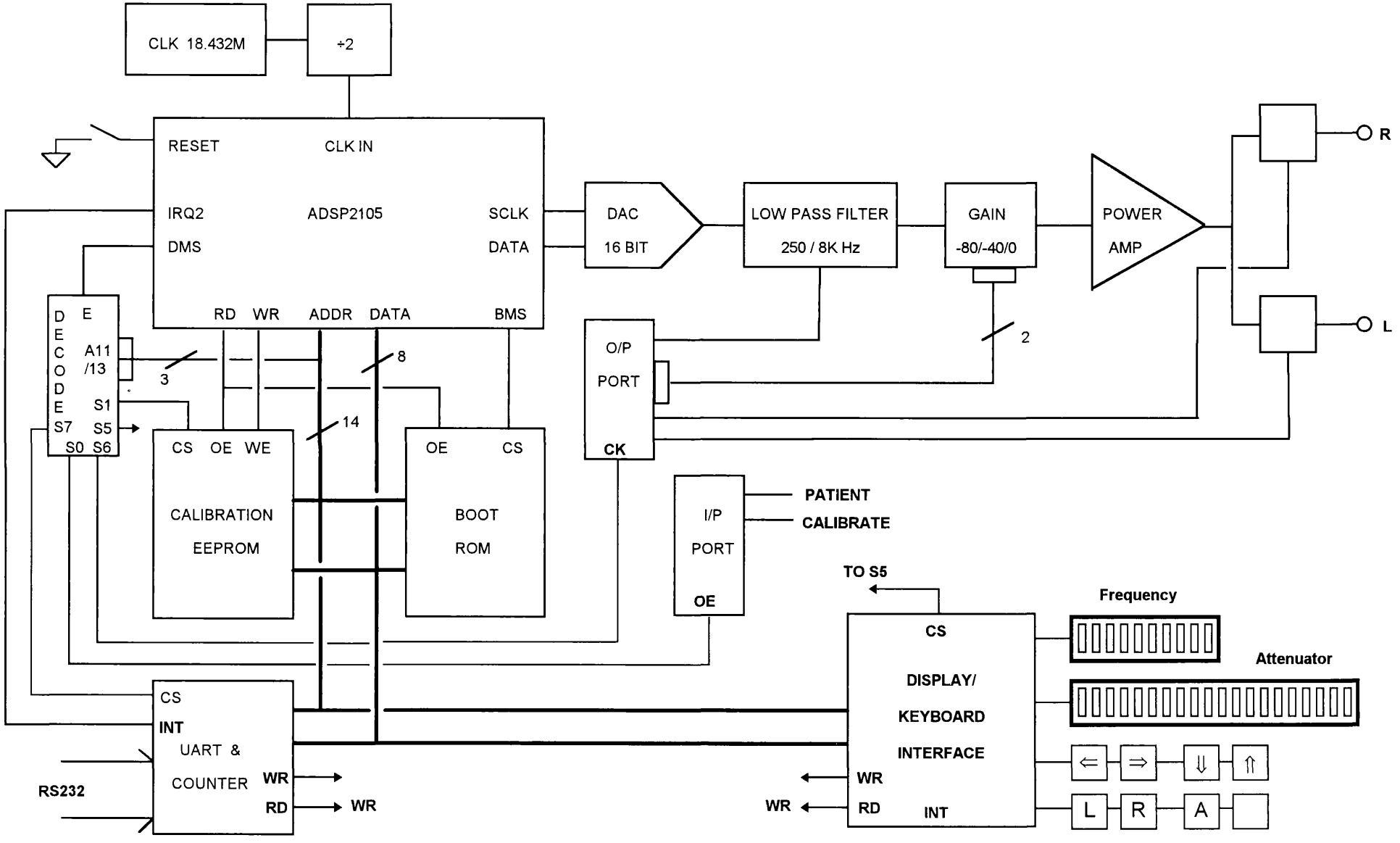
The British manufacturer Amplivox has a range of audiometric instruments which includes the 2100 and 50 model manual screening audiometers. The 2100 is almost identical to the AS7 in facilities and circuit configuration. The functions of tone interrupter and earphone selection are combined in right and left tone interrupter buttons. A simple toggle circuit drives two reed relays which connect the appropriate earphone. The model 50 is similar to the Madsen MM304 with two lines of LEDs indicating HL and frequency in the form of an audiogram. Again the circuitry is almost identical with CMOS digital control and analog signal generation. Both the 2100 and 50 make use of presets for calibration adjustment. The Amplivox CA850 is an industrial screening audiometer with a similar set of facilities to the Interacoustics BA20. Extensive use is made of CMOS analog switches to control the oscillator circuit and the output attenuator (which is composed of a resistive ladder network) and a voltage controlled amplifier implements the tone interrupter circuit.

The USA company Micro Audiometrics specialises in microprocessor-based, screening audiometers. The Earscan instrument is a compact unit with a front panel consisting of an LCD display and a sixteen button keypad. The unit can perform manual or automatic tests and results can be viewed on the four line display. An external printer or computer can be attached to a serial port to allow tests to be printed out or stored to disk. Again use is made of analog switches to control the signal generation circuits however in this case a multiplying DAC is used to implement part of the attenuator. The pure tone signal is applied to the input of the DACs R-2R resistor chain and the attenuated signal is taken from the output. By applying an 8-bit code to the digital input of the DAC the attenuation of the resistor chain can be altered to vary the amplitude of the output signal.

A number of other companies were found to produce instruments which fell into the categories detailed above. The Italian company Amplaid produces the 99 and 121 manual screening audiometers which are similar to the AS7 and MM304 instruments. The British company Peters has the manual AP26 and AP27 audiometers which are virtually identical to the AS7 in terms of facilities and circuitry. Peters also briefly produced a computer based instrument called the AP250. This was similar to the ASRA system but the audiometer hardware made use of the circuitry from an old stand-alone industrial audiometer (the AP25) with a digital control interface. The USA companies Maico and Qualitone produce AS7 type instruments the MA25 and AS90/AS110 respectively.

In the review of audiometers detailed above it is apparent that all of the instruments make use of analog, digital and microprocessor technology. At the time of the review the author could not locate any evidence of an audiometer which made use of DSP technology.

FIGURE 3.1 - HARDWARE BLOCK DIAGRAM



3. HARDWARE

3.1 INTRODUCTION

The block diagram outlined in Figure 3.1 provides an overview of the circuitry that has been developed for the audiometer. Section 3.2 provides an overall description of the circuit, while the remaining sections provide detailed descriptions of the four circuit diagrams contained in Appendices 1A to 1D.

The practical configuration of the audiometer circuits follows the documentation with four main circuit boards: the DSP and digital circuit, the analog circuit, the display and keyboard interface and the power supply and isolated RS232 circuit. All of the circuitry was laid out and built by the author during the course of the project. The physical layout of the circuitry is most important in the analog circuit where very low AC signals are present. Great care was taken in this part of the circuit with the routing of ground connections and signal lines to minimize the possibility of noise pickup from the digital circuit.

3.2 BLOCK DIAGRAM

The heart of the digital circuit is the Analog Devices ADSP-2105 digital signal processing chip. This is a second generation DSP with a 16 bit architecture. It has 1k words of program memory and 512 words of data memory on-chip. It has the facility to automatically load (boot) the internal program memory from a standard 8 bit external EPROM at reset. External interfacing is carried out via a 14 line address bus and a 24 line data bus. Standard read (RD) and write (WR) lines are utilised together with three memory select lines: program memory select (PMS), data memory select (DMS) and boot memory select (BMS). In addition the chip has a serial port, a programmable timer and three computational units (arithmetic/logic, multiplier and shifter).

Referring to the block diagram, the ADSP-2105 is interfaced to two memory chips. The boot ROM contains the program code which is loaded into the internal program memory at power on or reset. The calibration EEPROM is an electrically erasable EPROM which is used to store calibration offsets together with a number of data tables and constants which are used by the DSP to define various signal parameters. The most significant three address lines (A11-A13) are decoded together with the DMS line to divide the data memory address space into eight 2k sections. One of theses lines (S1) is used to select the calibration EEPROM.

Two lines from the address decoder, S0 and S6, are used to select memory-mapped ports. The input port interfaces the DSP to the patient response button and a switch which selects the calibration mode of the instrument. Eight lines from the output port are used to provide control signals for the analog circuitry.

Line S7 is used to select the UART/timer chip. This IC provides two main functions; a four-wire RS232 interface (allowing hardware handshaking) with software programmable baud rate and a programmable counter/timer module. The counter is configured to interrupt the DSP at 1ms intervals to provide a timing signal for patient response times, tone lengths etc. The facility to optically isolate the RS232 port has been built into the design to allow the instrument to be electrically isolated when connected to a standard PC.

Line S5 selects the keyboard and display interface. This circuit drives two bargraph modules which indicate frequency and attenuator values. It also scans a number of push switches which control frequency increase and decrease, tone presentation etc. The interrupt output of the chip is not utilised. Instead a software poll is carried out by the DSP to determine if a key has been pressed.

The clock signal for the DSP is derived from a standard crystal oscillator module with a frequency of 18.432 MHz. This is divided by two to give a master clock of 9.216 MHz for the DSP. This frequency was chosen as a direct multiple of the 48 kHz sampling rate utilised to generate the sine wave signals. It is also within the maximum operating speed (10 MHz) of the ADSP-2105.

The analog signal is generated by a sixteen bit digital-to-analog converter (DAC). A serial chip is used which interfaces directly to the serial port of the DSP. The chip used is an industry standard part which is commonly used in domestic digital audio equipment such as compact disc players.

The signal from the DAC is fed to a low pass filter. This is a simple 12dB/octave filter to remove the sampling harmonics from the DAC waveform. The Nyquist bandwidth of the DAC output is theoretically 24 kHz with a 48 kHz sampling rate. However, as the highest frequency of interest is 8 kHz (possibly 12 kHz) the filter cut-off frequency is set to approximately 10 kHz. This provides enough attenuation at, and above, the Nyquist frequency to reduce the sampling harmonics to a negligible level. The filter cut-off frequency can be switched to 250 Hz to provide extra attenuation of harmonics at signal frequencies of 125 and 250 Hz.

The next stage of the circuit is a combined gain and current amplifier stage. A considerable amount of time was spent developing this section of the circuit as it is crucial to the performance of the instrument. An audiometer requires a dynamic range of at least 140dB. This range cannot be achieved without some form of gain switching arrangement - the ASRA Series 1 (& 2), for example, splits this into two ranges. The sixteen bit DAC has a theoretical dynamic range of 96dB however, as the signal level produced by the DAC is decreased the total harmonic distortion (THD) of the generated sine wave increases. The performance of the DAC in this respect was found to be very close to its published specification - at a level of 40dB down from full scale the THD was found to have risen to about 0.1%. It was therefore decided to split the range into three sections. The first two sections each consist of a 40dB attenuation range while the lowest range extends to 60dB attenuation. The extra 20dB attenuation can be achieved at low levels because the harmonics are well below the threshold of hearing at these low sound pressure levels.

The amplifier stage is necessary to provide current amplification to drive the low impedance (10Ω) audiometric earphones. Low levels of noise and THD are also important requirements for this stage. A number of commercially available amplifier chips were investigated for this part of the circuit. However, these chips were found to be too noisy and, in some cases,

accurate gain setting was difficult to achieve. A discrete approach was then investigated with a low noise op-amp driving a push-pull transistor output stage. This was found to be capable of a very low noise performance with low distortion characteristics. The gain could also be precisely controlled via the op-amp feedback path.

The final stage of the analog circuit switches the signal to the right or left earphone and this is achieved via photoMOS relays. These devices are opto-coupled MOSFET switches with low on resistance. Opto-isolation between the control input and the signal path ensures that switching transients do not appear at the outputs of the switches. This "silent" switching capability allows the appropriate earphone to be coupled to the amplifier stage only during the presentation of a tone. At all other times both photoMOS relays are off, therefore eliminating the possibility of any extraneous signals entering the earphones.

The power supply unit uses an inverter module to generate a ±9V DC supply from a single 5V DC input. Standard regulators are used to provide a ±6V supply for the analog circuitry and +5V for the digital circuitry. The inverter provides electrical isolation of the 5V input - this allows the 5V supply of a PC to be utilised while avoiding the problems of earth loops and noisy supply lines.

3.3 DIGITAL CIRCUIT DIAGRAM

Appendix 1A contains the detailed schematic drawing for the digital signal processor and its associated circuitry. The ADSP2105 DSP (U1) lies at the heart of the circuit. Its address and data lines are connected to the BOOT ROM Program Memory (U2) and the System Data EEPROM (U3). U2 is a 27C256 EPROM providing 32K bytes of storage space. As each boot page occupies 4K this allows a maximum of 8 boot pages to be stored in this chip. In order to ensure backwards compatibility with the ADSP2101 Analog Devices DSP the upper address lines A12 - A14 of U2 are connected to A13, D22 and D23 of U1. (The ADSP2101 has 8K boot pages and makes use of A12 to address the larger memory space.) The data lines of U2 and U3 are connected to the middle byte of the ADSP2105 data bus. For most 16-bit operations of the DSP this is the least-significant byte of a sixteen bit word. The chip enable (CE) for U2 is connected to the boot memory select line (BMS) of the DSP. This line is only enabled during the boot cycle of the DSP. The CE of U3 is connected to the Data Memory address decoder (U8). The write enable (WE) pin of U3 is connected to the DSP write pin via an OR gate, U10b. This gate enables the write line only when the calibration switch (SW1) is activated. This ensures that the data in U3 can only be altered when the instrument is in calibration mode and minimises the possibility of erroneous writes to the chip when the instrument is in normal mode.

The clock signal for the DSP is derived from a crystal oscillator module, XTAL1. The 18.432 MHz clock from this module is applied to a D-type flip-flop (U4b) configured as a divide-by-two counter. A 74ACT family chip is used for this divider to provide a low impedance drive for the DSP clock input at a frequency of 9.216 MHz. This frequency was chosen as an integer multiple of the 48 Khz sampling frequency utilised by the digital-to-analog converter, while keeping within the 10 MHz maximum operating frequency of the ADSP2105.

A supply voltage supervisor (U11) is utilised to provide the reset signal for the DSP and the UART (U6). This IC provides both active low and active high reset signals. The reset pulse width is determined by capacitor C2 and in this case a value of 100nF provides a 1.3 mS pulse time. U11 monitors the supply voltage during power-up and will only supply the reset pulse when the supply has reached its nominal voltage - in this case a minimum of 4.5V. If at any time the supply drops below the threshold point the reset outputs are activated until the supply is restored. This IC therefore minimises the possibility of a power supply transient causing the DSP to enter an unknown state. This could, in turn, lead to a dangerously high SPL being presented to the patient. A push button is also interfaced to U11 to provide the facility of a manual reset.

The ADSP2105 has a 16K byte Data Memory address space. This space is split into eight 2K pages by address decoder U8. Appendix 2A details the memory map for the Data Memory. The first page starting at address &h0000 selects an 8-bit wide input port named "inout_0". The page starting at address &h0800 maps the keyboard and display circuit .The next three pages are not utilised (these are available for future expansion, if required). The fifth page starting at address &h2000 maps the 2K of calibration memory U3. The next page selects the UART at address &H2800. Address &h3000 selects the output port "inout_6". The final 2K page is reserved for the internal data memory of the DSP (512 words) and various memory mapped control registers used by the DSP.

The input port is implemented with an octal tri-state buffer, U9. The DSP read line is ORed with the page select line from U8 to drive the output enable line (OC) of the buffer. Thus a read from address &h0000 will place the 8 bit port data onto the DSP data bus. The least significant bit of the port is connected to the patient response circuit. U10a is configured as a non-inverting buffer and its input is held low by resistor R7. If the patient response button is pressed the ring and tip connections of the jack plug will be shorted and U10a input will be driven high. This will drive U9 input 1D high and LED1 will illuminate to provide a visual indication of the patient response. Input 2D of the buffer is connected to the calibration switch. This is a simple two position switch which is configured to generate a logic high level in the normal position and logic low in the calibrate position. As described above, this line is also used to enable writing to U3 during calibration. It should be noted that the status of this port is polled by the DSP rather than using a more complicated interrupt driven approach.

A tri-state octal latch (U7) implements the output port. The page select line from U8 and the DSP write line are combined by OR gate U10d. The output of the OR gate drives the clock input of the octal latch so that a low to high transition on this line will latch the 8-bit data from the DSP. The output enable pin of U7 is hard-wired low. The two output lines **-80** and **-40** select the attenuation of the analog output circuit. The **Filt** line controls the 250 Hz low-pass filter which is enabled for 125 and 250 Hz test signals. Three lines are used to control the talk through circuit; **Talk** to enable the microphone, **Step** to change the gain and **Up/Dn** to set an increase or decrease in gain. Finally the **Right** and **Left** lines select the appropriate earphone output. A 74ACT chip is used for the buffer as this allows the earphone opto-relays and LEDs to be driven directly.

The UART interfaces directly to the DSP address and data lines. The IC has 8 control registers which occupy the eight addresses selected by A0 to A2. The IC requires an active high reset signal and this is supplied by U11. The 3.6864 MHz crystal supplies the master clock for the

internal baud rate generator and the 16-bit counter/timer. The UART interrupt line is connected to the DSP IRQ2 pin. This active low line is asserted whenever a character is received via the serial input or the counter reaches zero (the DSP has to determine through software which condition has occurred). The serial port consists of the transmit and receive lines **Tx** and **Rx** together with two handshaking lines. **MPO** is configured as a Request to Send (RTS) line and **MPI** is configured as a Clear to Send (CTS) line. This allows hardware handshaking to be implemented between the DSP and a controlling PC.

3.4 ANALOG CIRCUIT DIAGRAM

Appendix 1B details the schematic for the analog section of the circuitry. It can be split into four main sections: the digital-to-analog converter and filter, the amplifier/gain stage, the output routing and the talk through circuit.

The digital-to-analog converter (U1) is an Analog Devices AD1851N-16 bit serial chip. This IC is an improved version of the industry standard PM56P part. (This chip was used in the first generation of compact disc players which appeared on the market in the early 1980s.) It has improved distortion and signal-to-noise figures over the PM56P and does not require a -5V digital supply line. A 74ACT02 NOR gate (U6a) is connected as an inverter to supply the clock signal for U1. This inverter is required because the DSP serial port can only be configured to provide a negative going clock waveform whereas U1 latches the serial data bits on the rising edge of the clock waveform. The analog output voltage produced by U1 is centered about 0V with a maximum full scale amplitude of 2.0V rms and the internal buffer is configured to provide a low impedance voltage output. An OP-27 low-noise operational amplifier (U2) together with R1, R2, C4 and C5 form a Sallen and Key low-pass filter. The component values have been calculated for a unity gain, Butterworth response and the cut-off frequency can be switched between 8 000 Hz and 250 Hz by switching R3 and R4 in or out of circuit via two CMOS 74HC4316 switches (SW1a and SW1b). DC biasing for U2 is obtained by DC coupling to the output of U1.

The amplifier stage is formed by an OP-27 (U3) and four transistors (Q1 - Q4) configured to provide a current gain stage. The signal from the output of the op-amp is fed via R17 to a BC109 (Q3) configured as an emitter follower. Another BC109 (Q4) is connected as a constant current load for Q3 (biasing is provided by R18, R19 and R20). The use of Q4 rather than a purely resistive load increases the current gain of Q3 and also helps to stabilise the bias point of the amplifier by providing a constant current through diodes D1 and D2. These diodes bias the push-pull output stage formed by a BD131 (Q1) and BD132 (Q2) complementary pair into Class AB operation. Again the transistors are connected in common emitter mode to provide current gain. The two emitter resistors R21 and R22 perform a dual role; they equalise any small differences in the current gains of the two output transistors and limit the maximum current that can flow through the transistors if the output is short-circuited. These resistors, together with the ON resistance of the MOSRelays, ensure that the amplifier can withstand a short-circuit at either earphone socket without damage. The amplifier is configured as an inverting stage with the gain (or in this case attenuation) determined by the resistor chain formed by R9 to R16. The 74HC4316 CMOS switches SW2b, SW2c and SW2d tap the

resistor chain to provide attenuations of 0 dB, -40 dB and -80 dB respectively. By connecting the analog switches to the inverting input of the op-amp any resistance or non-linearity in the switch will be effectively eliminated due to the virtual earth formed at the inverting input of the op-amp. This in turn means that the attenuation of the circuit is determined solely by the ratio of the resistors in the input and feedback paths of the amplifier stage. Using 1% resistors allows the attenuation to be set with an accuracy of better than 0.1 dB without any trimming. The signal from the filter stage is AC coupled to the amplifier by C3. The low resistance in the feedback path of the amplifier stage coupled with the good DC performance of the OP-27 allows the output to be directly coupled to the load. DC offsets at the output are also minimised by the fact that the DC voltage gain is unity. An OR gate (U6b) is used to combine the **-40** and **-80** control lines so that 0 dB is selected when both lines are low while LED2, LED3 and LED4 serve to indicate the selected gain.

The output signal from the amplifier is connected to the right or left earphone via photoMOS relays RL1 and RL2. The **R** or **L** control lines drive the relay activating LEDs and also LED5 and LED6 which indicate the earphone selected. A 74ACT02 gate wired as an inverter (U6d) allows this ouput switching to be disabled when its output goes to a logic HIGH state. The input to this gate is wired to the **NReset** signal from the digital circuit. This ensures that the earphones are disconnected from the amplifier output when the digital circuit is in an undetermined state during reset or a power interruption. It has been found that the photoMOS relays exhibit some tone breakthrough in their off state. This appears to be due to a leakage capacitance caused by two protection diodes placed in parallel with the output MOSFETs. In order to eliminate this breakthrough a small-value capacitor is connected from the common MOSFET pin to earth. However, if this capacitor is left connected to the pin when the relay is operated clicks are introduced at the output. The solution adopted is to switch in the capacitor only when the opposite earphone output is selected. This ensures that high hearing level settings do not cause breakthrough to the opposite earphone.

The final section of the analog circuit comprises the talk through circuit. An OP-27 is used (U5) to buffer and amplify the signal from the microphone. Initially a simple rotary potentiometer was employed to provide level control for the talk through signal. However, a more elegant solution was found in the form of the X9313WP digitally controlled potentiometer IC (U4). This allows the talk through gain to be controlled by the DSP with the added advantage that U4 retains the wiper setting in an EEPROM memory even when power is removed. The **Talk** control signal from the DSP controls CMOS switch SW2a which connects the output of U5 via R30 to the inverting input of U3. The value of R30 is chosen such that

with the main amplifier set to an attenuation of -40 the signal from the output of U5 has a gain of unity at the output of the amplifier stage. LED7 indicates when the talk through circuit is active. The **Talk** line also controls the chip select input of U4 via inverter U6c. With the Step input to U4 LOW a pulse on the Up/Dn line will increment the potentiometer wiper position. If the step input is high then a pulse will decrement the wiper position. The potentiometer in U4 has a value of 10K and 30 discrete steps. It is connected to the feedback path of U5 together with R28 and R29 to provide and approximate logarithmic gain range of between 0 and 36 dB. This allows a suitable talk through level to be set for the type of microphone in use.

