SECTION B

RELAXATION MEASUREMENTS
In general, NMR experiments are of two types: continuous wave type (CW) and pulsed measurements. CW-NMR experiments correspond to measurements in the frequency domain where absorption and/or dispersion spectra are recorded as a function of Larmor frequency $\omega$. Pulsed-NMR experiments are performed in the time domain where the recovery of the macroscopic magnetization from a non-equilibrium value, specifically created by a chosen set of r.f. irradiation pulses, is recorded as a function of time. Each has its advantages as well as drawbacks. However, data in both the domains are related by the Fourier transform, and this fact is exploited in the FT-NMR techniques.

**CW-EXPERIMENTS - PRINCIPLE OF OPERATION**

The imaginary part of the magnetic susceptibility ($\chi''$) at resonance frequencies is connected with resonance absorption of energy by the nuclear spins, while the real part is related to the frequency dispersion. CW experiments are mostly aimed at obtaining the absorption spectrum. In an experiment, this is detected as the change in $x''$ of an inductance coil $L$ in which the sample is kept, while scanning the resonance condition commonly by slowly varying the quantizing field $H$. In crossed-coil type spectrometers an induced e.m.f. in an inductance
coil kept in the \textbf{XY-plane} perpendicular to the irradiating coil is detected and is proportional to $g(\omega)$, the absorption shape function. In single coil detection schemes, the resonant absorption of energy is detected as a change in the output level of a limited oscillator. In both techniques, a modulating field parallel to the Z-axis is applied to facilitate efficient signal detection and processing. The frequency of the modulating field, $\omega_m$, is chosen to be few order of magnitudes less than the Larmor frequency, and the carrier frequency $\omega$ can be filtered out after detection with the help of a low pass filter.

The rate at which the \textit{Zeeman} field can be scanned is restricted by the "adiabatic fast passage" condition \cite{Abragam, 1970}

\begin{equation}
\frac{1}{T_1} \ll \left| \frac{1}{H_1} \right| \frac{dH_0}{dt} \ll \gamma H_1 \tag{B-1.1} \end{equation}

and this condition is satisfied in the case of solids, where $T \ll T$. The intensity of the irradiating r.f. should also be kept at a low enough level to avoid 'saturation'. This condition can be expressed by

\begin{equation}
\frac{\gamma^2 H_1^2}{T_1 T_2} \ll 1 \tag{B-1.2} \end{equation}

The amplitude of the modulating field too is restricted by the adiabatic fast passage condition

\begin{equation}
\omega_m H_m \ll \gamma H_1^2 \tag{B-1.3} \end{equation}

In order to calculate the second moment, $M$, the derivative spectrum obtained due to phase sensitive detection is integrated numerically to
generate the absorption curve and the $M$ is computed by using the relation

$$M_2 = \frac{\int \omega^2 f(\omega) \, d\omega}{\int f(\omega) \, d\omega}$$

In practice, the integral needs to be replaced by a suitable summation to a very good approximation.

**PULSED EXPERIMENTS - PRINCIPLE OF OPERATION :**

The basic principle in these experiments is to create a chosen non-equilibrium state for the magnetic state of the spin subsystem, and then observe, through suitable coupling to the measuring apparatus, the relaxation of the spins to their equilibrium state, in the absence of external r.f. fields. In practice, such an observation involves coupling to the magnetization components perpendicular to the applied static field and the resultant signal is called the free induction decay (FID).

The methodology of the pulsed NMR experiment is appreciated in a more lucid fashion by considering a description of the dynamics of the magnetization in the laboratory and "rotating" frames. In the laboratory frame, a static Zeeman field $H$ is applied along the $Z$-direction and an oscillating r.f. field is applied along, say, the $X$-axis, at the Larmor frequency $\gamma H$. This field can be decomposed into right circularly polarized and left circularly polarized components (Fig. B-1.1a).

Now, if we transform from the lab frame to a frame of reference
{B - 1.1a) Diagram showing the static field $H_o$ and rotating field

— 1.16) Diagram showing the effective field in a frame rotating with angular velocity $\Omega$ about Z-axis in the laboratory frame.
Diagram showing the tipping of the magnetization $M_0$ about the field $H_1$ in the rotating frame.
rotating with an angular velocity $\Omega$ about the Z-axis, the Z-axis magnetic field in this rotating frame is given by $(H - \Omega/\gamma)$ and the magnetic moment of any given spin experiences in this frame an effective field $H_e = (H - \Omega/\gamma)k + H_1i$ (Fig. B-1.1b). Now, when $\Omega = \gamma H_0$, the effective field in this frame is given by $H_1i$ alone and this corresponds to the resonance condition. Thus, at resonance, the magnetic moment in the rotating frame experiences a static field $H_0$ in the XY-plane, along X-axis. For an ensemble of non-interacting spins, a macroscopic magnetization $M$ is present along the Z-axis, which is the sum of the Z-components of the individual magnetic moments. This magnetization tips by an angle $\gamma H \Delta t$ in an interval of time $\Delta t$ about this field (Fig. B-1.2). The time for which the field $H$ is applied can be so chosen that $\Delta \theta = \gamma H \Delta t = \pi/2$, and after this interval the magnetization $M$ will be in the XY-plane, along the Y-axis. This corresponds to the populations in the two levels of energy becoming equal as evidenced by the absence of a net Z-axis magnetization. The r.f. pulse applied for this characteristic time to tip $M$ by an angle $\pi/2$ is called the $\pi/2$ pulse. If this time interval is doubled, the magnetization precesses all the way to the $-Z$ direction, and then this pulse is referred to as the $\pi$-pulse. This condition corresponds to the inversion of the equilibrium populations between the two energy levels.

The free induction decay is recorded as an observable emf induced in a pick up coil due to the time dependent (rotating at $\omega$, and irreversibly decaying to zero due to $T_1$ process) magnetization. Normally the same inductance coil is used both for irradiation and subsequent detection, and a typical FID at this stage would be a decaying (exponentially) sinusoid.
Measurement of the spin-lattice relaxation time $T_1$

By recording the growth of the longitudinal magnetization to its equilibrium value $M$ progressively in time, the relaxation time $T$ can be calculated. Such monitoring of $M$ can be achieved in different ways, and some of the popular schemes are discussed below.

1. Saturation recovery sequence: 
   \[(n/2 - T - n/2 \ldots)\]

   This is the simplest of all the three sequences. It contains a preparation $n/2$ pulse which is followed by 'detection' $n/2$ pulse after a time interval $T$ (Fig. B-1.3a). Assuming that the spin system is at equilibrium to begin with, the first $n/2$ pulse tilts the magnetization $M$ on to the XY-plane thereby creating the non-equilibrium condition. If we wait for time $T$ after this pulse, then the Z-axis magnetization $M$ would have grown to a non-zero value and when the second $n/2$ pulse is applied at this point, it tilts the ambient magnetization on to the XY-plane which induces an FID in the inductance coil.

   From Bloch equations it is seen that

   \[(B-1.5)\]

   \[M_z = M_0 (1 - e^{-T/T_1})\]

   for the initial condition $M = 0$ at $T = 0$, and describes the growth of magnetization with $T$ with the above pulse sequence.
(B – 1.3a) Saturation recovery.

(B – 1.3b) Inversion recovery.

(D – 1.3c) Saturation burst.
2. Inversion Recovery sequence: \((\pi - T - \pi/2)\)

This sequence is shown schematically in (Fig. B-1.3b). The preparation pulse is a \(\pi\) pulse and this causes the magnetization \(M\) to be inverted to the \(-Z\) direction. A time delay \(T\) is applied and the magnetization would have evolved in the \(Z\)-direction from \(-M\) to a larger value in this time, which is followed by a \(\pi/2\) detection pulse which gives rise to an FID whose amplitude depends on the value of \(M\) after its recovery from \(-M\) in an interval of time \(x\). Thus plotting the magnetization \(M\) as a function of \(x\), the delay time, we get an exponential recovery whose time constant is \(T\). If we solve the \(Z\)-component of Bloch's equations with an initial condition that at \(T=0\), \(M_z = -M_o\), we get,

\[
M_z = M_o \left(1 - 2e^{-T/T}\right)
\]

(F-1.6) Fitting the set of \((x, M)\) pairs to this equation will enable us to calculate \(T\). In order to ensure that we satisfy the initial condition, namely \(M = -M\) at \(x = 0\), we must allow enough time to elapse after a given pulse sequence so that the magnetization will recover back to its equilibrium value \(M\) and the application of a \(\pi\) pulse will make \(M = -M\) at time \(x = 0\). In an actual measurement, the sequence with a given \(x\) itself is repeated several times to average out the baseline noise and to improve the signal to noise ratio (S/N) of the FID.
3. Saturation Burst or Saturation Comb sequence :

\[(n/2 - A - n/2 - \ldots - A - n/2) - x - (\text{Burst}) - \ldots\]

This sequence is improvement over the saturation recovery method. The primary drawback of that method was the error incurred due to, either incorrect pulse width or the r.f. field inhomogeneity. This problem is taken care of in this method by the repeated application of \(n/2\) - pulses separated by time \(A\), where \(A\) is chosen so that, \(T < A << T\). Even in the presence of the above mentioned problems, after repeated application of sufficient number of these pulses, the magnetization along \(Z\)-axis is made zero exactly. This bunch of \(n/2\) pulses are referred to as the "saturation burst" or "saturation comb". After the application of a given comb, a time delay \(T\) is inserted after which another burst is applied. As in the case of saturation recovery method, the burst serves a dual purpose in that, the first \(n/2\) pulse of a given burst acts as the detection pulse, apart from the burst itself acting as a preparation sequence. Measurement of \(M\) as a function of \(x\) provides us a recovery curve, which is fitted to equation (B-1.5) and \(T\) is thus calculated.

