AN AUTOMATED SAMPLE LINE FOR THE PREPARATION OF O^{18}/O^{16} ISOTOPE ANALYSES FROM WATER SAMPLES

by

PETER WHAITE

B.Eng. (Electrical)

The University of New South Wales, 1973

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in the Department

of

Geophysics and Astronomy

We accept this thesis as conforming to

the required standard.

The University of British Columbia

Abstract

An automated sample preparation line has been developed to equilibrate water samples for determination of their oxygen isotope ratio. Preliminary estimates put the repeatability of the sample preparation methods at approximately $0.04^{o}/_{oo}$, a figure that compares very favourably with the present state of the art.

A noteworthy feature of our sample line, is that temperature control is unnecessary during sample equilibration. Errors arising from non-constant temperature conditions are prevented by simultaneously saving all of the equilibrated gas samples in separate reservoirs when the equilibration reaction is complete.

Several other innovations in sample rack design are also described. These are: a circular rack geometry; an improved, inexpensive, magnetic stirrer design to agitate water samples; a Peltier cooling device to trap water vapour; and the use of standard Pyrex test tubes as equilibration flasks.

The preparation line is highly modular. Up to sixteen racks, each capable of preparing sixteen samples, can be included in the system. Racks may be removed, repaired or modified, and replaced, without disturbing the operation of any other racks in the system. The current configuration is a minimum system with only one rack.

The programming concepts used to control operation of the system are new to this application, and hence are a significant contribution. A multi-tasking executive allocates resources amongst the racks on a priority basis. By using linked list structures, the operating system maximises resource and

processor utilisation, but does not compromise flexibility and modularity. The operator can submit any rack for preparation at any time, and the system could, with sixteen racks, prepare a full load of 256 samples in a day.

A simple handshaking interface has been provided to control the release of samples for analysis. This should make it possible to connect the sample line to any mass spectrometer capable of the automated analysis of carbon dioxide. The user controls sample line operation by commands entered on a teletype keyboard. The command language is deliberately unstructured, and users can type in "natural" English sentences if they wish. All system operations and user sentences are printed on the teletype to provide a permanent record for later scrutiny.

Finally a manual command repertoire has been provided. It allows the operator complete control over any rack. All solenoids, registers, and control lines, can be manipulated on an individual basis from the teletype keyboard.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF PROGRAMS

LIST OF TABLES

LIST OF EQUATIONS

Acknowledgements

The successful completion of large project such as this would not have been possible without the support and encouragement of many people.

Primarily, I must thank my superviser, R.D. Russell. His acceptance of my credentials, and his immediate offer of financial support, after I arrived unannounced, unknown, and unexpected, cannot be forgotten.

The early influences of R.D. Russell, W.F. Slawson, T.K. Ahern, and R.D. Meldrum were very important. The need for a sample preparation line was perceived after separate visits by R.D. Russell and T.K. Ahern to Dansgaard's laboratory in Denmark. The technique of using sample reservoirs arose from discussion of Dansgaard's apparatus by members of the above group. This, and many of their other ideas, were given to me during the initial stages of the project.

Support from the technical staff at the Department of Geophysics and Astronomy was essential as the design of the sample rack reached it's construction stages. In particular K.D. Schreiber and W. Seip gave me the substantial benefit of their accumulated experiences, and pounded many of my impractical designs into something workable.

For the final stages the formation of the Geophysical Instrumentation Group has produced a very conducive and exciting laboratory environment. I am particularly grateful to B.B. Narod and J. Bennest for the discussions with them that led to the development of a new design in magnet stirrers to agitate the water samples.

The U.B.C. Computing Centre has provided continuing, and reliable support. The maintenance of an HDLC link to our mini-computers, and the provision of extensive file and editing facilities, have both made my task considerably easier.

Both G.K.C. Clarke and R.D. Russell were responsible for reading this thesis. I suspect that my writing would have been impossible to read without their many comments, suggestions, and revisions.

During the project, financial assistance was received in the form of teaching and research assistantships, and of a U.B.C. Graduate Fellowship. The entire project was funded by NSERC operating and strategic grants (A720 and G852) given to R.D. Russell. Again, my appreciation for his considerable financial support.

Mostly, my thanks to Petra for her personal support in a new country.

CHAPTER I. INTRODUCTION

1.1 CONTRIBUTIONS OF THIS THESIS

An automated sample preparation line has long been recognised as essential to oxygen isotope analysis. Dansgaard had established such a facility in Denmark by 1975, and recently VG Micromass have realised the commercial possiblities by introducing their model MM5020 CO_2/H_2O equilibration system (Micromass, 1978). The Micromass sample line is a more up-to-date version of Dansgaard's. It features a constant temperature air bath to maintain an even equilibration temperature, an adjustable lateral oscillating motion for mixing the water samples, and a remotely situated "mimic" diagram which may be employed to operate the system. This thesis describes a sample line incorporating new solutions to the problems of CO_2/H_2O equilibration.

Traditionally, equilibrated CO_2/H_2O samples are maintained at a constant temperature whilst awaiting analysis. The extreme temperature sensitivity of the equilibration reaction puts severe constraints on the temperature regulator, and requires the use of extremely stable controllers. In our system equilibrated CO_2 from every sample is simultaneously isolated in a bank of reservoirs. The temperature stability becomes unimportant provided that all samples experience the same temperature <u>history</u> .

Sample fractionation during evacuation of the air space above the water sample is avoided if every sample is pumped equally. Instead of pumping through rate-determining capillaries, the sample line uses circular symmetry to achieve the same effect. The problems of capillary mounting, cleaning, and slow pump down are thus avoided.

The sample preparation system is highly modular. The basic unit is the rack which may, at any time, be connected or disconnected from the system, without interfering with other racks. Modularity facilitates repair and modification of racks with a minimum of disturbance, and allows the system to be extended at will.

Writing the software to run the sample preparation line has been the most satisfying aspect of this project. I have viewed the mass spectrometer as a resource, and the sample racks as users in a multi-user environment. By designing an operating system from this perspective it is possible to queue racks for preparation and analysis at any time, irrespective of the status of other racks in the system. To my knowledge no other automated line operates in this manner. Usually all racks must be loaded with samples before any preparation can begin, and the system is then fully occupied until the last analysis ends.

Operating system techniques in memory management, resource sharing, and task scheduling have been used extensively. The resulting list-structured approach allows programs and tasks to be added, changed and executed with great flexibility and simplicity. Using some of the more primitive aspects of artificial intelligence, a readily understandable command language with an extensive vocabulary has been developed.

Critical areas are provided with software protection against operator interference, the rack processing algorithm checks extensively for forseeable problems, and a task can be scheduled to run periodic system testing procedures. A complete log of system events is output on teletype and should allow correction of unforeseeable conditions as well as providing data for future program development.

An extremely promising contribution is the development of magnetic stirrers to agitate the samples during equilibration. These are of a unique design that uses no moving parts. They can be assembled quickly from inexpensive and commonly available components, and operate at extremely low power levels. It is believed that such stirrers will be valuable to many people for a variety of applications.

Other less radical contributions deserve to be mentioned. A cold trap has been constructed that uses Peltier coolers to remove water vapour from equilibrated gas samples. Cooling is easily controlled using electrical signals, and the constant replenishment of liquid nitrogen is no longer a problem. Finally, but also importantly, an optically isolated interface, coupled with a simple handshaking protocol, minimises the dependency of the sample line on the mass spectrometer.

1.2 <u>THE MEASUREMENT OF THE OXYGEN ISOTOPE RATIO IN WATER</u>

In addition to the normal mass 16, there are two more stable isotopes of oxygen. The masses 17 and 18 occur with abundances of about 0.04% and 0.02% respectively. The variation of the O^{18}/O^{16} ratio in natural oxygen reflects the physiochemical history of the collected sample. In particular this ratio in water and ice samples can be used to trace the origins of the sample, and in the case of ice samples from deep cores, can indicate the paleoclimatic conditions that precipitated the original snow.

Such data is essential to a workable understanding of the dynamics of large bodies of ice and water. In an era of expanded northern development this knowledge can only be beneficial.

1.2.1 Terms used in Isotope Analysis

Because the natural variation of stable isotope ratios is very small it is convenient to report analyses in terms of the difference from an agreed international standard. This is the DEL function (Equation 1.1). The internationally agreed

The DEL value is defined to reflect the small variation in natural isotope ratios. DEL values are usually reported as parts per thousand (per mil) relative to an internationally agreed standard water, V-SMOW.

$$\delta(x/s)=1000(Rx/Rs -1) \text{ per mil } (^{\circ}/_{\circ\circ}) \text{ .... } (1.1)$$

where:

Rx	n_{18}/n_{16} of the unknown sample.
Rs	n_{18}/n_{16} of the agreed standard.
n_{18}	number of O^{18} atoms.
n_{16}	number of O^{16} atoms.

Equation 1.1 Definition of the DEL function.

standard is Vienna-Standard Mean Ocean Water (V-SMOW) but in practice comparisons take place with a local standard and results are corrected to agree with V-SMOW.

The natural range of DEL values in water is typically from $-50^{\circ}/_{\circ\circ}$ in arctic regions to $+20^{\circ}/_{\circ\circ}$ in ocean water at the equator.

1.2.2 Preparaton of Water Samples

The direct analysis of O^{18}/O^{16} in water is impractical due to the polar nature of the molecule. The charged molecules tend to "stick" to surfaces within the mass spectrometer and contaminate subsequent samples. Also, the hydrogen atoms introduce so many mass combinations that the mass spectrum becomes very difficult to interpret.

Instead, the method now generally in use is that described by Epstein & Mayeda (1953) and proceeds in four basic steps.

1. Air is removed from the equilibration vessels.
2. Carbon dioxide is admitted.
3. The water samples are equilibrated with CO_2 at a defined temperature.
4. The equilibrated CO_2 is withdrawn for measurement.

With the original technique, the water was frozen to -78°C before the air was removed. This avoided water loss and isotopic fractionation of the samples. Thawing, freezing, and pumping were repeated to eliminate the possible contaminating effects of gases dissolved within the water.

Roether (1970) described and tested a sample line which eliminated all the time consuming, and labour intensive freezing steps. Using capillaries and controlled shaking he firmly established the feasibility of modern automated lines.

Sample preparation relies on the chemical reaction of carbon dioxide with water to transfer oxygen atoms from the liquid to the gas. Mills and Urey (1939,1940) have studied this reaction in detail and point out that it depends on the pH of the solution (equation 1.2).

The first reaction in equation 1.2 is known as hydration and

The exact equilibration reaction depends upon the pH of the solution.

$$CO_2 + H_2O \rightleftharpoons H_2CO_3 \quad \text{.......} (1.2a)$$

is the hydration reaction and occurs if the pH of the solution is less than 8.

$$CO_2 + OH^- \rightleftharpoons HCO_3 \quad \text{.......} (1.2b)$$

is a bimolecular reaction that predominates as the pH rises above 10.

Equation 1.2 The equilibration reaction.

proceeds with a relaxation time of approximately 1 minute. The alternative bimolecular reaction is about one hundred times slower so the advantage of forcing hydration is obvious. Usually this is not necessary as most natural water is slightly acidic.

The DEL value of the final equilibrated CO_2 will depend on the isotopic ratio of the carbon dioxide, of the water, and on the relative amounts of water and carbon dioxide. Equation 1.3 is simply derived from a consideration of mass balance. When

The final DEL value is a result of the initial del values of both CO_2 and H_2O, as well as the relative amounts of each. A mass balance gives the following relationship.

$$\delta(w/sw) = (\rho+\alpha)/\rho \ \delta(c/sc) - \alpha/\rho \ \delta(c_0/sc) \text{ per mil} \quad \text{.......} (1.3)$$

where:	$\delta(w/sw)$ ..	DEL of the sample relative to standard H_2O.
	$\delta(c/sc)$ ..	DEL of CO_2 equilibrated with the sample relative to CO_2 equilibrated with the standard.
	$\delta(c_0/sc)$.	DEL of unequilibrated CO_2 relative to CO_2 equilibrated with the standard.
	ρ	number of oxygen atoms in the water divided by the number of oxygen atoms in the CO_2.
	α^*........	the separation factor of CO_2 and H_2O in equilibrium. Experimentally measured at 1.046 at 25°C.

Equation 1.3 Relationship between DEL values of initial water and final CO_2.

the same CO_2 is used both as a standard and to equilibrate the water samples, the last term of the above expression drops out.

It should be noted that in such a case the standard water is a virtual standard, and that later correction is made its isotopic composition.

The problems and errors introduced during this stage of the analysis will be discussed later. They include the possibility of sample fractionation, contaminating effects from residual gases, "memory" effects from adsorbed water, the sensitivity of the separation factor to temperature, and the transport of carbon dioxide into a dissolved state in the sample.

1.2.3 Analysis of Samples

After sample equilibration DEL values may be calculated by measuring the relative abundances of O^{16} and O^{18} in the equilibrated gas. The ratio will be almost identical to half that exhibited by the abundances of CO_2 of mass 44 and 46. It can be measured directly in a mass spectrometer.

The mass spectrometer is usually of the McKinney-Nier type (Nier,1947; Nier et.al,1947; McKinney et.al,1950). It features a dual-sided inlet system to facilitate rapid comparison of equilibrated CO_2 against a working reference, a dual collector assembly which allows the mass 44 and mass 46 ion beams to be measured simultaneously, and compensation of the smaller ion current with a portion of the larger one.

The isotope facility at UBC uses a mass spectrometer of the McKinney-Nier type originally constructed by Kollar and Russell (Kollar,1960) for the precise measurement of the heavier lead isotope ratios. Consequently it has higher resolving power and dispersion than is usual in the mass 44 to 46 range and allows the elimination of the mass 45 corrections (Craig, 1957). Since its conversion as an oxygen machine Ahern and Russell have made extensive modifications with a view towards completely automated analysis (Ahern, 1972; Russell & Ahern, 1974; Ahern, 1980). As well as completely reworking the collector, the source, and the measuring system, the original Interdata control algorithm (Russell, Blenkinsop et. al., 1971) has been continually modified, and several hardware additions made to effect increased computer control.

Random fluctuations within the mass spectrometer and its measuring system contribute signifiant uncertainty to the measurement of an isotopic ratio. Much of this uncertainty can be eliminated by alternately admitting gas to the mass spectrometer from unknown and reference gas reservoirs. This generates a series of unknown sample isotope ratios interleaved with a similiar series from a known reference. By interpolating between the data points an average difference in ratio can be derived, and from this a DEL value of the sample relative to the standard.

To avoid systematic errors the gases must flow from the inlet line to the source in the mass spectrometer under identical conditions. Such conditions are met by having the inlet line equalise the mass 44 peak signals of both gases before analysis begins. The inlet line on our system was originally designed and constructed by Ahern (1980). It features a servo controller using two mercury columns to equalise the two peak signals and a rather cleverly designed microprocessor to control valve sequencing. This latter apparatus is capable of admitting and queuing a new sample even though analysis of another may be taking place. A time-sharing arrangement such as this helps optimise sample throughput.

Unfortunately the original configuration of the inlet line proved incapable of automated operation, mainly due to the susceptibility of the controller to electrical interference, and to the presence of many small leaks throughout the plumbing. The inlet line is of crucial importance to this thesis so, in co-operation with R.D. Russell, I rebuilt it in a modular, and therefore more serviceable, fashion. Reliable operation was attained by systematically removing the leaks and by installing decoupling circuitry in the electronics.

1.2.4 Collection of Data

Amplifiers in the collector of the mass spectrometer produce two voltages proportional to the intensities of the mass 44 and mass 46 ion beams. After some conditioning in an analogue circuit, the voltages are measured and digitised by a ratioing digital voltmeter. The digital values produced (known as PHI values) can be related to the isotopic ratio using equations described by Russell & Ahern (1974). An Interdata mini-computer collects about four samples a second from the digital ratio meter, and filters these to produce an average PHI value every eleven seconds.

By examining standard deviations of individual PHI values, and the difference between consecutive ones, an experienced operator can usually judge the quality of data quite well. An analysis would be terminated when enough good data had been collected. To do the same in an automated situation is a much more difficult proposition, mainly because human judgement is somewhat intuitive. Consideration must be given to the problem else data collected from the system may be either useless or redundant.

Ahern (1980) attempted this with a simple algorithm. He calculated DEL values on-line using linear interploation of the PHI values. The error of the measurement could be estimated by dividing the standard deviation by the square root of the number of DEL values used in its calculation. Analysis was terminated when the error fell below a certain preset limit ($0.04^{o}/_{oo}$).

The validity of estimating the measurement error in this way is undisputed but the precision of the analysis is another matter. Like most physical signals the production of DEL values will be a Markoff process (Feller,1968). The DEL cannot be truthfully considered as a random variable so estimates of the mean and error do not accurately reflect the sample being analysed. In other words, there is a significant component of low frequency noise throughout the system, and DEL values derived during a sample analysis can only be considered independent when separated by an impractically long time. Overall improvement in the precision of the analysis can only be improved when the low frequency noise is reduced.

A case can still be made for improving measurement precision. There are often instances when the presence of a random step, spike or glitch makes it unlikely that an algorithm based on simple statistical measures will take an optimum course of action. The removal of bad data is essentially a problem of pattern recognition, and is is still best handled by the human mind.

I have therefore taken the attitude that automated analyses can still benefit from human judgement. The machine has been configured for what it is good at; collecting the data. The same quantity of data is collected from every sample, good or bad. If data is consistently bad then fundamental problems exist within the mass spectrometer and must be corrected before routine analyses can proceed. The human talent for pattern recognition has been left with the user. Data reduction takes place off-line in an interactive environment that enables the user to judge, accept, and reject individual data points.

1.2.5 Data Reduction Methods

Several different methods have been used in this laboratory to derive DEL values from the two PHI series output by the measuring system. Samples analysed for the Polar Continental Shelf Project (Russell & Koerner, 1969) were reduced by fitting a Tchebychev polynomial to the data values. Two similiar curves differing by a multiplicative factor were constructed through the two series. From the curves and the average PHI value of the standard (PHISTD) a DEL value can be calculated from the equations discussed by Russell & Ahern (1974).

Later Ahern (1975) used a method employing cubic splines to fit data points in the two series. Two smooth curves were produced, but were not constrained to be related in any way. PHIDIFF was obtained by calculating the average difference between them, and an estimate of measurement error was made from the standard deviation of PHIDIFF. The linear interpolation carried out on-line (Ahern 1980) works in a similar fashion to the cubic splines, but the data points are connected by staight line segments instead of smooth curves. Ahern quotes negligible error from the approximation.

The methods used by myself are not unsimiliar to Russell's Tchebychev polynomial. Two parallel, but straight,lines are fitted to the PHI values using the method of least squares. A constant difference exists between the two lines and this, along with the average PHI value of the standard, is used to calculate the measurement DEL. Likewise the error is estimated from the deviation of the data points from the straight line. Correction of the DEL values to agree with the international SMOW standard is included in the data reduction package. Each rack analysed contains several known standards amongst the unknowns, (usually four standards and twelve unknowns). Data reduction takes place by the rack so the average measured value of the standards can be used to correct for SMOW. In addition their standard deviation gives the user an indication of the precision of analyses for that rack.

The program used to reduce the data is menu driven and self explanatory. It allows the user to see results and graphically examine the raw data. Suspect points may be flagged as such, and will be ignored in subsequent regressions. The process can be repeated until the operator is satisfied with the quality and validity of the analyses. When ready the results are presented in a printable form to a high quality Zerox printer. The printer output is of publishable quality, and can be sent directly to the owner of the samples.

1.2.6 Precision of Analysis

The error of the final DEL value can be attributed to both preparation and analysis of the sample. In particular the precision or repeatability of the sample preparation stage by current state-of-the-art sample lines should be met or bettered by this research.

The Micromass MM602 serves as an appropriate benchmark for stable isotope mass spectrometers. Recent promotional literature from Micromass defines the internal reproducibilty of the machine as calculated from a series of twelve alternate measurements of isotopic ratio from sample and standard gas. For a CO_2 analysis it was better than $0.017^{o}/_{oo}$ at the two sigma level. Our own machine fares somewhat poorly by comparison. After running 22 isotopically identical CO_2 samples on his newly constructed inlet line, Ahern (1980) reported a precision of $0.03^{o}/_{oo}$ at the one sigma level. Rough tests by myself indicate an even worse figure; probably around $0.15^{o}/_{oo}$. Until the instabilities causing these poor results are located and remedied it will be impossible to test the sample preparation line to a better precision.

Large errors are contributed during the sample preparation stage. Ahern (1975) has estimated that the manual methods previously employed in this laboratory were responsible for approximately $\pm 0.08^{o}/_{oo}$ of the final error. Roether (1970) on the other hand concluded that his line had a reproducibility of better than $\pm 0.03^{o}/_{oo}$ even though he eliminated all the freezing and thawing steps. It is rather surprising, therefore, to find that Micromass will only guarantee a precision of $\pm 0.1^{o}/_{oo}$ for their currently available MM5020 equilibration system (Micromass, 1978). This may be due to Micromass's choice of a constant temperature air bath in which to equilibrate the samples. Roether used a less convenient water bath, but because of its higher specific heat stable temperature control would have been much easier.

The sample line developed through this thesis also "immerses" the samples in air. It should certainly be capable of a ±0.1$^{o}/_{oo}$ reproducibility, but ultimately be able to better a mass spectrometer precision of ±0.03$^{o}/_{oo}$.

1.3 PROBLEMS IN SAMPLE PREPARATION

1.3.1 The Temperature Sensitivity of Equilibration

During the equilibration reaction, atoms of oxygen move freely between the CO_2 and water molecules. Because it requires more energy to move the heavier O^{18} molecules they tend to "gravitate" to the lowest energy state of the complete reaction. The final isotopic ratio of the CO_2 will therefore differ from that of the water as the reaction nears completion. It is then convenient to define a separation factor (α) as the isotopic ratio of the CO_2 divided by the isotopic ratio of the water when the exchange is in equilibrium

As might be expected temperature affects the amount of energy available to move the O^{18} molecules, and varies the separation factor. Such behaviour is precisely what makes stable isotope analysis valuable, but unless the sample preparation temperature is well controlled it is also a major source of error. An outline of the sensitivity of the reaction

Consider a closed system with water and CO_2 in chemical and isotopic equilibrium. A change in temperature will alter the separation factor (α) and the measured DEL value of the equilibrated CO_2. Differentiating equation 1.3 with respect to temperature gives:

$$\partial\delta/\partial T = (1/T)(\partial\alpha/\partial T)(\delta+1000) \quad \ldots\ldots (1.4a)$$

where: α^* .... the separation factor.
δ .... the measured DEL value of the equilibrated CO_2.
and the isotopic ratio of the water is assumed constant.

Staschewski (1964) has derived an empirical relationship describing the variation of α with temperature.

$$\alpha^* = 0.9779 \exp(18.989/T) \quad \ldots\ldots (1.4b)$$

where: T .... is the temperature in °K.

Differentiating (1.4b) and substituting to (1.4a) yields the temperature sensitivity.

$$\partial\delta/\partial T = -18.989\,(1000 + \delta)/T^2 \quad \ldots\ldots (1.4c)$$

For the normal range of DEL values, and for small temperature variations, 1.4c can be approximated at 25°C as:

$$\Delta\delta = 0.21\,\Delta T \quad \ldots\ldots (1.4d)$$

where: $\Delta\delta$ .... the change in Del of the equilibrated CO_2 (per mil).
ΔT .... the change in temperature (°C).

Equation 1.4 Temperature sensitivity of equilibration.

to temperature is given in equations 1.4.

If a permissible variation in the DEL value of a rack of samples is $\pm 0.03^{o}/_{oo}$ then equation 1.4d places a limit of $\pm 0.12^{o}$C on the effective temperature difference between samples. This limit must apply not only in a spatial sense, but also for the time taken to complete the analysis of every sample on the rack. Such stability is relatively easy to attain in a water bath but is much more difficult in air.

If it were possible to analyse all samples simultaneously the severe restriction on time stability could be avoided. As long as the spatial variation is within limits, the DEL values of all samples will change identically and no error will be introduced. Our sample line uses reservoirs to store the equilibrated gas samples simultaneously, with the same effect.

1.3.2 Sample Fractionation During Pump Down

Craig et.al. (1953) give the following relationship for an equilibrium batch distillation of water.

$$\ln(1+\delta)-\ln(1+\delta_0) = (\alpha-1)\ln(f) \qquad (1.5a)$$

where: f .... is the fraction of water remaining.
δ_0 .... DEL value of water at $f=1$.
δ .... DEL value of water after $f-1$ has evaporated.
α^* .... separation factor of water vapour.

Using the approximation that $\ln(1+\Delta)=\Delta$ when Δ is small, then 1.5a can be approximated as:

$$\Delta\delta = -\epsilon^* \Delta m/m \qquad (1.5b)$$

where: $\epsilon^*=1-\alpha^*$ .... Experimentally determined as -9.1 per mil at 25°C.
Δm .... mass of water sample lost to evaporation.
m .... original mass of the water sample.
$\Delta\delta$ .... change in DEL value of the water.

Equation 1.5 Batch distillation fractionation.

Before CO_2 can be admitted to the sample flasks, most of the air must be removed by pumping. It is awkward to freeze the samples on an automated line, so water constantly evaporates as the pressure drops. The resulting isotopic fractionation shifts the DEL values of the samples and becomes a potential source of error.

Using equations for Rayleigh distillation (equations 1.5) a change in the DEL value of a sample can be conveniently expressed as a function of the residual air pressure (equations 1.6). The tolerance of an analysis to residual air is, however, hard to determine. Roether (1970) has allowed that pressures of a few hundred microns of mercury are tolerable when samples were equilibrated with CO_2 at a pressure of 400 torr. His preferred lower level of 20-50μHg was to facilitate rapid transfer of the gas from the sample flasks, by freezing CO_2 at the gas sample tube.

Since this technique is not used on automated lines the higher level is probably quite adequate. At an equilibration pressure of one atmosphere (760 torr), residual air pressures should therefore be less than 1 torr.

Consider a container from which a mixture of air and water vapour is being pumped at F(t) litres/second. Assuming that the partial pressure of the water vapour (Pw) is held constant by evaporation from a mass, m, of liquid water, then the ideal gas laws show that:

$$dm/dt = (Pw.W.F(t))/(62.36\ T) \text{ grams/sec} \quad \ldots\ldots(1.6a)$$

and that:

$$dP/dt = -(P-Pw).F(t)/V \text{ torr/sec} \quad \ldots\ldots(1.6b)$$

Eliminating F(t), integrating, and then substituting into equation 1.5b yields an expression for the change in DEL value of the water that is <u>independent of the flow rate.</u>

$$\Delta\delta = (\epsilon^{*}.V.W.Pw)/(62.36m.T)\ \ln\{Pr/(Pa-Pw)\} \text{ per mil} \quad \ldots\ldots(1.6c)$$

where:

ϵ^{*}	$1-\alpha^{*} = -9.1\%$ per mil.
V	gas volume in litres.
W	molecular weight = 18gms for water.
Pw	partial pressure of water at T°C (23.75 torr @ 298°K)
m	mass of liquid water (grams)
T	temperature in °K.
Pr	final residual air pressure (torr).
Pa	initial pressure above the sample- usually 760 torr.

Equation 1.6 DEL error as a function of residual air pressure.

The actual sample fractionation in reaching this and lower pressures is insignificant. With a 10cc water sample and a 60cc air space, the change in DEL value (at a temperature of 25°C) is $0.008^{o}/_{oo}$, $0.011^{o}/_{oo}$, and $0.014^{o}/_{oo}$ for residual pressures of 1, 0.1, and 0.01 torr respectively (equation 1.6c).

A more serious problem is the lack of symmetry in pump geometry from sample to sample. Conventionally samples are arranged on their rack in a straight line, and are evacuated through tubing from one end. The sample nearest the pump can

If samples are pumped down by a vacuum pump of high capacity, through tubing of uniform cross section, then the volumetric flow rate is well represented by:

$$F(t) = F.P(t) \quad \ldots\ldots (1.7a)$$

where: F .... constant that depends upon the geometry of the tubing.
$P(t)$ ..... time varying pressure above the sample.

Substituting (1.7a) in equation (1.6b), then integrating gives the time taken to reach a residual air pressure, Pr.

$$t = V/(F.Pw)\ \ln\{(Pw+Pr)(Pa-Pw)/(Pa.Pr)\} \quad \ldots\ldots (1.7b)$$

where: All variables were defined in equations 1.6.

Consider two samples pumping at unequal rates, F_1 & F_2, where $F_1 > F_2$. Let the times taken to reach the same residual air pressure be t_1 & t_2 respectively. Under these conditions, equation 1.6a places an upper bound on the additional mass lost from sample 1.

$$\Delta m = W.F_1.Pw.(Pw+Pr)/(62.36\ T)\ (t_2-t_1) \text{ grams} \quad \ldots\ldots (1.7c)$$

Substituting appropriate values from 1.7b & 1.7c into equation 1.5b places an upper limit on the final difference in DEL values between the samples as a function of the ratio of the flow rates.

$$\Delta\delta = C.(Pw+Pr).\ln\{(Pw+Pr)(Pa-Pw)/(Pa.Pr)\}.(F_1/F_2-1) \text{ per mil} \quad \ldots (1.7d)$$

where: F_1 .... conductance of the tubing through which sample 1 is being pumped.
F_2 .... tubing conductance for sample 2:
and $C = \epsilon^*.V.W/(62.36m.T)$

Note that this estimate of the upper bound converges to the true change as the residual air pressure drops.

Equation 1.7 DEL error due to mismatch in tubing conductance.

then have an effective tubing conductance well over an order of magnitude greater than those at the far end. By considering the simple configuration outlined in equations 1.7 the difference in DEL values of two samples, due to a mismatch of tubing conductances, can be expressed as a function of residual air pressure. Figure 1.1 presents equation 1.7d graphically, for the conditions used previously.

If the mass spectrometer is capable of a $0.03^{o}/_{oo}$ precision then sample error due to pump-down fractionation should be less than $0.01^{o}/_{oo}$ to be considered negligible. The tubing conductances of every sample must then be matched within a

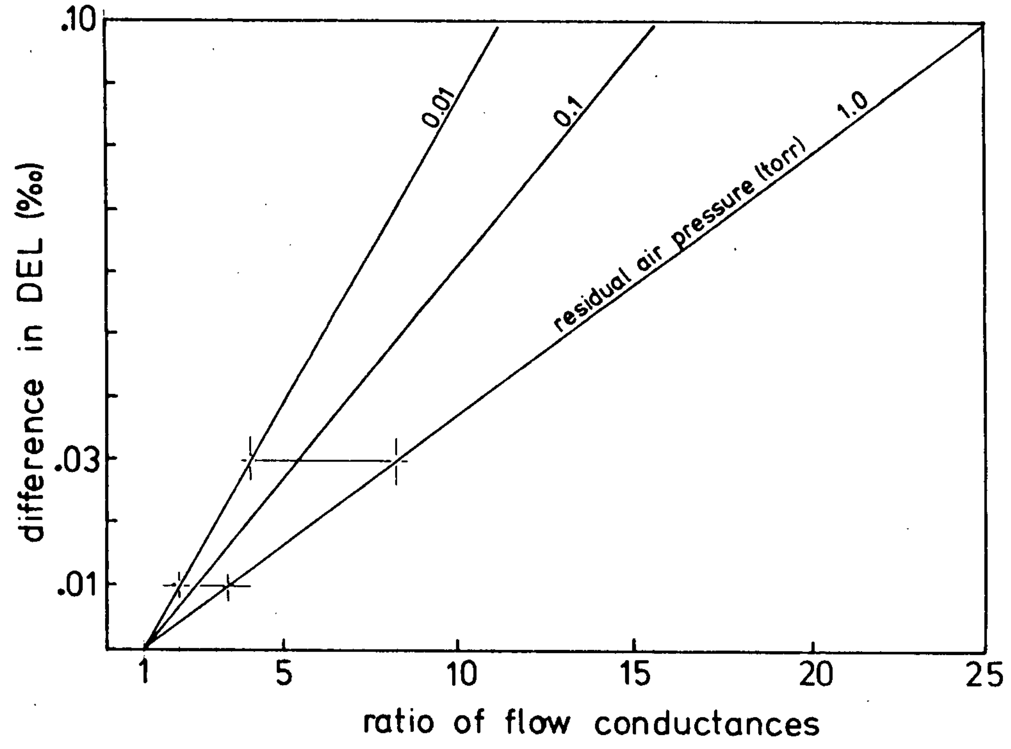


Figure 1.1 DEL Error Due to a Mismatch in Tubing Conductance

factor of 2 to 3 depending on the final residual air pressure.