3.5 DISPLAY AND KEYBOARD CIRCUIT DIAGRAM

Appendix 1C details the circuit diagram of the display and keyboard interface for the audiometer. The circuit is based around an Intel 8000 series chip the 8279 (U1). The 8279 is an 8-bit microprocessor compatible, programmable device which can display data on an array of alphanumeric LED displays or discrete LED indicators. The information to be displayed is written into the on-chip memory and the 8279 then provides the appropriate row and column outputs to drive a multiplexed array of indicators. In a similar fashion a matrix of simple push switches can be scanned by the chip and key presses stored in a first in first out (FIFO) memory.

The Data, RD, WR and Reset pins of U1 are connected to the appropriately designated DSP lines in Appendix 1A. The CS line is taken from the address decoder U8. A single address line (A0) is required to select the data or control registers within the 8279. The clock signal for U1 is derived by dividing the master DSP clock by two to give an operating frequency of 4.608 MHz; the maximum clock frequency for the 8279 being 5 MHz. The display lines comprise OB0 to OB3 and OA0 to OA3. These eight lines are split into two groups of four to allow easy interfacing to different types of displays. In this case the eight lines are buffered by an 74ACT244 (U2). This IC provides enough current sourcing capacity to drive the display LEDs directly. The scan lines SL0 to SL3 provide the column signals for the multiplexed display and the output is in the form of a binary count which is decoded by a 74ACT138 (U3). Each column of LEDs is selected when the appropriate output of U3 goes low. If this coincides with a high row output from U2 then the appropriate LED will be illuminated. The 8279 can be programmed to scan a maximum of sixteen columns however in this case it is programmed to scan eight columns once every 5 ms.

The audiometer frequency is indicated by one of eleven LEDs: LED 1a to 1j and LED 2. The attenuator setting is indicated from -20 to 125 dB HLS by LED 3a to 3j, LED 4a to 4j and LED 5a to 5j. The prototype display utilised 10 segment, dual-in-line LED packages to provide a compact display.

The keyboard switches are scanned in a similar manner to the LEDs. The column select lines are common to the display and key presses are detected on the key return lines RL0 and RL1. If a switch is depressed a low input will appear on the appropriate return line and the corresponding byte loaded into the internal FIFO memory of U1. An interrupt will also be generated to indicate that a key press has occurred however this signal is not utilised by the

DSP. Switch de-bounce and multiple key press routines are built into the 8279 scan routines, virtually eliminating the possibility of erroneous key detections.

Switches are provided to increase (INC) and decrease (DEC) the frequency and attenuator settings, to present the test tone to either the right or left earphone (OUPUT R or L), to enable the automatic test mode (AUTO) and to enable the talk through mode (TALKTHROUGH).

The flexibility of the 8279 chip makes a number of different interfaces possible without requiring any significant extra hardware. The circuit can be readily expanded to accommodate additional audiometer functions by simply adding LEDs or switches to the appropriate scan lines. The frequency and attenuator displays could be replaced with 7-segment numeric LEDs to provide a digital readout of the audiometer settings. Rotary stepped controls with a 2-bit Grey Code output could replace the INC and DEC buttons for frequency and attenuation control. The additional DSP software required to interface these additions would be minimal due to the programmability of the 8279.

3.6 POWER SUPPLY AND ISOLATED RS232 INTERFACE

Appendix 1D details circuits for an isolated power supply and RS232 interface for the audiometer. The circuit was designed to utilise the +5V digital supply from a host computer running audiometer software (a "black box" type of configuration). By isolating the circuits from the PC problems such as supply noise, ground loops and patient/operator electrical safety are effectively eliminated. It also provides a cost saving by eliminating the need for a mains power supply module. However, a stand-alone audiometer utilising the keyboard and display interface would require a mains power supply.

The NMH0509H module (IC1) is an isolated DC-DC converter requiring a 5V input to generate a nominal ±9V output. The +9V line is filtered by C1 and fed to a 7805 regulator (U3) to derive the +5V digital supply for the audiometer. This +9V line is also fed to a three terminal adjustable regulator (U1) the output of which is set to +6V by resistors R1 and R2. In a similar manner the -9V line from IC1 is regulated to -6V by U2 and resistors R3 and R4. This ±6V supply is used to power the analog circuitry of the audiometer.

The isolated RS232 interface is implemented with 4 opto-couplers. The output signals Tx and RTS from UART U6 (see Appendix 1A - Digital Circuit) are buffered by two Schmitt inverters (U5a and U5b) which drive the transmit LEDs of the TIL193 opto-couplers. The opto-coupler receive transistors are tied to the inputs of a MAX232 interface chip (U4). This chip converts the +5V logic levels to the RS232 voltage levels of ±10V and the output lines are connected to the appropriate pins of a standard 9-pin D connector. The Rx and CTS pins of the D connector are connected to the input pins of U4 and the +5V logic outputs from the chip are used to drive the otpo-coupler LEDs. The transistor collectors are connected to Schmitt inverters U5c and U5e via pull-up resistors R11 and R12. The signals are then passed through two more inverters, U5d and U5f, to restore the correct signal polarity. The input hysterisis of the inverters ensures that a clean signal is applied to the Rx and CTS inputs of the UART.

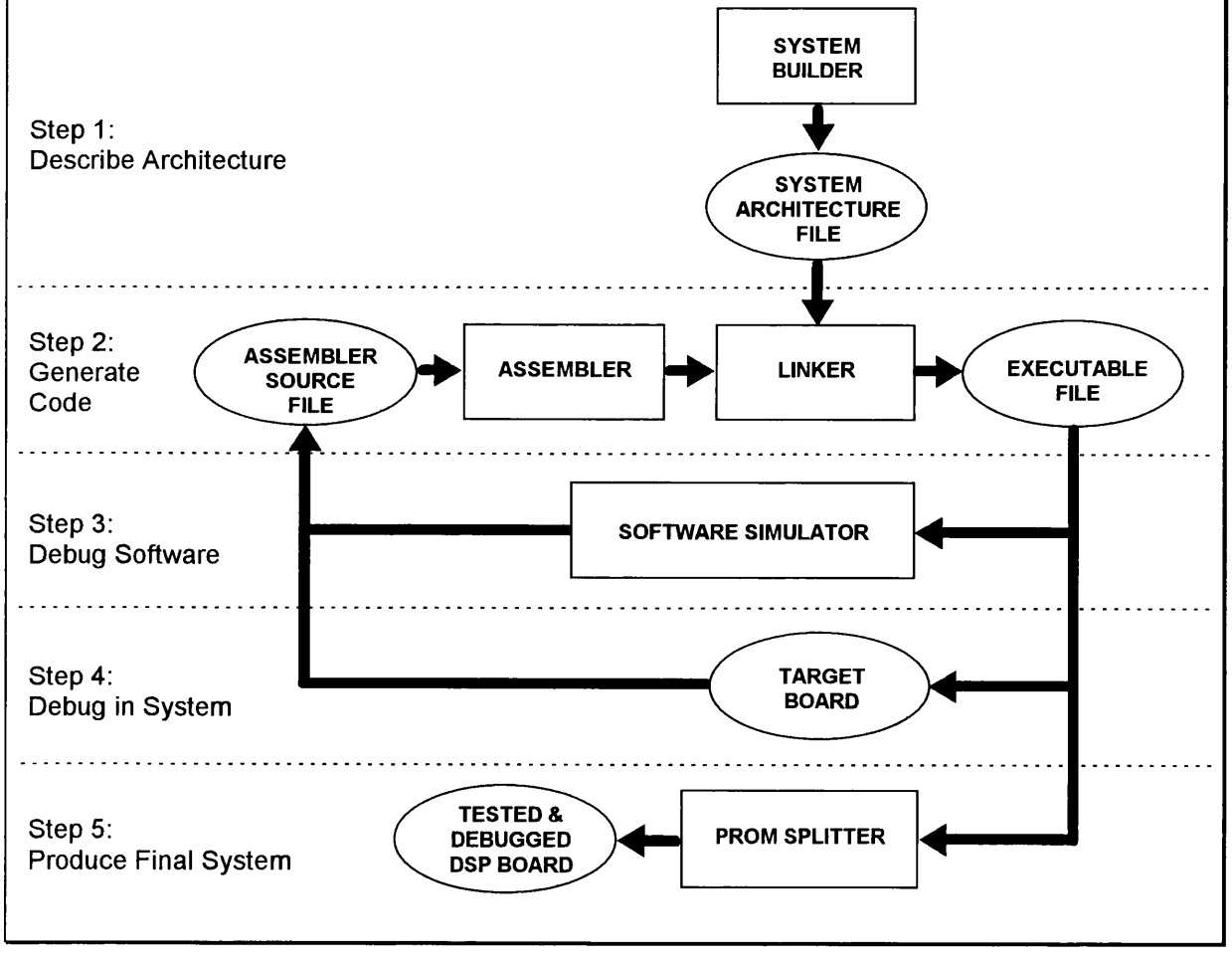
4. DSP SOFTWARE

4.1 INTRODUCTION

A block diagram and overview of the software developed for the DSP is detailed in section 4.2. Section 4.3 describes the System Specification File which defines the hardware setup for the DSP system. The main program listing is documented in Section 4.4, while 4.5 and 4.6 document the UART subroutines. Appendix 2 contains a detailed memory map for the system while Appendix 3 provides the System Specification file listing. Appendix 4 contains the complete source code listing for the DSP software and this is commented on a line by line basis.

The development of software for the ADSP-2105 is carried out in a similar manner to any

FIGURE 4.1 - SOFTWARE DEVELOPMENT PROCESS



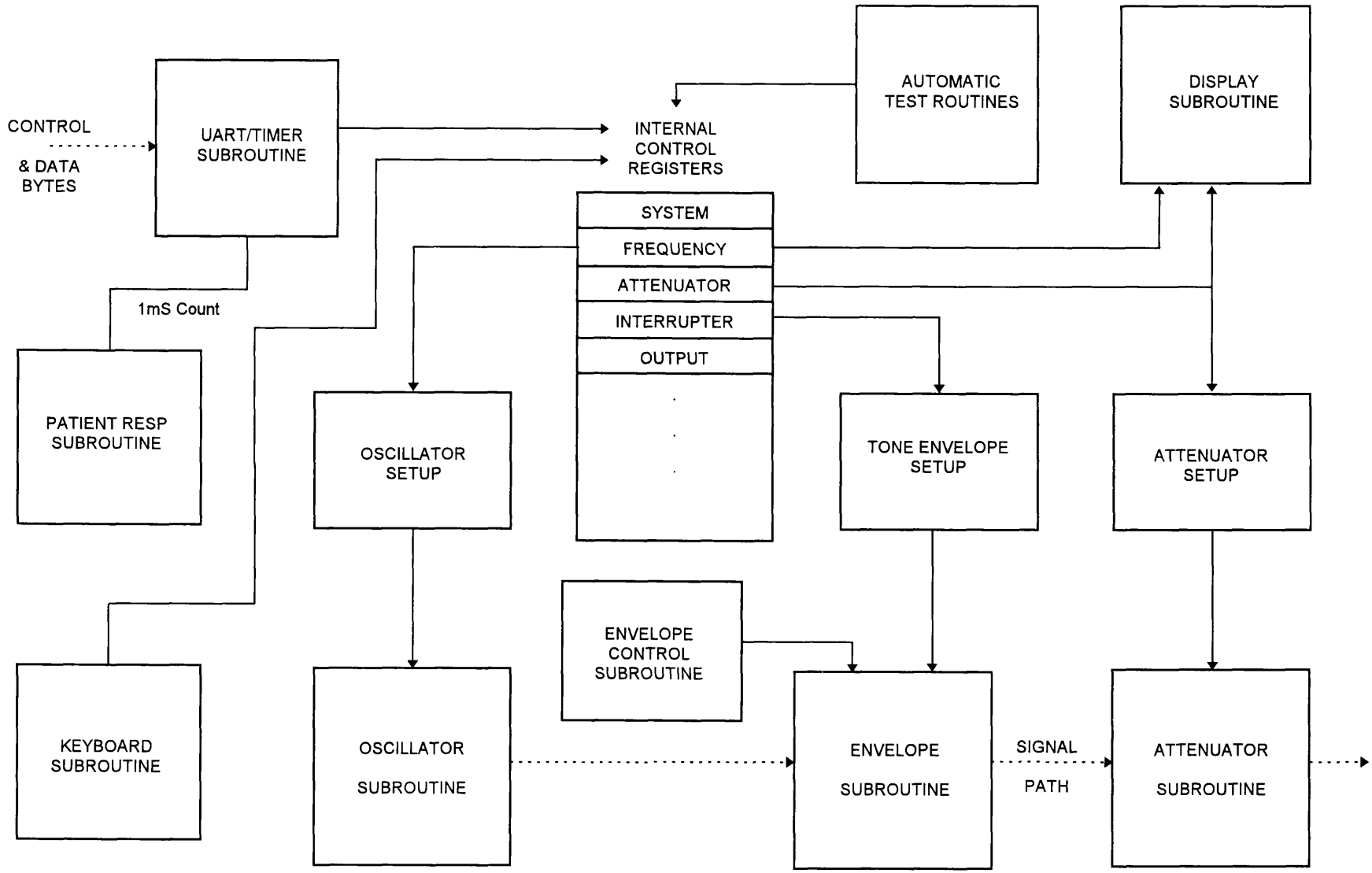
other microprocessor-based system. A set of software development tools supplied by Analog Devices for the ADSP-2100 series of chips was used to write and debug the DSP code using a standard PC. These tools include an Assembler to produce object code, a Linker to combine

object modules into an executable file, a Simulator to test and debug code and a PROM splitter to generate files for EPROM programming.

Figure 4.1 provides a diagramatic representation of the development process and the type of files produced at each stage. The system hardware setup is specified in a System Specification file and this is processed by the System Builder to produce a set of addresses for the Linker program. The DSP program is written using a standard text editor and the file is saved with extension .DSP. This file is utilised by the Assembler to produce the ADSP2105 machine code instructions which are stored in an object file with extension .OBJ. The Assembler can also produce a listing file which has extension .LST. The Linker is then used to link together the object files (there may be other object files containing subroutines, initialisation routines etc.) and resolve all the address labels within the routines. This produces the final executable file with extension .EXE. This file contains the code which is loaded into the DSP program memory. This file can also be loaded into the Simulator to provide a software simulation of how the program would run in the DSP itself. The Simulator has a number of advanced facilities which allow the program to be debugged and these include register viewers, memory viewers, single-step execution, break points, simulation of input and output signals etc. In practice it was found that the simulator was only used when a program bug could not be diagnosed from the behaviour of the hardware.

It is beyond the scope of this document to describe in detail the architecture of the ADSP2105 processor, however, a block diagram detailing the main features of the chip is reproduced in Appendix 6. This diagram also shows the processor registers associated with each functional unit and details the internal data and address buses for the program and data memory. Full details of the ADSP2105 can be found in the Analog Devices data sheet.

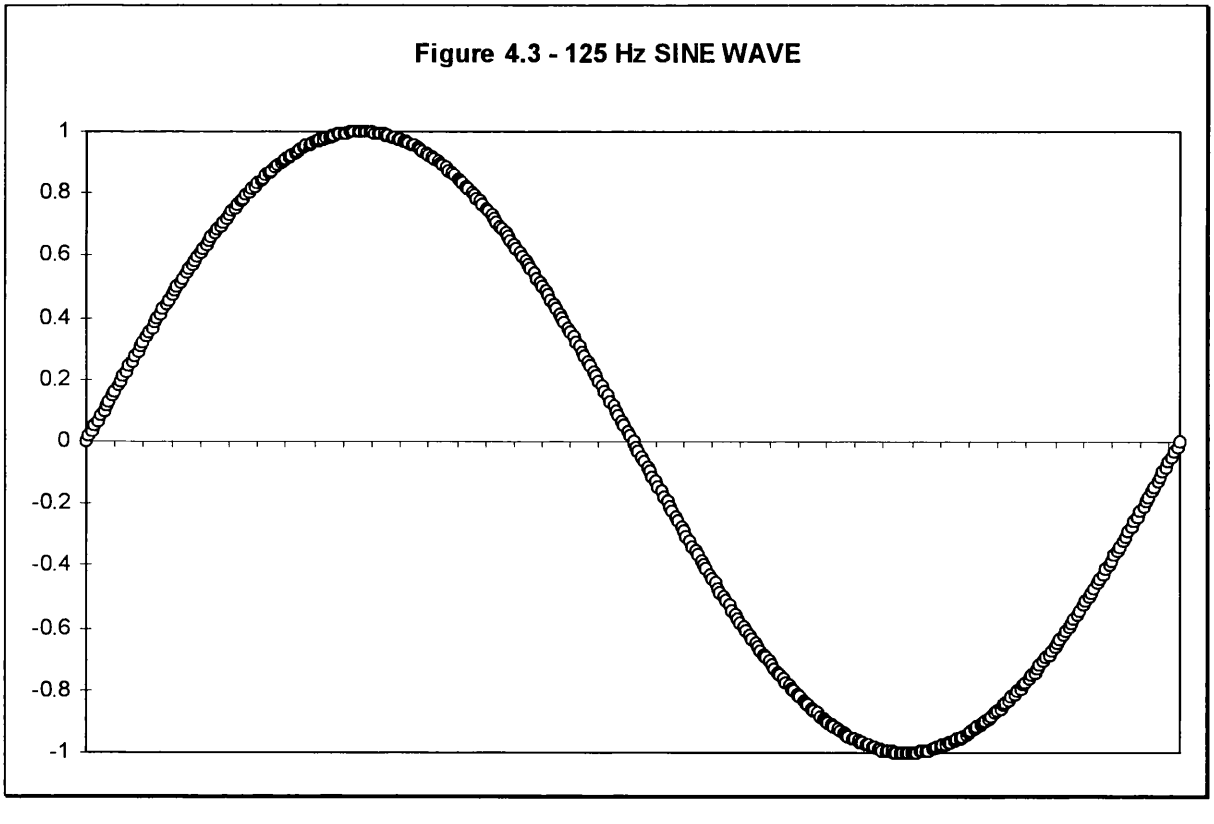
FIGURE 4.2 - SOFTWARE BLOCK DIAGRAM



4.2 SOFTWARE BLOCK DIAGRAM

Figure 4.2 outlines the structure of the DSP software. The software can be thought of as operating at two different levels. The first is to implement the algorithms to generate the sine wave test signal at the correct frequency, amplitude etc. This signal generation has to be carried out in real time so that each sample can be sent to the DAC at the 48 000 Hz sample rate and the coding of the signal generation algorithms has to be compact so that sufficient DSP processing time is left between samples to allow the other functions to be carried out. The second level that the DSP operates on is to implement the control functions traditionally carried out by a microprocessor (see Section 2.3, Figure 2.2) such as setting the frequency and attenuator level, scanning the control panel, writing to the display etc. These control routines are not as time critical as the signal generation routines and so the interrupt structure of the DSP has been configured so that the signal generation routines will always take priority. This ensures that no discontinuity occurs in the samples sent to the DAC, as a single missed sample would result in an unacceptable glitch appearing at the output of the audiometer.

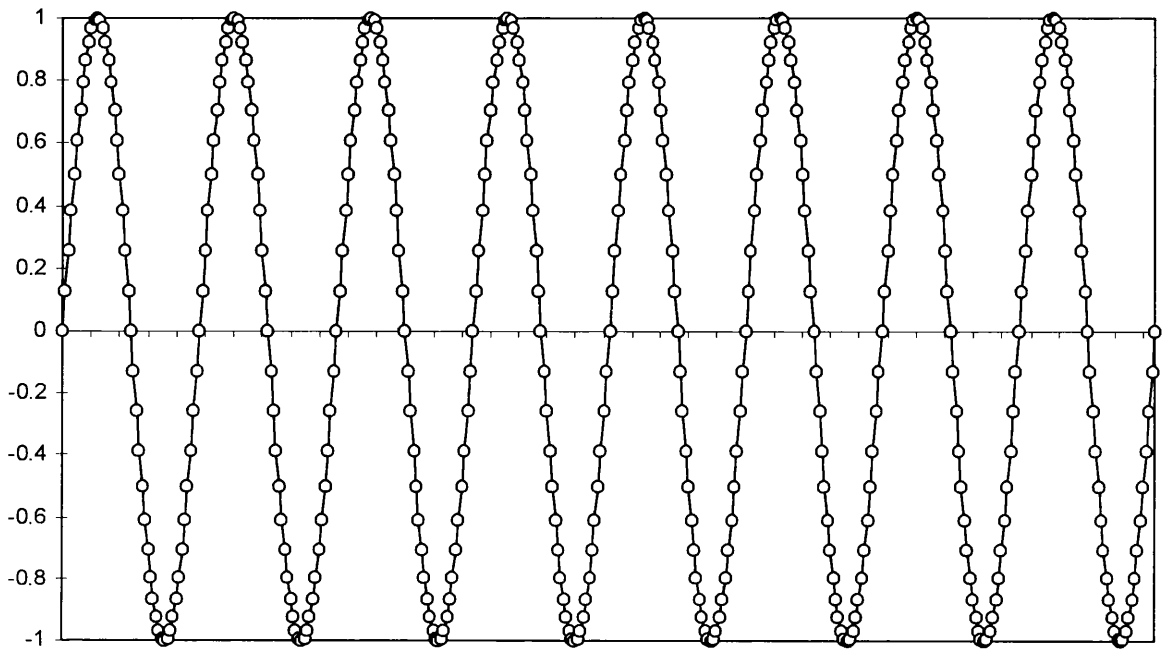
Figure 4.3 - 125 Hz SINE WAVE



The software can therefore be categorised into two main areas: the signal generation routines which comprise the oscillator, envelope generator and attenuator blocks and the control routines which comprise the UART, automatic test, patient response, display and keyboard blocks.