Saturation burst sequence incorporates some of the advantages of both the previously described sequences. It does not put heavy load on the transmitter, as the duty cycle is kept at a low value by the inherent delay time \(x\) used in the experiment, which is always larger than the duration of the burst. Long waiting times are avoided as each burst prepares the spin system in its initial condition. But it does not match the dynamic range of \(M\) available in the inversion recovery sequence. The
relevant schematic diagram of this sequence is provided in Figure (B-1.3c).

An interesting and useful spin-off of the inversion-recovery method is the 'fly-by' measurement of $T$ using the zero-crossing method. From equation (B-1.6), it can be seen that $M$ becomes zero when

$$\tau_{zero} = T_1 \ln 2$$

If we can find out the time delay $\tau$ for which the magnetization is zero along the $Z$-direction, from that information, $T$ can be estimated. But this method is not as accurate as the other methods due to the inherent ambiguities present in measuring the zero-crossover time delay $T_{zero}$, but this is quite a useful technique in situations where a necessity for quick succession of measurements of $T$ may arise.

The methods given here are the widely used schemes. Numerous other methods are also reported in literature and these methods aim at improving the sensitivity of the measuring technique as well as reduce the experimental time involved [Canet et al., 1975; Freeman et al., 1971; Heatley, 1973; Edzes, 1975; Sezginer et al., 1991; Moore et al., 1993]. Very useful tips and elaborate discussions are available on these matters, in the treatise by Fukushima and Roeder (1981).

**Measurement of spin-spin relaxation time $T$:**

The rate of decay of the FID depends not only on the spin-spin relaxation time $T$ but also on the broadening caused by the Zeeman field
inhomogeneity. Thus, the time constant determined from the FID is to be denoted as $T$ while the true spin-spin relaxation time as $T^*$, and these two are related by

$$\frac{1}{T^*} = \frac{1}{T_2} + \gamma \Delta H$$

where $\gamma \Delta H$ is the broadening of the line caused by the field inhomogeneity.

Spin Echoes:

When a n/2 pulse is applied, $M$ is tipped on to the XY-plane. $M$ precesses in the XY-plane about Z-axis. Due to inhomogeneity in the d.c. field as well as in local fields within the sample, different pockets of spins experience different Larmor frequencies and thus the total magnetization, which comprises of the magnetic moment of these different pockets, phase out to zero. Even though the macroscopic magnetization $M$ becomes zero in the XY-plane, the magnetization within the pockets of spin which experience a reasonably homogeneous field ($H + \Delta H$) takes more time to decay truly irreversibly to zero, which is of the order of the true spin-spin relaxation time $T$. Suppose that these defocused spin pockets are refocused by some means, the non-zero magnetic moment in each spin pocket will add up to give a total non-zero magnetization in the XY-plane once again, and the reappearance of the magnetization in the XY-plane is observable as the spin-echo. This fact was demonstrated by Hahn in his famous paper [Hahn, 1950], in which he has shown that two n/2 pulses applied in succession with an interval of time $T$ separating them produces after time $\tau$ from the second pulse a non-zero magnetization, which is
the Hahn echo. An echo essentially looks like two FID-s superimposed back to back. The more widely used Hahn-echo sequence, which consists of a n/2 pulse followed by a n pulse after time $\tau$ is given in Fig.(B-1.4a). It is much easier to form Hahn echoes in liquids as their $T'$s are large. One can understand the formation of echoes by looking at the evolution of the magnetization in the rotating frame as follows (Fig.B-1.4b). We shall use primed symbols ($X', Y', Z'$) to denote the directions in the rotating frame. Let us assume that in the $Z'$-direction there is a net magnetization $M (A)$. After the $\pi/2$ pulse is applied along the $X'$ direction, the magnetization is tipped on to the $Y'$ axis (B). Now, if we allow for some time $\tau$ to elapse, because of the static field inhomogeneity, different spin pockets having different Larmor frequencies begin to dephase in the $XY$-plane (C). Spins which experience a Larmor frequency greater than $\omega$ ($\omega + \Delta \omega$, say) rotate in one direction with an angular velocity $\Delta \omega$ and spins which are slower i.e. whose Larmor frequency is less than $\omega$ move in the opposite direction. After $x$, a $\pi$-pulse is applied along $X'$ direction and this makes the various spin pockets to precess about the field along $X'$ direction by an angle $n$. For the sake of clarity, in Fig.(B-1.4b), the magnetization of only two spin pockets are shown whose Larmor frequencies are away from $\omega$ by equal amounts $\pm \Delta \omega$. Now, if one waits for the time $x$ after the pulse, the spin pockets refocus on to the $Y'$ axis and this net magnetization gives a spin echo ($\ell$). Now, to measure $T$, we can record the echo amplitude for increasing $\tau$-values and a plot of the echo amplitude versus $\tau$-values is an exponential decay with a time constant given by $T_2$. 

(B — 1.4a) Hahn echo sequence.

(B — 1.46) Echo formation in stages in rotating frame.
Carr-Purcell Sequence:

In this sequence [Carr et al., 1954] a train of π pulses are applied along \(-X'\) direction at intervals \(x, 3x, 5x, \ldots\) etc. after the first \(n/2\) pulse, applied in the \(X'\) direction and echoes will form at the intervals 2\(x\), 4\(x\), 6\(x\) and so on. Because the refocusing pulses are applied along \(-X'\) direction, the echoes form along \(Y'\) and \(-Y'\) direction alternatively. This method has the advantage that unlike Hahn sequence, a single application of this sequence will give us the entire echo amplitude decay envelope from which \(T_\perp\) can be measured readily. But this method still suffers from the limitation that, if the \(n\) pulses are not exact in their width, the cumulative error on the echo amplitude leads to a smaller measured value of \(T_\perp\). This problem can be taken care of by phase shifting the adjacent \(n\) pulses by 180°. The Carr-Purcell sequence is shown in Fig. (B-1.5a).

Carr-Purcell-Meiboom-Gill sequence (CPMG):

A simple modification to the CP (Carr-Purcell) sequence is to apply all the refocusing pulses along \(Y'\) direction after applying the preparation \(\pi/2\) pulse along the \(X'\) direction (Fig. B-1.5b). In this way all the echoes form along the \(Y'\) direction and the sequence with this modification is known as the CPMG sequence [Meiboom et al., 1958]. This sequence can be used for signal enhancement of the echo also.

Measurement of Diffusion Coefficient:

It was shown by Hahn and later by Carr and Purcell that the
(B - 1.5a) 
Carr-Purcell sequence.

(B - 1.5b) 
CPMG sequence.
formation of spin echoes can provide us very useful information on the translational diffusion in liquids [Hahn, 1950; Carr et al., 1954]. In the Hahn echo sequence, as $\tau$ is made very large, the decay of the echo amplitude is not only determined by the spin-spin relaxation process but the self-diffusion of the nuclei also contribute to this decay. This can be understood in the following way. Let us consider a case when diffusion is not present. In this case, after the preparation pulse, in time $T$ a given spin pocket with a Larmor frequency $(\omega + \Delta \omega)$ would have dephased by an angle $\Delta \omega T$ or we can say that this spin pocket has a phase $\Delta \omega T$, in the rotating frame. After the refocusing pulse, in time $T$, this spin pocket accumulates the same phase $\Delta \omega T$ but in the opposite direction and at the end of $\tau$ s the net phase of this spin pocket is zero. This is true of all such spin pockets and therefore all of them coincide at the same point to give rise to the echo of the maximum amplitude. But when diffusion is present, during the time interval $\tau$, the spins move from a region of a given Larmor frequency to a region of another frequency, and thus as the time elapses there will be a change in the phase of any given spin pocket as the quantity $\Delta \omega$ itself is time dependent. Thus the phase accumulated by a given spin pocket before and after the $n$ pulse are no longer same, because of which a complete refocusing of the spin pockets does not take place at the end of $T$ s after the $n$ pulse.