In the conventional linear rack such a match is possible only when the side arms to the samples are made much more restrictive than the connecting line. Because the volume of the connecting line must be kept low to give adequate CO_2 pressure at the mass spectrometer, the only choice is to pump the samples through small bore capillary tubing. These are easily matched within 20%, and make sample fractionation errors insignificant.

Capillaries do have some practical disadvantages. The fine bore increases the possibility of blockage, and necessitates their frequent removal for cleaning. This increases the system maintenance cost, decreases throughput, and presents some tricky problems of mechanical design. If rack geometry is restructured around a circularly symmetric form the use of capillaries can be avoided. Samples can be pumped equally from a central point and conductances are easily kept within 20%, using standard tubing whilst maintaining a low pump-line volume. If necessary the pump rate of all samples may be controlled using a single adjustable leak between the pump and the rack.

1.3.3 Equilibration Time

The most time-consuming step of the sample preparation is that taken for completion of the equilibration reaction. Minimising the equilibration time will decrease the rack turnaround time. Depending on the particular system configuration this either allows more samples to be run during a working day, or decreases the number of racks required to optimally utilise the mass spectrometer.

Three factors influence the equilibration time. The first has been previously discussed (section 1.2.2) and is the rate at which the equilibration reaction proceeds. At a pH of less than 8 the hydration reaction has a time constant of approximately 66 seconds. As only the dissolved CO_2 can react with the water sample, the gas in the equilibration flask must be exchanged several times before equilibration of all the CO_2 is complete. Therefore, the total reaction rate depends on the rate at which CO_2 transfers across the water surface, and the ratio of the volume of CO_2 dissolved in the water sample to the total volume of CO_2. The volume of CO_2 in the water is the product of the volumetric solubility (0.98 cc of CO_2 per cc of H_2O at STP) and the sample volume.

The determining factor is the rate at which dissolved CO_2 can be transferred from the surface into the bulk water sample. Roether (1970) has investigated equilibration time as a function of shaking frequency on a rack of samples. He notes that this remains rather high at low frequencies, but that it drops rapidly when the water sample begins to resonate within its container, and shows no furthur decrease after this.

Samples are most commonly agitated by shaking or oscillating the complete rack. Not only is this mechanically complex, but the need for flexible vacuum couplings must cast doubt on the ultimate reliability.

Another contribution of this thesis is the first use of magnetic stirrers. These generate no heat, are silent and vibration free. Teflon encased stirrer bars, placed in the samples before connection to the rack, generate a vortex that constantly replenishes the surface layer with a complete lack of sample splash.

1.3.4 Contamination and Mixing

When CO_2 is transferred into a reservoir, sample container or into the mass spectrometer inlet line, any remnant gas will mix with it, and alter the DEL value. The error introduced is a function of the difference in DEL values, and of the ratio of the gas pressures. The calculations for conceivable situations in this system, though straightforward, are tedious, and are not reproduced here. Instead it can be stated that using a simple rotary vacuum pump to reduce pressures to 0.01 torr completely removes contamination as a source of error.

The same argument applies to cross mixing of water samples during pumpdown. Even if allowed to mingle freely, the quantity of water in gaseous form provides insignificant contamination of the remaining liquid. No special precautions need be taken to prevent back mixing between water samples.

1.3.5 Water Adsorption

The least understood source of error is due to water adsorption on the internal surfaces of the sample preparation line. In developing his concept of adsorption, Langmuir (1918) had the following explanation for the phenomena.

> "The atoms forming the surface of a solid are held to the underlying atoms by forces similiar to those acting between atoms inside the solid. From Bragg's work on crystal structure, and from many other considerations we know that these forces are of the type usually classed as chemical. In the surface layer, because of the asymmetry of the conditions, the arrangement of atoms must always be slightly different from that in the interior. These atoms will be unsaturated chemically and thus are surrounded by an intense field of force."

The polar water molecule will be strongly attracted by the surface forces of the solid. Even at pressures as low as 20μHg, experimental measurements show the presence of a monolayer of water molecules on glass surfaces (Frank,1929). It is convenient to consider the adsorbed water as a solid, with a correspondingly low vapour pressure.

Unfortunately it is still possible for the adsorbed water to re-equilibrate with CO_2 passed over the surface. Ahern (1975) found it necessary to prevent water vapour entering his preparation line during the transfer of the equilibrated CO_2 from above the sample. When this was not done, the DEL value of that sample was unreliable. Roether (1970) has also noted such an effect and quotes its magnitude as 0.05±0.05% of the difference in DEL value from the previous sample. He attributed this to water adsorbed on the manifold and stopcock sidearms of his apparatus; and to its re-equilibration with the next sample at a different temperature (approximately 3.5°C higher).

Obviously there are too many factors to attempt an analytical appraisal of the memory effect. If Roether's figures are an indication, it is not a serious problem. Even in the unlikely possibility of a difference in DEL of 50°/oo between consecutive samples the error introduced is at most 0.05°/oo. Extrapolations to other systems are, however, difficult to make. Traditionally accepted techniques scrupulously avoid the presence of water and water vapour throughout the sample line. This is not the case for this sample line, nor for Dansgaard's, and the possibility that memory effects could introduce significant error is an uneasy one to live with.

If necessary, a better compromise is to flush common areas with dry air between analyses. This could prove effective in removing the adsorbed film, but could delay the rate at which samples can be released to the mass spectrometer and reduce throughput.

1.4 ISSUES IN AUTOMATION

The mechanical, repetitive nature of automation might alienate the human viewpoint, but it does allow a significant increase in the accuracy of routine oxygen isotope analyses. In this instance it also replaces an increasingly uninteresting chore, and frees personnel for the interpretive aspects more suited to their abilities. Despite its advantages, automation brings its own problems, and requires a careful examination of the application. This section outlines both the possibilities, and the inherent difficulties.

1.4.1 Advantages of Automation

Slight alterations in the absolute isotopic ratio of a sample are inevitable by the time the ratio is measured. Comparison with a known standard that has been subjected to identical processes, allows these small changes to be measured and corrected. Automation has the advantage that it imposes a rigid repeatibility to the whole process. The confidence that measured changes in the standards are a true measure of those ocurring in unknown samples must be significantly greater.

The development of microprocessor technology has made it possible to implement time-sharing systems in specialised situations. Executing a complex sequence of instructions in microseconds allows the microprocessor to perform a large number of jobs simultaneously. By providing a sufficient number of sample holders it becomes possible to admit samples to a mass spectrometer on a continuous, twenty-four hour basis.

The skill required of human personel to perform manual analyses is not insignificant. Operators must be trained in laboratory techniques, and have experience and knowledge of mass spectrometry. It has proven difficult to find and retain staff for what is essentially a monotonous job. Automating sample preparation allows the laboratory techniques to be de-emphasised, so that the scientific challenges can receive greater attention.

A well designed automated system can be built to exhibit intelligence and adaptability. It can be designed to run self check procedures on its vital organs; thus to document, and possibly rectify, problems that would otherwise go unnoticed. The flexibility possible with a microprocessor-controlled system enables strategies to be developed, implemented and tested in response to the changing needs and criteria of the analytic procedures.

1.4.2 The Disadvantages of Automation

The replacement of manual methods by automated ones is a course fraught with difficulty and misunderstanding. The constant and loose references to intelligent systems are more indicative of wishful thinking than anything real. Engineering problems created when interfacing to an often unpredicatable world necessitate a re-appraisal and evaluation of design tradeoffs and techniques.

Any automated system will be partially blind. The designer must balance the advantages of increased sight against the cost and overheads require to obtain and analyse the extra information. Inevitably, the probability of occurrence of some problems is not large enough to warrant the provision of a sensor to detect it. Should such a problem occur the assumptions upon wich the system is based become invalid but the system will continue to operate.

No matter how superficially clever a system appears on the surface it is never capable of intuitive, and usually incapable of adaptive, decisions. Automation cannot usually deal with problems that the programmer has not contemplated, but list structured languages (eg LISP) can support programs of surprising adaptability. These are designed in such a way that program and data are indistinguishable. It is possible to generate and execute new programs in response to the changing state of the system.

Automated systems can be extremely uncommunicative. Unless provision is made to indestructibly record relevant system operations, faults become very difficult to diagnose, and even to repeat. A system that is too verbose is just as bad.

A grey area that is difficult to define and implement is the provision of operator intervention. On one hand it is tempting to give the operator absolute overriding priority and a large degree of manipulative power. On the other hand, parts of the system that the operator is not properly concerned with should be protected from inadvertent manipulations.

There is, finally, the issue of understandability. Unless any operator finds it easy to understand and use the system, then its introduction will simply result in the replacement of a trained laboratory technician, with a trained computer technician. Again the most promising approaches to these problems are found in the field of artificial intelligence. Program/structures have been developed that can interpret natural English sentences, infer the intended meaning, act upon it, and reply...... in English (Winogrand,1972).

CHAPTER II. INNOVATIONS IN THE MECHANICAL DESIGN

2.1 ORGANISATION

2.1.1 Innovations in Rack Construction

This chapter presents an overall description of the mechanical design of the sample preparation line but emphasises those aspects that make our equipment unique. Specifically there are three major innovations: the use of equilibrated gas reservoirs, a circular rack geometry, and implementation of a new design in magnetic stirrers.

The mechanical organisation can be clarified by delineating three functional areas. These are associated with sample preparation, the mass spectrometer, and general services. The system takes form by interconnecting hardware associated with each of the areas.

2.1.2 Sample Preparation - Sample Racks

Each sample to be equilibrated in the system requires a dedicated, unshared collection of mechanical hardware termed the sample unit (see figure 2.2). In this system its parts are the sample flask assembly where the equilibration reaction takes place, a small reservoir to store the equilibrated gas until it can be used during analysis, and two interconnecting solenoids that isolate the above containers from each other and from the rest of the system.

In the final analysis DEL values are calculated by comparing isotopic ratios obtained from known standards and from unknown samples. The results thus obtained will be valid only if both standard and unknown samples have undergone identical processes. To keep conditions between the individual sample units as similiar as possible it becomes necessary to group them together as one unit in some physically manageable configuration. Historically such a unit is known as the <u>sample rack</u> (figure 2.1).

Choosing the number of samples on each rack involves a trade off between conflicting demands. The number finally decided upon was sixteen, the length of the microprocessor word. This choice simplifies design of both hardware and software. Sixteen samples a day represents a doubling of the manual analysis rate previously possible in this laboratory (Ahern 1980) and gives our research real value. The cost of constructing a rack for sixteen samples is relatively inexpensive, and the physical size is small enough to make it possible to remove the rack for repair and modification.

The microprocessor architecture supports a system with between one and sixteen racks, and the operating system design can schedule a mass spectrometer of adequate stability to analyse all the racks each day. Allowing for downtime, weekends, and holidays, our laboratory would have the capability of producing about 4000 analyses each month, a figure entirely adequate for any forseeable needs.

2.1.3 <u>The Mass Spectrometer Line</u>

Although, after equilibration, the CO_2 has a DEL value related to the water sample, it is not yet suitable for admittance to the mass spectrometer. The gas must be "cleaned", mainly of water vapour, before it enters the high vacuum of the measuring system. Traditionally a vapour trap is used to cool the CO_2 below the liquification temperature of any condensable contaminates present in the gas. The most convenient (and least expensive) scheme is to pass the CO_2 through such a trap as it enters the mass spectrometer inlet line[1].

The vapour trap, and the manifold that enables the samples to pass from the racks to the inlet line can be regarded as logically and physically grouped. This collection of hardware that is responsible for transporting the equilibrated gas to the mass spectrometer is termed the <u>mass spectrometer line</u> .

2.1.4 <u>General Services - The Main Line</u>

Common operations are performed on both the sample racks and the mass spectrometer line. Typically these are the evacuation of gases, the admittance of CO_2, and associated pressure testing. There is little point in duplicating these expensive services for the mass spectrometer line and each rack, especially as such services are infrequently used. Under software control a common facility can be efficiently shared by the whole system with little or no loss of throughput.

The hardware necessary to provide such services is termed

[1] There is no point in removing water vapour before the equilibrated CO_2 is stored in the reservoirs. (see section 2.2.5)

the main line . It consists of a vacuum pump, a CO_2 cylinder (and regulator), pressure gauges, and a distribution manifold. The pump, and the CO_2, are connected or isolated from the manifold using electrically operated valves. The pressure gauges are permanently connected.

2.1.5 Interconnections

Having defined three basic components it remains to connect them together in a manner that not only allows the system to function, but also provides the ability to add and remove racks without interference. Such a scheme is shown in figure 2.1.

Each rack is connected to the mainline and the mass spectrometer line by a corresponding solenoid driven valve (V.MNL and V.MSL respectively). A swagelock tubing connector is installed between these solenoids and the rack body. Provided the two valves are closed the rack may be be disconnected without affecting the rest of the system. The closure of the two solenoids is forced by driving them, via an electrical connector, from within the rack. In order to turn a wrench on the swagelock fitting, the solenoid connector must first be removed.

There is no direct connection between the main and mass spectrometer lines. The later must be evacuated by pumping it through a rack. However since both the rack and the mass spectrometer line must be at vacuum before a sample is transferred from its holding reservoir to the mass spectrometer, then such a connection has little disadvantage. It is possible that the need for a direct route will arise if more advanced testing procedures are introduced, or if it becomes necessary to periodically clean the vapour trap. In such a case the extra

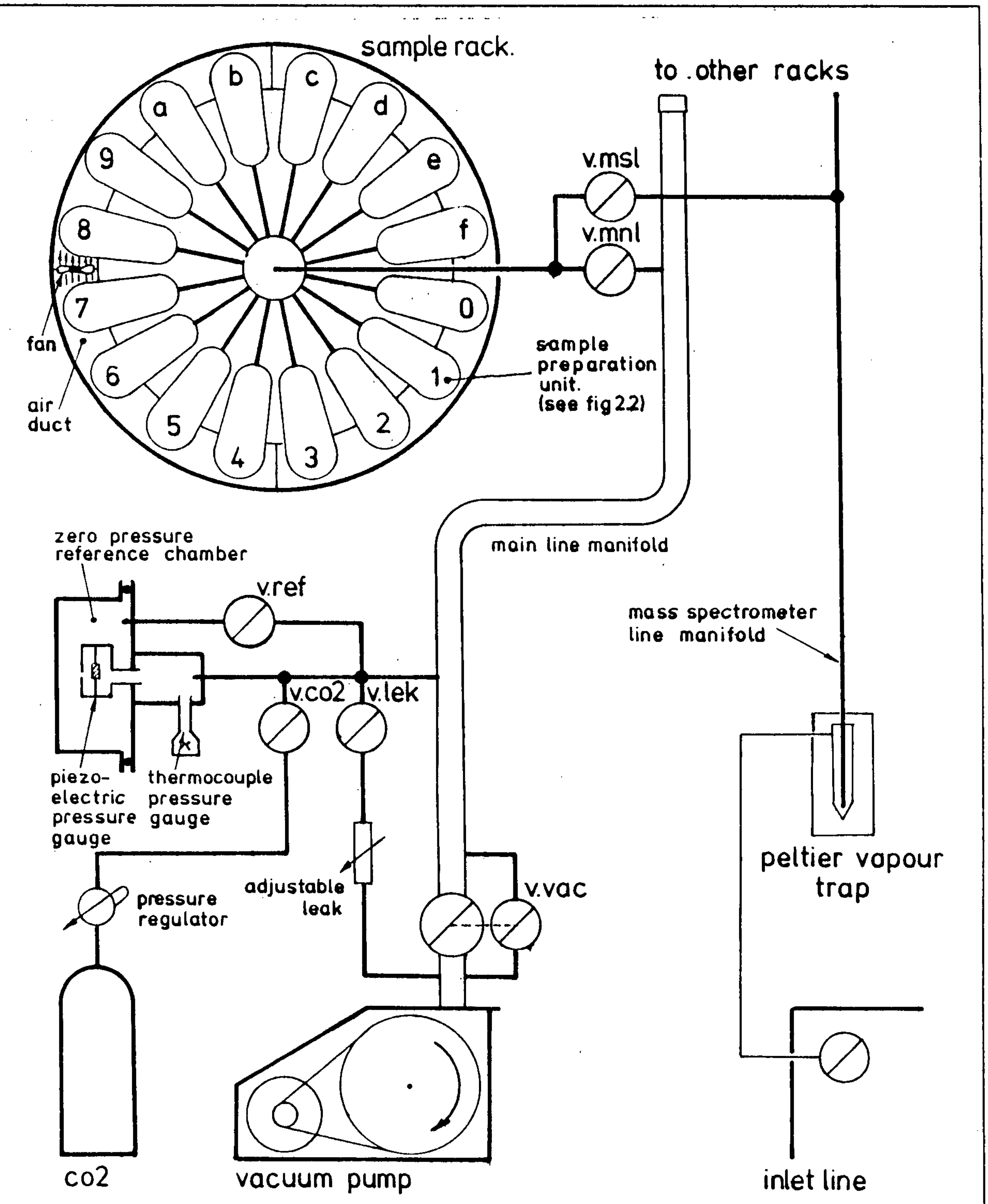


Figure 2.1   Mechanical Organisation of the Sample Line

plumbing is not difficult to add.

Separately connecting the rack to both the main and mass spectrometer lines is necessary for two reasons. Firstly, it allows a reasonable degree of parallelism. One rack can send a sample to the mass spectrometer while another simultaneously uses the mainline. For a minor increase in cost and complexity the system is more efficiently utilised. The second reason is more important. In order to pump gas rapidly the mainline manifold must be of considerable diameter and volume. If separate connections were not made the sample gas would have to expand into both the main line and mass spectrometer lines. The large volume would then reduce sample pressure to an unusable value.

Finally it is interesting to note the equivalence of this interconnection system with that of the parallel bus structure in microprocessor architecture. Both enable units to be added to ,and deleted from, the bus independently of the operation of others. It is the employment of parallel structures in any form that makes modular systems workable and convenient.

2.2 THE SAMPLE RACKS

2.2.1 Rack Overview

Three major decisions have been taken that give the present sample racks a unique design. They have brought advantages, but also new problems.

The choice of a symmetric geometry that could assure equal pumpdown, and hence fractionation of the water samples, is responsible for the circular configuration that finally evolved. Although convenient for the above reason, and also for a certain compactness, there are problems of machining and access.

The decision to use sample reservoirs, rather than sensitive stable temperature controllers, has eased the restrictions of temperature constancy. Disadvantages arise from the extra space, and cost, incurred by the reservoirs and their associated solenoids. In common with previous preparation systems the samples (and reservoirs) must be kept at <u>equal</u> temperature. A rack must be designed to accommodate an area where such a condition is met.

Finally, the use of magnetic stirrers in place of a rack shaking mechanism probably allowed more freedom with the shape, size and mass of the rack than could otherwise have been possible but the actual stirring mechanism, though elegant and simple, was not arrived at easily.

Satisfying these demands required a good deal of compromise and interaction so the following sections are more descriptive than deductive. An overall picture of the rack organisation is first presented to identify the relationship of the various components before they are discussed. No detailed mechanical description is given as it was felt irrelevant to this thesis.

2.2.2 <u>Rack Organisation</u>

The arrangement and interconnection of the various components within the rack is best understood by referring to figure 2.1. For reasons disclosed earlier, sixteen sample flasks are distributed circularly around a central point[1]. Each sample flask is connected to a stainless steel reservoir using tubing and a solenoid valve (V.SMP). Likewise every reservoir is isolated from the common central point by another solenoid (V.RSV). From the central distributor a line leads out of the rack to a swagelock tubing connector. This links the rack to two solenoids which can connect the central distributer to the main and mass spectrometer lines (see section 2.1.5).

All the sample flasks and reservoirs are physically located in a duct through which a fan circulates air (in an attempt to maintain equal temperature over all samples). The air duct is formed by a circular wall, and a top and bottom rim. Four clear plexiglass doors clip around the extreme perimeter of the two rims to complete the enclosure, but still allow convenient access to the sample flasks. The doors are held and separated by four small sections of aluminum plate. These provide corridors through which the various services enter and leave. They are also a convenient place to mount electric connectors and indicator lamps. The empty section of air duct behind them can just accommodate a small fan.

2.2.3 The Temperature Problem

The precision of a sample preparation line depends largely upon the repeatability of the CO_2/H_2O equilibration conditions. In particular the temperature between samples must not differ by

[1] Each sample position in the rack has been labelled with a hexadecimal digit (0-9, A-F), for computing convenience.

any significant amount[1].

The restrictions on temperature stability are worsened by the traditional approach to automated sample preparation and analysis. Usually the CO_2 and water remain in equilibrium until the gas can be admitted to the mass spectrometer. Several hours may elapse between the analysis of the first and last samples, and temperature conditions must be constant during this time.

By including several known standards throughout the rack the constancy condition could possibly be relaxed. However accurate interpolation between the standards is uncertain for conditions of random temperature fluctuation and non-constant analysis time. A stable and sensitive temperature controller is still needed.

2.2.4 Using Sample Reservoirs

Constant temperature during analysis is not necessary if all the equilibrated CO_2 samples are simultaneously isolated from the water in the sample tubes and stored pending analysis. Provided that the temperature difference between any two samples is always within prescribed limits, all samples will vary identically. The DEL values of the equilibrated gas above the unknown samples and the known standards to which they are compared change equally, cancelling the temperature instability.

Such a scheme is easily implemented by providing reservoirs in which to store the equilibrated gas (see figure 2.2). By opening the V.RSV and V.SMP solenoids, air in the reservoir and above the water sample is removed. Carbon dioxide gas can then

[1] Section 1.3.1, equation 1.4d, prescribes a temperature difference of $\pm 0.05^{\circ}C$ for an equilibration reproductibility of $\pm 0.01^{\circ}/_{oo}$.

be admitted to the tube and remains there while a magnetic

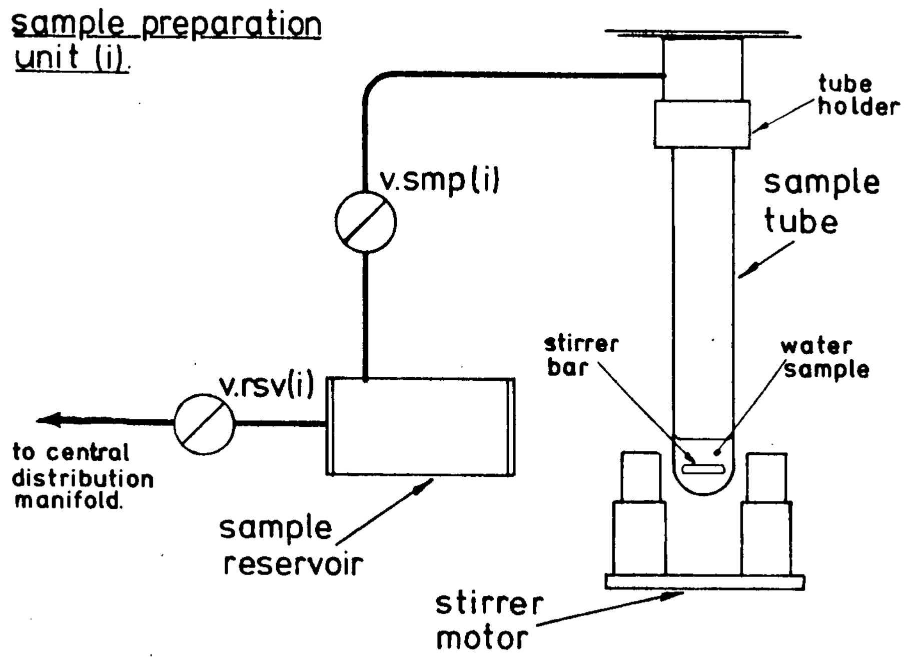


Figure 2.2 The Sample Preparation Unit

stirrer mixes the water. When equilibration is complete the reservoir is evacuated then isolated. The CO_2 gas is stored by opening V.SMP until the gas pressure between the sample tube and the reservoir equalises. The gas can remain in the reservoir, unaffected by the prescence of the water sample, until it is analysed.

2.2.5 Re-equilibration in the Reservoirs

Even though the V.SMP solenoid effectively isolates the CO_2 gas from the sample, some water is inevitably transferred in the form of vapour. Because pressures have been allowed to stabilise then the partial pressure of the water will be close to its saturated vapour pressure at that temperature. Adsorption isotherms on silver and glass surfaces indicate that under such conditions films form up to 100 molecules thick (McHaffie & Lehner, 1925). The first few layers mask the effect of surface forces, so the film behaves very like the bulk liquid. If the reservoir is at a different temperature from the sample then re-equilibration of the stored CO_2 occurs, introducing an error into the measured DEL value.

An estimate of the effect can be made using equations 1.3 and 1.4b, and by evaluating ρ from typical pressures and volumes[1]. If the separation factor for bulk liquid is also representative for the adsorbed film then an error of $0.01^{o}/_{oo}$ results when the temperature difference between sample and reservoir is $0.9^{o}C$.

Such a large difference is unlikely to exist but precautions have been taken in the mechanical layout to minimise the possibility. Firstly, both samples and reservoirs are located in close proximity in case a spatial temperature difference is significant. Secondly, potential heat sources have been located as far as possible from the reservoirs. The most likely sources are the V.RSV solenoids. These will remain on until the inlet line accepts a sample. The time is both long and variable so the temperature rise becomes significant and unpredictable. The solenoids are therefore located towards the center of the rack and are separated from the reservoirs by at least 10 inches of 1/4 inch stainless steel tubing. Any heat transfer should be negligible. Placing these solenoids close to

[1] ρ will be the partial pressure of water vapour divided by that of the CO_2 in the sample. When the sample tube and the reservoir are of equal volumes. Typically these will be 24 and 740 respectively, giving a value for ρ of 0.0032.

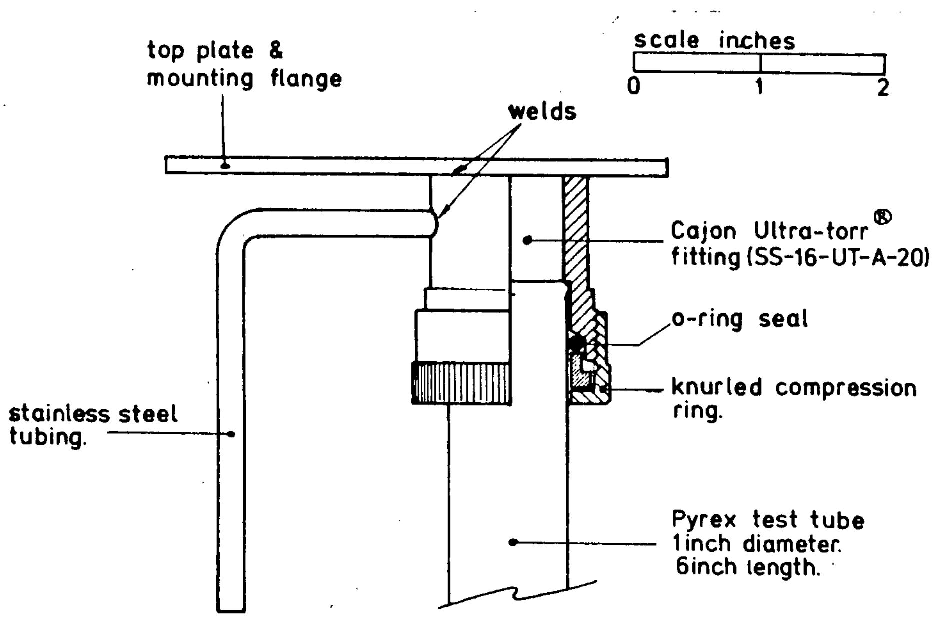
the center also minimises the volume of the common distributor point, helping conserve sample gas, and decreasing pump down time.

2.2.6 The Sample Flask Assembly

The sample flask assembly consists of two components. There must be a flask, or tube, which contains the water sample while it is equilibrating, and a tube holder that both supports and connects the tube into the rack.

The tube holder should be designed to make rapid mounting of the tube a simple operation. The tube must be capable of holding a vacuum indefinitely, and be transparent. This enables the operator to check sample volume, monitor the magnetic stirrers, and watch for sample splash during pumpdown. The volume of the assembly must be large enough to deliver the required amount of gas at the mass spectrometer. The maximum size will be limited by what is available, what fits in the allocated space, and to a lesser extent the ratio of the number of CO_2 and water molecules. If this is too large then the appropriate correction factors may be uncertain (see equation 1.3). Most lines use flask volumes from 50 to 100 cc. Finally it is important to point out that commercially available fittings usually result in significant cost savings, even when they require modification.

The sample assembly used is shown in figure 2.3. The sample tube is simply a one inch diameter pyrex test tube, approximately six inches in length. These have the advantages of transparency, strength under vacuum, and chemical inertness. The rounded bottom makes it possible to stir sample volumes as small as 2.0 cc effectively. The cost is so low that tubes



scale inches
0
1
2
top plate & mounting flange
welds
Cajon Ultra-torr® fitting (SS-16-UT-A-20)
o-ring seal
knurled compression ring.
stainless steel tubing.
Pyrex test tube 1inch diameter. 6inch length.

Figure 2.3 The Sample Flask Assembly

could be discarded after use.

A modified Cajon Ultra-Torr fitting forms the tube holder. The test tube is connected by inserting it into the body of the fitting. A vacuum tight connection is made when a hand tightened knurled ring clamps an O-ring around the outside of the tube. In our experience these fittings have never leaked, even when used on the high vacuum of the mass spectrometer.

The open end of the Ultra-Torr fitting is sealed by welding a stainless steel plate across it. A length of 1/4 inch stainless steel tubing, welded into a hole bored in the side of the fitting, allows gas to enter and leave the assembly. The plate also doubles as a mounting bracket that suspends the tube assembly from the top rim of the rack chassis.

The total volume of the sample flask assembly has been measured at approximately 65cc. The pressure within the tube should not exceed that of the atmosphere surrounding it. Positive pressures will force the test tube from the Ultra-Torr fitting, and increase the possibility of the glass test tube exploding.

As it stands the assembly is quite adequate but some changes could be made in future versions. For a start, it has been found that minuature stirrer bars are available and work extremely well. The neck of the sample tube could therefore be much narrower, possibly around 3/8 of an inch. Smaller Ultra-Torr fittings could be used. The whole assembly would be cheaper, lighter, and more compact. The tubes could be standard laboratory flasks, or hand blown if necessary. They would be much squatter enabling the height of the sample racks to be decreased significantly. Lastly the use of Nalgene as a flask material would be worth investigating. The high surface tension of the polymer makes it harder for water droplets to adhere to the tube walls. These can result in significant DEL errors if they equilibrate at a different temperature to the rest of the sample, as has been previously discussed.

2.2.7 The Reservoirs - Construction and Volume

The reservoir construction shown in figure 2.4 is simple and straightforward. Because no commercial item of the required shape and volume could be found it was necessary to machine parts from suitable stainless steel stock, and to have them welded together. This proved advantageous. It led to the design of a self supporting configuration that fitted neatly into the rack's air duct.