At the start of the signal path is the oscillator block which generates the sine wave test signals. There are a number of different DSP algorithms which can be used for the generation of a sine wave. These are based upon calculating either the amplitude of the wave form in real time, using an appropriate equation, or by utilising some form of look-up table. It was decided that a look-up table would be the most efficient algorithm as it would reduce the amount of processor time spent on calculations. As each audiometric frequency is based on a whole number multiple

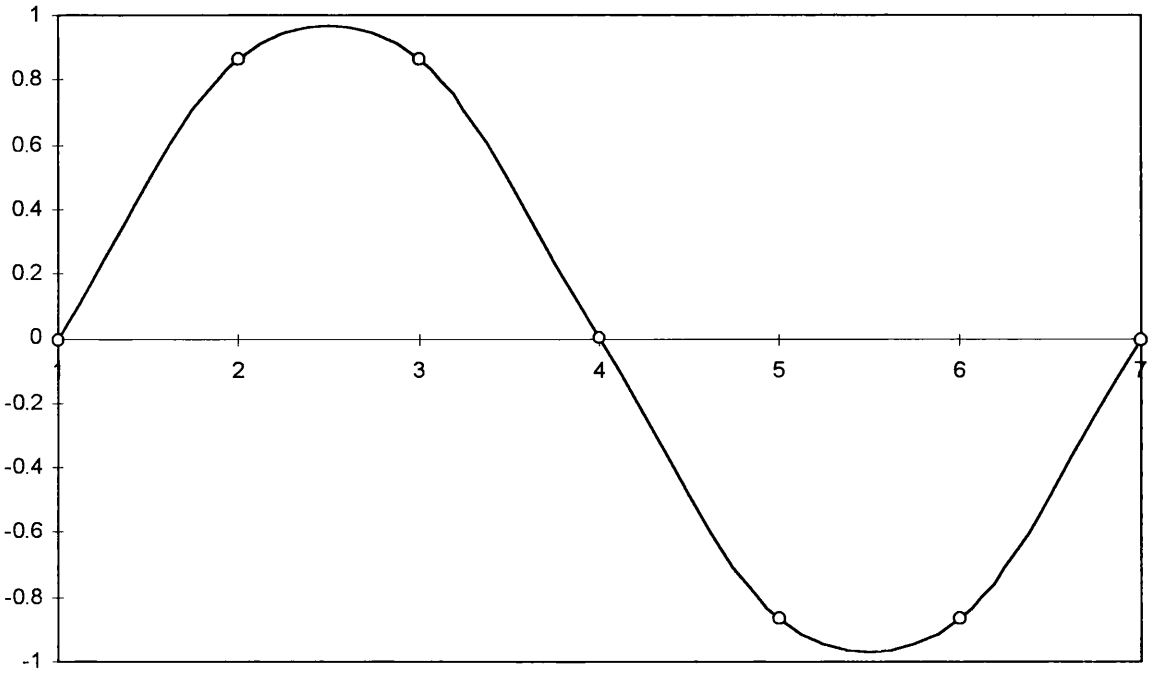
FIGURE 4.4 - 1 kHz SINE WAVE



of 125 Hz there are two basic techniques which could be utilised to generate the sine waves. A fixed length look-up table could be used with a variable sample rate or a fixed sample rate could be used with a varying length of look-up table. A fixed sampling rate was chosen as the best option as this allows a fixed frequency, anti-aliasing filter to be used after the DAC output. This approach also avoids any problems with the generation of time-dependent parameters which could be disrupted by a variable sampling rate. The look-up table consists of one cycle of a 125 Hz sine wave which is automatically initialised in the ADSP-2015s on-chip memory as part of the boot procedure. For a 48 kHz sampling rate 384 samples are required, and these are stored in two's complement 16 bit format. To generate the sine wave signal the DSP on- chip serial port timer is initialised to generate an interrupt every 1/48000 of a second. The oscillator subroutine then reads the appropriate sample from the look-up table and writes it to the serial port buffer. The serial port then converts the parallel data to a serial data stream and clocks it into the DAC. To generate 125 Hz the program steps through the table one

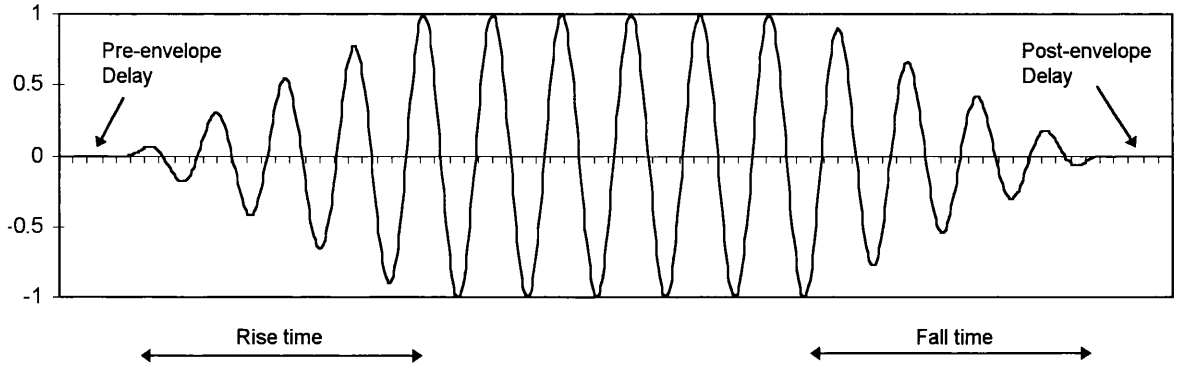
sample at a time. Figure 4.3 graphs the 384 sample points which comprise the 125 Hz

FIGURE 4.5 - 8 kHz SINE WAVE



waveform. The large number of samples make it difficult to discern the individual points on the waveform. Other frequencies are generated by skipping the appropriate number of steps through the table - two steps for 250 Hz, eight steps for 1 kHz etc. - and figure 4.4 graphs the waveform of a 1 kHz sine wave. In this case the waveform has been generated by cycling

FIGURE 4.6- TONE ENVELOPE



through the look-up table eight times to produce 384 samples of the 1 kHz sine wave. Figure 4.5 graphs one cycle of the 8 kHz sine wave which is generated using six sample points. An accurate sine wave is obtained if suitable low-pass filtering is employed to eliminate the sampling harmonics.

The presentation of the test tone requires an envelope to be applied to the basic sine wave with accurately defined rise and fall times and no under/overshoot or discontinuities. The envelope routine achieves this by multiplying the sample from the look-up table with a weighting which defines the shape of the envelope over time. Figure 4.6 details a typical tone envelope. This figure also shows the pre- and post-envelope delays which are utilised when connecting the earphones to the output amplifier via the photoMOS relays. The appropriate relay is energised just prior to the start of the tone envelope and then de-energised once the decay portion of the envelope has been completed. This eliminates the possibility of audible clicks at the beginning, or end, of the tone presentation and ensures that the earphone is only connected to the amplifier during the presentation of a tone.

FIGURE 4.7 - ATTENUATOR TABLE

HEX	Decimal	Actual	Nominal	Error
7FFF	32767	100.00	100	0.00
7214	29204	99.00	99	0.00
65AC	26028	98.00	98	0.00
5A9D	23197	97.00	97	0.00
50C3	20675	96.00	96	0.00
47FA	18426	95.00	95	0.00
4026	16422	94.00	94	0.00
392C	14636	93.00	93	0.00
32F5	13045	92.00	92	0.00
2D6A	11626	91.00	91	0.00
287A	10362	90.00	90	0.00
2413	9235	89.00	89	0.00
2027	8231	88.00	88	0.00
1CA8	7336	87.00	87	0.00
198A	6538	86.00	86	0.00
16C3	5827	85.00	85	0.00
1449	5193	84.00	84	0.00
1214	4628	83.00	83	0.00
101D	4125	82.00	82	0.00
0E5D	3677	81.00	81	0.00
0CCD	3277	80.00	80	0.00
0B68	2920	79.00	79	0.00
0A2B	2603	78.00	78	0.00
0910	2320	77.00	77	0.00
0813	2067	76.00	76	0.00
0733	1843	75.00	75	0.00
066A	1642	74.00	74	0.00
05B8	1464	73.00	73	0.00
0518	1304	72.00	72	0.00
048B	1163	71.00	71	0.00
040C	1036	70.00	70	0.00
039B	923	69.00	69	0.00
0337	823	68.00	68	0.00
02DE	734	67.01	67	0.01
028E	654	66.00	66	0.00
0247	583	65.00	65	0.00

HEX	Decimal	Actual	Nominal	Error
0247	583	65.00	65	0.00
0207	519	63.99	64	-0.01
01CF	463	63.00	63	0.00
019D	413	62.01	62	0.01
0170	368	61.01	61	0.01
0148	328	60.01	60	0.01
0124	292	59.00	59	0.00
0104	260	57.99	58	-0.01
00E8	232	57.00	57	0.00
00CF	207	56.01	56	0.01
00B8	184	54.99	55	-0.01
00A4	164	53.99	54	-0.01
0092	146	52.98	53	-0.02
0082	130	51.97	52	-0.03
0074	116	50.98	51	-0.02
0068	104	50.03	50	0.03
005C	92	48.97	49	-0.03
0052	82	47.97	48	-0.03
0049	73	46.96	47	-0.04
0041	65	45.95	46	-0.05
003A	58	44.96	45	-0.04
0034	52	44.01	44	0.01
002E	46	42.95	43	-0.05
0029	41	41.95	42	-0.05
0025	37	41.06	41	0.06
0021	33	40.06	40	0.06
001D	29	38.94	39	-0.06
001A	26	37.99	38	-0.01
0017	23	36.93	37	-0.07
0015	21	36.14	36	0.14
0012	18	34.80	35	-0.20
0010	16	33.77	34	-0.23
000F	15	33.21	33	0.21
000D	13	31.97	32	-0.03
000C	12	31.27	31	0.27
000A	10	29.69	30	-0.31

The final signal path block controls the attenuation of the test signal. In a similar manner to the envelope routine the attenuator is implemented by multiplying the full-scale sample from the sine wave look-up table with a weighting corresponding to the required attenuation level. The appropriate weightings for 1dB steps are stored in a look up table in the calibration EEPROM together with the corresponding gain settings for the output amplifier. Calibration offsets are stored in a separate table and are added to the attenuator setting to index into the look-up table and retrieve the appropriate weighting. Figure 4.7 tabulates the values held in the attenuator table, in hex, together with the actual attenuation values in dB. The final column of the table lists the errors between the nominal attenuator setting and the actual setting. It can be seen that over a 60 dB range (from 100 to 40 dB) the maximum error is less than ±0.1 dB. If the range is extended by a further 10 dB the maximum error increases to ±0.3 dB. This therefore is the maximum useable range of attenuation using this technique.

The UART/TIMER block performs two important control functions. It provides a serial interface which conforms to the RS232 specification and it generates a 1mS interrupt signal which is utilised by the DSP for timing in the patient response and tone presentation routines. The serial port allows remote control of the audiometer functions. Commands for the DSP are sent from a controlling PC in a two byte format. The first byte is a command byte and this signifies the parameter to be altered - frequency, attenuator etc. The second byte is a data byte and this indicates the setting of the parameter specified by the command byte - 125 Hz, 35dB HLS etc. The DSP internal data memory contains a set of control registers and which are written to by the UART routine. A flag is then set to indicate that a parameter has been altered and the signal path is set up by an appropriate routine. In order to minimise transients all parameter changes are carried out at the zero crossing point of the sine wave. Hence the subroutine to change a parameter is only called when the start of the sine wave table has been reached. Appendix 5 lists the control codes and data formats for the serial interface.

The patient response block detects when the response button has been pressed and, in addition, provides a measurement of the time between the start of the tone presentation envelope and the closure of the patient switch. When this routine detects the start of a tone presentation (the envelope multiplier reaches its maximum amplitude) a software counter is initialised to zero. This counter is then incremented at 1mS intervals, using the interrupt from the UART timer, until a patient response is detected. The response time can then be read by sending an appropriate command byte to the UART - two bytes are returned (MSB and LSB) giving the response time in mS . A maximum response time can be set beyond which any response will be ignored and this facility is utilised by the automatic test routines.

The keyboard routine provides the facility of manually controlling the audiometer functions via push switches. The routine reads the value of the key pressed and translates it into appropriate control and data bytes which are written to the appropriate registers. If, for example, the frequency increment key had been pressed the routine would read the current data byte from the frequency control register increment it by one and write the new value to the register.

The display block is called whenever a new value is written to either the frequency or attenuator control registers. The data byte is converted to a bit pattern which will illuminate the appropriate frequency or attenuator bar graph LED. This byte is then written to the display chip which handles the scanning of the display matrix.

The envelope block controls the generation of "one-shot" (the tone is presnted for a fixed time length) or pulsed tone presentations. The 1mS interrupt used by the patient response subroutine is also utilised by this routine to provide accurate timings for the length of tone presentations. In the case of the "one-shot" presentation the on-time of the tone envelope is defined by a software counter. The pulse presentation mode allows a pulsing tone to be generated with the off and on time intervals controlled by two different software counters. The timings for both the one-shot and pulsed tone signals can be altered through software.

The automatic test routine performs an automated hearing test according to a recognised test protocol ("BSA Recommended Procedures for Pure Tone Audiometry"). This test consists of increasing the attenuator in 5 dB steps until a valid response from the patient occurs. The attenuator is then reduced in 10 dB steps until no response is obtained at which point the level is again increased in 5 dB steps until a valid response is obtained. A valid tone detection (or reversal) is defined as the point were the attenuation reverses from +5 to -10 dB. The process is then repeated until a second reversal is obtained. If the two reversals occurred at the same attenuator setting then this point is taken as the hearing threshold for that test frequency. If the two reversals do not match then the test is continued until at least two out of four reversals match. Once a valid threshold has been determined the test progresses to the next frequency and the process is repeated until a complete series of thresholds has been obtained.. A random delay is given to the time between tone presentations so that the patient cannot anticipate the start of the tone. Various test parameters can be set up in software including the order of tone presentation and tone lengths. A valid response time can also be set up to ensure that a response is only accepted if it occurs within a certain time frame from the start of the tone presentation.

At the completion of the test (or if the test is aborted) the test results are transmitted from the serial port in a format suitable for printout. Figure 4.8 details a sample audiogram.

FIGURE 4.8 - SAMPLE AUDIOGRAM PRINTOUT

```
PATIENT AUDIOGRAM

LEFT      1000      10
LEFT      2000      15
LEFT      3000      15
LEFT      4000      10
LEFT      6000       5
LEFT      8000      25
LEFT       500      10
RIGHT     1000      30
RIGHT     2000      40
RIGHT     3000      45
RIGHT     4000      50
RIGHT     6000      55
RIGHT     8000      70
RIGHT      500      15
LEFT      1000      10
```

4.3 SYSTEM SPECIFICATIONS

Appendix 3 provides a listing of the system specification file for the DSP hardware. This file is required by the ADSP-2105 assembler to resolve address labels in memory and data space and to specify both internal and external memory size. The first three lines of the listing specify the system name, the processor type and the state of the MMAP pin. In this case MMAP is held low which indicates that the system has an external boot memory. The next four lines specify four pages of 1K boot memory. The 1024 word internal program memory and 512 word internal data memory are specified next followed by the 2048 byte calibration EEPROM. In addition to the memory size (the figure in square brackets) the specification file also contains the start address of the memory block (prefixed by "ABS="). The remaining statements use the "PORT" identifier to define absolute addresses for the hardware mapped into the DSP external data memory space. These comprise the data and command registers for the 8279 keyboard/display chip, the eight interface registers for the SCC2691 UART chip and the input and output data latches which interface to the analog circuitry.

The system specification file facilitates hardware changes to the DSP system by defining all memory and port address labels. Thus changes can be made to the system hardware and provided that the system specification file is updated none of the program files will require alteration.

4.4 MAIN PROGRAM

4.4.1 Software Initialisation

The main audiometer program module is listed in Appendix 4A. The first section of the program consists of a number of initial configuration steps. The PORT commands set up the hardware addresses from the DMC2105 system file. The program variables are then defined in either data memory or program memory. The variables take the form of a table of memory words (the number of words is enclosed in square brackets) or a single storage location. In the case of the sine wave look-up table the memory space is defined as a circular buffer with the command "CIRC". This instructs the DSP to automatically loop back to the beginning of the buffer once the end has been reached. Other tables include the control register (reg_tab) and subroutine register (sub_tab) which control the audiometer functions, automatic testing tables and strings to create an audiogram print out and the calibration tables stored in the calibration EEPROM. A number of single word variables are used and they are placed in the internal data memory of the DSP.

The "INIT" command is used to load variables with the appropriate data. This data can either be from a disk file, as in the case of the sine table, or specified in the program code as in the case of the text strings used for audiogram printing.

The "GLOBAL" command makes the variable names accessible to other program modules so that subroutines called by the main program can share data.

The "EXTERNAL" command defines the names of any subroutines called by the main program. The "uart_set" subroutine initialises the SCC2691 UART while the "uart" subroutine services the interrupt generated by the UART timer. Both of these subroutines are described in sections 4.5 and 4.6 respectively.

The interrupt vector table specifies the program jumps in response to the ADSP-2105 interrupt sources. Although seven interrupts are listed the ADSP-2105 has only one serial port (SPORT1) and so the SPORT0 interrupts are listed only for compatibility with the parent ADSP-2101 processor. Program control addresses the first interrupt vector on a hardware reset and a jump instruction moves to the start of the main program. The next vector services the hardware interrupt from the UART and directs program control to the UART Subroutine. The SPORT1 transmit interrupt signals that the serial port buffer is ready for the next waveform sample and a jump is made to the appropriate part of the code. The SPORT1 receive interrupt is not utilised by the program. Finally the DSP timer interrupt jumps to the envelope generation

routine. Then interrupt priority is reflected in the table order with a hardware reset being of the highest priority and the timer of lowest priority.

The "restart" label marks the start of the program where the first instruction calls the initialisation subroutine for the UART. On return from this routine the 8279 display/keyboard chip is initialised. The next code segment configures the DSP Data Address Registers, or DAG. There are eight sets of DAG and each set consists of three registers. The Index (I) register points to a data or program memory location. The Modify (M) register is added to the I register after a memory access to allow post-modification of the address pointer. The Length (L) register specifies the length of a circular buffer in memory and is set to zero for all other memory accesses. I registers are set up for the three circular tables (sine, threshold and random) and the main control register.

The addresses of the audiometer routines are loaded into the subroutine table in data memory. These are the routines which are called when a data byte is written into the corresponding control register. To allow for future developments space is provided for a second channel of control routines.

The next stage of the program configures the ADSP-2105 internal control registers. SPORT1 is configured for a 48 kHz sampling rate and initialised to a sixteen bit data word format. The DSP TIMER is configured to provide an interrupt once every 50 clock cycles. Wait cycles are programmed for the EEPROM and interface chips and SPORT1 is loaded with 0. The final commands set up the interrupt structure and enable the IRQ2, SPORT1 and TIMER interrupts.

The final section of the initialisation routines loads the control registers with an initial audiometer setting of 1000 Hz, 30 dB Hearing Level and left earphone output. Calls are made to the output and tone interrupter routines to implement these settings.

The next stage of the program, after initialisation has been completed, comprises the main loop. This is simply a two instruction section of code which puts the processor into an idle state while waiting for an interrupt to occur. The ADSP-2105 enters a reduced power mode in the idle state as it is not processing any program instructions. When an interrupt occurs the appropriate address from the interrupt vector table is loaded into the DSP program counter and program execution continues from there.

4.4.2 Envelope Routine

The envelope weighting varies from 0x0000 (tone off) to 0x7FFF (tone on) and is incremented or decremented in steps of three to create the rise or fall portions of the tone envelope. If the

DSP master clock period is multiplied by the number of steps in the tone envelope (0.108μS x 10922) the minimum rise or fall time interval can be calculated (1.2 mS). By programming the TIMER to divide the master clock by figures greater than unity the rise and fall times can be increased proportionately. In the initialisation section of the program the TIMER divisor was set to 50 and so this gives a rise/fall time of 60 mS (50 x 1.2mS).

The first section of the envelope routine program listing tests the current envelope weighting (held in MX1) to see if it has almost reached the maximum value of 0x7FFF. If this is the case the millisecond count generated by the UART timer is reset to zero. This count is used to measure the time from the presentation of the tone (ie when the tone envelope reaches maximum) to a response from the patient. The next stage of the routine takes the envelope increment and adds it to the current envelope value. This increment will be either +3 or -3 for the rise or decay portions of the envelope respectively. Checks are made to ensure that the weighting does not increase above its maximum value (this would be an overflow) or decrease below zero (resulting in a negative weighting). At the end of the routine the new envelope weighting is loaded into the MX1 register and a return made from the interrupt.

4.4.3 Sample Routine

The first section of this routine retrieves the sine wave sample from the look-up table. The sample is then multiplied by the envelope weighting in MY1 and the attenuator weighting in MX0. The resulting value is loaded into the serial port transmit register TX1. The serial port converts the sixteen bit word to a suitable format for transmission to the DAC and controls the timing of the digital to analog conversion.

The next section of the listing controls the pre- and post-envelope delays. The delay count variable is decremented and tested to ascertain if it has reached zero. In the case of the pre-envelope delay the TIMER interrupt is then enabled to start the tone envelope. For the post-envelope delay the control byte is output to switch off the appropriate earphone relay. If the delay counter is greater or equal to zero the program checks to see if the tone envelope has reached zero. If this has occurred the delay count variable is initialised with the post-envelope delay value.

The keyboard/display chip is polled by the sample routine to determine if a keypress is present in the FIFO buffer. Although the 8279 can be programmed to produce a hardware interrupt this method was not used because the single ADSP2105 external interrupt line is connected to the UART chip. If a keypress has been stored the program first checks to determine if the code corresponds to a frequency increment or decrement. If this is the case the current frequency is

obtained from the corresponding control register and incremented or decremented as appropriate. The address of the frequency change routine is then loaded into the control register to ensure that the new frequency is set up. A similar procedure is followed to test for the attenuator increment/decrement, the right/left and the auto keys.

The final section of the sample routine tests the index register for the sample table, I4, to ascertain if the start of the table has been reached. If I4 is pointing to the beginning of the table a jump is made to the routine address stored at the start of the control register. If, for example, the stored address was that of the frequency routine then this routine would be run to set up the new frequency parameters. The beginning of the sine table corresponds to the zero-crossing point of the sine wave, hence any changes in signal parameters will be carried out at this point in the waveform. This, in turn, will eliminate the possibility of any discontinuities in the generated output waveform.

4.4.4 Frequency Routine

The first step of the frequency routine loads the start of the control register with the program label "return". This label simply points to a return from interrupt instruction (RTI) which ensures that the program does not make any further calls to parameter changing routines when the start of the sine wave sample table is reached. The routine continues by multiplying the frequency code by two and then adding this offset to the base address of the frequency table held in the calibration EEPROM. Appendix 2 details the structure of this table where each frequency has two corresponding bytes of information. The first byte contains the SPORT sampling rate, while the second byte contains the step size between samples. (The facility to vary the sample rate is not utilised in this version of the audiometer code however, it was incorporated to allow for future developments.) The sample rate is read first from the table and masked to extract the lower byte from the sixteen bit word (the calibration memory is a standard 8-bit memory chip and so the upper byte of any sixteen bit accesses will be undefined). The SPORT1 sample clock is then updated with the new value. The sample step size is then read, masked and stored in the modification register, M4. The output control register is then inspected to determine which earphone is currently selected for output. This value, together with the frequency code is used to offset into the calibration table. Appendix 2 shows the structure of the calibration table. For each frequency there is an calibration offset byte and values are stored for both the right and left earphones. The pointer to the appropriate calibration offset is stored in the variable "cal_off".

The final section of the frequency routine updates the display registers of the 8279 chip so that the correct frequency LED is illuminated. The frequency code is decremented and loaded into the shifter control register. A single digit is then shifted to appropriate bit location in a sixteen bit word where each bit corresponds to one display LED. The 8279 display registers are then loaded with the least and most significant bytes of the word to update the display.

4.4.5 Attenuator Routine

In a similar manner to the frequency routine the first step of the attenuator program segment is to load the start of the control register table with the label "return". The next section of the program indexes into the attenuator table to retrieve the correct weighing for the desired hearing level. Appendix 2 details the memory map for the attenuator table which is stored in the calibration EEPROM. Three bytes are utilised for each attenuator step. The first and second bytes comprise the least and most significant bytes of the sixteen bit word which forms the attenuator setting. These correspond to the values tabulated in Figure 4.4 above. The third byte provides the setting for the output amplifier attenuation stage. The table covers a range of -20 to 120 dB in 1 dB steps. The range from -20 to 40 dB uses an output attenuation of -80 dB, while the 40 to 80 dB and 80 to 120 dB ranges use attenuations of -40 and 0 dB respectively. Although the actual weightings follow the same pattern for each of the three ranges, having an individual weighting for each step allows the attenuation value to be varied if required. This facility could be used to fine tune the attenuator at low settings without having to perform any hardware modifications.