Effect of diffusion on spin echoes can be quantitatively understood by suitably modifying the Bloch equations. [Torrey, 1956]. It can be shown that the effect on the echo amplitude in the case of a Hahn echo sequence is different from that of the Carr-Purcell echo sequence. The Bloch equation modified suitably to include the effect of translational diffusion is given by
where $H$ is an average value of the Zeeman field and $G$ is the field gradient. Let $H = H = 0$ and the $Z$-component of the magnetic field will be the applied field which changes from one point to another and this variation of the $H$ can be expressed as

(B-1.10) \[ H_z = H_0 + (G \cdot \mathbf{r}) \]

where $H_0$ is an average value of the Zeeman field and $G$ is the field gradient. Substituting this expression in equation (B-1.9) and writing $m = M + iM$, the modified Bloch's equation can be written as

(B-1.11) \[ \frac{\partial \mathbf{m}}{\partial t} = i\omega_0 \mathbf{m} - \frac{\mathbf{m}}{T_2} - i\gamma (G \cdot \mathbf{r}) \mathbf{m} + D \nabla^2 \mathbf{m} \]

and introducing $\psi(\mathbf{r}, t)$ by the relation $\mathbf{m} = \psi \exp(i\omega t - t/T)$ this equation modifies to

(B-1.12) \[ \frac{\partial \psi}{\partial t} = -i\gamma (G \cdot \mathbf{r}) \psi + D \nabla^2 \psi \]

We can look at the solution of equation (B-1.12) in the light of Carr-Purcell sequence or Hahn sequence and we shall do the first in this case. In the case of Carr-Purcell sequence, $n$ pulses are applied at intervals $\tau, 3\tau, 5\tau, \ldots (2n-1)\tau$ and the echoes form at intervals $2\tau, 4\tau, 6\tau, \ldots 2n\tau$ and so on. Assuming that the $n$ pulses are phase shifted by $90^\circ$ with respect to the preparation $n/2$ pulse, for an interval $(2n-1)\tau$ to
(2n+l)T, the solution $\psi$ of (B-1.12) in the absence of the diffusion term is given by

(B-1.13) \[ \psi = A \exp\{-i\gamma(G.r)(t-2n\tau)\} \]

and the influence of the term $D \nabla \psi$ can be seen as making the amplitude $A$ time dependent. Substituting this solution back in (B-1.12), we get the following equation for the rate of change of $A$

(B-1.14) \[ \frac{dA}{dt} = -AD\gamma G^2 (t-2n\tau)^2 \]

and integrating this equation within the interval $(2n-1)\tau$ to $(2n+1)x$, we get

(B-1.15) \[ A((2n+1)\tau) = A((2n-1)\tau) \exp\{-&(2/3)D\gamma G^2 \tau^3\} \]

from which it can be deduced that

(B-1.16) \[ A(2n\tau) = A(0) \exp\{-&(2/3)D\gamma G^2 \tau^3 n\} \]

and $A(0)$ is the amplitude of the unattenuated echo at time $x = 0$. The attenuation of the echo amplitude at the time $t=2nx$ is

(B-1.17) \[ A(t) = A(0) \exp\{-&(1/3)D\gamma G^2 \tau^2 t\} \]

In contrast to this result, the attenuation of the echo amplitude in the case of a Hahn echo sequence can be shown to be [Abragam, 1970]
One can see from these equations that the effect of diffusion on the amplitude of a \textit{Carr-Purcell} echo train is seen as an exponential decay as a function of the observation time $t$. But in the Hahn echo sequence, the echo amplitude is attenuated by the factor whose functional dependence on the observation time $t$ is $\exp(-t^3)$. By a careful measurement of the echo amplitude attenuation and a prior knowledge of the field gradient $G$, the Hahn echo sequence can be used as an excellent method for the measurement of the self-diffusion coefficient $D$. In a typical diffusion measurement scheme, the echo attenuation due to the translation of the spins is exploited selectively, by imposing on the static Zeeman field $H$ a linear static field gradient $G$ along the $Z$-direction and using formulae like (B-1.18), the diffusion coefficient $D$ is calculated. In these experiments, it is assumed that the field gradient $G$ which is inherent in the static field is much smaller compared to the applied field gradient $G$. The sensitivity of the measuring technique depends on the highest gradient one can produce and also on how long the observation time interval $t$ can be. The former is limited by the instrumentation being employed and the latter by the spin lattice relaxation time of the given sample. Currently, \textit{self-diffusion} measurement using NMR has become a vast and active field of research in itself and one can find excellent reviews in literature like the one by Karger and others [Karger \textit{et.al.}, 1988].

As a part of the present work, a diffusion measurement set up has been developed by the author by augmenting the existing home-built pulsed NMR spectrometer suitably and the instrumentation details are given in Part-3 of this section (sub-section B-3). The setup is designed to make

\begin{equation}
A(t) = A(0) \exp\left\{-\frac{1}{12}D_\gamma G^2 t^3\right\}
\end{equation}
it possible to measure diffusion coefficients in liquids and liquid crystals. Liquid crystals are interesting anisotropic systems in which the spin-spin relaxation time is much smaller than it is in the case of liquids owing to the strong unaveraged dipolar interactions among nuclei.

There are certain difficulties in making diffusion measurement on this kind of systems using the above mentioned methods. First of all, due to strong spin-spin interaction in these compounds, it is tough to form the spin echoes. Secondly, these systems have high viscosity and it is expected that the diffusion coefficient is relatively small in these systems. This puts a high demand on the largest field gradient one can create, as can be seen from the attenuation factor given in equations (B-1.15) and (B-1.16). It can also be observed that the time of observation has to be made as large as possible to enhance the sensitivity of the measurement and this is restricted by the order of T in the compound. To take care of the first problem, line narrowing techniques are employed like Magic Angle Spinning (MAS) and there are other multiple pulse techniques which selectively average out the dipolar interaction and make the T longer. It is difficult to create d.c. field gradients of very large values, but high field gradients in a pulsed manner can be sustained for small intervals of time (of the order of a few hundred μs). One can analyze the effect of these pulsed field gradients on the echo amplitudes and derive an expression for the attenuation factor from which the diffusion coefficient may be calculated [Karger et al., 1988]. It can be said that most of the spin echo techniques to study diffusion in various systems are all Pulsed Field Gradient (PFG) techniques. The first ever PFG technique was that of Stejskal and Tanner, and the pulse sequence is provided in Fig. (B-1.6)
(B - 1.6) Stejskal-Tanner sequence.
This is a simple sequence in that, apart from the Hahn sequence, two field gradient pulses in between the \( n/2 \) and \( n \) pulses are inserted, and the amplitude of the echo which forms at \( 2T \) is monitored as a function of either the duration of the gradient pulses or the strength of the pulses. The echo attenuation factor is given by

\[
\Psi = \exp \left[ -\gamma^2 D \delta^2 g^2 (\Delta - (1/3)\delta) \right]
\]

Here, \( g \) is the strength of the field gradient pulse and \( \delta \) is the duration of the pulse and \( \Delta \) is the time interval separating the two gradient pulses.

