The development of a portable Earth's field NMR system for the study of Antarctic sea ice

A thesis presented in partial fulfilment
of the requirements for the degree of
Master of Science in Electronics at
Massey University

Robin Dykstra

2001

Abstract

A portable Nuclear Magnetic Resonance (NMR) spectrometer based on digital signal processor (DSP) technology has been developed and applied to the study of the structure of Antarctic sea ice. The portability of this system means that external sources of noise can be minimised and remote sites can be investigated.

A new sea-ice probe has been developed in conjunction with the spectrometer allowing in-situ measurement of water content, relaxation times and self diffusion. The new probe minimises disturbances to the sea ice sample which have been a problem with previous techniques.

The core of the spectrometer consists of a Motorola DSP56303 DSP which controls the NMR experiment under the supervison of a host computer which in this case is a PC laptop. Communication between host and DSP is via either a PCMCIA card or USB interface. DSP software runs the experiment, controls acquisition and performs digital filtering of the NMR data before sending it to the PC for analysis and display.

The flexibility of the DSP based core means that this system could be adapted to other control applications with relative ease.

Acknowledgments

Firstly I would like to thank my supervisor, Dr. Craig Eccles, for his friendship, and his guidance and contributions throughout this work, in particular in the area of Windows programming. His scrupulous and thorough approach to experimental work and proof reading has been greatly appreciated.

I gratefully thank Professor Paul Callaghan for his encouragement, enthusiasm and support, and for providing me with the opportunity to go to Antarctica.

Associate Professor Robert O'Driscoll is also somebody who has been a great help over the years with the sharing of his ideas, experiences, and his passion for electronics.

Thanks to the staff in the workshops for their assistance and for always delivering on my sometimes demanding requests.

The person I would like to thank the most and whom I love is my wife Christine. Her support, patience and love have been a tremendous help and I am deeply indebted. Also our children, Kerry, Haidee, Hannah and Cameron, have been very understanding ("Dad is doing his thesis again") and now it's payback time.

Contents

1.0 Introduction

To most people the continent of Antarctica has little or no impact on their lives and exists only in their minds as an icy wasteland at the bottom of the globe. However it has captured the attention of a few and has drawn them into its strange and amazing environment. An environment that has unspoiled beauty, yet can be extremely treacherous. Although Antarctica may seem insignificant to some, it plays a very significant role in global ecology and climate due to the vastness of the continent and its surrounding sea ice. It is also considered by some to be the weather engine of the planet, and is sometimes described as the coldest, windiest and driest place on Earth.

Each winter the waters around Antarctica freeze and form a 1 to 2 m thick layer of sea ice over an area of ocean of approximately 20 million square kilometres (figure 1.0). This sea ice behaves as a blanket in that it insulates the cold polar atmosphere from the warmer ocean, thus restricting heat exchange [4]. The exchange of carbon dioxide is also limited by this relatively impermeable covering. The sea ice is usually covered by a layer of snow, which assists it in reflecting over 90% of sunlight in summer. This is much greater than the 5% reflected sunlight from the open ocean and demonstrates the effectiveness of the sea ice as a massive solar reflector. As most of the sunlight is reflected, the amount that is absorbed at the surface, and that is available for photosynthesis in the underlying ocean, is significantly reduced. Furthermore, a layer of algae grows on the underside of the sea ice and further restricts the light available to the ocean. This layer of algae equates to a vast quantity of biomass that is later released into the ocean in summer.

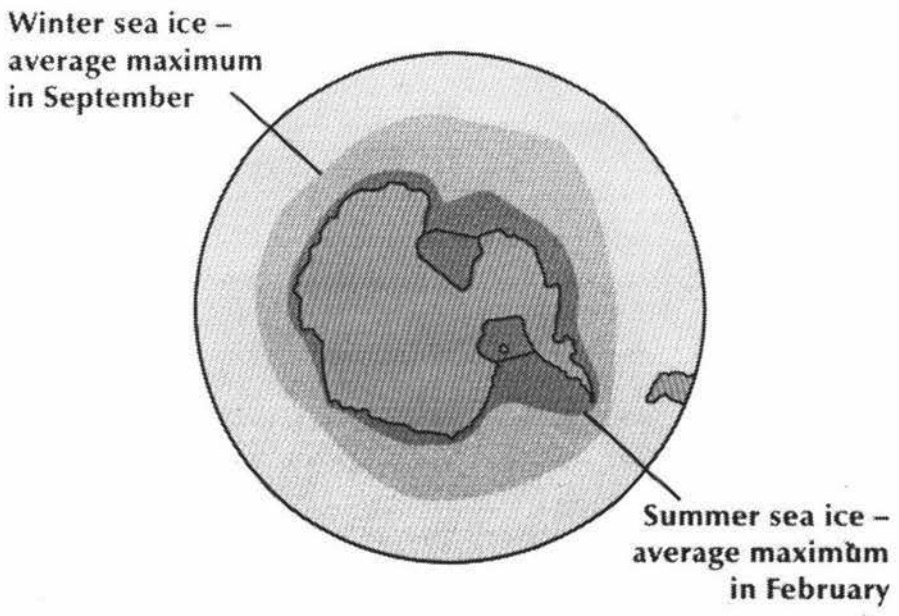


Fig 1.0 Seasonal variation in Antarctic sea ice cover [8].

When the seawater freezes, small cavities of concentrated brine form inside pure ice. Some of the brine drains out through vertical channels as the sea ice grows in thickness, but some of it remains trapped. As the ocean and atmosphere are at different

temperatures, a temperature gradient exists within the sea ice, which then causes convection processes to occur within the cavities. The cavities slowly migrate downwards due to the melting that occurs at the bottom of the cavity and the subsequent refreezing at the top. Over time these cavities migrate towards the ocean and eventually drain out, releasing the brine into the ocean. In summer the sea ice slowly breaks up and moves north releasing vast quantities of fresh water and algal spores into the ocean, again influencing the climate and ecology. The enormous seasonal variation in the surface area of the sea ice is a large annual event that is much intertwined with the global climate, and therefore any effective climate modelling depends upon an understanding of the optical, thermal and mechanical properties of this complex and unusual material.

For a number of years now the Physics department at Massey University has been using Nuclear Magnetic Resonance (NMR) techniques to study the microstructure of Antarctic sea ice. This is part of a larger Antarctic sea ice research program in which the study of the structure and mechanical properties of sea ice has been undertaken, allowing predictions to be made about the strength of sea ice, how it formed and how it breaks up under wave action. Our involvement is at the microscopic end of the scale where we are studying the behaviour of the brine cavities.

There are two types of experiments that we perform on sea ice:

1. The first is the determination of the brine content within the ice. This information is used to make predictions about the age of the ice and how it formed. The concentrated brine has a very low freezing point and significant free water can still be found in sea ice at temperatures below $-20\,^{0}C$.
2. The second is the measurement of the brine diffusivity. Here we get information about the diffusion and convection occurring within the cavities which will help in making predictions about the thermal behaviour of the sea ice. It can also give information about the brine pocket dimensions.

This thesis describes some work that I have been doing part time over a period of about three years, during which time a portable Earth's field NMR system designed primarily for Antarctic research, but with other applications in mind, was developed. This work is an extension to previous work done within the Physics department, where a system for studying sea ice was initially put together using a number of commercial units and some homebuilt electronics [3]. This early system itself grew out of a laboratory experiment [1] that was developed a number of years ago within the department.

My task was to design and construct a totally in-house system. The project could be divided into four parts:

1. Probe design and construction.
2. Antarctic Earth's Field NMR System design and construction.
3. The design and construction of a general purpose NMR system "core".
4. Experimental work in Antarctica.

This work started in the latter part of 1997 when I designed and built a new probe for the Antarctic visit at the end of the year. Some improvements and modifications were

also made to the existing NMR system electronics that were used with the new probe. In 1999 another visit was planned, so I then put together a totally new portable system that would allow us to make remote measurements. Some improvements were also carried out on the probe that I built for the previous season. Since returning from the last Antarctic visit, I have continued with the development of the system, in order to optimise it for any future visits, and to lay the foundation for some future work where systems will be developed for other NMR and non-NMR applications.

The following chapter presents a brief introduction to NMR, with some further information regarding the areas concerning my project. The third chapter describes the design and construction of the new portable Earth's field NMR system and the new probe, and concludes with an evaluation of the system and probe's performance. Chapter four gives the details of the design and construction of the "system core" component mentioned in chapter 3. Included is a description of the digital signal processing techniques used. The end of the chapter describes some work that has been done to improve and extend the capabilities of the system core. An overview of some of the work done in Antarctica is given in chapter 5, while the final chapter presents some conclusions.

2.0 Nuclear Magnetic Resonance

2.1 Introduction to Nuclear Magnetic Resonance [†]

Nuclear Magnetic Resonance (NMR) is a technique that allows one to non-invasively analyse the physical and chemical properties of materials by probing the nuclear dipole moments within the material. Nuclei of certain atoms have an associated magnetic moment and spin much like a small spinning magnet. When these nuclei are placed in a magnetic field ($\boldsymbol{B}_0$), the nuclear magnetic moments will start to precess about the field, like a spinning top (figure 2).

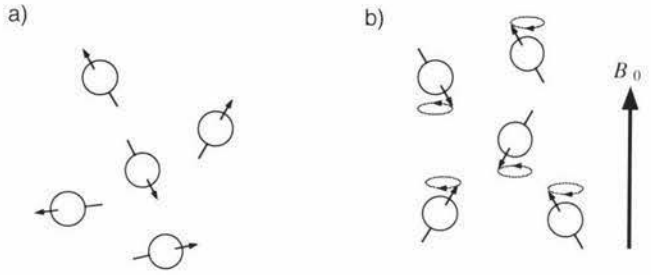


Fig 2.0 Nuclear magnetic moments before (a) and after (b) the application of a magnetic field.

The frequency of this precession (known as the resonant or Larmor frequency) depends on the type of nuclei and the strength of the applied magnetic field $\boldsymbol{B}_0$. The equation for this relationship is:

$$f = 2\pi\gamma B_0$$

γ is the gyromagnetic ratio, which for hydrogen nuclei is relatively high giving a precession frequency of 42.8 MHz/Tesla. The higher the gyromagnetic ratio the better, as it allows greater signal to noise ratios to be obtained due to the increased coupling between the spinning magnetic moments and a pick up coil. For the sea ice experiments, NMR allows one to selectively probe the hydrogen nuclei found in liquid water. The hydrogen nucleus can have either of two states in a magnetic field, one with its magnetic moment aligned along the field direction and the other anti-aligned. At room temperatures there is a slightly greater number of aligned nuclei than anti-aligned, and the sum of all the nuclear magnetic moments results in a small net magnetization $\boldsymbol{M}$, in the direction of the applied field (figure 2.1). The magnetization is proportional to the applied field and sample volume.

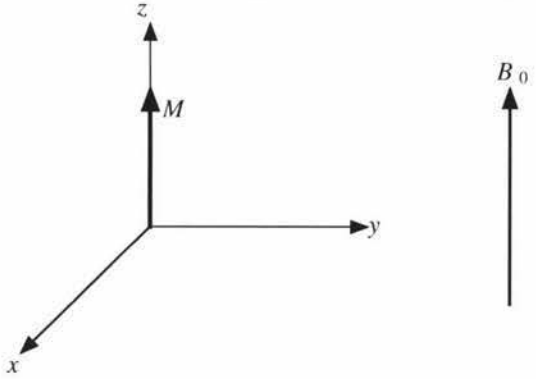


Fig 2.1 Net equilibrium magnetization $\boldsymbol{M}$ aligned with the applied field $\boldsymbol{B}_0$.

[†] Note that this is a grossly simplified description of an inherently quantum mechanical phenomena. For a more in depth description the reader is directed to a standard NMR text such as reference 5.

To interact with the nuclear magnetization $\boldsymbol{M}$, a radio frequency (RF) pulse, at the Larmor frequency, is applied to a coil surrounding the sample and is orientated in such a way so that its magnetic field is at right angles to $\boldsymbol{B}_0$. The RF signal is sinusoidal and when applied to the coil an oscillating magnetic field is generated which can be represented as two counter rotating field vectors. Only one of the vectors ($\boldsymbol{B}_1$) interacts with the magnetization and so the oscillating magnetic field can be represented as a single rotating field vector that is at right angles to $\boldsymbol{B}_0$ (figure 2.3).[†]

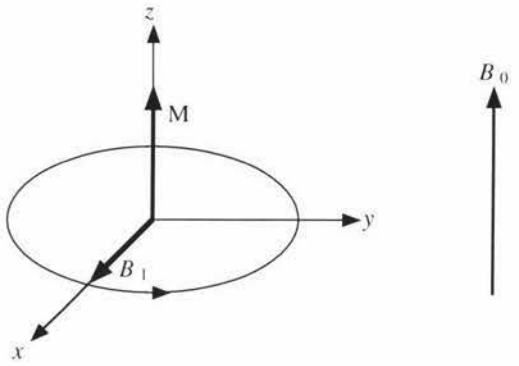


Fig 2.3 Rotating $\boldsymbol{B}_1$ vector produced by RF pulse.

The torque applied by the rotating field $\boldsymbol{B}_1$ causes the magnetization vector $\boldsymbol{M}$ to spiral down around the z axis (figure 2.4). When it reaches the x,y plane the RF field is removed and the precessing magnetization can then be observed as an induced EMF in a coil surrounding the sample. The $\boldsymbol{B}_1$ field used to achieve this is known as a 90-degree pulse, and the coil used to apply it is often used to receive the induced signal as well.

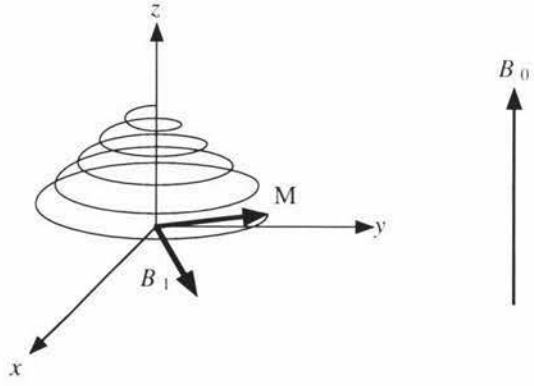


Fig 2.4 Magnetization vector $\boldsymbol{M}$ spirals down into the x,y plane due to applied $\boldsymbol{B}_1$ pulse.

The induced signal doesn't last forever as relaxation processes will cause the magnetization to return to its equilibrium position, while inhomogeneities in the $\boldsymbol{B}_0$ field will cause the magnetization to dephase, as not all the components will precess at the same frequency (figure 2.5). The resultant signal is called the free induction decay (FID) and consists of a combination of decaying sinusoids. The $\boldsymbol{B}_0$ inhomogeneities are caused by imperfections in the applied $\boldsymbol{B}_0$ reference field and also by the localised magnetic effects of other surrounding nuclei. The effect of the surrounding nuclei is

[†] For more details on how this works refer to reference 5 or any other standard NMR text.

much greater in solids than in liquids, which is good for sea ice experiments as it allows one to distinguish between the two materials. The signal from the ice only lasts a few microseconds whereas the signal from the liquid water can last up to a second.

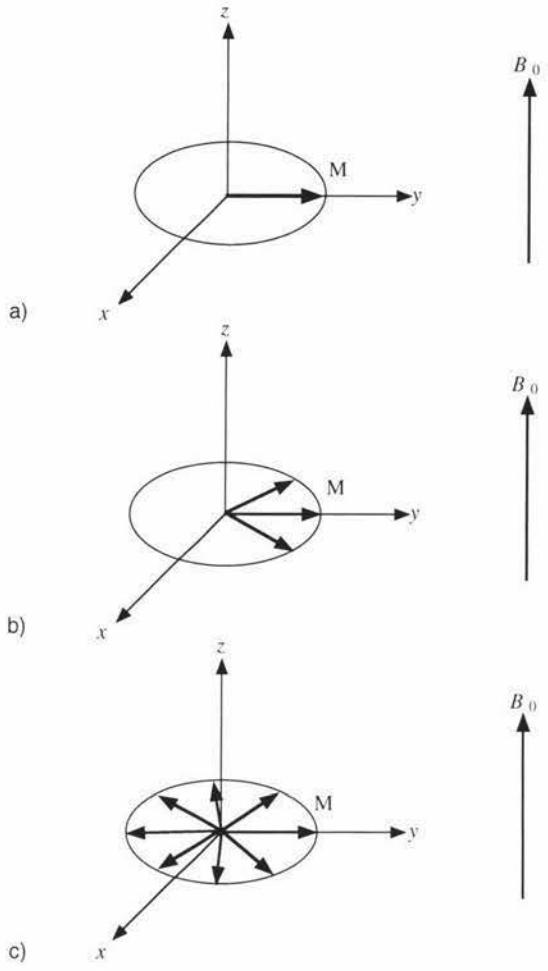


Fig 2.5 Dephasing of the magnetization due to variations in the $\boldsymbol{B}_0$ field.

Pulsed NMR is in some ways similar to the impulse response techniques used in electronic systems, in which a pulse is used to excite a system and the system's response or emission reveals some properties of the system (Figure 2.6).

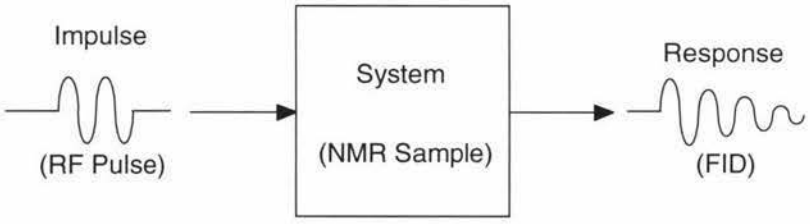


Fig 2.6 The NMR impulse response.

With NMR, the strength of the emission tells us about the number of nuclei present in the sample and is proportional to the sample volume when the nuclear density is uniform. The emission spectrum tells us about the magnetic environment of the nuclei. This forms the basis for NMR chemical spectroscopy where the spectrum can provide

information about the type of nuclei and their bonding arrangements, thereby telling us something about the makeup of a chemical sample (figure 2.7).

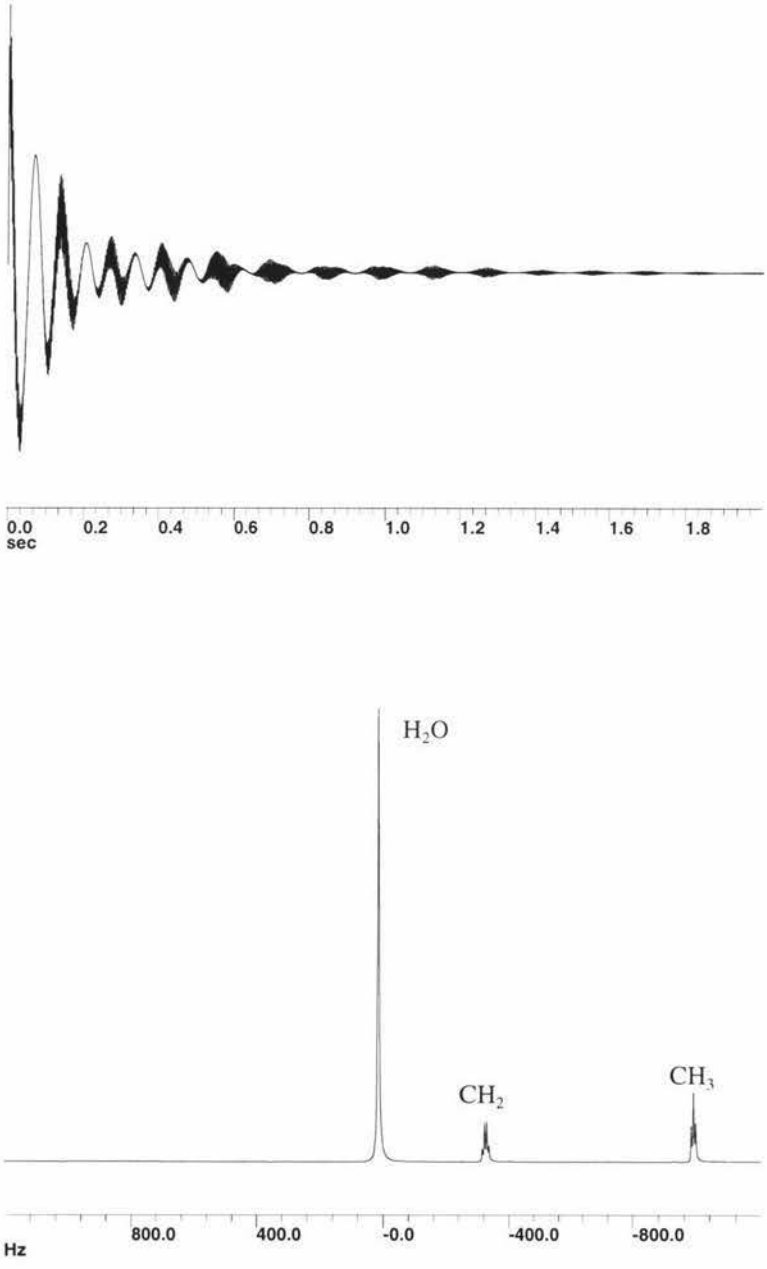


Fig 2.7 The FID and corresponding spectrum for a Tequila sample [12].

2.2 Determination of water content

For our sea ice experiments a simple experimental sequence consisting of a single 90-degree pulse and the observed FID is all that is required to determine the water content within a sea ice sample (figure 2.8). The initial amplitude of the FID is proportional to the amount of liquid water in the sea ice, and this is compared to a calibration experiment where a known water sample is used (figure 2.9).

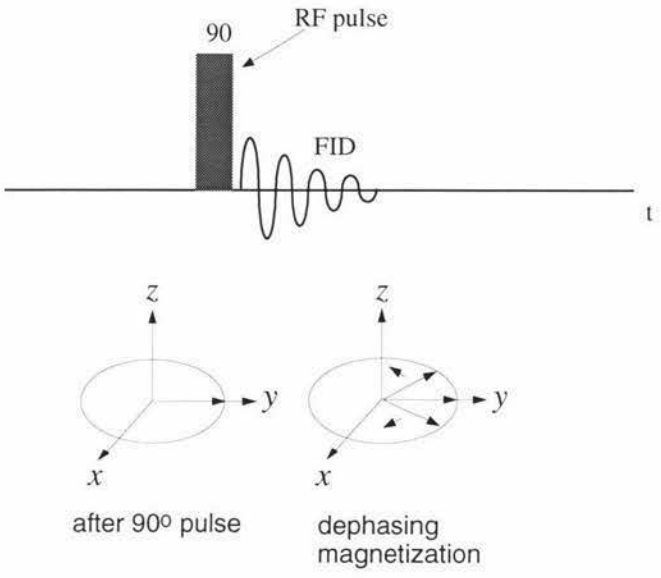


Fig 2.8 Simple 90 degree pulse sequence.

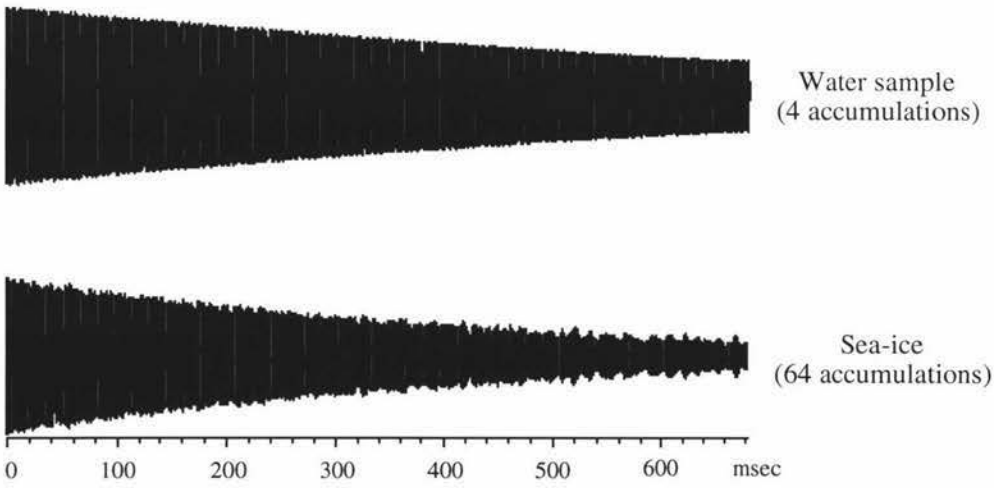


Fig 2.9 FID's for water sample and sea ice [11].

2.3 Diffusion measurements

For sea ice diffusion experiments, a more complicated sequence is required which is based on a "spin echo" technique. In an NMR experiment the decay of the signal is largely due to the dephasing of the magnetization, caused by the nuclei precessing at slightly different rates at different positions in the sample. If another RF pulse is applied a short time after the FID, so that the magnetization components are flipped over by 180 degrees, a "spin echo" will form due to the rephasing of the magnetization (figure 2.10).

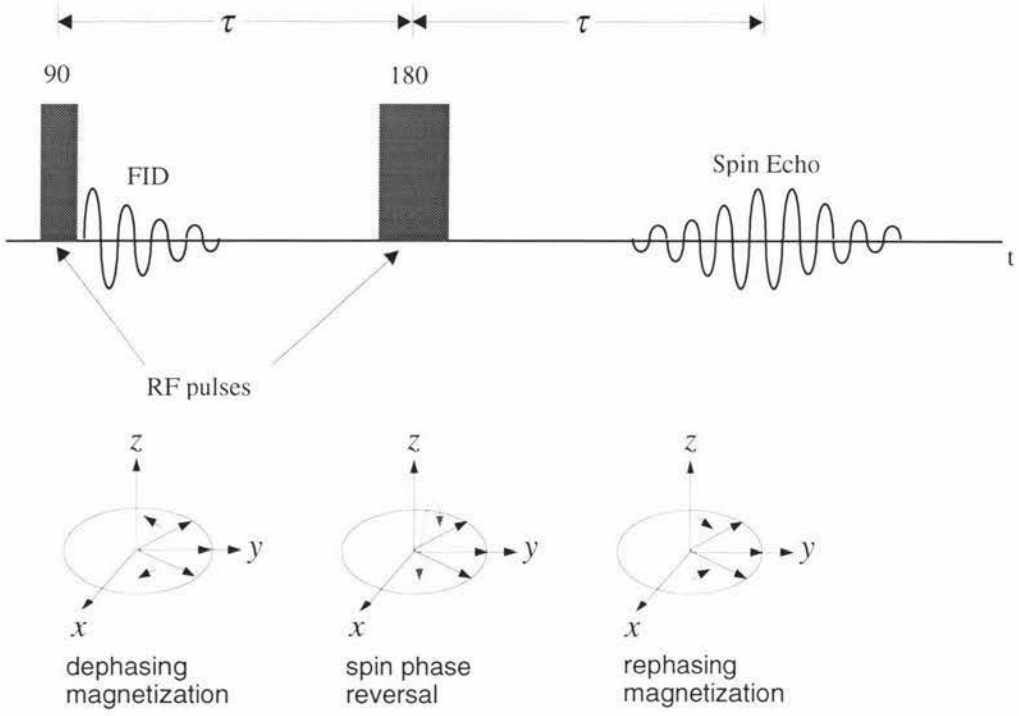


Fig 2.10 Spin echo formation due to the refocussing of the magnetization.

Not all of the spins will refocus perfectly as some will experience slight changes in their magnetic environment ($\boldsymbol{B}_0$) during the time between the 90° pulse and the spin echo formation. Apart from relaxation processes the main contributor to partial refocusing is the movement of the nuclei to different positions in an inhomogenous $\boldsymbol{B}_0$ field, and hence a change in their magnetic environment. This leads to a slight attenuation of the spin echo that is proportional to the motion of the nuclei. We can utilise and enhance this effect by applying additional magnetic field gradients, which cause the spins to rapidly dephase and rephase and therefore make the echo attenuation much more sensitive to nuclear movement. This forms the basis for the diffusion measurements, as the diffusion coefficient is related to the spin echo attenuation. The experiment used to determine the diffusion coefficient is called "Pulsed Gradient Spin Echo" (PGSE) [5], and consists of the standard spin echo sequence with two additional matched gradient pulses, one on either side of the 180° RF pulse (figure 2.11).

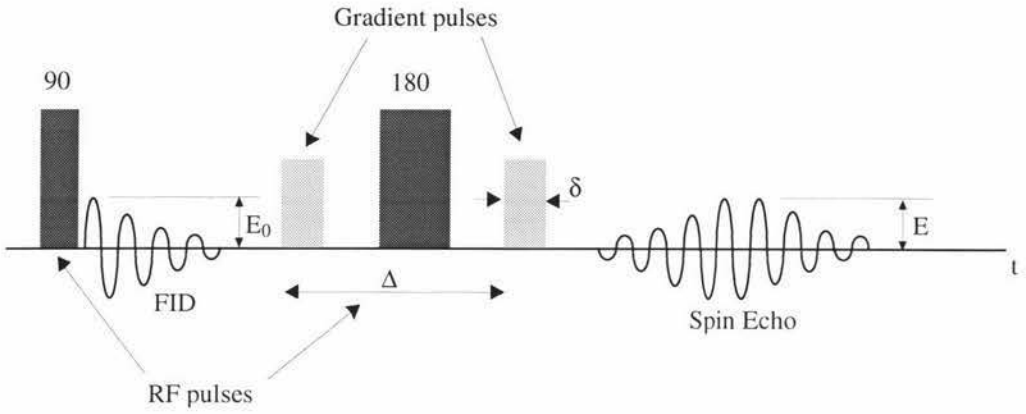


Fig 2.11 Diffusion measurement using pulsed gradients.

The equation describing the relationship between the normalised spin echo amplitude E_n and the self-diffusion coefficient D is [5]:

$$E_n = \exp [-\gamma^2 g^2 \delta^2 D (\Delta - \delta / 3)]$$

Where $E_n = E/E_0$ and Δ is the separation between two gradient pulses of duration δ and amplitude g.

A series of experiments are performed with different gradient pulse lengths δ from which a corresponding series of spin echo attenuations are obtained. The diffusion coefficient D is the negative of the slope of the graph of $\ln(E_n)$ versus $\gamma^2 g^2 \delta^2 (\Delta - \delta/3)$, (figure 2.12). Note that it is also possible to vary the gradient pulse amplitude g or the delay time Δ.

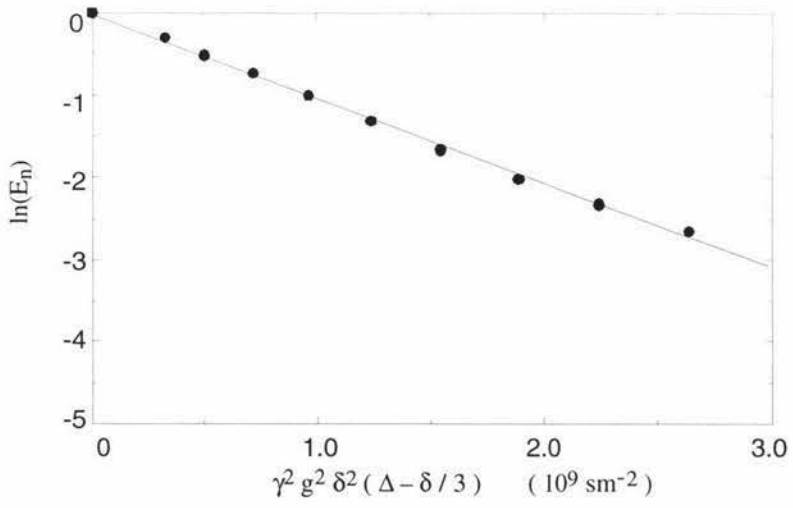


Fig 2.12 PGSE echo attenuation plot for a water sample at 7^0C [4].

2.4 Other applications and advantages of NMR

NMR has become a standard tool for chemical spectroscopists as it has greatly enhanced their ability to analyse the chemical make-up of solutions and the determination of the structure and environments of various molecules. Another very important application of NMR is imaging. Magnetic field gradients are applied to make the Larmor frequency position dependent, and therefore localised information can be obtained. This information can then be used to construct images. This technique, known as Magnetic Resonance Imaging (MRI), has become extremely popular in the medical profession, but is also very useful in materials science research. By combining the imaging technique with some other techniques, such as PGSE, it is possible to get weighted images, where the intensity is proportional to a parameter of interest, e.g. diffusion. Some of the weighted imaging techniques [11] that can be performed are:

- Nuclear density.
- Diffusion and velocity (flow) imaging.
- Relaxation imaging.
- Chemically selective imaging.

The major advantage of MRI over other imaging techniques is that it is largely non-invasive. It doesn't usually require any physical alteration of the sample, unlike optical techniques, and it doesn't require the use of harmful radiation, such as x-rays or the emissions of radioactive substances.

When spatial localisation is not required the NMR technique is capable of making averages over the sample volume which with other techniques might be more time consuming and difficult. This property is of special interest to this work in which total liquid water content is required in the sea ice sample. It is also relevant when making diffusion measurements, as the sea ice brine cavities are randomly distributed and have a range of dimensions.

NMR with its ability to analyse the physical and chemical properties of materials, has become a very important research tool, while its variant, MRI, has made a major impact as a medical diagnostic tool.

3.0 Earth's field NMR

3.1 Introduction to Earth's field NMR

Earth's field NMR makes use of the freely available magnetic field of the Earth to provide the B_0 field and is therefore inexpensive, easy to implement, highly portable, and able to be used in remote regions and under harsh environmental conditions, thereby enabling *in situ* measurements to be performed. One doesn't require a large expensive heavy magnet and a whole console full of electronics. In Antarctica the field strength is about 64 μT, which equates to a resonant frequency of 2.7 kHz (RF then becomes AF !). This field may be small, but it is nearly vertical and very uniform, so FIDs lasting a few seconds can be obtained. Also, since the field is so uniform, large samples can be used, and therefore satisfactory signal levels can be obtained despite the low Larmor frequencies. As the frequency is in the audio range, quite simple transceiver circuitry is required, and the signal can be sampled directly, eliminating the need for intermediate frequency (IF) stages.

The sensitivity of the experiment can be further increased by the application of a polarization field at the start of the pulse sequence. This is applied to increase the sample magnetization and therefore increase the subsequently received signal strength. A solenoid is used to generate the polarization field (B_p), while a saddle coil or solenoid is used to generate and receive the perpendicular B_1 field (figure 3.1). The configuration used is determined by the sample orientation and access requirements. Generally, a solenoid is used whenever possible as better signal performance is obtained.

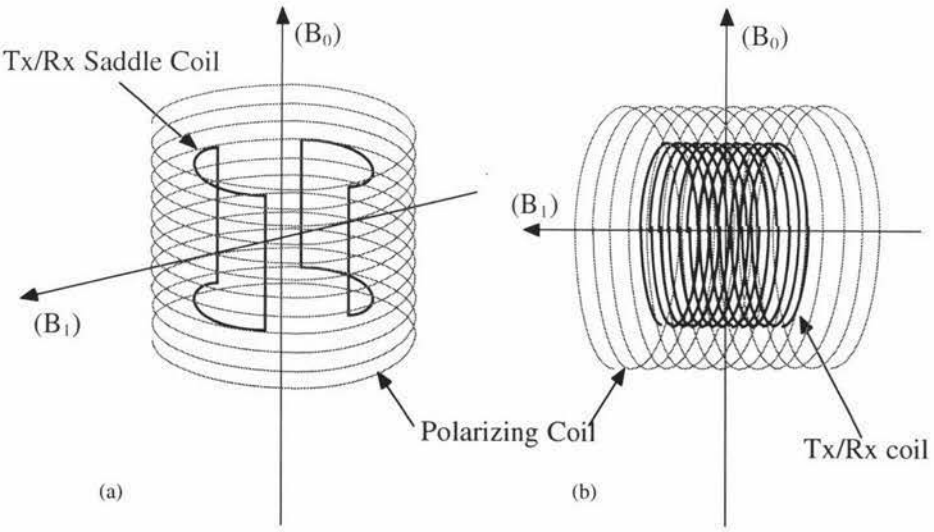


Fig 3.1 Saddle (a) and solenoid (b) probe configurations.

The earliest Antarctic sea ice NMR work [3] was based on a solenoidal probe (figure 3.1b), as vertical ice cores were extracted from the ice sheet and then inserted horizontally into the probe. For later work a saddle coil based probe was developed, as it enabled direct insertion of the probe into the ice sheet. The pulse sequences for Earth's field NMR experiments are like conventional NMR sequences but with the addition of an initial 5 second polarizing pulse (fig 3.2a). With the saddle coil probe,

the polarization field is in the same direction as the Earth's field and so the magnetization grows in the Earth's field direction (fig 3.2b). The solenoidal probe has its polarization coil oriented orthogonally to the Earth's field, and so the magnetization grows orthogonally to the Earth's field (fig 3.2c). But if the field is removed adiabatically[†] [14], the magnetization will reorientate along the Earth's field direction. This adiabatic switching is also required for the saddle probe as the polarizing field is not entirely homogeneous or in the direction of the Earth's field.

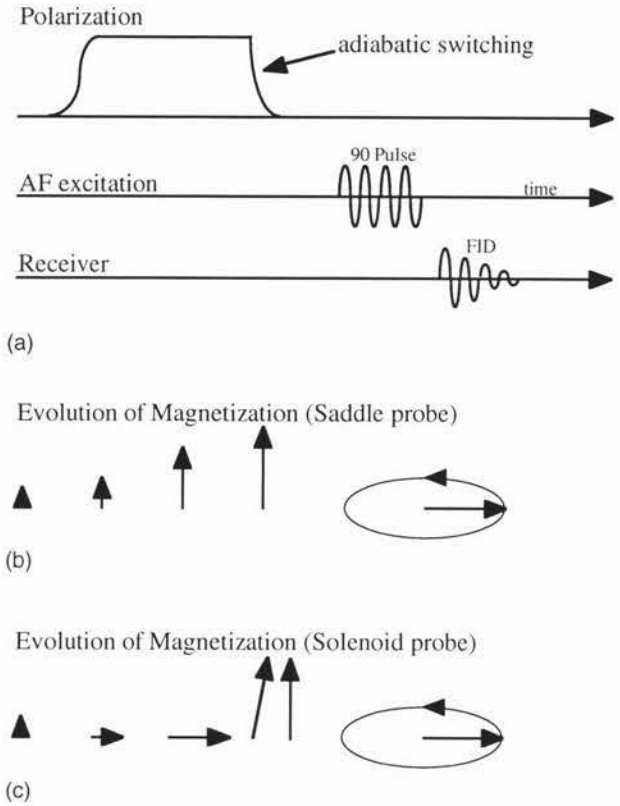


Fig 3.2 Simple pulse sequence and magnetization evolution for saddle and solenoidal probes.

3.1.1 Applications of Earth's Field NMR

Earth's field NMR is quite novel and can be used for many different types of applications. Some of these applications are the same as for conventional high field NMR systems. Some examples are:

- Water content determination of soil, fruit, sea ice, etc.
- Self diffusion and flow of water [15].
- Relaxation studies of materials [2].
- Imaging of proton density and even diffusion and relaxation weighted imaging [16,17].

The performance may not be great, but then one isn't bound to a very expensive system. The low cost and versatility of Earth's field NMR also makes it an ideal system for teaching. The finding of ground water [3] and the accurate measurement of the Earth's magnetic field are two applications that are specific to Earth's field NMR.

[†] By adiabatically we mean that the polarization field is removed slowly enough that the magnetization is not left behind but follows it.

3.2 Earth's field NMR system overview

Although sea ice research was the primary motivation behind the development of this system we also had in mind a wider range of applications. A number of design criteria therefore had to be met; the system should be:

- Portable and battery powered, to allow *in situ* sampling at different sites.
- Compact, as it needed to be able to be loaded easily into a helicopter, Skidoo, or other vehicle.
- Robust, to withstand extreme environments and rough handling.
- Flexible, with full control of software and parameters so experiments can be changed in the field.
- Low cost, so multiple systems can be used, this would also and make it available as a teaching instrument.
- Have the ability to perform water content and diffusion experiments.

In many ways the Earth's Field NMR system is very similar to a conventional laboratory system but operates at audio frequencies instead of hundreds of MHz. The system is built up from a number of interconnected units as shown in figure 3.3.

The "RF" transceiver section which really operates at audio frequencies (AF) uses standard operational amplifiers and other audio components, such as an analogue switch which is used to connect and disconnect the transmit amplifier. The power required for the B_1 excitation is minimal, therefore a low noise preamp can be connected permanently to the probe. The inner block known as the "system core" is responsible for generating all the required signals and also for digitizing and processing the FID. The probe being the "transducer" end of the system consists of a single transmit/receive coil, a polarizing coil, gradient coils, and some embedded temperature sensors. The rest of the components that make up the system are:

- Two variable 10 A constant current power supplies required for driving the polarizing and gradient coils.
- An air circulating pump to cool the polarizing coil between acquisitions.
- Some other power supply electronics so that the whole system can be run from a 24 V battery.

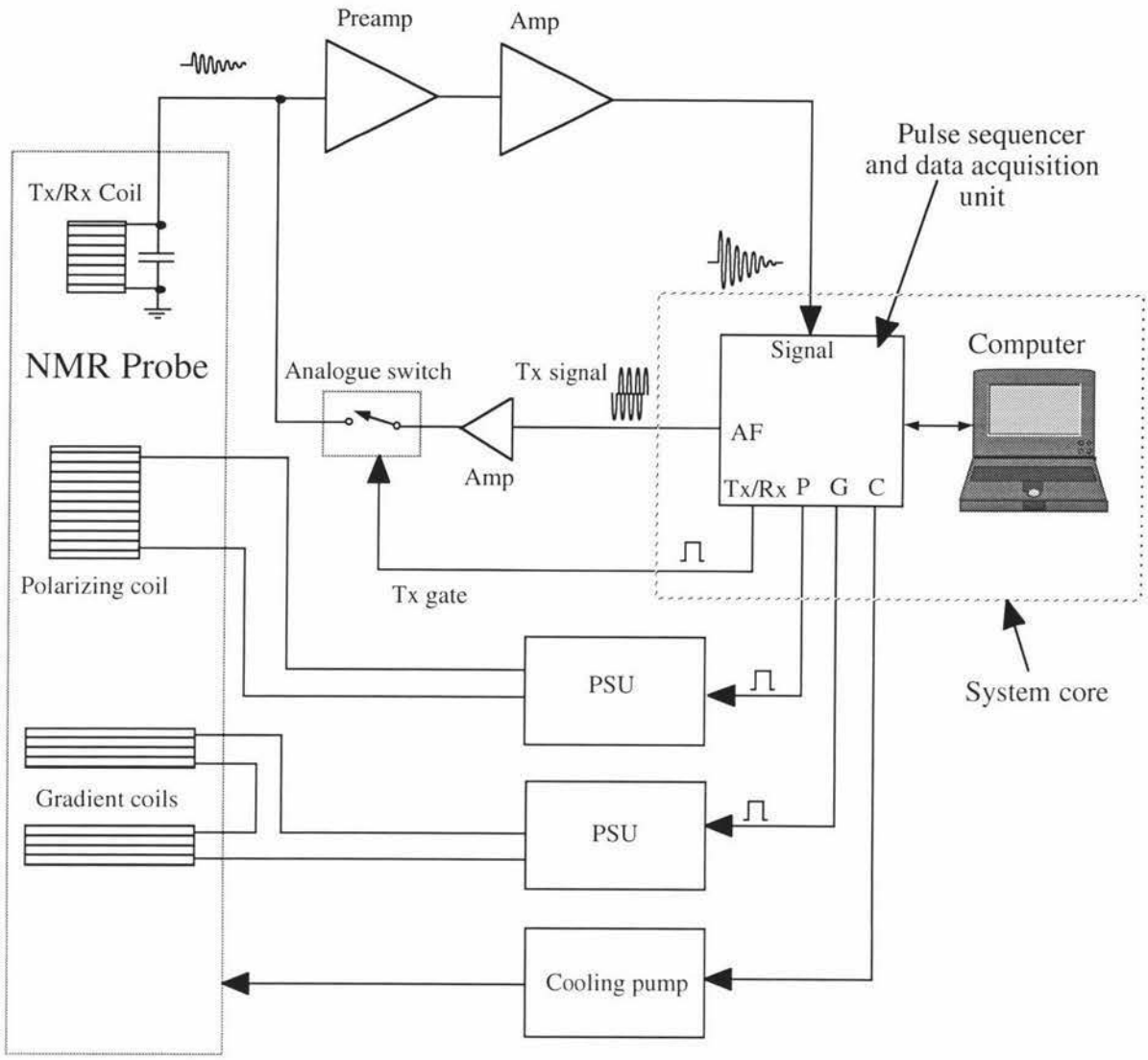


Fig 3.3 Block diagram of the Earth's field NMR system.

The early NMR work performed in Antarctica used a system built up from readily available units and some home built electronics [3]. This system, as shown in figure 3.4, was rather large, mains powered, and therefore not at all portable. Two Kepco [32] OPS 25-10M power supplies (bottom) were used to drive the polarising and gradient coils. A data acquisition and pulse programmer unit (LEO/J) from Tecmag [31] and an old Macintosh [33] IIci computer provided all the control signals and handled the digitizing and processing of the data. The transceiver section (next to the oscilloscope) we constructed ourselves and consisted of two diecast metal boxes. One box contained a preamplifier and a duplexer that I designed for the 1997 Antarctic season. This unit worked very well so I used the same design for the new portable system. The other box contained the rest of the transceiver.

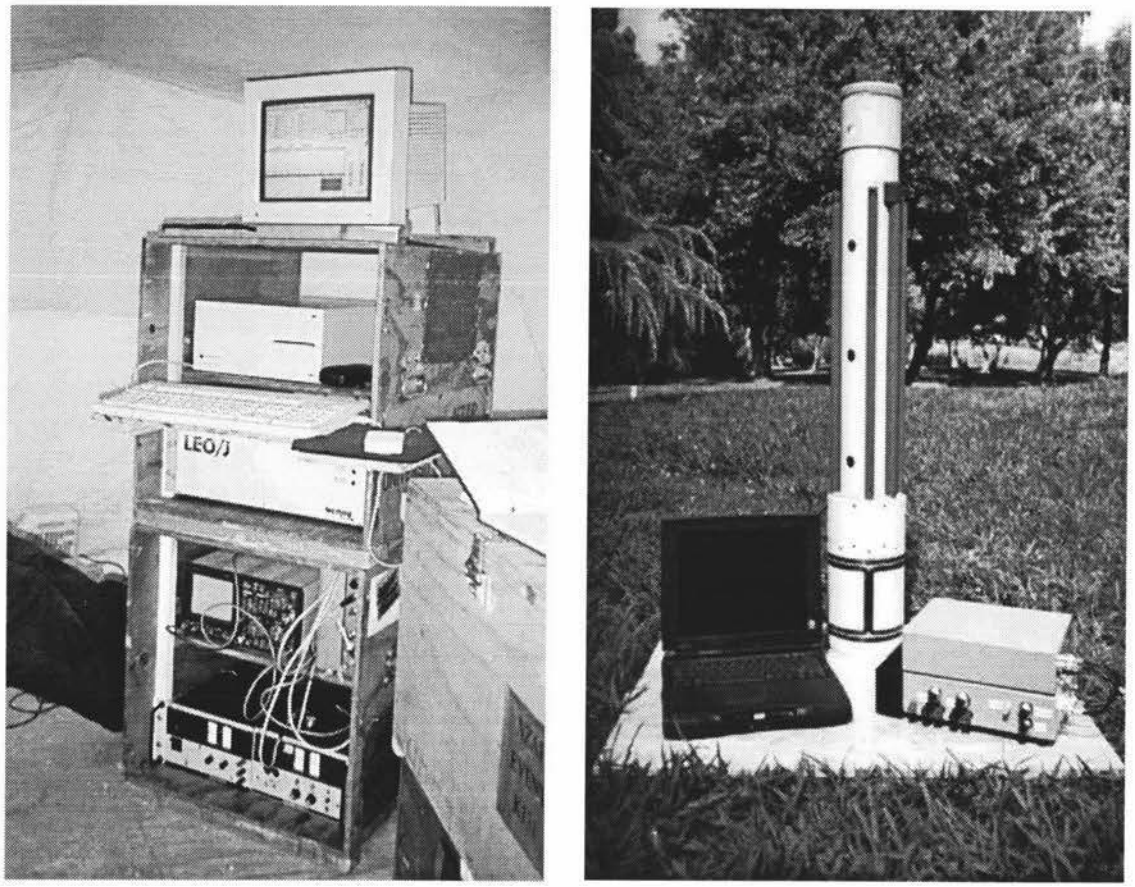


Fig 3.4 Old system (left) and new portable system with probe (right).

The new system that I developed is housed in two plastic boxes, each measuring 250 mm x 200 mm x 65 mm, and is controlled by a laptop computer. In Antarctica metal components need to be avoided, as they can be uncomfortable to handle at cold temperatures. The batteries used (not shown) are dependent on the expected power usage, the duration of the experiments, and the operating environment. In Antarctica special low temperature 40 Ah 12 V "Gel cells" were used, and 2 battery packs of about 300 mm x 200 mm x 200 mm were required to allow a whole "day's" worth of experiments.

3.3 Earth's field NMR system in detail

3.3.1 The probe

The probe plays a crucial role in an NMR system, as it is the interface between the sample and the system electronics and can therefore be thought of as a transducer. The initial NMR work performed in Antarctica [3] used a solenoidal B_1 based probe, which necessitated the removal of ice-core samples from the ice sheet (figure 3.5) since the probe couldn't be inserted into the sheet due to the B_1 orientation.

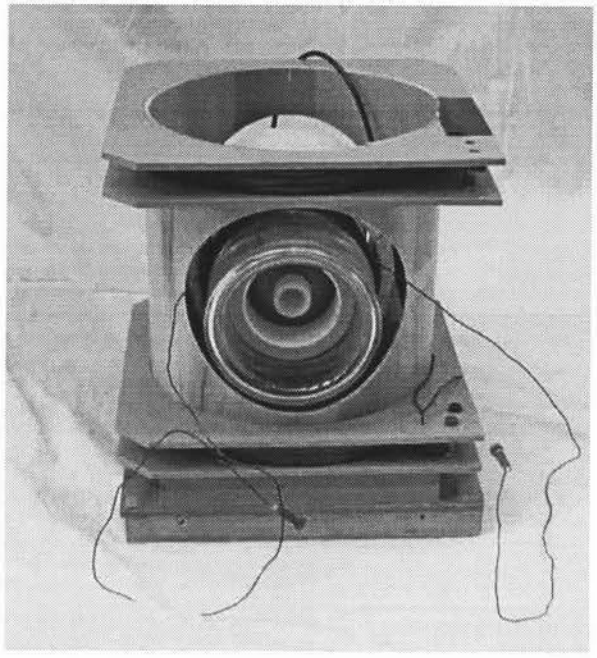


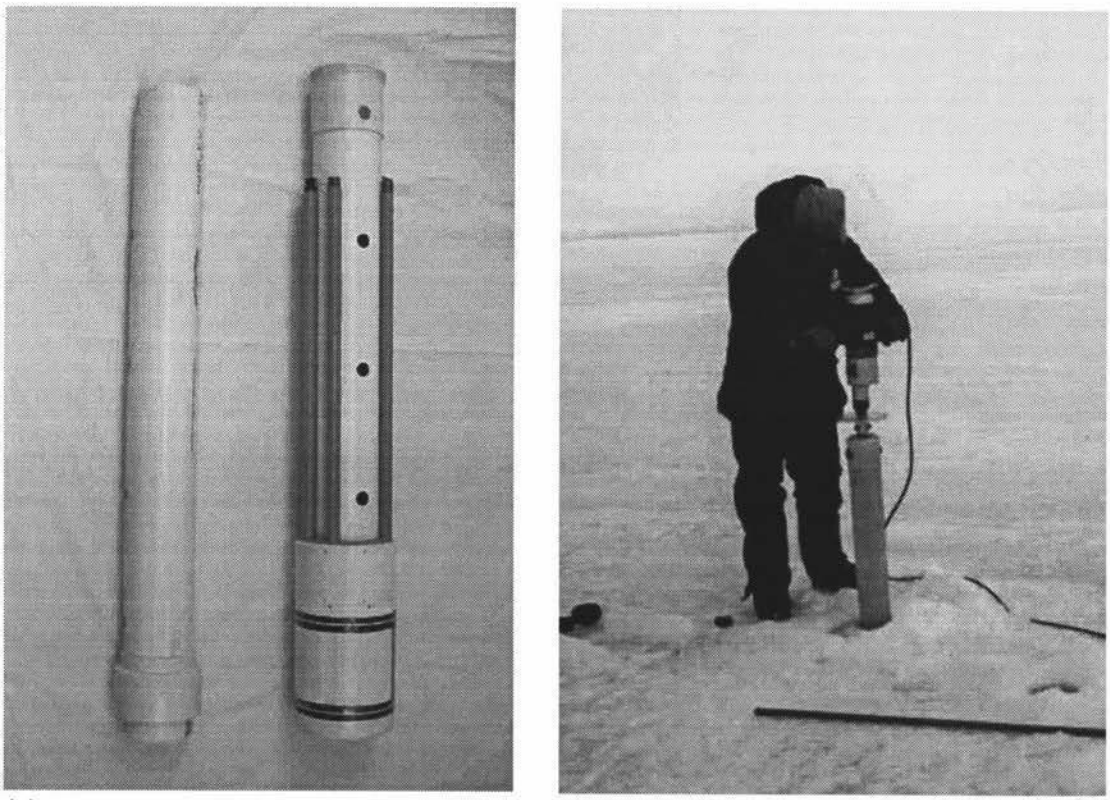
Fig 3.5 Extracted core probe

The experiments performed with this extracted core probe were very successful, but the question of the impact of removing the ice core and reorientating it needed to be answered. Disconnecting the sample from the ice sheet and reorientating it may disturb the hydrostatic equilibrium, and any change in temperature will disturb the convection and diffusion processes. Because of these potential problems, a less invasive saddle coil based probe was developed that has the ability to be inserted directly into the ice sheet. Unfortunately the saddle coil is less sensitive than a solenoid, so there is a trade off between signal strength and invasiveness.

The ice sheet probe head also had to meet a number of design criteria, in particular it should:

- be constructed as a sleeve so that it could be placed around an intact pillar of sea ice.
- be robust and water proof.
- be able to be constructed out of standard PVC pipe sizes.
- be of a suitable size so that an ice cutter could be constructed and used easily.
- be able to be lowered 2 m below the ice sheet surface.
- allow easy insertion and removal from the ice sheet.
- provide minimal degradation in signal strength.
- provide minimal degradation of the sample ie the sea ice pillar.
- monitor the sample temperature.
- use the same Tx/Rx coil.

The design was based on the availability of some 110 mm and 160 mm diameter PVC pipes and also on the ease of construction and use of an ice cutter. The ice cutter cuts out a 35 mm thick annulus leaving a pillar of ice of 95 mm diameter. This ice pillar is still connected at the base to the ice sheet. A high power drill is used to speed up the cutting process (figure 3.6b).



(a) (b)

Fig 3.6 (a) Probe with extension tube, (b) preparing an ice core.