The attenuator setting is retrieved from the control register table and initially multiplied by fifteen. The attenuator value is expressed in 5 dB steps and this figure, together with the three byte storage size, gives the factor of fifteen. The fine attenuator value and calibration offsets are then retrieved and multiplied by three (these are both 1 dB steps) and summed with the previous figure to achieve the final offset into the attenuator table. A check is then made to determine if the end of the table has been reached. If so the attenuator value is decremented and the new value stored. A jump is then made to the start of the procedure and the process repeated to see if the reduced value is within range. This process will repeat until the value is brought within the calibration table. The next program segment masks and combines the least and most significant bytes to form the weighting word for the sine wave sample. This value is stored in the multiplier register MX0.

The program then sets up the control byte for the output attenuation setting. A check is made to determine if the attenuation is 0 dB and the frequency is 125 or 250 Hz. In this case the output

filter cut-off frequency is reduced from 8 KHz to 250 Hz and this provides extra filtering of the sampling harmonics at these settings. The control bits are then loaded into the step nibble variable and the delay count is initialised to control the timing of the change in attenuator setting.

The final section of the attenuator code sets the 8279 LED display to indicate the correct attenuator setting. In a similar manner to the frequency routine a single bit is loaded into the shift register and moved to the appropriate display position. In this case four display bytes are used to represent the attenuator setting and these are sequentially loaded into the 8279 display registers.

4.4.6 Output Routine

The output routine is a short section of code which enables the right or left earphone. When running this routine the program first jumps to the label "oput_sub" where a call is made to the output routine. On return from the routine program execution continues with the frequency setting program to load the calibration offsets for the new earphone and set up the control byte to output to the analog hardware.

The output routine (at label "set_oput") checks the right/left bit of the control register. It then enables the appropriate bit in the control byte and stores the nibble in the variable "cont_nib". This variable is accessed by the frequency routine to enable the appropriate earphone.

4.4.7 Tone Interrupter Routine

The tone interrupter routine controls the set up of the one-shot and pulsed tone presentation and the set up of the tone envelope itself. The first section of the routine gets the tone interrupter control byte and tests the one-shot enable bit. If this bit is true the variable "tone_off" is loaded with the length (in milliseconds) of the one-shot presentation. This variable is used by the UART timer routine to turn off the tone after the allocated time. In a similar manner the pulsed tone bit is tested and the variables "tone_off" and "puls_len" are set up if the bit is enabled. The tone_off value controls the length of the tone presentation while the puls_len value sets the time at which the tone presentation repeats.

The final code segment controls the tone envelope. If the tone presentation bit is not selected the envelope increment value is set to -3 and stored in "env_inc". This will start the fall portion of the tone envelope. If the bit is enabled the envelope increment is set to +3 and stored. A check is then made to determine if the tone is currently on and the routine terminated if this is the case. Otherwise the set tone bit must signal the start of the tone envelope. The control byte is

assembled from the two nibbles and output to the control port to setup the analog output stage circuitry. The delay variable is then initialised and the TIMER interrupt disabled. The TIMER will be enabled by the envelope routine after the delay period to implement the rise portion of the tone envelope. The codes ends with a return from interrupt instruction.

4.4.8 Auto Control Routine

The final section of the main program controls the operation of the automatic testing routines and provides the final test results in the form of a printable table. A call is made to the auto control routine at the start of an automatic test (ie when the auto button is pressed) or at the end of a test (this may also be at the end of a partly completed test). The auto control byte is first tested to determine if a new test has been initiated, in which case the automatic test parameters are initialised, or if a test has ended, in which case a jump is made to the audiogram printing section of the program.

The automatic test initialisation code begins by setting up the variables "auto_off" and "auto_len". These variables define the auto tone length and repitition rate in a similar manner to the pulse tone parameters however, a random variation to the time between tone presentations is added during the automatic test. The next section of code clears both the threshold table and the audiogram table. The threshold table stores the hearing levels corresponding to consecutive tone detections at a single frequency and consists of three memory locations. The audiogram table stores the final threshold results for each earphone and frequency tested as defined in the auto test table. The routine then indexes into the start of the auto test table to determine the settings for the start of the test. The frequency, attenuator and output control registers are loaded with the appropriate values and the tone interrupter register is set up for a tone presentation. Finally the output routine is called and a jump made to the tone interrupter routine to initiate the automatic test.

The section of code starting at the label "auto_audg" prints the test results to the serial port and makes use of the subroutine at the label "send_chars". This subroutine reads the character pointed to by the I5 register and writes it to the UART transmit register. The UART status register is then read and a loop is entered until the byte has been transmitted. The process is repeated for each character in the string until the terminating character, zero, is reached and a return is then made from the routine. The first section of the code sends a blank line followed by a title for the audiogram and two further blank lines. Two pointers are then set up: the first points to the audiogram result table while the second points to the auto control table. The first test frequency is read from the auto table and the address of the corresponding print string is

calculated. This pointer is stored while the auto table is used to determine whether to print a right or left string. Two spaces are then printed and the frequency pointer is used to print the appropriate frequency. The line is completed by reading the attenuator value from the threshold table and indexing into the attenuator string table to print the corresponding characters. After printing a carriage return and line feed the process is repeated for each threshold stored in the table. Finally a return from interrupt instruction is executed to return to the main program loop.

4.5 UART SUBROUTINE

The program listing in Appendix 4B comprises the routines which are called in response to a hardware interrupt from the SCC2691 UART. The interrupt is generated by the on-chip timer or by the receipt of a byte from the serial port. The UART interrupt register is interrogated by the program to determine the source of the interrupt and a jump is made to the appropriate handling routine.

The initialisation section of code names the program module, defines variable names and sets up the entry point for the subroutine.

4.5.1 Timer Routine

The first section of code (at label "uart") begins by enabling the secondary set of registers. This stores the processor registers so that when the UART subroutine is exited the original register values can be restored to enable the main program execution to continue without disruption. The interrupt status byte is then read from the UART and tested to determine if the source of interrupt was the counter or serial port. If a byte has been received a jump is made to the Serial Port Routine otherwise a control byte is sent to the UART to stop the counter. The "mS_count" variable is finally incremented and stored in memory to complete the routine.

4.5.2 Patient Response Routine

The status of the patient response button is tested by the next section of code. By testing the response button at each timer generated interrupt a 1 mS resolution can be obtained for patient response timings.

The input port is read and the resulting byte tested to determine if the button is pressed (a high bit) or not (a low bit). The "resp_flag" variable is loaded with the response byte to store the current status and then the byte is compared with the previous status to ascertain if there has been any change. If the button has been pressed (a low to high transition) or released (a high to low transition) a status byte is output from the serial port to indicate the change in the button state. In the case of a button press the next section of code calculates the time from the tone presentation to the patient response.

The previous response time is interrogated to determine if the button has been pressed prior to the current response. A response time of zero indicates that there has been no previous response and the program then tests to see if the timing is greater than the maximum response time. If

the maximum response time is exceeded a jump is made to the next section of code otherwise the response time is stored in a sixteen bit format in two consecutive memory registers.

4.5.3 Automatic Test Routine

The first section of the automatic test routine controls the timings for the one-shot and pulsed tone presentations. The code beginning at the label "test_off" compares the current millisecond count to the length of the one-shot tone presentation. If the values are equal the tone interrupter parameters are set to implement the decay portion of the tone envelope. In a similar manner the code at label "test_puls" tests the current count against the pulse tone length and implements the rise portion of the tone envelope if the values are equal.

The code for the automatic test starts at label "test_aoff" and this begins by testing for the end of the automatic tone envelope in the same manner as the one-shot routine. The next section of code tests for the start of the auto tone envelope. In the case of the automatic test an extra random delay is added between the tone presentations. This delay is therefore added to the auto pulse length before it is compared to the current counter value. If the start of the tone envelope has been reached the next step is to determine if a valid patient response has been detected during the previous tone presentation. If the patient response time is zero (ie no response) the attenuator offset is set to +5 while if the response time is non-zero (ie a valid response has been detected) then the attenuator offset is set to -10. The previous attenuator change is then compared to the current value to determine if a reversal from +5 to -10 has occurred. Such a reversal indicates that a valid threshold has been detected and a comparison is then made to the three previous thresholds (stored in the threshold table). If a match is not found a jump is made to the "set_attn" label to set up the attenuator for the next tone presentation and continue the test. If a match is found the threshold is deemed to be valid and is stored in the audiogram table at the appropriate frequency location. The threshold table is then filled with 0x00FF to clear the previous thresholds and the current attenuator setting is increased by 20 dB to give the initial tone level for the new test frequency. The pointer to the auto table is used to retrieve the next test frequency and this is tested to determine if the end of the test has been reached. If the test has finished the address of the audiogram printing routine is loaded into the control register so that the test results are printed to the serial port at the next zero crossing of the sine wave. If the test has not finished the code at the label "set_freq" sets up the new frequency and test earphone before jumping to label "start_cnt" to restart the UART counter and exit the subroutine. The code at label "set_attn" is used to set up the new attenuator value between tone presentations while the code at label "set_tint" is used to set up either a rise or fall setting for the tone envelope.

4.5.4 Serial Port Routine

The final section of code in the UART subroutine handles the reception and transmission of data via the serial port. The program retrieves the received byte from the UART and tests it to determine whether it is a control or data byte. In the case of a control byte a pointer is set up in register I6 which identifies the appropriate control register. A further test is made of the control byte and if a data read is indicated the UART transmit register is loaded with the contents of the control register. If the control byte indicates a data write a flag is set to indicate that the next byte received will be a data byte.

When a data byte is received program execution jumps to the label "data_byte". The flag is then tested to ensure that the byte is, in fact, data and an exit is made from the subroutine if the flag is not set (ie the byte is discarded). The control register is loaded with the byte value and a pointer is set up which is used to retrieve the address of the routine which will implement the new audiometer setting. The routine address is then loaded into the start of the register table so that at the next zero crossing of the sine wave the routine will be executed to set up the new parameter. The UART subroutine finishes by disabling the secondary set of registers and returning to the main program module.

4.6 UART INITIALISATION SUBROUTINE

The UART initialisation subroutine is listed in Appendix 4C. This subroutine sets up the serial port parameters, configures the on-chip counter/timer, enables a hardware interrupt and resets the chip.

The entry point of the subroutine is at address "uart_set" and the first instructions set up mode registers one and two. The serial data transmission is set to an eight bit word with one stop bit and no parity. Hardware handshaking is configured by enabling the RTS and CTS lines. The clock select register is then set up to provide a 9600 baud rate for serial communication. The interrupt mask register is configured to provide hardware interrupts when the counter has reached zero and on receipt of a serial byte. The counter/timer is then loaded with an upper and lower byte (forming a sixteen bit word) which sets the interrupt period to 1 millisecond. Finally instructions are sent to the command register to reset the transmit and receive registers, assert the RTS line (to signal that the serial port is ready for data) and clear and enable the UART. A RTS instruction completes the subroutine.

5. PERFORMANCE MEASUREMENTS

5.1 INTRODUCTION

BS EN 60645-1, "Specification for audiometers" provides detailed performance specifications for pure tone audiometers. To assess the performance of the DSP audiometer the Audiometric Calibration facility of EMIS was utilised to make objective measurements of the parameters defined in BS EN 60645-1 for an air conduction audiometer. Before the tests were carried out the audiometer SPLs were adjusted to within ±0.5 dB of the reference SPLs at each frequency (ie the audiometer was calibrated). The following sections cover the frequency accuracy of the test tones, the signal purity (total harmonic distortion and noise), the attenuation accuracy and the tone envelope.

To carry out the tests the audiometer earphone was applied to a Bruel & Kjaer 4152 Reference Coupler. (The Reference Coupler is a form of "artificial ear" which couples the earphone under test to a 1.0" laboratory standard, condenser microphone via a 6 cubic cm cylindrical volume.) The electrical signal from the coupler microphone was then fed to a precision pre-amplifier and filter unit to boost the signal and limit the bandwidth to the measurement range of interest. The signal was then fed to a Philips PM6669 frequency counter to measure the fundamental frequency of the test signal. A Solartron 7151 computing multimeter was used to measure the dB SPL of the audiometer tones. Finally an Advantest R9211B spectrum analyser was used to produce the time and frequency domain plots which are reproduced in the following sections. The calibration of the Reference Coupler used for the measurements is directly traceable to the National Physical Laboratory, Teddington while the calibration of the measuring instruments can be traced, via UKAS accredited laboratories, to national standards.

5.2 FREQUENCY

Figure 5.1 provides a table of the frequency measurement results. The first column lists the standard audiometric frequencies from 125 to 8 000 Hz. The second column lists the measured frequencies to a resolution of six digits. The third column of the table calculates the percentage error between the measured and nominal frequencies. Inspection of this column reveals a consistent error of -0.002%. This error is almost insignificant when compared to the maximum tolerance of ±1.0% specified in BS EN 60645-1 for a clinical audiometer (a specification of ±3.0% is quoted for a screening audiometer). The frequency accuracy of the audiometer depends directly on the DAC clock rate and this, in turn, is derived from the DSP master clock. The crystal oscillator used in the DSP circuit has a specification of ±100 ppm (parts per million) worse case leading to a maximum frequency error of ±0.01%. The measured values are well within the crystal specification as the final column of the table details an error of -24 ppm maximum. The required accuracy is therefore designed into the circuit and it is not necessary to provide any adjustment to trim the output frequencies.

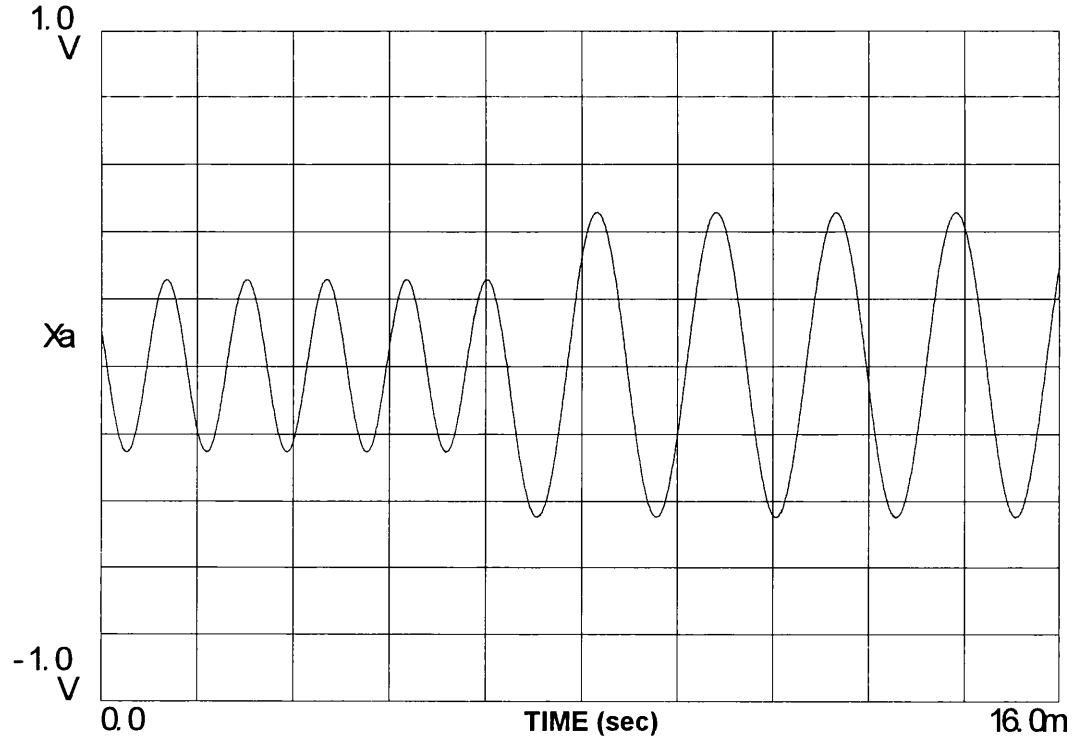
FIGURE 5.1 - FREQUENCY

Nominal	Measured	Error (%)	Error (ppm)
125	124.997	-0.002%	-24
250	249.995	-0.002%	-20
500	499.990	-0.002%	-20
750	749.985	-0.002%	-20
1000	999.980	-0.002%	-20
1500	1499.97	-0.002%	-20
2000	1999.96	-0.002%	-20
3000	2999.94	-0.002%	-20
4000	3999.92	-0.002%	-20
6000	5999.87	-0.002%	-22
8000	7999.83	-0.002%	-21

The DSP software routines are configured so that a change in frequency will only occur at the zero crossing point of the generated sine wave signal. This technique ensures that unwanted transients are kept to a minimum when the test frequency is changed. Figure 5.2 is a plot of the acoustical waveform measured from the earphone when the frequency is decreased from 1000 Hz to 750 Hz. A change in sound pressure level also occurs due to the difference in calibration offsets for each frequency. It can be seen that the change in frequency does occur at the zero

crossing point and that a totally seamless transition is made between the two different frequencies.

FIGURE 5.2 - CHANGE IN FREQUENCY



5.3 ATTENUATOR

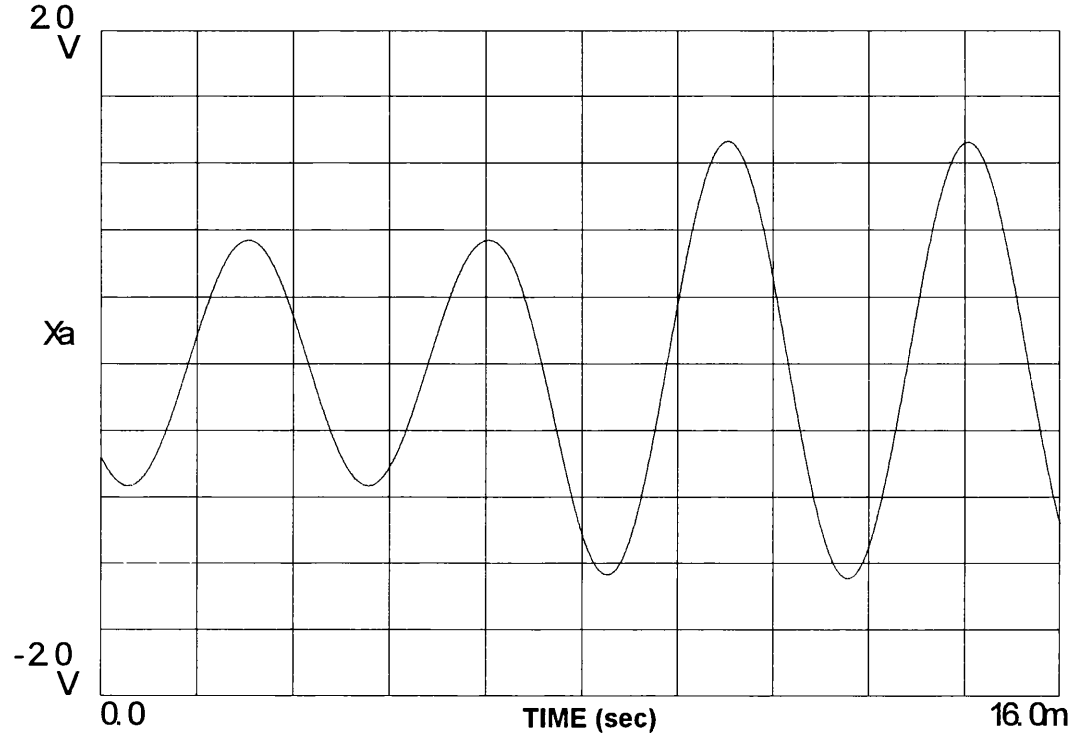
To measure the accuracy of the signal attenuation the difference in SPL produced by the test earphone was measured for each HL step. Figure 5.3 tabulates the results that were obtained for the three frequencies 125, 1000 and 4000 Hz. Measurements were also made at the other frequencies and similar results were obtained. It can be seen that the step error is within ±0.1 dB from maximum to 0 dB HL for all three frequencies. From 0 to -10 dB HL the error is within ±0.3 dB. BS EN 60645-1 specifies a maximum error of ±1.0 dB in any 5 dB step and so the results obtained comfortably meet this specification. The measurement system itself has an uncertainty of ±0.1 dB above 0 dB HL and ±0.3 dB below 0 dB HL (the increased uncertainty below 0 dB is due to the difficulty in accurately measuring the test tone at very low SPLs). Referring back to Figure 4.7 of section 4 the measured values agree with the theoretically calculated values detailed in the table.

FIGURE 5.3 - ATTENUATOR STEPS

Attenuator Step	Measured - 125Hz	Measured - 1kHz	Measured - 4kHz
120/115	---	4.9	4.9
115/110	---	4.9	4.9
115/110	---	5.0	5.0
110/105	---	5.0	5.0
105/100	---	5.0	5.0
100/95	---	5.0	5.0
95/90	---	5.0	5.0
90/85	---	5.0	5.0
85/80	5.0	5.0	5.0
80/75	5.0	5.0	5.0
75/70	5.0	5.1	5.0
70/65	5.0	4.9	5.0
65/60	5.1	5.0	5.0
60/55	4.9	5.0	5.0
55/50	5.0	5.0	5.0
50/45	5.0	5.0	5.0
45/40	5.0	5.0	4.9
40/35	5.0	5.0	5.1
35/30	5.0	5.0	5.0
30/25	5.0	5.0	5.0
25/20	5.0	5.0	5.0
20/15	5.0	5.0	5.0
15/10	5.0	4.9	5.0
10/5	5.0	5.1	5.0
5/0	5.0	5.1	5.0
0/-5	4.9	5.0	4.8
0/-10	4.8	4.9	4.7

In a similar manner to a change in frequency the DSP software ensures that a change in attenuator setting only occurs at the zero-crossing point of the sine wave. Figure 5.4 details the waveform of a 250 Hz tone when the attenuator setting changes from 80 to 85 dB HLS. It can be seen that the change between the two levels occurs at the zero crossing point of the sine wave and that there is no visible discontinuity in the waveform.

FIGURE 5.4 - CHANGE IN ATTENUATION



5.4 SIGNAL PURITY

The purity of a sine wave can be analysed by breaking the signal down into its individual frequency components. This technique is termed Fourier Analysis. The spectrum analyser utilised to record the spectral plots of this section makes use of the digital algorithm called the Fast Fourier Transform, FFT. The purity of the signal is assessed by measuring the Total Harmonic Distortion, or THD of the waveform. The harmonic distortion is defined as the unwanted frequency components present at whole number multiples of the fundamental frequency. Hence for a 1000 Hz signal the 2nd harmonic would occur at 2000 Hz, the 3rd harmonic at 3000 Hz etc. The THD is defined as the ratio of the power contained in these harmonics to the total power of the waveform. This ratio is most commonly expressed as a percentage although a dB factor is sometimes used. An ideal sine wave with no distortion products would therefore have a THD of 0%.