PFG techniques also have their own limitations. The fast rising field gradient pulses may create eddy currents in the NMR probe body and reflections from the pole caps of the magnets also cause residual field gradients which may persist for a considerable amount of time. Any mismatch in the gradient pulse amplitudes leads to attenuation of the echo amplitude which may be construed as due to the diffusion effect. Various other pulse schemes are available in literature which tend to correct these problems [Meerwall et al., 1989; Murday, 1973 Holz et al., 1991; Price, 1991; Heink et al., 1991]. There are methods reported to circumvent specific difficulties pertaining to the systems studied thereof. For instance the scheme suggested by Karlicek and Lowe addresses the problem of large background fields created in powdered samples of considerable bulk susceptibility and circumvents this problem [Karlicek et al., 1980]. Novel schemes of measuring diffusion using RF field gradients is reported [Canet et al., 1989]. Techniques are suggested for measurement of transverse relaxation rates as well as diffusion.
parameters in a fast manner [Moore et al., 1993]. Some of the other methods are available in the following references [Counsell, 1993; Latour et al., 1993; Merril, 1993]. Apart from the new methods of measurements, considerable importance is given to the interpretation of the results as well as theoretical modeling of diffusion processes, in literature [Brooklevinson et al., 1993; Murad et al., 1993; Araujo et al., 1993]. The field of diffusion measurements in micellar, porous and polymeric systems as well as biological systems is a rapidly expanding one and especially diffusion studies in systems with restricted and fractal geometries offers understanding of several new concepts and potential applications as well. As, discussing about the finer details of this interesting branch of NMR spectroscopy is beyond the scope the present work, the reader is referred to the above said articles for further details.
REFERENCES


(B-1.8) EDZES H.T., J.Magn.Reson., 17, (1975) 301


(B-1.11) HAHN E.L., Phys.Rev., 80, (1950) 580

(B-1.12) HEATLEY F., Faraday Trans. II, 69, (1973) 831


(B-1.16) KARLICEK AND I. J. LOWE, J.Magn.Reson., 37, (1980) 75

(B-1.18) MEIBOOM S. AND D. GILL, Rev. Sci. Instrum., 29, (1958) 688


(B-1.22) MURDAY J.S., J. Magn. Reson., 10, (1973) 111


(B-1.26) TORREY H.C., Phys. Rev., 104, (1956) 563

SECTION B

PART 2

PULSED NMR SPECTROMETER

This section contains a brief description of the pulsed NMR spectrometer, employed for measurements in the current work. The details of the construction of various units and their functions are found elsewhere [Venu, 1988].

Fig.(B-2.1) provides a block diagram of the pulsed NMR spectrometer. The spectrometer can be broadly divided into the transmitter, the NMR probe and receiver. The transmitter has to generate the appropriate pulsed r.f. radiation and amplify it to typically 100s of volts and deliver it to the NMR probe. The probe is a matching network made of passive elements L, C and R, and the sample is kept inside the inductance coil. At resonance, the impressed voltage is multiplied several-fold and the nuclei are subject to a high voltage r.f. field. The receiver picks up the very weak signal from the NMR coil, and amplifies it in stages, to be detected by the phase sensitive detector and processed through the low pass filter. Finally, the signal is routed to a signal averager, which averages the signal for improved signal to noise (S/N) ratio.

Transmitter:

1. **Waveform synthesizer**: This provides the continuous sinusoidal voltage at the required Larmor frequency. Output voltage of this
(B. 2.1) Block diagram of the pulsed NMR spectrometer.
unit is few hundreds of millivolts which is gated by appropriate pulse sequence and amplified. A WAVETEK make frequency synthesizer (model No. 2500A) is used here, which has a range of frequencies from 0.2 MHz to 1100 MHz.

2. The power divider: This splits the input from the waveform synthesizer into two parts and routes one to the r.f. mixing circuit and the other to the PSD for detection purposes.

3. Pulse Programmer: This unit generates the required pulse sequences. A detailed description of the \( \mu \)P-based pulse programmer developed as a part of this work is postponed till the next sub-section (B-3). In a simple minded configuration to perform T experiments, the pulse programmer consists of a master clock, two pulse generators and one delay generator. In the case of inversion recovery sequence, the master clock provides the repetition rate of the pulse sequence. For saturation burst, the master clock is not needed as the delay generator produces the trigger at the required repetition rate to the first pulse generator and the output is used to gate the second generator which is in free running mode. This produces a burst of pulses and the number of them in a burst can be modified with the help of the gating pulse and the rate of generation of the second pulse generator. The delay generator is BNC model 7010 and the pulse generators are BNC 8010.
4. **RF mixing circuit**: For this, double balanced mixers manufactured by Mini Circuits company are used. A current driver circuit is fabricated to bias the DBMs, by this method ON/OFF ratio better than 80 dB is achieved [McLachlan, 1982].

5. **Power amplifier (two stages)**:
   - Medium power amplifier of ENI make (model no. 310L) is used, which has a gain of 50 dB and has a bandwidth of 250 kHz to 110 MHz.
   - High power amplifier is a home built circuit with a 3E29 dual tetrode tube, providing a gain of 10.

**Probe**:

Two designs of home built probes were employed. One is the series resonance probe [W.G. Clark, 1973]. The other is a parallel resonance circuit. The parallel resonance circuit has the facility to change the Q factor and has wider bandwidth than the series resonance circuit.

**Receiver**:

1. **Pre-Amplifier**: The preamplifier must have the ability to recover from the overload due to the transmitter pulse and it should also be able to provide reasonable gain to pick up the weak NMR signal. Two commercial units were used: MATEC 252 and
MATEC 253. Model 252 is tunable over a range of 0.5 MHz to 25 MHz and has an overall gain of 30 dB. Model 253 is a wide band amplifier with a bandwidth of 0.5 MHz to 40 MHz having an overall gain of 20 dB.

2. **Wide-band amplifier**: The signal output from the preamp stage is further amplified by this unit. Commercial unit of MATEC make, model No. 625 with a bandwidth of 2 to 200 MHz is used.

3. **Phase Sensitive Detector**: Reference from the power divider is taken in and multiplied with the incoming signal to detect the signal selectively at the Larmor frequency and the phase information is also kept intact. A Hewlett-Packard make DBM is used.

4. **Low Pass filter**: Recovers the FID envelope and filters out all unwanted higher frequencies. This is a home-built \( \pi \)-section filter.

5. **Signal Averager**: A TEKTRONIX make model 2230 signal averager is employed to collect the FID signals sequentially and average out the random noise and thereby improve S/N ratio.
Temperature Control:

Home made P.I.D. circuit using calibrated Chromel-Alumel as the sensor, provided good temperature control up to ±0.5 K. A gas flow type cryostat is used. For low temperature work liquid nitrogen vapours are needed while high temperature work utilizes dry air.
REFERENCES


(B-2.3) VENU K., Ph.D thesis, *University of Hyderabad*, 1987
The basic pulsed-NMR spectrometer was described in the previous section (B-2). The flexibility, versatility and ultimately the utility of the instrument depends on the design of the pulse programmer. The existing instrument uses, as mentioned earlier, two pulse generators coupled through a digital delay generator. The limitation of such an arrangement include restriction to only two pulse sequences, necessity for only manual operation of the delay generator and the lack of facility for automatic data collection synchronous with such a pulse sequencer. In this connection, the fabrication of a microprocessor (μP) based pulse programmer overcoming some of the above problems, and interfacing of the spectrometer to a PC for automatic execution of the experiment are the author's efforts to upgrade the instrument. One of the useful byproducts of such an exercise has been the demonstration of the flexibility of the arrangement to perform, in principle, diffusion measurements with a fairly complex rf and field gradient pulses. This aspect of the instrumentation has been taken up more as a logical extension of the effort to upgrade the facility available to the author.

This Section discusses in two parts, the general pattern of pulse programming including the present work of the author, as well as interfacing of the different subunits to a PC, through GPIB/RS-232-C buses.
PULSE PROGRAMMERS – A OVERVIEW

The utility of the pulse programmer depends on the flexibility of the instrument, and in order to appreciate various considerations in attempting its design, it is useful to list all the relevant features, as expected from experimenter’s point of view.

(i) It should generate pulses of variable widths through independent channels at TTL levels (typically 1 \( \mu \text{sec} \) pulses on the lower side and few milliseconds on the higher side for NMR purposes).

(ii) It generate variable delays between any two pulses. For most of the simple experiments, a resolution of, say, 100 \( \mu \text{sec} \) in the delay time is adequate. The demand on the total range of delay required is sometimes more stringent.

(iii) The pulse programmer should generate adequate number of triggering and gating pulses to serve various other units like the power amplifier, signal averager etc.

(iv) It should have flexible scheme of pulse mixings and provide a desired pulse sequence in a given channel.

The programmer should have a flexible user-interface for changing the various settings as well as triggering the generator into action.
There are a variety of methods reported in literature, for the implementation of pulse programming, each one of them having a unique design philosophy pointed towards a particular set of objectives and requirements of a given spectrometer. Majority of the pulse programmers can be categorized into four broad groups:

1. Single purpose hardware PP-s (or) hardwired PP-s
2. Hardware PP-s with flexible hardware control
3. Hardware PP-s with software control
4. Software pulse generators

1. The first category of pulse programmers are hardwired circuits, designed for a specific pulse sequence. The early models of pulse programmers were of this type [e.g.: Lind, 1972; Franconiet.al., 1970; Shenoy et.al., 1976], and obviously these are not flexible enough for modern purposes.