The 110 mm pipe forms the inner sleeve that surrounds the ice core, and itself consists of two lengths as shown in figure 3.6a. One length is for the main probe assembly

(figure 3.9) and the other is an extension to allow the probe to be inserted 2 m below the ice sheet surface. A series of holes along the side of the inner sleeve are used to provide support by inserting a rod through them. The small tubes glued to the outside of the inner sleeve house the various interconnecting cables and connectors, and provide inlet and exhaust pipes for air cooling of the polarising coil. As the connectors are magnetic they need to be mounted as far away as possible from the probe's sensitive region (figure 3.8b). The saddle coils as shown in figure 3.7 are mounted on the outside of the inner sleeve and each were constructed by winding 700 turns of 0.2 mm diameter wire around a former. The resistance and inductance for each of the coils is 197 Ω and 142.5 mH. The coils are connected in series, giving a total resistance and inductance of 394 Ω and 285 mH. The formers themselves were carefully machined out of PVC pipe and then glued onto the sleeve. Above the saddle coils are two sets of PVC rings with O-rings. The O-rings combined with silicon grease provide the necessary sealing for the outer sleeve. The space (connection chamber) between the PVC rings is for all the connections that are required, the tuning capacitors and a manifold for the air cooling. The outer sleeve is made out of two pieces of 160 mm diameter PVC pipe. One piece is used to cover the connection chamber and is easily removed to allow probe tuning. The other piece houses the solenoidal polarising coil and also acts as a former for the gradient coils (figure 3.8a).

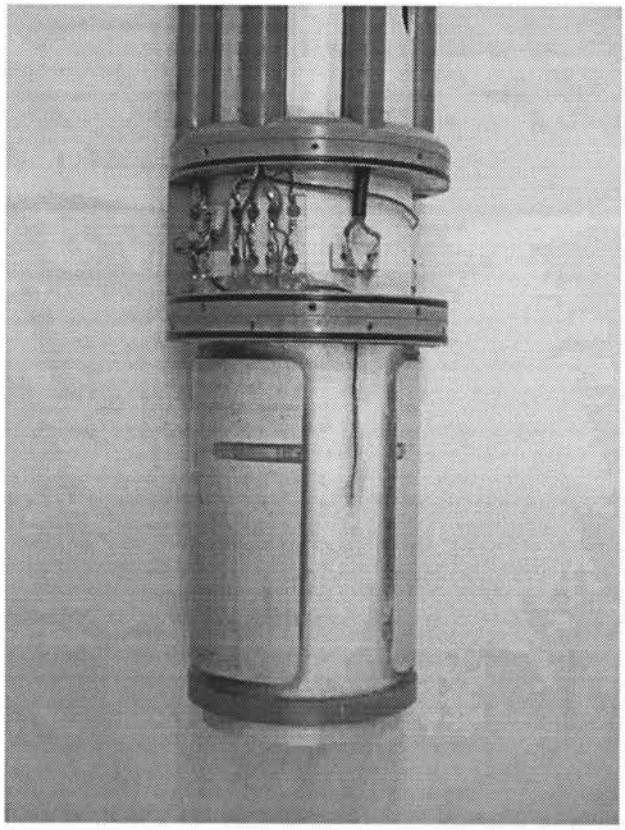
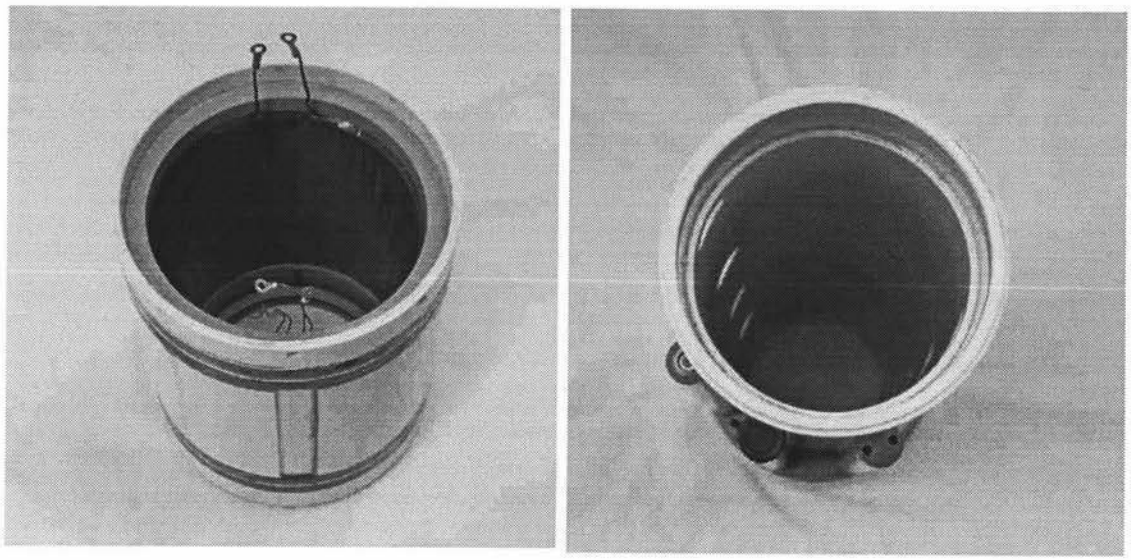


Fig 3.7 The inner part of probe assembly showing the saddle coil arrangement and connection chamber.



(a) (b)

Fig 3.8 Polarizing coil mounted in outer sleeve (a) and a birds eye view of the probe showing its sleeve form and the connectors at the end of the small tubes (b).

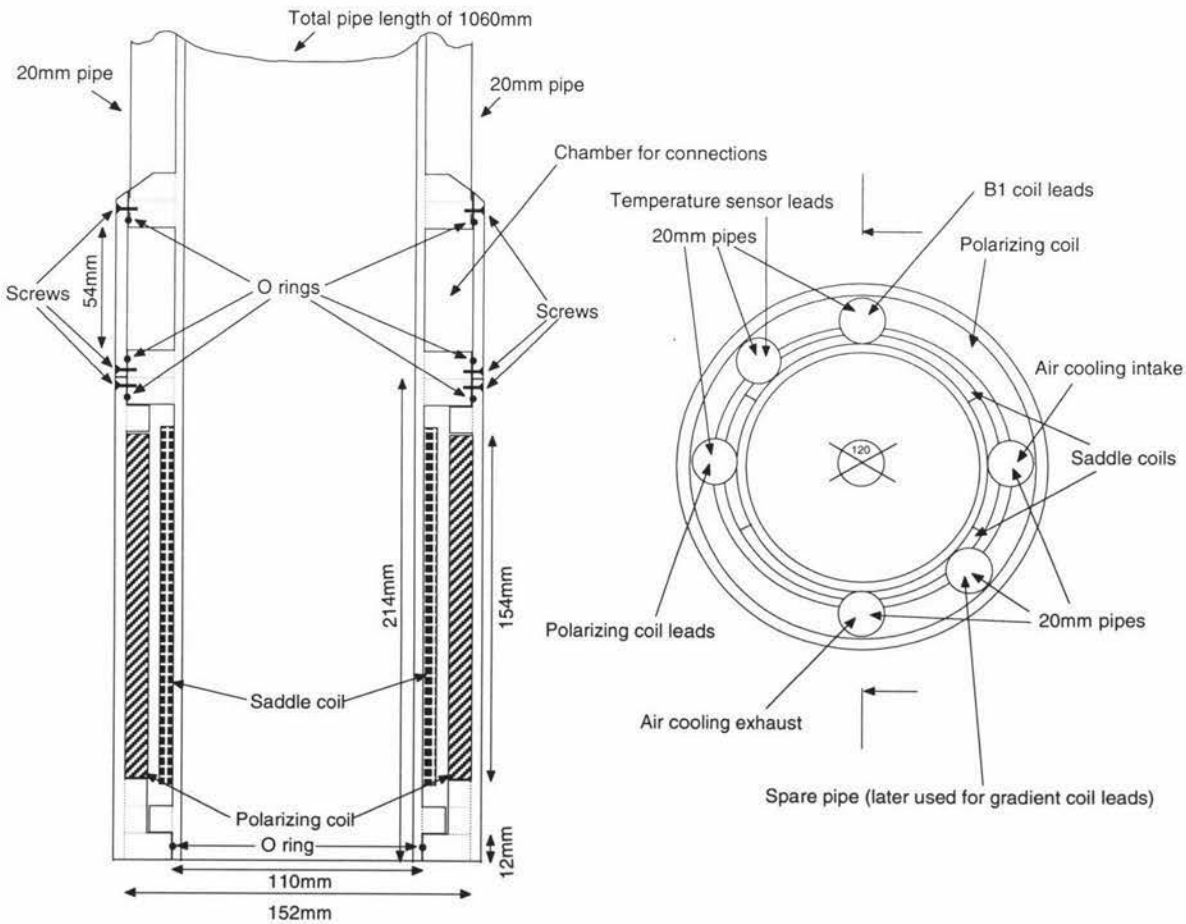


Fig 3.9 Probe details and dimensions.

The polarizing coil was made by winding 6 layers of 84 turns of 1.6 mm copper wire around a 127 mm diameter cylinder. The coil length is 152 mm and has a mean diameter of 140 mm and a resistance of 1.8 Ω for a total number of 504 turns. For these dimensions the field strength inside the solenoid would be approximately 3 mTA^{-1}. Typically a 6 A polarising pulse (18 mT) of 5 seconds duration is used for each experiment scan. So for an experiment that repeats every 10 seconds, the mean power dissipation of the coil is 32 W.

When the probe is assembled there is a small air gap between the polarising and saddle coils where cool air is forced in to provide the necessary cooling of the polarising coil. The cool air is generated by a small electric air pump and a 10 m length of silicone tubing. In Antarctica the air temperature is always below freezing so the silicone tubing acts as a moisture trap and also dissipates any heat generated by the pump. Silicone tubing is used as it remains flexible at –30 °C temperatures. Due to the forced air cooling, the probe is slightly pressurised which helps to keep out any water. There is always the risk of the probe becoming frozen into the ice sheet due to ice particles falling down the hole, and rising water. One method to remove a frozen in probe would be to turn off the air-cooling and apply continuous current to the polarising coil. This would heat up the probe and melt the surrounding ice. The top edge of the outer sleeve and uppermost PVC ring is chamfered to assist in the easy removal of the probe from the ice sheet. In all our time in Antarctica we have not had a problem with the probe freezing in, but it always pays to be prepared.

To monitor the temperature of the ice sample, two PT100 platinum resistance temperature sensors were embedded into the inner sleeve. They are mounted on either side of the probe, in between the saddle coils (figure 3.7).

The B_1 coil is a crucial part of the NMR probe, as its performance can dictate the performance of the whole system. Therefore a lot of thought was required when designing the B_1 coil, as a number of things needed to be considered, such as shape, sensitivity, Q, self resonance and signal to noise (S/N) performance. The B_1 coil basically consists of an inductor with some inherent series resistance. Capacitors are placed across this inductor to form a parallel resonant circuit, which is then tuned to resonate at the Larmor frequency (f_0). Q is known as the quality factor of the coil and is a measure of the coil's ability to resonate and is determined mainly by the coil's resistance (R) and inductance (L). The Q also determines the response time and bandwidth of the tuned circuit, which must be broad enough to cover the bandwidth of interest. A very high Q coil is not always good as it will have a very narrow bandwidth (Δf) and will take a long time to respond to any change in the input signal.

$$Q = \omega L/R \quad \text{and} \quad \Delta f = f_0/Q$$

The resistance of the coil is minimised by using the largest possible wire size and the shortest length of wire. But the sensitivity of the coil is proportional to the number of turns and so the geometry and volume of the coil limits the wire size that can be used. Therefore there is a trade off between the number of turns, wire size and Q. For high frequencies the skin effect also needs to be taken into account, but for Earth's field NMR this is not a problem. For the probe operating at 2.7 kHz a capacitance of 9 nF was used for tuning and the Q was calculated and measured to be about 12.

When a coil is wound it has capacitance between the windings and this causes the coil to be self resonant. This self resonant frequency must always be greater than the tuning frequency as capacitance can only be added and therefore the resonant frequency can only be lowered. This effect therefore places limits on the number of turns and geometry of the coil. The self resonant frequency for the probe was measured to be 18 kHz. The shape of the coil is largely determined by the shape and access requirements of the sample. The saddle coil as shown in figure 3.10 has been quoted as having a factor of about $\sqrt{3}$ to 3 lower S/N performance [13,19] than the efficient solenoid configuration and is therefore only used when absolutely necessary. The optimum saddle coil design for maximum B_1 homogeneity has H/W= 2 [21,19], but this is not always possible due to physical construction limitations as well as wire length and its associated problems of low Q and self resonance. The probe used in this work has an H/W ratio equal to 1.4 and is therefore not optimal.

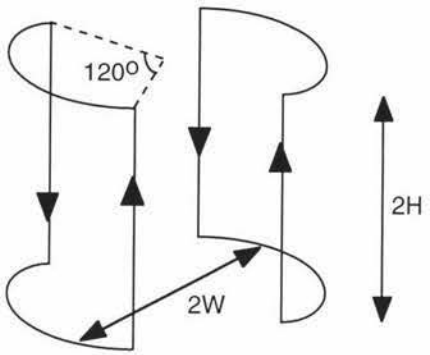


Fig 3.10 Saddle coil configuration.

To determine the viability of a design a simple S/N ratio calculation can be performed. Figure 3.11 shows a simplified schematic of the probe, where the signal and noise are represented by voltage generators in series with the coil.

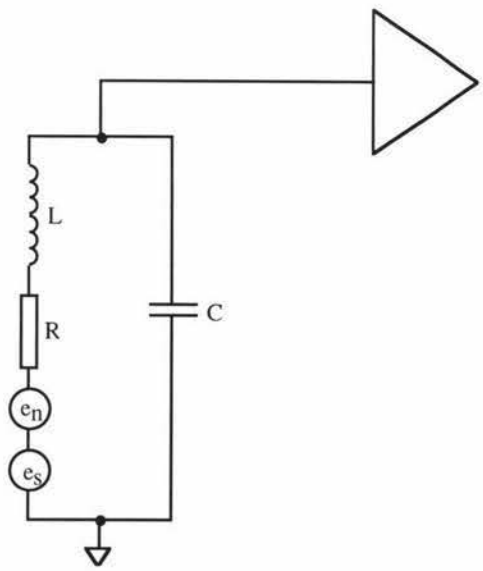


Fig 3.11 Simple model of probe for signal to noise estimation.

The strength of the signal induced in the B_1 coil from the spinning magnetization can be derived using Faraday's law, where the EMF induced in a wire loop is proportional to the rate of change of the magnetic flux within the loop. This relationship written in a more useful form [11] is given below:

$$e = -\frac{d}{dt}\left(\frac{\tilde{B}_1}{I} \cdot \tilde{m}\right)$$

Where B_1/I is the B_1 field due to unit current and m is the net magnetic moment. The magnitude of the peak voltage for a sinusoidally varying magnetization can then be calculated using:

$$e_s = \frac{B_1}{I} M_0 V_s \omega_0$$

Where M_0 is the sample magnetization, V_s the sample volume, and ω_0 the precessing rate. The peak voltage e_s is then determined by the B_1 coil's ability to couple this field. The coupling performance is gauged using the principle of reciprocity [18], where the coil's ability to receive is directly proportional to its ability to generate a magnetic field. For a saddle coil, the field at the centre for one turn and one amp can be calculated using [19]:

$$\frac{B_1}{I} = \frac{\sqrt{3}\mu_0}{\pi}\left\{\frac{WH}{\left(W^2+H^2\right)^{\frac{1}{2}}} + \frac{H}{W\left(W^2+H^2\right)^{\frac{1}{2}}}\right\}$$

The saddle coil has 700 turns with W equal to 55 mm and H equal to 75 mm and therefore has a calculated field strength of 9.6 mTA^{-1}. A measurement was also performed on the coil and a figure of 10.5 mTA^{-1} was obtained which is very close to the calculated figure.

The peak magnetisation M_0 can be estimated using [11]:

$$\tilde{M}_0 = \frac{N_n \gamma^2 \hbar^2 \tilde{B}_0}{4kT}$$

Where N_n is the number of protons per unit volume (for water $N_n = 6.691 \times 10^{28}\ m^{-3}$), and T is the sample temperature. For a water sample at 20 ^{0}C and a B_0 field strength of 1 Tesla, $M_0 = 3.3 \times 10^{-3}\ Am^{-1}$. In our case we use the polarizing field strength ($B_0 = B_p$) to determine the magnetization. Using all of these relationships, a signal level of 4.8 µV peak was calculated for a water sample at 20 ^{0}C, a volume of 500 ml, a polarizing field of 18 mT and an operating frequency of 2.7 kHz.

The noise voltage (e_n) is thermal noise (Johnson) associated with the resistance of the coil and can be estimated using:

$$e_n = \sqrt{4k_B T \Delta f R}$$

With a bandwidth Δf of 225 Hz and operating at 0 ^{0}C we get an RMS noise voltage of 37 nV. This leads to an estimated signal to noise ratio of 130, which means that viable experiments should be able to be performed. The real signal to noise will be less, due to many factors such as: further noise introduced by the system and the environment,

and inhomogenieties in the polarizing and B_1 fields. Note that in this signal to noise calculation, the peak signal and the RMS noise values are used. Normally one would be consistent about the use of peak or RMS values, but in NMR experiments the signal and noise values are usually determined from the FID. The peak amplitude at the start of the FID is used as the signal value as it is easily measured and is constant regardless of inhomogeneities. It would be difficult to estimate the RMS value of the decaying FID, as it would depend on the duration of the signal used in the calculation. The RMS noise is usually determined from the peak to peak value of the noise in an experiment without a sample, and with NMR for which the noise has a Gaussian distribution, this peak to peak noise value is typically about 5 times its RMS value [22].

At resonance the B_1 circuit can be represented as a single voltage generator with a series resistor as shown in figure 3.12. The voltage generator output is Q times the sum of the signal (e_s) and noise (e_n) voltages and demonstrates an advantage of high Q coils. Unfortunately the source resistance is very large and is equal to Q^2 times the resistance of the coil [11] and therefore leads to the requirement of a high input impedance preamplifier. For the saddle coil probe, Q^2R is equal to 56 kΩ.

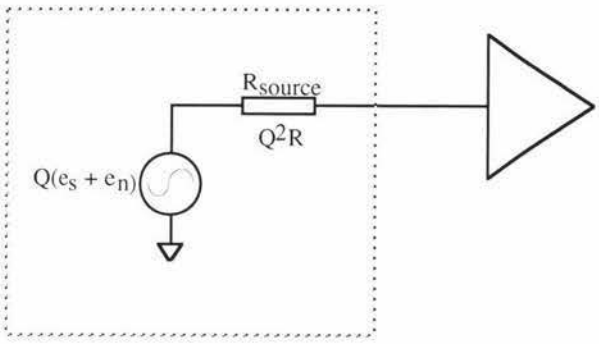


Fig 3.12 B_1 tuned circuit Thevenin equivalent at resonance, modeled as a single voltage generator with a series resistor.

One other consideration when designing a B_1 coil is the amount of power required and the duration of the RF pulses. For an inhomogeneous B_0 field there will exist a range of Larmor frequencies. The method used to excite this band is to use a sufficiently narrow B_1 RF pulse. The excited bandwidth is proportional to the reciprocal of the pulse duration, this relationship being based on the well known Fourier transform property, where a narrow pulse in the time domain results in a broad spectrum in the frequency domain. If the RF pulse is very narrow, the power level required will be very high as a certain amount of energy is required to tip the magnetisation. The B_1 field strength and the magnetisation tipping rate are related by the following relationship:

$$\omega_1 = \gamma B_1$$

Where γ is the gyromagnetic ratio and ω_1 is the tipping rate in radians per second. For a 90^0 ($\pi/2$) pulse we can derive the following relationship:

$$\vartheta = \frac{\pi}{2} = \omega_1 t = \gamma B_1 t \Rightarrow B_1 t = \frac{\pi}{2\gamma}$$

Where $B_1 t$ is constant and is the product of the field strength and duration of the pulse. With Earth's field NMR, B_0 is very uniform, therefore a long pulse can be used as we only need to excite a very narrow band of frequencies. A 3 ms 90^0 pulse length is used and therefore a B_1 field strength of 1.96 μT is required to tip the magnetisation within that time. The B_1 coil sensitivity is 10.5 mTA^{-1}, and therefore a current of 0.19 mA is needed. Remember that the probe is a parallel resonant circuit, so the actual current required to drive the probe will be Q times less than that in the coil itself. As Q is equal to 12, the actual current now required is only 16 μA, but also remember that the impedance is 56 kΩ, hence a voltage of 0.9 V is required. The power is only 15 μW and these calculations show how ordinary operational amplifiers can be used to drive the probe. In some NMR systems [42] narrow pulses of tens of kilowatts maybe required.

The ice sheet probe was initially constructed with a separate uniform gradient set consisting of a Maxwell pair of 100 turns of 1.6 mm diameter wire wound onto 352 mm diameter formers, separated by 313 mm (figure 3.14b). With a current of 2 A the gradient magnitude was 5.3 mTm^{-1}. The various gradient directions were obtained by suitably orienting the coils. This arrangement was used in the 1997 season and proved to be very time consuming to set up and was highly invasive. A large hole had to be cut around the pillar of ice with a chainsaw to enable the gradient coils to be inserted (figures 3.13 and 3.14). This operation took about half an hour and therefore left the ice core exposed to a different environment for a considerable amount of time before any experiments were performed.

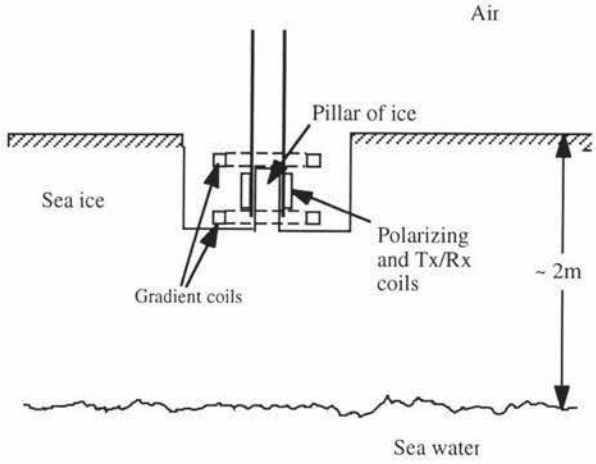
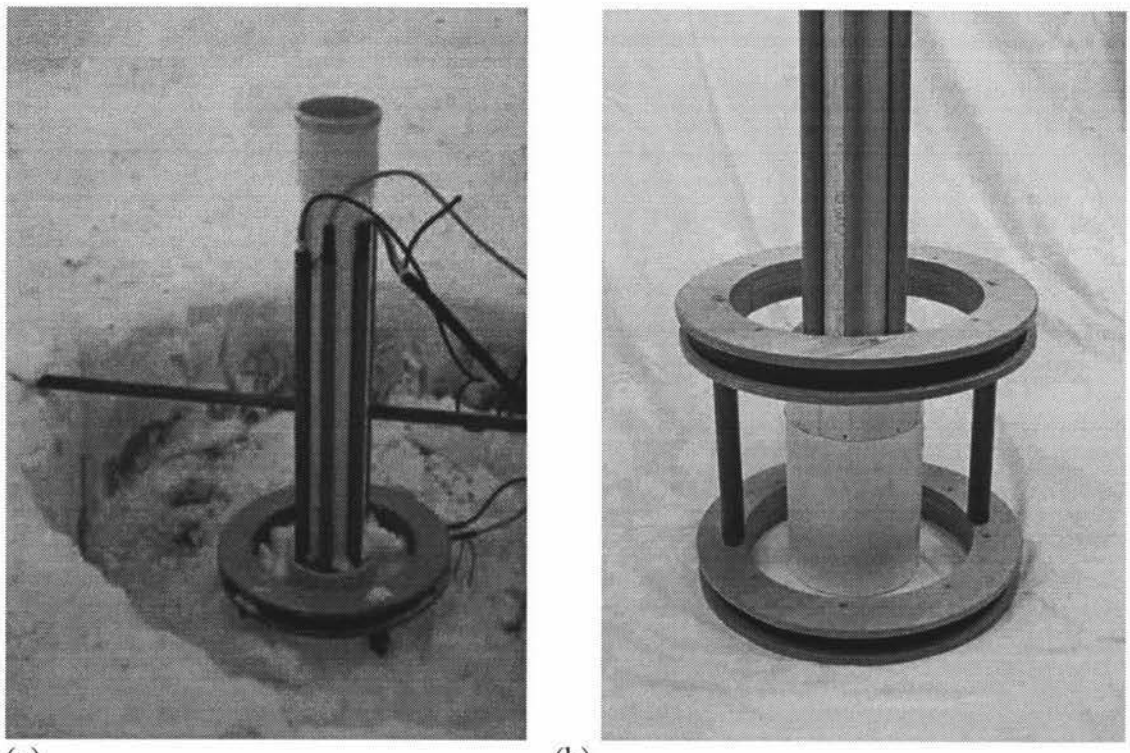


Fig 3.13 Sample arrangement with external Z gradient set used in 1997.



(a) (b)

Fig 3.14 Probe with external Z gradient set.

In the 1999 season, two sets of gradient coils (Z and X) were wound directly onto the ice sheet probe itself, leading to a much easier and faster sampling arrangement which worked very well (figures 3.15 and 3.16). It took only five minutes to cut the hole and get the probe ready for experiments.

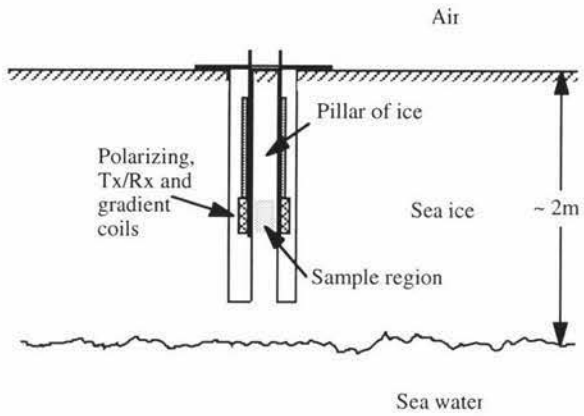


Fig 3.15 Sample and probe arrangement used in the 1999 season.

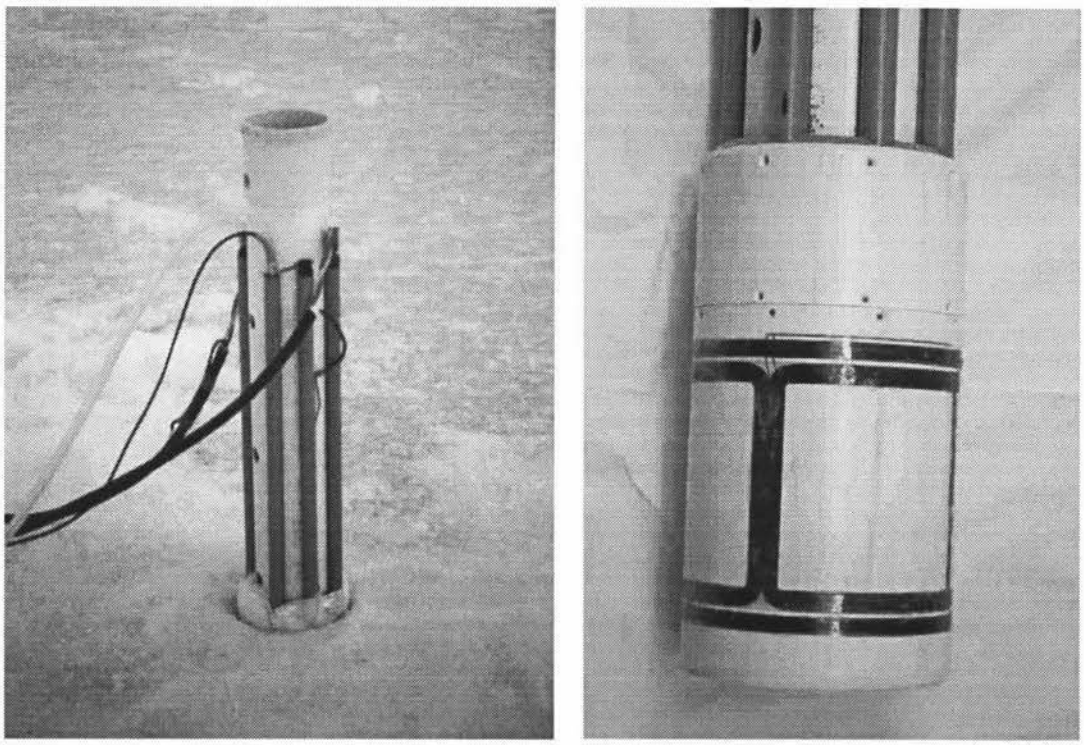


Fig 3.16 Probe inserted into ice sheet (a), probe with gradient coils (b).

For diffusion experiments highly uniform gradients are normally used, but due to the physical constraints of the probe, uniform gradients could not be obtained. However, the non-uniform gradients could be compensated for using a special data analysis technique that would give reliable diffusion data. The development of this technique was a joint project and will not be discussed in this thesis, but details may be found in reference 43.

0.71 mm copper wire was used in the new Z and X gradient sets which were wound into groves machined in the outside of the outer sleeve. The Z gradient set, as shown in figure 3.17a, consists of a pair of 154 mm diameter coils separated by 152 mm, with each coil having 26 turns. For 4 A, the gradient strength in the centre of the sample was calculated to be 11.8 mTm^{-1} and the resistance was measured to be 1.1 Ω. The X (or Y) gradient set as shown in figure 3.17(b), consists of a saddle type arrangement but with $\theta = 180^0$. Each half has 25 turns with a height of 125 mm and a diameter of 154 mm. The resistance was measured to be 1.6 Ω and a gradient strength of approximately 1 mTm^{-1} was calculated for a current of 4 A.

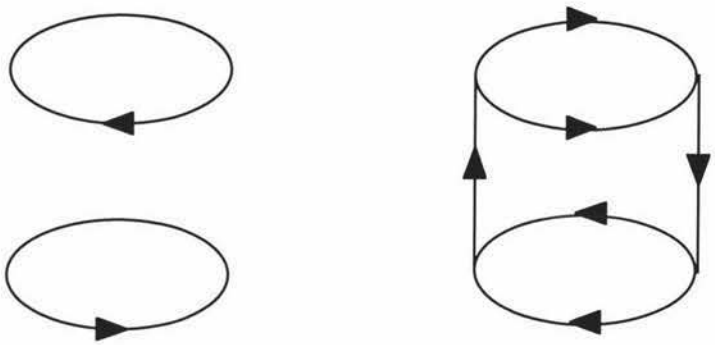


Fig 3.17 The Z gradient set (a) and the X or Y gradient set (b).

The spare 20 mm PVC pipe on the probe was used to provide support for the connections to the integrated gradient sets.

3.3.2 The transceiver section

With NMR the sample is firstly "excited" with a short radio (or audio) frequency burst, and then a small signal is detected. The part of the system that performs the "transmit" and "receive" operations is often called the "transceiver" and is very similar to that used in radio communications. With Earth's field NMR, the transceiver is very simple as no mixers, IF stages or quadrature detectors are required. The transceiver itself is split into two parts known as the transmit and receive sections. The circuit for the transmit section as shown in figures 3.18 and 3.19 consists of a band pass filter followed by an amplifier and a transmit switch. During the transmit phase of a pulse sequence, the system core generates a sinusoidal waveform that is fed into the input of the filter stage and a TTL signal to turn on the transmit gate.

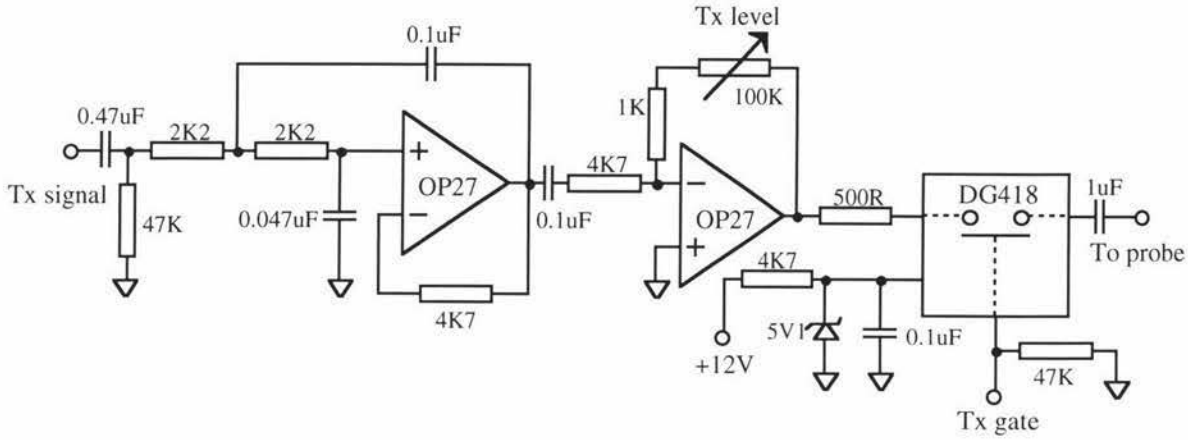


Fig 3.18 Transmit section.

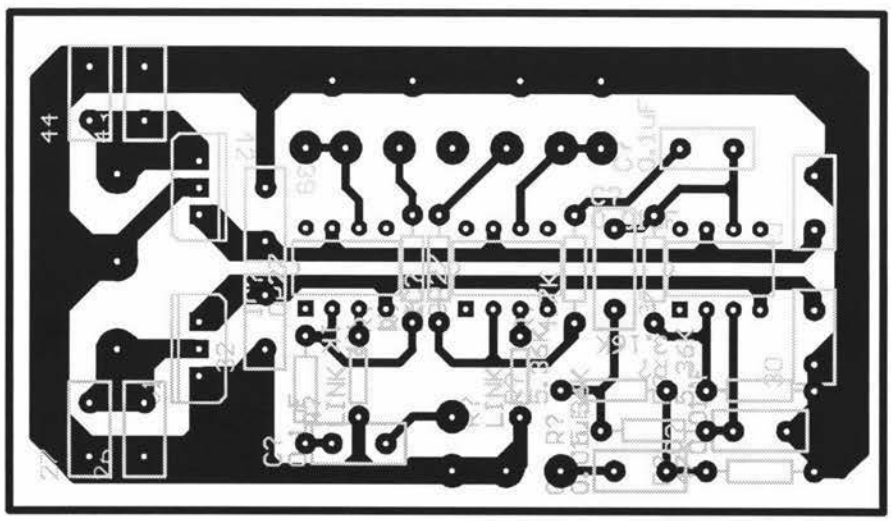


Fig 3.19 Circuit layout for Transmit section.

The filter is used to remove any noise before the waveform is amplified to the required output level. The transmit section must be disconnected during the receive phase, otherwise any noise produced by it would be detected and greatly amplified by the receive section, and would most likely swamp out any signals of interest. An analogue switch is used as it is much faster, smaller, and consumes much less power than a relay. The 1 µF capacitor is used to block any unwanted residual DC during the transmit and receive phases, and hence prevent any B_0 field perturbations.

The circuit for the receive section as shown in figures 3.20 and 3.21, consists of a preamp stage followed by a variable gain amplifier. The signal from the probe will be of the order of tens of microvolts, so a total gain of about 10^5 is required to produce a suitable signal level for the analogue to digital converter in the system core. The 180 pF capacitor improves the preamp's stability and noise figure by reducing the bandwidth. OP27 [26] operational amplifiers are used throughout the transceiver due to their low noise and generally good performance.

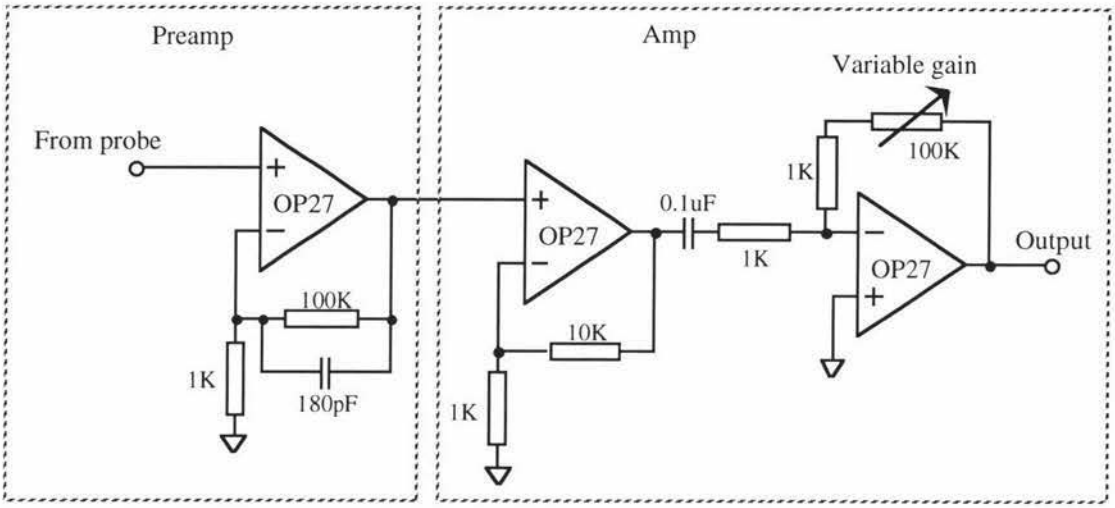


Fig 3.20 Receive section

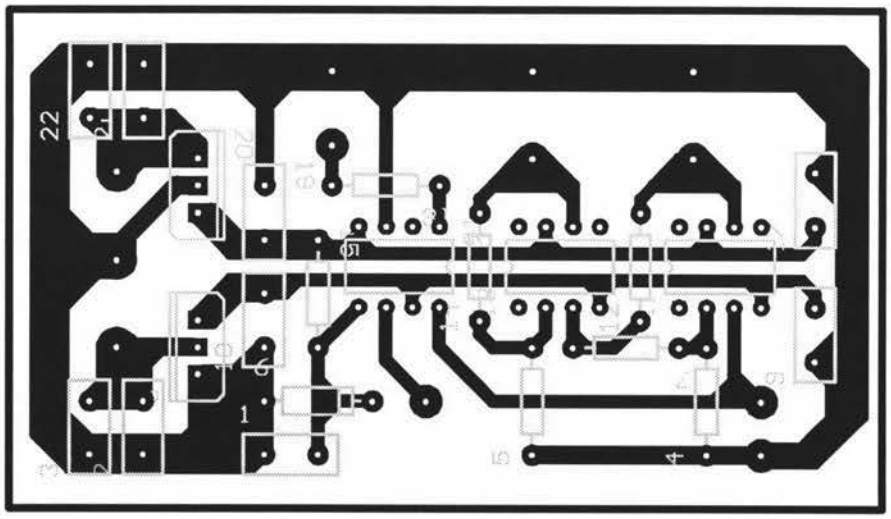


Fig 3.21 Circuit layout of receive section.

After the receiver was constructed, the gain was measured and it was found that at 2.7 kHz, the gain could be varied between 836 and 83600. The calibration for the A-D converter is such that it gives an output of $8x10^3$ for a 1 V input signal. Using this information it is possible (given a digitised signal) to calculate the signal level at the input of the receiver preamp.

3.3.3 The polarizing and gradient power supplies

At the beginning of each experiment scan, a current is passed through the polarizing coil to generate the B_p field required to increase the sample magnetization. Ideally B_p should be the same for all scans as often comparisons are made between them. Therefore a constant current source is required as the resistance of the polarising coil can vary depending upon its temperature. A current range of 0 to 10 A is also required to allow the system to operate with different probes. The requirements for the gradient coils are similar but with the addition of fast switching. A general purpose power supply module as shown in figures 3.22 and 3.23 was therefore developed that could be used to drive either the polarising or gradient coils. The "heart" of the power supply consists of an operational amplifier and a transistor power stage capable of sourcing 10 A continuously. The operational amplifier operates as an error amplifier within a control loop and provides the necessary drive to the power stage to maintain the requested current. A 0.1 Ω 25 W current sensing resistor is used to provide the necessary negative feedback to the operational amplifier. The high wattage resistor is needed to minimise any change in resistance due to the current passing through it and heating it up. The requested current level is determined by the voltage applied to the "Vref" or non inverting terminal of the operational amplifier. A voltage of 1 V corresponds to a current of 10 A. This level can be set by dialing the required amount on the 1 kΩ potentiometer and can also be gated on and off by applying TTL pulses to the "current control gate". When the power supply is gated off, a small negative voltage limited to -0.4 V, is applied to the non inverting terminal of the operational amplifier to make sure that it doesn't drive the output stage. The LM258 [40] operational amplifier was chosen as it can operate from a single supply, is able to withstand small -V on the inputs, and comes in a dual amplifier package.

When the circuit was constructed, a small very high frequency oscillation close to 10 MHz was found on the output. This was eliminated by placing a 100 Ω resistor and a 47 pF capacitor at the output of the operational amplifier. The combination of resistor and capacitor operate as a low pass filter, and therefore lower the high frequency gain of the control loop. Since the oscillation frequency was very high, the instability was most likely caused by parasitic coupling, resulting in some positive feedback. The 100 Ω resistor also limits the current from the operational amplifier and therefore protects it from overload. As the load is highly inductive, some further frequency compensation may be required to maintain stability, but with the new probe this was found to be unnecessary. However as narrow current pulses are required for the gradients, care must be taken not to over compensate and end up with slow rise and fall times. A 6 A current pulse was applied to the Z gradient coil, and the rise and fall times were measured to be about 200 µs. For highly inductive loads the rise time is limited by the maximum voltage that the power supply can source, which is itself limited by the rail voltage of 24 V. This rail voltage was chosen as it gave enough headroom to be able to drive 8 A into a 2 Ω load as well as being easily made up from standard batteries. The leads from the power supply units to the probe can be up to 20 m long and therefore contribute to the load resistance. They must therefore be taken into account when determining the rail voltage required. In Antarctica, 2.5 mm^2 low temperature cable was used that remained flexible at –30 °C.

One of the main points to consider when designing power circuits is the possible power dissipation. An estimate can be made based on the typical operating conditions, and in the case of the polarising coil requirements these are: a 24 V supply rail, 6 A into 1.8 Ω, and a 50% duty cycle. Using these parameters the average power that will be dissipated by the power supply will be about 36 W.

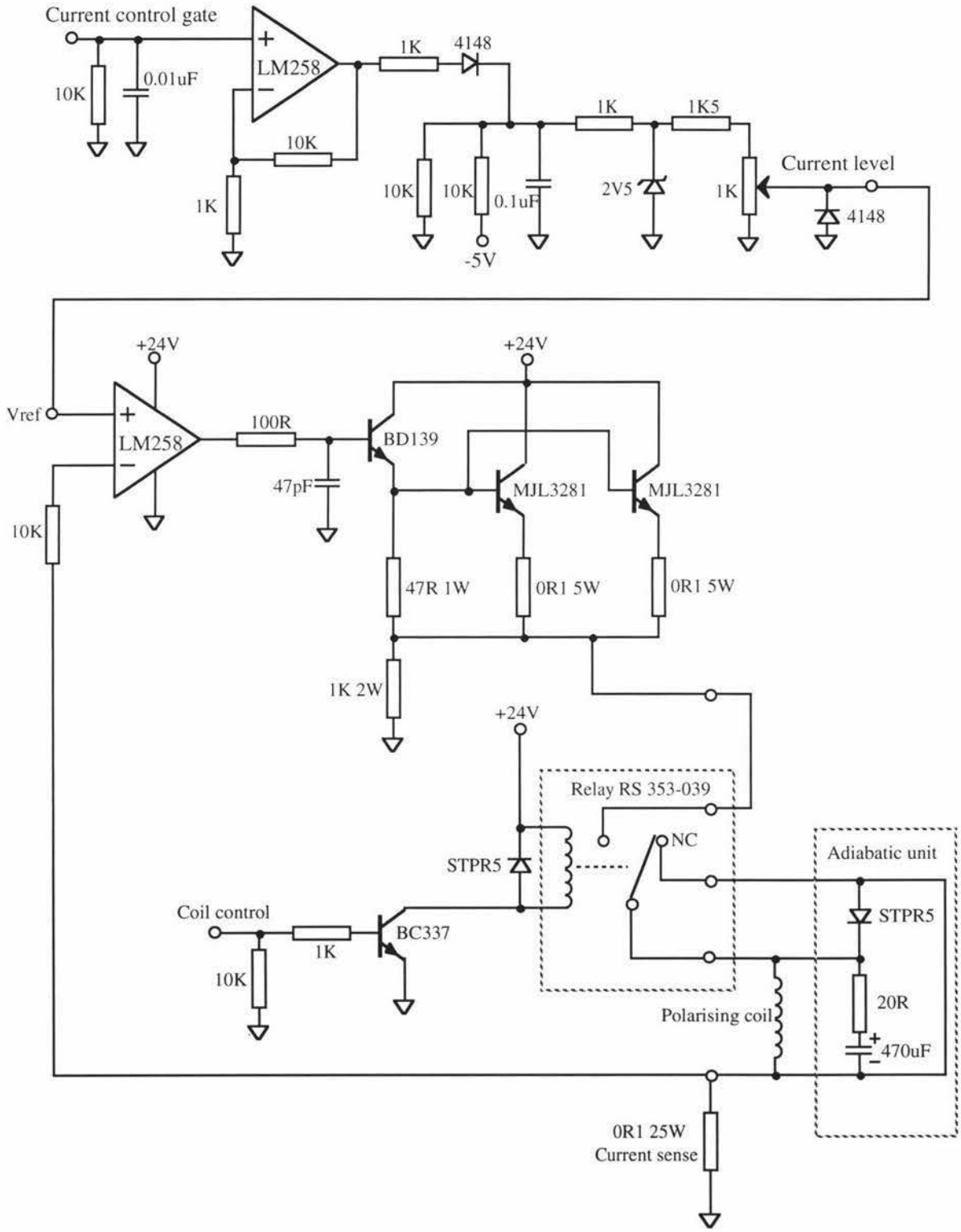


Fig 3.22 Constant current power supply with adiabatic unit.

To aid in dissipating the heat, a single heatsink with a rating of 0.59 ^{0}C/W is used for both power supplies. The power demands for the gradient power supply will be much less than for the polarising power supply, so a total power dissipation of about 40 W could be expected, resulting in an increase in the temperature of the heatsink of 24 ^{0}C. The MJL3281 [28] power transistors were chosen because, being an insulated TO-264 package, they can be easily mounted onto a heatsink, they have high gain, a rated power dissipation of 200 W, a continuous current rating of 15 A, and were designed for 100 W audio frequency applications. A single MJL3281 would be sufficient to handle all the current, but two were used in the final design so that the power supply module could be used for later applications. As two devices are used, a 0.1 Ω resistor is required in series with each to guarantee that they share the current equally.

A relay is used to disconnect and short out the polarising or gradient coils when they are not being used. This is done to guarantee that no current will be flowing within them and hence the B_0 field will not be disturbed. The relay is only switched when there is little or no current flowing, and this is done to protect the relay contacts. In the case of the polarising coil, the relay disconnects and shorts out the coil immediately after the end of the polarisation phase. A diode is placed in parallel with the coils to handle any back-EMF generated during the end of a current pulse. For adiabatic polarising field removal, a resistor and capacitor in series is also placed across the polarising coil to alter the current decay shape at the end of the polarising pulse. The combination of the resistor, capacitor, and diode is known as the adiabatic unit. In the future, shaped current pulses could be generated by computer controlling the current level through the use of a digital to analogue converter.

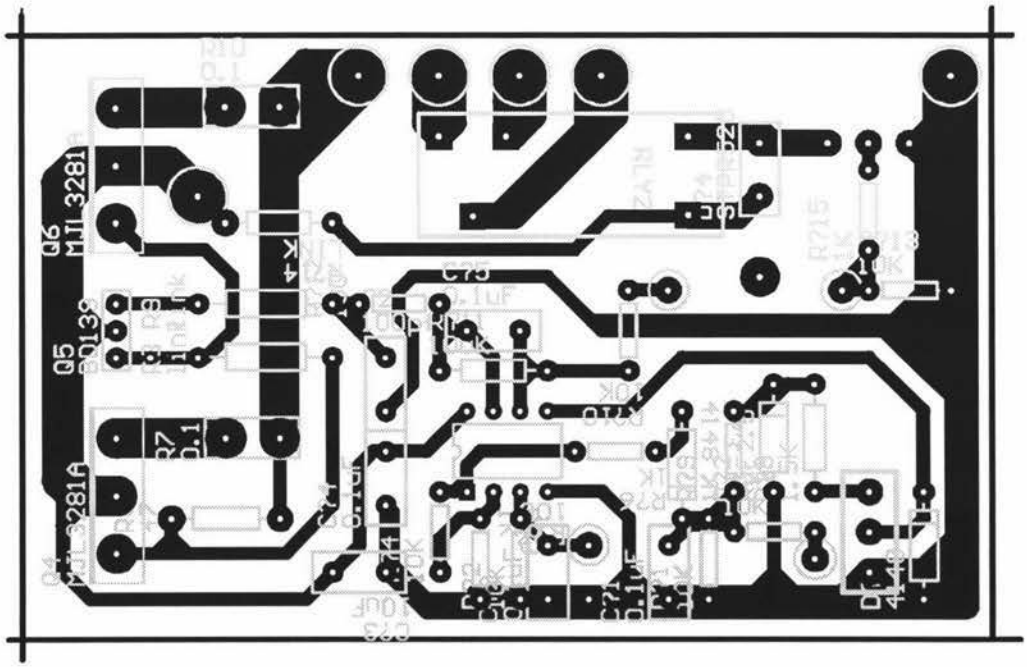


Fig 3.23 Circuit layout for the constant current source.

3.3.4 An overview of the system core

The system core is responsible for generating all the necessary signals for an experiment and is also responsible for digitising, processing, displaying, and storing the received FIDs. It is the "brains" of the whole system and consists of a laptop computer and a digital signal processor (DSP) based processing and control unit, as shown in figure 3.24.

The system core has been designed to be flexible, in that it can be used for a number of different applications by merely adding other units. It therefore forms the basis for low and medium field NMR applications where low cost and compact systems are required. By using a DSP a lot of functional blocks can be implemented in software instead of hardware, and therefore greater flexibility can be achieved. Two examples of this are the digital filtering of the FID and sinusoidal waveform generation using a digital oscillator algorithm. The DSP is interfaced to a number of units, the main ones being:

- Memory, to store the acquired signal and parameters and to act as a buffer between the DSP and the host computer.
- A 2 channel A-D/D-A converter with 16 bit resolution and a sampling rate of 48 k samples per second.
- A high speed digital interface to control/monitor other units.
- A laptop interface, to control the transfer of data to/from the host computer.

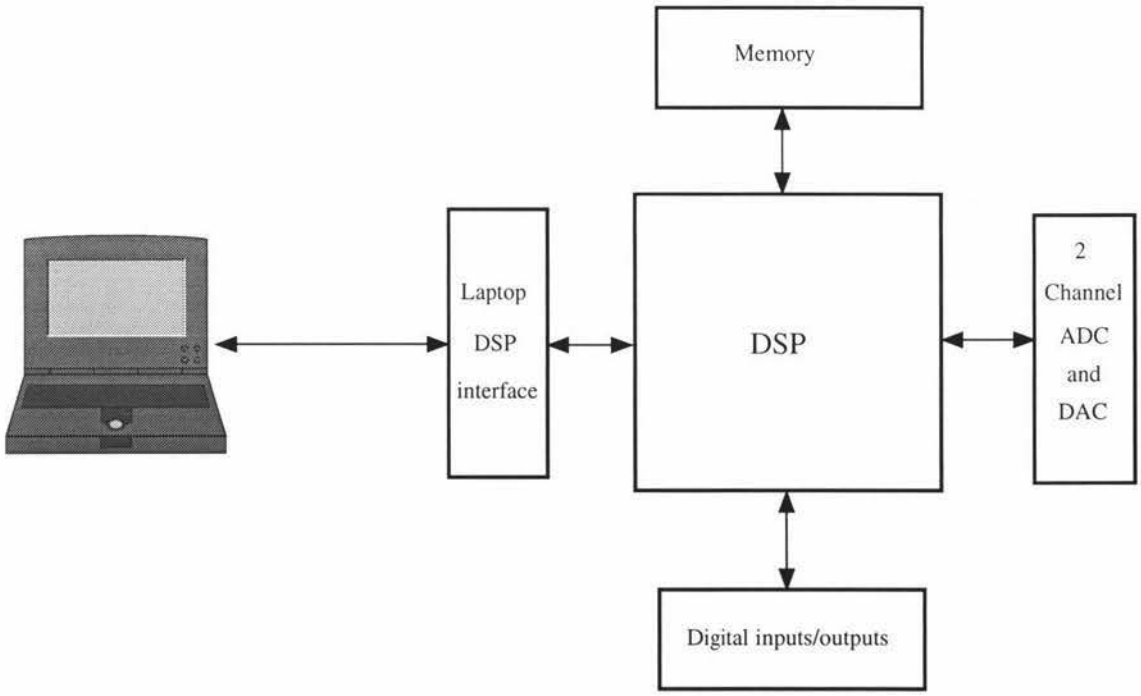


Fig 3.24 System core block diagram.

The pulse program takes the form of executable code for the DSP, written with an assembler or C compiler, allowing complete pulse sequence flexibility. The timing resolution of the DSP is currently 12.5 ns, but in a later version will be changed to 10 ns.

The software running on the laptop computer is called "Prospa" [23] and is a multipurpose NMR data processing package that has now been adapted to process and control applications. This enables the user to run the whole system and analyse the data within one application. Prospa has a built in scripting language, like "Matlab" [25], and coupled with the DSP programmability allows complete flexibility in setting up experiments.

The system core will be explained in greater detail in the next chapter.

3.3.5 Additional electronics and other components

As mentioned previously, the whole system, apart from the Laptop, batteries, pump, and probe is housed in two plastic boxes. One box houses the DSP part of the system core and is the "digital box", while the "analogue box" houses the transceiver, the gradient and polarising power supplies, and some other electronics. Two separate boxes were used for a number of reasons. Firstly the system core needs to be kept separate so that it can be used by itself for other applications. Secondly, physical separation of the units will help to minimise the coupling of noise generated by the high speed digital electronics into the very sensitive analogue receiver.

Figures 3.25 and 3.26 show how the units are arranged within the analogue box. The transceiver section is contained within a diecast metal box to provide shielding and good grounding. Within this unit the transmit and receive sections are separated by a copper sheet and have their own voltage regulators to further minimise coupling. An 8 V DC @ 300 mA supply for the DSP part of the system core was constructed by using four 3-Terminal regulators in series, mounted on a small heatsink and driven from the 24 V supply. This design is not efficient but I wanted to avoid using a switching regulator as it would have produced lots of unwanted electromagnetic interference. Unfortunately some negative voltages for the transceiver and constant current power supplies were required, and therefore a switching voltage invertor had to be used. It was mounted inside a small diecast metal box to contain any electromagnetic interference generated by the switching process. However this unit was not found to cause any problems. A diode was placed in series with the 24 V supply input to protect against accidental incorrect polarity when connecting batteries, and a fuse was also used to provide protection if a fault occurred. It is very important to use fuses when working with batteries, as they have very little source resistance and can therefore supply large currents.

To provide cooling for the probe, a small air pump with its own battery pack was used, which could be remotely operated via a control cable. A small relay was placed inside the analogue box to control the pump.