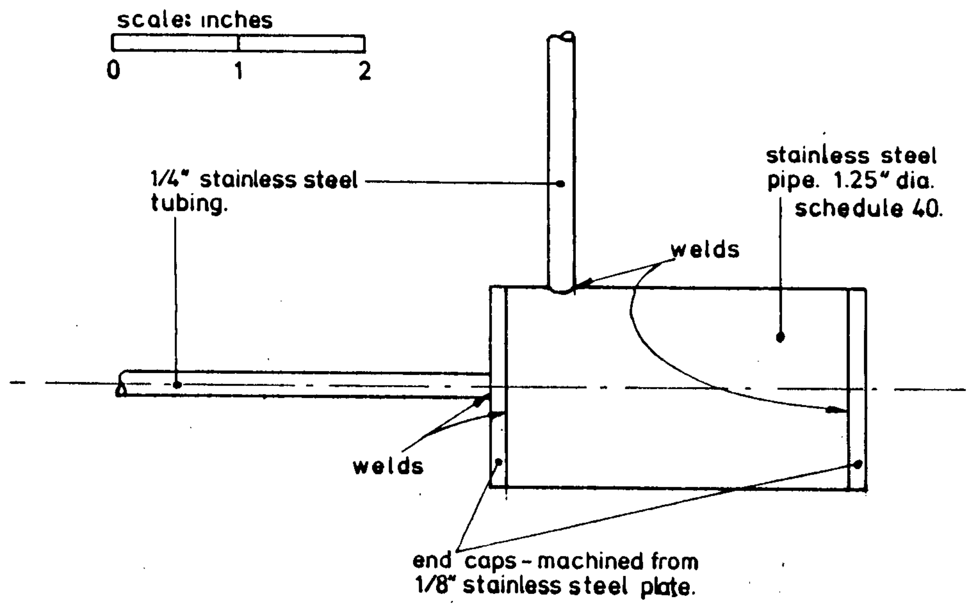

scale: inches
0
1
2
1/4" stainless steel tubing.
stainless steel pipe. 1.25" dia. schedule 40.
welds
welds
end caps - machined from 1/8" stainless steel plate.

Figure 2.4 Reservoir Construction

The capacity of the reservoir was choosen to maximise the sample pressure at the mass spectrometer. The optimum value derived from a consideration of the Boyle gas law is not a critical one. Large changes in the volume of the reservoir make little difference in the end pressure of the gas at the mass spectrometer inlet line. For the regions of interest little would have been gained if the volume of the reservoir exceeded that of the sample. Consequently reservoir capacity was set at 65cc.

Assuming that samples are equilibrated with CO_2 at atmospheric pressure, and that the volume of the inlet line does not exceed 50cc, approximately 25 metres of 1/4 inch o.d. stainless steel tubing can be used to connect the racks to the mass spectrometer before the sample becomes unusable (a pressure below 5cm of Hg). The use of 1/8 inch tubing, and separated pumping manifolds could increase this beyond any forseeable shortfall, but is not likely to be necessary.

2.2.8 Sample Agitation

To minimise the completion time of the reaction continuous agitation of the water samples is necessary during equilibration (section 1.3.3). Traditionally samples have been mixed by oscillating the sample rack as a single unit. Usually a system using eccentric cams and an electric motor is employed (Roether,1970) but recently Micromass (1978) have used solenoids to shake the rack from side to side.

Any shaking system has fundamental drawbacks. For automated operation a flexible vacuum connection must be used; this may fail. Continuous vibration, in any form, severely tests the durability of all components, whether mechanical or electronic. An intensive maintenance program is mandatory for reliable operation. There are other problems. Noise can create an irritating work environment for the operator, and the mechanical complexities can be a major project for the designer. Optimising the shaking motion is no trivial task. Both frequency and amplitude must be adjusted to maximise the reaction rate without splashing any water onto the sides of the sample tube.

Magnetic stirrers seem to suffer none of these disadvantages. They are quiet, vibration free, and mechanically simple. Even at low spin rates the vortex generated by teflon-encased stirrer bars in the sample seems particularly efficient. A drop of ink placed on the surface is completely mixed within twenty seconds. Sample splash is non-existent no matter how rapidly the stirrer rotates.

2.2.9 The Magnetic Stirrers

The stirring system relies on a rotating magnetic field to couple with small teflon encased magnets in the sample tubes. Usually the field is generated by rotating a strong magnet with a small electric motor. However the size and expense of commercially available units was unacceptable, the cost of assembling stirrers of similiar design would be prohibitive, and there were fears that differences in the efficiencies of the motors could generate uncontrollable temperature gradients between samples.

Instead of electric motors a source of compact and non-expensive water turbine stirrers was located. It was felt that temperature gradients would not be a problem with these devices because of the rapid flow of water through them. In practice other problems developed. Firstly the plumbing necessary to supply an adequate flow rate was cumbersome, and volume of water used was high enough to require a recirculating pump. More seriously, PVC turbine bearings in the stirrers proved unreliable and incapable of sustained operation. When the rack was loaded with identical samples significant deviations in DEL could be attributed to the differing spin rates of the faltering stirrers. It became obvious that a more reliable system was absolutely necessary, and that the stirrers should spin in synchrony.

After discussion with others in our group two alternatives arose. One was to couple the stirrers with a toothed belt drive. The other involved construction of a synchronous electric motor. The second approach proved both feasible and simple. After some encouraging experiments the final design shown in figure 2.5 was adopted. Each motor on the rack is assembled from readily available parts.

A large steel washer forms the base, steel bolts and nuts the pole pieces, and reed relay coils the field windings. Machining is limited to four holes drilled in the base washer. Assembly is completed within minutes.

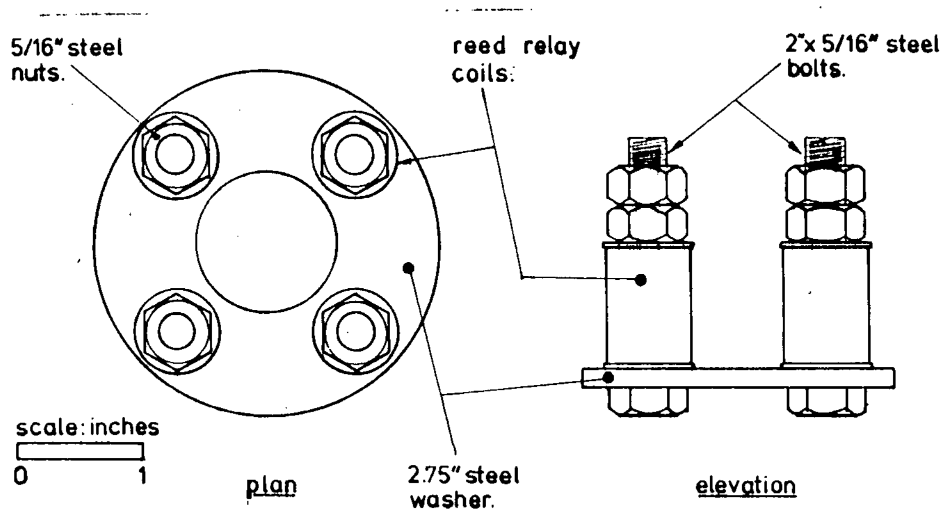


Figure 2.5 Schematic Diagram showing Construction of the Magnetic Stirrers

The 'secret' of the stirrers is not the construction of the motor but the manner in which the magnetic fields are produced. The coils on opposite pairs of poles are wired to generate a field between them. The field may be reversed by reversing the direction of current with a simple solid state switch. By utilising two such switches it is possible to generate a sequence that rotates the magnetic field in eight separate steps. The circuitry needed to generate the drive sequence required only three integrated circuit packages, four transistors, several resistors and capacitors. Two solenoid driver packages and the transistors are used to switch the coil currents, a four-bit Johnson counter generates the correct switch sequence, and spare parts of the drive chips are used as an oscillator to clock the Johnson counter.

Finally the low energy consumption of the agitation system is one of its most striking aspects. Whereas the previous scheme needed a 120 watt motor to pump an adequate supply of water, the present power requirement is about two orders of magnitude lower. No problems can arise from differential heating in the field windings.

2.3 THE MAIN LINE

2.3.1 The Main Line Manifold

The manifold allows the racks to be connected to the vacuum pump and the CO_2 cylinder. It should be of sufficient diameter to allow rapid pump down, and capable of sustaining vacuums below the 10 millitorr level.

Compared to the high vacuum systems needed in the mass spectrometer these specifications were easily met using standard one inch copper pipe and fittings. Entirely adequate joins were made with lead/tin solder. The tubing was easily machined to take a variety of fittings and the geometry of the manifold could be changed in the lab without a special trip to the welding shop.

Some doubts might be raised when considering the rather reactive nature of the copper but such a possibility is unlikely to cause problems. Carbon dioxide is only in contact with the manifold on its way to the sample tubes, and then only for one minute. In any case the gas is dry and unlikely to react. Even if it does, any shift in its DEL should affect both standards and samples equally and contribute no error to the final results.

2.3.2 The Vacuum Pump

Previous calculations showed that residual pressures of 10 millitorr would cause no significant errors, even when the residual gas was entirely CO_2. Such conditions are well within the capabilities of any double-stage rotary vacuum pump.

More importantly, the pump must be capable of passing rather large amounts of water vapour and air without contamination of the pump oil. Generally the larger the pump chambers are and the more oil there is, the less a problem contaminants become. A ballast valve is valuable as it allows air to purge the pump chambers, and to strip the water molecules from the oil.

For the above reasons an old model Welch pump was pressed into service. Although bulky by modern standards it was known to be trouble free, and could handle the water vapour.

Finding a solenoid to isolate the vacuum pump from the manifold presented some problems. There are many kinds which can adequately seal a one inch diameter pipe; there are few that open or close against a pressure of over one atmosphere. This requires a plunger-type solenoid to pull with a force of over five kilograms, so a butterfly or door type mechanism is usually used, possibly pneumatically driven.

Instead of resorting to such expensive complications a rather simpler method was implemented. This used an air admittance valve (Edwards model SVA-25) connected in the reverse of the usual manner. High pressures in the manifold push the plunger against its seating, helping to maintain a good seal. The solenoid that operates the plunger is totally incapable of operating against pressures above two or three psi so it is necessary to bypass the large solenoid with another that has a smaller orifice. This can operate against pressures of over 100 psi, and will evacuate the manifold until the large plunger can open. In practice this occurs after approximately five seconds.

2.3.3 Pressure Gauges

Two gauges have been provided on the system to monitor evacuation of the racks and admittance of CO_2.

The lower range gauge is a standard thermocouple type. It gives reliable readings from 5 to 500 millitorr and is useful for evaluating the removal of residual gas from the racks, and for detecting small leaks.

The higher range measures absolute pressures from 5 to 1500 torr. This can monitor pump down of the water samples, and admittance of CO_2 to the sample tubes. Because reliable absolute gauges of this range are extremely expensive, a piezoelectric gauge pressure transducer was used. The entire transducer, normally at atmospheric pressure, has been placed in an o-ring sealed chamber that can be evacuated under computer control. The pressure port connects to the vacuum manifold to measure absolute pressure against the zero reference.

The readings from both gauges are visible to the operator on two meters, and are available to the controlling microprocessor through two analogue-to-digital converter channels.

2.3.4 The CO_2 Supply

Carbon dioxide is piped directly from the same cylinder that supplies the reference side of the mass spectrometer inlet line.

A standard regulator is used to maintain a small positive pressure in the distribution line (approximately 5 psig). Admittance to the manifold is via a small solenoid (Skinner type B2DX70).

2.4 THE MASS SPECTROMETER LINE

2.4.1 The Gas Lines

The mass spectrometer line transports gas samples from the racks to the inlet line. There are three main components; a manifold that connects all racks to a peltier cooled vapour trap, the trap itself, and its connection to the mass spectrometer inlet line. The volume throughout has been kept as low as possible to conserve sample gas.

Gas flows to the inlet line along approximately 1.5 metres of small bore stainless steel tubing (1.6mm i.d.). Evacuation times would be excessively long if it were not for the low volume of this section, and the dryness of the gas passing through it. Much higher residual pressures can be tolerated before contamination and re-equilibration errors become at all noticeable.

More care is required with the manifold preceding the vapour trap. Its tubing must be large enough to pump away adsorbed water during the time between analyses, but must be small enough to conserve sample gas volume. The current configuration uses about one metre of stainless steel tube, with an inside diameter of 0.32 millimetres (1/8"), to connect the rack and the trap.The mass spectrometer line is pumped for ten minutes between samples, and there is no indication of error from water adsorbed on its surfaces.

2.4.2 A Peltier Cooled Vapour Trap

The removal of condensible contaminants from vacuum systems is an old problem, and many methods have evolved to meet it. Unfortunately, available commercial traps lacked in several aspects, and it proved necessary to design something more suitable. There were three important criteria to meet: a small internal volume, reliable unattended operation, and the ability to freeze away water vapour without liquefying any of the CO_2.

The basic trap is machined from a small stainless steel block (Figure 2.6). A blind, 3/8" diameter, hole forms the internal cavity. This has been tapped and sealed with two Cajon Ultra-Torr o-ring fittings. These minimise thermal conduction between the trap and the rest of the line by allowing connections to be made with 1/4" diameter glass tubing. The top Cajon fitting has been bored through so the inlet tube can protrude into the bottom of the cavity. This forces the gas to

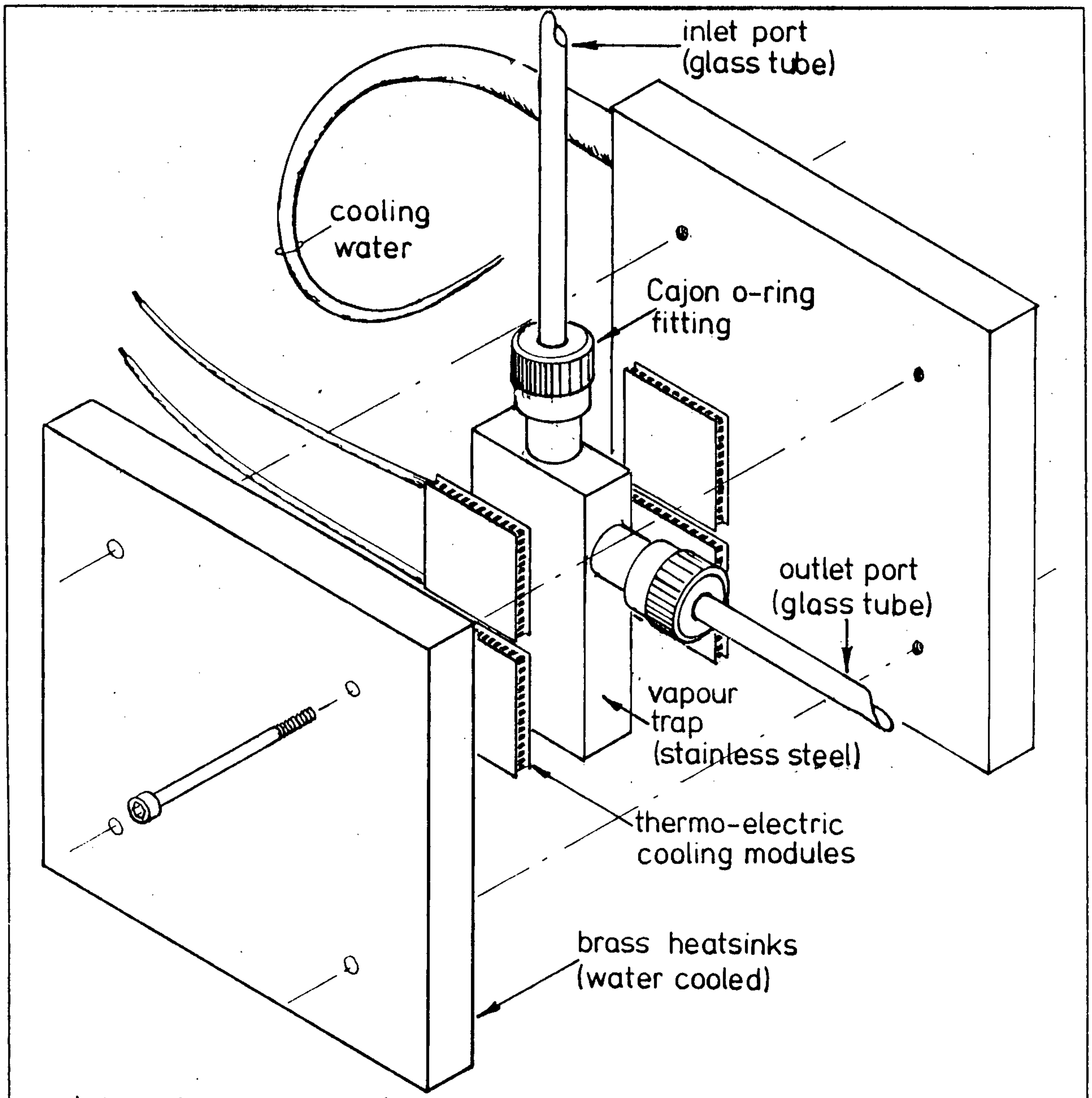


Figure 2.6 Peltier Cooled Vapour Trap Assembly

flow in a thin layer over the cold stainless steel surface on its way to the exit port. Not only does the gas cool faster, but small ice particles should fall to the bottom of the trap and remain there.

Four thermo-electric cooling modules (Melcor, CP 1.4-71-

06L) refrigerate the block. The modules are an extremely rugged and compact array of thermocouple junctions that operate through a phenomenon known as the Peltier effect. Electric power, dissipated in the junctions, pumps heat from one face of the module to the other.

The cold faces of four cooling modules are cemented to the stainless steel block with a heat-sink compound. This assembly is then sandwiched between two large brass heatsinks, and the air space inside is packed with a foam insulation. Tap water flows through holes bored in the heatsinks to keep them and the outer faces of the modules at approximately 10°C. The modules are wired in series so they may be driven from the 24 volt solenoid power supply. Power consumption is around 100 watts.

The surface temperature of the stainless steel block has been measured at a constant -25°C, even when wet gas is passing through the trap. There is absolutely no indication of water arriving at the mass spectrometer, nor do there appear to be any "memory" effects associated with water adsorbed by the trap and the manifold.

CHAPTER III. HARDWARE ORGANIZATION AND OPERATION

3.1 SYSTEM ORGANIZATION

Although there is nothing particularly innovative in the design of the electronics hardware, its construction was necessary to implement the ideas described in this thesis. This chapter presents the overall organisation of the microprocessor controller (figure 3.1) and gives a non-technical description of its component parts. Others who seek a more detailed technical explanation may find it in Appendix I.

I found it convenient to use a pre-assembled microprocessor system know as the University Kit board. Although originally intended for university level tutorials in microprocessor design the board adapts easily to other tasks, and comes with an impressive array of hardware devices and software tools.

3.1.1 System Buses

It is evident from figure 3.1 that the various parts or modules of the system communicate with each other electrically using a parallel bus. The bus may be divided into three separate parts.

1) The unbuffered TTL level bus on the University Kit board. This bus is adequate when communicating with all on-board parts but must be buffered if the system is expanded.
2) The TTL level expansion bus on the expansion board is used to interface TTL level devices that the University Kit does not

provide (such as the analogue to digital converter (ADC) and

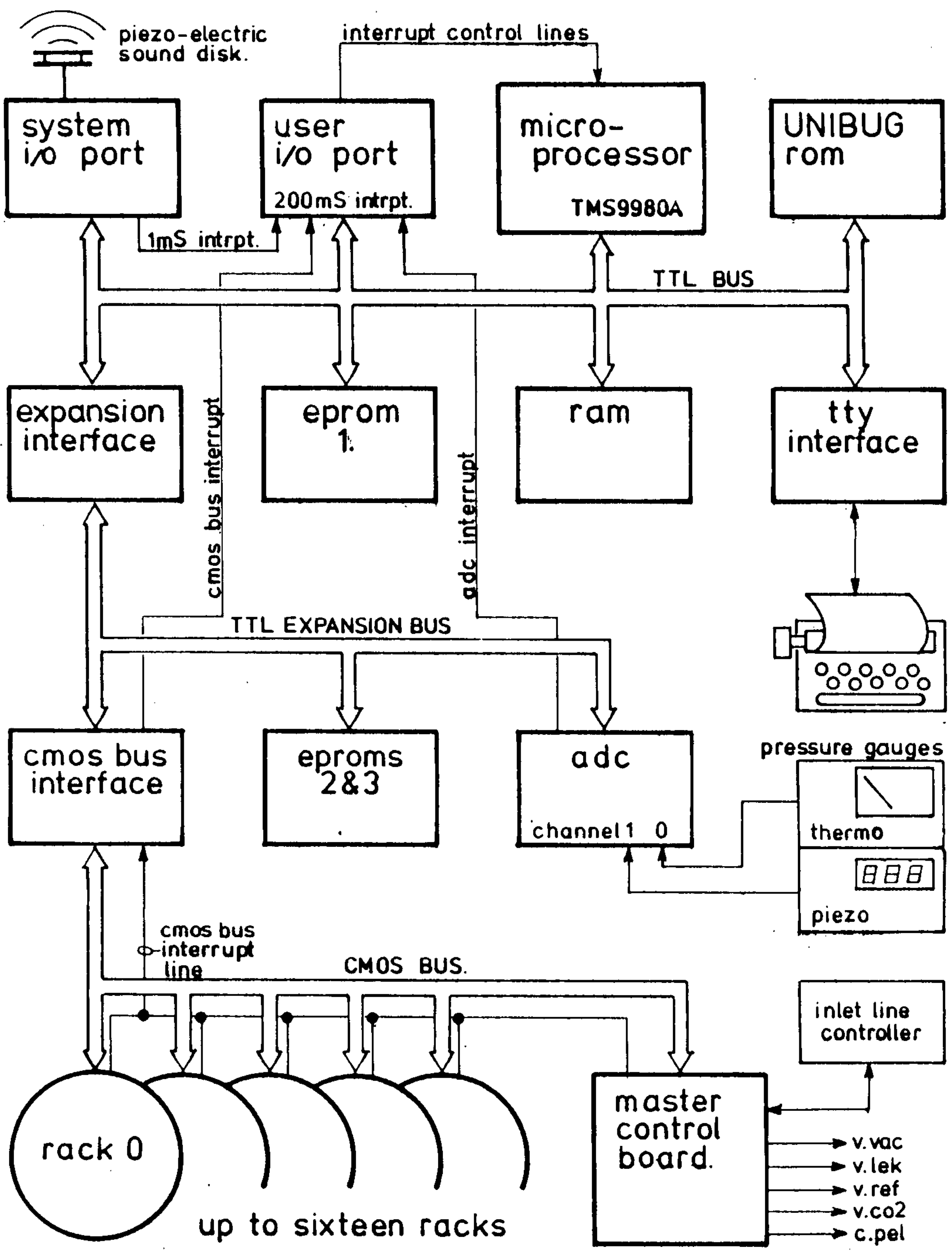


Figure 3.1 System Organisation

extra EPROM).

3) <u>CMOS level expansion bus</u> that is used to communicate with

"noisy" devices. These are primarily any modules associated with solenoid drivers which tend to create large surges and spikes in any signal lines within reasonable proximity. CMOS circuitry is less susceptible to this interference because it is capable of operating at a higher voltage level (12 volts as opposed to 5 volts). The usual penalty in speed of such CMOS devices is unimportant here, as are the concurrent advantages of low power.

3.1.2 System Modules

The hardware may be broken down into the following important parts.

1) The microprocessor (TMS 9980A) - a 16 bit machine with a 2 MHz clock.
2) Random Access Memory (RAM) - up to 1K bytes for program variables.
3) Read Only Memory (ROM) - The UNIBUG monitor and EPROM for user programs.
4) Input/Output Ports. a)
   System I/O port - controls the keyboard, display, visual and acoustic indicators.
   b) User I/O port - is a 16 bit programmable I/O port that doubles as an interrupt interface.
   c) Asynchronous Communications Controller - suitable for operating a teletype.
5) Bus interface - to buffer and expand the system from the onboard bus.
6) A sixteen channel, interrupt driven, analogue-to-digital

converter.

7) CMOS bus interface and expansion - level translation, handshaking and interrupts.

8) Master control board - handles sample preparation functions used by all racks. This included communication with the inlet controller, the vacuum pump, the pressure reference, the peltier vapour trap, and the carbon dioxide cylinder.

9) Rack control boards - control all LEDs and solenoids; sense door switches and the relay power supply for each rack.

3.1.3 System Memory Map

With the exception of the three I/O ports all the system modules can be considered as part of memory. The memory map of the current system is shown in Table 3.1, but because the hardware has been provided with addressing switches it is possible to relocate the racks, the master control board, and the analogue to digital converter. This would allow the addition of another EPROM.

Physical Device	Memory Address
RAM	0000 to 07FF
EPROM.1	0800 to 0FFF
EPROM.2	1000 to 17FF
EPROM.3	1800 to 1FFF
RACKS	2000 to 20FF
MASTER	2100 to 21FF
ADC	2200 to 2203
unused	2204 to 2FFF
unused	3000 to 3FFF

Table 3.1 System Memory Map

The master control board uses two words (status and control registers) but these are selected with only the two least significant bits of the CMOS address bus. Thus the master control board inefficiently occupies all of the memory shown in table 3.1. Such redundancy is not uncommon in microprocessor design. Usually savings in the decoding circuitry are great when not all of the memory space is needed.

3.1.4 Modularity

It is important to design the complete sample line preparation system with a large expansion capability. It should be possible to construct new racks as they are needed and to connect them into the system without making any mechanical, hardware or software changes. This need is fulfilled by using a buffered, parallel I/O bus on which the racks may be placed, and by using switches to set up the address of each rack.

3.2 THE MICROPROCESSOR (TM9980)

The University Board is built around the Texas Instruments TM9980A microprocessor. Its instruction set is compatible with the TI 990/9900 family of microcomputers and processors but there were several features that made its choice very attractive.

1) A 16 bit central processing unit. The ability of any microprocessor to operate directly on a 16 bit word enhances software capability two ways.
   a) Any address in the range from 0 to 65,535 can be referenced directly. When processor operations are

limited to an 8 bit word the corresponding range is usually 256 bytes. References to subroutines, or I/O are often furthur away, so there is constant shuffling of high and low bytes between EPROM and registers

b) Arithmetic and logical operations on 16 bit word involve several steps with an 8 bit processor. These are accomplished with one instruction in this machine, and this allows convenient manipulation of blocks of sixteen samples. To limit samples to blocks of eight was not practical.

2) An 8 bit, multiplexed data bus. Operations on 16 bit words usually require a 16 bit data bus if optimal speed is to be attained. By sacrificing speed it is possible to reduce the bus size and to gain in terms of hardware cost and complexity. In the TM9980, memory contents are read and written in two cycles, one 8 bit byte at a time. The instruction cycle takes approximately twice as long to execute but this is of no consequence in oxygen isotope sample preparation. However with bus interfaces to be provided on sixteen racks the savings in wiring and components are considerable.

3) Workspace Pointer. Usually machine operations take place in an internal register set. This processor contains no internal work registers but uses a pointer to define a block of RAM that is equivalent. The context in which the CPU works is therefore easily changed by changing workspace pointers. This has allowed the design of a simple, yet elegant way of scheduling tasks (see Chapter 4).

Additionally the chip has the following features which make life considerably easier for the hardware designer and programmer.

1) Six prioritised interrupts simplify detection and service in both a hardware and a software sense.
2) An addressable memory of 16 kilobytes is more than adequate for most needs.
3) An addressable single bit I/O interface, separate from memory mapped I/O allows times, teletypes, keyboards, displays, LEDs, and interrupts to be easily referenced, tested, and changed. This is especially true given that the interfaces to all these devices already exist on the University Kit board.

On the whole a powerful instruction set, six different addressing modes, and the extensive on-board interfacing have made application of this board particularly easy and clean.

3.3 RANDOM ACCESS MEMORY

RAM is used to store data (both input and output), and temporary and immediate results of program execution. The University Kit board has provision for 1028 bytes (512 words) of RAM which is quite adequate to system needs. RAM has the disadvantage that a power failure will destroy any information stored within it, but battery backup could be provided if required.

3.4 EPROM MEMORY

All program is stored within eraseable/programmable read only memory. This is provided using three INTEL2716 chips with a capacity of 1024 words each. One chip is on the University Kit board and two more have been added to the TTL expansion bus. An extra INTEL2716 could be added if furthur decoding circuitry was designed, and if the addresses of the ADC, the CMOS bus, and the master board were changed.

EPROM's are erased by removing them from their sockets and placing them under strong ultra violet light for about thirty minutes. They are re-programmed in this laboratory using a Pro Log Programmer and a Model 74 Interdata computer. Machine code is generated from a cross assembler on the university supported Amdahl computer, and is transferred across an HDLC link to the model 74 in our laboratory. I am much indebted to R.D. Russell for his work in implementing the link. Without it my task would have been extremely laborious. The cross assembler was written in the GASS language for the Department of Electrical Engineering. However, it was deficient in several respects and I had to rewrite extensive sections of the code to remove some annoying bugs and implement several convenient enhancements. A separate subroutine was needed to produce a suitable object file for transfer to the Model 74 minicomputer.

3.5 THE SYSTEM I/O PORT

As with all University Kit board components a more comprehensive description will be found in the Users Guide (Texas Instruments, 1979). The system I/O port is based around a programmable systems interface (TMS9901), and interfaces to the CPU through the communications register unit (CRU). The system I/O port controls the following system parts.

1) The display timer which generates a periodic interrupt every 1 ms.
2) The alphameric keyboard can be used to input UNIBUG commands.
3) The ten digit, seven segment LED display on which UNIBUG ROM can translate all ASCII characters to seven segment codes.
4) A piezoelectric sound disc which flexes whenever a voltage is applied to it. Pulsing the SPKDRV line of the system I/O port generates a tone of the pulse frequency.

The sample line operating system uses the 1 ms timer to refresh rack registers and to control the sound disc. UNIBUG uses the timer to sense the keyboard and to drive the display.

3.6 THE USER I/O PORT

This operates in an identical manner to the system I/O port, but maintains a different CRU address. There are two important functions dedicated to the user I/O port in the current system.

1) Its internal timer is set to generate an interrupt every 200 ms. This periodic signal keeps track of real time.

2) All interrupts are input to the system through the user I/O port (see sections 4.13, 4.14, & 4.15 for a description of interrupt operations).

3.7 THE SERIAL COMMUNICATION PORT

This port consists of an asynchronous communications controller (TMS9902), and various drivers capable of RS-232-C or 20mA current loop operation. In the present configuration the on-board options have been wired to drive a 20 mA teletype current loop.

3.8 BUS EXPANSION INTERFACE

All address, data and control lines are available at the bus expansion interface. For a comprehensive tabulation of pinouts and characteristics, refer to the Users Guide (Texas Instruments, 1979).

3.9 THE ANALOGUE-TO-DIGITAL CONVERTER

The analogue-to-digital converter is based upon the INTERSIL ICL8052/ICL1704-12 pair (Intersil,1981) and features a 10 volt input range with 12 bits of accuracy, plus polarity and overrange indication (actually equivalent to a 13 bit binary number). An on-chip reference voltage and clock simplify design and conserve board space. Auto-zeroing compensates for any internal amplifier offset drifts. Any one of sixteen single-ended inputs can be selected through an INTEL IH6116 multiplexer.

The programmer initiates a conversion cycle by writing the channel number of the input to be measured to the ADC memory address. When, after approximately 20mS, the conversion is complete, an ADC interrupt will be generated. This causes the microprocessor to temporarily abandon its current task, and to begin execution of an interrupt service routine. The service routine must read the conversion from the output buffers of the ADC and store it in some specified location. The ADC circuit has been designed to reset the interrupt line and to reinitialise the converter during the read cycle, thus allowing the interrupted task to be resumed once the data are stored.