FIGURE 5.6 - 1 kHz SIGNAL SPECTRUM

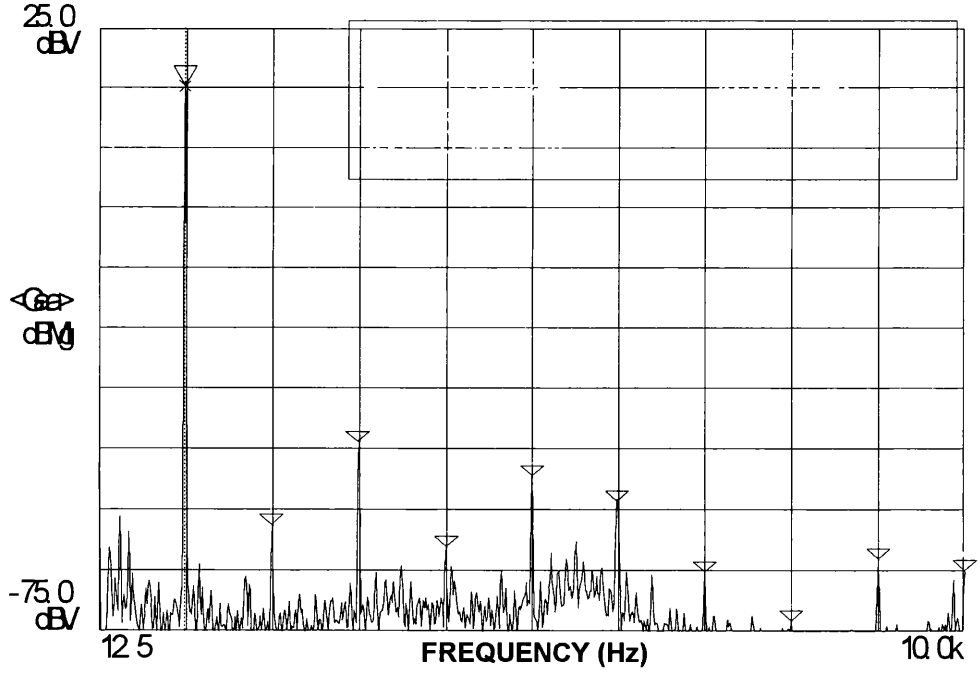


Figure 5.6 graphs the spectrum of a 1 kHz tone at a hearing level setting of 90 dB. Each of the harmonics is marked from the 2nd at 2 kHz to the 10th at 10 kHz. The spectrum analyser has an automatic THD calculating algorithm and the THD for this waveform is displayed as 0.13% (It is also displayed as a dB calculation beside "THP"). The hearing level of 90 dB is a worst case

setting at the bottom of the 40 dB range referred to in 3.2 and the measured THD of 0.13% is very close to the theoretical target of 0.1%.

FIGURE 5.5 - TOTAL HARMONIC DISTORTION

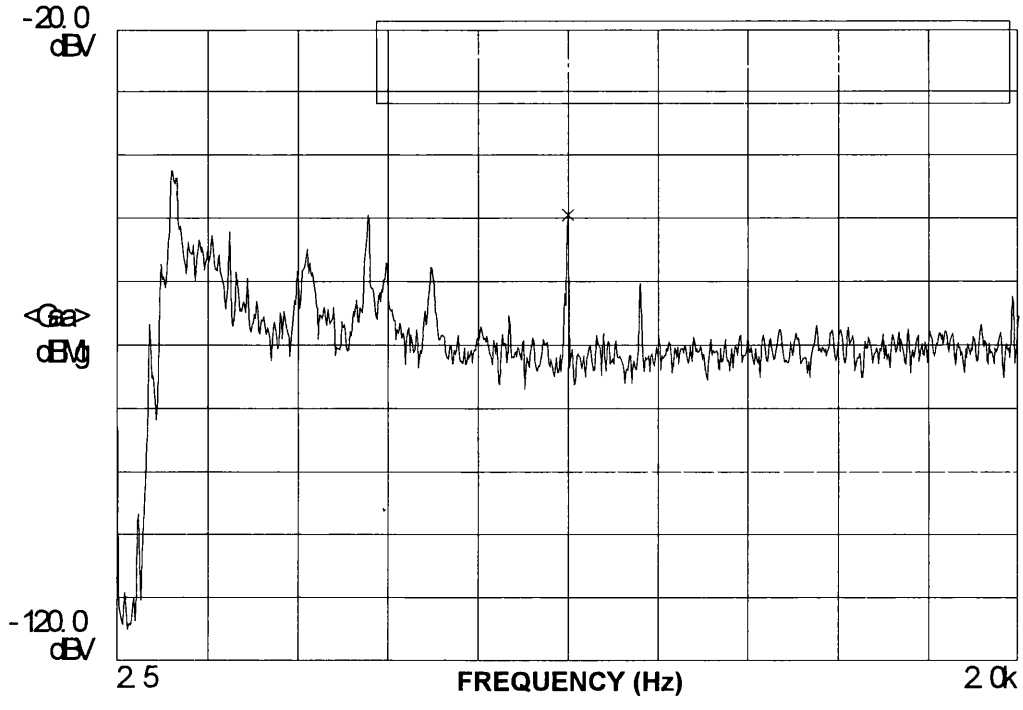
Attenuator	125 Hz THD (%)	1kHz THD (%)	4kHz THD (%)
120	---	0.28	0.35
110	---	0.09	0.18
100	---	0.04	0.09
90	---	0.13	0.06
80	0.11	0.02	0.03
70	0.05	0.02	0.02
60	0.05	0.06	0.05
50	0.06	0.18	0.17
40	0.03	0.53	0.42
30	0.03	---	---
20	0.08	---	---
10	0.28	---	---

In order to determine the distortion performance of the audiometer measurements of the output waveform were made at 10 dB attenuator intervals across the frequency range of the instrument. The table of Figure 5.5 lists the results that were obtained for the three frequencies 125, 1000 and 4000 Hz. The results follow a similar pattern for each frequency: at low HL the THD readings increase as the measurements start to reflect the noise level of the system (the noise signal is being interpreted as harmonics) while at high levels the readings again increase as the earphone (and to lesser extent the output amplifier) begins to distort. Between these two extremes the THD rises to a maximum of about 0.1%. BE EN 60645-1 specifies a maximum THD of 2% for a HL of 70 dB at 125 Hz and a HL of 110 dB at 1 kHz and 4 kHz. It can be seen that within these parameters the measured values do not exceed 0.2% which is a factor of ten better than the maximum specification.

In addition to the THD measurement it is important that the sine wave test signal is free from extraneous noise. The signal could be corrupted by mains hum or electrically generated amplifier noise for example. In order to investigate the output of the audiometer for noise the spectrum of a -10 dB was inspected at each output frequency. Figure 5.7 details the spectrum for the 1 kHz tone. The wide-band signal at approximately -20 dB relative to the amplitude of the tone is the noise floor of the measurement system. Apart from a few minor peaks in the noise spectrum there is no discernible hum or excess noise components in the waveform. The

slight peaking of the waveform at lower frequencies is due to the pickup of acoustically borne noise. A 250 Hz filter is utilised to minimise the effect of this noise at very low frequencies.

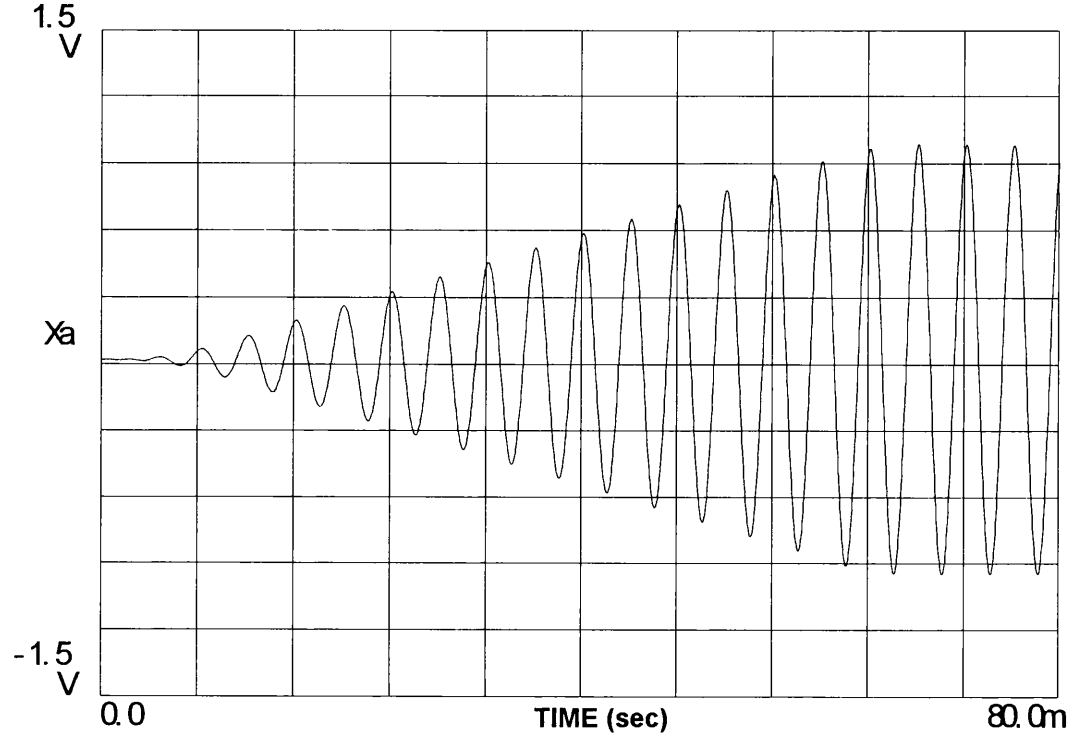
FIGURE 5.7 - -10 dB SIGNAL SPECTRUM



5.5 TONE ENVELOPE

Figure 5.8 details a plot of the rise portion of a 250 Hz tone envelope. The amplitude of the waveform is precisely controlled from minimum to maximum level. There are no discontinuities or overshoot discernible in the wavefrom. BS EN 60645-1 specifies a rise and fall time of between 0 and 250 ms for the tone envelope. It can be seen from figure 5.8 that the rise time is 60 ms. This agrees with the calculated figure which is set up in the DSP software. The tone envelope was also inspected at other frequencies and hearing levels and in all cases the accuracy of the envelope was found to be the same.

FIGURE 5.8 - TONE ENVELOPE



6 CONCLUSION

6.1 REVIEW OF PROJECT AIMS

Section 1.2 lists seven main aims which were set out at the start of the project. This section reviews each of these aims to ascertain if the prototype instrument meets the initial expectations.

The first aim was to implement as many of the audiometer functions through DSP processing as possible and reduce analog circuitry to a minimum. The DSP implements the circuit functions of sine wave oscillator, envelope generation and attenuation over an 80 dB range and eliminates the analog circuitry which would normally be required for these functions. The actual analog circuitry which is used in the prototype consists of five main blocks; DAC, filter, gain switching, amplifier and talk through. In order to reduce the existing component count one or more of these blocks would have to be simplified or eliminated. The DAC cannot be eliminated from the circuit as there has to be a means to convert the digital signal from the DSP to an analog waveform. However, use of a different type of converter could lead to a reduction in component count. Use of an over-sampling converter (where a high sample rate is utilised) or a delta-sigma device would simplify the filtering requirements by making it possible to incorporate the required single pole filter into the final amplifier stage. A converter with a higher resolution could be utilised (20 and 24 bit converters are currently available) and this could be used to obtain a greater attenuation range from the digital signal making it possible to reduce the gain switching from three to two ranges. A potential problem with this option would be the increased DSP processing required to generate the higher resolution word length, as the ADSP2105 is optimised for a 16 bit word size. A further consideration would be the cost of the higher resolution DAC. It is most likely that the extra cost would be greater than the cost of the components (including the saving in board space) which would be eliminated. The power amplifier could be simplified by replacing the discrete stage with an IC equivalent. However, as stated in section 3.2, the author was unable to locate an IC which performed satisfactorily in this part of the circuit. A possible option for component reduction would be to make use of an integrated device such as a SoundPort Codec (commonly used in personal computer soundcards). The Analog Devices AD1847 is one such device which contains 16-bit ADC and DAC converters, programmable gain and attenuation, filtering and interfacing circuitry on one chip. Such an IC could potentially replace a number of sections of the existing circuitry however, because of the complexity of such a device, significant changes would be required to the existing DSP routines. The audio performance of the device might also be unsatisfactory

(eg noise and distortion) as the IC is designed to a consumer audio specification. In summary it is felt that the analog circuit presented is a good compromise between component count, cost and performance.

The measurements presented in section 5 demonstrate that the aim of easily meeting the requirements of BS EN 60645-1 has been achieved in full. The use of the DSP achieves a high level of signal purity and accuracy.

The prototype audiometer meets the aim of being able to perform an automatic hearing test in a stand alone mode together with storage of the test results for printing or download via a serial link. There are a number of areas in which the automatic testing facility of the instrument could be developed. The facility to store a number of test results in non-volatile memory would be desirable. This would allow a series of hearing tests to be carried out and the results downloaded to a suitable PC for further analysis and archiving (the existing ASRA software could in fact be used for this purpose). Automatic categorisation of the audiogram would be a useful addition. The necessary calculations could be carried out by the DSP and the category either added to the audiogram printout or displayed on a suitable series of LED indicators. The automatic test could be expanded to include other test types such as a fixed frequency Bekesey test or an automatic screening test. It is likely that a commercially viable instrument would need to incorporate some or all of these features.

The facility of remote control of the instrument is provided via the serial interface. Section 4.2 gives details of the interface. During the development process a simple Visual Basic program was used to test the operation of the serial commands. This program implemented a "virtual audiometer" on the PC display which was used to control the operation of the DSP audiometer hardware. The operation of the instrument in this mode is similar to the existing ASRA audiometer and it therefor follows it should be possible to interface the DSP hardware to the ASRA software (modification would be required to the software).

The objective of reducing component count has been partly addressed in the discussion of the analog circuitry above. The DSP and digital circuitry was designed for minimum chip count and it is difficult to envisage how this could be reduced further. For example the UART chip chosen also includes a timer which would otherwise have required a second component to implement. A possible route would be to adopt a DSP with enhanced on-chip facilities such as the ADSP2181 from the Analog Devices family. This DSP has an additional serial port, parallel input/output ports, increased clock speed and larger program and data memory. This DSP could potentially eliminate three or four chips from the present circuit however, the cost

of the device is presently four times that of the ADSP2105. The extra cost of this device would therefor not justify its use, although falling prices may change this situation in the future.

An important aim of the project was to eliminate any manual adjustment or set-up of the circuitry. This has been fully realised by the total elimination of preset potentiometers from the design.

The final aim of the project was to allow easy set-up and calibration of the audiometer. The calibration process is straightforward, the calibration mode is entered by a single switch and the SPL can be adjusted in 1 dB steps. A possible enhancement to the calibration process would be to allow calibration via the serial port, this would allow a totally automatic calibration to be carried out by a suitably equipped audiometer calibration system. A further useful addition would be to add routines to allow the system variables and tables held in the EEPROM to be inspected and altered via the serial port. This would allow signal parameters such as tone rise/fall times, repetition rate and maximum HLS to be altered remotely. It could also provide the facility to trim the attenuator linearity at low HLS by modifying the offsets held in the look-up table in EEPROM.

6.2 PRACTICAL AUDIOMETERS

The development work outlined in this thesis has culminated in the design of a prototype DSP, air conduction audiometer. The design could be used as the basis for a number of different practical audiometers. A stand-alone manually operated instrument could be developed for educational and community screening use. An industrial screening instrument could be developed which carried out an automatic test and then either printed out the audiogram results or down-loaded the results to a PC for processing. The serial control facility of the design would also make it feasible to configure the instrument in a similar manner to the existing ASRA audiometer as a "black box" instrument controlled from a PC running suitable software. The bulk of the work required to realise these practical instruments would be concerned with the design of a suitable enclosure and control layout. It is hoped that if working prototypes can be produced it would be possible to manufacture the DSP audiometer in a similar manner to the existing ASRA instruments.

6.3 FURTHER DEVELOPMENTS

The project could be developed further to include the design of diagnostic and clinical audiometers. These instruments would require a second channel of hardware to drive an extra transducer. The instrument would also be required to produce a filtered noise test signal and speech waveforms to allow various additional audiological tests to be carried out. The present DSP would not have the processing power to implement all of these functions. However the Analog Devices 2100 DSP family has newer processors with larger program memory and faster execution speed which would be capable of implementing the extra algorithms required for diagnostic and clinical instruments. Software compatibility throughout the family of processors would allow the existing code to be used with little or no modification as the basis for the new instruments.

6.4 UPDATE TO AUDIOMETER REVIEW

At the end of the project a further review of the market was carried out to determine if DSP technology was being adopted by any of the audiometer equipment manufacturers. Two instruments were found which first became available in 1997; the Interacoustics AS216 and the Micro Audiometrics DSP Pure Tone Audiometer.

The Interacoustics AS216 is a portable, air conduction audiometer with a frequency range of 125 to 8000 Hz and -10 dB to 120 dB HL range. It can perform an automatic Hughson Westlake test and can either print out the audiogram on a suitable printer or download the thresholds to a suitable PC (Interacoustics have Windows database software for this use). The specification of this instrument is remarkably similar to the prototype audiometer presented in this thesis. However, the AS216 does not have the facility of remote control via the serial interface.

The Micro Audiometrics DSP Audiometer is a manual only, air conduction device with a 250 to 8000Hz frequency range and a -10 to 80 dB HL range. The device is extremely compact, measuring 150 x 100 x 25 mm in size, and is powered by an internal battery pack. Frequency and HL indication are by means of discrete LEDs and membrane push buttons control frequency, HL and left/right tone presentation. The audiometer has no external interfacing capabilities and has no facilities to store threshold results.

The ready availability and dropping cost of DSP technology makes it likely that increasing numbers of audiometer manufacturers will base new instrument designs around a DSP chip. There has been a rapid growth in the use of digital circuitry in hearing instruments in recent years. DSP and digitally programmable hearing instruments are currently available from most of the major manufacturers and it is most probable that, in a similar manner, digital technology will rapidly supersede analog techniques in audiometric instrumentation.

7 BIBLIOGRAPHY

1. ADS2105 Data Sheet, Analog Devices, Norwood, MA., 1990.

2. Analog Devices, ADSP-2100 Family User's Manual, Prentice Hall, Englewood Cliffs, N.J., 1994.

3. Analog Devices, Assembler Tools and Simulator Manual, Analog Devices, Norwood, MA., 1994.

4. Analog Devices, Conversion Handbook, Analog Devices, Norwood, MA., 1991.

5. Analog Devices, Digital Signal Processing Applications Using the ADSP2100 Family, Prentice Hall, Englewood Cliffs, N.J., 1992.

6. BS EN60645-1:1995, Audiometers - Part 1: Pure tone audiometers.

7. BS 2497:1992, Acoustics - Specification for standard reference zero for the calibration of pure tone air conduction audiometers.

8. Bekesy, G., Experiments in Hearing, McGraw-Hill, New York, 1960.

9. Berry, B. F., John, A. J., Shipton, M. S., A Computer-Controlled Audiometry System, Proc. Inst. Acoust., Spring Conf., 20.P.3, 1979.

10. Bolster, A. A., Computer Controlled Audiometry, Ph D Thesis, University of Glasgow, 1991.

11. BSA, Recommended Procedure - Computer Coding of Audiometric Thresholds, British Journal of Audiology, 29, 355-358, 1995.

12. BSA, Recommended Procedures for Pure-tone Audiometry Using a Manually Operated Instrument, British Journal of Audiology, 19, 281-282, 1985.

13. BSA, Recommended Format for Audiogram Forms, British Journal of Audiology, 23, 265-266, 1989.

14. Campbell, R.A., Computerised Audiometry, Journal of Speech and Hearing Research, 17 , 134-140, 1974.

15. Dadson, R.S., King, J. H., A Determination of Normal Threshold of Hearing and Its Relation to the Standardisation of Audiometers, J. Laryngol. Otol., 66, 366-378, 1952.

16. Fletcher, H., Speech and Hearing, MacMillan and Co Ltd, London, 1929.

17. Gardner, M. B., A Pulse-Tone Clinical Audiometer, J. Acoust. Soc. Am., 19, 592-599, 1947.

18. Harris, D., Microprocessor versus Self-Recording Audiometry in Industry, The Journal of Auditory Research, 19, 137-149, 1979.

19. Harris, D., A comparison of Computerised Audiometry by ANSI, Bekesy Fixed Frequency and Modified ISO Procedures in an Industrial Hearing Conservation Program, The Journal of Auditory Research, 20, 143-167, 1980.

20. Higgins, R. J., Digital Signal Processing in VLSI, Prentice Hall, Englewood Cliffs, N.J., 1990.

21. Hirsh, I. J., The Measurement of Hearing, McGraw-Hill, New York, 1952.

22. Hood, J. L., The Art of Linear Electronics, Butterworth-Heinemann Ltd, Oxford, 1993.

23. ISO 8253-1:1989, Acoustics - Audiometric test methods - Part 1: Basic pure tone air and bone conduction threshold audiometry.

24. Jung, W. G., IC Op Amp Cookbook, Howard Sams, Indianapolis, 1986.

25. Katz, Jack, Handbook of Clinical Audiology, Williams and Wilkins, Baltimore, MD., 1978

26. Kinsler, L. E., Frey, A. R., Fundamentals of Acoustics, John Wiley and Sons, New York, 1962.

27. Lockhart, G. B., Cheetham, B. M. G., BASIC Digital Signal Processing, Butterworth, Oxford, 1989.

28. Newby, H. A., Popelka, G. R., Audiolgy, 5th Edition, Prentice Hall, 1985.

29. Pelmear, P. L., Hughes, B. J., Self-recording Audiometry in Industry, British Journal of Industrial Medicine, 31, 304-309, 1974.

30. Pohlmann, K. C., Principles of Digital Audio, Howard Sams, Indianapolis, 1989.

31. Putnam, B. W., RS-232 Simplified, Prentice-Hall, Englewood Cliffs, 1987.

32. Reger, S. N., A Clinical and Research Version of the Bekesy Audiometer, Laryngoscope, 62, 1333-1351, 1952.

33. Robinson, D. W., Shipton, M. S., Whittle, L. S., Audiometry in Industrial Hearing Conservation Part I and II, NPL Acoustics Report Ac64 and Ac71, 1975.

34. Shipton, M. S., Manual Audiometry: A Comparison of Two Methods, NPL Acoustics Report Ac88, 1978.

35. Tyler, R. S., Wood, E. J., A Comparison of Manual Methods for Measuring Hearing Levels, Audiology, 19, 316-329, 1980.