Ten-turn potentiometers are used for the gain and current settings so that they can be precisely set and then locked.

All the circuit boards within the analogue box are single-sided, are tin plated and were produced by myself with the aid of "Protel" [38], a circuit board design package.

The batteries that we used in Antarctica were sealed lead acid "Dryfit" batteries produced by Sonnenschein [41] and were found to operate well at temperatures down to $-30\,^{0}$C. The average current consumption of the system when performing experiments is about 4 A, so for eight hours continuous operation two 12 V batteries of at least 40 Ah capacity are required.

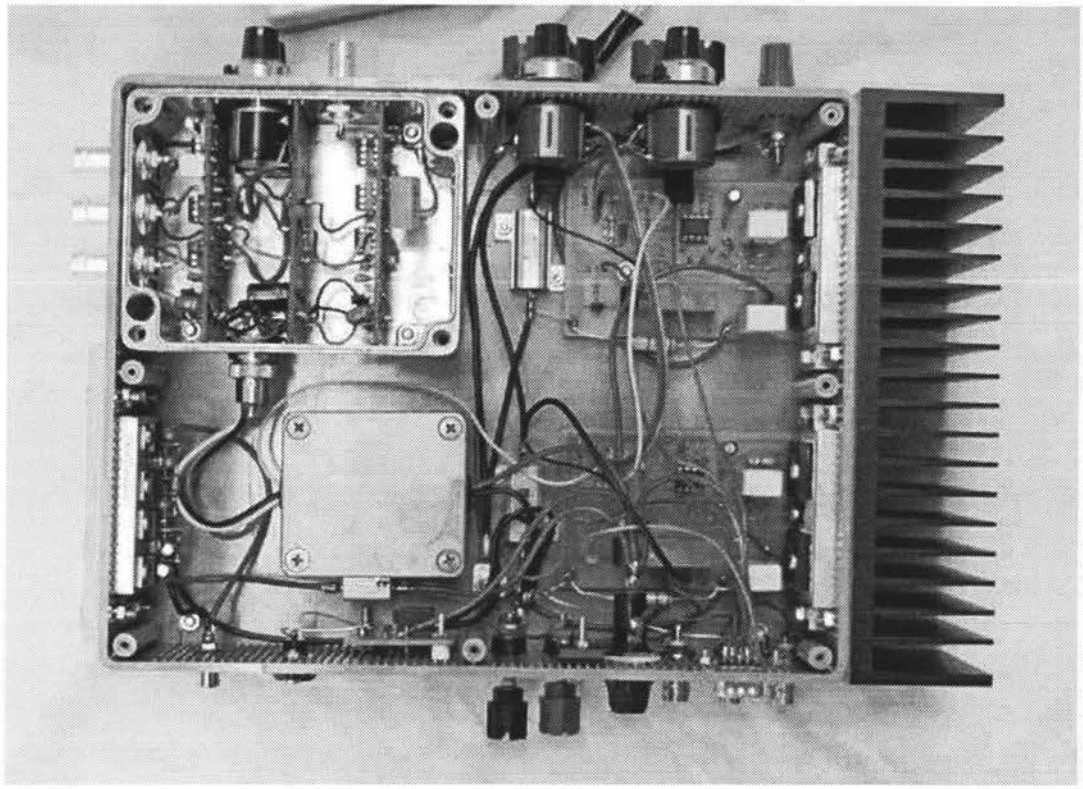


Fig 3.25 Transceiver and power supplies box.

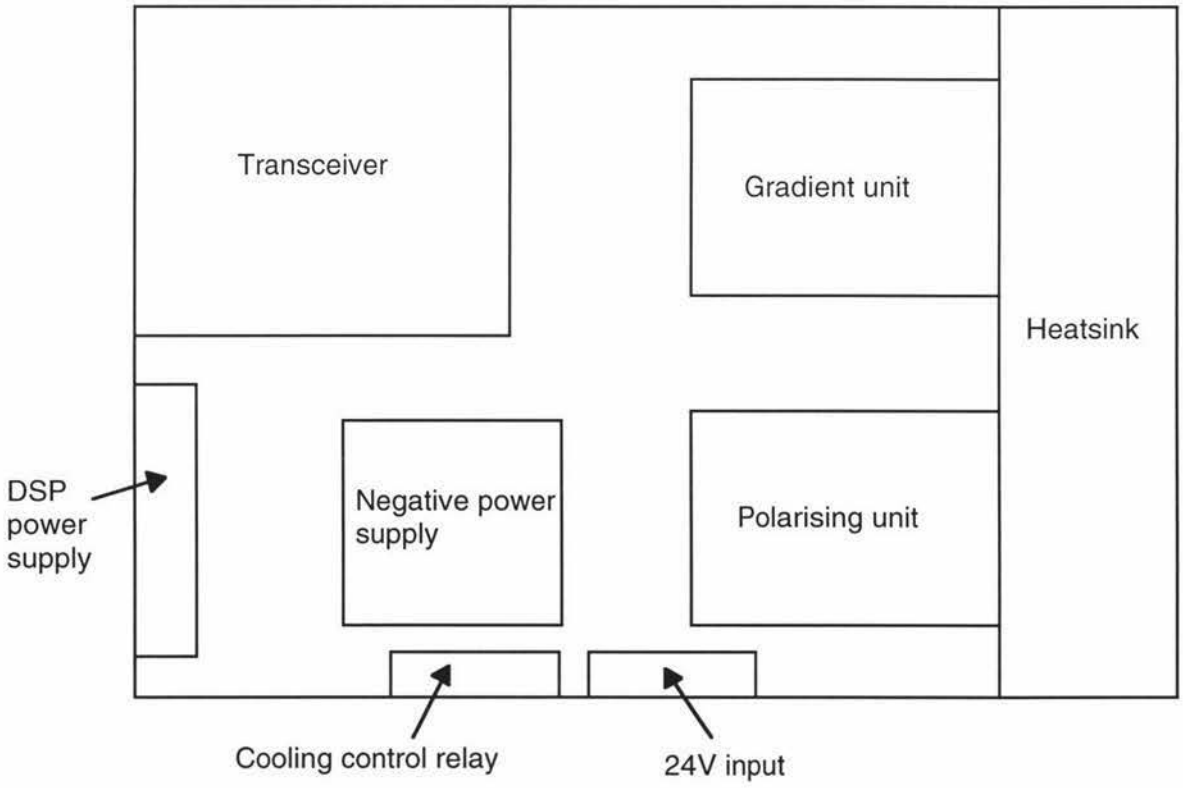


Fig 3.26 Overlay diagram of transceiver/power supplies box.

3.4 Earth's field NMR system performance.

3.4.1 Signal/Noise performance

The overall performance of the system is largely determined by three main factors:
1. Probe design.
2. Interference sources.
3. Receiver electronics.

As already discussed, the probe is the 'transducer' end of the system, and in our case it only generates a very small signal, which must then be greatly amplified by a very sensitive low noise receiver before it can be digitised. This process is very prone to signal degradation due to external interference, and noise introduced by the receiver, and therefore a lot of effort has been made to minimize these effects.

Interference is a big problem, especially when operating near mains powered equipment. The B_1 coils in the probes act as very good pick-up coils, and therefore screening is required. The solenoidal probe is far less sensitive to interference than the saddle coil probe, due to the screening effect of the polarizing coil. In the solenoidal probe the polarizing coil is in the same orientation as the B_1 coil, and when it is shorted during the receive phase it acts as a very good screen. This is not the case for the saddle coil probe, so other means of screening are required. When working in New Zealand an old copper hot water cylinder with one end removed is used to provide some level of screening by placing the drum over the entire probe. In Antarctica the background noise level is much less, so the copper drum is not necessary, however we discovered that the noise level increased dramatically when the saddle coil probe was inserted into the ice sheet. The ice sheet is electrically conductive and can therefore assist in coupling electrical interference into the probe. Grounded aluminum foil wrapped around the probe solved this problem by reducing the capacitive coupling between the probe and the ice sheet and resulted in a four fold reduction in pick up. Figures 3.27 and 3.28 show the effect of using the foil screening and how the interference was reduced. These data were obtained with the old NMR system at our 1999 Antarctic base camp with the instruments located 20 m away from the diesel generator that provided all the power to the remote site. The probe was located a further 20 m away from our instruments so that it would be as far away as possible from sources of interference.

The major interfering signals are 100 Hz spikes from rectified mains. These spikes cause the B_1 coil to ring and this shows up as the evenly spaced dominant peaks in figure 3.28. Periodically exciting a resonant circuit results in a spectrum with discrete peaks separated by the excitation frequency. The overall envelope of the spectrum is the frequency response of the circuit, and the main central peak is at the resonant frequency of the circuit. Since spikes cause the coil to ring at its resonant frequency it is a good idea to slightly tune the coil away from the Larmor frequency of interest so that the signal and interference can be more easily distinguished. Unfortunately this results in a slight decrease in signal strength depending on the frequency offset and the Q of the coil.

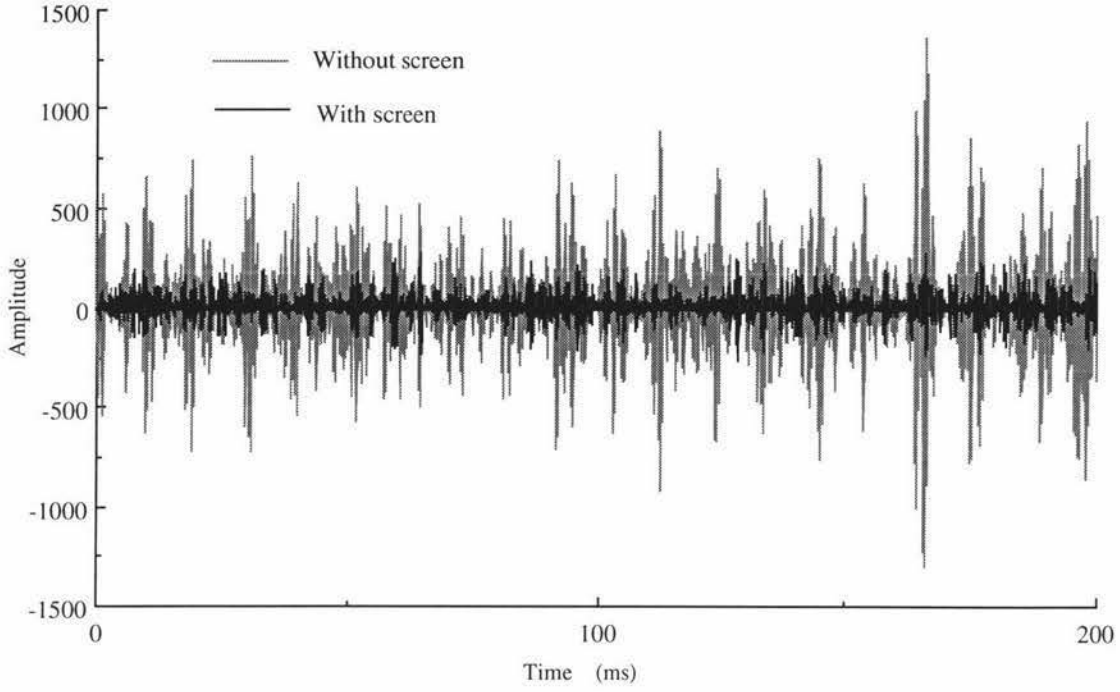


Fig 3.27 Noise signals at Antarctic base camp, with and without foil screening

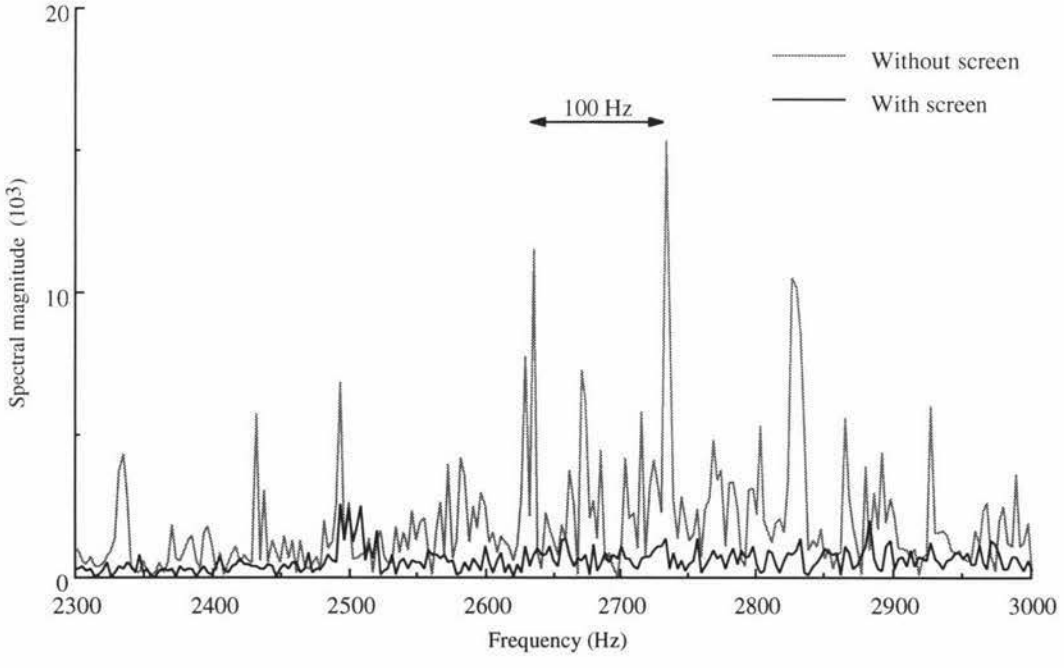


Fig 3.28 Noise spectrum at Antarctic base camp, with and without foil screening

With the new system I was able to travel away from the base camp and be completely free from mains operated equipment. Some noise measurements were performed and it was found that the noise (and interference) level was 20% less than that of the noise level at the base camp. This was not as good as I had expected and was probably due to insufficient distance between equipment and base camp. Figure 3.29 shows the spectrum of one of the noise data sets obtained at the remote site and clearly shows the frequency response of the digital filter in the system core. The noise level measured

equates to 58 mV RMS in the time domain, at the output of the receiver section, which is also the input to the A-D converter. Some experiments were also performed using a 500 ml water sample, and a peak signal level of 2.5 V was obtained. This leads to a signal to noise ratio of 43. This is much less than the calculated figure of 133 and is due to three main reasons:

1. Less signal due to probe B_1 and B_P inhomogeneities.
2. Losses in the transmission of the signal from the B_1 coil to the receiver.
3. Noise introduced by the receiver and interference.

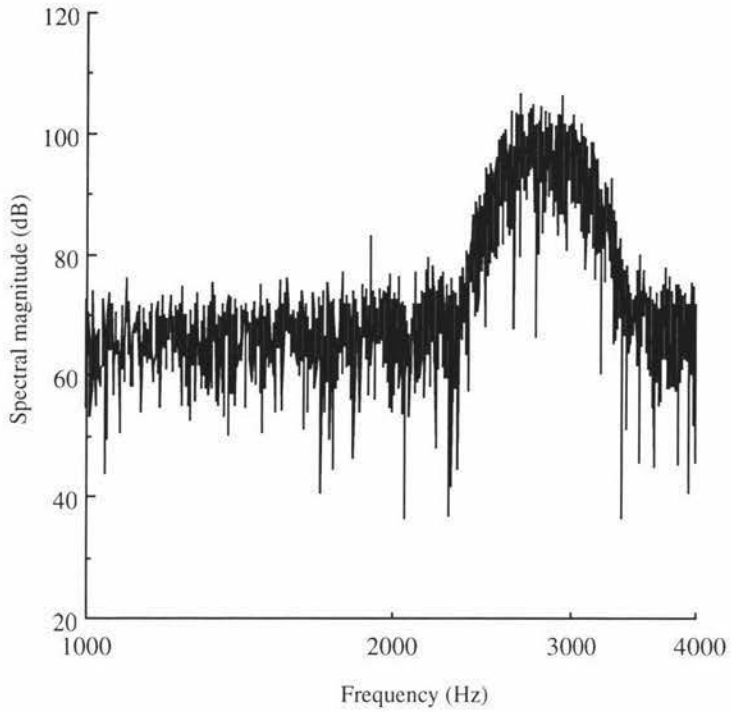


Fig 3.29 Noise spectrum at Antarctic field site.

The peak signal level of 2.5 V equates to an input signal to the receiver (gain = 83600) of 30 μV. Dividing by the coil Q of 12 gives 2.5 μV as the peak signal level induced into the coil. The calculated figure was 4.8 μV and is about a factor of two greater than actually received. This is quite reasonable considering that the calculations assumed homogenous B_1 and B_P fields and no losses.

The noise level of 58 mV RMS equates to an input signal to the receiver of 694 nV. Dividing by the coil Q of 12 gives the noise level produced by the coil resistance as 58 nV. The calculated noise level was 37 nV and is roughly a factor of two less than that received. The increase in the noise level is due to external interference and noise introduced by the amplifier, and as the level is so small it is hard to distinguish between "real" noise and interference.

Back in New Zealand I hunted around to find an electrically quiet location and finally ended up in the Kahuterawa valley, which is about 20 km from the University. The spot chosen was at the end of a road that followed a stream into the foot hills of a mountain range and was a good distance from any power lines and other sources of interference. It is also a very popular spot for trampers and mountain bikers as some tracks lead off from there. The area is surrounded by native bush and has a nice stream

running through it, so it is not only electrically quiet but is also a very pleasant place to work. Some experiments were performed with a 500 ml water sample and the results are shown in figures 3.30 and 3.31.

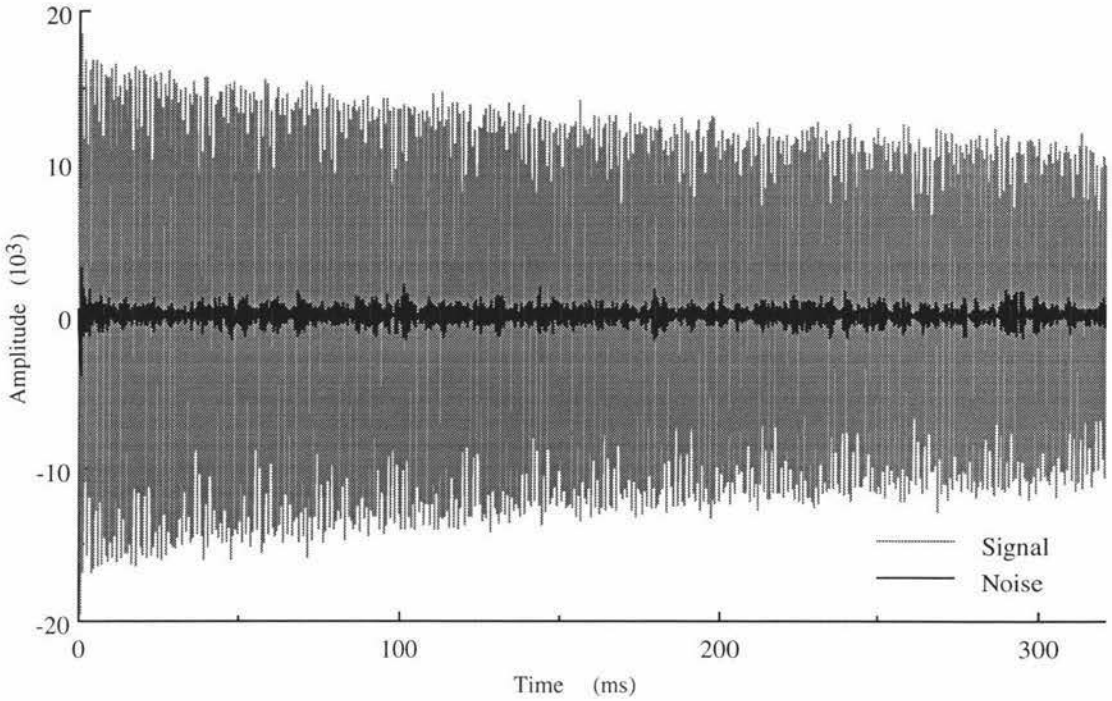


Fig 3.30 Signal and noise data obtained in the Kahuterawa valley.

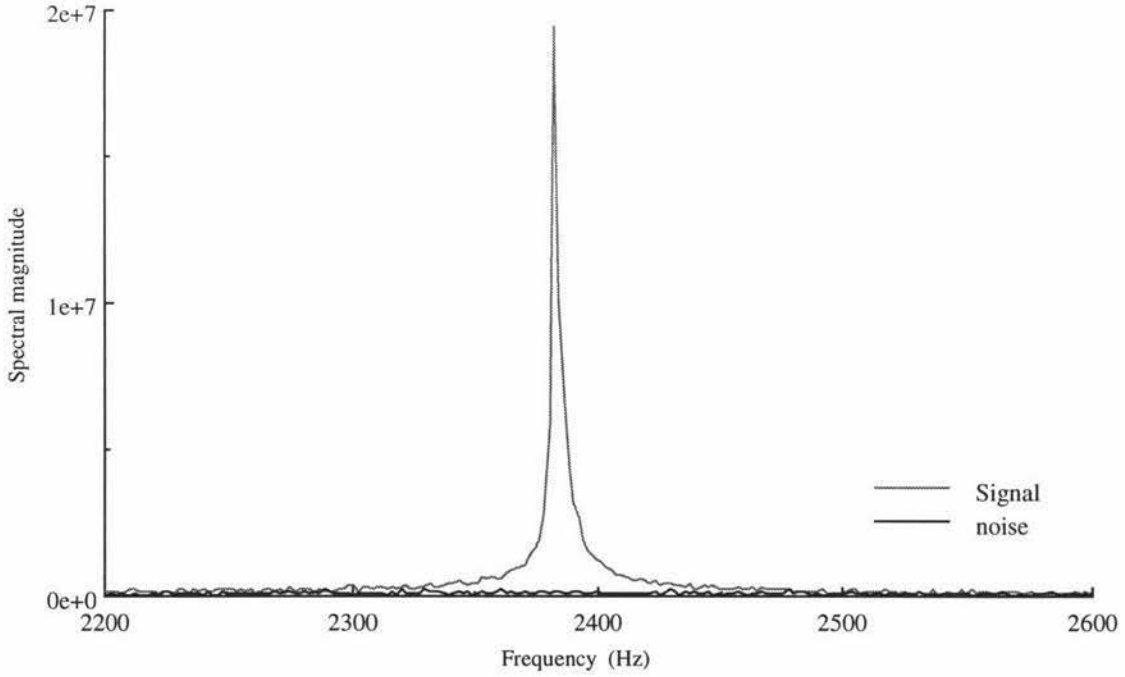


Fig 3.31 Spectrum of Kahuterawa signal and noise data.

When the copper drum is used as well as the foil, the noise level at the Kahuterawa site is slightly less than that at the Antarctic base camp, which verifies the suitability of the location for performing Earth's field NMR experiments. The signal level is slightly down due to the reduction in the strength of the Earth's (B_0) field (2.4 kHz instead of 2.7 kHz). The B_0 field is remarkably uniform, demonstrated by the long FID and corresponding narrow peak in the spectrum. An estimate of the S/N is 31.

For comparison, signal and spectra from a sea ice experiment are shown in figures 3.32 and 3.33. These were obtained using the old system and the saddle probe at the Antarctic base camp. Sea ice has about a 4% water content, therefore the signal level is about 4% of that for a water sample. This results in very poor S/N, so we use signal averaging to improve the S/N to a satisfactory level. With signal averaging, the signal level will accumulate in a linear fashion, but the noise will only increase by the square root of the number of accumulations. Signal averaging requires excellent experiment repeatability as signals from each scan need to have the same frequency and phase for them to combine constructively. This makes the requirements of precision timing and a constant magnetic environment essential. As a consequence we try to minimise the experiment time as there is always the possibility of frequency drift due to a slight change in the magnetic field or drift in the system electronics master clock. The Earth's magnetic field is very stable but can be perturbed by metallic objects so it is very important to make sure that no vehicles in the vicinity are moving during an experiment. A crystal oscillator is used as the master clock to minimise the drift in the B_1 excitation frequency and the sampling frequency of the digitiser. With sea ice the possibility of melting, due to the heat generated by the polarizing coil, is a major factor that limits the experiment time. So there is a trade off between S/N and experiment time, which means that we can't signal average for ages and get wonderful data. Therefore we want an NMR system that has the highest possible S/N performance, but this is controlled by the sample access requirements and hence the type of probe that is used.

For the sea ice experiment 24 scans were performed, resulting in an experiment time of four minutes. The S/N of the resultant data is only about 7 which is what we would expect for sea ice with 4% free water. The signal may look very noisy, but in the frequency domain a clearly defined peak emerges out of the noise floor. By picking out the peak and integrating it, one can get a good indication of the initial signal level in the time domain. Picking out the peak is like applying a very selective narrow band pass filter, but in this case it is done in the frequency domain. As the peak frequency is usually very stable during a series of experiments, the process can be automated through the use of appropriate software ("Prospa"). Working in the frequency domain can sometimes be advantageous.

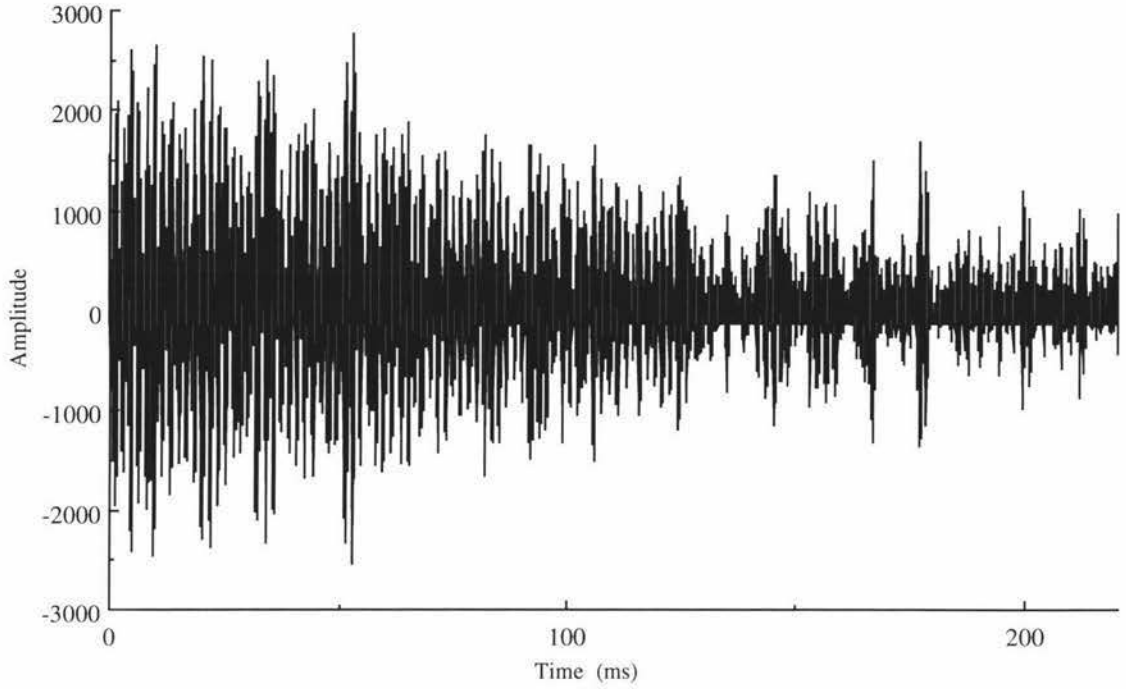


Fig 3.32 Signal from sea ice at Antarctic base camp (24 scans).

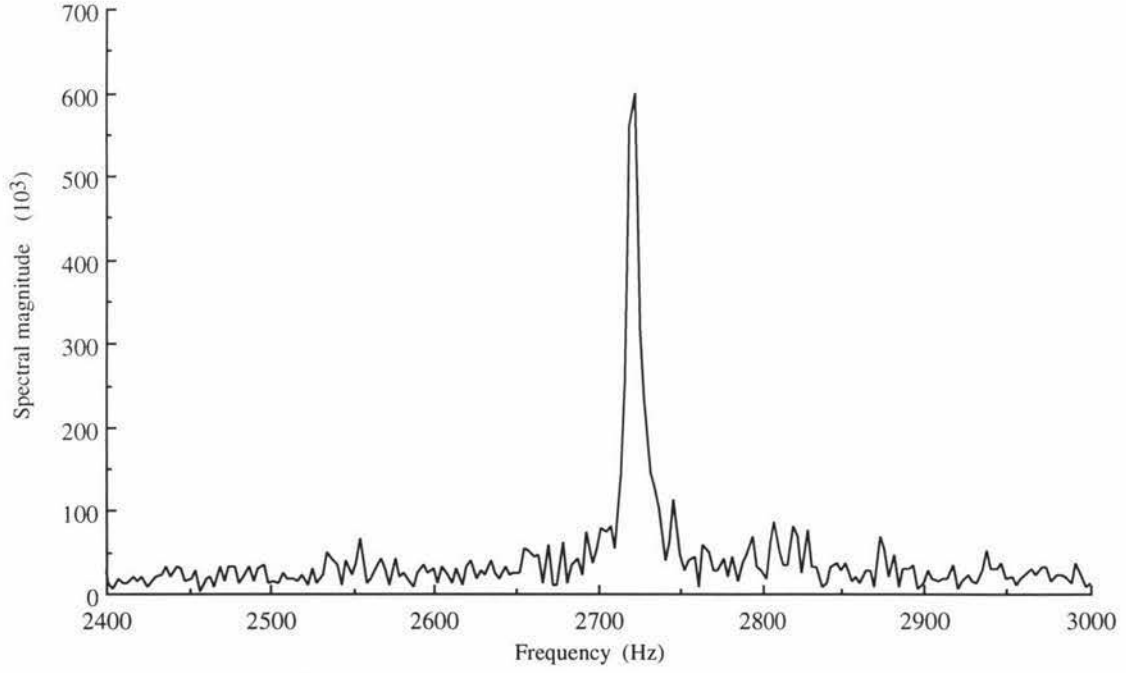


Fig 3.33 Spectrum of sea ice data, obtained at Antarctic base camp (24 scans).

To get an indication of the success of the saddle probe design, a comparison with the solenoidal probe is useful. The solenoidal probe has a calculated polarising field strength of 19 mT for an applied current of 6 A and a B_1 field strength of 24.7 mT/A. For a sample volume of 500 ml a signal level of 13.3 μV peak would be expected. This is 2.8 times the calculated figure for the saddle probe. The B_1 coil has a Q of 12 and a resistance of 236 Ω resulting in an expected noise voltage of 28 nV RMS, again better than that expected for the saddle probe. Using these figures the expected signal to

noise ratio for the solenoidal probe is 475, 3.5 times that expected for the saddle probe. In Antarctica some measurements were done to compare the performance of the two probes, and it was found that the solenoidal probe signal level was 2.3 times that of the saddle probe. The noise level from the solenoidal probe was about 20% higher than that of the saddle probe. This is not what was expected as the solenoidal probe has less resistance, indicating that it is most likely to be interference and that some screening is required. Overall it was found that the solenoidal probe had about a factor of two better signal to noise performance.

3.4.2 Suggestions for improvements

To improve the overall signal to noise performance, we need to achieve either, but preferably both, of the following:

1. Increased signal strength.
2. Reduction in amplifier noise and less sensitivity to external interference.

One way to increase the signal strength would be to simply increase the polarising current. This would not require any modifications to the probe, but depending on the current, it may require modifications to the power supply. Another way to increase the signal strength would be to keep the current the same, but wind a new polarising coil with a greater number of turns and thinner wire. This would increase the resistance of the coil and again may require modifications to the power supply, as higher voltages would be needed to drive the coil. This demonstrates that the polarising coil and the power supply need to be designed simultaneously, as power supplies are usually designed for specific load impedances. A too high rail voltage and a low impedance load will result in excessive power dissipation in the power supply, and a power supply with a low rail voltage will not be able to drive a high impedance load. Unfortunately all of the methods mentioned above for increasing the signal strength lead to an increase in the power dissipation in the polarising coil and hence heating of the sample. This is a problem for Antarctic work but for other applications it may not be.

As mentioned previously, interference is a big problem. Some things that could be done that may improve the performance of the system are:

1. To replace the aluminium foil with a permanent copper sleeve. Some copper sheet could be wrapped around and glued to the outside of the outer sleeve and then connected to the probe's earth point. Vertical slots would be needed to be cut in the copper sheet in order to prevent eddy currents, due to the application of the polarising and B_1 fields.
2. The cables between the probe and the electronics also pick up interference so they should be as short as possible. We have always used 20 m cables as we wanted to keep the probe as far away as possible to minimise the interference from the mains powered equipment. With the new system we can now work a lot closer and so the cable length can now be reduced to less than 5 m.
3. For the signal, coaxial cable is currently used between the probe and the electronics as the preamp input is single ended. The cable could be replaced by a twisted pair and a differential input preamp. This should reduce the pickup but

would require a much more complicated preamp and transmit switch arrangement and may end up with an increased amount of noise introduced by the electronics.

4. A preamp could be placed in the probe to reduce the effect of the pick up in the signal cable. This has been tried before but some of the components were magnetic and caused large perturbations in the B_0 field. The preamp was mounted in the connections chamber so was very close to the sample but it could be possible to mount one at the top of the probe by the connectors. Batteries and the transmit switch would also have to be included with the preamp.

The preamp plays a very important role in any receiver system as its noise performance dictates the noise performance of the whole receiver. Therefore every step needs to be taken to minimise the noise introduced by the preamp. The present preamp uses an OP27 operational amplifier. This amplifier was chosen for a number of reasons, namely: its good noise performance, its low input bias current, and its availability. The low input bias current feature is important, as any current in the B_1 coil will perturb the B_0 field. However this effect can be eliminated by using a suitable coupling capacitor. Some further investigation into the performance of the preamp could be undertaken, and one thing that could be looked at is separating the preamp from the rest of the receiver by placing it in a separate metal case. This would improve its stability and reduce the possibility of interference coupling from the rest of the receiver.

From all these points one can see that some further work could be done to improve the performance of the system. It is a case of optimising every little thing to squeeze out the last little bit of signal to noise improvement.

4.0 System core

The system core is the controller, data acquisition and processing part of the NMR system, and is therefore responsible for generating all the necessary signals, and the digitising, processing, displaying and storing of the captured FID data. In commercial NMR systems it typically consists of a host computer and some additional electronics.

4.1 Requirements

In order to design a system core that would be flexible and powerful enough to suit a range of NMR applications, I canvassed a number of people and also studied the details of some NMR systems, and came up with a list of requirements that my system should have. These design criteria are listed below:

- The system core should have a control/acquisition unit that is separate from a host computer. This is to allow different computers to be used as the user interface for the system. Experience with scientific instruments has shown that it is not a good idea to have the control/acquisition unit and the computer all in one, as often, the computer rapidly becomes outdated technology and therefore limits the possible performance of the instrument. Using a separate control/acquisition unit allows the host computer to be upgraded when necessary.
- Ideally the control/acquisition unit should be host platform independent and be able to be used with a wide range of computers and operating systems.
- A standard interface should be used between the control/acquisition unit and the host computer in order to meet the previous criteria. This interface should also be fast, reliable and not involve complicated cabling.
- The system core should be as flexible as possible, so that it can be used as the basis for many different NMR and even non-NMR systems. Because of this it will probably be overkill for Earth's field NMR requirements.
- The control/acquisition unit should have a range of digital I/O interfaces for example, a flexible high speed data I/O interface capable of being used with a wide range of A-D/D-A units and able to handle a high throughput of data, and some high speed digital outputs to be used in conjunction with the pulse program to control some other units within the NMR system.
- Good dynamic range is necessary for the A-D conversion stage and the signal processing stages that follow it.
- The acquisition part should be able to take advantage of signal processing techniques such as signal averaging, oversampling, undersampling and digital filtering.
- The control part should be able to generate a sequence of pulses and waveforms on multiple digital and analogue channels with a timing resolution of 10 ns.
- The system user should have the maximum flexibility when designing NMR experiments and pulse sequences. Looping and nested looping and the ability to rapidly alter parameters between individual experiments through the use of parameter tables, such as delay tables, are essential if multi-dimensional NMR experiments are to be performed.
- A single software package running on the host computer should be able to control the system and analyse the data.
- Rapid updates on the computer screen of acquired data should be possible.
- The system should be compact so that it can be portable.

4.2 Suggested design

To use a DSP as the central element in the control/acquisition unit was a decision made right at the beginning. The reasons for this are as follows:

- DSPs offer a large amount of processing power at low cost, and some are housed in packages that are relatively easy to work with.
- DSPs are designed for real time processing of signals and so are ideal for digital filtering.
- 100 MHz processors are available that will give the required timing resolution of 10 ns.
- DSPs, being programmable devices, are very flexible and should easily handle complicated multidimensional pulse programs.
- A range of intelligent high speed parallel and serial interfaces are also provided with most DSPs, so that they can easily be connected to other devices such as A-D/D-A converters with minimal glue logic.

For the Antarctic work we wanted an NMR system that was portable and battery powered. Obviously a laptop computer would be the most suitable as a system host and so this decision was also made at an early stage. The core of the design was therefore a laptop computer and a DSP based control/acquisition unit. The next phase was to determine what type of interface to use as the communications link between the two units.

A number of different interfaces exist, some are standard such as Ethernet, and are available on many different computer platforms, and some are platform specific such as the PC parallel printer port. When choosing an interface, many things need to be considered, such as the throughput, the hardware/software complexity at either end, and the physical cabling required. In the end, there are many tradeoffs between such things as complexity/performance/cost, hardware/software and standard/non-standard/developer support.

For many years most digital I/O devices were easy to program, one would simply write directly to the I/O hardware using in or out (Peek and Poke) commands or memory reads or writes. This was simple, fast, and efficient and required a minimal learning curve. Today with modern operating systems, direct reading and writing to hardware is forbidden and now the only way to communicate with hardware is through the use of device drivers. A device driver is effectively a trusted operating system extension that has more privileges than an ordinary user mode application. Its task is to manage the device and to provide a software interface to users' applications. Device management is important with multi-user, multi-tasking operating systems, as the device may be shared among multiple applications and/or threads. Device drivers were also introduced to prevent applications from the unauthorized use and/or misuse of devices that could potentially bring the whole system down. Through the use of device drivers, operating systems should become more robust and stable. However a big disadvantage of the device driver approach is the huge overhead it places on the system and hence results in a much reduced performance of the device and the application. The choice is stable but slow, or fast and dangerous. It looks like the trend is for more conservative systems where reliability is the main goal. Therefore device drivers are something that we have to learn to live with. One of the unfortunate things

about using device drivers to access hardware is that you don't know exactly what or when things may happen. You may send a request to the driver, but it will decide when it will do it, depending on the other tasks that the system has to perform. This means that guaranteed timing and precise synchronisation of events is impossible. Taking all this into account, any hardware interface that is used will need to be well supported by a good device driver that is optimised for good data transfer rates and has a straight forward application programmer's interface (API).

I evaluated a number of different interfaces in order to choose one that best met my design criteria, and these are listed as follows:

- PC parallel printer port. Unfortunately this is limited to PC's only and will be soon redundant. Manufacturers want to move away from using this port, as it takes up a lot of board and panel real estate, and requires a large expensive cable. The support for Windows98/2000 is not very good, as there are no readily available general purpose device drivers.
- RS232. This interface is too slow and is also becoming redundant, as it is no longer supported by some platforms, such as the Apple iMAC.
- Ethernet. This is a possible solution, as it is very common and available on most computer platforms. But there are a number of big drawbacks, one of which is that the bandwidth can't be guaranteed if the system is connected to a network. In addition it is not supported on some laptops unless a PCMCIA card is used, and it requires a large amount of complicated software at both ends to support it, as there are many protocol layers to work through and hence massive overheads.
- USB. This is a fairly new interface that is being rapidly adopted by most platform manufacturers and therefore will be around for a while. USB is a 4 wire serial interface that is cheap to implement, compact and well supported by Windows98/2000, MacOS and Linux. Reasonable data transfer rates of 12Mbits/s can be obtained. USB peripheral chips are available that can be easily interfaced to DSP devices and there is also a whole host of developer information available.

I chose to use USB as the interface between the laptop and the DSP based control/acquisition unit, as it looked to be the most suitable for my application.

One of the design criteria was that the control/acquisition unit should be able to be interfaced to a range of A-D/D-A converters, so that the system could be easily adapted to a range of NMR applications. The idea therefore was to make a DSP board with a USB interface, and also with most of the DSP I/O lines brought out to a connector, so that it could be connected to a separate A-D/D-A board. This way only the A-D/D-A board would ever need to be changed. Using this design would give me a fairly general purpose signal processing and control box that could be used for many applications other than NMR, such as the controller for a scanning tunneling microscope (STM).

For NMR, the pulse programs would be executable program code written for the DSP using an assembler or C compiler on the laptop. The code would then be downloaded to the DSP board when required. For the user interface and controlling application on the laptop, I will be using a modified version of Prospa, an NMR processing application, written in-house. Using Prospa will save considerable amounts of time as I will only have to write code that deals with the DSP board, and not a whole visual interface.

4.3 Implementation

I had my plan for the system core, which was to build a DSP board with a USB interface, but I soon realised that I wouldn't be able to get it ready in time for the trip to Antarctica. That date was set, so the main aim changed to getting a system going as soon as possible. Therefore I had to come up with an alternative design that made use of some readily available units. Instead of building my own DSP board, I purchased an evaluation board. A communications link to a laptop computer was implemented by purchasing a PCMCIA PIO24 parallel digital I/O card [46] and building an interface board to go between the I/O card and the DSP board as shown in figure 4.1.

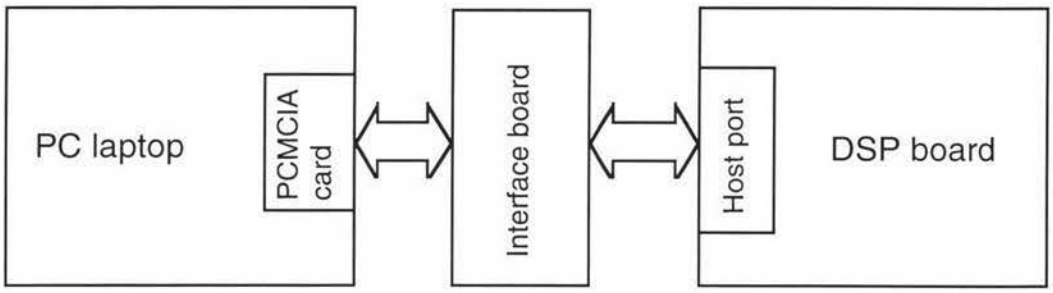


Fig 4.1 Alternative system core design.

This alternative design met most of my criteria, but one major exception relates to the DSP/laptop interface. This design is limited to PC's only, as a non-standard interface is used. A minor deviation from my criteria is that an 80 MHz, instead of a 100 MHz clock is used for the DSP as this is the maximum rate that is available on the evaluation board.

4.3.1 Acer laptop

The laptop used is an ACER EXTENSA 502T [34] with Windows98. One of the main requirements for the computer was that it should have an active matrix display. This is necessary because we want reasonably fast screen updates, and because of the experience of others using computers in Antarctica who have found that active displays still work well below –20 °C. The Acer was chosen as it was one of the cheapest available and yet still had an active matrix display. The harsh Antarctic conditions were also something that had to be considered, so I looked at some of the ruggedized laptops available, but was shocked by the price. I could buy three ordinary laptops for the same cost, which meant that we could take the risk and afford to lose one. Anyway the main problem was temperature and I don't think a ruggedized one would have been any better off, as they are only stronger and splash proof. Some mechanical and thermal protection can be provided by using a foam lined carry case. So in the end we purchased a reasonably cheap laptop (figure 4.2) just in case we destroyed it.

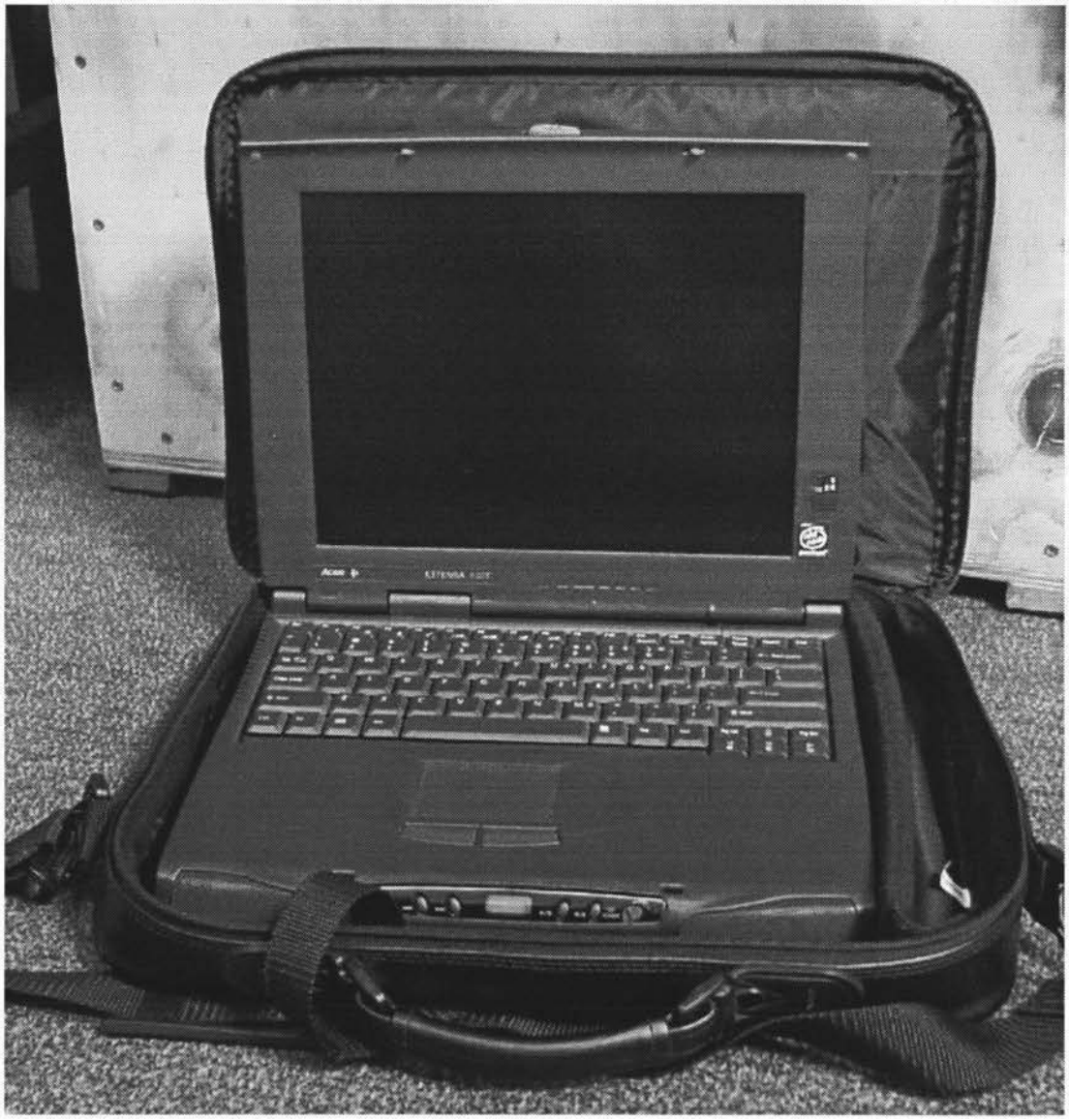


Fig 4.2 The ACER EXTENSA 502T with carry case.

4.3.2 Motorola DSP56303 evaluation Board

I chose to use a Motorola DSP56303 evaluation board [30] as it is a board that was designed for real time audio signal processing applications and has the following valuable features:

- An on-board stereo 16 bit A-D/D-A converter capable of 48K samples per second on each channel simultaneously.
- It uses the Motorola 24 bit DSP56303 chip [55].
- Most of the DSP chip interfaces are brought out onto connectors, allowing other units to be connected to the evaluation board.
- It has 32K words of high speed static RAM, ideal for storing FIDs.
- It comes with a whole host of development software.
- It is very reasonably priced.

The DSP56303

The DSP56303 is one of a new series of 24 bit DSP chips from Motorola that can execute most instructions in a single clock cycle. As I intended to build my own DSP board, the chip packaging is a very important consideration. Most high end devices are now only available in Ball Grid Array (BGA) packages, which are extremely difficult to work with as they require special techniques and equipment to solder them to a circuit board. However the 56303 is available in a 100 MHz, 144 pin quad flatpack version and this makes it one of the fastest DSPs available in a standard surface mount package. It can be soldered to a board without any special equipment apart from some soldering paste and a reflow soldering tip. I also wanted a DSP chip that had a data width greater than 16 bits, but I avoided floating point processors as they are not as efficient at performing fixed point operations as a fixed point processor, and often have much poorer response times to interrupts, which can make high speed real time applications more difficult. Floating point processors are also physically large devices with many pins and consume much more power than a fixed point device. A 24 bit processor was ideal as it had the dynamic range needed for signal averaging, while for digital filtering, 24 bit coefficients could be used, resulting in better filter performance. The 56303 is similar to the older 56000 series devices which I have some experience with, so this was another consideration when choosing this device. Another point in its favour is the availability of good, free development software such as a C compiler, an assembler and a debugger, as well as lots of application notes and software examples of signal processing algorithms. Overall the DSP56303 is a good performer and is well suited to NMR signal processing and control applications, and it is also well supported by a large international company and many semiconductor retailers.

The DSP56303 consists of a DSP56300 core and a peripheral and memory expansion area as shown in figure 4.3. The 56303 is one of the 56300 family and so shares a common core but has different peripherals and memory from other devices. The memory is divided up into three parts, program, X and Y data memories. This is common with DSP devices as many operations can be done in parallel, such as an instruction fetch and an X and Y data move. It is this level of parallelism that makes DSP devices so fast. The separate data memories are very useful when implementing digital filters as one memory may contain the filter coefficients and the other the data. Part of the program memory can be used as an instruction cache and it is also possible to make the X and Y data memories larger by allocating some of the program memory

to them. Off chip memory expansion is also possible through the use of the external data, address and control buses. The peripheral expansion area consists of four units. The first is the triple timer module which consists of three independent 24 bit timer/event counters each with an associated I/O pin. The timers can perform a number of functions such as counting, pulse width measurements and pulse width modulation. The next module is the host interface which consists of an 8 bit I/O port and a number of control lines. The host port has been designed to allow a glueless connection to a number of industry standard microcomputers, microprocessors and DSPs that may operate as the master (or host) processor. In this configuration the DSP behaves as a memory addressed peripheral to another processor. The host port can also be configured as 16 general purpose I/O lines and it is this mode that I have used to interface the DSP to the PIO24 card. The next module is the dual enhanced synchronous serial interface which is often used to communicate to high speed serial devices such as A-D and D-A converters. The last module is a serial communications interface that is asynchronous and can be used as an RS232 port. The external address, control and data buses are common to all the devices within the 56300 family, as well as the interrupt pins, the Joint Action Test Group (JTAG) port, and the On Chip Emulation module (OnCE).

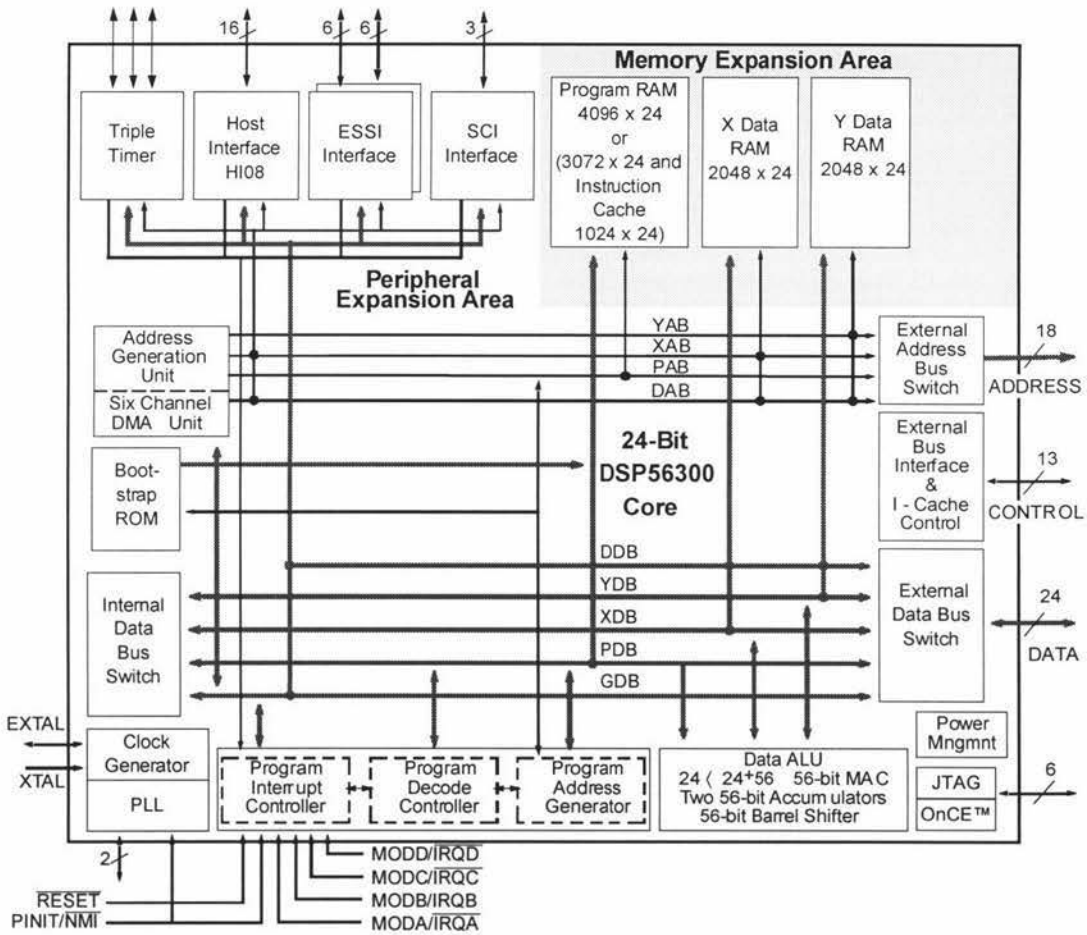


Fig 4.3 The DSP56303 block diagram [51].

The JTAG port can be used to verify the device's hardware and is a feature that is mainly used during manufacturing. The OnCE port can be used for debugging as it

allows one to examine and change registers, memory, and on chip peripherals and as such is a very useful feature.

The DSP56303 programming model (figure 4.4) is the same as all the other devices in the 56300 family and therefore all the software is code compatible. This means that all the software that I write for 56303 can be easily ported to planned future higher performance devices.