3.10 <u>THE CMOS EXPANSION BUS</u>

A parallel bus structure enables the microprocessor to communicate with a master control center (for the vacuum pump, CO_2 cyclinder etc.) and up to sixteen racks. However the racks will be spatially separated in an electrically noisy environment. Induced voltages can unpredictably appear on the the bus lines and result in unforeseen processing errors. In such a situation CMOS logic, operating on a 12 volt supply has at least five times the noise immunity of equivalent TTL circuitry so its choice is obvious. The inherent slowness of the CMOS devices is not a disadvantage in the current application. The bus is accessed very infrequently, and a simple handshaking scheme compensates for any bus delays.

3.10.1 Bus Interrupts

A simple interrupt strategy has been used for the CMOS bus. When any device attached to the bus requests processor service it lowers the CMOS bus interrupt request line. The user must provide software to identify the source of the request, reset the interrupt flag, and execute an appropriate service routine. Given the extreme improbability of simultaneous interrupts priority encoding schemes are completely unwarranted.

Most interrupts will be generated from micro switches mounted in the racks. As with most mechanical switches these bounce when changing position and take a long time to settle. It is possible for the microprocessor to poll the switch position between bounces, and to miss the interrupting rack. Some form of switch debouncing must be incorporated to prevent this possibility. There are various software solutions to the problem but these tend to occupy the processor whilst waiting for the bounce to subside. Instead I have used a circuit incorporating a retriggerable monostable. It prevents bus interrupts from reaching the microprocessor unless the switch has been steady for 10 ms. As the bounce period is nearer 1 ms, the interrupt request will be delayed until 10 ms after the switch settles.

3.11 THE SAMPLE RACK HARDWARE

3.11.1 Rack Hardware Organisation

The rack hardware is organised into functional modules operating in parallel from the CMOS bus. Physically the modules are wired on printed circuit cards that plug into edge

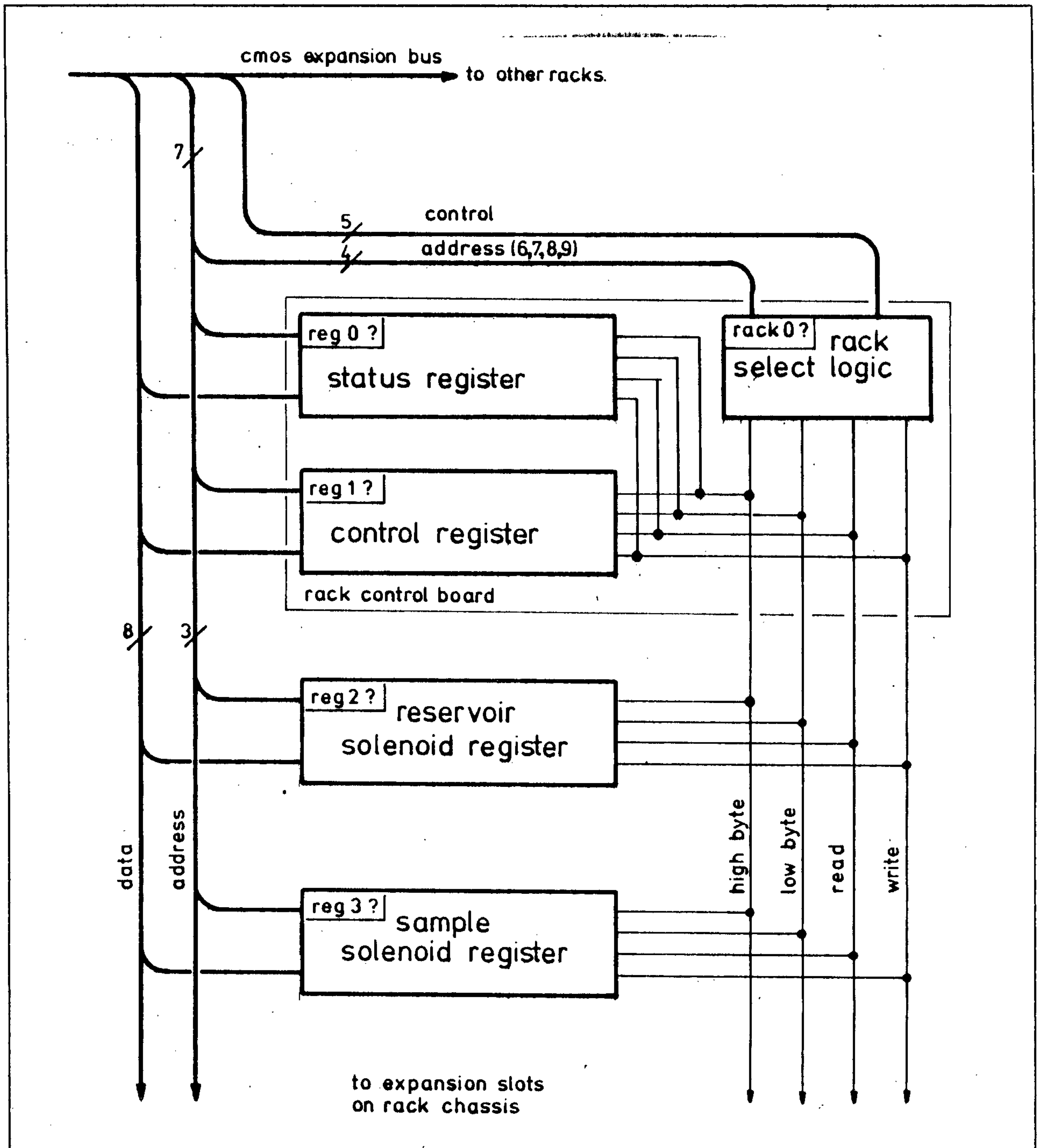


Figure 3.2 Rack Register & Hardware Organisation

connectors on a circular chassis. The CMOS bus is brought into the rack on ribbon cable and is wired across the back of each edge connector. The circular chassis mounts in the center of the rack on five push-fit connectors. These provide the necessary electrical connections to the solenoids, the physical chassis mounting, and allow the chassis to be easily removed for repair and modification. Of the six card positions, three are currently used, so there is ample room for expansion. The bus organisation is shown in figure 3.2.

Rack registers are selected with the aid of four signals produced by the rack select logic on the rack control board (figure 3.2). The signals become active only when the rack field of the address bus (figure 3.3) equals the rack number as set by switches mounted on the rack control board. The four logic lines determine the nature of the cycle (read or write) and the byte to be accessed (high or low). The rack select logic also handles the generation of interrupts and the bus read handshaking protocol.

The other two boards drive the sample and reservoir solenoids, and are electrically identical. Their address is set using board-mounted switches. This flexibility is handy for setup and servicing but care must be taken not to duplicate register addresses.

3.11.2 Rack Register Addressing

The memory address word of the sample line system has been divided into fields to facilitate software readability and hardware decoding. For the racks, an address has four fields. The first specifies the memory block in which the racks are located, the second specifies the page within that block, the third is the rack number, and the fourth is the register number (see figure 3.3).

A page of 256 bytes is capable of supporting sixteen racks,

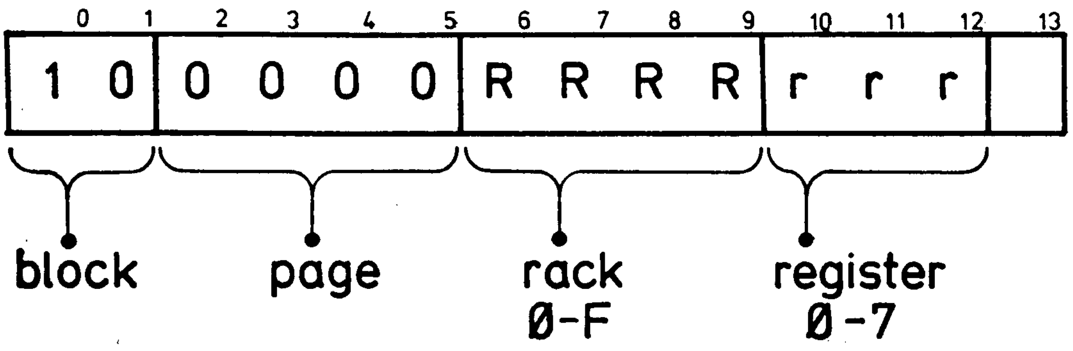


Figure 3.3 Rack Register Addressing

with eight registers for each rack. At the current level only four of these are assigned, the rest being reserved for future hardware expansions (figure 3.2).

It should be noted that the hardware does not support both read and write from the same register. It was felt that more advantages would be gained by having an image of the rack register in RAM, from which the rack register is continually refreshed (see section 4.17.3). This allows the rack state to be restored even if it should be removed totally from the bus and power supply. It also saves on hardware, and mates well with the sample line operating system.

3.11.3 The Rack Status Register

When read, the rack status register provides information about the status of the addressed rack. The flags of the status word are shown in figure 3.4. The four least significant bits monitor micro-switches sensing the position of the four doors that enclose the samples. The fifth is true when the solenoid power supply (+24 volt) is present and false otherwise.

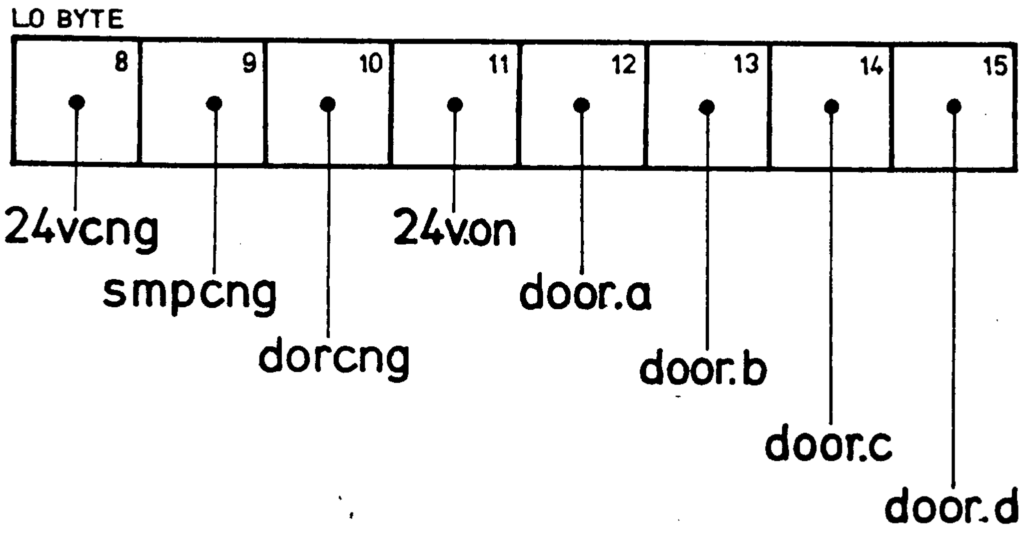


Figure 3.4 The Rack Status Register

This allows the processor to sense whether a rack is active or present on the system.

If any of the three most significant bits (Bits 8,9,10) are set, then a bus interrupt will be generated. These flag the origin of the bus interrupt to the service routine executed by the central processor, and are automatically reset after the status register has been read.

Note however, that these bits are set whenever there is a change in state since the previous reset. For example, if a door was open the DORCNG line will be set when the door is closed. Reading the status word resets this line, but it will be set again when a door is opened.

The SMPCNG is inactive in the current system, and is provided in case future system expansion incorporates sample switches and interface logic on another board. Operating in a similar manner to the doors it should be set whenever a sample tube is inserted or removed from a holder.

3.11.4 The Rack Control Register

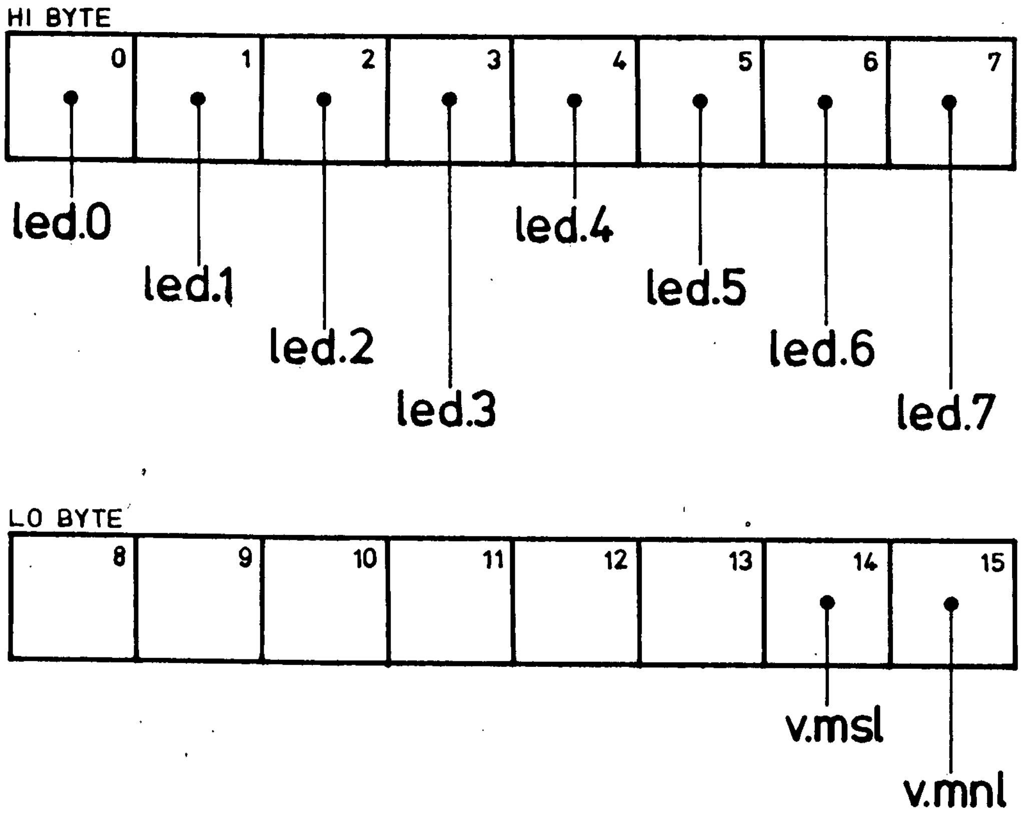


Figure 3.5 The Rack Control Register

Setting appropriate bits in the rack control register operates the solenoid, and lights the front panel light emitting diode shown in figure 3.5. Unassigned bits are slated for future rack functions that give more control over the fans and the stirrers. Temperature controllers are another possibility.

3.11.5 The Solenoid Driver Registers

Setting a bit in a solenoid driver register operates the corresponding solenoid, and opens a valve. The number of the solenoid specified by manual operations corresponds to the numbering on the rack, but is reversed from the bit numbering system (bit #0 is solenoid #15 and vice versa). This made software for manual operations much simpler to design, and is transparent to the operator using manual control from the sample line operating system.

3.12 THE MASTER CONTROL BOARD

3.12.1 Master Control Functions

The system's master control board operates and senses all equipment that is shared amongst the racks. Previously the shared facilities have been divided into two functional areas: the mass spectrometer line, and the mainline.

Functions associated with the mainline all require the operation of solenoids to provide a particular shared service. The solenoids connect the main line to the vacuum pump (V.VAC), the CO_2 cylinder (V.CO2), the slow pump rate line (V.LEK), and the pressure gauge reference chamber (V.REF).

The only device physically present on the mass spectrometer line is the Peltier cooler, and a control line (C.PEL) is provided to power this on and off. However communication with the mass spectrometer's inlet line controller should be logically included. This is effected by coupling four TTL level signals from the inlet line controller to the CMOS master control board via optical isolators (Hewlett Packard Type 6N136).

Three of the lines provide information about the state of the sample line to the inlet line controller. The first flags

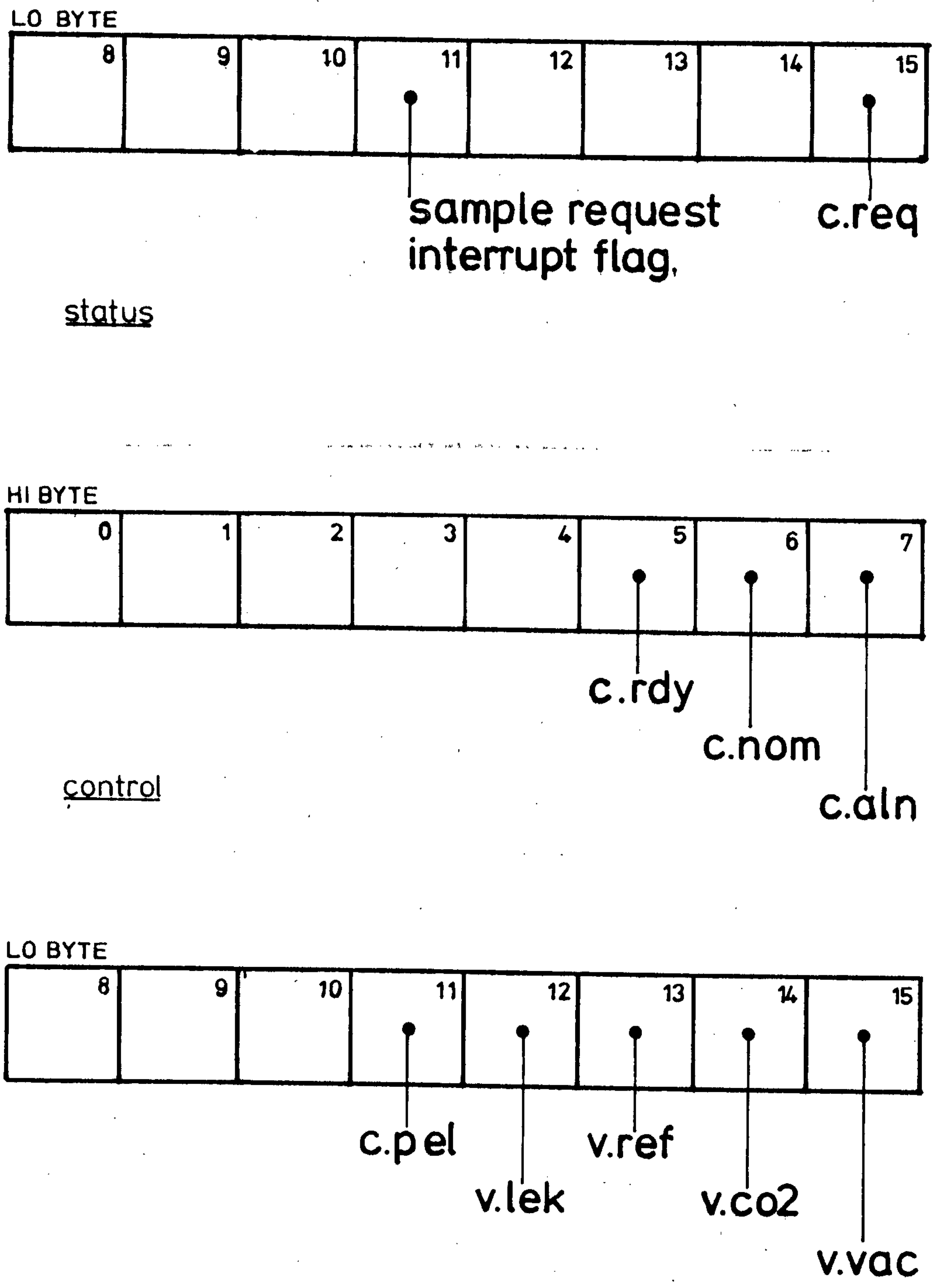


Figure 3.6 The Master Control and Status Registers

the availability of the sample line (C.ALN-autoline), the second indicates no more samples to analyse (C.NOM), and the third is set to indicate that a sample is present in the mass spectrometer line and ready for analysis (C.RDY).

The fourth line is a flag <u>from</u> the inlet line requesting a sample (C.REQ). The inlet controller will set this line when it needs a new sample, and will reset it when told a sample is ready (C.RDY is set). The master board generates a CMOS bus interrupt when the sample request line is raised, and will be reset when the master status register is read.

Two registers are more than adequate for all the common system functions. The bit designations for these, the master status, and master control register, are shown in figure 3.6. On board light emitting diodes give a visual indication of the state of all master board lines.

CHAPTER IV. THE OPERATING SYSTEM

4.1 SYSTEM OVERVIEW

The system software must be designed to execute a variable number of asynchronous tasks. Because it is possible to represent a number of different tasks by the same procedure (program) using different data, a task in becomes synonymous with a (program, data) pair. The tasks are considered asynchonous as there is usually no way of predicting when the system will be asked to execute them. The following lists several examples of tasks.

1. Process the samples on rack (i) where i=0,1,2,3..or 15.
2. Manually control rack (i) where i=0,1,2,3..or 15.
3. Periodic pressure testing.
4. Accept, decode and execute user system commands.
5. Update clocks.

Execution obviously should proceed as rapidly and as economically as possible but it must be remembered that tasks are likely to be competing for the following resources:

1. The central processing unit (CPU).
2. Random access memory (RAM).
3. Various external devices, including:

   a) The teletype and its associated routines.

   b) The main vacuum line, the vacuum pump, pressure gauges and CO_2 cylinder.

   c) The inlet line, and the peltier cooler.

d) The analogue-to-digital converter.

It becomes necessary to design an operating system to oversee the allocation of these and other resources, and to develop a general structure upon which the operating system can effectively work.

4.2 TMS9980 SOFTWARE TECHNIQUES

To understand the sample line operating system it is necessary to have some knowledge of the methods used by the microprocessor to execute its instructions. The TMS9980 microprocessor is well suited to task-oriented, list-structured approaches, mainly due to its workspace/context switch concept. To explain these ideas some basic terminology must be introduced.

4.2.1 Workspace Pointer (WP)

Historically the first computers performed all arithmetic and logical operations in an internal register known as an accumulator . Because most of the programming effort usually went into fetching and storing accumulator results, multiple accumulators were developed. As programming concepts developed it became necessary to break tasks into modules or subroutines, and to enable the processor to jump between them. This context switch usually meant that all the multiple registers had to be saved in a known place, then replaced with new values for the new context. Despite the presence of special instructions (e.g. STM in INTERDATA computers) to lessen programming effort, a large amount of processor time and bookkeeping was still spent keeping track of the registers and performing the switch.

Texas Instruments has avoided this problem by retaining one internal register called a workspace pointer that is loaded with the address of a block of sixteen contiguous words in RAM. The block then behaves as if it were 16 general purpose registers, but the real advantage is that changing register sets can be effected simply by changing the workspace pointer.

4.2.2 The Program Counter (PC)

Any processing unit works by repeating the following basic algorithm.

1. Fetch the instruction at the address in the program counter.
2. Increment the program counter to give the address of the next instruction.
3. Execute the fetched instruction.

The program counter is an internal register that contains the address of the next instruction to be executed.

4.2.3 The Status Register (ST)

The status register is another internal register which contains status or flag bits set as a result of arithmetic and logical operations. These bits can be tested by conditional jump and branch instructions so the value of the status register determines the future course of program flow.

The status register also contains four bits that comprise the interrupt mask. The interrupt mask determines which external interrupt lines will be allowed to pre-empt an executing task.

4.2.4 The Program State Vector (PSV)

The program state vector is simply all the information that describes the current state of an executing program. If a program is interrupted then restoring the PSV will allow execution to proceed from where it left off. Obviously for the TMS9980 the PSV consists of:

a) The workspace pointer so the general purpose registers may be found.

b) The program counter so the next instrucion may be found.

c) The status register so that future conditional jumps and interrupts have the correct information.

4.2.5 Context Switching

I will consider a context switch as a re-allocation of the processor to another task (a program/workspace pair). It is assumed that the original task will resume execution at some later time, so the program state vector must be saved and later restored. There are two basic kinds of context switches:

a) The CPU is voluntarily relinquished by the current task. The two instructions that cause a voluntary context switch are:

i) The BLWP instruction (Branch and Load Workspace Pointer).

ii) The XOP instruction (external operation).

b) The CPU is pre-empted by an external interrupt. This can be explained by pointing out that the basic processing algorithm shown in section 1.2.2 is somewhat more complicated in practice. For the TM9980 three external lines are read after each instruction is executed. If their binary value exceeds that of the interrupt mask (see section 4.2.3), then the CPU will be forced to execute in a new context. The context is selected by using the binary value of the three external interrupt lines as an index to load a new workspace pointer and program counter from a dedicated table in low memory.

The main difference between a relinquished and a pre-empted task is that the program state is known when relinquishing so only a partial PSV need be saved to resume execution. Pre-emptions on the other hand are determined by external signals so the state of the machine can rarely be predicted, and the complete PSV must be saved. However the architecture of the TMS9980 requires such a small PSV (WP, PC and ST) that there is little real point in distinguishing between the two types.

All context switches use the same basic mechanism (Figure 4.1). The PSV is saved when the workspace pointer, program counter and status register are copied into registers 13, 14, and 15 of the new workspace. The switch is completed when the workspace pointer and the program counter are loaded with their new values from the vector specifing them. All context switches use this vector to define the new workspace pointer and program counter. The vector is just two consecutive memory words specifying the new values of the workspace pointer and the program counter. For the BLWP instruction these may start at any address, but the XOP's and interrupt service routines require them in fixed, and reserved, locations.

Returning from a context switch is simply achieved using the RTWP (return workspace pointer) instruction. It moves registers 13, 14 and 15 from the current workspace into the

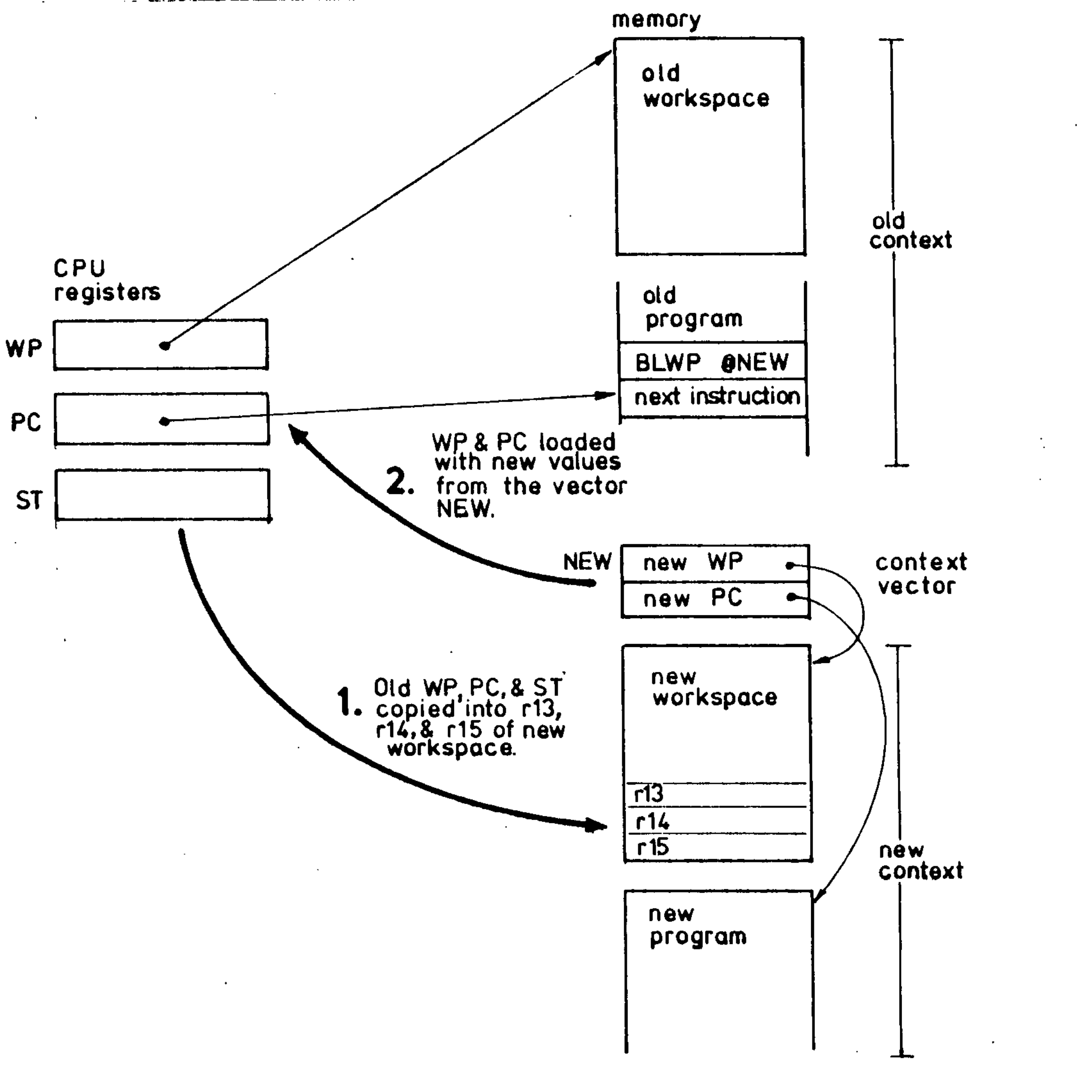


Figure 4.1 Context Switch using BLWP Instruction

workspace pointer, program counter and status register respectively.

4.3 THE TASK CONTROL BLOCK (TCB)

If the execution of tasks in the system is to be under control of the operating system, then information to effect this control must be maintained somewhere in RAM. Because task workspaces require blocks of 16 contiguous words it is convenient to assign a similiar sized block for task control.

```
TYPE
  tcb=
    RECORD
      pfre:@tcb;
      pxec:@tcb;
      ptmr:@tcb;
      psem:@tcb;
      ptsk:@tcb;
      perr:@tcb;
      wsp:@workspace;
      rsc:PACKED ARRAY [0..15] OF boolean;
      abt:boolean;
      nme:integer;
      alm:integer;
      pry:integer;
      wp:@workspace;
      pc:@instruction;
      st:register;
    END;
```

Notes:
1. The symbol @ designates the Pascal pointer type and can be read as "points to".
2. The types workspace, instruction, and register are <u>not</u> standard Pascal types. However they do give a true indication of the type of information, and could be defined.

Program 4.1 Pascal structure of a task control block

The <u>task control block</u> can be represented by the Pascal structure in program 4.1 and contains the following information:

1. List pointers which enable TCBs to be linked together.
2. The address of the task's workspace.
3. Resource flags which show the resources currently reserved by the task.
4. A flag which an executing task may reference to decide if a deferred abort has been set.
5. The name (or rather the number) assigned to the task.

The name identifies the task uniquely in the system.

6. The time that the task has to remain in an inactivated state. This value is set whenever the task is placed on the timer queue (see section 4.10). It is decremented each second, until zero. The waiting task is then reactivated.

7. The priority currently assigned to the task. This is a signed integer number.

8. The program state vector (PSV) is stored in three contiguous locations in the TCB (WP, PC and ST).

There is at present only one spare word in the TCB but more judicious use of the list pointers could free up another three. The task control block can be thought of as the basic building block of the control system structure.

4.4 LINKED LISTS

The most convenient and flexible way of joining TCBs together to build the system structure is to use linked lists. Because of the confusion between a linked list and the far more common stack structures a brief description of linked list processes is presented here.

4.4.1 The Headpointer

Also called the list base the headpointer holds the address of the first TCB on a list. Usually a headpointer is defined for each list in the system and an empty list is flagged by a zero value.

4.4.2 The Forward Pointer

The forward pointer is within the TCB at a known position. It holds the address of the next TCB on the list. If the address is zero then it is the last TCB on the list. A simple forward linked list is shown in figure 4.2.

