DEDICATED MULTICHANNEL A/D CONVERTER MODULE

by

Andrew G. Bell

A Project Submitted to the Graduate

Faculty of Rensselaer Polytechnic Institute

in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF ENGINEERING

Approved:

Dr. Michael Savic
Project Advisor

Rensselaer Polytechnic Institute
Troy, New York

May 1985

CONTENTS

(Continued -- CONTENTS)

LIST OF TABLES

LIST OF FIGURES

(Continued -- LIST OF FIGURES)

Forward

I would like to thank Dr. Michael Savic for his help on this masters project. Keith French and Bob Hamel provided me with valuable insight that made my job easier. Several vendors from Analog Devices provided me with technical assistance on the applications of their parts that reduced the amount of debug time that was spent. In addition, I'd like to thank Bob Gwozdz for building the module, Bev Nichols for drawing the schematics, and Mrs. Nancy Shepherd for taking the time to type this report. Finally, I would like to thank my wife, Linda, for her loving support, my son, Nathan, who was born during the writing of this report who gave me great joy and, God my father, who answers prayers.

Abstract

State of the art multichannel analog I/O modules are generally designed for data acquisition without regards to digital control applications. An external processor interfaces to the module in some sort of synchronous scheme which by its very nature limits either the number of channels that can be selected or the maximum rate of change of these signals. Thus, it is apparent that the design of a data acquisition module with emphasis on digital control applications is needed.

A dedicated multichannel analog I/O module was designed to provide an asynchronous sampling of multichannel input signals and a synchronous interface to an external processor for initializing and the accessing of data. This will allow the external processor to access the binary data and not retard the conversion process. The module is initialized by programming the rate of the sample clocks and the sequence of the analog input channels. The module was designed to interface to Intel's MULTIBUS bus structure and to process analog signals which are preconditioned.

The sections used on this dedicated module can typically be found in analog I/O modules with the exception of the timing controller state machine and the 16 by 16 dual port SRAM. The onboard timing controller provides the intelligence needed to convert the analog input signals independent of the external processor. The analog input signals are multiplexed and converted to 13 and 16 bit words at a through-put rate of 80KHz and 47KHz, respectively. The dual port SRAMs allow the external processor to access the converted data at its leisure without

inhibiting the conversion process. Other unique sections include the decode, programmable clock and nonlinear gain amplifier sections.

PART I

1.INTRODUCTION

The dedicated multichannel A/D converter module (DMA/DCM) differs from currently available analog I/O modules in the following ways. Two independent conversion paths are available. A 16 bit path, for slow rate of change signals, has seven single ended input channels and a 12 bit DAC input channel to provide a 16 bit wide conversion path with a through-put rate of about 47.0 kilosamples per second. The other path is a 13 bit path for fast rate of change signals. It has seven single ended input channels and a 12 bit DAC input channel. The 13 bit conversion path has a through-put rate of about 80.0 kilosamples per second. In addition, the 13 bit path can achieve a 16 bit dynamic range with the aid of a nonlinear gain amplifier (NGA).

The intelligence of the DMA/DCM lies within its ability to perform onboard tasks independent of an external processor. There are five performed tasks that vary in levels of complication. First, the simplest task is the generation of -XACK for MULTIBUS interfacing. This signal is generated by the module when the module has been addressed on a read or write command. Next, if an analog input signal is being converted and a memory read is initiated by the external processor, then a write lockout command will be issued to the module memory to prevent the converted data being read from being corrupted by more data from the converter. Third, the 13 bit path contains hardware to vary the gain of the input signal based on a threshold. This

provides a greater dynamic range for the 13 bit path. The 13 bits contain 12 bits of conversion data and 1 bit, the most significant bit, that indicates whether the gain of the NGA was one or eight based on the threshold. Fourth, the module generates two independent programmable sample clocks used to synchronize the converters and support hardware. These clocks can be programmed to a value between 6.0MHz to less than 1.0Hz by dividing 6.0MHz by an integer value. Fifth, there are two independent sequencing memories. These memories operate as variable length FIFOs and are used to independently control the channel selections.

The emphasis for designing this module was not for data acquisition but rather for digital control applications. Thus, it became necessary to enhance the independence of channel selection via the use of a programmable timing state machine which would control the sequencing of the channels and rate of channel selection.

PART II

2.0 TECHNICAL BACKGROUND

The reasons for designing a "dedicated" multichannel A/D converter module with onboard intelligence arise in part from the evaluation of some background information on analog I/O modules and digital control systems. Part II presents in a general sense some of the reasons, methods and requirements needed for the design of this dedicated (having a specific purpose) module.