2. The next generation of PP-s are the hardwired programmers with control capability implemented through hardware [Lind, 1972; Taylor et.al., 1974; Conway et.al., 1977; Aducci et.al., 1977; Lalanne et.al., 1970; Shenoy et.al., 1976; Ellet et.al., 1971] This category of PP-s are also known as modular programmers since several modules each with a specific task are combined together to generate the required pulse sequences. The respective modules have facilities to reorganize the parameters and thus a hardware-based flexible control is possible in these systems. Typically the modules are divided into
the clock module, a series of pulse channel modules and a control module. The clock module has a master clock which may be operating within a frequency range of 10-20 MHz and this module provides the clock inputs to the pulse channels, sets the repetition rate of a given pulse sequence being generated which is achieved through a separate counter -timer circuit in-built and important of all, should start the pulse sequence. The pulse channel modules have the capability to produce pulses of variable width and the width variation may be done in an analog fashion using trigger pulses and monostable circuits or it may be digitally produced with help of a dedicated counter. The clock input from the clock module provides with counting rate and that determines the resolution of the pulse width generation. Other counters within this module, set the delay between pulses and a start trigger for the next pulse channel module in the series. In this type of programmers, invariably, the user interface is through thumb wheel switches or rotary switches. For a fairly complicated pulse sequence, the number of thumb wheels to be operated becomes unmanageable.

3. To alleviate these problems, hardware pulse programmers with software control were designed, and here the parameters, connected with a given pulse sequence, are programmable through a µP-based circuitry or a µP-based development board, the parameters of the experiments being stored in memory. Essentially, the µP replaced the cumbersome process of changing thumb wheel settings to set different parameters. The basic organization of this programmer is similar to
that of a microcomputer. We can identify the memory unit, the control logic circuitry and address counter. The data and instructions are stored in the memory, and the control and logic circuitry reads a given data from the location pointed by the address counter. Several authors have adopted this approach of software controlled hardware pulse generators [Ellet et al., 1971; Matson, 1977; Aducci et al., 1979; Mohr et al., 1983; Ader et al., 1978; Hale et al., 1986; Saint-Jalmes et al., 1982]. One finds slight differences in these approaches also. In one system, only the timing data is stored in the memory but not the instructions. A hardwired program selects these values sequentially to generate a particular sequence [Lapray et al., 1976]. Some of them have the looping characteristic in built, such that, a burst of identical pulses can be produced [Caron et al., 1978]. Aducci et al., have used a Motorola M6802 μP with few peripheral chips like the 4K EPROM, 2K RAM and a console connected to the μP via a serial interface controller chip M6850 and this configuration provides the necessary software support to a microprogrammable logic system called the sequencer and these two units are connected via parallel I/O port controller M6820. All the necessary information about the pulse sequence is stored in the RAM of the microprocessor and the actual generation of the levels, widths, etc. are implemented by the sequencer [Aducci et al., 1979]. An excellent review on the various kinds of pulse programmers is available [Geiger et al., 1980].

The last category of pulse programmers are the so called "software"
pulse generators. Here the time delay between pulses is essentially generated by the finite execution time of the program and this fact is used in the form of setting up "software counters" to count the necessary delay [Wright et al., 1973; Huang et al., 1977]. In some cases, instead of the execution time of instruction providing the delay, the computer clock gives a hardware interrupt to the CPU after the required amount of delay time, and thus a gating pulse synchronous with the interrupt is generated. In all the methods, except the last one, a dedicated, somewhat complex logic circuitry is needed to generate a pulse sequence, and a μP based system or a microcomputer is inevitable to provide the necessary user interface. However, these PP-s have the unique advantage that almost any kind of pulse sequence can be produced with minimal hardware support from outside. Since the pulse programming is effected by the microcomputer, other useful functions of the computer like data collection and manipulation is integrated with the pulse programming.

The pulse programmer developed as part of the current work belongs to the last category. It is built around a stand-alone microprocessor development board which is interfaced via RS-232-C serial interface bus to a PC. Such an interface provides the following flexibility, namely, the pulse programmer can function either in the "slave" mode by receiving all relevant information connected with any pulse sequence from the PC or it can use the relevant pulse sequence data already stored by the user in its own memory and function independently.
The 8085 based Pulse Programmer:

SPECIFICATIONS

LOGIC AND CONTROL

Based on INTEL \( \mu \text{PD}-8085 \) chip.

USER INTERFACE

Hex-pad input and 7 segment display

CONTROL LOGIC

Internal - Software delay counter
External - using \( \mu \text{PD}-8253 \)

INTERFACES

Parallel Interface - using PPI 8255
Serial Interface - USART 8251

PULSE WIDTH GENERATION

Using 74123 buffered monostable circuit and buffered OR gates 7432

PULSE WIDTH CONTROL

Continuous variation with 10 turn potentiometer.

PULSE WIDTH RANGE

1.5 \( \mu s \) to 100 \( \mu s \)

DELAY RANGE

23 \( \mu s \) to = 100 s

ERROR IN DELAY

\( \leq 0.2 \% \)

SYSTEM CLOCK FREQUENCY

3 MHz
PULSE SEQUENCES GENERATED

1. Simple single pulse sequence with a variable time period

2. Inversion recovery sequence

\[(\pi - \tau - \pi/2) -- 5T_1 --\]

3. Saturation Burst sequence

\[(\pi/2-\Delta-\pi/2-\Delta-\ldots) -- (\pi/2-\Delta-\pi/2-\Delta-\ldots) --\]

4. Jeener-Brokaert sequence for \(T_{1D}\) measurements

\[(\pi/2 - \Delta - \pi/4 - \tau - \pi/4) -- \ldots\]

5. Spin locking Sequence for \(T_{1P}\) measurements

\[(\pi/2 - P_{\text{spin-lock}}) -- \ldots\]

4. Hahn echo sequence

\[(\pi/2 - \tau - \pi) -- 5T_1 --\]

5. CPMG sequence

\[(\pi/2 - \tau - \pi - 2\tau - \pi -) -- 5T_1 --\]

6. Stejskal - Tanner sequence (\(P_g\) = gradient pulse trigger)

\[(\pi/2 - \Delta - P_g - (\tau - \Delta - \pi - \Delta - P_g) -- 5T_1\]

7. Liquid crystal sequence

(all these pulse sequences are provided in Fig.B-3.1 and Fig.B-3.2)

Note; 1. Appropriate trigger pulses are provided for the digitizer and amplifier units.

2. Upgradation to any other pulse sequence requires minimal effort as the available codes can be combined suitably to generate a given sequence.
— 3.1) Simple pulse sequences.
THE BLINC SEQUENCE
CONTAINS CARR PURCELL SEQUENCE

WAUGH SEQUENCE

FIELD GRADIENT PULSES

\[(B - 3.2) \quad \text{Liquid Crystal sequence (Blinc et. al., 1971)}\]
OTHER FEATURES:
# Cassette interface for **hardcopy** of assembly codes
# CRT Monitor and ASCII keyboard included - Monitor code provided
# EPROM Assembler-Dissembler of 8085 instruction set included
# EPROM recorder available on board - for hardcopy backup of codes

The Block diagram in Fig.(B-3.3) gives a simple representation of the pulse programmer, and the pulse programmer is based on a minimal number of modular units, resulting in a simple construction.

LOGIC AND CONTROL UNIT:

All the information regarding the pulse sequence is processed in this module. This includes the CPU (**μPD-8085**) with its memory, the programmable timer/counter (**IC 8253**) and a few other passive components. Necessary information on the sequence generation is stored in the memory as data and the algorithm which generates a given pulse sequence is stored in the form of relevant 8085 instruction sets in the memory. In one method, where the sequence is entirely software driven, appropriate delay loops are executed within the program when the code is initialized and executed. The sequence can also be generated with the help of **μPD-8253**. The appropriate number corresponding to the delay to be
(B - 3.3) Block diagram of pulse programmer.
generated is loaded in the counter/counters of the 8253 and at the end of the delay counting, a hardware interrupt is sent to the CPU for a trigger pulse generation. With this method, we can achieve a better resolution for the delay increment.

USER INTERFACE:

A set of "hexadecimal keys", along with some utility functions (like selection of the desired memory location or relocation of the contents of a set of locations to some other memory area), are available on the development board for the user to interact with the \( \mu P \)-system and a seven segment LED display is also provided. Both these functions are controlled by \( \mu PD-8279-5 \) keyboard/display controller.