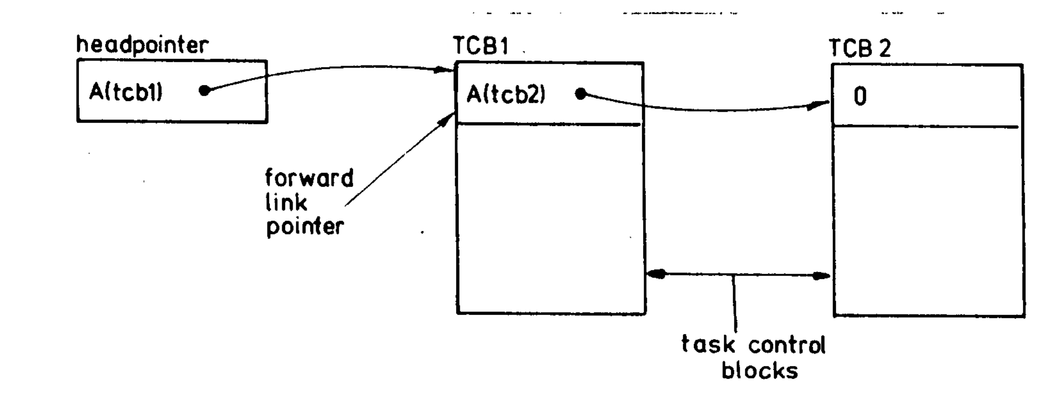


Figure 4.2 A Simple List Structure

4.4.3 Advantages of List Structures

List structures allow a far greater flexibility and simplicity than the stack structures commonly employed. Operating systems generally benefit from their introduction for many reasons.

1. There is no wholesale movement of memory contents. List structures are changed by replacing the link pointers.
2. Insertions and deletions are simply performed in a linked list by changing the values of the forward pointers in the appropriate TCBs. Stack operations would require wholesale movement of the stack contents to make room for new insertions, or to close the gap left by a deletion.
3. Searching a list for a TCB with a given keyword is

relatively simple.

4. By using multiple forward pointers it is easy to place one TCB on several lists at once. This is impossible on a stack without making multiple copies.

5. Finally a list is an inherently powerful structure, capable of great variety and form. Incorporating head pointers into the TCB allows the construction of sub-lists, branches, trees, and even potentially dangerous loops. List processing languages, such as LISP, are the "workhorses" of the artificial intelligence community.

4.4.4 List Operations

There is a basic set of list operations that are frequently necessary. They are presented here in terms of a call to a Pascal procedure, and a short explanation of their operation. For a detailed understanding it is best to refer to the list subroutines in Appendix II.

1. POPL(LIST,NEW,EMPTY): This routine removes the first TCB from the list that the headpointer "LIST" marks. It returns a pointer "NEW" that is the address of the popped TCB. In the event that the list was empty the boolean flag "EMPTY" is set to true (otherwise false) and "NEW" is unchanged. This routine mimicks the usual stack popping operation.

2. PUSHL(LIST,NEW): PUSHL puts a NEW TCB on the front of LIST. It is the complementary operation to POPL but must always succeed.

3. DELETE(LIST,NEW,KEY,KEYWORD,NOTFOUND): The action of DELETE is similiar to POPL except that it searches through the list until the KEYWORD is found. KEY specifies the position of the keyword in the TCB and NOTFOUND is set true if the keyword isn't found (possibly because the list is empty).

4. INSERT(LIST,NEW,KEY): INSERT behaves in a similiar fashion to PUSHL, only the NEW TCB is inserted before the first TCB it finds with a keyword at KEY less than its own. This is handy when inserting tasks of equal priority into a queue as it ensures that the later arrivals are queued <u>after</u> the earlier ones. It is also important to note that this routine assumes the keyword is already present in the new TCB. The new TCB is inserted at the end of the list if all other TCBs have larger keywords.

5. FIND(LIST,NEW,KEY,KEYWORD,NOTFOUND): Searches the LIST for a KEYWORD at KEY, and returns the pointer to its TCB in NEW if found. NOTFOUND is set true and the value of NEW is unchanged if the search fails. The list structure is left unchanged so this routine is handy when testing for the existence of tasks.

All system operations so far devised can be performed using these five list operations. Others that could possibly be of value would be the ability to add and remove TCBs from the <u>end</u> of lists, thus simulating first in, first out stacks.

The other important point regarding list operations is that they should be <u>indivisible</u>. This term means that interrupts must not be allowed to pre-empt these routines while they are in the process of changing list structures. Indivisibility is simply achieved by loading the interrupt mask with a zero (LIMI 0 instruction). Interrupts are thus prevented until a RTWP instruction is executed. This point applies to any part of the system that is actually manipulating the system structures.

4.5 MEMORY MANAGEMENT

Microprocessor systems consist of two very different types of memory with different properties. The first is read only memory (ROM) and while it is not possible for the microprocessor to alter ROM contents erasable/programmable types have been developed and given the acronym EPROM. Usually an ultra-violet lamp is required to do the erasing, and special high current drivers for the re-programming. The EPROM is convenient for programs and constants.

The other form of memory is known, confusingly, as random access memory (RAM) and it can of course be read and written. For any system, all data that is to be changed must reside in RAM and this includes task control blocks, workspaces, flags, semaphores and counters.

Consider now that a system has the potential to execute a large number of tasks. It is extremely unlikely that all these tasks will be in the system at once, especially those that only have a short duration (such as printing a message). A given amount of RAM can therefore support a larger task load if means are devised to dynamically allocate and de-allocate blocks of memory.

4.5.1 The Free List (Q.FRE)

This is simply a linked list of blocks of RAM that are available for use. One of the main problems of memory management is the process of fragmentation . If processes take as much RAM as they require, and if they require different amounts, there are always odd bits and pieces left over which are impossible to use. As time increases the number of the fragments increases until the memory management system becomes unusable. The simplest way to avoid this problem is to determine the largest block that the system needs and to give this to every task. There is undoubtedly some lack of economy with the approach but the wastage is not serious. Fortunately our system uses very similiar sized blocks for all its processes.

1. Workspaces : are 16 words long by necessity. Clearly no overlap of workspaces can be allowed in a dynamic situation.
2. Task Control Blocks : with all pointers, flags, names etc. considered a TCB requires 15 words of RAM. This could be reduced to 12.
3. Conversational Buffers : used to store "sentences" from the teletype for furthur processing. At two characters per word, thirty-two characters can be held in a sixteen word buffer. This size has proven very convenient, and in any case buffers are returned to the free list relatively rapidly.

The free list is therefore initialised with forty-two blocks of RAM each one 16 words long. This is enough to satisfy the requirements for sixteen rack tasks (32), two conversational buffers (2) and four other tasks (8).

```
PROCEDURE tcbget (VAR new:@tcb;none:boolean);
  VAR newwp:@workspace;
  BEGIN
    popl(qfre,new,none);
    IF NOT none
      THEN BEGIN
        popl(qfre,newwp,none);
        IF NOT none
          THEN new@.wsp:=newwp
          ELSE pushl(qfre,new)
      END
  END
```

Program 4.2 TCBGET - creates a TCB/workspace structure

4.5.2 Allocating and De-allocating Memory

With the construction of a free list of equal sized blocks of RAM memory management is an extremely straight-forward procedure. When a block is required it is simply POPLed from the free list, and likewise returned by using the PUSHL operation. There is no point in maintaining any kind of order in the free list and this therefore represents the optimal strategy.

Because new tasks commonly require both a TCB and a workspace a procedure has been defined to obtain these. It can be found under the name TCBGET in the operating system listing (Appendix II) and its operation is described by the Pascal procedure in program 4.2.

Likewise decomposing task structures is a relatively common operation and the procedure DISOLV, that does this, is listed in program 4.3.

4.6 ACCESSING TASKS

```
PROCEDURE disolv(VAR taskname:integer;notfound:boolean);
  BEGIN
    delete(qtsk,old,taskname,notfound);
    IF NOT notfound
      THEN BEGIN
        IF racktask(taskname,racknum)
          THEN BEGIN
            ledtab(racknum):=0;
            rcktab(racknum):=0
          END
      pushl(qfre,old);
      pushl(qfre,old@.wsp)
      END
  END
```

Notes:
1. "Racktask" is a convenient procedure that tests the task name to see if it was using a rack. The racknumber is returned when this is true.
2. "Ledtab" and "racktab" are system tables responsible for refreshing the contents of rack registers. Clearing these deactivates the appropriate rack (see section 4.17)

Program 4.3 DISOLV - dissolves the TCB/workspace structure

Although most system operations involve moving TCBs between various linked lists it sometimes happens that it is necessary to find a TCB when its position in the system structure is not known. Examples of this encountered so far are:

1. Checking for the existence of a new task name before generating it. Allowing two TCBs into the system with the same name would be fatal.
2. Check for the existence of a task whose purpose conflicts with a new one. The prime example of this is that tasks for both manual and automatic control of the same rack can interact with completely unforeseen results.

3. Aborting tasks is a tricky business. Unless a task is in some known condition (such as an error condition) aborts should only be performed by the task itself. This is made possible by setting a flag in the TCB which the task can test to conditionally execute a self destruct procedure (called a deferred abort). Resource de-allocation and shutdown is then performed in an orderly manner. By testing for existence of the task before setting the flag the possibility of an operator error is minimised.

4.6.1 <u>The Resident Task List (Q.TSK)</u>

Instead of performing an extensive search of all lists, queues and pointers in the system it is much simpler to retain all tasks currently active on a special linked list, known as the <u>resident task list</u> .

A task has its TCB PUSHLed here when first introduced into the system, and then DELETEd when finished. In between time a given task name is easily found by using the FIND operation. Not only is the existence of the task verified but parameters in its TCB are easily tested, altered, set or changed.

4.7 <u>EXECUTION OF TASKS</u>

The primary function of any operating system must be to execute tasks in a logical fashion. It is reasonable to expect that some tasks will also have a greater priority for the CPU than others. Structures and routines must be devised to account for this.

4.7.1 The Execution Queue (Q.XEC)

The simplest way of executing tasks logically is to define a linked list of task control blocks that are priority ordered. When the CPU becomes available the highest priority task can be removed from the queue and executed. Our system varies the scheme slightly. Rather than having the first task on the execution queue as the highest priority task waiting for the processor, it is actually the task currently executing . When it no longer requires the CPU (of its own volition or if it is pre-empted) then it is either POPLed from the queue or has a higher priority task INSERTed before it. Operation always continues with the execution of the first task on the queue.

There is the question of what to do when the execution queue becomes empty. Usually the processor is sent into a loop or an idle state that can be interrupted in some fashion. Because our system is driven by user commands from teletype it makes sense to execute a tight loop that monitors the keyboard for operator input. Such a program could easily be a part of the operating system, but a neater solution is to place the loop within a low priority task. The task is forever checking the teletype or decoding commands so it never really finishes, and the operating system never has to contend with an empty execution queue. This task is called the command task (TSKCMD) and it is necessarily the lowest priority task in the system. A task of lower priority might never execute.

4.7.2 Dispatching or Allocation

The execution queue ensures that the highest priority task will be allocated the CPU at the first possible opportunity.

However routines must be devised to actually dispatch the task to the processor. This is simply achieved by remembering that a task is executed when its program state vector is loaded into the appropriate internal registers. As already mentioned, the only practical way of doing this is to use the RTWP instruction. Dispatching is, therefore, a simple procedure which moves the PSV from the first TCB on the execution queue (procedure PSVGET) and into registers 13, 14 and 15 of the supervisor workspace. These steps are followed by the RTWP instruction. Figure 4.3 should clarify the operations.

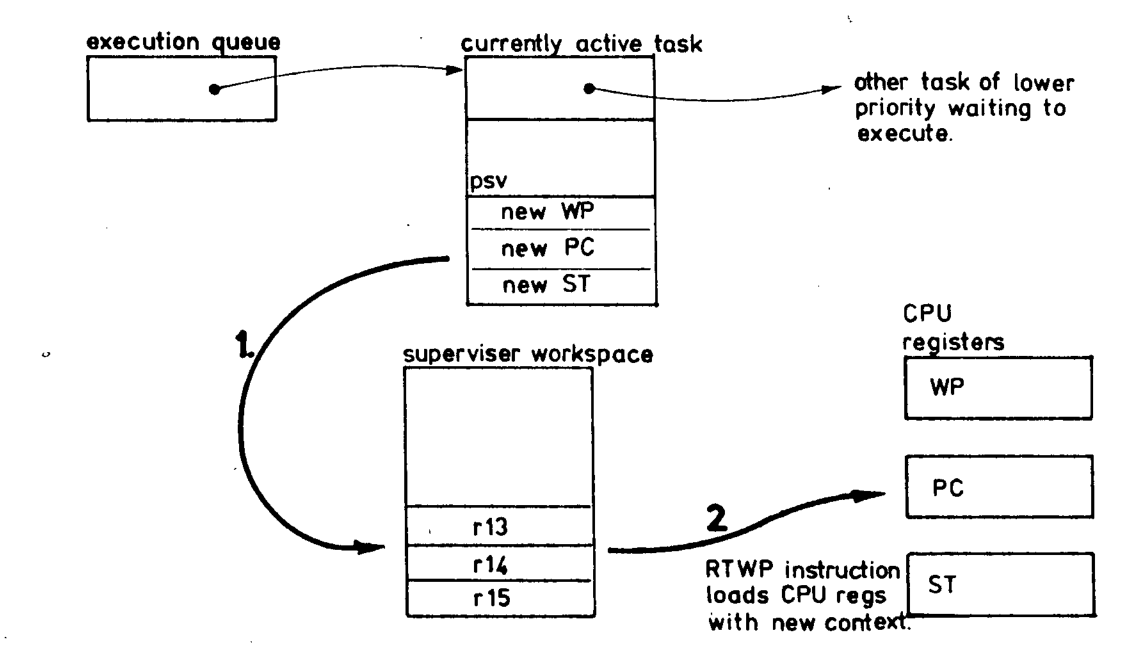


Figure 4.3 Dispatching a Task for Execution

4.7.3 De-allocation

The operating system routines are used by relinquishing tasks calling them from the BLWP instruction, or in response to external interrupts pre-empting the currently active task. In either case the PSV of task is saved in registers 13, 14 and 15 of the supervisor workspace. If the task is to be correctly dispatched at some later point then the PSV must be moved from the supervisor workspace to the first TCB on the execution queue before any changes are made in the queue. A procedure PSVSAV has been written to do this. In some cases no new tasks may be inserted into the execution queue so there is no point in wasting time and memory saving the PSV. As long as the three last registers of the supervisor workspace are not altered, task execution is simply resumed using only the RTWP instruction.

4.8 STARTING THE SYSTEM (STARTUP)

Before any system can function correctly there are inevitably a whole series of devices, flags, pointers, counters, etc. that must be initialised. Upon entry the operating system performs the following sequence of initialisations which can be sought in greater detail in the program listing and in the instruction manual for the microprocessor (Texas Instruments, 1979).

4.8.1 Startup Procedure

1. Initialise a periodic 200 ms timer in the user I/O port, set up the interrupt vector, and enable its interrupt capability.
2. In a similiar fashion, initialise a 1ms timer in the system I/O part.
3. Initialise the XOP vectors (see instruction manual).
4. Initialise the analogue/digital converter hardware, permit

interrupts, and set its interrupt vector.

5. Initialise and allow interrupts from the CMOS expansion bus (from racks and inlet line).

6. Create the free list.

7. Clear all lists and queue headpointers.

8. Initialise semaphores.

9. Clear rack and LED refresh tables.

10. Initialise workspace for display timer interrupt service routine.

11. Initiate ASCII clock and set current prompt or prefix character.

12. Initiate the clock update task and place it on the clock interrupt wait pointer.

13. Initialise the command task.

14. PUSHL the command and clock update tasks to the resident task list.

15. INSERT the command task on the execution queue.

16. Dispatch to the first task on the execution queue.

After this sequence is completed the command task will be executed and will start scanning the teletype input port for a character. Interrupts will arrive from the clock timer every 200 ms and the clock update task will consequently be queued and executed every second. Every 1.0 ms the service routine for the display timer interrupt will clear rack registers thus making sure that all solenoids and control lines are off or inactive 16 ms after the STARTUP procedure is entered.

In the present implementation the sample line operating system (SLOS) is entered using the J command from the UNIBUG monitor (Texas Instruments, 1978). The monitor itself is entered whenever power is applied to the microprocessor board, or when the board mounted load switch is closed. A more satisfactory procedure would be to enter SLOS directly on power up but this would require changing the load vector in the UNIBUG program. Various hardware approaches could possibly be used but all of these involve making physical changes to the printed wiring on the microprocessor assembly. These modifications will be made when time permits.

4.9 INTRODUCING NEW TASKS (RELINQ)

After start up, the processor is occupied with updating the time-of-day clock and scanning the teletype port for input commands. At some stage the command task will receive a user command, that to be executed must result in the generation and introduction of a new task into the system. In order to do this the command task must relinquish control of the CPU once it has composed and initialised a TCB and workspace. A system routine RELINQ is provided expressly for this purpose, but is in fact useful for introducing any task, whether new or just previously inactive, back into the execution queue. The RELINQ operation is shown in Program 4.3. It simply saves the PSV of the relinquishing task, inserts the TCB of the new task into the execution queue (and the resident task list if not already there), and then dispatches the highest priority task to the CPU. There is the added complication that two tasks with the same name (but using different TCBs) must not be allowed to co-exist. If this condition exists the name of the new task is incremented until it is unique. This proved convenient,

```
PROCEDURE relinq(VAR new:@tcb);
  CONSTANT the-universe-dies=false;
    VAR
      same:@tcb;
      notfound:boolean;
    BEGIN
      limi(0);
      IF new=NIL THEN systemerror;
      REPEAT
        BEGIN
          find(qxec,same,name,new@.name,notfound);
          IF notfound OR same=new
          THEN BEGIN
            psvsav;
            insert(qxec,new,priority);
            psvget;
            rtwp
          END
          new@.name:nw@.name+1
        END
      UNTIL the-universe-dies
  END
```

Notes:
1. "Systemerror" is a diagnostic routine in the event of system error.
2. Control leaves this procedure when either "systemerror" or "rtwp" is executed.
3, The procedure could repeat indefinitely if no unique new name is ever found (i.e. until the universe dies).

Program 4.3 RELINGQuishing the CPU

especially when logging rack interrupts, but could prove troublesome if the task name is incremented too far (highly improbable).

4.10 INACTIVATING TASKS FOR A GIVEN TIME

It is frequently the case in a real time system that a task activates something external (i.e. in the real world) and then must wait for a certain time until it can proceed. In our case common examples involve waiting for gas flow rates to stabilise during transfer stages, or for pressures to stabilise during pump down or leak testing.

As the CPU can do nothing for the task except loop continuously, it is wasteful to use this valuable resource in such a manner. Instead it makes a good deal more sense to inactivate the waiting task until it is ready to execute again. In the meantime the processor can proceed with other tasks (usually the command task).

4.10.1 The Timer Queue (Q.TMR)

Structuring the operating system around linked lists allows the programmer to create as many special purpose queues as the system requires. Consequently it is easy to define a queue on which tasks are placed until a certain time period has elapsed. The timer queue can be updated at a convenient unit of time (every second) and TCBs can be deleted from it and replaced on the execution queue when their time has elapsed. As usual it is convenient and straightforward to order tasks in terms of their priority.

4.10.2 Setting the Alarm (WAIT)

Tasks that wish to wait for a given period of time execute a BLWP to the superviser/operating system routine WAIT, with register zero of the workspace set to the time period in seconds. The WAIT routine transfers the period to its appropriate place in the task control block after POPLing the block from the execution queue. The waiting TCB is INSERTed into the timer queue, and the next highest priority task on the execution queue is allocated the CPU. The procedure is simple and straightforward. It is shown in Program 4.4.

```
PROCEDURE wait(VAR time:integer);
  VAR temp:@tcb;empty:boolean;
  BEGIN
    limi(0);
    psvsav;
    popl(qxec,temp,empty);
    IF empty THEN systemerror;
    temp@.alarm:=time;
    insert(qtmr,temp,priority);
    psvget;
    rtwp
  END
```

Program 4.4 Procedure to place TCB on timer queue

4.10.3 The System Clock (SRVCLK)

The University Kit board has two interface devices, both of which have internal, programmable timers. The timers generate a periodic interrupt which pre-empts current program execution. In particular the user I/O port is initialised to generate an interrupt every 200 ms. This interrupt causes entry to the service routine SRVCLK, which in turn executes the following actions (Program 4.5).

1. Resets and re-enables the clock interrupt.
2. Decrements a tic counter. This is set to a value of five during the start up procedure and therefore reaches zero after one second has elapsed.
3. If the tic counter is still positive then an RTWP instruction is executed, continuing the pre-empted program. Otherwise the following actions are performed.
4. Check the interrupt pointer for the clock (INT.CLK). If

this is zero (or empty) then the clock update task is busy, and no action can be taken, so the RTWP sequence is executed.

5. If however the clock update task is ready then the pointer will be non-zero. The TCB is inserted in the execute queue, the pointer is zeroed (flagging that the update task is busy), and the highest priority task is dispatched to the CPU.

The highest priority task is usually the clock update task itself, and so will execute as soon as the clock interrupt has been serviced.

```
PROCEDURE srvclk
  VAR intdk:@tcb;ticctr:integer;
  BEGIN
    limi(0);
    resetclockinterrupt;
    ticctr:=ticctr-1;
    IF ticctr.0
      THEN rtwp;
      ELSE
        IF intclk=NIL
          THEN rtwp
          ELSE BEGIN
            psvsav;qxec
            insert(qxec,intclk,priority);
            intclk:=NIL;
            psvget;
            rtwp
          END
  END
```

Program 4.5 System clock service routine (SRVCLK)

4.10.4 The Clock Update Task (TSKCUD)

Primarily designed for updating and removing tasks from the timer queue, it actually performs two related but important functions.

1. All messages output on the teletype are prefixed by the time since system startup. This time is stored internally in the form of ASCII characters for the hours, minutes and seconds (the format is HHMM:SS) suitable for direct output to the teletype. The clock update performs an ASCII increment of its value.

2. The tic counter must also be updated, but this is simply a matter of subtracting five from its value. Using the tic counter in this fashion has two advantages. It allows clock interrupts to continue even though the clock update is executing, and it insures that clock interrupts cannot be "missed" if, say, the clock update task happened to be pre-empted for a rather long time. This is why the service routine for the interrupts is able to RTWP if the update task is not ready (INT.CLK is zero). The main function though is for the clock update task to work its way through the timer queue decrementing the alarm times of each TCB it finds. When such a decrement results in a zero value the TCB is DELETEd from the timer queue (and the error queue if there as well - see section 4.11.). The clock update task, then RELINQuishes, but because of its high priority continues processing the timer queue. In this way tasks are returned for execution once their time period has expired.

As mentioned before the clock update task alters the structure of the system when it deletes TCBs from the timer (and error) queues, then inserts them to the execution queue. Nothing else must be allowed to alter structure during this time so all interrupts are masked out to prevent the possibility (See

Program 4.6). When the clock update task has finished it waits for the next clock interrupt on the clock wait pointer (see section 4.13).

```
PROCEDURE tskcud
  VAR curr,new,intdk:@tcb;
      ticctr,beeper:integer;
      empty,notfound:boolean;
  BEGIN
    increment ascii clock;
    ticctr:=ticctr+5;
    curr:=qtmr;
    WHILE curr=NIL DO
      BEGIN
        curr@.alm:=curr@.alm-1;
        IF curr@.alm.0
          THEN BEGIN
            popl(curr,new,empty);
            delete(qerr,new,name,new@.name,notfound)
            IF NOT notfound THEN beeper:=beeper-1;
            relinq(new)
          END
        curr:=curr@.tmr
      END
    waitint(intclk);
  END
```

Notes:
1. "Increment ascii clock" is a tedious procedure with effect suggested.
2. "Beeper" controls the rate of the audible alarm beep. Re-executing a task in an error condition requires resetting this.
3. "Waitint" sits the task on the interrupt pointer specified.

Program 4.6 The clock update task (TSKCUD)

4.11 ERROR CONDITIONS

To function correctly a real time processing system such as ours requires a prescribed set of conditions in the real world. Although it is possible to design a task to behave intelligently when conditions vary such an undertaking usually requires a good knowledge of the most probable faults and errors. Because such experience is lacking at an early stage of development, and because it is impossible for the processor to correct all errors, then tasks finding it impossible to proceed must be inactivated under the following criteria.

1. Facilities must be provided to notify the operator that the error has occurred. Information must be presented to identify the error.
2. The operator must be able to decide the fate of such a task. Should the task be aborted or is it possible to effect some recovery action?
3. Some conditions are temporary. An example of this is that evacuating the sample reservoirs fails on a pressure test. This may be due to the presence of a small droplet of water so the evacuation would have been successful if it was tried again (or a number of times). It therefore seems reasonable to provide a time out procedure whereby the task initiates its own recovery action if the operator does not respond within a given period.

4.11.1 The Error Queue (Q.ERR)

As usual it is simplest to define a priority-ordered, linked list of TCBs for tasks that are unable to proceed because external conditions are not correct. The task itself must "discover" the condition and is responsible for placing itself on the queue.

4.11.2 Getting on the Queue (WAIT.ERR)

When a task needs to wait for operator intervention it executes a BLWP to WAIT.ERR (Program 4.6). The following steps are executed.

1. Increase the beeper rate (If not on, the audible alarm will sound).
2. Insert the calling task into the error queue.
3. If a time out period is specified (non-zero) then insert the calling task into the timer queue as well.
4. Dispatch to the highest priority task on the execution queue.

Note that this routine <u>does not</u> generate any messages regarding the nature of the error; it was found more practical to let the calling task do this. Nor does the routine releases any resources it has held; this can be useful when a resource, such as the vacuum pump, is found to be at fault. Not releasing the resource will prevent other tasks from using it.

4.11.3 Getting Off the Error Queue

There are three different ways of getting off the error queue and two of them are initiated by operator commands to the <u>command task</u> (see chapter 5).

1. In the event that the operator feels that there is no simple way of correcting the error (the rack has to be dismantled, say) then the operator issues an ABORT command for the appropriate task number. This will DELETE the task from the error queue,

the timer queue (if there), will deallocate any resources held,

```
PROCEDURE waiterr(VAR time:integer);
  VAR temp:@tcb;empty:boolean;
  BEGIN
    limi(0);
    psvsav;
    popl(qxec,temp,empty);
    IF empty THEN systemerror;
    beeper:=beeper+1;
    pushl(qerr,temp);
    IF time=0
      THEN BEGIN
        temp@.alm:=time;
        insert(qtmr,temp,priority)
      END
    psvget;
  rtwp
  END
```

Program 4.6 Waiting for errors (WAIT.ERR)

and will return the TCB and workspace to the free list.

2. If however the fault was simply corrected (e.g. opening that forgotten valve on the carbon dioxide cylinder) then issuing the RECOVER command (and RETRY, REDO, or RESTART) will DELETE the task from the error and timer queues, and re-insert it into the execution queue. When the task finally resumes execution it will do so with the first instruction <u>after</u> the call to WAIT.ERR. It is up to the task itself to take appropriate recovery action.

3. When the operator fails to respond within the time out period the clock update task initiates the actions in (2) above.

4.12 RESOURCE MANAGEMENT

Any system inevitably shares its resources amongst the various tasks. So far the system has been designed to allocate two of these: the central processing unit and blocks of random access memory. There are however a number of external resources which must not be used simultaneously.

1. The teletype and its associated routines can only be used by one task at a time. Even if it was possible to re-enter the teletype XOPs, messages would be confusingly mixed together.
2. The main vacuum line is used for evacuation, transfer of CO_2, and for pressure measurement. Obviously competing tasks should not be allowed to change any of these conditions.
3. The mass spectrometer line to the inlet controller must likewise be dedicated to one task at a time if sample mixing and contamination are to be avoided. The peltier cooler and the inlet line controller can be grouped in with this resource.
4. The analogue/digital converter should not be re-initialised until it has delivered a requested result.

4.12.1 Semaphores and the Critical Section

It is usual to term the area of a task that updates or uses common resources as the critical section (CS), (Shaw, 1974, pg 59) and many different schemes have evolved to provide the protection needed. The most general and easily implemented approach, first developed by Dijkstra (1965, 1968), uses two semaphore primitives to ensure mutual exclusion of the critical section. If S is a semaphore variable then the following two operations are defined:

1. V(S) : this simply increases S by 1 in a single indivisible action (i.e. no interrupts).
2. P(S) : decrements S by 1 unless S=0 in which case the task proceeds no furthur until it can (ie until a V(S) is executed by another task).

It is possible to protect an indefinite number of critical sections using the code in program 4.7.

```
PROCEDURE system
  VAR mutex:integer;
  BEGIN
    mutex:=1;
    .. .
    PROCEDURE task;
      BEGIN
        ...
        ...
        P(mutex);
        criticalsectionij
        V(mutex);
        ...
        ...
      END                             {taski}
    ...
    ...
  END                                 {system}
```

Program 4.7 Using semaphore primitives to protect a Critical Section

The initial value of the semaphore variable is actually a measure of the number of units of the resource that are available. Although all resources in our system should be initialised to one, an important exception could be the solenoid power supply. With a bank of solenoids consuming approximately 4 A, then a power supply capable of supplying

20 A could be protected from over use by initialising a semaphore to 5. The primitive P(S) is used every time a bank of solenoids is activated; and V(S) after they are de-activated. This is an example of the "producer-consumer problem" that Dijkstra's semaphores were designed to solve. The primitives V(S) and P(S) have been implemented for the sample line operating system as the procedures RELESE and RESERV respectively, but before describing these routines there is the ever present issue of structure to solve.

4.12.2 Semaphore Structure - the Semaphore Table (SEMTAB)

Dijkstra's semaphore primitives are conceptually simple (and elegant) but implementing them requires the following information.

1. A place to store the value of the semaphore, or the number of available resource units.
2. The primitive P(S) requires that the calling task wait until the value of S is positive. It is easiest to define a list of TCBs that are waiting to use the semaphore's resource. In that case a place is needed for the semaphore wait list head pointer.
3. It is also extremely convenient to keep tabs on the resources currently held by a task. For this reason a word in the task's TCB is allocated for resource flags . When a flag is set its corresponding resource is in use by the task.

A semaphore is conviently thought of as a pair of words, the first the number of resource units available, and the second the address of the first TCB on its wait list. All the semaphores are grouped together in a semaphore table and the position of a semaphore in a table corresponds to a bit in the TCB word assigned to resource flags. In this way semaphores can be referred to by name when writing Assembly language programs, yet can also be inserted and deleted from the semaphore table at will. The appropriate bit flag is easily calculated from the semaphore CALCBIT (see the operating system listing in Appendix II). The Pascal structure for the semaphore table is in Program 4.8.

```
TYPE
  semaphore=
    RECORD
      units:integer;
      waitlist:integer;
    END;
CONSTANT
  semmax=4; {in current implementation}
VAR
  semtab:ARRAY[1..semmax]OF semaphore;
```

Program 4.8 Semaphore structure (SEMTAB)

4.12.3 Reserving a Resource (RESERV)

This routine is entered by a task wishing unimpeded use of the named resource (passed as an argument - the address of a semaphore). The following steps are executed. (Program 4.9)

1. The number of resource units is decremented.
2. If the result of (1) is positive or zero, then no other tasks are using the resource so it is permissible to proceed. The appropriate resource flag is set in the tasks TCB and execution continues with an RTWP instruction.
3. If however step (1) generates a negative number of units

then the resource is unavilable. After saving the PSV for the calling task its TCB is POPLed from the execution queue then INSERTed into the semaphore wait queue. The first task on the execution queue is then dispatched to the CPU in the usual manner.

It may be useful to note here that this scheme differs slightly from Dijkstra's in that the number of resource units can be negative. However this makes no real difference to semaphore operation and is sometimes useful when examining the state of the wait lists. The magnitude of a negative semaphore indicates the number of tasks waiting to use that resource.