Section one of part two establishes a minimal acceptable sampling frequency by evaluating spectral distortion and phase shifting as functions of the sampling frequency. Section two of this part presents, in a brief overview, the performance of state of the art analog I/O modules. Section three of part two compares 12 and 16 bit successive approximation A/D converters with respects to their conversion speeds and resolutions. The final section presents certain design criterion for multichannel A/D converter module design.

2.1 Sampling Requirements for Digital Control Systems

Many digital control systems (DCSs) are designed today to replace existing analog control systems (ACSs). These DCSs must typically meet the same performance requirements placed upon the ACSs. These requirements can be divided into two groups: time (step) response and frequency response. The time response of the system portrays the transient and steady state performances. The frequency response (open and close loop) helps to illustrate the stability and bandwidth of the system.

The Nyquist sampling theory suggests that a signal must be sampled (multiplied by an impulse train) at a sampling frequency which is at least twice the highest frequency of interest.[7] This will, in theory, preserve the frequency information by preventing spectral overlap (aliasing). There are two major difficulties with sampling at the Nyquist rate for DCSs. First, phase due to sampling is added to the system's open loop frequency response. If the added phase is too great, it could lead to system instability and reduced bandwidth. In addition, simple compensation techniques may not provide adequate phase correction. Second, an impulse train is, in reality, a series of square wave pulses which translate into a spectrally overlapping sinc pulse train in the frequency realm. The spectral overlap (aliasing) leads to degraded servo performance. Thus, it becomes necessary to establish a minimal acceptable sampling frequency (MASF), other than the Nyquist rate, by determining how phase and spectral overlap effects the DCS's performance.

Spectral overlap becomes a significant factor when the sampling

frequency (ω_s) is slowed or the sample pulse width is increased. Figure 2.1 shows how spectral overlap of adjacent sinc pulses is a function of the normalized sampling frequency (NSF) and the normalized duty cycle (NDC) of the sampling waveform. It should be noted that generally the MASF for 1% of distortion or less is 10 samples per sample period with a 5% NDC.

The stability of an ACS is illustrated in the frequency realm via the phase margin of the open loop response. A stable system will have less than -180 degrees of phase at the crossover frequency (ω_c). Generally, the phase margin should be 30 degrees or more for both a stable and realizable ACS.[6]

The phase response of a low pass analog filter and its digital equivalence will be compared to show how the normalized sample rate (T) effects the phase response. The normalized sample rate, the inverse of the NSF, is

