REAL TIME MULTIPLE TAU AUTOCORRELATOR AND ITS APPLICATION IN DYNAMIC LIGHT SCATTERING

A thesis presented in partial fulfilment of the
requirements for the degree of Doctor of Philosophy
in Physics at Massey University

YI LUO

1996

DEDICATION

To my parents, my father Luo Binjie and my mother Yang Xinghua

ABSTRACT

Dynamic light scattering (DLS) has recently been extended from a study of translational diffusion coefficients in dilute solution to a means of obtaining distributions in relaxation times over wide ranges of decay times. Many different data analysis algorithms have been developed to extract information on distribution functions (radius distribution, diffusion coefficient distribution, etc.) from photon correlation functions. Obtaining these distribution solutions usually involves the inversion of the Laplace Transform, and this can lead to ill-conditioned Fredholm integral equations of the first kind. These problems can be minimised by using an optimised time scale for the autocorrelation function. The need to handle such a dynamic range within one correlation function led to the development of the Multiple Tau Autocorrelator.

This thesis describes the design and construction of a real time multiple tau autocorrelator and its application in DLS.

An introduction to the theory of DLS is given. The theoretical background of the experiments is discussed and DLS techniques are reviewed. Particular emphasis is placed on experimental techniques and experimental data analysis procedures. The relationship among the intensity, amplitude, and photon count correlation functions are discussed.

Data analysis methods based on obtaining distributions in relaxation times over wide ranges of decay times are discussed. Different hardware correlator system design techniques are reviewed. A correlator based on multiple tau techniques and symmetric normalisation is discussed.

The advantages of using a multiple-instruction-multiple-data (MIMD) system to perform multiple tau autocorrelation is examined. The novel multiple digital signal processor (DSP) architecture for real time implementation of multiple tau autocorrelation is developed based on the task scheduling analysis, interconnection performance, and parallel processing. Detailed explanations of the operation of the Motorola DSP56001 digital signal processor are given, including architecture, addressing modes, instruction sets and peripheral access.

The design and construction of the real time multiple tau autocorrelator is described. Detailed descriptions of hardware circuits and software are given. The correlator has proved satisfactory in its applications. The instrument can also works as spectrum analyser or other real time digital signal processing station.

Two sets of experiments on ternary polymer solutions were carried out using the multiple tau correlator. The results of dynamic light scattering measurements are discussed within the broad framework of the Borsali-Benmouna theory. Experimental data are analysed using the constrained regularisation method. The interdiffusion coefficient and cooperative diffusion coefficient of PS/PMMA/thiophenol and PS/PMMA/toluene system under "optical tracer" conditions are discussed, and the interesting features of polymer polymer interaction as the temperature is varied are also discussed.

Advantages of a multiple tau correlator over a linear correlator in the more complicated ternary polymer solution studies is demonstrated.

ACKNOWLEDGMENTS

First of all, I would like to thank my parents, for their love, support, encouragement, and sacrifice during my Ph.D study.

I wish to express my sincere gratitude to my chief supervisor, Assoc. Prof. D.N. Pinder, for his support, constant inspiration, supervision, patience and encouragement throughout this research. His willingness to provide advice and assistance in both research and personal aspects at any time is genuinely appreciated.

I wish to express my sincere gratitude to my co-supervisor, Assoc. Prof. R.C. O'Driscoll, for his support, continuous supervision, expertise, encouragement, constructive ideas, and valuable guidance during this study.

They both have given up much of their time to constructively criticise the writing of this thesis. Their enthusiastic support ensured that every encouragement and opportunity was given to complete this thesis. They have imparted skills that will invaluable in the future.

I would like to express my appreciation to my good friend, Mr. R. Dykstra from the Electronics Workshop, for his helpful assistance, constructive discussions, useful suggestions and technical support. As a good friend, his friendship will always be remembered.

I am also grateful to Professor R.R. Brooks and Professor P.T. Callaghan, for their support of my Ph.D study. Thanks to the Massey University for my appointment as Graduate Assistant.

The staff of the Department of Physics have been very helpful throughout the duration of my Ph.D project.

In particular to the technical staff of Electronics Workshop, Mr. K. Whitehead, Mr. U. Von Mulert, Mr. G. Harrigan, and Mr. P. Lewis, for their cooperation, advice, free access to workshop facilities, good humour, and friendship. Also Mrs K. Owens and Mr. B. Ford, for providing apparatus.

It has been a pleasure to associate with the postgraduate students I have met here at Massey during the past few years.

TABLE OF CONTENTS

MAIN GLOSSARY

$A(\tau)$	Distribution function of relaxation time
A_+	Amplitude of the fast mode
a	Radius of the particle
B	Photon correlation function baseline
D	Diffusion coefficient
D_C	Cooperative diffusion coefficient describing the relaxation of concentration fluctuations
D_I	Interpenetration diffusion coefficient describing the relaxation of composition fluctuations
D_{S2}	Self diffusion coefficient of polymer 2 in polymer solution
e	Electronic charge
$\boldsymbol{e}_y$	Unit vector in the direction of polarisation vector
$\boldsymbol{e}_z$	Unit vector in the direction of beam propagation
$F(x)$	Kawasakifunction
$G(r)$	Variable accessible experimentally in the Fredholm integral of the first kind, $G(r) = \int K(r,s)A(s)ds$
$A(s)$	Function characteristic for the system under investigation in the Fredholm integral of the first kind
$K(r,s)$	Specific for the experimental method, in our case $K(r,s) = e^{-rs} = e^{-\Gamma t}$, and $r = t$, $s = \Gamma$
$g^{(1)}(t)$	Normalised amplitude or first-order correlation function
$g^{(2)}(t)$	Normalised intensity or second-order correlation function
$G^{(1)}(t)$	Unnormalised amplitude or first-order correlation function
$G^{(2)}(t)$	Unnormalised intensity or second-order correlation function

$G_n(k)$	Photon correlation function
$g_n(k)$	Normalized photon correlation function
$G_e(k)$	Constitutes an unbiased estimator for the photon correlation function $G_n(k)$
$G(k)$	Raw correlation function
$g(k)$	Normalized raw correlation function
$g^{(sym)}(k)$	Symmetric normalised raw correlation function
I	Intensity (expressed as the number of photons reaching the detector in unit time)
k_B	Boltzmann constant
k	Magnitude of the propagation vector $k = \frac{2\pi}{\lambda}$
k_i	Initial wave vector
k_f	Final wave vector
m	Mass of the particle
M	Finite number of samples
n_c	Direct signal in the clipping correlator
n'	Clipped delay signal in the clipping correlator
n^2	Square of average count rate
n_{e0}	Standard monitor channel
n_{ek}	Monitor channel for an individual correlation channel
n_j	Number of photon detection pulses counted during a particular sampling time interval $j\Delta T$
$n(t)$	Number of photons detected in period ΔT
N_1	Degrees of polymerisation of polymers 1 in ternary polymer solution

N_2	Degrees of polymerisation of polymers 2 in ternary polymer solution
p	Quantum efficiency of the detector
$P(q)$	Particle form factor
$\boldsymbol{q}$	Scattering vector
q	Magnitude of the scattering vector $q = \lvert\boldsymbol{q}\rvert = 2k\sin\left(\frac{\phi}{2}\right)$
R_g	Random coil radius of gyration
$R_j(\tau)$	Photon current density correlation function at one point of the photocathode surface
$S_j(\omega)$	Power spectrum of the photocurrent density
t_d	Counting dead time
t_r	Hydrodynamic relaxation time
t	Time
T	Temperature
T_c	Critical temperature of the solution
x	Relative abundance of polymer 2 in ternary polymer solution
$w(\Gamma)$	A continuous distribution function of decay rates Γ
β	Intensity intercept
χ	Polymer-polymer interaction parameter
χ_c	Critical value of the polymer-polymer interaction parameter
ΔT	Sample time
ε	Reduced temperature given by $(T - T_c)/T$
ϕ	Polymer volume fraction
θ	Scattering angle

Γ	Decay rates
η	Viscosity of the solvent
μ	Mean number of detection pulses per sample time ΔT
λ	Wavelength of the incident laser light
ρ	Density of the particle
τ	Lag time
υ	Polymer excluded volume
ω	Angular frequency of the incident laser light
ξ	Dynamic correlation length

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.0 Background

Dynamic light scattering (DLS) is an experimental technique commonly found in laboratories concerned with fundamental studies of macromolecular systems. The technique has the space and time resolution necessary for testing a variety of theoretical models of macromolecular solutions.

Digital correlation techniques are widely used to extract spectral information from stochastic processes such as, for example, the analysis of very small frequency variations in the light scattered from dynamic systems (Berne and Pecora, 1976), (Pecora, 1985). In recent years, important advances have occurred which have transformed the utility of the technique.

While the general structure of digital correlators has not changed much, new ways have had to be found to evaluate correlation functions with a large spread of time scales. Many experiments involve stochastic processes with a frequency dynamic range of more than 10^{10}. In such dynamic light scattering experiments, one would like to resolve large ranges of coherence times (and hence of scattering particle size or diffusion constant) in a single measurement. In addition, dynamic light scattering has recently been extended from a study of translational diffusion coefficients in dilute solution to a means of obtaining distributions in relaxation times over wide ranges of decay times (Brown, 1993).

Many different data analysis algorithms have been developed to extract information on distribution functions (radius distribution, diffusion coefficient distribution, etc.) from photon correlation functions (Johnsen, 1988). Obtaining these distribution solutions usually involves the inversion of a Laplace Transform, and this can lead to ill-conditioned Fredholm integral equations of the first kind (Provencher, 1982). These problems can be minimised by using an optimised time scale for the autocorrelation function to ensure, first, that the initial part of the autocorrelation function is sampled at a sufficiently short sample time to provide the required high frequency response, and secondly, that the time scale extends far enough for the autocorrelation function to decay well into the noise level.

The need to handle such a dynamic range within one correlation function led to the development of the Multiple Tau Correlation Technique (Schatzel, 1985). With a multiple tau correlator, it is possible to measure accurately autocorrelation functions

covering many decades, and it is also possible to use new data analysis methods to determine the distribution function $A(\tau)$ of decay times τ.

The DLS group of the Physics Department at Massey University decided to investigate possible applications of modern computing and electronics to photon correlation spectroscopy by designing a new autocorrelator employing the multiple tau correlation technique to calculate the full autocorrelation function in real time on a pseudo-logarithmic time scale.

Real time implementation of the Multiple Tau Correlation Technique is an important requirement for the instrument not only because it speeds up data acquisition, but also because it permits monitoring of the data as they are collected so that any data nonstationarity can be readily detected. Real time correlation requires successive input samples to be processed quickly in order to produce an output channel data stream.

The real time hardware implementation of logarithmic time scale correlation can be expensive and complex. However recent developments in computer architectures have led to a new class of microprocessor, the digital signal processor (DSP). The high degree of parallelism of the DSP together with a hardware multiply-and-accumulate unit make it possible to calculate logarithmic time scale autocorrelation functions in real time. The very efficient hardware use of the Multiple Tau Correlation Technique enabled us to design a correlator as a kind of real time signal processing station. A logarithmic time scale can be achieved by increasing the sampling interval in proportion to the autocorrelation lag time. The overall signal processing speed can be enhanced by the use of multiple processors.

However, particular problems like bias and unsatisfactory statistical accuracy typically arise at very large lag-times when the multiple tau technique is used. A new normalisation scheme has been developed which solves the accuracy problem and provides reliable photon correlation data at large lag-times within significantly reduced total measurement times (Schatzel, Drewel, 1988).

This thesis describes the design of a real time pseudo-logarithmic time scale autocorrelator employing the Multiple Tau Correlation Technique. Real time implementation of pseudo-logarithmic time scale autocorrelation algorithms using a multi-DSP system is investigated. The method is based on a parallel block processing approach, where continuously supplied input data are processed concurrently by being assigned to four digital signal processors in the system. This approach requires only a simple interconnection network, thereby significantly reducing the number of communication links among the processors. In addition, various digital signal

processing algorithms can be implemented on the same multi-processor system by simple software changes.

1.1 Research Project

The aim of the research presented in this thesis was to design and construct a real time pseudo-logarithmic time scale autocorrelator, and use the instrument to perform new measurements of dynamic properties of polymer solutions.

The instrument was required to:

(1) work in a real time mode,

(2) have a pseudo-logarithmic time scale,

(3) have multi bit processing accuracy, and

(4) be compatible with the existing apparatus.

1.2 Correlator Overview

The system is based the use of the Motorola DSP56001 digital signal processor as the correlator processing unit. Efforts were made to build a parallel processing system which took advantage of the dedicated parallel architecture of the DSP56001 in order to optimise the utilisation of hardware resources.

The correlator is controlled by a Macintosh IIvx computer. The user interface program was created using National Instruments LabVIEW-2 software. LabVIEW is an acronym for Laboratory Virtual Instruments Engineering Workbench. It is a data flow language, as distinct from control flow languages like FORTRAN or PASCAL.

The correlator uses a multiple instruction multiple data (MIMD) multi-DSP parallel processing system to realise the multiple tau technique. The main features of the instrument are:

(1) Pseudo-logarithmic time scale in real time

(2) Simultaneous lag times 3.2μs...2s with 24x24 bit processing

(3) Symmetric normalisation with individual monitors

(4) Interfaces to NuBus computer, or PC Bus computer by NATIONAL INSTRUMENTS DIO-24 interface boards

(5) Standard window real time display

(6) Multiple working mode - autocorrelator, spectrum analyser or other real time digital signal processing station are selected simply by push button control

(7) High-level language (LabVIEW or C) used for control and analysis software

1.3 Thesis Organisation

This thesis is organised in the following sections. Chapter 2 reviews Dynamic Light Scattering theory and includes a review of the data analysis methods used in DLS experiments. Chapter 3 discusses the correlator system and photon correlation analysis techniques required to design a pseudo-logarithmic time scale autocorrelator. Chapter 4 reviews autocorrelator design methods, this chapter also gives the detailed design of the autocorrelator and the whole structure of the correlator system. Details of circuit operation are presented in Chapter 5. The correlator construction and testing are described in Chapter 6. Chapter 7 demonstrates the operation of the correlator in DLS experiments. Two different sets of experiments are discussed. Chapter 8 includes conclusions and suggestions for further work. Included in the appendices are the correlator specifications, the circuit diagrams and publications.

2 DYNAMIC LIGHT SCATTERING

2.0 Introduction

Dynamic light scattering technology is based on the high-speed processing of stochastic optical signals scattered from the polymer samples under study. There are two important technical aspects of DLS to be considered. First, there must be a correct relationship between the polymer samples and the scattered stochastic optical signal. Time dependent correlation functions provide a concise method for expressing the degree to which two dynamic properties are correlated over time, and this property makes the correlation function the most important mathematical tool available in dynamic light scattering experiments. Secondly, the DLS data must be collected as precisely and as rapidly as possible. This second aspect is the major concern of this thesis.

This chapter presents a brief review of DLS theory. After that, it will focus on the details of obtaining an estimate of the correlation data and on proper normalisation procedures.

2.1 Dynamic Laser Light Scattering

When light impinges on matter, the electric field of the light induces an oscillating polarisation of the electrons in the molecules. The molecules then serve as secondary sources of light and subsequently radiate (scatter) light. The frequency shifts, the angular distribution, the polarisation, and the intensity of the scattered light are determined by the size, shape, and molecular interactions in the scattering material.

For visible light at room temperature T, the energy of a photon (given by the product of light frequency and Planck's constant) is significantly larger than $k_B T$, (the average energy in a single degree of freedom of a physical system at thermal equilibrium), where k_B is the Boltzmann constant. This fact allows single photons of light to be detected with moderate effort.

Thus from the light scattering characteristics of a given system it should be possible, with the aid of electrodynamics and the theory of time dependent statistical mechanics to obtain information about the structure and molecular dynamics of the scattering medium. These are the general principles of the dynamic light scattering experiment.

A typical DLS system uses a single, moderately focused, linearly polarised laser beam to illuminate a number of scattering particles. The scattered light then enters a detector as shown in Figure 2.1. The signal analysis system could be either an autocorrelator or a spectrum analyser.

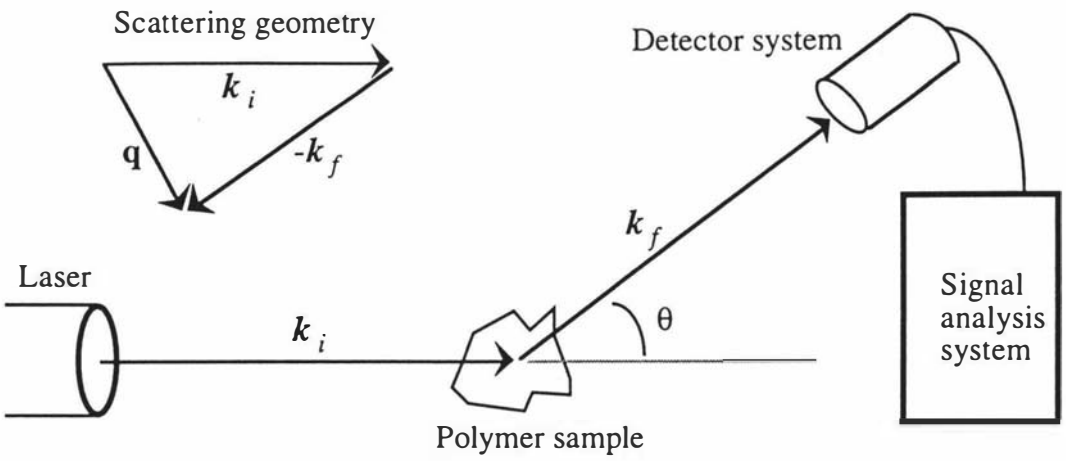


Figure 2.1 Scattering system and geometry

Several excellent collections of books and conference proceedings provide a detailed discussion of the field (Pusey and Vaughan, 1975), (Berne and Pecora, 1976), (Degiorgio, Corti, 1980), and (Pecora, 1985). This section presents an analysis of the system and a review the intensity statistics obtained in typical dynamic light scattering experiments. A purely classical treatment of electrodynamics is perfectly adequate for the purpose.

The incident laser beam can be well described by the plane wave approximation

$$\boldsymbol{E}(x,t) = \boldsymbol{e}_y E_0(x,y) e^{(ikz - i\omega t)} \tag{2.1}$$

where $\boldsymbol{e}_y$ is the unit vector in the direction of the polarisation (or electric field) vector; k denotes the magnitude of the propagation vector equal to $\frac{2\pi}{\lambda}$, λ and ω are the wavelength and angular frequency of the incident laser light, and t is the time.

The scattered light can be observed by a point detector placed in the (z, x) plane at a scattering angle θ with respect to $\boldsymbol{e}_z$, where $\boldsymbol{e}_z$ is the unit vector in the direction of beam propagation. For quasi-elastic light scattering, the magnitude of the final wave vector or the final propagation constant is given by k.

This scattering process involves an initial wave vector

$$\boldsymbol{k}_i = k\boldsymbol{e}_z \tag{2.2}$$

and a final wave vector

$$\boldsymbol{k}_f = k\boldsymbol{e}_x \sin\theta + k\boldsymbol{e}_z \cos\theta \tag{2.3}$$

From the scattering geometry shown in Figure 2.1, we can obtain their difference, the scattering vector

$$\boldsymbol{q} = \boldsymbol{k}_i - \boldsymbol{k}_f \tag{2.4}$$

with magnitude

$$q = |\boldsymbol{q}| = |\boldsymbol{k}_i - \boldsymbol{k}_f| = 2k\sin\left(\frac{\theta}{2}\right) \tag{2.5}$$

which may be varied between zero and $4\pi/\lambda$ by a proper choice of the scattering angle θ.

The light scattered by the single particle j can be characterised by a scattering amplitude b_j and an optical phase determined by the scalar product of particle position x_j and scattering vector $\boldsymbol{q}$:

$$\boldsymbol{E}_j(\boldsymbol{q},t) = \boldsymbol{e}_y E_0(x_j, y_j) b_j e^{(i\boldsymbol{q}\boldsymbol{x}_j - i\omega t)} \tag{2.6}$$

Assuming that the scattering process does not change the initial state of polarisation (or using a suitable polariser in front of detector), and omitting the vector character of the field as well as its explicit time dependence, the scattered light can be expressed as:

$$u_j(\boldsymbol{q},t) = E_0(x_j, y_j) b_j e^{i\boldsymbol{q}\boldsymbol{x}_j} \tag{2.7}$$

The weak focusing of the illuminating beam makes $E_0(x_j, y_j)$ a weakly varying function of particle position. The temporal behaviour of the complex amplitude is clearly dominated by changes of the phase factor, and it is quite useful to absorb $E_0(x_j, y_j)$ into a new scattering amplitude

$$a_j = E_0(x_j, y_j) b_j \tag{2.8}$$

This scattering amplitude a_j now contains the whole geometry of the experiment, including the particle position x_j within the beam, as well as the beam power and the form factor, which describes the scattering of the particle j for a scattering vector $\boldsymbol{q}$.

To make a connection between molecular properties and light scattering measurements it is convenient to introduce the assumption that the light is scattered from particles and that the dielectric constant fluctuations are due solely to fluctuations in the number density of particles. For the N particle system, we need to sum the scattered amplitudes of all N particles in the measurement volume to obtain a complex amplitude

$$u_f(\boldsymbol{q},t) = \sum_{j=1}^{N} a_j(x_j) e^{i\boldsymbol{q}\boldsymbol{x}_j} \tag{2.9}$$

Note the implicit time-dependence of this amplitude due to motion of the particles, that is, the x_j are functions of time $x_j(t)$. After few assumptions (Berne and Pecora, 1976) and the use of the complex temporal autocorrelation mathematics, we can determine the statistical properties of the complex amplitude $u_f(q,t)$ as a function of time.

Important results follow: Not only does the complex amplitude $u_f(q,t)$ yield Gaussian statistics if considered it at a single time, but also the two-time probability density yields joint Gaussian statistics. More importantly, the procedure just outlined can be continued to prove all higher-order statistics to be Gaussian. This scattering process thus engenders scattered light whose statistics are Gaussian.

This property, the common occurrence of Gaussian statistics for the complex amplitude at the detector in light scattering experiments, underlines the great practical importance of autocorrelation measurements. The temporal autocorrelation

$$G^{(1)}(t) = \left\langle u_f(0) u_f(t)^* \right\rangle \tag{2.10}$$

is the lowest-order time-dependent moment. All higher-order moments may be decomposed into expressions involving just the temporal autocorrelation function $G^{(1)}(t)$, also known as amplitude or first-order correlation.

Because of these results, we can focus our discussion on the intensity or second-order correlation, which is measurable experimentally, as will be shown in chapter 3.

$$G^{(2)}(t) = \left\langle \left|u_f(0)\right|^2 \left|u_f(t)\right|^2 \right\rangle \tag{2.11}$$

Such a moment is obtained by summing over all possible distinct permutations of the amplitudes involved. Non-zero expectations are only obtained for pairs of amplitudes, where the members of the pair are complex conjugate. The following very important result is obtained:

$$\begin{aligned} G^{(2)}(t) &= \left\langle \left|u_f(0)\right|^2 \right\rangle \left\langle \left|u_f(t)\right|^2 \right\rangle + \left\langle u_f(0) u_f(t)^* \right\rangle \left\langle u_f(0)^* u_f(t) \right\rangle \\ &= G^{(1)}(0)^2 + \left|G^{(1)}(t)\right|^2 \end{aligned} \tag{2.12}$$

which is the well known Siegert relation. It gives the relationship between the amplitude correlation, which is related to the physical attributes of the solution, and the intensity correlation function which can be measured in a photon correlation experiment. Hence the time dependence of the dielectric constant fluctuations can be studied by measuring the intensity autocorrelation function.

2.2 Single Decay Rate System

2.2.1 Intensity and amplitude autocorrelation function

The temporal first-order autocorrelation function for the case of non-interacting scattering particles undergoing their individual Brownian motion can be easily calculated. This is an adequate model for all suspensions at sufficient dilution. Brownian motion of a colloidal particle is driven by molecular collisions. The random character of this driving force results in a highly irregular particle motion. The velocity autocorrelation of a Brownian particle decays on a time-scale of the order

$$t_r = \frac{m}{6\pi\eta a} = \frac{\rho a^2}{9\eta} \tag{2.13}$$

known as the hydrodynamic relaxation time (Pecora, 1985). Here m denotes the mass, a the radius, and ρ the density of the particle; η is the viscosity of the solvent.

Usually t_r is much less than typical time scales accessed in photon correlation experiments. Hence we may expect particle displacements on our time-scale to be composed of very many independent small displacements. Application of the central limit theorem then predicts Gaussian statistics for each particle displacement δx_j with a second moment that increases linearly with time:

$$\left\langle \delta x_j^2 \right\rangle = 6Dt \tag{2.14}$$

The proportionality factor is six times the particle diffusion coefficient:

$$D = \frac{k_B T}{6\pi\eta a} \tag{2.15}$$

Treating the far-field complex amplitude as a sum over N single-particle contributions, the first-order or amplitude correlation function becomes

$$G^{(1)}(t) = N\left\langle \left| a_j \right|^2 \right\rangle e^{-q^2 Dt} \tag{2.16}$$

where we have used the different time-scales of the a_j and the phase factors to separate their expectations, and the statistical independence of x_j and x_m for j not equal to m to eliminate non-diagonal terms in the double sum. The final expectation over the phase factor is then recognised as the spatial Fourier transform of the particle displacement $\delta x_j(t)$ over a time interval t, which yields an autocorrelation function that is a Gaussian in q and a negative exponential in t. The corresponding normalised first-order correlation function reads (Berne and Pecora, 1976)

$$g^{(1)}(t) = e^{-q^2 Dt} \tag{2.17}$$

Using the Siegert relation, the second-order correlation of the spatially integrated intensity can be obtained

$$G^{(2)}(t) = \langle I \rangle^2 \left[1 + \beta e^{-2q^2 Dt}\right] \tag{2.18}$$

where β is intensity intercept. Equation (2.18) can be rewritten as

$$G^{(2)}(t) = B\left[1 + \beta e^{-2q^2 Dt}\right] \tag{2.19}$$

where B is the baseline. In normalised form

$$g^{(2)}(t) = \beta e^{-2q^2 Dt} \tag{2.20}$$

The intensity correlation function can be estimated by performing a photon correlation experiment. The analysis in terms of a negative exponential readily yields the diffusion coefficient and hence the size of the colloidal particles.

2.2.2 Data analysis of single decay rate system

Eqn (2.17) can be rewritten in a more general form to generally treat the data in terms of decay rates Γ

$$g^{(1)}(t) = e^{-\Gamma t} \tag{2.21}$$

then eqn (2.19) becomes

$$G^{(2)}(t) = B(1 + \beta e^{-2\Gamma t}) \tag{2.22}$$

This relationship is fitted to the data to provide an estimate of the parameter of interest Γ. Basically this can be done in two different ways, either a linear or a non-linear method, depending on whether the value of the baseline is reliably known or not.

If B is reliable and known with sufficient precision, then the simplest way to determine Γ is to transform the data into

$$\ln\left(\frac{G_2(t)}{B} - 1\right) = \ln\beta - 2\Gamma t \tag{2.23}$$

and to fit eqn (2.23) as a linear function of t, with -2Γ the slope and $\ln\beta$ the intercept.

This linear method has the disadvantage that for values of $G^{(2)}(t)$ close to B (ie. for large t), $G^{(2)}(t)/B - 1$ is very small, which greatly enhances the scatter of the data and

introduces some uncertainty in the values of the fitted parameters. Moreover, this procedure is very sensitive to the correct value of B.

Better results can be obtained with a non-linear method, where an initial guess for the parameter Γ is introduced and its value corrected in successive iterations until the desired precision is reached. In this procedure all the three parameters Γ, β, and B are determined by the fit, and no assumption is made about the baseline (Rektorys, 1976), (Press, 1986).

Once the results of the fit are obtained, the goodness of fit has to be analysed. Most descriptive for this purpose is a distribution of residuals,

$$\varepsilon = G^{(2)}(t) - f(t) \tag{2.24}$$

where $f(t) = B\left(1 + \beta\left|g^{(1)}(t)\right|^2\right)$ are the calculated values of the correlation function.

2.3 Multi-decay System

2.3.1 Intensity and amplitude autocorrelation function

The problem of analysing multiexponential autocorrelation functions has existed from the early days of dynamic light scattering and over the past 20 years many analytical procedures have been reported (Brown, 1993). Multiexponential autocorrelation functions are observed under various circumstances. Examples include mixtures of particles or polymers of different sizes (known as polydisperse systems), distributions of particle sizes, and combinations of various diffusion processes, relaxation mechanisms and internal modes of polymers.

Also the occurrence of particle interactions and internal modes of motion (as in flexible macromolecules) result in a broadened distribution of decay times similar to that for a polydisperse sample. An extreme example is represented by colloidal systems approaching a glass transition, where decay times may be spread over many decades (Brown and Fundin, 1991).

For particles of different sizes, we must average over the particle sizes. Note that the scattering amplitude factors a_j depend strongly on particle size and act as weights in the averaging procedure.

As shown above, the intensity correlation functions obtained from diffusing particles typically show exponential relaxation, where the decay times may be distributed over a considerable time range. If there are several decay times τ_i, ie. decay rates Γ_i, present in

the system under investigation, then the field correlation function $g^{(1)}(t)$ is the weighted sum of the individual contributions:

$$g^{(1)}(t) = \sum_i w_i e^{-\Gamma_i t} \tag{2.25}$$

where w_i is the amplitude corresponding to the decay rate Γ_i. For many decay processes, the summation in eqn (2.25) can be replaced by an integral, leading to the expression

$$g^{(1)}(t) = \int w(\Gamma) e^{-\Gamma t} d\Gamma \tag{2.26}$$

where $w(\Gamma)$ is a continuous distribution function of decay rates Γ (inverse decay times). We thus see that $g^{(1)}(t)$ and $w(\Gamma)$ are related by a Laplace transformation. That means that the analysis of autocorrelation data obtained from polydisperse samples typically requires an inverse Laplace transform of the first-order autocorrelation function.

The quantity really measured in an autocorrelation experiment is the intensity autocorrelation function $G^{(2)}(t)$, which is related to $G^{(1)}(t)$ by eqn (2.12) above, then the normalised field correlation $g^{(1)}(t)$ can be obtained.

2.3.2 Data analysis of multi decay rate systems

Most inversion programs are based on an estimate of $g^{(1)}(t)$ such as

$$g^{(1)}(t) = \left[\frac{\frac{G_2(t)}{B} - 1}{\beta} \right]^{\frac{1}{2}} \tag{2.27}$$

in order to obtain $w(\Gamma)$ directly. From eqn (2.26-27), the distribution function of decay times $w(\Gamma)$ is obtained from the measured intensity correlation function $G^{(2)}(t)$ by an inverse Laplace transformation.

This is a special kind of a more general integral transformation

$$G(r) = \int K(r,s) A(s) ds \tag{2.28}$$

called the Fredholm integral of the first kind (Provencher, 1982). $G(r)$ is the variable accessible experimentally and $A(s)$ is a function characteristic for the system under investigation, $K(r,s)$ is specific for the experimental method, in our case $K(r,s) = e^{-rs} = e^{-\Gamma t}$, and $r = t$, $s = \Gamma$. It is well known that the inversion of eqn (2.26) is ill-conditioned (Provencher, 1982a). This means that the solution to eqn (2.26) is unique only in the theoretical case when no noise is present in the data and no rounding errors

occur in the computation of the Laplace inversion. But in practice, measurements are noisy and rounding errors do occur. Thus an absolute answer cannot be extracted from the data and alternative methods must be used to obtain a good estimate.

In the case of DLS, some reasonable constraints can normally be applied to $w(\Gamma)$ in order to find a solution closely approximating to the true one (Johnsen, 1988), (Brown, 1993).

One popularly used method is "parsimony regularisation". The principle of parsimony dictates that the simplest of all possible solutions compatible with the data be taken. Such a solution does not necessarily have all the details of the true solution, but it does fit the data and is less likely to have artificial components.

Parsimony and the Tikhonov's regularisation method have been used by Provencher (Provencher, 1982b) to produce the well-known and widely used program package CONTIN.

2.3.3 CONTIN

Most experiments in the natural sciences are indirect. That is the observed data y_k are related to the desired function of vector x by operators O_k,

$$y_k = O_k x + \varepsilon_k \qquad k = 1, \ldots, N_y \qquad (2.29)$$

where the ε_k are unknown noise components. One is then faced with the problem of estimating x from the noisy measurements y_k.

Many different algorithms have been used to extract x information from eqn (2.29), ranging from simple cumulants to constrained regularisation methods such as CONTIN.

CONTIN is a general purpose program using a regularisation parsimony method for automatically inverting noisy linear algebraic and integral equations. This method has been encoded in a very popular and widely used program package by Provencher (Provencher, 1984).

CONTIN can handle cases where the equations are ill conditioned, this includes Fredholm integral equations of the first kind:

$$y_k \approx \int_a^b K(g,t_k)s(g)dg + \sum_{i=1}^{N_L} \beta_i L_i(t_k) \quad k = 1, \ldots, N_y \qquad (2.30)$$

where the function $K(g, t)$ and the values of the independent variable t_k are known, and $s(g)$ is to be estimated. The extra optional sum over the known $L_i(t)$ and N_L unknown β_i

permits, for example, a constant background term β_i, to be included by setting N_L=1 and all the $L_1(t) = 1$.

In the photon count autocorrelation experiment, eqn (2.30) becomes

$$g^{(2)}(t) = \beta \int_0^\infty e^{-\Gamma t} w(\Gamma) d\Gamma + \varepsilon_k \tag{2.31}$$

The first step in solving eqn (2.30) is to convert it into a set of linear algebraic equations. CONTIN can automatically do this by numerical integration of eqn (2.32),

$$y_k \approx \sum_{m=1}^{N_g} c_m K(g_m, t_k) s(g_m) + \sum_{i=1}^{N_L} \beta_i L_i(t_k) \quad k = 1, \ldots, N_y \tag{2.32}$$

where c_m are the weights of the quadrature formula. The solution, $s(g)$, is then determined at the N_g grid points g_m.

Solving eqn (2.29-30) is generally an ill posed problem, this means that even for arbitrarily small noise levels in the y_k, there still exists a large set of solutions $s(g)$ that all fit the y_k in eqn (2.29-30) to within the noise level.

For example eqn (2.32) can be rewritten as

$$y_k \approx \sum_{j=1}^{N_x} A_{kj} x_j \tag{2.33}$$

then one member of solution $s(g)$ can be the ordinary least squares of eqn (2.33), ie., the set of x_j that satisfies

$$VAR = \sum_{k=1}^{N_y} w_k \left(y_k - \sum_{j=1}^{N_x} A_{kj} x_j \right)^2 = \min \tag{2.34}$$

where the w_k are optional weights that can be assigned. But it is extremely unlikely that this solution will be close to the true solution.

CONTIN computes a constrained regularised solution, which is the set of x_j that satisfying

$$VAR + \alpha^2 \sum_{j=1}^{N_{reg}} \left(r_i - \sum_{j=1}^{N_x} R_{ij} x_j \right)^2 = \min \tag{2.35}$$

subject to the constraints in

$$\sum_{j=1}^{N_t} D_{ij} x_j \geq d_i, \qquad i = 1, \ldots, N_{\text{ineq}}, \tag{2.36}$$

$$\sum_{j=1}^{N_t} E_{ij} x_j = e_i, \qquad i = 1, \ldots, N_{\text{eq}}, \tag{2.37}$$

where the arrays D, E, d, and e can be specified by the user, these options allow user to use a priori knowledge by imposing linear inequality and equality constraints on the solution x_j, for example, to constrain the solution to be nonnegative.

The second term on the left of eqn (2.35) is called the regularizor. Its form is determined by specifying the arrays r and R, its strength is determined by specifying α, the regularisation parameter. The regularizor penalises a solution for deviations from behaviour expected on the basis of statistical a priori knowledge or on the basis of the principle of parsimony. CONTIN generally presents a progression of solutions with increasing α. Once the regularizor and constraints are specified, CONTIN finds the unique solution to eqn (2.35).

CONTIN first computes a Reference Solution with $\alpha = 0$, which provides the globally optimal non-negative exponential fit. The Fisher F-test is used to calculate the probability of rejecting the penalised solution. The solution which best corresponds to this F-test is called the chosen solution.

The availability of correlation data over a large range of lag times is an important prerequisite to obtain an accurate solution using CONTIN. It is important to measure at large enough t_k such that the correlation data have decayed well into the noise, since some slowly decaying components could otherwise be taken as part of the background and vice versa. Also, it is important to measure at small enough t_k to sample the initial rapidly decaying part of the correlation data finely enough. The best way to satisfy these demands is to sample data in a logarithmic time space, increasing the time spacing of the later correlation data - that implies a logarithmic time scale autocorrelation function.

2.4 Power Spectrum and Autocorrelation Function

The scattered light from DLS can be analysed not only in time domain using the autocorrelation techniques but also in the frequency domain using power spectrum techniques (Chu, 1974). According to the Wiener-Khintchine theorem, the power spectrum $S_j(\omega)$ of the photocurrent density and the photon current density correlation function $R_j(\tau)$ at one point of the photocathode surface are related through the equations:

$$R_j(\tau) = \langle j(t+\tau) j(t) \rangle = \int_{-\infty}^{\infty} S_j(\omega) \cos \omega\tau d\omega \tag{2.38}$$

$$S_j(\omega) = \frac{1}{2\pi}\int_{-\infty}^{\infty} R_j(\tau)\cos\omega\tau d\tau \tag{2.39}$$

which means the power spectral density is the time Fourier transform of the photon current density correlation function (where $j(t)$ is the photocurrent density).

And

$$R_j(\tau) = \langle j(t+\tau)j(t)\rangle = e\langle j\rangle\delta(\tau) + \langle j\rangle^2 g^{(2)}(\tau) \tag{2.40}$$

where e is the electronic charge. If $g^{(1)}(\tau)$ has a single decay, then using the Siegert relation, the power spectrum can be shown to be given by

$$S_j(\omega) = \frac{1}{2\pi}e\langle j\rangle + \langle j\rangle^2\delta(\omega) + \langle j\rangle^2\frac{\frac{2\Gamma}{\pi}}{\omega^2 + (2\Gamma)^2} \tag{2.41}$$

and the "light beating" power spectrum is a Lorentzian of half width $\Delta\omega_{1/2} = 2\Gamma$, centred at $\omega = 0$.

The analysis demonstrates that photon count spectroscopy is capable of measuring the decay time Γ from either the autocorrelation function or the power spectrum, especially when there is only one decay Γ. In practice, photon count spectroscopy spectrum analysis becomes too complex whenever there is more than one decay. Signal correlation is a much more efficient way of obtaining the desired information since the rate of data collection with a correlator is faster than that of data collection with a spectrum analyser. A correlator utilises virtually every photocount to compute the time dependent correlation function.

3 CORRELATOR DESIGN I - CORRELATOR MODEL

Two important technical aspects of the DLS experiment were mentioned in chapter 2. The first aspect - establishing the correct relationship between the polymer sample and the scattered stochastic optical signal was discussed in chapter 2. The second aspect, the application of technology to high speed data collection and processing, will be discussed more detail in this and subsequent chapters. This is the major part of the thesis.

This chapter will review a theoretical basis for photon correlation, and then it will discuss how to use the digital correlator to realise a correct photon correlation function. The effect of noise on photon correlation techniques will also be considered. After that the digital correlator system will be discussed, and the correlator model used to design the digital correlator will be established. These considerations formulate the basis for the correlator design. Chapters 4, 5, and 6 will give more detail about the design, manufacture and testing of the correlator system.

3.0 Introduction

As mentioned in chapter 2, the intensity autocorrelation function can be obtained from experiment, and can be used to calculate the decay information of the polymer system.

This chapter will introduce the signal analysis system for the light scattering experiment. Normally the intensity correlation function is not be obtained directly from experiment, rather the photon correlation function is determined. The relationship between the photon correlation function and the intensity correlation function will be established. The photon correlation function can be determined experimentally using a digital correlator. Finally the design of a digital correlator able to measure the photon correlation function as precisely as possible will be discussed.

3.1 Single Photon Counting System

The scattered light must be converted into data appropriate for processing in a digital electronic instrument. This can be achieved using photon counting. The photon count is obtained from a light detector with single photon detection capability. A single photon absorption event typically leads to the separation of a single pair of charge carriers. However, the charge of a single electron is much less than the detection limit of common electronic devices, hence the need for an extremely sensitive amplification mechanism - the photomultiplier tube.

In the photomultiplier tube, the single photo-electron is accelerated by a suitably arranged electric field and hits a dynode. Several secondary electrons are emitted from

the dynode on absorption of the primary electron. After several similar dynodes, a sizeable current pulse reaches the final anode. A resistor in the anode line converts the current pulse into a voltage pulse. These voltage pulses are processed by a preamplifier and discriminator which, in our experiment, produces a single TTL level pulse of 25ns duration for each detected photon.

If the optical signal to be analysed is so intense that many photons are absorbed within the response time of the photomultiplier, photon counting is not possible and the output electric current can be considered as an analog signal proportional to the intensity of the light beam.

In the light scattering experiment considered at here, the optical signal is so weak that it is very unlikely for more than one photon to be detected within the response time of the photomultiplier tube. The resulting output consists of a random train of pulses. Samples of duration ΔT are obtained by integrating all the incoming pulses over the time interval ΔT, using specially designed counters.

At the end of each sample time, determined by the sample time clock (STC), the data recorded in the counter are transferred to the subsequent processing instrument, and the counter is reset for the next sample. In this method the incident data are all treated identically, and the samples are non-overlapping and hence independent. Poisson statistics are obeyed for the number of photon detection pulses. This process can be cycled repetitively until enough samples have been taken to give an adequate estimate of the desired information. Based on this analysis, the counter system model shown in Figure 3.1 was used to obtain the data.

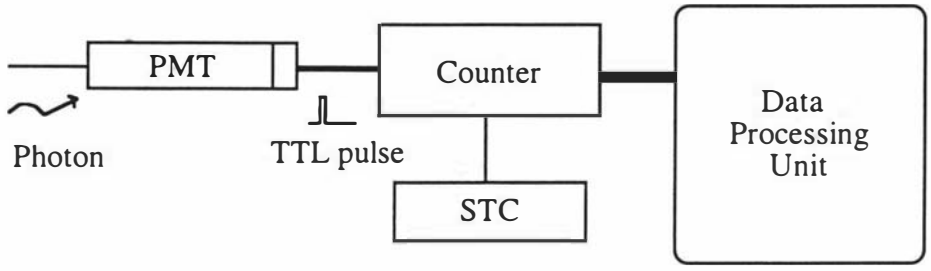


Figure 3.1 The counter system

There is a practical problem which should be noted in this photon counting system. All real devices have a finite temporal resolution. This may be formally described by a suitable counting dead-time model. In real time processing, the counter is inactive while the counter data are transferred to the processing unit and the counter is reset. This introduces a counting dead time t_d after every counting process, so some pulses are lost which gives rise to an error (which can be expressed as follows)

$$\text{Error} = [\text{ counting dead time / sampling time }] 100\% \qquad (3.1)$$

Such a counter system approaches a finite maximum count rate as the sampling rate is raised to very high levels. The practical counter system used in this thesis employs a zero counting dead time technique which reduces the dead time error. This system will be discussed in more detail in chapter 4.

3.2 Photon Correlation

The digital data which can be directly processed by a digital electronic system can be obtained using this kind of photomultiplier and counter system. The intensity statistics obtained in dynamic light scattering experiments have already been established in chapter 2. The connection between time averaged intensities and photon counting statistics will now be considered.

3.2.1 Equivalence of photon correlation and intensity correlation

Photon counting is the recording of the number of detected photons in regularly spaced time intervals as shown in Figure 3.2.

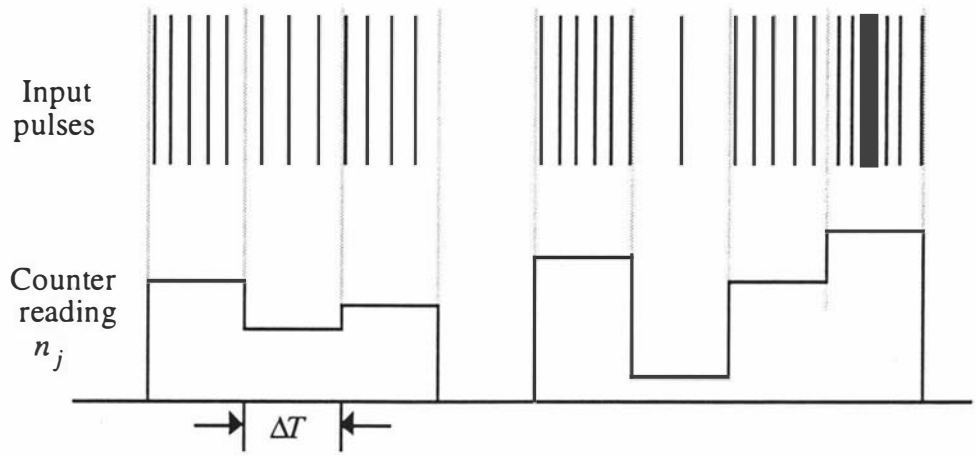


Figure 3.2 Photon counting

Assume n_j is the number of photon detection pulses counted during a particular sampling time interval $j\Delta T$, and that such an ordered sequence of random variables n_j is a realisation of a stochastic process. Then

$$G_n(k) = \langle n_j n_{j+k} \rangle = \langle n_j n_{j-k} \rangle \qquad (3.2)$$

is known as the autocorrelation of the stochastic process, or the photon correlation function.

Each photon detection event is essentially independent of all the others so pulse arrival times are expected to be completely random. Hence the number of photon detection pulses counted during some finite sample time interval ΔT is expected to be governed by Poisson statistics. If I denotes intensity (expressed as the number of photons reaching the detector in unit time) and p is the quantum efficiency of the detector, then the mean number of detect pulses per sample time ΔT is simply

$$\langle n \rangle = \mu = pI\Delta T \tag{3.3}$$

The mean is given by the time integral of the intensity over the sample time interval:

$$\langle n_j \rangle = \mu_j = p\int_{(j-1)\Delta T}^{j\Delta T} I(t)dt \tag{3.4}$$

Hence using basic statistical theory, and assuming stationarity for $I(t)$ (Berne and Pecora, 1976):

$$\langle n_j n_{j-k} \rangle_n = \delta_{k0}\langle \mu_j \rangle_\mu + \langle \mu_j \mu_{j-k} \rangle_\mu \tag{3.5}$$

where $\langle ... \rangle_\mu$ denotes averages over intensity statistics and $\langle ... \rangle_n$ is averages over photon statistics. $\delta_{k0}\langle \mu_j \rangle_\mu$ vanishes for non-zero k. Eqn (3.5) is an expression of the equivalence of photon correlation and intensity correlation. This connection between time-averaged intensities and photon counting statistics is a most important rule of light scattering.

This equality is the very basis of the photon correlation technique. It makes it possible to analysis the intensity correlation function by means of the measured photon count correlation function.

3.2.2 Linear photon correlation function

In a DLS experiment, the photon correlation function $G_n(k)$ is not obtained directly. Instead a finite number of samples M of photon count are obtained over some finite total measurement time $T_m = M\Delta T$. The most common algorithm used to calculate the real-time correlation function yields a photon correlation estimator

$$G_e(k) = \frac{1}{M}\sum_{j=1}^{M} n_j n_{j-k} \tag{3.6}$$

at a lag time $k\Delta T$. The subscript 'e' is used to denote estimators. $G_e(k)$ constitutes an unbiased estimator for the photon correlation function $G_n(k)$. The expectation of $G_e(k)$ is,

$$\langle G_e(k)\rangle = G_n(k) = \langle n_j n_{j-k}\rangle \tag{3.7}$$

No measurement will yield estimator data $G_e(k)$ exactly equal to $G_n(k)$, there will always be some statistical error $G_e(k) - G_n(k)$ (Schatzel, 1990). However, at small lag times, M has almost always to be a very large number (10^6 or more) in order to obtain sufficient averaging over photon counting noise.

3.2.3 Correlator accuracy and correlation function normalisation

The modern inversion programs (CONTIN or maximum entropy analysis) for the analysis of measured correlation functions require precise noise estimates in order to achieve optimum performance, so a more extensive treatment of noise on photon correlation data is needed.

There are two types of noise, due to photon detection as well as classical intensity statistics, and both types of noise must be considered. Detailed analysis of a noise model (Schatzel, 1990) reveals that at small count rates, photon noise dominates experiment, as count rates increase, both the photon detection noise and classical intensity noise decrease. At high input count rates, classical intensity fluctuations noise dominates over photon noise, noise on measured photon correlation functions is largely due to classical intensity fluctuations noise.

One way to reduce the noise is to use baseline subtraction techniques as used in autocorrelation function normalisation. This allows a partial cancellation of count rate fluctuation effects in the unnormalised correlation estimator and the count rate estimator. If the shape rather than the absolute magnitude of an autocorrelation function is of interest (Oliver, 1979), then the normalised photon autocorrelation function can be used to produce noise cancellation. In section 3.3, we will discuss more about noise reduction.

For photon correlation, n_j and n_{j-k} approach statistical independence for large values of k. Consequently the autocorrelation will approach $\langle n_j\rangle^2$ for large values of k. This asymptotic value of the autocorrelation can be used as its baseline, and the normalised correlation function is obtained by baseline subtraction and division as follows

$$g_n(k) = \frac{\left[G_n(k) - \langle n_j\rangle^2\right]}{\langle n_j\rangle^2} \tag{3.8}$$

The normalised autocorrelation function will decay to zero for large values of k, corresponding to large temporal displacements.

The lag time is

$$\tau = k\Delta T \tag{3.9}$$

where ΔT is the sampling time, the time used to obtain the n_j.

Unfortunately, $\langle n_j \rangle^2$ is generally unknown, and therefore a measured value has to be introduced as a best estimate for this expectation. Various schemes have been devised for baseline estimation. The most common involve the use of "far point channels" (measured correlation data at some large lag-time values), or "monitor channels" (special counters which measure the average count rate n), where

$$n = \frac{1}{M}\sum_{j=1}^{M} n_j \tag{3.10}$$

In the monitor channel method, the correlator must provide a count rate estimator n, measured simultaneously with the correlation data, which is used to keep track of all incoming input samples for the complete experimental duration and thus gives an estimate of the expectation $<n>$. The square of this estimator is commonly used to estimate the baseline. Normalisation is then carried out using

$$g(k) = \frac{G(k) - n^2}{n^2} \tag{3.11}$$

where $g(k)$ is the normalised raw correlation function, $G(k) = MG_e(k)$ are the raw correlation channel data.

It is common practice to implement such a monitor channel within digital correlators and use it for normalising the correlation function.

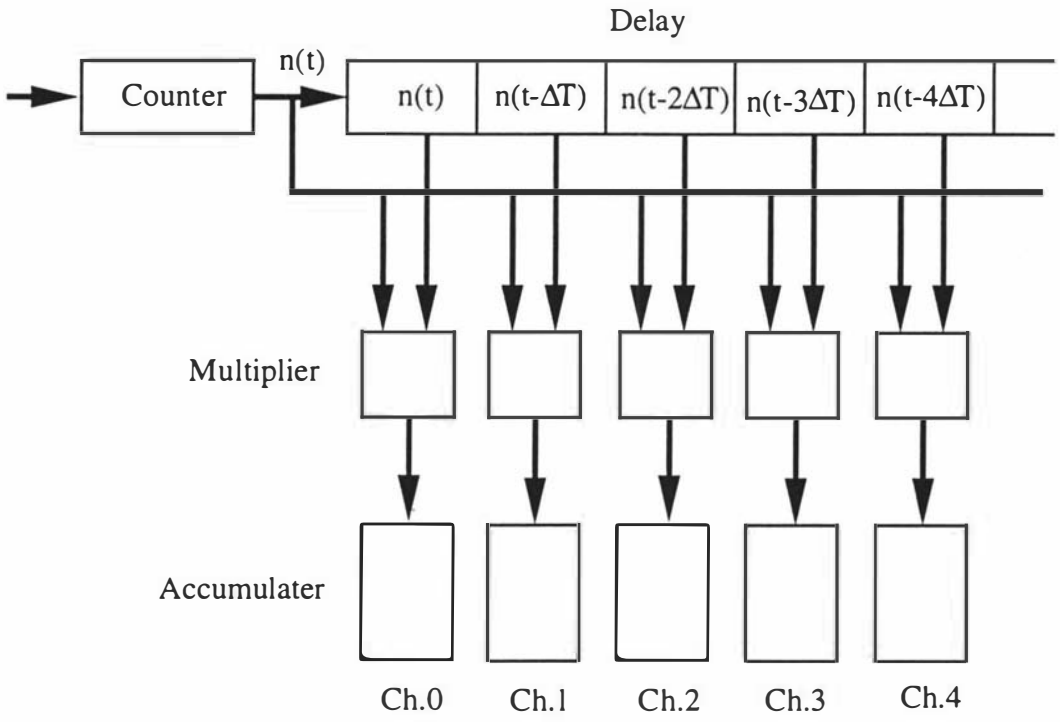
3.2.4 General correlator model

The general model used for the design of correlator hardware is shown in Figure 3.3. A non-zero sampling time for sampling the direct and delayed input samples has to be used, and the traditional correlator design introduces lag times that are separated by one sampling time interval each, thus the lag time for channel i is

$$t_i = i \cdot \Delta T \tag{3.12}$$

where ΔT is the (adjustable) sampling time. To implement the algorithm implied by eqn (3.11) with a correlator, the correlator hardware must perform four basic tasks: (1) count input pulses over sampling intervals spaced on some grid defined by a sample time ΔT; (2) delay these counts for some lag time $k\Delta T$; (3) multiply current input counts

with the delayed counts; (4) accumulate these products; that is, calculate their sum. One accumulation must be performed during each sample time for each channel for real time implementation of the autocorrelation function. Such a correlator may be called a linear correlator, as the lag time increases linearly with the correlation channel number.



Channel 0: $G(0) = \sum n(t)\, n(t)$

Channel 1: $G(1\Delta T) = \sum n(t)\, n(t-\Delta T)$

Channel 2: $G(2\Delta T) = \sum n(t)\, n(t-2\Delta T)$

Channel 3: $G(3\Delta T) = \sum n(t)\, n(t-3\Delta T)$

Channel 4: $G(4\Delta T) = \sum n(t)\, n(t-4\Delta T)$

Figure 3.3 Autocorrelator model

Obviously the lag time range of a linear correlator is limited by the number of correlation channels used. This is not a problem for applications where the correlation function consists of a single exponential. However processes exist which have important spectral information in a lag time range of $1{:}10^{10}$. To obtain reliable time discrete correlation functions for such processes, a linear correlator would need to have at least 10^{10} channels to maintain resolution. Cost and size limitations would make such a device unrealisable with present day electronics.

3.3 Multiple Sample Times Correlation Function

3.3.1 Correlation at multiple sample times

It was noted in chapter 2 that it was the ill-posed nature of the Laplace inversion problem together with a growing interest in samples of strongly interacting Brownian particles which motivated a desire for correlators with logarithmically spaced lag times. Analysis of the problem shows that a correlogram obtained with channels spaced logarithmically in lag time will be able to produce better inversion results (Provencher, 1982), (Schatzel, 1988). Also, channels spaced logarithmically in lag time can cover extremely large lag time ranges in one single experiment. This feature is often necessary for the analysis of strongly interacting colloids such as dispersions at high concentration.

Hence the ideal correlator should provide something like a logarithmic spacing of lag times in order to cover a large lag-time range with a reasonably small number of channels. In this section, we will discuss the possibility of using this multiple sample time correlation function.

Practical problems arise while using a logarithmic time scale correlator. Such a sampling scheme results in some correlation channels with very small sampling times as well as some correlation channels with much larger sampling time, and this may very well lead to distortion of the correlation function. This effect is called triangular distortion.

For photon counting events, a Poisson distribution is expected for the pulse counts n_i with a mean value given by the classical light intensity $I(t)$ integrated over the particular sample time interval i under consideration, as shown in eqn (3.4), which is rewritten here taking $p = 1$

$$\mu_i = \int_{(j-1)\Delta T}^{j\Delta T} I(t)dt \qquad i = 1, 2, 3, \ldots \qquad (3.13)$$

A calculation proves that the expectation value given in eqn (3.7) essentially equals the corresponding intensity correlation function for all non zero k (Schatzel, 1988).

$$G_n(k) = \langle \mu_i \mu_{i-k} \rangle = \int_{-\Delta T}^{\Delta T} \langle I(t) I(k\Delta T + t) \rangle (\Delta T - |t|) dt \qquad (3.14)$$

Eqn (3.14) has two results under two different conditions.

1. Case A, small sample time.

The sample time ΔT is constant and well below the time scale of typical fluctuations in $I(t)$ (that is, the coherence time t_c), then the triangular effect in the final expression of eqn (3.14) may be ignored, and one obtains the relation

$$G_n(k) \approx \Delta T^2 \langle I(0) I(k\Delta t) \rangle \tag{3.15}$$

2. Case B, multiple sample time

The sample time ΔT increases with lag time, so some sample times are comparable with t_c, then the triangular effect as given in eqn (3.14) must be taken into account, which will lead to slight distortions of the measured photon correlation function, as the following shows.

Considering a correlation function with one negative exponential and time constant Γ, the normalised intensity correlation function is

$$g^{(2)}(k) = \frac{\langle I(0) I(k) \rangle}{\langle I(0) \rangle^2} - 1 = \beta e^{-2\Gamma k} \tag{3.16}$$

then the triangular effect in eqn (3.14) will be

$$\begin{aligned} g_n(k) &= \beta \Delta T^{-2} \int_{-\Delta T}^{\Delta T} e^{-2\Gamma(k\Delta T + t)} (\Delta T - |t|) dt \\ &= \beta \Delta T^{-2} (2\Gamma)^{-2} \left[2\cosh(2\Gamma \Delta T) - 2 \right] e^{-2\Gamma k \Delta T} \end{aligned} \tag{3.17}$$

Series expansion for small $\Gamma \Delta T$ yields

$$g_n(k) = \beta e^{-2\Gamma k \Delta T} \left[1 + \tfrac{4}{12} \Gamma^2 \Delta T^2 + O\left(\Gamma^4 \Delta T^4 \right) \right] \tag{3.18}$$

Then the deviation between $g_n(k)$ and $g^{(2)}(k)$ caused by the triangular effect can be calculated as (Schatzel, 1990)

$$g_n(k) - g^{(2)}(k) \approx \left[\frac{(2\Gamma \Delta T)^2}{12} \right] \beta e^{-2\Gamma k \Delta T} \tag{3.19}$$

Which increases with ΔT at small values of the sampling time until it reaches a maximum of

$$\max = \beta e^{-2} (\Gamma \Delta T)^2 / 3 \tag{3.20}$$

at

$$k \Gamma \Delta T = 1 \tag{3.21}$$

then eqn (3.20) becomes

$$\beta/(3e^2k^2) \approx 0.045\beta/k^2 \qquad (3.22)$$

For larger sampling times ΔT the decay of the exponential will quickly reduce the absolute amount of distortion.

Eqn (3.20 - 22) show that the triangular effect can be reduced below given limit by keeping $\Gamma\Delta T$ sufficiently small. For example if we choose $\Gamma\Delta T = 1/8$, this triangular effect is less than 10^{-3} and is certain to be covered by noise in almost any application, which means if $k \geq 8$, then the triangular effect can be neglected.

In conclusion, triangular effect due to a finite sample time leads to a small increase in measured photon correlation data. This increase is a constant factor for a given sample time in the case of a single exponential normalised correlation function. The factor is very close to unity for sample times significantly less than the decay time (for example by a factor of 8). For the more complicated case of multi component correlation functions and/or multiple sample-time data, triangular averaging errors may be kept negligibly small by restricting oneself to the use of lag times considerably larger than the sample time, for example $k\Delta T/\Delta T \geq 8$.

Triangular distortion sets some limit to the accuracy one can obtain using the multiple sample time correlation algorithm. Such distortion is systematic, and does not depend on anything other than the shape of the theoretical correlation function, and the sampling times used. So for a given shape, "triangular effect distortion", can easily be predicted and is typically quite small. If a measured correlation function is obtained with absolute noise contribution much smaller than the triangular distortion (and only under these circumstances does such distortion becomes visible), the correlation function is already precise enough to use it for estimating distortions (as they only depend on shape and sampling time) and account for them. It is always possible to compute the effective contribution of triangular distortion on the correlation function and correct for it. One can directly use this measured correlation function to compute the distortion with high accuracy and simply subtract it from the measured correlation function.

3.3.2 Symmetric normalisation and noise reduction

Standard normalisation includes baseline subtraction and division by the square of the count rate estimator n^2. In a multiple sample time correlator, this normalisation procedure leads to some problems, because the total number of samples M may not be a

very large number at large lag times. The accuracy of the correlation baseline estimate decreases with increasing sampling time.

The total number of samples taken during an experiment must always be a very large number (some 10^6 or more) in order to obtain sufficient averaging over photon-counting noise. Therefore if the sample time ΔT is small, the time displacement between the first correlation channel and another $(t_j - t_1)$ is much smaller than the total duration of the experiment $(10^6 \cdot \Delta T)$. In this case n^2 is a good estimate of the baseline for all correlation channels.

However if the sample time were increased in the same experiment, the number of samples taken might only be of the order of 10^2 for larger sample times, and the total measurement duration would not exceed the largest lag time by orders of magnitude. The time displacement of these sampling times would contribute significantly if n^2 were used for normalisation. This time displacement leads to an increasing variance for the large sampling time region of the normalised correlation function caused by fluctuations of the boundary terms in the correlation estimator (Schatzel, 1988), so particular care is required to reduce this estimator variance during the necessary normalisation of the raw data.

For the multiple sample time correlator system, it is more efficient to use symmetrical normalisation to cancellate all boundary contribution so to reduce estimator noise (Schatzel, 1988). In this normalisation method, a symmetrical baseline estimator $n_{e0}n_{ek}$ is used for the baseline subtraction, where n_{e0} is the standard monitor channel,

$$n_{e0} = \frac{1}{M}\sum_{j=1}^{M} n_j \tag{3.23}$$

and n_{ek} is a monitor channel for an individual correlation channel

$$n_{ek} = \frac{1}{M}\sum_{j=1-k}^{M-k} n_j \tag{3.24}$$

then the normalised autocorrelation function obtained from symmetric normalisation is given by

$$g^{(sym)}(k) = \frac{G(k) - n_{e0}n_{ek}}{n_{e0}n_{ek}} \tag{3.25}$$

Where $g^{(sym)}(k)$ is the symmetrically normalised raw correlation function.

As the sample time increases, there is a proportional increase in the mean number of photons counted per sample time period. This reduces the photon noise contributions to the estimator (Schatzel, 1990). At lag times well beyond the time scale of the fastest intensity fluctuations, the sample time is long enough to provide efficient averaging over these fluctuations, and so reduce the classical intensity noise.

In conclusion, the use of increasing sample times together with the symmetric normalisation scheme can lead to greatly enhanced signal to noise ratio.

3.4 Multiple Tau Techniques

There are several operating principles required to implement a model of an autocorrelator in hardware. The model of a typical linear time scale autocorrelator was given in section 3.2.4.

A non-zero sampling time for sampling the direct and delayed input samples is used in the correlator, and the correlator design simply introduces lag times that are separated by one sampling time interval each. The lag time increases linearly with the correlation channel number, as shown in Figure 3.4.

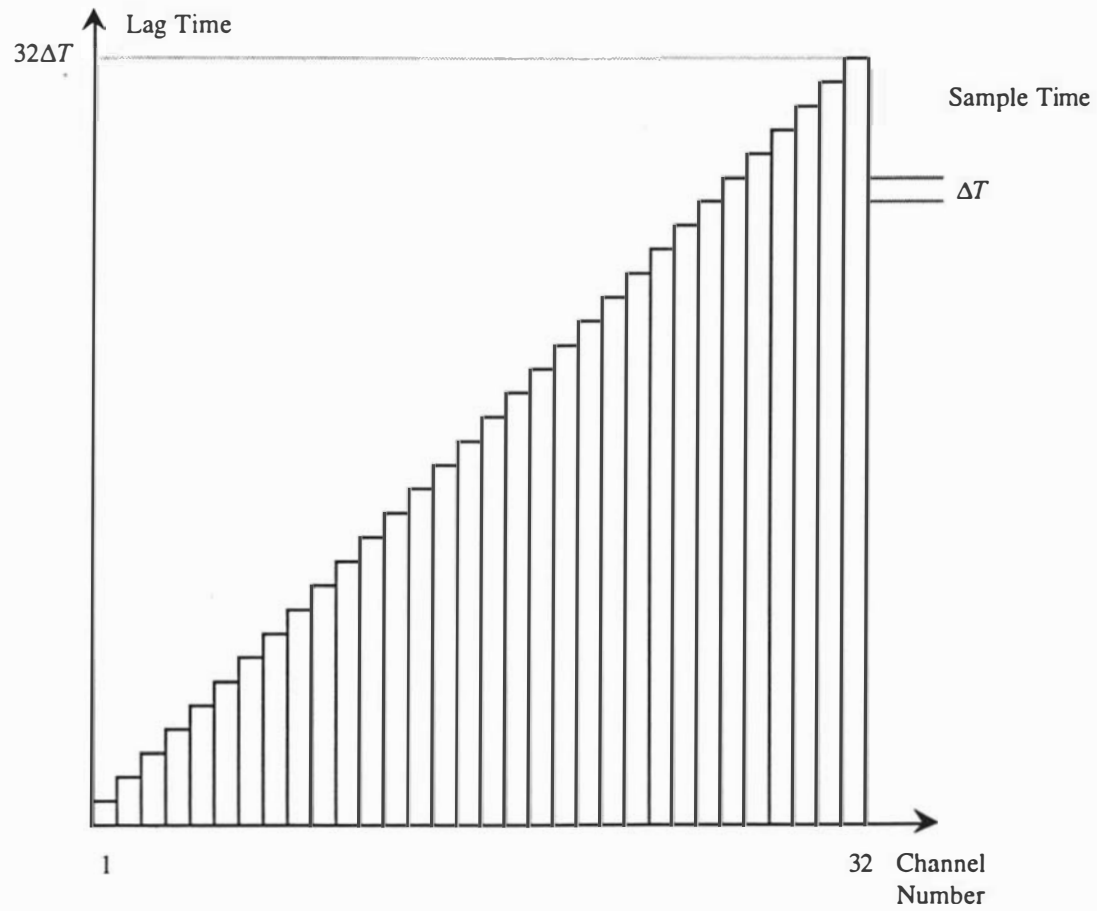


Figure 3.4 Linear time scale, step height is the lag time $k\Delta T$

In a multiple sample time correlator, the sampling time is no longer constant, but rather increases with the lag time. A multiple sample time scale correlator increases the temporal dynamic range of a correlator without increasing the number of correlation channels. This restricts the actual calculation of the sums of products to certain lag times only.

A logarithmic time scale distribution is a natural distribution scheme for these lag times. The lag times follow

$$t_{i+1} = t_i \cdot k \tag{3.26}$$

ie.

$$\begin{aligned} t_1 &= \Delta T \\ t_2 &= k\Delta T \\ t_3 &= k^2\Delta T \\ &\cdots\cdots \\ t_N &= k^{N-1}\Delta T \end{aligned} \tag{3.27}$$

where k is the delay factor and remains constant over the complete lag time range.

The correlator samples with a constant sampling time ΔT at logarithmically increasing lag times t_i, leaving logarithmic increasing gaps of lag times. Within such gaps the correlator does not calculate the correlation function and these gaps can be considered as delay times.

From eqn (3.27), the number of correlation channels needed to cover a given lag time range is now a function of the delay factor and may easily be computed as

$$N - 1 = \frac{\log \frac{\tau_{max}}{\Delta T}}{\log k} \tag{3.28}$$

which leads to values of about 240 correlation channels for a temporal dynamic range of $1{:}10^{10}$ and a delay factor of 1.1.

The major feature of an logarithmic delay correlator is, that not all possible products of direct and delayed samples are calculated. This is an advantage, since while the lag times and hence the number of such possible products increase logarithmically, the number of correlation channels and hence the number of calculated products increases only linearly.

However it would not be wise to double the sampling time for each individual correlation channel, because such a channel structure would become very coarse on the lag time grid (a temporal dynamic range of 10^{10} is reached using just 33 correlation

channels). Also such a correlator simply neglects more and more useful information as the lag time increases. The algorithm covers a vast temporal dynamic range but the temporal resolution is poor.

An algorithm can be found using the Multiple Tau Correlation Technique (Schatzel, 1985). This technique is a combination of the linear and logarithmic models. Many L channel linear correlator are used in series and the lag times between the L channel linear correlators are arranged logarithmically.

The major difference between this technique and the linear or logarithmic delay correlator technique is that, instead of increasing the sample time for each channel, blocks of $L = 8$ autocorrelation channels with constant sampling time are calculated and the sampling time is doubled from one block to the next. This makes the technical realisation of varying lag and sample times much simpler.

This scheme has been termed "multiple tau". The time scale of the ideal logarithmic and the multiple tau are compared in Figure 3.5.