36. Wilkinson, B., Digital System Design, Prentice-Hall, Englewood Cliffs, 1992.

37. Van Valkenburg, M. E., Analog Filter Design, Holt-Saunders, New York, 1982.

APPENDIX 1A - DSP AND DIGITAL CIRCUIT DIAGRAM

PSU
Appdx 1D

+5V

0V

XTAL1
OSCH0D

U4b
74ACT74

R3
10K

Clk

SData
SWord
SClk

ANALOG
CIRCUIT
APPDX 1B

U1
ADSP2105

XTL CkI CkO G1 G2 G3 G4 V1 V2 V3 DT TF RF DR Ck
RESET
IRQ2
BR
BG
FLAG
PMS
DMS
BMS
WR
RD
NC
A13 A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0
D23 D21 D19 D17 D15 D13 D11 D9 D7 D5 D3 D1
D22 D20 D18 D16 D14 D12 D10 D8 D6 D4 D2 D0

U2
27C256

Vpp OE Vcc
A14 A13 A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0
D7 D6 D5 D4 D3 D2 D1 D0
CE 0V

U3
2816

WE OE Vcc
A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0
D7 D6 D5 D4 D3 D2 D1 D0
CE 0V

U10b
74ACT32
GND
WR
VCC

U11
TL7705

R1
10K

R2
10K

SW2
C1 C2

C2
100nF

C1
100nF

R4
10K

U4a
74ACT74

U8
74ACT138

G1 G2A G2B A B C
Y0 Y1 Y2 Y3 Y4 Y5 Y6 Y7
GND VCC

U6
SCC2691

RDN WRN Vcc
A2 A1 A0
D7 D6 D5 D4 D3 D2 D1 D0
RESET X1 X2 TxD RxD MPO MPI
INT CEN GND

C3
22pF

XTAL2
3.6864M

C4
22pF

SW1
SPDT
C NC NO

R8
10K

LED1
LED

R5
470R

U7
74ACT374

U9
74ACT373

1D 2D 3D 4D 5D 6D 7D 8D
1Q 2Q 3Q 4Q 5Q 6Q 7Q 8Q
G CK OC
VCC GND

-80
-40
Filt
Talk
Step
Right
Left
Up/Dn

ANALOG
CIRCUIT
APPDX 1B

U10c
74ACT32

RD

U10d
74ACT32

WR

U10a
74ACT32

R6
1K

R7
10K

SK1
STEREOSKT
Ring
Tip
Gnd

CTS
RTS
Rx
Tx

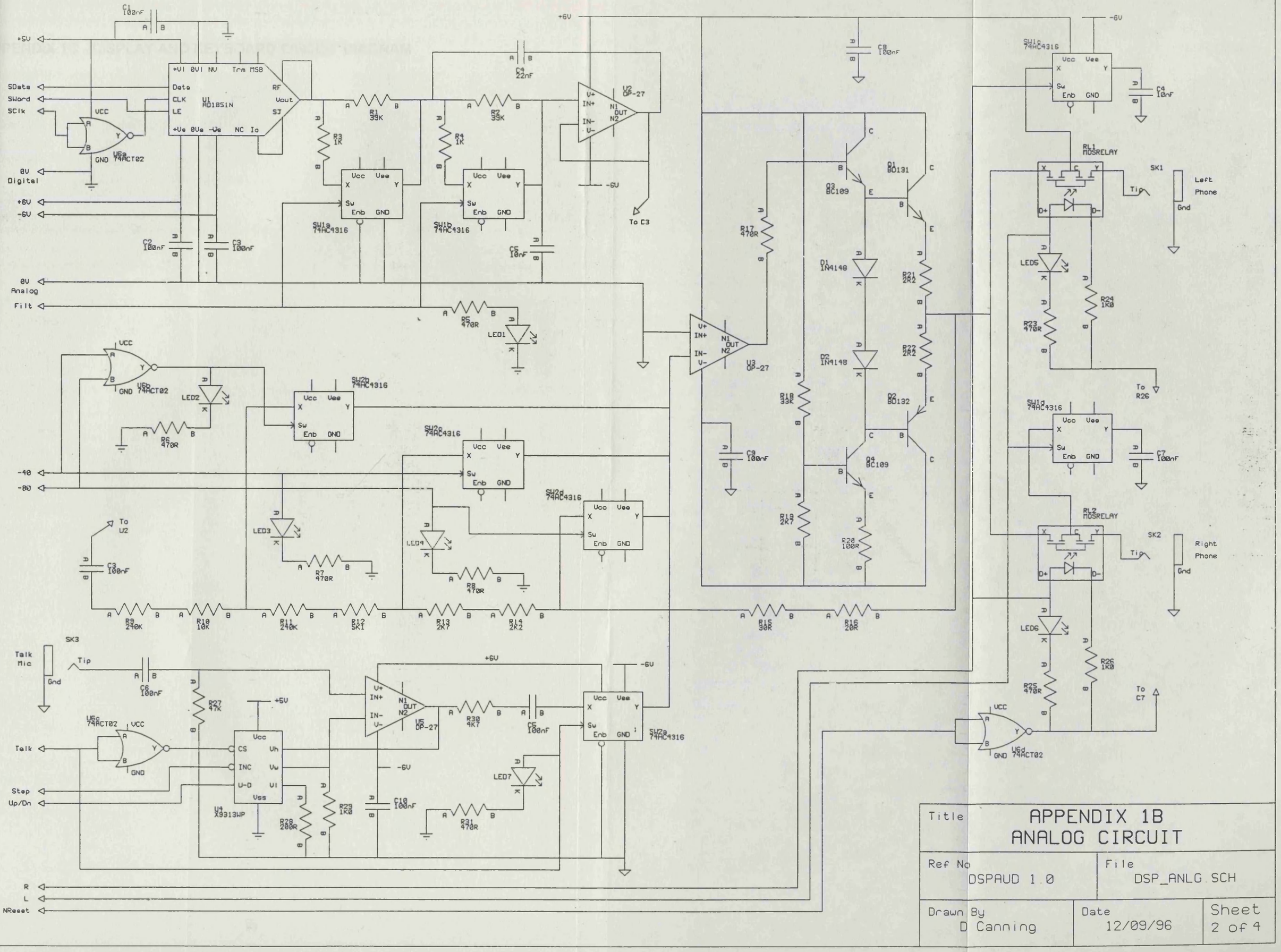
RS232
Appdx 1D

A0 RD WR Reset CS D15 D14 D13 D12 D11 D10 D9 D8 Clk/2

DISPLAY AND KEYBOARD APPDX 1C

Title	APPENDIX 1A DSP AND DIGITAL CIRCUIT	
Ref No DSPAUD1 0	File DSPAUD SCH	
Drawn By D Canning	Date 12/09/96	Sheet 1 of 4

APPENDIX 1B - ANALOG CIRCUIT DIAGRAM

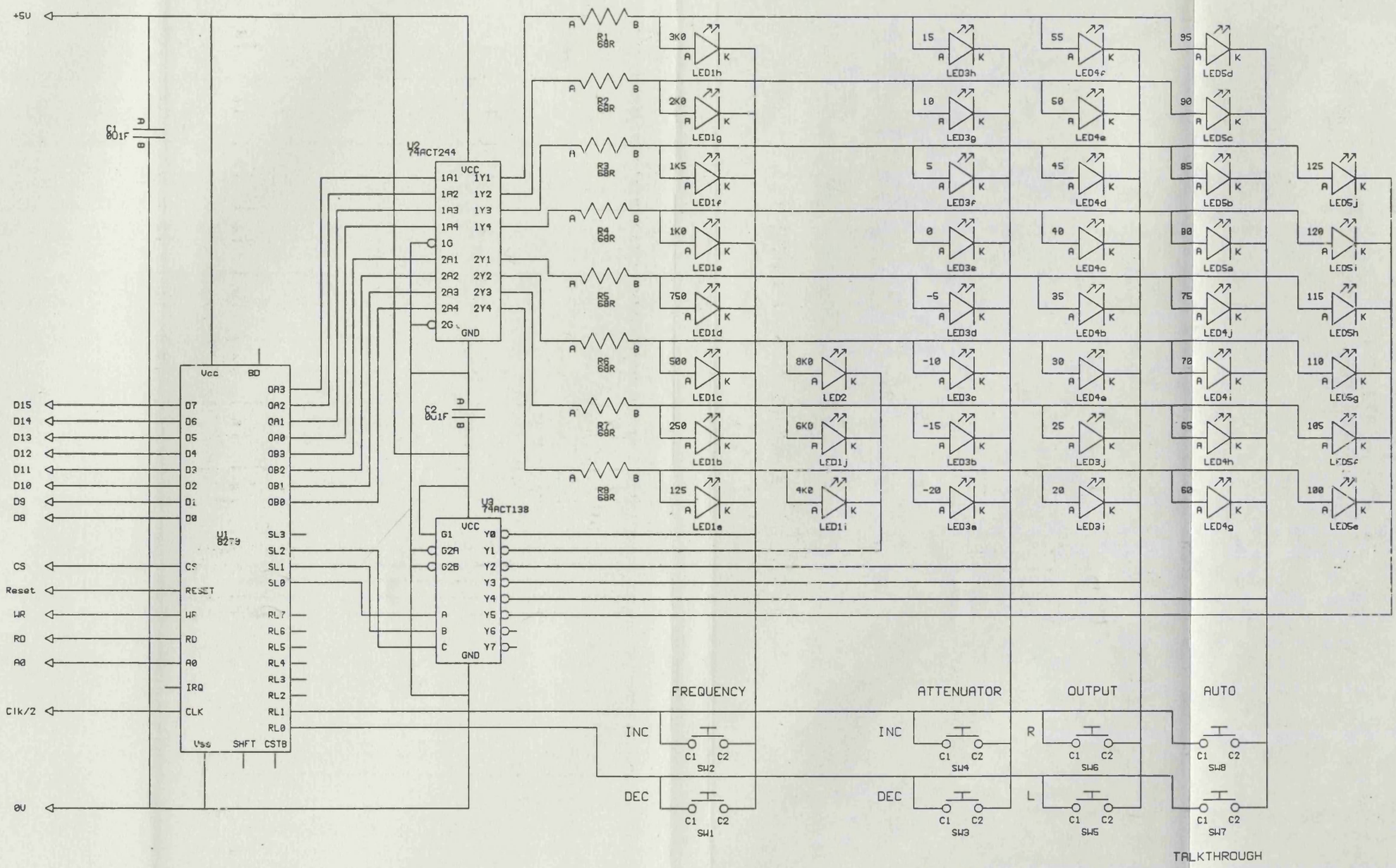


Title	APPENDIX 1B ANALOG CIRCUIT	
Ref No DSPAUD 1.0	File DSP_ANLG.SCH	
Drawn By D Canning	Date 12/09/96	Sheet 2 of 4

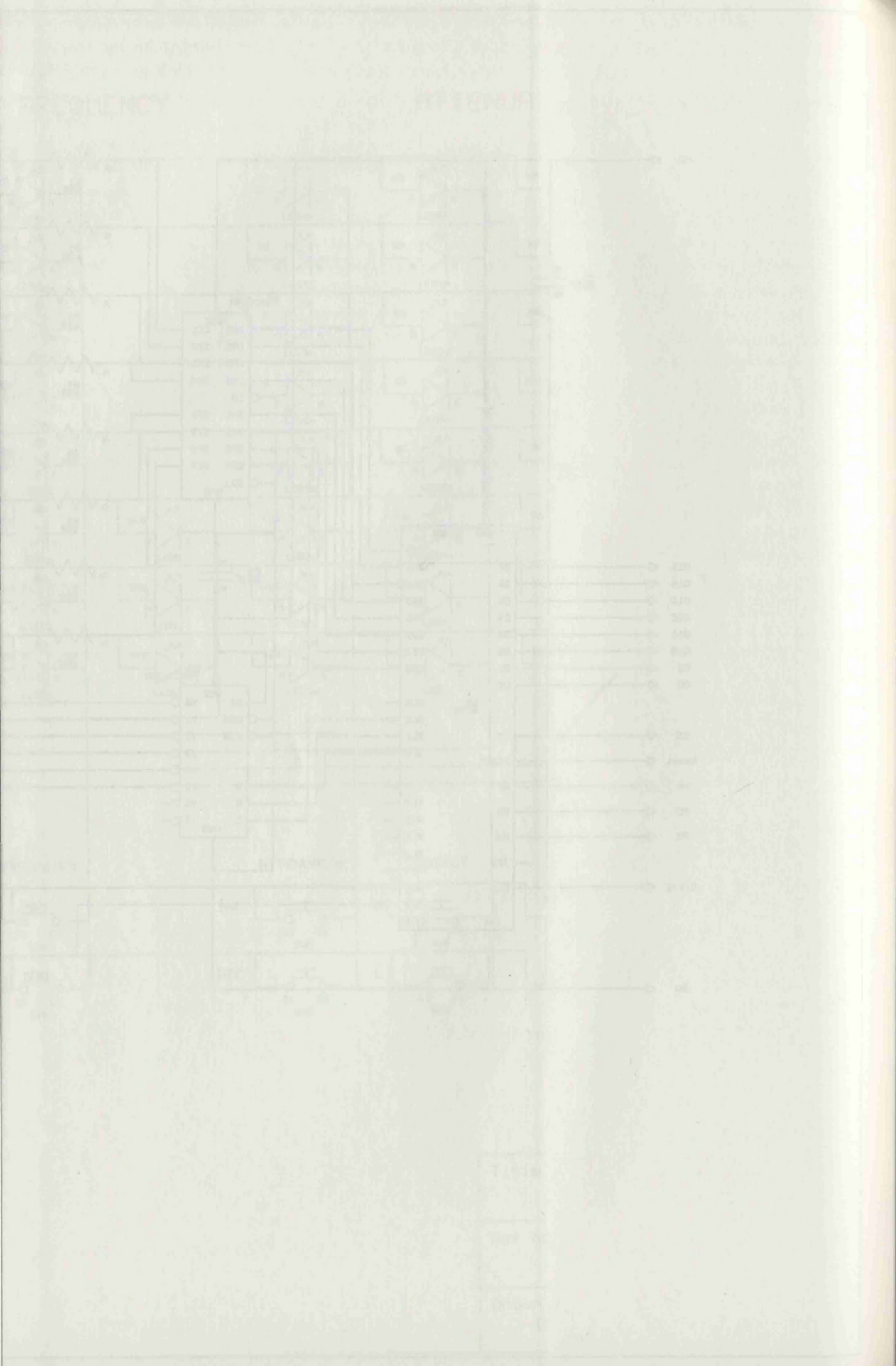
APPENDIX 1C - DISPLAY AND KEYBOARD CIRCUIT DIAGRAM

FREQUENCY

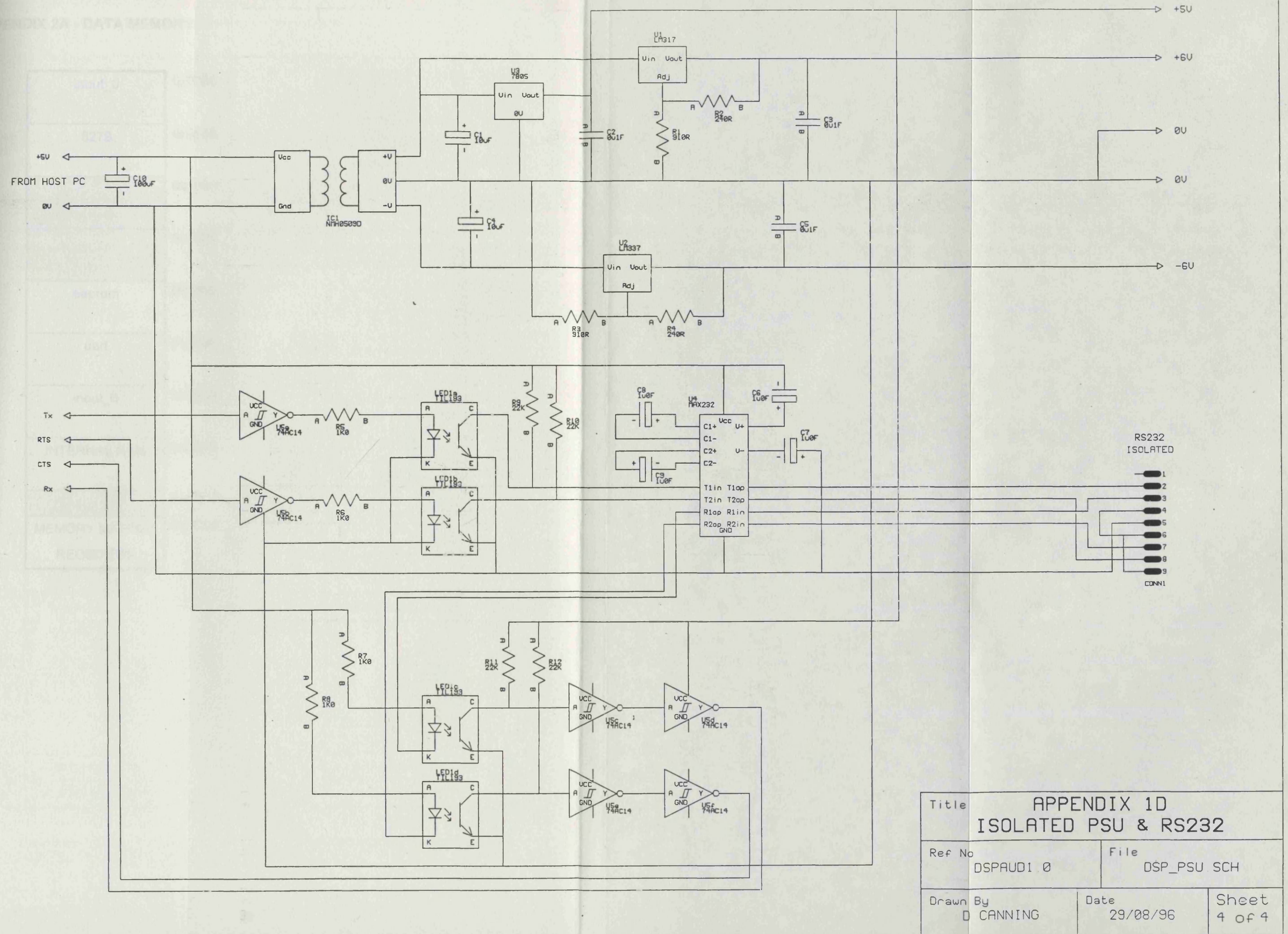
ATTENUATOR



Title	APPENDIX 1C DISPLAY AND KEYBOARD	
Ref No DSPAUD1 0	File DSPDISP.SCH	
Drawn By D CANNING	Date 20/08/96	Sheet 3 of 4

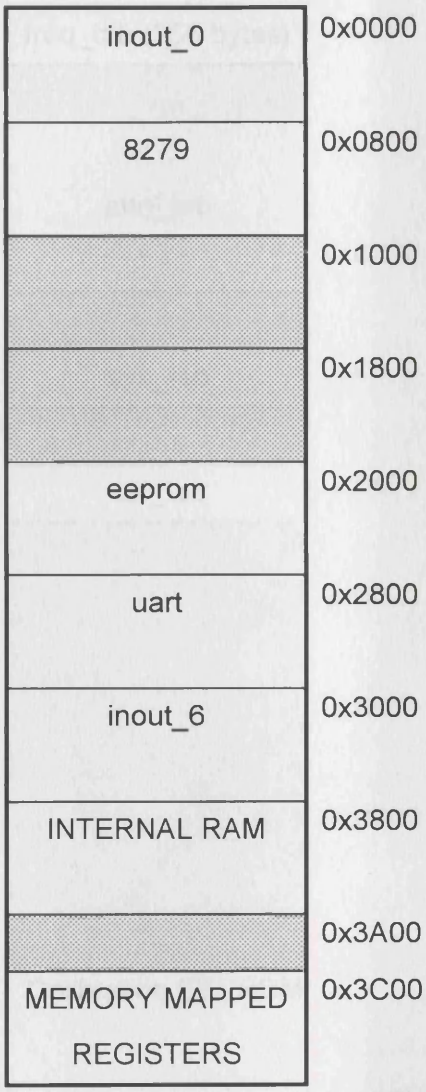


APPENDIX 1D - POWER SUPPLY AND ISOLATED RS232 CIRCUIT DIAGRAM

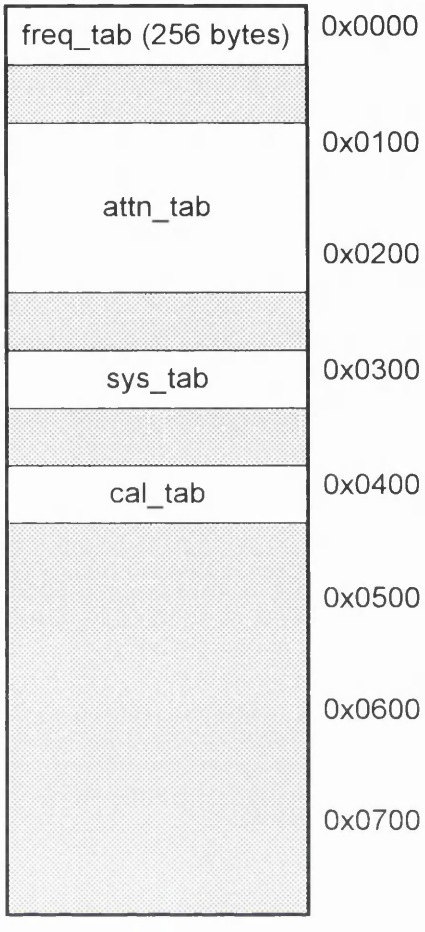


Title	APPENDIX 1D ISOLATED PSU & RS232
Ref No DSPAUD1 0	File DSP_PSU SCH
Drawn By D CANNING	Date 29/08/96
Sheet 4 of 4	