PARALLEL INTERFACE:

The parallel port of the \( \mu PD \) system is controlled by the programmable peripheral interface (PPI), \( \mu PD-8255 \). The data and control registers are accessible by the host CPU and appropriate trigger pulses at the end of the necessary delay generation is sent through the output ports of 8255 to the pulse width control unit, where the appropriate pulse widths can be set.
PULSE WIDTH CONTROL:

The schematic diagram of the circuitry in pulse width controller is provided in Fig (B-3.4). The task of the pulse width controller is to receive appropriate trigger pulses from the Logic and Control Unit and generate pulses of variable width. It is made of three 74LS123 buffered monoshot circuits which provide six independent channels, and the outputs at each of these channels are further buffered through the OR gates 7432. The OR gate circuits are also used to do the pulse mixing within the pulse programmer itself, for some simple two pulse sequences. Appropriate trigger pulses generated at the parallel port of the Logic and Control Unit is routed into the pulse width controller with help of the edge connector J. These pulses trigger the array of buffered monoshot circuits, and for each trigger input a pulse is generated whose width can be varied by a variable RC combination. In principle, by choosing a set of capacitors C, we can generate the desired large range of pulse widths. But in the present system, we have chosen a fixed value of C with a variable R provided by a 10-turn trimpot to achieve a maximum pulse width of 100 μsec. The circuit has in-built d.c. power supply for all the ICs. The whole circuit is housed in a sturdy metal casing and the outputs are brought out through BNC connectors, so that the desired pulses can be sent to the respective inputs in the spectrometer.
Schematic representation of pulse width controller circuit.
PULSE AND RF MIXING CIRCUIT:

After the pulse sequence is generated, the pulses are used to gate r.f. voltages which are later amplified to high levels to be applied to the NMR probe and this module provides such an r.f. mixing facility. It consists of a combination of double balanced mixers and power combiners (Mini Circuits) (Fig.B-3.6) and a current driver circuit [McLachlan, 1982] which is used to bias the double balanced mixers for the best ON/OFF ratio (a 80 dB) and the circuit diagram of the current driver circuit is shown in Fig.(B-3.5). In Fig.(B-3.6) we have shown two independent channels of mixers, but as can be made out from the figure this configuration can be extended to any number of independent channels of r.f. mixing, in a very simple way. The TTL pulses from the pulse sequencer is fed into the current driver circuit (Fig.B-3.5) and by adjusting the variable $1\Omega$ resistors, the mixing characteristics can be optimized. The outputs like A and B are fed into the DBM module (Fig. B-3.6) and the r.f. is fed to the input "L". The quadrature hybrid unit is shown to indicate the option that, if phase shifting of 90° is needed between two pulses, r.f. outputs of phase 0° and 90° can be tapped from this unit and mixed with the individual pulses. For such applications, these individual pulses are not mixed initially in the pulse width control unit and they are generated separately on individual channels and later mixed with the r.f. in the pulse and r.f. mixing unit, and this feature is taken care of by the appropriate program.
3.5) Current driver circuit to bias the Double Balanced Mixers.
(B - 3.6) Block diagram of pulse and r.f. mixing configuration.
SERIAL INTERFACE:

The serial interface provides the necessary interlink between the pulse programmer and an IBM PC-AT, which is being used as the master controller for the entire environment. All the assembly codes developed for the µPD controller are interfaced with other application programs written in BASICA language in the IBM PC-AT, such that, pulse generation is performed independently by the µPD programmer but it is synchronized with the other instruments in the spectrometer by communicating with the PC through the serial interface. In fact, with a small boot strap code loaded in the µP programmer, the entire assembly code pertaining to a particular pulse sequence can be loaded on to the memory of the µP from the PC-AT via the serial port. The serial port is built around the µPD-8251, which is also programmable from the host CPU. It has the maximum baud rate of 9600 but since data throughput need not be very high for the occasional control messages or information to be exchanged between the computer and µP-programmer, we have programmed it to operate at a low baud rate of 300. For such a small rate, hand shake is not necessary with the computer and this makes the wiring as well as the programming much simpler. In this configuration itself we have found the interface to work without any problems.

It may be instructive at this juncture to look at some programs for generation of some specific pulse sequences. Few of these programs are provided in Appendix-I. For each program a flow-diagram is given tracing the logic of the program, and the corresponding assembly program is
provided next to it. Considering, for example, the simple problem of generating a single pulse sequence, the corresponding flow diagram is given in Fig. (B-3.7). The flow-chart is self explanatory and the steps given in the flow chart are implemented in the form an assembly program derived from the instruction set of 8085, and this programme is provided in (Appendix I). With this program, the PP generates the pulse sequence with a set of tau values stored in specified locations of memory. The pulse sequence is generated ad infinitum and advancing from one tau value to other is done through an interrupt communicated to the processor by the master controller, which is the PC, via the serial port. Here the delay is generated in a separate subroutine. Division of the programs into such general modules is quite desirable, as such routines with general applicability can be readily adopted into other programs where similar functions may be needed. Several other programs were developed as a part of developing the pulse programmer, and a sample of some important programs are all provided in Appendix-I. The programs provided in the appendix are all recorded on cassette for a hard-copy storage. These programs can be either retrieved from the cassette and loaded onto the appropriate memory locations and executed, or they can be loaded by the master controller, the IBM PC-AT, via the serial interface also. Separate BASICA codes are developed, to interface the μP to the PC via the serial communication interface and these programs are provided in Appendix-II, and are discussed under the heading 'SOFTWARE' in the text, later.
Flow diagram of one pulse sequence.
The need for automation of a group of instruments like the pulsed NMR spectrometer for control of the experiment, unaided data collection, besides analysis and display of data etc., is well recognized. But one of the major challenges in achieving this task seems to be to interface different kinds of measuring instruments and make them transfer data/information among them in a reliable fashion. Initially, different manufacturers were defining their own standards of communication and interface between their own instruments so that it was not possible to interface any two arbitrarily chosen commercial instruments for data communication. With the advent of "General Purpose Interface Bus" (GPIB) defining an industry standard for the connection of various measuring instruments and computers, such interfacing of different instruments was made possible.

FEATURES OF GPIB:

The GPIB environment is basically made of instruments which are classified as Talkers, Listeners and Controllers. Talkers are those which only transmit data, Listeners are those which only receive data and the Controller is both a Talker and Listener, and further it controls all the activities in the bus. For a given Controller there can be a total of 15 devices, Talkers and Listeners put together. Each device is connected to the other or to the Controller with an IEEE cable whose maximum length can be 2m only. The recommended standard says
that the total length of the cable must not exceed 20 meters within the entire environment.

Each device is assigned a "primary address" and this address can either be set by dip switches on instruments or one can set the address using software commands sent to the instrument. It is the Controller which decides which device should transmit or "talk" and which are all the devices which ought to receive the message or "listen". There can be more than one Listener in the environment but only one instrument is allowed to be a Talker at a given time. The GPIB is a 24 line bus with 8 dedicated lines for data transfer (contrast the fact that this type of data transfer is bit parallel byte serial in comparison to the serial interface which is bit serial byte serial). It has 3 hand shake lines and 5 bus management lines. It is important to note that all the lines in GPIB are active LOW. This means that, to assert a signal on a given line a device has to make that line 0 V from 5 V.

HANDSHAKE PROTOCOL:

For a reliable data transfer a handshake protocol is needed in the GPIB environment and this facility is provided. A handshake protocol is nothing but an exchange of electrical pulses via dedicated conducting lines between instruments to provide a synchronization of the instruments with one another, for data transfer. Normally a handshake protocol needs at least two lines out of which one is an input and the other is an output line. The device can place a signal on the output
line and wait for acknowledgement from the other device. The moment such an acknowledgement is received, the handshake, for this simple purpose, is complete. After this, successful data transfer can take place on the bus. The handshake lines in the case of GPIB are DAV (DAta Valid), NRFD (Not Ready For Data), NDAC (Not Data ACcepted). The complete handshake sequence for one data byte is shown in Fig.(B-3.8). DAV is the signal controlled by the active bus Talker. It is used to communicate the state of the bus data lines. When this signal is low, the data on the data lines is valid and have had time to settle to the correct logic levels. NRFD is the signal that the active Listeners use to hold off transmission of a byte until all are ready to receive. NDAC is the signal used by the active Listeners to force the Talker to hold data on the data lines until all active Listeners have had time to accept the information. The sequence of events shown in Fig.(B-3.8) for the bus handshake is simple:

1. NRFD goes high, signifying that all Listeners are ready for the next piece of data.
2. The Talker recognizes this by placing the data on the data bus and driving DAV low after waiting for the data to settle.
3. Listeners acknowledge by driving NRFD low.
4. When all of the Listeners have acquired the data, they allow NDAC to go high, signifying their acceptance.
5. The Talker then releases DAV to end its part of the handshake.
6. The Listeners then pull NDAC low to prepare for the next bus transaction.
- 3.8) Handshake protocol in GPIB.
THE BUS MANAGEMENT LINES :

To manage the GPIB environment, the Controller sends several commands to all the devices and these commands can be divided into uniline commands and multiline commands. The uniline commands are simply signals ascertained on the dedicated lines of the bus management. The uniline commands or functionalities of the bus management lines can be seen as follows :

1. **ATN** (Attention) - The ATN line is one of the important management lines. The state of the ATN line determines whether Controller information on the data is to be considered data or a multiline command as described below.