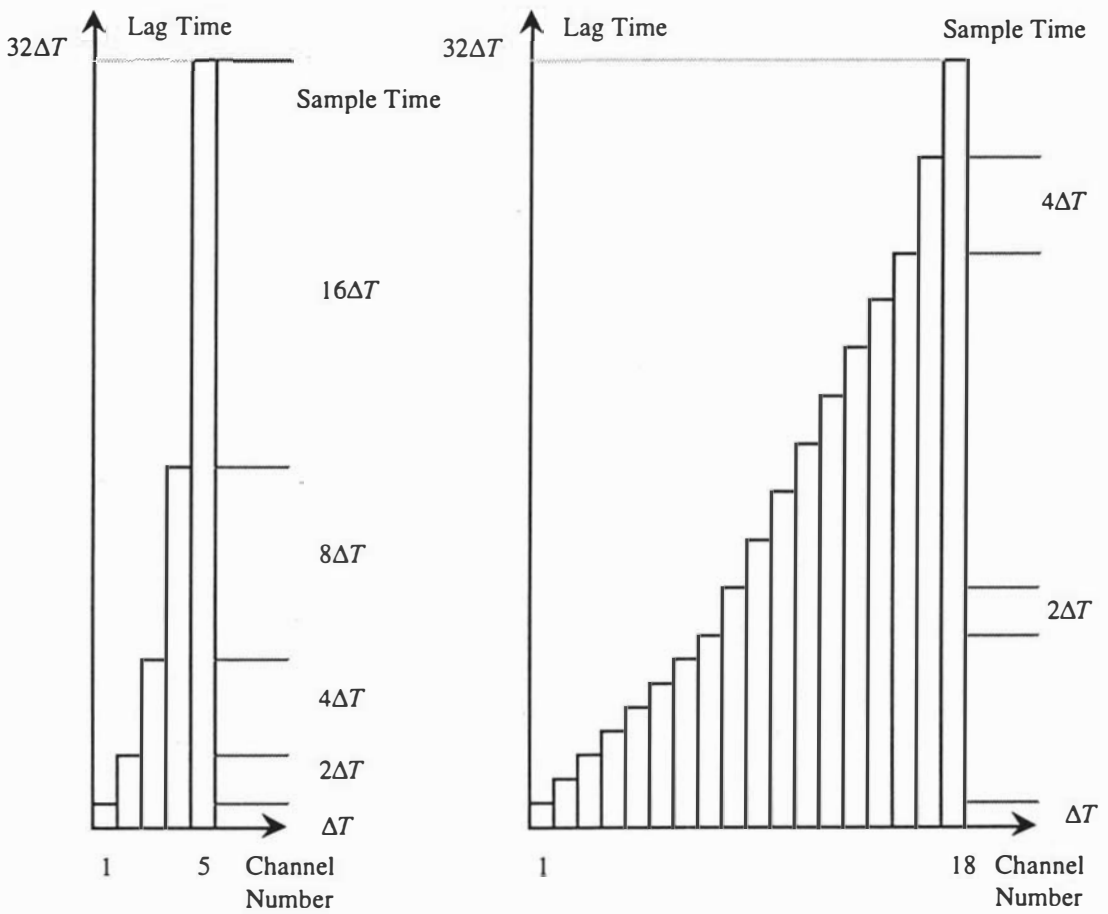


Figure 3.5 Logarithmic time scale and multiple tau time scale

The Multiple Tau Correlation Technique with 8 linear channels per octave therefore reduces to a simple algorithm:

(a) Calculate a number of 8 linear correlation channels with sampling time ΔT.

(b) Increase the sampling time by a factor of two.

(c) Repeat step (a) until the required temporal dynamic range is reached.

The design of the hardware structure of the correlator is based on this algorithm. First, the channel structure can be designed as follows. The smallest sampling time chosen is 3.2 μs and the sampling time increases over the complete lag time range for each block of 8 channels. The 128 correlator channels are divided into 16 blocks, each with 8 linear time scale channels. This means all the channels belonging to a particular block have the same sample time. The samples of a later block are simply obtained by the addition of pairs of samples of the previous block, thereby producing a doubled sample time.

Multiple sample time theory is an active research area, especially in baseline estimation. The hardware monitor structure should be designed to be flexible so the correlator can take advantage of any subsequent developments.

4 CORRELATOR DESIGN II - CORRELATOR STRUCTURE

The design of a real time autocorrelator based on the multiple tau technique is discussed in this chapter. Firstly techniques for implementating the autocorrelation function algorithm will be discussed, starting with a review of basic correlator techniques. This will give an overview of correlator hardware design. After that the use of a multi-DSP (digital signal processor) system to implement the multiple tau autocorrelation function algorithm will be illustrated. Chapter 5 will give the detailed circuit and software designs and descriptions.

4.0 Hardware Correlator Design Techniques Review

4.0.0 Ideal correlator

The model of an ideal photon correlator was given in Figure 3.3. It operates as follows:

(i) A sample is obtained by counting pulses in a given sample period. Every pulse is counted to ensure no information is lost.

(ii) Each sample is stored in some form of memory. These stored samples represent the history of the measured intensity fluctuations obtained from a light scattering experiment.

(iii) A real time autocorrelation function is generated by multiplying the most recent sample with each of the earlier samples and adding the products to the sums of the previously obtained products for each of the different lag times.

Calculation of the ideal correlation function involves a large number of multiplications and additions and represents a significant computational load. There are several possible hardware designs for a real time correlator. Various schemes have been designed in the past which take advantage of specialised circuits to simplify the hardware so that the speed requirement can be satisfied.

4.0.1 Clipping correlator

A correlator based on clipping techniques has been successfully designed and constructed by R.C. O'Driscoll in 1970's in the Department of Physics, Massey University. This technique was developed in 70's and the correlator is still successfully used in DLS experiments (O'Driscoll, 1982). This kind of correlator can work at very short sample times, 50ns, in real time.

There are two techniques used in this kind of correlator. The first involves the clipping of the delayed signal to single-bit accuracy before the correlation function is computed (Oliver, 1974). The output of a clipper is '0' if the number of counts detected in a given sample time ΔT is less than or equal to the clipping level, and '1' if the number of counts detected exceeds the clipping level.

The second technique takes advantage of clipping to simplify multiplication. If n_c is the direct signal (the number of count pulses into the system), and n' is the clipped delayed signal ('1' or '0'), then the required multiplications can be replaced by a series of additions:

$$n_c n' = \sum_{n'} n_c \tag{4.1}$$

These additions are triggered by the input pulses n'.

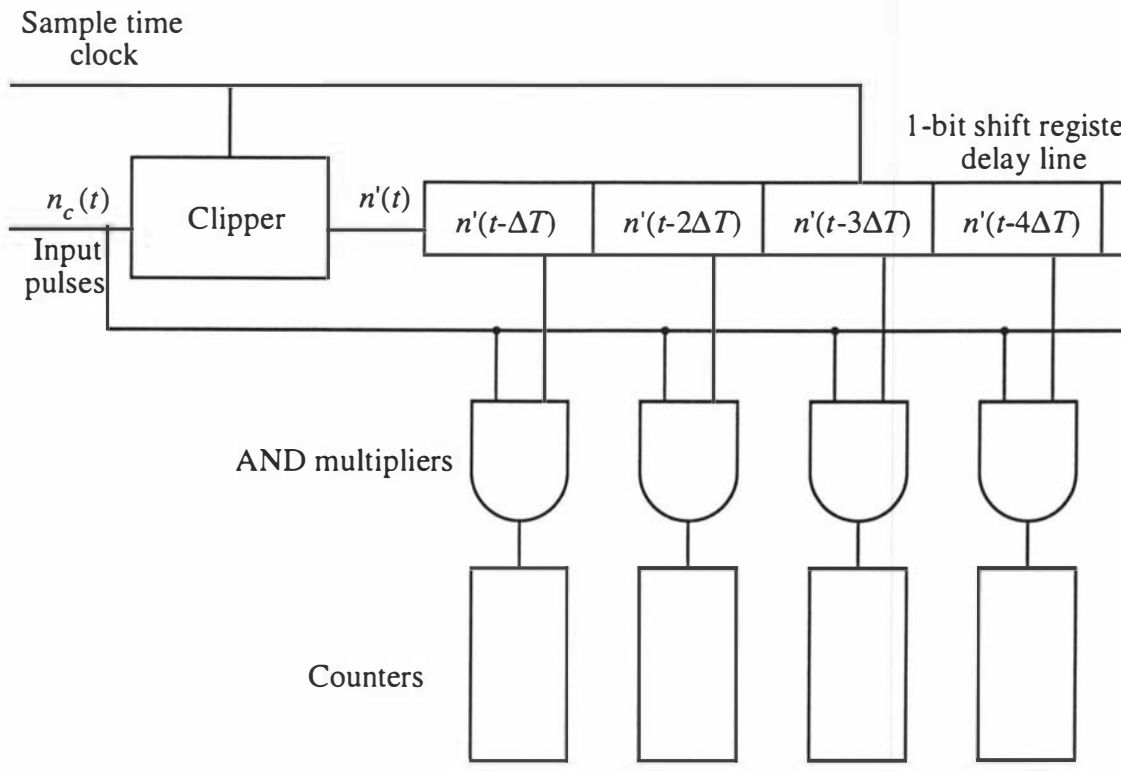


Figure 4.1 Single-clipping correlator

In this kind correlator (Figure 4.1), input pulses are passed through a clipping gate into a single-bit shift register operating under the control of sample time clock. All input pulses are passed in parallel to a bank of AND gates, which is used to gate the contents of the shift register into a bank of counters. In this manner, because of the necessary restriction to single-bit signals in the delayed input only, each counter accumulates one channel of the $1\times N$-bit correlation function. Additional counters are provided for the monitor channels (number of samples; number of input counts; number of clipped

counts). Further circuitry allows for real-time readout for display and storage, selection of a sample time as a multiple of 50 ns, and auxiliary functions such as start and stop, and clearing of the store (bank of counters). The O'Driscoll correlator also included the 'blinker' technique to minimise the effect of dust on the measured intensity correlation function.

The data are counted and sent to the shift register delay line which stores the delayed data. Thus the delayed data can be generated as fast as the direct data for the correlation calculation.

Figure 4.2 shows the data structure in the first 4 channels of the shift register delay line at times t, $t+\Delta T$, $t+2\Delta T$, $t+3\Delta T$.

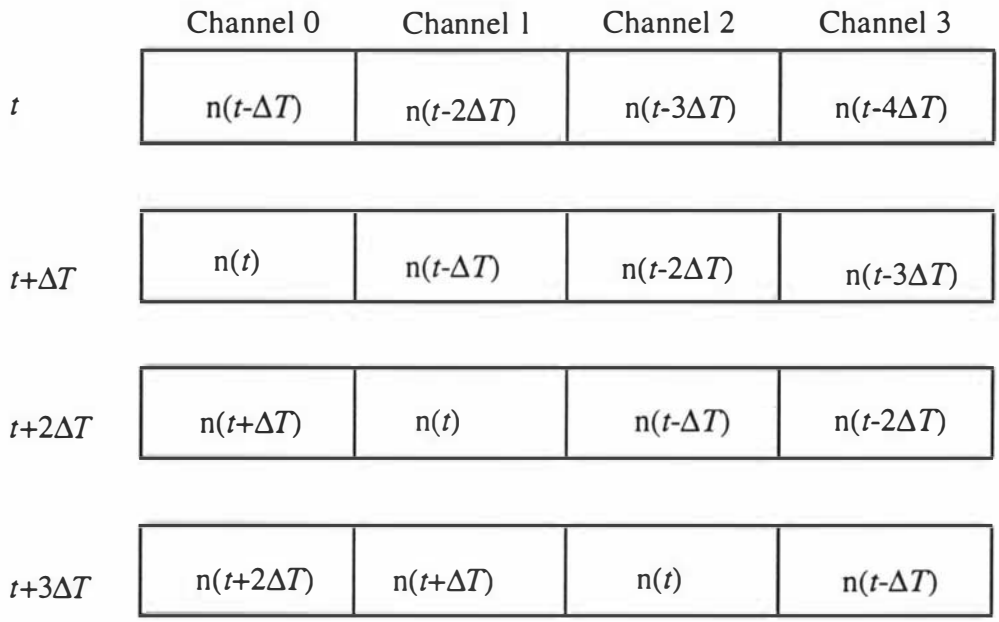


Figure 4.2 Data structure for clipping correlator using a shift register

4.0.2 N×M bit correlator

A second kind of correlator, which employs multi-bit processing, is an N × M-bit correlator. These devices still perform multiplications by repeated additions triggered by the input pulses. An example is a 4 × N-bit correlator (Brookhaven Instruments). The structure is similar to that of the single-bit or 1 × N-bit machines. The clipping gate is replaced by a 4-bit counter, the shift register has a width of four bits, and the simple AND gates are replaced by 4-bit adders with internal accumulators, which feed their overflows into the channel counters. The more complex channel structure results in a decrease in the speed of operation. Microprocessors have been introduced to control

such instruments, organise the display and data storage, and allow for more sophisticated user interaction.

4.0.3 Multiplying correlator

A third kind of correlator is the multiplying correlator (Schatzel, 1985) (ALV Laser Vertriebsgesellschaft mbH). This correlator uses special multiplier /accumulators for the correlation calculation and works on the fixed M × M bit input data format on both the delayed and direct channels.

4.0.4 Microprocessor based correlator

A fourth kind of correlator is the microprocessor based correlator (Thomas, 1983), (Noullez, 1986), (Subrahmanyam, 1987), and (Bruge, 1989). A correlator can be built by interfacing the output of a photon counting detection system to an on-line microprocessor system. These correlators are slow and sample time is limited to the millisecond regime. The software approach allows the computation of either the full correlation function or the power spectrum over as many channels as necessary, within the limitations of the processor memory and real time requirements. They may also be used at smaller sample times, if batch processing is used (this technique will be discussed below in section 4.0.5).

In the microprocessor type correlator, the shift register is replaced by a data storage buffer, normally a fast memory. Normal types of processor will need several tens of machine cycles for the elementary operation of updating the correlation estimate for a single channel. This leads to performance estimates expressed as the time needed for one such elementary operation. The fastest possible sample time for true real-time operation is obtained by multiplication of the time required for one elementary operation by the desired number of channels. As an example, a processor which needs just 1 μs for one elementary operation may compute a real-time correlation function with 100 channels for sample times of 0.1 ms or more.

4.0.5 Batch processing correlator

A fifth kind of correlator is the batch processing correlator (Noullez, 1986). This is used when the hardware design problems are overwhelming and real time processing is abandoned. In this mode, data may be sampled continuously (as in real-time processing), but in rather short 'batches' only. Then the sampling stops and the data are processed. After completion of processing, the device samples the next batch, etc. This approach increases the total duration of the measurement by approximately a factor of

processing time divided by sampling time. The inverse of this factor is known as the duty cycle and gives a straightforward measure of performance reduction.

This batch correlation function can then be computed as follows: If R is the ratio of the time interval between two accumulations to the sampling time

$$R = \frac{N\Delta T + T_c}{\Delta T} \tag{4.2}$$

where T_c is the computation time for one batch of data, N the total number of accumulations, and ΔT is the sampling time, then

$$G(j) = \sum_{i=1}^{N} n(iR\Delta T)n(iR\Delta T - j\Delta T) \tag{4.3}$$

Thus the batch mode correlator has a signal utilisation efficiency of $1/R$ and should be used only when short sampling times are required and under conditions of high stability of the sample, longer accumulation times are necessary in this mode.

The major time consuming operation in the batch correlator is the $N\Delta T$ delay time before each accumulation. Of course most correlation data could be processed using a batch system. However there is a requirement for much faster responses to input data than is available from batch systems, hence the need for real time systems.

4.1 The Techniques Used to Design the Multiple Tau Correlator

Each of the above well established hardware techniques for photon correlation has their own advantages and disadvantages. This section will discuss the appropriate electronic techniques that can be used to build a multiple tau correlator to meet the design requirements, and the reasons for the choices made will be given.

Our purpose is to design a multiple tau correlator to be as near ideal as possible using available electronics techniques. The autocorrelator must operate in real time at all sampling and lag times. Large input data words should be processed with full multiplication procedures for M × M bit correlation. Built in symmetric normalisation procedures must be available with individual channel monitors and a block monitor for each sampling time block to ensure optimum statistical accuracy.

The design described here is combination of the multiplier correlator technique and the software correlator technique using a parallel processing multi-DSP system. There are several reasons for this.

(1) Utilisation of the clipping technique provides a very fast and efficient way to design a correlator. But a multiple tau correlator using clipping would require different clipping levels at each sample time, and the splicing together of different sample time measurements could become a formidable task.

(2) A digital correlator works on input data with only finite accuracy, determined by the maximum input data width, and care must be taken to ensure that minimum quantisation noise occurs for the larger lag times (those lag times where even the M bit format overflows and preprocessing of the input data has to be introduced). The design of a multiple tau system, using different accuracy in the direct and the delayed inputs (N x M bit) is very difficult, and no simple way exists of breaking the required operations into subsequent steps which could be triggered by single input pulses.

(3) Microprocessor based correlators are unable to operate at sufficiently short sample times. General purpose microprocessors generally examine the data and make decisions on the type of operation required, using a host of conditional codes with alternatives and branches. However correlation algorithms and other digital signal processing algorithms are more directed, often with only one possible direction between input and output data flow. Hence much of the power of the general purpose microprocessor is not required. Also the lack of a fast multiplier/accumulator makes the correlation calculation too slow. In short, normal microprocessor based autocorrelators have been inapplicable or too expensive for real time application in DLS because the very short sample times required are not achievable.

(4) Multiplier correlators designed with special hardware architectures tend to be for special purposes only. Dedicated digital hardware circuitry may indeed meet the demands of high computational rates, but at a cost of hardware complexity and loss of flexibility. A system based on such devices could not easily be altered to implement improved correlation algorithms. A preferred system would be able to be easily reprogrammed. The current trend is to use more "off-the-shelf" programmable components, such as digital signal processors, to produce more generic architectures.

(5) A DSP is a microprocessor dedicated to the rapid execution of complex algorithms in real time (Mosely, 1990). DSPs typically feature high-speed multiply and accumulate hardware that facilitates iterative algorithmic processing by enabling the DSP to perform signal-instruction-cycle multiplication and accumulation. DSPs use 16-, 24-, or 32-bit words and are designed for either fixed-point or floating-point arithmetic.

The key differences between normal microprocessors and programmable digital signal processors result mainly from design decisions aimed at real-time computing. Many DSPs function as embedded real-time processors in special-purpose hardware.

Since DSPs are just fast microprocessors with specialised instruction sets, they enjoy all the advantages of microprocessors, ie. they are easy to use, flexible, and economical. Also the use of DSPs has, for the most part, traded the original hardware system problem for a programming problem within a fixed structure.

The advantages of using DSPs are becoming more compelling as they become faster and more cost effective. Modern DSPs can deliver up to 30 MIPS (million instructions per second) and 60 MFLOPS (million floating operation per second) of computing power. In addition to large address spaces, these chips feature multiple register files, general addressing modes, bit-manipulation instructions, and integrated on-chip peripherals.

Despite this speed and processing power, a single DSP cannot provide the processing needs for a multiple tau correlator system. This problem can be solved by using a multi-DSP system.

(6) Currently many real time system architectures consist of multiprocessors, or networks of uniprocessors, or networks of both uniprocessors and multiprocessors (Bond, 1987), (Wilson, 1989), (Fukuda, 1991), and (Agnello, 1992). These multiprocessor architectures have been acknowledged as the cost effective alternative to custom implementations for handling the enormous amount of computation in digital signal processing. Also most modern DSPs are designed to be able to support this kind of multiprocessor or parallel application. So increasingly, these real time DSP systems can offer multiprocessor solutions to immensely enhance processing power.

A correlator built with multiple DSPs is a multi-purpose instrument. Different normalisation methods and different working modes for the correlator can be implemented. Moreover the instrument can be made to operate as a spectrum analyser, or other digital signal processing unit just by changing the software.

Also the 'full' correlation function can be formed. DSPs are capable of 8-bit, 16-bit or even 24-bit resolution of the correlation function. Full correlation will give a significant increase in signal to noise ratio.

The above considerations governed the selection of the hardware system for the correlator. Hence the major topic of this thesis could be summarised as the use of real

time multiple DSP techniques to design and construct a multiple tau autocorrelator. The first step is the selection of a suitable DSP for the system.

4.2 Selection of A Suitable Digital Signal Processor

4.2.1 DSP families

DSPs began to appear approximately 14 years ago, with the AT&T Bell Laboratories DSP1 and the NEC μPD7720 leading the way, followed by the Texas Instruments TMS32010 in 1982.

DSP performance has improved dramatically since 1982. Devices with clock "speeds" of over 60 MHz are now in production, and "speeds" are expected to increase to as much as 100 MHz over the next few years. Also while speeds were increasing fourfold, data paths and address space have moved from eight to 32 bits.

The most interesting recent development is the emergence of DSP cores. These cores are programmable DSPs, surrounded by customer-specific circuitry. If customer design support can be delivered effectively, these devices will provide formidable competition for ASICs (application specific integrated circuits).

Table 4.1 is a review of some of DSPs currently available.

Texas Instruments	Motorola	Analog Devices
TMS320C40: 40/50 MHz floating point DSP; extensive parallel processing support through 6 buffered byte-wide 20 Mb/s links and 6 channel DMA.	**DSP56001**: 20/27/33 MHz fixed point DSP. 24 bit data bus, 16 bit address bus, 56 bit accumulators (2), host interface port, serial ports (2), general purpose I/O pins. 512 words program RAM, 512 words data RAM on chip.	**DSP21020**: 20/25/33 MHz floating-point DSP; Supports 32-bit fixed point, IEEE format 32-bit floating point, and 40-bit floating point; 40-bit registers plus two 80-bit fixed-point multiply-accumulators; 32 word instruction cache allows two data accesses in a single cycle.
TMS320C50: Enhanced TMS320C25; low overhead looping; 10 Kwords SRAM on chip.	**DSP56000**: Mask programmed version of DSP56001.	**ADSP21010**: Slower and cheaper version of '020 (16 MHz). Limited to 32-bit fixed and floating point.
	DSP56002: DSP56001 with On-Chip Emulation (OnCE) debug port and clock PLL. Also has a four cycle double precision multiply and support for block floating point. Available up to 40 MHz.	
	DSP96002: Floating point DSP; 32 bit data and address bus, two complete external buses; Designed for multi-processing. Has a bus oriented, tightly coupled architecture. It has five on-chip buses and bus-arbitration logics, plus instruction cache and data memories.	

Table 4.1 DSP families

4.2.2 Why the DSP56001 was chosen

The choice of a DSP is an important step in system design. The design must be carried out with what is available and feasible at the time, especially in the rapidly developing field of processing electronics. There are two criteria affecting the choice of DSP. First,

of course, is the capability of the DSP itself. This decision is based on the performance of the DSP and ease of application of both hardware and software. The second is the availability of a DSP application development system and application support from the manufacturer. The DSP must have a suitable software development environment. The lack of adequate development support for the programmer can render even the most sophisticated and powerful hardware useless.

For this application a Motorola 24-bit DSP56001 was the ideal choice for the following reasons. First it provides 24 bits of intermediate storage for each correlator channel and this 24-bit capacity is sufficient to ensure that a reasonable input count rate can be accepted without causing the counters to overflow. Second this DSP has a very good support system development kit. Third this DSP's design structure makes it easy to use and program if the developer is familiar with the Motorola 68000 microprocessor family. Fourth it was one of the fastest DSPs on the market at the time the project was started and it was the most cost effective DSP on the market at that time. Fifth the DSP56001 provides on-chip parallel interfaces interfacing with a host processor to support parallel processing using multiple DSPs.

The latest version of the DSP56001 (33MHz) was used.

The correlator was developed using the Motorola ADS56000 development kit. The DSP56000 application development system (ADS) is a three component system which acts as a development tool for designing real time signal processing systems. The basic philosophy of the ADS is to incorporate all complex user interaction into a low cost workstation environment with a well supported operating system. This significantly decreases the overall hardware complexity and cost yet increases the capabilities of the system. The three components consist of an application development module (ADM) which contains a DSP56001 processor and control circuitry, a computer interface board, and a software program which interacts with the user and controls the ADM, as indicated in Figure 4.3.

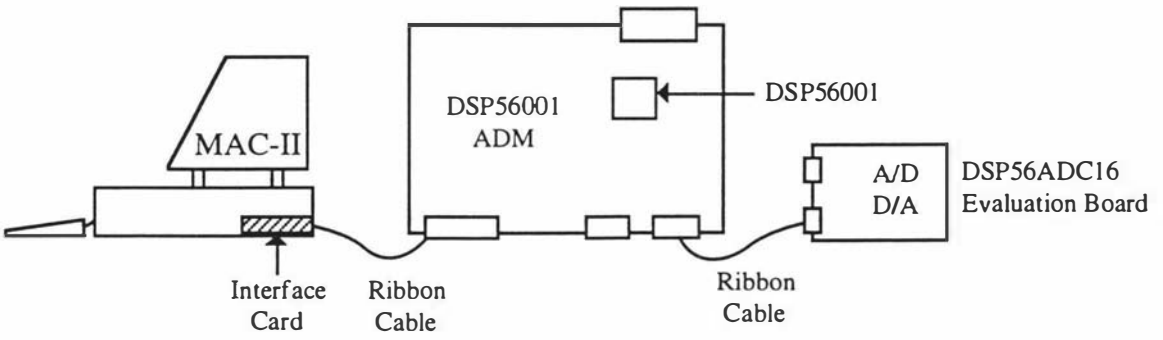


Figure 4.3 DSP56001 application system

The DSP56000CLASX, design-in software package, is a recommended companion product for the ADS. This software package runs on a Macintosh II computer, and includes the DSP56001 simulator program and the DSP56001 macro cross assembler program. All command entry occurs from a fixed command line on the computer screen. A fixed error line is used to flag any errors in the command line entry.

Although "C" compilers are available for the DSP56001, they were not used because the efficiency of the compiled code is usually low due to the complexity of the architecture, especially in multiple tau correlation calculations.

4.2.3 The DSP56001 Digital Signal Processor

The basic features of the Motorola DSP56001 digital signal processor will be introduced first. Then the software and hardware performance of the DSP will be explained. Most of the features introduced in this section will be used in later chapters concerning correlator circuit design.

The DSP56001 is the first member of Motorola's family of HCMOS, low-power, general purpose Digital Signal Processors. The DSP56001 features 512 words of full speed, on-chip program RAM (PRAM) memory, two 256 word data RAMs, two preprogrammed data ROMs, and special on-chip bootstrap hardware to permit convenient loading of user programs into the program RAM. The core of the processor consists of three execution units operating in parallel, the data arithmetic logic unit (ALU), the address generation unit, and the program controller. The DSP56001 has multiplier/accumulator (MCU) style on-chip peripherals, program and data memory, as well as a memory expansion port.

The high throughput of the DSP56001 makes it well-suited for communication, high-speed control, numeric processing, computer applications, and audio applications. The main features facilitating this throughput are the following:

Speed	At 16.6 million instructions per second (MIPS) with a 33.3MHz clock, the DSP56001 can execute a 1024 point complex Fast Fourier Transform in 1.96 milliseconds.
Precision	The data paths are 24 bits wide, providing 144 dB of dynamic range; intermediate results held in the 56-bit accumulators can range over 336 dB.
Parallelism	Each on-chip execution unit, memory, and peripheral operates independently and in parallel with the other units through a sophisticated bus system. The data ALU (arithmetic logic unit), AGU (address generation unit), and program controller operate in parallel so that an instruction prefetch, a 24-bit x 24-bit multiplication, a 56-bit addition, two data moves, and two address-pointer updates using one of three types of arithmetic (linear, module, or reverse-carry) can be executed in a single instruction cycle. This parallelism allows a four-coefficient IIR filter section to be executed in only four cycles, the theoretical minimum for single-multiplier architecture. At the same time, the two serial controllers can send and receive full-duplex data, and the host port can send or receive simplex data.
Integration	In addition to the three independent execution units, the DSP56001 has six on-chip memories, three on-chip MCU-style peripherals (serial communication interface (SCI), synchronous serial interface (SSI), and host interface), a clock generator, and seven buses (three address and four data), making the overall system low cost, low power, and compact.
Invisible Pipeline	The three-stage instruction pipeline is essentially invisible to the programmer, allowing straightforward program development in either assembly language or a high-level language such as a full C.
Instruction Set	The 62 instruction mnemonics are MCU-like, making the transition from programming microprocessors to programming the DSP56001 as easy as possible. The orthogonal syntax supports the parallel execution units. The hardware DO loop instruction and the repeat (REP) instruction make writing straightline code obsolete.
DSP56001 memory	The DSP56001 has following features: 512-word×24-bit, on-chip program RAM instead of 3.75K program ROM 32-word×24-bit bootstrap ROM for loading the program RAM from either a byte wide, memory-mapped ROM or via the host interface -On-chip X and Y data ROMs preprogrammed as positive Mu-law and A-law to linear expansion tables and a full, four-quadrant sine-wave table, respectively
Low Power	As a CMOS component, the DSP56001 has inherently very low power consumption.

Table 4.2 DSP56001 main features (Motorola, 1991a)

Figure 4.4 shows how the multiplier/accumulator, memories, and program controller are configured in the DSP56001. Three independent memories and memory buses are used to move two operands to the multiplier/accumulator while concurrently fetching a program instruction. The address generation unit (AGU) is divided into two arithmetic units used to independently control the X and Y memories and feed operands to the multiplier/accumulator.

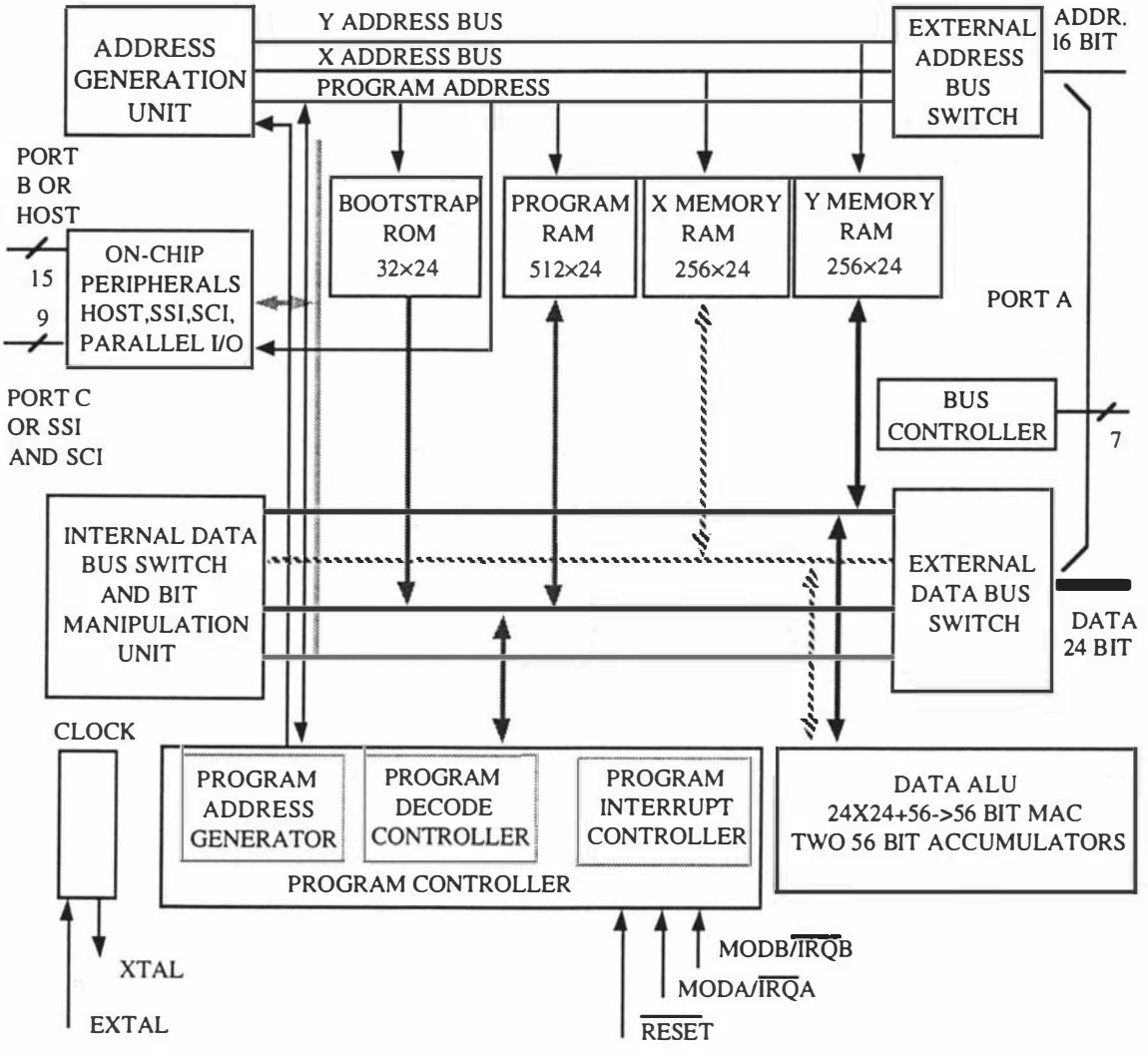


Figure 4.4 DSP56001 Block diagram (Motorola, 1991a)

An additional block labelled I/O is also shown in Figure 4.4. Many DSPs need additional parts to interface with their input and output circuits (such as A/D converters, D/A converters, or host processors). The DSP56001 provides on-chip serial and parallel interfaces to simplify this connection problem. Figure 4.4 also shows the blocks with their interconnecting buses. The DSP56000 Family of processors has a dual Harvard architecture optimised for multiply/accumulate operations.

Its input and output signals are organised into several functional groups, as shown in Figure 4.5. They are:

1. Port A Address and Data Buses
2. Port A Bus Control
3. Interrupt and Mode Control
4. Power and Clock
5. Host Interface or Port B/O
6. Serial Communications Interface or Port C I/O
7. Synchronous Serial Interface or Port C I/O

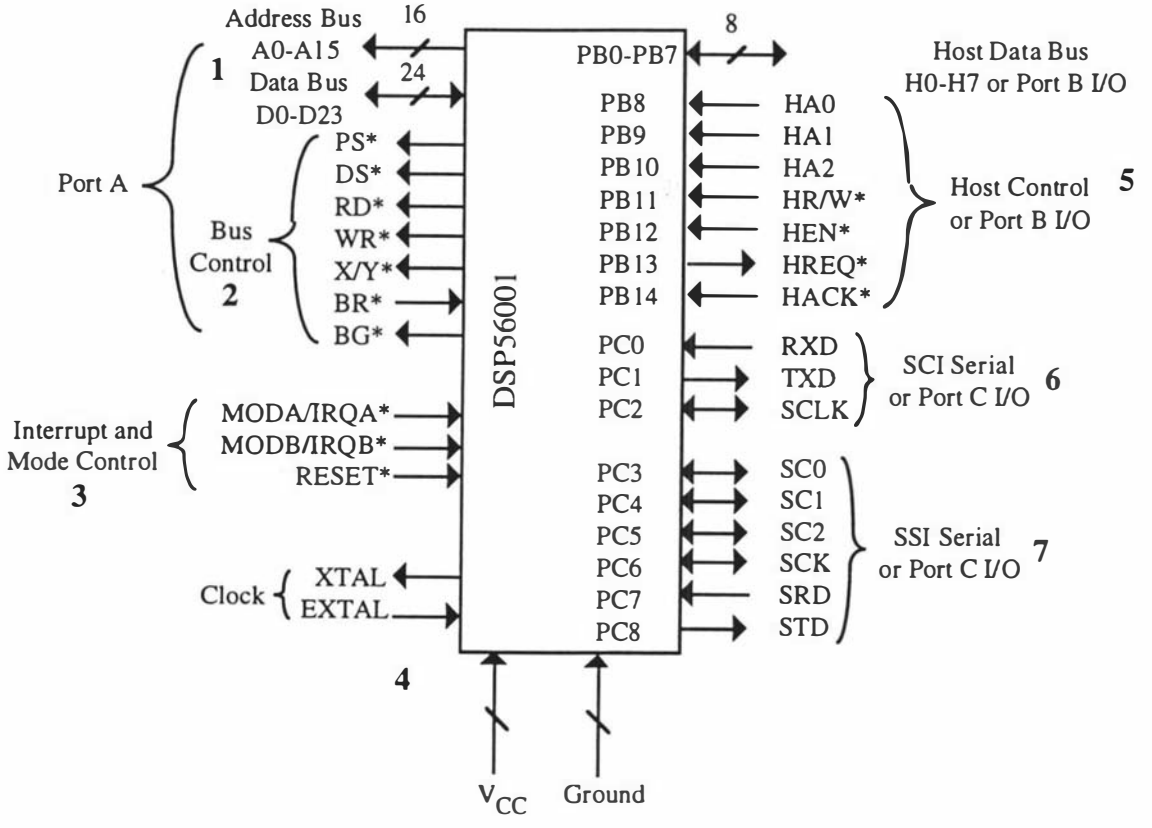


Figure 4.5 DSP56001 functional groups

Table 4.3 shows the brief signal descriptions of DSP56001.

Port A Address and Data Bus
Address Bus (A0-A15) These three-state output pins specify the address for external program and data memory accesses. To minimise power dissipation, A0-A15 do not change state when external memory spaces are not being accessed. Data Bus (D0-D23) These pins provide the bidirectional data bus for external program and data memory accesses. D0-D23 are in the high-impedance state when the bus grant is asserted.
Port A Bus Control
Program Memory Select (PS*). This three-state output is asserted only when external program memory is referenced. Data Memory Select (DS*) This three-state output is asserted only when external data memory is referenced. X/Y Select (X/Y*) This three-state output selects which external data memory space (X or Y) is referenced by data memory select (DS*). Read Enable (RD*) Write Enable (WR*) Bus Request (BR*/WT*) Bus Grant (BG*/BS*)
Interrupt and Mode Control
Mode Select A/External Interrupt Request A (MODA/IRQA*) Mode Select B/External Interrupt Request B (MODB/IRQB*) Reset (RESET*)
Power and Clock
Power (Vcc), Ground (GND) Crystal Output (XTAL)
Host Interface
Host Data Bus (H0-H7) Host Address (HA0-HA2) Host Enable (HEN*) Host Request (HREQ*) Host Acknowledge (HACK*)
Serial Communications Interface (SCI)
Receive Data (RXD) Transmit Data (TXD) SCI Serial Clock (SCLK)
Synchronous Serial Interface (SSI)
Serial Control Zero (SC0) Serial Control One (SC1) Serial Control Two (SC2) SSI Serial Clock (SCK) SSI Receive Data (SRD) SSI Transmit Data (STD)

Table 4.3 DSP56001 pins function (Motorola, 1991a)

The DSP56001 instruction set has been designed to be as orthogonal as possible to allow flexible, independent, concurrent control of the data ALU, address generation unit, and program control execution units during each instruction cycle. this maximizes throughput and minimizes program storage requirements. The instruction set execution time is minimized by the hardware looping capabilities, use of an instruction pipeline, and parallel moves. The instruction set can be divided into the following groups:

1, Move
2, Arithmetic
3, Logical
4, Bit manipulation
5, Loop
6, Program control

Detailed information on each instruction is given in the reference Motorola, 1991.

The complete range of instruction capabilities combined with the flexible addressing modes used in this processor provide a very powerful assembly language for implementing digital signal processing algorithms.

4.2.4 Correlation and the DSP56001

The multiply/accumulate is the basic operation used in the computation of a correlation function. The internal structure of a DSP56001 allows it to perform two moves, a multiply and an accumulate, in a single operation. For correlation calculations, a DSP56001 can capture the present data and delayed data, and perform a multiply/accumulate operation in a single instruction cycle.

To calculate a correlation function in real time, the most recent data and delayed data for different channels must be stored before the most recent data is up-dated. A circular buffer is used to store the data. The address pointers of the DSP56001 are used to select the data. During each sample time, the oldest data are replaced by present sample. Thus the circular list always contains fresh information, and a new calculation can start at the next clock pulse after the end of the current calculation. The data structure for a correlator using a circular list is shown in Figure 4.6.

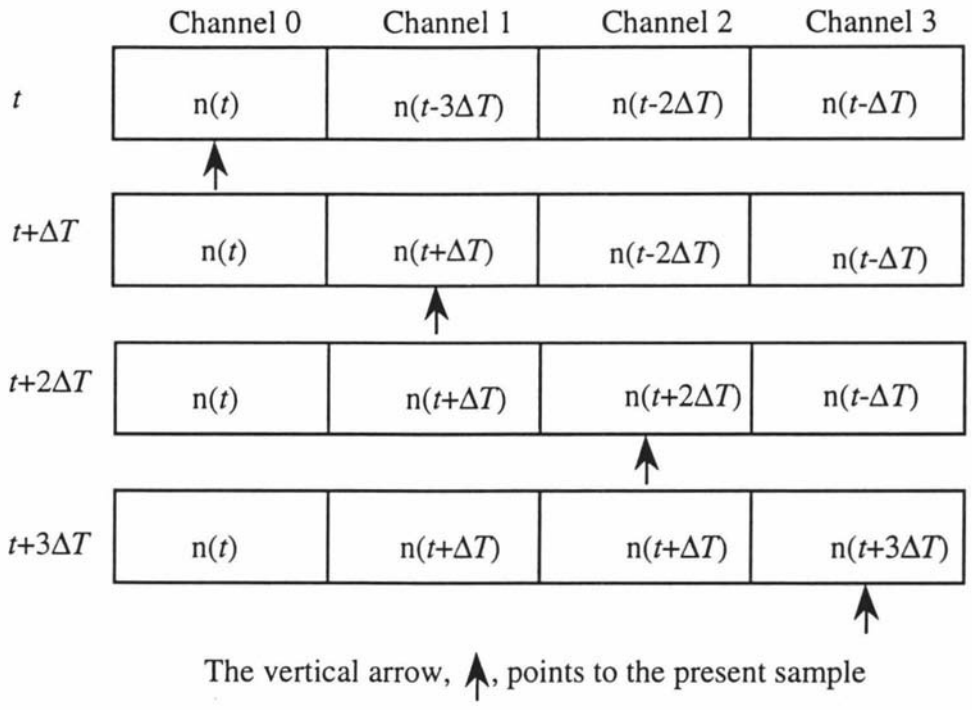


Figure 4.6 Data structure for correlator using a circular list

Based on this structure, the correlation calculation can be very efficient and fast, and the implementation of the correlation algorithm with the DSP is strictly a matter of software.

4.3 Multi - DSP Correlator System

In considering the type of multiprocessor system best suited to a multiple tau time scale autocorrelator task, the two basic considerations are the time schedule requirements for the system and the advantages and disadvantages of the different architectural configurations. The discussion of these systems will focus on the interconnection networks and the programming of the multiple tau autocorrelation algorithms. Also careful attention needs to be paid to the I/O system in a real time environment.

In this section, the basic concepts of multiprocessors and of parallel processing will be introduced and the hardware and software structures which are popularly used in multiprocessor systems will be discussed. The correlator design methodology and task scheduling will be presented. A generic multi-DSP system architecture will be presented and the key resource parameters associated with the multiprocessor system design will be discussed. Then the performance parameters influencing the design will be presented. Finally, a detailed implementation will be derived for the autocorrelator system in order to illustrate the use of the technology described. Implementation considerations in hardware and software will be discussed.

4.3.1 Multi-DSP correlator system design considerations

Figure 4.7 shows a general DSP system for performing digital correlation computation using a single or multiple DSPs.

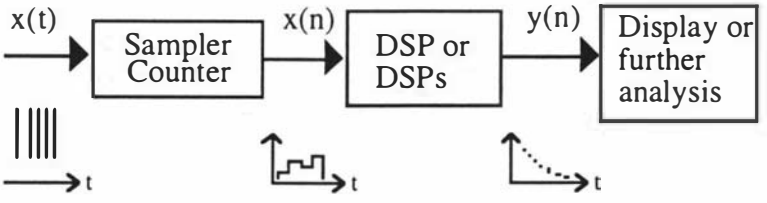


Figure 4.7 General DSP system.

The following specification must be met by the design of the multi-DSP correlator.

(1) The system must have high speed I/O including the capability of handling simultaneous multiple inputs from a multiple sample time counter system.

(2) The system should be versatile, to allow implementation of different normalisation procedures as desired (see section 3.3.2). Also the development of the system should allow the same system to function in a wide range of different applications to make it a multi-function instrument. The architecture of the multi-DSP system should have a

flexible bus structure. The optimised interconnection structure for one algorithm may not be appropriate for another algorithm.

(3) The behaviour of the real time correlator system must be predictable. That is, it should be possible to show at the design stage that all the timing constraints of the application will be met. Since the need for predicability has significant impact on the design of real-time systems, it should be carefully studied at the design stage.

(4) The correlator architecture should be easily expanded to allow for an increase in the number of processing units within the computing resource as additional funds become available.

(5) The system should be able to be controlled by a personal computer such as a Macintosh II or an IBM 486. The system should have a task allocation mechanism to allow the computer to easily distribute the program code to the processors in the system.

The execution time of a multi-DSP system is dependent on the system hardware, the software compiler, and the programming algorithm. There are many hardware features that speed up the execution time. For example, wide bandwidth data communication buses, parallel processing, and fast RAMs will lead to highly efficient hardware behaviour. Similarly, compiler optimisations can contribute to efficient code execution times. Additional good behaviour will depend on the solutions to the complicated problems in the system interfaces due to interrupt handling, and shared data references. The discussion about how system architectures affect the digital correlator execution time and how the multiple tau digital correlation function algorithm is implemented will be presented in the following sections.

4.3.2 Scheduling

One of the primary requirements of a multi-DSP system is that it must have a task allocation analysis to allocate data and tasks among the DSPs (Fukuda, 1991), (Liao, 1994), and (Malloy, 1994). Given the number of processing resources in the system, the task assignment and scheduling must be done to determine where and when each task will be executed so that the multiple tau autocorrelation function can be calculated in real time. This also means the tasks and data are scheduled onto the DSPs in such a manner that the timing behaviour of the system in understandable, predictable and maintainable.

Consider the design of a real-time multiple tau autocorrelator with 128 channels, with the channel structure and timing constraints as shown in Table 4.4.

Channel 1 ... 128, Format: 24 x 24 bit

Channel structure:	Lag times:
8 ch. @ 3.2 µs	8 lags 3.2 ... 25.6 µs
8 ch. @ 6.4 µs	8 lags 32.0 ... 83.2 µs
8 ch. @ 12.8 µs	8 lags 96.0 ... 198.4 µs
8 ch. @ 25.6 µs	8 lags 224.0 ... 384.0 µs
8 ch. @ 51.2 µs	8 lags 435.2 ... 793.6 µs
8 ch. @ 102.4 µs	8 lags 896.0 ... 1.613 ms
8 ch. @ 204.8 µs	8 lags 1.818 ... 3.251 ms
8 ch. @ 409.6 µs	8 lags 3.661 ... 6.528 ms
8 ch. @ 819.2 µs	8 lags 7.347 ... 13.08 ms
8 ch. @ 1.638 ms	8 lags 14.72 ... 26.19 ms
8 ch. @ 3.277 ms	8 lags 29.46 ... 52.40 ms
8 ch. @ 6.554 ms	8 lags 58.96 ... 104.8 ms
8 ch. @ 13.11 ms	8 lags 117.9 ... 209.7 ms
8 ch. @ 26.21 ms	8 lags 235.9 ... 419.4 ms
8 ch. @ 52.43 ms	8 lags 471.8 ... 838.8 ms
8 ch. @ 104.9 ms	8 lags 934.7 ... 1678 ms

Table 4.4 Multiple tau correlator channel structure and lag times

The total number of tasks required to realise the above channel structure and lag times at all times, as well as the computation and resource requirements of all these tasks must be known. It is unlikely that all this information is available at the design stage, and it is not possible to predict which task will meet all its constraints, so worst case values are assumed to predicate the processing ability to meet the proposed multi tau correlator channel structure.

Now consider the proposed 128 channel real-time multiple tau autocorrelator with channel structure, lag times and the shortest practical sample time ΔT_0, as shown in Table 4.4. To achieve real time processing, the multiplication/accumulation calculation for the new input data in all the multiple tau channels must be completed within the shortest sample time ΔT_0. If the calculation time for 8 linear channels is T, then the time taken to calculate the 128 channels is 16T in a single processor unit system in which the processing device takes time T to calculate 8 linear channels. Hence the shortest possible sample time ΔT_0 must be greater than $16T$.

But for a large sample time channel, say $10\Delta T_0$, processing will not take more than a single ΔT_0 cycle out of the $10\Delta T_0$ available. This means that most of the internal clock intervals do not receive an input pulse and no computations are actually performed in these larger sample time intervals. The fast and expensive hardware is idle for more than 90% of the total measurement time.

However, a more complex calculation layout allows a full M × M bit operation to be completed for all 128 multiple tau channels within a time $2T$. Thus much higher input count rates can be accommodated if the system schedule is designed so that a particular cycle can be chosen for processing. Using known idle intervals is then possible to calculate "more channels", ie. more correlator data at larger lag times.

The above idea can be further extended by assigning processing tasks to four processors with the same processing speed. The number of parallel processors required depends on the amount of data that is to be processed in a unit of time. The assignment of tasks to four DSPs with this given channel timing structure is dependent on the number of DSPs used, the type of message-passing used by the system and on the interconnection network of the system.

The schedule problem can be simplified by using a parallel block processing scheme. This scheme achieves the parallelism needed for multiprocessing by dividing the channels into blocks, and assigning a block of channels to each processor. The inputs of the DSPs are connected to the counter array in parallel. The parallel connection equally distributes the different sample time data to the corresponding DSPs, and thereby reduces the I/O overhead in each DSP. A DSP in the system performs all the operations needed for the processing of the given block, and the intermediate results normally do not flow from one DSP to another DSP. Only the completed correlation information is sent to the master or host DSP.

The result of this distributed scheme is that the processor working on the first 8 channels determines the shortest sample time that can be used.

Table 4.5 shows a possible schedule for this system with four processors. The correlation calculations are assigned in parallel to four DSPs. This schedule satisfies the precedence and timing constraints of all tasks.

Autocorrelator Channel Structure:

DSP1 counter sample time = 3.2 μs
Channel Sample time --- Block 1, 8 channels (1 to 8) @3.2 μs, 8 Linear Time Lags
DSP2 counter sample time = 6.4 μs
Channel Sample time --- Block 2 - 3, 16 channels (9 to 24) @6.4 - @12.8 μs, 16 Multi-Time Lags
DSP3 counter sample time = 25.6 μs
Channel Sample time --- Block 4 - 8, 40 channels (25 to 64) @25.6 - @409.6 μs, 40 Multi-Time Lags
DSPH counter sample time = 819.2 μs
Channel Sample time --- Block 9 - 16, 64 channels (65 to 128) @819.2 μs - @104.9 ms, 64 Multi-Time Lags

Table 4.5 Data distributing scheme

In this scheme, 128 channels of the correlator are divided into 16 blocks, each with 8 linear time scale channels. This means all the channels belonging to a particular block have the same sample time. The samples of a later block are simply obtained by the addition of pairs of samples of the previous block, thereby producing a doubled sample time, for example two 3.2μs samples make one 6.4μs sample, two 6.4μs samples make one 12.8μs sample etc, as shown in Figure 4.8.

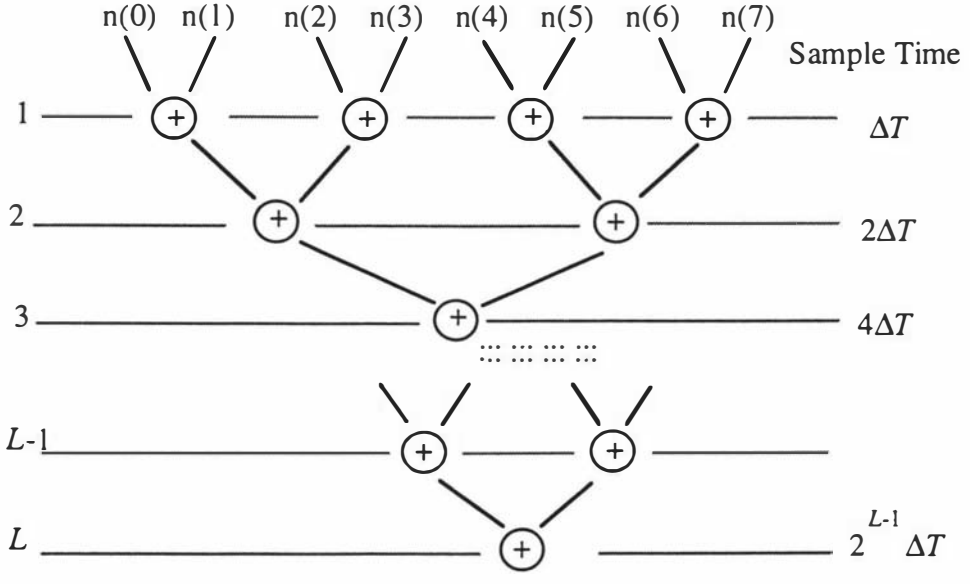


Figure 4.8 Multi-rate sample data preparation

The first DSP (DSP1) works in block 1 (channel 1 to 8). The counter for this group samples at 3.2 μs. The second DSP (DSP2) works in blocks 2 and 3 (channel 9 to 24). The counter for block 2 samples at 6.4 μs. The third DSP (DSP3) works in blocks 4 to 8 (channel 25 to 64), the counter of block 4 samples at 25.6 μs. The forth DSP (DSPH) works in blocks 9 to 16 (channel 64 to 128), the counter for block 9 samples at 819.2 μs.

Apart from the first DSP, each of the DSPs requires only part of the available time to calculate all the channels of the shortest sample time block assigned to it. The remaining computation time is used to process data corresponding to larger sample times.

The execution time of the system schedule is optimised as shown in Figure 4.9. Where T1, T2 etc. are the processing time sequences.

128 Channels / 8 = 16 blocks, 1 block = 8 Channels

	DSP1	**DSP2**	**DSP3**	**DSP4**
	ΔT = 3.2 μs	ΔT = 6.4 μs	ΔT = 25.6 μs	ΔT = 819.2 μs
$T1$	(block 1)	(block 2) + (block 3)/2	(block 4) + (block 5)/2 + (block 6)/4 + (block 7)/8 + (block 8)/16	(block 9) + (block 10)/2 + (block 11)/4 + (block 12)/8 + (block 13)/16+ ...
$T2$	(block 1)	(block 2) + (block 3)/2	(block 4) + (block 5)/2 + (block 6)/4 + (block 7)/8 + (block 8)/16	(block 9) + (block 10)/2 + (block 11)/4 + (block 12)/8 + (block 13)/16+ ...

Figure 4.9 Timing scheme

Each processor in the system has its own copy of the correlation core code and the execution time is varied as shown in Figure 4.9. During time $T1$, DSP1 processes the channels of block1, DSP2 processes the channels of block2 and the first half number of channels of block3, DSP3 processes the channels of block4, first half number of channels of block 5, ..., the remaining channels of those blocks are processed in the next sample time, and so on. During time $T2$, DSP1 processes the channels of block1, DSP2 processes the channels of block2 and the second half number of channels of block3, DSP3 processes the channels of block4, second half number of channels of block 5, ... The above analysis indicates that four DSP56001 can meet the processing requirements.

Careful inspection of this multiple tau algorithm leads to the conclusion, that the required total processing time for each DSP to process the channels assigned to him has an upper bound of exactly twice the processing time needed to compute the first 8 channels. This leads for the required processing time F_n to

$$F_n = 8\left(1+\frac{1}{2}+\frac{1}{4}+\frac{1}{8}+\frac{1}{16}\cdots\right)\Delta T_n = 16\Delta T_n \qquad (4.4)$$

where F_n is the required processing time for DSPn (n = 1, 2, 3, 4), and ΔT_n is the sample time for the first block channels assigned to DSPn.

4.3.3 Multiprocessing architectures

A multiprocessor system can be defined as one containing two or more processors of approximately similar capabilities. The idea behind multiprocessor systems is to exploit parallel activity, and so achieve the concurrent execution of two or more processes (Bond, 1987), (Wilson, 1989), (Lenoski, 1992), and (Agnello, 1992). Most real time systems now are designed using this concept of parallel activity. Also the use of efficient software can further enhance the performance of the system. In any multiprocessor system the available speed of a processor depends on how many operations can proceed in parallel on that processor, this parallelism is achieved by having several different on-chip execution units working in parallel.

4.3.3.1 Multiprocessing architectures review

There are many kinds of multiprocessor architecture (Andrews, 1992), and they must be analysed and their various merits considered within the context of real correlator applications. Analysing the factors which influence the performance of a multi-processor system is a very complex task since many factors jointly determine system performance and the modification of some factors affects many others.

The normal hardware multiprocessor structures, known as SISD, SIMD, and MIMD, will be described in this section.

1) SISD (Single-Instruction-Single-Data) computing is the traditional single-processor model. The SISD architecture is illustrated in Figure 4.10.

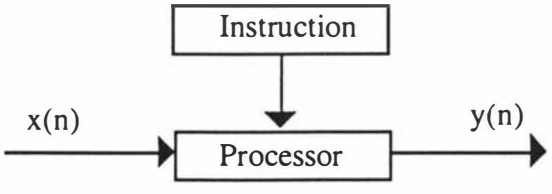


Figure 4.10 SISD structure

In a SISD system, an application is run on a single processor under the control of a single instruction stream (one instruction is taken from the program at a time), and each instruction operates on a single datum at a time. SISD machines are often given the

appearance of parallelism through operating system features for supporting multitasking. When equipped with time-multiplexing of tasks, a fast SISD machine can support a form of concurrence, but true parallelism is not supportable. Therefore, SISD hardware is incapable of parallel computing.

2) SIMD (Single-Instruction-Multiple-Data) seems restrictive at first, but is perhaps the most useful paradigm for massively parallel scientific computing. The SIMD architecture is illustrated in Figure 4.11.

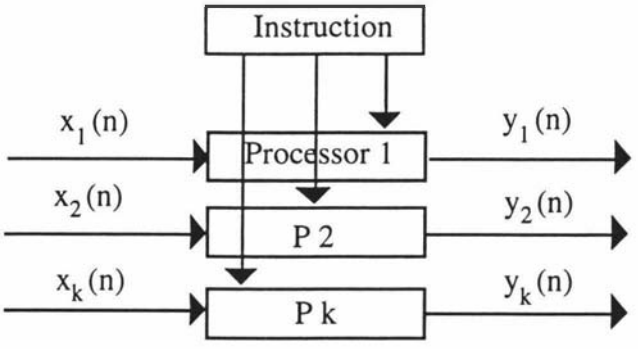


Figure 4.11 SIMD architecture

In a SIMD computer, a single instruction stream is acted upon by many processing elements, in sequence. That is, one instruction counter is used to sequence through a single copy of the program. The data that are processed by each processing element differs from processor to processor. Therefore, a single program and a single control unit simultaneously act on many different collections of data. Many scientific and engineering applications naturally fall into the SIMD paradigm. SIMD is an example of synchronous data parallel computing. Any general correlation function may be implemented using this single primitive. However, SIMD structures require substantial data transfer and storage capacity for digital signal processing applications, especially for logarithmic correlation applications. In addition, secondary memory access is normally required because it is not practical to store an entire large input data set in primary memory.

3) MIMD (Multiple-Instruction-Multiple-Data) is the most general model of parallelism. The structure is shown in Figure 4.12. MIMD is useful when the problem requires multiple different tasks to be performed at the same time. Synchronisation is achieved explicitly and locally rather than through a global synchronisation mechanism. This makes the system flexible. Because of the flexibility of MIMD, a variety of programming paradigms may be used.

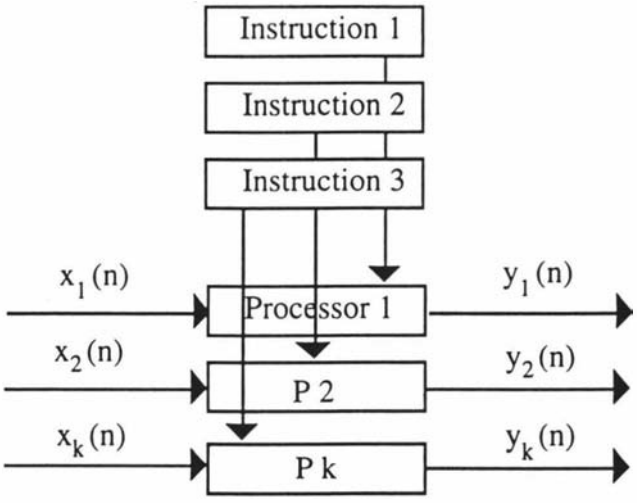


Figure 4.12 MIMD structure

The members of the MIMD group are commonly referred to as multiprocessors. One of the major architectural differences among the various types of parallel processing system is the interconnection technology used. In recent years, various interconnection networks have been studied for parallel machines designed for communication intensive applications. Normally the interconnection network for this kind of system could be either shared bus, or special switcher, or multistage network. The input and output signal data can be transferred through multiplexer and demultiplexer circuits and various types of structure can be used to interconnect the processors and memories.

There are two common forms of MIMD interconnection structure. These are the 'tightly coupled' and the 'loosely coupled' multiple architecture. A tightly coupled architecture uses shared memory as a communication medium, as shown in Figure 4.13.

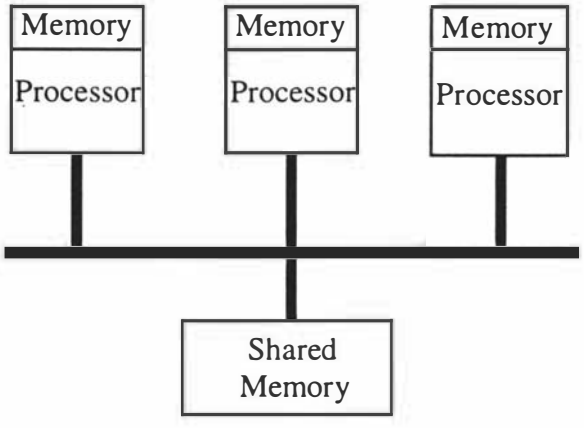


Figure 4.13 Tightly coupled shared bus system

A loosely coupled multi-DSP system contains a number of processors with local memory, interconnected via point-to-point communication links. The individual

processing units tend to be autonomous, working on their own tasks with little interprocessor communication, as illustrated in Figure 4.14.

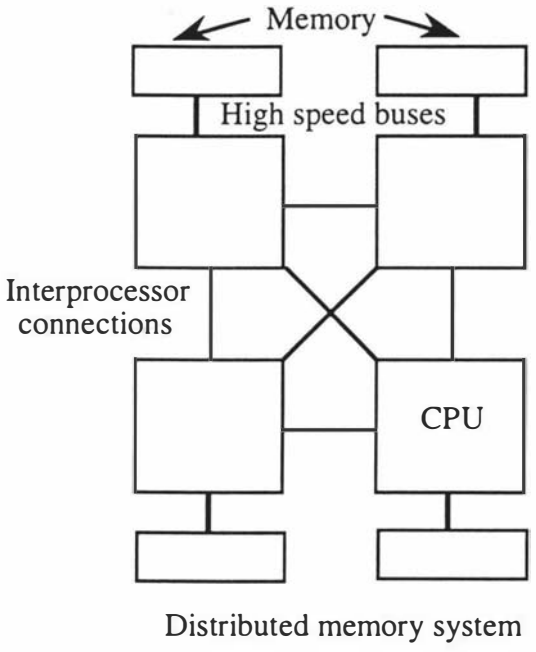


Figure 4.14 Loosely coupled distributed memory system

4.3.3.2 Parallel architecture performance analysis

There are several ways to measure the performance of this parallel architecture. Most obviously, we can count the total time taken T and the number of processors used P. It is also interesting to compare the speed attained by the parallel algorithm with that of the best sequential algorithm. The speed enhancement S of a parallel algorithm is defined to be the ratio of the running time L of the fastest sequential algorithm to the running time T of the parallel algorithm, ie.,

$$S = \frac{L}{T} \tag{4.5}$$

To obtain meaningful results, accurate estimations are needed for the computation times, memory requirements, and the time cost due of interprocessor communication. The execution time for implementing a multiple tau correlation function algorithm with DSPs can be divided into three parts: autocorrelation arithmetic, counter data reading, and interprocessor communication operations. We denote the unit time cost of each operation as C_a, C_{io}, and C_c, respectively. If the number of arithmetic, counter reading, and communication operations per sample of output is denoted as N_a, N_{io}, N_c, respectively, then the execution time per sample can be represented as follows:

$$t = \frac{T}{L} = C_a N_a + C_{io} N_{io} + C_c N_c \tag{4.6}$$

One of the objectives of the algorithm design is the minimisation of the execution time.

The execution time of the single DSP and multi-DSP algorithms can be represented as in eqn (4.7 and 4.8), where the subscript s denotes the single processor and the subscript m indicates the multiprocessors. In many cases, a parallel algorithm needs more arithmetic operations compared with its sequential counterpart, and as a result, $N_{a,m}$ can be larger than $N_{a,s}$.

$$t_s = C_a N_{a,s} + C_{io} N_{io,s} \tag{4.7}$$

$$t_m = C_a N_{a,m} + C_{io} N_{io,m} + C_c N_{c,m} \tag{4.8}$$

Three performance measures: (1) throughput, (2) speed enhancement, and (3) efficiency are used for evaluating multi-DSP implementations.

(1) The throughput

The throughput can be obtained by dividing the number of processors by the execution time per sample time of the system. The throughput using four processors, $R(4)$, is represented by the following equation

$$R(4) = \frac{4}{C_a N_a + C_{io} N_{io} + C_c N_c} \tag{4.9}$$

(2) The speed enhancement

The speed enhancement is the ratio of the throughput in the multi-DSP system to that in a reference single processor system.

$$S(4) = \frac{4\left(C_a N_{a,s} + C_{io} N_{io,s}\right)}{C_a N_{a,m} + C_{io} N_{io,m} + C_c N_{c,m}} \tag{4.10}$$

(3) The overall efficiency

The overall efficiency is defined as the ratio of the speed enhancement and the number of DSPs:

$$E(4) = \frac{C_a N_{a,s} + C_{io} N_{io,s}}{C_a N_{a,m} + C_{io} N_{io,m} + C_c N_{c,m}} \tag{4.11}$$

The above equations clearly indicate that the reduction of the communication cost is vital for increasing the efficiency or speed enhancement ratio.

4.3.3.3 Selecting a loosely coupled MIMD architecture

A loosely coupled MIMD architecture was chosen to implement the multiple tau correlation algorithm for the following reasons.

(1) Multiple tau correlation computing requires several independent activities to occur at the same time. MIMD machines are typically composed of SISD devices, and each device operates independently and asynchronously. Several devices can cooperate to solve a complex problem and can also operate independently to solve different problems. So different DSPs can work in different sample time blocks and hence dramatically improve the response time of the system. Such a processing system makes a very cost-effective transaction processing system for a multiple tau autocorrelator.

(2) A loosely coupled MIMD structure allows the system to be designed using a simple communication, scheduling and synchronisation mechanism. The most serious problem with a shared memory system is the possibility that data transfer between the processing boards will cause a bottleneck in the system's performance. In order to access the shared memory, tightly coupled systems require a complex arbitration scheme to ensure mutual exclusion of the common data bus. This arbitration mechanism can be a hindrance to system expansion. Loosely coupled systems with point-to-point communication structures do not require any shared arbitration logic.

(3) Loosely coupled MIMD structures can be readily extended. In shared-memory machines, adding processors to a system can degrade performance by bus saturation. With distributed-memory MIMD, the computing power increases almost linearly with the number of processing units. Also the correlator minimum lag time can be reduced and number of channels extended by adding much more powerful processing units in the future.

In summary a loosely coupled MIMD structure requires minimal interprocessor communication and I/O operations and can use the same multiprocessor architecture to implement alternative digital signal processing algorithms.

4.3.4 Design of Parallel Algorithms

In a multi-processor system, the real challenge is in developing the corresponding parallel application software (Chaudhary, 1993) and (Singh, 1993). The requirement for a parallel algorithm is to dynamically control the task allocation and execution within a multi-DSP system. This adds complexity to the design and unless care is taken, the

software overheads incurred by the parallel algorithm can degrade the performance of the system.

In order to minimise the complexity of the task allocation system it is best to perform the allocation of code to the DSPs prior to run time. Fortunately, using a loosely coupled multi-DSP architecture, the scheduling of tasks to DSPs can be determined without the requirement for extensive development tools. The application can be written as a series of tasks and assigned to the appropriate DSPs with the communications mechanisms between the tasks defined. This enables code to be quickly developed and tested on a single DSP prior to network scheduling.

The method adopted here uses different DSPs to calculate different blocks of channels at the same time. The algorithm simply divides the total computational work required into equal parts and subtasks each part to a different DSP. The aim is to keep the algorithm as simple as possible while minimising data transfer between processors.

There has been no single dramatic breakthrough in parallel programming to match the hardware advances in the last few years (Curtis, 1994). Programming a parallel multi-processor system is still more difficult than sequential programming, although the choice of a distributed MIMD multi-DSP system has eliminated a great deal of the heartbreak from the programming.

Challenging problems in the future will rely on the development of algorithms appropriate for parallelism for this system. It should be no surprise that much of what is known about multiple tau time scale autocorrelation algorithms will have to be revisited in the context of parallelism in the future.

4.4 Correlator Architectures

The architecture of the correlator depends on the system hardware and the software architecture, including the data communication between the processors. In the correlator system, successive input samples must be processed to produce an output channel data stream within a relatively short time delay. The correlator must execute the algorithm fast enough to avoid any loss of input data.

The structure of the correlator designed using a loosely coupled MIMD multi-DSP system is shown in Figure 4.15. The correlator is designed using separate sampling and processing sections which are connected through a fast 24-bit counter system. The counter system offers a multiple sample-time mode, where ΔT and the lag-times are both doubled every 8 channels.

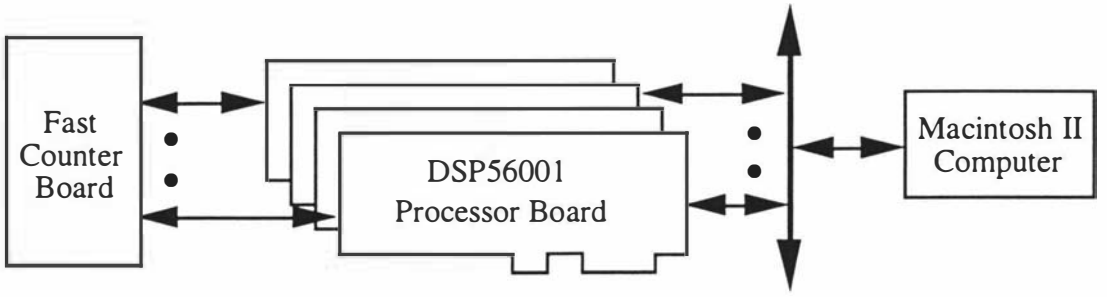


Figure 4.15 Correlator system based on MIMD structure

This section will describe the hardware architecture of the correlator. The design is based on the above analysis (sections 4.1 - 4.3). The system works as follows. The multi-DSP system receives signals from the fast counter system, the sampling rate is changed, the autocorrelation calculations are performed. The completed unnormalised correlation and monitor data are transfered to the computer to display and perform any additional required analysis. Each processor performs its tasks with different priorities. These are (from highest to lowest): counter sampling, accumulation, data transfer and graphic display.