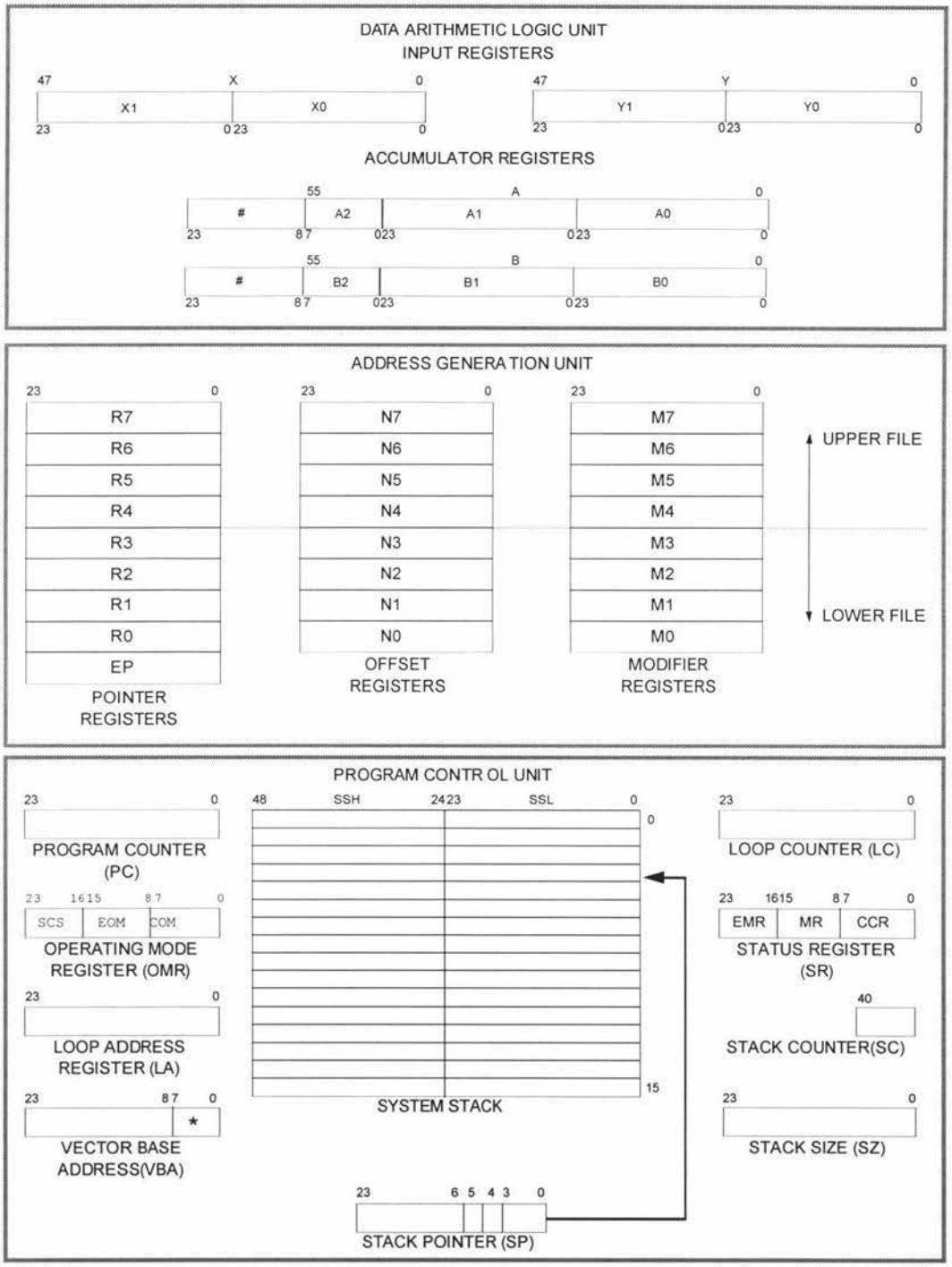


Fig 4.4 The 56300 series programming model [50].

The programming model is divided into three sections, each associated with a different device unit. The Data Arithmetic Logic Unit performs all the arithmetic and logical operations on the data and is made up of a number of 24 bit registers and two 56 bit accumulators. One of the most popular operations that has been optimised for the DSP is the multiply-accumulate (MAC). This is where the contents of two 24 bit input registers are multiplied and then added to the contents of one of the accumulators. During the multiplication phase two new values can be read from the X and Y data memories and loaded into the input registers, ready for the next multiply. If the MAC operation is performed repeatedly (in a loop) it is then possible to perform one multiply accumulate per clock cycle. This technique has become the core of digital filtering where one needs to quickly multiply and accumulate an array of filter coefficients with an array of signal data. The next section is the address generation unit, which consists of a number of 24 bit registers that are used as memory pointers. Offset and modifier registers are also included and are used when implementing circular buffers and/or modulo "n" indexing. Circular buffers are very useful for digital filtering, as a FIFO buffer is needed for the signal data. The last section is the program control unit, which is very similar to that of a standard microprocessor with its program counter, stack pointer and status register. The loop counter and loop address registers are used to implement zero overhead hardware "DO n" looping. Again this is another very useful feature for digital filtering.

The evaluation board

The DSP56303 evaluation board as shown in figures 4.5 and 4.6 is as an evaluation system produced by Motorola and is designed to help developers to become familiar with the DSP device hardware and software and thus kick start the development of custom DSP boards.

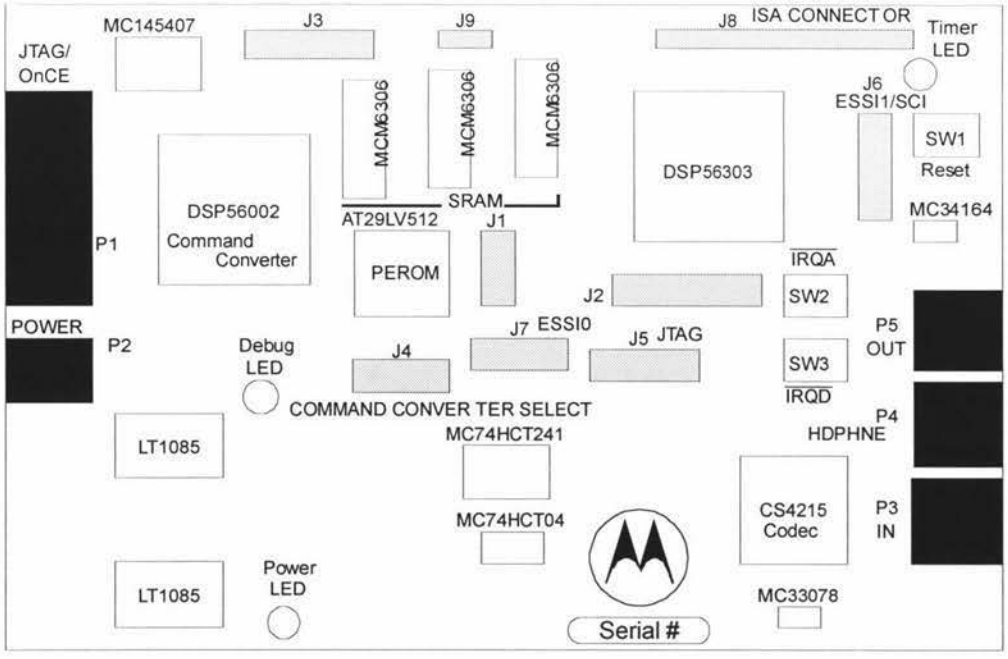


Fig 4.5 Motorola DSP56303 evaluation board layout [30].

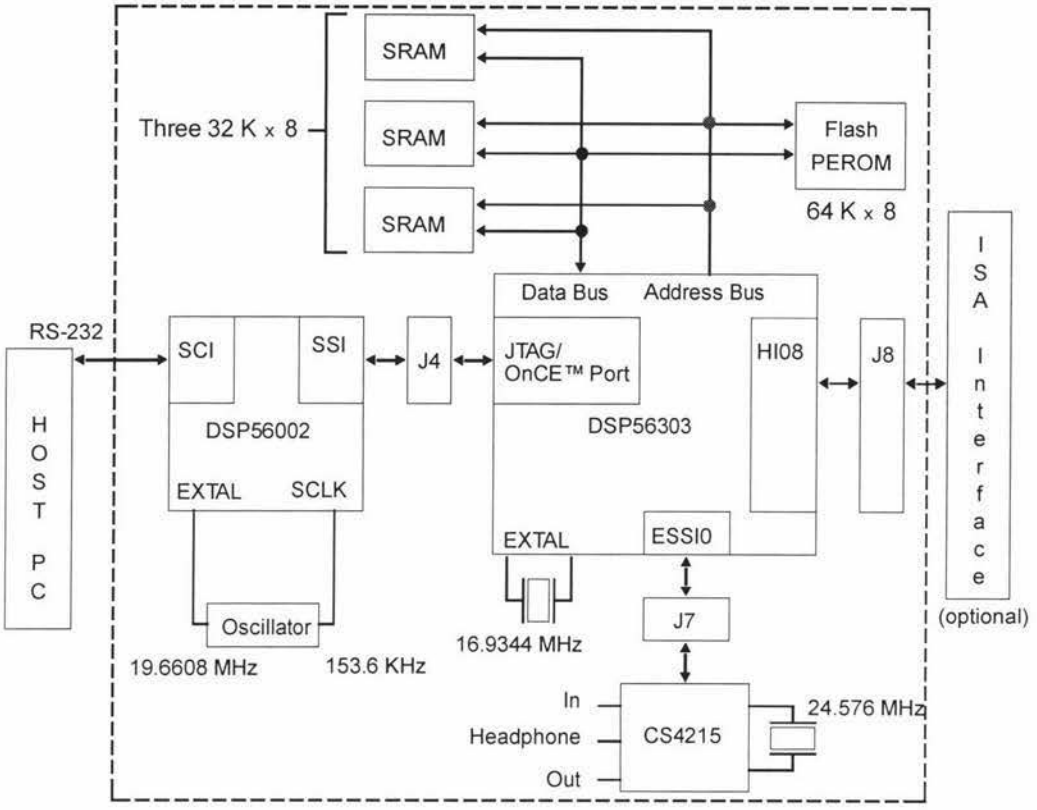


Fig 4.6 Motorola DSP56303 evaluation board simplified schematic [30].

The evaluation board consists of a DSP56303 chip and some other peripherals such as some high speed SRAM, a bootstrap EEPROM, a JTAG/OnCE command converter and a stereo audio A-D/D-A converter chip. The host port, the reset and three interrupt input pins are brought out onto a two row 2 mm pitch 40 way male header connector (J8). This connector is used to couple to the DSP/PIO24 interface board. The JTAG/OnCE command converter is an older DSP device that is used to interpret commands sent from a host PC via an RS232 link, and then control the 56303 chip via the JTAG/OnCE port. This is used extensively when debugging, as the debugger allows one to down-load programs into the 56303's memory and then execute them using breakpoints and single stepping. The 56303's registers and memory can also be monitored and modified using the debugger. The EEPROM is a "bootstrap" ROM that contains program code that can be loaded into the 56303's memory and then executed following a reset. The CS4215 stereo A-D/D-A converter is described by its manufacturer [27] as a 16 bit multimedia audio codec that was developed for high quality music processing. The sampling rate is programmable and ranges from 8 to 48 kHz. Each of the two A-D channels consist of a programmable gain stage followed by a 64 × oversampled sigma-delta converter. The D-A channels each consist of a delta-sigma converter followed by a programmable attenuator. The CS4215 communicates with the 56303 via a synchronous serial interface. The default memory map of the 56303 evaluation board is shown in figure 4.7, where the external 32 K words of static RAM appears as a unified block of X and Y data memory between addresses $10000 and $18000. This however can be changed by programming one of the address attribute registers (AAR0) of the 56303.The amount of internal program and data memory can also be changed by suitably programming the cache enable and memory switch bits of the 56303's control registers.

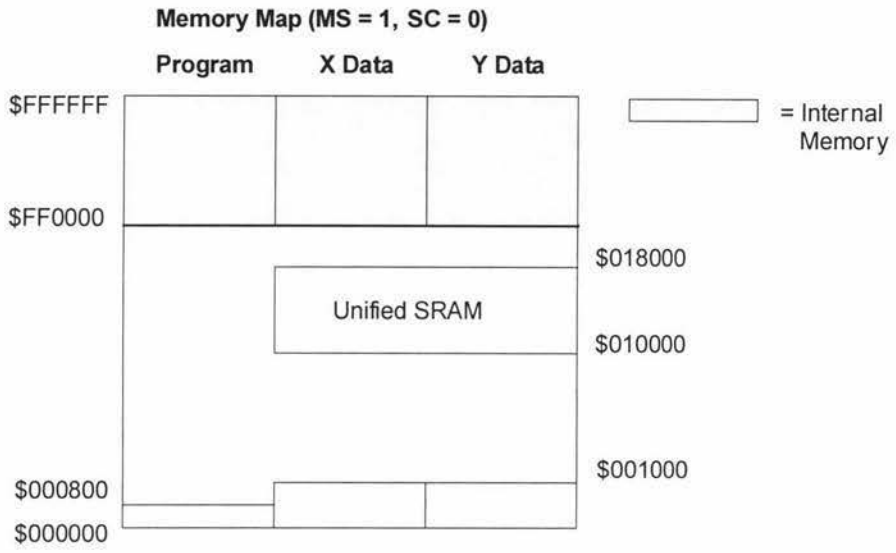


Fig 4.7 Motorola DSP56303 evaluation board memory map [30].

The external static RAM is very useful for storing acquired FIDs, as there is not enough memory within the DSP to do this.

4.3.3 The PIO24 PCMCIA digital interface card

The PIO24 card emulates the common Intel 8255 [46] Programmable Peripheral Interface but with reduced functionality. It only operates in basic mode, in that each bit can only be used as standard I/O. The PIO24 has three 8 bit ports, A, B and C, and each individual bit of each port can be configured as either an input or an output. The PIO24 hardware allows its bits to be reconfigured at any time, but unfortunately with Windows98 and future operating systems this is not possible as the card must be globally configured as multiple applications, or threads may use it simultaneously. With "DOS" it was quite common to use one 8 bit port as an I/O port, with the port directions being set accordingly for a read or write operation, essentially emulating the data bus of a microprocessor. This arrangement is no longer possible. The PIO24 card also has a clock input pin that is used to synchronise the reading or writing of data with an externally generated clock pulse. The card fits into any standard PCMCIA slot provided on most laptop computers, and it comes with a cable that brings out all the connections to a D37 female connector as shown in figure 4.8.

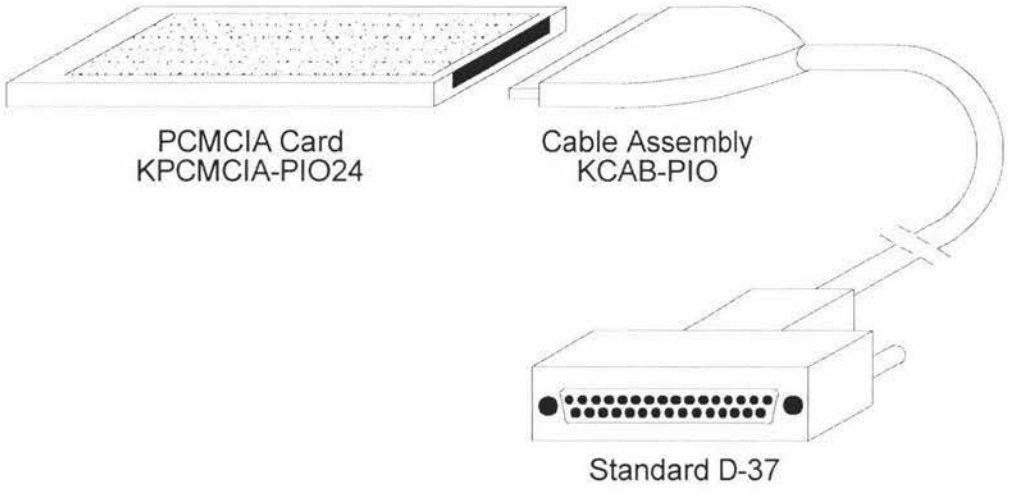


Fig 4.8 I/O card and cable assembly [46].

The digital I/O channels each consist of an open collector driver and a 10 kΩ pull up resistor connected to 5 V (figure 4.9). When the channel is configured as an input, the open collector driver is disabled. Upon reset all the channels are set as inputs.

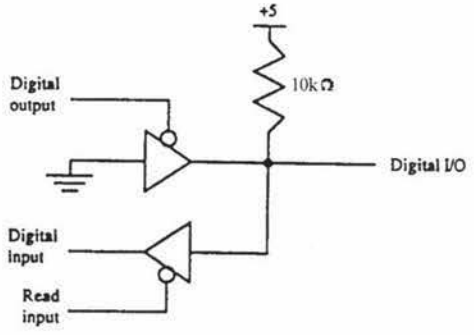


Fig 4.9 PIO24 interface arrangement

4.3.4 DriverLINX

Device drivers must be used to access the PIO24 card, as Windows98 doesn't allow one to access the card's hardware directly. This results in extra overheads and constraints and therefore reduced performance. DriverLINX [47] is the device driver provided with the card and is a language and hardware independent applications programmers interface (API) that has been designed to support a range of different I/O devices, such as parallel I/O cards and data acquisition boards. DriverLINX is a multiuser, multitasking data acquisition resource manager that provides many services for foreground and background data acquisition tasks. Communication between the users application and the device driver is through the use of a service request data structure. The data structure is used to define the required operation and also contains pointers to data buffers that are needed for data transfer operations. The structure once configured is passed to the DriverLINX driver for execution, and when the driver has finished it passes the structure back with some additional status information such as success or failure and/or the number of bytes transferred. Special functions are also provided with the card to handle the data structure and some other operations. Due to the fact that many applications can access the card a single application cannot configure the card. This must be done through a Windows98 control panel.

Logical devices

A single DriverLINX driver can support multiple boards, so during configuration each physical device is assigned a logical device number that DriverLINX then uses to access the device.

Logical subsystems

DriverLINX maps a hardware independent, or abstract, data acquisition model onto the PIO24 hardware capabilities. The PIO24 card is treated by DriverLINX as four logical subsystems:

1. Device, which is the PIO24 card as a whole.
2. Digital input, this refers to the 8 bit digital I/O ports as well as the clock input line.
3. Digital output, again refers to the 8 bit digital I/O ports.
4. Counter/Timer, refers to the clock channel for pacing input/output tasks.

Logical Channels

Each part of the PIO24 is assigned a logical channel number that is used by DriverLINX.

Logical channel	DriverLINX Function	KPCMCIA-PIO24 connector
0	Digital I/O port	PA0 – PA7 (D0 – D7)
1	Digital I/O port	PB0 – PB7 (D8 – D15)
2	Digital I/O port	PC0 – PC7 (D16 – D23)
3	Clock input	Clock input

Operating modes

DriverLINX also operates in a number of different modes, which are of two types.

1. Foreground or synchronous. Here the calling application doesn't regain control until DriverLINX completes the service request. This type of mode is used for simple single value I/O operations and some housekeeping functions.
2. Background or asynchronous. Here the calling application regains control as soon as DriverLINX initiates the task. This type of mode is used for burst I/O. Polling, which is a foreground task, can be used to determine when the task has completed.

The three modes that can be used with the PIO24 card are listed as follows:

1. Polled mode. This is a foreground operation where only a single I/O transfer is performed.
2. Interrupt Mode. This is a background operation involving multiple I/O transfers under the control of an internal or external clock.
3. Other mode. This is a foreground mode and is used for initialization and configuration operations.

The Service Request data structure

The Service Request is a C data structure that is initially set up by the user's application before its address is passed to DriverLINX for execution. The structure itself consists of four major property groups.

1. Request group. This specifies the target Logical Device and Logical Subsystem, the data acquisition mode and the operation to perform.
2. Events group. This specifies how DriverLINX should time or pace the data acquisition.
3. Select group. This specifies the Logical Channels to use and the number and length of the data buffers.
4. Results group. DriverLINX uses this group to return result codes and single data values.

Types of data transfers

Single I/O (Polled mode)

Here only a single value is read or written to a port using a single I/O request structure for each transfer. Multiple data transfers can be done by software loops. I tried this and found that I could only get a maximum transfer rate of about 600 bytes per second and this demonstrates the immense overhead of the device driver.

Burst I/O with internal clocking (Interrupt mode)

Here multiple values can be transferred with a single I/O request structure and an internal clock is used to synchronise the transfers. Again this was tested and surprisingly was found to be even slower than the previous technique. A maximum transfer rate of only 200 bytes per second was obtained.

Burst I/O with external clocking (Interrupt mode)

Here multiple values can be transferred with a single I/O request, but this time an external clock is used to control the transfer of each value. Using this technique it is possible for short periods of time to get transfer rates of up to 50 K bytes per second. For large data transfers of about 10 K bytes, a transfer rate of about 32 K bytes per second was found to work reliably. Sometimes the computer will go off and do some other task such as access the disk drive and then it is not possible to sustain a 50 K byte per second data transfer rate. For the Earth's field NMR system, 32 K bytes per second transfer rates are acceptable as the experiments run over a long period of time, unlike some other systems were an experiment may be repeated every 100 ms with thousands of data points.

I decided to use single I/O transfers to send commands to the DSP board and then use burst I/O with external clocking to transfer data to and from the DSP board. The DSP board had the task of generating the necessary clock pulses to control the transfer of the data.

To get the DriverLINX component working required wading through hundreds and hundreds of pages of manuals and many hours trying to work out exactly how it all functioned and what the limitations and pitfalls were. This is what happens when you are removed from the hardware and have to work with a complex software interface that most likely has bugs and is poorly documented, as expected for most software manuals. Using the card took up a lot more time than was initially intended for a temporary solution and this was mainly due to having to work with the bureaucracy imposed upon by modern super protective operating systems. One wonders how much time has been wasted on this kind of work when under DOS a couple of lines of code would have done the same job !

4.3.5 DSP board/ PCMCIA card, interface board

The Laptop/DSP relationship has been organised as a Master/Slave arrangement. The Laptop has complete control over what the DSP board is doing, and this has been implemented using a command/response technique where the laptop issues a command and then waits for the DSP to execute the command. The DSP board spends most of its time waiting for a command from the laptop. The DSP board can't initiate anything, not even a response due to an error, therefore the laptop being the master has the responsibility to poll the board periodically to determine its status. If things go wrong, the laptop can force a reset in the DSP board and therefore bring it to a known state. This master/slave arrangement is simple and works very well. Sometimes it is very difficult and cumbersome to implement communication between devices that do not have a simply defined control structure.

Some of the commands that the laptop can initiate are as follows:

- Download, here a block of data or pulse program is received from the laptop and stored into the DSP's program memory.
- Run Pulse Program, here the previously loaded pulse program is executed.
- Upload data, here a 2048 byte block of data is sent to the laptop.

Communication between the laptop and the DSP board is a combination of a software protocol and a hardware interface. A simplified diagram of the interface is shown in figure 4.10. Note that the DSP host interface has been used, allowing the other high speed interfaces to be available for other devices, such as high speed A-D converters.

The way the laptop sends a command to the DSP is as follows:
A command consisting of 4 bits is placed on the outputs D16-D19 of the PIO24 card. The D21 output is driven high and then low generating a pulse edge that is used to interrupt the DSP. The DSP then reads the command and places it into a buffer and then executes it once it has finished completing its present task. The laptop being the host most often knows what the DSP is doing and therefore doesn't need a response from the DSP. There is one exception and that is when the DSP is running a pulse program. In this case after the laptop issues a run pulse program command, it waits for the DSP by monitoring the pulse program finished signal that is connected to an input pin (D23) of the PIO24 card. Polling, command generation and the interrupt signalling are all done by issuing single I/O requests to DriverLINX.

Burst I/O with external clocking is used when down-loading pulse programs and uploading data to and from the DSP board. For down-loading pulse programs the laptop first issues a download pulse program command to the DSP board using single I/O DriverLINX requests. The laptop then issues a "burst I/O write with external clocking" request to DriverLINX. Within the burst I/O request structure is a pointer to a buffer containing the pulse program data, and a parameter containing the number of bytes to transfer. The first 3 bytes (24 bits) of the pulse program data specify where the program should be loaded into the DSP memory and the next 3 bytes specify the number of bytes to transfer. The DSP, after receiving the command, sets up the port lines PB0-PB7 as inputs and also enables the PIO24 portA buffer. It then waits for 10 ms for the laptop to get ready and then reads the first 6 bytes indicating the load address and transfer size. Immediately after this the DSP reads the rest of the data and

loads it into its memory. The DSP reads the data from the laptop by sending clock pulses to the clock input of the PIO24 card to initiate the transfer of each byte.

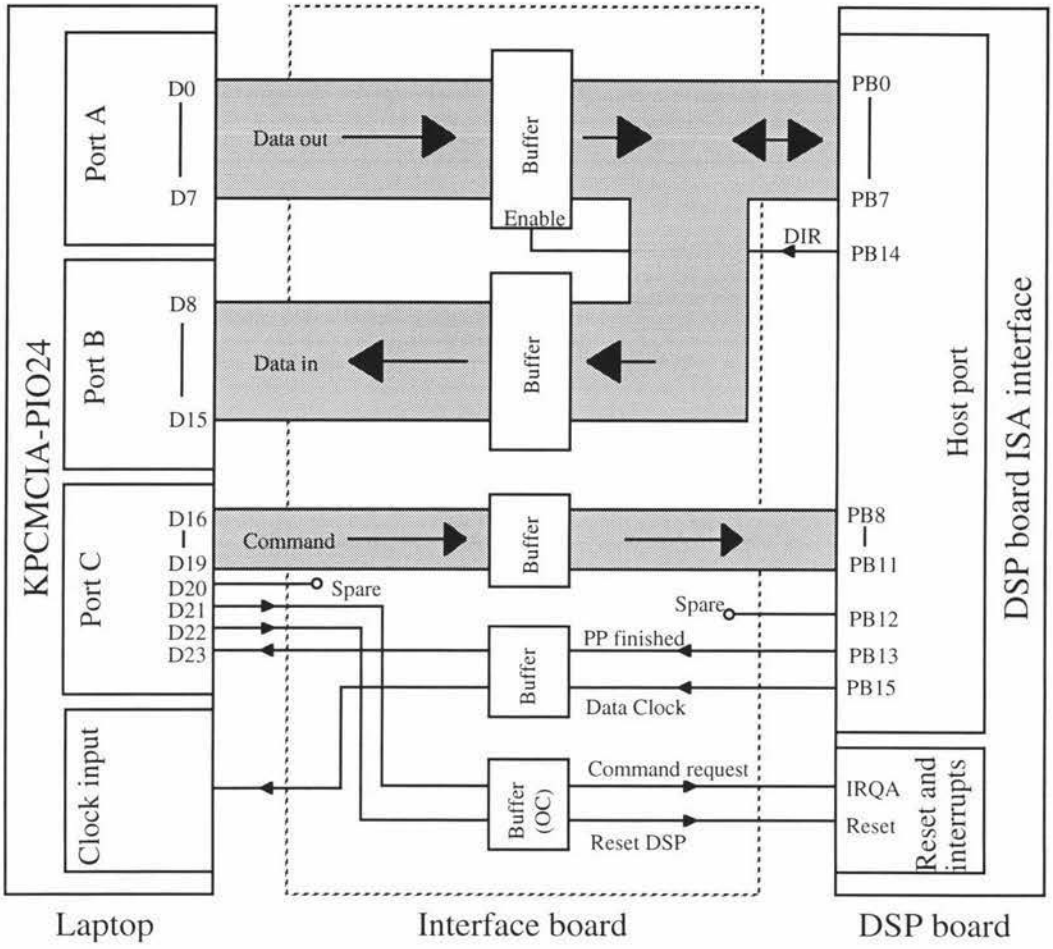


Fig 4.10 Simplified diagram of the PIO24/DSP board interface.

A timing diagram of a PIO24 write is shown in figure 4.11. Here the DSP first outputs a clock pulse. DriverLINX then outputs the data on the PIO24 portA pins and this should occur within 20 µs of the clock pulse. The DSP waits for 25 µs from the time it output the clock pulse and then reads the data. If further bytes need to be transferred, the DSP outputs another clock pulse.

For uploading data the laptop first issues an upload data command to the DSP board using single I/O DriverLINX requests. The laptop then issues a "burst I/O read with external clocking" request to DriverLINX. Within the burst I/O request structure is a pointer to a buffer where the data will go. After receiving the command the DSP sets up the port lines PB0-PB7 as outputs and then waits for 10 ms for the laptop to get ready. The DSP writes a 2048-byte block of data to the laptop and simultaneously outputs clock pulses to the clock input of the PIO24 card to initiate the transfer of each byte.

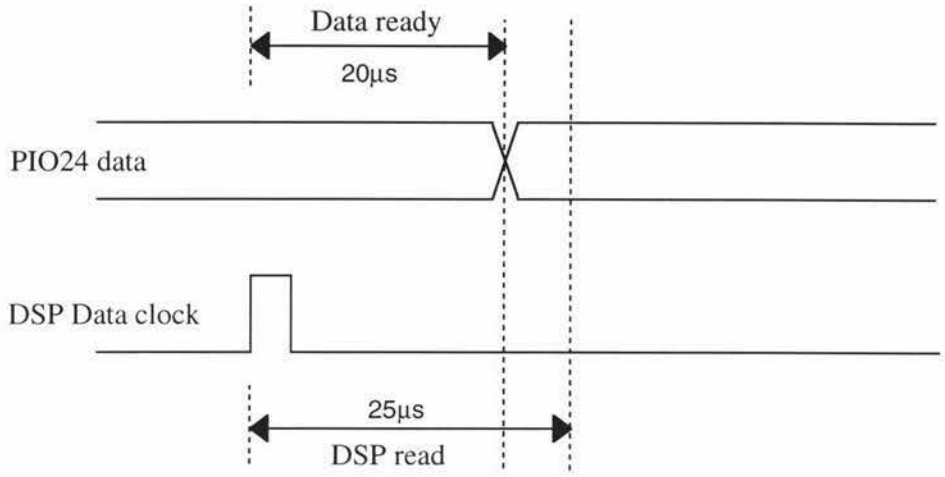


Fig 4.11 Burst I/O write timing.

A timing diagram of a PIO24 read is shown in figure 4.12. Here the DSP outputs the data and generates a clock pulse. DriverLINX then reads the data on the PIO24 portB pins and this should occur within 20 µs of the clock pulse. If further bytes need to be transferred, the DSP outputs another byte and a clock pulse.

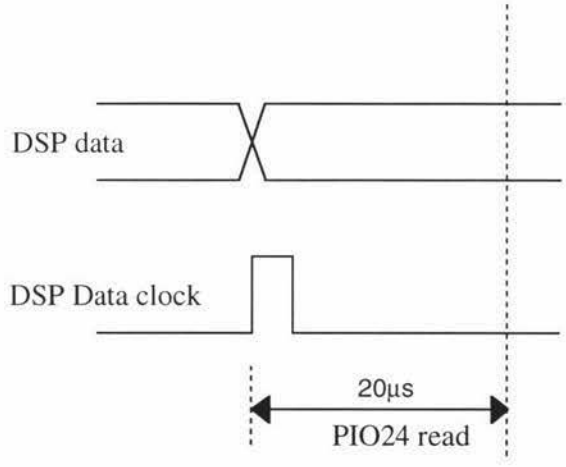


Fig 4.12 Burst I/O read timing.

Using the burst I/O with external clocking technique is the fastest way of transferring data between the laptop and the DSP, and data transfer rates of about 30 K bytes per second can be obtained with reasonable reliability. Unfortunately sometimes transmission errors do occur, so this must be taken into account and compensated for within the software. The main source of transmission errors is DriverLINX not always being able and ready to transmit bursts of data. One way to overcome this is to slow everything down. This is not very efficient and when tried the errors reduced, but some still did occur, usually when the laptop accessed the hard drive. In Antarctica the problem was exacerbated due to problems with the connectors in the extreme cold. The software functions that I provided for the user applications would try to detect errors and if found would retry the transmission. The number of retries was limited to

5 as any number of successive errors greater than that would indicate that something was severely wrong with the communication link. This could occur if the laptop and DSP get out of synchronisation with each other. This can be remedied by the laptop resetting the DSP board.

A detailed schematic of the PIO24/DSP board interface is shown in figure 4.13. The DB37 connector (J1) connects to the PIO24 card via an interconnecting cable provided with the card and the connection to the DSP board is through the 40 way header connector. The DSP board uses a 3.3V supply for the DSP56303 device, therefore all the signals that go from the DSP to the PIO24 board need to be buffered and converted to 5V logic, and this is done with U3 and parts of U2. The buffer U4 is used to isolate PIO24 port A from the DSP host port when the lower 8 bits of the host port are configured as outputs. This is done through the use of the buffer enable pin, which is connected to PB14 of the host port. The DSP host port, reset and interrupt inputs are all 5V tolerant, therefore no level translation is required. However the reset and interrupt inputs are connected to other circuitry on the DSP board in a "wired OR" configuration, therefore open collector output gates (U1) are required if a connection is made to these pins. U1A buffers the IRQA signal and likewise U1B the RESET. The two resistors (3K3Ω) R2 and R3 are needed to ensure that when the PIO24 card is disconnected, the U1 buffer doesn't pull down the DSP IRQA or RESET inputs. The pin PB15 of the DSP host port, which is pin 13 of the header connector, is used to provide the clock required for data transfers. It is buffered and then connected to pin 1 of the DB37 connector (J1). The "pulse program finished" signal was a later addition and used up one of the two spare host port pins (PB12 and PB13) that I brought out onto J6 and J2. J5 is a buffered version of PB13 and is connected via a wire link to J8, which is pin D23 of the PIO24.

Most of the lines associated with the DSP peripheral ports can be used for general purpose I/O. The lines of the SCI and one of the ESSI ports of the DSP are used as high speed digital output lines for controlling the NMR system. There are 9 lines altogether and they are brought out onto a 12 pin header connector (see figure 4.14). A ribbon cable is used to bring the lines from the evaluation board (J6) onto the interface board for buffering. An octal buffer is used so one of the lines (E1) remains unused and the others are brought out to a male DB9 connector.

All the logic devices used in the interface board are CMOS 74HCT series as they are compatible with the logic levels generated by devices powered from 3.3V. A two layer tin plated circuit board is used and the layouts are shown in figure 4.15.

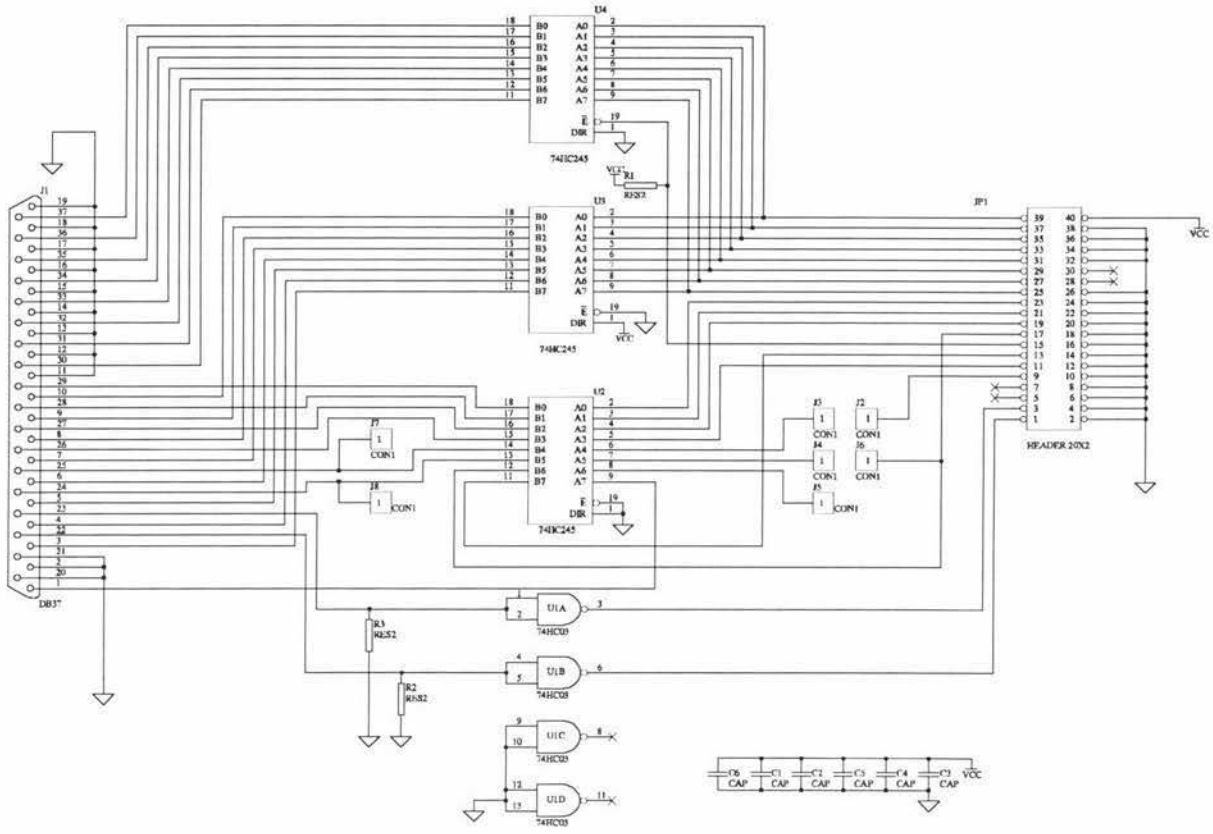


Fig 4.13 PIO24/DSP board interface schematic.

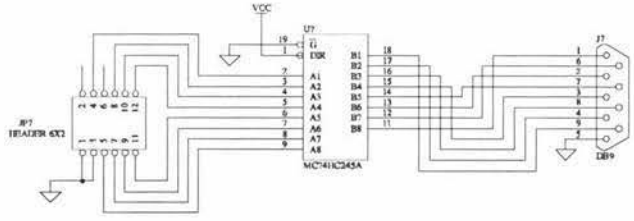
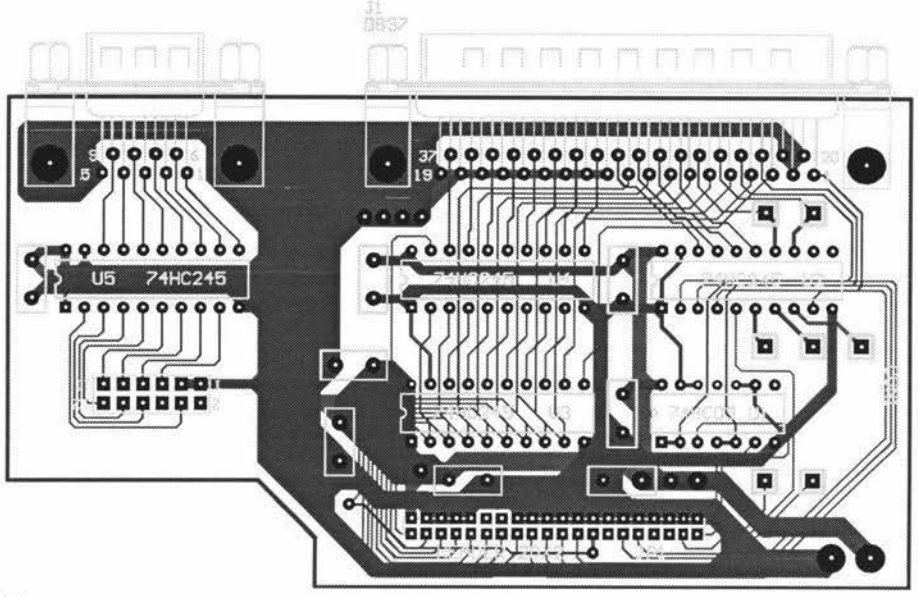
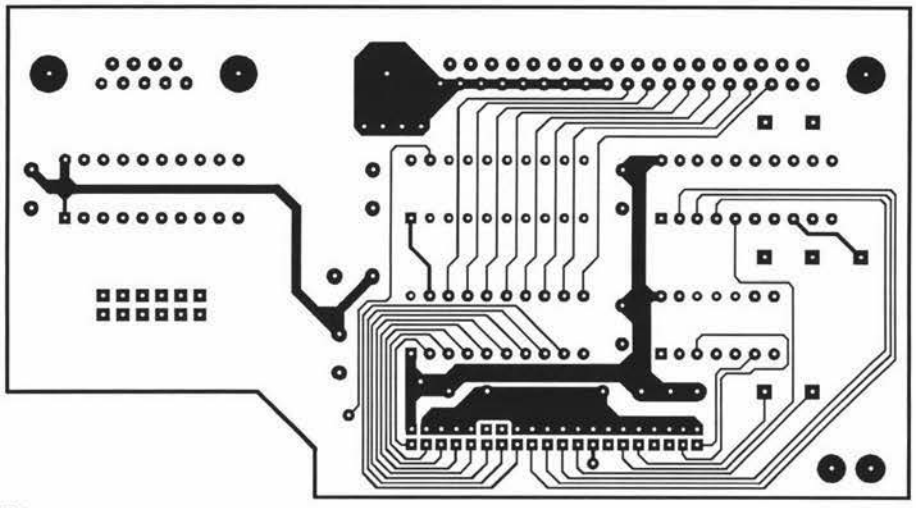


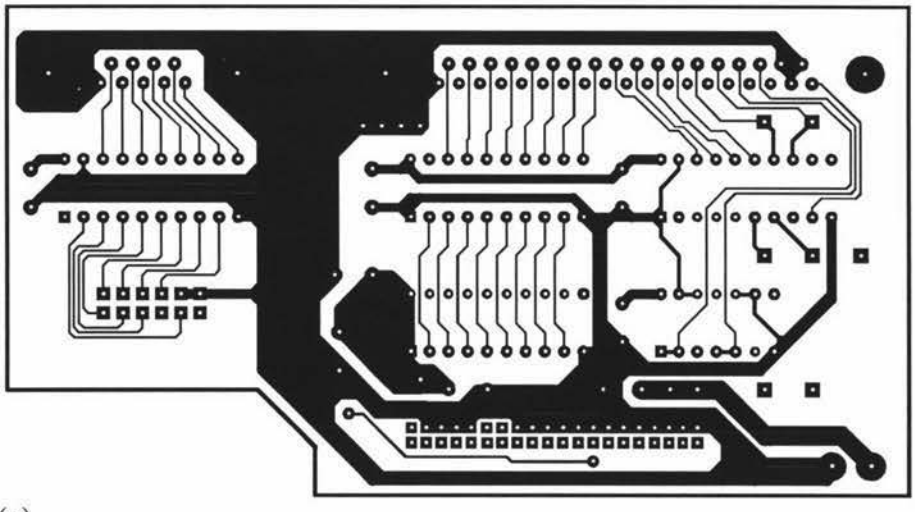
Fig 4.14 DSP board high speed digital outputs schematic.



(a)



(b)



(c)

Fig 4.15 PIO24/DSP-board interface, circuit board layout, overlay (a), top layer (b), bottom layer (c).

Figure 4.16 shows the arrangement of the two boards within the "digital box". The ribbon cable in the top left hand corner is used to bring the RS232 port (JTAG/OnCE) out onto a DB9 female connector mounted on the outside of the case. The interface board is directly connected to the DSP board via a 40 pin 2 mm pitch female connector that is mounted on the underside of the interface board. The ribbon cable pictured in the centre carries the high speed digital output lines, and the twisted wires on the right hand side connect one channel of the A-D/D-A to BNC connectors. The digital box can be used separately from the rest of the NMR system, by simply using a DC plug-pack instead of the power provided by the "analogue box". The power (8V DC) is supplied to the DSP board via a plug pack connector mounted on the outside of the case and the wire running along the bottom. The interface board derives its power from the DSP board.

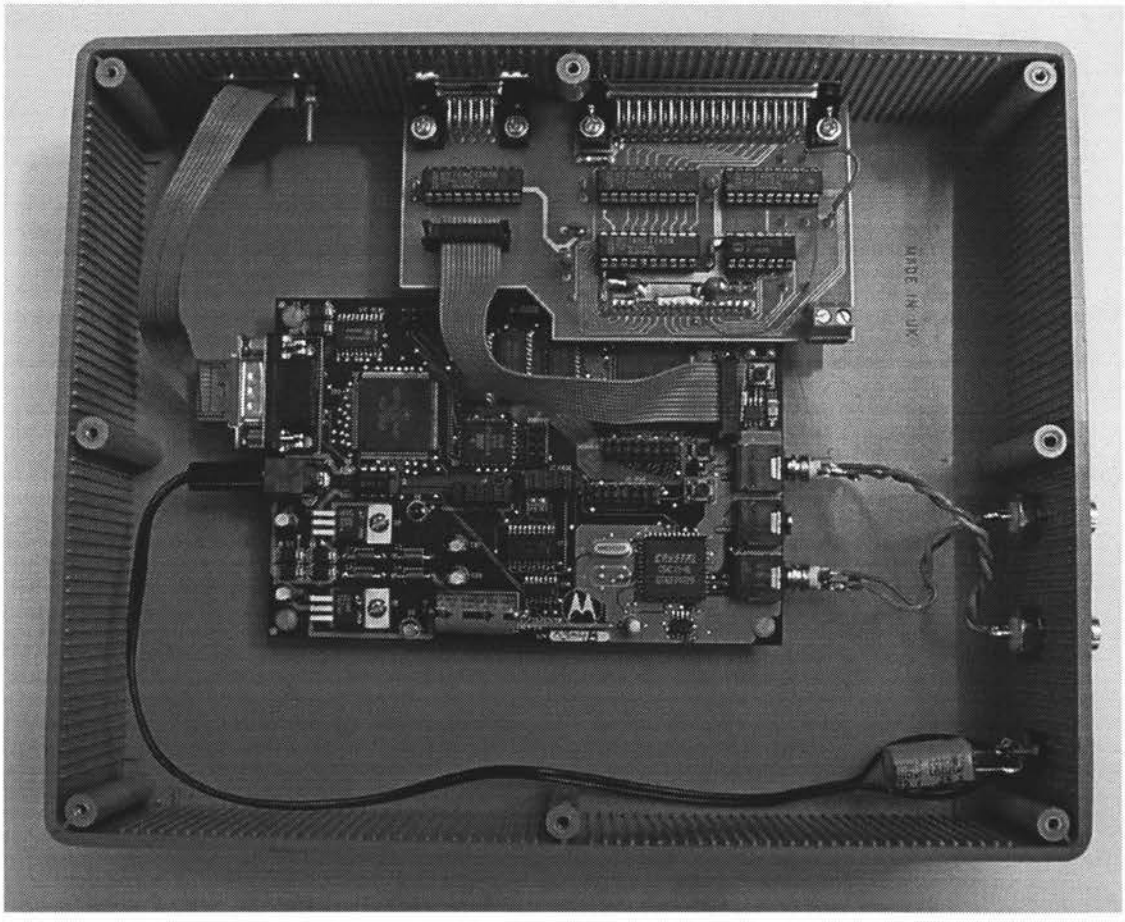


Fig 4.16 "Digital box" consisting of the DSP and interface boards.

4.3.6 The DSP and Laptop software

As mentioned previously the laptop operates as a master and the DSP board as a slave. This is done by the laptop issuing commands, which the DSP board then executes. The laptop and the DSP board each operate independently using their own software. The software also manages the inter-device communication so that as far as an application programmer is concerned the hardware doesn't exist, as the communication appears to be between two software systems. This is shown in figure 4.17 where a virtual link is formed between the upper layers of the communications protocol stack. The application doesn't know about the hardware and "thinks" that it is communicating directly with the DSP board's operating system. In reality, when the application calls one of the functions provided by "MyAPI", commands are sent down through the layers and across to the DSP board, and then the DSP board's operating system takes the appropriate action. When the DSP board's operating system receives a command, it takes action by calling the specified command handler, which then uses I/O and/or other routines to perform the task.

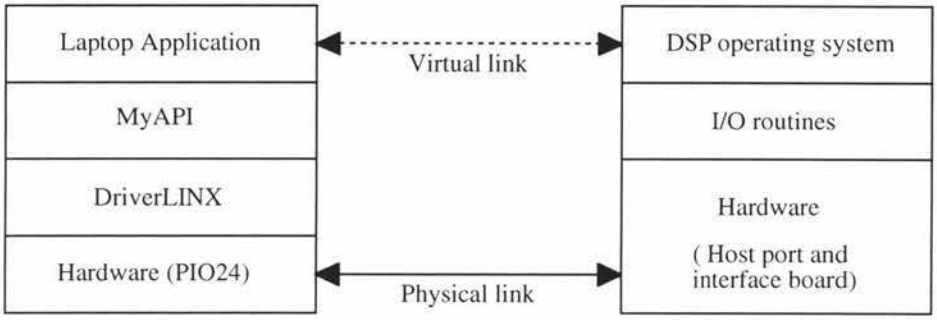


Fig 4.17 Communications protocol stack.

Each layer within the protocol stack only communicates with neighbouring layers. Using a protocol stack model is helpful when developing software for two systems that need to communicate, as it clearly defines what each part of the system has to do. Being modular makes it a lot easier when modifications are required. For example, if the communication hardware is replaced with alternative hardware, only the bottom parts of the protocol stack such as the DSP I/O routines and the lower parts of the MyAPI layer, would need to be rewritten. The main part of the application running on the laptop would not require any modifications.

The MyAPI layer that I developed provides a simple software interface to the users' application, therefore eliminating the need for the application programmer to have any knowledge about DriverLINX and the interface hardware. The functions provided by MyAPI that are available to the application are as follows:

- InitDSP. This function initialises DriverLINX and resets the PIO24 card.
- DownloadPP. This is the download pulse program function and is used to load a pulse program into the DSP that is contained within a file on the laptop.
- DownloadData. This is another download function, but this time is used to transfer the contents of a data buffer within the application to the DSP board.

- Upload. This function uploads a block of data from the DSP and stores it into a buffer within the application. The actual transfer is broken up into a series of 2048 byte transfers but this is transparent to the application program.
- Run. This function signals the DSP to execute the pulse program that has been loaded into it.
- Set. This is used to set a single parameter within the DSP, such as the acquisition sample rate.

A typical NMR experiment would initially consist of loading a pulse program into the DSP, and then the setting of some parameters. Execution of the pulse program would be the next phase, followed by the uploading of the FID data. Once loaded the pulse program can be repeatedly executed and the data uploaded after each scan. The user can also alter some parameters between each scan and so does not have to download a new pulse program every time.

When the application running on the laptop wants to communicate with the DSP board's operating system, it calls one of the functions provided by MyAPI. This function, depending on the task, may then send a whole series of commands to the DSP board. For example, with the upload function, the following occurs: First a command is sent to the DSP board, requesting it to set its output data buffer pointer to the beginning of the data buffer. The requested data transfer is then broken up into a series of 2048 byte transfers. An upload data command is then sent to the DSP board for each 2048 byte block that needs to be transferred. If transmission errors occur, the same block is transferred again, but if 5 successive block transmission errors occur the complete transmission request is aborted and a message is passed back to the application indicating that there was an unrecoverable error. An increment block pointer command is sent between each successful transfer.

The functions provided by MyAPI can be thought of as "high level" commands that are then broken down into a series of "low level" commands that are subsequently sent to the DSP board. Using this technique makes a lot of the I/O transparent to the application and therefore makes life a lot easier for the applications programmer.

4.3.6.1 The DSP software

The software running on the DSP board consists of two parts, the operating system and the pulse program that gets loaded and executed. The pulse program itself can be thought of as an application, as it is loaded into memory, and when the program has finished it terminates and hands back the control to the operating system. The operating system running on the DSP board is responsible for initialising and maintaining the hardware and the memory, and for receiving commands and dispatching control to the command handlers. The DSP operating system software was written in assembly language to minimise the memory requirements and to achieve good speed performance, and a listing of it (mprog1.asm) can be found in appendix A. Figure 4.18 shows a simplified structure diagram for the operating system software, the I/O and other low-level modules are not shown.

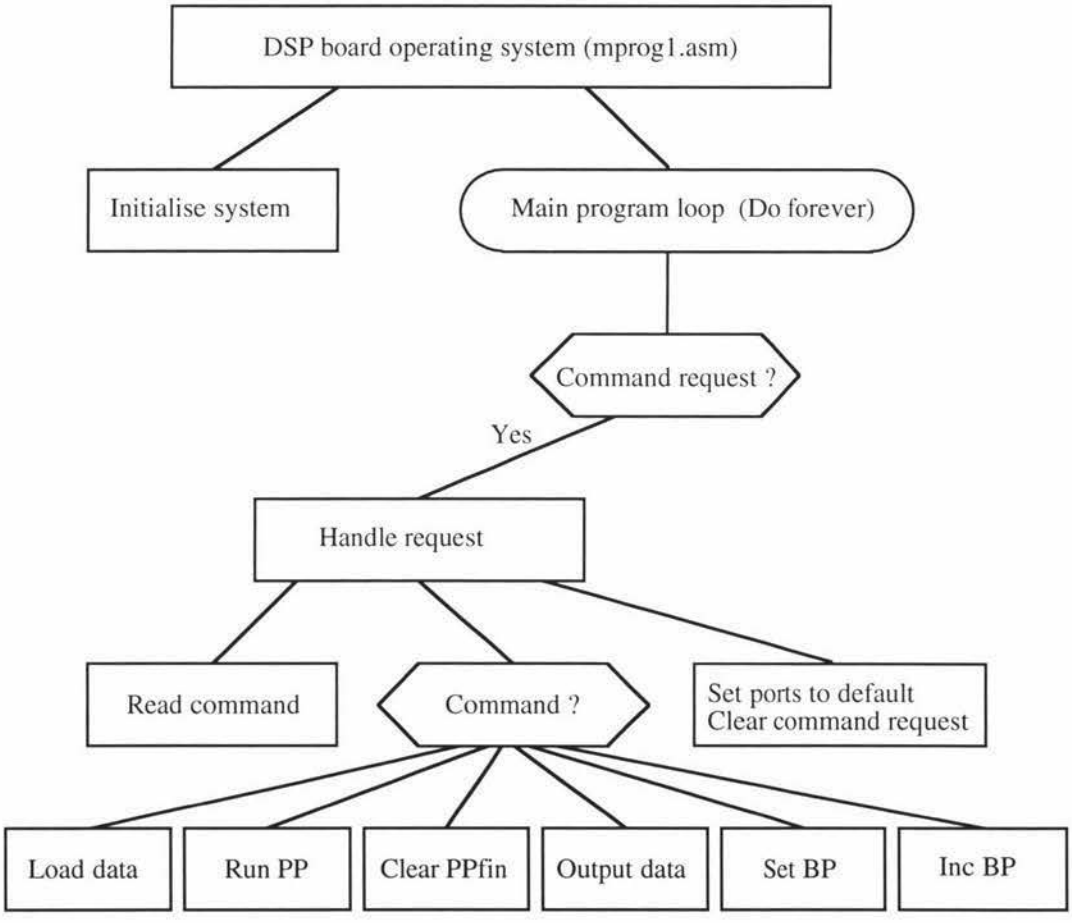


Fig 4.18 Structure diagram of DSP board operating system.

After a reset, the system is firstly initialised before entering an infinite command processing loop. The initialisation phase is concerned with setting various DSP hardware and software parameters such as: the DSP operating frequency (80 MHz), the external static RAM address and wait states, the system and user stack pointers, the modulus of the index registers, the peripheral I/O ports, various flags and buffers, the interrupts and finally the initialisation of the A-D/D-A converter unit.

When a command request interrupt occurs, the interrupt routine sets the command request flag, and for debugging purposes it also turns on an LED. On detecting the set flag, the main program loop reads the command value from the Host port and then calls as a subroutine, the specified command handler. When the command is completed, control is passed back to the main program which then turns off the LED, clears the command request flag, does some other house keeping and then goes back to waiting for another command.