$$T = \frac{\omega_c}{\omega_s}. \qquad (2.1)$$

Where ω_c is the frequency at crossover and ω_s is the sampling frequency

The transfer function (H(s)) of the low pass filter shown in figure 2.2 is

$$H(s) = \frac{1}{RCs + 1}. \qquad (2.2)$$

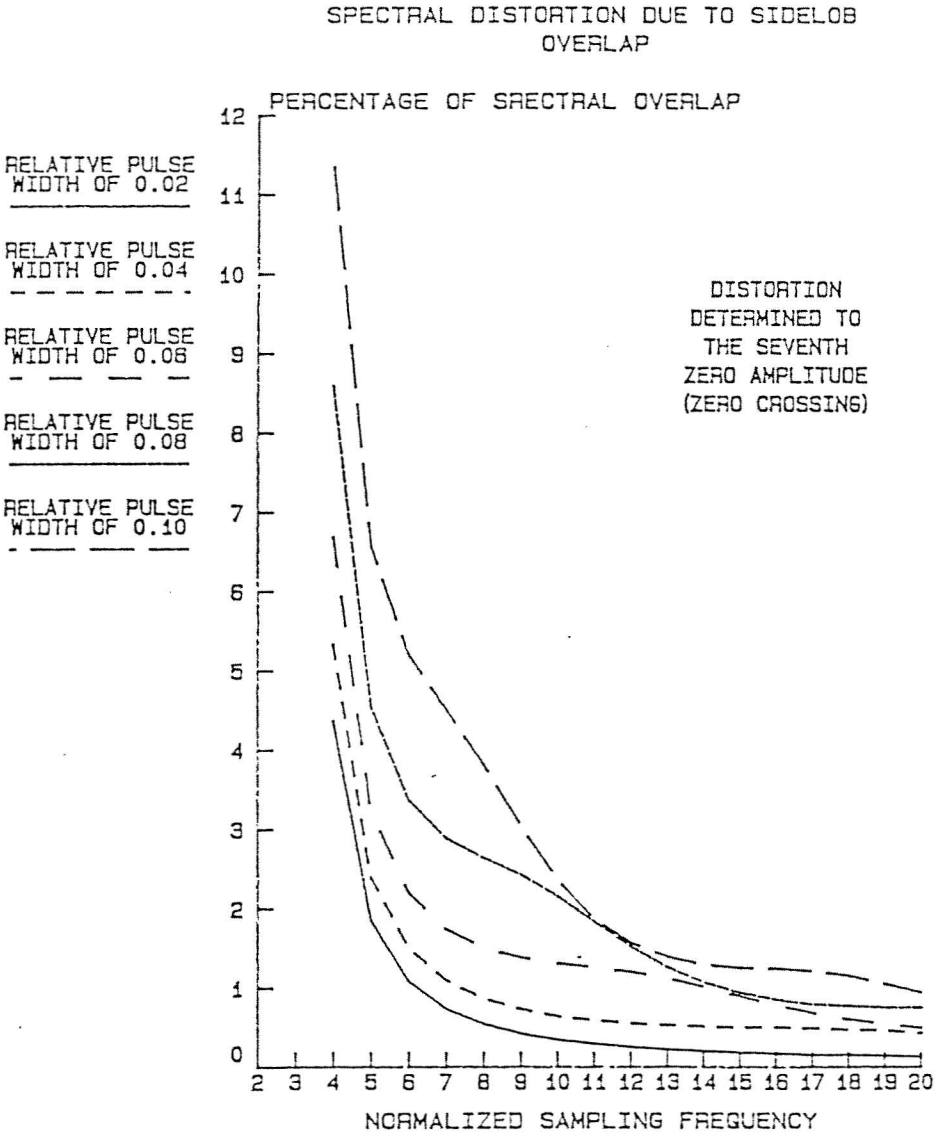

Figure 2.1
Spectral Distortion Due to Sampling

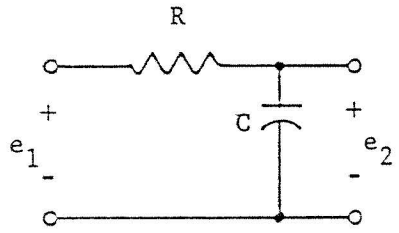

Figure 2.2
Low Pass Analog Filter

(This assumes no initial charge on the capacitor.) The frequency response ($H(j\omega)$) is determined by substituting "$j\omega$" into equation 2.2 for "s". Then $H(j\omega)$ is rationalized into its real ($ReH(j\omega)$) and imaginary ($ImH(j\omega)$) parts. Next, the magnitude ($M(j\omega)$) and phase ($P(j\omega)$) of $H(j\omega)$ can be determined. Thus, $H(j\omega)$ becomes

$$H(j\omega) = \frac{1 - jRC\omega}{1 + (RC\omega)^2} \tag{2.3}$$

and the magnitude and phase are

$$M(j\omega) = [((ReH(j\omega)^2) + (ImH(j\omega)^2)]^{0.5} \tag{2.4}$$

and

$$P(j\omega) = \tan^{-1} [-ImH(j\omega)/ReH(j\omega)] \tag{2.5}$$

or

$$P(j\omega) = \tan^{-1}(-RC\omega). \tag{2.6}$$

The phase response for the analog filter at 1 radian/second, the -3DB point, is -45 degrees when

$$R=C=1. \tag{2.7}$$

This is shown in Figure 2.3.

The digital equivalent of the analog filter can be determined by

using the Pole-Zero placement method[4] to translate H(s) into the Z domain. Thus, the low pass digital filter's transfer function (H(s)) becomes

$$H(z) = \frac{K}{z - e^{-T}}. \qquad (2.8)$$

The value of K, the gain of the digital filter, has no effect on the phase of the filter. The phase (P(z)) of the digital filter is

$$P(z) = \tan^{-1} [-\mathrm{Im}H(z)/\mathrm{Re}H(z)] \qquad (2.9)$$

or

$$P(z) = \tan^{-1} [-\mathrm{Sin}(T) / (\mathrm{Cos}(T) - e^{-T})] \qquad (2.10)$$