4.4.1 The counter array system

The need to provide a large counter storage capacity (24-bit) for each correlation channel increases the cost of the device and requires a complex logic network to enable the counter contents to be read and displayed in real time. The counter system provides the input digital data streams to the signal processors. Multiple sample time clocks are used to control the counter array at different sample times, as scheduled in section 4.3.2. The system is shown in Figure 4.16.

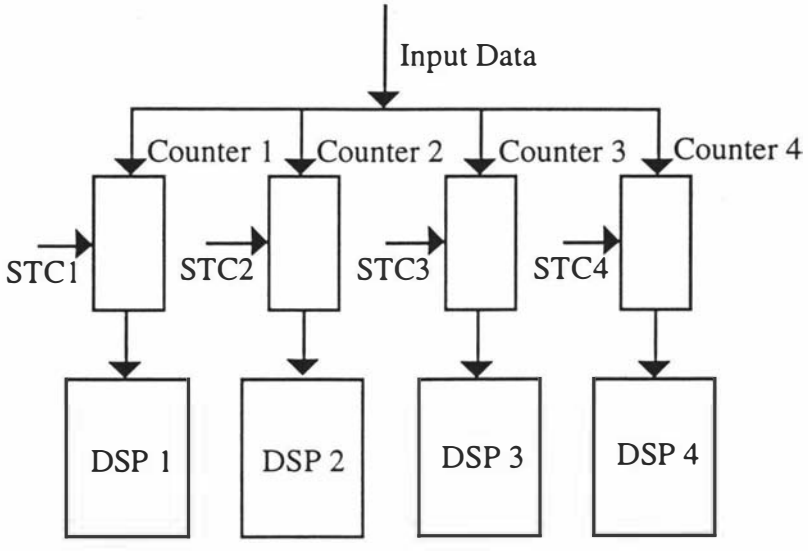


Figure 4.16 Counter array system

The pulses from the photomultiplier are shaped to standard TTL form by a preamplifier and discriminator circuit. These output pulses can be accepted by the counters, and then accumulated over the sample times. The sample time clock system should be able to generate four sample time clocks STC1, STC2, STC3, and STC4. The time relationship between STC1, 2, 3, and 4 should be changeable, so different time structures can also be implemented.

As mentioned in section 3.1, photon counting systems could exhibit a counting dead time interval, during which time incoming pulses are not counted. A derandomiser and a synchroniser can be used to make a zero dead time counter system to ensure that all pulses from the photomultiplier are counted (O'Driscoll, 1982).

The counted data are passed directly to a DSP via the auxiliary input port to the DSP data memory for instant real time processing. This method makes full use of the DSP's computing power to make the system work in real time. In this method, an interrupt routine causes the counted data to be loaded into the DSP memory. This interrupt service routine is a piece of software that is designed to react to an interrupt generated by the sample time clock. The DSP system uses one storage area of fast RAM to store the delayed and immediate data. The correlation calculation routine which processes the incoming data is interruptable, so that new values continue to be read while the accumulation calculation is taking place. The data interrupt routine has the highest interrupt priority.

The data are stored in a circular list, so that new data replace records which have been processed and are no longer needed. Computations are started from the last channel. This contains the oldest data and is therefore the next one to be overwritten. Clearly the sample time ΔT should not be less than the sum of the calculation time t_c and the interrupt routine execution time t_i, ie.

$$\Delta T > t_c + t_i \tag{4.12}$$

4.4.2 The signal processor system

The MIMD correlator architecture involves issues at the node and system levels. Each node is a processor unit comprising a DSP56001 with a memory segment and I/O interfaces. The memory subsystem also has to be carefully designed to provide fast and reliable communication within the node so that each node becomes a self-contained computer. The nodes must provide speed and predictability in executing real-time tasks, in handling interrupts, and in interacting with the counter and control computer. These signal processing nodes perform the primary multiple tau autocorrelation functions required by the DLS application.

The nodes cannot access each other's private memory space, and hence a communication network is required to transfer data between nodes. As mentioned in section 4.3, the performance of the system is as much affected by the speed of this communication network as it is by the speed of the processing elements. Thus the balance between communication-time and computation-time in this message-passing machine is an important design issue.

Data communication between DSPs could cause synchronisation problems unless a care is taken at the design stage. Also as mentioned in section 4.3, the major obstacle to the prevalent use of the multiprocessor systems has been the absence of efficient tools to automatically schedule programs onto the multiprocessor structure. This is especially true for inter-DSP communication. The only way to keep these problems firmly under control is to plan very carefully, and solve the problems at the design stage. It is much more difficult and expensive to try to solve these problems at the code stage. The design of an architecture which needs less interprocessor communication and fewer I/O operations is an important requirement.

There are four different kinds of data access requirement in the system. The first kind is DSP access to its own data. This can go through the DSP's own 24-bit data bus which is connected to fast RAM.

The second kind is access to the counted data. Each DSP also accesses the counter system through the DSP56001 24-bit data bus for high speed processing.

The third kind of data access requirement is data exchange between DSPs and the control computer. To make the system more of a general purpose signal processing station, the placement of the I/O controllers needs to be done in such a way that all nodes are able to communicate with the control-computer. In the correlator scheme, all nodes are connected to the interface manager which is used to down load the processing program from the control computer to the nodes. The control computer accesses each DSP node through a computer-DSP interface bus.

The fourth kind of data access requirement is data exchange between different DSPs. This can be achieved by using the DSP56001 host ports as the interface for a separate 8-bit data bus.

The DSP views its host interface port (HI) as a memory-mapped peripheral occupying three 24-bit words in the data memory space. Separate transmit and receive data registers are double buffered to allow efficient data transfer at high speed. Memory mapping allows the DSP CPU to communicate with its HI registers using standard instructions and addressing modes. In addition, the MOVEP instruction allows HI-to-

memory and memory-to-HI data transfers without going through an intermediate register.

Figure 4.17 is a block diagram showing the registers in the HI. The registers can be divided vertically down the middle into registers visible to the host processor on the left and registers visible to the DSP on the right. They can also be divided horizontally into control at the top, DSP-to-host data transfer in the middle (HTX (host transmit register) and RX (receive register)), and host-to-DSP data transfer at the bottom (TX (transmit register) and HRX (host receive register)).

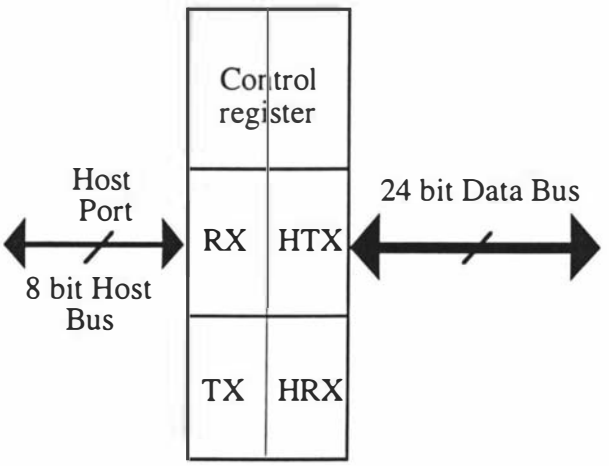


Figure 4.17 Host port data transfer structure

Multiple buses have been used to separate very high speed processor to memory functions from slower I/O peripherals (Andrews, 1989) and (Marsan, 1982). Use of the host bus allows high processor utilisation and efficient data communications, because it preserves the DSP 24-bit data bus for counter reading and memory access only.

When the multiple tau correlation processing is completed, the channel information located in the slave DSPs are passed to the host DSP through the host 8-bit data bus. There may be more than one DSP with data to transfer. The host DSP is the bus master and it gains control of the bus to transfer a block of data between processors.

Based on the above, the resulting correlator system scheme is illustrated in Figure 4.18. The multi-sample time counters are clustered together and each counter is assigned to a DSP.

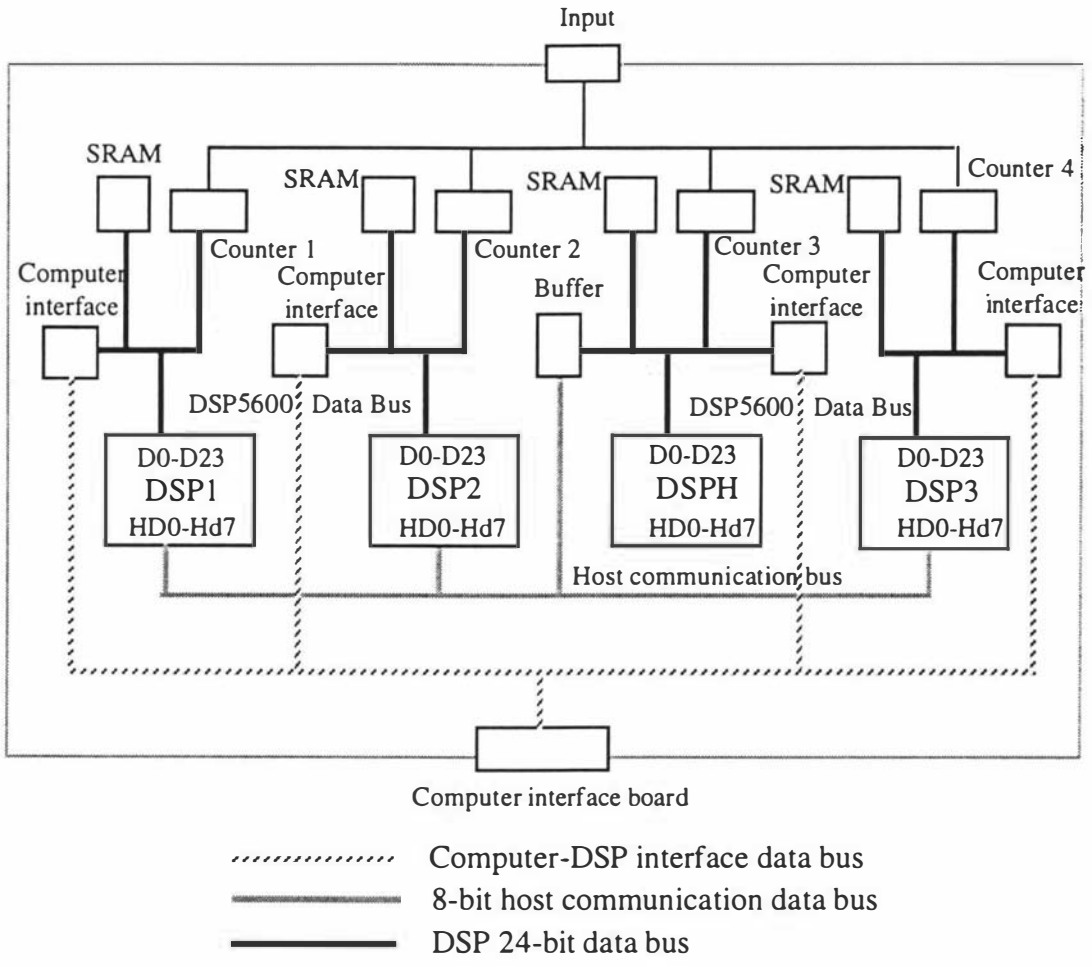


Figure 4.18 Communication bus structure

The main function of the host DSP is to execute operations necessary to deliver the calculated channel data to the on-line control computer and to control the processing of the other DSPs. When the control computer needs correlation data for further analysis and display, it communicates with the host DSP to ensure that the operation of other DSPs is not interrupted. These other DSPs may be performing calculations on the small sample time channels. The host DSP is also responsible for functions in the network and the data links of the system. The host DSP must establish connections between the source and destination DSPs.

There is a control computer (Macintosh IIvx) available to work with the correlator. The control computer performs the overall management of the correlator. This includes processing option selections, mode selections, mode parameter changes, input/output channel selections, counter system control, display system control, and fault monitoring. An interface program provides the correlator-to-computer interface. The display program residing in the control computer provides the required data processing and formatting necessary for display purposes. Many forms of postprocessing

algorithms can be used, including noise analysis and signal to noise enhancing techniques.

4.4.3 The entire structure

The boards inter relationships in the correlator system is shown in Figure 4.19.

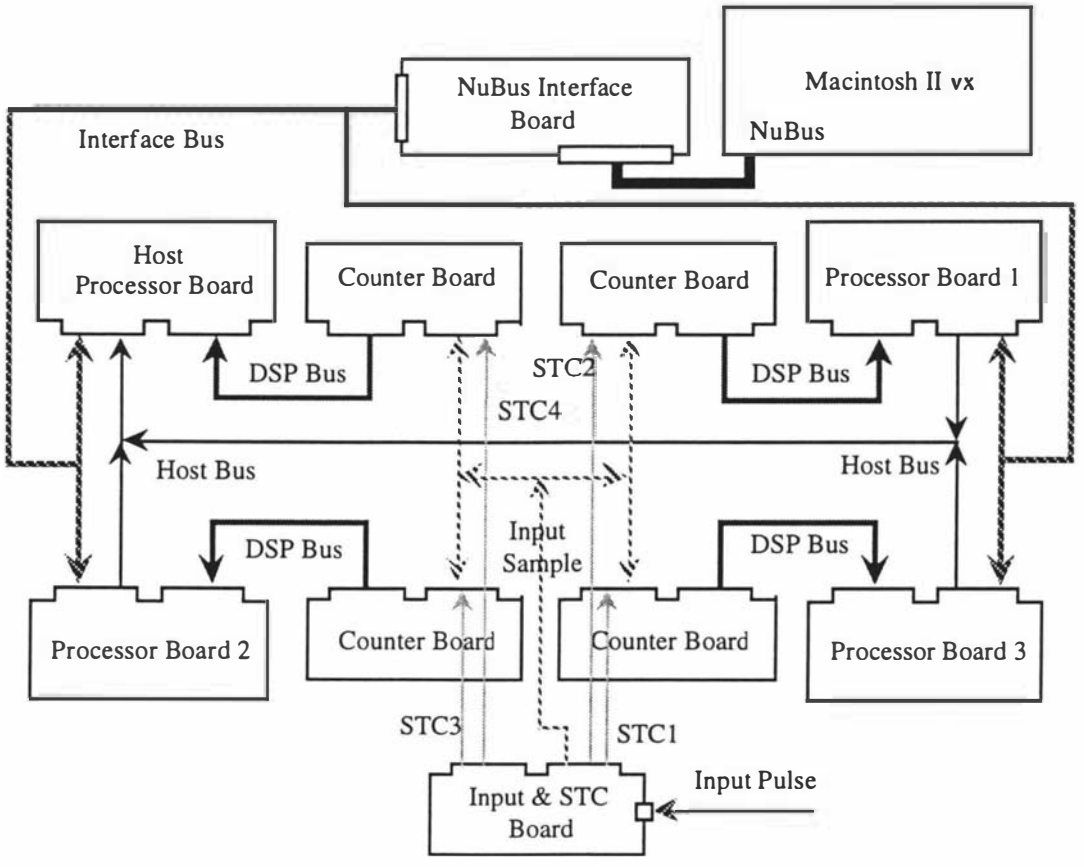


Figure 4.19 Boards inter relationships in the correlator system

Each DSP performs all the operations needed for the processing of the given block of channels, and the intermediate results do not normally flow from one processor to another. Each DSP board contains a resident monitor program for performing the following functions; (1) initialisation, (2) control and communication.

On power-up the resident monitor program is automatically loaded and executed by the DSP board. The DSP board then enters an idle state waiting for commands from the control computer. Then the DSP board loads in the processing program from the control computer, different processing program can be loaded in. The correlator system currently has four processing boards but is expandable to more processing units if desired.

The entire correlator system consists of three parts, the correlator station itself, the control computer, and the correlator to computer interface system which is the instrumentation "front end" for the data interface board, see Figure 4.20. Also included in the diagram is a spectrometer.

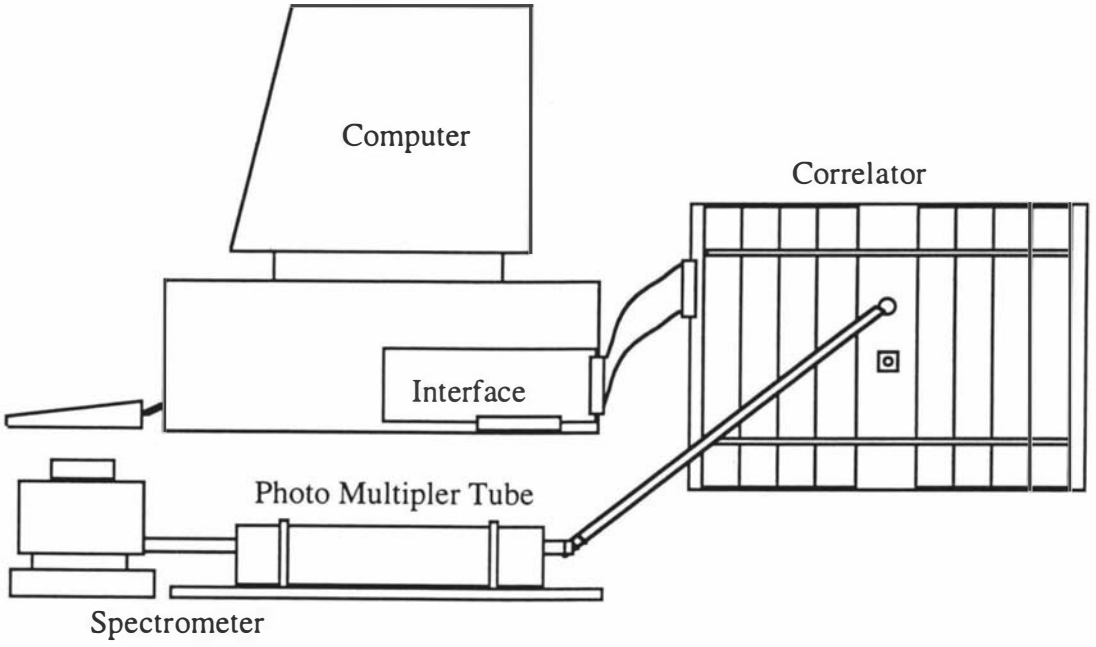


Figure 4.20 Correlator system structure

5 CORRELATOR DESIGN III - CIRCUITS AND SOFTWARE DESCRIPTION

The circuit design and software operation of the multi-DSP autocorrelator are described in this chapter. The emphasis will be on the electronics used in the correlator hardware, and on detailed descriptions of the hardware circuits and software programs.

5.0 Overall Correlator System

Figure 5.1 shows the overall block diagram of the correlator system. The system is designed with an open structure to make it extendable and flexible. The entire correlator can be considered to be composed of 4 functional blocks: the input and clock (one clock board), the counter (four counter boards), the processor (four DSP boards) and the computer interface block (interface circuit and interface card).

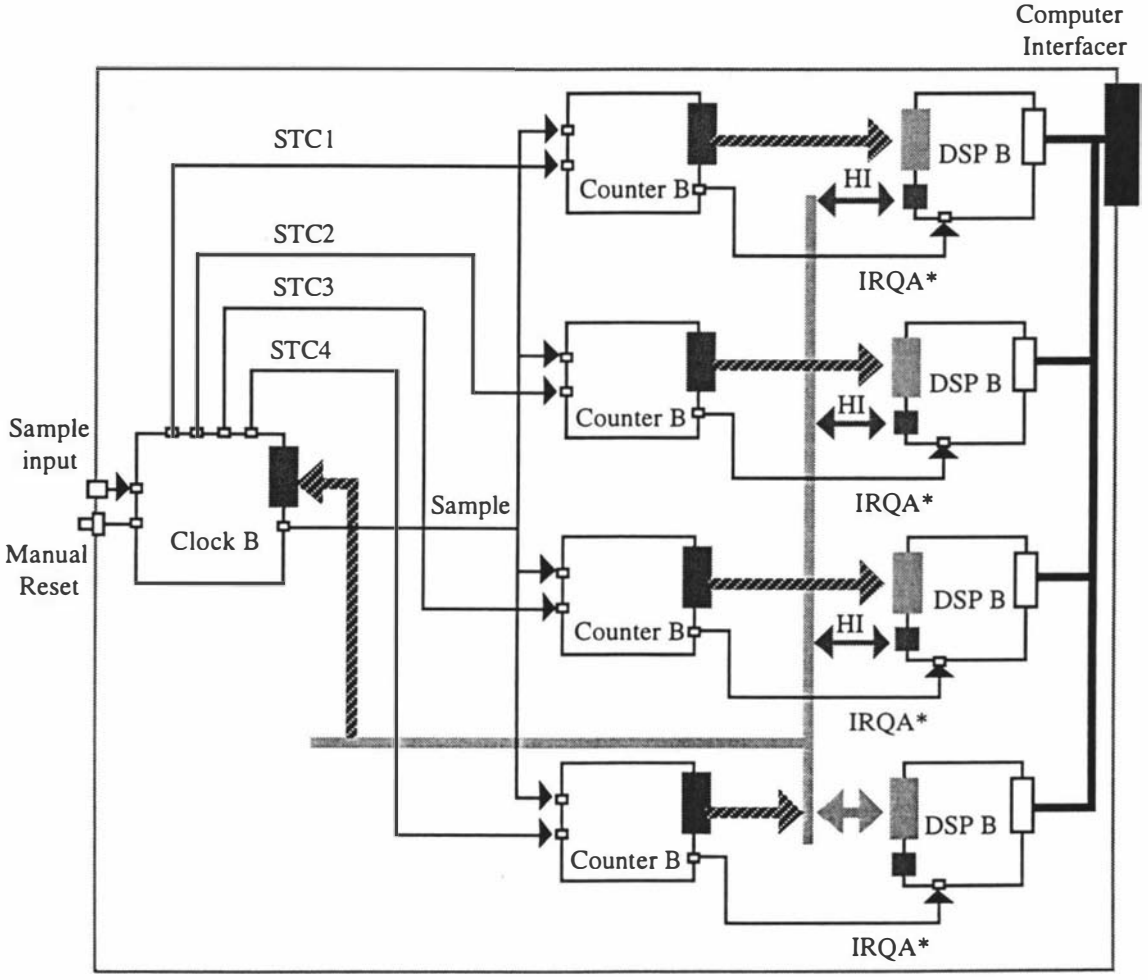


Figure 5.1 Overall block diagram of correlator system

5.1 Input and Clock Board

There are three parts to the input and clock system, as shown in Figure 5.2. They are the input system, the programmable sample time clock system, and the reset system. This block is made on a single printed circuit board and two 64 pin Euro DIN connectors are used to connect this board to the rest of the correlator.

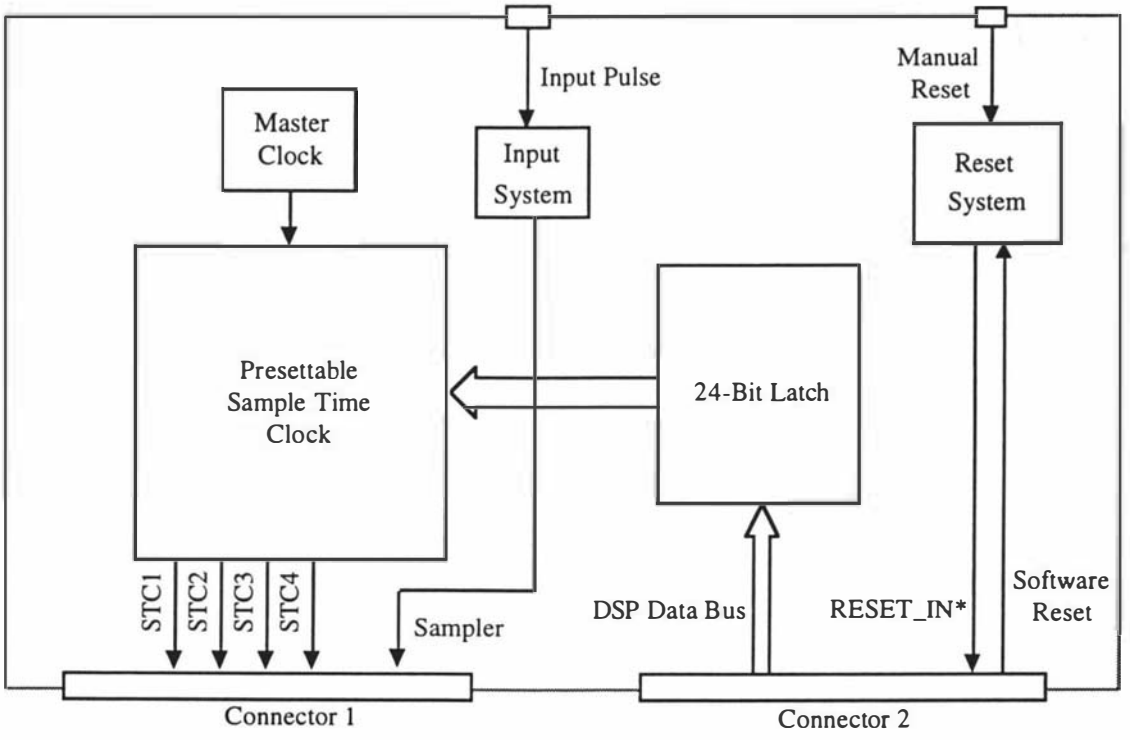


Figure 5.2 Input and clock system block diagram

5.1.1 Input amplifier

The input amplifier circuit is based on that used in the clipping mode correlator in our laboratory (O'Driscoll, 1982). Adopting this input circuit makes the new correlator compatible with the existing DLS system. The circuit is redrawn here (Figure 5.3).

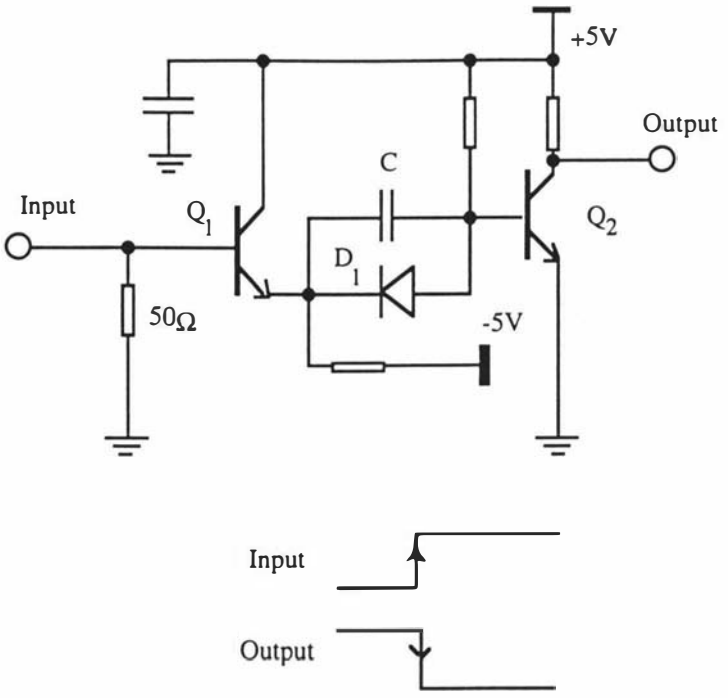


Figure 5.3 Input amplifier

The input amplifier is designed to have a 50Ω input resistance and to operate with 1V pulses of positive input polarities. An emitter-follower input buffer ensures that a constant 50Ω resistance appears at the input for input voltages in the range -5V to +5V. The second stage is a discrete component diode transistor logic inverter. The voltage on the base of Q_2 is zero if the input voltage is also zero. This assumes equal voltage drop across the base-emitter junction of Q_1 and diode D_1. A 1V input turns on Q_2 causing the collector voltage to change from logic '1' to logic '0'. The positive going edge of the input voltage causes a '1' to '0' transition at the output of Q_2. Capacitor C is included to reduce the transition time of the Q_2 output waveform.

5.1.2 Programmable sample time clock

As mentioned in chapter 4, four counters are used in the correlator system. Because they simultaneously sample at different sampling times, multiple sample time clocks are needed for the counter system.

The 74F series FAST TTL Logic family was selected for this system. 74F TTL products are made by combining advanced oxide-isolated fabrication techniques with standard TTL functions. The high operating speed of the 74F family enables a system to operate at speeds previously only possible with 10K ECL, but with simple TTL design rules and single 5V power supplies. Low input loading allows the user to mix the LS TTL, ALS TTL, and HCMOS logic families in the same system without the need for translators and restrictive fanout limitations.

There are four main parts in the clock system. They are: the 20 MHz master reference clock; the multistage clock divider system; the magnitude comparator system; and the data latch system controlled by a DSP.

The 20 MHz master reference clock is based on a 40 MHz crystal oscillator clock module as shown in Figure 5.4. The clock must attain specified rise and fall times. The crystal oscillator output is divided by two to guarantee approximately 50-50% high-and-low duty cycle, and the output of the system is buffered so that the clock is not loaded by other logic circuits, thus ensuring that its rise and fall time requirements are met (5ns).

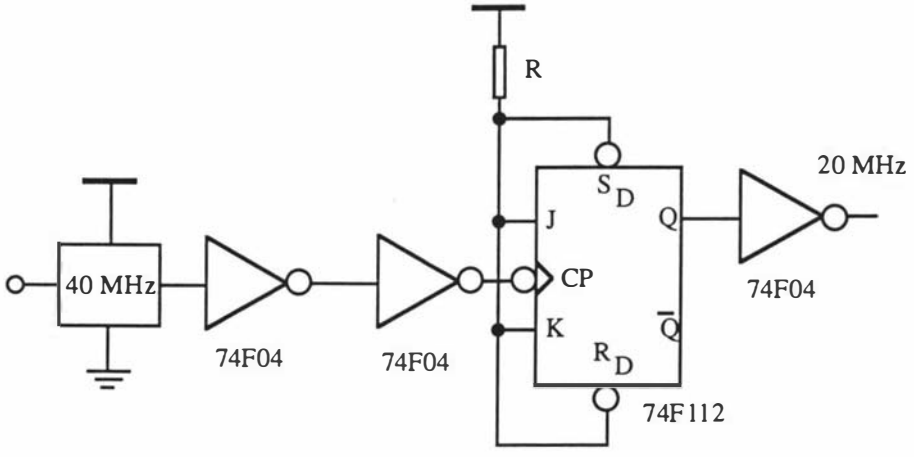


Figure 5.4 20 MHz master clock

Figure 5.5 shows the sample time clock block diagram. The multistage sample time clock dividing system is made using five 4-Bit Binary Counters to form a synchronous multistage counter, which counts clock pulses from the 20 MHz master clock. The comparator system is made using five 4-Bit Magnitude Comparators. It compares the output of the counter with the data stored in a 24-bit latch. Sample time clock pulses STC1, STC2, STC3, and STC4 are generated when the numbers in the counter equal the numbers stored in the latch. STC1 - STC4 are the outputs gating by bits 0 - 3 of the 24-bit latch.

The clock system can work in a very flexible way with four different sample time clocks controlled by a DSP. This enables the whole experiment to be performed under software control.

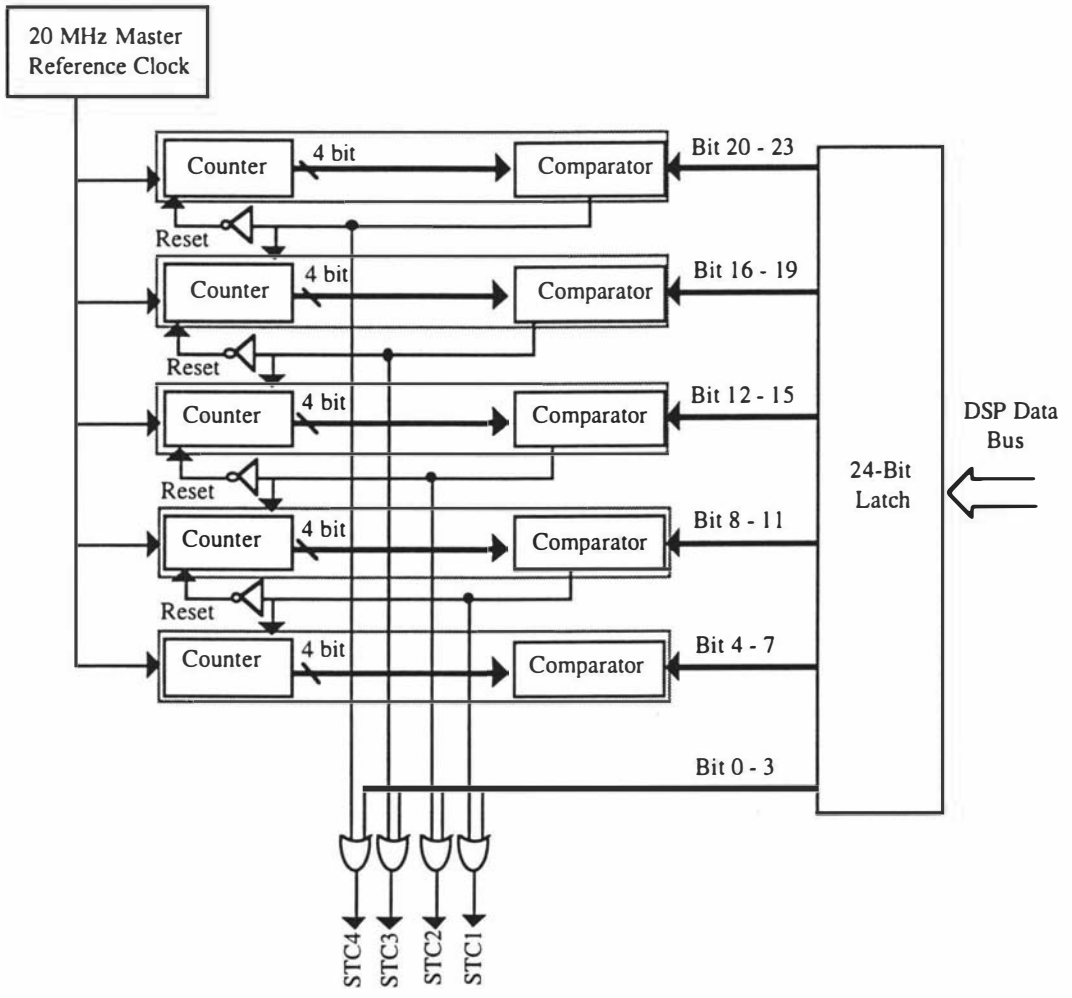


Figure 5.5 Sample time clock block diagram

The circuit diagram of the sample time clock is shown in Figure 5.6. The counters can count and reset synchronously. A low level at the Reset (SR*) input sets all four outputs of a counter (Q0 - Q3) to low levels after the next positive-going transition on the Clock (CP) input. The carry look ahead simplifies synchronous cascading of the counters.

One 4-bit binary counter (74F163) and one magnitude comparator (74F85) comprise one group. If the data A input and data B input of a magnitude comparator are the same, the A=B output is high. The high output is used to reset the counter via an Inverter (74F04) and generates a carry pulse to the next counter group by enabling that group to count the next master clock pulse.

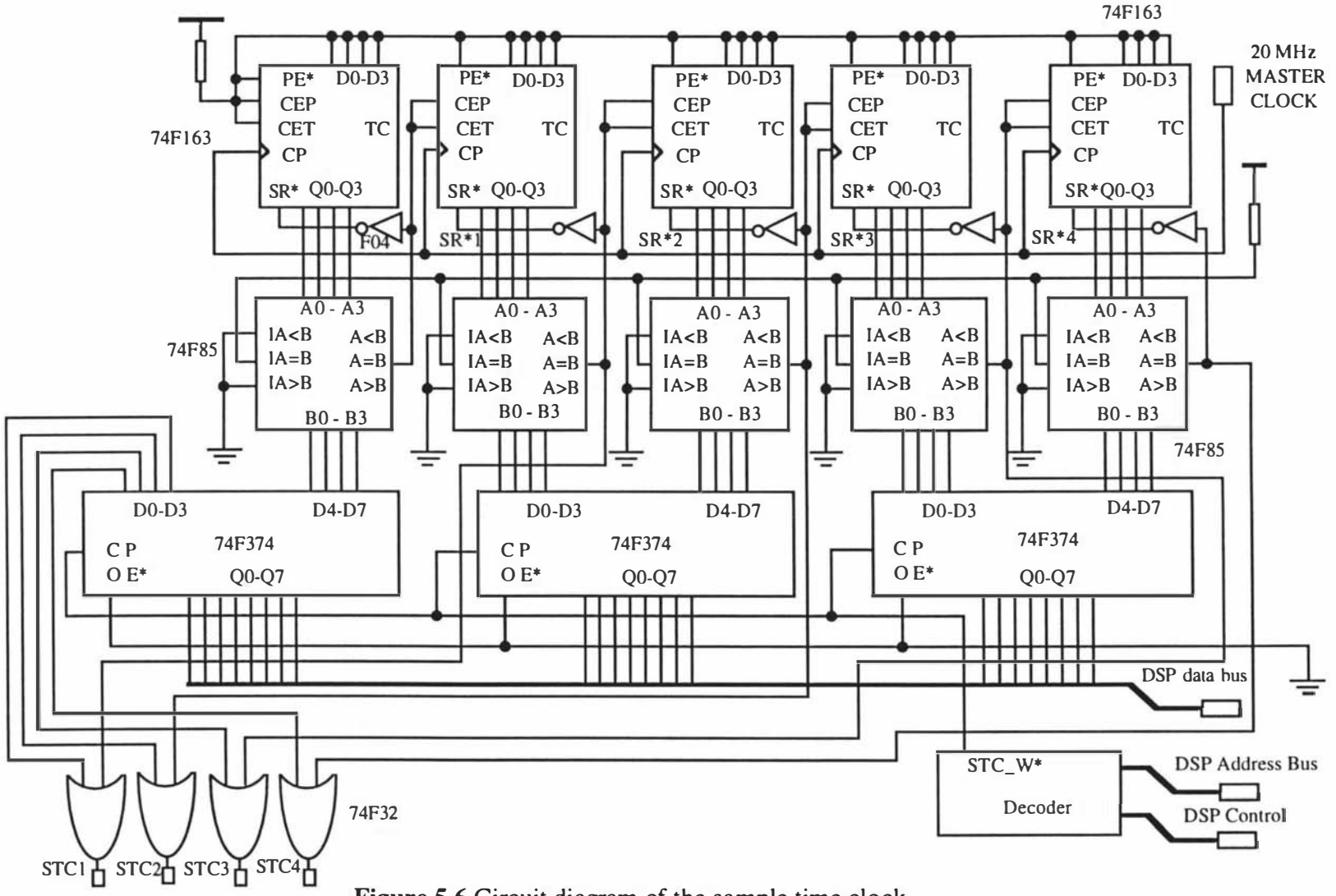


Figure 5.6 Circuit diagram of the sample time clock

Three Octal D Flip-Flops (74F374) are used to form a latch with 24 tri-state outputs, only 20 of which are used by the comparator. The other four control four 2-Input OR Gates (74F32) which function as switches to allow each of the four sample time clocks to be turned on or off. The DSP can simply write different data into the input of the Octal D Flip-Flop to obtain a different sample time.

5.1.3 The reset system

The reset system (Figure 5.7) is based on a 555 timer and provides a 150 millisecond reset time for both power up and manual reset operations. The 1N914 Diode (D1) is used to detect any power glitches. When power is momentarily interrupted, this diode quickly discharges the timing capacitor which triggers the timer, causing a reset operation.

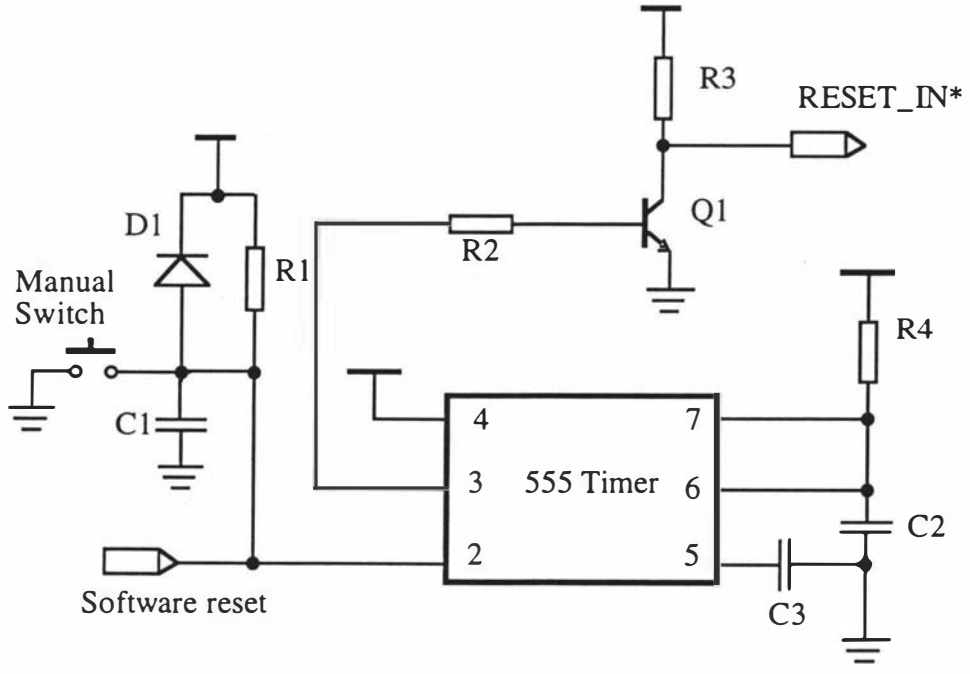


Figure 5.7 The reset circuit

The reset state can be generated either manually or by the computer. This causes the timer to generate a RESET_IN* signal. The RESET_IN* signal is used to reset the DSP boards and to setup the initial operating mode of the DSPs.

5.2 The Counter Board

5.2.1 Counter overview

The counter accumulates the number of photoelectron pulses occurring during each sample time. The 24-bit output of this counter is transferred into the DSP and the counter is reset at the end of each sample period.

The counter board layout is shown in Figure 5.8. The whole system consists of five functional blocks: 1, Derandomiser. 2, Synchroniser. 3, Clear signal generator. 4, Counter circuit. 5, Buffer.

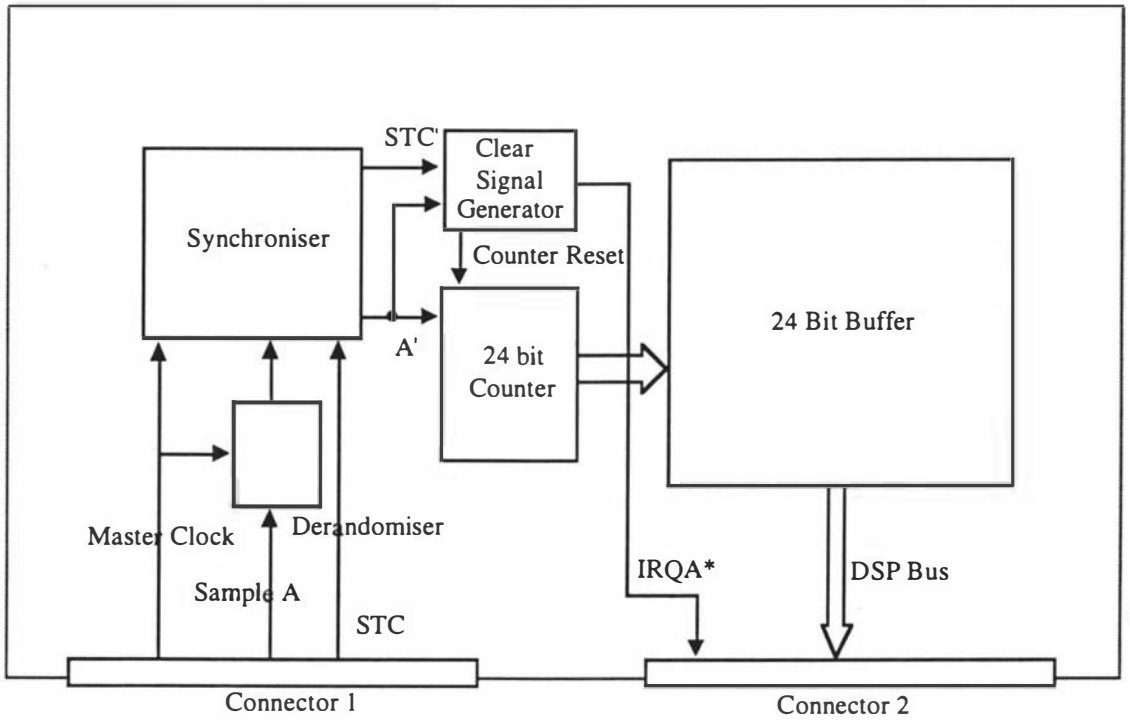


Figure 5.8 Counter board block diagram

5.2.2 Derandomiser and Synchroniser

The design of the correlator can be simplified if input pulses occur only at certain times. This is achieved by using a derandomiser and a synchroniser (Oliver, 1974), (O'Driscoll, 1982). The derandomiser defines time intervals equal to 50 ns, and shifts the input pulses within these intervals to be synchronised with the 20 MHz master clock. This time shift does not effect the correlator operation since for a given sample period it is necessary to know only the number of pulses occurring and not the actual time of occurrence within the sample period. Knowing the time of arrival of the input pulses enables the counter to be reset between pulses, which ensures that there is zero dead time and all input pulses

are counted. The synchroniser is used to match the input pulses and the sample time clock (STC) pulses with the master clock pulses. All pulses have a duration of 25 ns. The input and STC pulses (when they occur) start at the same time as the master clock pulse.

The derandomiser and synchroniser circuits used were taken from an earlier clipping autocorrelator (O'Driscoll, 1982).

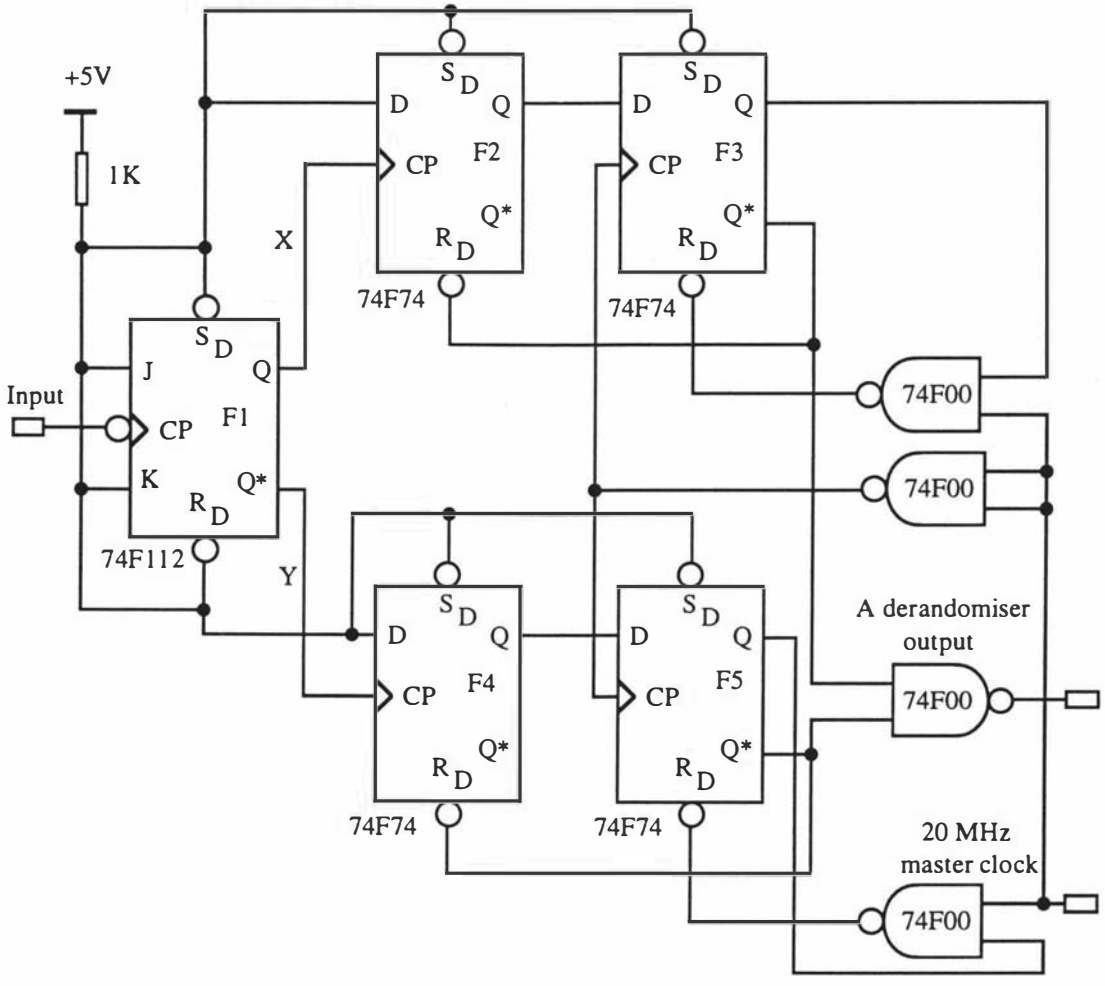


Figure 5.9 Circuit diagram of the derandomiser

The derandomiser circuit has two identical signal paths to ensure that all input pulses are processed. If a single path were used a pulse would be lost if it arrived while the preceding pulse was being delayed for synchronisation with the master clock. Input pulses are directed by F1 alternately into paths X and Y. The occurrence of a pulse in path X is recorded by the output of F2 going '1'. This logic '1' applied to the D-input of F3 causes the Q* output of F3 to go '0' on the positive edge of the master clock pulse, clearing F2 and setting the derandomiser output '1'. The negative edge of the master clock pulse clears F3 and returns the derandomiser output to '0'. Thus the pulse from the derandomiser is locked to the master clock and delayed with respect to it by the device propagation delays. The process is repeated in path Y for the next pulse.

Operation without loss of input pulses requires that paths X and Y be as nearly identical as possible. If one path is slightly slower than the other it is possible for a pair of closely spaced input pulses to appear at the outputs of paths X and Y at the same time. Such pulses would overlap and thus one of them would be lost. Mismatching is minimised by using dual flip flops so that F2 and F4 are on the same integrated circuit chip, as are F3 and F5.

The synchroniser matches the pulses from the derandomiser and the two clocks by means of latches L1, L2 and L3 which are loaded on the negative edge of the 20 MHz master clock and cleared on the following positive edge, as shown in Figure 5.10.

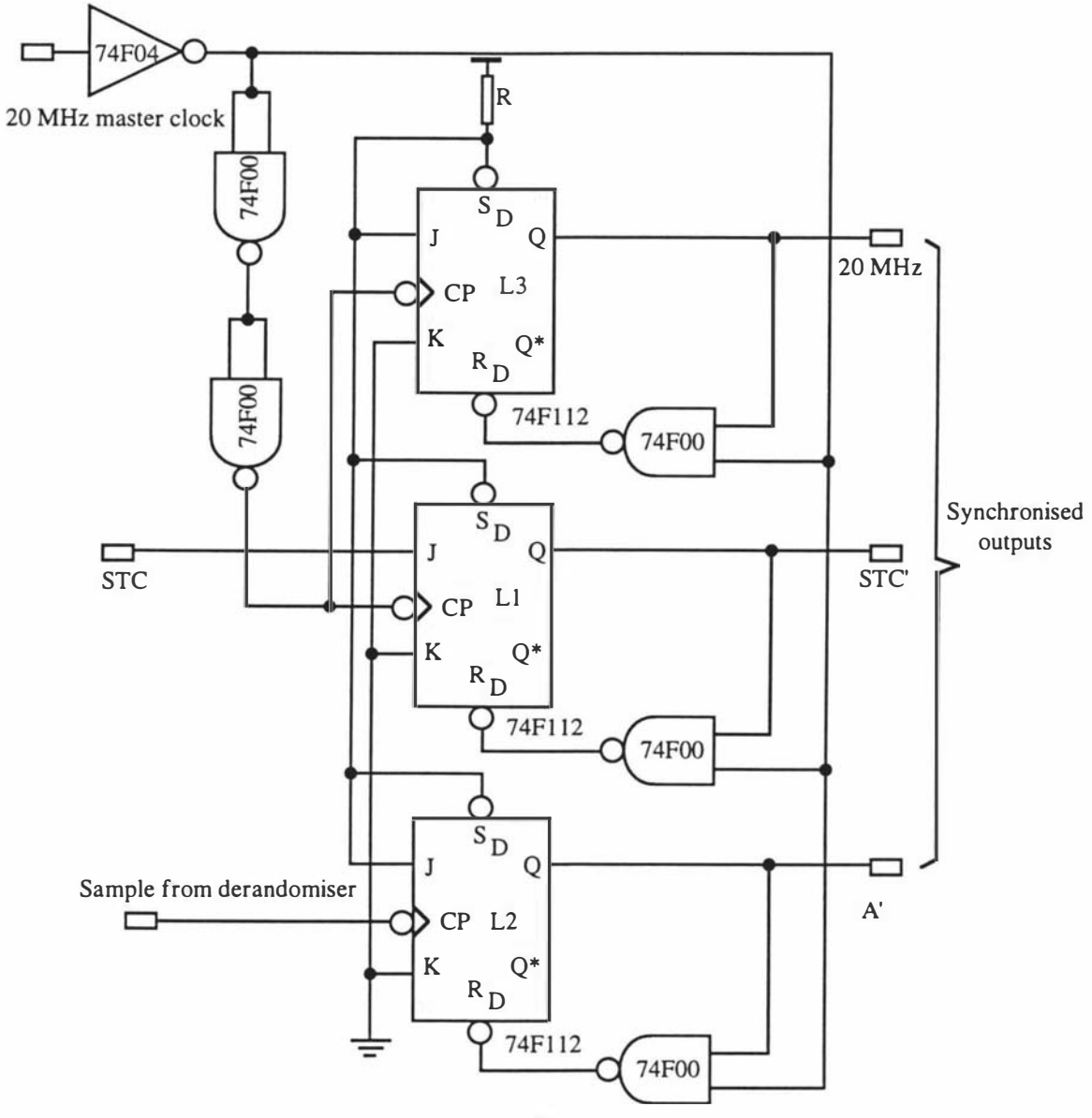


Figure 5.10 Circuit diagram of the synchroniser

Clocking latch L2 from the derandomiser output ensures that all pulses from the derandomiser are counted. The negative edge of the derandomiser output pulse is delayed

with respect to the 20 MHz master clock by four gates in series since the 'clear' input and Q* output of F3 (and F5) are linked internally by a single gate. An equivalent four gate delay (gates 5-8) ensures that the clocking of L1 and L3 coincides with that of L2. Typical pulse sequences for the synchroniser are shown in Figure 5.11.

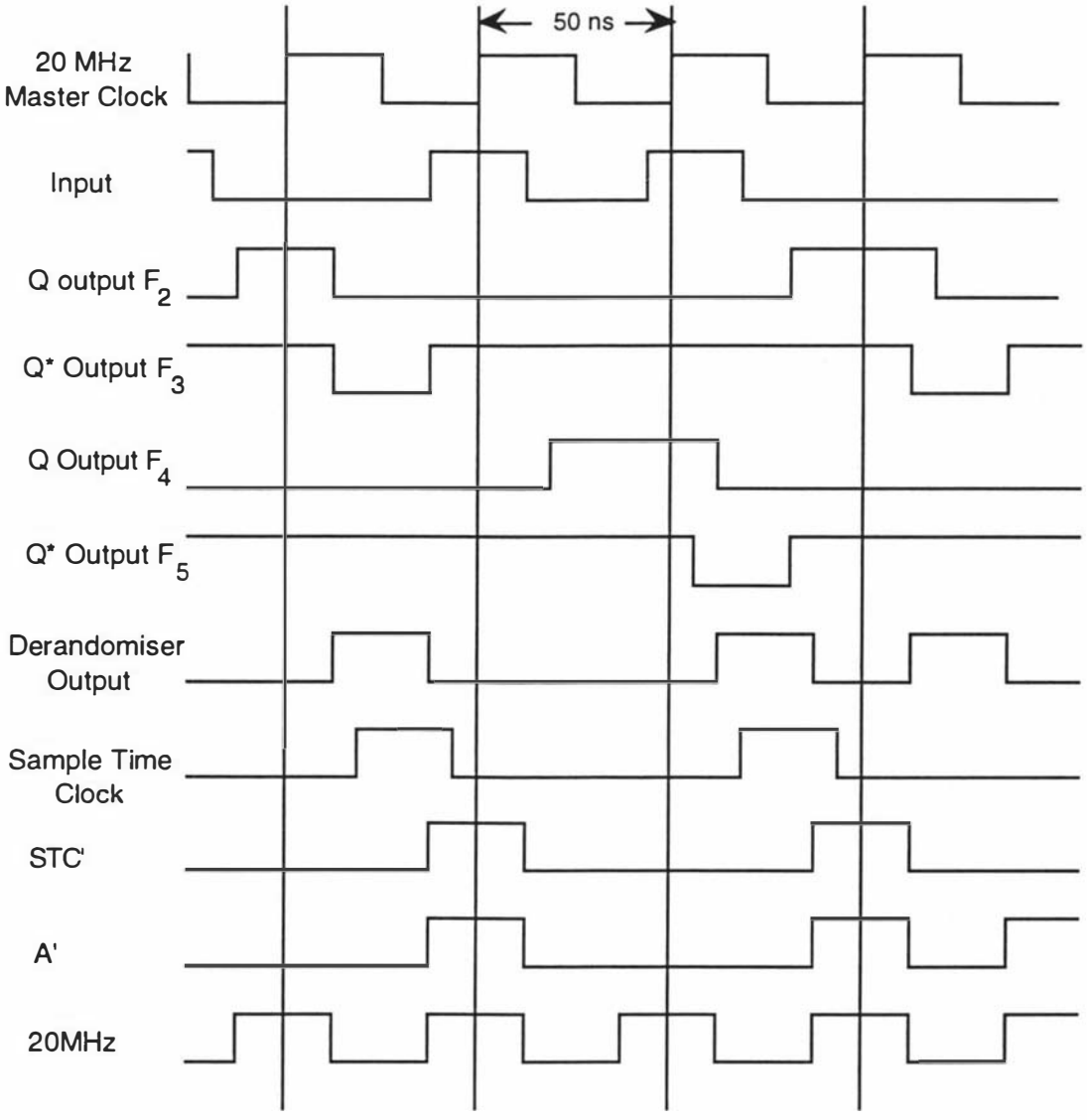


Figure 5.11 Pulse sequences for the derandomiser and the synchroniser (O'Driscoll, 1982)

5.2.3 Synchronous cascade counter

Six 4-Bit Binary Counters (74F163) are used to form a synchronous multistage counter system as shown in Figure 5.12. The scheme allows the formation of a cascaded counter system. The active rising edge of the pulses synchronously operates the counting. Three Octal D Flip-Flops (74F374) are used to form the latch system. The two sections of the device are controlled independently by the Clock (CP) and the Output Enable (OE*) control pins.

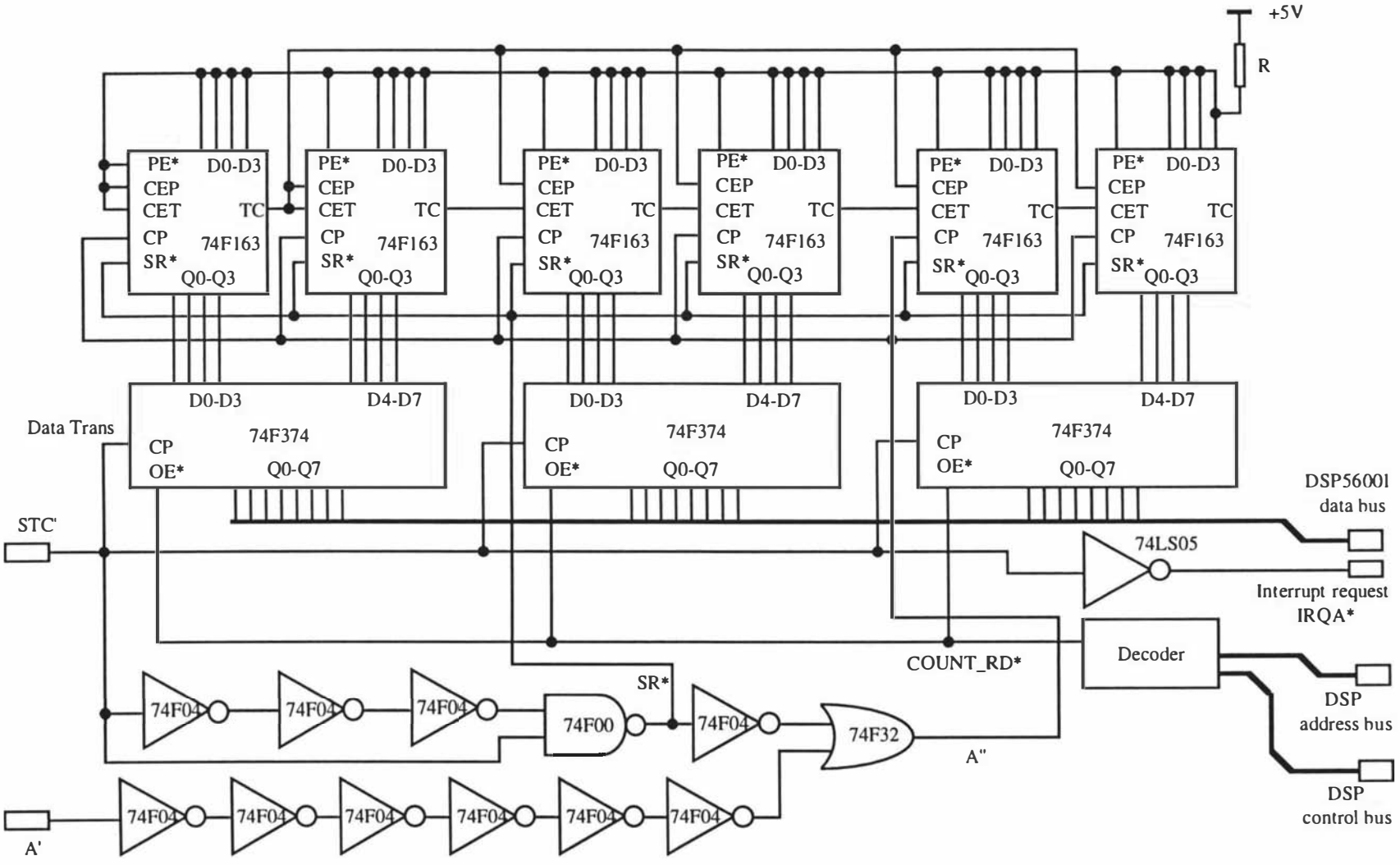


Figure 5.12 Counter circuit diagram

Typical pulse sequences of the counter system are shown in Figure 5.13.

1. When the Low-to-High transition of the STC' occurs, the state of each D input of the Octal D Flip-Flop (74F374), (previous counting data), is transferred to the flip-flop's Q output. They are held there until the next Low-to-High edge of the STC' occurs.

2. Then the counter gives the signal IRQA*, a negative edge interrupt request signal to the DSP which causes the DSP to jump into a fast interrupt service routine.

3. After the SR* signal is sent to the Synchronous Reset Input (SR*) of the Binary Counter (74F163), the next positive edge of the CP will reset the counters.

4. An equivalent six-gate delay ensures that the input sample A' will appear at the counter after the counters are reset.

5. Finally the DSP interrupt service transfers the data held in the Q output of the Octal D Flip-Flop (74F374) to the DSP's data memory. Data are read by the DSP once each STC' cycle.

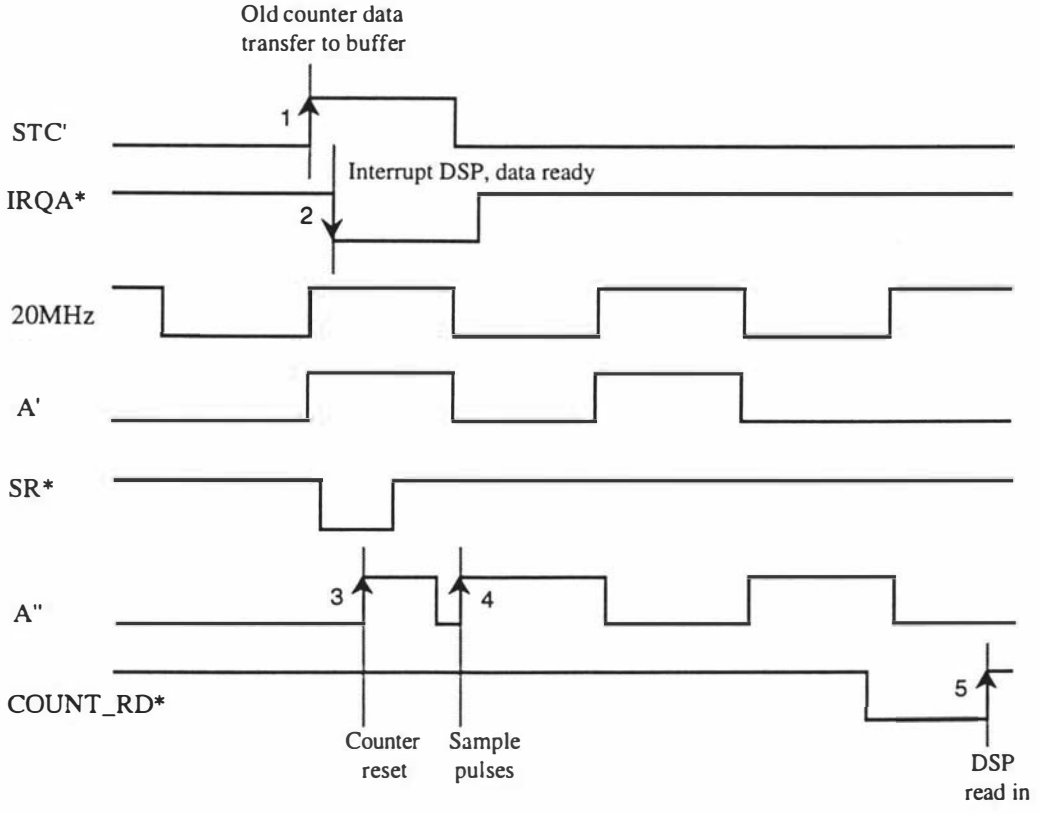


Figure 5.13 Typical pulse sequences of the counter system

5.3 Single DSP Board

The DSP board includes a 33MHz 24-bit General Purpose Digital Signal Processor (DSP56001RC33), an 8K × 24-bit fast static RAM (MCM56824) with 30ns access time and a 4K × 8 bit EPROM (NMC2764) with 150ns access time, as shown in Figure 5.14. The buffer system is used for data transfer between the counters and the DSP56001. Also on the board is the computer to DSP interface. This will be described in section 5.5. The host bus which is used for DSP to DSP communication will be described in section 5.4.

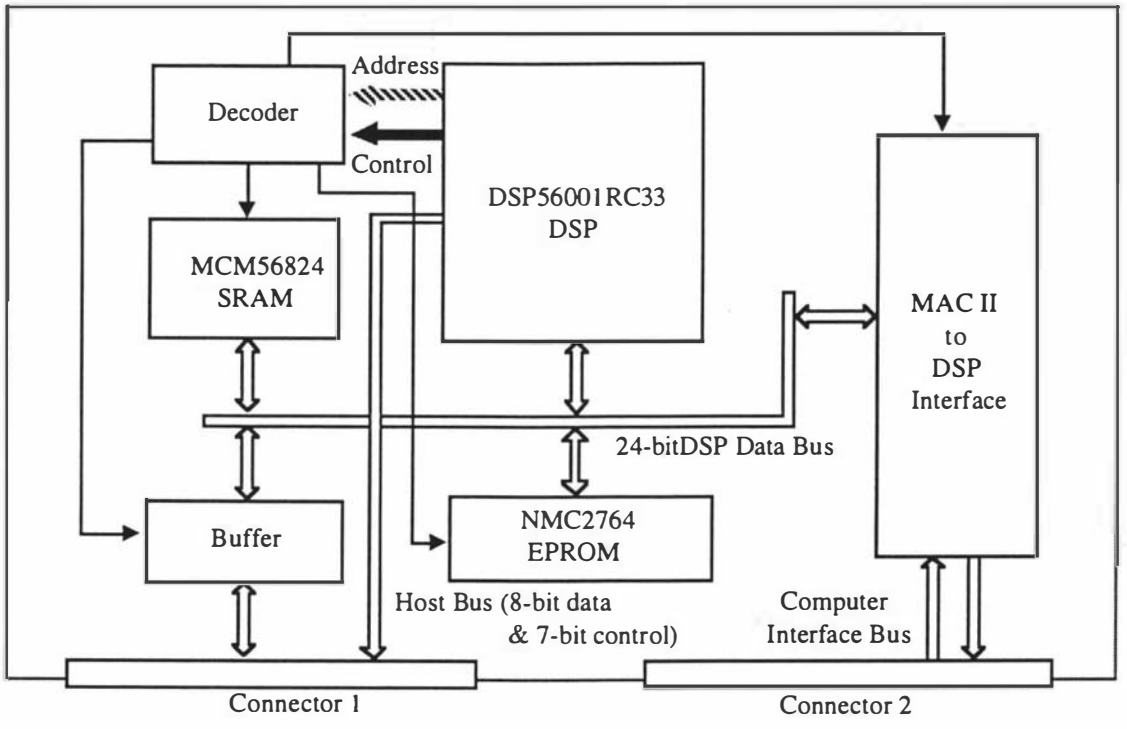


Figure 5.14 DSP board block diagram

5.3.1 DSP56001 data bus and access timing

The DSP board data bus (Figure 5.15) is built around the DSP56001 Port A, which is divided into three functional groups: a 24-bit wide data bus, a separate 16-bit address bus and a control bus. The 24-bit bidirectional data bus is used for external memory and I/O access.