The low level commands that the laptop can request are as follows (the subroutine names are given in brackets):

- Load data (cmd_3r); here a block of data is received from the laptop and stored into the DSP's program memory. The size of the transfer and the load address are specified in the first six bytes of the data. This command is used to load pulse programs or data into the DSP's program memory.
- Run PP (cmd_5r); here the previously loaded pulse program is executed, and when finished the pulse program finished output pin is set to indicate this to the laptop.
- Clear PPfin (cmd_1r); this command is used to reset the pulse program finished pin.
- Output data (cmd_4r); here a 1024x16 bit block of data within the external static RAM is sent to the laptop. The start address is specified by the current block pointer, and by default is $011000.
- Set BP (cmd_6r); this command resets the current block pointer to the default address ($011000).
- Inc BP (cmd_7r); this command increments the current block pointer so that it points to the next 1024x16 bit block.

The commands themselves make use of a set of I/O and timer routines, the most important two are called "inbyt" and "outbyt". These routines look after the actual transfer of each byte between the laptop and DSP board.

The pulse program is executable code for the DSP and is loaded into the program memory at address $000600. It is called as a subroutine and therefore it must make sure that it preserves the system's memory and variables. The pulse program does not necessarily have to be an NMR sequence, as being executable code it can perform any function that the user requires. This means that the system core can be used for applications other than NMR.

4.3.6.2 The laptop software

The communications protocol stack for the laptop part of the system is again shown in figure 4.19, but this time in more detail, with the MyAPI layer now being split into three layers.

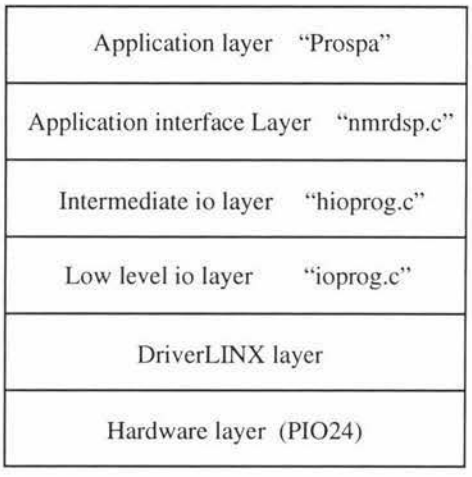


Fig 4.19 Laptop communications protocol stack.

The application layer on the top is the user's application, and in my case is Prospa, an NMR data processing package that was developed by Dr Craig Eccles a number of years ago and is currently being redesigned to include data acquisition and control. The next layer down is my application interface layer (nmrdsp.cpp). This layer consists of the set of functions (described in section 4.3.6) that the application can call. It is responsible for the communication link and tries to guarantee success by using multiple attempts if timeout errors occur. The intermediate I/O layer (hioprog.cpp) is a set of four commonly called functions that bridge the gap between the low level I/O routines layer and the upper applications interface layer. The functions provided by this layer are listed below:

- OutCmd. This function is responsible for sending a command to the DSP board which it does by calling a series of low level output requests.
- OutData. This function sets up some parameters and then calls a low level burst output routine.
- InData. This function sets up some parameters and then calls a low level burst input routine.
- Polldsp. This function samples the "Ppfin" pin by calling a low level single input routine.

The low level I/O (ioprog.cpp) layer consists of a set of single and multiple I/O transfer routines that call DriverLINX.

C source code listings for these 3 layers are contained within appendix B.

One problem of using protocol stacks is the often large numbers of layers involved and hence high overheads. In my case the DriverLINX layer overheads are far greater than any other layer so it makes little sense to try and optimise my layers. Flexibility comes at a cost.

4.3.6.3 Prospa

Prospa is a program that was initially developed for NMR data processing on Macintosh and UNIX platforms, but has now been ported over to the PC platform and extended to *control* NMR systems. This allows us to capture and analyse data within one software package, and as Prospa is developed in-house it can be modified and expanded easily to suit many applications, including ones other than NMR. The main features of Prospa are the built-in scripting language and the user programmable graphical window interfaces. Figure 4.20 below is a screen dump from the laptop with Prospa running and shows how the user interface can divided up into a number of sections. The individual windows can be resized and moved to any position, and other window interfaces can be generated if desired using a window layout script. In this way a window can be created that may contain some user defined buttons and a plotting area for displaying graphs. For now I have only been using three of the standard Prospa windows.

The first is the 1D plot window and appears at the top of the screen. This window is used to display the FID data or the corresponding spectrum, and included are some buttons that allow the data to be saved and later retrieved. A zoom button is also included to allow one to examine a part of the data in detail.

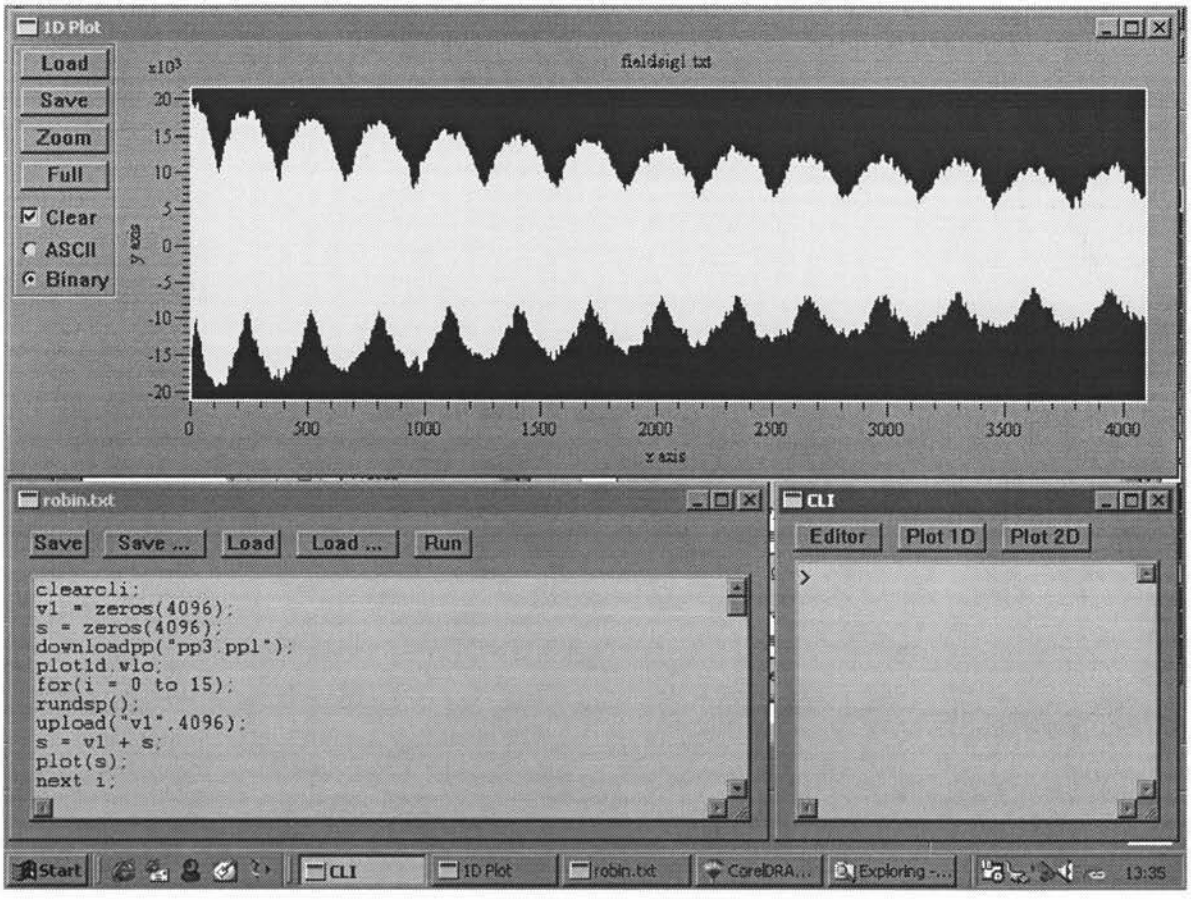


Fig 4.20 Screen dump of laptop with Prospa running.

The command line interface (CLI) is the next window and is shown on the bottom right hand side. It is used when only a single line command is required to be executed. An example is the "ft" command, which can be used to perform a Fourier transform of the data displayed in the 1D plot window. Prospa has many such commands (or functions) contained within its function library. The library itself can be easily expanded as it allows C code modules to be easily linked into it. The functions in the application interface layer that I wrote (nmrdsp.cpp) have been included within Prospa's function library and therefore can be executed from the CLI.

The macro editor and executor is the last window and is shown on the bottom left hand side of the screen dump. Prospa provides a macro scripting language that can be used to automate commonly used command sequences. All the functions contained within Prospa's function library can be used within macros. For-Next loops and If-Then-Else switches are also included to provide a full programming-like language. The macro displayed in this window (robin.txt) is one that I use to run a series of NMR experiments. It is reproduced below, with comments:

```
clearcli;
```
This command is used to clear the command line interface of any previous error messages.
```
v1 = zeros(4096);
```
This statement sets up a 4 K zeroed buffer for storing FIDs received from the DSP board.
```
s = zeros(4096);
```
This statement sets up a 4 K zeroed buffer for storing accumulated FIDs.
```
downloadpp("pp3.ppl");
```
Here the pulse program is downloaded to the DSP board. Note that "downloadpp" is one of the functions that I wrote and is contained within "nmrdsp.cpp"
```
plot1d.wlo;
```
This is a window layout script that generates the plot 1D window for displaying the FID etc.
```
for(i = 0 to 15);
```
16 individual scans are performed.
```
rundsp();
```
This command tells the DSP board to execute the pulse program, and is another one of the functions contained within "nmrdsp.cpp".
```
upload("v1",4096);
```
This command uploads the FID from the DSP board and stores it into the `v1` buffer, and again is a function contained within "nmrdsp.cpp".
```
s = v1 + s;
```
Here the FID is added to the previous ones. This is where the signal averaging is performed.
```
plot(s);
```
This command plots the contents of buffer `s` in the 1D plot window.
```
next i;
```
The end of the loop.

4.3.7 Pulse programs

Pulse programs are code that is written for the DSP and which implement the sequence of events that are required to perform an NMR experiment. For example, the polarizing of the sample, B_1 excitation and the subsequent sampling and storage of the FID. The pulse program is developed and stored on the laptop, and when it is required it is downloaded to the DSP board. It is then executed as a subroutine.

The pulse programs can be written with an assembler or C compiler which generate an object file (*.cld) that is stored in the Motorola's common object file format (COFF) [68]. This file contains a lot of additional information such as symbol tables, so I run the file through a conversion program (Cof2ppl) [66] that generates a file that can then be downloaded to the DSP board via Prospa. This conversion program firstly strips off all the unnecessary information, leaving just the machine code instructions, and then six bytes containing the load address and file size are inserted at the beginning of the file. The whole process of assembling, linking and file conversion is automated using a 'batch' file on the laptop, and an example batch file (robnz.bat) is listed below:

```
asm56300 -a -b -l -z robnz.asm
Cof2ppl robnz.cld
```

An individual pulse program is written for each type of experiment that the user wants to perform, but many sections are common to all experiments and most pulse programs follow a standard form. An example of a pulse program (robnz.asm) that performs a simple FID experiment that could be used to determine the water content of a sample, is given in appendix C. This program basically does the following:

1. Firstly it initialises the digital filter and A-D/D-A converter that are used later on in the sequence.
2. A polarising pulse of 4 seconds is then generated.
3. After a short delay a $90^0 B_1$ pulse is generated using a sine generator algorithm and a D-A converter.
4. The FID is then sampled, filtered and stored into memory.

The program contains additional code that is required to synchronise and precondition parts of the system.

In order to be able to do signal averaging we have to guarantee that for each FID the timing of the sample points is the same. The A-D/D-A converter, being a sigma-delta converter, samples continuously, and periodically interrupts the DSP in order to transfer the data. An interrupt routine then receives the data and places it into a buffer. This can be thought of as a task running in the background, as other programs running on the DSP system are unaware that this is happening. Therefore, at the beginning of the pulse program, the DSP needs to be synchronised with A-D/D-A converter sampling time. This is done by continuously monitoring the synchronous serial port in order to detect a transfer frame.

Before the $90^0 B_1$ pulse is generated, a dummy B_1 pulse is generated, but with the transmit gate open. This is required to settle the filters in the delta-sigma D-A converter and in the analog section. Also the sampling of the FID is started earlier than

needed, and this is done in order to preload the digital filter with data that is similar to the data of interest. This is often required, as the initial data values within the filter are all zeroes, so it takes a while for the filter to start working correctly.

The memory requirements of the DSP are also specific to the pulse program. Figure 4.21 shows the memory usage within the X and Y data memories of the DSP. The FID data is stored in the external static RAM, starting at address $011000 and the pulse program is loaded in program memory beginning at address $000600. Further parts of the memories could be used to store parameters and/or tables.

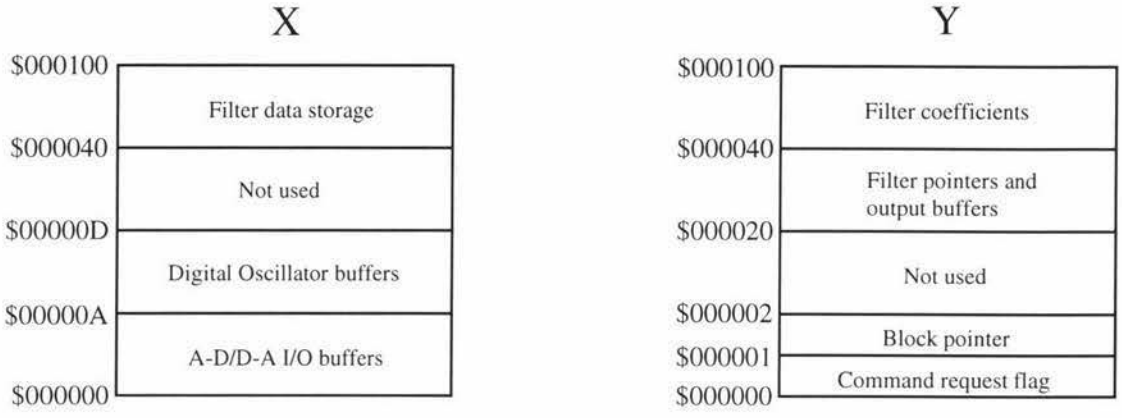


Fig 4.21 DSP X,Y memory allocation.

The "block pointer" and "command request flag" memory words are used by the DSP's operating system (mprog1.asm).

4.3.7.1 The digital oscillator

The digital oscillator used to generate the B_1 pulse can be thought of as a second order infinite impulse response (IIR) filter, but with no inputs, and its poles located on the unit circle [48]. The structure of the oscillator is shown in figure 4.22, and only one multiplication and subtraction is required for each iteration.

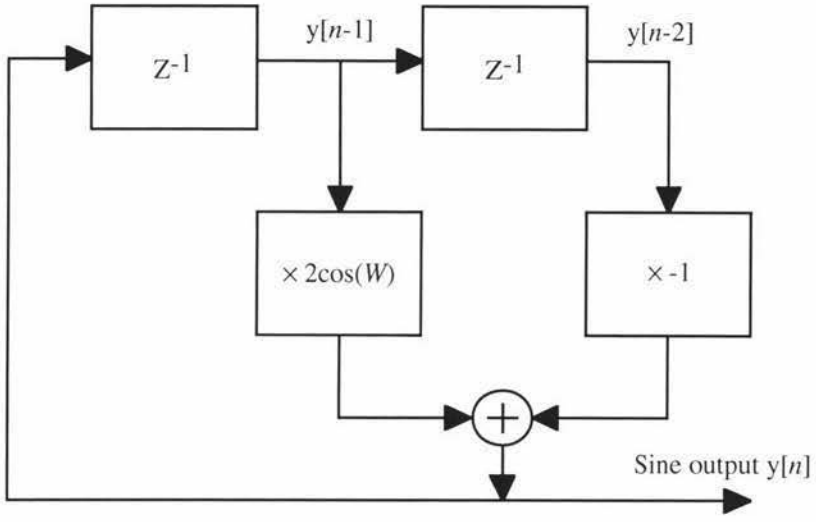


Fig 4.22 Digital oscillator structure.

The governing equation for the oscillator is:

$$y[n] = 2\cos(W)y[n-1] - y[n-2]$$

Where W is the phase increment for each iteration and is:

$$W = 2\pi f_{osc} / f_s$$

Where f_{osc} is the required output frequency and f_s is the output sample rate. Two seed values (y[-1] and y[-2]) are required to get the oscillator going and are simply the previous two values for the sine function. They can be calculated using:

$$y[-2] = G\sin(B - 2W)$$

$$y[-1] = G\sin(B - W)$$

Where B is the initial phase angle (-180^0 to $+180^0$) and G is the amplitude of the output which can be set from 0 to 1. The finite resolution of the DSP can lead to drift in amplitude and frequency [49] over long time periods, and this is another good reason for using a 24 bit processor. With NMR we only use short bursts, so any long term drift is not a problem.

The proof that this oscillator works is as follows:

Recall the complex exponential:

$$e^{jx} = \cos(x) + j\sin(x)$$

This represents a unit complex number whose angle with the positive real axis is x radians. The cosine of the angle is the real (horizontal) component and the sine is the imaginary (vertical) component. Now let $y[n]$ be an array of complex numbers (with $n = 0,1,2,3\ldots.$) defined as follows:

$$y[n] = e^{j(nW+B)}$$

Thus $y[n]$ is a unit complex number rotating about the origin. A complex number rotating at a fixed rate like this is called a phasor. The phasor starts at B radians when n=0, and rotates W radians per sample thereafter.

The trick in this proof is to consider the sum $y[n+1] + y[n-1]$, i.e. the sum of the previous sample and the next sample bounding the arbitrary phasor sample $y[n]$.

$$\begin{aligned}
y[n+1] + y[n-1] &= e^{j(n+1)W + jB} + e^{j(n-1)W + jB} \\
&= e^{jnW + jW + jB} + e^{jnW - jW + jB} \\
&= e^{jnW + jB}e^{jW} + e^{jnW + jB}e^{-jW} \\
&= y[n]e^{jW} + y[n]e^{-jW} \\
&= y[n](e^{jW} + e^{-jW}) \\
&= y[n](\cos(W) + j\sin(W) + \cos(-W) + j\sin(-W)) \\
&= y[n](\cos(W) + j\sin(W) + \cos(W) - j\sin(W)) \\
&= 2\cos(W)y[n]
\end{aligned}$$

From which we obtain the final result:

$$y[n+1] = 2\cos(W)y[n] - y[n-1]$$

or:

$$y[n] = 2\cos(W)y[n-1] - y[n-2]$$

4.3.7.2 The Digital filter

The digital filter is used to reduce unwanted interference and noise from the incoming FID. To do this a sufficiently narrow bandpass filter is used with its passband centred on the Larmor frequency of the expected FID. I chose to use a digital filter as they can be changed easily and rapidly to suit different conditions, such as different Larmor frequencies caused by variations in the Earth's magnetic field. Digital filters offer many advantages over analogue filters such as lack of drift due to temperature changes and aging, and they also offer good selectivity as they can be designed with very narrow bandwidths. Digital filters can also be designed with a linear phase response, therefore guaranteeing that no phase distortion will occur, unlike analogue filters where this can be a big problem.

With the digital filtering technique, most of the filtering is performed after the digitisation of the signal, unlike the traditional method where the signal is initially passed through analogue filters and then digitised. In order to prevent aliasing, the bandwidth of the signal must be limited before it is digitised. With the traditional method this was automatically performed as a result of the band pass filtering. With the digital filtering technique, an analogue low pass filter is therefore required before the digitiser, but its requirements can be greatly reduced by taking advantage of a process known as oversampling. Here the sample rate used is many times faster than is actually required to capture the signal, and hence the input bandwidth is much broader. This means that a simple first order low pass filter can be used, that has its cut off frequency way beyond that of the highest frequency of interest. As a result, no phase distortion occurs within the bandwidth of interest and the anti-aliasing requirements are met. After the signal has been sampled at the excessive sampling rate, it is bandwidth limited using a digital filter so that the sample rate can then be reduced. A spin off from this process is that the quantisation noise due to the analogue to digital conversion is reduced. The quantisation noise after the high speed sampling process is distributed within a wide part of the frequency spectrum. Low pass filtering performed by the digital filter greatly reduces the spectrum used and therefore large amounts of quantisation noise is eliminated. Using oversampling techniques it is possible, for example, to get 16 bit performance using 12 bit converters. This idea has been taken to the extreme with sigma-delta converters, which use a 1 bit quantiser and massive oversampling.

Normally the process of digital filtering and sample rate conversion is done as two separate steps in the signal processing chain; bandwidth limiting using a digital filter and then removal of some of the outputs to reduce the data rate (decimation). If samples are discarded then why calculate them in the first place ? This appears to be very inefficient in terms of processing time. The smart thing to do is to combine the filtering and the decimation phases, which can be done if a finite impulse response (FIR) filter is used [67] as shown in figure 4.23. A FIR filter only needs inputs, so there is no need to compute all the outputs if you intend to throw some away. The FIR filter does not feed previous outputs back into the filter as an infinite impulse response (IIR) filter does. If an IIR filter was used, it would still need to compute all the outputs in order to compute the next ones, even though it would later on throw some away. A FIR filter usually has many more coefficients than an IIR filter, but as it can take advantage of the down conversion phase, it can actually end up being more efficient in terms of processing time than an IIR filter.

Recall that filtering is a convolution process where the filter coefficients are multiplied with the input samples and the results summed to give a single output value. This is repeated for each new sample, with the new sample shifted in one end and the oldest sample discarded (FIFO). This is shown below (figure 4.23) for a six coefficient FIR filter.

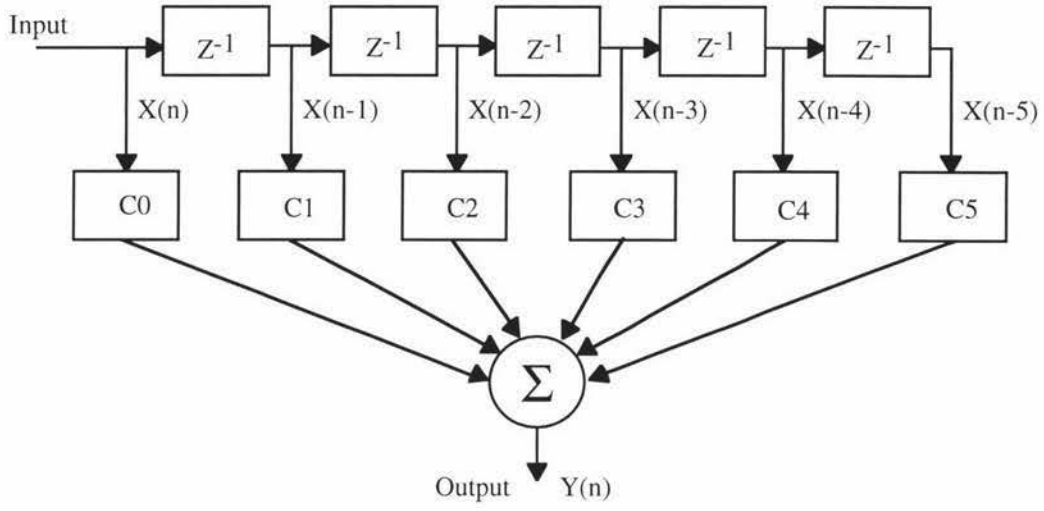


Fig 4.23 6 tap FIR filter structure.

The table shown in figure 4.24 shows how the input data and the filter coefficients contribute to the outputs.

Filter coefficients (C) and input data (X)

Input time	C0	C1	C2	C3	C4	C5	Output
n=0	X0	X-1	X-2	X-3	X-4	X-5	Y0
1	X1	X0	X-1	X-2	X-3	X-4	Y1
2	X2	X1	X0	X-1	X-2	X-3	Y2
3	X3	X2	X1	X0	X-1	X-2	Y3
4	X4	X3	X2	X1	X0	X-1	Y4
5	X5	X4	X3	X2	X1	X0	Y5
6	X6	X5	X4	X3	X2	X1	Y6

Fig 4.24 6 tap FIR filter table.

Where,

$$Y0 = C0*X0 + C1*X-1 + C2*X-2 + C3*X-3 + C4*X-4 + C5*X-5$$

If this type of filter is followed by a 2:1 down-sampler, only every second output will be used. This is shown in figure 4.25.

	Filter coefficients (C) and input data (X)							
Input time	C0	C1	C2	C3	C4	C5	Output	Used ?
n=0	X0	X-1	X-2	X-3	X-4	X-5	Y0	Yes
1	X1	X0	X-1	X-2	X-3	X-4	Y1	No
2	X2	X1	X0	X-1	X-2	X-3	Y2	Yes
3	X3	X2	X1	X0	X-1	X-2	Y3	No
4	X4	X3	X2	X1	X0	X-1	Y4	Yes
5	X5	X4	X3	X2	X1	X0	Y5	No
6	X6	X5	X4	X3	X2	X1	Y6	Yes

Fig 4.25 6 tap FIR filter with 2x down conversion table.

For the outputs that are kept, the following pattern emerges:

X0 uses C0,C2,C4
X1 uses C1,C3,C5
X2 uses C0,C2,C4
X3 uses C1,C3,C5 etc

There are two subfilters within this filter, (C0,C2,C4) and (C1,C3,C5), and alternate input data are switched between them. This is demonstrated in figure 4.26.

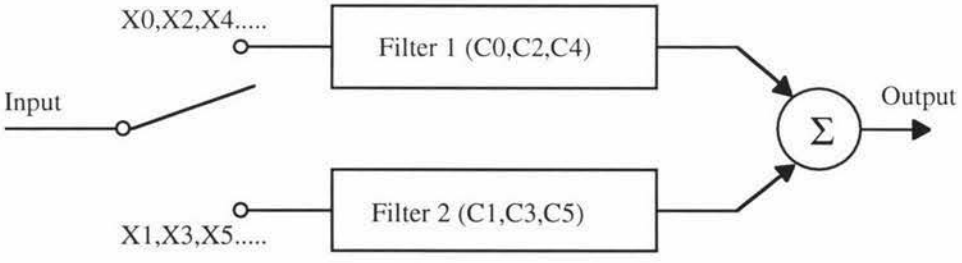


Fig 4.26 6 coefficient FIR filter with 2x down conversion, broken down into two subfilters.

Each subfilter, upon receiving a sample, computes an output, and when all the subfilters have received a new sample, all their outputs are summed together to generate one output value at the lower sampling rate. One big advantage using this filter structure is that the processing is broken up into smaller blocks. Only the computation of a single subfilter output is required every input sample time, which makes it easier to do the processing in real time.

The digital filter that I designed has 192 coefficients and 6 subfilters to give an output sample rate of 8 kHz for a 48 kHz input rate. Initially the filter is designed as a single 192 coefficient FIR filter. The coefficients are then distributed in order among the subfilters as shown in figure 4.27.

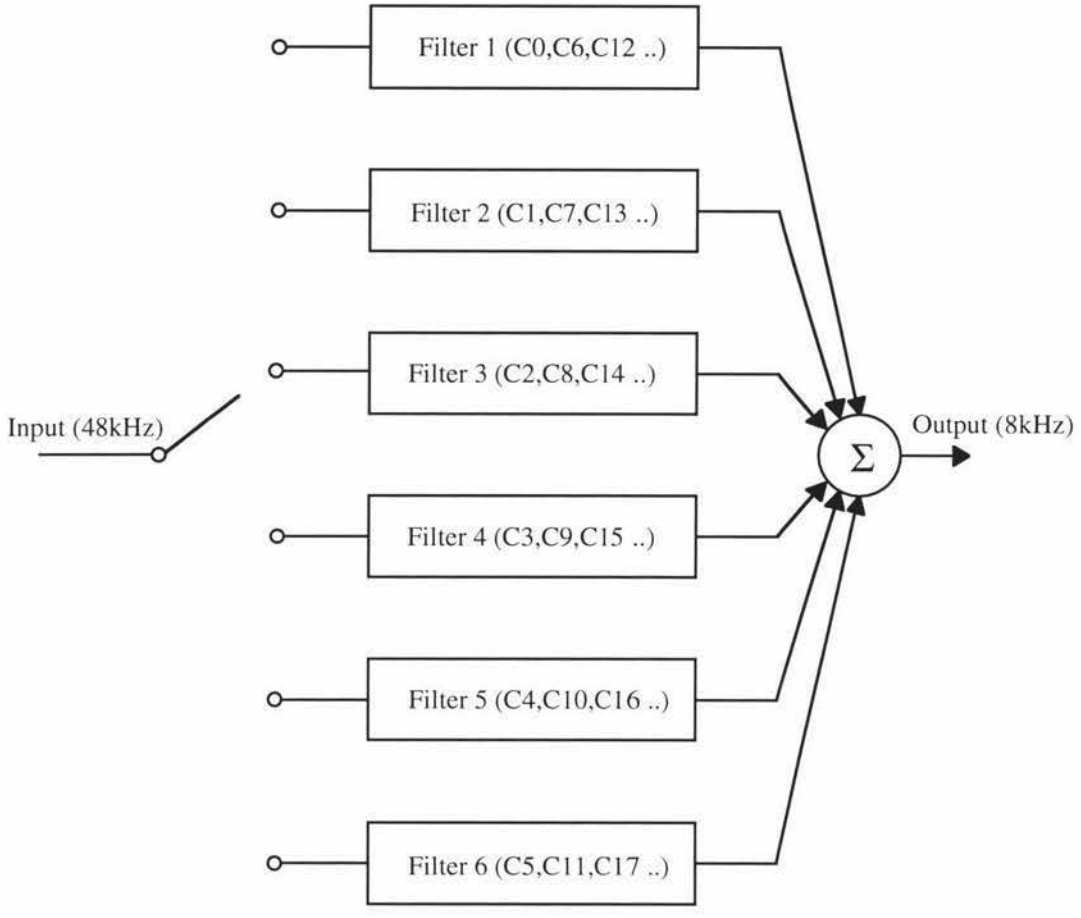


Fig 4.27 192 coefficient FIR filter with 6x down conversion, broken down to six subfilters.

This filter structure has been implemented for the 56303 DSP as a filter routine (do_fltr) that is called every time a new sample arrives. The listing of it can be found in appendix C. For the six subfilters, twelve memory buffers are used, six for the filter coefficients and six for the data. The data buffers are implemented as circular buffers using special DSP modulo addressing modes. The data is not actually shifted, but circular pointers are used to point within a block of memory. The pointer, when reaching the end of a data block, is automatically reset to point to the beginning. In this way new data is just stored in the location of the previous oldest data, so the start of the actual data moves around the block of memory. This is all handled automatically by the DSP and therefore makes programming a lot easier and results in much faster execution. The filter coefficients are also stored in data buffers, but require no special addressing as the filter output calculation always starts with the coefficient at the beginning of the buffer.

Register r2 is used as a filter number counter and determines which subfilter should be used for the incoming sample. It is updated every time the routine is called, but can be

reset by the calling routine so that it can be synchronised if needed. A block of DSP Y data memory is used to store the intermediate results of each subfilter, and when the filter number counter selects the last subfilter, all the intermediate results are summed together to generate an output sample, which is then stored into the external data memory of the DSP board. Some other DSP Y data memory is used to store a set of pointers that are used to point to the oldest data within the data buffers. When the pulse program is run, the filter data and coefficient buffers and pointers are initialised from the data that appears at the end of the pulse program, which is stored in the DSP's program memory.

The subfilters themselves use a standard FIR filter program structure that takes advantage of the DSP's multiply-accumulate instruction (mac) and the simultaneous X and Y data moves. The piece of program code that is the filter core is reproduced below:

```
move    a,x:(r0)              ;input sample in memory
clr     a       x:(r0)+,x0  y:(r4)+,y0     ;clear accumulator
move    r0,y:(r1)             ;store data pointer
rep     #ncoef-1              ;do the next instruction 31 times
mac     x0,y0,a x:(r0)+,x0  y:(r4)+,y0     ;
macr    x0,y0,a (r0)-
move    a,y:(r3)              ;output filtered sample and
move    (r2)+                 ;increment filter number counter
```

Where r0 points to the signal data and r4 to the filter coefficients. The line:

```
clr     a       x:(r0)+,x0  y:(r4)+,y0
```

clears the accumulator and gets the first data and coefficient values ready for the first multiply-accumulate.

```
move    r0,y:(r1)
```

saves the pointer that points to the location for storing the next subfilter input sample.

```
rep     #ncoef-1
```

This line tells the DSP to execute the following instruction 31 times.

```
mac     x0,y0,a x:(r0)+,x0   y:(r4)+,y0
```

This line is executed 31 times (for a 32 coefficient filter) and coupled with the next line:

```
macr    x0,y0,a (r0)-
```

performs the convolution between the signal data and the filter coefficients. The "macr" instruction is the same as the "mac" instruction but performs rounding of the result.

4.3.7.3 Digital filter design

To design my digital filter I used Matlab and one of its extensions, the "signal processing toolbox". Matlab allows macros to be written which are called "m" files, and one of these that I wrote to design my filters is listed in appendix D. The first part of my filter design macro is reproduced below:

```
% Filter for Earths field NMR
% This one is for working in Antarctica
% The sample rate is 48kHz
% The filter is a bandpass type with a passband
% between 2.6 and 2.9 kHz.
% It is an FIR filter with 192 taps.
n = 192;
m = [0 0 1 1 0 0];
f = [0 0.09 0.11 0.12 0.14 1];
b = remez(n-1,f,m);
[H,w] = freqz(b);
L = 10*(log(H));
plot(24*w/pi,L);
```

The function "remez" which is contained within the signal processing toolbox of Matlab is an optimal FIR filter design function that is an implementation of the Parks-McClellan algorithm [49]. The filter type and frequency response requirements are passed to the function, which then generates a set of filter coefficients that can be used to implement the filter. The filter design algorithm tries to match the desired response as close as possible by performing a number of iterations. The array "m" specifies the desired filter type, where "1's" are used to indicate the pass band. The array "f" specifies the frequencies (as a fraction of the bandwidth) of the transition points between the pass and stop bands defined by array "m". The example above uses six frequency components to describe a bandpass filter. Four components to define the stopband/passband transition frequencies (and slopes) and two to define the operating frequency range of the filter. In my case the operating frequency range is from DC to 24 kHz.

The function "freqz", also contained within the signal processing toolbox, is used to generate a frequency response plot from the filter coefficients.

The rest of my filter design macro, which is not shown above, formats and arranges the coefficients, so that they can be copied and pasted into the pulse program. It is planned to write our own remez function for Prospa, based on a C source listing that I have obtained [48]. This would enable us to rapidly compute and then down-load new digital filters to the DSP board.

For Antarctica I needed a sufficiently narrow band-pass filter that was centred around 2.7 kHz. The macro listing above was used to generate this, and a frequency response plot of the resultant filter is given in figure 4.28.

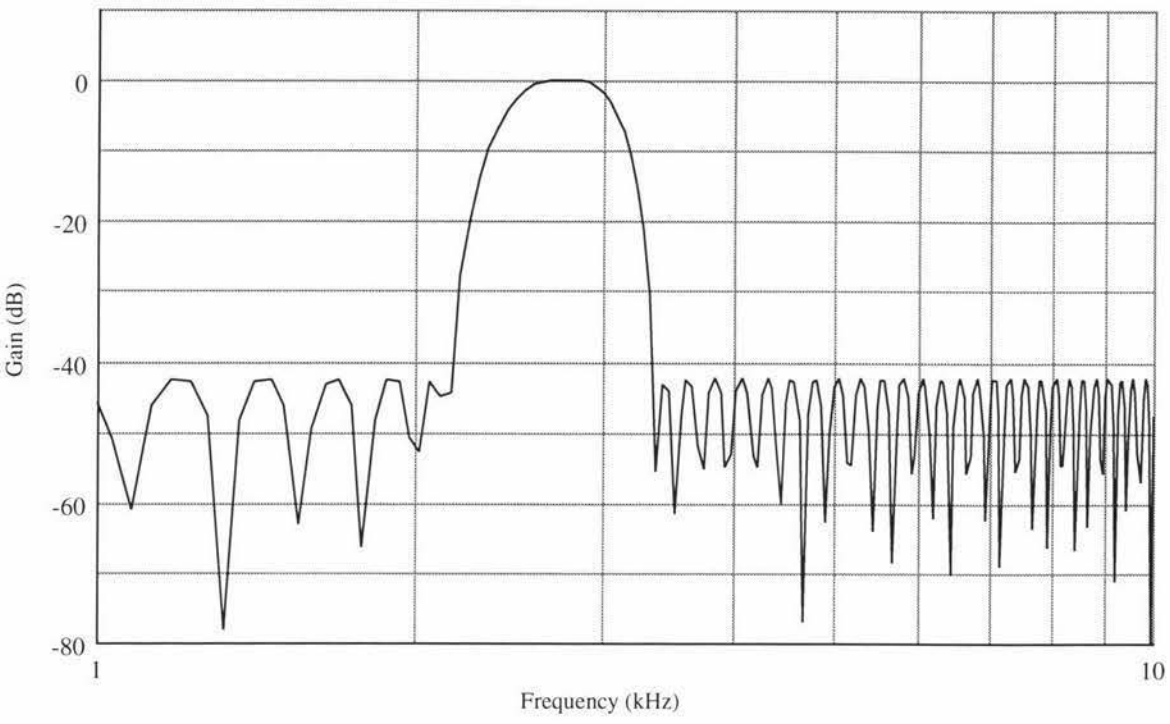


Fig 4.28 Frequency response of the digital filter used in Antarctica.

4.4 New developments

Since returning from Antarctica the aim of my work was to continue developing the system core. I decided that the next step would be to replace the PCMCIA/DSP interface board with a USB/DSP interface board, this would be a big step in the development of my own DSP board that contained a USB interface. This approach would also allow me to thoroughly debug the USB part of the system before committing everything to a final board design.

4.4.1 Introduction to USB

USB is still in its infancy as most of the documentation is rather poor and lacking in detail and examples. But a few books [63, 64] have recently become available that provide some help and examples. Some web sites [69] are now also available with further information and resources for USB developers. Reverse engineering is often required to work out some details of the USB hardware/software protocols.

USB is a network that uses a tiered star topology, to connect multiple devices. This is done with the use of hubs, which are the nodes of the stars within the network. Differential signalling is used to transmit and receive packets between a host controller and a device, in a half duplex arrangement that is capable of transferring data at 12 Mbit/s. The packets are organised into 1 ms duration frames and there are different types of packets, which are listed as follows:

- The start of frame token packet (SOF) is sent by the root hub every 1 ms and is used as a "heartbeat" signal to synchronise devices. Within the packet is an 11 bit frame number counter that rolls over every 2 seconds and can be used by devices as a real time clock. This packet is very short in total, to allow the possible transmission of 1500 bytes of data between each SOF packet.
- The root hub also sends out other token packets when it wants to communicate with devices. The first two of these are the IN and OUT token packets, and are used by the controller to indicate that it wants to read or write data from/to the device. The last of these is the SETUP token packet, which is used to send data (usually a request) to the device, which it must accept and action.
- The next type of packet are the DATA0 and DATA1 packets. They are used by the controller or device to transfer data and they alternate to provide a way of detecting lost packets. The length of the data packet can vary from 0 to 1023 bytes, this is to allow for the variation in required transfer sizes and therefore bandwidth wastage is avoided. Multiple data packets can be sent during a single 1ms frame.
- The last type of packet are the handshake packets, ACK, NAK and STALL, and are used to indicate successful or various levels of unsuccessful transmission.

Each packet starts with an 8 bit SYNC sequence that is used to synchronise the transmit and receive clocks, this is then followed by an 8 bit packet identifier. Within a 1 ms frame, multiple packets can be sent between the controller and the devices.

USB devices can have a number of different "endpoints", which are individual buffers that allow parts of the USB device to be uniquely addressed. Using this method multiple transfer types can be in progress simultaneously. Before a data transfer can

occur, the host and the device must establish a "pipe", which is a logical connection between a device's endpoint and the host controller's software.

Each USB device contains a set of "descriptors", which are tables that contain information about the device, its configuration and its endpoints. When a device is connected to a USB network, the host controller interrogates the device and obtains the information stored in the descriptors.

USB uses four different types of transfers, which are predefined sequences of packets used to define data movement between the controller and a device's endpoint. These are listed as follows:

- Interrupt transfers. This is when the controller is set up to poll the device periodically (up to 1000 times a second) to determine if it needs attention. This would be used for devices such as computer mice, where only small amounts of data need to be transferred but without any significant delay.
- Bulk Transfers. This type is used to transfer large amounts of data, but without any guaranteed delivery time or rate. It is a low priority transfer type, which makes use of any available spare bandwidth that is not being used for other transfers. If many busy devices are connected to the USB network, bulk transfers can become very slow. But if network activity is low, transfer rates of greater than 1 Mbyte/s are easily obtainable. This type of transfer is commonly used for devices such as printers and scanners.
- Isochronous Transfers. This is used to transfer a packet of data between the controller and a device every frame. Here the host computer ensures that there is enough available bandwidth before it agrees to the connection. Once set up the device is guaranteed a slice of every frame. This type of transfer is used for transferring real time data, such as that from modems or audio/video systems.
- Control Transfers. These are the most complicated transfers and are used for the system control of devices. The controller uses these types of transfers to request devices to perform various standard and vendor defined functions. A control transfer consists of three transaction phases. Firstly a setup phase, which consists of a SETUP packet, a DATA0 packet and a handshake (ACK) packet. The data0 packet (figure 4.29) always contains 8 bytes, which specify the transfer request type, the actual request and some other parameters particular to the request. The next phase is an optional data phase and is used to transfer data in either direction between the controller and the device. The setup phase will specify if the optional data phase is required and what the transfer direction is. The last phase of the transfer is the status phase, which is used to indicate the successful execution of the requested function and the data transmission. Each USB device has a control endpoint (endpoint0) that handles control transfers.

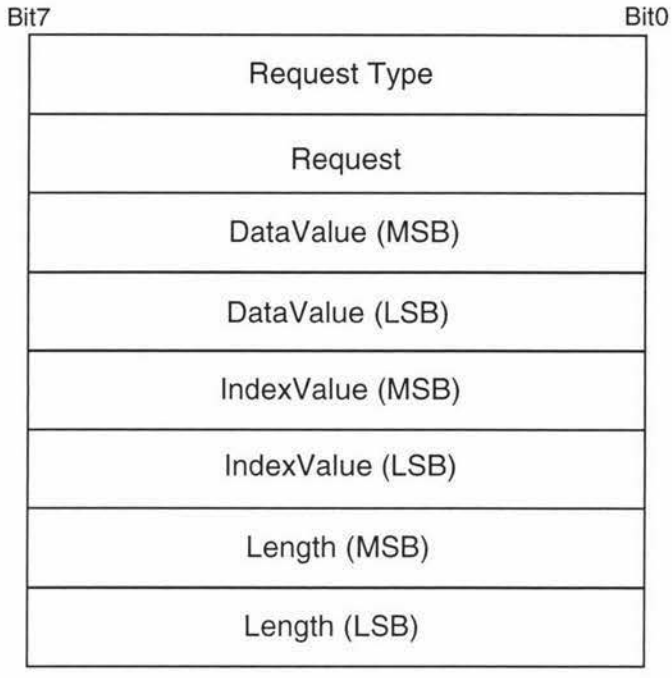


Fig 4.29 8 byte DATA0 packet within the setup phase of a control request.

The 8 byte data packet (figure 4.29) contained within the setup phase of a control request specifies what is required of the device. The first byte specifies the "Request Type" and is divided into a number of bit fields which are defined as follows:

- Bit 7 is used to indicate the direction of the optional data stage and a "0" indicates the host to device direction.
- Bits 6 and 5 are request type bits and specify if the request is a standard USB request or a user defined or class request.
- The rest of the bits are used for the optional addressing of the various components within a USB device.

The next byte is the "Request" value and indicates what function is being requested of the USB device. The "Data Value" and "Index Value" are words that the host may use to pass information to the device. The final word is the "Length" and it specifies the length of the optional data phase that follows the setup phase.

The standard requests are those that all USB devices need to support, in order to be able to be connected to the network. They are defined in chapter 9 [58] of the USB specification, and some of them are listed as follows:

- Set_Address. This request sets the USB device's address for all future accesses. When a USB device is first powered up it uses the default address of 0.
- Get_Descriptor. This request is used by the host to collect information about the device, so that it knows what it is, what it is capable of, and how it should communicate with it.

Vendor requests are requests that are specific to each device and allow a way for vendors to implement their own functions.

When a USB device is connected to the network, either through plugging it in or powering it up, it and the host go through an enumeration process which configures

the device and the host for correct operation. The steps that are performed are listed as follows:

- Firstly the host resets the USB device's interface to bring it into a known state with a device address of 0.
- The host then reads the device descriptor using the Get_Descriptor control request in order to discover what kind of device has been attached.
- An address is then assigned to the device using the Set_Address control request.
- The host then reads other descriptor information to build up its knowledge about the device.
- Using the information that the host has gathered, the host then tries to select a suitable device driver that applications will need to use in order to communicate with the device. Windows 98 and 2000 use "INF" files to describe what device driver to use for each type of device. These files and device drivers are usually provided with USB devices.
- After the device driver is loaded, the USB device is brought into a "configured" state indicating that everything is functioning and that the operating system can support the device.

An application running on the host computer can then use a series of standard and vendor requests to initiate the device to perform various functions. USB is very much a master/slave arrangement, with a single master and many slave devices. The host computer is responsible for monitoring the status of the devices and for initiating any transfers if required.

4.4.2 Philips PDIUSBD12

The USB interface chip that I chose to use is the PDIUSBD12 from Philips semiconductors [59]. It is a peripheral chip that can be connected to the data bus of a microprocessor and accessed as memory. The heart of the D12 chip is the Serial Interface Engine (SIE) which implements the full USB protocol layer. It is hardwired for speed and needs no firmware intervention as it looks after many functions including: synchronisation pattern recognition, parallel/serial conversion, bit stuffing/de-stuffing, CRC checking/generation, address recognition, and handshake evaluation/generation.

The D12 also includes some internal data buffers, which help it to achieve a 1 Mbyte/s data transfer rate for bulk modes, and the interface to the host microprocessor is capable of transferring data at the rate of 2 Mbytes/s. The D12 has three endpoints, a control endpoint (0) that handles all the control requests, a generic endpoint (1) that can be used for interrupt or bulk transfers, and a main endpoint (2) that can be used for isochronous and bulk transfers. The control and generic endpoints both have a maximum packet size of 16 bytes and the main endpoint is limited to a packet size of 64 bytes.

The D12 uses a set of commands that a host microprocessor sends to it in order to control its operation. Associated with each command is a data phase, where data is transferred between the D12 and the host microprocessor. Some examples of these commands are listed as follows:

- Set Address/Enable. This is one of the initialisation commands, and is used to assign the address, and enable the endpoints of the device. The command byte is followed by a single data byte write from the host microprocessor which specifies the address.
- Select Endpoint. This command initialises an internal pointer to the start of the selected endpoint buffer. Optionally, this command can be followed by a data read which sends a status byte to the host microprocessor indicating whether the buffer is full or empty or not functioning.
- Read Buffer. This command would be sent after a select endpoint command, and is used to initiate the transfer of multiple bytes of data from an endpoint buffer to the host microprocessor.
- Write Buffer. Like the Read buffer command it is used to transfer data, but in the opposite direction.

The D12 also has a "goodlink" output that can be connected to an LED to indicate when the device is successfully enumerated and configured. The LED also blinks when data is being transferred between the host and the device. The goodlink feature has proved to be very useful when debugging the system.

Also available from Philips is a D12 evaluation system [60] that consists of a circuit board with an 8 bit microcontroller and a D12 chip mounted on it. The board behaves as a USB device and can be used to kick-start the development of custom USB devices. I purchased one of them so that I could learn about USB hardware/software by doing a bit of reverse engineering. The evaluation system came with all the source code for the boards firmware as well as a device driver, an INF file, and a simple test application for Windows98.

USB development consists of two parts, firstly the design, construction and programming of a USB device, and secondly the development of a device driver, an INF file, and an application for the host computer. I also decided to split my USB development into two parts. The first part would be to get the D12 chip working with the 56303 DSP and essentially emulate the Philips evaluation board. This would mean that I could use the Philips device driver and test application to test my own USB device. The next phase would be to develop my own device driver and test application.

4.4.3 USB/DSP interface board.

A USB/DSP interface board was designed to be a plug in replacement for the PCMCIA/DSP interface board and this was done to minimise the modifications to the "digital box". Again the host port of the DSP was used and this is shown in the schematic diagram (figure 4.30) for the USB/DSP interface board.

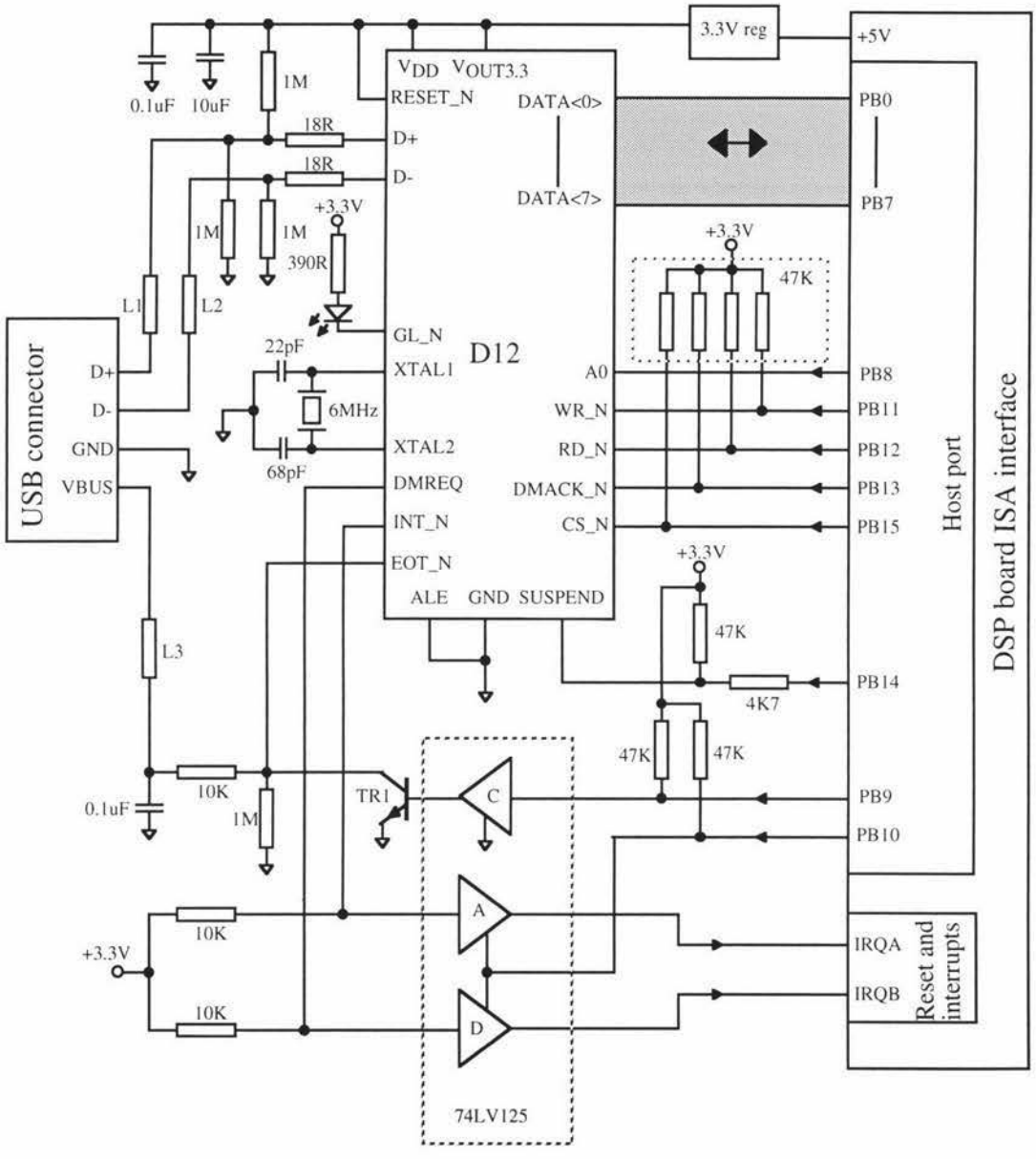


Fig 4.30 USB/DSP interface board schematic.

The DSP host port is configured for general purpose I/O with the lower 8 bits acting as a bidirectional data bus, and this is done by suitably altering the port data direction bits when required. All the other host port bits are permanently configured as outputs and are used to drive the various inputs of the D12 chip. The software running on the DSP generates the necessary control signals to allow it to communicate with the D12.

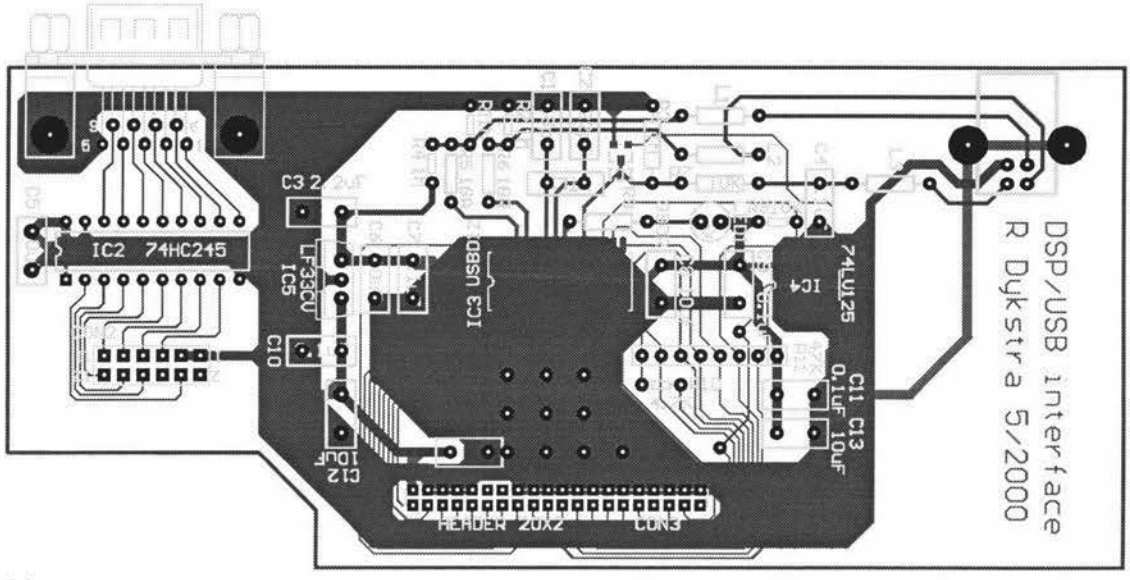
The D12, with the A0 input set high, accepts single byte commands that are written into it. This command phase is followed by a variable length data phase, where data is transferred a byte at a time between the D12 and the DSP, with A0 set to 0. In this way the D12 behaves like two successive memory locations that are addressable through the use of the A0 input pin. The D12 can gain the attention of the DSP through the use of the DSP interrupt inputs. The IRQA interrupt is used when a packet is transmitted or received and the D12 then requires that the DSP read the received data or write some data for the D12 to transmit. The other interrupt (IRQB) input can be used by the D12 to signal the DSP that it wants to perform DMA between its main endpoint buffer and the DSP's memory. At the moment this feature is not used but the hardware has been put in place in case it is required in the future.