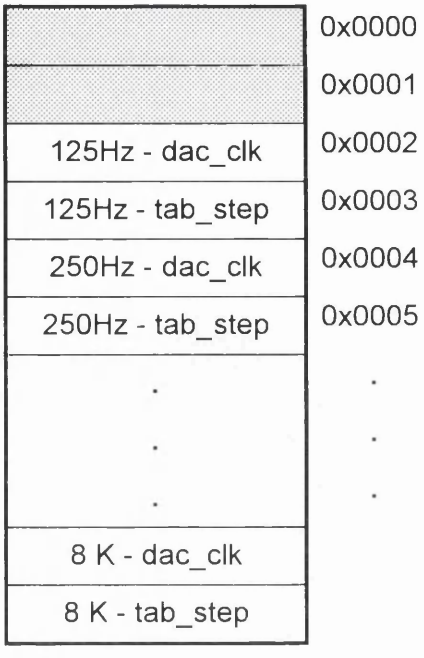
APPENDIX 2A - DATA MEMORY



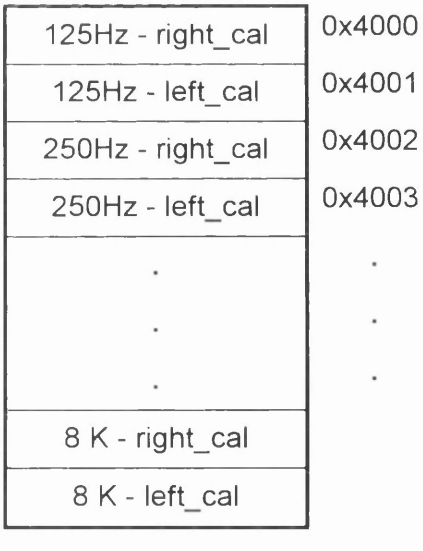
APPENDIX 2B - EEPROM MEMORY



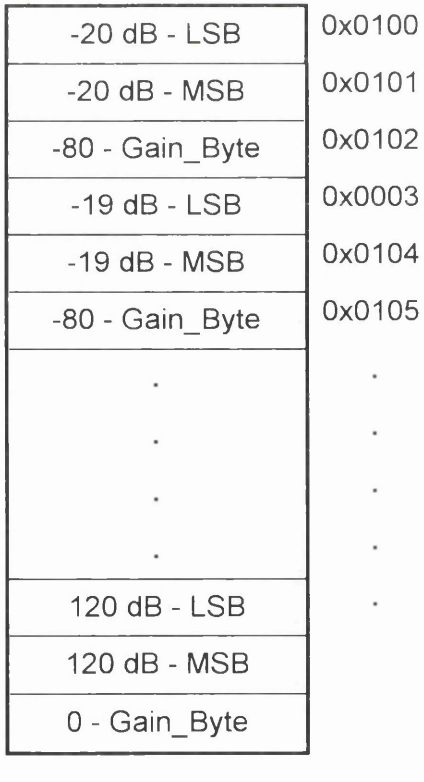
Calibration EEPROM



Frequency Control Table



Calibration Offsets Table



Attenuator Control Table

APPENDIX 3 - SYSTEM SPECIFICATION FILE

```
{====================SYSTEM SPECIFICATION - DMC2105====================}

.SYSTEM dmc2105;                                    {System name}
.ADSP2105;                                          {Processor type}
.MMAP0;                                             {State of MMAP pin}
.SEG/ROM/BOOT=0 boot_mem0[1024];                    {Four pages of BOOT memory}
.SEG/ROM/BOOT=1 boot_mem1[1024];
.SEG/ROM/BOOT=2 boot_mem2[1024];
.SEG/ROM/BOOT=3 boot_mem3[1024];
.SEG/PM/RAM/ABS=0/CODE/DATA int_pm[1024];           {1K internal prog memory}
.SEG/DM/RAM/ABS=14336/DATA int_dm[512];             {1/2K internal data memory}
.SEG/DM/RAM/ABS=8192/DATA eeprom_dm[2048];          {2K external EEPROM}
.PORT/DM/ABS=0 inout_0;                             {In/out port}
.PORT/DM/ABS=2048 disp_data;                        {8279 data register}
.PORT/DM/ABS=2049 disp_cmd;                         {8279 command register}
.PORT/DM/ABS=10240 uart_mr12;                       {UART mode register}
.PORT/DM/ABS=10241 uart_sr;                         {UART status register}
.PORT/DM/ABS=10242 uart_cr;                         {UART command register}
.PORT/DM/ABS=10243 uart_data;                       {UART data register}
.PORT/DM/ABS=10244 uart_acr;                        {UART auxiliary register}
.PORT/DM/ABS=10245 uart_int;                        {UART interrupt register}
.PORT/DM/ABS=10246 uart_ctu;                        {UART counter/timer upper}
.PORT/DM/ABS=10247 uart_ctl;                        {and lower registers}
.PORT/DM/ABS=12288 inout_6;                         {In/out port}
.ENDSYS;
```

APPENDIX 4A - MAIN PROGRAM LISTING

```
{======================MAIN AUDIOMETER PROGRAM=========================}

.MODULE/RAM/ABS=0       aud_main;               {Program Name}

{PORT commands set up addresses of hardware from DMC2105 architecture file}

.PORT inout_0;                                  {Data Latches..}
.PORT inout_6;

.PORT disp_data;                                {8279 Display Chip..}
.PORT disp_cmd;

.PORT uart_mr12;                                {SCC2691 UART..}
.PORT uart_sr;
.PORT uart_cr;
.PORT uart_data;
.PORT uart_acr;
.PORT uart_int;
.PORT uart_ctu;
.PORT uart_ctl;

{VAR commands name variables and places them in data or program memory..}

.VAR/DM/RAM/CIRC/SEG=int_dm   sin_tab[384];     {Sine wave sample table}
.VAR/DM/RAM/SEG=int_dm        reg_tab[16];      {Main Control Registers}
.VAR/DM/RAM/SEG=int_dm        sub_tab[16];      {Subroutine adresses}
.VAR/DM/RAM/SEG=int_dm        audg_tab[25];     {Audiogram table for auto}
.VAR/PM/RAM/CIRC/ABS=0x0380   rand_tab[20];     {Random table for auto}
.VAR/DM/RAM/CIRC              thrsh_tab[3];     {Threshold table for auto}
.VAR/PM/RAM                   audg_titl[22];    {Text strings to create}
.VAR/PM/RAM                   audg_crlf[3];     {audiogram print out..}
.VAR/PM/RAM                   audg_space[3];
.VAR/PM/RAM                   audg_r[6];
.VAR/PM/RAM                   audg_l[6];
.VAR/PM/RAM                   audg_freq[60];
.VAR/PM/RAM                   audg_attn[124];
.VAR/DM/RAM/ABS=0x2000        freq_tab[32];     {Frequency table in EEPROM}
.VAR/DM/RAM/ABS=0x2100        attn_tab[465];    {Atten table in EEPROM}
.VAR/DM/RAM/ABS=0x2400        cal_tab[32];      {Calib table in EEPROM}
.VAR/DM/RAM/ABS=0x23C0        auto_tab[50];     {Auto table in EEPROM}

.VAR/DM/RAM/SEG=int_dm        cont_nib;
.VAR/DM/RAM/SEG=int_dm        step_nib;
.VAR/DM/RAM/SEG=int_dm        delay_count;
.VAR/DM/RAM/SEG=int_dm        env_incr;
.VAR/DM/RAM/SEG=int_dm        cal_off;
.VAR/DM/RAM/SEG=int_dm        resp_flag;
.VAR/DM/RAM/SEG=int_dm        mS_count;
.VAR/DM/RAM/SEG=int_dm        max_resp;
.VAR/DM/RAM/SEG=int_dm        auto_inc;
.VAR/DM/RAM/SEG=int_dm        audg_pntr;
.VAR/DM/RAM/SEG=int_dm        auto_pntr;
.VAR/DM/RAM/SEG=int_dm        tone_off;
.VAR/DM/RAM/SEG=int_dm        puls_len;
.VAR/DM/RAM/SEG=int_dm        auto_len;
.VAR/DM/RAM/SEG=int_dm        auto_off;

{The INIT command initialises the appropriate variables..}

.INIT                   sin_tab:    <SIN_TAB.DAT>;
.INIT                   rand_tab:   <RAND_TAB.DAT>;
.INIT                   audg_titl:  '  PATIENT AUDIOGRAM',0;
```

```
.INIT                   audg_crlf:  0x000D00,0x000A00,0;
.INIT                   audg_space: '  ',0;
.INIT                   audg_r:     'RIGHT',0;
.INIT                   audg_l:     'LEFT ',0;
.INIT                   audg_freq:  '    ',0,' 125',0,' 250',0,' 500',0,'
750',0,'1000',0,'1500',0,'2000',0,'3000',0,'4000',0,'6000',0,'8000',0;

.INIT                   audg_attn:  '-20',0,'-15',0,'-10',0,' -5',0,'
0',0,'  5',0,' 10',0,' 15',0,' 20',0,' 25',0,' 30',0,'35',0,'40',0,'45',0,'
50',0,' 55',0,' 60',0,' 65',0,' 70',0,'75',0,'80',0,'85',0,'90',0,'
95',0,'100',0,'105',0,'110',0,'115',0,'120',0,'125',0,'N/T',0;

{The GLOBAL command extends variable use to subroutines..}

.GLOBAL                 reg_tab, sub_tab, freq_sub, attn_sub, tint_sub;
.GLOBAL                 oput_sub, auto_sub, inout_0, inout_6;
.GLOBAL                 resp_flag, mS_count, max_resp, auto_inc, audg_pntr;
.GLOBAL                 auto_pntr, tone_off, puls_len, auto_len, auto_off;
.GLOBAL                 uart_mr12, uart_sr, uart_cr, uart_data, uart_acr;
.GLOBAL                 uart_int, uart_ctu, uart_ctl;

{The EXTERNAL command defines subroutine names..}

.EXTERNAL               uart_set, uart;

{*****************************Interrupt Vectors*****************************}

                JUMP restart;           {Hardware reset - jump to start}
                nop; nop; nop;          {of program}

                CALL uart;              {IRQ2 Interrupt line from UART}
                RTI;
                nop; nop;

return:         RTI;                    {SPORT0 transmit not available}
                nop; nop; nop;

                RTI;                    {SPORT0 receive not available}
                nop; nop; nop;

                JUMP sample;            {SPORT1 transmit-sample for DAC}
                nop; nop; nop;

                RTI;                    {SPORT1 receive not used}
                nop; nop; nop;

                JUMP envelope;          {TIMER used for envelope routine}
                nop; nop; nop;


{*****************************Start of Program*****************************}

restart:        CALL uart_set;          {Initialise UART}

                AR = 0x003F;            {Initialise 8279..}
                DM(disp_cmd) = AR;      {Program scan frequency}
                AR = 0x0002;
                DM(disp_cmd) = AR;      {Program N-Key rollover}
                AR = 0x00C1;
                DM(disp_cmd) = AR;      {Clear all}

{Initialise Data Address Registers..}

                I0 = ^reg_tab;          {point to Control Registers}
                I1 = ^thrsh_tab;        {point to thrsh_tab}
```

```
                I4 = ^sin_tab;          {point sin_tab start}
                I7 = ^rand_tab;         {point to rand_tab}
                M0 = 1;                 {increment by 1}
                M1 = 0;                 {no increment}
                M5 = 0;                 {no increment}
                M7 = 1;                 {increment by 1}
                L0 = 0;                 {Not circular buffer}
                L1 = %thrsh_tab;        {length of thrsh_tab}
                L2 = 0;
                L3 = 0;
                L4 = %sin_tab;          {length of sin_tab}
                L5 = 0;
                L6 = 0;
                L7 = %rand_tab;         {length of rand_tab}

{Load Subroutine Addresses into Sub Tab..}

                I0 = ^sub_tab;          {point to table}
                DM(I0,M0) = 0x0000;     {system routine}
                DM(I0,M0) = ^freq_sub;  {stimulus routine}
                DM(I0,M0) = ^freq_sub;  {frequency change routine}
                DM(I0,M0) = ^attn_sub;  {attenuator change routine}
                DM(I0,M0) = ^attn_sub;  {attenuator fine routine}
                DM(I0,M0) = ^tint_sub;  {interrupter routine}
                DM(I0,M0) = ^oput_sub;  {output routine}
                DM(I0,M0) = ^return;    {patient response}
                DM(I0,M0) = ^return;    {patient response}
                DM(I0,M0) = ^attn_sub;  {attn change routine II}
                DM(I0,M0) = ^attn_sub;  {attn fine routine II}
                DM(I0,M0) = ^tint_sub;  {interrupter routine II}
                DM(I0,M0) = ^oput_sub;  {output routine II}
                DM(I0,M0) = ^return;    {common}
                DM(I0,M0) = ^auto_sub;  {auto tests}
                DM(I0,M0) = ^return;    {response}

{Initialise 2105 Control Registers..}

                I0 = 0x3FF1;            {SCLKDIV}
                DM(I0,M0) = 5;          {Set 48kHz sample rate}
                DM(I0,M0) = 0x4A0F;     {SPORT1 initialise}
                I0 = 0x3FFB;            {TIMER register}
                DM(I0,M0) = 0;          {TSCALE =0}
                DM(I0,M0) = 49;         {TCOUNT = 49}
                DM(I0,M0) = 49;         {TPERIOD = 49}
                ENA TIMER;              {Start Timer}
                DM(I0,M0) = 0x00C0;     {DWAIT2 = 3 for EEPROM}
                DM(I0,M0) = 0x0C18;     {PWAIT=0,BWAIT=3,PORT1 SER}
                TX1 = 0x0000;           {Prime SPORT1 with 0}
                AY0 = I4;               {sin_tab start}
                ENA AR_SAT;             {Set saturation mode}
                ICNTL = 0x00;           {Disable interrupt nesting}
                IFC = 0x003F;           {Clear all interrupts}
                IMASK = 0x0025;         {Enable IRQ2, SPORT1, TIMER}

{Initial Tone Settings..}

                DM(I0,M0) = ^return;
                DM(I0,M0) = 0x0005;     {set initial stimulus}
                DM(I0,M0) = 0x0005;     {set initial frequency - 1K}
                DM(I0,M0) = 0x000A;     {set initial attenuator - 30dB}
                DM(I0,M0) = 0x0000;     {set initial attenuator fine}
                DM(I0,M0) = 0x0040;     {set initial tone on}
                DM(I0,M0) = 0x0020;     {set initial o/p left}
                DM(auto_len) = M1;      {disable auto}
                DM(auto_off) = M1;      {disable auto}
```

```
                PUSH STS;
                CALL oput_sub;
                PUSH STS;
                CALL tint_sub;
                MX1 = 0;

                AR = 1000;              {set max response time to 1sec}
                DM(max_resp) = AR;

{Main Program Loop..}

mainloop:       IDLE;                   {wait for interrupt}
                JUMP mainloop;

{*****************************Envelope Routine*****************************}

envelope:       AY1 = MX1;              {Get current envelope value}
                AR = 0x7FFB;            {Check if nearly at maximum}
                AR = AR - AY1;
                IF NE JUMP envinc;      {If not jump}
                AR = 0x0090;
                DM(uart_cr) = AR;       {Stop UART counter}
                AR = PASS 0;
                DM(mS_count) = AR;      {Set mS_count to 0}
                DM(reg_tab + 7) = AR;   {Clear response time registers}
                DM(reg_tab + 8) = AR;
                AR = 0x0080;
                DM(uart_cr) = AR;       {Start UART counter}
envinc:         AX1 = DM(env_incr);     {Get envelope increment}
                AR = AX1 + AY1;         {Calculate new envelope value}
                IF AV AR = PASS AR;     {If overflow clear flags}
                IF LT AR = PASS 0;      {If < 0 then set to 0}
                MX1 = AR;               {Save new envelope value in MX1}
                RTI;                    {finished}

{*****************************Sample Routine*****************************}

sample:         MY1 = DM(I4,M4);        {Get sample from sin_tab}
                MF = MX1 * MY1 (RND);   {Multiply by envelope}
                MR = MX0 * MF (RND);    {Multiply by amplitude}
                TX1 = MR1;              {Output to DAC}

                AR = DM(delay_count);   {Get delay count}
                AF = PASS AR;
                IF EQ JUMP change;      {If = 0 then jump}
                AF = AF - 1;            {Decrement count}
                IF GT JUMP change;      {If > 0 then jump}
                AR = DM(step_nib);      {Get step nibble}
                AY1 = DM(cont_nib);     {Get control nibble}
                AR = AR OR AY1;         {Combine to form output byte}
                DM(inout_6) = AR;       {Output to port}
                POP STS;
                IMASK = 0x0025;         {Enable TIMER interrupt}
                PUSH STS;               {on exit from this routine to}
                IMASK = 0x0000;         {start envelope}
change:         AR = PASS AF;
                DM(delay_count) = AR;   {Update delay count}
                AR = MX1;               {Get current envelope value}
                AF = PASS AR;
                IF NE JUMP key_chk;     {If <> 0 then jump}
                AR = DM(cont_nib);      {Get control nibble}
                AF = PASS AR;
                IF EQ JUMP key_chk;     {If outputs off then jump}
                AR = PASS 0;
                DM(cont_nib) = AR;      {Otherwise set to 0}
```

```
                AR = 200;               {and setup delay}
                DM(delay_count) = AR;

{This part of the routine checks the 8279 for a key press..}

key_chk:        AR = 0x0050;            {program 8279 to read FIFO}
                DM(disp_cmd) = AR;
                AR = 0x0007;            {Set up mask}
                AF = PASS AR;
                AR = DM(disp_cmd);      {Get key data}
                AR = AR AND AF;
                IF EQ JUMP check;       {Jump if no characters in FIFO}

{Each set of buttons are now checked starting with the Frequency INC/DEC..}

                AR = DM(disp_data);
                AY1 = 193;              {Key code for frequency INC}
                AR = AR - AY1;
                IF GT JUMP key_attn;    {Jump if not frequency INC/DEC}
                AF = PASS AR;
                IF EQ AF = AF + 1;      {If INC set to +1}
                AR = DM(reg_tab + 2);   {Get current frequency}
                AR = AR + AF;           {Increment or decrement}
                IF EQ AR = PASS 1;      {Make sure lowest frequency is 1}
                DM(reg_tab + 2) = AR;   {Store new frequency}
                AR = ^freq_sub;
                DM(reg_tab) = AR;       {Point to frequency subroutine}

{Check for attenuator INC/DEC in same manner as frequency..}

key_attn:       AY1 = 16;
                AR = AR - AY1;
                IF GT JUMP key_rl;
                AF = PASS AR;
                IF EQ AF = AF + 1;
                AR = DM(reg_tab + 3);
                AR = AR + AF;
                DM(reg_tab + 3) = AR;
                AR = ^attn_sub;
                DM(reg_tab) = AR;

{Check for Right/Left keypress in same manner..}

key_rl:         AY1 = 8;
                AR = AR - AY1;
                IF GT JUMP auto;
                AF = PASS AR;
                AY1 = 32;
                IF LT AR = PASS AY1;
                DM(reg_tab + 6) = AR;
                AR = ^oput_sub;
                DM(reg_tab) = AR;

{Check for AUTO keypress in same manner..}

auto:           AY1 = 8;
                AR = AR - AY1;
                IF GT JUMP check;
                AF = PASS AR;
                AY1 = 32;
                IF LT AR = PASS AY1;
                AR = DM(auto_off);
                AR = PASS AR;
                AR = 0;
                IF EQ AR = PASS AY1;
```

```
                DM(reg_tab + 14) = AR;
                AR = ^auto_sub;
                DM(reg_tab) = AR;

check:          AX0 = I4;               {Current position in sin_tab}
                AR = AX0 - AY0;         {If at start of sin_tab}
                IF NE RTI;              {jump to appropriate routine}
                I5 = DM(reg_tab);       {address held in reg-tab}
                JUMP (I5);

{When setting the output enter here..}

oput_sub:       CALL set_oput;          {Call routine and set frequency}

{**************************Frequency Routine*****************************}

freq_sub:       AR = ^return;           {Clear routine address..}
                DM(reg_tab) = AR;
                SI = DM(reg_tab + 2);   {Get frequency code from table}
                AY1 = SI;               {Save in AY1}
                SR = LSHIFT SI BY 1(LO);{Multiply code by 2}
                I2 = ^freq_tab;         {Index into frequency table}
                M2 = SR0;               {in EEPROM..}
                MODIFY (I2,M2);
                I3 = ^cal_tab;          {Index into calibration table}
                MODIFY (I3,M2);         {in EEPROM}
                AR = 0x00FF;            {Load mask for low byte..}
                AF = PASS AR;
                AR = DM(I2,M0);         {Get sample rate from freq table}
                AR = AR AND AF;         {and mask LSB}
                DM(0x3FF1) = AR;        {Load SPORT1 clock rate}
                AR = DM(I2,M2);         {Get sample step from frequency}
                AR = AR AND AF;         {table and mask LSB}
                M4 = AR;                {Update M4 with new step}
                SI = DM(reg_tab + 6);   {Get current output}
                SR = LSHIFT SI BY -5(LO);
                M2 = SR0;               {If left earphone then add 1 to}
                MODIFY (I3,M2);         {index address}
                AR = DM(I3,M0);         {Get calibration offset}
                AR = AR AND AF;         {and mask LSB}
                DM(cal_off) = AR;       {Store in variable}

{This section of code sets up the 8279 LED frequency display..}

                AR = AY1 - 1;           {Get freq code and decrement}
                SE = AR;                {Load shift control register}
                AR = 0x0090;            {Setup 8279 display position..}
                DM(disp_cmd) = AR;
                SI = 0x0001;            {Load shifter with 1}
                SR = LSHIFT SI (LO);    {Shift to appropriate frequency}
                DM(disp_data) = SR0;    {Load 8279 with LS display byte}
                SR = LSHIFT SR0 BY -8 (LO);
                DM(disp_data) = SR0;    {Load 8279 with MS display byte}

{**************************Attenuator Routine****************************}

attn_sub:       I2 = ^reg_tab;          {Clear address..}
                M2 = 3;
                DM(I2,M2) = ^return;
                ENA M_MODE;             {Set integer multiplication mode}
                MR = 0;                 {Clear MR register}
                MR0 = ^attn_tab;        {Get base address of attn tab}
                MX0 = DM(I2,M0);        {Get attenuator setting}
                SE = MX0;               {Store attenuator value in SE}
                MY0 = 15;               {Multiply attenuator by 15}
```

```
                    MR = MR + MX0 * MY0(UU);{and index into table}
                    MX0 = DM(I2,M0);        {Get attenuator fine value}
                    MY0 = 3;                {Multiply by 3 and index into}
                    MR = MR + MX0 * MY0(UU);{table}
                    MX0 = DM(cal_off);      {Get calibration offset}
                    MR = MR + MX0 * MY0(UU);{Index into table}
                    DIS M_MODE;             {Set fractional mode}
                    AY1 = 0x22C2;           {Test to see if end of}
                    AR = MR0 - AY1;         {attenuator table reached}
                    IF LE JUMP multi;       {Jump if OK}
                    AY1 = DM(reg_tab + 3);  {Otherwise get current}
                    AR = AY1 - 1;           {value, decrement and store}
                    DM(reg_tab + 3) = AR;   {Go back to start to see}
                    JUMP attn_sub;          {if this value is OK}

{This section sets up the attenuator multiplier in MX0..}

multi:              I2 = MR0;               {Load I2 with attenuator address}
                    AR = 0x00FF;            {Load 8-bit mask..}
                    AF = PASS AR;
                    AR = DM(I2,M0);         {Get attenuator LSB}
                    AR = AR AND AF;         {and mask}
                    SR0 = AR;               {Load shifter}
                    SI = DM(I2,M0);         {Get attenuator MSB}
                    SR = SR OR LSHIFT SI BY 8 (LO);
                    MX0 = SR0;              {Load MX0 with attenuator word}

{This section sets up the control byte for the analog output stages..}

                    AR = DM(I2,M0);         {Get control byte from table}
                    AF = PASS AR;           {Save in AF}
                    AY1 = 3;                {Check if gain is}
                    AR = AR AND AY1;        {set to 0dB}
                    IF NE JUMP filter;      {Jump if not}
                    AR = DM(reg_tab + 2);   {Get current frequency}
                    AY1 = 3;                {Check to see if frequency}
                    AR = AR - AY1;          {is 125 or 250 Hz}
                    AX1 = 8;                {If so switch filter from}
                    IF GE AF = AX1 + AF;    {8K to 250 Hz cut off}
filter:             AR = PASS AF;           {Store in AR}
                    DM(step_nib) = AR;      {Load nibble}
                    AR = DM(reg_tab + 1);   {Get delay for change and}
                    DM(delay_count) = AR;   {load delay count}

{This section of code sets up the 8279 LED attenuator display..}

                    AR = 0x0092;            {Setup 8279 display position..}
                    DM(disp_cmd) = AR;
                    SI = 0x0001;            {Load shift register with 1}
                    SR = LSHIFT SI (LO);    {Shift to appropriate attenuator}
                    SI = SR1;               {value - stored in SE}
                    DM(disp_data) = SR0;    {Load four display bytes..}
                    SR = LSHIFT SR0 BY -8 (LO);
                    DM(disp_data) = SR0;    {into 8279}
                    DM(disp_data) = SI;
                    SR = LSHIFT SI BY -8 (LO);
                    DM(disp_data) = SR0;
                    RTI;                    {Finished}

{*****************************Output Routine*****************************}

set_oput:           SI = DM(reg_tab + 6);   {Get output byte}
                    SR = LSHIFT SI BY -5 (LO);
                    SE = SR0;               {Load shifter with Left/Right}
                    SI = 32;                {select bit}
```

```
                SR = LSHIFT SI (LO);    {Set Right or Left bit and}
                DM(cont_nib) = SR0;     {store in nibble}
                RTS;                    {Finished}

{************************Tone Interrupter Routine************************}

tint_sub:       AR = ^return;           {Clear address..}
                DM(reg_tab) = AR;
                AX1 = DM(reg_tab + 5);  {Get tone interrupter byte}
                AR = 0;                 {Clear AR}
                DM(tone_off) = AR;      {Assume one shot is disabled}
                AY1 = 0x0008;           {Test to see if one-shot is}
                AR = AX1 AND AY1;       {selected}
                IF EQ JUMP tint_puls;   {If not jump}
                AR = 0x01F4;            {Otherwise load tone_off with}
                DM(tone_off) = AR;      {length of one-shot}
tint_puls:      AR = 0;                 {Clear AR}
                DM(puls_len) = AR;      {Follow same procedure to check}
                AY1 = 0x0010;           {if pulsed tone bit is enabled..}
                AR = AX1 AND AY1;
                IF EQ JUMP tint_on;     {If not jump}
                AR = 0x00FF;
                DM(tone_off) = AR;      {Set tone on length}
                AR = 0x0500;
                DM(puls_len) = AR;      {Set repetition time}

{This section controls the presentation of the tone..}

tint_on:        AY1 = 0x0040;           {Check to see if tone on bit is}
                AR = AX1 AND AY1;       {selected}
                AR = 3;                 {If so set envelope increment +3}
                IF EQ AR = -AR;         {Otherwise set to -3}
                DM(env_incr) = AR;      {Store variable}
                IF EQ RTI;              {If tone off then exit}
                AR = DM(cont_nib);      {Check to see if tone is}
                AF = PASS AR;           {currently being presented}
                IF NE RTI;              {If so exit}
                CALL set_oput;          {If not this must be start of}
                AR = DM(step_nib);      {tone envelope}
                AY1 = DM(cont_nib);     {Get step and control nibbles}
                AR = AR OR AY1;         {Combine for control byte}
                DM(inout_6) = AR;       {output to port}
                AR = 200;               {Setup delay prior to start}
                DM(delay_count) = AR;   {of tone envelope}
                POP STS;                {Disable TIMER interrupt - it}
                IMASK = 0x0024;         {will be enabled after delay to}
                PUSH STS;               {start tone envelope}
                RTI;                    {Finished}

{*****************************Auto Routines*****************************}

auto_sub:       AR = ^return;           {Clear address}
                DM(reg_tab) = AR;
                AX1 = DM(reg_tab + 14); {Get auto control byte}
                AR = 0;                 {Clear AR}
                DM(auto_off) = AR;
                DM(auto_len) = AR;      {Assume auto disabled}
                AY1 = 0x0020;           {Check to see if auto routine is}
                AR = AX1 AND AY1;       {finished}
                IF EQ JUMP auto_audg;   {Jump if it is}
                AR = 0x01FF;            {Otherwise set up auto tone}
                DM(auto_off) = AR;      {length and repetition time..}
                AR = 0x0500;
                DM(auto_len) = AR;
                AR = 0x00FF;
```

```
                    CNTR = 3;               {Fill threshold tab with FF..}
                    DO clear_thrsh UNTIL CE;
clear_thrsh:        DM(I1,M0) = AR;         {ie clear it}
                    I3 = ^audg_tab;         {Point to audiogram table}
                    AR = 30;
                    CNTR = 25;              {Fill audiogram tab with 30..}
                    DO clear_audg UNTIL CE;
clear_audg:         DM(I3,M0) = AR;         {This corresponds to "N/T"}

                    AY1 = 0x00FF;           {Load 8 bit mask}
                    I3 = ^auto_tab;         {Point to auto table}
                    AR = DM(I3,M0);
                    AR = AR AND AY1;        {Get frequency byte}
                    DM(reg_tab + 2) = AR;   {Setup frequency}
                    AR = DM(I3,M0);         {Get Right/Left byte}
                    AR = AR AND AY1;
                    DM(reg_tab + 6) = AR;   {Setup output}
                    AR = 0x000A;            {Load AR with 30 dB HLS}
                    DM(reg_tab + 3) = AR;   {Set attenuator}
                    AR = 64;
                    DM(reg_tab + 5) = AR;   {Set tone ON}
                    DM(auto_pntr) = I3;     {Save auto pointer}
                    I3 = ^audg_tab;
                    DM(audg_pntr) = I3;     {Save audiogram pointer}
                    AR = 1;                 {Set last HL increment to 5}
                    DM(auto_inc) = AR;      {and store}
                    MX1 = 0;                {Make sure tone is OFF}
                    PUSH STS;
                    CALL oput_sub;          {Call output and tone}
                    JUMP tint_sub;          {interrupter routines to start}

{This section prints out the audiogram to the serial port..}

auto_audg:          I5 = ^audg_crlf;        {Print a carriage return and}
                    CALL send_chars;        {line feed}
                    I5 = ^audg_titl;
                    CALL send_chars;        {Print the title}
                    I5 = ^audg_crlf;
                    CALL send_chars;
                    I5 = ^audg_crlf;
                    CALL send_chars;        {Print two CR/LF}
                    I0 = ^audg_tab;
                    I3 = ^auto_tab;         {Set up pointers into tables}
                    ENA M_MODE;             {Enable integer multiply}
                    MR = 0;                 {Clear MR}
auto_again:         MR0 = ^audg_freq;       {Get pointer to frequency string}
                    MY0 = 5;                {Each frequency is five chars}
                    AY1 = 0x00FF;           {Mask}
                    AR = DM(I3,M0);         {Get frequency}
                    AR = AR AND AY1;        {and mask}
                    IF EQ JUMP auto_end;    {If end of table jump to end}
                    MR = MR + AR * MY0 (UU);{Index into frequency string}
                    AR = DM(I3,M0);         {Get Right/Left earphone}
                    AY1 = 0x0020;
                    AR = AR AND AY1;
                    I5 = ^audg_r;           {Point to Right..}
                    IF EQ JUMP auto_next;
                    I5 = ^audg_l;           {or Left string}
auto_next:          CALL send_chars;        {Print string}
                    I5 = ^audg_space;
                    CALL send_chars;        {Print a space}
                    I5 = ^audg_space;
                    CALL send_chars;        {Print a space}
                    I5 = MR0;
                    CALL send_chars;        {Print the frequency}
```

```
                    I5 = ^audg_space;
                    CALL send_chars;        {a space}
                    I5 = ^audg_space;
                    CALL send_chars;        {a space}
                    MR0 = ^audg_attn;       {Pointer to attenuator string}
                    MY0 = 4;                {4 chars in string}
                    AY1 = 0x00FF;           {Mask}
                    AR = DM(I0,M0);         {Attenuator value}
                    AR = AR AND AY1;
                    MR = MR + AR * MY0 (UU);{Index into attn string table}
                    I5 = MR0;
                    CALL send_chars;        {Print threshold}
                    I5 = ^audg_crlf;
                    CALL send_chars;        {Move to next line}
                    JUMP auto_again;        {Repeat for next line}
auto_end:           DIS M_MODE;             {Restore multiply mode}
                    RTI;                    {Finished}

{This routine sends the 0 terminated string pointed to by I5 to the UART..}

send_chars:         AR = 0x0008;            {Load UART test byte}
                    AF = PASS AR;           {Store in AF}
next_char:          AR = PM(I5,M7);         {Get character}
                    AR = PASS AR;           {Check if it is 0..}
                    IF EQ RTS;              {If so return}
                    DM(uart_data) = AR;     {Write to UART}
uart_wait:          NOP;                    {Delay}
                    AR = DM(uart_sr);       {Get UART status}
                    AR = AR AND AF;         {Mask status and loop if}
                    IF EQ JUMP uart_wait;   {transmit register not empty}
                    JUMP next_char;         {Repeat for next character}
                    RTS;                    {Finished}
.ENDMOD;
```