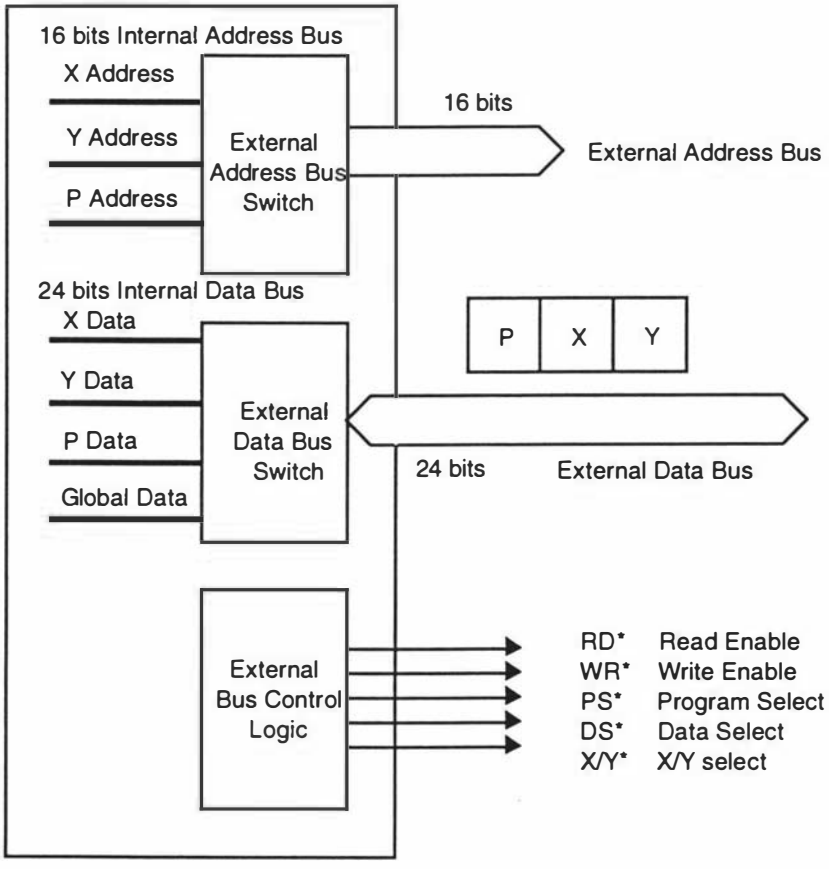


Figure 5.15 Data bus signal

Three address buses (XAB, YAB, and PAB) and four data buses (XDB, YDB, PDB, and GDB) are available for internal memory accesses during one instruction cycle, but only one address bus and one data bus are available for external memory accesses. If all memory sources are internal to the DSP, one or more of the three memory sources may be accessed in one instruction cycle. However, when one or more of the memories are external to the DSP56001, memory references may require additional instruction cycles. So during the programming period, care must be taken to use internal memory for critical data and programs. For accessing slow I/O and memory, an internal wait state generator can be programmed to insert up to 15 wait states (Motorola, 1990).

The DSP56001 external bus timing is defined by the operation of the address bus, data bus, and bus control pins. The transfer of data over the external data bus is synchronous with the clock. Timing is relative to the edges of an external clock, as illustrated in Figure 5.16. This timing is essential for the design of the single DSP board and synchronous multiprocessor systems. One instruction cycle takes two clock cycles or four clock phases. The clock phases, which are numbered T0-T3, are used for timing on the DSP56001.

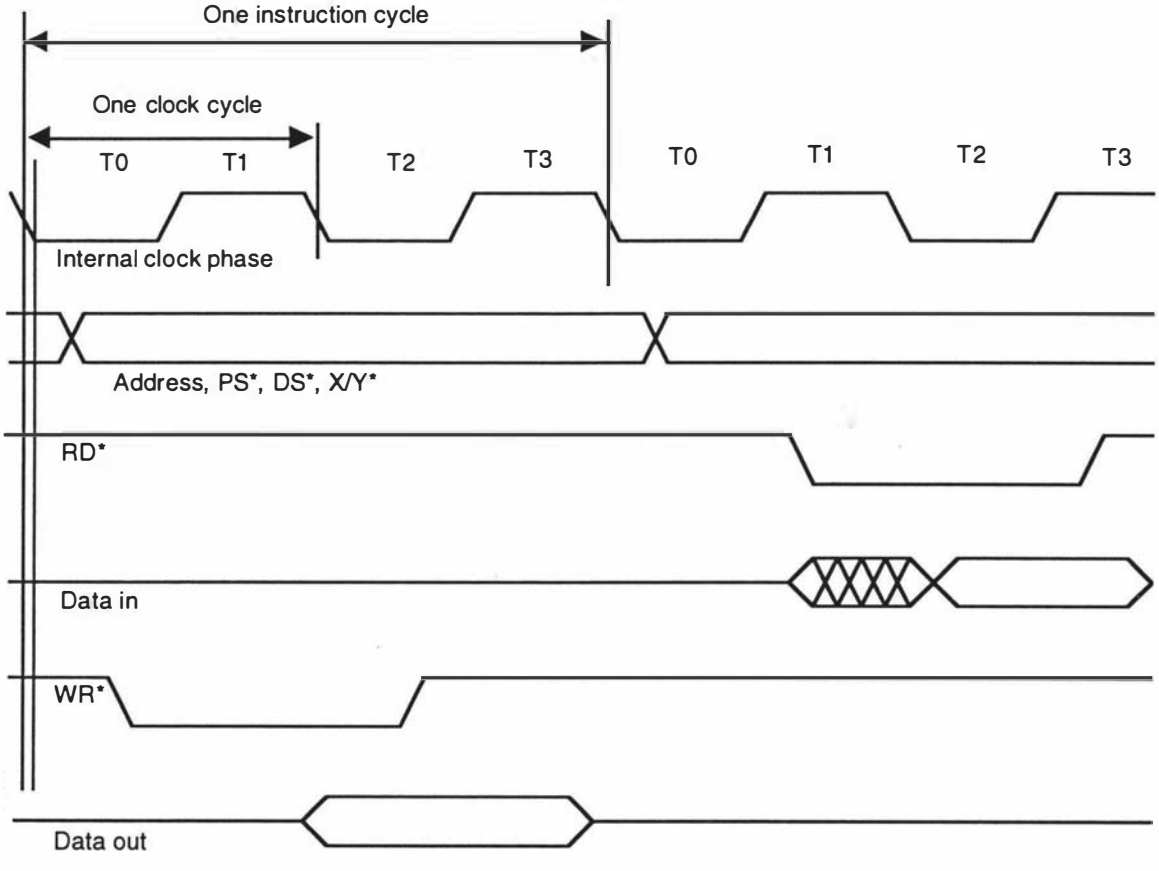


Figure 5.16 PORT A bus operation (from DSP56001 data sheet)

5.3.2 Mixed speed system

The DSP board is a mixed speed system. Figure 5.17 shows the system used for combining different memory speeds and memory-map peripherals in different address spaces. The internal memory and the external MCM56824 SRAM require no wait states. The NMC2764 EPROM memory uses six wait states, and the counter uses 1 wait state. Ten wait states are used in the computer interface system.

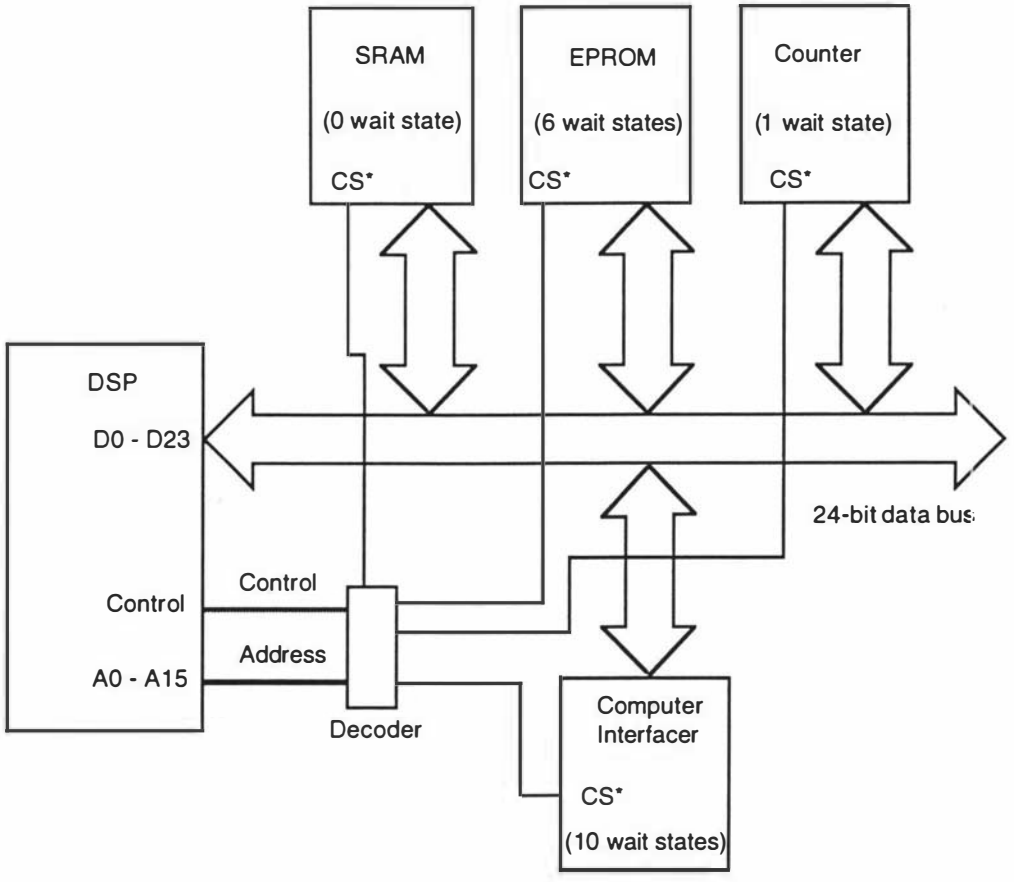


Figure 5.17 Mixed speed system

The following procedure is required to enable the DSP to access the external SRAM, EPROM and memory mapped I/O (counter, sample time clock, and computer interface).

The external memory address is defined by the address bus (A0-A15) and the memory reference selects (PS*, DS*, and X/Y*). These signals change in the first phase (T0) of the bus cycle. Since the memory reference select signals have the same timing as the address bus, they are used as additional address lines in the correlator circuits to generate chip-select signals for the appropriate devices. These chip-select signals change the memory chips from low-power standby mode to active mode and begin the read access time.

When the address and memory reference signals are stable, data transfer is enabled by read enable (RD*) or write enable (WR*). RD* or WR* is asserted to "qualify" the address and memory reference signals as stable and to perform the read or write data transfer. RD* and WR* are asserted in the second phase of the bus cycle (if there are no wait states). Read enable is typically connected to the output enable of the memory chips and simply controls the output buffers of the chip-selected memory. Write enable is

connected to the write enable (WE) or write strobe (WS) of the memory chips and is the pulse that strobes data into the selected memory.

For a read operation, RD* is asserted and WR* remains deasserted. Since write enable remains negated, a memory read operation is performed. The DSP data bus becomes an input, and the memory data bus becomes an output. For a write operation, WR* is asserted and RD* remains deasserted. Since the read enable remains deasserted, the memory chip OUTPUT remains in the high-impedance state. This state assures that the DSP and the chip-selected memory chips are not enabled onto the bus at the same time. The DSP data bus becomes an output, and the memory data bus becomes an input.

When RD* or WR* are deasserted at the start of T3 in a bus cycle, the data are latched in the destination device—ie., when RD* is deasserted, the DSP latches the data internally; when WR* is deasserted, the external memory latches the data on the positive-going edge. The address signals remain stable until the first phase of the next external bus cycle. The memory reference signals (PS*, DS*, and X/Y*) are deasserted (held high) during periods of no bus activity.

During the slow EPROM and I/O access period, a few wait states are added to the PORT A access time by using the DSP bus control register (BCR). The write signal is also delayed from the T1 to the T2 state when one or more wait states are added to ease interfacing to the port.

5.3.3 DSP board memory system

The DSP56001 design provides a low parts-count configuration with fast or slow memory, and memory mapped devices. Figure 5.18 shows the memory map of the DSP system.

The memory of the DSP can be partitioned in several ways to provide high-speed parallel operation and additional off-chip memory expansion. Program and data memory are separate, and the data memory is, in turn, divided into two separate memory spaces, X and Y. Both the program and data memories can be expanded off-chip. There are also two on-chip data read-only memories (ROMs) that can overlay a portion of the X and Y data memories and a bootstrap ROM that can overlay part of the program random-access memory (RAM).

The three independent memory spaces of the DSP (X data, Y data, and program spaces) are configured by control bits in the operating mode register (OMR). The operating mode control bits (MA and MB) in the OMR control the program memory map and select the

reset vector address. The data internal ROM enable (DE) bit in the OMR controls the X and Y data memory maps and enables/disables the internal X and Y data ROMs.

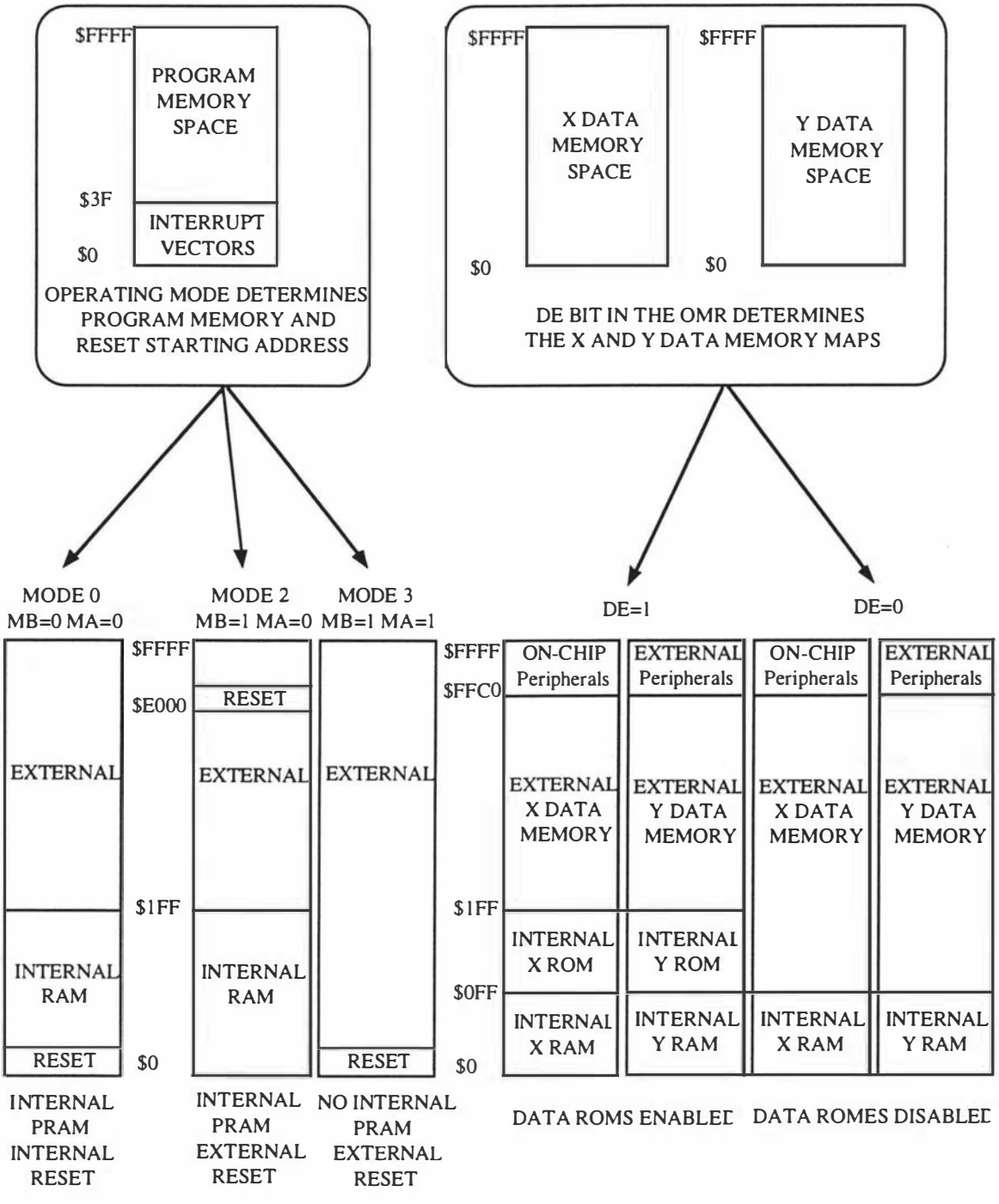


Figure 5.18 DSP56001 memory map (from DSP56001 data book)

The on-chip X data RAM is a 24-bit-wide, internal, static memory occupying the lowest 256 locations (0-255) in the X memory space. The on-chip X data ROM occupies locations 256-511 in the X data memory space and is controlled by the DE bit in the OMR. The on-chip peripheral registers occupy the top 64 locations of the X data memory

($FFC0-$FFFF). The 16-bit addresses are received from the X address bus, and 24-bit data transfers to the data arithmic logic unit (ALU) register occur on the X data bus.

The DSP board is controlled by a monitor program and the user processing program. The monitor program is stored permanently in a 64K by 8-bit EPROM and the user processing program is loaded into the external SRAM from the Macintosh IIvx computer. When the DSP is reset, it first goes into the bootstrap state and loads the monitor program into its internal RAM from the EPROM. It then passes control to the monitor program. Under monitor control the user processing program is loaded from the computer through the DSP-computer interface to the external SRAM.

The monitor program continuously polls its computer port address until it receives a request from the computer. The monitor program performs the basic tasks necessary to transfer memory and register data to and from the computer, it also loads the processing program from the computer, and passes control to user programs.

The NMC27C64 EPROM incorporates a transparent quartz window allowing the user to erase the programmed bit pattern by exposing the device to ultraviolet light. A new program can then be written into the device. It has 15 address inputs, 8 tri-state data outputs, a chip select input (CS*), and an output enable input (OE*).

The chip select (CS*) is the primary control signal that enables a read or write access, the output enable (OE*) signal controls the enabling of data output buffers. The EPROM OE* is permanently connected to digital ground in this design, since the EPROM memory is only valid for a read operation. This simplifies the design significantly. Both the chip enable CE* and output enable OE* control signals of EPROM memory interface must be asserted to obtain valid data at the outputs. The CE* controls the device selection, and when it is negated, puts the EPROM in standby mode. The ROM_RD* signal is derived from the address decoder and is connected to CE* on the EPROM.

Special provisions have been made to allow the DSP to load a program from the EPROM into internal program memory during a powerup reset. On powerup, the wait-state generator adds 15 wait states to all external memory accesses so that slow memory can be used. Bit 23 of external memory is set to logic one, so the DSP will load the contents of external EPROM into internal program memory (if bit 23 is a logic zero, it will load from the host port). The bootstrap program uses the bytes in three consecutive memory locations in the external EPROM to build a single word in the internal program memory. Figure 5.19 shows the system that enables internal program memory to be loaded from an external EPROM during powerup. The base address of the EPROM is $C000.

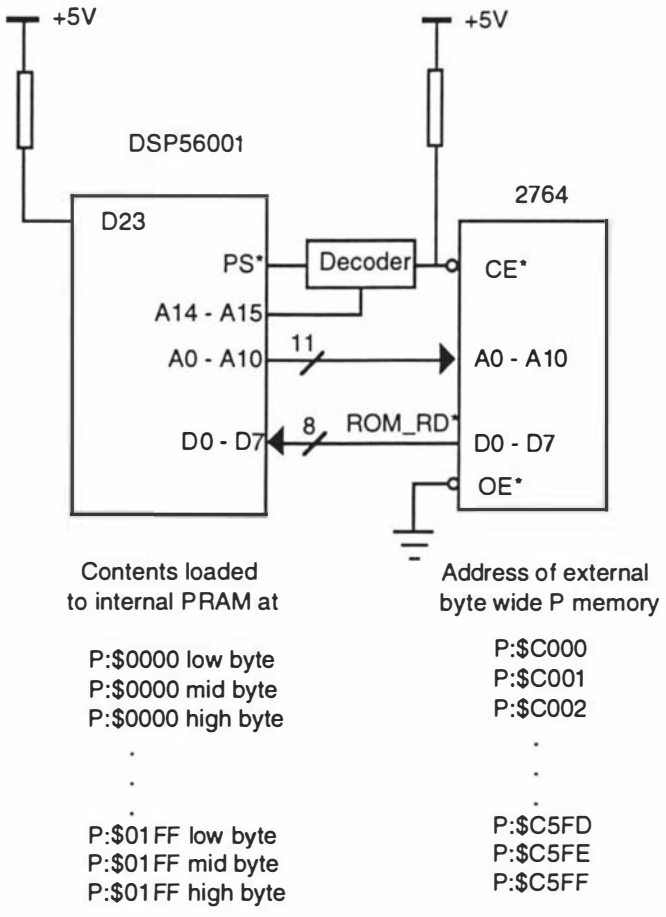


Figure 5.19 EPROM Bootstrap Circuit

The MCM56824 is a CMOS static random access memory which integrates an 8K × 24-bit core with multiple chip enable inputs, output enable, and an externally controlled single address pin multiplexer. These functions allow for simple connection to the DSP56001.

Figure 5.20 shows the external SRAM system. The PS* signal is used to select the A11 and V/S* inputs (vector/scalar address multiplexer control) to enable the program memory at the appropriate time. The X/Y* input is used to select two data memories X and Y. The three external memory spaces (program, X data, and Y data) reside in the same physical device.

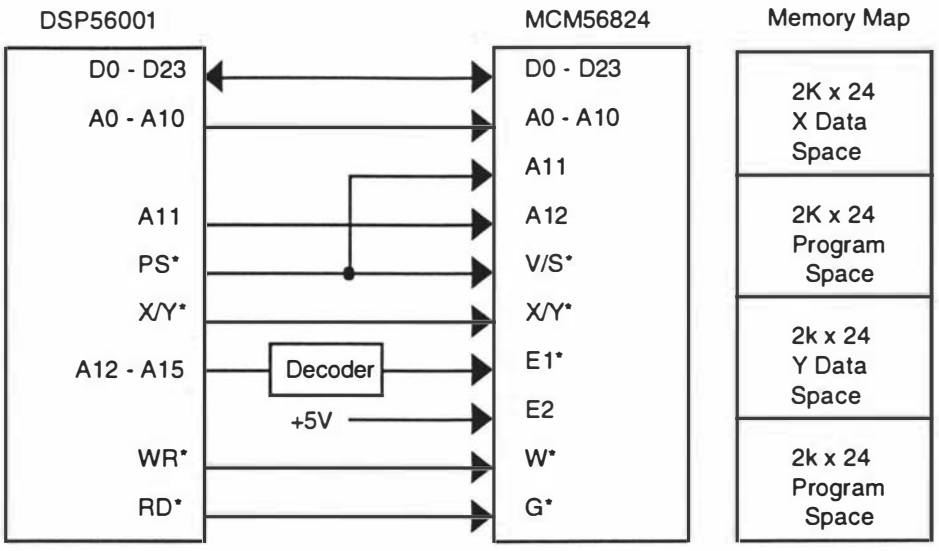


Figure 5.20 DSP56001 and MCM56824

Figure 5.21 shows the MCM56824 SRAM timing when connected to a DSP56001.

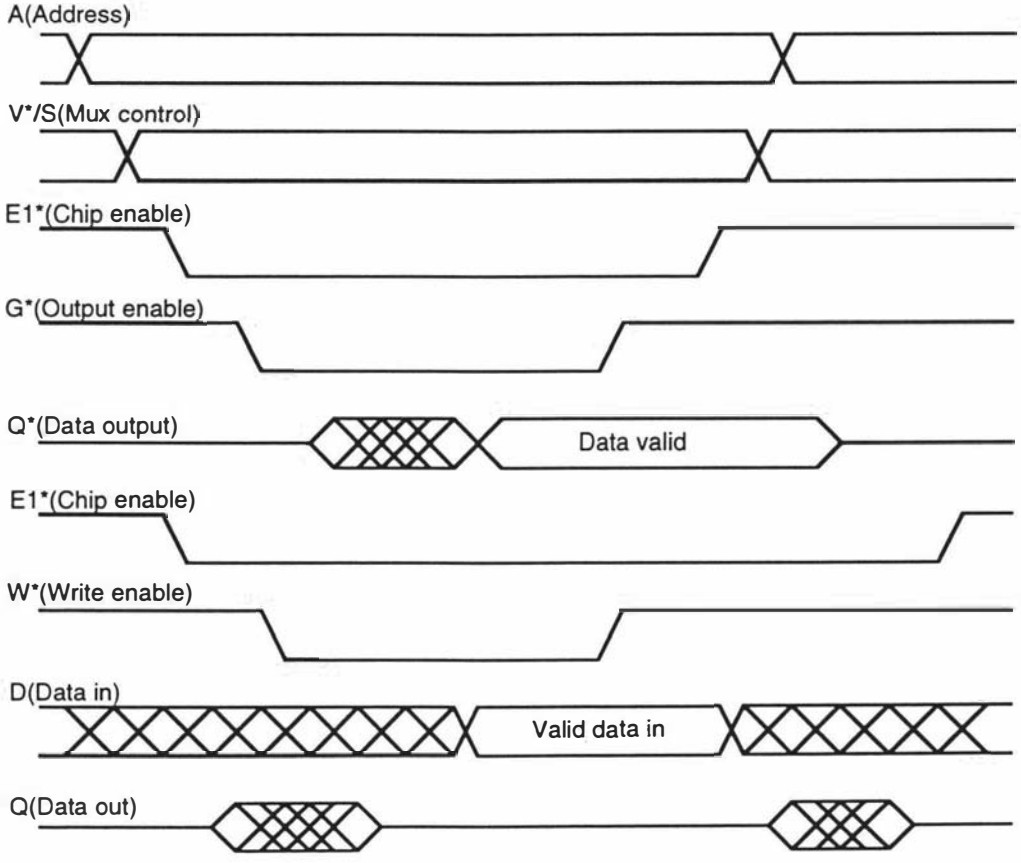


Figure 5.21 MCM56824 SRAM timing (Motorola, 1992)

The circuit diagram of the external EPROM and RAM connected to PORT A of the DSP is shown in Figure 5.22.

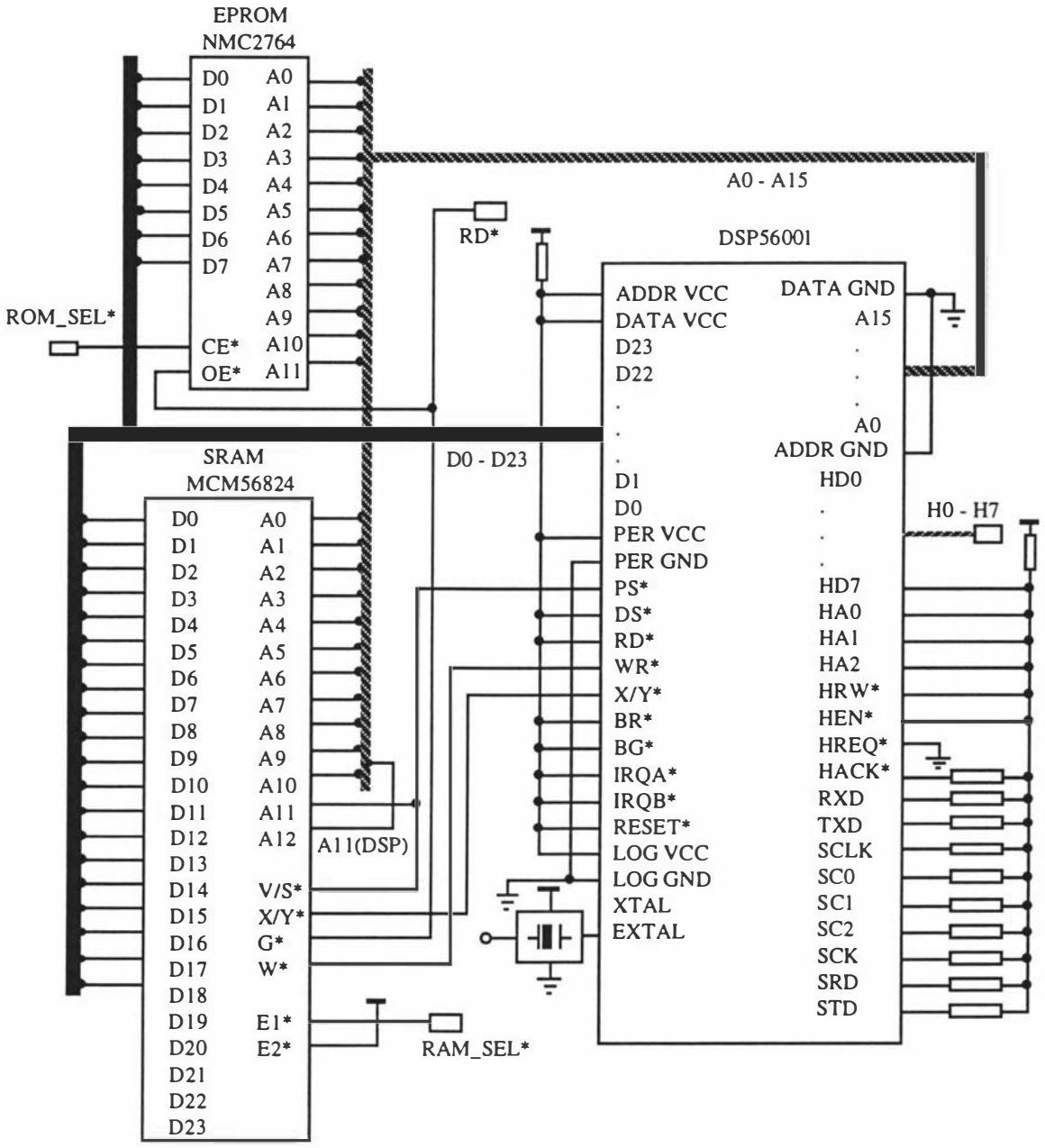


Figure 5.22 DSP board memory circuit

Because floating inputs cause erroneous operation and draw excessive power, all unused input pins are pulled up or down by a resistor. Without pullup resistors, the DS* and other chip select signals may become active, causing two or more memory chips to try to simultaneously drive the external data bus, which can damage the memory chips. A pullup resistor in the 50KΩ range is sufficient.

5.3.4 Decoder system

The purpose of the address decoder is to enable the device that is to be accessed. The first step in designing an address decoder is to define the memory map. As shown in Figure 5.23, an organised memory map has separate areas of address space assigned to memory type devices and peripheral I/O controllers.

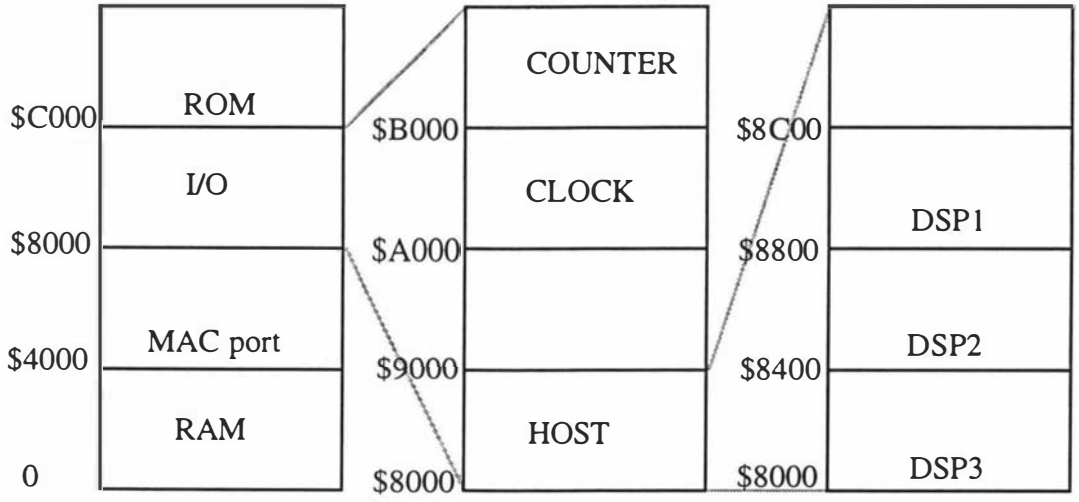


Figure 5.23 DSP board memory map

The read/write controls of the DSP56001 are self-descriptive. They can be used as decoded read and write controls, the write signal (WR*) can be used as the read/write control, and the read signal (RD*) can be used as an output enable (or data enable) control for the memory. Decoding in this fashion simplifies connection to the high-speed SRAM. The program memory select (PS*), data memory select (DS*), and X/Y select (X/Y*) can be considered additional address signals. The X/Y* output signal has the same timing as the address lines.

Since external logic delay is large relative to the timing margins of devices, timing becomes more difficult when different speed devices are introduced. The separate read and write strobes used by the DSP56001 are mutually exclusive, with a guard time between them to avoid two data buffers being enabled simultaneously.

The decoding circuit is shown in Figure 5.24. Four 1-of-4 Decoders (74F139) are used for device selection.

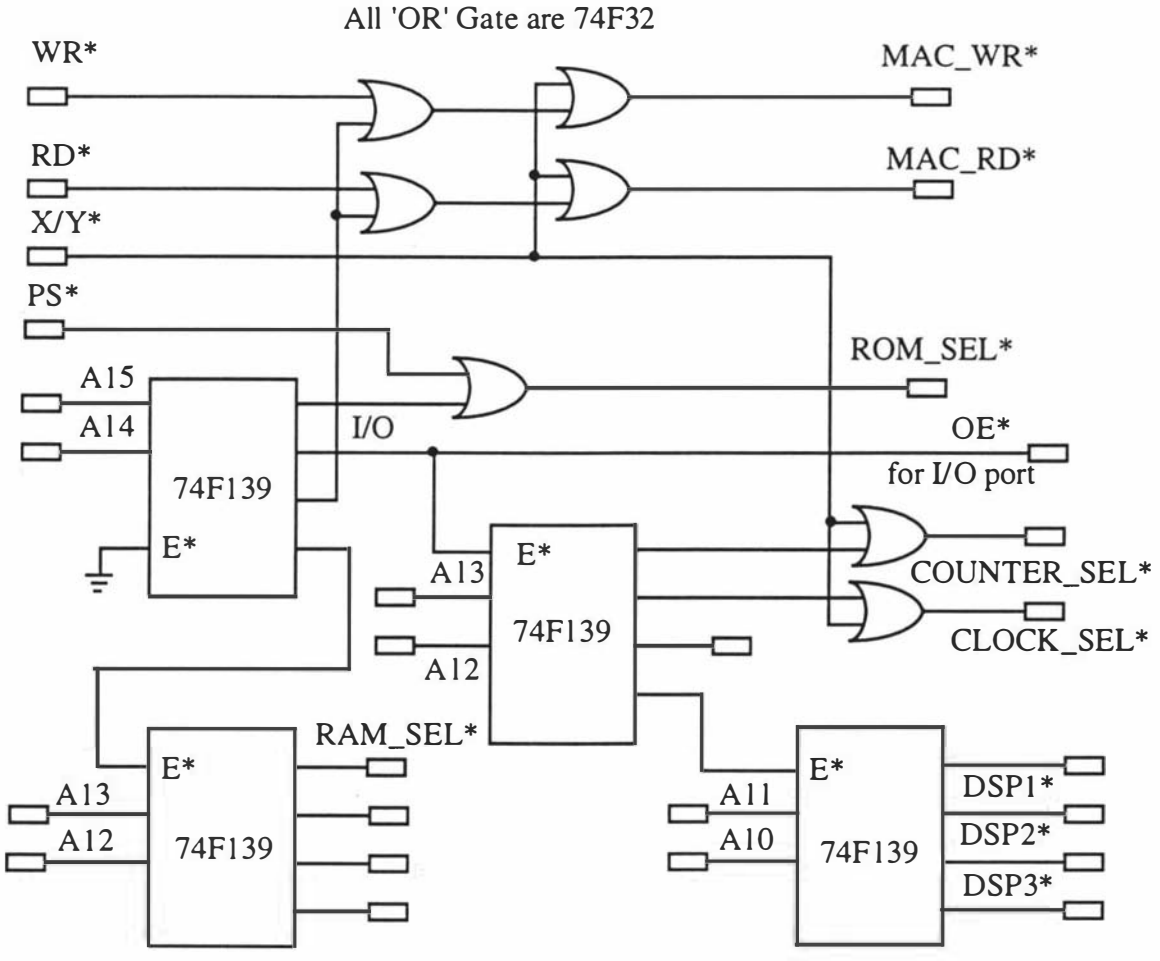


Figure 5.24 Decoding circuit

ROM_SEL* is used to select the EPROM at address P:$C000, OE* is used to select I/O buffer at address X:$8000, COUNTER_SEL* is used to select the counter board at address Y:$B000, CLOCK_SEL* is used to select the sample time clock board at address Y:$A000, RAM* is used to select MCM56824 SRAM at address $3000, DSP1*, DSP2*, and DSP3* are used to select DSP board 1, DSP board 2, and DSP board 3 respectively, at addresses Y:$8000, Y:$8400, and Y:$8800.

5.3.5 Clock signal

The DSP56001 requires a TTL compatible clock input. The clock input must meet the specified rise and fall times, as well as the limits on low and high width times. The rise and fall time of this external clock should be 5 ns maximum. Since the rise and fall times are small, the clock interface circuit should be located close to the DSP56001 in order to eliminate the possibility of any reflections or ringing of the clock signal.

In the system, an external 33 MHz clock module is used. The externally supplied square wave is connected to the EXTAL pin of the DSP, as shown in Figure 5.25.

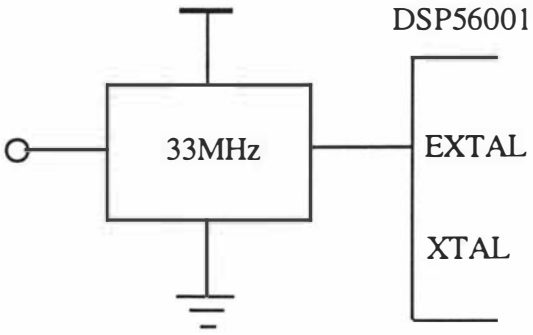


Figure 5.25 External clock

5.3.6 Reset operation, external interrupt and mode select circuit

When the RESET* signal line is asserted by external logic, the DSP and its peripherals are reset. This operation is necessary at power up time. After powerup, RESET* has to be asserted for a minimum of 100 milliseconds.

The MODA and MODB inputs are read and internally latched in the DSP when the processor exits the RESET state. Therefore these two pins should be forced into the proper state during reset. After leaving the RESET state, the MODA and MODB pins automatically change to external interrupt requests IRQA* and IRQB*. Table 5.1 shows the mode assignments. The special bootstrap mode (Mode 1) is chosen as the initial working mode for the correlator system,

Operating Mode	MODB	MODA	Description
0	0	0	Single Chip Mode
1	0	1	Special Bootstrap Mode
2	1	0	Normal Expanded Mode
3	1	1	Development Mode

Table 5.1 Initial DSP56001 operating mode

The mode select circuit is shown in Figure 5.26.

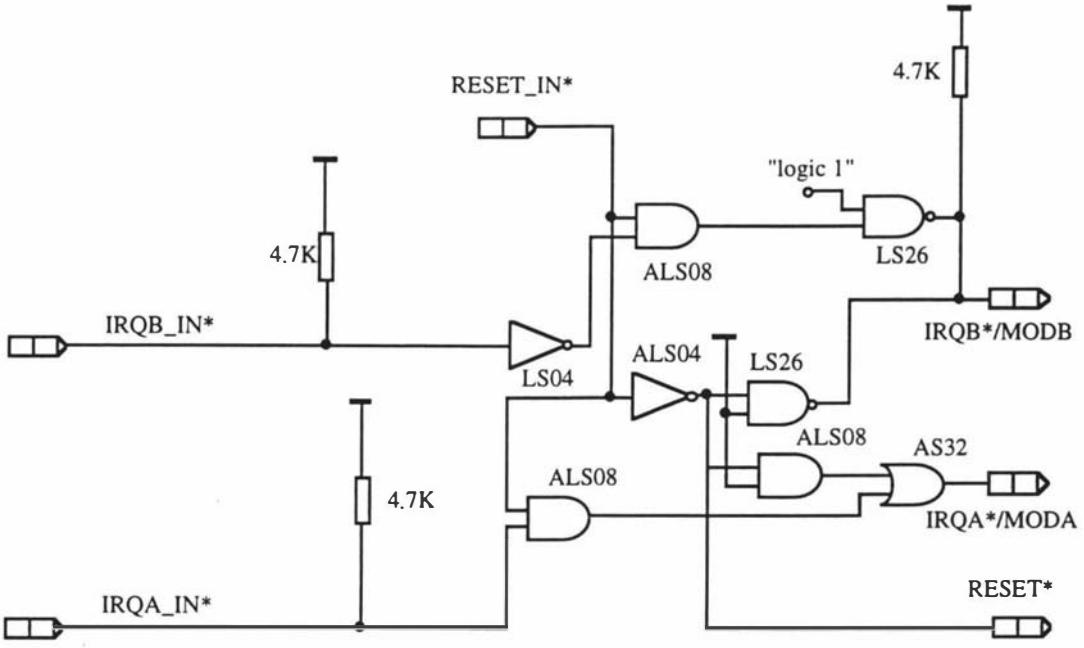


Figure 5.26 Mode select circuit

The circuit ensures that, following a reset, MODA will be set to logic "1" and MODB to logic "0".

5.3.7 DC electrical characteristics and signal buffering

An important DC electrical parameter of the DSP56001 is the maximum current that its output drivers can sink. We have to choose the appropriate logic family and decide which output signals require buffers, based on the fanout and circuit speed.

The data and signal buffers are shown in Figure 5.27. Octal Transceivers (74F245) were chosen as buffers. The tri-state outputs are capable of sinking 60 mA and sourcing 15 mA, producing very good capacitive drive characteristics. Address and control signals are buffered by a Two-Input OR Gate (74F32), which has a 20 mA low level output current.

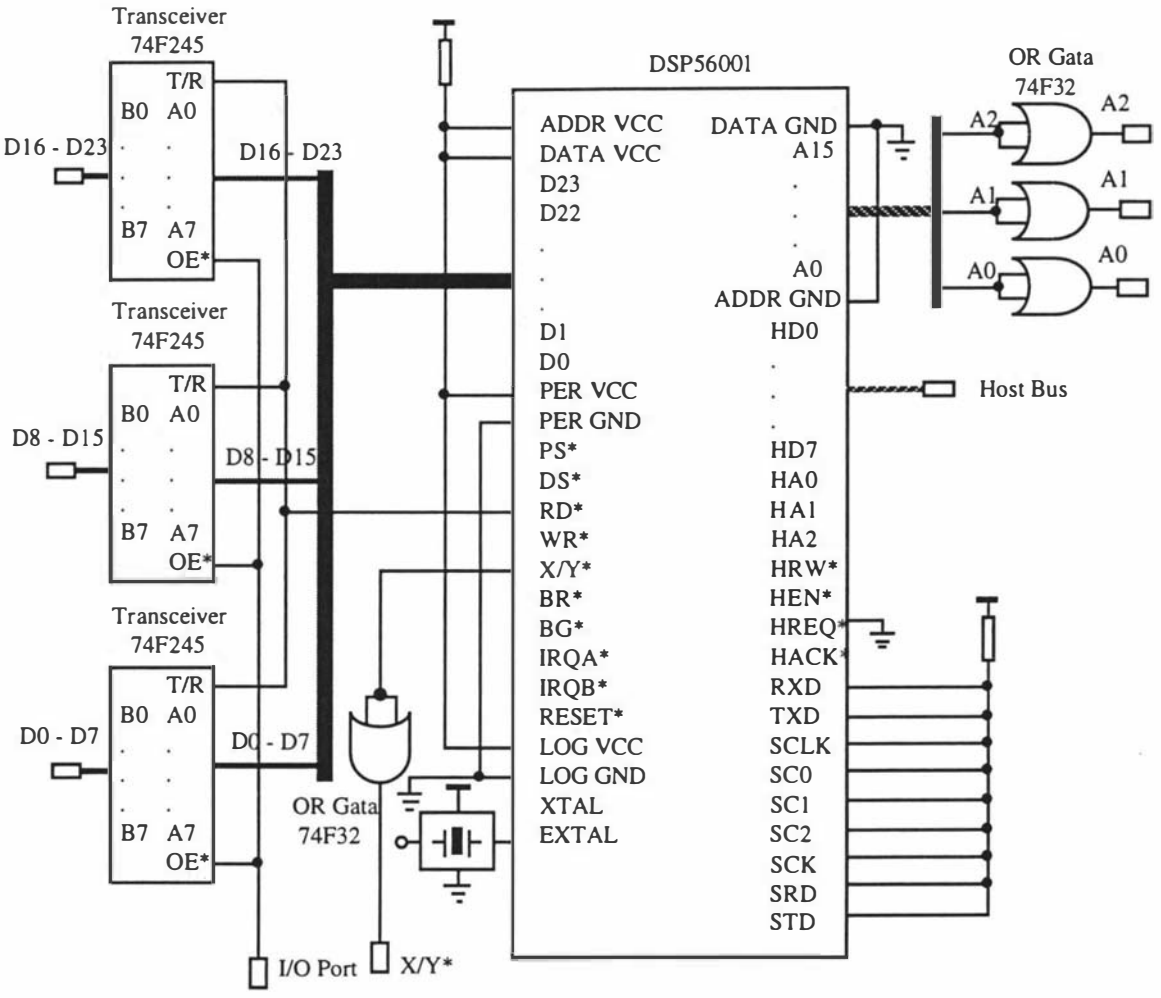


Figure 5.27 Data and signal buffers

5.4 The Correlator-Computer Interface

5.4.0 Introduction

The control computer requires an interface to communicate with the correlator. In selecting the appropriate interface, two factors need to be considered. The first factor is speed. The real time requirement of the correlator system demands rapid data transfer between the correlator and computer. This can be achieved using a parallel interface to connect the correlator directly to the Macintosh NuBus. The second factor is generality. Generality is an important requirement in modern electronic instrument design. The Macintosh IIvx has been chosen as the controlling computer. However one may also wish the correlator to function with a different computer, and so a standard parallel interface was chosen to build the interface system.

The control computer sends commands and the processing program to the correlator and requests data from the correlator. The correlator is connected to the computer interface card by a 50-pin ribbon cable.

The interface system consists of a interface card, an interface cable, two interface programs (the computer interface program which resides in the personal computer and the correlator monitor program which is executed by the DSP), and the correlator circuits associated with the computer interface card, as shown in Figure 5.28.

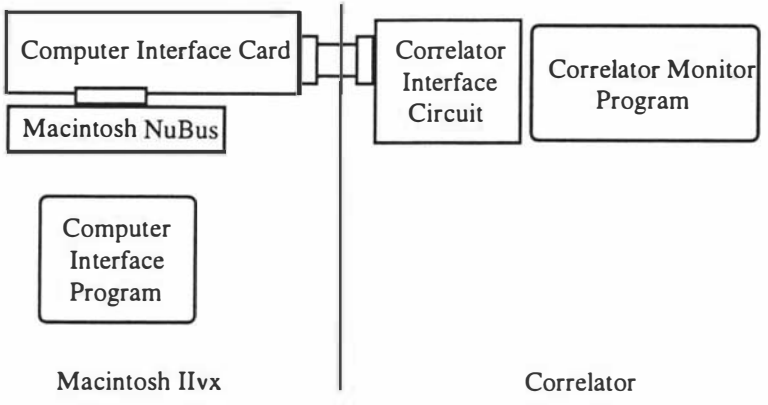


Figure 5.28 Correlator-Computer Interface System

5.4.1 The computer Interface Card

5.4.1.1 Macintosh NuBus

The correlator is directly connected to the NuBus 96-pin Euro-DIN connector through a data I/O board. The intent of NuBus is to implement a 32-bit high speed interface that

uses a very simple and direct design methodology. It runs at a transfer rate of 10 MHz and is a synchronous bus. All signal lines on the NuBus are active low signals.

The NuBus lines are as follows:

AD0/through AD31/ These 32 tri-state, active low address and data lines carry both address and data information. They are multiplexed to minimise the number of lines. Address information is first broadcast. The responding slave then may place 8,16, or 32 bits of data on these lines, depending on the type of transfer being performed.

TM0/ and TM1/ These two tri-state, active low transfer mode lines are used to indicate the type of transfer being performed. The master initiating the transfer indicates the type of transfer to be performed and the responding slave indicates the success of the transfer over these lines.

ACK/ The acknowledge signal is an active low, tristate line used to indicate the completion of the data bus operation. ACK/ and START/, when used together, can signal that attention cycles have occurred on the bus.

START/ This active low, tristate signal is driven low to indicate the beginning of an operation. ACK/ and START/, when used together, can signal an occurrence attention cycles on the bus.

NMRQ/ The active low non master request line is used by boards not capable of becoming bus masters but which are in need of service.

5.4.1.2 The digital I/O card

The digital I/O card (NB-DIO-24) is a National Instruments 24-bit parallel digital I/O interface for Macintosh NuBus computers. All signals are connected to an I/O connector on the end of the board which makes cabling uncluttered and easy to install. An Parallel Peripheral Interface (PPI) (AMD8255A) controls the 24-bit digital I/O. The PPI can operate in either a unidirectional or bidirectional mode, can generate interrupt requests, and can be programmed for 8-bit or 16-bit operation.

The key functional components of the hardware are illustrated in the block diagram of Figure 5.29. The digital I/O port on the NB-DIO-24 board contains three ports of eight digital lines each. These ports are labelled PA, PB, and PC. The eight digital lines making up Port PA are labelled PA7 to PA0. These digital I/O are controlled by the Parallel Peripheral Interface chip. Port A and Port B can be used for both latched (handshaking) and non latched (non-handshaking) modes. Port C can be used for non

latched mode only. The digital lines making up Port C are used as handshaking lines for both Ports A and B whenever either is programmed for latched mode.

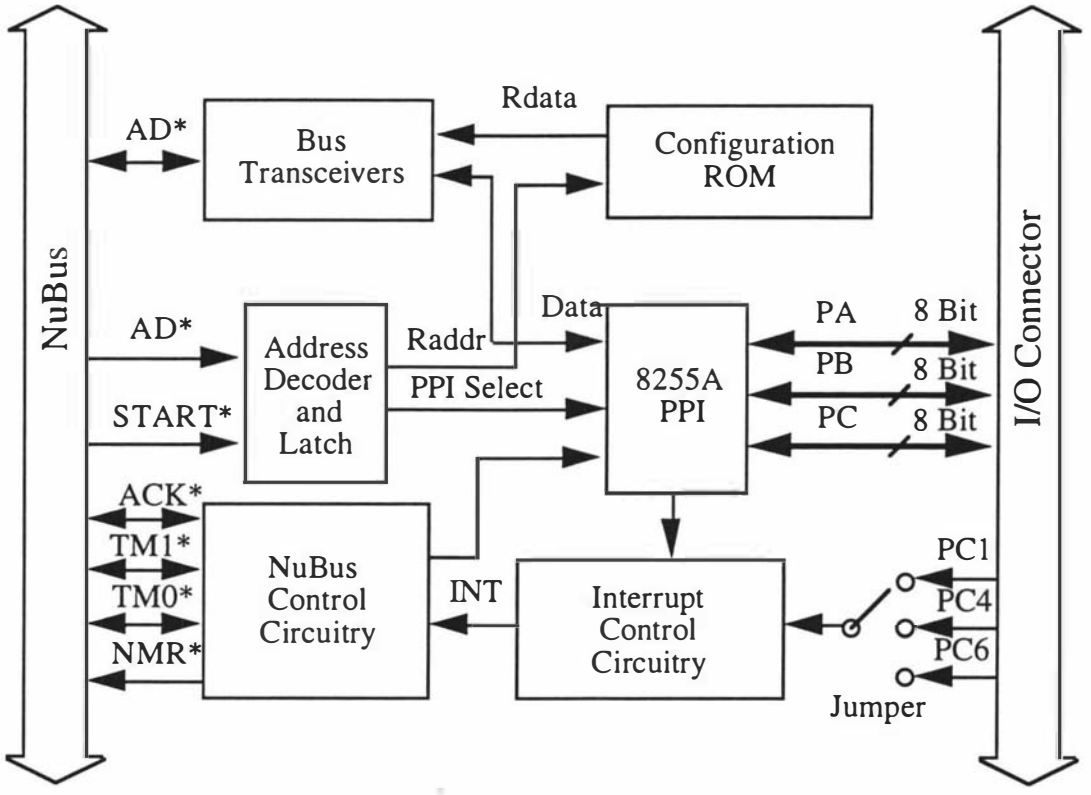


Figure 5.29 NB-DIO-24 Block Diagram

The digital I/O card comes with a software C package (NB-DAQ) for the control computer. This package has a library of functions that can be called from an application program written in C to control the I/O card. National Instruments maintains a consistent hardware structure and software interface among different versions of I/O card and software, so one can switch between platform computers with minimal modifications to the system. The NB-DIO-24 digital I/O card can also be used with LabVIEW, a software system that features interactive graphics, and a powerful graphical programming language.

5.4.1.3 Signal specification of the I/O card

The I/O card was programmed for Mode 0 operation. Mode 0 provides the following features. Two 8-bit ports (A and B) and two 4-bit ports (upper and lower part of Port C). Any port can be input or output, outputs are latched, inputs are not latched.

All digital I/O is transmitted through a standard 50-pin connector. All even pins on this connector are connected to logic ground, and pin 49 is connected to +5 VDC, which is often required to operate I/O module mounting racks.

5.4.2 Correlator interface logic and circuits

The correlator interface circuits are built on the DSP board, and are used to generate handshake signals. The I/O board signal lines comprise four parts: address, control, byte data and handshake.

There are three output control bits in the handshake. These bits are DSP56B_INT, DSP56B_REQ, and DSP56B_ACK. DSP56B_REQ and DSP56B_ACK act as handshake lines for reading and writing data. DSP56B_INT acts as a flag to indicate whether a DSP board is requesting computer service. These three control bits are enabled when the computer selects the DSP board.

There are three input control bits in the handshake. These bits are MAC_ACK, MAC_REQ, and INT_ACK. MAC_ACK and MAC_REQ are handshake inputs from the computer for reading and writing data. INT_ACK informs the monitor that the computer has received its service request and is ready to communicate. Figure 5.30 shows the handshake signals between the computer and the correlator for data transfer to the correlator.

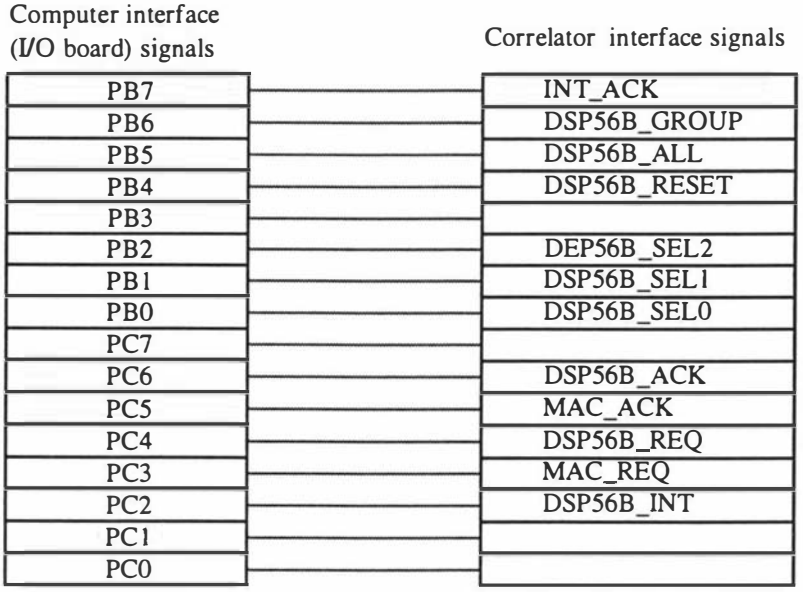


Figure 5.30 Handshake signals

Two handshake signals originate at the computer I/O card and are used to pass data to and from a DSP board. MAC_REQ initiates a data byte transfer to a DSP board while MAC_ACK acknowledges receipt of a data byte from a DSP board. The data signal lines between computer I/O card and the DSP board form an eight-bit bi-directional data bus. Since the DSP56001 has a 24-bit data bus, all data are passed to the high order byte first (bits 23-16), the middle order byte second (bits 15-8), and the low order byte third (bits 7-0). Figure 5.31 shows the circuit diagram of the correlator interface.

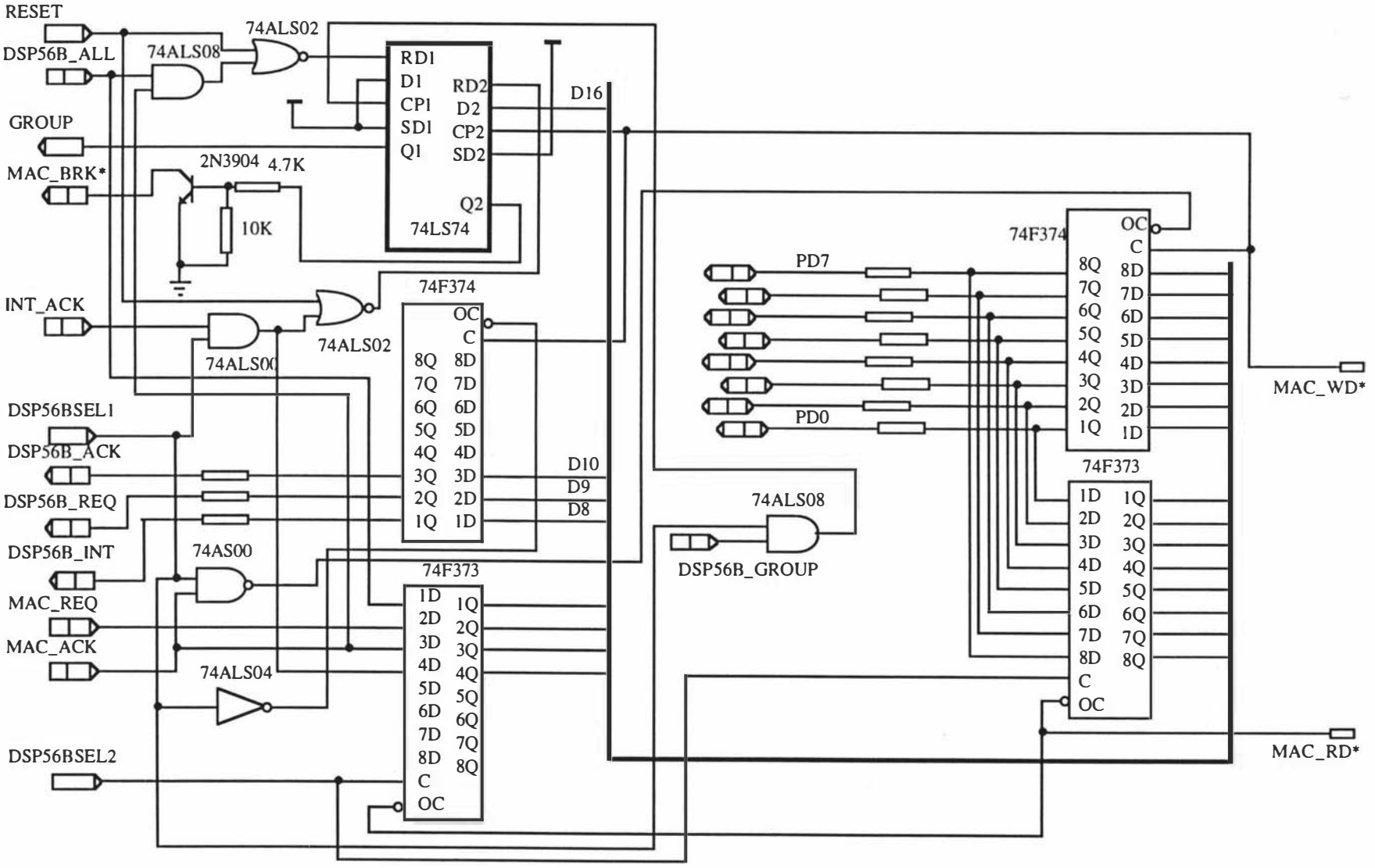


Figure 5.31 Correlator-computer interface circuit

Two Octal Transparent Latchs (74F373) are used for data transfer from computer to correlator, two Octal D Flip-Flops (74F374) are used for data transfer from correlator to computer, and data transfer is controlled by MAC_WR* and MAC_RD*. The open collector output of the transister (2N3904) generates the MAC_BRK* signal to interrupt the computer if it is needed. DSP56BSEL1 and DSP56BSEL2 (generated from address select circuit) are used to enable the interface circuit.

5.4.3 Communication protocol

All data transfers to and from the Macintosh IIvx use a fixed protocol format. A description of the handshake timing associated with command and data transfers is presented in this section, together with the various types of command.

5.4.3.1 Command packet description

There are two types of commands recognised by the DSP board monitor program. Each command type has a different packet length. Observing or modifying memory or register values requires a memory command packet while program control transfer from the monitor to a user program requires a function command packet.

A memory command packet consists of five bytes. The five bytes received are in the following order: command, 16-bit start-address high byte, 16-bit start-address low byte, 16-bit count-value high byte, and 16-bit count-value low byte.

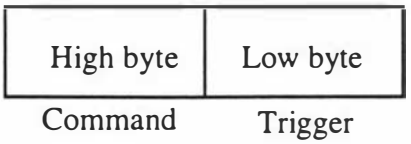
Byte	High byte	Low byte	High byte	Low byte
Command	Start address		Count value	

Memory Command Types

PMEM READ = $14
PMEM WRITE = $15
XMEM READ = $16
XMEM WRITE = $17
YMEM READ = $18
YMEM WRITE = $19

A memory command may be either read or write. Register values pertinent to a user's program also reside in memory. References to registers are references to a particular location in the development mode RAM map of the monitor.

A function command packet may be one or two bytes long.



Function Command Types

GO_IMMEDIATE = $10
GO_DELAYED = $11 + trigger

If a single DSP board is addressed, the function command is one byte (GO_IMMED). If a group of DSP boards are to be addressed simultaneously the function command is two bytes (GO_DELAYED). The second byte of the function command is a trigger byte ($55) which informs all the DSP board monitors in the group to transfer program control to their respective user program simultaneously. This trigger byte does not appear until all DSP boards in the group have been armed with the GO_DELAYED command.

5.4.3.2 Data packet description

All data transfers occur when a memory command is received by a DSP board. The DSP 24-bit data word value is divided into three 8 bytes which are received and transmitted in the given order: high-order byte (bits 23-16), middle-order byte (bits 15-8), and low-order byte (bits 7-0).

5.4.3.3 Hand shake description

The computer program and the DSP board monitor program use handshake lines to initiate and acknowledge transfer of each data byte.

Figure 5.32 illustrates the timing sequence for a data write from the computer and a data read by a DSP board. The DSP board polls the MAC-Port address for a change in the MAC_REQ line. The MAC-PORT address resides at program memory address $4000 of the DSP memory map. This address requires ten wait states whenever it is accessed. The DSP Y data address $4000 (MAC-Port) is arranged such that bits 0-7 are used for 8-bit data transfers, bits 8-15 are used for data transfer control and bit 16 is used for Macintosh IIvx service requests.

Following is a brief summary of the timing events:

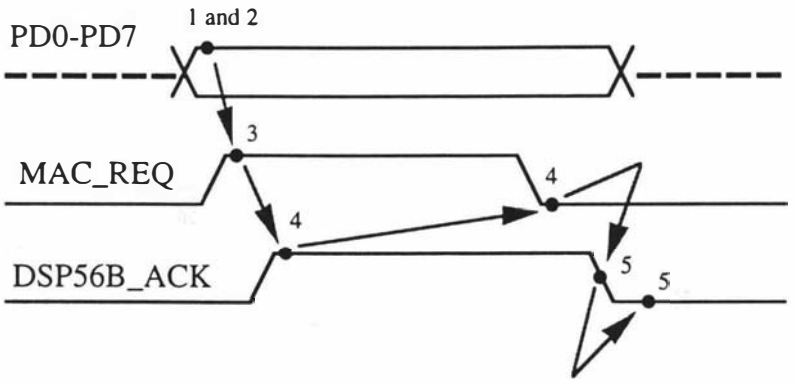


Figure 5.32 The time sequence for a data transfer from computer to correlator

1. Macintosh IIvx places data byte on data bus latch.
2. Macintosh IIvx places DSP board address on control latch.
3. Macintosh IIvx asserts MAC_REQ line and begins timeout for response from DSP board.
4. DSP board detects a logic one on MAC-Port Address bit 9 and asserts MAC-Port Address bit 10 (DSP56B_ACK). The Macintosh IIvx Data Bus will become valid at this time. Macintosh IIvx de-asserts MAC_REQ and begins timeout for DSP56B_ACK deassert.
5. DSP board reads data byte. DSP board de-asserts DSP56B_ACK. Macintosh IIvx recognises DSP56B_ACK de-assert and ends cycle.

The following control signals remain deactivated during this handshake sequence: DSP56B_INT, DSP56B_REQ, MAC_ACK, MAC_BRK, DSP56B_BRK, DSP56B_RESET.

Figure 5.33 illustrates a data write from an DSP board and a data read by the Macintosh IIvx program. The Macintosh IIvx must have the DSP board address selected and be polling the DSP board DSP56B_REQ line for a change in state. Following is a brief summary of the timing events:

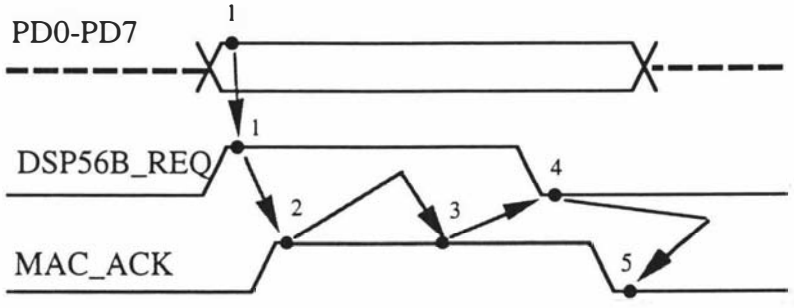


Figure 5.33 Time sequence of a data write from correlator to computer

1. DSP board places data byte on MAC-Port low-order byte and asserts DSP56B_REQ signal.
2. Macintosh IIvx detects a DSP56B_REQ, asserts MAC_ACK and starts timeout for DSP56B_REQ deassert. Macintosh IIvx Data Bus transparent latch is enabled.
3. Macintosh IIvx reads data byte.
4. DSP board de-asserts DSP56B_REQ and waits for MAC_ACK deassert.
5. Macintosh IIvx deasserts MAC_ACK signal. DSP board recognises MAC_ACK deassert and ends cycle.

The following control signals remain deactivated during this handshake sequence: DSP56B_INT, DSP56B_BRK, DSP56B_RESET, DSP56B_ACK, MAC_REQ, MAC_BRK.

5.4.4 Computer interface program

5.4.4.1 LabVIEW program

The acquisition program was created using National Instruments LabVIEW® 2 software. LabVIEW is an acronym for Laboratory Virtual Instruments Engineering Workbench. It is a data flow language, as distinct from control flow languages like FORTRAN or PASCAL. Creating an application in the LabVIEW environment is similar to designing a real electronic device. Computer programs or subprograms are constructed like instruments by building front panels on the screen with knobs, buttons, digital displays, and other controls and by making block diagrams. These virtual instruments (VI) are themselves building blocks for higher program structures. While dedicated for automated measurements, LabVIEW is a also general purpose language.

LabVIEW programs have three main parts: the front panel, the block diagram and the icon and connector. The block diagram is program source code. The front panel is interfaced to the program, and provides the means through which one interacts with it. The icon and connector together represent the program in a manner analogous to a subroutine call statement when the program is used as a sub-program in another LabVIEW program block diagram.

LabVIEW graphical programming relieves the user of writing debugging codes in conventional programming languages and replaces this with the simple and intuitive task of interconnecting functional blocks and control structures. The iconic presentation of blocks and structures and the familiar block diagram lay-out make the application almost self-explanatory with little need for additional documentation. The hierarchical nature of the virtual instruments concept allows complex systems to be divided into a set of more manageable sub-systems in a structured fashion.

The norm in LabVIEW is data flow control, where functions may be executed as soon as their data inputs become available. Therefore, function blocks with no data dependency are not guaranteed to execute in a given order. In situations where execution should follow a certain order in time, sequence structures should be used. Since a group of such structures are stacked to occupy the same area of the block diagram space, they are useful in the development of large applications. They help structure an application into a sequence of stages and allow stages to be easily added, removed, or relocated within the sequence.

Figure (5.34-35) contains some LabVIEW symbols and functions used later in this thesis.

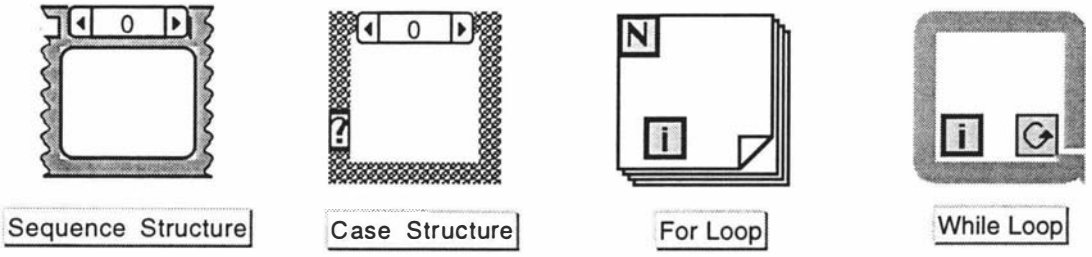


Figure 5.34 LabVIEW structure symbols

A Sequence Structure has multiple frames that execute in order starting from frame number 0. Each frame contains some program code. The sequence structure specifies the order of execution when there is no data dependency between parts of the program but the order of execution is important. A Case Structure has two or more cases that execute depending upon the value of the selection variable [?]. A For Loop executes its subdiagram [N] times (the value contained in the count terminal [N]), the iteration terminal [i] contains the current number of completed iterations. A While Loop executes computer code until a boolean value of a conditional terminal [⟳] is false. The iteration terminal [i] behaves exactly as it does in the For Loop.