A comparison of the analog and digital filter responses shows that as the NSF increases the phase difference between the two filters decreases. Thus, in general terms, a "good" approximation for P(z) is

$$P(z) = P(j\omega) - 180T/\pi. \qquad (2.11)$$

Figure 2.3 illustrates the actual and approximate phase of the digital

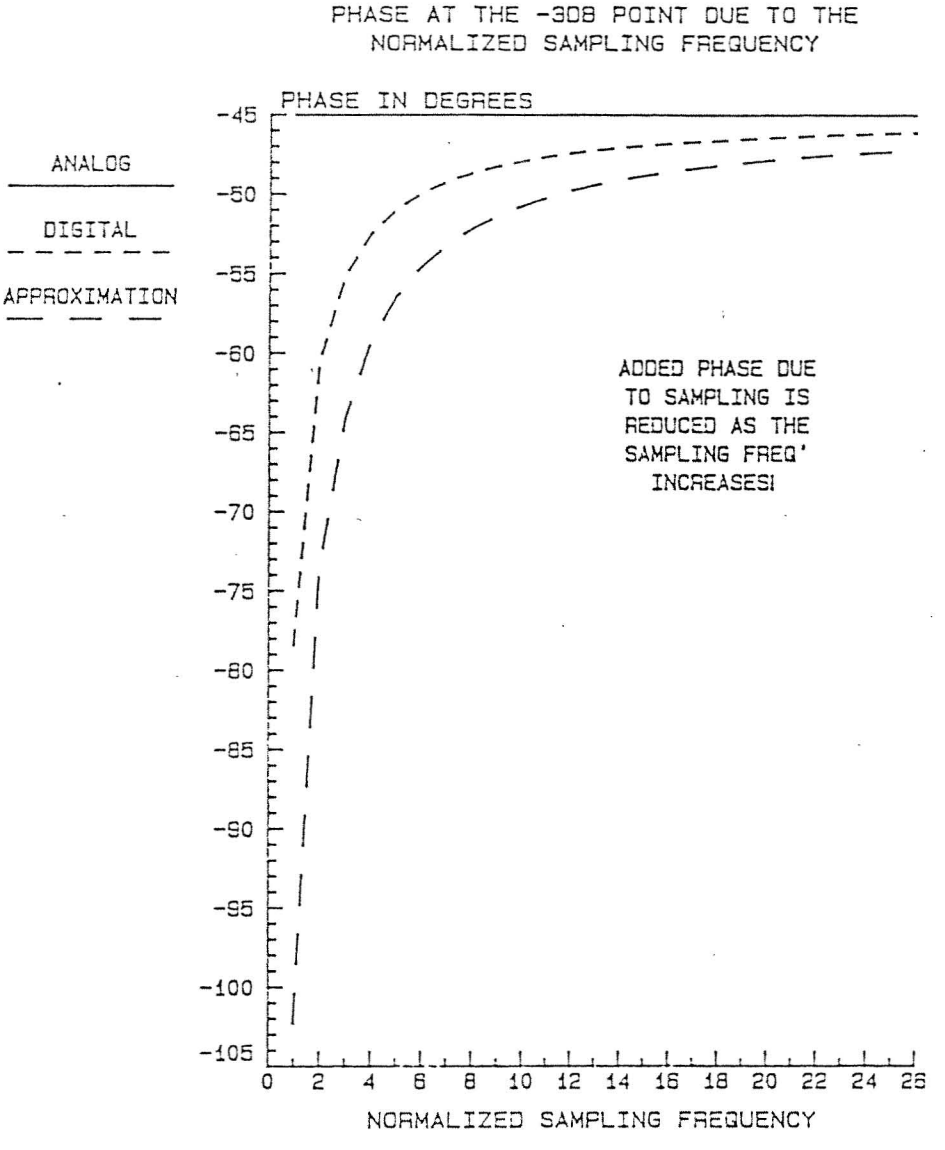

Figure 2.3
Phase at the -3DB Point Due to the
Normalized Sampling Frequency

filter as functions of the NSF and the phase of the analog filter at the -3DB point. It should be noted that if the NSF is 10 samples per sample period. Then the added phase due to sampling is approximately 3 degrees. This is a relatively insignificant amount of added phase. However, if the NSF is at the Nyquist frequency of 2 samples per sample period, then the added phase due to sampling is approximately 16 degrees. This may introduce an unacceptable amount of phase and results in a greatly reduced phase margin. The MASF is dependent upon how much the phase margin can be reduced.