4.12.4 Releasing a Resource (RELESE)

When a task has finished using a resource it should be released for use by other tasks using a BLWP @RELESE insruction. The procedure is equivalent to the P(S) primitive except that it checks to see if any tasks are waiting to use that resource. If they are, it POPLs the highest priority one from the semaphore wait queue and INSERTs it into the execution queue. The highest priority task then begins executing. The equivalent Pascal procedure in Program 4.9 should be self explanatory.

4.12.5 Using RESERV and RELESE

Reserving and releasing resources is usually a matter of calling RESERV, using the resource, then calling RELESE when finished (with the appropriate semaphore address of course). However when several resources are held by two or more tasks concurrently it is possible to arrive at a circular wait

```
PROCEDURE reserv(VAR sa:@semaphore);
  VAR
    empty:boolean;
    temp:@tcb;
    snum:integer;
  BEGIN
    limi(0);
    sa@.units:=sa@.units-1;
    IF sa@.units.0
      THEN BEGIN
        psvsav;
        popl(qxec,temp,empty);
        IF empty THEN systemerror;
        insert(sa@.waitlist,temp,priority);
        psvget;
        rtwp
      END
      ELSE BEGIN
        calcbit(sa,snum);
        qxec@.rsc[snum]:=true;
        rtwp
      END
  END
```

Program 4.8a Reserving resources (RESERV)

condition known affectionately as *deadlock* . A simple case of this for two tasks is shown in figure 4.3 where task A holds the main line and requires the teletype, while task B holds the teletype but cannot release it until it gets the main line. Clearly this condition is impossible to exit, hence the name deadlock.

Not only is deadlock disastrous, but it is also common. Many methods have evolved to detect and prevent it (Shaw, 1978, Chapter 8) but the most pertinent in our case is for the programmer to *order* resource calls so that the condition never occurs. In the example shown in figure 4.3 deadlock would be prevented if *both* tasks always asked for the mainline first. Similarly if a critical section requires

```
PROCEDURE relese(VAR sa:@semaphore);
  VAR
    temp:@tcb;
    empty:boolean;
    snum:integer
  BEGIN
    limi(0);
    calcbit(sa,snum);
    qxec@.rsc[snum]:=false;          {reset resource flag}
    sa@.units:=sa@.units+1;
    IF sa@.units.0
      THEN BEGIN
        popl(sa@.waitlist,temp,empty);
        IF empty THEN systemerror;
        temp.rsc[snum]:=true;          {set resource flag}
        psvsav;
        insert(qxec,temp,priority)
        psvget;
        rtwp
      END
      ELSE rtwp
  END
```

Program 4.9 Releasing resources (RELESE)

several resources at once then deadlock is avoided by requesting resources in the sample line system in the following order.

1. The mass spectrometer line. (S.MSL)
2. The main line. (S.MNL)
3. The teletype. (S.TTY)
4. The analogue/digital converter. (S.ADC)

Importantly they must be RELESEd in the reverse order.

4.13 WAITING FOR INTERRUPTS (WAIT.INT)

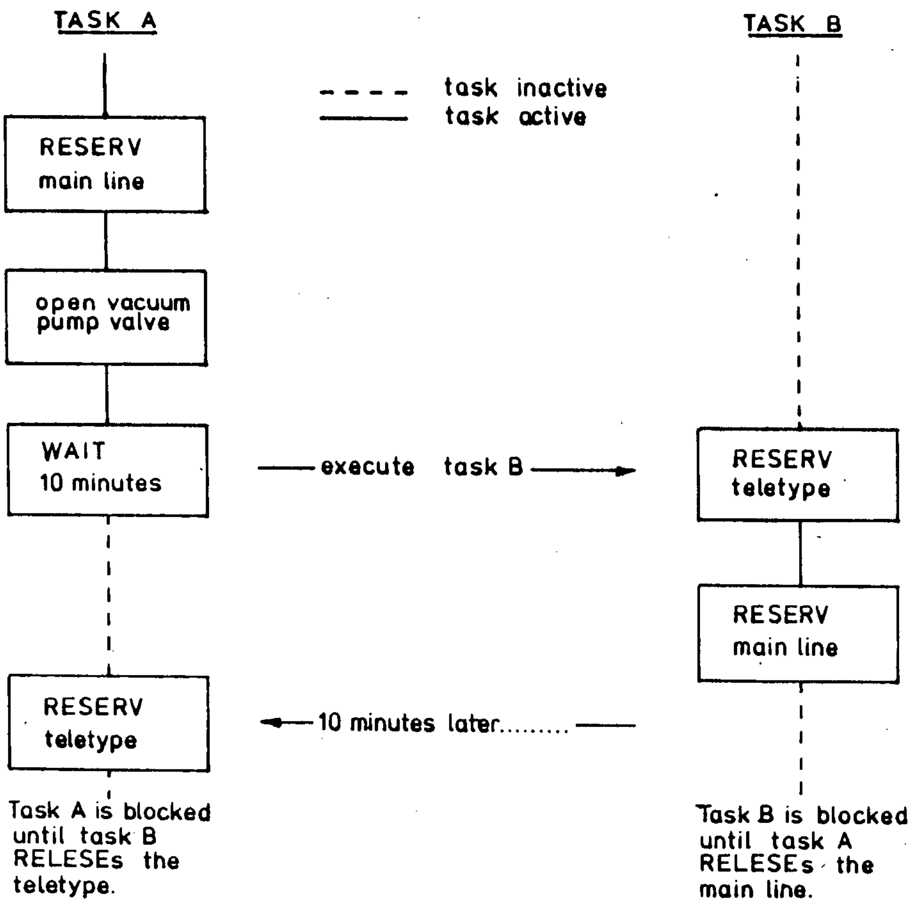


Figure 4.4 An Example of Deadlock

Devices in the real world are usually very slow to respond to the CPU. In some cases interrupts are completely asynchronous and no prediction can be made about their arrival. To avoid unnecessary use of the CPU under these conditions it is necessary to define pointers on which task TCBs can be placed while waiting for an interrupt from a particular source. The procedure in program 4.10 is basically very straightforward. The calling task is removed from the execution queue, is placed on the interrupt pointer specified, and the next highest priority task is executed.

At present interrupt pointers exist for three sources: the inlet controller (INT.INL), analogue-to-digital converter (INT.ADC) and the clock timer (INT.CLK).

```
PROCEDURE waitint(VAR intptr:@tcb);
  VAR
    temp:@tcb;
    empty:boolean;
  BEGIN
    limi(0);
    psvsav;
    popl(qxec,temp,empty);
    IF empty THEN systemerror;
    intptr:=temp;
    psvget;
    rtwp
  END
```

Program 4.10 Waiting for Interrupts (WAIT.INT)

Removing TCBs from the interrupt pointers is actually a function of the appropriate interrupt service routine (ISR) but in general once the source of the interrupt is determined its pointer is checked for the presence of a waiting task (the pointer is non-zero). If a task is waiting then it is inserted in the execution queue and the highest priority task is dispatched to the CPU.

4.14 SERVICING INTERRUPTS

The TMS9980 has six levels or priorities of interrupts of which four are immediately accessible to the user. Each level has a dedicated vector in RAM in which is specified the workspace and execution address of the interrupt service routine. An interrupt effectively causes a BLWP to be executed from this vector after the current instruction is completed. It also forces execution of the first instruction in the ISR enabling the interrupt mask to be set, and hence preventing pre-emption of the routine. The table is shown in more detail in the RAM definition section of the operating listing (Appendix II), and in the Unibug Users Manual (Texas Instruments, 1978).

4.15 SERVICING CMOS BUS INTERRUPTS (SRVBUS)

Interrupts arriving from the CMOS bus can have one of two sources.

1. A sample request from the inlet line controller.
2. As a result of a change in the state of a door switch or relay power supply on a rack.

The routine first tests for the inlet line controller interrupt. If this was the source then it will queue the task waiting on its wait pointer (INT.INL). When this pointer is empty nothing else is done and the pre-empted task regains the CPU after an RTWP instruction is executed.

If the inlet line was not the interrupt source each of the sixteen rack status words are tested in turn. The first rack found with its interrupt flag set will queue a task that logs the event on the teletype (TSKLOG). This task generates a message indicating the rack number and its status word then FINISHes (see Appendix II). After inserting the log task into the execution queue the usual dispatch procedure is executed.

Note that this scheme is not the best way to handle sequential interrupts because of the return as soon as one rack has been found causing the condition. Sequential interrupts are of such low probability however that there is no real advantage in designing for them.

```
PROCEDURE srvbus
  VAR
    rnum:0..15;
    new:@tcb;
  BEGIN
    limi(0);
    IF interrupt(master)
      THEN BEGIN
        psvsav;
        insert(qxec,intinl,priority);
        psvget;
        rtwp
      END
      ELSE
        FOR rnum:=0 TO 15 DO
          IN
            IF interrupt(rack[rnum])
              THEN BEGIN
                createlogtask(new,rnum);
                psvsav;
                insert(qxec,newpriority);
                psvget;
                rtwp
              END
    rtwp
  END
```

Program 4.11 CMOS Bus interrupt service routine (SRVBUS)

4.16 USING THE ANALOGUE-TO-DIGITAL CONVERTER

This is most conveniently accessed by executing the instruction BL @ADCGET (see Appendix II for argument specifications). This routine RESERVs the teletype, initialises its channel number (from 0 to 15), waits for an ADC interrupt, reads the ADC value from the output latches (resetting the interrupt at the same time) and stores it in the workspace of the waiting task. When the waiting task resumes execution it RELESEs the ADC for other users. Both the ADCGET and the interrupt service routine are listed in Program 4.12.

One furthur point is that the value initially extracted from the ADC contains two flags indicating polarity and overflow (Intersil, 1981). ADCGET tests these and alters the value to give a correct 2's complement integer, and flags an overflow if it occurred.

4.17 CHANGING RACK REGISTERS

The rack hardware registers have been designed to resemble part of memory, and it is quite feasible to set them by writing a word to the appropriate address. However in the interests of hardware simplicity it is not possible to read from these registers and many convenient machine operations will therefore function incorrectly (SOC, SZB, shifts and arithmetic operations). As well, it is possible to lose the contents of these registers in the event of a power failure on a rack and it was therefore felt that an advantage could be gained by holding an <u>image</u> of the rack registers somewhere in RAM. Realising that the image could be altered far more conveniently than the registers themselves, a structure and a process were devised to continually <u>refresh</u> the rack registers from RAM.

4.17.1 <u>The Rack Refresh Table (RCKTAB)</u>

```
PROCEDURE adcget(VAR channel:0..15;
                     value  :integer;
                     overflo:boolean)
  BEGIN
    devadc:=channel    {set channel number}
    reserv(sadc);
    waitint(intadc);
    relese(sadc);
    correct(value)     {correct sign and test for overflow}
  END

PROCEDURE srvadc
  BEGIN
    limi(0);
    IF intadc=NIL THEN systemerror
    value:=devadc;     {get adc value}
    psvsav;
    insert(qxec,intadc,priority);
    intadc:=NIL;
    psvget;
    rtwp
  END
```

Program 4.12 Getting ADC values (ADCGET & SRVADC)

The rack refresh table is a block of sixteen words in RAM. Each word holds the <u>address</u> of (or "points to ") an image of the rack registers. I have termed this the <u>rack state vector (RSV)</u> . If a table entry is zero then the corresponding rack is inactive. The RSV can be conveniently held in the workspace of the task using the rack. The pointer to the RSV is initialised by the task when it first executes, and is cleared as it FINISHes.

4.17.2 <u>The LED Table (LEDTAB)</u>

Eight LEDs are provided in a conspicuous location on the rack front panel and these can be set by writing ones to the appropriate bits of the rack control word. As well a facility to <u>flash</u> LEDs at the same rate at which the audible alarm beeps has been provided by setting bits in the appropriate entry of the LED table. This is a block of 16 <u>bytes</u> (8 bits each) of RAM, one byte being dedicated to each rack.

4.17.3 Refreshing the Racks (SRVDSP)

Every 1 ms the display timer pre-empts the currently executing task and executes its service routine. The display service routine conveniently handles a number of jobs and operates in a sense that is independent of the operating system. It has its own dedicated workspace, has no TCB, and alters none of the system structure, but performs the following jobs.

1. Sounds the audible alarm at the rate determined by "beeper".
2. Flashes the LEDs marked in the LED table, at a rate dependent on "beeper". Note however that LEDs can always be made to flash even if the audible alarm is silent.
3. Refreshes one rack every 1 ms from the contents of the rack state vector. If the rack table entry is zero it clears the rack register.
4. Besides the rack registers, the <u>master control word</u> is refreshed every 1 ms from its corresponding RAM image, as well.

The scheme has proven extremely convenient because of the ease it allows for altering rack registers. The other approach would be to write routines specifically for this pupose, but by the time code and argument lists are considered such routines would not only be uneconomical in terms of EPROM space, but also for their awkwardness.

```
PROCEDURE finish
  VAR
    fin,new:@tcb;
    rnum:0..15;
    empty,nameunknown:boolean;
  BEGIN
    limi(0);
    popl(qxec,fin,empty);
    IF empty THEN systemerror;
    disov(qtsk,fin,name,fin@.name,nameunknown);
    rnum:=0 IF nameunknown THEN systemerror;
    WHILE anyflagset(fin@.rsc) DO
      BEGIN
        IF fin@.rsc[rnum]
          THEN BEGIN
            semtab[rnum.units:=semtab[rnum].units+1;
            IF semtab[rnum].units.0
              THEN BEGIN
                popl(semtab[rnum].waitlist,new,empty);
                IF empty THEN systemerror;
                new@.rsc[rnum]:=true;
                insert(qxec,new,priority);
              END
            rnum:=rnum+1
          END
        psvget;
        rtwp
  END
```

Notes:
1. "Anyflagset" is a boolean function. It returns a value of true if it finds any true flags in the packed array in the argument list.

Program 4.13 Finishing Tasks (FINISH)

Detailed operation of SRVDSP is best understood by referring to the listing of the operating system (Appendix II). It is a straightforward application of various flags to pulse the piezo-electric transducer, and continuously incremented indexes to enter the rack and LED tables.

4.18 FINISHING TASKS (FINISH)

When a task has finished it must be able to inform the operating system of the fact. Its resources can then be deallocated, its TCB and workspace returned to the free list, and the CPU assigned to the next pending task. These operations are all accomplished by executing a BLWP @FINISH instruction (Program 4.13).

CHAPTER V. THE OPERATOR INTERFACE

5.1 A DEFINITION

Our sample preparation line has been presented as an almost self-contained system, capable of operating in an unattended mode, with a minimum of human intervention. Under normal conditions this is true, but there are many instances where the human operator must play a more significant role.

There are, for example, testing procedures that can be introduced at the operator's discretion, and it is sometimes necessary to abort tasks, either because they have been introduced inadvertently, or because they are no longer needed. Overriding manual control is vital, and may allow the operator to avoid a recurrent error by manually stepping through the troublesome procedure. In such cases it is useful to provide a number of entry points to the preparation algorithm, and to allow the operator the power of continuing processing at any one of them. Manual overrides also permit new procedures to be tested and refined before committing them to unattended operation.

The sample line therefore requires a sophisticated channel of communication between its controller and the human operator. Unfortunately, both participants speak different languages, and it is necessary to bridge the gap with both hardware devices, and software protocols. This is the operator interface.

5.2 IDEAS

There is much conjecture as to the form of any operator interface, but it is hard not to be impressed with the ideas in Xerox's Star Information System (Smith, Irby, et. al.,1982). A very useful interface could be constructed if the Xerox procedures were adapted for our system. For example, manual control of a rack would be through manipulation of its schematic diagram as displayed on a graphics terminal. Solenoids could be selected by positioning a cursor over the appropriate solenoid in the schematic, and could be operated by pressing a button on the cursor positioning device (commonly called "the mouse"). Such a scenario enjoys the directness and readibility of customised "mimic" panels, but in the flexibility of a software environment. It is hard to settle for anything less.

5.3 PRACTICALITIES

The operator interface suggested above requires expensive hardware devices, and would need many programmer-months to bring on-line. There is also a definite need for similiar interfaces on the inlet line, the mass spectrometer, and even the measuring system. Such duplication is not only expensive, but is likely to be intimidating from the operator's point of view. Instead, a centralised interface is required, and a more generalised set of software tools must be written to run it. At the hardware level a compatible set of interfaces and protocols must be defined to link all components of the complete oxygen isotope analysis system. Such an undertaking requires a complete re-appraisal and re-organisation of the system, and is obviously well beyond the scope of this thesis.

In any case, the sample preparation line has been designed with an eye towards its use in other laboratories, and with other mass spectrometers. To aid in these ends, communication with the host system has been kept deliberately simple. Under such conditions, the operator interface must be part and parcel of the sample line itself, and cannot rely on the existence of another machine.

Finally, it must be pointed out that the task oriented nature of the sample line operating system allows different operator interfaces to be used in different situations. The one described in the following paragraphs is universally useful for a stand-alone system, but may be replaced if, for example, the users would rather handle the human end from another machine.

5.4 DESIGNING THE INTERFACE

5.4.1 Choosing Input and Output Devices

An operator interface relies heavily upon the devices that transfer information between the human and the machine. Those employing visual graphics probably result in the clearest and friendliest communication, but as pointed out earlier, other considerations do not make this currently practical.

A viable alternative is the antique, but still useful, teletype. The familiar layout of its standard typewriter keyboard should prove unintimidating to most operators, and the flexibility to interpret keystrokes in software allows the programmer to write natural-feeling command languages. Furthermore, every teletype is equipped with a print head that can be made to serve a dual purpose. By echoing keystrokes on the printer, a degree of feedback is provided that can only enhance the confidence of the operator in his ability to control the system. With suitable software, errors in a command sentence can be seen and corrected before submitting it to the system. The other purpose is to type messages originating from the system. These might be in immediate response to some input, or could be reports issued some time later, marking progress or errors during processing.

The necessity of a permanent printed record cannot be overemphasised. During unattended operation, error, test, and progress reports can be printed and available for later examination. When combined with the echoed keyboard input, such a record provides valuable data, not only for the detection and correction of errors, but to guide future system development.

A more modern, video type, display could be used to control the sample line but it would be necessary to provide a printer for permanent records. On the whole, a teletype is the least expensive of the available options, and is the easiest to implement. Its choice was made easier, in this case, by the availability of one in our laboratory, of the existence of a suitable interface on the microprocessor board, and of the ability to access input and output routines already resident within the UNIBUG monitor program.

In addition to the teletype, two other forms of output are implemented to attract the operators attention. Eight light emitting diodes are mounted on the front panel of the prototype rack. These can be programmed to light, or flash, in any sequence, and give an instant visual indication of the rack status. Similarly a small piezo-electric sound disk is mounted on the processor board, and has been utilised as a 500Hz audible alarm. Whenever errors requiring operator attention occur, the alarm "beeps" until the operator responds. The rate at which the alarm beeps increases with the number of outstanding errors, and gives an impression of urgency as they build up.

5.4.2 Developing a "Natural" Command Language

When faced with the task of providing unskilled operator communication through a typewriter keyboard there are two basic approaches.

The first is commonly known as a menu-driven command mode, and it has the advantage of being self-documenting. The user is presented with a list of numbered choices (the menu) and, after reading through it, can select one of them by entering its number on the keyboard. However, the method becomes cumbersome when a slow output device, such as a teletype, must be used to display the menu. This is especially true of systems requiring large, and complicated, menus.

The second approach is therefore more practical for our purposes. It requires the programmer to define a command language using a set of command words (the vocabulary), and a set of rules for combining them (a grammar). System operations can then be initiated by entering command sentences at the keyboard.

Unfortunately the scheme is not self-documenting, and the programmer must provide clear documentation manuals that will enable a naive user to quickly learn the language. This will be a less formidable task for operator and programmer alike if the command language appears natural or if, for most of us, it is similiar to written English.

A good deal of work has been done on front-end natural language processors. The most sophisticated programs operate in several stages beginning with the input of a sentence from the operator. After some preprocessing the sentence is broken up into words and then submitted to a parser. The parser uses grammatical rules to break the sentence into its structural units (e.g. subject, verb, object), which are then used as templates in the next stage to construct subroutines related to the meaning of the sentence. Finally, the subroutines are executed to perform the intended actions. Such programming primarily takes place within the arena of artificial intelligence, but has increasing applicability for data bases, electronic offices, and teaching systems. The most striking example must be Terry Winograd's SHRDLU program (Winograd,1972). It maintains dialogue with a simulated robot, moving objects around in the simulated environment of its "blocks world".

Such programs are far too complex, and lengthy, to be realised on a small microprocessor, but they did grow from simpler things. In particular an earlier program named ELIZA (Weizenbaum, 1965), gives an amazing impression of intelligence for something that is totally idiotic. ELIZA relies on a pattern matcher to recognise certain cleverly choosen keywords. The keywords trigger a simple response, vaguely related to the context the keyword was used in, but mainly based on the rather cynical notion that conversations can be kept up by just appearing interested (by "uhuh" at appropriate moments, for example).

The use of a pattern matcher is a valuable technigue, especially in an environment where nearly all sentences are commands to the system. Here it is most usual for a simple word in the system to carry all of the meaning, and its recognition by the matcher is often a sufficient condition to execute it. For example, if the keywords "START", "BEGIN", and "PROCESS", all initiate sample processing then inputing the following sentences will cause the correct action in each case.

```
START
PROCESS THE SAMPLES
PLEASE BEGIN PROCESSING THE SAMPLES
```

Of course it is easy to fool the system with negatives such as "DO NOT START", but allowing for such wilful errors is of little real value when it seems reasonable to expect co-operation from the operator.

By giving the pattern matcher a table of keywords we can increase the vocabulary of the command language. The matcher trys each keyword in turn, stopping at the first that matches. Some care must be taken with the table to avoid ambiguities in the language. For example, suppose that "TEST" is added to the above commands, and that it initiates some system test routines. Then the sentence "START THE SYSTEM TEST" can take two meanings because of the presence of "START" (begin processing samples, or begin testing). This is easily resolved by specifying the keyword table in the following order.

```
PROCESS
TEST
BEGIN
START
```

Matching of "TEST" will then succeed before "START" or "BEGIN" and the sentence takes on its intended meaning.

It is understandably tedious entering a complete sentence when a single word or even a letter will suffice, and there is a general tendency for the user to eventually favour abbreviated forms of input. Fortunately it is easy to modify the matching routine to recognise abbreviations, and even extensions of keywords. Consider a table that contains the following keywords in the order shown.

```
ABORT
TEST
START
STOP
```

Then the words "T", "TE", "TES", and even "TESTING" will all match with "TEST"; whereas "S", "ST", and "STA" likewise match to "START". However "STO" is the minimum abbreviation for "STOP" because the routine will attempt a match of "START" first.

A case above that is potentially troublesome is the confusion between "A" used as a determiner, and as an abbreviation for "ABORT". The sentence "RUN A SYSTEM TEST" could attempt an abort procedure. Placing "ABORT" after "TEST" in the keyword table solves the problem in this particular instance.

In summary a rather simple pattern matching scheme proves capable of quite "clever" processing. Apart from the advantage of a non-structured command syntax, the programmer can, through extensive use of "look up" tables, make drastic changes and additions to processor commands without altering the decoding setup.

5.4.3 Software Tools

Two important subroutines are necessary. The first is code that can obtain the sentence from the teletype and store it in an accessible location. This is really no problem mainly, because of the list structured approach used in designing the operating system. On receipt of the first character from the teletype, a buffer is removed from the free list. Additional characters are stored in this buffer until the end of the sentence is flagged by a RETURN character (or when the buffer is filled). The buffer is returned to the free list after command processing is completed or, optionally, when a CANCEL character is input (CNTRL X). CANCEL allows the operator to abort a command if he makes a mistake typing it in, and is also handy for documenting the printed record. Because the standard size of blocks on the free list is thirty-two bytes, this is the maximum number of characters in a sentence (including spaces). It has proven more than adequate for present needs. Additionally it is possible to alter the contents of the buffer before the sentence is released for furthur processing but such a line editor has not been implemented. A listing of the subroutine may be found in Appendix II under the name STRGET.

The second subroutine is the pattern matcher discussed earlier. It requires two input parameters; the location of a keyword table, and the location of the sentence to be searched for the keywords. A word has been defined as a string of characters preceded by a space, and ended with any one of a set of valid terminators (space, comma, left bracket, or return). Some examples are underlined below.

<u>TEST</u> <u>RACK</u>(1) <u>VAC</u>, <u>#?*</u>

Two exit points are provided for the subroutine. The first is used when none of the keywords in the table can be found in the sentence. Usual responses to such an exit are to search another table, or to print an error message. The other eventuality is a successful match. When this is the case the other exit point is used, with the <u>index</u> of the matching table entry being returned in one of the workspace registers. The index may be used to access data in tables with a one-to-one correspondence to the keyword table. Common examples are tables of subroutine addresses, data, data types, or argument types. For added flexibility the routine also returns the location of the matching word in the sentence. This allows other tables to search for keywords in the rest of the sentence, and permits the programmer to build ordering into the command syntax. Such a feature has proven useful during the manual mode when it is often necessary to perform several operations at once. For more detail the subroutine MATCH should be examined in the program listing of Appendix II.

5.5 THE MANUAL AND COMMAND TASKS

As can be seen from the above, the operator interface falls naturally into two separate and distinct categories. It can therefore be represented by two tasks running under the control of the sample line operating system. These are the command task and the manual task.

The command task uses operator input from the teletype to alter the structure of the operating system queues. Manipulative examples include the power to abort, and hence delete, tasks from the system; and the ability to move tasks from the error queue, back into the execution queue. The most powerful commands result in the formation of a new task, to which the command task relinquishes control. Examples of this include tasks to perform sample preparation, equilibration, and analysis; pressure testing; and manual rack control.

This last example results in the introduction of the manual task to the execution queue. It allows the operator to manipulate all solenoids and control lines necessary to the operation of an individual rack. These include shared facilities from the mass spectrometer and main lines, so some protection must be provided to prevent the operator inadvertently affecting their use by other racks. The manual task therefore RESERVes both the main and mass spectrometer lines for its exclusive use before requesting operator input. It will not release these resources until issued the "STOP" command, and will prevent other tasks from accessing them during the interim. This feature may prove inconvenient on a multi-rack system, but presents no problems on the current single rack configuration. In any case, as repeatedly pointed out, the task oriented nature of the sample line operating system permits the the operator interface tasks to be modified and replaced at will.

Both manual and command tasks operate in a similiar manner. They continuously interrogate the teletype keyboard until a sentence has accumulated. The sentence is searched for keywords from tables peculiar to each task, with further operation depending on the keywords found. For those readers requiring a more detailed understanding, it is best to consult the program listing in Appendix II, under the programs labeled TSKCMD & TSKMAN.

Despite the similarity of their operation there is a fundamental difference between the two tasks. The command task is introduced to the system during the initialisation phase. It never finishes, but always remains in the <u>background</u> as the lowest priority task in the system. The manual task, in common with all tasks introduced from the command mode, is <u>transient.</u> It is created, used, then destroyed, as needed.

All tasks, except the manual and command tasks, execute very rapidly before removing themselves from the execution queue to wait on the timer or resource queues. In fact their total utilisation of the central processor is so minimal that the operator never notices their presence and always appears to be talking to either the manual or command task. To avoid operator confusion the current task is identified by an <u>input prefix.</u> This is a single character output on the teletype printer whenever the current task expects input. If the character is a

"#" then input is being requested by the command task; and if it is a ":" the manual task is in control.

The commands available from the user interface are currently being compiled, and will be available from this laboratory in the form of an operating manual for the mass spectrometer.

CHAPTER VI. AUTOMATED SAMPLE PROCESSING METHODS

6.1 THE RACK PROCESSING TASK

Only one more component need be described to complete a working sample preparation system. This is the task that runs the rack, and processes the samples. The rack task was probably the easiest part of the system to implement, but only because of the extensive groundwork that prepared the way for its introduction. In particular, the bookkeeping overheads, and the convoluted flow of logic that is often a characteristic of complicated control programs, are missing because of the supervisory nature of the sample line operating system. The capacity of the operating system to oversee task requests for resources, the ability of its routines to inactivate tasks for periods ranging from seconds to hours, and the provision of an orderly method that traps tasks faced with real-world obstructions, are just some of the areas where extensive simplification of the rack task has resulted.

There is no point in presenting the detail of the rack processing algorithm, but a brief description of each of its stages is given in the following sections. These paragraphs will concentrate upon the more unusual aspects of the task, and the reader is refered to the program listing in Appendix II, under the label TSKRCK, for the exact sequences.

6.1.1 Sample Preparation

Samples are prepared by pumping away the air above the water in the sample tubes, and replacing it with carbon dioxide. It is important to note that pumping must continue until gases dissolved within the water have been removed. This is evidenced by vigorous boiling when the tube is first evacuated, and of a continual decrease in activity over the next ten minutes.

There was originally some concern that problems could arise from droplets of water splashed onto the tube walls during this phase, but the rapidly spinning stirrer bars tend to break up the larger bubbles, and minimise splash. It has also been observed that large droplets adhering the walls do evaporate and, presumedly recondense on the sample surface, as would be expected from their higher vapour pressure. In any case, the test tube will be approximately uniform in temperature, and errors resulting from equilibration in the droplets should be negligible.

The final point of note is the pressure of carbon dioxide in the sample tubes. It was choosen to be slightly above atmospheric to minimise possible fractionation and contamination resulting from small leaks to the outside air. For the same reason carbon dioxide is left in the adjacent sample reservoirs until the equilibration reaction is complete. In this case the presence of dry CO_2 should help strip water vapour from the reservoir walls before the equlibrated gas is stored.

6.1.2 Equilibration

After completing the sample preparation stage, the rack task enters an idle state, and remains there until the

Sec 6.1.2

equilibration reaction has completed. During equilibration it is important that the magnetic stirrers operate continuously if efficient mixing of the gas and the water is to occur. The reaction proceeds exponentially towards completion, with a time constant dependent upon several factors, including the ratio between the number of molecules of carbon dioxide and of water in the sample tube. A typical time constant is thirty minutes, but the total equilibration time depends upon the initial DEL difference between the water and the CO_2, and the maximum permissible error in the DEL of the final equilibrated gas. Currently we only analyse one rack per day, and the equilibration time of fourteen hours has been chosen to fit into this schedule.

The time constant mentioned above also applies to small changes in temperature, and it is necessary to keep the samples at equal temperatures for about the last two hours of equilibration. A small fan circulates air rapidly around the rim of the rack in an attempt to meeet this condition as simply as possible.

6.1.3 Sample Storage

When equilibration is complete the reservoirs are evacuated and closed off. The connecting valve between them and the sample tubes is opened until the pressure between the two containers equalises. Closing the connecting valve isolates the gas in the reservoir from any further exchange with the water sample. It can remain there, its isotopic value stable, until required for analysis.

6.1.4 Sample Analysis

The analysis of samples is the most complicated section of the processing algorithm. A rack with samples ready to analyse reserves the mass spectrometer line (thus preventing other racks from releasing samples), starts up the Peltier-cooled vapour trap, and signals the mass spectrometer interface that samples are ready. Having completed this initialisation sequence, the rack task waits for a sample request from the mass spectrometer. When it receives one, the common area of the rack and the mass spectrometer line are evacuated, then sealed. The reservoir solenoid on the first rack position is opened, and equilibrated sample gas flows down the mass spectrometer line manifold, through the Peltier vapour trap where any water vapour is removed, and onto the line connecting the trap to the mass spectrometer's inlet line. At the same time a signal is sent to the mass spectrometer indicating that the sample is ready.