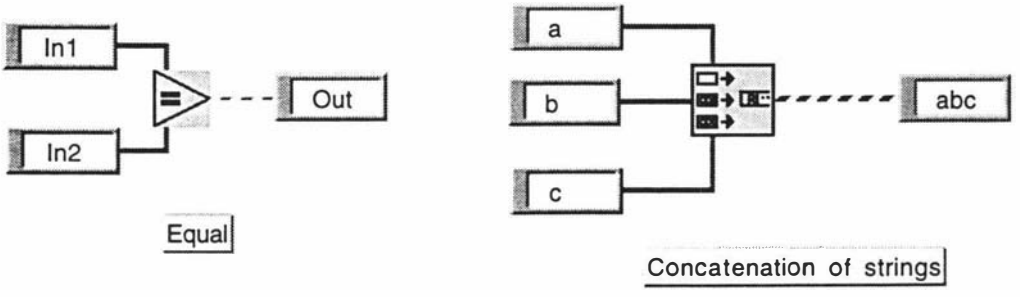


Figure 5.35 LabVIEW functions

5.4.4.2 Using the LabVIEW program in the correlator system

The autocorrelator computer interface software was developed on a Macintosh IIvx computer with 5 MB of memory running an operating system version 7. The software consists of a main virtual instruments control and a set of sub-virtual instruments for communicating with the autocorrelator, implementing the autocorrelator command language, and emulating autocorrelator functions for data acquisition, display, listing, storage and retrieval.

The entire computer control program is integrated in the single software front panel shown in Figure 5.36.

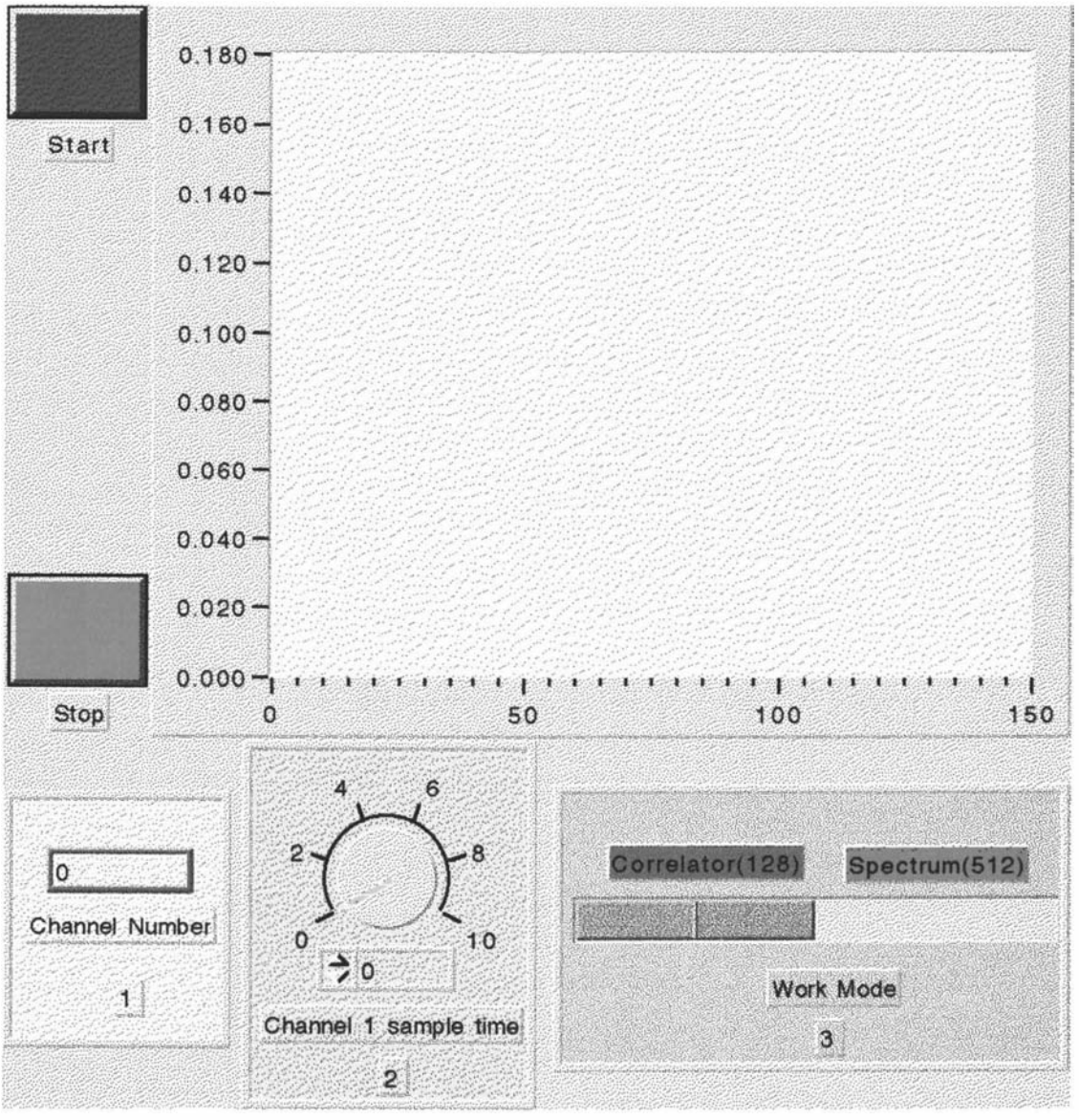


Figure 5.36 Computer monitor front control panel

The front panel serves as an interactive interface for supplying inputs to and observing outputs from the instrumentation system. A variety of controls can be conveniently located in palette menus. When all virtual instruments are complete, the correlator system can be controlled by the front panel which also provides real time feedback from the system.

All functions of the computer control and interface software are performed using a single top level LabVIEW virtual instrument. The instrument calls other sub-virtual instruments for communication with the autocorrelator, performing the various functions for data collection and display, and management of the resulting spectra.

After setting the front-panel controls to perform the desired acquisition, the **Start** switch is initiated, and the program performs the acquisition. The top level virtual instrument continuously reads the designated window of the data memory segment by calling a sub virtual instrument. Once data acquisition has started, the plots are repeatedly updated until the **STOP** switch is turned to off.

The top level virtual instruments of correlator consists of three main sequence structures with the first two used to set up the interface board and initialise the autocorrelator. The third structure contains the main infinite WHILE loop which continuously monitors front panel controls for user commands and input changes and updates the correlation function display. Eight sequence structures are used within the WHILE loop to perform these functions. Functional blocks were selected from palette menus and connected with wires to pass data from one block to the next.

LabVIEW has a group of digital I/O virtual instruments (VIs) which can be used in the LabVIEW environment to operate the digital I/O card (NB-DIO-24).

DIG_Ptr_Config Configures the specified port for direction (input or output) and handshake mode.

DIG_In_Port Reads digital data from the specified digital I/O port.

DIG_Out_Port Writes digital data to the specified digital I/O port.

DIG_In_Line Returns the digital logic state of the specified digital input line in the specified port.

DIG_Out_Line Sets or clears the specified digital output line in the specified port.

DIG_Ptr_Status Returns the handshake state of the specified port.

Some example LabVIEW programs will now be described. In the following example LabVIEW diagrams, virtual instruments with a graphic icons are standard LabVIEW virtual instruments used mainly for communications.

Example 1. Sending a command to the correlator.

The steps required to send a command to the correlator are as follows,

1 I/O board configuration (Figure 5.37)

The port 0 and port 1 of digital I/O board are configured for the command transfer to the autocorrelator. The port configure virtual instrument (PRT_CONFIG) can be used to configure the port working mode (ie. handshake mode or non handshake mode, input or output mode). Board number 5 represents that the I/O board is plugged into NuBus expansion slot No.5 in the Macintosh IIvx (available slot numbers are 4 through 6).

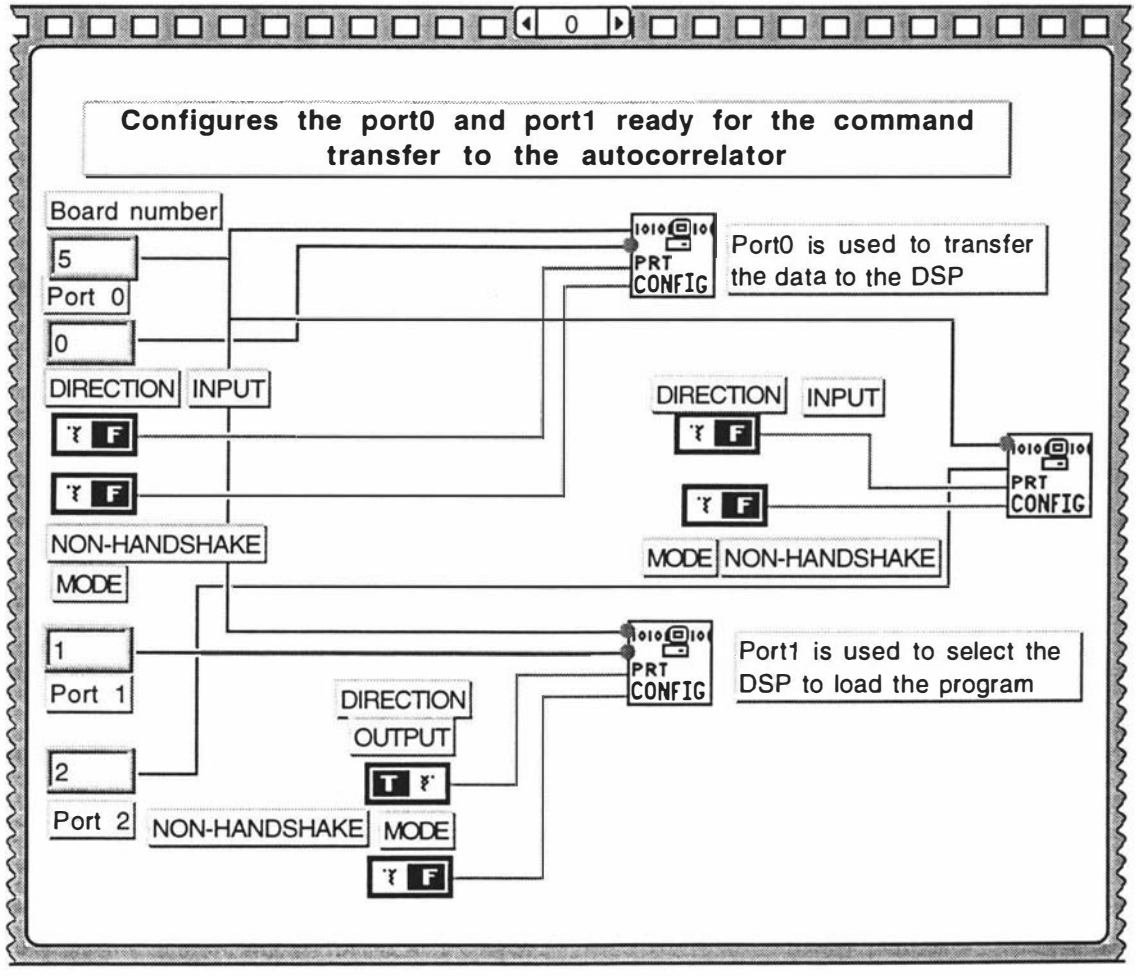


Figure 5.37 The LabVIEW program for the I/O card port configuration

2 Sending command (Figure 5.38)

The command is send to the correlator by the OUT PORT virtual instrument.

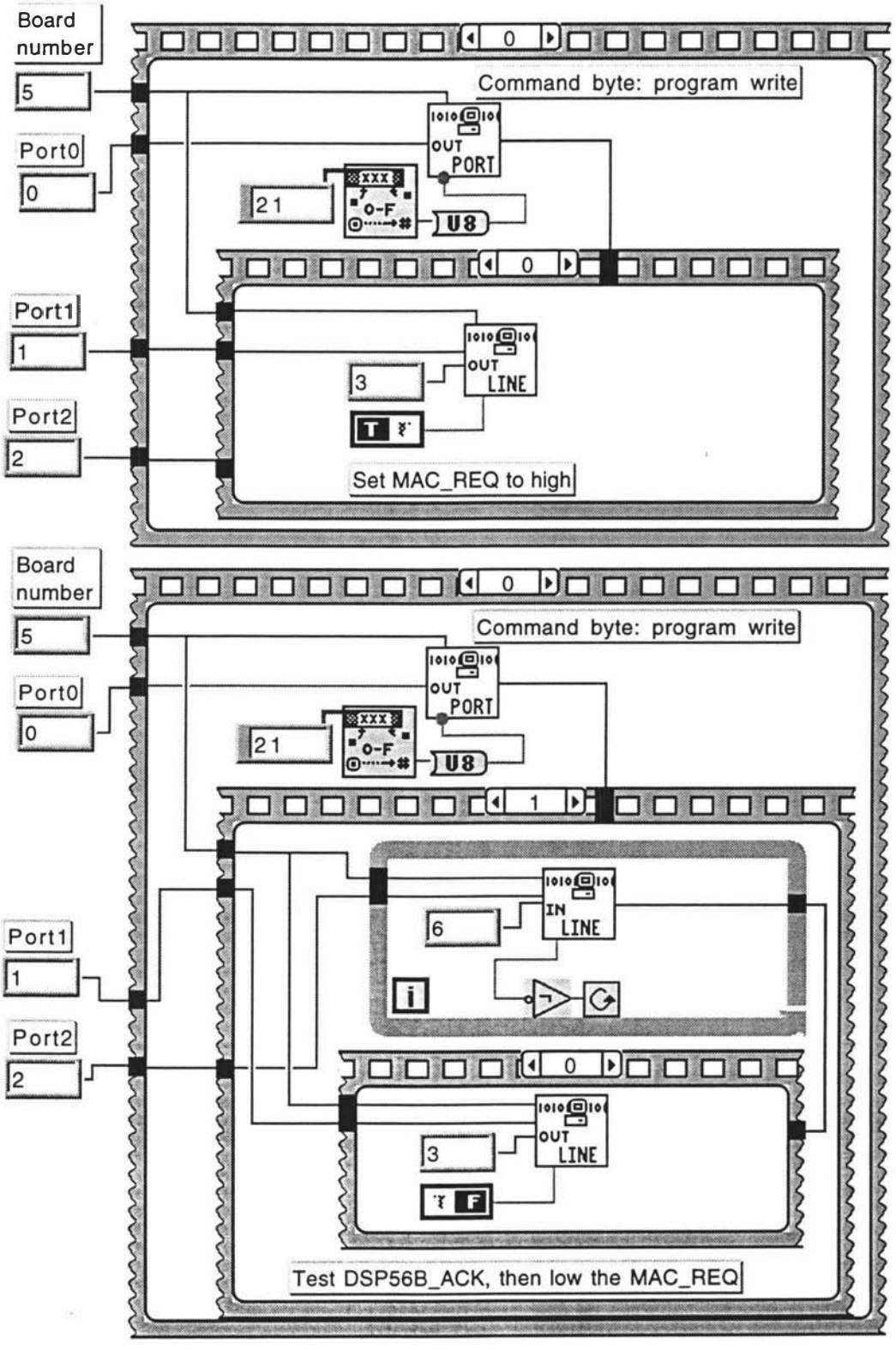


Figure 5.38 LabVIEW program: Send command to correlator

First, OUT PORT puts the command data to port 0 of I/O card, then the OUT LINE virtual instrument sets the MAC_REQ pin to logic "1". Following that, the IN LINE virtual instrument returns the digital logical state of DSP56B_ACK pin, and tests. Finally the OUT PORT resets MAC_REQ pin to logic "0" to complete the command transfer.

Example 2. VI example, Opening a program file.

Figure 5.39 is a LabVIEW screen showing the front panel of the Opening a program file virtual instrument which opens a program file for transferring to the autocorrelator.

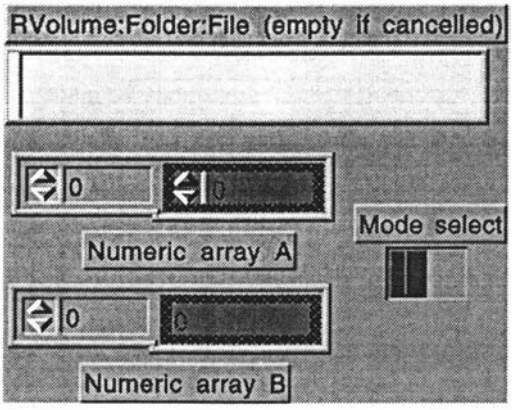


Figure 5.39 The front panel of the Open program file VI

The RVolume window is used to input the file name of the program to be transferred to the correlator. Numeric arrays A and B represent the format of the data file for transfer (A is input format and B output format). Mode select allows different files to be opened, so different programs can be transferred for operation in the correlator mode or spectrum analyser mode. The virtual instrument runs automatically when loaded and remains active until aborted manually by the user. Figure 5.40 is a LabVIEW print out of the program diagram of the virtual instrument used.

Example 3. Reading data from the correlator and displaying them.

Figure 5.41 is part of the print out of LabVIEW program for reading data from the correlator and displaying them.

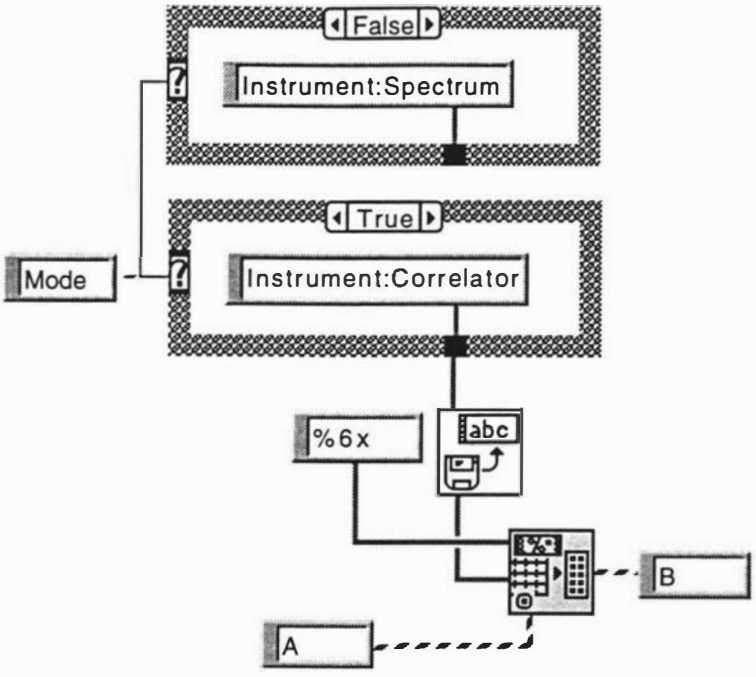


Figure 5.40 LabVIEW print out of the diagram of the VI

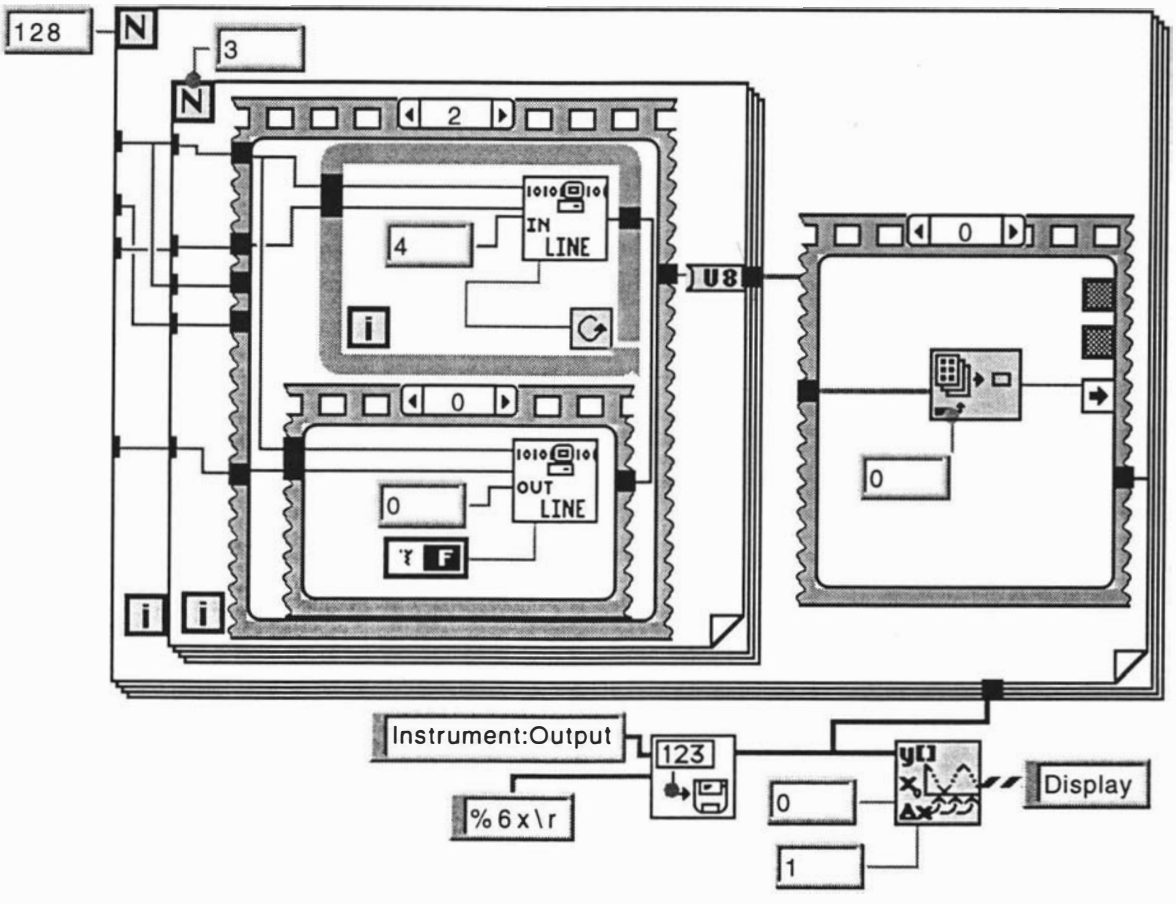


Figure 5.41 LabVIEW program for reading data from correlator and display them

Two For Loops are used in the program. The first For Loop repeats 3 times to obtain 3 bytes from the I/O card to form 24-bit data. The second For Loop repeats 128 times to

obtain data for 128 autocorrelation channels. The data are stored in a file under the name 'Output' in the folder 'Instrument', and then displayed on the computer monitor screen.

5.4.5 Correlator resident monitor program

The monitor EPROM address resides in the fixed external program address space of the DSP board at $C000 as illustrated in Figure 5.42.

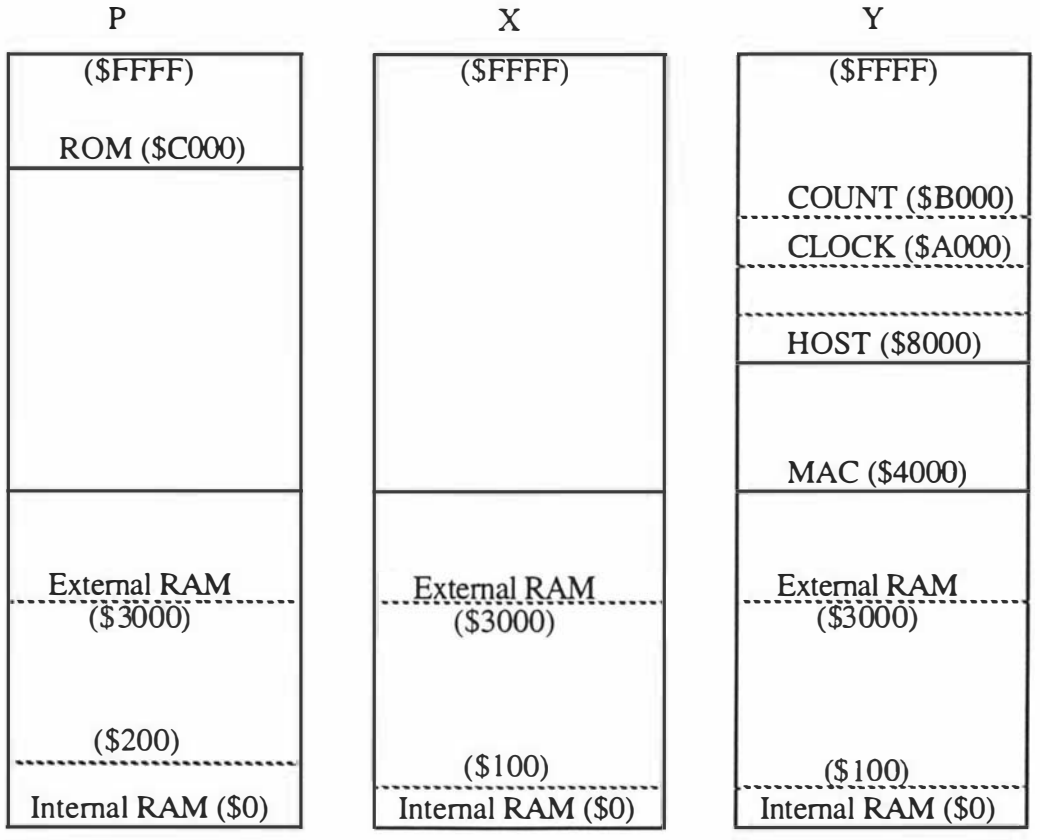


Figure 5.42 Default DSP board memory map

The monitor program performs the basic tasks necessary to transfer memory and register data to and from the Macintosh IIvx. It continuously polls its MAC-PORT address until it receives a request from the Macintosh IIvx program. The task is completed when the appropriate command is received from the Macintosh IIvx program.

The hardware reset exception is used to pass control to the DSP bootstrap program which loads the DSP monitor program into the DSP board from external PRAM immediately upon powerup or when a hardware reset occurs.

5.4.5.1 DSP56001 bootstrap mode

The bootstrap feature of the DSP56001 consists of three special on-chip modules: the 32 words of bootstrap ROM, the bootstrap control logic, and the bootstrap firmware

program which loads data from a byte-wide memory starting at location P:$C000. The external bus version of the bootstrap program is selected when the location P:$C000, bit 23 is read as a one. A byte wide EPROM is connected to the DSP56001 Address and Data Bus. The data contents of the EPROM are organised as shown below.

Address of External Byte Wide P Memory	**Contents Loaded to internal PRAM**
P:$C000	P:$0000 low byte
P:$C001	P:$0000 mid byte
P:$C002	P:$0000 high byte
.	.
.	.
.	.

The reset returns control from a user program to the monitor program whenever a reset occurs. Figure 5.43 shows the bootstrap program flowchart.

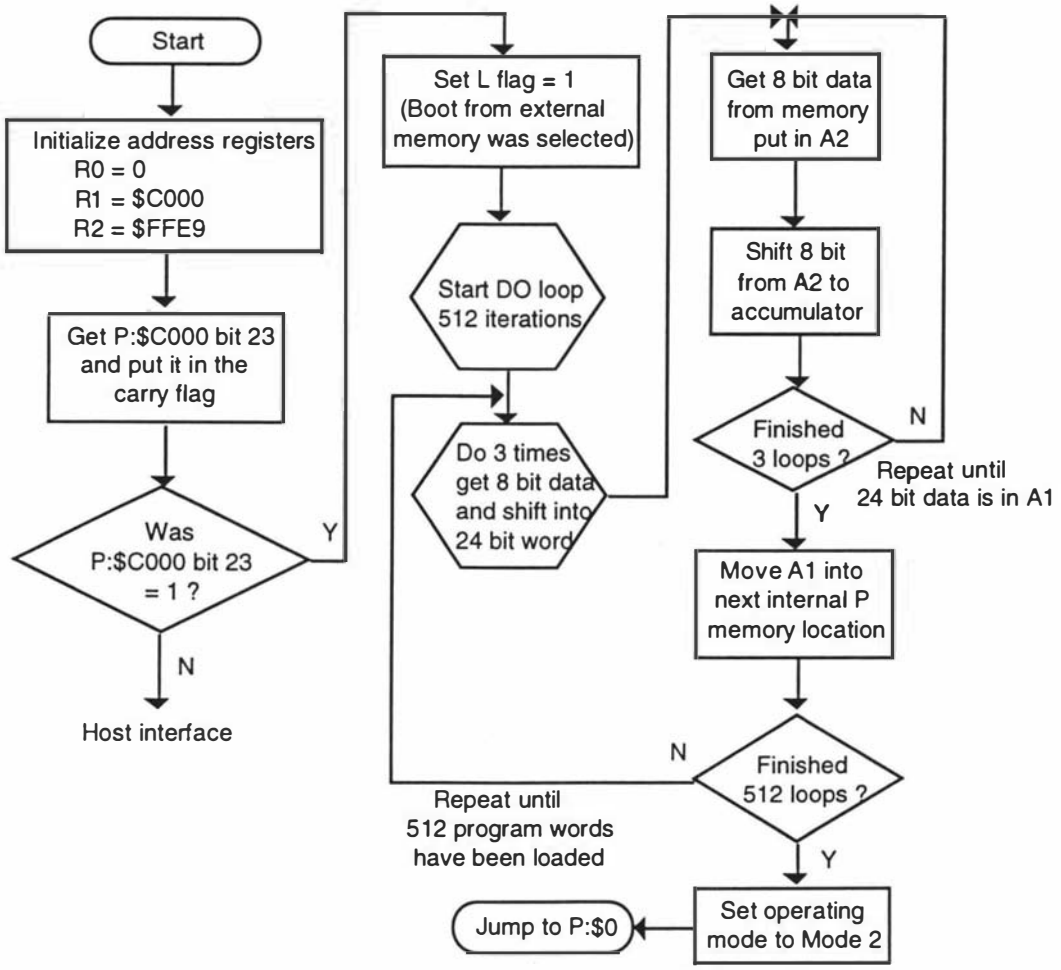


Figure 5.43 Bootstrap program flowchart

5.4.5.2 DSP board monitor program

Figure 5.44 shows the flowchart of the monitor program.

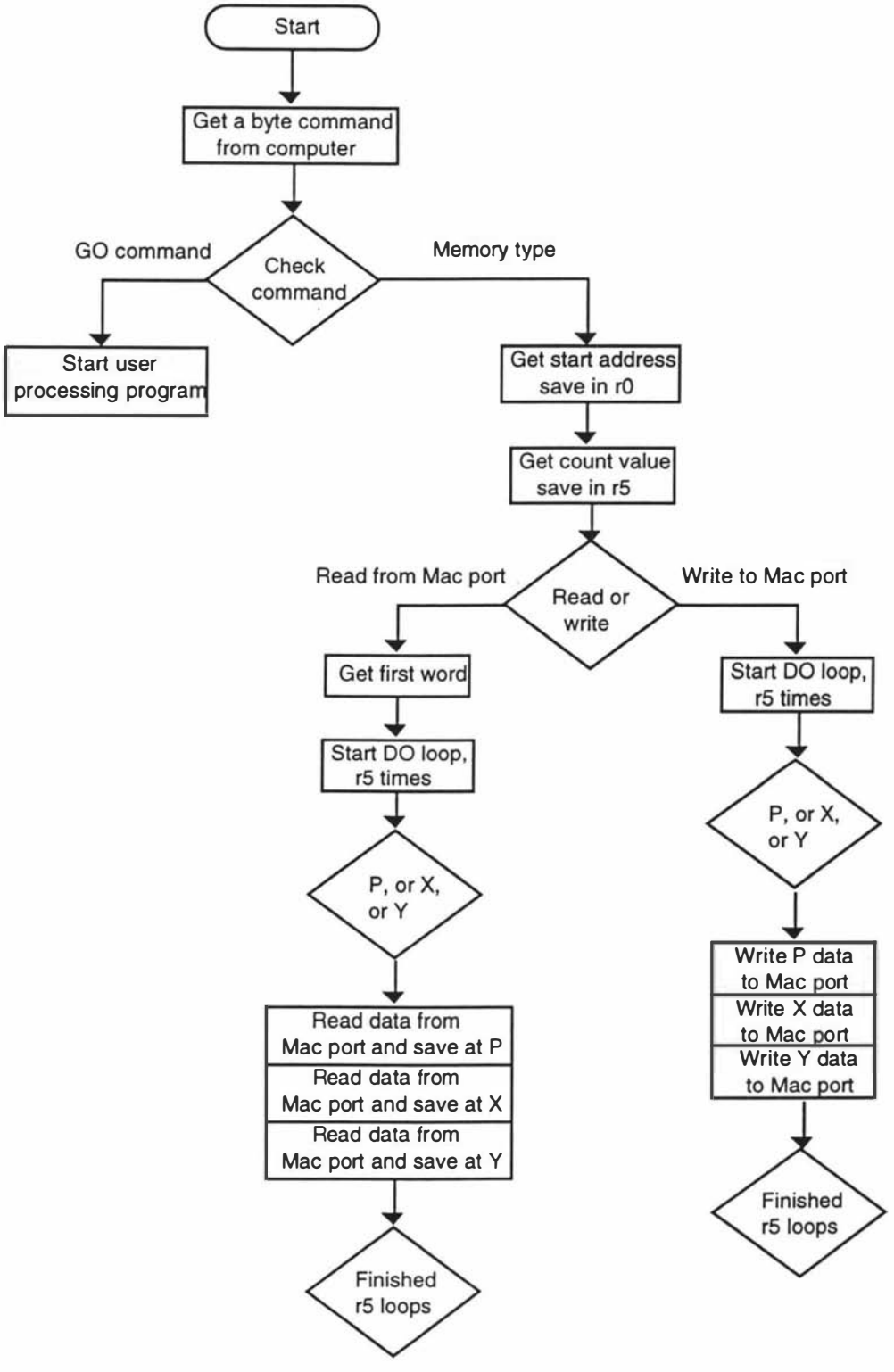


Figure 5.44 Monitor program flowchart

Figure 5.45 shows the flowchart of the program to either write data to the MAC-Port or read data from the MAC-Port.

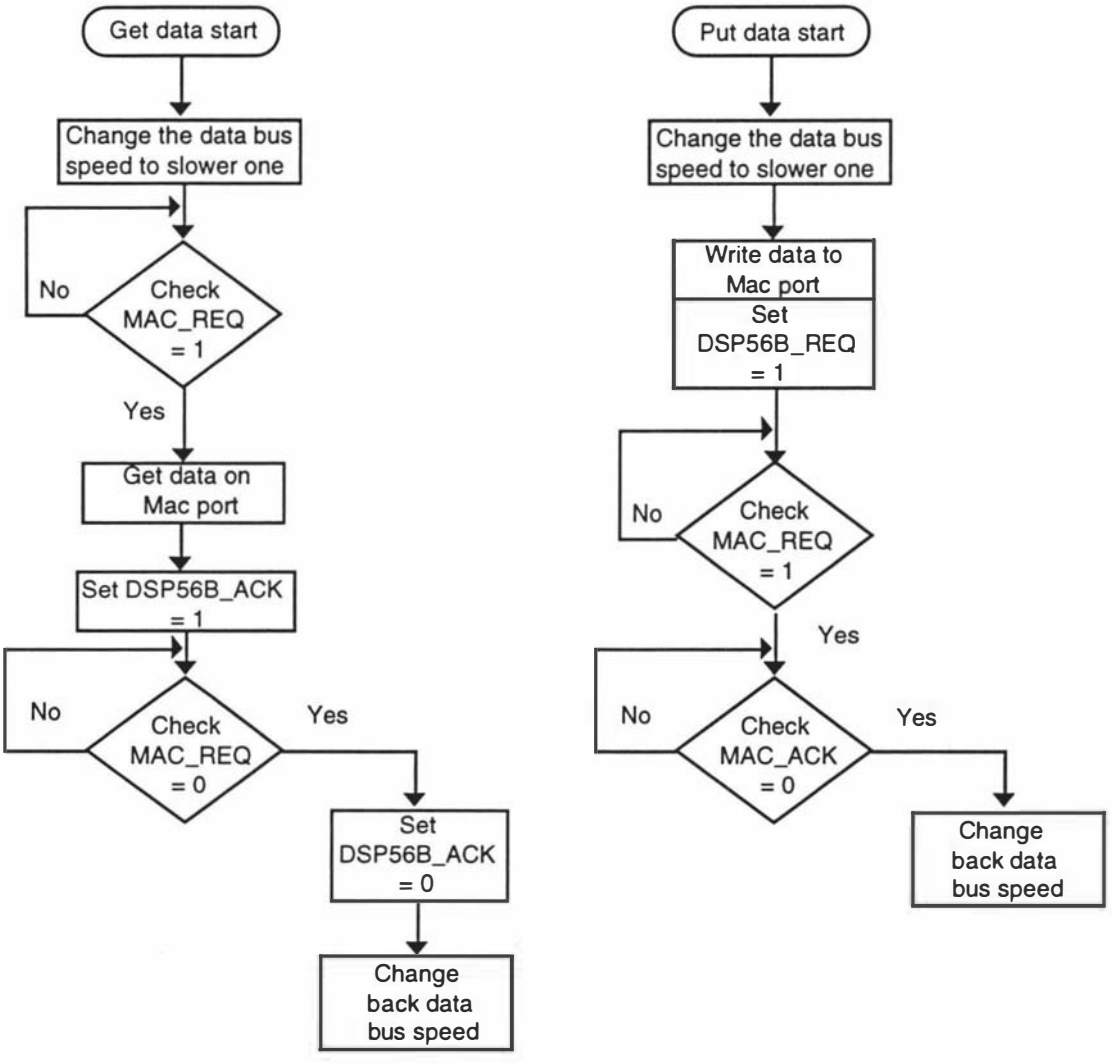


Figure 5.45 Flow chart of program for reading data from the MAC-Port or writing data to the MAC-Port

5.5 Multiple DSP System

This section contains a discussion of the Multiple instruction and multiple data (MIMD) structure, the data input of the multiple DSP system, data transfer between multiple DSP boards and the computer, and data transfer between the host DSP board and other DSP boards.

5.5.1 Multiple instruction and multiple data structure

This multiple DSP system is an MIMD parallel system, as described in chapter 4. Four DSPs are synchronised with their own sample time clocks derived from a master clock. The same input data are sampled simultaneously by the counter units of each DSP. Each DSP processes its own data, and the combined results from all DSPs produce the multiple tau autocorrelation function. This structure is shown in Figure 5.46.

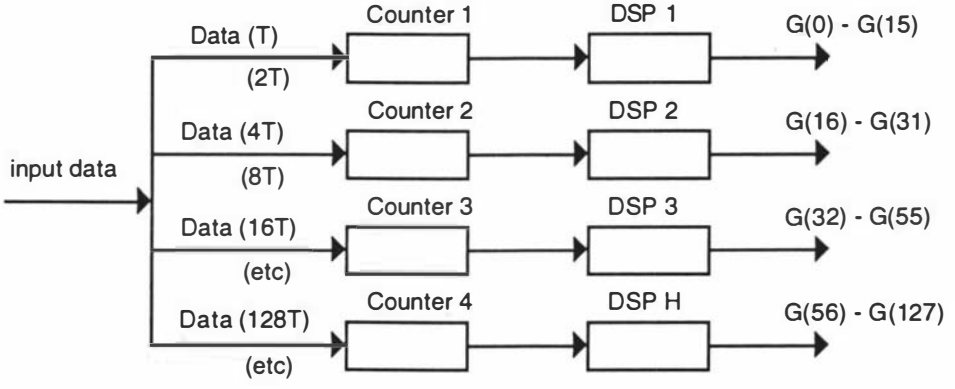


Figure 5.46 The MIMD system

One DSP board (DSP H) is specially designed to be the host DSP board. The other three DSP boards function as slave processing units. Data can be transferred to the host DSP board through a special bus (using the DSP56001 Host port). The combined processing results are sent to the control computer by the host DSP board. In addition, each DSP board can communicate directly with the control computer through its own interface circuit. All DSP boards are controlled by the control computer.

5.5.2 Data communication between multiple DSPs and the computer

Figure 5.47 shows the DSP boards and their interfaces with the computer. The processing program is loaded from the control computer into each of the DSPs via an interface circuit which also transfers the results back to the computer.

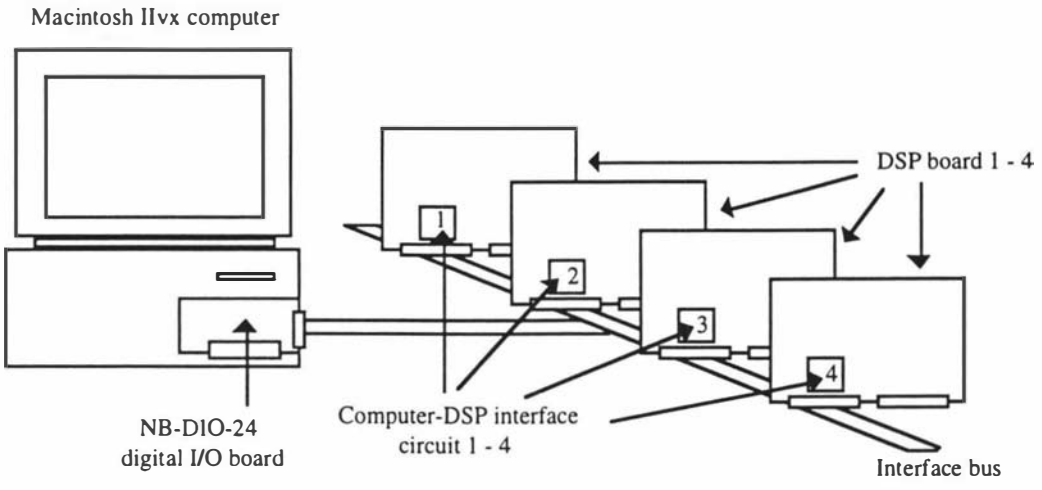


Figure 5.47 The computer and DSP boards

The DSP boards are unable to communicate with each other via the parallel interface bus, even through they are all connected to the same computer interface card. Communication among them must go through the separate 8-bit Host bus built on the mother board.

The address selector circuit is shown in Figure 5.48. Each DSP board has a unique address 0, 2, 4 or 6. Address programming is done via jumpers JGI, JG2, and JG3. The DSP boards may be addressed individually, in group or all together.

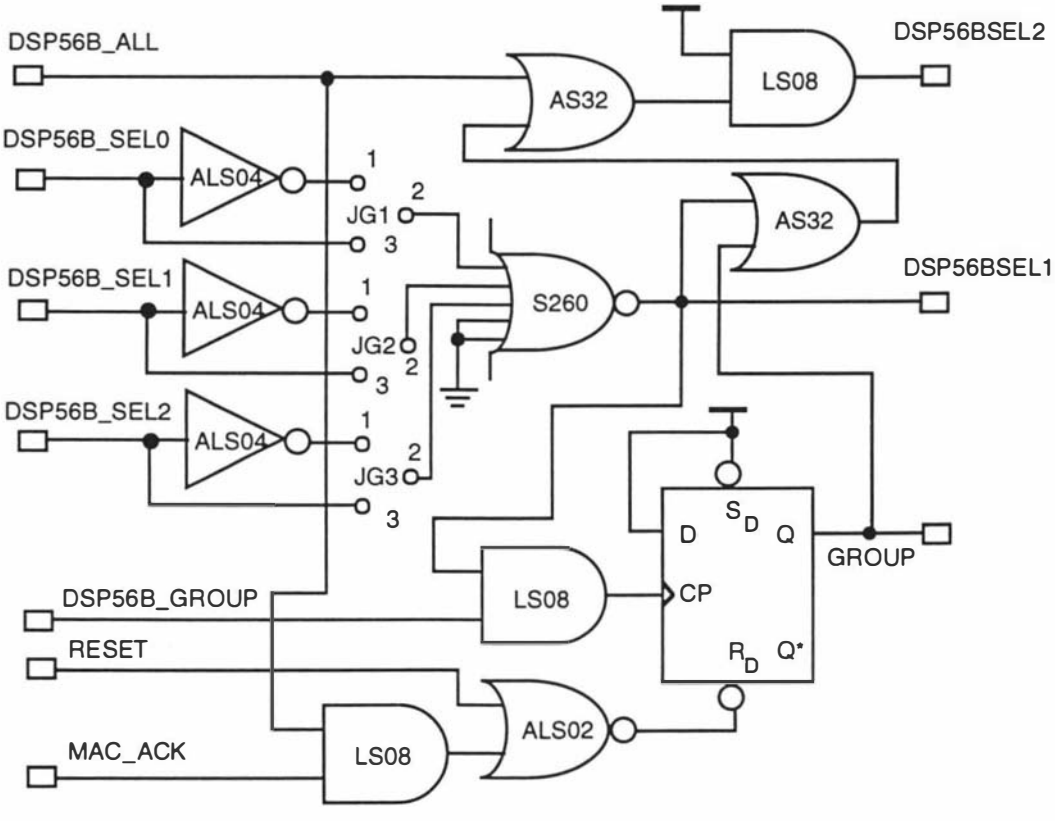


Figure 5.48 Address selector circuit

When the computer needs to communicate with a particular DSP board (to send a program or to get data), that board's address must be placed on the output control lines of the I/O board. This address is held on the control bus until communication has ended.

The GROUP input allows groups of DSP boards to be selected. Computer first selects a group member and then asserts and deasserts the DSP56B_GROUP line, this action latches the DSP board address on until unlatched by the computer program. Once all DSP boards within a group are armed, a trigger byte is put on the data bus so that each DSP board passes control to the user defined program simultaneously.

The computer may also select all DSP boards simultaneously with the DSP56B_ALL control signal. This signal is used to unlatch group members addresses or reset all DSP boards simultaneously.

There are two ways to transfer the data from the correlator to the computer. The first is transfer of data from the host DSP. In the second method each DSP board transfers data directly to the computer via the interface bus. Communication is established by the computer in response to all ID message sent by a DSP board.

5.5.3 Data communication between DSPs

Data communication between different DSP boards is through the separate 8-bit host data bus as shown in figure 5.49. The address decoder in the host DSP board generates three chip select signal S1*, S2*, and S3*, which are used to select one of three slave DSP boards.

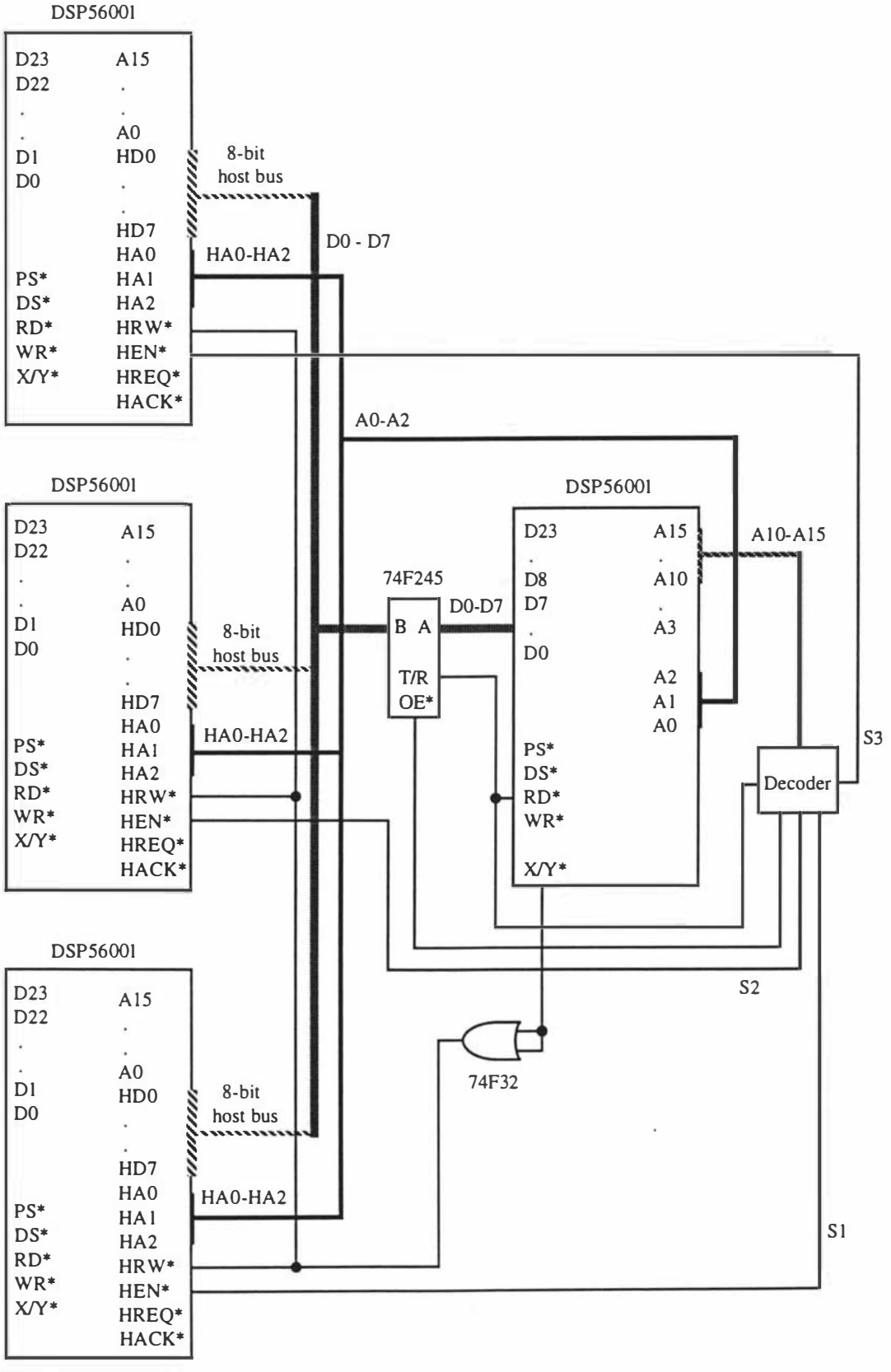


Figure 5.49 Communication between different DSP boards

5.5.3.1 Host interface bus

The host interface port of a DSP56001 is a byte-wide, full-duplex, double-buffered, parallel port which is connected directly to the data bus of the host DSP56001. The key features of the host interface are summarised in Table 5.2.

Signals (15 Pins)

H0-H7	Host data bus
HA0-HA2	Host address select
HR/W*	Host Read/Write Control
HEN*	Host Transfer Enable
HREQ*	Host Request
HACK	Host Acknowledge

Interface - DSP CPU Side

Mapping:	Three X Memory Locations
Data Word:	24 Bits
Transfer Modes:	DSP to Host
	Host to DSP
	Host Command
	Software Polled
	Interrupt Driven (Fast or Long)
	Direct Memory Access

Interface - Host Side

Mapping:
Eight Consecutive Memory Locations
Memory-Mapped Peripheral for Microprocessors.

Data Word:
Eight Bits

Transfer Modes:
DSP to Host
Host to DSP
Host Command
Mixed 8-,16-, and 24-Bit Data Transfers

Handshaking Protocols:
Software Polled
Interrupt Driven

Dedicated Interrupts:
Separate Interrupt Vectors for Each Interrupt Source. Special host commands force DSP CPU interrupts under host processor control.

Table 5.2 DSP56001 Host Interface Port

The host interface is asynchronous and consists of two banks of registers - one bank accessible to the host processor and a second bank accessible to the DSP CPU as shown in Figure 5.50.

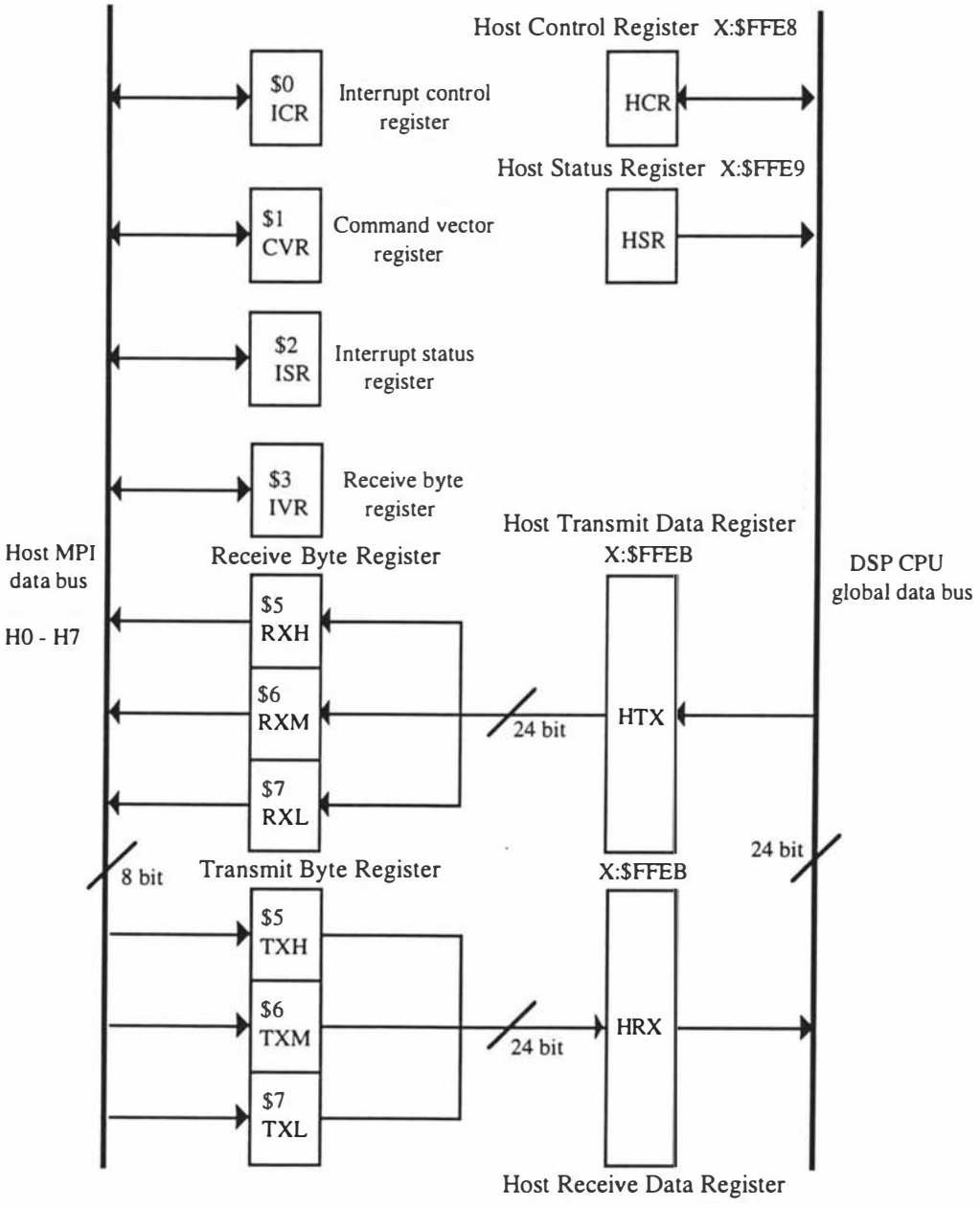


Figure 5.50 Host interface block diagram

5.5.3.2 Host interface communication

Three slave DSP host interface ports are connected to the host DSP 24-bit data bus. The slave DSP CPU views its own host interface as a memory-mapped peripheral occupying three 24-bit words in data memory space. The DSP uses the host interface as a normal memory-mapped peripheral, using either standard polled or interrupt programming techniques. Separate transmit and receive data registers are double buffered to allow

efficient data transfer at high speed. Memory mapping allows DSP CPU communication with the host registers to be accomplished using standard instructions and addressing modes. In addition the MOVEP instruction allows host interface-to-memory and memory-to-host interface data transfers without going through an intermediate register.

Initialisation the host interface communication system takes two steps. The first step is to initialise the slave DSP side of the host interface, which requires that the interrupt options are selected and then the host interface is selected. The second step is for the host DSP to clear the Host Command (HC) bit of Command Vector Register (CVR) and select the data transfer method.

1 Host DSP to slave DSP Data Transfer.

Care is required when reading multibit registers that are written to by another systems. Handshake flags are required for both interrupt driven and polled data transfers to indicate when the transmit byte registers are empty. This guarantees that the slave DSP will read stable data.

The slave DSP can poll the host interface port states to see when data has arrived, or it can use interrupts to read data when the host interface interrupt is enabled and data are ready at the host interface port. The host DSP can cause three types of interrupts in the slave DSP. These are: receive data full interrupt, transmit data full interrupt, and command interrupt. The host command interrupt can be used to control the DSP by forcing it to execute any of thirteen subroutines that can be used to transfer data and process data. Also the host command interrupt can cause any of the other 19 interrupt routines in the slave DSP to be executed.

The program initialises the host port and then enters a wait loop and allows interrupts to transfer data from the host DSP to a slave DSP. The first step is to enable the host interface and configure the interrupts. The slave DSP enables HRIE (Host Receive Interrupt Enable) in the Host Control Register (HCR), which allows the interrupt routine to receive data from the host processor.

The slave DSP polls the Host Receive Data Full (HRDF) of Host Status Register (HSR). Data transfer occurs when HRDF = 0.

Figure 5.51 shows the program steps for data transfer from the host DSP to a slave DSP.

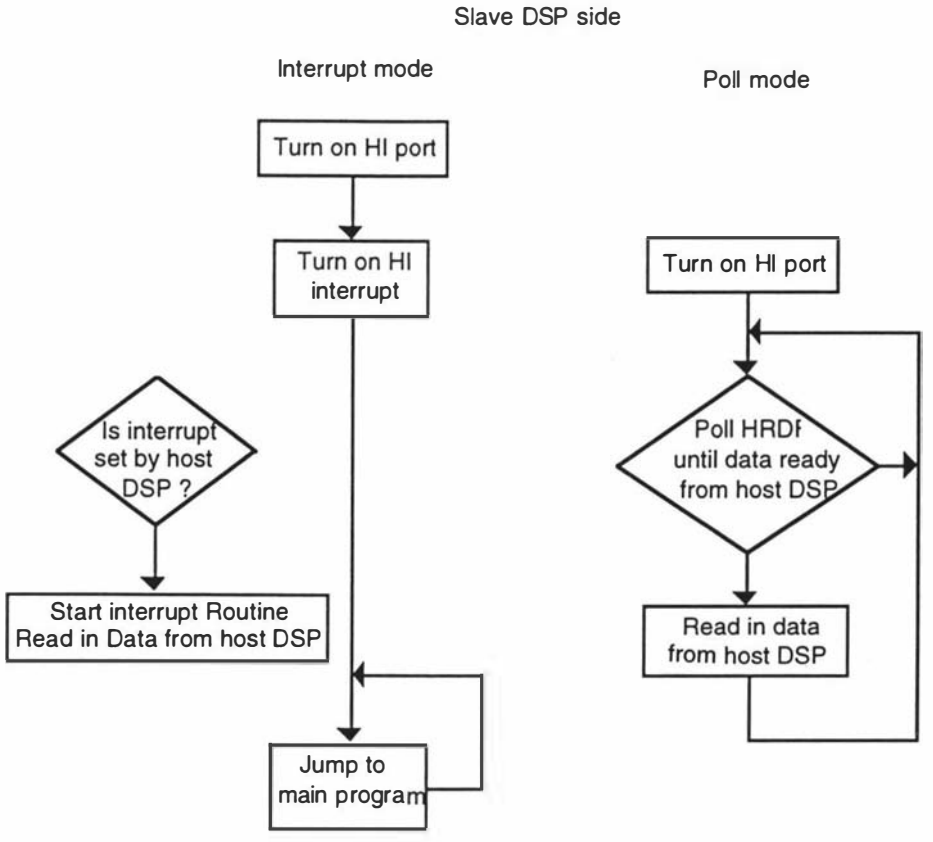


Figure 5.51 Program steps for data transfer from the host DSP to a slave DSP

The receive data interrupt routine is implemented as a long interrupt, (the instruction at the interrupt vector location is a JSR). The MOVEP instruction moves data from the DSP host interface to a buffer area in memory and increments the buffer pointer so that the next word received will be put in the next sequential location.

2 Slave DSP to host DSP Data Transfer.

The slave DSP host interface appears to the host DSP as eight words of byte wide static memory. The host DSP can access the slave DSP host interface asynchronously by using polling techniques or interrupt based techniques to indicate when data is available. This guarantees that the data in the receive byte registers will be stable. Separate transmit and receive data registers are double buffered to allow the slave DSP and host DSP to transfer data efficiently at high speed.

Data is transferred from the slave DSP to the host DSP in a similar manner as from the host DSP to the slave DSP. The slave DSP CPU polls the Host Transmit Data Empty (HTDE) bit in its own Host Status Register (HSR) to see when it can send data to the host DSP.

The interrupt mode can also be used in data transfer. The host interface port of the slave DSP can be interrupt enabled by the Host Transmit Interrupt Enable (HTIE) in the Host Control Register (HCR).

Figure 5.52 shows the program steps for data transfer from a slave DSP to the host DSP.

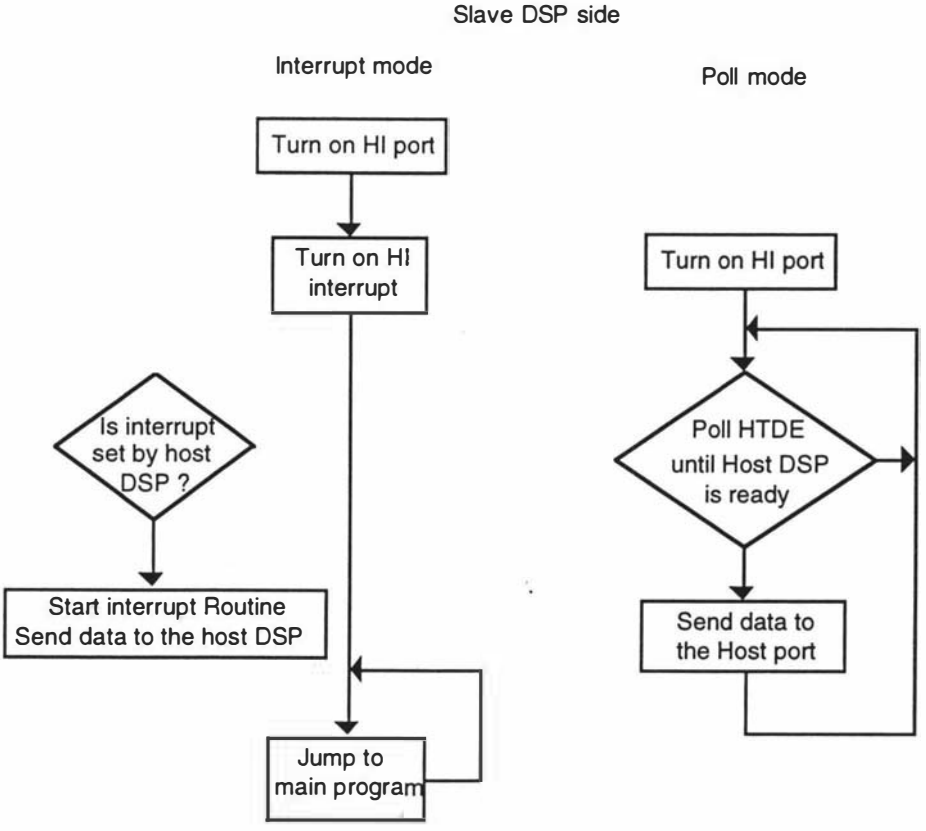


Figure 5.52 Program steps for data transfer from a slave DSP to the host DSP

5.6 Programming the Multiple DSP56001 System

The programming of the multiple DSP correlator will be discussed in this section. The correlator program development consists of: analysis of task requirements, software design, coding and documentation, followed by debugging and testing.

Software developed for the correlator consists of three different parts. The first part, the DSP monitor program, is written in DSP56001 assembly language and was developed to initialise the correlator and to permit communication between the correlator and the host computer. This part was presented in section 5.4 as part of interface system. The second part, written in LabVIEW, enables the user to control the correlator and the real time display of the correlation function. This part was also presented in section 5.4. The third part, written in assembly language to optimise real time performance, is the correlator-dedicated software to process the multiple tau correlation function. The characteristic features of this software will now be described.

5.6.0 Program development with the DSP56001

The DSP56001 assembly language is a symbolic language defined by Motorola to code DSP56001 source programs and is used to avoid coding directly into object code. The assembler, a program itself, translates a program expressed in assembly language into object code which consists of executable binary machine code.

The complete programming model for the DSP56001 central processor is shown in Figure 5.53. The programmer's model of DSP56001 registers is divided into three parts: the control unit, the data arithmetic unit, and the address register.

X1:X0, Y1:Y0 are 24 bit general purpose arithmetic logic unit input registers. They serve as input pipeline registers between the X data bus and Y data bus and the multiply and accumulator unit. The six data Arithmetic Logic Unit (ALU) accumulator registers A2:A1:A0, B2:B1:B0 form two general purpose 56 bit accumulators, these are shown in Figure 5.54.

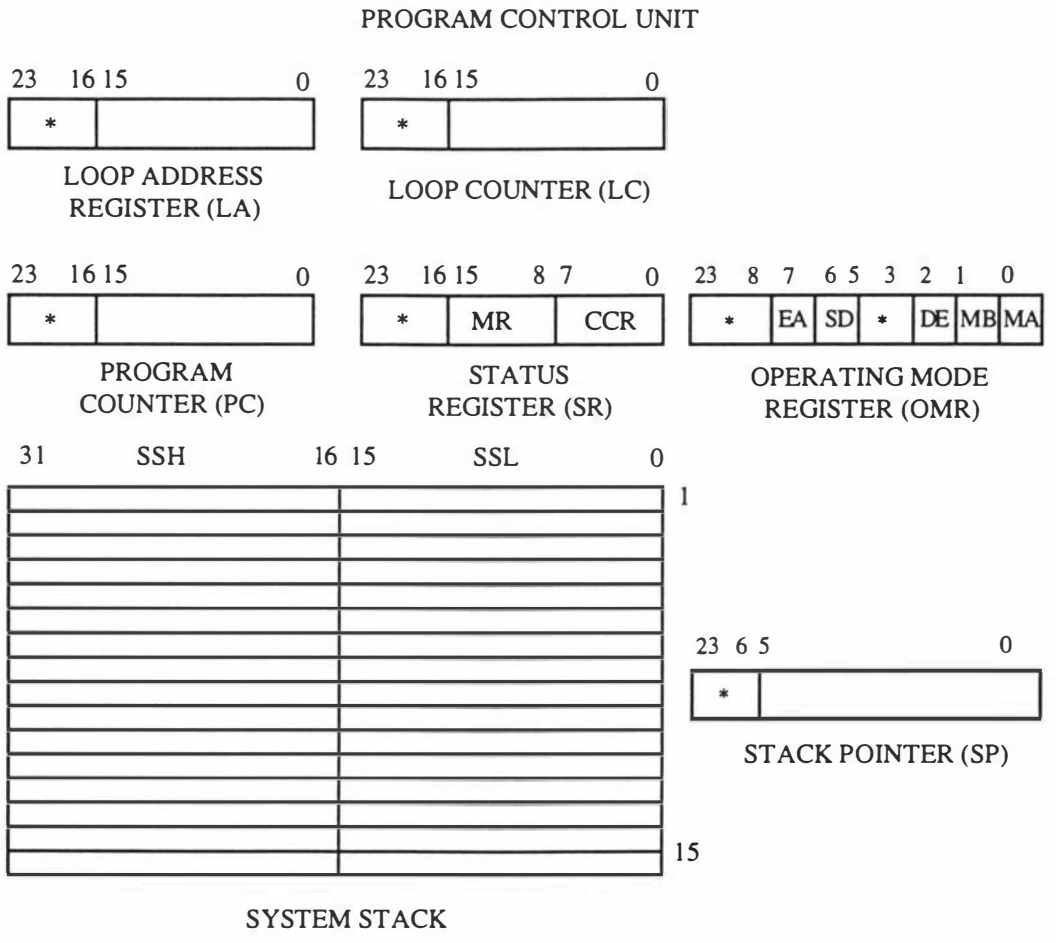


Figure 5.53 DSP56001 central processor programming model 1-1

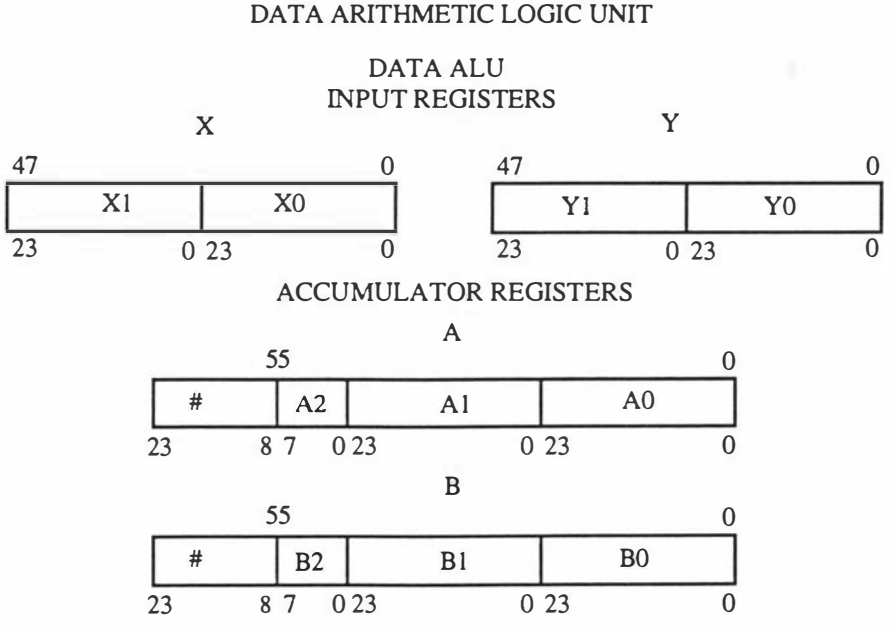


Figure 5.54 DSP56001 central processor programming model 1-2

The address, offset and modifier registers are shown in Figure 5.55. The eight pointer registers, R0-R7, are 16 bits wide and may contain addresses or general purpose data. When supporting parallel X and Y data memory moves, the pointer registers must be viewed as two separate files, R0-R3 and R4-R7, one file for each bus. The eight offset registers, N0-N7, are 16 bits wide and may contain offset values used to increment and decrement address registers in indexed address register update calculations. The content of modifier registers M_n defines the type of address arithmetic to be performed for addressing mode calculations. These address arithmic modifiers allow the DSP address generation unit to support linear, reverse-carry, and modulo address arithmetic for all address register indirect modes. These special address arithmetic types allow the creation of data structures in memory for FIFOs (queues), delay lines, circular buffers, stacks, and bit reversed FFT buffers. Data are manipulated by updating pointer registers rather than moving large blocks of data.