2. **IFC** (Interface Clear) - Setting the IFC line true (low) causes the bus to go to a known state.

3. **REN** (Remote Enable) - Setting the REN line low sends the REN command. This sets up instruments on the bus for remote operation.

4. **EOI** (End or Identify) - The EOI line is used to send the end signal which usually terminates a multi-byte transfer sequence.

5. **SRQ** (Service Request) - The SRQ line is set low by a device when it requires service from the Controller.
One of the important features available in the GPIB standard is the serial polling function. When an instrument or instruments request(s) attention of the Controller, the Controller begins a Serial Polling sequence and gets a status word from each of the instruments which had requested for service. Depending on the information available in the status word, the Controller takes necessary action.

**MULTILINE COMMANDS**: 

The multiline commands are sent by the Controller on the data bus by ascertaining the ATN line. These commands can be further divided into Universal, Addressed and Unaddressed groups.

**Universal Commands**

- **LLO** (Local Lockout) - This command is used to lock out front panel controls on devices so equipped.

- **DCL** (Device Clear) - After a DCL is sent, instrumentation equipped to implement the command will revert to a known state. Usually, instruments return to their power up conditions.

- **SPE** (Serial Poll Enable) - The SPE command is the first step in the serial polling sequence, which is used to determine which instrument has requested service with the SRQ command.

- **SPD** (Serial Poll Disable) - The SPD command is sent by the Controller to remove all instrumentation on the bus from the serial poll mode.
Addressed Commands

Addressed commands are multiline commands that must be preceded by a listen command derived from the device's primary address before the instrument will respond. Only the addressed device will respond to each of these commands.

1. **SDC** (Selective Device Clear) - The SDC command performs essentially the same function as the DCL command except that only the addressed device will respond. Instruments usually return to their default conditions when the SDC command is sent.

2. **GTL** (Go to Local) - The GTL command is used to remove instruments from the remote mode of operation. Also, front panel control operation will usually be restored, if the LLO command was previously sent.

3. **GET** (Group Execute Trigger) - The GET command is used to trigger devices to perform a specific action that depends on device configuration. Although GET is considered to be an addressed command, many devices respond to GET without being addressed.

Unaddressed Commands

The two unaddressed commands are used by the Controller to remove
all Talkers and Listeners from the bus simultaneously. ATN is low when these multiline commands are asserted.

1. **UNL** (Unlisten) - All Listeners are removed from the bus at once when the UNL command is placed on the bus.

2. **UNT** (Untalk) - The Controller sends the UNT command to clear the bus of any Talkers.

Apart from these specific commands, there are other addressed commands which are MLA (My Listen Address), MTA (My Talk Address) and these commands are derived from the primary addresses of the devices.

After looking at the various commands, we can see the typical way by which the Controller will be able to coordinate a byte transfer and in the case of service request, how the Controller carries out a serial poll of all the devices. For data transfer, the Controller first sends an IFC command to clear the bus and asserts the REN line to place the devices in remote operation. After this, UNL and UNT commands are placed on the data bus by asserting the ATN line, and the Controller then places the MTA for the specific device, by placing one by one the MLA command for all the Listeners. During these commands the ATN is line is still asserted. After completion of the task, the Controller unasserts ATN line. Now, the Talker and Listener(s) communicate to transfer data. At the end of all data transfer, the Talker is configured to assert the EOI line, which tells the Controller that the transaction is complete.
and thus the Controller places the **UNT** and **UNL** commands on the line and thus the bus is ready for the next set of data transfer.

When devices have requested for service by asserting the **SRQ** line, the Controller enters into serial polling in the following way.

1. The Controller sets the **ATN** line true.
2. The **SPE** (serial poll enable) command is placed on the bus by the Controller.
3. A device on the bus is addressed to talk.
4. The instrument then places its status byte on the bus to be read by the Controller. The **ATN** line is made false at this point.
5. The Controller then sets the **ATN** line low and places **SPD** (serial poll disable) on the bus to end the serial polling sequence.
6. Steps 3 to 5 are repeated for all the other devices.

**Device Dependent Commands:**

All the commands, discussed so far pertain to the basic control of the bus. Now in order to provide a facility by which the Controller can communicate with a specific device, certain "device dependent commands" are needed. Device dependent commands are string of characters sent by the Controller to a particular device as data. The device interprets such a string as a command and performs a specific function. As examples we can consider two instruments, like the TEKTRONIX 2230 digitizer/signal averager and the **Keithley** 195A digital multimeter. In
the case of the 2230 system, when a string of characters given by "CH1?" is sent on the data bus, the scope responds by placing on the data bus the following information "CH1 VOL:<nl>, COU:AC". The string "CH1?" is interpreted by the digitizer as a query as to what are the current settings in channel 1 of the scope and sends a reply back to the Controller indicating that the VOLTS/DIV setting is n1 and the coupling is AC. In the case of DMM 195A, a string like "R1X" will be interpreted as a command to set the resistance measurement range. Thus, different instruments treat different sets of characters as device dependent commands. As of now, there is no accepted industry standard for the format of device dependent commands but some manufacturers like TEKTRONIX have defined their own Codes and Formats Standard for device dependent commands. Few examples from this standard are considered below.

Each command has what is called a "header" and this header is as self explanatory as possible for the function it performs.

example : INIT

a command which initializes the scope. A header may have an "argument" also which is to be separated from the header with a space.

example : SAVEREF REF4

When argument itself requires another argument, the two arguments are
separated by a colon.

example: ACQ REP SAMPLE

Where the header has multiple arguments, the arguments (or argument pairs, if the argument has its own argument) must be separated by commas, such as

example: DAT ENC BIN,CHA CH2

Multiple commands can be put in one line and each command can be separated from the other with a semicolon.

example: DAT ENC BIN,CHA CH1;WFMPRE

Here DAT is the header of first command which has multiple arguments and WFMPRE is the header of another command.

**GPIB PROGRAMMING IN IBM-PC ENVIRONMENT:**

Though stand alone GPIB Controllers are available from commercial sources, integrating the ability of a GPIB Controller into a computer has obvious advantages. Apart from the controlling ability, the computer's computational and programming power can be put to good use. Standard circuit boards called "Add on" cards are available which can be connected to one of the expansion slots of the IBM-PC, which makes the
PC function as a GPIB Controller. Some of the standard cards available in the market are the PC-II and PC-IIA cards manufactured by National Instruments Inc. In the configuration implemented for the present work, a National Instruments PC-II card was used. This card is treated as any other peripheral device attached to the computer and a device driver need be placed in the configuration table of the computer before booting. The manufacturer provides the necessary device driver and also a high-level language interface program, and by far the most popular high level language being used for this purpose is BASICA. The flexibility of BASICA to perform even machine level operations with ease and the impressive graphics manipulation capabilities makes it one of the ideal languages for GPIB programming. Device drivers in other languages like PASCAL and C are also available.

The BASICA programme:

Appendix - II gives the listing of some of the programs developed to control and perform experiments in the GPIB environments along with the \textit{\textmu}P-based pulse programmer. Every BASICA - GPIB program has a declaration block and in fact this is provided as a separate file along with the driver. In the programs listed, one normally finds this declaration block to be from statements 1 to 6. This declaration block loads a machine level routine known as the "\texttt{BIB.M}" into specific location of the system RAM and this works in cooperation with the BASICA interpreter. This routine acts as the translator between the BASICA statement calls and the machine level language driver. Before one
attempts to write a program, one configures a table of the GPIB environment with a utility called **IBCONF.EXE**. This utility allows one to specify, for each device, all its important characteristics starting from the primary address of the device. Thus, the table gives a realistic information to the device driver about what the devices are, which are connected in the bus. The device driver is made of a number of routines, each one performing a specific task of I/O or control between the Controller and the device(s). The manual gives a complete listing of all the routines. Depending on the level of operation, these routines can also be divided into categories. With the group of routines at the lowest level, we can perform some of the bus management tasks like setting the REN line, for instance. At the highest level, certain routines just take the BASICA variables as operands, which are either used to send a string of data to a device or receive data from a device and the routine takes care of all the handshake formalities and other control tasks to complete the communication automatically.