The rack task then waits until it receives a request from the mass spectrometer for another sample. The reservoir that is currently open is sealed off, and the procedure repeats with the next sample in the rack. The sequence continues until every sample on the rack has been released. The rack task finishes by releasing the mass spectrometer line, so that other rack tasks can submit their samples for analysis.

Note that sample gas is conserved wherever possible, so that gas remains in the reservoir after the first analysis. On the present system there is enough left for two more analyses, so, if time permits repeats can be performed to increase the accuracy of the result.

6.2 THE PUMPING ALGORITHM

The most frequent operation during rack processing is the evacuation of gases from various sections of the system. This is also the area most likely to cause serious problems, either from leaks, pump failure, or solenoid malfunction. A special subroutine was written to take account of the peculiarities of the vacuum pump. It has been designed to check the pressure in such a manner as to detect small leaks, and to use a special error exit if certain criteria are not met.

Before calling the pump routine, the user is responsible for opening all valves leading to the area to be evacuated. The routine must be provided with the location of a special table of pumping parameters, and an address to which the program will branch in case of error.

The first table entry is the type of pressure gauge on which all pressure measurements will be performed (piezo-electric or thermocouple). The second is a time (in seconds) for which the vacuum pump will be connected to the main manifold. At the end of this time, the pressure will be measured, and compared with the third table entry. Any gross leaks will fail the first test, and the subroutine will be able to report a pump-down failure. However, small leaks will not be noticed due to the small orifice in the valves through which rack sections are pumped. The leaks are detected by waiting after pump down for the period specified in the fourth table entry. This is chosen to allow pressure throughout the evacuated section to equalise, and increase in the case of a leak. At the end of this time, pressure is again measured and compared with the fifth and last table entry. Failure is marked by a pressure greater than that specified.

The pumping algorithm is specified in Appendix II, and can be found under the name PUMP in the rack task.

6.3 RACK ERRORS

Most errors that occur are detected by the pump routine described above. When this is the case the rack task issues an error message, then places itself on the operating system error queue. Retrys, from the error queue, usually take place at the beginning of the stage in which the error occurred (preparation, equilibration, storage, or analysis). Retrys are either initiated by the operator, or if operator attention is not forthcoming, are automatic after a specified time has elapsed. Sometimes conditions are such that another attempt will succeed without operator intervention. An example of this could be a small drop of water left in one of the reservoirs. Several attempts at pumping may be required before the droplet evaporates.

However some errors are extremely unlikely to disappear. and must be rectified before rack processing can proceed. An example is the lack of carbon dioxide pressure when attempting to backfill the water sample tubes. In this case the rack task will remain on the error queue indefinitely, and the operator must respond before the task can proceed.

The doors surrounding the samples in the rack are necessary during the last stages of equilibration, so the rack task is prevented from proceeding until they are all in place. Though this ensures that the rack is set up correctly, there is no easy way to prevent their removal later on. In fact, it is not necessarily advantageous to do so. Sometimes the operator may wish to make repairs "on the run", and these can often be done without adversely affecting sample equilibration. There could be conditions where small errors are preferable to the loss of the whole rack, a day of time, or irreplaceable samples. Door openings, and closures, are therefore not considered as errors; but they are reported by the operating system. When such an event generates a CMOS bus interrupt, the operating system tests the rack refresh table to see if the interrupting rack is currently in use. If it is, status words showing the state of the rack doors are printed on the teletype; but if it is not then the event is ignored (see TSKLOG in APPENDIX II).

CHAPTER VII. PRELIMINARY TESTS AND RESULTS

7.1 PRECISION OF SAMPLE PREPARATION

7.1.1 Methods and Results

The precision of sample preparation is conveniently estimated by preparing equilibrated CO_2 samples from a rack loaded with identical water samples, and then measuring the DEL of the CO_2 samples on the mass spectrometer. The standard deviation of the DELs gives an estimate of the combined error from sample preparation and measurement.

The following test was run to evaluate errors arising from the measurement stage. Using the manual control facilities, the sixteen sample reservoirs were evacuated, then filled with tank carbon dioxide. The gas in the reservoirs was analysed in the normal manner by evoking the ANALYSE command from the sample line teletype. The results of one such test (run #025) using this method are shown in figure 7.1.

To obtain the overall precision of analyses performed on the system, water samples were prepared and analysed in the usual manner. The sixteen sample tubes were loaded with samples pipetted from a well mixed beaker of tapwater. The results from two such runs (#020a and #022) are shown in figure 7.2.

All the results have been plotted as the residuals left after subtracting out the mean DEL value obtained from that run. A linear regression has been used to relate the measured DEL to rack position, and the line of best fit is plotted amongst the

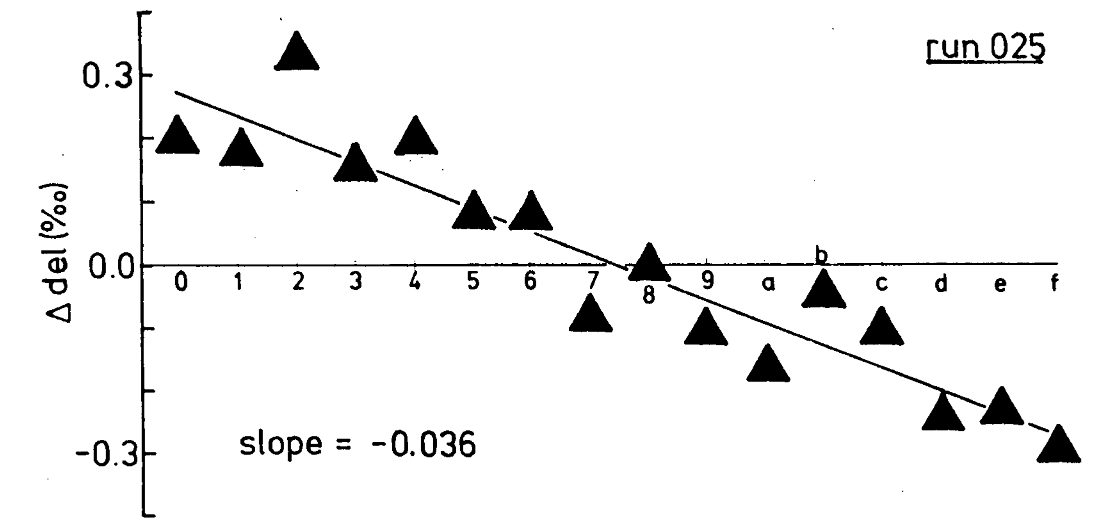


Figure 7.1 Results from Identical Gas Samples

data points.

7.1.2 Discussion of Results

Turning our attention to the results of the measurement evaluation experiment (figure 7.1) a strong linear trend is immediately apparent. The residual DEL value is therefore strongly correlated to either rack position, elapsed time, or the number of samples analysed. The first possibility is unlikely given the fact that no equilibration was involved in the test. It would be even more remarkable if time dependent effects, such as the presence of small leaks in every reservoir, manifested themselves in such a fashion. The last possibility is the most likely, and the prime suspect is a slow constant fractionation of the standard gas as it is depleted from its reservoir on the inlet line. If such is the case then the observed trend can only be explained if lighter molecules are being removed in preference to the heavier ones. Such behaviour

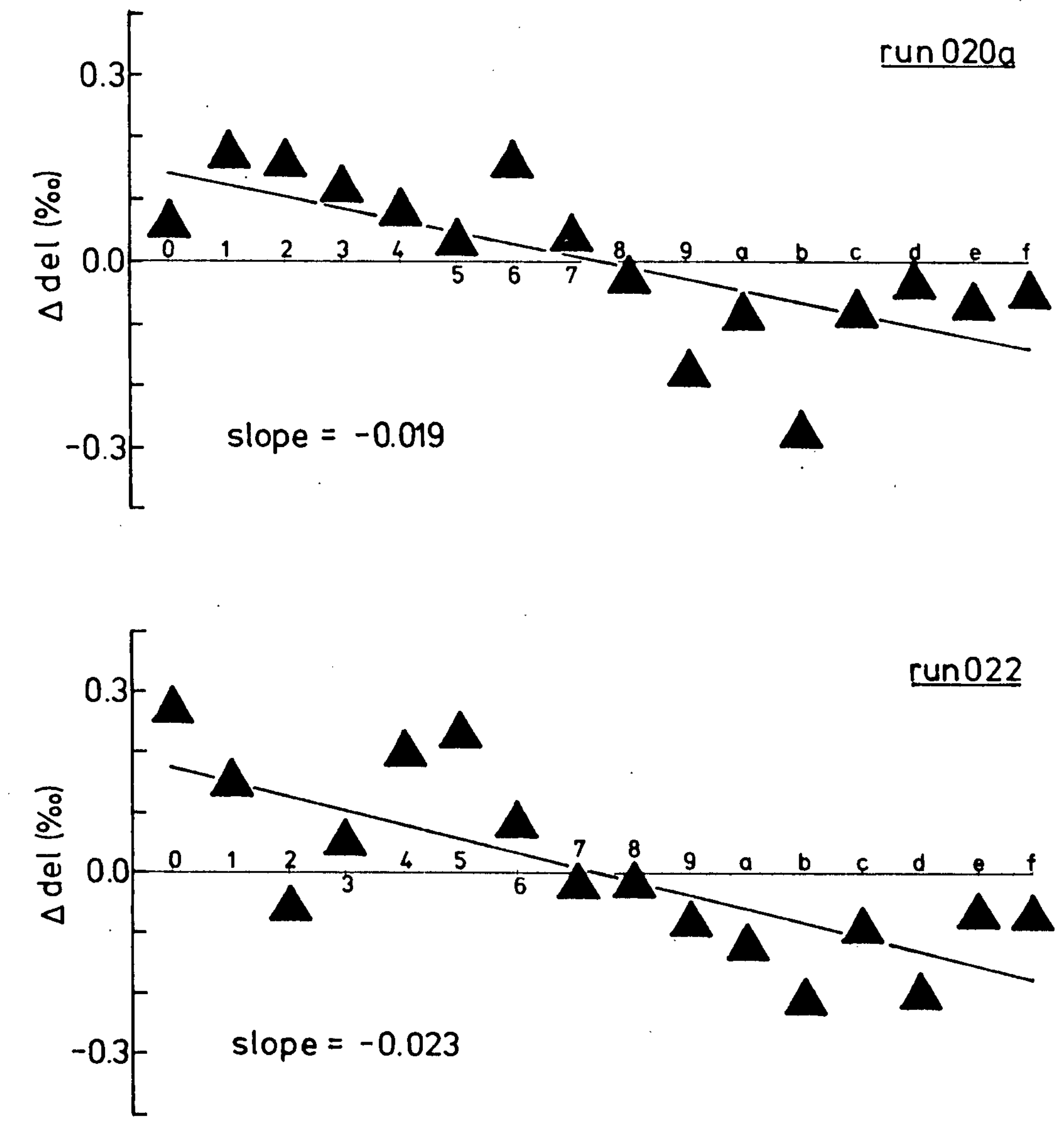


Figure 7.2 Results from Identical Water Samples

is possible if the CO_2 flows through a molecular leak at the inlet line to the mass spectrometer. This leak is suspected as a source of instability in the mass spectrometer analyses for other reasons, and it will shortly be replaced with a more stable glass one. The results in figure 7.2 show a similiar trend, but the correlation is not as pronounced. These earlier runs suggest that the problem is unpredictable, and could also be growing more serious.

In view of the above, a separation of the repeatibility of sample preparation from that of the rest of the system is somewhat difficult. Some idea can be obtained by removing the linear trend from each set of results, then calculating a standard deviation from each set of residuals. When this is done the standard deviation is found to be $0.06^{\circ}/_{oo}$, $0.09^{\circ}/_{oo}$, and $0.10^{\circ}/_{oo}$ for run #025, run #020a, and run #022 respectively. These are probably reasonable estimates of the precision that will be attainable when the drift problems are remedied. I, therefore, tentatively place the overall precision of the measurement at around $0.10^{\circ}/_{oo}$, with about $0.06^{\circ}/_{oo}$ of that being due to errors introduced during analysis. The remaining $0.04^{\circ}/_{oo}$ is probably attributable to lack of constancy in conditions throughout the sample rack and is, when compared with other systems, a very encouraging figure.

7.2 ERRORS FROM CROSS CONTAMINATION

7.2.1 Methods and Results

A second kind of error results when a sample, released to the mass spectrometer, has its DEL value altered by residual gas and adsorbed water left in common connecting lines from the release of the previous sample. To test these effects, the rack was loaded with a blocks of four tubes containing identical tapwater samples, alternating with blocks of four empty tubes.

After equilibration, the DEL of the CO_2 in the tubes containing

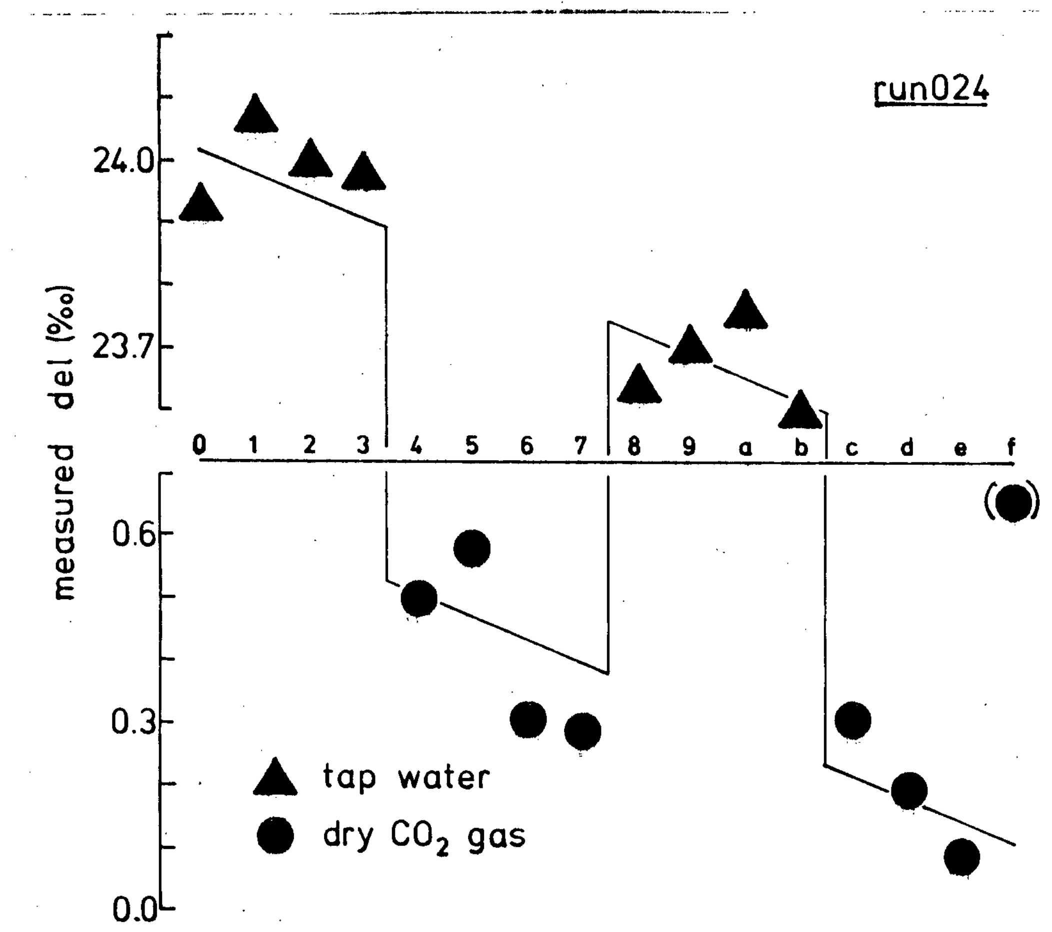


Figure 7.3 Results of the Cross Contamination Test

water changes by about 24°/₀₀, and the gas becomes thoroughly saturated with water vapour. During analysis the dry CO_2 sample immediately following the block of wet ones will be subjected to the maximum effects of contamination from residual gas and adsorbed water, with the effect tailing off as each new sample in the dry block is analysed. Similarly, the wet samples following the dry block should also show the decreasing effects of contamination, but only those arising from their mixture with residual gases.

Analyses obtained from a rack configured in such a manner are shown in figure 7.3.

7.2.2 Discussion of Results

Examining the results shows the expected pattern of high and low DEL values. However, the low DEL values appear quite variable, and the trends expected from contamination could be lost in the "noise". Still, it is possible to attribute an absolute maximum error of $0.30^{\circ}/_{oo}$ to the effect, or as it is usually expressed, about one percent of the difference in DEL between adjacent samples.

There is only one transition from dry to wet samples, but it is better behaved. Here the magnitude of the contamination error is below $0.10^{\circ}/_{oo}$ or around 0.3 percent of the difference. These figures are acceptable for most routine analyses.

APPENDIX I. CIRCIUT DESCRIPTION AND OPERATION

A1.1 THE TMS9980 MICROPROCESSOR

A1.1.1 Microprocessor Operation

A detailed and extensive discussion of the operation of the TMS9980 microprocessor is beyond the scope of this thesis. In any case operation is well documented and explained in the appropriate reference manual (Texas Instruments, 1979). Still, the basic way in which memory is read and written should be explained if furthur sections of this chapter are to be comprehensible.

There are five control lines that synchronise read and write operations. These are briefly explained below and by reference to figure A1.1.

03-: This is derived from the clock (phase three) and is used as a timing reference.

DBIN: When high this indicates that the microprocessor is ready to receive data on the data bus.

MEMEN-: (Memory enable). When low the address bus has a valid memory address.

WE-: (Write enable). When low the data bus holds valid ouput data and can be written to memory.

READY: The ready signal allows extended memory cyles. When high it indicates to the microprocessor that data can be read or written during the next clock cycle. The TM9980 enters a

<u>wait state</u> until this line is high.

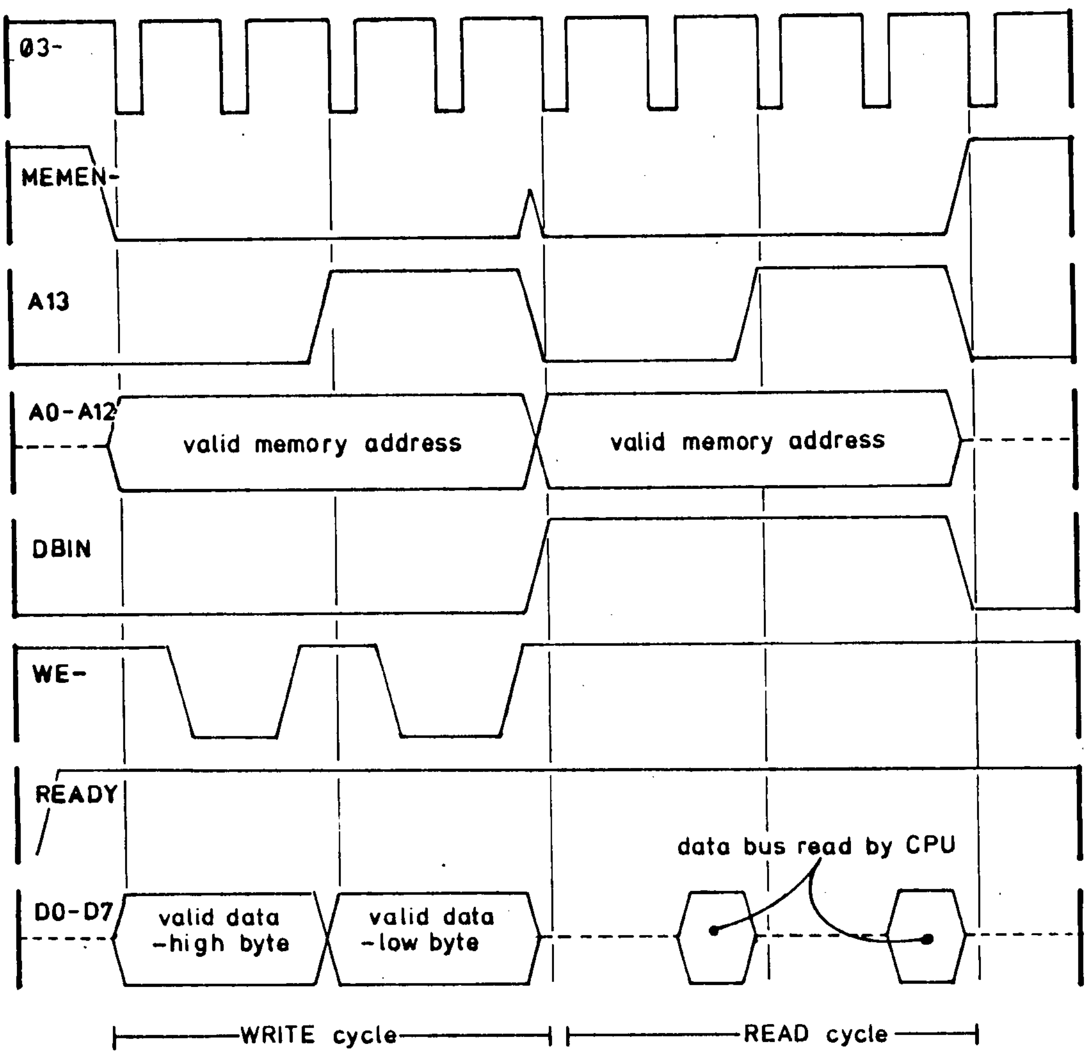


Figure A1.1 Typical WRITE and READ cycle

When the microprocessor wishes to write, it places the low byte of data on the data bus, and the address on the address bus. The valid address is flagged by lowering MEMEN- and valid data by lowering WE-. The high byte of data is placed on the data bus after WE- returns high and the address is incremented by raising A13. WE- returns low when the second byte of output data becomes valid.

A read cycle performs in an identical manner except a request for input data is flagged by raising DBIN. The interface circuitry is then responsible for placing the data on the data bus before the second clock pulse arrives.

If an external device is not ready to receive or transmit data to the data bus it should lower READY before the next clock pulse 03-, and raise READY when the data is valid. The long propagation delays through the CMOS bus, have made this "handshaking" necessary when reading from registers on the rack or master control boards.

A1.2 OPERATION OF THE ANALOGUE-TO-DIGITAL CONVERTER

Operation of the ADC may be understood by referring to figure A1.2. An ADC conversion is initiated when the CPU writes an analogue channel number to the ADC address. This causes both ADCEN- (ADC enable) and WRSTB- (write strobe = EXPWE- . EXP03-) to both become active (low). Consequently the output of G1 will go high, latching the four least significant bits of the data bus into the multiplexer to select the analogue channel number. At the same time F1 will clock its D input and raise the R/H- (run/hold) input on the ICL7104. The conversion then begins and after a short time STTS (status) goes high to flag a conversion in progress. In the process it lowers the output of F2, but because STTS is high the state of USERINT4- remains inactive.

When the conversion is complete STTS goes low, activating the interrupt, and resetting F1. This places the ADC in a hold state until the interrupt is serviced.

At the first opportunity the processor services the

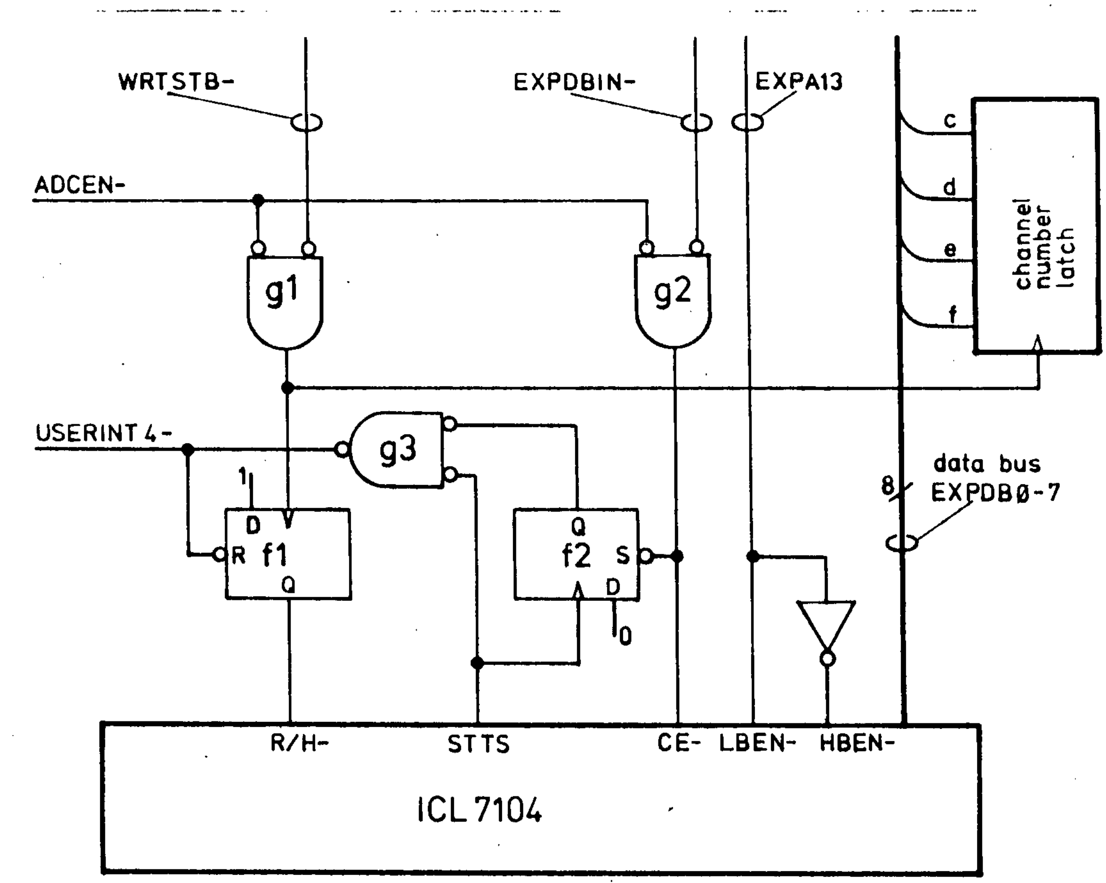


Figure A1.2 Simplified Schematic of ADC

interrupt by reading from the ADC address. This causes both ADCEN- and EXPDBIN- to become active, setting F2 and resetting the interrupt line through G3. As well the ICL7104 output buffers are enabled because CE- and LBEN- (low byte enable) will both be low. The lowest byte of the conversion value is thus placed on the data bus and the processor reads it. The state of EXPA13 is changed in the second part of the read cycle lowering HBEN- (high byte enable) and so allowing the rest of the conversion value to be read. When the cycle is completed the ADC remains in a hold state, the output buffers are disabled, and the interrupt line is inactive (HIGH). The ADC is then ready for another conversion request.

For an in-depth look at the actual conversion procedure it is best to refer to the device data sheets (Intersil,1981) but it could be described as an auto-zero, dual slope technique.

Another point to note is that the ADC address is completely selectable using twelve dual in line rocker switches. It may be placed anywhere in the range 2000 to 2FFE.

A1.3 THE CMOS BUS INTERFACE

A1.3.1 CMOS Bus Lines

As with any microprocessor bus, the CMOS bus can be grouped into three different functions. Control lines to and from the microprocessor supervise its actions, address lines select the desired device, and data lines transmit the two way flow of data. The lines are named and described in the following paragraphs.

CRACKEN- : (Rack enable) This line becomes active (low) whenever a rack address is output on the address bus by the micropressor. Valid rack addressess reside in a block of contiguous memory from 2X00 to 2XFF. "X" is a hexadecimal digit selectable from four switches located on the expansion board. In this version X is set to zero.

CMSTREN- : (Master enable) Will become active (low) whenever an address in the range 2Y00 to 2YFF is accessed. Y is selectable using another board mounted switch but should never be the same as X.

CWE-: (write enable) becomes active (low) whenever an valid output data from the microprocessor is present on the data bus (CD0- to CD7-).

CBUSIN-: (Bus input) Becomes active (low) to flag a request from the microprocessor for input from the CMOS bus.

CREADY: This control line is used to inform the processor when a device on the CMOS bus has data ready to be read (in response to a CBUSIN-). Because two read cycles are used, a simple encoding scheme generates the correct timing on the microprocessor (see section A1.3.3).

CINT-: (Bus Interrupt Request) A device on the CMOS bus lowers this line whenever it wishes to interrupt the microprocessor. Because interrupts are mainly generated from mechanical switches the line is debounced before it enters USER INT3- on the User I/O port (see section A1.3.4).

CA6 to CA13: (The Address Bus). The eight least significant address lines are divided into two fields. The most significant field (CA6 to CA9) selects the rack while the other (CA10 to CA12) addresses one of the registers within the rack. The CA13 line is used to flag the high or low byte of the addressed register (see figure 3.3).

CD0- to CD7-: (The Data Bus). These eight lines form an inverted image of the microprocessor data bus. They are used to transmit data values to and from the rack registers.

Additionally, power (+12 volt) is sent to all CMOS devices through the bus. The high noise immunity and the ultra low power consumption of the supplied electronics make this feasible.

A1.3.2 Level Translation

Logic levels on the CMOS bus are much higher than their TTL equivalents so level translators must be used to interface the two. There are three different modes of translation.

The first group of bus lines involves those signals originating at the microprocessor and requires translators to shift the level upwards. The second group is similar except the direction and shift are reversed. Both these groups of lines are termed unidirectional and a suitable translator was found to be manufactured by RCA (Type CD40109). The quad low-to-high voltage translator is intended primarily for TTL to CMOS conversions but it may be used in the opposite sense by reversing the appropriate power supply leads. The penalty paid for this convenience is an extremely long propagation delay in the reverse mode (up to 1.6μS !!). However for the CMOS bus there is no real disadvantage given that only two lines require high to low translation (CINT- and CREADY) and that extra delays are unimportant to their operation.

The third group of bus lines requiring translation are bidirectional. Signals on the data bus (CD0- to CD7-) can originate at either the processor or a bus device, and the translator is required to function in both directions. Another RCA device (CD40115) conveniently fulfilled all the above requirements. A single chip (24 pin DIP) can translate eight bidirectional lines between CMOS and TTL levels. Propagation delays are very low in both directions (below 30nS) even when outputs are capacitively loaded by long bus lines.

In fact this device was so much faster than the unidirectional translators that it could switch direction and begin transmitting data before CMOS buffers, just read, could be disabled. Although no short term problems were experienced when these two devices attempted to drive in opposite directions, logic was incorporated onto the data translator to throw it into a high impedance state whenever the CMOS bus was not being used.