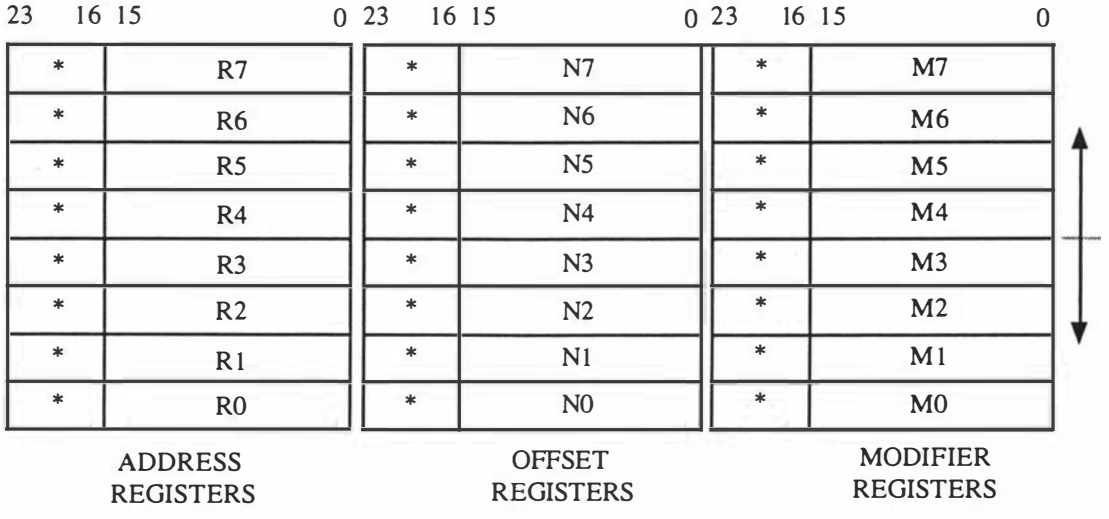


Figure 5.55 DSP56001 central processor programming model 1- 3

5.6.1 Correlator programming

The assembly program for the correlator algorithm is structured as several main routines: an interrupt routine to read in counter data (INTERRUPT), calculation routine (PROCESS), and a data transfer routine (DISPLAY), plus one subroutine for initialisation. After the initialisation routine, the correlation routine is executed, and then the transfer routine. The code then alternates between the calculation and the transfer routines indefinitely. This code is simply repeated N times incrementing the correlator channel address and the RAM destination addresses each time.

The 24-bit channel storage capacity of the device means that the input count rate can be kept reasonably large before the channels overflow between read cycles. If the carry flag is not set, the correlator stops the correlation processing schedule when it reaches the run

number limit set by the control computer. The 128 correlation channel data and monitor channel data are then sent to the computer.

A flow diagram of the program is shown in Figure 5.56. This shows the order in which the various portions of the algorithm are executed.

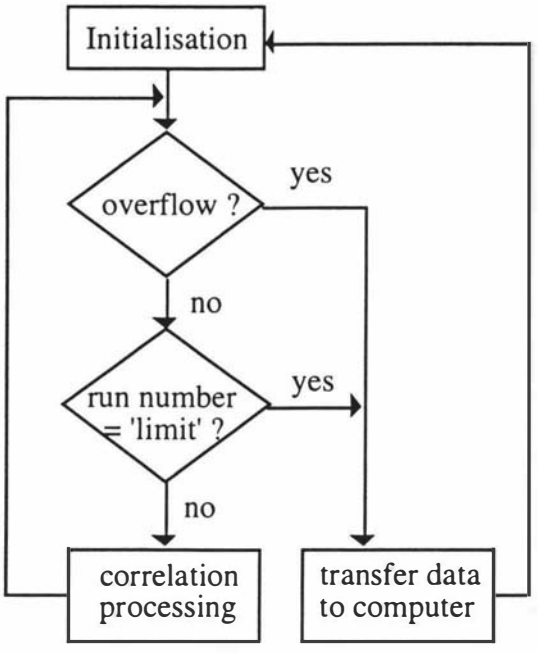


Figure 5.56 A flow diagram of the correlator program

The program continually loops through the data transfer routines collecting and displaying data until it is halted through the keypad. After halting the correlator, the user may either invoke a user function to alter the correlator operation by entering another digital signal processing algorithm or resume data collection. Figure 5.57 is a flow chart of the correlator software operation.

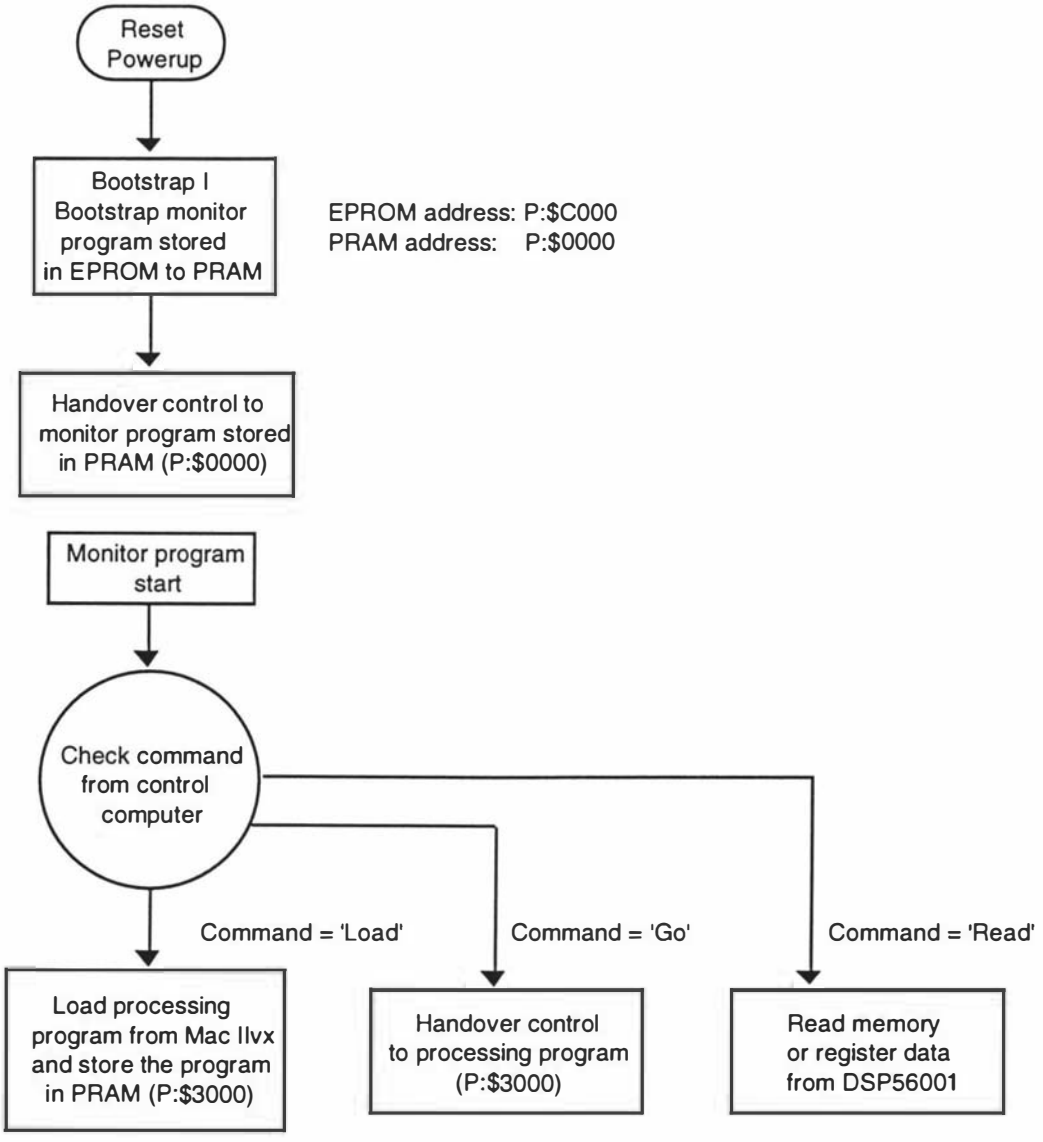


Figure 5.57 Flow chart of the correlator software operation

Figure 5.58 illustrates the arrangement of the data blocks in RAM. Eight blocks (DATA_BUF1 - DATA_BUF8) have been assigned to hold the data for different sample time processing. The COE_BUF area is used for coefficients. IN_BUF holds the current data from the counter for correlation function processing. CH_BUF holds the calculated autocorrelation channel data for the DISPLAY routine. The tasks are to add the fresh data stored in IN_BUF to different sample time data stored in different DATA_BUF (1 - 8) and then update the CH_BUF accordingly.

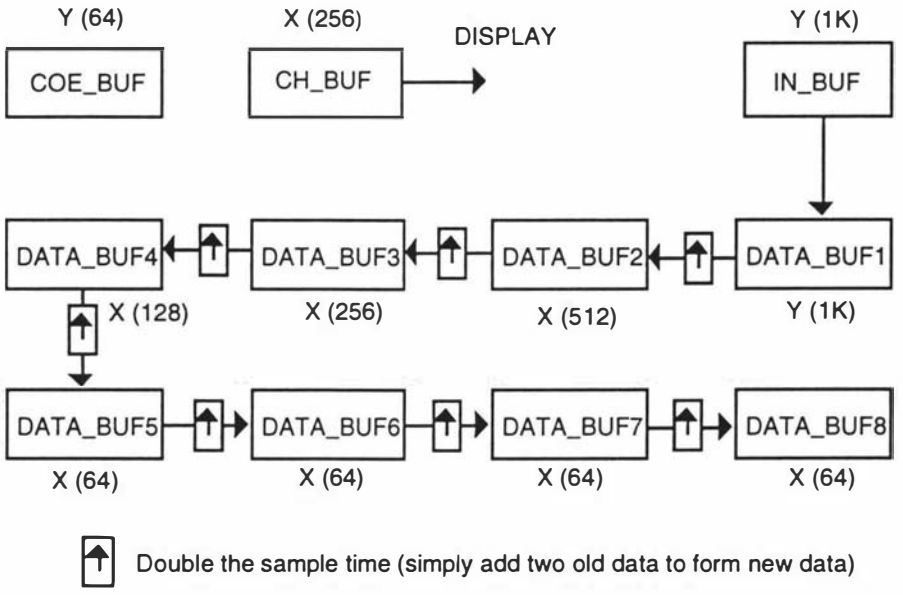


Figure 5.58 Illustration of data block arrangement

The most important task of these programs is to ensure that after a DSP is interrupted, it reads in the counter data, stores the data in IN_BUF. After that the data are coverted into different sample time data and are stored in DATA_BUF(1 - 8). Then the data are processed to autocorrelation coefficients. Finally the data are transferred to the Macintosh IIvx computer and the display is refreshed.

5.6.2 Initialisation

The initialisation subroutine accomplishes three main tasks: initialising the DSP, initialising program variables, and initialising data buffer pointers and modulo registers. At the beginning of the program the address pointers are initialised. The channel buffers (to store the accumulating correlation data) are cleared for each experiment. This includes initialising all data and coefficient buffers, initialising all important pointers and registers, initialising sample time clock and initialising the computer port.

The DSP initialisation consists of enabling the bootstrap routine and setting the DSP Bus Control Register (BCR) for zero wait state external program memory access. The zero wait states for external program memory are needed so the algorithm can run in real time. After that the interrupts are enabled, and the program jumps into the main program loop.

5.6.3 Correlator interrupt routine

The effective use of real-time systems requires an efficient interrupt handler capable of receiving events from the real world with a minimum of delay. An interrupt handler simply causes counter data to be read and stored. This handler has the highest priority in the whole system.

By reserving enough registers for the interrupt and correlation processing, the code can be made very efficient and the processing speed can be enhanced. Address registers are assigned for IN_BUF accumulation, for DATA_BUF (1 - 8), and for correlation function processing.

The handler essentially creates a high-priority buffer in which to store interrupt events. The code for this interrupt handler is listed as following,

```
P: $0008          JSR   INTERRUPT          ;jump to interrupt routine

P: INTERRUPT      move  Y:$BFFF,Y0         ;read in data from counter
                  move  Y0,y:(r1)+         ;save data into input buffer
                  rti                      ;return from FIFO interrupt
```

The interrupt involves the use of a well organised input FIFO (first-in, first-out) buffer IN_BUF. The input can pass its data to the buffer as quickly as possible and thus be available to input external events for as much of the time as possible so no data is lost.

Using this FIFO buffer, sufficiently large data can be sampled and processed. Such a buffer provides storage for incoming signals from the input handler while passing the earliest signal to the computing process.

The FIFO buffer is organised as using three pointers, head, tail, and stop, keep track of the position of the data within the buffer. Head points to the next element of the buffer to receive an input, tail points to the next element of the buffer to do an output, and stop points to the element in the buffer directly behind tail. The buffer itself can be thought of as a circular buffer in which tail is always chasing head. When data are read from the input process, head advances; when data are passed to the computing process, tail advances.

As shown in Figure 5.59, data element five is the most recent input, while data element zero is the next to be emitted. The SIZE is defined to be of size buffer, and MAX is one less than buffer size. If an input does take place, the head pointer is incremented. Because FIFO is a circular buffer, the increment operation is done modulo of the buffer size. Any input is received on counter interrupt request.

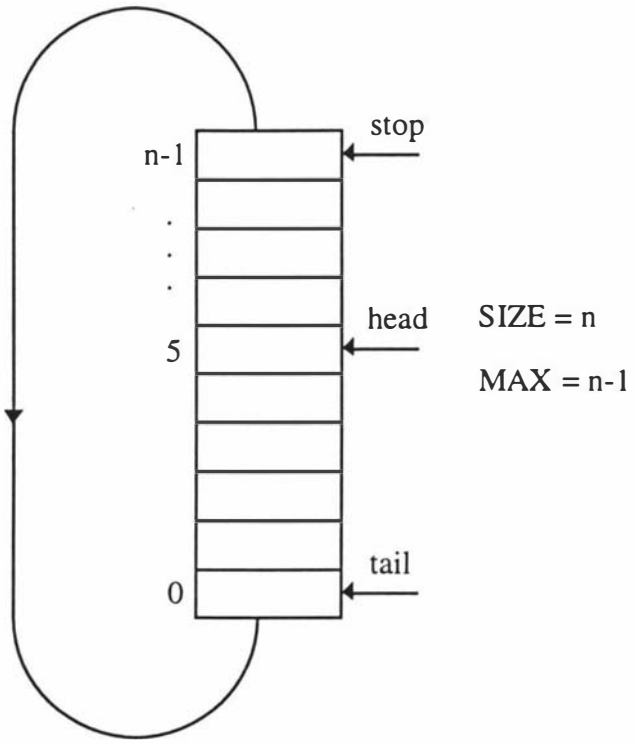


Figure 5.59 FIFO pointers

When the computations for the correlation are completed, the input buffer (IN_BUF) is copied into DATA_BUF1 and a new set of computations begin. During all time, IN_BUF continues to get new data samples.

The output value is assigned to the data element that tail points to, which is the next data element to be passed to the computing process. Before each copying process, tail is initialised to one element before head value to ensure that the oldest sample is copied before the next input sample overwrites it. The stop points to the element just behind tail, the last element in which new data can be stored before data already in the buffer will be overwritten. All other samples will have adequate time to be copied before head point equal stop point. Then the data copying process starts, tail is updated, followed by stop. The counter sampling time ensure that head is always behind stop.

Another method is for the interrupt routine to include a correlation processing routine to make a long interrupt. This means that every input data are processed at once. Under this method, the shortest sample time is dependent on the time to complete the interrupt routine. Test results comparing the performance of these interrupt handlers show clearly that for short sample times requiring a relatively short processing time, a simple three-process interrupt handler can provide very good performance if it includes a simple FIFO buffer. For longer sample times which require a relatively long processing time, handlers with a long interrupt provide much better performance.

5.6.4 Correlation routine

Figure 5.60 shows a block diagram of the code algorithm for the multiple tau autocorrelator, where $T' = 2T$ (double sample time), $G(\ )$ represents a correlation channel, and $M(\ )$ represents a monitor channel.

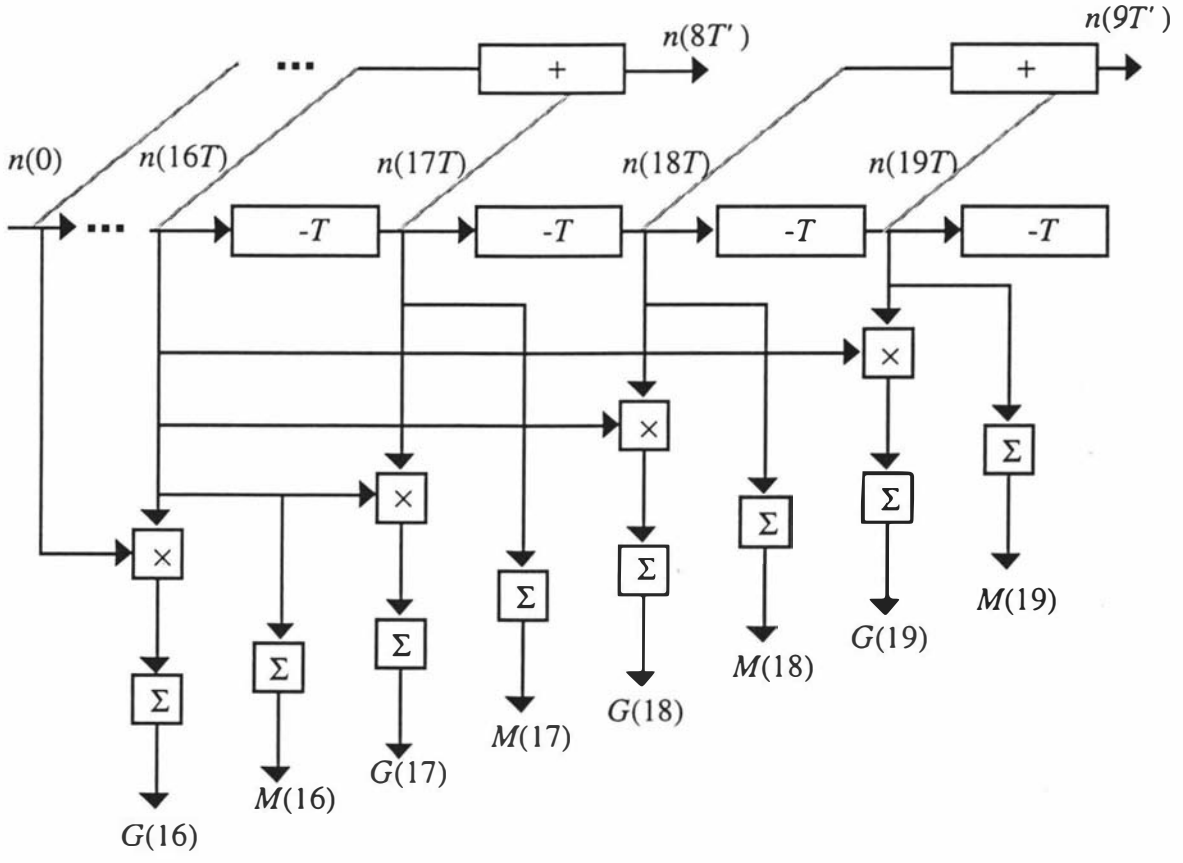


Figure 5.60 Autocorrelator coder

As illustrated in the diagram, the autocorrelator coder must meet several requirements. The first is to provide correlation channel data. The second is to provide the monitor channel data. The third is to provide sample data for doubled sample time processing.

It continually loops through the sum-and-store (for double sample time data) and correlation channel calculations. The host computer, having started this program, is not busy with this task and can run any other program during the measurement process.

The correlation routine has been written in in-line code for the whole block of channels belonging to each DSP56001. The assembly code to read, multiple and accumulate, and store one correlation channel is:

```
lua     (r6)-n6,r2          ;r1 new data, n1 = 4
move    x:(r6)+,x0
```

```
        move    x:(r3),A1
        clr     B           x:(r2)-,Y1
        add     Y1,A
        mpy     x0,Y1,B     A1,x:(r3)+
        asr     B           x:(r5),A0
        add     A,B
        move    B0,x:(r5)+
```

5.6.5 Display routine

The display process is repeated for the subsequent 128 channels. The assembly routine which sends the correlation data to the computer for display is also written in in-line code and is listed as follow.

```
        move    #>DSP56B_REQ,x1   ;set DSP56B_REQ bit
        move    a1,b0
        or      x1,a
        move    #>$ffff,x1
        and     x1,a
        move    a1,y:MACPORT      ;write byte and DSP56B_REQ to
                                   MACPORT
        move    #>MAC_ACK,x1
        jsr     p_test            ;wait for MAC_ACK to go high
        move    b0,a1
        move    #>MAC_ACK,x1      ;check if MAC_ACK = 0
        move    a1,y:MACPORT      ;wait for MAC_ACK to go low
p_test  move    y:MACPORT,a1      ;read flags
        and     x1,a
        jeq     p_test
```

5.6.6 Implementation on the DSP56001

The algorithm described above has been implemented on the multiple DSP system. The assembly source code has been written and optimised by the Motorola DSP56000CLASB development system. The system contains the simulator, linker, librarian and assembler software package. The source code is set up to run the algorithm in real time on the four DSP56001s, each running at 33 MHz. The code can also be easily modified for much more complicated configurations.

The correlator executes DSP56000 object code which is generated using the DSP56001 Macro Assembler program (ASM56000). The reformatted object code is loaded into the

correlator through LabVIEW and the Macintosh IIvx-Correlator interface system. The I/O routines allow programs running on the DSP boards to access data in ASCII files on the host computer. These routines provide a convenient method for accessing the correlator source code which is distributed in ASCII format. Any source code containing ASCII characters may be used as input files to the correlator system.

The implementation of the correlator code on the DSP56001 uses several optimisation techniques to obtain real time performance. The goal of these techniques is to obtain the minimum execution speed for the entire algorithm.

The code is simulated by Motorola SIM56000 simulation program. The DSP56000 simulator program (SIM56000) is a software tool for developing programs and algorithms for the DSP. This program exactly duplicates all of the functions of a DSP56001, including all on-chip peripheral operations, all memory and register updates associated with program code execution, and exception processing activity. The highly pipelined bus activity of the devices is exactly simulated. This enables the simulator to provide an accurate measurement of code execution time which is critical in a real time correlator application.

The correlation channels and monitor channels are stored in internal data memory, and some of critical variables are stored below address $40 so that the short immediate addressing mode can be used when accessing them.

If possible, subroutines are avoided. The overhead of passing parameters and calling subroutines requires a significant portion of the total execution time, enough to prohibit real time performance. Eliminating subroutines also allows the parallelism of the DSP56001 to be exploited as much as possible since less data movement is required. The disadvantage is that more program memory is required.

Memory can be sacrificed to gain execution speed because of the multiplexed external bus of the DSP56001. The external bus will allow one external access per instruction cycle with no penalty in execution speed. Some data and variables are stored separately and these allow the parallel bus structure of the DSP56001 to be taken advantage of.

REP and DO instructions take extra cycles to set up the loop registers, so if possible, the REP and DO are replaced by repeating the code many times. For the whole algorithm, these savings add up to allow several cycles of execution time to be saved for each sample.

Hardware DO loops execute without overhead once the loop is started. After a three cycle initialisation of the DO loop, the body of the loop executes as if it were straight line code.

Since the DO loop does not require any overhead cycles for each pass, the need for straight line code is eliminated.

Parallel moves are taken advantage of whenever possible. This makes the assembly code more difficult to understand and modify since values may be fetched from memory long before they are actually used. Dual parallel moves on the DSP56001 usually require the use of an addressing register for accessing memory.

The order of execution of the routines and the worst-case processing time in instruction cycles for each routine is simulated by SIM56000 program. An instruction cycle (Icycle) is defined as two clock cycles on the DSP56001. For a 33 MHz DSP56001 an Icycle is 60 ns. Sampling at a rate of 3.2 μs translates into 53 Icycles. The worst case calculations indicate the total of instruction cycles for doing both counter read and channels calculation. These worst-case times calculations were based on worst-case branches and delays in each routine. Taking into account all these conditions we have a minimum sample time of about 3.2μs.

6 AUTOCORRELATOR CONSTRUCTION AND TESTING

The production of the plug-in printed circuit boards and the assembly and construction of the entire instrument are just as important as the circuit and system design for the completion of a properly functioning multiple tau autocorrelator. This chapter describes the production of the printed circuit boards, the construction of the entire instrument and the correlator testing.

6.1 Construction Considerations

To successfully construct an electronic instrument, the instrument's appearance, reliability, ease of construction and ease of testing, all need to be considered. Based on these considerations, the correlator is constructed as follows:

(1) The instrument has a standard Eurocard structure.

(2) The circuits are constructed on printed circuit boards.

(3) The circuit boards are connected using Euro DIN connectors and flat ribbon-cable connectors.

(4) A specially designed printed circuit motherboard is used for inter-card connection.

(5) The individual circuit cards are housed in a card "cage", plugged into the motherboard.

The following sections describe the step by step construction of the instrument. These are; the printed circuit board construction, treatment of interconnections, controls, and enclosures.

6.2 PC Board Fabrication

The best method of constructing an electronic circuit is to use printed circuit (PC) boards. They offer the most reliable fabrication technique.

The first step in the design of a printed circuit board is to convert the schematic circuit diagram into a corresponding pattern of desired copper foil traces. The main task is to figure out how to make all the interconnections the circuit demands by running lines around a board.

Working from the circuit schematics designed in chapter 5, MacCad software was used to design the set of interconnection paths and produce a set of precision machine-drawn plots. Beginning with the circuit diagram, the trial sketches of component layouts and interconnections were established, and eventually these were worked together into a final layout drawing. Then pads (terminal areas for component connections) and patterns (used for IC and transistor pads and for ribbon and edge connectors) were then put in place. After this the plots were printed to make a negative film image of the board design.

A single sided board provided all the required connections (aided by a few wire jumpers) for the mother board. Other circuit boards are double-sided with plated through holes. Lines of 0.0215 or 0.020 inch width are used for signal paths. Power supply lines are 0.05 inch wide and the wider ground paths are 0.2 inch wide.

Figure 6.1 shows a completed actual-size top layer of a DSP printed circuit board on which the opaque pattern delineates the desired circuit traces and pads. Figure 6.2 shows the bottom layer of the same board.

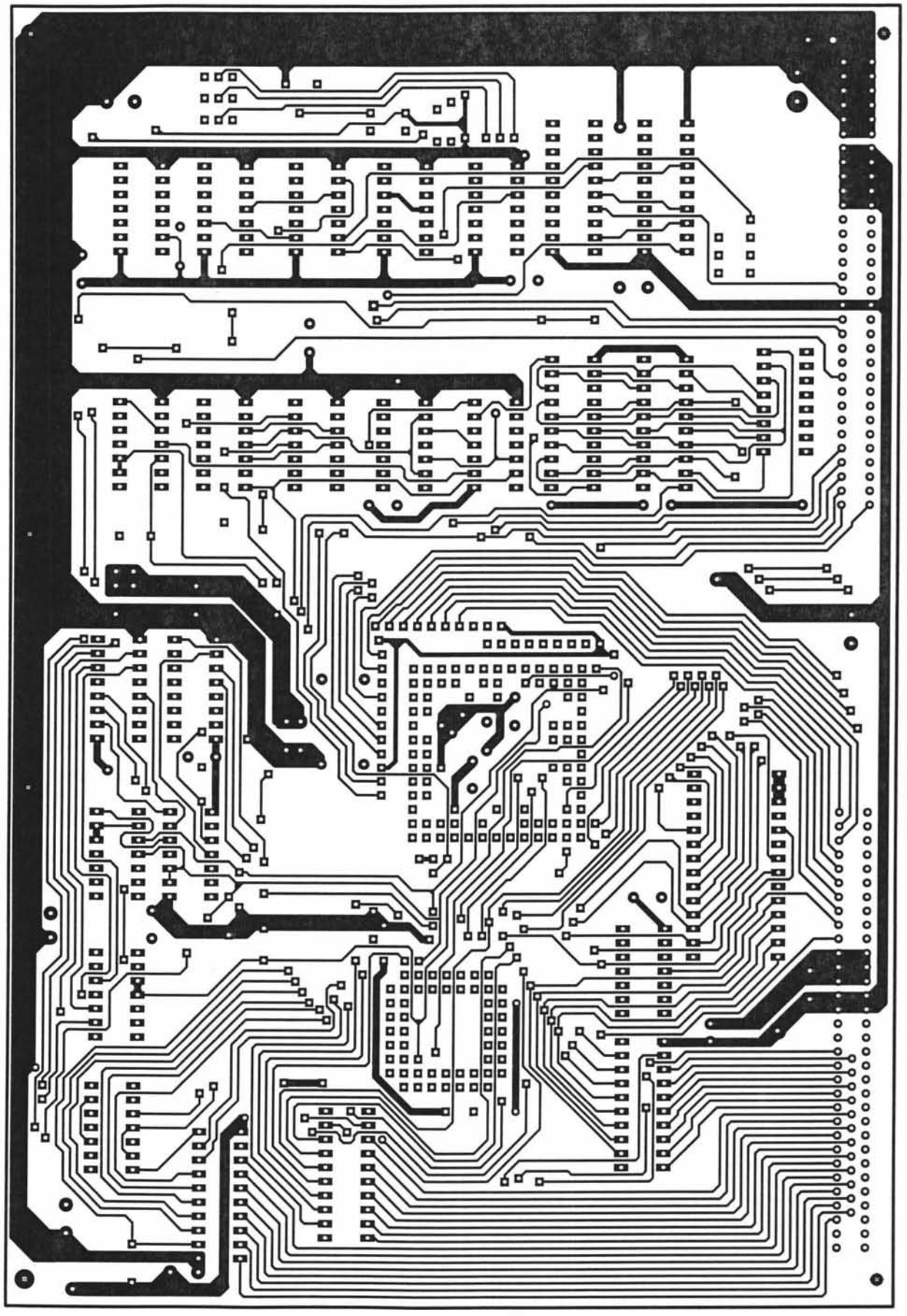


Figure 6.1 The top layer of DSP board

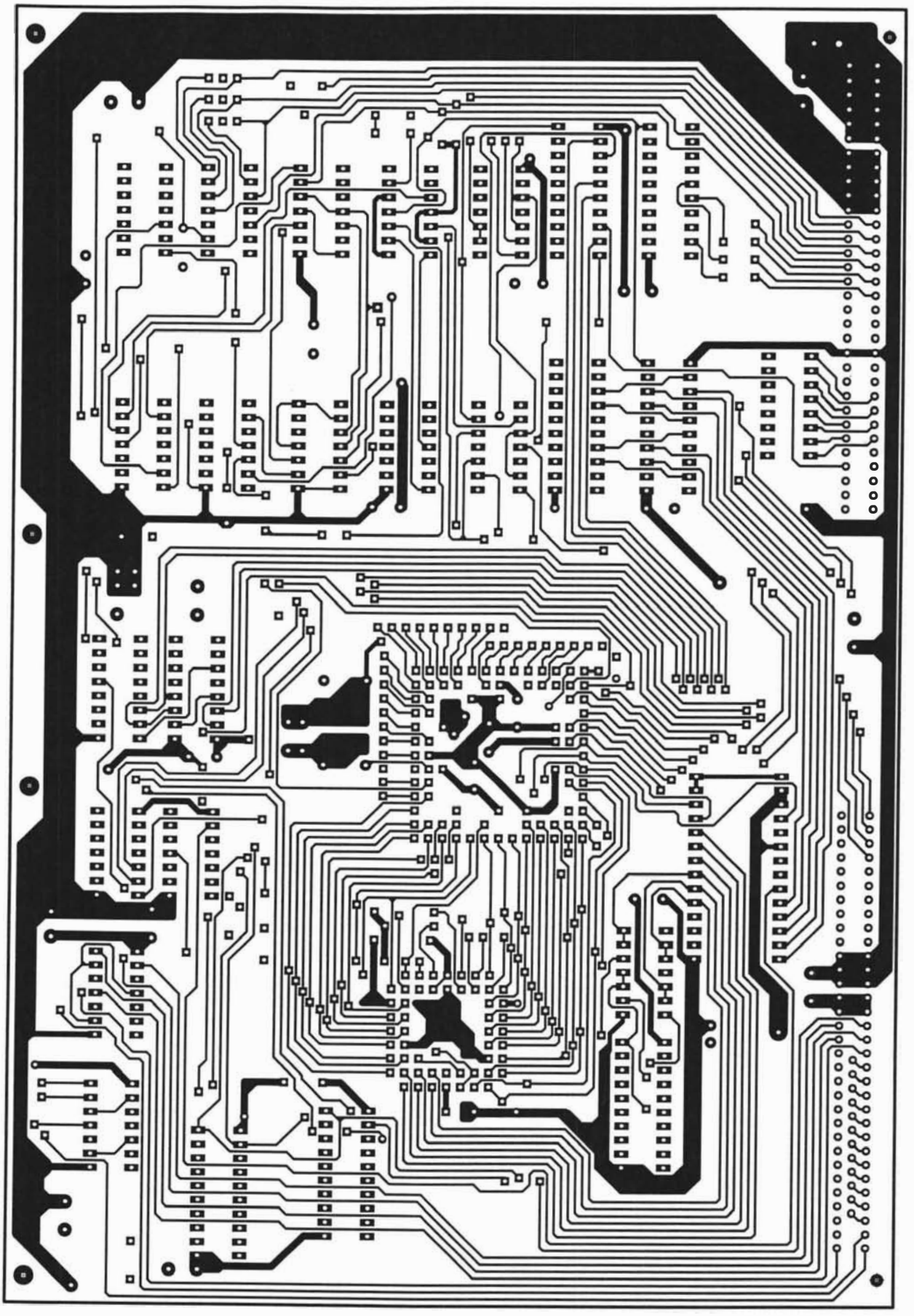


Figure 6.2 The bottom layer of DSP board

All the boards used were produced in the Electronic Workshop of the Physics Department, Massey University. The board material (usually 1/16 inch of so-called FR-4 board, a fire-resistant epoxy-bonded fibreglass) comes clad on both sides with copper. The boards are exposed with photo-resist film through a full-size negative and the unexposed part is chemically removed.

After drilling holes using an automated drilling machine keyed to the full-size board, the IC sockets, capacitors and resistors were soldered onto the board. Many bypass capacitors were put around the ICs, the memory and the DSPs. All connections to the boards were brought out through Euro DIN connectors. Figure 6.3 shows a completed DSP board.

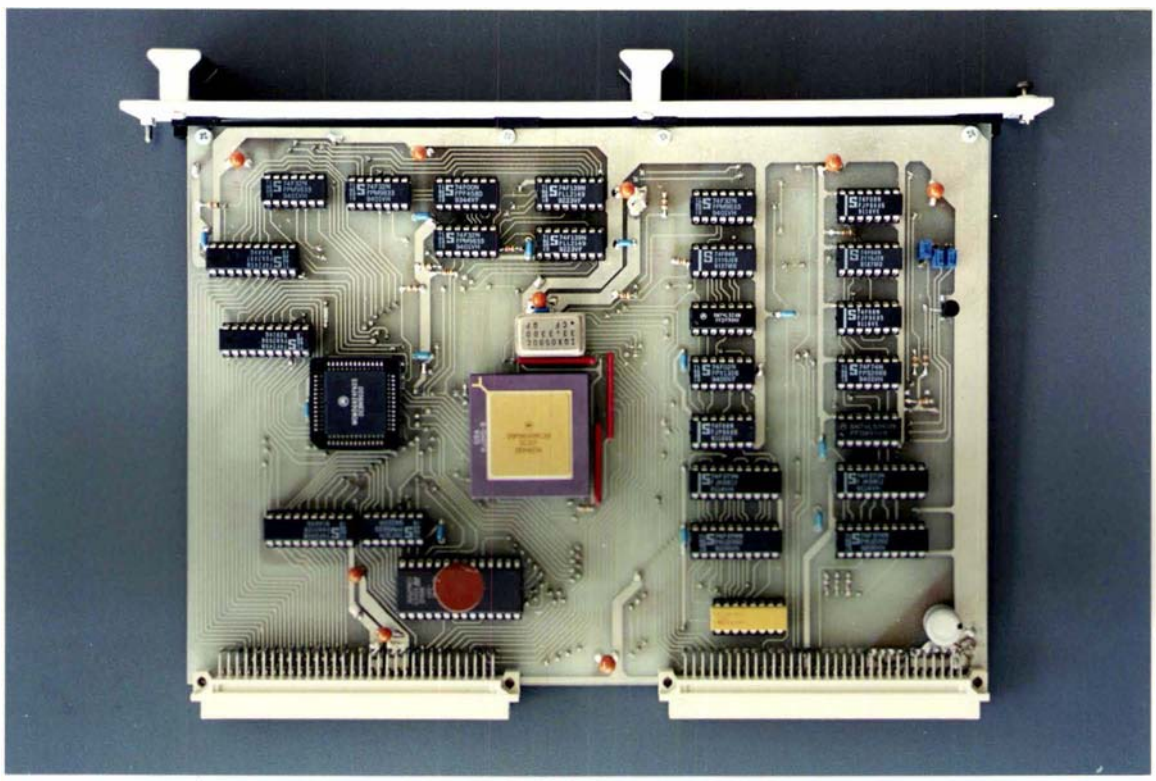


Figure 6.3 DSP printed circuit board

6.3 Instrument Construction

After production of the PC boards, the completed printed circuits boards were mounted in an enclosure and connected to power supplies, panel controls and connectors, and other necessary circuitry. The aim was to assemble the instrument so that the circuits were neatly mounted and accessible for testing and repair.

6.3.1 Circuit card mounting

This correlator system has ten printed circuit boards, so the best way to arrange things is to use a card cage. This is a flexible assembly with guides for individual cards to slide into and aligned holes along the rear so that Euro DIN connectors can be mounted to mate with the cards. This method spreads the circuit over a number of plug-in circuit boards and so achieves a modularity which permits ease of repair and yet is flexible for any further development. This power supply is also mounted in the card cage.

The correlator is constructed using the Eurocard sub-rack modular system (6Ux84E). The sub-rack provides a comprehensive range for all commonly used sizes of equipment. All sub-racks feature ruggedly designed aluminium extrusions for maximum strength and all joints are pre-tapped for ease of assembly and reliability. The sub-racks are designed to accommodate printed circuit boards (233.3mm wide × 160mm high).

Two-part 64-pin Euro DIN connectors are used with one mating portion soldered to the circuit board as a component. Also multiple groups of Euro DIN connectors are put on a single mother board card. Cards are spaced 0.5 inch apart to allow plenty of room, if necessary more cards could be add later.

The printed circuit cards are mounted vertically. The front panel can be removed and plug-in extension cards used to make the circuit cards accessible.

Computer and correlator connections are brought out by using flat ribbon cable terminated in DIP plugs. Such cables plug into IC sockets on the mother board of the correlator. The cable is about 0.5 meter long. This ribbon cable also connects to the computer I/O board via Euro DIN connectors which are polarised to prevent wrong connection.

6.3.2 Grounding

The correlator system uses 33 MHz clock, so a low-inductance grounding system must be provided to the set of boards by way of the connectors at the ends of the boards. Several methods are used on the circuit boards. Firstly, the critical circuitry of the DSP system is placed in one limited well grounded area. Throughout that area, wide surface lines are used to make ground connections between separated portions of the circuit. Secondly, the boards are interconnected with a PC motherboard which is provided with wide ground traces to achieve better ground distribution. Finally, we use as many redundant ground connections as possible - (multiple connections to the ground, doubled connector pins and wires, etc.) - to reduce the inductance through which ground currents may flow.

6.3.3 Connection

A motherboard back plane connection (a PC board designed just to hold the card connection sockets) is used to make all the signal bus and power-supply connections. Connections are made with both Euro DIN connector sockets and flat cable terminated with a connector to mate with a plug on the board. With Euro DIN and ribbon connectors, the connections will support the connector adequately, so no extra connector supports are required.

Care must be taken in the design and layout of the circuit to ensure the stability of the signal during high speed signal transfer. The correlator system is designed with large ground lines to minimise any induced spikes, and all critical circuits are kept together on one card to keep wiring capacitance to a minimum. It is important to ensure that all signals are clean and free from noise, so unused pins of the Fast logic are tied either to ground or to the power supply, as appropriate.

Transmission line effects due to the length, inductance and capacitance of the signal line must be considered when the sub circuits are connected together by a data bus. The propagation delay differences in different paths of combinatorial circuits may cause the generation of an unwanted pulse or a logic spike. The chip enable signal is asserted after the other inputs, allowing sufficient time to avoid any momentary transients in the outputs. Particular attention is given to the quality of the HOST Enable (HEN*) signal.

Each Vcc pin on the DSP56001 is provided with a low impedance path to +5 volts. Each GND pin is also provided with a low impedance path to ground. And the Vcc power supply is bypassed to ground using 0.2 μF capacitors located underneath the chip. During the layout, the printed circuit trace interconnection lengths were minimized to minimise undershoot and reflections caused by fast output switching times. This is especially important for the address and data buses as well as the RD*, WR*, IRQA*, IRQB*, and HEN* pins.

6.3.5 Cooling

The instrument is cooled adequately by simple convection cooling through perforated top and bottom covers. Circuit boards are better ventilated when mounted vertically, although heat dissipation on the circuit cards is negligible. The system has a large card spacing permitting relatively unobstructed airflow, between the cards, from the bottom to the top of the instrument.

The completed multiple tau autocorrelator is shown in Figure 6.4.

Figure 6.4 Multiple tau autocorrelator

Figure 6.5 shows the front control panel of the multiple tau correlator system.

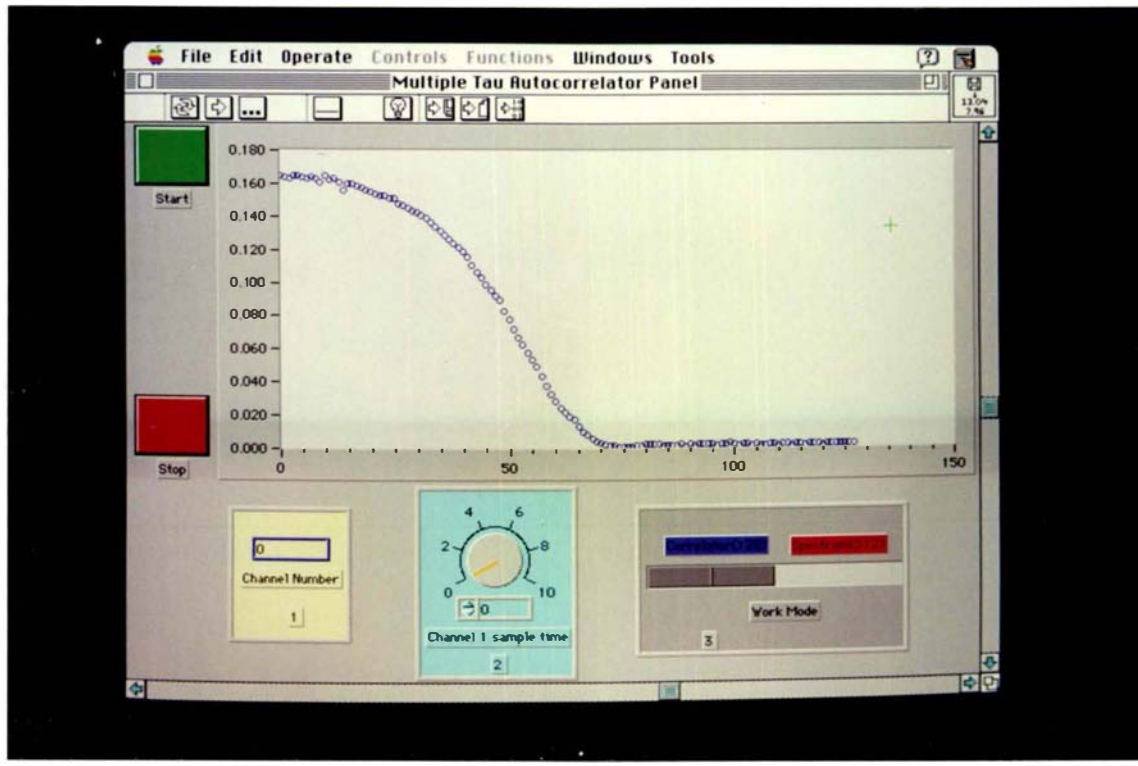


Figure 6.5 Autocorrelation function displayed by the control computer of the multiple tau correlator

6.4 Autocorrelator Tests

This section describes the correlator tests and demonstrates the correct operation of the correlator. The test procedures were carried out on separate units and also on the whole correlator. These tests were carried out both during construction and after completion of the instrument.

The correlator system was tested step by step during the construction. Each circuit board was tested to ensure that it worked properly individually. Then the whole system was put together and tested with the control computer. In this section some of the test results are reviewed.

6.4.1 Sample time clock test

The four output sample time clock pulses were recorded to see if they were correctly controlled by the host DSP. The host DSP sent a control byte to the sample time clock control register. The output sample times are shown in Figure 6.6 as predicated. For the test, number $4448F0 was sent to the control register. Correct sample time sequences were generated.

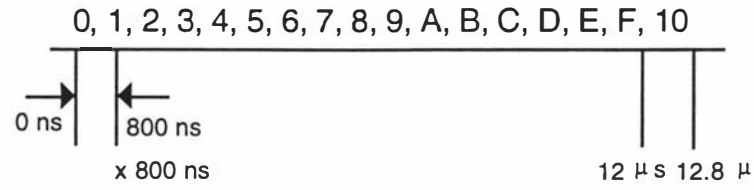


Figure 6.6 Sample time clock time sequence controlled by the first byte of control register

6.4.2 Counter test

Counter boards were first tested using the DSP56001 development system. A counter board was connected to the ADS56001 board by a 96 pin edge connector. The sample time clock time pulses were generated by the sample time clock board. Constant 50 ns width TTL pulses were generated by a PM 5715 PHILIPS pulse generator, and the results were obtained using the DSP56001 development system software. The data obtained from the counter board by the ADS56001 board showed that the counter was functioning correctly.

Sine wave inputs were also used to test the counter board. The 50 ns width TTL input pulses were achieved by using two 3311A Hewlett-Packard function generators and a PM 5715 PHILIPS pulse generator as shown in Figure 6.7. One HP function generator acted as the waveform generator, and the other acted as a voltage controlled oscillator

producing TTL level pulses. These pulses were connected to the pulse generator TRIGG/GAT IN input to trigger the corresponding TTL 50 ns width pulses. A correct sine waveform was obtained from the counter board.

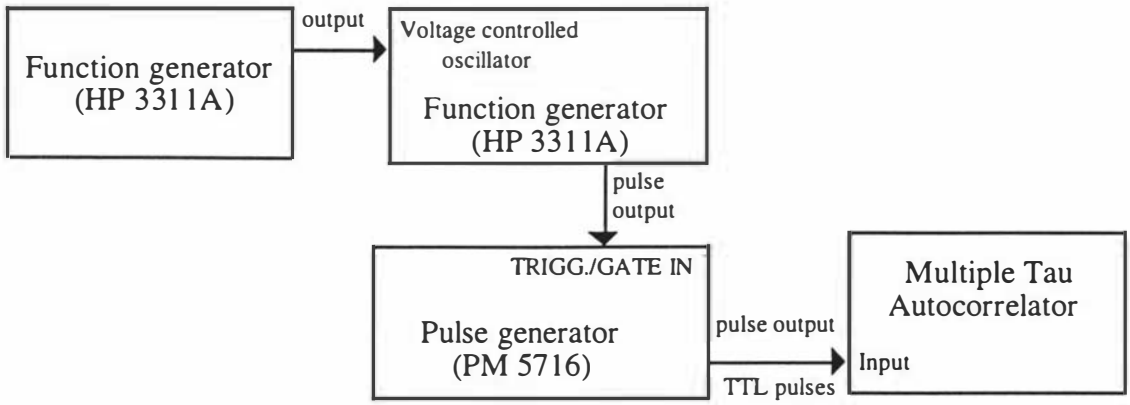


Figure 6.7 Experimental arrangement for correlator test

This experimental arrangement was also used for the correlator test system.

6.4.3 DSP board test

The DSP boards were tested using specially designed software resident in the EPROM. The program produced special pulses at test points in the circuit, and each test point was monitored with a 100 MHz oscilloscope. A separate reset pulse generator was connected to the DSP board to reset the DSP board. After the reset stage, the DSP would load the test program resident in the EPROM into internal RAM and run the program. Correct pulse sequences were monitored at the test points. Proper functioning of the DSP56001 and the EPROM was thereby established.

Subsequently a more complicated program was used to test the RAM system, the decoder system and the I/O system. Test results show that the DSP board worked as designed.

6.4.4 Interface test

The interface system was tested with the Macintosh IIvx, interface I/O card (NB-DIO-24), the LabVIEW program, and the DSP board. Special communication programs were designed using both LabVIEW language (resident in the Macintosh IIvx) and DSP56001 assembly language (resident in the DSP board EPROM). The communication flag was monitored using an oscilloscope. Correct communications were established.

After these tests, the desired monitor program was loaded into the EPROM, and a test processing program was loaded from the Macintosh IIvx into the DSP board RAM. Then the procedure for sending data to the computer was tested. The correct program was

transfered into the DSP board RAM, and correct data were received by the Macintosh IIvx computer.

6.4.5 Multi-DSP test

Communication between the host DSP and the slave DSPs was tested by specially designed software. The data sent by three slave DSPs were combined by the host DSP and sent to the Macintosh IIvx for viewing. Communications were established as designed.

6.4.6 Correlator system test

The experimental arrangement used in the counter test was also used in the correlator test. When the START button is pressed, LabVIEW sends a 'Load' command to the correlator monitor program, the monitor program acknowledges LabVIEW and loads the processing program into PRAM and starts the processing program. Then the results are transferred to LabVIEW by the Macintosh IIvx - Correlator interface.

A constant input was used for checking operation of the counter. A special LabVIEW program was built to interogate the input buffer memory of each DSP. The signal was generated by a PM 5716 pulse generator set to a frequency between 2.501 - 2.502 MHz (period 399.8 ns - 400.2 ns). The smallest sample time clock was programmed at 3.2μs and also at 9.6μs. The data read out from the first DSP are shown in Figure 6.8 and Figure 6.9 and are as expected.

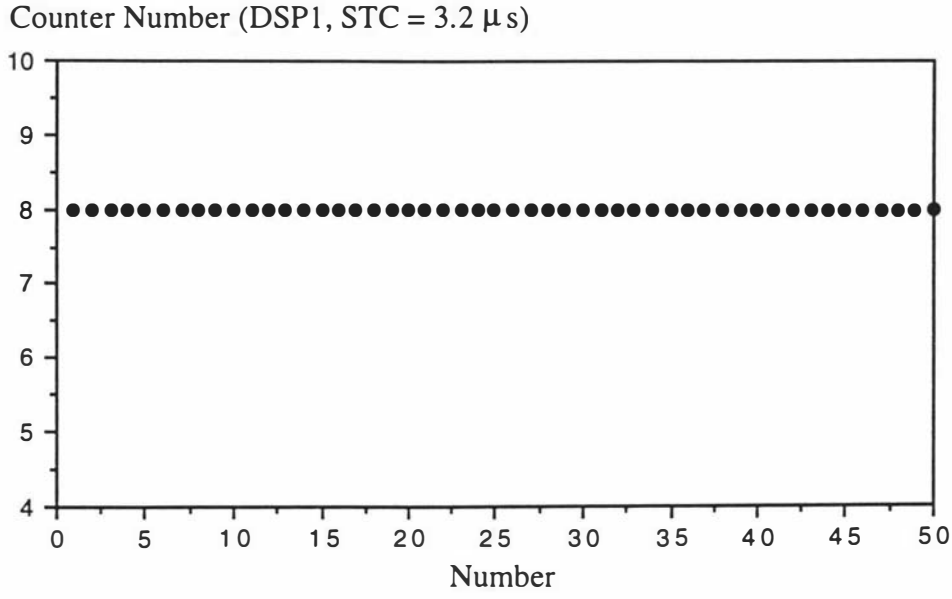


Figure 6.8 Counting data loaded by DSP (3.2μs sample time)

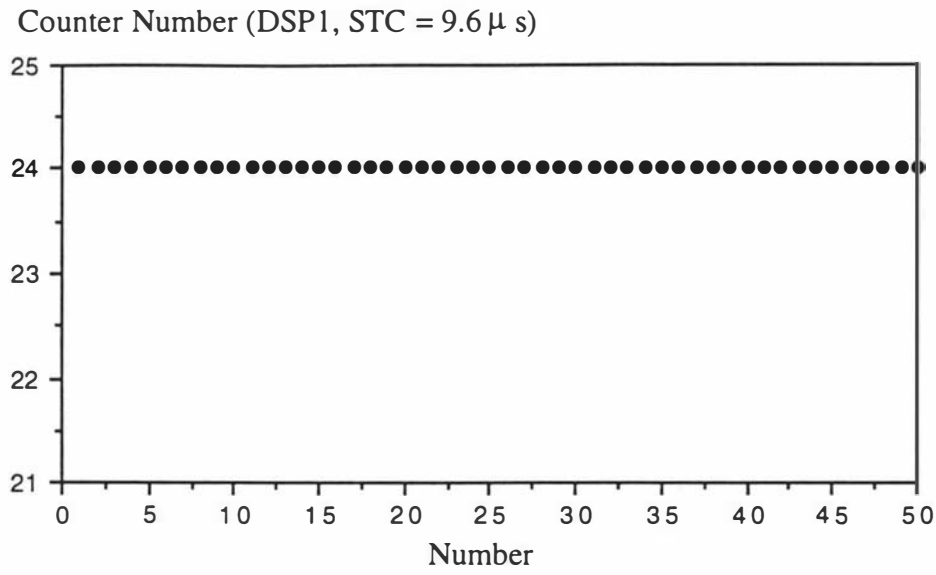


Figure 6.9 Counting data loaded by DSP (9.6μs sample time)

A 2.0 Hz sine wave input was analysed by the correlator. The smallest sample time was chosen as 9.6 μs. The normalised correlation function has the expected form

$$g^{(2)}(t) = \cos(2\pi ft) \tag{6.1}$$

where f is the signal frequency. The result of measuring the correlation function of a 2.0 Hz sine wave is shown in Figure 6.10.

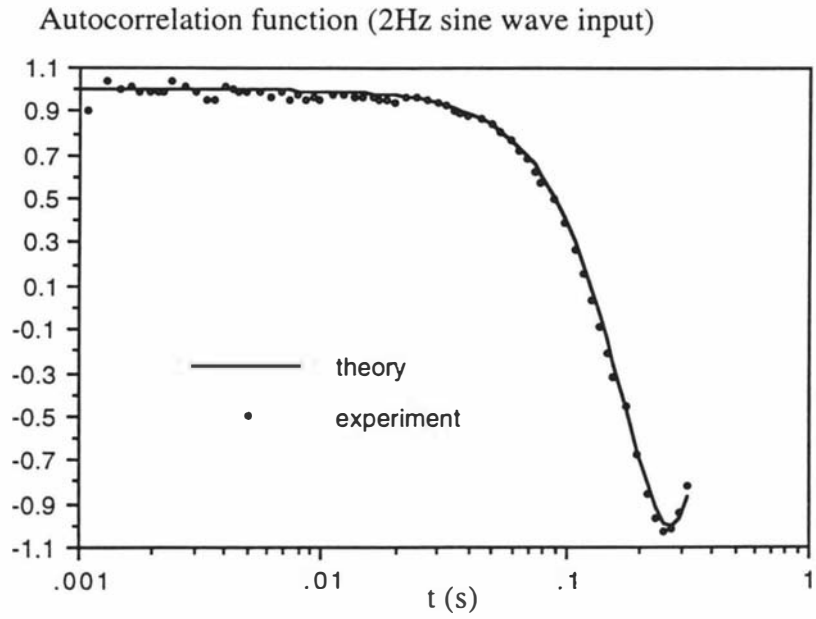


Figure 6.10 Correlation function, 2.0 Hz sine wave

The correlation function obtained had the expected cosine form, demonstrating that the correlator works as designed. Figure 6.11 shows the test data obtained from an 8.0 Hz sine input.

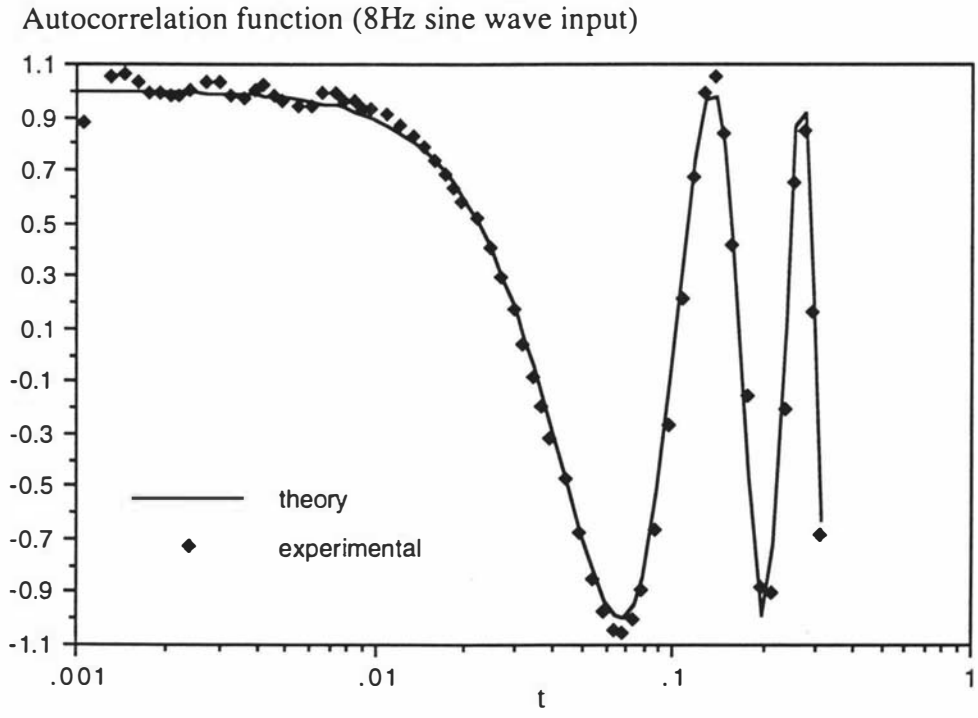


Figure 6.11 Correlation function, 8.0 Hz sine wave

7 INVESTIGATIONS OF TERNARY POLYMER SOLUTION USING THE MULTIPLE TAU CORRELATOR

The intensity autocorrelation function of laser light scattered from semidilute ternary solutions of random coil polymers can exhibit a range of decay processes. Hence such solutions can provide data that are ideal for demonstrating the effectiveness of the multiple tau correlator.

In this chapter two dynamic light scattering experiments on ternary polymer solutions will be described and discussed. The correlator output data are analysed using the CONTIN program developed by Provencher (Provencher, 1984).

The performances of the multiple tau and a linear correlator are compared when both correlators receive the same photomultiplier pulse train simultaneously. It is shown that under some circumstances the interpretation of the linear correlator output can introduce artefacts.

7.1 Experimental System

The understanding of the properties of linear flexible chain polymer solutions has increased greatly in recent years through the application of scattering techniques, which probe small scale spatial correlations of concentration fluctuations. The length scale probed is expressed in chapter 2 as

$$q = \left(\frac{4\pi}{\lambda}\right) n \sin\left(\frac{\theta}{2}\right) \tag{7.1}$$

where λ is the wavelength of the incident light in a vacuum, n is the refractive index of the medium and the measurement is performed at the angle θ, see Figure 2.1.

The technique of photon correlation spectroscopy of scattered laser light is well established and enjoys widespread application, particularly in the study of the solution properties of macromolecules and other particles in the size range 10-1000 nm. Figure 7.1 is a schematic diagram of the laser light scattering system used in this work.

This is a typical DLS experimental arrangement. The light source is a Spectra-Physics 165-08 Ar+ ion laser operating well above threshold. The exit aperture of the laser is reduced to ensure operation in the TEM mode. The 488 nm line is selected and a variable neutral density filter is used to reduce the intensity of the laser light, output powers of up to two hundred milliwatts at 488 nm are typically used. The laser beam is focused into the cylindrical scattering cell by a 10 cm focal length lens, and the light scattered through an

angle θ is imaged onto an ITT FW130 photomultiplier tube. The photomultiplier pulses are converted to TTL level by an amplifier/discriminator, as discussed in chapter 3.

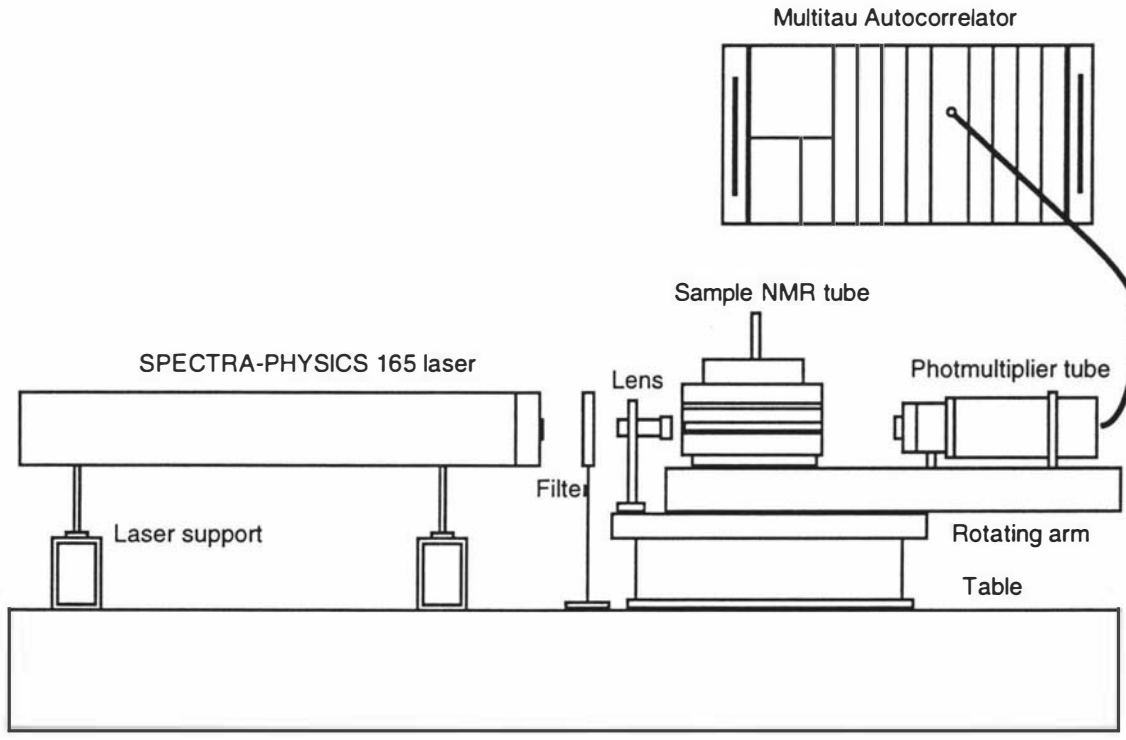


Figure 7.1 DLS experiment apparatus

A closed circuit cooling system containing distilled water in thermal contact with a heat exchanger was used to cool the laser. The cylindrical scattering cell is positioned at the centre of a Malvern Instruments RR102 spectrometer, consisting of a refractive index matching and temperature control bath, a spectrometer rotation unit and a photomultiplier assembly including collection optics, photomultiplier tube and photoelectron pulse amplifier/discriminator. A Malvern Instruments RR56 temperature controller was used to control the temperature. The standardised photomultiplier output pulses are processed by the multiple tau correlator to generate the photon count autocorrelation function (intensity autocorrelation function of the scattered light). The correlator is interfaced to a Macintosh IIvx computer which performs all the necessary data processing tasks.

7.2 Data Analysis

Rapid advances in experimental aspects of the DLS technique, particularly autocorrelator design and construction, have paralleled developments in data analysis. This combination of developments has permitted a fresh approach to several problem areas such as those in which several relaxation modes feature in the time correlation function. Such problems

stretch to the limit the capacity of the technique both as regards the time window that is accessible and the achievable resolution.

The full autocorrelation function, $G^{(2)}(q,t)$, of the scattered intensity was obtained using the multiple tau autocorrelator. The intermediate scattering function $g^{(1)}(q,t)$ is related to the measured intensity autocorrelation function through the Siegert relation

$$\left(\frac{G^{(2)}(q,t)}{B}-1\right)^{\frac{1}{2}} \approx \int_0^\infty A(\tau)e^{-\left(\frac{t}{\tau}\right)}d\tau = g^{(1)}(q,t) \qquad (7.2)$$

The analysis of the autocorrelation function was made on-line with the Macintosh IIvx computer.

For the analysis of the measured autocorrelation curve, an inverse Laplace transformation is performed using the constrained regularisation method CONTIN to obtain the distribution $A(\tau)$ of relaxation times, or $A(\Gamma)$ the distribution of relaxation times. The experimental autocorrelation function generated by the autocorrelator is formatted into a special input data set for CONTIN. More than 40 Control Variables are defined in the CONTIN program to make the analysis as flexible as possible. The Control Variable NLINF is of particular interest for this variable determines whether or not a base line is fitted to the data. If NLINF is set to zero the theoretical base line is used, if NLINF is set to unity an experimental base line is calculated. CONTIN is usually run in a default mode which calculates the $A(\tau)$ distribution twice, once with NLINF set to zero and once with it set to unity. The $A(\tau)$ distributions should then be compared by the operator. However, there is no substitute for good experimental technique and the experimenter should take every care to ensure that the solutions are free of dust or other high molar mass contaminants. Also it is important for the autocorrelation function to be measured at lag times so long that the autocorrelation function has decayed well into the noise.

The output of CONTIN includes a statistical parameter, P, "probability to reject" which is calculated for each $A(\tau)$ distribution computed. The operator is advised to choose that $A(\tau)$ distribution with P closest to 0.5.

7.3 Theory of DLS from Ternary Polymer Solutions

7.3.1 Ternary polymer solutions far from phase separation

In the last decade, the scattering properties of ternary polymer solutions have been the subject of intensive studies, and there has been wide-spread interest in diffusion in ternary polymer systems, the dynamic behaviour of ternary solutions of random coil polymers has been investigated by various researchers.

1. D_C and D_I

Dynamic light scattering from ternary random coil polymer solutions can be described by the Borsali-Benmouna mean field theory, provided the several assumptions made in the development of the theory are valid (Benmouna, Benoit, 1987). One of these assumptions is that the solvent is "equally good" for the two polymers, ie, the solvent is of equal and good quality for both polymers. Another assumption is that the compatibility of the polymers can be represented by a single parameter χ, the polymer-polymer interaction parameter.

This theory introduces two polymer diffusion coefficients, the fast cooperative diffusion coefficient (D_C) describing the relaxation of concentration fluctuations, and the slow interpenetration diffusion coefficient (D_I) describing the relaxation of composition fluctuations.

The total intermediate scattering function is assumed to be given by

$$g^{(1)}(t) = A_+ e^{-D_C q^2 t} + A_- e^{-D_I q^2 t} \tag{7.3}$$

where q is the scattering vector and t is time. D_C and D_I are given by

$$D_C, D_I = \frac{D_{S2}}{P(q)}\left[A \pm 0.5\sqrt{F - E + Q}\right] \tag{7.4}$$

where

$$A = 0.5\left[\frac{N_2}{N_1} + 1 + \upsilon\phi N_2\right] \tag{7.5}$$

$$E = -4x\upsilon\phi N_2\left[1 - \frac{N_2}{N_1}\right] \tag{7.6}$$

$$F = 8x(1-x)(\upsilon\chi N_2)^2 \frac{P(q)\chi}{\upsilon} \tag{7.7}$$

$$Q = 4\left(A - \frac{N_2}{N_1}\right)^2 \tag{7.8}$$

In the above ϕ is the polymer volume fraction, x the relative abundance of polymer 2, N_1 and N_2 are the degrees of polymerisation of polymers 1 and 2, D_{S2} is the self diffusion coefficient of polymer 2, $P(q)$ is the particle form factor, and υ and χ are the polymer excluded volume and polymer-polymer interaction parameters respectively.

The mode amplitudes A_+ and A_- are also complicated functions of N_1, N_2, ϕ, x, υ and χ, and moreover they also depend on the refraction indices of the polymers and the solvent. In the special case, that polymer 1 is isorefractive with the solvent and polymer 2 is present as a trace only, A_+, the amplitude of the fast mode vanishes and the field autocorrelation function of the scattered light reduces to a single exponential with decay rate governed by the interpenetration diffusion coefficient D_I.

When $N_1 = N_2$, equation 7.4 can be simplified to

$$\Gamma_c = \Gamma_s(q)\left[1 + V\Phi NP(q)\right] \tag{7.9}$$

and

$$\Gamma_I = \Gamma_S(q)\left[1 - \frac{\chi P(q)}{\chi_C}\right] \tag{7.10}$$

$$\text{where } \Gamma_S(q) = q^2\left(\frac{kT}{N\zeta}\right)\frac{1}{P(q)} = q^2\frac{D_S}{P(q)} \tag{7.11}$$

k is the Boltzmann constant, T the absolute temperature, and ζ is the friction coefficient of a monomer unit. χ_c is the critical value of the polymer-polymer interaction parameter, $\chi_c = 1/[2\phi Nx(1-x)]$.