Immediately after the declaration statements, the functions IBFIND are executed which return into the integer variables like BRDX, the so-called device descriptors, which are derived from the primary address of each device. Even the GPIB board has to be accessed via IBFIND and a device descriptor value is returned in the variable BRD%. After the device descriptors are assigned to each device on the bus, all the functions are accessed through the basica CALL statements and each of these statements have two parameters. The first one is the device descriptor of the particular instrument we are accessing, and the second
one will be either a string variable (e.g. CMD$) or a numerical variable (e.g. vX), which will be taken as the argument for the subroutine functions. The string variable will normally contain the device dependent command strings. For instance, to write a command like "CH1? VOL" onto the 2230, we assign this string to a variable called CMD$, say, and we execute

CALL IBWRT (SCOPE%, CMD$)

The function then completes the task of placing the string on the bus to be read by the scope. Here SCOPE% contains the device descriptor for the TEK 2230 scope. After this function call if we execute

CALL IBRD (SCOPE%, ANS$)

the response to the previously sent query will be read into the variable ANSS. It is a remarkable fact that just to perform I/O between devices, these are the only two function calls needed.

ERROR CAPTURE AND STATUS MONITORING :

When a reasonably complex program is developed with many devices on the bus, it is desirable to have some error capture and status monitoring facilities and the handler provides this facility. For the execution of each function, a status variable IBSTA% is updated. Each bit in the IBSTA% variable being set or reset conveys a specific
condition in the bus. For example, if the data transfer is successfully completed, then the eighth bit is set. If a device has requested for service by asserting the SRQ line, the 12th bit in IBSTA% is set. If an error has occurred, then the 15th bit of this variable is set. The program should be coded in such a way that, after finding an error condition, it must go and examine another variable called the IBERR%. Each bit in IBERR% conveys a specific error, which occurred in the bus. It may be worth mentioning that some of the Controllers provide the facility that the SRQ assertion can be brought into the BASICA program as a hardware interrupt using the "ON PEN ..." statement. Such a facility is implemented in PC-IIA, for instance. Another feature implemented with the handler or driver is the auto-serial polling facility by initiating which the program can be made to automatically poll the devices at regular intervals and collect the status bytes from the respective devices into appropriate variables.

Some of the programs are provided in the Appendix-II. The subunits are configured, and programs are developed such that even in the absence of the µP based pulse programmer, standard experiments like T measurements can be carried out with all the units entirely under the GPIB environment, provided of course they are all GPIB compatible (Appendix -II). One can further develop other useful programs performing different tasks, like retrieving waveforms from the digital storage oscilloscope onto disk files and performing signal averaging to improve the S/N ratio, etc. The GPIB facility developed by the author has been used successfully in other applications like automation of a Field
Cycling NMR spectrometer and pulsed NQR spectrometer, in the same laboratory.

SOFTWARE DEVELOPMENT

Apart from the programs developed in specific reference to the pulse programmer development and the automation environment, an integrated approach was taken to write several other programs to analyze the data collected from specific experiments and provide outputs of them in presentable formats. A list of program names and their categorization are provided in Appendix - III.

The software which was developed as part of this work can thus be categorized under three broad headings. They are:

1. 8085 - Assembly programming
2. BASICA language programming
3. ASYST language programming

Programmes of first type are presented in Appendix-I. BASICA programs are of two types. The first type pertains to the programming for automation of the spectrometer. The second contains programs to extract $T_1$ values from the magnetization recovery data from standard experiments like the inversion recovery or saturation burst, and also non-linear least square fit programs to fit the $T_1$ data as a function of temperature to a specific microscopic model. These programs were written using the grid and gradient search algorithms. A group of programs were
also written to do simple manipulation of data like sorting of a data file and to plot the data in desired formats. Finally, the third type of programs written in the ASYST scientific programming language. ASYST is a commercial software which is made of compact scientific subroutines and it has the complete structure as that of any other programming language. Examples of these programs are available in Appendix-III. We have written programs in ASYST programming environment also, to fit our T data to the microscopic models. Because, it borrows all the salient features of the important programming languages like, FORTRAN, PASCAL and BASICA, the ASYST code is very compact and very quick in running. It works on an IBM compatible PC-AT with 512 Kbytes of base memory and with a math coprocessor like the INTEL 80287.

With this effort by the author on the development of a software controlled pulse programmer and different types of programs to control the experiment, collect and process data, as well as analyze and display relaxation measurements, the available pulsed NMR instrument could be upgraded to perform an automated experiment with reasonable flexibility and convenience in one environment.

**DIFFUSION MEASUREMENT**

As a part of developing assembly level programs for somewhat involved pulse sequences also, programs are developed to effect measurements on translational diffusion based on nuclear spin echo modulations in the presence of pulsed field gradients, (including the
sequence for measurements in liquid crystals [Blinc et al., 1971; Zupancic et al., 1974]). In the process, the necessary hardware (like field gradient coils, power driver etc.) is also fabricated.

For generating large field gradients to enhance the sensitivity of the diffusion apparatus, a circuit which can send large pulses of current to a pair of Helmholtz coils were needed. For this purpose, we have adopted the electrical and mechanical design reported by Karlicek and Lowe [Karlicek et al., 1980], which was originally intended for diffusion measurements in solids with large bulk susceptibility. Fig.(B-3.9) shows the circuit diagram of one of the two current pulsers, which are used in the generation of large magnetic field gradients. Fig.(B-3.10) shows the gradient coil arrangement used to generate the gradients across the sample and Fig.(B-3.11) provides a view of the cross-section of this arrangement. It consists of a cylindrical Teflon former in which, four vertical grooves of 1.5 mm width and 2.5 mm depth are made, which are at 90 to each other. About 20 turns of two independent set of wires (SWG 33) are wound in the opposite directions into the grooves so that, d.c. or pulsed field gradients in the opposite senses can be obtained from these two sets of coils, when appropriate d.c. or current pulses are applied to them. The teflon former has a cylindrical hole coaxial to the axis of the cylinder and its inner diameter is 6 mm. A support glass tube is inserted into this hole in which the sample is kept with the r.f. coil wound on it. Standard signals have been tested in the presence of these field gradient pulses. Applying a field gradient modifies the FID as well as the spin echoes,
\((B - 3.9)\) Current pulser circuit for diffusion apparatus.
(B - 3.10) Gradient coil arrangement.
\((B - 3.11)\) Cross section of gradient coil.
and the signal takes the shape of a Bessel function, approximately. The distance between the adjacent nodes of the wiggles gives us an estimate of the field gradient strength [Murday, 1973]. Our preliminary measurement on water and benzene showed that, we could get a field gradient of about 250 Gauss/cm with an application of 3 Amps of d.c. current.

The d.c. sources used for pulsing the current in the circuit are an array of batteries used in automobiles. The circuit (Fig.B-3.9) consists of an inverting amplifier formed by Q and Q and a Darlington type current amplifier made up of Q and Q. The bias to the Darlington configuration can be varied and in effect the current delivered to the helmholtz coils can be varied by adjusting the bias of the inverting amplifier configuration. The input I is only to trigger the pulser to turn the inverting amplifier on and this is derived from the source of the pulse programmer. Thus it can be seen that the pulse amplifier is operated in its extreme operating regions, namely the cut-off and saturation regions. This circuit was fabricated and tested successfully.

Now it remains to apply the relevant pulse sequence to different systems to measure diffusion coefficients and calibrate the unit for optimal performance. Since the pulse programmer is readily equipped with the necessary pulse sequences to measure diffusion coefficients, both in liquids (like the CPMG and Stejskal and Tanner sequence), as well as liquid crystals (the liquid crystal sequence, refer to Fig.(B-3.2)), the above arrangement is in principle ready for use. However, necessary
effort is still required to standardize and calibrate this set-up for regular diffusion measurements.
REFERENCES


(B-3.3) ADUCCI D.J. AND B.C. GERSTEIN, Rev. Sci. Instrum., 50, (1979) 1403


(B-3.15) LIND A.C., Rev. Sci. Instrum., 43, (1972) 1800