The interrupt input pins have a secondary function (mode control), such that when the DSP undergoes a reset, the status of the interrupt pins determine what operating mode the DSP will use. This feature is used to allow the hard-wired programming of which source the DSP uses to boot up from. A range of boot sources can be used, such as an external EEPROM or another processor connected to the host port. The D12 INT_N and DMREQ outputs are held low after reset and if they were directly connected to the interrupt inputs of the DSP they would interfere with the mode selection phase. Open collector tri-state buffers are therefore used to disconnect the signals until the DSP enables them. Also, during a DSP reset, the host port is tri-stated, therefore some pullup resistors are required to prevent erroneous signals from appearing on the input pins of the D12 and to make sure that it is disabled.

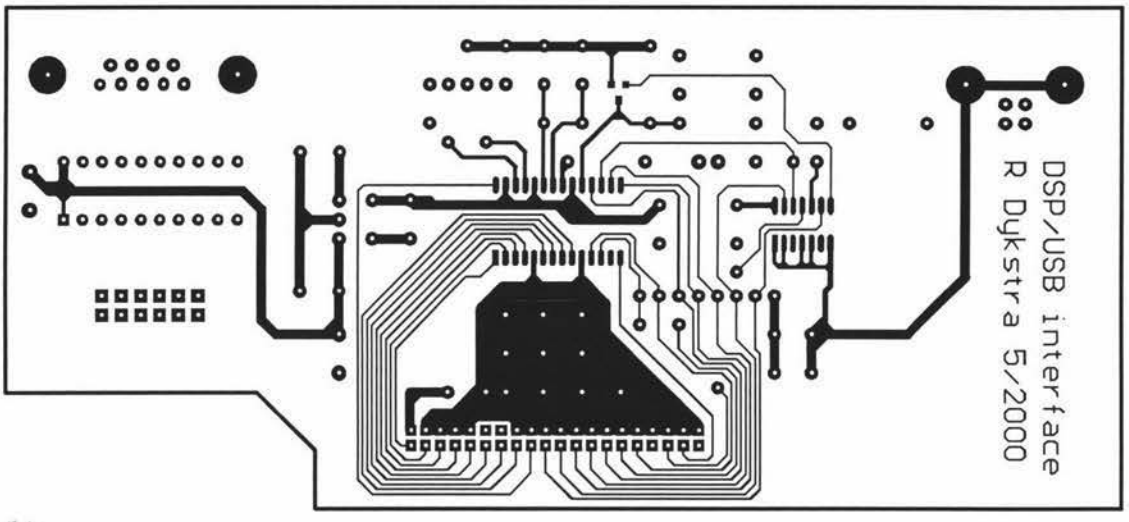
If a device draws its power from the bus and sees no activity on the bus for 3 ms, it must enter a suspended state and then consume minimal power. The suspend pin of the D12 is an open collector output that is used to signal external hardware that a suspended state has been entered into. This output is normally low and is released when the D12 is suspended. The suspend pin is also an input, to allow a signal to initiate a remote wake up of the D12 when it is suspended, and therefore force it out of the suspended state. PB14 of the DSP's host port is connected to the suspend pin in order to provide the remote wake up feature if required. The 4K7 resistor in series with the PB14 pin is used as protection against the faulty situation in which the port output pin is high and the D12 pulls the suspend pin low.

The EOT_N pin of the D12 is an input pin that is used by other hardware to signal to the D12 when a DMA transfer is to be terminated. This pin is also used to monitor the 5V VBUS supply voltage of the USB bus in order to determine if the bus exists before the D12 tries to connect to it.

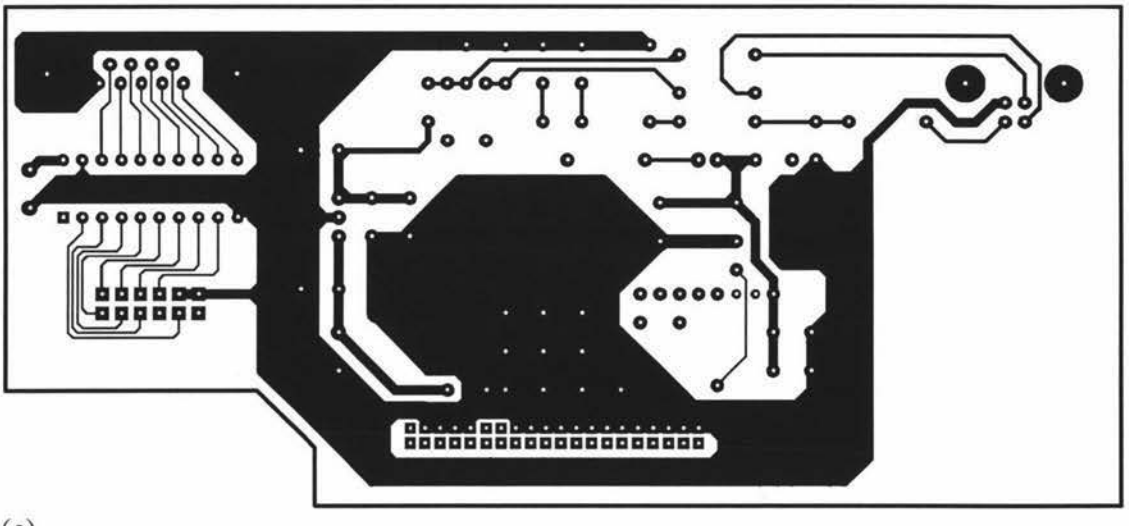
The D12 requires a 3.3V supply, which is good as it is the same as the 56303 and therefore no level translation is required. The supply is provided from a 5 V feed from the DSP board, which is then regulated down to 3.3 V. The LED in series with the 390 Ω resistor is the "goodlink" indicator and the D12 uses its own clock, derived from a 6 MHz crystal. I laid out the circuit onto a two layer board as shown in figure 4.31 and figure 4.32 shows the "digital box" with the replacement board.



(a)



(b)



(c)

Fig 4.31 Circuit board layouts for the DSP/USB interface board, overlay (a), top layer (b) and bottom layer (c).

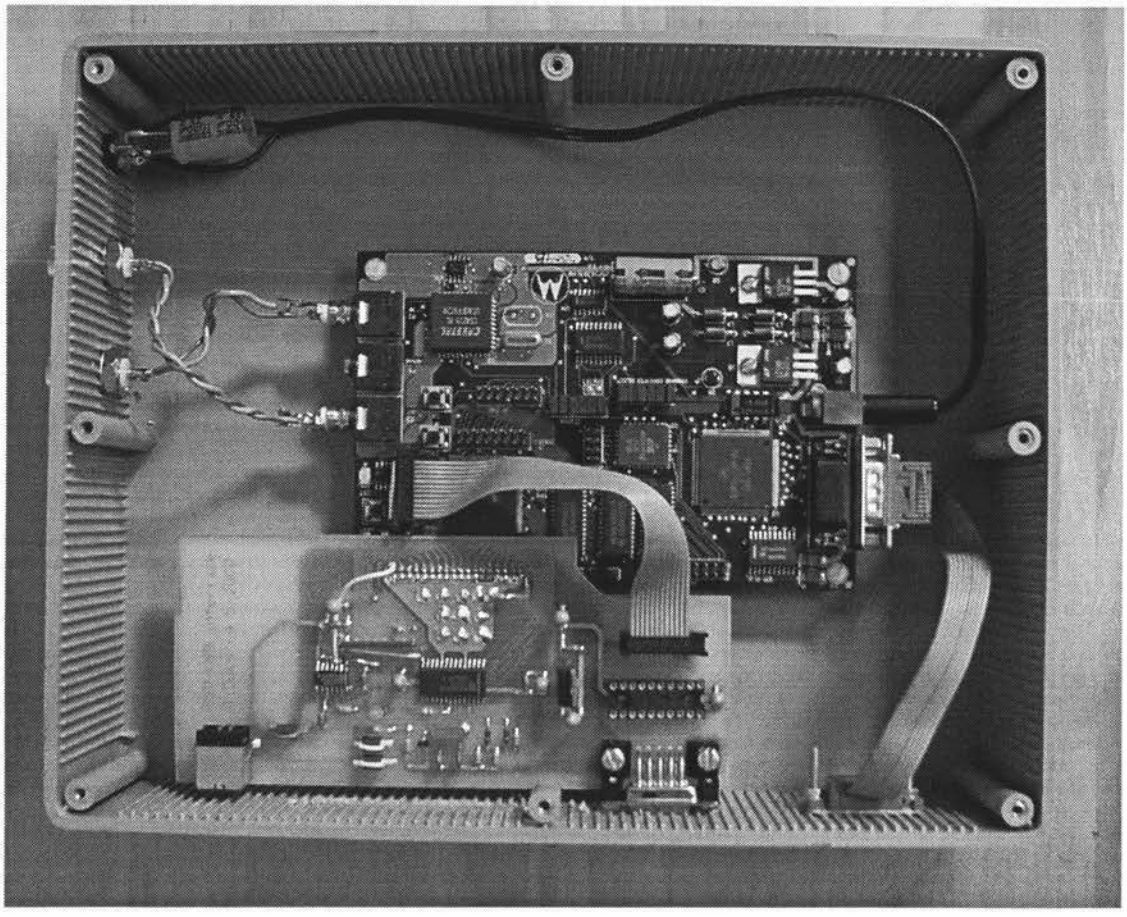


Fig 4.32 Arrangement of the USB/DSP interface board within the "digital" box.

4.4.4 USB/DSP software

For a number of reasons I also wanted to change from using an assembler to a C compiler to generate DSP code. The first reason is that the USB development system that I purchased, provided all the C source code listings for the board's firmware as it was developed in C. Therefore I could use large parts of the provided software and hence save vast amounts of time. Another reason is that the DSP is rather a difficult "beast" to program using assembly language, as it has many complicated instructions, data representations and addressing modes. Another reason for using C is that the source code is much more readable and intelligible, which is important if others want to understand how it works, in order to modify it.

There are some disadvantages to moving over to C. The first one is that the firmware is much larger and therefore more program memory is required. I thought about this, as it could be a major problem with the DSP56303. But my problem was solved, when Motorola released a new larger memory version of the 56303. It is called the 56309 [56] and is identical to the 56303 apart from the memories. The program memory has been expanded from 4 K to 20 K and the X and Y data memories each have been expanded from 2 K to 7 K. The 56309 is also available in a 100 MHz flat pack surface mount package, the same as the 56303. This new DSP device should have enough memory to cope with the extra demands of the C based operating system and therefore will be the device that I will use for the final DSP board design.

Another disadvantage with using C is that the code is usually slower, but the C compiler used is an optimising compiler [54], in that it tries to produce the fastest possible code. If the programmer wants maximum performance and/or the precise control over the timing of events, an assembler can be used to write some routines that can be linked in and called by the C program.

To write the new operating system for the DSP board with the USB interface in C, required the following:

- Firstly, setting up the C compiler so that it was compatible with the DSP board and the configuration of the C executing environment, which requires the allocation of parts of the memory for the C core's variables and stack.
- Secondly, a number of I/O and interrupt routines had to be written in assembler and then linked [53] into the C environment. With C, the caller and the called routine share the responsibility of preserving the DSP registers, which is done during compilation with each looking after half the registers. This is a big trap when writing routines in assembler that call C functions, such as interrupt routines, as one would normally expect the C function to preserve the registers. This makes it quite inefficient and slow for interrupt routines that call C functions as half the registers need to be saved and later retrieved every time an interrupt occurs. However, one could look through the assembly listing of the compiled C function to determine what registers it uses, and then only worry about preserving those. This would improve the interrupt handling performance of the system.
- Thirdly, the C source code had to be modified for a 24 bit big endian processor as it was originally written for an 8 bit little endian processor [60].

The USB/DSP operating system consists of a number of software modules that interact with each other in a layered hierarchical fashion as shown in figure 4.33.

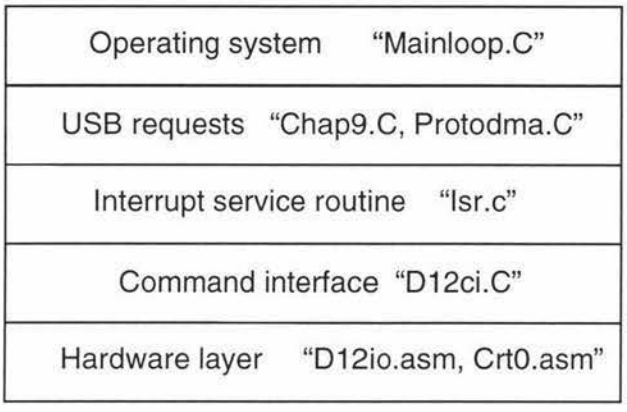


Fig 4.33 Layered model of the software.

The hardware layer is the interface between the hardware and the C part of the operating system. D12io.asm contains some I/O functions that the upper layer (D12ci.C) can call in order to transfer commands and data to the D12. Crt0.asm looks after the system initialisation following a reset and then passes control to Mainloop.C. It also handles the hardware interrupt from the D12 and calls the C interrupt service routine, Isr.C, which is responsible for transferring data between memory buffers and the D12, and for the setting of flags to indicate events to the upper layers. The module D12ci.C provides a set of command interfaces, that are available to all the layers above it, that encapsulate all the functions used to access the D12. Standard USB requests are handled by the module Chap9.C, and likewise Protodma.C handles all the vendor requests. The main loop (Mainloop.C) monitors event flags and calls the appropriate subroutines to handle them. For example, when a USB request is received by the interrupt service routine, a flag is set indicating a pending request. The mainloop then detects the pending request, works out if it is a standard or a custom vendor defined request, and then calls the appropriate request handler. All the source code is available in the USB folder of the CD contained in the back of this thesis.

After many hours of debugging, the USB/DSP system was fully functional and successfully emulated the D12 evaluation board. In fact it proved to be extremely reliable and robust, unlike the D12 evaluation board which often crashed. Testing of the board was carried out using the test application and device driver that was provided with the D12 evaluation system. The test application sends 256 byte blocks of data to the USB device, receives it back, and then checks for any errors. This is repeated in an endless loop so it can be left running to make sure no errors occur. The test application uses bulk transfers to send and receive the data, and vendor requests to initiate them. The test was left running over a day and no errors occurred.

To make the USB/DSP system a stand alone device, I had to get it to boot itself up after a reset. To do this, the operating system and a bootloader routine had to be stored in the EEPROM mounted on the DSP board. This was done by using the hardware debugger to load the operating system software and all the C predefined variables into the DSP's program memory. An EEPROM programming routine that

contained the code for a bootloader was then also loaded into the DSP's program memory which, when executed, firstly copied the bootloader and then the operating system and variables data into the EEPROM. When the DSP is reset in the EEPROM bootstrap mode, it loads in the first part of the EEPROM that contains the bootloader into a region of program memory and then starts executing it. The bootloader then copies the operating system software and all the predefined C variables into the appropriate DSP memories and then starts executing the operating system. This technique is known as a double bootstrap. The first six bytes of the bootloader, which are stored at the beginning of the EEPROM, tell the DSP how many bytes it has to load and where to save them. The load address also specifies where the DSP should start executing from after it has finished loading.

Getting the C and USB parts working took considerable amounts of time as many manuals and software source code listings had to be read and deciphered in order to gain enough knowledge to enable me to develop my own system. Also countless hours were spent debugging the various components of the system software and hardware.

4.4.5 Device driver and API

The next phase was to write my own device driver and modify MyAPI so that Prospa could then be used with the USB/DSP system. Unfortunately at this stage of the project I had ran out of time to be able to do this work and then present it in this thesis. I have done some preliminary groundwork so this should be completed in the near future.

Device drivers are not the easiest things to develop, but there is now very good developer support available from Microsoft [70] and various publications [65], which provide driver development kits (DDKs) and wizards. Windows98, for USB, uses a series of kernel mode (trusted) drivers organised in hierarchical layered fashion as shown in figure 4.34. The driver that I will develop is a kernel mode custom driver that sits between the application and the USB bus driver. The driver that came with the D12 evaluation board is D12test.sys.

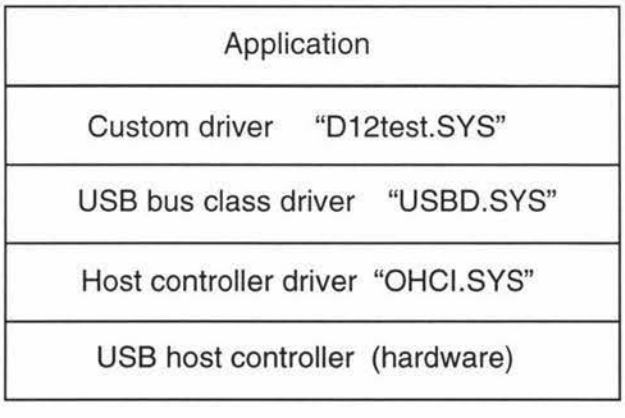


Fig 4.34 Layered device driver model.

Applications can communicate with the uppermost driver (custom) through a set of C functions provided by the system, known as an application programmers interface (API). Some of the functions provided by theWin32 API that can be called by the application are listed as follows:

- CreateFile. This is used to obtain a unique identifier (handle) to the device that is used for all subsequent accesses.
- ReadFile. Is used to retrieve data from a device using bulk transfers.
- WriteFile. Used to send data to a device using bulk transfers.
- DeviceIOControl. This is a more flexible function that is used to perform many different operations including reading and writing data to the device. The required operation is selected by using a predefined I/O control code.

The Windows API converts these function calls into an I/O request packet (IRP), and then passes them down to the uppermost device driver. Each IRP contains an I/O control code, which describes what operation is required. The task of the device driver is to process the IRP, by calling a function that matches the I/O control code. The called function then has the task of generating a series of USB request blocks (URBs) that are then passed down to the USB bus driver (USBD.SYS). The URBs also use control codes to indicate what function is being requested. Custom I/O control codes can be used within the IRPs in order to implement special features, but the URBs are standard and are defined by the USB bus driver. Some URB functions that can be requested from USBD.SYS are listed as follows:

- GET_DESCRIPTOR_FROM_DEVICE. This function retrieves the device descriptor from a USB device.
- VENDOR_DEVICE. This function sends a vendor control request.
- BULK_OR_INTERRUPT_TRANSFER. This function performs a bulk or interrupt transfer.

4.4.6 DSP board with USB interface

The final phase of the project was to layout a new circuit board that contained the 56309 DSP and the D12 USB device. Again I was unable to complete it in time to be included within this thesis, but the design has essentially been verified by the success of the USB/DSP interface board.

I intend to continue using the host port of the DSP to communicate with the D12, as this will leave the DSP's high-speed memory expansion port free so that it can be used to interface with some high speed memory and external I/O devices. The JTAG/OnCE port will be brought out onto a connector so that a debugger can be used, and an EEPROM will also be included for boot strapping. No A-D or D-A converters will be included on the new DSP board, as the intention is to construct daughter boards with the appropriate devices to suit different applications.

4.4.7 Future modifications

One small modification that should be made before another visit to Antarctica is to provide power to the laptop computer so that it doesn't use its internal battery. This is because when operating from its own battery the laptop dims the screen, making it difficult to see in the bright Antarctic environment. The batteries used to power the system core electronics, could easily cope with the extra demand.

5.0 Antarctic sea ice research

The sea ice research was conducted in the Ross Island region of the Ross dependency, an area of Antarctica that is managed by New Zealand (figure 5.1). This area has a rich history of early exploration. In particular several groups used Ross Island as a base before setting out to reach the South pole.

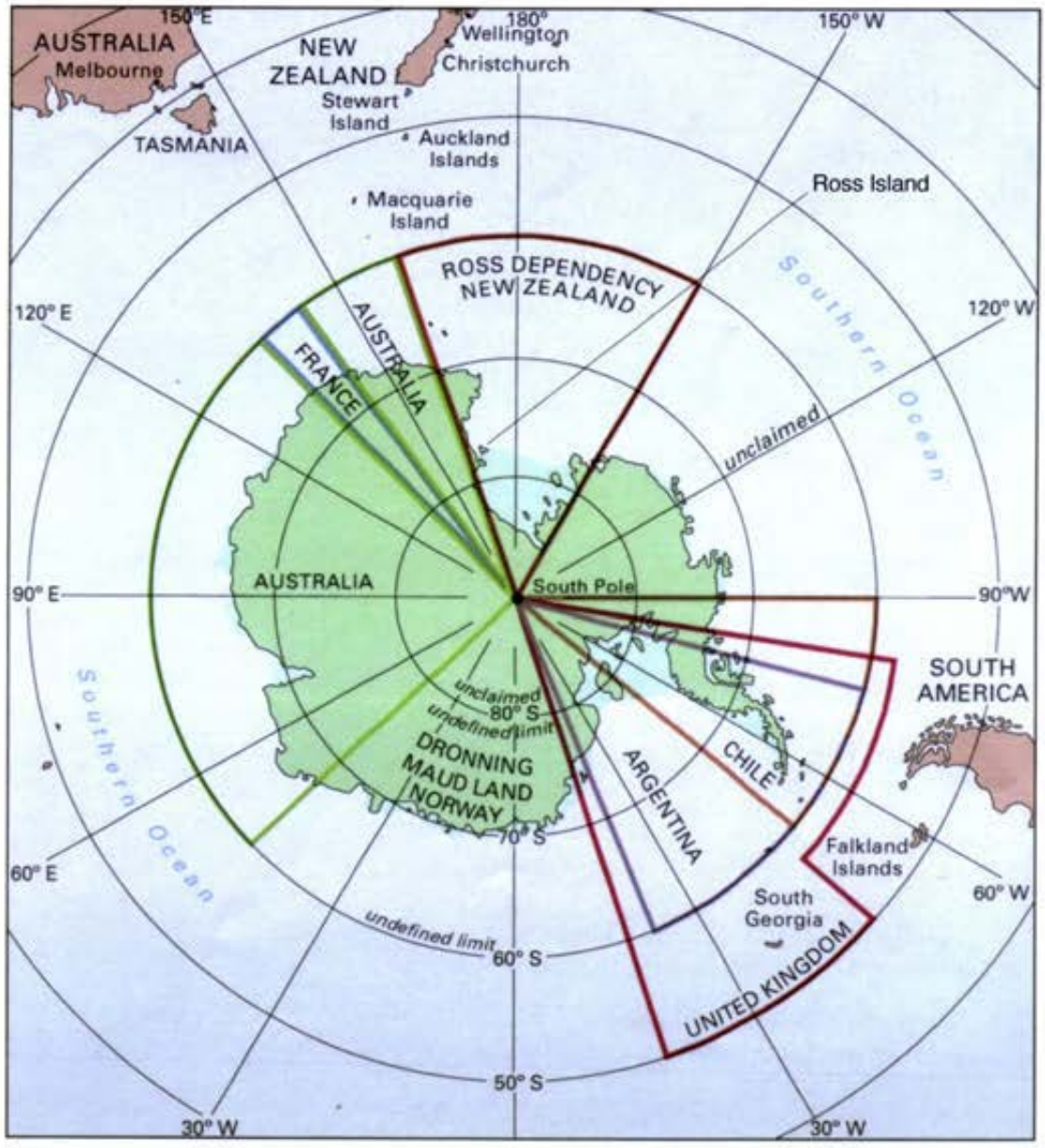


Fig 5.1 Territorial claims on the Antarctic continent [8].

New Zealand has a research facility on Ross Island, called Scott Base, which is located at Pram Point, at the end of the Hut Point Peninsula (figure 5.2). Near Scott Base is McMurdo Station, a base operated by the United States of America. Both bases are supported from Christchurch, where all the supplies and flights are coordinated.

Fig 5.2 The region of Ross Island where our research was undertaken [72].

Once at Scott Base we travelled out to our research area which was located on the sea ice in between Inaccessible and Big Razorback Islands and along the main flagged travelling route that ran almost in a direct line from Hutt Point to Cape Evans. The sea ice around our camp was approximately 2 m thick and suitable for performing experiments. The camp (figures 5.3 and 5.4) consisted of a number of shipping containers that had been previously refrigerated, but as they are well insulated, made ideal huts that were used as work rooms and living quarters. The polar tents shown in figure 5.3 were used for sleeping and a larger tent (not shown), located as far away as possible from the containers, was used to house our equipment. This was done to minimise perturbations in the Earth's magnetic field and reduce interference from the camp generator as much as possible.

Fig 5.3 Tents on the sea ice with Big Razorback Island in the background.

Fig 5.4 The main part of our camp and a little visitor.

In 1997 the old system with a new preamp and duplexor unit, new saddle coil probe and external gradient coils was used to collect sea ice diffusion data from around the base camp. As we had to use a chainsaw to set up the PGSE experiments, we could only get data for one depth. During this time we had a lot of trouble with interference and grounding caused by a dubious power connection and a generator operating nearby. In 1999, we initially used the old system and the modified probe incorporating the gradient coils to do a series of experiments at different depths around the base camp. The plan was to follow this with experiments using the new system further from the camp, but unfortunately this could not be done as we had to evacuate the camp site a week earlier than expected. However I did some preliminary experiments and performance checks with the new system at the camp and out in the field, as described in section 3.4. Indications were that the system worked satisfactory. In addition we did some tests using another alternative probe design that one of our team members was investigating [71]. The system described in this thesis was used to do these tests as we travelled away from camp in order to get the maximum possible signal to noise performance, but this exercise proved to be unsuccessful as the alternative probe had very low signal to noise performance, even with a water sample.

The main purpose for the 1999 visit was to collect more and better sea ice data in order to confirm our previous results. Each season the sea ice environment is slightly different so data must be collected over a number of seasons before a clear understanding can be obtained of the sea ice properties. Each time we go we gain experience and knowledge, which is then used in the on-going process of optimising the equipment and the experiments so that better data can be collected. The phenomena that we have observed in previous seasons and that was confirmed in the last visit (1999), is that when we perform diffusion experiments on sea ice, we obtain data that shows that there are multiple brine self diffusion rates. This is demonstrated in figure 5.5, where four successive diffusion experiments were performed on some sea ice using three different Δ times. †

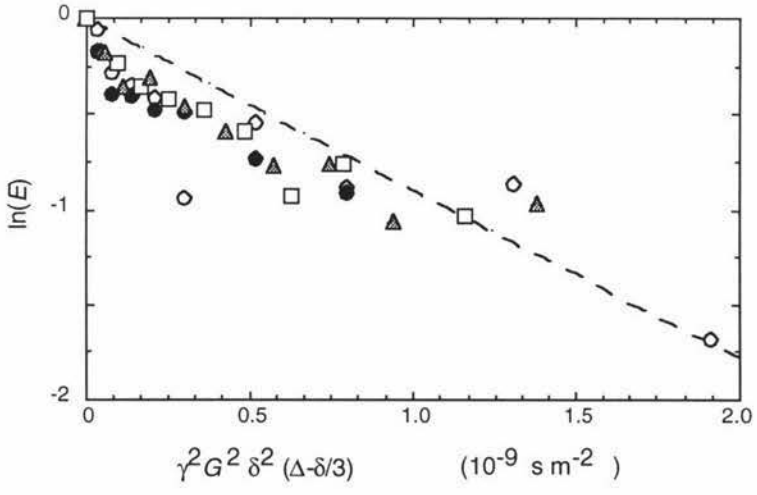


Fig 5.5 Diffusion data for sea ice [4], obtained using different Δ times: 175 ms (circles), 208 ms (squares) and 245 ms (triangles).

† This graph is reproduced from appendix F [4], which is a paper that discusses the sea ice aspects of our work in detail, and should be consulted if a greater understanding is required.

The dashed line in figure 5.5 is the expected echo attenuation for a water sample. The sea ice diffusion data exhibits bi-exponential behaviour and is very different from that of a water sample. Initially there is a rapid decay, then the slope of the data reduces significantly to a decay rate less than that for water. The different decay rates indicate that there are a number of different diffusion rate components, one faster and one slower than free water. The fast diffusion component we have attributed to a convection process occurring within the brine cavities, known as Rayleigh convection. Here a thermal gradient across the brine cavity causes the brine to circulate and results in the melting of the bottom of the cavity and the subsequent refreezing at the top. The brine is also more concentrated at the bottom of the cavity, which helps in the melting process. The net result is the enhanced transfer of heat from the bottom to the top, ie, the ocean to the atmosphere. The diffusion component slower than free water is most likely due to restricted diffusion of the brine within certain cavities. This occurs when the cavity dimensions are small enough to confine the water molecules and therefore restrict their movement. The PGSE experiment can therefore be used to get an indication of the pore sizes by varying the observation time Δ.

6.0 Conclusions

Information about Antarctic sea ice microstructure is very important as it enhances our understanding of its thermal, optical and mechanical properties and therefore better predictions can be made about the strength of the ice, when it will subsequently break up, and how the sea ice influences the global climate.

The NMR technique gives us a unique ability to non-invasively probe the microstructure of the sea ice in a way that would be very difficult with other techniques.

The initial NMR work performed in Antarctica used a system that was constructed out of a number of commercial units and some homebuilt electronics and a probe. This system suffered from a number of problems such as: it was bulky, it required considerable time to set up, it required mains power, it was not robust enough for the harsh conditions and therefore had reliability problems, the type of computer that could be used as the host was very limited, the solenoidal probe was very invasive and not robust.

The objectives of this thesis were to develop a new system and probe that addressed the problems of the old system and also took into account future work which could be undertaken. The new system and probe are portable and quick to set up on arrival to a test site. Flexibility has been achieved through the use of a DSP and a standard interface to a host computer. The new probe is much less invasive.

The new system worked satisfactorily in Antarctica but unfortunately little time was available to use it, however it will be used extensively on any future visits. We can also use the system in New Zealand to perform other experiments or use it for teaching purposes. The probe was used extensively during the last two Antarctic visits and proved to be robust and very easy and quick to set up. The data obtained with the new probe confirmed and enhanced our understanding of the sea ice structure and brine mobility, where diffusion data showed that there are two different main brine diffusion components. One component, slower than free water, is probably due to the restricted movement of the brine within very small cavities, while another component, faster than free water, is most likely due to convection processes occurring within the brine cavities.

The system core part of the new NMR system can and will be used for other applications and could have commercial potential as a general purpose control/acquisition system. Two applications that we have currently in mind are a scanning tunnelling microscope and a compression rheometer. For NMR applications some further work that could be undertaken is the development of a pulse program editor. This could be a graphical or a custom language editor that would not require the user to know anything about C or assembly language for the DSP. The editor would take the user input and compile executable code for the DSP. The possibility of linking the PC programmers interface (MyAPI) into other applications such as Matlab is something that could also be explored. Currently the system core has only been used for Earth's field NMR, but through the use of some digital communications

transceiver devices such as the AD6620, AD9856, available from Analog Devices [37], medium and high field NMR systems could be implemented.

The design and development of equipment for a harsh environment and a one shot testing period has been a considerable challenge. Flexibility, robustness, reliability and guaranteed performance are all needed to minimise the risk of not been able to collect any useful data during a brief Antarctic visit. As well as that, to develop the NMR system, knowledge and experience were required in a range of disciplines.

References

[1] P. T. Callaghan, *Nuclear Magnetism and Nuclear Magnetic Resonance – laboratory experiment* , (1980).

[2] P. T. Callaghan, M. LeGros, Am. J. Phys., **50(8)**, 709, (1982).

[3] P. T. Callaghan, C. D. Eccles, and J. D. Seymour, Rev. Sci. Instrum., **68**, 4263, (1997)

[4] P. T. Callaghan, R. Dykstra, C. D. Eccles, T. G. Haskell, and J. D. Seymour, Cold Regions Science and Technology, **29**, 153, (1999)

[5] P. T. Callaghan, *Principals of Nuclear Magnetic Resonance Microscopy*, (1993)

[6] P. T. Callaghan, and C. D. Eccles, Bulletin of Magnetic Resonance, **18**, p62-64, (1996)

[7] P. T. Callaghan, C. D. Eccles, T. G. Haskell, P. J. Langhorne, and J. Seymour, Journal of Magnetic Resonance, **133**, 148, (1998)

[8] International Antarctic Centre, Christchurch, New Zealand.

[9] R. Dykstra and C. D. Eccles, Proceedings of the Fourth Electronics New Zealand Conference, 46, (1997).

[10] C. D. Eccles, C. J. Clark, S. L. Codd and R. Dykstra, Proceedings of the Fifth Electronics New Zealand Conference, 45, (1998).

[11] C. D. Eccles, *Introduction to NMR imaging – Lecture notes*, (1999).

[12] P. Edwards, Massey University – Chemistry, Personal communication.

[13] P. G. Morris, *Nuclear Magnetic Resonance Imaging in Medicine and Biology*, (1986).

[14] B. F. Melton and V. L. Pollak, Journal of Magnetic Resonance, A-**122**, 42, (1996).

[15] J. Stepisnik, M. Kos, G. Planinsic and V. Erzen, Journal of Magnetic Resonance, A-**107**, 167, (1994).

[16] A. Mohoric, J. Stepisnik, M. Kos, and G. Planinsic, Journal of Magnetic Resonance, **136**, 22, (1999).

[17] G. Planinsic, J. Stepisnik and M. Kos, Journal of Magnetic Resonance, A-**110**, 170, (1994).

[18] D. I. Hoult, Concepts in Magnetic Resonance, **12(4)**, 173, (2000)

[19] D. I. Hoult, and R. E. Richards, Journal of Magnetic Resonance, **24**, 71, (1976)

[20] D. I. Hoult, Progress in NMR Spectroscopy, **12**, 41, (1978)

[21] D. M. Ginsberg, and M. J. Melchner, Review of Scientific Instruments, **41**, p122, (1970)

[22] R. R. Ernst, Advances in Magnetic Resonance, **2**, 121, (1966)

[23] PROSPA, Data analysis software developed by C. Eccles, Massey University, (1999).

[24] C. D. Eccles, *Command Syntax for PC version of Prospa* (1999).

[25] Matlab, a Numeric Computation and Visualisation Software package, The Math Works Incorporated, (1994)

[26] PMI, Analog integrated circuits data book, volume 10, 5-155, (1990).

[27] Crystal Semiconductor Corporation, CS4215 data sheet, (1993).

[28] Motorola, MJL3281A data sheet, (1998).

[29] Atmel, AT29LV512 data sheet, (1999).

[30] Motorola, *DSP56303EVM user's manual*, rev 2.

[31] Tecmag Inc, 6006 Bellair Blvd, Houston, TX 77081, USA.

[32] www.kepcopower.com

[33] www.apple.com.

[34] global.acer.com.

[35] www.microsoft.com.

[36] www.metroworks.com.

[37] www.analog.com/comms.

[38] www.protel.com

[39] R. Kleinberg, Encyclopedia of Nuclear Magnetic Resonance, 4960, (1995).

[40] National Semiconductor, LM258 data sheet.

[41] www.sonnenshein.at.

[42] Fraunhofer Institut, *NMR-INSPECT*, (1999).

[43] P. T. Callaghan, R. Dykstra, C. D. Eccles, S. A. Furkert, E. D. Minot, *Use of non-uniform gradients to measure multi-component diffusion in Antarctic Sea Ice*, (still to be published).

[44] Keithley, *Using DriverLINX with Your Hardware.*

[45] Keithley, *DriverLINX Digital I/O Programming guide.*

[46] Keithley, *KPCMCIA-PIO24 User's Manual.*

[47] Keithley, *DriverLINX Technical Reference manual.*

[48] P. M. Embree, *C Algorithms for real-time DSP*, (1995).

[49] S. K. Mitra, *Digital signal processing*, (1998).

[50] Motorola, *DSP56300 24-BIT DIGITAL SIGNAL PROCESSOR FAMILY MANUAL.*

[51] Motorola, *DSP56303 User's Manual.*

[52] Motorola, *Suite56 DSP Tools User's Manual*, Release 6.3.

[53] Motorola, *MOTOROLA DSP LIMKER/LIBRARIAN REFERENCE MANUAL.*

[54] Motorola, *DSP563CCC MOTOROLA DSP56300 FAMILY OPTIMIZING C COMPILER USER'S MANUAL.*

[55] Motorola, DSP56303 semiconductor data sheet.

[56] Motorola, DSP56309 semiconductor data sheet.

[57] Motorola, *DSP56309 User's Manual.*

[58] Compaq, Intel, Microsoft, NEC, *Universal Serial Bus Specification*, Revision 1.1, (1998).

[59] Philips, PDIUSBD12 integrated circuits data sheet, (1998).

[60] Philips, *D12 SMART USER'S MANUAL.*

[61] Philips, *Firmware Programming Guide for PDIUSBD12*, Version 1.

[62] Philips, PDIUSBD12 integrated circuits data sheet, (1998).

[63] J. Hyde, *USB Design by Example*, (1999).

[64] J. Axelson, *USB COMPLETE,* (1999).

[65] W. Oney, *Programming the Microsoft Windows Driver Model*, (1999).

[66] Coff2ppl, a file conversion utility written by P. Ryland, Massey University.

[67] K Runtz, *Multirate signal processing*, (1999).

[68] Motorola, *DSP ASSEMBLER REFERENCE MANUAL.*

[69] www.USBorg.com.

[70] Microsoft, *Windows98 Driver Development Kit.*

[71] S. A. Furkert, *A Nuclear Magnetic Resonance investigation of brine inclusions in Antarctic and artificial sea ice,* (2000).

[72] United States Geological Survey, *Ross Island Antarctica*, (1970).

Appendix A

An assembler source listing of the DSP operating system software used with the PCMCIA/DSP interface.

mprog1.asm

```
;=====================================================================
;
;       Mprog1.ASM  Ver.1.0
;       Robin Dykstra
;       10/8/99
;
;=====================================================================
;
;       The DSP communicates with the laptop via port B
;
;       The lower 8 bits (b0-b7) of port B are used for transfer of data
;       between the laptop and DSP. Therefore these 8 bits are bidirectional
;       and are configured as input or output when required. As default they
;       are set as inputs.
;
;       The next 4 bits  (b8-b11) are used as inputs to read the command
;       requested from the laptop.
;
;       The next bit (b12) is not used, but is set up as an output.
;
;       The next bit (b13)is an output and is used to tell the laptop
;       when the pulse program has finished.
;
;       The next bit (b14)is an output and is used to enable the "read"
;       tri-state buffer. It should be brought low when a read operation
;       from the laptop is required. note: the lower 8 bits of port B should
;       be configured as inputs before this pin is driven low.
;
;       The last bit (b15) is an output and is used to clock data in and
;       out to/from the laptop. The laptop looks for the rising edge.
;
;=====================================================================
;
        nolist
        include  'ioequ.asm'
        include  'intequ.asm'
        include  'ada_equ.asm'
        include  'mvectors.asm'
        list
;
;=====================================================================
;
;       The following EQUates will define the operatonal parameters
;       of the codec.  Please refer to the ADA_EQU.ASM source file
;       for a description of the parameters selections available.  The
;       variables defined by the EQUates are sent to the  codec via
;       the transmit buffer, TX_BUFF.
;
;       Since the parameters are defined in ADA_EQU.ASM, these lines must
;       follow the include statement.
;=====================================================================
;
CTRL_WD_12      equ     NO_PREAMP+HI_PASS_FILT+SAMP_RATE_48+STEREO+DATA_16
;CLB=0
CTRL_WD_34      equ     IMMED_3STATE+XTAL1_SELECT+BITS_64+CODEC_MASTER
CTRL_WD_56      equ     $000000
CTRL_WD_78      equ     $000000

TONE_OUTPUT     equ     HEADPHONE_EN+LINEOUT_EN+(0*LEFT_ATTN)+(0*RIGHT_ATTN)
```

```
TONE_INPUT      equ     MIC_IN_SELECT+(15*MONITOR_ATTN)

   ;---Buffer for talking to the CS4215

        org     x:0
RX_BUFF_BASE    equ     *
RX_data_1_2     ds      1       ;data time slot 1/2 for RX ISR
RX_data_3_4     ds      1       ;data time slot 3/4 for RX ISR
RX_data_5_6     ds      1       ;data time slot 5/6 for RX ISR
RX_data_7_8     ds      1       ;data time slot 7/8 for RX ISR

TX_BUFF_BASE    equ     *
TX_data_1_2     ds      1       ;data time slot 1/2 for TX ISR
TX_data_3_4     ds      1       ;data time slot 3/4 for TX ISR
TX_data_5_6     ds      1       ;data time slot 5/6 for TX ISR
TX_data_7_8     ds      1       ;data time slot 7/8 for TX ISR

RX_PTR          ds      1       ;Pointer for rx buffer
TX_PTR          ds      1       ;Pointer for tx buffer

;       Parameter table and various variables

        org     y:0
PT_BASE         equ     *
CMD_REQF        ds      1       ;Command request flag
BLK_PTR         ds      1       ;Pointer to block in data buffer

D_BUFFA         equ     $011000 ;Data buffer start address
PP_LOADA        equ     $000600 ;Pulse program load address

;       Main program

        org P:$100
START
main
        movep   #$A40033,x:M_PCTL       ;PLL = 52/11 x 16.9344Mhz = 80.05MHz
        movep   #$010921,x:M_AAR0       ;Set up SRAM address y:010000
        movep   #$012421,x:M_BCR        ;Set up wait states
        movec   #0,SP                   ;clear stack pointer
        move    #0,SC                   ;clear stack counter
        ori     #3,mr                   ;disable interrupts
        movep   #$11,x:M_TCSR0          ;turn LED off
        move    #$200,r6                ;initialize stack pointer
        move    #-1,m6                  ;linear addressing
        move    #-1,m5
        move    #-1,m4
        move    #-1,m3
        move    #-1,m2
        move    #-1,m1
        move    #-1,m0
        move    #-1,m7
        move    #RX_BUFF_BASE,x0
        move    x0,x:RX_PTR             ;Initialize the rx pointer
        move    #TX_BUFF_BASE,x0
        move    x0,x:TX_PTR             ;Initialize the tx pointer
        movep   #$01,x:M_HPCR           ;Set up portB
        movep   #$F000,x:M_HDDR         ;Upper 4 bits as outputs
        movep   #$4000,x:M_HDR          ;Deselect "read" tri-state buffer
        movep   #$00,x:M_PCRD           ;Set up portD
        movep   #$3F,x:M_PRRD
        movep   #$00,x:M_PDRD
        movep   #$00,x:M_PCRE           ;Set up portE
        movep   #$07,x:M_PRRE
        movep   #$00,x:M_PDRE
        move    #$000000,r1
        move    r1,y:CMD_REQF           ;Clear the command request flag
        move    #D_BUFFA,r1
        move    r1,y:BLK_PTR            ;Initialise current block pointer
        move    #$00000C,r1
```

```
        move    r1,p:PP_LOADA           ;Put "RTS" at the start of pulse prog
mem
        movep   #$06,x:M_IPRC           ;Set up Interrupt IRQA, neg edge
trig, priority 2
;
;
; --- INIT THE CODEC ---
        jsr     ada_init                ;Jump to initialize the code
;
        clr     a
        move    a,x:TX_BUFF_BASE        ;transmit left value
        move    a,x:TX_BUFF_BASE+1      ;transmit right value
        move    #TONE_OUTPUT,y0         ;set up control words
        move    y0,x:TX_BUFF_BASE+2
        move    #TONE_INPUT,y0
        move    y0,x:TX_BUFF_BASE+3
;
;=========================================================================
;
;       Main program loop
;       Waits for a command from the Laptop, then executes it.
;
loop_1  jclr    #0,y:CMD_REQF,*         ;Wait for command request ?
        dor     #$FFF,dly_1             ;Wait for port to settle
        nop
        nop
dly_1   clr     a
        movep   x:M_HDR,a1              ;Read command
        and     #$000F00,a              ;Mask out unwanted bits
        cmp     #$000100,a              ;Command 1?
        beq     cmd_1
        cmp     #$000200,a              ;Command 2?
        beq     cmd_2
        cmp     #$000300,a              ;Command 3?
        beq     cmd_3
        cmp     #$000400,a              ;Command 4?
        beq     cmd_4
        cmp     #$000500,a              ;Command 5?
        beq     cmd_5
        cmp     #$000600,a              ;Command 6?
        beq     cmd_6
        cmp     #$000700,a              ;Command 7?
        beq     cmd_7
        bra     cmd_fin
cmd_1   bsr     cmd_1r                  ;Handle command 1
        bra     cmd_fin
cmd_2   bsr     cmd_2r                  ;Handle command 2
        bra     cmd_fin
cmd_3   bsr     cmd_3r                  ;Handle command 3
        bra     cmd_fin
cmd_4   bsr     cmd_4r                  ;Handle command 4
        bra     cmd_fin
cmd_5   bsr     cmd_5r                  ;Handle command 5
        bra     cmd_fin
cmd_6   bsr     cmd_6r                  ;Handle command 6
        bra     cmd_fin
cmd_7   bsr     cmd_7r                  ;Handle command 7
cmd_fin movep   #$11,x:M_TCSR0          ;turn LED off
        movep   #$00,x:M_PDRE           ;Set port bits low
        move    #$00,r1
        move    r1,y:CMD_REQF           ;Clear the command request flag
        bra     loop_1                  ;Loop back.
;
;=========================================================================
;       Command 1 routine
;       This routine clears the acknowledge flag if set.
;
cmd_1r  movep   #$4000,x:M_HDR          ;Set PB13 to low
        rts
```

```
;
;=============================================================================
;       Command 2 routine
;       spare
cmd_2r  movep   #$0c,x:M_PDRD
        rts
;
;=============================================================================
;       Command 3 routine
;       This routine reads a block of data from the laptop
;       and stores it into program memory.
;       The first 3 bytes of the data block specify what address
;       to start loading the data into.
;       The next 3 bytes specify the number of 24bit words to transfer.
;       The data then follows.
;
cmd_3r  bsr     wait10m         ;Wait for laptop to get ready to transmit
        bsr     get3b           ;Load the address
        move    r3,r4           ;Put into r4
        bsr     get3b           ;Load the number of words
        move    r3,r5           ;Put into r5
cmd_31  move    r5,b            ;All the data loaded ?
        cmp     #$00,b
        beq     cmd_3e          ;Branch if so otherwise loop again
        bsr     get3b           ;Get another word
        move    r3,p:(r4)+      ;Store into memory and increment pointer
        move    (r5)-           ;Decrement word count
        bra     cmd_31          ;Loop again
cmd_3e  rts                     ;Done
;
;=============================================================================
;       Command 4 routine
;       This is the upload data to laptop routine.
;       It sends the 1024 16bit block of data to the laptop that the
;       current block pointer is pointing at.
;
cmd_4r  bsr     wait10m         ;Wait for laptop to get ready to receive
        move    y:BLK_PTR,r5    ;Make r5 point to the start of the block
        move    #$00,r4
cmd_41  move    r4,b            ;All the data sent out?
        cmp     #$00400,b
        beq     cmd_4e          ;Branch if so
        move    y:(r5)+,r3      ;Read word from memory and increment pointer
        move    (r4)+           ;Increment count
        bsr     out2b           ;Output the word to laptop
        bra     cmd_41          ;Loop again
cmd_4e  rts
;
;=============================================================================
;       Command 5 routine
;       This is the run pulse program routine.
;       At the end it sets an acknowledge flag for the laptop and
;       sets the block pointer to the start of the buffer.
;
cmd_5r  movep   #$4000,x:M_HDR          ;Clear flag if set
        jsr     PP_LOADA                ;Run the user pulse program
cmd_5e  move    #D_BUFFA,r1
        move    r1,y:BLK_PTR            ;Set block pointer to start of buffer
        movep   #$6000,x:M_HDR          ;Set PB13 high to indicate end of pp
        rts
;
;=============================================================================
;       Command 6 routine
;       This routine sets the current block pointer so that it points to
;       the start of buffer memory. That is it sets it to y:10000.
;
cmd_6r  move    #D_BUFFA,r1
        move    r1,y:BLK_PTR            ;Set block pointer to start of buffer
cmd_6e  rts
```

```
;
;=============================================================================
;       Command 7 routine
;       This routine increments the current block pointer
;       so that it points to the next 1K block.
;
cmd_7r  move    y:BLK_PTR,a
        add     #$400,a
        move    a,y:BLK_PTR
cmd_7e  rts
;
;=============================================================================
;       Output 2 bytes to laptop
;       The 16bit word to be sent is in the upper part of r3
;
out2b   move    r3,a1           ;Output MSB first
        lsr     #16,a1
        bsr     outbyt
        move    r3,a1           ;Output next byte
        lsr     #8,a1
        bsr     outbyt
        rts
;
;=============================================================================
;       Output 3 bytes to laptop
;       The 24bit word to be sent is in r3
;
out3b   move    r3,a1           ;Output MSB first
        lsr     #16,a1
        bsr     outbyt
        move    r3,a1           ;Output next byte
        lsr     #8,a1
        bsr     outbyt
        move    r3,a1           ;Output last byte
        bsr     outbyt
        rts
;
;=============================================================================
;       Input 3 bytes from laptop and format as a 24bit word in r3
;
get3b   clr     b
        clr     a
        bsr     inbyt           ;read MSB from laptop
        and     #$0000FF,a      ;mask out unwanted bits
        lsl     #16,a           ;move up
        move    a1,b1           ;and put into b
        bsr     inbyt           ;read next byte from laptop
        and     #$0000FF,a      ;mask out unwanted bits
        lsl     #8,a            ;move up
        move    a1,x0
        or      x0,b            ;and or into b
        bsr     inbyt           ;read last byte from laptop
        and     #$0000FF,a      ;mask out unwanted bits
        move    a1,x0
        or      x0,b            ;and or into b
        move    b1,r3           ;put result into r3
        rts
;
;=============================================================================
;       Input byte from laptop routine
;       using a clock pulse to get data from laptop.
;       The byte read is placed into the lower part of a1
;
;       Need to send out a rising edge clock to initiate transfer of data
;       from the laptop. Wait for 25uS and then read the data.
;
inbyt   movep   #$8000,x:M_HDR  ;Output clock edge and set up for read
        bsr     wait25u         ;Wait for laptop to output data
        bsr     wait25u
```

```
        movep   x:M_HDR,a1      ;Input byte
        movep   #$4000,x:M_HDR  ;Set clock level low and disable buffer
        bsr     wait25u
        rts
;
;=============================================================================
;       Output byte to laptop routine
;       using a clock pulse to load into laptop.
;       The byte to be sent should be placed into a1.
;
;       Need to firstly set port as outputs and then output data to
;       port with a rising edge clock and then wait 25uS for
;       the laptop to read the data.
;
outbyt  movep   #$4000,x:M_HDR  ;Disable input buffer
        movep   #$F0FF,x:M_HDDR ;Set lower bits as outputs
        and     #$0000FF,a      ;format for port
        or      #$00c000,a
        movep   a1,x:M_HDR      ;Output data to port with clock high
        bsr     wait25u         ;Wait
        and     #$0040FF,a      ;Set clock level low
        movep   a1,x:M_HDR
        bsr     wait25u
        movep   #$F000,x:M_HDDR ;Set lower bits as inputs
        rts
;
;=============================================================================
;       Wait 10mS routine
;
wait10m dor     #400,w10e
        bsr     wait25u
w10e    rts
;
;=============================================================================
;       Wait 25uS routine (80MHz clock)
;
wait25u move    #2000,x0
        rep     x0
        nop
        nop
        rts
;
;=============================================================================
;       IRQA Interrupt service routine
;       This interrupt is used by the laptop to indicate a request
;       The routine basically sets a flag to tell the main program
;       loop that the laptop has a command waiting at the command
;       port.
;
cmdint  move    r1,x:(r6)+              ; Save R1 to the stack.
        move    #$000001,r1
        move    r1,y:CMD_REQF           ; Set the command request flag
        movep   #$2800,x:M_TCSR0        ; TURN ON LED
        move    x:-(r6),r1              ; Restore R1.
        rti
;
;=============================================================================
;
      include         'ada_init.asm'  ; load the code init routines
;
;=============================================================================
```

Appendix B

C source listings of the "MyAPI" layer running on the laptop.