This Borsali-Benmouna mean field theory has been tested in many experimental investigations (Daivis & Pinder, 1993), (Giebel, 1993), (Brown, 1993) etc., and found to give an adequate description of DLS from ternary polymer solutions formed with equally good solvents far removedfrom phase separation.

7.3.2 Ternary polymer solutions close to phase separations

The simple Borsali-Benmouna theory of DLS from ternary polymer solutions ignores the so-called "memory effects", it also adopts a simple treatment of the effects of hydrodynamic interactions. The theory is able to provide an adequate account of the behaviour of ternary polymer solutions that are far from phase separation where the effects of hydrodynamic interactions are negligible.

But ternary polymer solutions close to phase separation are characterised by the correlation length ξ diverging so long range hydrodynamic effects are significant in the critical region where ξ varies with reduced temperature as

$$\xi \propto \varepsilon^{-n} \tag{7.12}$$

where ε is the reduced temperature given by $(T-T_c)/T$, and T and T_c are the temperature and critical temperature of the solution. The mean field theory, three dimension Ising, and Fisher renormalisation values for this critical exponent n are 0.5, 0.63 and 0.71 respectively. [Note this exponent is usually represented by ν, but in this work ν is used to represent the polymer excluded volume parameter so the symbol n is used here instead.]

Benmouna et al (1993) have extended the simple Borsali-Benmouna theory to include hydrodynamic interactions for the symmetric case in which the polymers have equal degrees of polymerisation, equal friction coefficients, equal and opposite refractive index increments and equal concentrations. They found that only the interpenetration mode was visible and its decay constant Γ_I was given by

$$\Gamma_I = q^2 D_{S2}\left(\frac{1}{P(q)} - \frac{\chi}{\chi_c}\right) + \frac{q^2 kT}{(2\pi)^2 \eta}\int dk \cdot f\left(\frac{k'}{q}\right)\left\{\frac{1-\chi/\chi_c P(q)}{1-\chi/\chi_c P(k')}\right\} \quad (7.13)$$

The interpenetration diffusion coefficient can be defined in the usual way as $D_I = \Gamma_I/q^2$

$$\therefore D_I = \Gamma_I/q^2 = D_{S2}\left(\frac{1}{P(q)} - \frac{\chi}{\chi_c}\right) + \frac{kT}{(2\pi)^2 \eta}\int dk' \cdot f\left(\frac{k}{q}\right)\left\{\frac{1-\chi/\chi_c P(q)}{1-\chi/\chi_c P(k')}\right\} (7.14)$$

The first term represents the Rouse contributions as predicted by the simple Borsali-Benmouna theory and is not new, the second term represents the long range hydrodynamic back flow effects which may be important in the critical region. Benmouna et al approximated $1/P(q)$ with $1+q^2 R_g{}^2/2$ in the second term and obtained

$$\Gamma_I/q^2 = D_I = D_{S2}\left[\frac{1}{P(q)} - \frac{\chi}{\chi_c}\right] + \frac{kT}{6\pi\eta\xi} F(q\xi) \quad (7.15)$$

Where $F(x)$ is the Kawaski function which is very nearly $3/4+x$ for $x \geq 2$, and approaches the value 0.75 as x approaches zero.

The unusual approximation for $P(q)$ was used to cast the integral into an analytical form. This approximation fits the higher qR_g range better than the low qR_g range.

This theory has not yet been extended to the general case of arbitrary polymer molar mass, refractive index increments and composition. However, one can reasonably assume that the essential features of the theory will remain the same. In particular one can assume that there will be two contributions to D_I, a Rouse term given by the simple Borsali-Benmouna theory and an hydrodynamic interaction term which will be similar to the second term in equation 7.15.

It should be appreciated that this extension of the Borsali-Benmouna theory includes hydrodynamic interactions but there are other dynamic processes occurring in ternary polymer solutions, some of which are susceptible to observation in a DLS experiment. Such process may be other fluid flow mechanisms (Brown, 1992), (Wang, 1981) or polymer aggregation (Konac, 1991) these processes are not included in the extended theory of Benmouna et al (1993).

Eqn (7.15) describes the variation of the interpenetration decay constant with q. It is convenient to discuss the two special cases which approximately describe the experimental investigations to be presented below.

Case A type solutions.

The solution conditions are:

(a) Both polymers sufficiently small for $P(q)$ to be taken as unity.

(b) Polymers have equal degrees of polymerisation, $N_2 = N_1$.

(c) Polymer 1 isorefraction with the solvent.

(d) Polymer 2 present as a trace only.

Under these conditions eqn (7.15) can be rewritten as:

$$\Gamma_I/q^2 = D_I = D_{S2}\left[1-\frac{\chi}{\chi_c}\right]+\frac{kT}{6\pi\eta\xi}F(q,\xi) \tag{7.16}$$

(i) If $q\xi \leq 0.5$ then $F(q,\xi) \approx 3/4$, and

$$D_I = D_{S2}\left[1-\frac{\chi}{\chi_c}\right]+\frac{3}{4}\frac{kT}{6\pi\eta\xi} \tag{7.17}$$

which is independent of q.

As the temperature is lowered towards phase separation χ approaches χ_c and ξ diverges, so both terms approach zero and D_I is expected to display a "critical slowing down" as phase separation is approached.

Also D_I can be used to define a dynamic correlation length ξ_d

$$D_I = \frac{kT}{6\pi\eta\xi_d} \tag{7.18}$$

ξ_d is expected to diverge as T approaches T_c.

(ii) If $q\xi > 2$ then $F(q,\xi) \approx q\xi$.

and $$\Gamma_1/q^2 = D_1 = D_{S2}\left[1 - \frac{\chi}{\chi_c}\right] + q\frac{kT}{6\pi\eta} \tag{7.19}$$

D_1 is q dependent due to the hydrodynamic interaction term.

The Rouse contribution does display a "critical slowing down", and

$$\lim_{q\to 0} D_1 = D_{S2}\left[1 - \frac{\chi}{\chi_c}\right] \tag{7.20}$$

Summary of results for case A type solutions.

(i)

1. D_1 is q independent.
2. D_1 displays a "critical slowing down" as T_c is approached.
3. D_1 can be used to define the dynamic correlation length ξ_d.
4. Hydrodynamic effects are "of same nature" as Rouse effects.

(ii)

1. D_1 is linearly dependent on q, the scattering vector.
2. Intercept on Γ_1/q^2 axis displays a "critical slowing down".

Case B type solutions.

The solution conditions are:

(a) Minority polymer molar mass so large that $1/P(q)$ is approximated by $1+q^2R_g{}^2/3$.

(b) $N_2 >> N_1$.

(c) Majority polymer isorefractive with solvent.

(d) Polymer relative abundances 3:1 by mass.

In this case both the fast and the slow modes are visible, however, only the slow mode will be discussed in detail here because this mode exhibits critical behaviour of phase separation before the fast mode exhibits critical behaviour of precipitation as the solution temperature is reduced.

No expression equivalent to equation 7.14 is available in this case. However one can expect D_f to be composed of two terms, a Rouse term given by the simple Borsali-Benmouna theory and an hydrodynamic term similar to that of equation 7.14.

The Rouse term is far more complicated in this case, but one can expect it to display two features, a q^2 dependence due to $P(q)$ and a critical slowing down as phase separation is approached. Hence, following Benmouna et al (1993) one may write the Rouse term as

$$D_{\text{Rouse}} = D\left[\frac{1}{P(q)} - \frac{\chi}{\chi_c}\right] \tag{7.21}$$

where D is a known but complicated function of polymer concentration. Since $1/P(q)$ can be approximated as $1+q^2R_g^{\,2}/3$ the Rouse term becomes

$$D_{\text{Rouse}} = \Gamma_1/q^2 = D\left(1+q^2R_g^2/3-\chi/\chi_c\right) \tag{7.22}$$

Including the hydrodynamic interaction term from equation (7.15), this becomes

$$D_f = \Gamma_1/q^2 = D\left(1+q^2R_g^2/3-\chi/\chi_c\right)+\frac{kT}{6\pi\eta\xi}F(q,\xi) \tag{7.23a}$$

where $F(q,\xi) = 0.75 + q\xi$.

Following Benmouna et al (1993) R_g^2 can be expressed as $2\xi^2(1-\chi/\chi_c)$ so equation 7.23a can be rewritten as:

$$\Gamma_1/q^2 = D_f = D\left(1-\frac{\chi}{\chi_c}\right)\left(1+\frac{2}{3}q^2\xi^2\right)+\frac{kT}{6\pi\eta\xi}F(q,\xi) \tag{7.23b}$$

Also ξ is expected to be larger in this case because the minority polymer molar mass is greater, so $F(q,\xi)$ can be approximated as $q\xi$. Hence the expression for D_f can be written in another form as:

$$D_f = \Gamma_1/q^2 = D\left(1-\frac{\chi}{\chi_c}\right)\left(1+\frac{2}{3}q^2\xi^2\right)+\frac{kT}{6\pi\eta}q \tag{7.23c}$$

Summary of results for B type solutions

1. From equation 7.23c the relative magnitude of the Rouse term decreases as χ approaches χ_c, ie. the hydrodynamic term becomes more significant as phase separation is approached as is to be expected.

2. The Rouse term introduces a q^2 dependence to Γ_1/q^2.

3. The hydrodynamic term introduces a q dependence to Γ_1/q^2, and this q dependence is expected to increase as phase separation is approached, because the relative magnitude of the Rouse term decreases.

4. The small q asymptote of Γ_1/q^2 is $D(1-\chi/\chi_c)$, it should demonstrate a "critical slowing down" as phase separation is approached.

5. At high temperature, where $\chi<<\chi_c$ and hydrodynamic term is negligible, a graph of Γ_1/q^2 against q^2 is expected to approximate a straight line of slope $R_g^2 D/3$ and intercept D.

6. At low temperatures, close to phase separation, a graph of Γ_1/q^2 against q^2 is expected to be nonlinear due to the q dependence of the hydrodynamic term.

7.4 Ternary Polymer Solution Experiment I

Following the usual optical alignment procedure employed for DLS experiments (Daivis, 1989), a set of measurements were made on a ternary polymer solution formed by dissolving 233000 molar mass polystyrene together with a trace amount of 330000 PMMA in thiophenol. The PMMA volume fraction was 0.00092 and the total polymer volume fraction was 0.080. This solution studied was very close to phase separation at 25°C, indeed phase separation could be induced by lowering the solution temperature to 13°C. The PMMA was presented as a trace amount only and the PS was very nearly isorefaction with the solvent, these conditions ensured that the cooperative mode was not detected.

This solution can be regarded as fulfilling the conditions for a case A type ternary polymer solution. Observations were made over scattering angles ranging from 20° to 135°, as the solution temperature was varied from 13°C to 45°C. Several runs were made at each angle at each temperature. Figure 7.2 shows a typical correlation function collected from the PS/PMMA/thiophenol system.

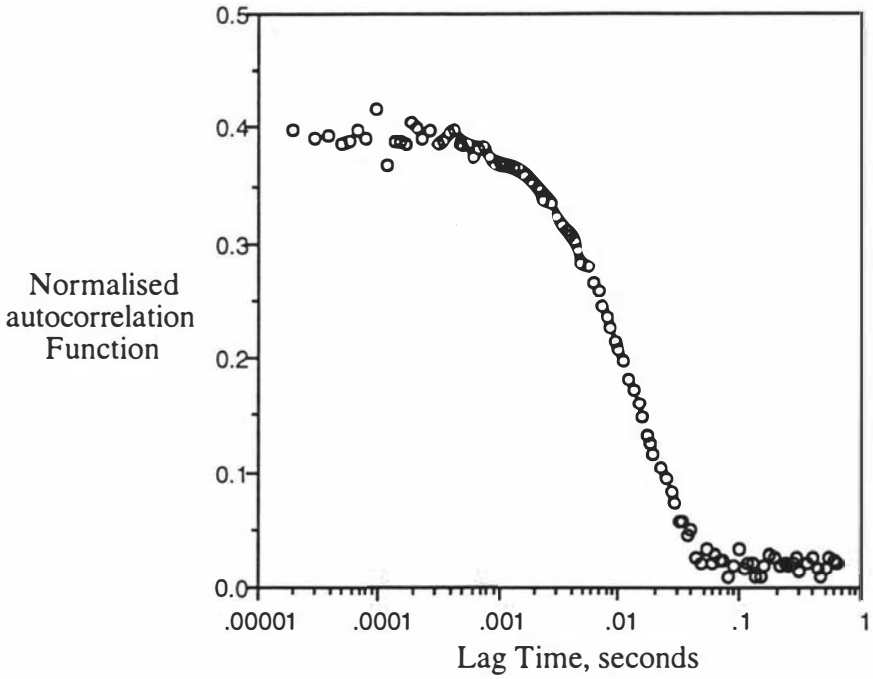


Figure 7.2 Normalised autocorrelation function versus time scale for the PS/PMMA/thiophenol sample. The intensity autocorrelation function $g^{(2)}(t)$ is obtained using the multiple tau autocorrelator, the scattering angle was 105° and temperature was 16°C.

The corresponding relaxation time distribution obtained by Laplace inversion is shown in Figure 7.3. Plotted in the form $A(\Gamma)$ versus $\mathrm{Log}_{10}(\Gamma)$.

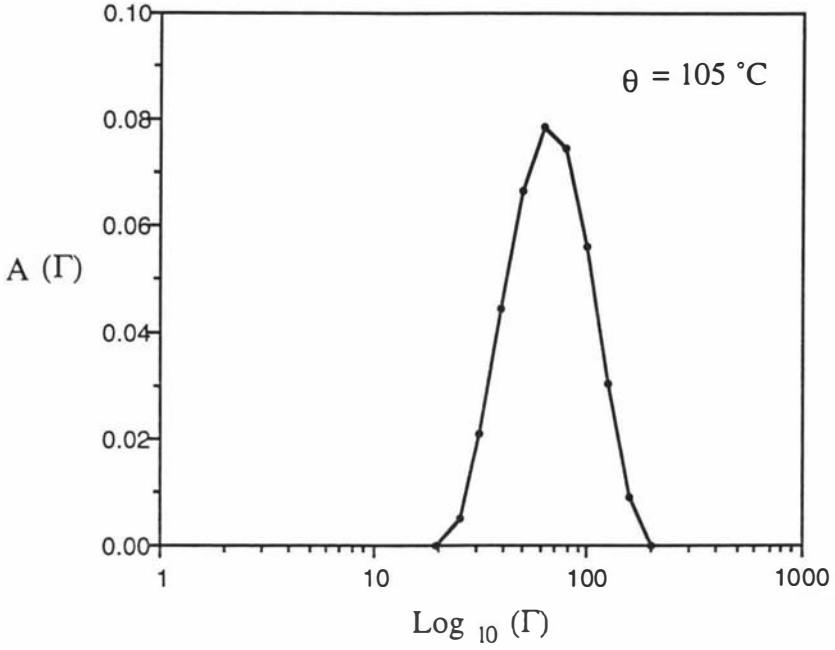


Figure 7.3 Relaxation time distribution corresponding to the autocorrelation function shown in Figure 7.2 obtained using Laplace inversion (CONTIN). As expected only a single slow mode is obtained.

Figure 7.4 shows the experimental data and the theoretical fit generated by CONTIN.

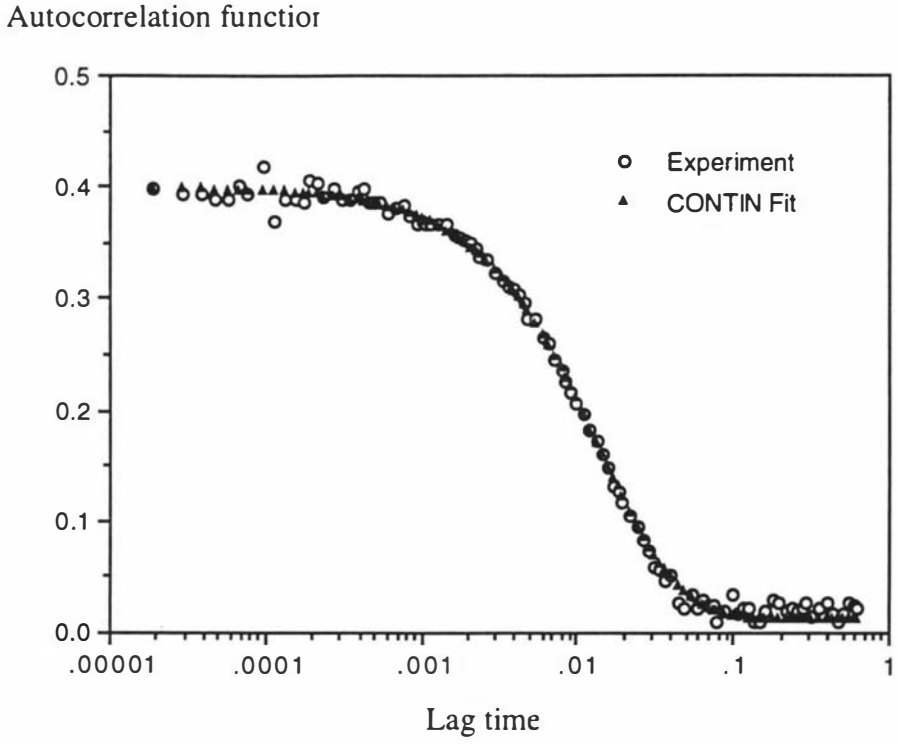


Figure 7.4 Experimental data of Figure 7.2 and the theoretical fit by CONTIN.

The refractive indices of polystyrene and thiophenol, 1.59 ±0.05 and 1.587 respectively, were so nearly equal and the volume fraction of the PMMA was so small that the fast mode was not observed.

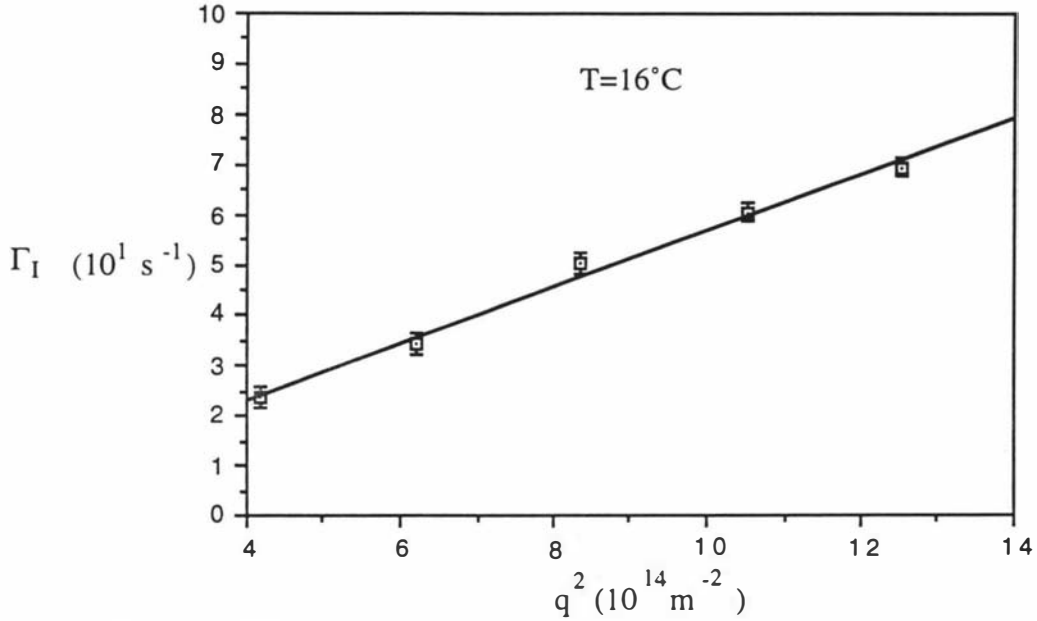


Figure 7.5 Variation of the relaxation frequencies Γ_I as a function of q^2. (PS/PMMA/thiophenol sample)

Figure 7.5 shows the variation of the decay rate Γ_I with q^2, for this solution at 16°C, a linear dependence is observed, implying that D_I is independent of q. These data are replotted in Figure 7.6 to illustrate the insensitivity of D_I to q.

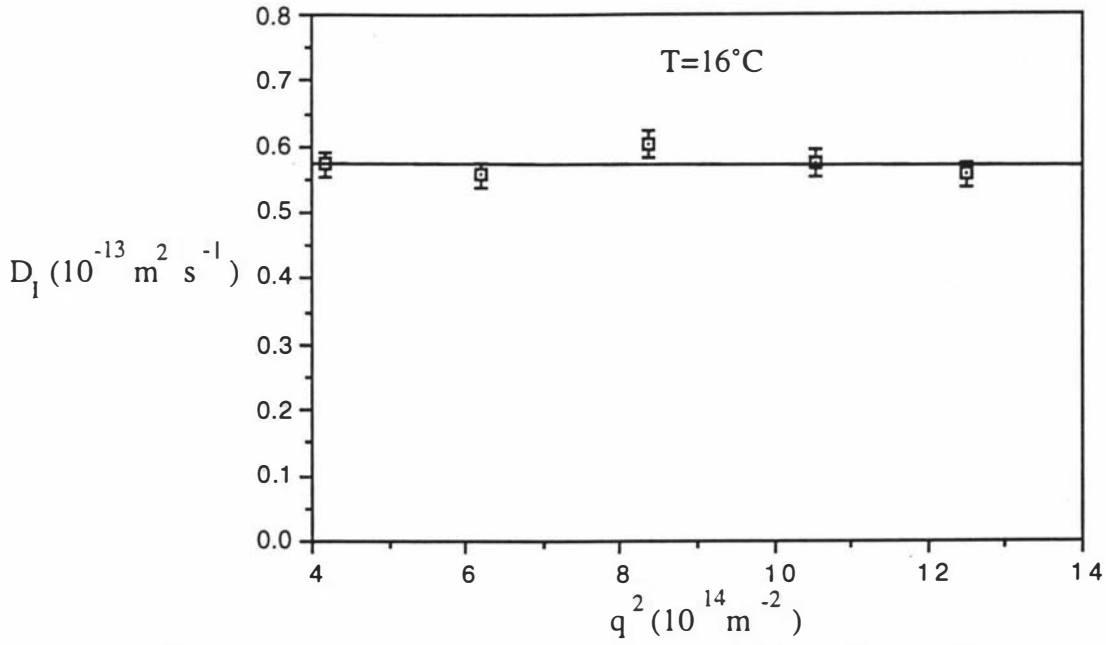


Figure 7.6 The scattering vector dependence of the effective diffusion coefficient. (PS/PMMA/thiophenol sample)

The q independence of D_I can be interpreted through eqn (7.17). $P(q)$ can be taken as unity for these polymers, so the Rouse term is q independent, the hydrodynamic term can be q independent if $q\xi$ is small (≤ 1). This solution was concluded to be a case A type solution and it was observed under the condition that $q\xi$ was always small.

Each term in eqn (7.17) is expected to vanish at the critical point (display "a critical slowing down"), since χ approaches χ_c and ξ diverges as T is reduced to T_c.

The graph of $\log D_I$ with temperature is shown in Figure 7.7.

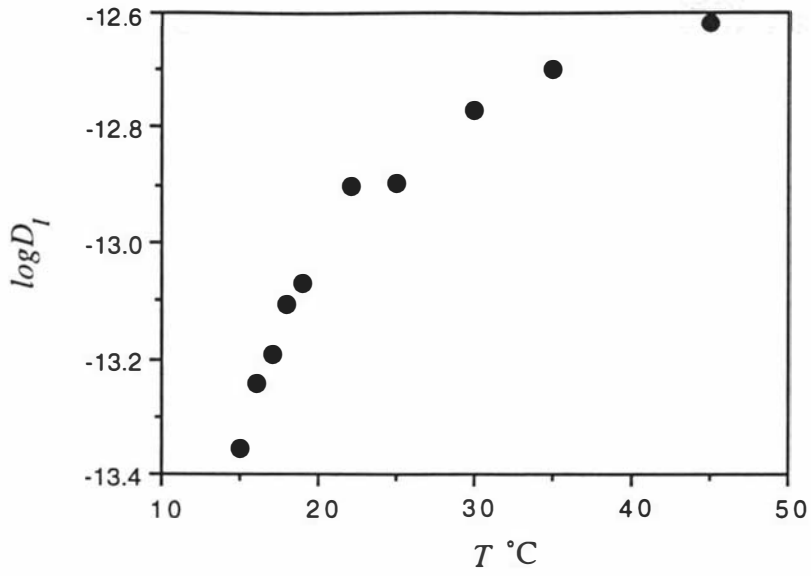


Figure 7.7 Graph of $\log D_1$ vs temperature T

D_1 decreases as the temperature is lowered. These data have been represented in Figure 7.8, where the correlation length ξ_d (defined as $kT/6\pi\eta D$) is plotted against ε ($\varepsilon = (T - T_c)/T$) to logarithmic scales.

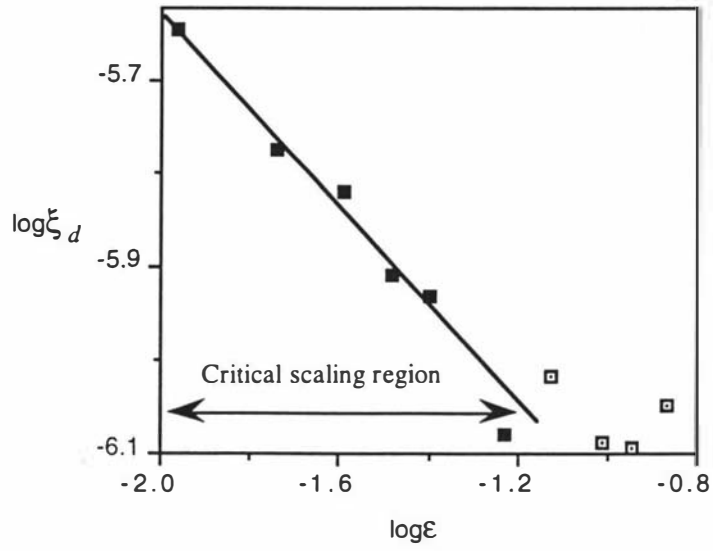


Figure 7.8 Graph of $\log\xi_d$ vs logε

Taking the critical temperature *Tc* to be 13°C, for this solution, the linear scaling region yields the correlation length critical exponent, *n*, to be 0.60 (±0.07). This is to be compared with the mean field theory, three dimensional Ising and Fisher renormalisation values of 0.50, 0.63 and 0.71 respectively. The scaling region is too narrow and the thermal stability of the experiment too poor for a more precise determination to be obtained. However, it should be noted that Seils et al (1994) were unable to quote a value for *n* from their DLS experiment on a similar system, although they did obtain the value

0.57 from static light scattering observations (no uncertainty stated). Migashitu et al (1994) have also used static light scattering to obtain the value 0.64±0.02 for n for similar ternary polymer solutions.

All the data collected from the PMMA/PS/thiophenal solution support the Benmouna et al (1993) theory of DLS from case A type ternary polymer solutions close to phase separation in which $q\xi$ was always less than unity.

7.5 Ternary Polymer Solution Experiment II

7.5.0 Ternary polymer solution

The solution studied in this experiment was a ternary polymer solution formed with 929000 molar mass polystyrene and 107000 molar mass poly (methyl-methacrylate) (PMMA) dissolved in deuterated toluene. The mass ratio of polystyrene to PMMA was 1:3 and the total polymer volume fraction, Φ, was 0.050.

The refractive index increments of polystyrene and PMMA in toluene at 25°C are 0.117 and 0.0028 respectively, so the solution closely satisfies the "isorafractive condition". However significant quantity of polystyrene was present so the amplitude of the fast decay mode is not expected to be zero. Detailed observations of only the slow decay mode are reported here.

This solution approximately satisfies the conditions for a case B type ternary polymer solution and the behaviour of the interpenetration diffusion coefficient will be discussed with reference to eqn 7.22, 7.23 (a), 7.23 (b) and 7.23 (c).

7.5.1 High temperature regime

Figure 7.9 shows the autocorrelation function measured at 25°C with the multiple tau autocorrelator, the scattering angle was 60°, and Figure 7.10 shows the distribution of relaxation rates A(Γ) calculated from the autocorrelation function by CONTIN.

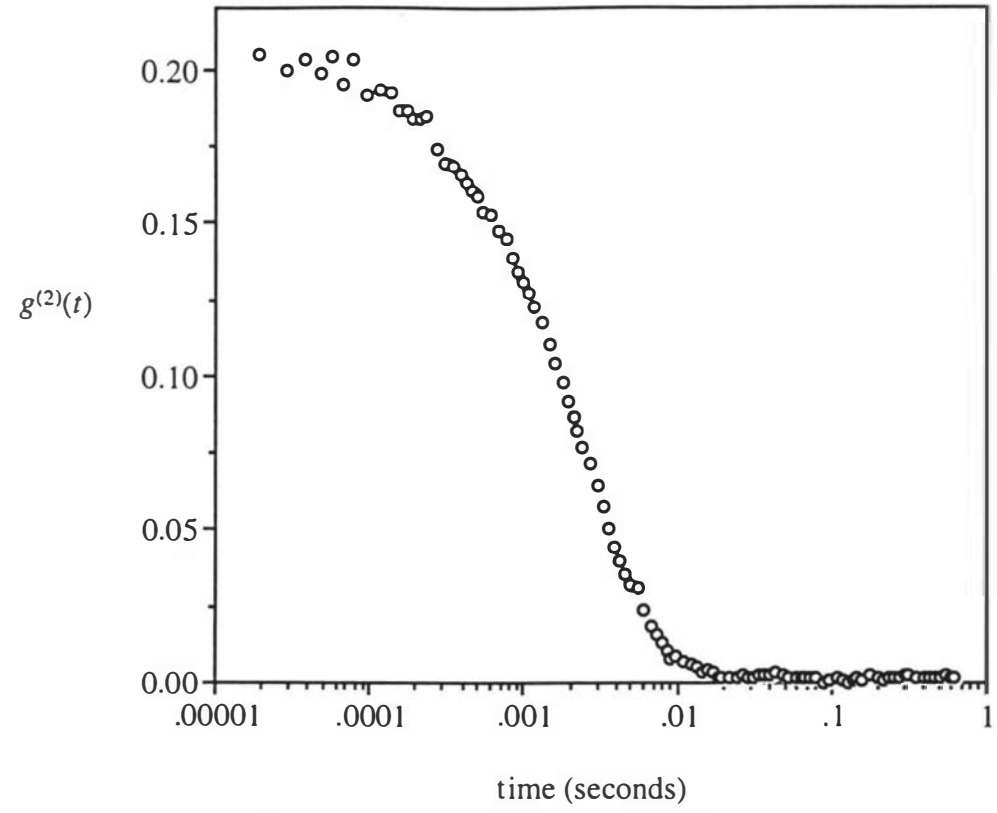


Figure 7.9 Autocorrelation function

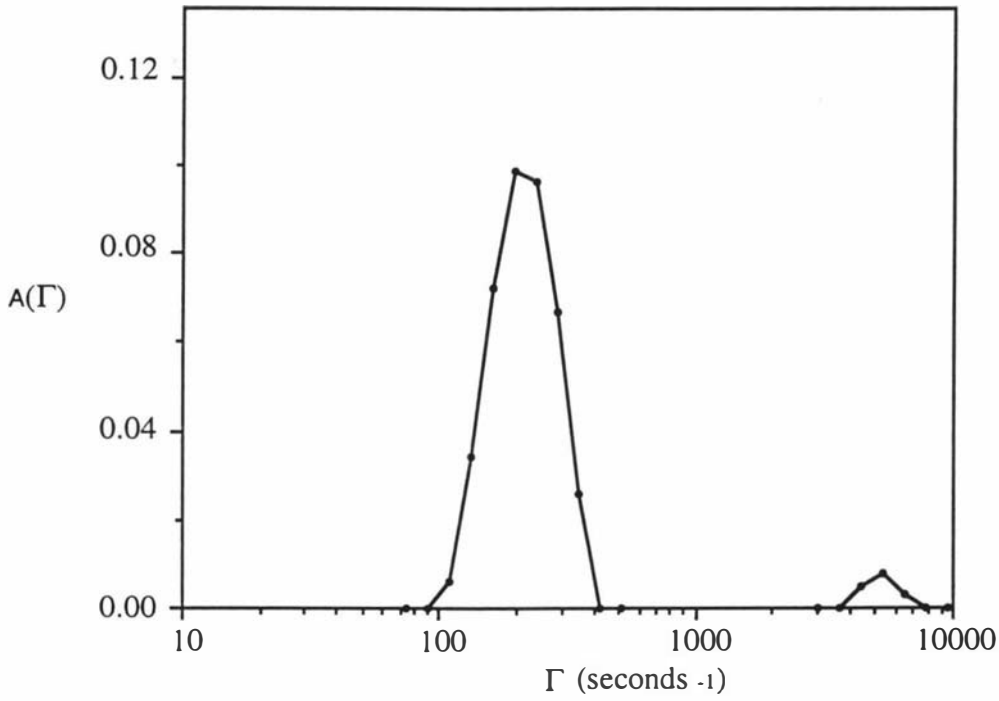


Figure 7.10 Distribution of relaxation times

Figure 7.11 shows the power spectrum obtained using the instrument in the power spectrum mode. As expected the average photo current is apparent at low frequencies and a large baseline is present due to a large shot noise background. On removing the background and ignoring the average photo current the spectrum is expected to be the

sum of two Lorentzians. The data are too scattered for the extraction of reliable half widths. The average photo current has a half width of about 1 Hz, these very slow variations could be due to laser power drift or mechanical vibrations. They are unlikely to effect the measurement of the quantities of interest.

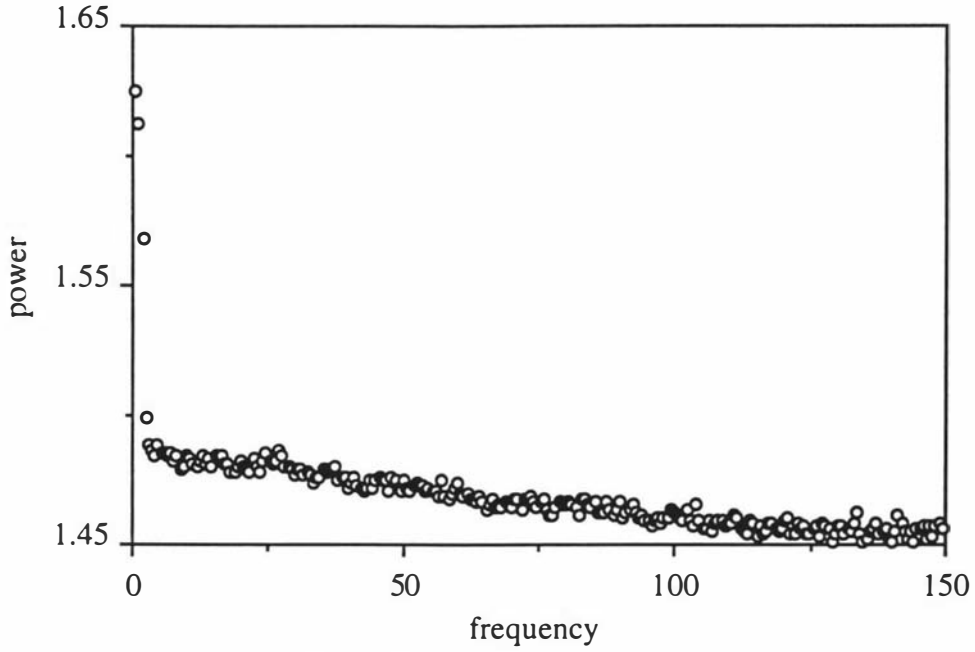


Figure 7.11 Power spectrum

The total time taken to accumulate the power spectrum was ten times that taken to accumulate the autocorrelation function, due partly to the use of batch processing in the power spectrum mode. This illustrates what is already well documented, autocorrelation analysis is a far more efficient way to treat DLS data than is power spectral analysis.

Notice two relaxation modes are observed in the autocorrelation function. The larger amplitude relaxation mode is the interpenetration mode and the smaller amplitude faster mode is the cooperative mode. The interpenetration mode is very sensitive to the chemical mismatch of the polymers, whereas the cooperative mode reflects total polymer concentration fluctuations and is not very sensitive to the chemical mismatch of the polymers. Also critical behaviour of phase separation occurs at higher temperatures than does critical behaviour of precipitation. It is for these reasons that only the interpenetration mode is studied in detail here.

The angular variation of Γ_1/q^2 has been considered by many workers, see for example (Daivis & Pinder, 1993). Figure 7.12 shows the variation of Γ_1/q^2 with q^2 for the data collected at 25°C by the multiple tau autocorrelator.

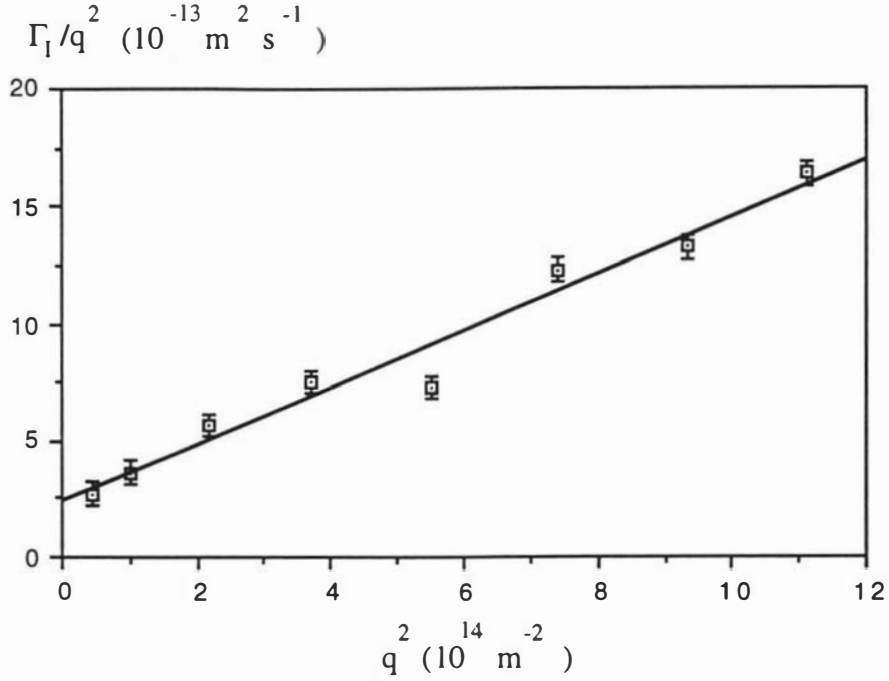


Figure 7.12 The scattering vector dependence of effective diffusion coefficient (experiment temperature is 25°C). (PS/PMMA/toluene sample)

These data can be fitted to a straight line, so the dependence on q as distinct from q^2 is negligible. Hence the data can be interpreted through eqn (7.22). The intercept is $D(1-\chi/\chi_c)+kT/8\pi\eta\xi$ and the slope is $DR_g^2/3$. Assuming both χ/χ_c and $kT/8\pi\eta\xi D$ to be small, since the solution is "far from phase separation", the values of D_I and R_g obtained are 2.4±0.5×10^{-13} (m^2 s^{-1}), and 124±10 nm respectively.

The value of D_I is less than a quarter of that expected and the value of R_g is more than a factor of two greater than expected. Either the theory is inappropriate or χ/χ_c is approximately 0.85. This would imply that 25°C is not "far from" phase separation for considerable "slowing down" has already occurred. Nevertheless assuming χ/χ_c to be 0.85, then D_I is 1.6±.2×10^{-12} m^2 s^{-1}, and R_g is 50±3 nm. This value of D_I is less than but compatible with the measures of D_I found by Daivis et al (1992) on different but similar solutions formed with toluene. [They investigated solutions formed with PVME and a trace amount of polystyrene, they found D_I to be approximtely 1.8×10^{-12} m^2 s^{-1}, at the same total polymer fraction.] The value of R_g is compatible with that quoted by Adam and Delsanti (1977) for polystyrene dissolved in benzene (54nm).

Daivis and Pinder (1993) have previously observed anomalously small values of D_I in their study of 929000 polystyrene and 110000 PVME dissolved in carbon tetrachloride. They found D_I to be approximately a factor of 2 smaller than that found for the same polymers dissolved in toluene at similar concentrations and compositions. They

suggested the effect was due to the carbon tetrachloride solution being close to phase separation because carbon tetratchloride has significantly different affinities for the two polymers. Even the simple Borsali Benmouna theory is inapplicable in this case for a basic assumption of the theory is that the solvent be equally good for the two polymers.

There are two intriguing features of the Daivis et al (1993) data. One is the anomalously small values of D_I reported for the carbon tetrachloride solutions, which has just been discussed. The other is the very small values of R_g found in that work, they found R_g to approximately 2.5 times smaller than expected. They used a 48 channel linear correlator for all their DLS data collection. The performance of a 48 channel linear correlator is compared with that of the multiple tau correlator in what follows.

In Figure 7.13 the variation of Γ_I/q^2 with q^2 for data collected at 25°C by the multiple tau correlator is compared with that collected by a 48 channel linear autocorrelator. These two data sets were collected by processing the same photomultiplier pulse train in each autocorrelator simultaneously.

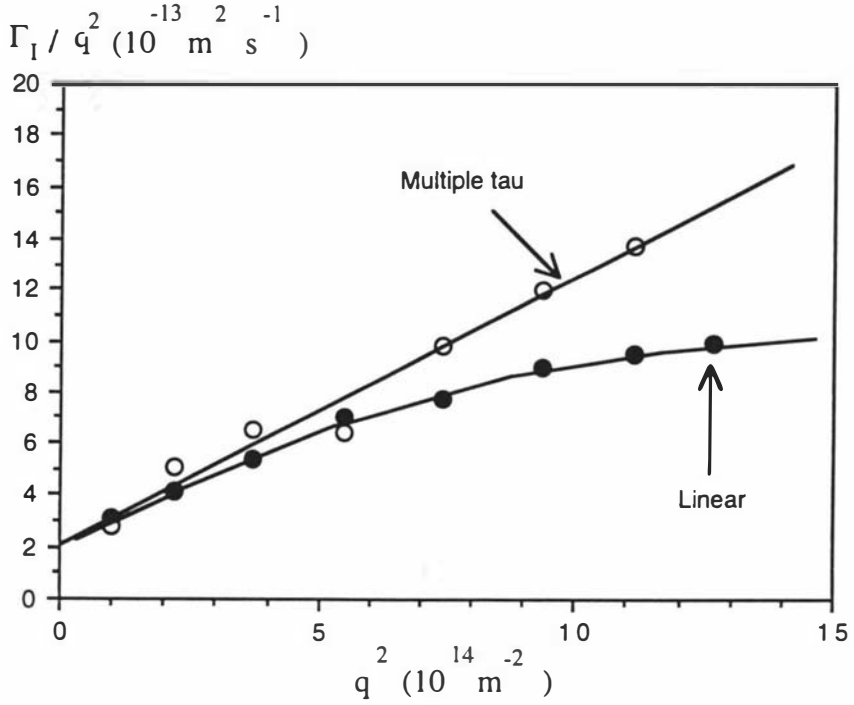


Figure 7.13 Γ_I/q^2 against q^2, for data collected by multiple tau and linear correlators

The data obtained from the linear correlator are nonlinear. These data were analysed using the method of cumulants, care had to be taken to ensure that only the interpenetration mode was studied and not the small amplitude faster cooperative mode. Care also had to be taken at forward scattering angles (small q) to ensure that data from supposed "aggregates" were not included. It seems that this "prior processing" of the linear

correlator data may have caused the data to be wrongly interpreted. Indeed an earlier study of the same solution at the same temperature under taken before the multiple tau correlator was available yielded the linear graph shown in Figure 7.14.

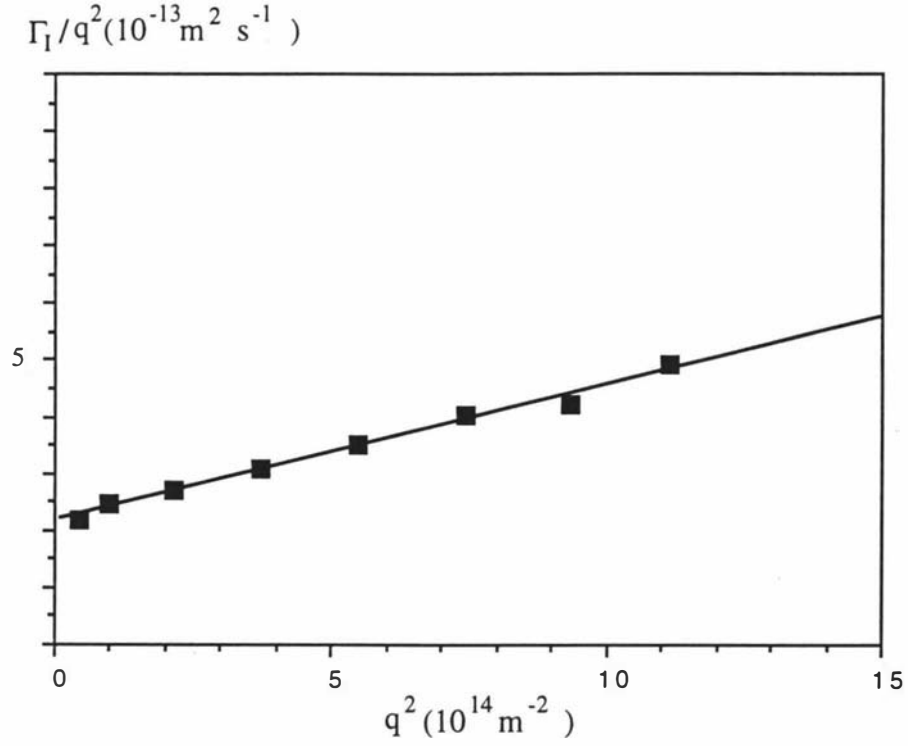


Figure 7.14 Γ_1/q^2 against q^2, data collected by linear correlator (old data)

It would appear that data collected at small scattering angles were processed to discriminate against low values of Γ_1/q^2 which were thought to be due to polymer aggregation effects. Notice these "old" linear correlator data are a linear extrapolation of the "new" high q data obtained with the linear correlator. These "old" linear correlator data provide values of R_g that are too small by a factor of two, echoing the very small values of R_g found by Daivis and Pinder (1993).

The "new" data collected with the linear correlator do agree with those collected with the maltiple tau correlator at small q values, see Figure 7.13. So the values of D_I and R_g obtained from a very careful analysis of the small q data collected with the linear correlator do agree with those values obtained using the multiple tau correlator.

The surprisingly large artefacts that can be introduced by the use of a linear correlator are quite startling. It is possible that these data processing effects are the cause of the anomolously small values of R_g found by Daivis et al (1993). Even supposedly single exponential DLS data should be analysed over an extended timescale of at least five times

the relaxation time to avoid the unwitting introduction of artefacts. A multiple tau correlator is ideal for this task.

7.5.2 Low temperature regime

Only the interpenetration mode will be discussed in this section, and only data collected with the multiple tau correlator will be presented.

Figure 7.15 is a graph of Γ_I/q^2 against q^2 for data collected from the solution at 16°C.

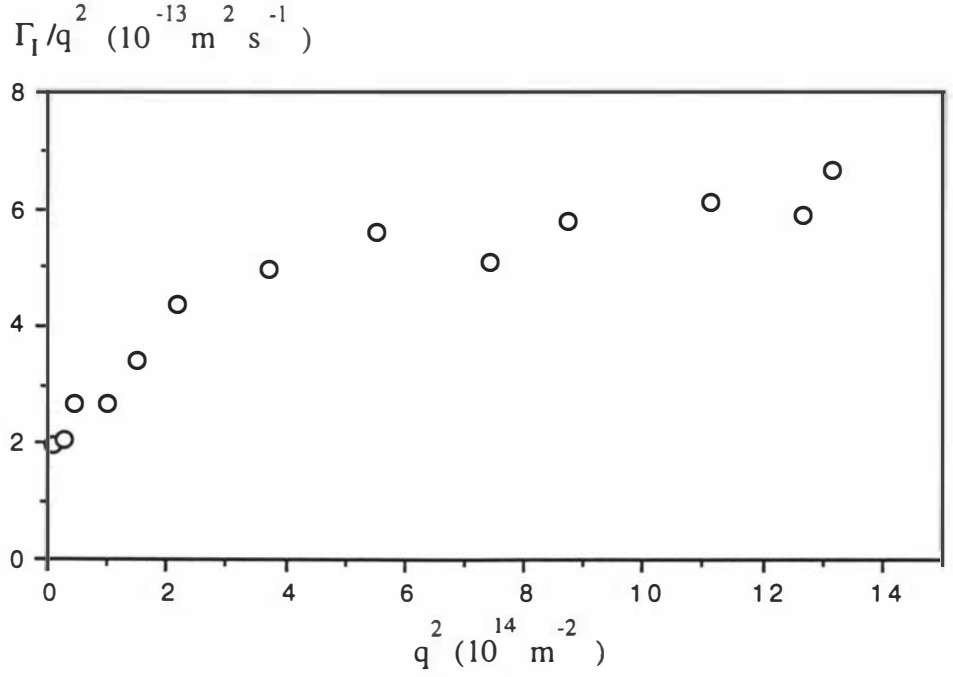


Figure 7.15 The interdiffusion coefficient as a function of q^2 at 16°C. (PS/PMMA/toluene sample)

The graph is nonlinear, moreover this nonlinearity is clearly due to Γ_I/q^2 varying with q to a lower power than q^2. These data can be interpreted through eqn 7.23 (c), where the nonlinearity of the Γ_I/q^2 against q^2 graph is ascribed to the effect of the hydrodynamic term. At 16°C the solution is close to phase separation so the relative contribution of the Rouse term is reduced, because χ/χ_c is close to unity, and the hydrodynamic term becomes significant at large q values.

The graph can be interpreted as being Rouse term dominated at small q values and hydrodynamic term dominated at large q values. The ordinate intercept is predicted to be $D_I(1-\chi/\chi_c)$ and the initial slope is predicted to be $D_I(1-\chi/\chi_c)2\xi^2/3$. Again D_I cannot be found since χ/χ_c is unknown.

Since χ increases to χ_c at phase separation, the Benmouna et al theory predicts the intercept of the Γ_1/q^2 against q^2 graph to be strongly temperature dependent, indeed the intercept is predicted to display a "critical slowing down". However, the initial slope of the graph is predicted to be less sensitive to temperature.

Figure 7.16 shows data collected from the solution at three temperatures. The most interesting feature of this figure is the insensitivity of the intercept to temperature.

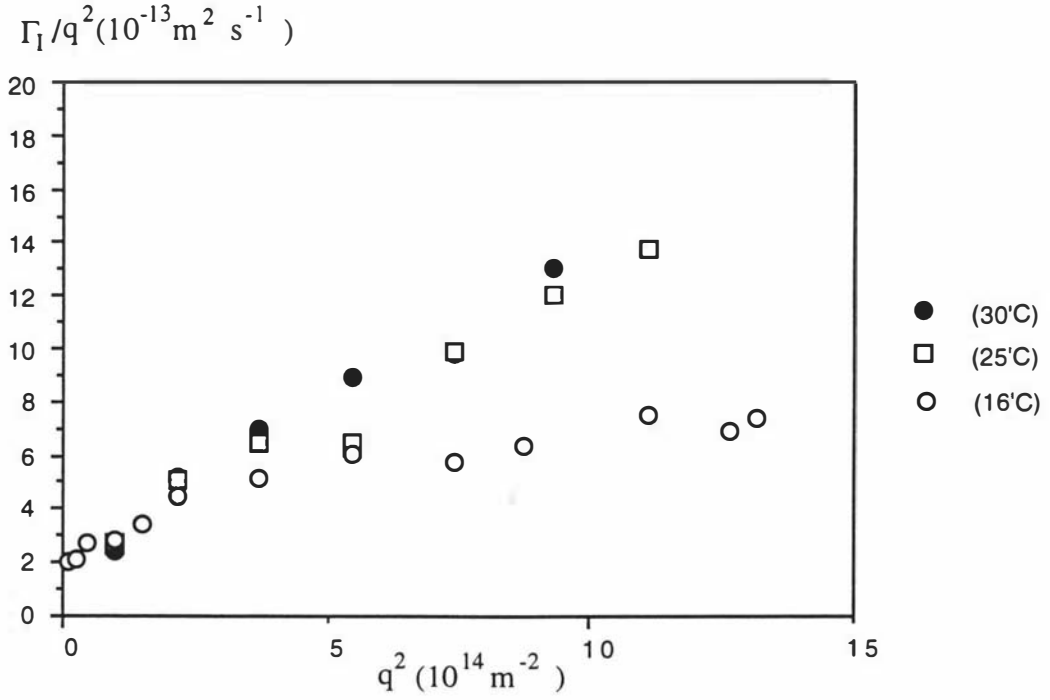


Figure 7.16 Data collected from the solution at three temperatures

This solution does not display the predicted "critical slowing down". Moreover the initial slope decreases with decreasing temperature implying that ξ^2 decreases with temperature which certainly is not expected. So the details of the Benmouna et al theory (1993) are not supported by these data. Although the general form of the variation of Γ_1/q^2 with q^2 can be interpreted through the Benmouna theory the details of the variation with temperature are not in agreement with theory.

Seils et al (1994) have studied the critical behaviour of a similar ternary polymer solution. These authors claimed to be able to resolve two separate slow modes at temperatures close to phase separation. They identified one mode as a Rouse like mode which they claim did to show a critical slowing down, although the authors were unable to present convincing data to support this claim. They also claimed that the amplitude of this mode decreased as the temperature was lowered. They identified the second mode with the hydrodynamic term in eqn 7.23c, the amplitude of this mode was claimed to increase as the solution temperature was reduced, its amplitude was negligible at high temperature.

The results presented by Seils et al are in broad agreement with those found here. However, our autocorrelation functions could not be resolved into two slow modes, indeed the Benmouna theory does not predict the existing of two distinct slow modes, rather it predicts that the single interpenetration decay rate varies in a complicated manner with temperature and scattering vector.

7.5.3 Conclusion

(i) The first solution studied was 323000 PMMA and 233000 polystyrene dissolved in thiophenol, the molar masses of both polymers were sufficiently small for the particle form factor, $P(q)$ to be taken as unity. The interpenetration diffusion coefficient was observed to vary in a manner predicted by the Benmouna theory. The interpenetration diffusion coefficient was observed to be independent of q, indicating that $q\xi \leq 1$ at all times.

A critical slowing down was observed and a measure of the critical exponent for correlation length was obtained. This value (0.60±0.07) is close to the three dimensional Ising value and is in agreement with measures published by other authors using static light scattering methods.

It should be noted that this is the first occasion that a value for this critical exponent has been obtained using DLS, although other authors have tried to obtain a value.

(ii) The second solution studied was 107000 PMMA and 929000 polystyrene dissolved in deuterated toluene. There are two major differences between this solution and the previous one. First the molar mass of the polystyrene is larger, so the particle form factor was taken as $1+q^2R_g^2/3$. Secondly the relative abundance of the visible polymer was significant. The variation of the Γ_1/q^2 with q^2 was observed to be in general agreement with the predictions of the Benmouna theory, but the details of the behaviour did not confirm the theory. In particular a critical slowing down was not observed. It would appear that the q^2 dependence introduced by using a polymer with a larger molar mass tended to mask the critical behaviour that was the prime object of the study. At 25°C the intercept value of Γ_1/q^2 is only a quarter of that expected which may indicate that at temperatures as high as 25°C considerable critical slowing down has already occurred and χ/χ_c is 0.85.

The fact that no further critical slowing down is observed as the temperature is reduced to 16°C is a matter of some concern. It is known that toluene has different affinites for the polymer PMMA and polystyrene so the solvent barely meets the assumption of the Benmouna theory that it is of equal and good quality for both polymers. Perhaps this

effect plays a role as phase separation is approached and the expression for the Rouse term is faulty.

(iii) A 48 channel linear correlator was compared with the multiple tau correlator by processing the same photomultiplier pulse train in both correlators simmultaneously. It was shown that the limited timescale of the linear correlator may cause the autocorrelation function to be misdiagnosed and so cause the introduction of some unexpected artefacts.

8 POSTSCRIPT

8.1 Conclusions

Current investigation of ternary polymer solutions demand high resolution decay rates distribution functions. The time scale of the autocorrelation function must be optimised if such functions are to be produced. These considerations have motivated the design and construction of the novel digital autocorrelator described in this thesis. The following results were achieved:

(1) A real time pseudo logarithmic time scale autocorrelator using the multiple tau technique has been designed and constructed, the multiple tau correlator provides a delay time range covering both fast and slow processes to achieve better Inverse Laplace Transform results.

The instrument is based on a parallel block processing approach which uses a loosely coupled MIMD (multiple-instruction-multiple-data) architecture. It has been shown that it is possible to design a real time multiple tau autocorrelator by employing this parallel DSP system, and this is the first multiple tau correlator constructed by a multi-DSP system. Four Motorola DSP56001 digital signal processors are used in the architecture, and continuously supplied input data are processed concurrently by these four DSPs. The architecture was designed to minimise the interprocessor communication. The autocorrelator has 128 channels each with a 24-bit count capacity. Sample times range from 3.2 μs to 2 seconds in real time.

The use of a multi-DSP system rather than a "hardware" logic system makes it possible for the instrument to be easily programmed to function as a spectrum analyser. Indeed the processing code could also be easily changed for different applications, various digital signal processing algorithms can be implemented on the same multiprocessor system.

It was shown that in the more complicated ternary polymer solution system study, the limited timescale of the linear correlator may cause the autocorrelation function to be misdiagnosed and so cause the introduction of some artefacts. The studies reported here confirm that the initial part of the autocorrelation function must be sampled at a sufficiently short sample times to provide the required high frequency response, and that the time scale must also be able to extend far enough for the autocorrelation function to decay well into the noise level.

(2) The multiple tau autocorrelator was used in two different sets of ternary polymer solution studies. In the first the molar masses of both polymers were sufficiently small for the particle form factor, $P(q)$ to be taken as unity. The interpenetration diffusion

coefficient was observed to be independent of q. The interpenetration diffusion coefficient was observed to vary in a manner predicted by the Benmouna theory. A critical slowing down was observed and a measure of the critical exponent of the correlation length was obtained. This is the first occasion that a value for this critical exponent has been obtained using DLS.

In the second solution studied the molar mass of the polystyrene was larger, so the particle form factor was taken as $1+q^2R_g^2/3$. The variation of the Γ_1/q^2 with q^2 was observed to be in general agreement with the predictions of the Benmouna theory, but the details of the behaviour did not confirm the theory. In particular a critical slowing down was not observed, which may indicate that at temperatures as high as 25°C considerable critical slowing down has already occurred and χ has assumed the value $0.85\chi_c$.

8.2 Suggestions for further work

(1) Ternary Polymer Solution Studies

The delay-time range of this multiple tau autocorrelator has been designed to encompass both the fast and the slow decay processes that occur in ternary polymer solutions. This multiple tau correlator should be used in further studies of ternary polymer solution in the critical region to fully explore the implications and predictions of the Benmouna et al theory (Benmouna, 1993). In particular solutions formed with equal quantities of 400000 Poly(methyl methacrylate) and 390000 polystyrene dissolved in bromo benzene should be studied. Such solutions meet the restriction requirements of the theory without causing the particle form factor, $P(q)$ to differ significant from unity.

(2) Multiple Tau Autocorrelator Developments

(i) The multiple tau autocorrelator was purposely designed to be flexible so to take advantage of any future advances in normalisation procedures.

(ii) The delay time range of the multiple tau autocorrelator could be extended both to shorter and to longer delay times

(a) Shorter delay times

Although shorter delay times could be readily attained by including an additional DSP unit, this was not deemed necessary or desirable. There are two reasons for this.

First the laser light scattering laboratory at Massey University already has a very fast 48 channel linear correlator which can operate at the shortest useable delay time of 50 ns. At this sample time the number of photo detections per sample time is either zero or unity so

the linear correlator calculates the autocorrelation function without error. The total lag through the 48 linear channels is 2.4 microseconds. The multiple tau autocorrelator is compatible with the linear correlator so the two can be ran together to span the lag time range 50ns to 1 second in one experiment. The linear correlator simply acts as an additional block of 48 channels with sample time 50ns.

Secondly at sample times less than a few micro seconds, there are so few photo detections per sample time in a typical DLS experiment that the signal to photon noise ratio is very small.

(b) Longer delay times

The multiple tau autocorrelator could be made to operate at maximum delay times of a few minutes by the use of an additional DSP unit (or even a 68000 family processor). Also the Macintosh IIvx computer could be used to process even longer delay times of several tens of minutes. But such a correlator would not make optimum use of the data, since measurements which include delay times of 1 second require accumulation times of the order of several hours. The integrity of the data would be challenged by the long term thermal stability of the laboratory and apparatus and also by the long term stability of the laser.

REFERENCES

1. T. Agnello, *Computer Design*, 25-27 (1992).

2. ANALOGIC, ANALOGIC, Designing with Multiple DSP Processors (1993).

3. A. J. Anderson, *Journal of Microcomputer Applications*, 327-346 (1992).

4. W. Andrews, *Computer Design*, 82-96 (1989).

5. W. Andrews, *Computer Design*, 69-79 (1992b).

6. W. Andrews, *Computer Design*, 78-98 (1992a).

7. R. Asch, N. F. Jr., *The Review of Scientific Instruments* **44**, 506-508 (1973).

8. M. R. Aven, C. Cohen, *Macromolecules* **23**, 476-486 (1990).

9. H. Baba, S. Hoshina, K. Sakurai, N. Takeuchi, *Rev. Sci. Instrum.* **56**, 1926-1929 (1985).

10. M. Benmouna, H. Benoit, Z. Akcasu, *Macromolecules* **20**, 1107-1112 (1987a).

11. M. Benmouna, H. Benoit, R. Borsali, M. Duval, *Macromolecules* **20**, 2620-2624 (1987b).

12. M. Benmouna, J. Seils, *Macromolecules* **26**, 668-678 (1993).

13. B. J. Berne, R. Pecora, *Dynamic Light Scattering* (John Wiley and Sons, New York, 1976).

14. L. Bhuyan, Q. Yang, D. Agrawal, *IEEE Computer* , 25-37 (1989).

15. J. Bond, *Computer Design* , 51-74 (1987).

16. R. Borsali, M. Duval, M. Benmouna, *Macromolecules* **22**, 816-821 (1989).

17. W. Brown, J. Fundin, *Macromolecules* **24**, 5171-5178 (1991).

18. W. Brown, P. Stepanck, *Macromolecules* **25**, 4359 (1992).

19. W. Brown, *Dynamic Light Scattering, The Method and Some Applications* (Clarendon, Oxford, 1993), vol. 1.

20. F. Bruge, P. San-Biagio, S. Fornili, *Rev. Sci. Instrum.* **60**, 3425-3429 (1989).

21. F. Bruge, P. S. Biagio, S. Fornili, *Rev. Sci. Instrum.* **60**, 222-225 (1989).

22. C.J.Oliver, in *Scattering Techniques Applied to Supramolecular and Nonequilibrium Systems.* S.H.Chen, B.Chu, R.Nossal, Eds. (1981).

23. V. Chaudhary, J. Aggarwal, *IEEE Transactions on Parallel and Distributed Systems* **4**, 328-346 (1993).

24. B. Chu, *Laser Light Scattering* (Academic Press, INC., New York, 1974).

25. C. Correia, F. P. Santos, *Rev. Sci. Instrum.* **60**, 2937-2939 (1989).

26. Z. Cummins, E. Pike, *Photon Correlation and Light Beating Spectroscopy* (Pienum Press, New York, 1974).

27. Z. Cummins, E. Pike, *Photon Correlation Spectroscopy and Velocimetry* (Plenum Press, New York, 1977).

28. B. Curtis, V. Madisetti, *IEEE Transactions on Signal Processing* **42**, 649-662 (1994).

29. P. J. Daivis, Ph. D Thesis, Massey University, New Zealand (1989).

30. P. J. Daivis, D. N. Pinder, *Macromolecules* **26**, 3381-3390 (1993).

31. A. Davis, J. Burke, *Byte* , 269-275 (1992).

32. V. Degiorgio, M. Corti, M. Giglio, *Light Scattering in Liquids and Macromolecular Solutions* (Plenum Press, New York, 1980).

33. R. Dorshow, *Rev. Sci. Instrum.* **61**, 186-188 (1990).

34. M. El-Sharkawy, *Real Time Digital Signal Processing Applications With Motorola's DSP56000 Family* (Prentics Hall, 1990).

35. J. Ferron, *Rev. Sci. Instrum.* **63**, 5464-5468 (1992).

36. A. Florek, A. Hanczak, K. Kozlowski, W. Wroblewski, *Journal of Microcomputer Applications* **14**, 27-37 (1991).

37. S. Friberg, J. Andersen, L. Mandel, *Rev. Sci. Instrum.* **53**, 205-209 (1982).

38. H. F. Fuchs, C. Jorde, O. Glatter, *Rev. Sci. Instrum.* **60**, 854-857 (1989).

39. A. Fukuda, K. Murakami, *IEEE Micro* , 16-61 (1991).

40. L. Giebel, R. Borsali, G. Meier, *Macromolecules* **23**, 4054-4060 (1990).

41. L. Giebel, M. Benmouna, R. Borsali, E. Fischer, *Macromolecules* **26**, 2433-2438 (1993).

42. P. Hoang, J. Rabaey, *IEEE TRANSACTIONS ON SIGNAL PROCESSING* **41**, 2225-2235 (1993).

43. M. Hobel, M. Wuthrich, J. Ricka, T. Binkert, *Rev. Sci. Instrum.* **65**, 2123-2129 (1994).

44. N. Instruments, *LabVIEW2 User Manual* (National Instruments Corporation, 1992b).

45. N. Instruments, *LabVIEW2 Data Acquisition VI Library Reference Manual* (National Instruments, 1992a).

46. N. Instruments, *NB-DIO-24 User Manual* (National Instruments, 1993).

47. E. Jakeman, , "Photon Correlation", in [26]

48. H. Janssen, *Nuclear Instrument and Methods in Physics Research* **A299**, 292-297 (1990).

49. R. M. Johnsen, W. Brown, *Laser Light Scattering in Biochemistry* **Chapter 6**, 77-91 (1988).

50. C. Konac, J. Podessva, *Macromolecules* **24**, 6502 (1991).

51. P. Lago, L. Rovati, D. Cuffaro, M. Corti, *Rev. Sci. Instrum.* **65**, 263-264 (1994).

52. J. Lane, G. Hillman, *Implementing IIR/FIR Filters with Motorola's DSP56000/DSP56001* (Motorola INC., 1990).

53. D. Lenoski, J. Laudon, K. Gharachorloo, *IEEE Computer* , 63-79 (1992).

54. A. Liao, G. Gao, V. Agarwal, *Journal of Microprocessor Applications* **17**, 171-196 (1994).

55. G. Lorusso, V. Capozzi, A. Minafra, *Rev. Sci. Instrum.* **63**, 2152-2156 (1991).

56. Y. Luo, D.N.Pinder, R.C.O'Driscoll, *Australian Instrumention and measurement conference* **AIM-92**, 153-158 (1992).

57. B. Malloy, E. Lloyd, M. Soffa, *IEEE Transactions on Parallel and Distributed Systems* **5**, 498508 (1994).

58. M. Marsan, G. Balbo, G. Conte, *IEEE Transactions on Computers* **c31**, 1179-1188 (1982).

59. T. Mimar, *Programming and Designing with the 68000 Family* (Prentice-Hall, INC, 1991).

60. J. Mosely, *Electronics World & Wireless World* , 412-417 (1990).

61. Motorola, *DSP56001 Digital Signal Processor User's Manual, Rev 2* (Motorola, 1990b).

62. Motorola, *DSPRAM 8K x 24 bit fast static RAM, Motorola Semiconductor Technical Data* (Motorola, 1990a).

63. Motorola, *Advance Information, 24 Bit General Purpose Digital Signal Processor, DSP56001, Motorola Semiconductor Technical Data* (Motorola, 1991b).

64. Motorola, *DSP Development Software-DSP56000CLASB Manual* (Motorola, 1991a).

65. A. Noullez, J. P. Boon, *Rev. Sci. Instrum.* **57**, 2523-2528 (1986).

66. R. C. O'Driscoll, Ph. D Thesis, Massey University, New Zealand (1982).

67. C. J. Oliver, , "Correlation Techniques", in [26]

68. C. Oliver, *in Photon Correlation and Light Beating Spectroscopy*. R. Pike, H. Cummins, Eds., (Plenum, New York, 1974).

69. C. J. Oliver, *J. Phys. A: Math. Gen.* , 591-617 (1979).

70. R. Pecora, *Dynamic Light Scattering* (Pienum Press, New York, 1985).

71. Philips, *Fast TTL Logic Series Data Handbook* (Philips Semiconductors, 1992).

72. D. N. Pinder, D. Clark, P. Daivis, *Macromolecular Reports* **A31**, 1119-1126 (1994).

73. D. N. Pinder, (1995a).

74. D. N. Pinder, (1995b).

75. W. Press, B. Flannery, S. Teukolsky, V. Wetterling, *Numerical Recipes* (Cambridge University Press, 1986).

76. S. W. Provencher, *Computer Physics Communications* **27**, 213-227 (1982b).

77. S. W. Provencher, *Computer Physics Communications* **27**, 229-242 (1982a).

78. S. Provencher, Max Planck Institut fur Biophysikalische Chemie, CONTIN (Version 2) User Manual (1984).

79. P. Pusey, L. Vaughan, Specialist Periodical Reports of The Chemical Society, London, Light Scattering and Intensity Fluctuation Spectroscopy (1975).