A1.3.3 Ready Handshaking

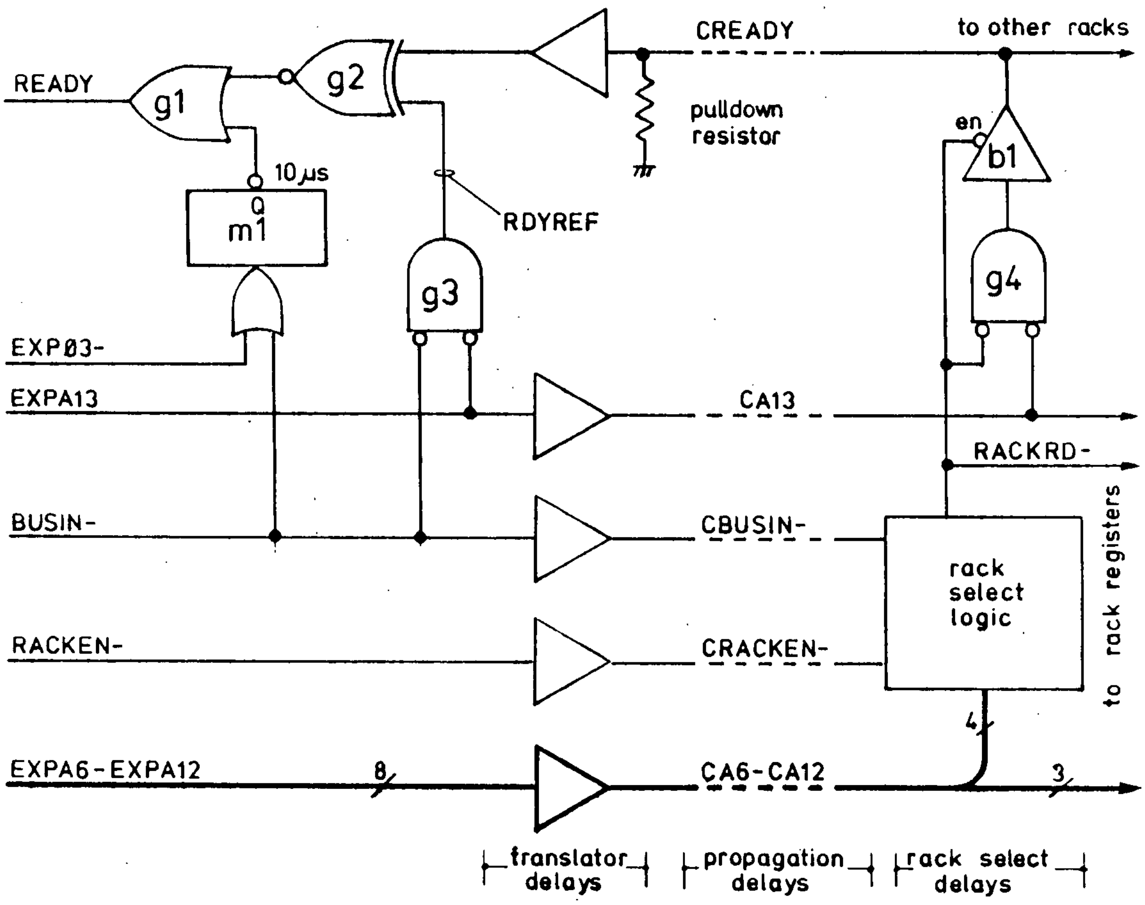


Figure A1.3 READY Handshaking Circuit

Long propagation delays through the CMOS bus, bus devices, and level translators make it impossible to read the devices without establishing a handshaking protocol. A simple encoding scheme has been used to generate signals appropriate to the

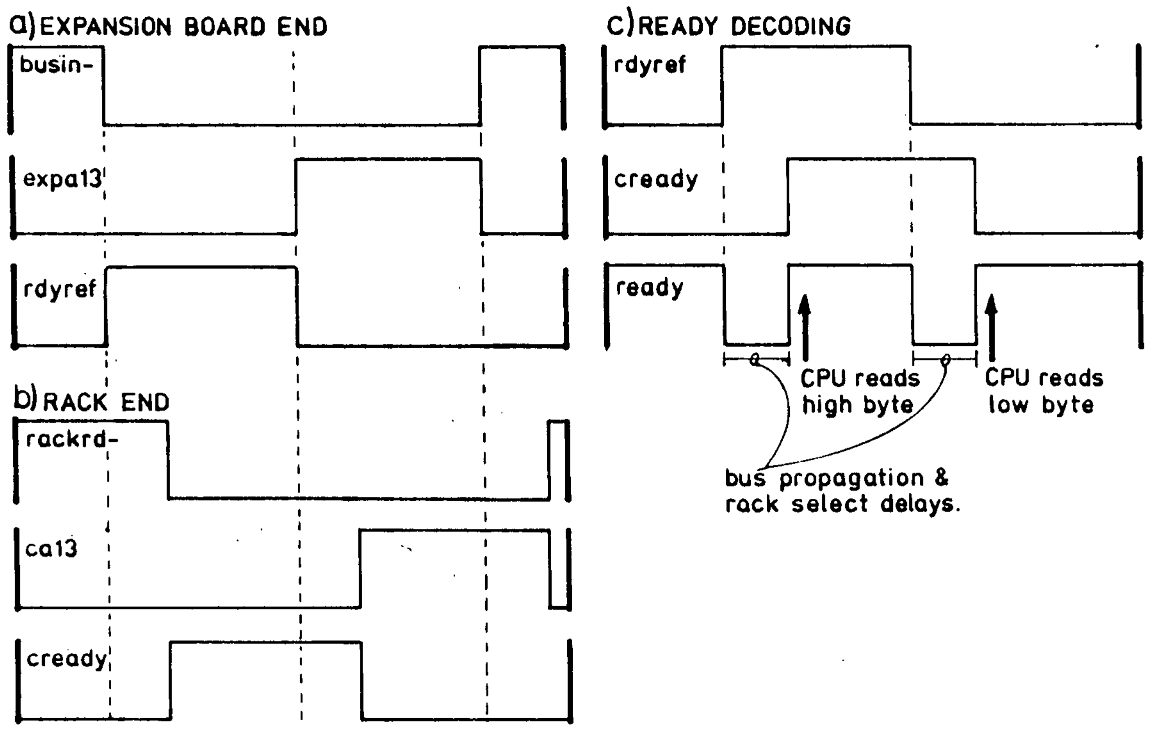


Figure A1.4 Handshaking Signals

operation of this multiplexed bus (figures A1.3 & A1.4).

At the expansion board end the BUSIN- line goes low whenever data is requested from a CMOS bus device. A reference signal, RDYREF, is generated through gate G3 that goes high when the first byte is requested. After the first byte is read, and the processor wants the second, the RDYREF line returns low (figure A1.4a).

Similiar circuitry at the bus device (G4, figure A1.3) will generate an identical signal on the CREADY bus line. When this signal arrives back at the CMOS bus interface it will be delayed from RDYREF by the sum of level translator delays, bus propagation delays, and device select delays (figure A1.4b).

Thus all CMOS bus signals are in transition when CREADY and RDYREF differ, data lines will <u>not</u> be valid, and the microprocessor must be suspended until they are. This is done by driving the microprocessor READY line from an exclusive NOR gate, G2, that directly compares the two signals. Figure A1.4c shows the relationship graphically.

The monostable, M1, provides a "time-out" facility if a non-present bus device is addressed. In such a case the READY line will go low, the microprocessor will go into coma, and the lack of bus acknowledgement will hang-up the system. Normally the system clock (EXP03-) continuously retriggers the monostable keeping the ouput low. When BUSIN- goes low the triggering is blocked and the monostable must eventually change state, sending READY high to wake the processor up. In practice a delay in the the monostable of ten microseconds allows adequate time for bus response without delaying processor operation unnecessarily.

A1.3.4 <u>Interrupt Line Debounce Circuit</u>

It is necessary to prevent switch contact bounce from influencing the microprocessor during the service of an interrupt (see section 3.10.1). The most common hardware approach is to use a two pole switch that sets or resets an RS flip-flop. Whilst a solid, reliable, and fast solution, the component and connector count is significant, especially when sample sense switches are contemplated. Where speed of response is unimportant a more economical solution is offered using software generated delays to defer interrupt service until the switch has had.time to settle (around 20 ms). However the delay "ties-up" the processor during the waiting period, and prevents other interrupts from functioning. Software schemes that do not

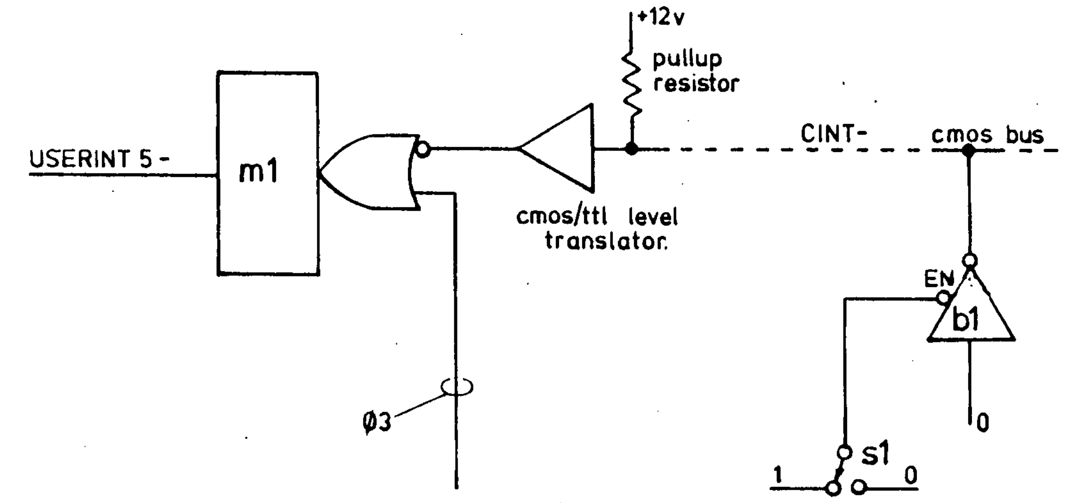


Figure A1.5 Bus Interrupt Line Debouncing

do this usually require a significant book-keeping overhead, and can become quite messy.

Instead a simple circuit has been used that incorporates the most useful features of both hardware/software debouncers. A simplified circuit is used to demonstate the debouncing circuitry in figure A1.5. When the switch S1 is closed it enables the tristate buffer B1, and lowers CINT-. This blocks the re-triggering signal to the monostable (M1). After a delay, USERINT5- will go low, generating an interrupt at the microprocessor via the system I/O port. If, at any time, CINT- goes high again the monostable instantly retriggers, resetting the delay. By choosing the monostable delay to be significantly greater than the longest time between bounces USERINT5- will remain high until after the swith has settled.

After observing the operation of the switches used on the rack a suitable time was found to be around 10 ms. This may be eternity to the microprocessor but it appears instantaneous to a human.

A1.4 THE RACK INTERFACE

A1.4.1 The Rack Select Logic and Control Lines

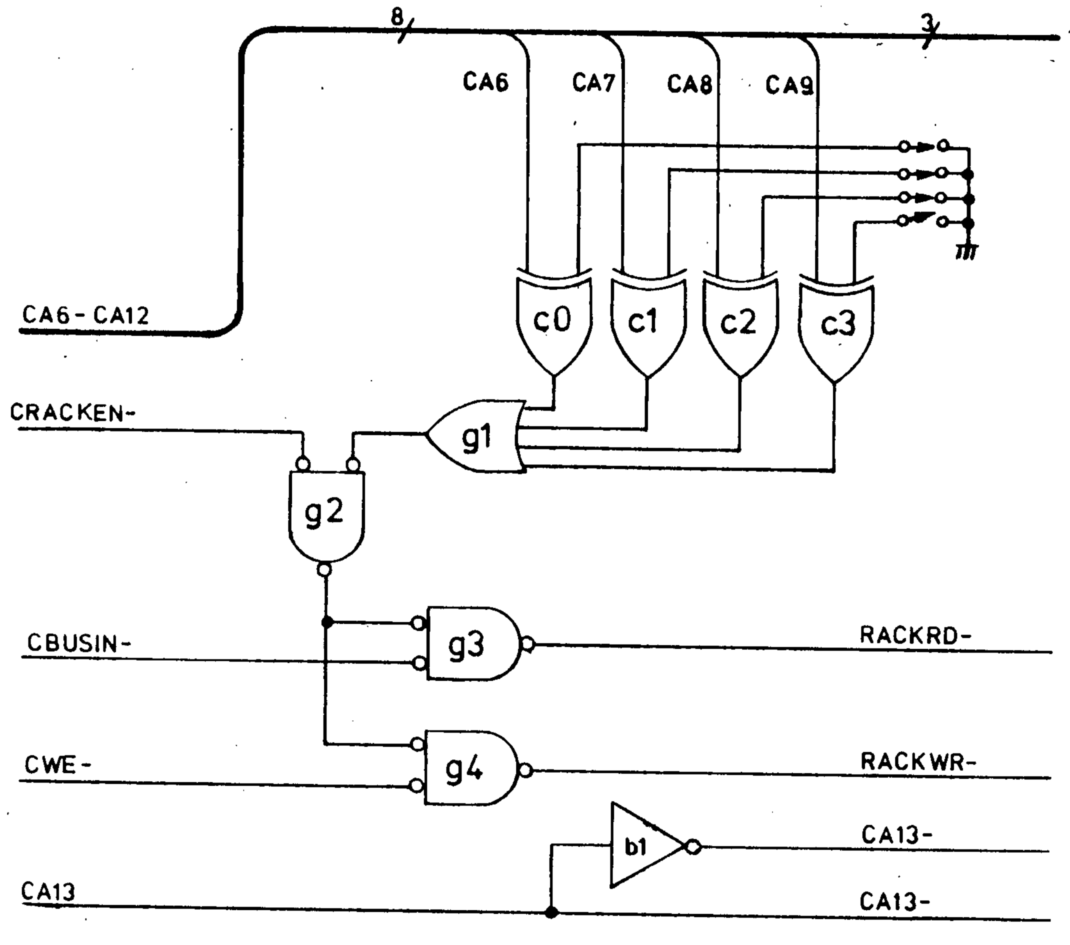


Figure A1.6 Rack Select Logic

Although rack registers are read and written directly from the CMOS bus, redundant decoding circuitry can be eliminated by centralising the rack select logic on one board, and by sending decoded control signals where needed.

A simple, straightforward scheme is used (see figure A1.6). The exclusive OR gates, C0 to C3, compare the configuration of four switches to the rack number field on the CMOS address bus (bits 6 to 9). When the two match, the output of G1 will go low. If RACKEN- is also low the rack will be selected by the output of G2. By gating the output of G2 with CRACKIN- (G3) and CWE- (G4), rack read (RACKRD-) and rack write (RACKWR-) signals are produced, and can control the operation of all rack registers. B1 inverts the byte select line (CA13) to avoid the necessity of repeating this at each register.

A1.4.2 <u>Rack Status Register and Interrupt Generation</u>

A simplified version of the rack status logic is shown in figure A1.7. Assume that the rack is in a quiescent state with no interrupts pending, and that all previous interrupts have been reset. The D type flip-flop, F1, will have input and output equal, the output of the comparator C1 will be low, the outputs of G3 and G4 will be low, and the CINT- bus line will be high. If the door switch, S1, changes position, then the comparator inputs become unequal, DORCNG goes high, the tristate buffer B3 is enabled and the CINT- line goes low. After a debouncing period (see section A1.3.4) USERINT5- on the system I/O port will also become low flagging a CMOS bus interrupt.

At the first available opportunity the microprocessor begins polling all devices attached to the CMOS bus. For the racks it will read the status register of each rack from 0 to F. When the rack generating the interrupt is read the RACKRD- line and the output of the register select comparator goes low

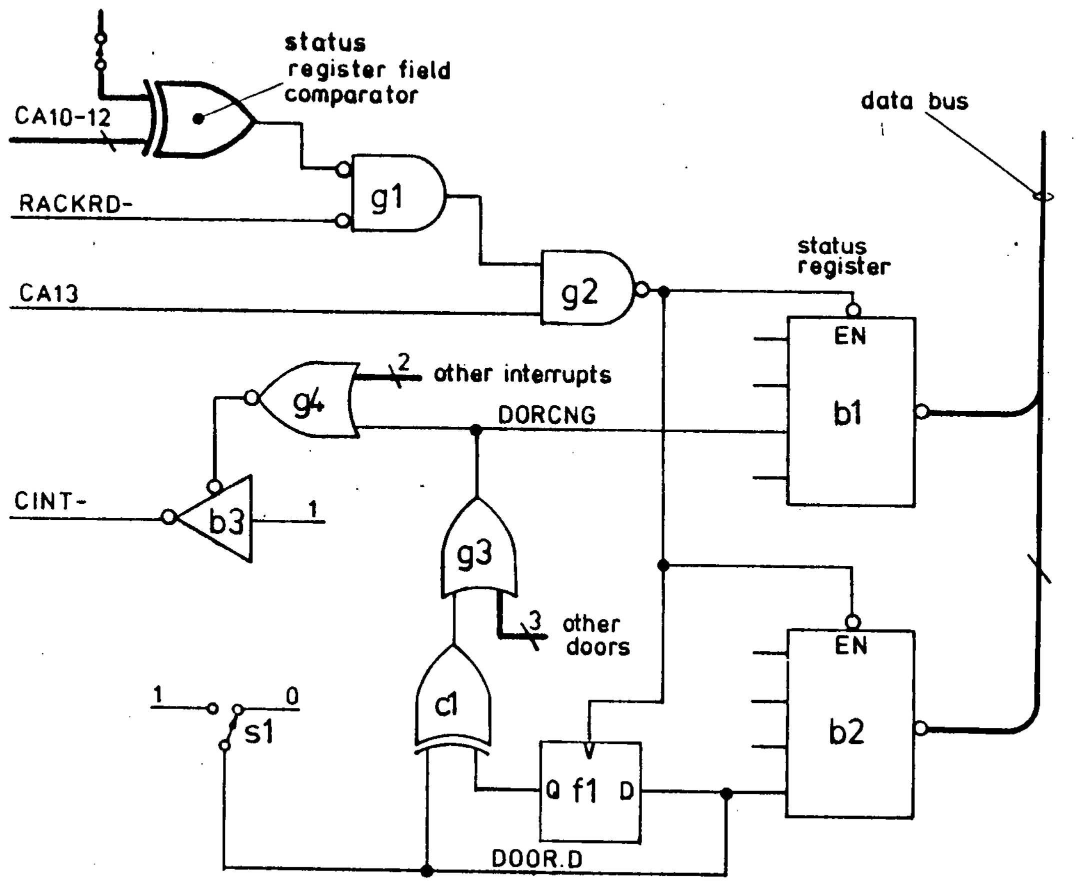


Figure A1.7 Rack Status and Interrupt Logic

sending G1 high. During the second part of the read cycle CA13 also goes high, enabling B1 and B2 via G2, and placing the status register lines on the data bus. When the processor completes its read cycle the output of G2 returns high, and clocks the edge-triggered latch, F1. The two inputs to the comparator C1 therefore equalise and CINT- returns high, resetting the interrupt.

In the meantime, having fetched the rack status register, and having tested it, the microprocessor software has determined the interrupt source, and takes approprate action.

Other doors use identical circuitry to that shown here for DOOR.D, sending DORCNG low through G3. In a similar fashion other interrupts lower the CINT- line through G4.

A1.4.3 Rack Write Operations

The operation of the write logic is straightforward, and a simplified circuit incorporating all its elements is shown in figure A1.8.

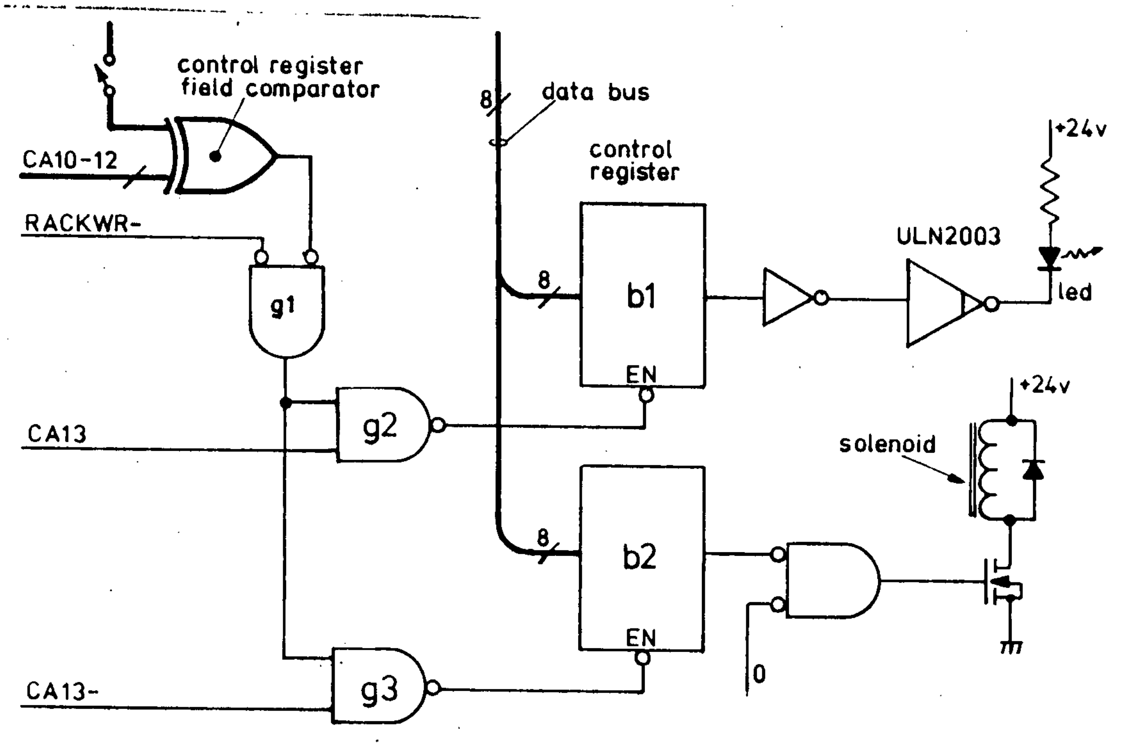


Figure A1.8 Rack Write Logic

When the rack and register are addressed the output of G1 will go high after valid data appears on the data bus. During the high byte part of the write cycle, CA13 will be high and will clock data, via G2, onto the output of B1 from the data bus. During the low byte CA13- will be high and will enable the latch of B2 via G3.

The solenoid drivers are MOS V-FETS choosen for their power capabilities and simplicity of interface. A diode protects them against destructive back emfs during shut off. The LED drive is part of a package designed to drive seven segment LED displays (Sprague Type ULN2003), and would probably be suitable for driving the solenoids as well. This is recommended in future versions.

Because the CMOS bus is an inverted image of the microprocessor one, inverters are placed between the latches and drivers. In the case of the solenoids a gate performs the function. The spare input is intended to act as a safety interlock when used in conjunction with sample sense switches. If a sample tube is not present the switch will hold the gate input high and prevent the solenoid from functioning.

A1.5 <u>THE MASTER CONTROL BOARD</u>

The control and status registers are read and written in an identical manner to their rack counterparts described earlier, but a cruder, more ambiguous address decoding scheme is used.

The circuitry that generates sample request interrupts uses master and slave flip-flops to effect the edge-triggered interrupt, and to prevent the fatal loss of interrupts arriving during a master status read cycle (figure A1.9). When there is no read cycle in progress, the buffer enable line will be high, the tristate buffers will be in their high impedance state, and the gate G2 will be transparent to signals arriving at its input. A sample request from the inlet line will lower

ISREQ- lighting the LED, and saturating the transistor in the

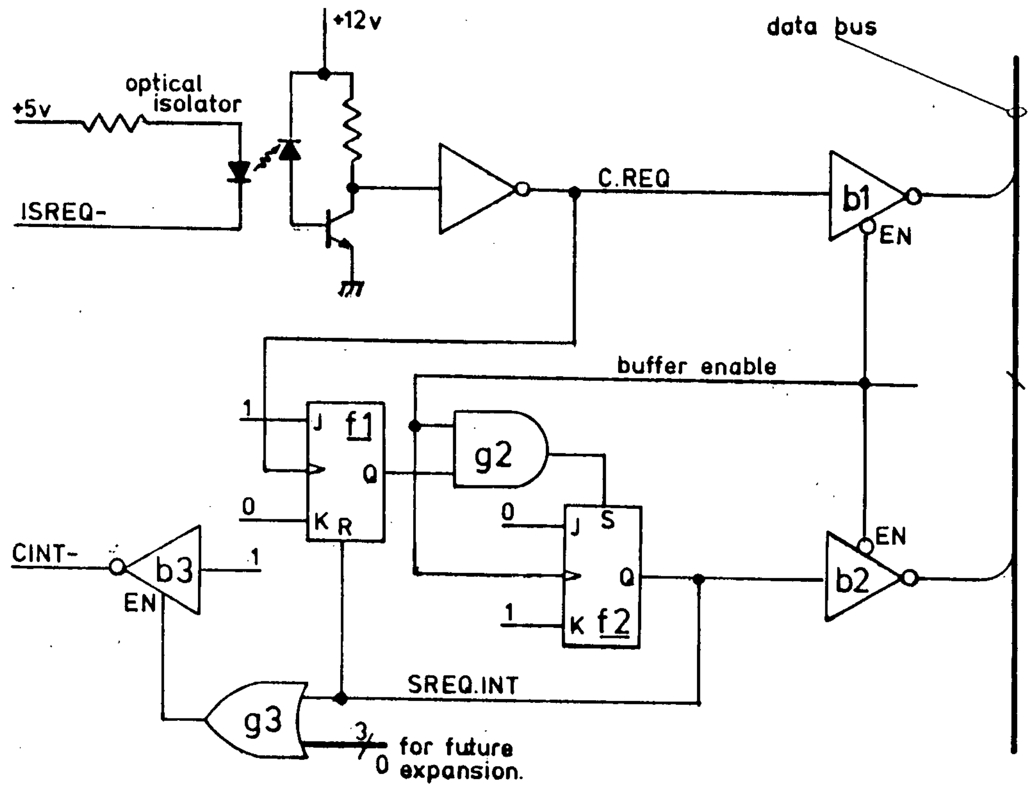


Figure A1.9   Sample Request Interrupt Logic

optical isolator. An inverter cleans up the isolator signal causing a sharp rise in the C.REQ line that clocks the output of the JK flip-flop (F1) high. When G2 is transparent the output of F1 sets F2, raising SREQ.INT and resetting F1 in preparation for the next interrupt. The CINT- line is lowered when the SREQ.INT enables the tristate buffer (B3) via G3.

When the master status register is read the buffer enable line will be lowered enabling B1 and B2, and placing the sample request (C.REQ) and interrupt flags (SREQ.INT) on the data bus. At the end of the read cycle the buffer enable line is raised, disconnecting B1 and B2, and resetting F2. If an interrupt arrived during the read cycle then F1 would have been set. Gate G2 would have prevented F2 from being set and F1 reset, but will allow the sequence to continue as soon as the buffer enable signal returns high.

APPENDIX II. PROGRAM LISTING OF THE SAMPLE LINE OPERATING SYSTEM

Microfiche filed separately in Special Collections

MICROFICHE INDEX

MICRO FICHE
in Special
Collections -
Bookcase 2

APPENDIX III. REFERENCES CONSULTED IN THIS THESIS

Ahern, T.K. (1975) "An O^{18}/O^{16} Study of Water Flow in Natural Snow". M.Sc. Thesis, University of British Columbia, 164pp.

Ahern, T.K. (1980) "The Development of a Completely Automated Mass Oxygen Isotope Spectrometer". Ph.D. Thesis, University of British Columbia, 181pp.

Craig, H. (1957) "Isotopic standards for carbon and oxygen, and correction factors for mass spectrometric analysis of carbon dioxide". Geochimica et Cosmochimica Acta. Volume 12, pp133-149.

Craig, H., Gordon, L.I., and Horibe, Y. (1963) "Isotopic exchange effects in the evaporation of water". Journal of Geophysical Research. Volume 68, number 17, pp5079-5087.

Dijkstra, E.W. (1965) "Co-operating sequential processes". Mathematics Department, Technological University, Eindhoven, The Netherlands.

Dijkstra, E.W. (1968) "Co-operating sequential processes". In F.Genuys (ed.), Programming Languages, Academic Press, New York, pp43-112.

Dushman, S. (1962) "Scientific Foundations of Vacuum Technique". 2nd Edition, John Wiley & Sons, Inc.

Epstein, S. and Mayeda, T. (1953) "Variations of the O^{18} Content of Waters from Natural Sources". Geochimica et Cosmochimica Acta., volume 4, 213-224.

Feller, W. (1968) "An Introduction to Probability Theory and its Applications". Wiley, New York, 2nd Edition.

Frank, H.S. (1929) "Low pressure adsorption on a washed glass surface". Journal of Physical Chemistry, Volume 33, pp970-976. Also see Dushman (1962) for a discussion of these and other results.

Intersil (1981) "Data Book 1981". pg 4-118, Intersil, Inc. 10710 N.Tantau Avenue, Cupertino, California, 95014, U.S.A.

Kollar, F. (1960) "The precise intercomparison of lead isotope ratios". Ph.D. Thesis, University of British Columbia, 107pp.

Langmuir, I. (1918) "The adsorption of gases on plane surfaces of glass, mica, and platinum". Journal of the American Chemical Society, Volume 40, pp1361-1402. Also see Dushman (1962) for a precis and furthur references.

McHaffie, I.R. and Lehner, S. (1925) "The adsorption of water from the gas phase on plane surfaces of glass and platinum". Journal of the Chemical Society, Volume 127, pp1559-1572.

McKinney, C.R., McCrea, J.M., Epstein, S., Allen, H.D. and Urey, H.C. (1950) "Improvements in mass spectrometers for the measurement of small differences in isotope abundance ratios". Review of Scientific Instruments. Volume 21, pp724-730.

Micromass (1978) "Water Isotopic Analysis", Publication 02.254/1, Dec 78 DL/WDU. VG Isotopes Ltd., Ion Path, Road Three, Winsford, Cheshire, CW7 3BX, England.

Mills, G.A. and Urey, H.C. (1939) "Oxygen exchange between carbon dioxide, bicarbonate ion, carbonate ion, and water". Journal of the American Chemical Society. Volume 61, pg534.

Mills, G.A. and Urey, H.C. (1940) "The kinetics of isotopic exchange between carbon dioxide, bicarbonate ion, carbonate ion and water". Journal of the American Chemical Society. Volume 62, pp1019-1026.

Nier, A.O. (1940) "A mass spectrometer for routine isotope abundance measurements". Review of Scientific Instruments. Volume 11, pp212-216.

Nier, A.O. (1947) "A mass spectrometer for isotope and gas analysis" Review of Scientific Instruments. Volume 18, number 6, pp398-419.

Nier, A.O., Ney, E.P., and Inghram, M.G., (1947) "A null method for the comparison of two ion currents in a mass spectrometer". Review of Scientific Instruments. Volume 18, number 5, pp294-297.

Russell, R.D. and Ahern, T.K. (1974) "Economical mass spectrometer ion current measurement with a commercial parametric amplifier". Review of Scientific Instruments. Volume 45, number 11, pp1467-1469.

Russell, R.D., Blenkinsop, J., Meldrum, R.D., and Mitchell D.L. (1971) "On-line computer assisted mass spectrometry for geological research". Mass Spectroscopy. Volume 19, number 1, pp19-36.

Russell, R.D. and Koerner, R. "DEL O^{18} variations in snow on the Devon Island ice cap, Northwest Territories, Canada". Volume 16, number 7, pp1419-1427.

Roether, W. (1970) "Water-CO_2 exchange set-up for the routine oxygen-18 assay of natural waters". International Journal of Applied Radiation and Isotopes. Volume 21, pp379-387.

Shaw, A.C. (1974) "The logical design of operating systems". Prentice-Hall, Inc., Englewood Cliffs, N.J., U.S.A.

Smith, D.C., Irby, C., Kimball, R., Verplank, B., and Harslem,

E. (1982) "Designing the Star user interface". Byte, Volume 7, number 4, pp242-282.

Staschewski, D. (1964) "Experimantelle bestimming der O^{18}/O^{16} trennfaktoren in den systemen CO_2/H_2O und CO_2/D_2O". Bunsen Gesellschaft fur Physikalische Chemie. Volume 68, pp454-457.

Texas Instruments (1979) "TM990/180 Microcomputer Users Guide". Publication #1602004-9701.

Weizenbaum, J. (1965) "ELIZA - A computer program for the Study of Natural Language Communication between Man and Machine". Communications of the Association for Computing Machinery, Volume 9, number 1. Also see Winston (1977), pages 323,333-335, for an easily read summary.

Winograd, T. (1972) "Understanding Natural Language". Ph.D. Thesis, Academic Press. Also see Winston (1977), pages 157-166.

Winston, P.H. (1977) "Artificial Intelligence", Addison-Wesley Publishing Company Inc., 444pp.