APPENDIX 4B - UART SUBROUTINE

```
{=========================SCC2691 UART SUBROUTINE=========================}

.MODULE/RAM       uart_sub;                {Name program}

.EXTERNAL         reg_tab, sub_tab, freq_sub, attn_sub, tint_sub;
.EXTERNAL         oput_sub, auto_sub, inout_0, inout_6;
.EXTERNAL         resp_flag, mS_count, max_resp, auto_inc, audg_pntr;
.EXTERNAL         auto_pntr, tone_off, puls_len, auto_len, auto_off;
.EXTERNAL         uart_mr12, uart_sr, uart_cr, uart_data, uart_acr;
.EXTERNAL         uart_int, uart_ctu, uart_ctl;

.ENTRY            uart;                    {Define entry label}

{****************************1mS Timer Routine****************************}

uart:             ENA SEC_REG;             {Use secondary registers}
                  AY1 = 0x0004;            {Load mask}
                  AX0 = DM(uart_int);      {Get UART interrupt register}
                  AR = AX0 AND AY1;        {and test for counter or byte in}
                  IF NE JUMP byte_in;      {Jump if byte received}
                  AR = 0x0090;             {Load control byte}
                  DM(uart_cr) = AR;        {Stop counter and clear}
                  AY1 = DM(mS_count);      {Get mS count}
                  AR = AY1 + 1;            {Increment count}
                  DM(mS_count) = AR;       {and store}

{***********************Patient Response Routine**************************}

                  AR = DM(inout_0);        {Read input port}
                  AY1 = 0x0001;            {Mask for response}
                  AR = AR AND AY1;         {Test input byte}
                  AY1 = DM(resp_flag);     {Get current response flag}
                  DM(resp_flag) = AR;      {Update with new value}
                  AR = AR - AY1;           {Compare new with previous}
                  AR = 0x00C0;             {Setup byte to output}
                  AY1 = 1;                 {If button pressed AR = 0xC1}
                  IF GT AR = AR + AY1;     {Button released AR = 0xC0}
                  IF EQ JUMP test_resp;    {Jump if no change}
                  DM(uart_data) = AR;      {Output response byte}
test_resp:        IF LE JUMP test_off;     {If button released jump}

{This section calculates the response time...}

                  AR = DM(reg_tab + 7);    {Get previous response time}
                  AR = PASS AR;            {byte and jump if not zero}
                  IF NE JUMP test_off;     {ie button already pressed}
                  AR = DM(mS_count);       {Get response time}
                  AY1 = DM(max_resp);      {Get max response time}
                  AR = AR - AY1;           {Compare}
                  IF GT JUMP test_off;     {If max exceeded then jump}
                  SI = DM(mS_count);       {Otherwise store response}
                  DM(reg_tab + 7) = SI;    {time in appropriate registers.}
                  SR = LSHIFT SI BY -8 (LO);
                  DM(reg_tab + 8) = SR0;

{************************Automatic Test Routines**************************}

{This section controls the one-shot and pulsed timings..}

test_off:         AR = DM(mS_count);       {Get mS count}
```

```
                AY1 = DM(tone_off);      {Get one-shot length}
                AR = AR - AY1;           {Test}
                IF NE JUMP test_puls;    {Jump if not equal}
                AY1=B#0000000000000000; {Otherwise set tone interrupter}
                JUMP set_tint;           {word and jump}

test_puls:      AR = DM(mS_count);       {Get count}
                AY1 = DM(puls_len);      {Get pulse length}
                AR = AR - AY1;           {Test}
                IF NE JUMP test_aoff;    {Jump if not equal}
                AY1=B#0000000001000000; {Otherwise set tone interrupter}
                JUMP set_tint;           {word and jump}

{This is start of auto routines, first check timings..}

test_aoff:      AR = DM(mS_count);       {Get count}
                AY1 = DM(auto_off);      {Get auto tone length}
                AR = AR - AY1;           {Test}
                IF NE JUMP test_auto;    {Jump if not equal}
                AY1=B#0000000000000000; {Otherwise set tone interrupter}
                JUMP set_tint;           {word and jump}

test_auto:      AR = DM(auto_len);       {Get auto pulse length}
                AR = PASS AR;            {If set to zero then jump}
                IF EQ JUMP start_cnt;    {ie auto disabled}
                AY1 = PM(I7,M5);         {Get random offset from table}
                AR = AR + AY1;           {Add to pulse length}
                AY1 = DM(mS_count);      {Get count}
                AR = AR - AY1;           {Test}
                IF NE JUMP start_cnt;    {Jump if not tone start}
                MODIFY (I7,M7);          {Move to next random offset}
                AR = DM(reg_tab + 7);    {Get patient response}
                AR = PASS AR;            {and test}
                AR = -2;                 {Set attenuator offset -10}
                IF EQ AR = PASS 1;       {If no response set to +5}
                AY0 = AR;                {AY0 = now_inc}
                AY1 = DM(reg_tab + 3);   {AY1 = now attn}
                AR = DM(auto_inc);       {AR = auto_inc}
                AR = AR - AY0;           {Test for reversal from +5}
                IF LE JUMP set_attn;     {to -10 - tone detected}
                AX0 = 0;                 {Set up flag}
                CNTR = 3;                {Set up counter}
                DO thrsh_test UNTIL CE; {Carry out three checks}
                AR = DM(I1,M0);          {Get stored threshold}
                AR = AR - AY1;           {Compare to current}
                IF NE JUMP thrsh_test;   {Jump if not equal}
                AX0 = AY1;               {Otherwise set flag}
thrsh_test:     NOP;
                DM(I1,M0) = AY1;         {Store threshold in table}
                AR = AX0 - AY1;          {Test for match in table}
                IF NE JUMP set_attn;     {If not jump}
                I3 = DM(audg_pntr);      {Threshold found so get pointer}
                DM(I3,M0) = AY1;         {audiogram table and store}
                DM(audg_pntr) = I3;      {Update pointer}
                AR = 0x00FF;
                CNTR = 3;                {Fill threshold tab with FF..}
                DO clear_thrsh UNTIL CE;
clear_thrsh:    DM(I1,M0) = AR;

                AR = 4;                  {Equivalent to +20 dB}
                AR = AR + AY1;
                DM(reg_tab + 3) = AR;    {Add 20dB to attn for new freq}
                AY0 = 0x00FF;            {Load 8 bit mask}
                I3 = DM(auto_pntr);      {Pointer to auto table}
                AR = DM(I3,M0);          {Load AR with next auto freq}
```

```
                AR = AR AND AY0;          {and mask it}
                IF NE JUMP set_freq;      {Jump if end not reached}
                DM(reg_tab + 14) = AR;    {Disable auto}
                AR = ^auto_sub;           {Load auto routine so that when}
                DM(reg_tab) = AR;         {this sub finished audiogram}
                JUMP pc_port_end;         {will be printed to port}

{This section sets up the next auto frequency..}

set_freq:       DM(reg_tab + 2) = AR;     {Load new frequency}
                AR = ^oput_sub;           {Output sub will be run to set}
                DM(reg_tab) = AR;         {up new freq and earphone}
                AR = DM(I3,M0);           {Get output byte}
                AR = AR AND AY0;          {and mask}
                DM(reg_tab + 6) = AR;     {Load register}
                DM(auto_pntr) = I3;       {Store auto pointer}
                AR = -2;                  {Set last increment to -10}
                DM(auto_inc) = AR;        {for freq change}
                AR = PASS 0;              {Reset counter to zero..}
                DM(mS_count) = AR;
                JUMP start_cnt;           {Jump to start counter}

{This section sets up the next auto attenuator..}

set_attn:       DM(auto_inc) = AY0;       {Store increment}
                AR = AY1;                 {Load AR with current attn}
                AR = AR + AY0;            {and inc or dec as appropriate}
                IF LT AR = PASS 0;        {Don't go below zero}
                DM(reg_tab + 3) = AR;     {Load register with new value}
                DIS SEC_REG;              {Swap registers}
                PUSH STS;                 {Call attenuator routine}
                CALL attn_sub;            {directly to set up new value}
                ENA SEC_REG;              {Restore registers}
                AY1=B#0000000001000000; {Set tone interrupter word}

{This section sets up the tone interrupter..}

set_tint:       AR = ^tint_sub;           {Point to tone interrupter}
                DM(reg_tab) = AR;         {routine}
                AR = DM(reg_tab + 5);     {Get current setting}
                AY0=B#0000000000111111; {Make sure bit is set to zero}
                AR = AR AND AY0;          {initially}
                AR = AR OR AY1;           {Then get new setting}
                DM(reg_tab + 5) = AR;     {Store in register}

start_cnt:      AR = 0x0080;              {Start control word}
                DM(uart_cr) = AR;         {Write to UART to start count}
                JUMP pc_port_end;         {Jump to exit from routine}

{*************************Serial Port Routines*****************************}

byte_in:        AY0 = 0x0080;             {Load mask for control/data}
                AY1 = 0x00FF;             {Byte mask}
                AX0 = DM(uart_data);      {Get byte from receive register}
                AR = AX0 AND AY1;         {Mask byte}
                AX0 = AR;                 {and store in AX0}
                AR = AX0 AND AY0;         {Test for control or data byte}
                IF EQ JUMP data_byte;     {Jump if data}
                AY0 = 0x0040;             {Mask for read/write}
                AY1 = 0x000F;             {Mask for lower nibble}
                AR = AX0 AND AY1;         {Mask control byte}
                M6 = AR;                  {Load modifier with reg offset}
                I6 = ^reg_tab;            {Load reg table base address}
                AF = PASS 1, AX1 = DM(I6,M6); {Load AX1 with address and}
                                          {set flag in AF}
```

```
                AR = AX0 AND AY0;       {Check for read or write}
                IF EQ JUMP pc_port_end; {Jump if write}
                AF = AF - 1, AR = DM(I6,M6);  {Get current setting from}
                                        {table and zero flag}
                DM(uart_data) = AR;     {Load UART send register}
                JUMP pc_port_end;       {Jump to end}

data_byte:      AF = AF - 1;            {Check flag}
                IF NE JUMP pc_port_end; {If not data byte then jump}
                DM(I6,M6) = AX0;        {Load reg table with data byte}
                I6 = ^sub_tab;          {Point to subroutine table}
                MODIFY (I6,M6);         {Index into table}
                AR = DM(I6,M6);         {Get appropriate subroutine}
                DM(reg_tab) = AR;       {address and store in reg table}

pc_port_end:    DIS SEC_REG;            {Restore registers}
                RTS;                    {Finished}

.ENDMOD;
```

APPENDIX 4C - UART INITIALISATION SUBROUTINE

```
{=================SCC2691 UART INITIALISATION SUBROUTINE=================}

{This subroutine is called by the main program to set up the UART after a
hardware reset}

.MODULE/RAM       uart_set_sub;           {Subroutine name}

.EXTERNAL         reg_tab, sub_tab, inout_0, inout_6;
.EXTERNAL         uart_mr12, uart_sr, uart_cr, uart_data, uart_acr;
.EXTERNAL         uart_int, uart_ctu, uart_ctl;

.ENTRY            uart_set;               {Define entry label}

uart_set:         AR = 0x0093;            {MODE REGISTER 1: Enable RTS,}
                  DM(uart_mr12) = AR;     {No Parity, 8-bit Data Word}

                  AR = 0x0017;            {MODE REGISTER 2: Normal Mode,}
                  DM(uart_mr12) = AR;     {Enable CTS, 1 Stop Bit}

                  AR = 0x00BB;            {CLOCK SELECT REGISTER: Rx=9600,}
                  DM(uart_sr) = AR;       {Tx=9600}

                  AR = 0x0038;            {AUXILLIARY CONTROL REGISTER: }
                  DM(uart_acr) = AR;      {Counter=Clk/16, MPO=RTS}

                  AR = 0x0014;            {INTERRUPT MASK REGISTER: Enable}
                  DM(uart_int) = AR;      {Counter and Receive Interrupts}

                  AR = 0x0000;            {COUNTER/TIMER UPPER: =0}
                  DM(uart_ctu) = AR;

                  AR = 0x00E6;            {COUNTER/TIMER LOWER: = 230 to }
                  DM(uart_ctl) = AR;      {give 1mS interrupt rate}

                  AR = 0x0020;            {COMMAND REGISTER: }
                  DM(uart_cr) = AR;       {Reset Transmitter, }
                  AR = 0x0030;
                  DM(uart_cr) = AR;       {Reset Receiver, }
                  AR = 0x00A0;
                  DM(uart_cr) = AR;       {Assert RTS Line, }
                  AR = 0x0045;
                  DM(uart_cr) = AR;       {Clear and enable UART}

                  RTS;

.ENDMOD;
```

APPENDIX 5 - SERIAL PORT CONTROL CODES

Control Byte

1	R/W	U	U	C	C	C	C

The control byte is identified by the most significant bit - this is always set to one. The R/W bit signifies whether the data byte is to be read or written to the Control Register. The U bits are undefined at present but could be utilised for further expansion of the serial interface. The four C bits allow sixteen Control Register locations to be addressed.

Data Byte

0	D	D	D	D	D	D	D

The most significant bit of the data byte is set to one. The seven D bits comprise the data value to be written (or read) from the Control Register.

Control Registers

0	SYSTEM
1	STIMULUS I
2	FREQUENCY I
3	ATTENUATOR I
4	ATTENUATOR FINE I
5	INTERRUPTER I
6	OUTPUT I
7	STIMULUS II
8	FREQUENCY II
9	ATTENUATOR II
10	ATTENUATOR FINE II
11	INTERRUPTER II
12	OUTPUT II
13	COMMON
14	TESTS
15	RESPONSE

The above table lists the function of each of the Control Registers. The registers have been defined to allow for future expansion of the audiometer interface. For example a second set of signal control registers allows for the control of a two channel clinical type of audiometer. The present audiometer does not utilise all of the registers and in the definitions which follow only the Control Registers which the audiometer responds to are detailed.

Frequency Control Register

0	F	F	F	F	F	F	F

VALUE	FREQENCY
0	OFF
1	125
2	250
3	500
4	750
5	1000
6	1500
7	2000
8	3000
9	4000
10	6000
11	8000
12	12000
13...	UNDEFINED -
...127	POSSIBLY 1/3 OCTAVE FREQ

Attenuator Control Register

0	U	U	A	A	A	A	A

VALUE	ATTENUATOR
0	-20
1	-15
2	-10
3	-5
4	0
5	5
6	10
7....	15
...29	125
30	130
31	135

Attenuator Fine Control Register

0	B	B	B	B	B	B	B

VALUE	ATTENUATOR FINE
0	0.0
1	0.1
2	0.2
3	0.3
4	0.4
5	0.5
6	0.6
7....	0.7
...125	12.5
126	12.6
127	12.7

Interrupter Control Register

0	I	R	P	O	U	U	U

BIT	DEFINITION
I	Present tone ON (1) / OFF (0)
R	Reverse Enable (1) / Disable (0)
P	Pulsed Enable (1) / Disable(0)
O	One Shot Enable (1) / Disable (0)
U	Undefined

Output Control Register

0	R	L	U	U	U	U	U

BIT	DEFINITION
R	Right Earphone (1)
L	Left Earphone (1)
U	Undefined

APPENDIX 6 - ADSP2105 PROCESSOR ARCHITECTURE

ADSP-21xx Registers