80. K. Rektorys, *Applicable Mathematics* (Iliffe Books, Massachusetts, 1976).

81. K. Schatzel, *Appl. Phys.* **22**, 251-256 (1980).

82. K. Schatzel, *Inst. Phys. Conf. Ser. 77* , 175-184 (1985).

83. K. Schatzel, M. Drewel, S. Stimac, *Journal of Modern Optics* **35**, 711-718 (1988).

84. K. Schatzel, *Quantum Opt.* **2**, 287-305 (1990).

85. J. Seils, M. Benmouna, *Macromolecules* **27**, 5043-5051 (1994).

86. J. Singh, J. Hennessy, A. Gupta, *IEEE Computer* , 42-50 (1993).

87. M. Smith, *IEEE Micro* , 10-23 (1992).

88. G. R. Sohie, *Implementation of Fast Fourier Transforms on Motorola's DSP56000/DSP56001 and DSP96002 Digital Signal Processors* (Motorola INC., 1989).

89. V. Subrahmanyam, B. Devraj, S. Chopra, *J. Phys. E: Sci. Instrum.* **20**, 1987 (1987).

90. Z. Sun, C. H. Wang, *Macromolecules* **27**, 5667-5673 (1994).

91. N. Suri, M. Hugue, C. Walter, *Proceedings of The IEEE* **82**, 41-53 (1994).

92. J. C. Thomas, Y. T. Lum, D. Kennett, *Rev. Sci. Instrum.* **54**, 1346-1355 (1983).

93. C. Wang, G. Fytas, D. Lilge, T. Dorfmuller, *Macromolecules* **14**, 1363 (1981).

94. D. Warner, F. Wells, *Rev. Sci. Instrum.* **60**, 2804-2806 (1989).

95. R. Wilson, *Computer Design* , 72-83 (1989).

APPENDIX I CORRELATOR SPECIFICATIONS

Channels: Number of channels and sampling time controlled by software. For 128 real time channels (Multiple Tau Sampling Scheme), the fastest sampling time is 3.2μs.

Modes: Real time multiple tau autocorrelator, real time spectrum analyser, or other real time digital signal processing.

Processors: Four 24-bit Motorola DSP56001 operating at 33MHz.

Input pulse specifications:

TTL compatible inputs on 50 Ω.
Pulse width: >20 ns
Pulse freq.: <200 MHz
Input current (0<Vin<5V): Min. -10mA, Max10mA

Dimension: 480mm x 180 mm x 260 mm, standard eurocard structure.

Computer and software requirements:

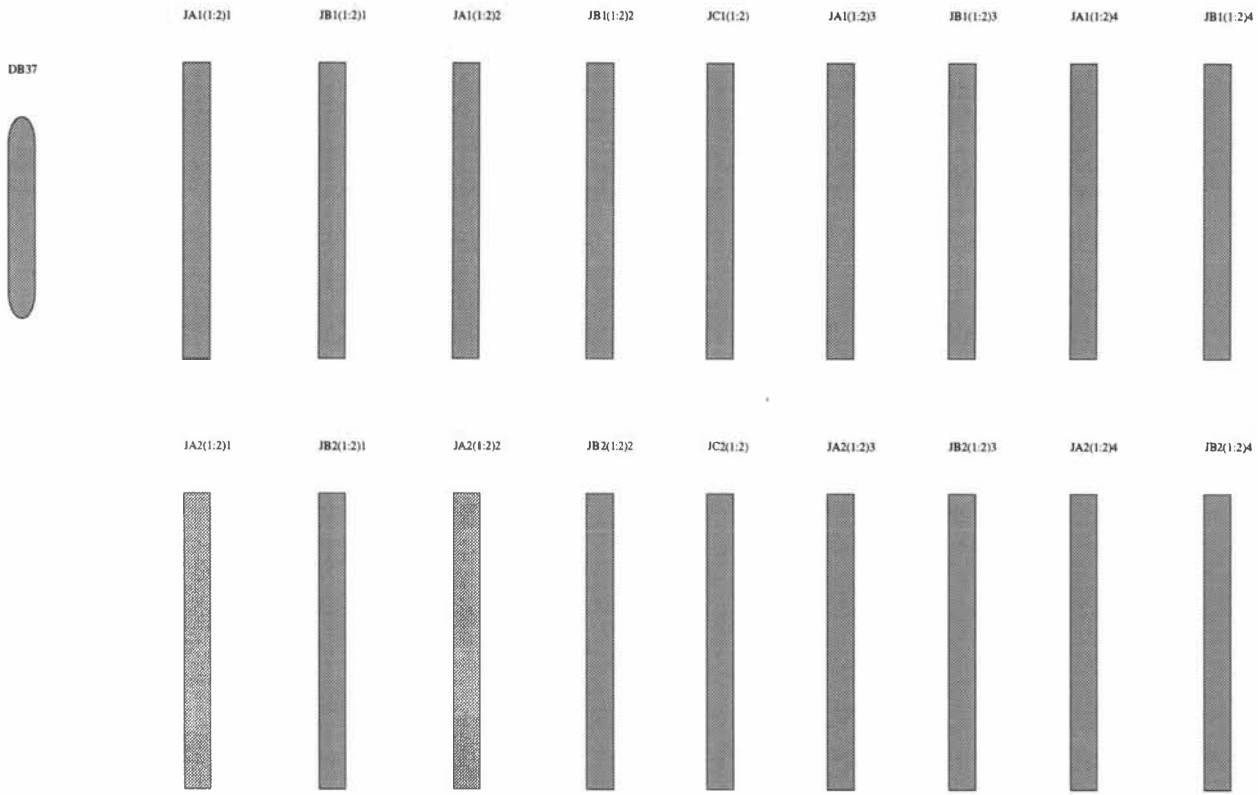
Macintosh II, PC/XT/AT or compatible, IBM PS/2. LabVIEW (NATIONAL INSTRUMENTS) for Macintosh (NuBus system), LabVIEW for Windows (PC/XT/AT and PS/2 system). LabWindows for DOS (PC/XT/AT and PS/2 system).

Interface requirements:

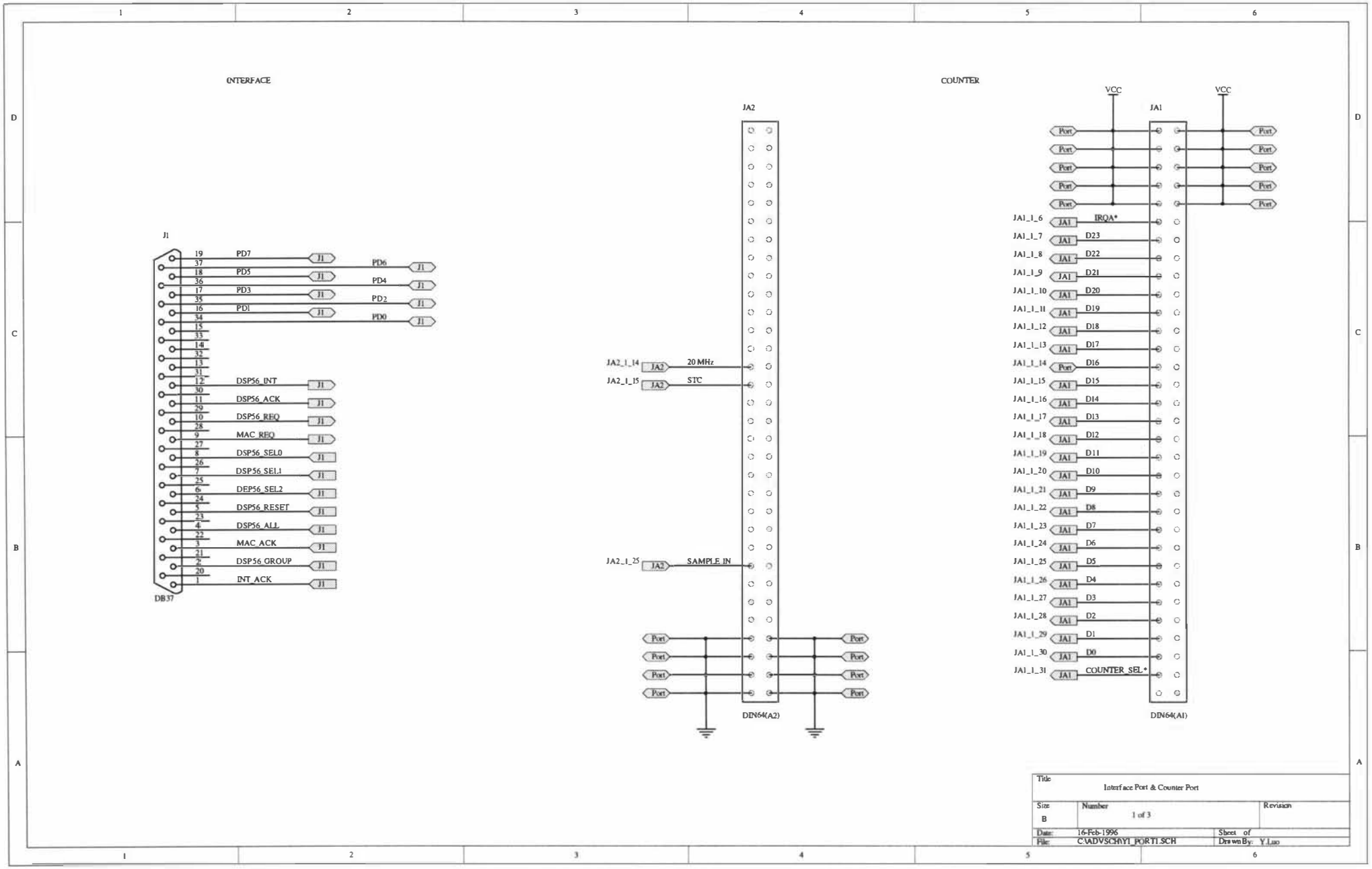
NATIONAL INSTRUMENTS DIO-24 Interface board. The correlator I/O connector is a 50-pin polarised male ribbon cable header with strain relief.

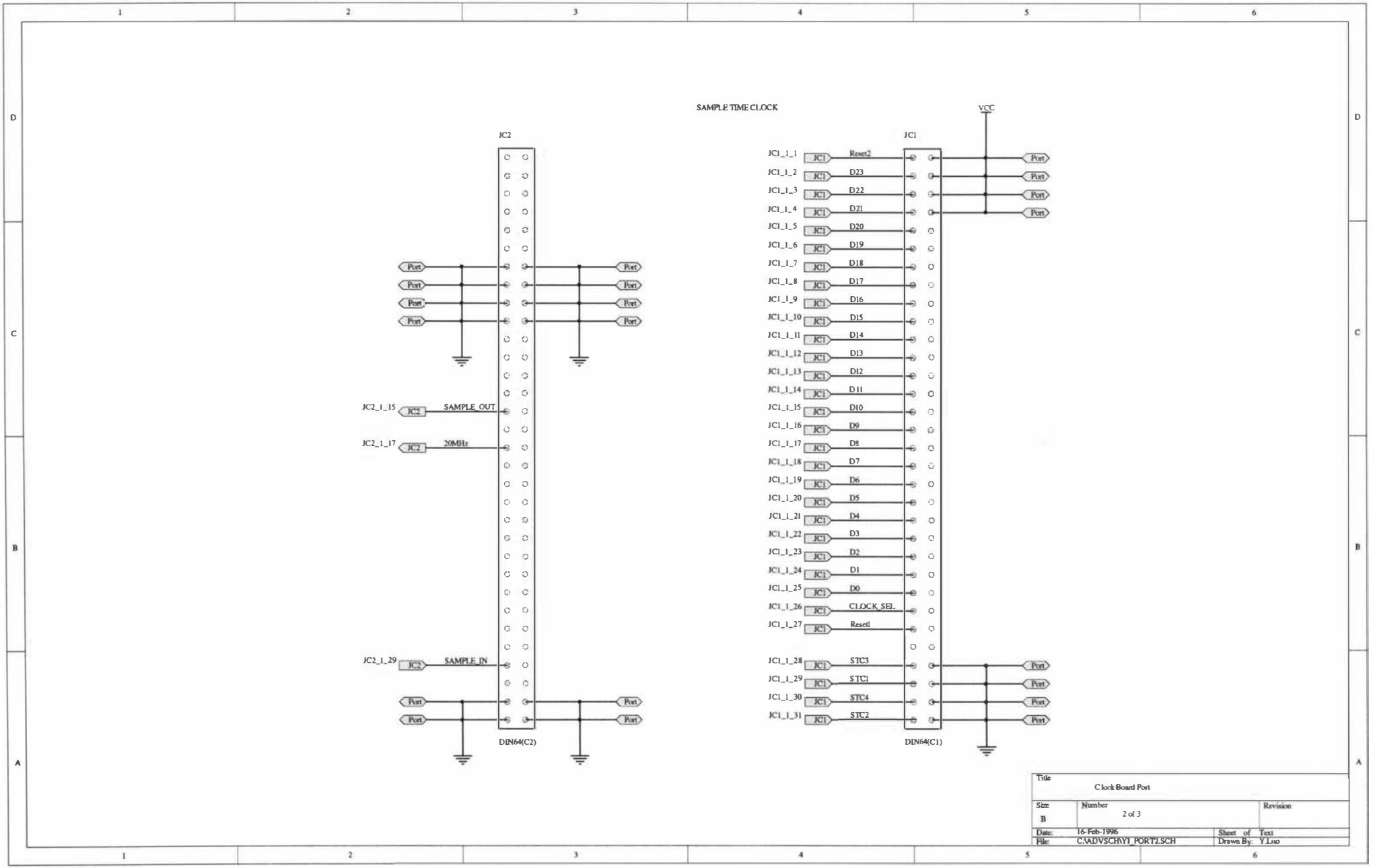
APPENDIX II

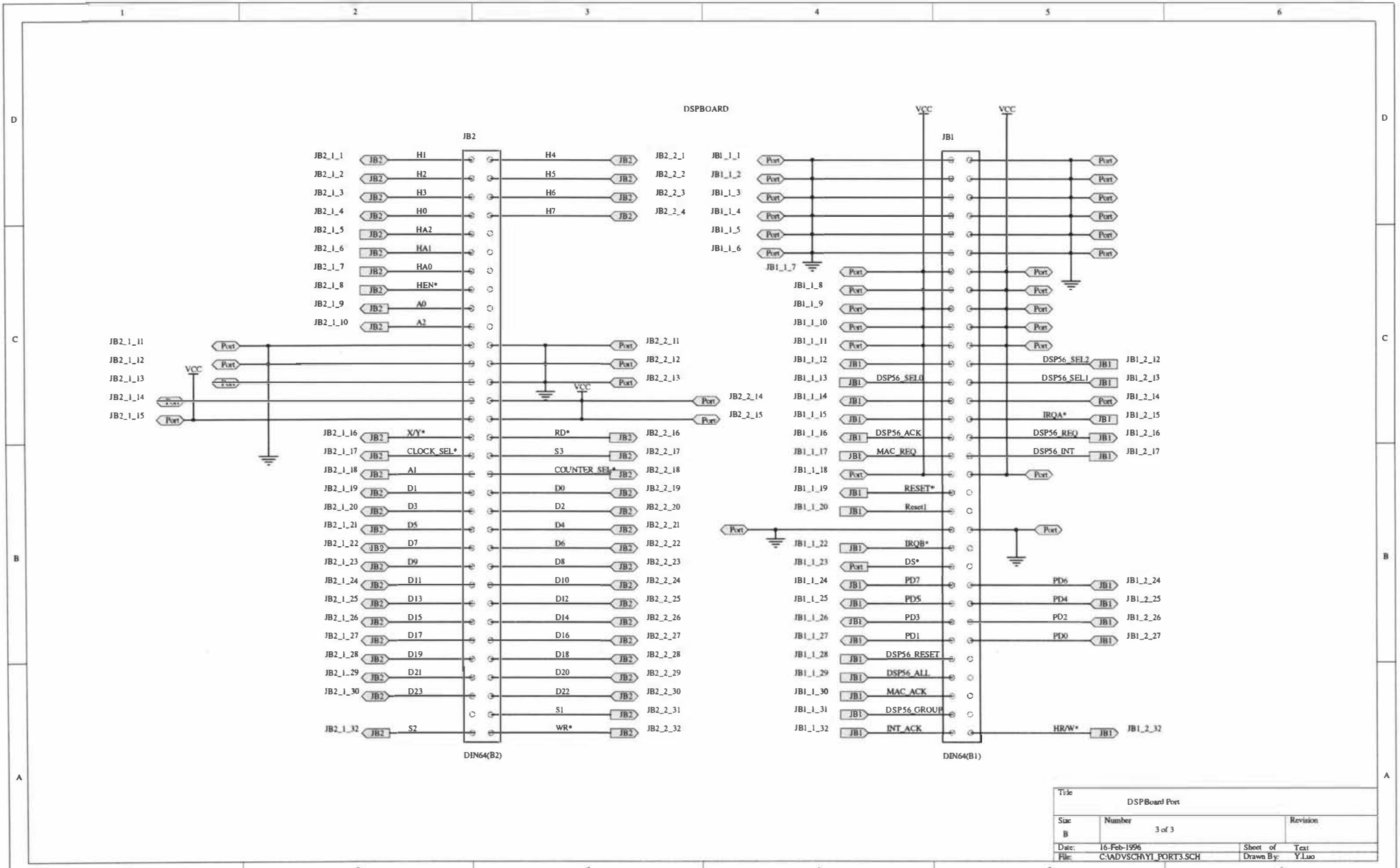
CIRCUIT DIAGRAMS



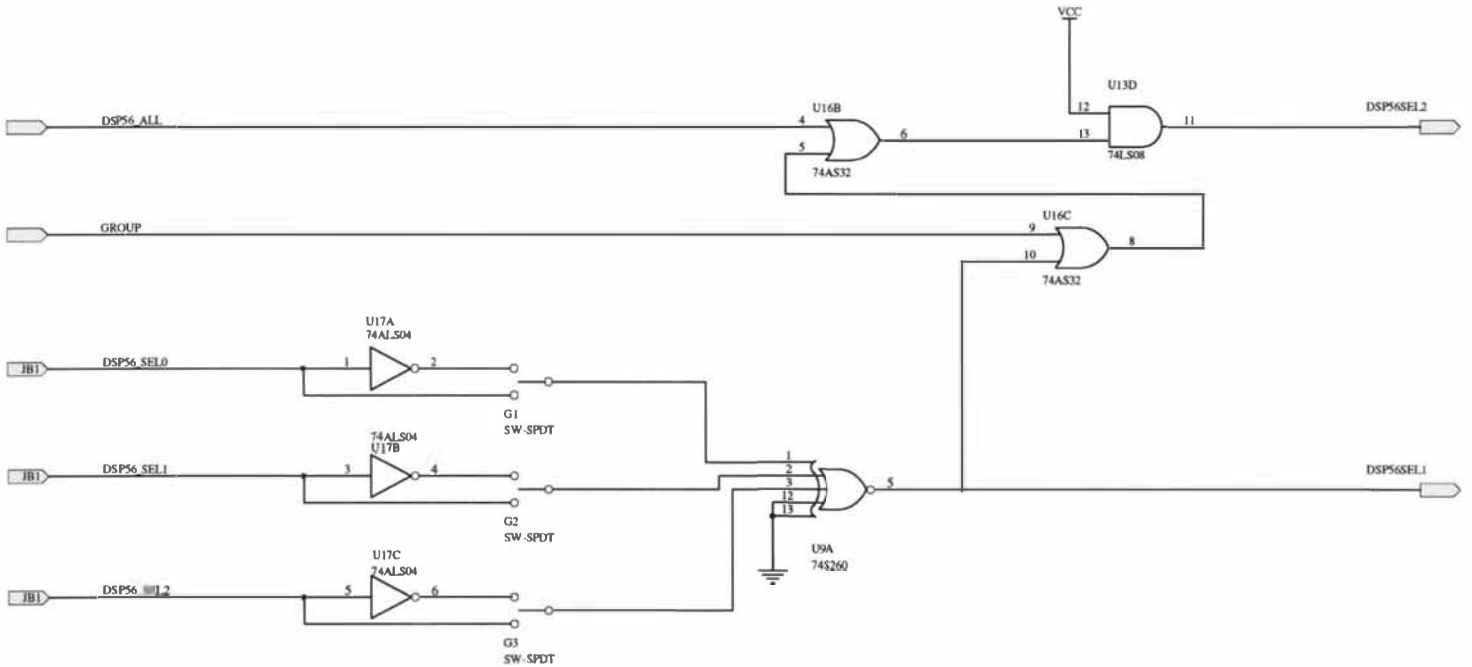
Title: Board Position		
Size: B	Number: 1 of 1	Revision:
Date: 16-Feb-1996		Sheet of
File: C:\ADVSCH\YI_BACKB.SCH		Drawn By: Y.Luo


INTERFACE
J1
19 PD7
37 PD6
18 PD5
36 PD4
17 PD3
35 PD2
16 PD1
34 PD0
15
33
14
32
13
31
12 DSP56_INT
30
11 DSP56_ACK
29
10 DSP56_REQ
28
9 MAC_REQ
27
8 DSP56_SEL0
26
7 DSP56_SEL1
25
6 DEP56_SEL2
24
5 DSP56_RESET
23
4 DSP56_ALL
22
3 MAC_ACK
21
2 DSP56_GROUP
20
1 INT_ACK
DB37
JA2
JA2_1_14 20 MHz
JA2_1_15 STC
JA2_1_25 SAMPLE_IN
DIN64(A2)
COUNTER
VCC
VCC
JA1
JA1_1_6 IRQA*
JA1_1_7 D23
JA1_1_8 D22
JA1_1_9 D21
JA1_1_10 D20
JA1_1_11 D19
JA1_1_12 D18
JA1_1_13 D17
JA1_1_14 D16
JA1_1_15 D15
JA1_1_16 D14
JA1_1_17 D13
JA1_1_18 D12
JA1_1_19 D11
JA1_1_20 D10
JA1_1_21 D9
JA1_1_22 D8
JA1_1_23 D7
JA1_1_24 D6
JA1_1_25 D5
JA1_1_26 D4
JA1_1_27 D3
JA1_1_28 D2
JA1_1_29 D1
JA1_1_30 D0
JA1_1_31 COUNTER_SEL*
DIN64(A1)
Title
Interface Port & Counter Port
Size
B
Number
1 of 3
Revision
Date: 16-Feb-1996
Sheet of
File: C:\ADVSCH\Y1_PORT1.SCH
Drawn By: Y.Luo

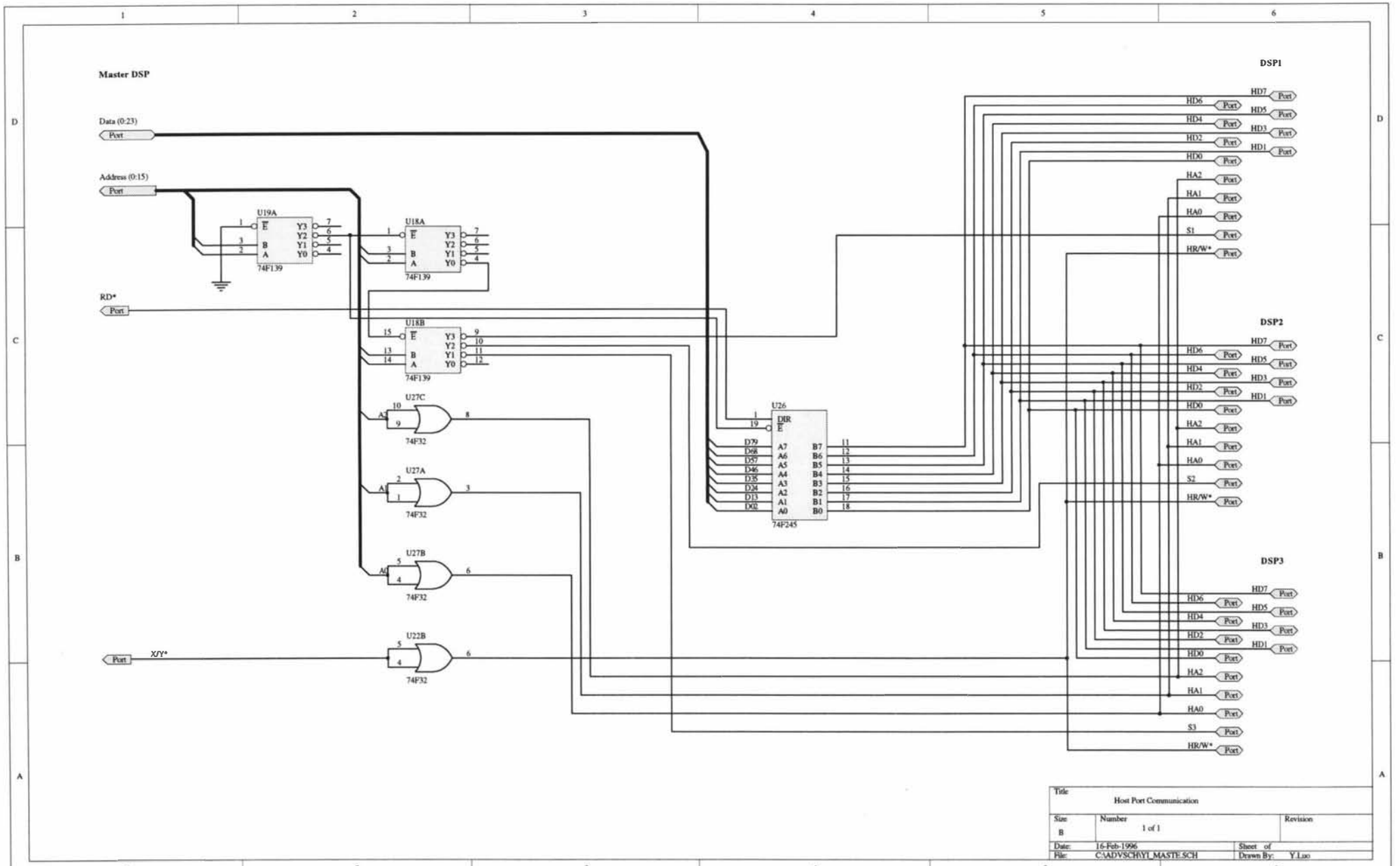

SAMPLE TIME CLOCK
JC2
JC1
VCC
JC2_1_15
SAMPLE_OUT
JC2_1_17
20MHz
JC2_1_29
SAMPLE_IN
DIN64(C2)
DIN64(C1)
JC1_1_1 Reset2
JC1_1_2 D23
JC1_1_3 D22
JC1_1_4 D21
JC1_1_5 D20
JC1_1_6 D19
JC1_1_7 D18
JC1_1_8 D17
JC1_1_9 D16
JC1_1_10 D15
JC1_1_11 D14
JC1_1_12 D13
JC1_1_13 D12
JC1_1_14 D11
JC1_1_15 D10
JC1_1_16 D9
JC1_1_17 D8
JC1_1_18 D7
JC1_1_19 D6
JC1_1_20 D5
JC1_1_21 D4
JC1_1_22 D3
JC1_1_23 D2
JC1_1_24 D1
JC1_1_25 D0
JC1_1_26 CLOCK_SEL
JC1_1_27 Reset1
JC1_1_28 STC3
JC1_1_29 STC1
JC1_1_30 STC4
JC1_1_31 STC2
Port
Title
Clock Board Port
Size
B
Number
2 of 3
Revision
Date: 16-Feb-1996
Sheet of
Text
File: C:\ADVSCH\Y1_PORT2.SCH
Drawn By: Y Luo

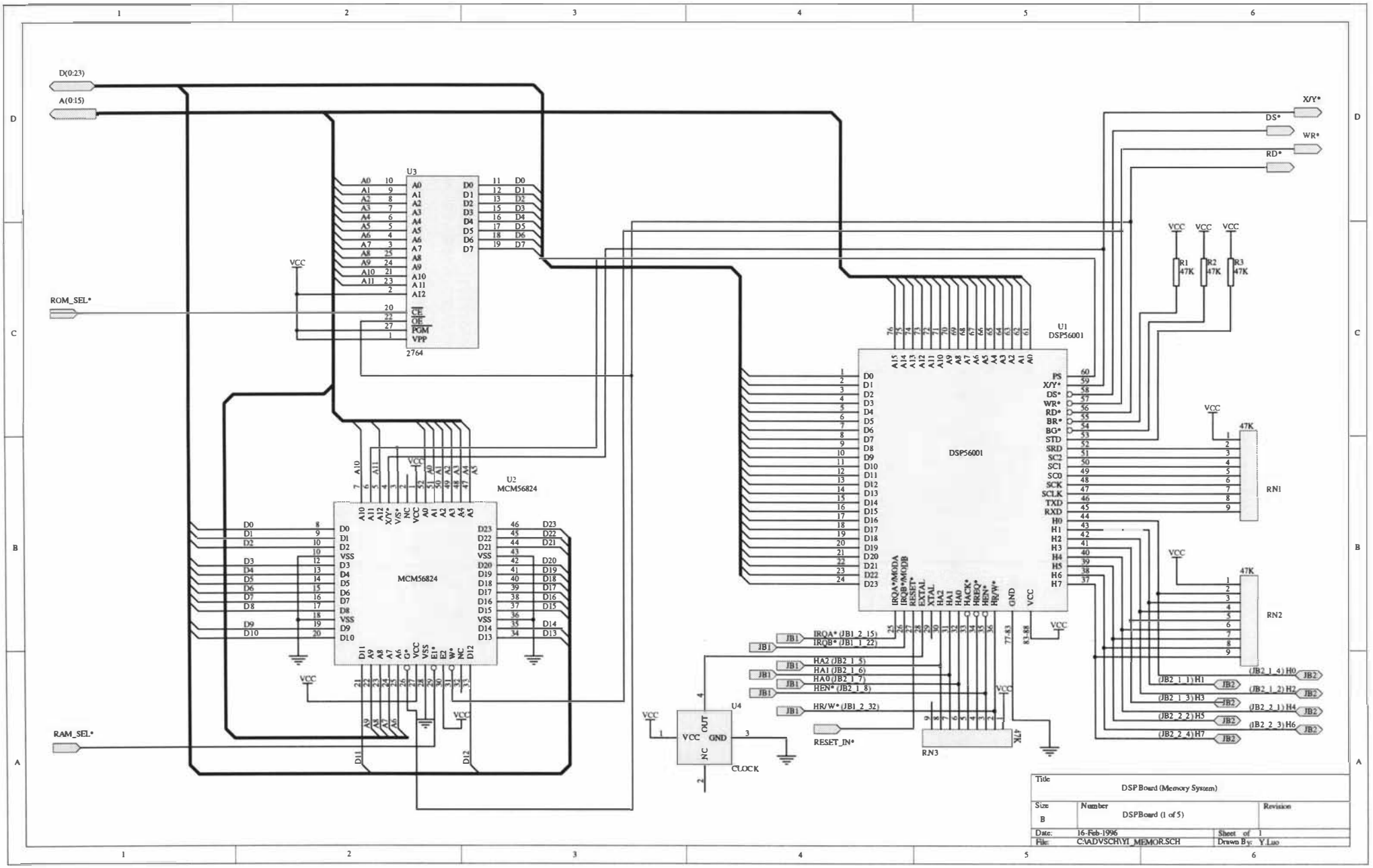


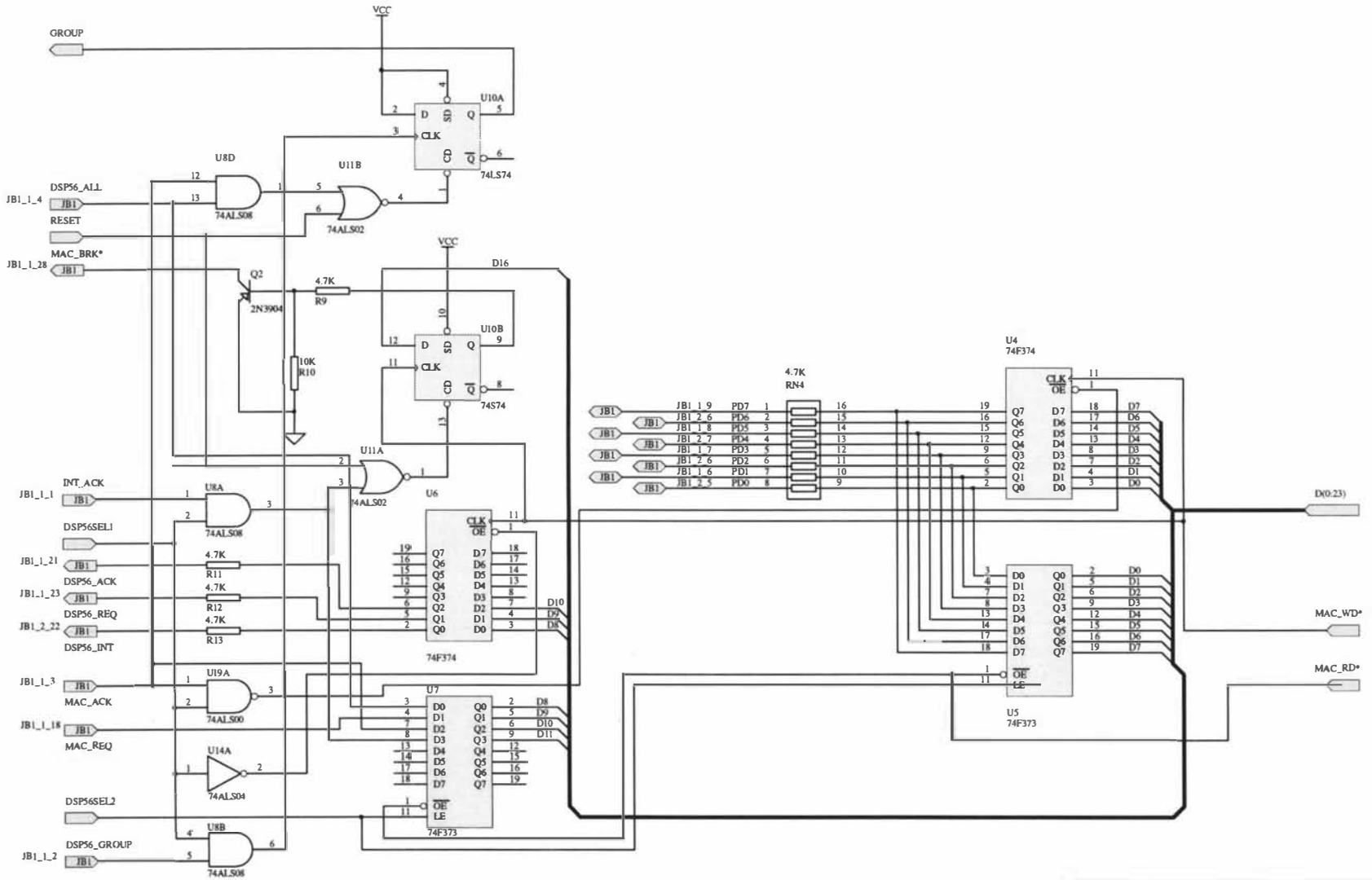
Title: DSPBoard Port		
Size: B	Number: 3 of 3	Revision:
Date: 16-Feb-1996	Sheet of	Text
File: C:\ADVSCH\YI_PORT3.SCH	Drawn By:	Y.Luo



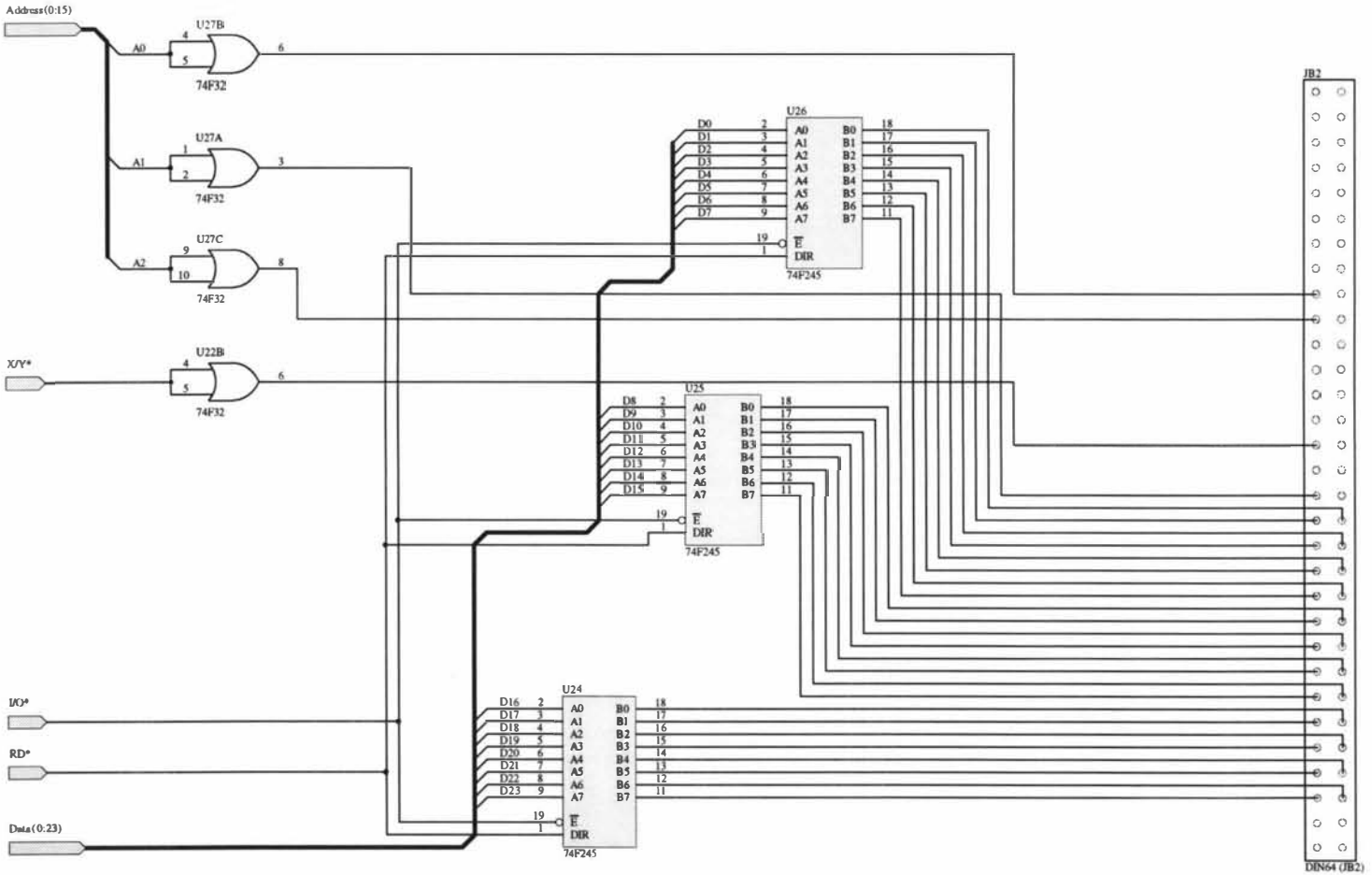
Title: DSP Boards Select		
Size: B	Number: 1 of 1	Revision:
Date: 16-Feb-1996		Sheet of
File: C:\ADVSCH\YI_BOARD.SCH		Drawn By: Y.Luo


Master DSP
Data (0:23)
Port
Address (0:15)
Port
RD*
Port
X/Y*
Port
U19A
74F139
U18A
74F139
U18B
74F139
U27C
74F32
U27A
74F32
U27B
74F32
U22B
74F32
U26
DIR
74F245
DSP1
DSP2
DSP3
HD7
HD6
HD5
HD4
HD3
HD2
HD1
HD0
HA2
HA1
HA0
S1
S2
S3
HR/W*
Port
Title
Host Port Communication
Size
B
Number
1 of 1
Revision
Date:
16-Feb-1996
Sheet of
File:
C:\ADVSCH\YI_MASTE.SCH
Drawn By:
Y.Luo

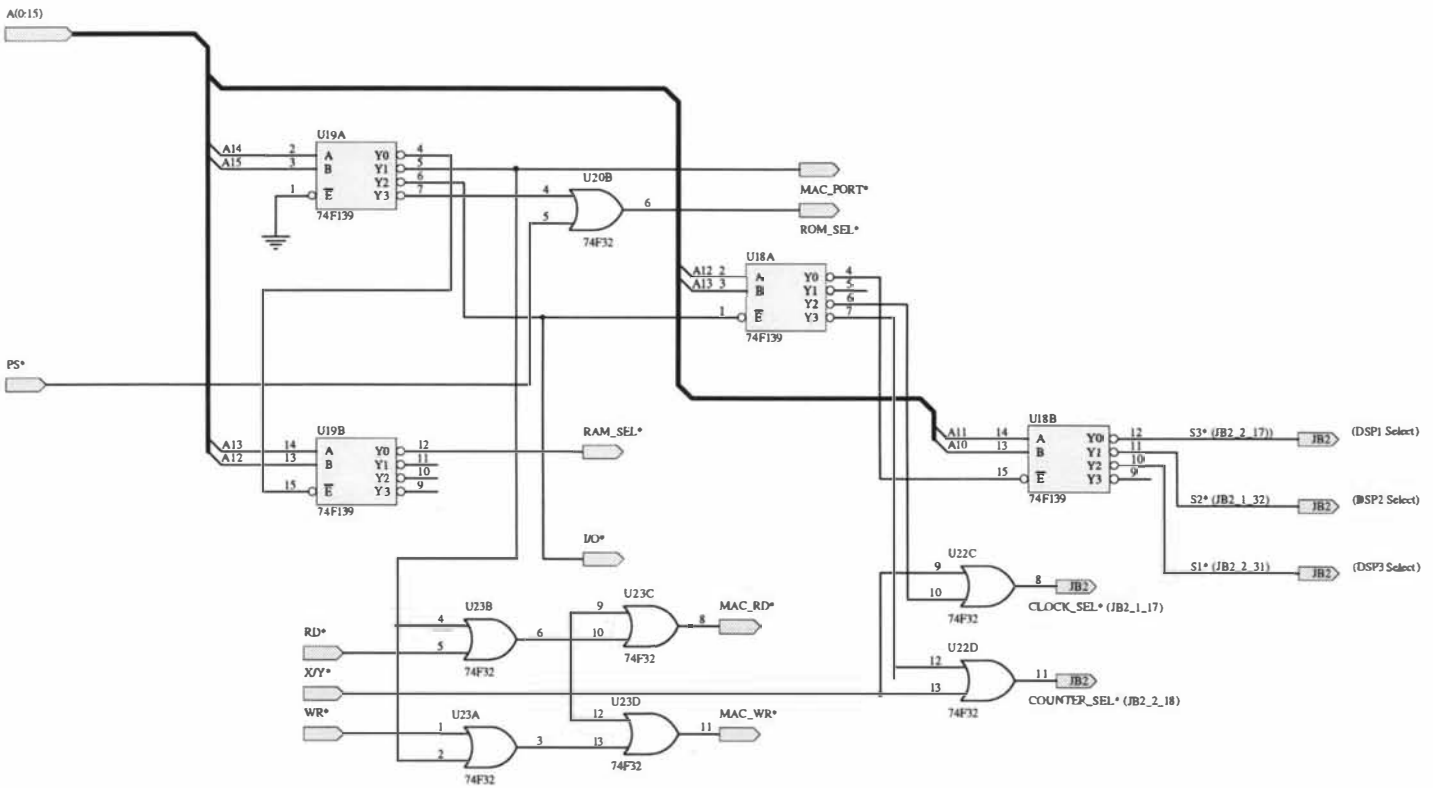

D(0:23)
A(0:15)
ROM_SEL*
RAM_SEL*
X/Y*
DS*
WR*
RD*
U3
2764
U2
MCM56824
MCM56824
U1
DSP56001
DSP56001
U4
CLOCK
RESET_IN*
RN1
RN2
RN3
R1 47K
R2 47K
R3 47K
47K
VCC
IRQA* (JB1_2_15)
IRQB* (JB1_1_22)
HA2 (JB2_1_5)
HA1 (JB2_1_6)
HA0 (JB2_1_7)
HEN* (JB2_1_8)
HR/W* (JB1_2_32)
(JB2_1_1) H1
(JB2_1_2) H2
(JB2_1_3) H3
(JB2_2_2) H5
(JB2_2_4) H7
(JB2_1_4) H0
(JB2_2_1) H4
(JB2_2_3) H6
Title
DSP Board (Memory System)
Size
B
Number
DSPBoard (1 of 5)
Revision
Date: 16-Feb-1996
Sheet of 1
File: C:\ADVSCH\YI_MEMOR.SCH
Drawn By: Y.Luo



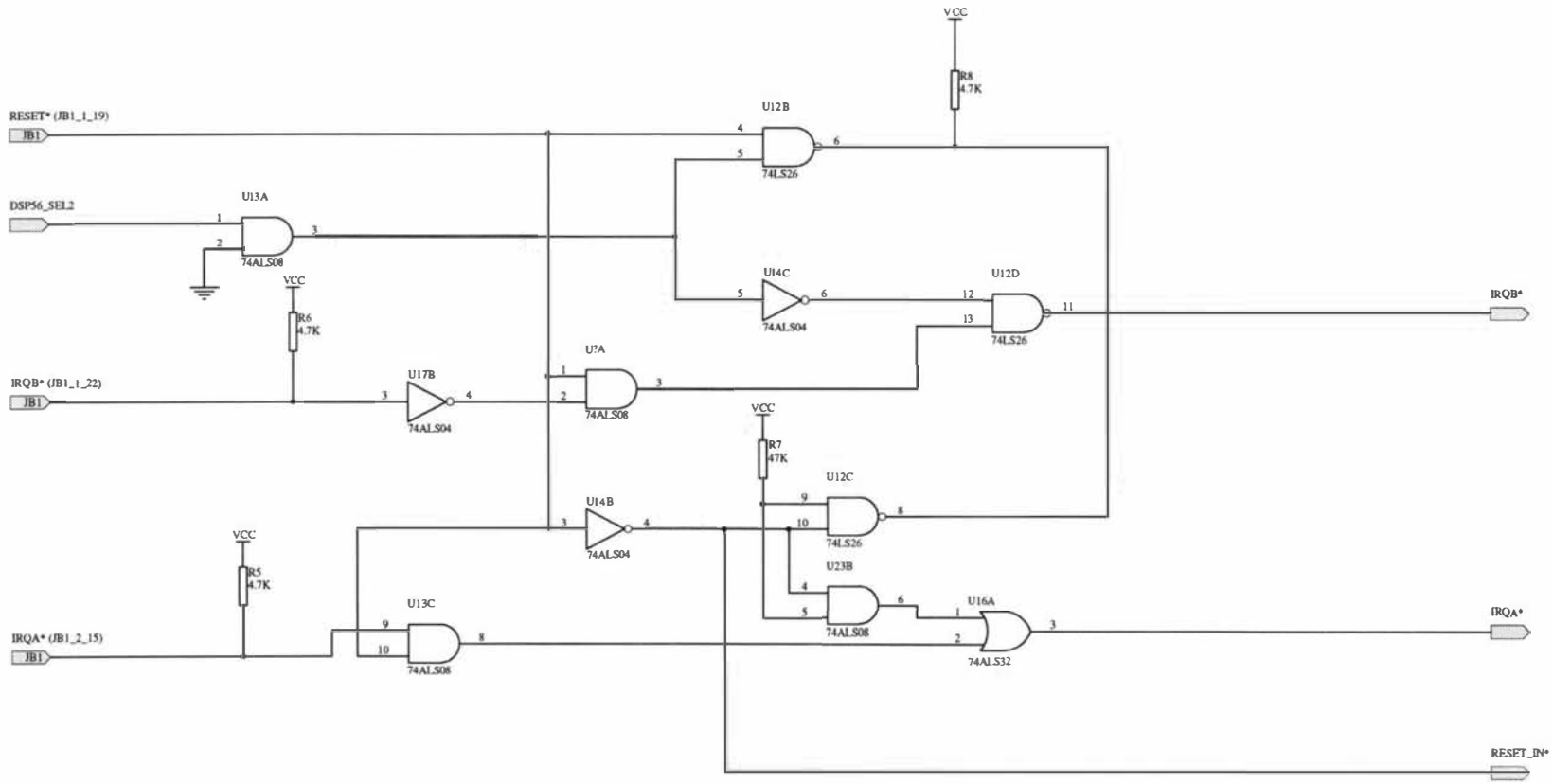
Title	DSP Board (DSP-Computer Interface)	
Size: B	Number: DSP Board (2 of 5)	Revision:
Date: 16-Feb-1996		Sheet of
File: C:\ADVSCH\YI_INTER.SCH		Drawn By: Y.Luo



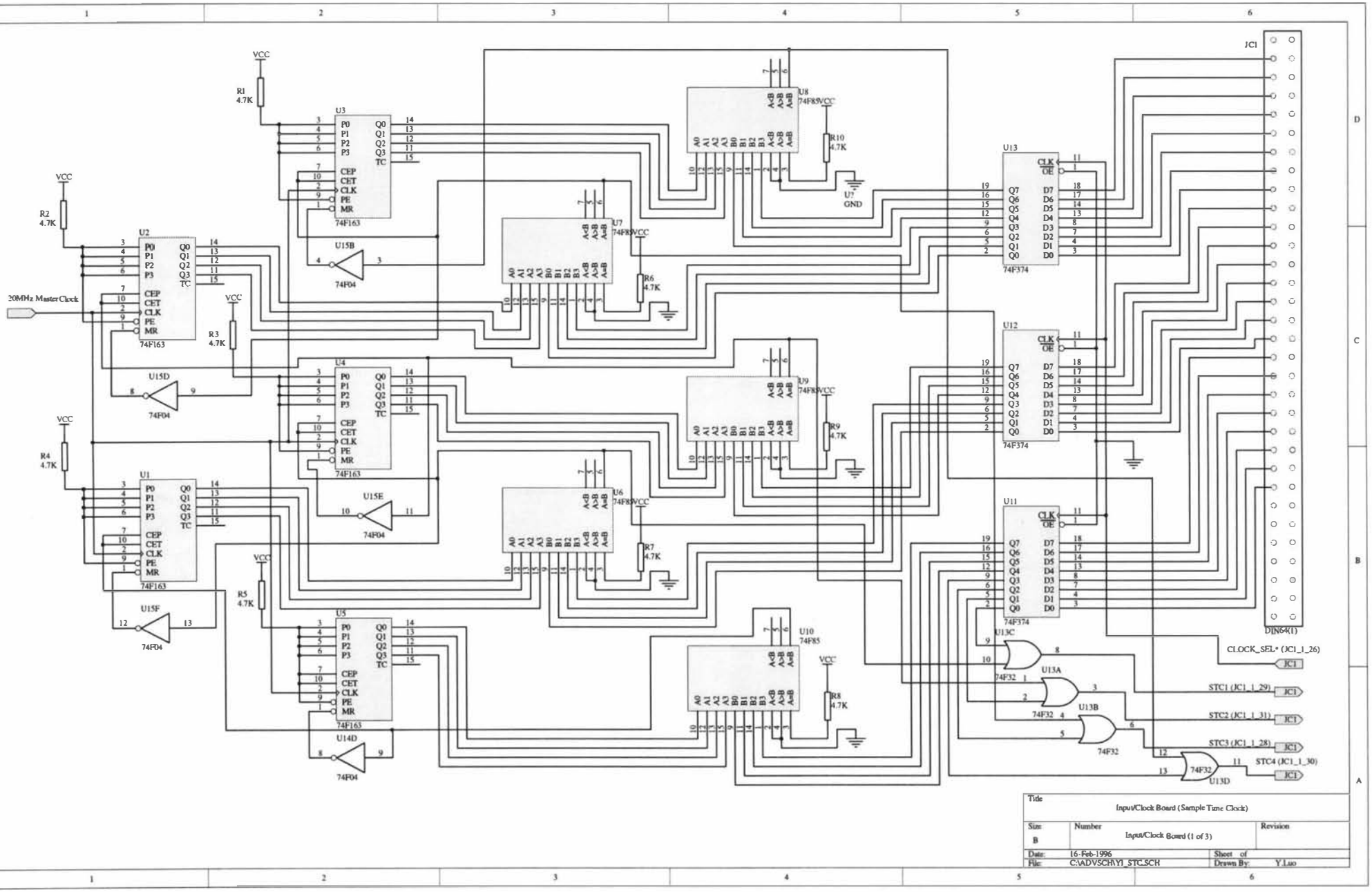
Title: DSP Board (Buffer)		
Size: B	Number: DSP Board (3 of 5)	Revision:
Date: 16-Feb-1996		Sheet of
File: C:\ADVSCH\YI_BUFFE.SCH		Drawn By: Y.Luo

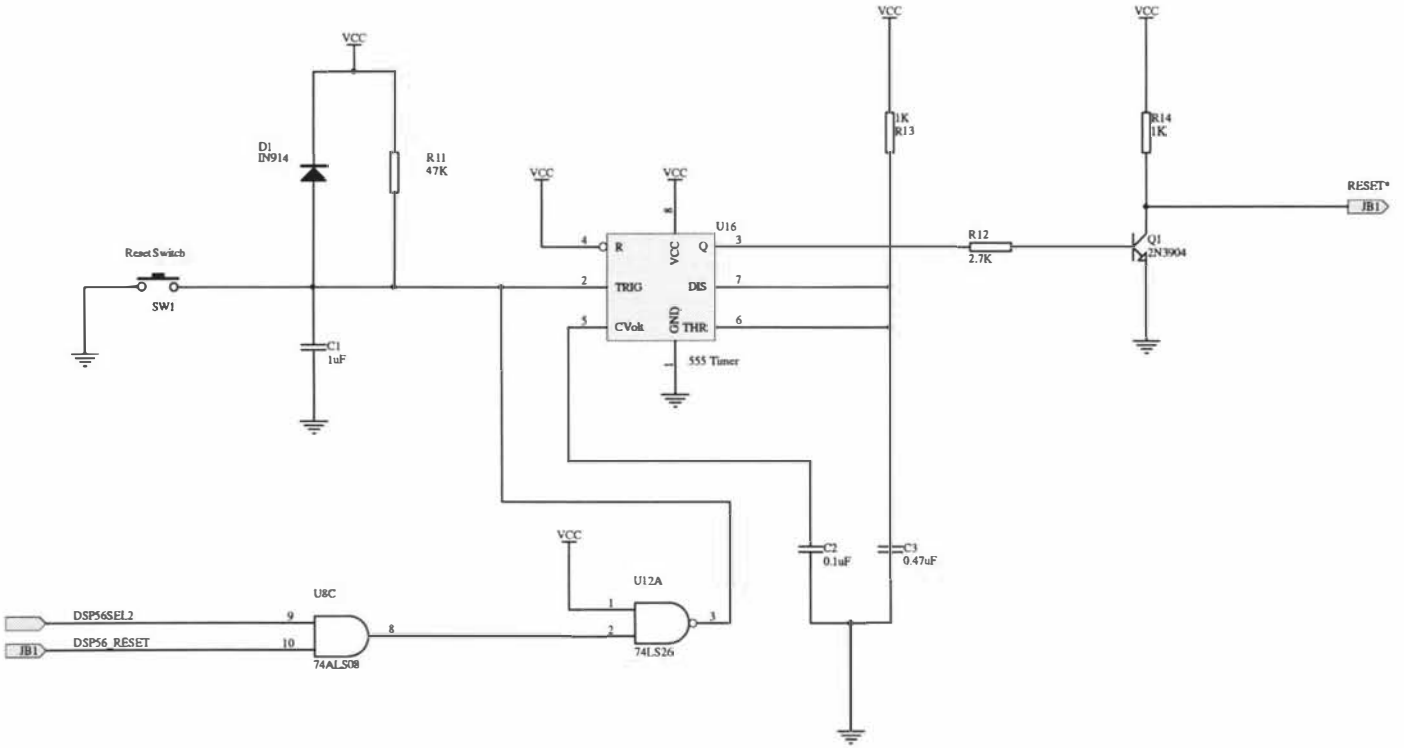


Title	DSP Board (Address Decoder)	
Size: B	Number: DSP Board (4 of 5)	Revision:
Date: 16-Feb-1996		Sheet of
File: C:\ADVSCH\YI_DECOD.SCH		Drawn By: Y.Luo

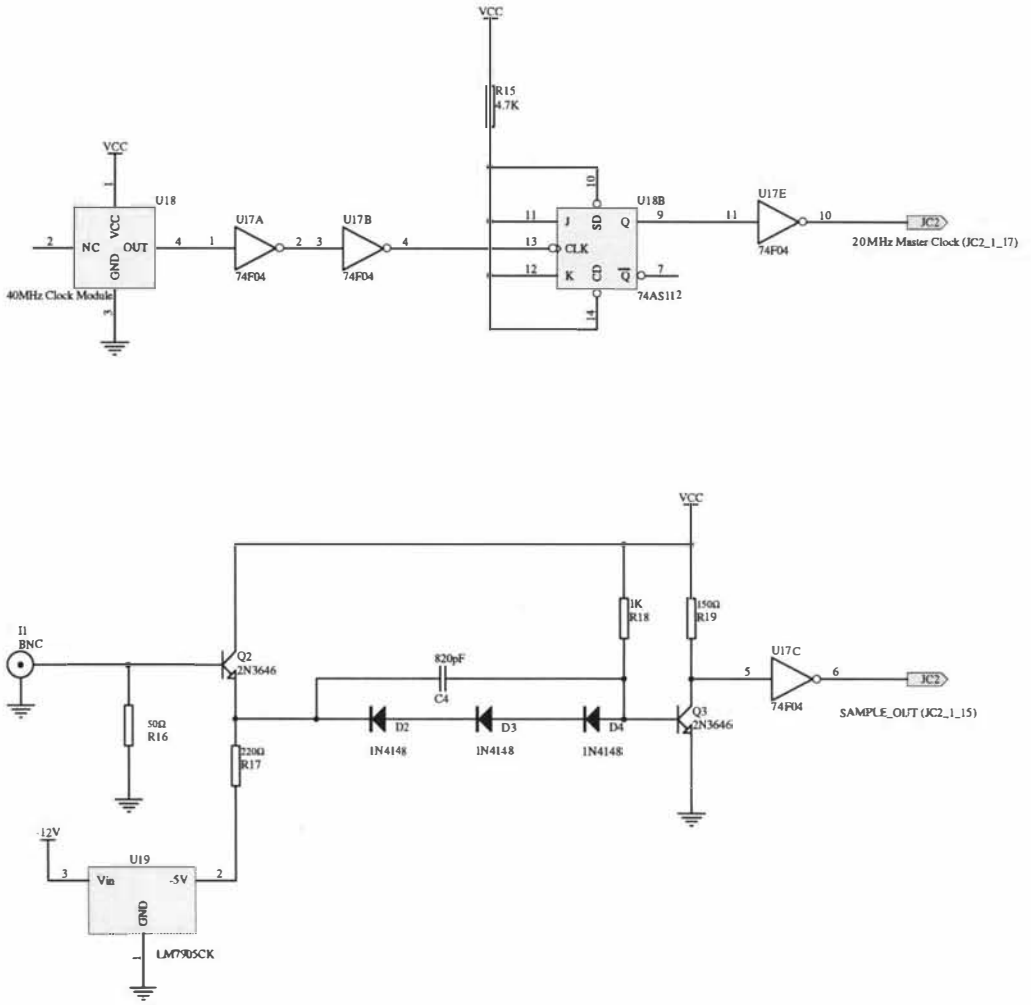


Title	DSP Board (Mode Select)		
Size: B	Number: DSP Board (5 of 5)		Revision:
Date: 16-Feb-1996		Sheet of	
File: C:\ADVSCH\YI_MODE.SCH		Drawn By: Y.Luo	

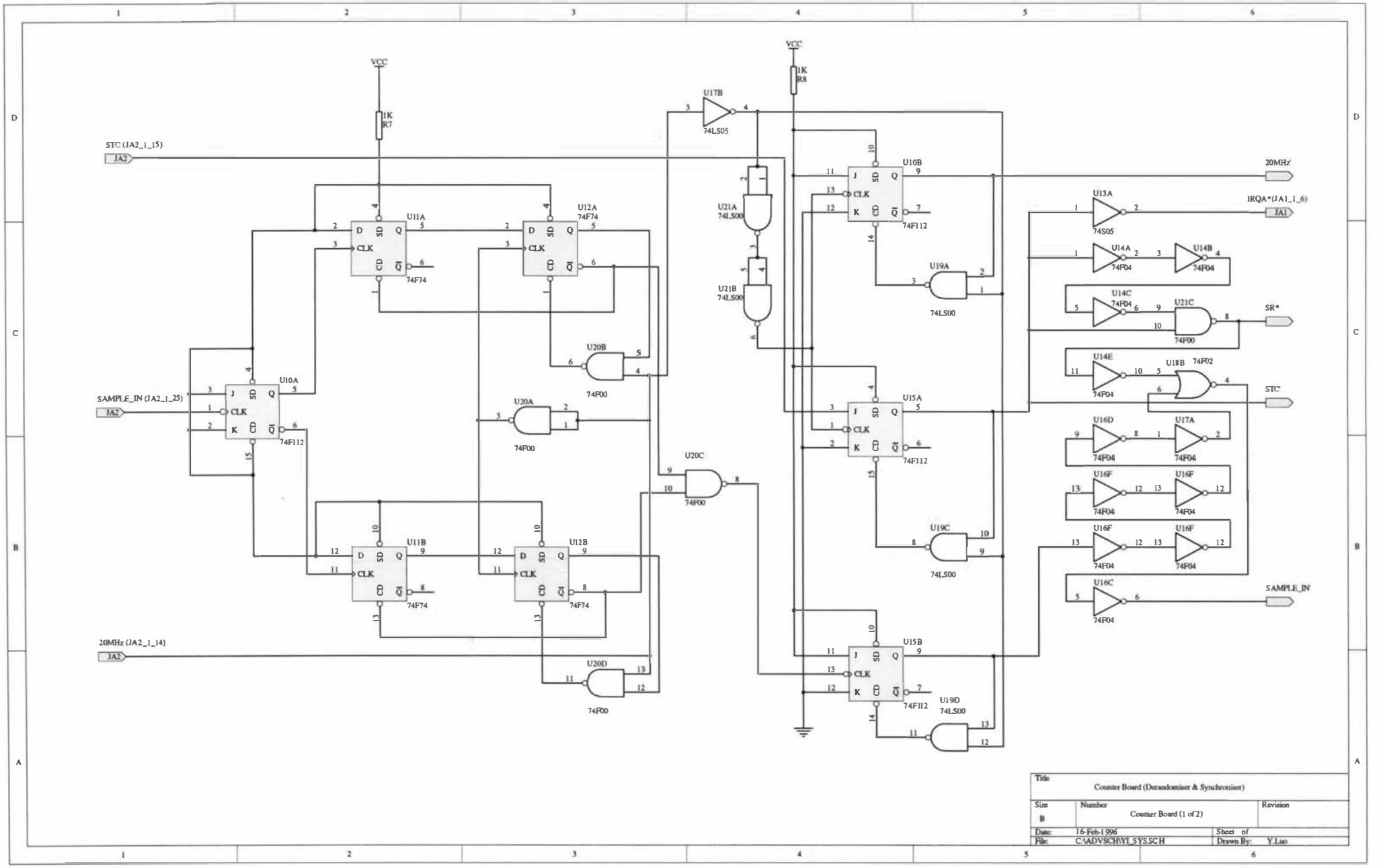

VCC
R1 4.7K
R2 4.7K
R3 4.7K
R4 4.7K
R5 4.7K
20MHz MasterClock
U1
U2
U3
U4
U5
74F163
P0 P1 P2 P3 Q0 Q1 Q2 Q3 TC CEP CET CLK PE MR
U15B
U15D
U15E
U15F
74F04
U6
U7
U8
U9
U10
74F85
A<B A=B A>B A0 A1 A2 A3 B0 B1 B2 B3
R6 4.7K
R7 4.7K
R8 4.7K
R9 4.7K
R10 4.7K
U7 GND
U11
U12
U13
74F374
CLK OE Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0 D7 D6 D5 D4 D3 D2 D1 D0
U13C
U13A
U13B
U13D
74F32
JC1
D[N64:1]
CLOCK_SEL* (JC1_1_26)
STC1 (JC1_1_29)
STC2 (JC1_1_31)
STC3 (JC1_1_28)
STC4 (JC1_1_30)
Title
Input/Clock Board (Sample Time Clock)
Size
B
Number
Input/Clock Board (1 of 3)
Revision
Date: 16-Feb-1996
Sheet of
File: C:\ADVSCH\YI_STC.SCH
Drawn By: Y.Luo

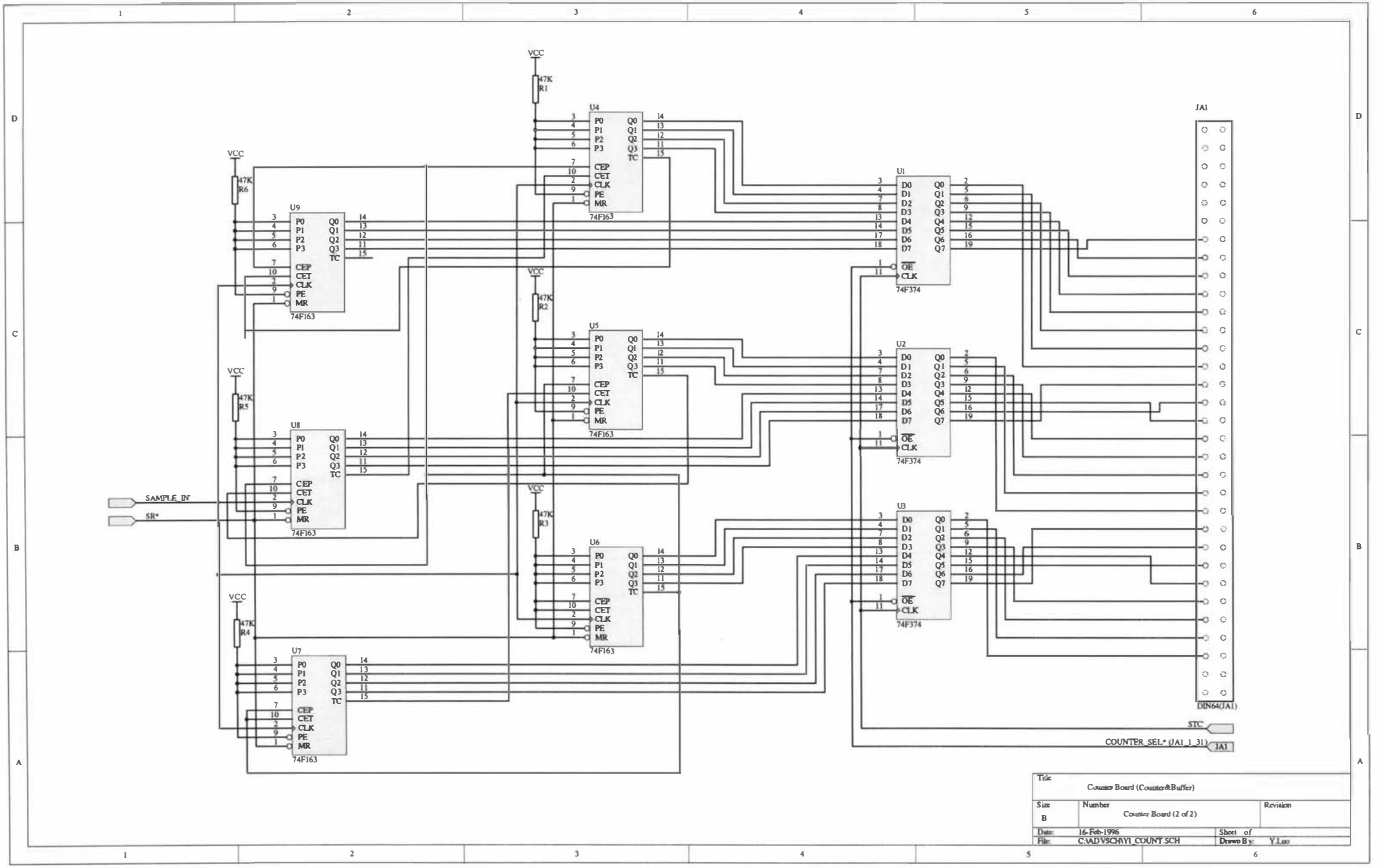


Title	Input/Clock Board (Reset)	
Size: B	Number: Input/Clock Board (2 of 3)	Revision:
Date: 16-Feb-1998		Sheet of
File: C:\ADVSCH\V1_RESET.SCH		Drawn By: Y.Luo



Title	Input/Clock Board (Input&20MHz Clock)		
Size: B	Number: Input/Clock Board (3 of 3)		Revision:
Date: 16-Feb-1996		Sheet of	
File: C:\ADVSCH\YI_CLOCK.SCH		Drawn By: Y.Luo	


VCC
1K
R7
STC (JA2_1_15)
JA2
U11A
U12A
74F74
U10A
74F112
SAMPLE_IN (JA2_1_25)
U11B
U12B
20MHz (JA2_1_14)
U20A
U20B
U20C
U20D
74F00
U21A
U21B
74LS00
U17B
74LS05
U10B
U15A
U15B
U19A
U19B
U19C
U19D
20MHz'
U13A
IRQA*(JA1_1_6)
JA1
U14A
U14B
74F04
U14C
U21C
SR*
U14E
U18B
74F02
STC
U16D
U17A
U16F
U16C
SAMPLE_IN'
Title
Counter Board (Derandomiser & Synchroniser)
Size
B
Number
Counter Board (1 of 2)
Revision
Date:
16-Feb-1996
Sheet of
File:
C:\ADVSCH\YI_SYS.SCH
Drawn By:
Y.Liao


VCC
47K
R1
R2
R3
R4
R5
R6
U1
U2
U3
U4
U5
U6
U7
U8
U9
74F374
74F163
D0
D1
D2
D3
D4
D5
D6
D7
Q0
Q1
Q2
Q3
Q4
Q5
Q6
Q7
OE
CLK
P0
P1
P2
P3
TC
CEP
CET
PE
MR
JA1
DIN64(JA1)
SAMPLE_IN
SR*
STC
COUNTER_SEL.* (JA1_1_31)
Title
Counter Board (Counter&Buffer)
Size
B
Number
Counter Board (2 of 2)
Revision
Date:
16-Feb-1996
Sheet of
File:
C:\ADVSCH\YI_COUNT.SCH
Drawn By:
Y.Luo

APPENDIX III

SOFTWARE PROGRAMS
(Enclosed in Disk)

APPENDIX IV

PUBLICATIONS