nmrdsp.cpp

```
#include "drvlinx.h"
#include "dlcodes.h"
#include "include.h"


int InitialiseNMRDSP(short mode, char arg[]);
int ResetNMRDSP(short mode, char arg[]);
int DownloadPPNMRDSP(short mode, char arg[]);
int DownloadDataNMRDSP(short mode, char arg[]);
int UploadNMRDSP(short mode, char arg[]);
int RunNMRDSP(short mode, char arg[]);
int SetNMRParameter(short mode, char arg[]);

void UnpackCharData(char *cdata, float *fdata, long length);

extern WinData rootWin;
extern UINT InitDevice (LPServiceRequest pSR, HWND win, UINT Device);
extern UINT OutData(LPServiceRequest dSR, HWND dwin, LPVOID dBuffer, DWORD
dLength);
extern UINT OutCmd(LPServiceRequest cSR, HWND cwin, DWORD dspcmd);
extern UINT InData(LPServiceRequest dSR, HWND dwin, LPVOID dBuffer, DWORD
dLength);
extern UINT   Polldsp(LPServiceRequest dSR, HWND dwin);

DL_SERVICEREQUEST sr;

//*************************************************************************
// Initialise DSP laptop driverlinx interface
//*************************************************************************

int
InitialiseNMRDSP(short mode, char arg[])
{
   HWND hWnd;
   WinData *win;
   HINSTANCE DLLAPI dLinx;
       UINT dLinxres;

   win = rootWin.FindWinByNr(1);
   hWnd = win->hwnd;
       dLinx = OpenDriverLINX(hWnd,"KPCPIO.DLL");
       dLinxres = InitDevice ((LPServiceRequest)(&sr),hWnd,(UINT)0);

   return(0);
}

//*************************************************************************
// Reset DSP
//*************************************************************************

int
ResetNMRDSP(short mode, char arg[])
{
   TextMessage("\rReset DSP\r");
   return(0);
}

//*************************************************************************
```

```
// Download pulse program to DSP ("filename")
//*********************************************************************

int
DownloadPPNMRDSP(short mode, char arg[])
{
   short r;
   static char fileName[200];
  // static char addressStr[50];
   FILE *fp;
   char *buffer;
       int iotimeout;

   extern int GetFileLength(FILE *fp);

   if((r = ArgScan(mode,arg,1,"filename","e","s",fileName)) < 0)
         return(r);

// Open the pulse-program file

   if((fp = fopen(fileName,"r")) == (FILE*)0)
   {
      ErrorMessage("file '%s' not found",fileName);
      return(-1);
   }

// Get file length and allocate space in buffer to contain file

   long length = GetFileLength(fp);

       if((buffer = new char[length+1]) == (char*)0)
       {
          ErrorMessage("out of memory in downloadpp");
          return(-1);
       }

// Read file into buffer and close file

       fread(buffer,1,length,fp);
       fclose(fp);

// Get parent window for error messages

   WinData *win = rootWin.FindWinByNr(1);
   HWND hWnd = win->hwnd;

// Send buffer to DSP            (if error try again upto five times)

       iotimeout = 5;
       r = 1;

       while( (r != 0) && (iotimeout != 0))
       {
       r = OutCmd((LPServiceRequest)&sr,hWnd,3);
   r = OutData((LPServiceRequest)&sr,hWnd,buffer,length);
       iotimeout = iotimeout - 1;
       }

   if(iotimeout == 0)            // major I/O error
             {
             delete[] buffer;
             ErrorMessage("I/O download failure");
             return(-1);
             }

   delete[] buffer;

   return(r);
```

```
}


//*************************************************************************
// Download data to DSP ("data",srcBuffer,size)
//*************************************************************************

int
DownloadDataNMRDSP(short mode, char arg[])
{
   short r;

   r = CountArgs(arg);

//   if(r == 3)
//   {
//      if((r = ArgScan(mode,arg,1,"mode filename
address","cee","sss",varName,title,range)) < 0)
//         return(r);


       // Get parent window for error messages

   //WinData *win = rootWin.FindWinByNr(1);
   //HWND hWnd = win->hwnd;

       // Send buffer to DSP

   //r = OutCmd((LPServiceRequest)&sr,hWnd,3);
   //if(r != 0) { delete[] buffer; return(-1);}
   //r = OutData((LPServiceRequest)&sr,hWnd,buffer,length);
   //if(r != 0) { delete[] buffer; return(-1);}

   return(0);
}


//*************************************************************************
// Upload data from DSP ("data",dstBuffer, size)
//*************************************************************************

int
UploadNMRDSP(short mode, char arg[])
{
   short r;
   static long length, blength = 1024;
   static char bufferName[50];
   short type;
       int iotimeout;
   char *data;
   float *fdata;
       Variable *var;
       static char flip[2];
       short *idata;
       long nblocks, bcount = 0;

   if((r = ArgScan(mode,arg,2,"buffername
size","ce","sd",bufferName,&length)) < 0)
         return(r);

// Get passed buffer

   var = GetVariable(bufferName,type);

   if(type == VECTOR)
   {
      if(var->dimx != length)
      {
```

```
            ErrorMessage("buffer length mismatch",bufferName);
            return(-1);
         }
      }
      else
      {
         ErrorMessage("invalid buffer '%s'",bufferName);
         return(-1);
      }

// Calculate number of 1024 word blocks

      nblocks = roundtol(length/blength);

// Allocate temporary buffer data

      if((data = new char[2*blength]) == (char*)0)
      {
         ErrorMessage("out of memory in rundsp");
         return(-1);
      }

// Get parent window for error messages

   WinData *win = rootWin.FindWinByNr(1);
   HWND hWnd = win->hwnd;


// Get data from DSP

      r = OutCmd((LPServiceRequest)&sr,hWnd,6); //Set DSP block pointer to
start

      while( nblocks != 0 )
            {

            // Get data block from DSP        (if error try again upto five
times)
            iotimeout = 5;
            r = 1;
            while( (r != 0) && (iotimeout != 0))
                  {
                  r = OutCmd((LPServiceRequest)&sr,hWnd,4);
            r = InData((LPServiceRequest)&sr,hWnd,data,2*blength);
                  iotimeout = iotimeout - 1;
                  }

            if(iotimeout == 0)                  // major I/O error
                  {
                  delete[] data;
                  ErrorMessage("I/O upload failure");
                  return(-1);
                  }

            // Copy char data to buffer
            idata = (short *)flip;
            fdata = (float *)var->data;
            for(long i = 0,j = 0; i < blength; i++,j+=2)
            {
                  flip[0] = data[j+1];
                  flip[1] = data[j];
                  fdata[bcount++] = idata[0];
            }

            r = OutCmd((LPServiceRequest)&sr,hWnd,7); //Increment DSP block
pointer

            nblocks = nblocks - 1;
```

```
            }

   delete[] data;

   return(0);
}


//************************************************************************
// Take the 16 bit data from the DSP and convert to 32 bit float
//************************************************************************

void
UnpackCharData(char *cdata, float *fdata, long length)
{
       char flip[2];
       short *idata;

       idata = (short *)flip;
       for(long i = 0,j = 0; i < length; i++,j+=2)
   {
              flip[0] = cdata[j+1];
              flip[1] = cdata[j];
              fdata[i] = idata[0];
   }
}


//************************************************************************
// Run pulse program
//************************************************************************

int
RunNMRDSP(short mode, char arg[])
{
   short r;

       int pptimeout;

       // Get parent window for error messages

   WinData *win = rootWin.FindWinByNr(1);
   HWND hWnd = win->hwnd;

       // Run pulse program

       pptimeout = 1000;                  //max allowable wait is 10 seconds

   r = OutCmd((LPServiceRequest)&sr,hWnd,5);
   if(r) return(-1);

       Sleep(10);           //Wait for pp to start executing

       while( !Polldsp((LPServiceRequest)&sr,hWnd) && (pptimeout != 0))
       // wait for DSP
       {
       Sleep(10);
       pptimeout = pptimeout - 1;
       }

       if (pptimeout == 0)                // time out error
              {
              ErrorMessage("I/O runpp failure");
              return(-1);
              }

   return(0);
}
```

```
//*************************************************************************
// Send an NMR parameter to the DSP
//*************************************************************************

int
SetNMRParameter(short mode, char arg[])
{
   short r;
   char parameterName[200];
   long parameterValue;

// Extract parameter name and value

   if((r = ArgScan(mode,arg,2,"parameter
value","ce","sd",parameterName,&parameterValue)) < 0)
      return(r);

// Process data

   if(!strcmp(parameterName,"numpoints"))
   {

   }
   else if(!strcmp(parameterName,"afperiod"))
   {

   }
   else if(!strcmp(parameterName,"afpulse"))
   {

   }
   else if(!strcmp(parameterName,"dwelltime"))
   {

   }
   else
   {
      ErrorMessage("invalid NMR parameter");
      return(-1);
   }

   return(0);
}
```

hioprog.cpp

```
/* Pulse programmer controller high level io routines */
/* Robin Dykstra 3/8/99                               */

#include      "drvlinx.h"
#include      "dlcodes.h"

UINT OutCmd(LPServiceRequest dSR, HWND dwin, DWORD dspcmd);
UINT OutData(LPServiceRequest dSR, HWND dwin, LPVOID dBuffer, DWORD dLength);
UINT InData(LPServiceRequest dSR, HWND dwin, LPVOID dBuffer, DWORD dLength);
UINT Polldsp(LPServiceRequest dSR, HWND dwin);

extern UINT InitDevice (LPServiceRequest pSR, HWND win, UINT Device);
extern DWORD Digin (LPServiceRequest pSR, HWND win, UINT Device, UINT
Channel);
extern UINT Diginmi (LPServiceRequest pSR, HWND win, UINT Device, UINT
Channel, LPVOID Buffer, DWORD Length);
extern UINT Diginme (LPServiceRequest pSR, HWND win, UINT Device, UINT
Channel, LPVOID Buffer, DWORD Length);
extern UINT Digout (LPServiceRequest pSR, HWND win, UINT Device, UINT
Channel, DWORD Value);
extern UINT Digoutmi (LPServiceRequest pSR, HWND win, UINT Device, UINT
Channel, LPVOID Buffer, DWORD Length);
extern UINT Digoutme (LPServiceRequest pSR, HWND win, UINT Device, UINT
Channel, LPVOID Buffer, DWORD Length);


//*********************************************************
// Output command to DSP routine
//*********************************************************

UINT OutCmd(LPServiceRequest dSR, HWND dwin, DWORD dspcmd)
{
       UINT cresult;
       UINT coutChan = 2;
       DWORD cmdpi;
       DWORD cmdni;

       cmdni = dspcmd;
       cmdpi = dspcmd + 64;        // set bit to interrupt DSP

       cresult = Digout (dSR, dwin, 0, 0, 0);   // set portA to 0

       cresult = Digout (dSR, dwin, 0, coutChan, cmdni);

       cresult = Digout (dSR, dwin, 0, coutChan, cmdpi);

       cresult = Digout (dSR, dwin, 0, coutChan, cmdni);

       return cresult;
}


//*********************************************************
// Output data to DSP routine
//*********************************************************

UINT OutData(LPServiceRequest dSR, HWND dwin, LPVOID dBuffer, DWORD dLength)
{
       UINT dresult;
       UINT doutChan = 0;

       dresult = Digoutme (dSR, dwin, 0, doutChan, dBuffer, dLength);

       return dresult;
}
```

```
//*********************************************************
// Input data from DSP routine
//*********************************************************

UINT InData(LPServiceRequest dSR, HWND dwin, LPVOID dBuffer, DWORD dLength)
{
       UINT dresult;
       UINT dinChan = 1;

       dresult = Diginme (dSR, dwin, 0, dinChan, dBuffer, dLength);

       return dresult;
}

//*********************************************************
// POll DSP routine. See if pulse program has finished
//*********************************************************

UINT Polldsp(LPServiceRequest dSR, HWND dwin)
{
       UINT pvalue;

       pvalue = Digin (dSR, dwin, 0, 2); // read port
       if(pvalue&0x20) return 1;                    //return 1 if flag set

       return 0;
}
```

ioprog.cpp

```
/* Pulse programmer controller low level io routines */
/* Robin Dykstra 1/8/99                              */

#include "include.h"
#include      "drvlinx.h"
#include      "dlcodes.h"

UINT InitDevice (LPServiceRequest pSR, HWND win, UINT Device);
DWORD Digin (LPServiceRequest pSR, HWND win, UINT Device, UINT Channel);
UINT Diginmi (LPServiceRequest pSR, HWND win, UINT Device, UINT Channel,
LPVOID Buffer, DWORD Length);
UINT Diginme (LPServiceRequest pSR, HWND win, UINT Device, UINT Channel,
LPVOID Buffer, DWORD Length);
UINT Digout (LPServiceRequest pSR, HWND win, UINT Device, UINT Channel, DWORD
Value);
UINT Digoutmi (LPServiceRequest pSR, HWND win, UINT Device, UINT Channel,
LPVOID Buffer, DWORD Length);
UINT Digoutme (LPServiceRequest pSR, HWND win, UINT Device, UINT Channel,
LPVOID Buffer, DWORD Length);


//*********************************************************
// Hardware initialization routine
//*********************************************************

UINT InitDevice (LPServiceRequest pSR, HWND win, UINT Device)
{
       UINT result;

       memset(pSR, 0, sizeof(DL_SERVICEREQUEST));
       DL_SetServiceRequestSize(*pSR);

       pSR->device = Device;
       pSR->hWnd = win;
       pSR->subsystem = DEVICE;
       pSR->mode = OTHER;
       pSR->operation = INITIALIZE;

       result = NoErr;
       if (DriverLINX(pSR) != NoErr) {
              result = pSR->result;
                     // Display error message
       pSR->operation = MESSAGEBOX;
       DriverLINX(pSR);
       }

       return 0;
}

//*********************************************************
// Hardware digital input routine, single value
//*********************************************************

DWORD Digin (LPServiceRequest pSR, HWND win, UINT Device, UINT Channel)
{
       UINT result;

       memset(pSR, 0, sizeof(DL_SERVICEREQUEST));
       DL_SetServiceRequestSize(*pSR);

       pSR->device = Device;
       pSR->hWnd = win;
       pSR->subsystem = DI;
       pSR->mode = POLLED;
       pSR->operation = START;
```

```
	pSR->channels.nChannels = 1;
	pSR->channels.chanGain[0].channel = Channel;

	result = NoErr;
	if (DriverLINX(pSR) != NoErr) {
		result = pSR->result;
			// Display error message
	pSR->operation = MESSAGEBOX;
	DriverLINX(pSR);
	return (DWORD) -1;
	}

	return pSR->status.u.ioValue;
}

//*********************************************************************
// Hardware digital input routine, multiple values, internal clocking
//*********************************************************************

UINT Diginmi (LPServiceRequest pSR, HWND win, UINT Device, UINT Channel,
LPVOID Buffer, DWORD Length)
{
	UINT result;

	memset(pSR, 0, sizeof(DL_SERVICEREQUEST));
	DL_SetServiceRequestSize(*pSR);

	pSR->device = Device;
	pSR->hWnd = win;
	pSR->subsystem = DI;
	pSR->mode = INTERRUPT;
	pSR->operation = START;
	pSR->timing.typeEvent = RATEEVENT;
	pSR->timing.u.rateEvent.clock = INTERNAL1;
	pSR->timing.u.rateEvent.gate = DISABLED;
	pSR->timing.u.rateEvent.period = 1000;

	pSR->start.typeEvent = COMMAND;
	pSR->stop.typeEvent = TCEVENT;

	pSR->channels.nChannels = 1;
	pSR->channels.chanGain[0].channel = Channel;

	pSR->lpBuffers = (LPBUFFLIST)malloc(DL_BufferListBytes(1));
	if (!pSR->lpBuffers)
		return Error(Abort, BufAllocErr);

	pSR->lpBuffers->nBuffers = 1;
	pSR->lpBuffers->bufferSize = Length;
	pSR->lpBuffers->BufferAddr[0] = Buffer;

	result = NoErr;
	if (DriverLINX(pSR) != NoErr) {
		result = pSR->result;
			// Display error message
	pSR->operation = MESSAGEBOX;
	DriverLINX(pSR);
	}

	return 0;
}


//*********************************************************************
// Hardware digital input routine, multiple values, external clocking
//*********************************************************************
```

```
UINT Diginme (LPServiceRequest pSR, HWND win, UINT Device, UINT Channel,
LPVOID Buffer, DWORD Length)
{
	UINT result;

	memset(pSR, 0, sizeof(DL_SERVICEREQUEST));
	DL_SetServiceRequestSize(*pSR);

	pSR->device = Device;
	pSR->hWnd = win;
	pSR->subsystem = DI;
	pSR->mode = INTERRUPT;
	pSR->operation = START;
	pSR->timing.typeEvent = RATEEVENT;
	pSR->timing.u.rateEvent.clock = EXTERNAL;
	pSR->timing.u.rateEvent.gate = DISABLED;
	pSR->timing.u.rateEvent.period = 20;

	pSR->start.typeEvent = COMMAND;
	pSR->stop.typeEvent = TCEVENT;

	pSR->channels.nChannels = 1;
	pSR->channels.chanGain[0].channel = Channel;

	pSR->lpBuffers = (LPBUFFLIST)malloc(DL_BufferListBytes(1));
	if (!pSR->lpBuffers)
		return Error(Abort, BufAllocErr);

	pSR->lpBuffers->nBuffers = 1;
	pSR->lpBuffers->bufferSize = Length;
	pSR->lpBuffers->BufferAddr[0] = Buffer;

	result = NoErr;
	if (DriverLINX(pSR) != NoErr) {
		result = pSR->result;
			// Display error message
	pSR->operation = MESSAGEBOX;
	DriverLINX(pSR);
	}

	Sleep(20+Length/15);                                    //wait for I/O to
happen
	pSR->operation = STATUS;
	DriverLINX(pSR);

	if(pSR->status.u.ioStatus.status == scheduled) { //Check if finished ?
		return 0;
		}

	pSR->operation = STOP;     //otherwise abort I/O

	if (DriverLINX(pSR) != NoErr) {
		result = pSR->result;
			// Display error message
	pSR->operation = MESSAGEBOX;
	DriverLINX(pSR);
	}
	ErrorMessage("I/O timeout");
	return -1;          //timeout
}


//*********************************************************
// Hardware digital output routine, single value
//*********************************************************

UINT Digout (LPServiceRequest pSR, HWND win, UINT Device, UINT Channel, DWORD
Value)
```

```
{
	UINT result;

	memset(pSR, 0, sizeof(DL_SERVICEREQUEST));
	DL_SetServiceRequestSize(*pSR);

	pSR->device = Device;
	pSR->hWnd = win;
	pSR->subsystem = DO;
	pSR->mode = POLLED;
	pSR->operation = START;

	pSR->channels.nChannels = 1;
	pSR->channels.chanGain[0].channel = Channel;

	pSR->status.typeStatus = IOVALUE;
	pSR->status.u.ioValue = Value;

	result = NoErr;
	if (DriverLINX(pSR) != NoErr) {
		result = pSR->result;
			// Display error message
	pSR->operation = MESSAGEBOX;
	DriverLINX(pSR);
	}

	return 0;
}


//*********************************************************************
// Hardware digital output routine, multiple values, internal clocking
//*********************************************************************

UINT Digoutmi (LPServiceRequest pSR, HWND win, UINT Device, UINT Channel,
LPVOID Buffer, DWORD Length)
{
	UINT result;

	memset(pSR, 0, sizeof(DL_SERVICEREQUEST));
	DL_SetServiceRequestSize(*pSR);

	pSR->device = Device;
	pSR->hWnd = win;

	pSR->subsystem = DO;
	pSR->mode = INTERRUPT;
	pSR->operation = START;
	pSR->timing.typeEvent = RATEEVENT;
	pSR->timing.u.rateEvent.clock = INTERNAL1;
	pSR->timing.u.rateEvent.gate = DISABLED;
	pSR->timing.u.rateEvent.period = 1000;
	pSR->timing.u.rateEvent.channel = DEFAULTTIMER;

	pSR->start.typeEvent = COMMAND;
	pSR->stop.typeEvent = TCEVENT;

	pSR->channels.nChannels = 1;
	pSR->channels.chanGain[0].channel = Channel;

	pSR->lpBuffers = (LPBUFFLIST)malloc(DL_BufferListBytes(1));
	if (!pSR->lpBuffers)
		return Error(Abort, BufAllocErr);

	pSR->lpBuffers->nBuffers = 1;
	pSR->lpBuffers->bufferSize = Length;
	pSR->lpBuffers->BufferAddr[0] = Buffer;
```

```
        result = NoErr;
        if (DriverLINX(pSR) != NoErr) {
                result = pSR->result;
                        // Display error message
        pSR->operation = MESSAGEBOX;
        DriverLINX(pSR);
        }

        return 0;
}


//*************************************************************************
// Hardware digital output routine, multiple values, external clocking
//*************************************************************************

UINT Digoutme (LPServiceRequest pSR, HWND win, UINT Device, UINT Channel,
LPVOID Buffer, DWORD Length)
{
        UINT result;

        memset(pSR, 0, sizeof(DL_SERVICEREQUEST));
        DL_SetServiceRequestSize(*pSR);

        pSR->device = Device;
        pSR->hWnd = win;

        pSR->subsystem = DO;
        pSR->mode = INTERRUPT;
        pSR->operation = START;
        pSR->timing.typeEvent = RATEEVENT;
        pSR->timing.u.rateEvent.clock = EXTERNAL;
        pSR->timing.u.rateEvent.gate = DISABLED;
        pSR->timing.u.rateEvent.period = 20;

        pSR->start.typeEvent = COMMAND;
        pSR->stop.typeEvent = TCEVENT;

        pSR->channels.nChannels = 1;
        pSR->channels.chanGain[0].channel = Channel;

        pSR->lpBuffers = (LPBUFFLIST)malloc(DL_BufferListBytes(1));
        if (!pSR->lpBuffers)
                return Error(Abort, BufAllocErr);

        pSR->lpBuffers->nBuffers = 1;
        pSR->lpBuffers->bufferSize = Length;
        pSR->lpBuffers->BufferAddr[0] = Buffer;


        result = NoErr;
        if (DriverLINX(pSR) != NoErr) {
                result = pSR->result;
                        // Display error message
        pSR->operation = MESSAGEBOX;
        DriverLINX(pSR);
        }

        Sleep(20+Length/15);                                    //wait for I/O to
happen
        pSR->operation = STATUS;
        DriverLINX(pSR);

        if(pSR->status.u.ioStatus.status == scheduled) { //Check if finished ?
                return 0;
```

```
            }

        pSR->operation = STOP;      //otherwise abort I/O

        if (DriverLINX(pSR) != NoErr) {
              result = pSR->result;
                    // Display error message
        pSR->operation = MESSAGEBOX;
        DriverLINX(pSR);
        }
    ErrorMessage("I/O timeout");
        return -1;              //timeout
}
```

Appendix C

An assembler source listing of a pulse program for the DSP.

robnz.asm

```
;*************************************************************
;FID PULSE PROGRAM:
;*************************************************************
;
; This program does a simple FID experiment. 4096 data points
;
; The filter passband is 1.9 to 2.9 KHz
; B1 f set to 2.4KHz
;*****************************************************************************
        nolist
        include  'ioequ.asm'
        include  'intequ.asm'
        include  'ada_equ.asm'
        list
;*****************************************************************************
;
;       The following EQUates will define the operatonal parameters
;       of the codec.  Please refer to the ADA_EQU.ASM source file
;       for a description of the parameters selections available.  The
;       variables defined by the EQUates are sent to the  codec via
;       the transmit buffer, TX_BUFF.
;
;       Since the parameters are defined in ADA_EQU.ASM, these lines must
;       follow the include statement.
;=============================================================================
;
CTRL_WD_12      equ     NO_PREAMP+HI_PASS_FILT+SAMP_RATE_48+STEREO+DATA_16
;CLB=0
CTRL_WD_34      equ     IMMED_3STATE+XTAL1_SELECT+BITS_64+CODEC_MASTER
CTRL_WD_56      equ     $000000
CTRL_WD_78      equ     $000000

TONE_OUTPUT     equ     HEADPHONE_EN+LINEOUT_EN+(0*LEFT_ATTN)+(0*RIGHT_ATTN)
TONE_INPUT      equ     MIC_IN_SELECT+(15*MONITOR_ATTN)

   ;---Buffer for talking to the CS4215

        org     x:$0
RX_BUFF_BASE    equ     *
RX_data_1_2     ds      1       ;data time slot 1/2 for RX ISR
RX_data_3_4     ds      1       ;data time slot 3/4 for RX ISR
RX_data_5_6     ds      1       ;data time slot 5/6 for RX ISR
RX_data_7_8     ds      1       ;data time slot 7/8 for RX ISR

TX_BUFF_BASE    equ     *
TX_data_1_2     ds      1       ;data time slot 1/2 for TX ISR
TX_data_3_4     ds      1       ;data time slot 3/4 for TX ISR
TX_data_5_6     ds      1       ;data time slot 5/6 for TX ISR
TX_data_7_8     ds      1       ;data time slot 7/8 for TX ISR

RX_PTR          ds      1       ;Pointer for rx buffer
TX_PTR          ds      1       ;Pointer for tx buffer

DOSC_BUFF_BASE  equ     *
coeff           ds      1       ; data location for osc. a's coeff.
s1              ds      1       ; data location for osc. a's sr1.
s2              ds      1       ; data location for osc. a's sr2.
;
;*****************************************************************************
```

```
;       Space for filter coeffs, samples etc
;
        org     y:$20
FPTR_BASE       equu    *       ;Memory for filter pointers
FLT1_PTR        ds      1
FLT2_PTR        ds      1
FLT3_PTR        ds      1
FLT4_PTR        ds      1
FLT5_PTR        ds      1
FLT6_PTR        ds      1
FOUT_BASE       EQU     *       ;Memory for filter outputs
FLT1_OUT        ds      1
FLT2_OUT        ds      1
FLT3_OUT        ds      1
FLT4_OUT        ds      1
FLT5_OUT        ds      1
FLT6_OUT        ds      1

        org     x:$40
SDATA_BASE      equ     *       ;Memory for filter data

        org     y:$40
COEFF_BASE      equ     *       ;Filter coefficients

;*************************************************************************
;       Digital oscillator config
;
Fs      set     48000.0                         ;Specify sampling frequency.
PI      set     2.0*@asn(1.0)                   ;Compute PI as
2.0*arcsin(1.0)
factor  set     PI/180.0                        ;Multiplier for degrees to
radians
eighteen set    18
hex_twenty  set   $20

; Specification for tone a.

freq_a  set     2400.0                          ;Specify frequency in Hertz.
phi_a   set     360.0*(freq_a/Fs)               ;Compute phi
phase_a set     0.0                             ;Specify the phase angle in
                                                ; degrees (-180 -- +180).
amp_a   set     0.3                             ;Specify amplitude (0-1).
theta2_a        set     (phase_a-(2.0*phi_a))   ;Compute theta2
theta1_a        set     (phase_a-phi_a)         ;Compute theta1
s2_a    set     amp_a*@sin(factor*theta2_a)     ;Compute s2
s1_a    set     amp_a*@sin(factor*theta1_a)     ;Compute s1
coeff_a set     @cos(factor*phi_a)              ;Compute rcoef in 2:14 format

;*************************************************************************
**
;
        org     p:$000600               ;start

        move    #fptrs,r1               ;Set up filter pointers
        move    #FPTR_BASE,r3
        do      #6,fp1
        move    p:(r1)+,y0
        move    y0,y:(r3)+
fp1     move    #fcoeff,r1              ;Set up filter coeffs
        move    #COEFF_BASE,r3
        do      #192,fc1
        move    p:(r1)+,y0
        move    y0,y:(r3)+
fc1     move    #0,y0                   ;Clear filter data
        move    #SDATA_BASE,r3
        do      #192,fdc
        move    y0,x:(r3)+
fdc
```

```
                                        ;Now need to synchronise with ADC
        jset    #2,x:M_SSISR0,*         ;Wait for frame sync to pass
        jclr    #2,x:M_SSISR0,*         ;Wait for frame sync

        clr     a
        move    a,x:TX_BUFF_BASE        ;transmit left
        move    a,x:TX_BUFF_BASE+1      ;transmit right

        move    #TONE_OUTPUT,y0         ;set up control words
        move    y0,x:TX_BUFF_BASE+2
        move    #TONE_INPUT,y0
        move    y0,x:TX_BUFF_BASE+3

; Turn on polarizing gate and wait for 50mS
        movep   #$04,x:M_PDRD
        do      #5,polzo
        bsr     wait10m
polzo
; Turn on Polarizing coil and cooling relay and wait for 4 seconds
        movep   #$2c,x:M_PDRD
        do      #400,polzt
        bsr     wait10m
polzt
; Turn off polarizing coil and cooling and wait for 50mS
        movep   #$04,x:M_PDRD
        do      #5,polzd
        bsr     wait10m
polzd
; Turn off polarizing gate and wait for 50mS
        movep   #$0,x:M_PDRD
        do      #5,polzg
        bsr     wait10m
polzg

; Output B1 dummy to settle filters
        move    #coeff_a,x0             ;Set up dig osc
        move    x0,x:DOSC_BUFF_BASE     ;Load coeff. for osc. a.
        move    #s1_a,x0
        move    x0,x:DOSC_BUFF_BASE+1   ;Load s1 for osc. a.
        move    #s2_a,x0
        move    x0,x:DOSC_BUFF_BASE+2   ;Load s2 for osc. a.

        move    #$0,r1
B1_d    move    r1,b                    ;Done?
        cmp     #$40,b                  ;n points ?
        beq     B1d_e                   ;Branch if so
        jset    #2,x:M_SSISR0,*         ;Wait for frame sync to pass
        jclr    #2,x:M_SSISR0,*         ;Wait for frame sync
        move    #DOSC_BUFF_BASE,r5      ;Load pointer to osc's coeff, sr1 and
sr2
        jsr     dosc_sin                ;Call oscillator routine.
        move    a,x:TX_BUFF_BASE        ;transmit left
        move    a,x:TX_BUFF_BASE+1      ;transmit right
        move    (r1)+                   ;Increment count
        bra     B1_d                    ;Loop back.

; Turn on B1 TX gate
B1d_e   movep   #$02,x:M_PDRD

; Output B1 90 degree pulse

        move    #$0,r1
B1_90   move    r1,b                    ;Done?
        cmp     #$80,b                 ;n points ?
        beq     B190_e                  ;Branch if so
        jset    #2,x:M_SSISR0,*         ;Wait for frame sync to pass
        jclr    #2,x:M_SSISR0,*         ;Wait for frame sync
```

```
        move    #DOSC_BUFF_BASE,r5      ;Load pointer to osc's coeff, sr1 and
sr2
        jsr     dosc_sin                ;Call oscillator routine.
        move    a,x:TX_BUFF_BASE        ;transmit left
        move    a,x:TX_BUFF_BASE+1      ;transmit right
        move    (r1)+                   ;Increment count
        bra     B1_90                   ;Loop back.
B190_e  clr     a                       ;Set outputs to zero
        move    a,x:TX_BUFF_BASE        ;transmit left
        move    a,x:TX_BUFF_BASE+1      ;transmit right

; Wait 250uS then turn off B1 TX gate
        do      #10,b111
        bsr     wait25u
b111    movep   #$00,x:M_PDRD
; Wait 5mS
        do      #200,delfid
        bsr     wait25u

; Sample FID

delfid  move    #$10fc0,r5              ;Make r5 point to the start of fid
memory
        move    #0,r2                   ;Filter counter
simp_l  move    r5,b                    ;Done?
        cmp     #$12000,b               ;4096 points ? + 64 preamble
        beq     simp_e                  ;Branch if so
        jset    #2,x:M_SSISR0,*         ;Wait for frame sync to pass
        jclr    #2,x:M_SSISR0,*         ;Wait for frame sync
        move    x:RX_BUFF_BASE,a        ;Load left channel data
        move    x:RX_BUFF_BASE+1,b      ;Load right channel data
        bsr     do_fltr                 ;Filter data, decimate and store
        clr     a
        move    a,x:TX_BUFF_BASE        ;transmit left
        move    a,x:TX_BUFF_BASE+1      ;transmit right
        move    #TONE_OUTPUT,y0         ;set up control words
        move    y0,x:TX_BUFF_BASE+2
        move    #TONE_INPUT,y0
        move    y0,x:TX_BUFF_BASE+3
        bra     simp_l                  ;Loop again
simp_e  do      #300,edel               ;wait for 3 seconds
        bsr     wait10m
edel
        rts
;
;***************************************************************
;       Wait 10mS routine
;
wait10m dor     #400,w10e
        bsr     wait25u
w10e    rts
;
;***************************************************************
;       Wait 25uS routine (80MHz clock)
;
wait25u move    #2000,x0
        rep     x0
        nop
        nop
        rts
;
;***************************************************************
; The following subroutine calculates the next sinusoidal output value as a
; function by the digital oscillator given that r5 points to the memory
; location that contains the "coeff" value followed by the memory location
; that contains the "s1" value, followed by the memory location that contains
; the "s2" value.  The formula and block diagram of the oscillator are:
;
```

```
;               s1[n] = coeff*s1[n-1] - s2[n-1] = coeff*s1[n-1] - s1[n-2]
;
;
;                  _______    _______
;                 |       |  s1   |       |  s2
;               +-->|  z^-1 |--+--->|  z^-1 |----+
;               |   |_______|  |    |_______|    |
;               |              |                 |
;               |            __V__             __V__
;               |           |     |           |     |
;               |           | coeff |         |  -1   |
;               |           |_____|           |_____|
;               |              |                 |
;               |              |                 |
;               |              |                 |
;               |              +----->( + )<-----+
;               |                      |
;               |                      |
;               +----------------------+---------> sine output

dosc_sin

        move    x:(r5)+,x0              ; Load coeff
        move    x:(r5)+,y0              ; Load s1 into a.
        move    x:(r5)-,b               ; Load s2 into b, r5 points to s1.
        mpy     x0,y0,a                 ;Get coef*s1 in a in 2:14 format
        subl    b,a          ;y0,b      ;Get (coef*s1 -s2)=sin_val in a
                                        ;  in fractional format
        move    a,x:(r5)+               ; Save new s1.
        move    y0,x:(r5)               ; Save new s2.
        rts
;
;*************************************************************
;
;       Filter routine
;       On entry a holds sample
;       r2 is the filter number counter
;
;       This routine band pass filters and decimates down to 8KHz
;       It does this simultaneously using 6 filters instead of 1
;       This is much more efficient.
;       48KHz divided by 6 = 8KHz
;
ncoef   equ     32                      ;Number of coeffs in each filter
nfilt   equ     6                       ;Number of filters
;
do_fltr move    r2,r4                   ;Make r4 point to coeff set
        move    (r4)+
        move    (r4)+
        move    r4,b1
        lsl     #5,b
        move    b1,r4
        move    #FPTR_BASE,r0           ;and r0 to the samples
        do      r2,saml
        move    (r0)+
saml    move    r0,r3
        move    y:(r3),r0
        move    r3,r1                   ;Make r1 point to filter pointer
        do      #6,samlo
        move    (r3)+                   ;make r3 point to filter out location
samlo   move    #ncoef-1,m0             ;set modulo arithmetic
        move    m0,m4                   ;for the 2 circular buffers
        nop
        nop
        move    a,x:(r0)                        ;input sample in memory
        clr     a       x:(r0)+,x0      y:(r4)+,y0
        move    r0,y:(r1)
        rep     #ncoef-1
        mac     x0,y0,a x:(r0)+,x0      y:(r4)+,y0
```

```
        macr    x0,y0,a (r0)-
        move    a,y:(r3)                ;output filtered sample and
        move    (r2)+                   ;increment counter
        move    r2,b
        cmp     #6,b                    ;overflow ?
        beq     fltr_j
        bra     fltr_e
fltr_j  move    #0,r2                   ;filter 1 next
        move    #FOUT_BASE,r3           ;Sum filter outputs
        clr     a
        do      #6,slp
        move    y:(r3)+,x0
        add     x0,a
slp     move    a,y:(r5)+               ;Write to memory
fltr_e  move    #-1,m4                  ;linear addressing
        move    #-1,m0
        rts

;*************************************************************
;
fptrs   dc  $40
        dc  $60
        dc  $80
        dc  $a0
        dc  $c0
        dc  $e0

fcoeff  dc -0.0036,-0.0024,0.0017,0.0024,-0.0015,0.0002,-0.0041,0.0036
        dc 0.0070,-0.0076,-0.0004,-0.0068,0.0092,0.0247,-0.0428,-0.0089
        dc 0.0592,-0.0254,-0.0326,0.0290,0.0028,-0.0049,-0.0019,-0.0063
        dc 0.0086,0.0012,-0.0035,0.0002,-0.0014,0.0030,0.0008,-0.0023
        dc 0.0036,-0.0021,0.0025,0.0016,-0.0014,-0.0001,-0.0043,0.0059
        dc 0.0045,-0.0078,0.0003,-0.0080,0.0161,0.0170,-0.0493,0.0091
        dc 0.0532,-0.0387,-0.0200,0.0297,-0.0024,-0.0028,-0.0038,-0.0042
        dc 0.0094,-0.0010,-0.0026,-0.0001,-0.0009,0.0034,-0.0001,-0.0018
        dc 0.0011,-0.0016,0.0031,0.0007,-0.0010,-0.0007,-0.0038,0.0079
        dc 0.0016,-0.0071,0.0001,-0.0078,0.0224,0.0062,-0.0510,0.0267
        dc 0.0420,-0.0474,-0.0065,0.0273,-0.0060,-0.0010,-0.0056,-0.0015
        dc 0.0091,-0.0027,-0.0016,-0.0006,-0.0002,0.0034,-0.0009,-0.0007
        dc -0.0007,-0.0009,0.0034,-0.0002,-0.0006,-0.0016,-0.0027,0.0091
        dc -0.0015,-0.0056,-0.0010,-0.0060,0.0273,-0.0065,-0.0474,0.0420
        dc 0.0267,-0.0510,0.0062,0.0224,-0.0078,0.0001,-0.0071,0.0016
        dc 0.0079,-0.0038,-0.0007,-0.0010,0.0007,0.0031,-0.0016,0.0011
        dc -0.0018,-0.0001,0.0034,-0.0009,-0.0001,-0.0026,-0.0010,0.0094
        dc -0.0042,-0.0038,-0.0028,-0.0024,0.0297,-0.0200,-0.0387,0.0532
        dc 0.0091,-0.0493,0.0170,0.0161,-0.0080,0.0003,-0.0078,0.0045
        dc 0.0059,-0.0043,-0.0001,-0.0014,0.0016,0.0025,-0.0021,0.0036
        dc -0.0023,0.0008,0.0030,-0.0014,0.0002,-0.0035,0.0012,0.0086,
        dc -0.0063,-0.0019,-0.0049,0.0028,0.0290,-0.0326,-0.0254,0.0592
        dc -0.0089,-0.0428,0.0247,0.0092,-0.0068,-0.0004,-0.0076,0.0070
        dc 0.0036,-0.0041,0.0002,-0.0015,0.0024,0.0017,-0.0024,-0.0036

;*************************************************************
;
```

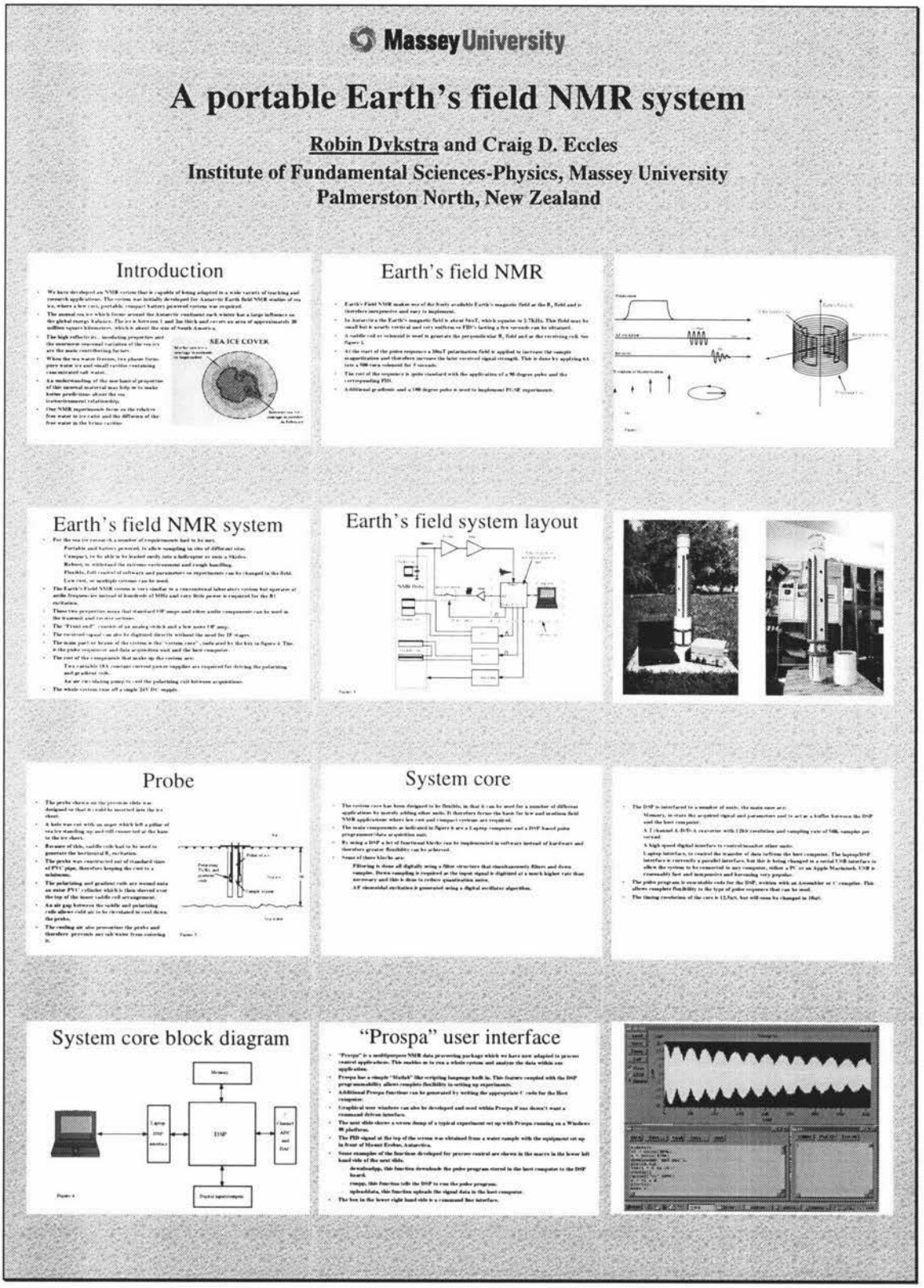
Appendix D

A Matlab listing of the filter design macro.

```
% Filter for Earths field NMR in Antarctica
% The sample rate is 48KHz
% The filter is a bandpass type with a passband between 2.6 and 2.9 KHz.
% It is an FIR filter with 192 taps.
n = 192;
f = [0 0.09 0.11 0.12 0.14 1];
m = [0 0 1 1 0 0];
b = remez(n-1,f,m);
[H,w] = freqz(b);
L = 10*(log(H));
plot(24*w/pi,L);
axis([1 5 -60 5]);
grid on;
j = 1;
for i = 1 : n/6;
   c1(i)= b(j);
   j = j+1;
   c2(i)= b(j);
   j = j+1;
   c3(i)= b(j);
   j = j+1;
   c4(i)= b(j);
   j = j+1;
   c5(i)= b(j);
   j = j+1;
   c6(i)= b(j);
   j = j+1;
end;

l=1;
for k = 1 : n/6;
   cf(l) = c1(k);
   l = l+1;
end

for k = 1 : n/6;
   cf(l) = c2(k);
   l = l+1;
end

for k = 1 : n/6;
   cf(l) = c3(k);
   l = l+1;
end

for k = 1 : n/6;
   cf(l) = c4(k);
   l = l+1;
end

for k = 1 : n/6;
   cf(l) = c5(k);
   l = l+1;
end

for k = 1 : n/6;
   cf(l) = c6(k);
   l = l+1;
end
b'
cf'
```

Appendix E

A poster presented at the ANZMAG2000 conference of the Australian and New Zealand Society for Magnetic Resonance.



Appendix F

The following pages are a paper that is relevant to this work.

ELSEVIER

Cold Regions Science and Technology 29 (1999) 153–171

cold regions science and technology

www.elsevier.com/locate/coldregions

A nuclear magnetic resonance study of Antarctic sea ice brine diffusivity

P.T. Callaghan [a,*], R. Dykstra [a], C.D. Eccles [a], T.G. Haskell [b], J.D. Seymour [c]

[a] *Institute of Fundamental Sciences-Physics, Massey University, Palmerston North, New Zealand*
[b] *Industrial Research, Gracefield, Lower Hutt, New Zealand*
[c] *New Mexico Resonance, Albuquerque, NM, USA*

Received 9 March 1999; accepted 9 June 1999

Abstract

We have measured the diffusive motion of water molecules in the brine inclusions of Antarctic sea ice using a specially constructed nuclear magnetic resonance (NMR) apparatus. The method relies on the use of pulsed magnetic field gradients in precise analogy to well established laboratory procedures. One version of the apparatus utilised core samples extracted from the ice sheet which were subsequently analysed on site while a later version utilised a probehead which was inserted into the ice sheet, thus minimising any sample perturbation. The diffusive motion of water molecules in the brine inclusions is found to be strongly anisotropic, and, over short length scales, exhibits a rapidity greatly in excess of that expected for thermal equilibrium Brownian behaviour, an effect which we attribute to convective transport. © 1999 Elsevier Science B.V. All rights reserved.

Keywords: NMR; Antarctic sea ice; Water molecules

1. Introduction

The annual sea ice which forms around the Antarctic continent each Southern Hemisphere winter exerts an important influence on the global energy, biomass and carbon dioxide balance and, as a consequence, is a crucial component in modelling the global climate and ecology. This ice, between 1 and 2 m thick, covers an area of about 20 million square kilometres in winter and whereas the open ocean of the summer reflects only 5% in sunlight, the sea ice reflects over 90%, a significant factor in determining the Earth's albedo. In winter the sea ice insulates the cold polar atmosphere from the warmer ocean surface, thus restricting heat exchange (Allison, 1997) In the carbon dioxide cycle, seasonal ice cover limits carbon dioxide exchange while subduction of surface water acts as a carbon dioxide sink (Savidge et al., 1996). Formation of the ice releases salt into the underlying ocean, increasing density and inducing the formation of dense, cold and saline Antarctic Bottom Water which is transported northward. The movement of ice northwards during the summer further influences the meridional heat and freshwater transport. Melting of the sea ice releases freshwater and algal spores to the ocean, increasing stability and stimulating phytoplancton growth (Savidge et al., 1996).

* Corresponding author. E-mail: p.callaghan@massey.ac.nz

0165-232X/99/$ - see front matter
PII: S0165-232X(99)00024-5

There are a number of reasons why an appreciation of the sea ice microstructure is important to any understanding of the behaviour of this geophysical medium, and thereby to any modelling of global energy and climate behaviour. First, the mechanical properties which derive from this structure determine the strength, rheology and resistance to fracture and breakup under the influence of wave action. Second, the structure of the composite brine pore/ice crystal medium is important in determining sea ice thermal conductivity, and the intra-pore brine mobility may well play an important role in aiding convective heat transport. Finally, and possibly most importantly, the interpretation of remotely sensed active and passive microwave observations of sea ice formation and dispersal depends on the correct modelling of the dielectric behaviour of sea ice (Golden, 1995). This behaviour is believed (Lin et al., 1988) to strongly correlate with the spatial distribution, size, anisotropy, alignment and fractional volume of brine inclusions.

In this article, we report on the use of a nuclear magnetic resonance (NMR) method to measure structural and dynamical properties of first year sea ice brine inclusions under Antarctic conditions. While NMR is a very old technique and one which has been used to measure brine volumes in laboratory-grown saline ice (Richardson and Keller, 1966) our work is novel in a number of respects. First, we are not aware of any other use of NMR in Antarctica, except for magnetometry studies, and certainly we are not aware of such measurements on sea ice being carried out in situ. Second our technique is somewhat unusual in that we are able to measure not only brine content, but also brine mobility. The particular method employed is standard in laboratory applications of NMR but we have pioneered the use of that method as regards it's portability and its ability to be used in remote regions or under environmentally harsh conditions (Callaghan et al., 1997, 1998). The key to that portability is our use of the Earth's magnetic field to observe nuclear precession, rather than by using a resistive or superconducting electromagnet more familiar to laboratory-based studies. Third, while our early measurements were made on extracted ice cores, we have recently improved our apparatus so that it can be inserted into the ice sheet, so as to examine the properties of ice samples which remain hydrostatically connected to their surroundings. Our ability to make measurements in situ under Antarctic conditions is particularly useful since the structure of sea ice may be crucially dependent on the growth history and on prevailing temperature and salinity profiles within the ice sheet (Veazy et al., 1994).

We shall report here on measurements made on sea ice over three spring seasons in the vicinity of Cape Evans, McMurdo Sound. We shall show that the NMR method provides a rapid measurement of brine volume, and that the results of those measurements are in excellent agreement with calculations based on the measured values of salinity and temperature in each sample. In addition, we demonstrate that the NMR measurements of brine diffusivity provide new insights, in particular indicating the existence of molecular motion much faster than expected on the basis of thermal equilibrium Brownian dynamics. This non-equilibrium fluid transport may possibly be associated with convection. Our measurements are made on samples on the order of several centimetres diameter and length and as a consequence, represent a statistical average over all brine inclusions. Clearly therefore any measurement of molecular motion will contain contributions from brine in a wide variety of inclusion geometries. Our method is able to distinguish the signals from molecules transported at differing rates. For example, we are able to see that, in addition to the water molecules undergoing rapid dispersion, a significant fraction of the brine exhibits restricted motion. Such restrictions are to be expected in a porous medium where the collision of water molecules with the walls of the enclosing pore reduces the effective Brownian motion (Callaghan, 1991). We shall show that this restricted motion is strongly temperature dependent and that it is orientationally anisotropic.

Before embarking on a detailed description of our work, we wish to acknowledge an inherent difficulty. NMR is a new technique as regards any study of sea ice. While we have confidence in our findings we realise that any new method applied to an old problem will necessarily be greeted with healthy scepticism, especially if its outcomes are incongruous with earlier findings. In fact we regard our method as entirely complementary to the more traditional means of probing sea ice structure and that far from being

inconsistent with known data, our results provide new, yet compatible insight.

2. Known structure of sea ice

Sea ice is a heterogeneous mixed phase material consisting of an array of platelets of pure ice, separated by rows of brine-filled inclusions. There exist a number of studies of sea ice structure by means of optical microscopy and a great deal is known about its morphological character (Weeks and Hamilton, 1962; Weeks and Assur, 1967; Doronin and Kheisin, 1975; Weeks and Ackley, 1982; Gow and Tucker, 1991; Jeffries et al., 1993). The initial freezing occurs near the surface of the seawater to produce a soup of frazil crystals which freeze together to produce flat sheet nilas, or in the presence of waves, solid ice pancakes which may consolidate to produce a continuous ice sheet. Direct freezing under the sheet produces congelation ice, typified by columnar-shaped crystals. This growth rate is determined by the rate at which latent heat can be extracted and hence by the thermal conductivity in the ice sheet along with the prevailing thermal gradient (Gow and Tucker, 1991). As the ice freezes, the salts are rejected in a concentrated brine thus successively lowering the freezing point of the remaining fluid. Some of this brine drains out through vertical channels as the sea ice grows in thickness, but much remains entrapped between ice crystals.

In the congelation ice which forms underneath the surface ice layer, there develops a strong vertical crystal elongation and, in the subsequent development of the ice, those crystals with c-axis horizontal dominate the growth, resulting in a structure in which pure ice plates are interspersed with parallel layers of brine inclusions. Eventually, many of these inclusions become segregated, trapping brine between the plates in narrow layers. The spacing of the plates is on the order of 1 mm while the brine layer thickness is typically an order of magnitude smaller, although these dimensions may differ from region to region according to the rate of ice crystal growth (Gow and Tucker, 1991; Jeffries et al., 1993).

The congelation zone is separated from the frazil layer by a transition zone, around 10 cm thick, in which the ice crystal orientation becomes more preferentially aligned. The depth of the congelation ice transition below the surface depends on growth conditions, and on the amount of snow layer which has consolidated above the frazil. Typically McMurdo Sound congelation ice can be found 10 to 20 cm below the surface and may be easily identified by optical observations of thin ice sections cut from near the surface of the sheet. One recent assessment of McMurdo Sound fast sea ice found that the columnar zone comprised around 60% of the sheet, with platelet and congelation/platelet ice making up the rest (Jeffries et al., 1993). At its base, the skeletal layer (around 10 to 30 mm thick of ice platelets separated by brine) forms the boundary with the sea water at its freezing point of $-1.8°C$. This is the region of dendritic ice growth in which convective heat transport in the underlying seawater is believed to be aided by the subsidence of cold brine-rich streamers as salt is expelled from the ice water interface above (Lewis and Weeks, 1971). In Antarctic ice the base of this skeletal layer is generally covered by an algal film.

Brine inclusions in the congelation ice exhibit a variety of geometries. Niedrauer and Martin (1979), in a study of laboratory-grown ice, refer to pockets, tubes and channels. Pockets are isolated bubbles of brine, typically on the order of 0.1 mm diameter, with limited or no communication to other brine masses while tubes are cylindrical brine-filled hollows with similar diameters but much greater vertical extent. Throughout the sea ice sheet, larger vertically oriented brine channels with diameters of around 0.5 mm (but with a few as large as 5 mm) interconnect the sheet in a tree-like array. It is believed (Gow and Tucker, 1991; Golden et al., 1998) that under sustained ice warming, initially disconnected brine inclusions may coalesce into vertical channels thus aiding further brine migration. As a consequence multi-year ice has a much lower salinity than first year ice.

There exist a number of important differences between Antarctic and Arctic first year ice. In particular first year Antarctic ice is considerably thicker at similar latitudes, than Arctic ice (Lewis and Weeks, 1971). In the Antarctic much of the ice consists of randomly oriented crystals characteristic of frazil (Paige, 1966; Gow and Tucker, 1990), a result of the turbulence of the southern ocean in which pancakes

are jumbled and rafted. Deep within the pack however, the open water areas become filled with ice in the Arctic manner, with ordered congelation ice forming a few centimetres below the surface (Jeffries et al., 1993). It is this latter type of ice sheet which we have examined in the McMurdo Sound area. Jeffries et al. (1995) have reported on similar structural characteristics in Arctic sea ice from the Beaufort Sea.

3. Nuclear magnetic resonance

3.1. Simple background to NMR

In this section we briefly review the underlying principles of the NMR method, while acknowledging that the subject is too broad (and too complex) to be adequately described in a few paragraphs. Readers familiar with NMR may wish to skip this section while those seeking a deeper understanding can find the method described in detail in a number of texts (Abragam, 1961; Slichter, 1963; Farrar and Becker, 1971; Mansfield and Morris, 1982; Shaw, 1984; Callaghan, 1991). NMR is one of the most important laboratory tools used in the investigation of condensed matter. It is particularly sensitive to molecular properties, including molecular structure, molecular order and alignment, and molecular rotational and translational dynamics. The method relies on the fact that atomic nuclei of non-zero angular momentum (or spin) undergo a precessional motion when placed in a laboratory magnetic field, at frequencies typically in the radio frequency range of tens to hundreds of megahertz. While the interaction of the spins with that field represents the dominant factor influencing their energies, there exist, in addition, a number of other interaction processes which subtlety alter precession rates and which can be observed with exquisite sensitivity by the NMR method. These additional terms arise principally from the molecular environment in which the atomic nucleus finds itself. It is beyond the scope of this article to review those interactions here and the interested reader can find them discussed in depth elsewhere (Slichter, 1963; Callaghan, 1991). However, we will focus on two particular phenomena of direct relevance to the present work, namely the inter-nuclear dipolar interaction, and the influence of molecular self-diffusion under the influence of a deliberately applied magnetic field gradient. As we shall see, both of these influence the phase coherence of an ensemble of nuclear spins and directly determine the NMR signal amplitude, each in a characteristic manner. Before discussing these however, we provide a simple description of the NMR method.

Of all stable atomic nuclei the hydrogen nucleus (proton) has the highest ratio of intrinsic magnetism to angular momentum (gyromagnetic ratio) and is therefore the most sensitive candidate for NMR observation, a factor which is aided by its abundance in condensed matter. In what follows we shall be concerned with NMR of protons residing in water molecules within sea ice. The very large number of protons present in any given sample of condensed matter (typically $> 10^{20}$) exhibit so-called "ensemble-average" properties in any detection of the nuclear spin precession, which means that under thermal equilibrium conditions, where the precessional phases are randomised, no precession may be directly observed. In such a state the spins simply exhibit a weak magnetisation along the polarising magnetic field direction, as a consequence of a slight preponderance of spin vector components along that axis. However, if that magnetisation vector is disturbed from equilibrium, by for example, causing it to be redirected along an axis normal to the field, then the underlying precession shows up in a rotation of the magnetisation vector (the transverse magnetisation). Such a re-alignment can be achieved by a resonant excitation of the spins using a weak magnetic field applied in a direction orthogonal to the polarising direction, and which oscillates at the intrinsic precession frequency.

The detection of transverse magnetisation free precession is performed using a coil antenna, in which is induced an oscillatory emf which gradually decays with time (the Free Induction Decay of FID) as the spins gradually lose phase coherence. This loss of coherence can be caused by slight variations in the magnetic field across the sample or by weak dipolar interactions between the spins themselves. In the latter case the loss of coherence is generally irreversible and known as transverse relaxation, a process which is typically exponential and characterised by a relaxation time T_2. Remarkably, this dipolar interaction process is strongly influenced by

the rotational motion of the host molecules since its strength depends on the angle subtended between the inter-nuclear vector and the external magnetic field. In the case of liquids, where the tumbling is very rapid, the dipolar interaction is motionally averaged and the relaxation time T_2 is very long leading to transverse magnetisation coherence lifetimes on the order of seconds. By contrast, in solids, the effect of these interactions is so strong as to destroy the coherence in a few tens of microseconds. This fact is crucial to our application of NMR methods in sea ice studies. By its selective sensitivity to nuclear spins whose host molecules reside in the liquid state, NMR has the potential to discriminate the protons of the ice from those which reside in the liquid water phase and hence to provide a long-lived signal arising entirely from the brine. One direct consequence of that fact is that the method can be used to quantify sea ice brine content (Richardson and Keller, 1966).

As pointed out in the previous paragraph, any inhomogeneity in the applied magnetic field may also cause phase spreading in the precession of the transverse magnetisation and hence a loss of signal following the resonant excitation. However, this phenomenon differs from the stochastic processes which lead to transverse relaxation, in that the phase spreading is inherently reversible and hence the signal recoverable. The means by which this may be achieved is shown in Fig. 1. After a dephasing period τ the spin phases are all reversed using a resonant excitation pulse whose effect is to reorient all spins by 180°. Consequently the spread of precession speeds, which previously caused the ensemble magnetisation to de-phase during τ, now serves to

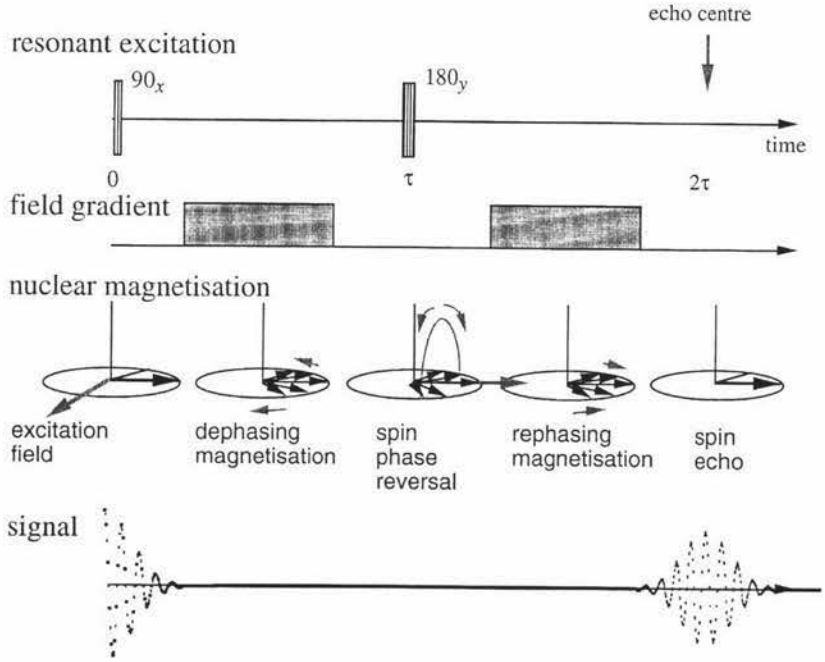


Fig. 1. Formation of a spin echo. Two resonant radio frequency pulses, oscillating at the precession frequency of the nuclear spins are used. The first (90_x) causes the longitudinal magnetisation to nutate into the transverse plane, subsequent to which it precesses. Application of a magnetic field gradient causes the spin isochromats to dephase, as shown. These phase shifts are all reversed by a subsequent 180_y rf pulse, so that application of a second gradient pulses causes the magnetisation vectors of the isochromats to come back into phase, to form a spin echo. This echo will be attenuated if diffusional motion of the molecules causes them to change location over the period between the application of the two gradient pulses.

bring them all back into phase coherence once a further period τ has elapsed following the 180° pulse. This process is known as a spin echo and it forms the basis of most of the sophisticated methodology underpinning modern NMR. We shall mention only one such method, of important to our sea ice work, namely the measurement of molecular self diffusion by means of pulsed magnetic field gradients, the so-called Pulsed Gradient Spin Echo (or PGSE) method (Stejskal and Tanner, 1965; Callaghan, 1991).

The precise angular frequency, ω, at which the nuclear spins precess depends upon the local magnetic field amplitude B_0 via the simple relation $\omega_0 = \gamma B_0$ where γ is the nuclear gyromagnetic ratio. Suppose we adopt the view that the net applied magnetic field, B, consists of a superposed uniform field $B_0\mathbf{k}$ (where the unit vector $\mathbf{k}$ defines the field direction) and a uniform magnetic field gradient $\mathbf{G} = \nabla B$. Then a nuclear spin at position $\mathbf{r}$ within the sample will have a local precession frequency given by

$$\omega(\mathbf{r}) = \omega_0 + \gamma \mathbf{G} \cdot \mathbf{r} \qquad (1)$$

This position dependence of frequency provides a means of labelling the spins by where they reside in the sample, the principle underlying magnetic resonance imaging. However, we shall be concerned with the consequences of positional **change** which result from molecular translational motion in the presence of the magnetic field gradient. This causes the precession frequency to be time dependent and in particular causes a perturbation to the rephasing process so crucial to the formation of the spin echo. Indeed, any motion occurring in the presence of such gradients will leave a characteristic signature in the spin echo amplitude and phase which may in turn be used to characterise the motion itself.

This underlying physics provides the basis for the PGSE experiment in which magnetic field gradients are applied during the spin echo sequence as well-defined pulses of amplitude $\mathbf{g}$, duration δ and separation Δ. There exists a significant literature on the subject of just what phase and amplitude effects are found in spin echo amplitudes for a wide variety of spin motions, including uniform flow, dispersive flow, free-diffusion, and restricted diffusion, the latter exhibiting features characteristic of the restricting geometry (Callaghan, 1991). We shall refer to some of this literature in what follows but for the moment we present just the simple relationship which exists between the normalised echo amplitude, E, and the molecular self-diffusion coefficient D in the case of free diffusion, namely (Stejskal and Tanner, 1965)

$$E = \exp\left[-\gamma^2 g^2 \delta^2 D(\Delta - \delta/3)\right] \qquad (2)$$

3.2. NMR and PGSE NMR in the Earth's magnetic field

NMR suffers from inherent signal-to-noise limitation because of the need to detect the spin precession in competition with thermal noise in the antenna. The best sensitivity is achieved by using the largest possible uniform magnetic field (typically 1 to 10 T), a factor which mitigates against apparatus portability. However it has long been realised that despite the fact that the Earth's magnetic field is extremely weak (around 50 μT), it can still be used to detect nuclear spin precession since it is so uniform that very large samples can be employed (Bene, 1971; Callaghan and Legros, 1982; Stepisnik et al., 1994). This makes Earth's field NMR an attractive candidate for use in outdoor applications. The sensitivity of the method can be further aided by using an additional, and rather rudimentary, resistive electromagnet to produce a large initial spin polarisation. This magnet, because it operates in pulsed mode and is not used to drive the spin precession during detection, avoids the need for coolant, has minimal constraints on uniformity, and may therefore be may be of very simple construction. Because the Earth's field is so low, spin precession frequencies are in the audio range of around 2.5 kHz, enabling the use of very large receiver and transmitter antennas (on the order of 10-cm diameter) and consequently large sample volumes are easily utilised. At such low frequencies quite simple transceiver circuitry may be employed and it is further possible to use direct digital detection of the FID using standard data acquisition systems. The principles behind the Earth field method are shown schematically in Fig. 2.

Note that the amplitudes of the polarising, precession (i.e., Earth's) and transmitted audio frequency fields are labelled B_p, B_e and B_1, respectively. The

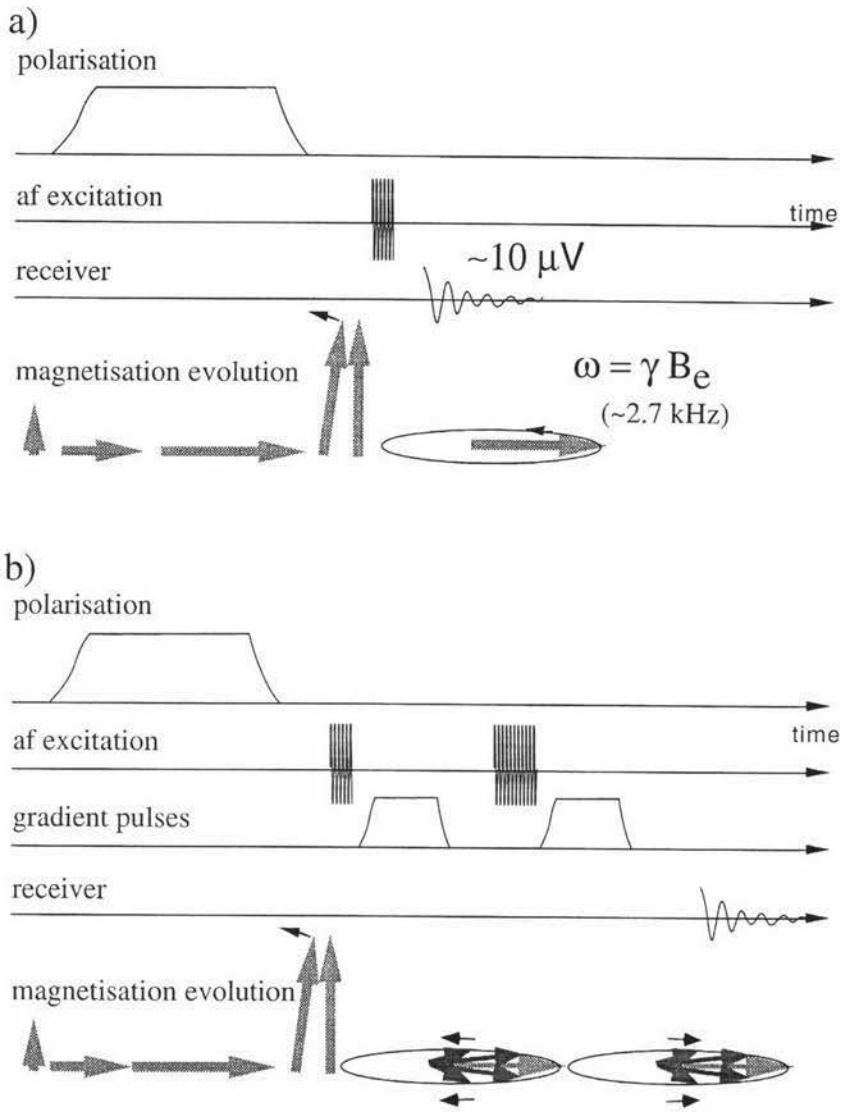


Fig. 2. (a) Schematic representation of the method used to obtain NMR signals using the Earth's magnetic field. (b) As for (a) but in the case where a spin echo is employed in conjunction with two magnetic field gradient pulses, according to the method of Fig. 1. Note that in the earths field the resonant pulses oscillate at audio frequencies.

polariser pulse needs to be applied for a duration sufficient for the establishment of the thermal equilibrium magnetisation. This is determined by the spin-lattice relaxation time, T_1, around 2 s for brine protons at the ambient temperatures experienced in Antarctica.

4. Apparatus description

We have developed a portable Earth's field NMR system suitable for use under Antarctic conditions. This system, which incorporates automated process control, uses a common clock for pulse sequencing, excitation pulse synthesis, and detection, thus resulting in a phase stability sufficient for accurate signal averaging. The instrument has additional Pulsed Magnetic Field Gradient coils and we have shown that it can be used for accurate PGSE measurements of diffusion (Callaghan, 1991). The details of the apparatus design as well as an explanation of the diffusion measurement methodology are described in an article in *Review of Scientific Instruments* (Callaghan et al., 1997). A key feature of that apparatus probehead is the need to place cylindrical-shaped samples in a horizontally aligned transceiver solenoid, as shown schematically in Fig. 3a. This was the mode in which we operated during visits to Antarctica in 1994 and 1995. Sea ice cores were extracted using a specially constructed auger (Rand

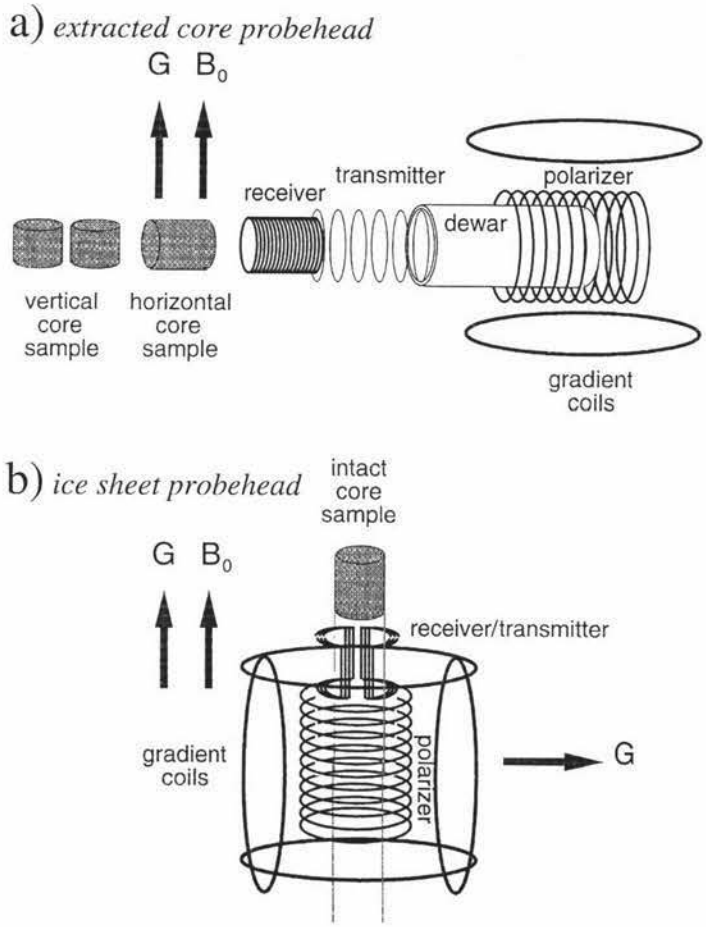


Fig. 3. Schematic representation giving an exploded view of (a) the *extracted core probehead* and (b) the *ice sheet probehead*. Note that in (b) samples remain fixed in position while the gradient coils are reoriented to enable both horizontal and vertical motion to be measured. The earlier apparatus shown in (a) was used in 1994 and 1995. In 1997 this apparatus was modified to permit gradient coil reorientation as in (b).

and Mellor, 1985) and placed within the NMR probehead which was situated in a tent close to the coring site. We shall refer to that apparatus, in what follows, as the *extracted core probehead*. This probehead was equipped with magnetic field gradient coils comprising a 100 turn Maxwell pair of 270 mm mean diameter and 240 mm mean spacing, fixed in a vertical orientation.

We subsequently developed an improved probehead which is capable of insertion directly into the icesheet, once a suitable annular hole is cut around the free standing core sample. This improved probehead system, which we shall refer to as the *ice sheet probehead*, was used during our third visit to McMurdo sound in 1997, and is shown schematically in Fig. 3b. It incorporates a vertically oriented solenoidal polarising coil comprising 504 turns of 1.6 mm diameter enamelled copper wire, (coil length 150 mm, diameter 140 mm, total resistance 1.8 Ω), along with a saddle coil pair wound from 0.2 mm diameter enamelled copper wire, (coil length 150 mm, diameter 110, total resistance 394 Ω) used as a common receiver and transmitter coil. A pump was used to drive external air down to the polarising coil so as to reduce any temperature rise associated with the polarising pulses.

When the apparatus surrounds a continuous cylinder, the effective sample length is determined by the rate at which B_1 drops off along the vertical axis away from the saddle coil. We have measured this effective length between half height contributions as approximately 100 mm.

The lower section of the probehead has an external diameter of 160 mm and the required annular hole in the ice sheet is cut using an auger whose blades produced a hole of 95 mm ID and 165 mm OD. By means of extension tubes the probehead could, in principle, be lowered through to the base of the sheet, some 2 m below the surface. However, our measurements in 1997 were carried out using a fixed depth of 300 mm to the sample centre.

In order to carry out diffusion measurements with the new apparatus it was necessary to cut an additional hole around the probe body so that a free standing Maxwell pair could be placed around the probe, centred on the sample centre. This coil set comprised 100 turns per winding of 1.6 mm enamelled copper wire held in wooden formers, to give a mean coil diameter of 352 mm and coil spacing of 313 mm. The cutting of the external hole was carried out using a chainsaw. The gradient coil could then be placed with its axis either horizontal or vertical, so as

Fig. 4. Photograph of the *ice sheet probehead* assembly in situ in the sea ice of McMurdo Sound, a few kilometres from Cape Evans. Mt Erebus is seen in the background. The cables connect the polarising, gradient, and receiver/transmitter coils to the instrument tent some 20 m distant.

to measure diffusion in either direction. Once the gradient set was in place the hole was packed with surface snow to provide mechanical rigidity and thermal insulation. Core temperatures were measured before and after all experiments, a complete set of which typically lasted on the order of 1 h per newly cut core.

Note that all the electronic details of the apparatus are, essentially, as described in our earlier article (Callaghan et al., 1997) with the exception of the duplexing switch circuitry required because of the use of a common receiver and transmitter coil. In our most recent measurements we used a polarising current of 6.0 A, resulting in $B_p = 19.5$ mT, and a gradient current pulse amplitude of 2.0 A resulting in an gradient of 5.28 mT m^{-1}. Polarising pulse durations were 5 s and the delay between experiments was 6 s, kept this long so as to minimise any heating of the sample by the polarising coil dissipation. The gradient pulses had durations ranging from 1 ms to 80 ms (the lower limit being fixed by finite pulse rise time) while the separations ranged from 175 ms to 245 ms, the lower limit being fixed by the need to allow sufficient time for complete decay of the first gradient pulse before application of the 180° af pulse, and the upper limit being determined by signal loss due to T_2 relaxation. A photograph of the *ice sheet probehead* under use at our 1997 site near Mt Erebus is shown in Fig. 4.

5. Experimental results

5.1. Calibration of the diffusion measurement under Antarctic conditions

A key factor in the experiments to be reported here concerns comparison of measured brine diffusivity with known values of equilibrium Brownian diffusivity at the same temperature. Fig. 5 shows results obtained in our laboratory, for the self-diffusion coefficient in 35 ppt NaCl (Antarctic sea water) water held in a capillary to prevent freezing below −1.8°C. The data are consistent with known values of water diffusivity above 0°C and therefore relatively insensitive to salt content.

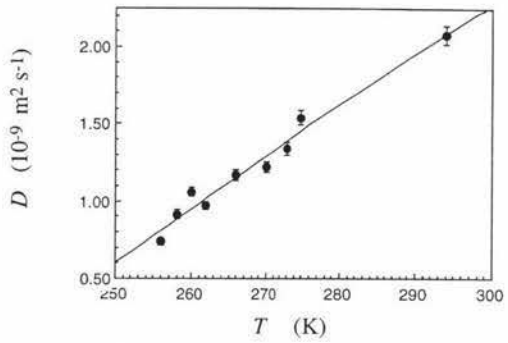


Fig. 5. Self-diffusion coefficient of water molecules in Antarctic sea water, as a function of temperature under thermal equilibrium conditions. These were measured by Pulsed Gradient Spin Echo NMR, in our New Zealand laboratory.

Fig. 6a shows the echo attenuation obtained, in the Antarctic, from a 400 ml water sample using the *extracted core probehead* placed in the polar tent. The water temperature was measured before and after the measurement, the entire data set being collected in a period of 5 min, during which time the water sample cooled by less than 2° and exhibited a mean temperature of 8°C. This sample had the same dimensions (120 mm long and 65 mm diameter) as the ice core samples. The gradient pulse amplitude, g, was 4.5 mT m^{-1}, the pulse duration, δ was varied from 1 to 80 ms and the pulse spacing, Δ, was 145 ms. The slight deviation from exponential decay is a consequence of the finite sample size and the variation in line gradient over that volume. Also shown in Fig. 6a is a theoretical curve based on the known diffusion coefficient of saline water at the same temperature. The agreement is very good.

In the *ice sheet probehead* the very different sample geometry and the facility to orient the gradient coils either vertical or horizontal, necessitated a water sample calibration for both coil orientations, in each case making a line gradient calculation using the known sample and gradient coil geometries and local Earth's field magnitude. The results are shown in Fig. 6b and c, where the experimental echo attenuations are shown alongside theoretical calculations based on the known diffusion coefficient of brine at the corresponding mean temperatures (5.8°C and 2.0°C for the vertical and horizontal cases, respectively).

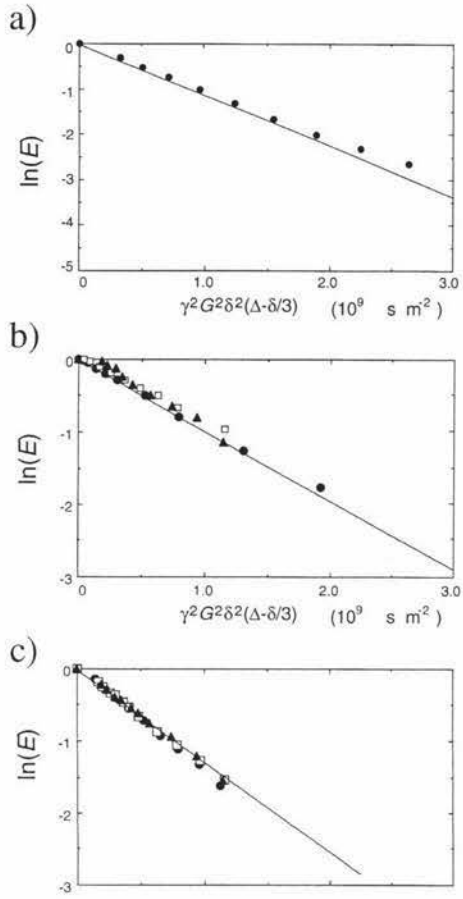


Fig. 6. (a) Comparison of Antarctic experimental data and theoretical calculation (based on the Biot-Savart calculation) for the echo attenuation of a water sample at 7°C for the *extracted ice core probehead*. The data were all obtained using a gradient current of 1.1 A and values of δ ranging from 20 ms to 85 ms with Δ fixed at 145 ms (closed circles). The solid line corresponds to a Biot-Savart calculation involving a line gradient calculation over the actual sample dimensions, using the known diffusion coefficient of water at 7°C. Note that the abscissa scale is based on the usual Maxwell pair high field approximation for G. (b) As for (a) but for the *ice sheet probehead* for a vertically oriented cylindrical water sample at 2°C, and with the gradient coils oriented horizontally. The data were all obtained using a gradient current of 2.0 A and values of δ ranging from 1 ms to 80 ms with Δ values of 175 ms (closed circles), 208 ms (open squares) and 245 ms (closed triangles). (c) As for (b) but for the *ice sheet probehead for* a vertically oriented cylindrical water sample at 6°C, and with the gradient coils oriented vertically.

5.2. Sea ice measurements — 1994 season

Between October 27 and November 5 1994 we carried out Earth field NMR measurements on core samples extracted from the first year ice sheet in the vicinity of Inaccessible Island, near Cape Evans, McMurdo Sound, Antarctica at which the terrestrial magnetic field is approximately 65 μT in magnitude and, vertical in orientation, giving a nuclear precession frequency of 2.72 kHz. These preliminary measurements indicated that we were indeed able to observe a significant signal from the brine and, most importantly, that the brine proton T_2 relaxation times were many hundreds of milliseconds, thus making the signal amenable to the PGSE measurement of diffusion.

5.3. Sea ice measurements — 1995 season

In 1995 we returned to McMurdo Sound, to a location near Cape Evans, Ross Island (77°38.668′S, 166°22.931′E), this time using an improved version of the apparatus in which gradient coils had been incorporated so as to enable the measurement of self-diffusion. In addition a glass dewar was placed between the polarizing coil and the transceiver coils containing the ice core, so as to minimise any temperature fluctuations of the sample. A total of 45 core samples were taken during the 1995 season at intervals of 200 mm from the surface to the ice–water interface at 2200 mm depth. The data reported here were all obtained between 6/11/95 and 9/11/95 during which time the air temperature was stable at around −15°C.

Following extraction from the requisite depth, each ice core was cut to a length of 130 mm and the temperature measured at each end. It was then wrapped in a plastic bag (a material which contributes no NMR signal at the coherence times utilised in our experiment) and inserted in the NMR probehead which we had placed in a standard polar tent. Cores were subjected to NMR measurement within 10 min of extraction from the ice sheet. In none of these samples did we observe brine drainage, all except the deepest cores (i.e., those within 200 mm of the skeletal layer) being dry in appearance. The Free Induction Decay data, used to obtain brine content, was acquired in around 5 min subsequent to

insertion in the probe while the spin echo measurements, used to obtain relaxation time and diffusion information, required another 15 min, following which the core temperature was re-measured. Typical temperature variations over the time of measurement were less than ± 2°C in the case of the diffusion measurements and less than ± 0.5°C in the case of the initial FID data acquisition.

Subsequently core samples were weighed and then melted in order to measure salinity. The known mass of the sea ice core and the NMR Free Induction Decay amplitude from a known mass of water can be used to normalise the ice core NMR Free Induction Decay amplitude so as to obtain a brine fraction value, once a (small) correction is made to allow for the slight difference in NMR sensitivity at the different temperatures of the ice core and the water reference. The ice core salinity and temperature data can be also used to calculate a brine fraction using the sea water phase diagram (Frankenstein and Garner, 1967). The correlation between the calculated brine content and that measured directly by NMR is demonstrated in Fig. 7a. The agreement is good, allowing for the uncertainty in the calculated values due to their sensitivity to small errors in the temperature measurement. Fig. 7c and d show the dependence of salinity and temperature on depth while Fig. 7b shows the depth dependence of the brine fraction below the sea ice surface, as measured by the NMR method. To our knowledge this is the first directly measured profile of brine content in natural sea ice.

In our measurements of self-diffusion on extracted sea ice core samples we observed a dramatic deviation from single exponential behaviour of the spin echo attenuation predicted by Eq. (2), indicating that the brine diffusivity could not be characterised by a single self-diffusion coefficient. Typical echo attenuation plots are shown in Fig. 8a and b, in each case from core samples extracted from the ice sheet at depths of 265 mm and 465 mm below the surface, respectively. These data were obtained from the same core samples used to measure brine content and therefore correspond to the salinity and temperature profiles shown in Fig. 7c and d. The PGSE experiments were carried out using a fixed gradient pulse amplitude of 0.0045 T m^{-1}, with δ values varying from 1 to 80 ms and with the diffusive observation time, Δ, fixed at 108 ms. In each graph we display echo attenuation data obtained with cores oriented vertically (i.e., in the same orientation as they are found in the ice sheet) and horizontally (i.e., orthogonal to their natural orientation). Because the gradient coils in the *extracted core probehead* had a fixed vertical axis, these orientations were necessary in order to measure brine diffusivity in directions parallel and perpendicular to the growth axis (i.e., the direction of the gravitational field).

In each graph we show for comparison the expected echo attenuation for brine at a diffusivity corresponding to the thermal equilibrium Brownian value at the ambient core temperature (i.e., the value shown in Fig. 5). In nearly every case we found that the sea ice, by contrast with a reference water sample, exhibited bi-exponential behaviour, characteristic of at least two diffusion rates, one much faster than equilibrium Brownian, and one much slower. In all the ice core samples examined at every depth the so-called vertical core measurements (i.e., those with the cores their natural orientation and with diffusion measured parallel to their in situ vertical axis) exhibited an initial spin echo attenuation rate consistent with enhanced motion. A summary of the data corresponding to the fast component diffusion rate at differing depths is shown in Fig. 9 along with the corresponding amplitude of the slow component. Note that while we have chosen to describe this rapid motion in terms of an effective diffusion coefficient, the motion cannot arise from molecular self diffusion and can only be due to dispersion caused by advective fluid motion. We shall argue here that such enhanced transport could be due to convective processes associated with the Rayleigh instability.

By contrast the data obtained using the so-called horizontal measurements (i.e., those with the cores orthogonal to their natural orientation and with diffusion measured perpendicular to their in situ vertical axis) were much more variable, as is apparent in the two examples shown in Fig. 8b. Indeed the echo attenuations ranged between mono-exponential behaviour, with diffusivities less than the corresponding free Brownian value, to strongly bi-exponential and similar to the vertical core data.

5.4. Sea ice measurements — 1997 season

A fundamental drawback of the methodology used to obtain the diffusion data presented above is that

the removal of the ice core from its ice sheet environment, and its exposure to temperature changes on the order of 1 or 2°C over the time of the measurements, may severely perturb the brine motion. In the case of the horizontal core measurements the reorientation of the core sample away from its natural alignment may disturb the hydrostatic equilibrium. In the southern hemisphere spring of 1997 we returned to McMurdo sound with the new probehead capable of in situ measurement. A total of 12 in situ measurements were made at an fixed distance from the surface of the ice sheet to the sample centre of 300

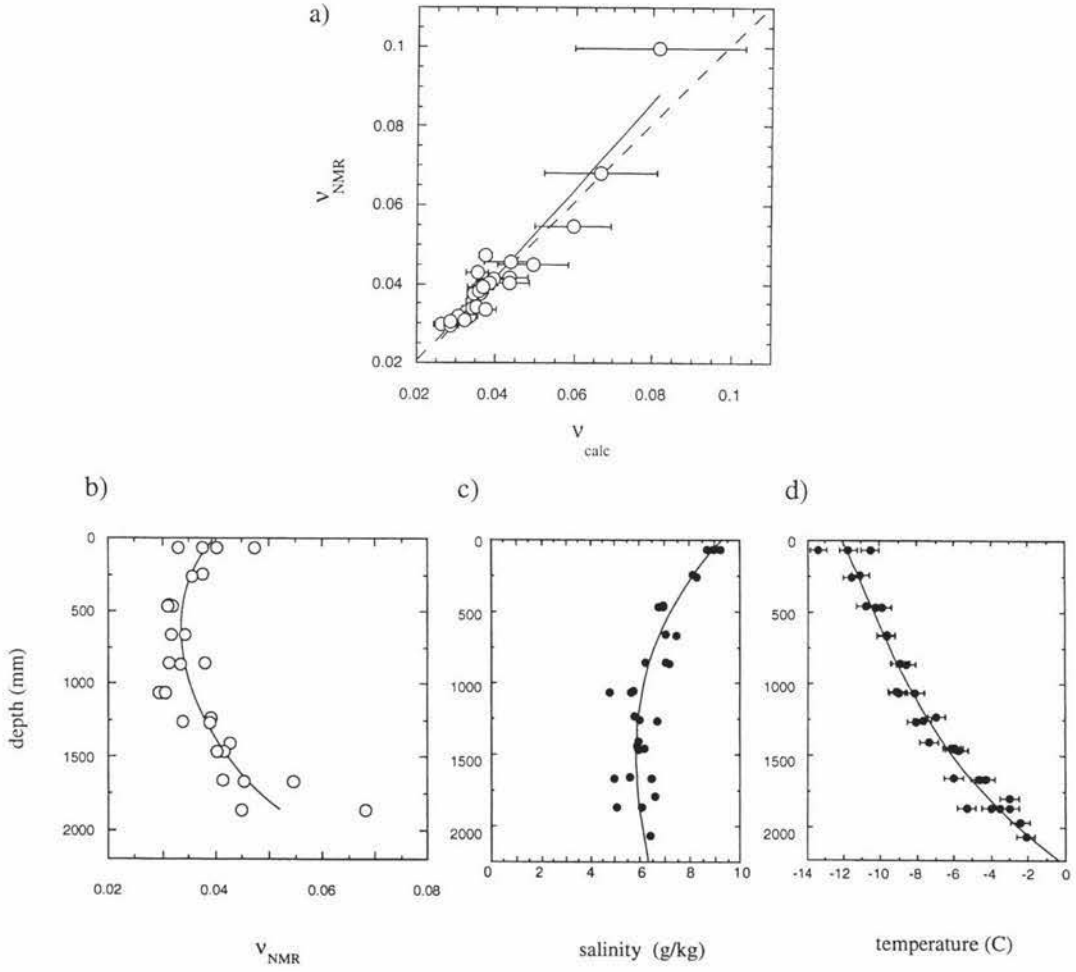


Fig. 7. (a) Correlation of NMR measured brine volume fraction, ν_{NMR} ($m^3_{brine}/m^3_{sea\ ice}$), in Antarctic sea ice from McMurdo Sound (1995) to calculated value based on measured sea ice temperature and salinity. Linear regression of the data (−) yields $\nu_{NMR} = -0.0035 + 1.12\,\nu_{calc}$, with correlation coefficient 0.952. The dashed line of slope 1 is shown for reference. Error bars in ν_{calc} arise from propagation of temperature uncertainty. (Note: of the 45 core samples taken, brine volumes and salinity and temperature were measured in 27, the remainder being used for diffusion studies only.) (b) Brine volume fraction as a function of depth in the ice measured directly by NMR. Note in (a) and (b) data are for 27 core samples for which brine content was measured. (c) Salinity as a function of depth in the ice. (d) Temperature as a function of depth in the ice.

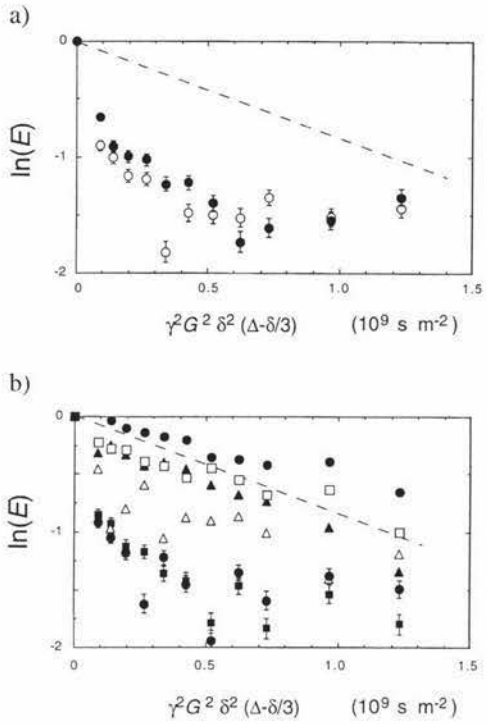


Fig. 8. Dependence of brine water spin echo attenuation on gradient pulse parameter $\gamma^2 g^2 \delta^2 (\Delta - \delta/3)$, for a sea ice sample obtained at a depth of 265 mm (open symbols) and 465 mm (closed symbols) using the *extracted core probehead* in 1995. The gradient direction is respectively parallel (a) and normal (b) to the sea ice vertical axis. The linear dashed line corresponds to the result expected for Antarctic sea water in equilibrium (see data in Fig. 5) at the ambient ice temperature (approximately -10°C. in each case). The PGSE experiments were carried out using a fixed gradient pulse amplitude of 4.5 mT m^{-1}, with δ values varying from 1 to 80 ms and with the diffusive observation time, Δ, fixed at 108 ms. Note that the brine shows evidence of fast and slow diffusion components (bi-exponential decay). The data corresponding to horizontal measurement (i.e., with the cores reoriented at 90° to their natural alignment in the ice sheet) show a much great variation in behaviour than is apparent when vertical diffusion is measured in vertically aligned cores. These data were obtained from the same core samples used to measure brine content and therefore correspond to the salinity and temperature profiles shown in Fig. 7c and d.

mm. The location of our field camp was again off Cape Evans at 77°40.744′S, 166°27.069′E on an ice sheet of 1.75 m total thickness. The data reported here were all obtained between 11/11/97 and 15/11/97 during which time the air temperature was stable at around -5°C. The ambient air temperature was very close to the temperature of the sea ice at the sample depth of 300 mm from the surface (typically between -4.4 and -5.0°C). The salinity, density, air volume and brine volume fractions measured for those samples were typically 5.6 ppt, 915 kg m^{-3}, 0.012 and 0.06, respectively.

The warmer air and ice temperatures presented a major contrast from our earlier work in 1994 and 1995. Nonetheless, as we shall see, some significant commonality in behaviour was observed. The ice sheet probehead has the major advantage that the cores are not disturbed from their environment and remain hydrostatically connected to the ice sheet, the temperature changes over the period of measurement are kept to a minimum (less than 0.5°C in most of the data reported here) and in situ measurements of horizontal and vertical diffusivity are possible. Fig. 10a summarises the echo attenuation plots obtained from a set of five separate measurements in which the gradient coil axis was vertical while Fig. 10b shows data obtained in separate experiments in which the coils were horizontal. Again, the calculated attenuation curves corresponding to equilibrium Brownian motion at corresponding temperatures are shown for comparison. All these data were obtained using a gradient pulse separation time Δ (i.e., diffusive ob-

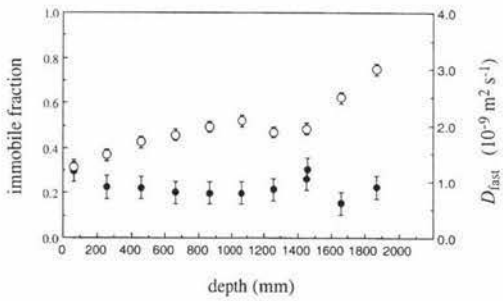


Fig. 9. Depth dependence of the fast component of diffusion (open circles) and the fraction of brine in the slow component (closed circles) for the 1995 data obtained using the *extracted core probehead* and in which vertical diffusion is measured in vertically aligned cores. The D_{fast} and slow component fraction provide a simple parameterization of the curves shown in Fig. 8a. Very little variation with depth is found through the congelation ice until the skeletal layer is approached.

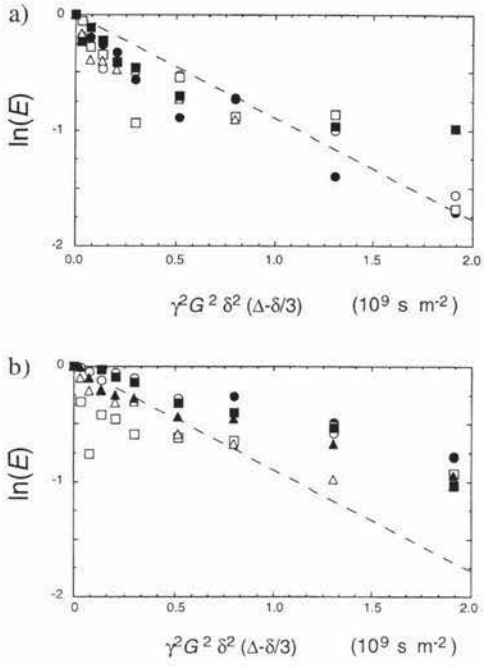


Fig. 10. Dependence of brine water spin echo attenuation on gradient pulse parameter $\gamma^2 g^2 \delta^2 (\Delta - \delta/3)$, for the complete set of sea ice samples obtained at a depth of 400 mm using the *ice sheet probehead* in 1997. The gradient direction is, respectively parallel (a) and perpendicular (b) to the sea ice vertical axis. The linear dashed line corresponds to the result expected for Antarctic sea water in equilibrium at the ambient ice temperature (−5°C). The PGSE experiments were carried out using a fixed gradient pulse amplitude of 5.3 mT m^{-1}, with δ values varying from 1 to 80 ms and with the diffusive observation time, Δ, fixed at 175 ms.

servation time) of 175 ms, this minimum time being longer than used in the extracted core probehead because of the larger size and larger inductance of the *ice sheet probehead* gradient coil set. Fig. 11 shows a vertical diffusion data set obtained using a variety of observation times on a single sample in which the independence of diffusivity on observation time is apparent. This timescale independence was a feature of all the vertical diffusion measurements made using the *ice sheet probehead*, during the 1997 season, a result which contrasts somewhat with the 1995 data.

In order to ascertain whether any differences in the 1995 and 1997 diffusivity data arose principally because of a difference in the sample treatment or because of a difference in the state of the ice sheet, we performed some of our 1997 measurements using the old *extracted core probehead*. The results exhibited the same general features apparent in the *ice sheet probehead* results and so we are able to draw the conclusion that the ice sheet difference is the principal factor influencing the differing degrees of bi-exponentiality seen in Figs. 8 and 10.

5.5. Summary of diffusion experiments

The PGSE diffusion data obtained during the 1995 season using the *extracted core probehead* and those obtained during the 1997 season using both the *extracted core probehead* and the *ice sheet probehead*, are broadly consistent in a number of respects, despite the very different air temperatures and ice sheet temperature gradients which prevailed in those two periods.

(i) All brine diffusivity measured in the vertical direction (and with the ice core naturally oriented) exhibited bi-exponential behaviour in which there existed a significant fraction of brine with dispersive motion much greater than that expected for thermal equilibrium brown conditions, along with a second component in which the brine motion was significantly attenuated.

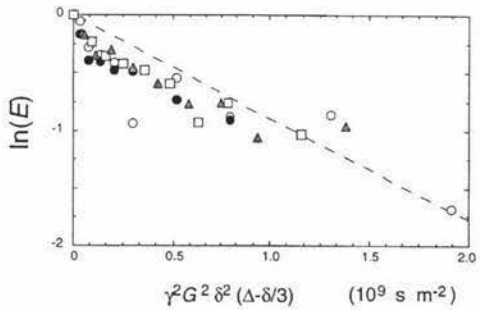


Fig. 11. As for Fig. 10a, vertical motion in *ice sheet probehead* (1997), but for a single sample where the observational time, Δ, is varied between 175 ms (circles) 208 ms (squares) and 245 ms (triangles). The data is approximately independent of observation time.

(ii) All brine diffusivity measured in the horizontal direction (with the ice core naturally oriented) exhibited a variety of behaviours between bi-exponential and mono exponential, in the latter case exhibiting diffusive motion which was significantly attenuated.

(iii) The behaviour in the case of vertical motion measured in 1997 was consistent between the ice core probehead measurements and the measurements in which the core samples were removed from the sheet but examined using their natural orientation.

(iv) The bi-exponential behaviour in the case of vertical motion was consistent within the data set obtained at a given ice sheet temperature gradient (i.e., the separate 1995 and 1997 data sets).

(v) In the case of the vertical diffusivity, the higher temperature gradient ice (1995) exhibits more severely bi-exponential behaviour (i.e., with a weaker amplitude and slower slow component) than in the case of the low temperature gradient ice experienced in 1997.

5.6. Interpretation of diffusion data

It is important to note that the bi-exponential behaviour observed in this work cannot be an instrumental artefact. We regularly checked our instrument using a standard water sample and always observed mono-exponential decay of the echo attenuation consistent with the known value of the water diffusion coefficient at that temperature.

The bi-exponential echo attenuation plots are indicative of at least two populations of water molecules, some with diffusivity much larger (D_{fast}) than the thermal equilibrium Brownian value and some much smaller (D_{slow}). It should be emphasised again that the PGSE NMR measurements give an average over all brine inclusions, and therefore sample a wide range of brine pore shapes and sizes. We shall argue that the population with D_{fast} belongs to brine in pores sufficiently large to allow convective instability but D_{slow} belongs to brine in smaller inclusions.

Let us consider first the slow component. If the inclusion dimensions limit the diffusive motion, in the sense that the wall spacing is smaller that the diffusion length $(2D\Delta)^{1/2}$, where D is the unrestricted diffusion coefficient of the water molecules, then the effective diffusivity would not only be significantly attenuated but would also show a significant decrease as Δ increased. This occurs because molecules are confined by reflections at the boundary to remain within a pore of dimension d then the effective diffusion coefficient is on the order of $d^2/2\Delta$. We do not observe such confinement, as is apparent in the data shown in Fig. 11.

Hence we have the paradox that in the case of the *ice sheet probehead* data the slow component of water appears unconfined yet the diffusion coefficient is less than the thermal equilibrium value in bulk fluid. We attribute this slowing to tortuosity in the path, possibly due to necking of the pores to form occasional narrower "throats" (Niedrauer and Martin, 1979; Weeks and Ackley, 1982) Given a molecular diffusion coefficient D_{slow} of around 10^{-9} $m^2\ s^{-1}$ and observation times on the order of 100 ms or greater, we must conclude that any restrictions to motion due to pore shape tortuosity must occur at dimensions smaller than $2D_{\text{slow}}\Delta$ or 20 μm, a surprisingly small distance. The existence of a slowly diffusing component where the molecular displacements are measured in the horizontal direction, suggests that this motion is impeded by brine inclusion morphology.

It is important to note that the two components of the bi-exponential decay represent water molecule populations which do not significantly exchange over the observational timescale from 100 to 250 ms. Any such exchange would lead to mono-exponential behaviour as Δ increased, with the associated effective diffusion coefficient approaching an average of D_{fast} and D_{slow}. We do not observe such a transition in any of our data. This is consistent with our model that D_{fast} and D_{slow} belong to brine in different populations of inclusions and also implies that the separation of these regions of water population must be greater than the distance $(2D_{\text{fast}}\Delta)^{1/2}$ diffused by the fast component, around 40 μm in 250 ms.

The biggest difference between the horizontal and vertical displacement measurements is the persistent observation of a fast diffusing component in the latter case while this effect occurs only sporadically in the former. The value of D_{fast} is typically between two and four times the equilibrium Brownian value of around 1×10^{-9} $m^2\ s^{-1}$ in the case of the *ice*

sheet probehead data for which the ambient ice temperature was around −5°C. Such enhancement could occur in brine inclusions sufficiently large to fulfil the Rayleigh criterion (Wooding, 1959) via convective displacements driven by buoyancy forces or by the chemical potential associated with salinity gradients. These forces, while directed vertically, must result in circulating brine flow if the water is to remain trapped in the inclusions. Such circulation will result in horizontal components of flow but less consistent and less intense. This is precisely what we observe.

The fact that the extracted ice cores and in situ ice cores obtained in the 1997 set exhibit broadly the same bi-exponential diffusive behaviour for vertical motion indicates just how robust these effects are in the face of temperature excursions. This robustness is consistent with our observation of broadly similar behaviour observed in 1995 for extracted cores obtained at all depths, irrespective of their ambient temperature in the ice sheet or their subsequent positive or negative temperature excursion, provided that temperature change is no more than 1 or 2°C. The indications from our comparison between 1995 and 1997 data that both the proportion and the value of D_{slow} is smaller when the ice core temperature gradient, $\partial T/\partial z$, is larger, is a potentially interesting finding. It suggests that the higher value of $\partial T/\partial z$ is consistent with a greater proportion of the brine under going convection in larger inclusions at the same time that the brine inclusion ''throats'' more effectively inhibit water for the water in the more slowly diffusing populations.

6. Concluding remarks

We have shown that Earth field NMR measurements can be used to obtain good estimates of the brine volumes of sea ice cores quickly and directly. We have also shown that a significant proportion of the brine experiences enhanced diffusivity, an observation which we attribute to brine convection. The existence of such convective processes have been inferred in studies of heat transport in sea ice using in situ thermometry (Lake and Lewis, 1970; Lytle and Ackley, 1996; Golden et al., 1998), and have been extensively studied in laboratory-grown sea ice by Niedrauer and Martin (1979). These latter authors showed that convective transport occurred in brine channels by using a dye introduced into the ice from the skeletal layer. Their explanation for this process was that the prevailing temperature gradient in the ice $\partial T/\partial z$ caused an salinity gradient within the brine inclusion, as a consequence of the thermal equilibrium relationship between salinity and temperature ($S = -17.5730T - 0.381246T^2 - 3.28366 \times 10^{-3}T^3$). Because there exists a strong dependence of brine density on salinity ($\rho_{\text{brine}} = 1 + 8 \times 10^{-4}S$ g cm^{-3}) this in turn implies that the brine density will increase with height in the brine, leading to convective instability should the Rayleigh condition (Wooding, 1959) be satisfied. For a temperature gradient of 0.006°C mm^{-1} (corresponding to the 1995 ice sheet) the convective instability is expected for radii in excess of 0.7 mm while for a gradient of 0.002°C mm^{-1} (corresponding to the 1997 ice sheet) the Rayleigh condition is exceeded for radii in excess of 1 mm. Of course these numbers would need to be appropriately modified in the case of plate-like rather than cylindrically shaped brine inclusions.

Based on this model we might postulate that the two populations of water molecules of the bi-exponential echo attenuation plots whose effective diffusion coefficients are D_{fast} and D_{slow}, respectively correspond to brine inclusions sufficiently wide or insufficiently wide to permit spontaneous convection. This proposition is consistent with our observation that a greater proportion of the brine underwent convective motion in the sea ice with the larger temperature gradient.

Niedrauer and Martin (1979) observed typical convective velocities of between 10 and 250 μm s^{-1} in their brine channels and brine tubes, although their laboratory imposed temperature gradient was quite large, at 0.06°C mm^{-1}. Such a speed represents the motion of fluid at the centre of the inclusion space, but the range of velocities are in fact spread between this value and zero owing to the effect of shear across the inclusion space due to the non-slip boundary condition. Such a velocity spread, when measured using the PGSE NMR method, returns an effective diffusion coefficient, $\langle v^2 \rangle \Delta$. We note with interest that the upper limit of the Niedrauer and Martin convective velocities yields values close to the D_{fast} values observed in our work.

We believe that the use of NMR methods in situ, has the potential to provide important information concerning both sea ice morphology and brine mobility in a relatively non-invasive manner. We see considerable potential for the method as a means of tracking seasonal changes in sea ice microstructure, monitoring sea ice microstructure during mechanical tests aimed at elucidating rheology and resistance to fracture, and as a means of providing ground-truthing information in connection with the dielectric modelling of sea ice, so important to any effective interpretation of passive and active microwave sensing experiments.

Acknowledgements

The authors wish to express their gratitude to Dr. Pat Langhorne of Otago University, Dunedin, New Zealand, for scientific advice and for assistance with salinity measurements. We are grateful to Antarctica New Zealand and the New Zealand Foundation for Research, Science and Technology for logistic and funding support. JDS wishes to thank the US National Science Foundation for Post Doctoral Fellowship support under the International Programme.

References

Abragam, A., 1961. Principles of Nuclear Magnetism. Clarendon Press, Oxford.

Allison, I., 1997. Physical processes determining the Antarctic Sea Ice environment. Australian Journal of Physics 50, 759–771.

Bene, G.J., 1971. Abstracts of 4th Int. Symposium on Magnetic Resonance, Rehovot, Israel 25–27 Aug. 1971. Weizmann Inst. of Science, unpublished.

Callaghan, P.T., 1991. Principles of Nuclear Magnetic Resonance Microscopy, Clarendon Press, Oxford.

Callaghan, P.T., Legros, M., 1982. Nuclear spins in the Earth's magnetic field. Am. J. Phys. 50, 709–713.

Callaghan, P.T., Eccles, C.D., Seymour, J.D., 1997. An earth's field NMR apparatus suitable for pulsed gradient spin echo measurements of self-diffusion under Antarctic conditions. Rev. Sci. Instrum. 68, 4263–4270.

Callaghan, P.T., Eccles, C.D., Haskell, T.G., Langhorne, P.J., Seymour, J.D., 1998. Earth's field NMR in Antarctica: a pulsed gradient spin echo NMR study of restricted diffusion in sea ice. J. Magn. Reson. 133, 148–154.

Doronin, Yu.P., Kheisin, D.E., 1975. Sea Ice. Gidrometeoizdat Publishers, Leningrad (Published by the Office of Polar Programs and the National Science Foundation, Washington, DC by Amerind Publishing, New Delhi, 1977).

Farrar, T.C., Becker, E.D., 1971. Pulse and Fourier Transform NMR, Academic Press, New York.

Frankenstein, G., Garner, R., 1967. Equations for determining the brine volume of sea ice from −0.5 to −22.9°C. J. Glaciol. 6, 943–944.

Golden, K., 1995. Bounds on the complex permittivity of sea ice. J. Geophys. Res. 100 (C7), 13699–13711.

Golden, K.M., Ackley, S.F., Lytle, V.I., 1998. The percolation phase transition is sea ice. Science 282, 2238–2241.

Gow, A.J., Tucker, W.B., III, 1990. Sea ice in polar regions. In: Smith, W.O., Jr. (Ed.), Polar Oceanography, Part A. Academic, San Diego, pp. 47–121.

Gow, A.J., Tucker, W.B., 1991. Physical and dynamic properties of sea ice in the polar ocean. CRREL Monogr. 91-1, US Army Cold Reg. Res. and Eng. Lab., Hanover, NH.

Jeffries, M.O., Weeks, W.F., Shaw, R.A., Morris, K., 1993. Structural characteristics of congelation and platelet ice and their role in the development of Antarctic land-fast sea ice. J. Glaciol. 39, 223–238.

Jeffries, M.O., Schwarz, K., Morris, K., Veazy, A.D., Krouse, H.R., Cushing, S., 1995. Evidence for platelet ice accretion in Arctic sea ice development. J. Geophys. Res. 100 (C6), 10905–10914.

Lewis, E.L., Weeks, W.F., 1971. Sea ice: some polar contrasts, Symposium on Antarctic Ice and Water Masses, Tokyo, Japan, September 1970. Published by Scientific Committe on Antarctic research.

Lin, F.C., Kong, J.A., Shin, R.T., Gow, A.J., Arcone, S.A., 1988. Correlation function study for sea ice. J. Geophys. Res. 93 (C11), 14055–14063.

Lytle, V.I., Ackley, S.F., 1996. Heat flux through sea ice in the western Weddell sea: convective and conductive transfer processes. J. Geophys. Res. 101 (C4), 8853–8868.

Mansfield, P., Morris, P.G., 1982. NMR Imaging in Biomedicine, Academic Press, New York.

Niedrauer, T.M., Martin, S., 1979. An experimental study of brine drainage and convection in young sea ice. J. Geophys. Res. 84 (C3), 1176–1186.

Paige, R.A., 1966. Crystallographic studies of sea ice in McMurdo Sound, Antarctica, US Navy Civil Engineeering Laboratory Technical report, R494, pp. 61–31

Rand, J., Mellor, M., 1985. Ice coring augers for shallow depth sampling, CRREL Monogr. 85-21, US Army Cold Reg. Res. and Eng. Lab., Hanover, NH.

Richardson, C., Keller, E.E., 1966. The brine content of sea ice measured with a nuclear magnetic resonance spectrometer. J. Glaciol. 6, 89–100.

Savidge, G., Priddle, J., Gilpin, L.C., Bathmann, U., Murphy, E.J., Owens, N.P.J., Pollar, R.T., Turner, D.R., Veth, C., Boyd, P., 1996. An assessment of the role of the marginal ice zone in the carbon cycle of the southern ocean. Antarctic Science 8, 349–358.

Shaw, D., 1984. Fourier Transform NMR Spectroscopy, 2nd edn. Elsevier, Amsterdam.

Slichter, C.P., 1963. Principles of Magnetic Resonance. Harper and Row, New York.

Stejskal, E.O., Tanner, J.E., 1965. J. Chem. Phys. 42, 288–292.

Stepisnik, J., Kos, M., Planinsic, G., Erzen, V., 1994. J. Magn. Reson. A 107, 167–172.

Veazy, A.L., Jeffries, M.O., Morris, K., 1994. Small-scale variability of physical properties and structural characteristicsof Antarctic fast ice. Ann. Glaciol. 20, 61–66.

Weeks, W.F., Ackley, S.F., 1982. The growth, structure and properties of sea ice, CRREL Monogr. 82-1, US Army Cold Reg. Res. and Eng. Lab., Hanover, NH.

Weeks, W.F., Assur, A., 1967. The mechanical properties of sea ice, CRREL Monogr. 11-C3, US Army Cold Reg. Res. and Eng. Lab., Hanover, NH.

Weeks, W.F., Hamilton, W.L., 1962. Petrographic characteristics of young sea ice, Point Barrow, Alaska. American Mineralogist 47, 945–961.

Wooding, R.A., 1959. The stability of a viscous liquid in a vertical tube containing porous material. Proc. R. Soc. Ser. A 252, 120–134.