A digital filter will typically interface to a continuous plant via a zero order hold (ZOH). By definition,[5] the transfer function for a zero order hold is

$$ZOH(s) = \frac{1-e^{-sT}}{s} \qquad (2.12)$$

or simply

$$ZOH(j\omega) = \frac{2\sin(180T)}{T\omega e^{j180T}}. \qquad (2.13)$$

The phase added due to the zero order hold can be normalized with the frequency of interest and is equal to

$$PH(j\omega) = -180T. \qquad (2.14)$$

(This phase and the phases associated with a first order hold and a

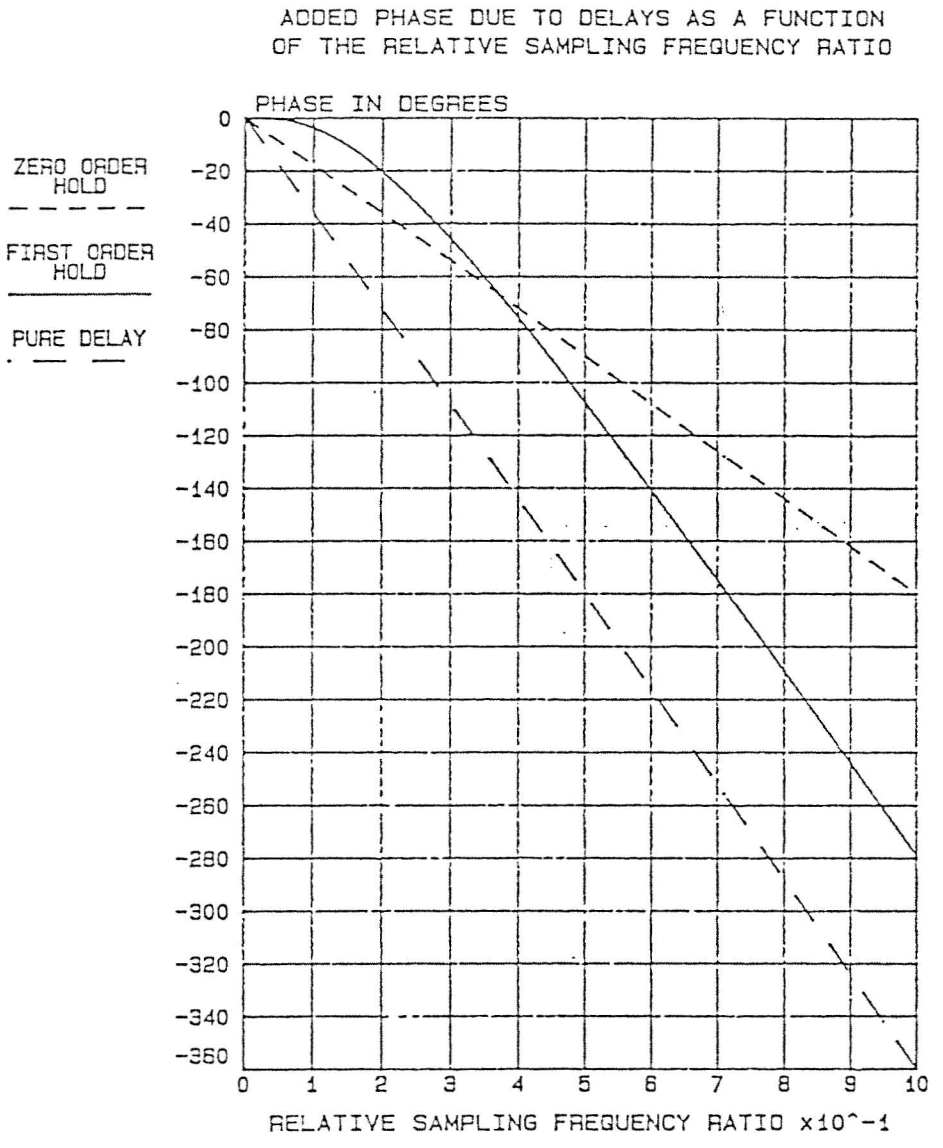

Figure 2.4
Phase Due to Holds Versus
the Relative Frequency

pure delay are plotted in Figure 2.4.) The phase added due to the ZOH is linear with respect to the NSF. If the NSF is 10 samples per sample period, then the added phase due the ZOH is approximately 18 degrees. However, if the NSF is at the Nyquist frequency of 2 samples per sample period, then the added phase due to the ZOH is approximately 90 degrees. As the NSF decreases it becomes more apparent that the phase margin will quickly reduce to nothing.

The total approximate phase added (PA(jω)) due to sampling and the ZOH to the analog system's phase response at the -3DB point can be expressed as

$$PA(j\omega) = 180T(1+1/\pi). \qquad (2.15)$$

The designer should determine how much of a phase margin reduction is acceptable and force T to the rate which will at least provide this phase margin. It has been suggested[4] that for a system with human feedback, that the NSF be at least 10 samples per sample period. From a spectral distortion viewpoint, the NSF should be at least 10 samples per sample period for a 1% or less distortion level with a NDC of 5%. Thus, it can be concluded that generally the MASF for DCSs should be at least 10 samples per sample period. This MASF will result in a combined added phase at the -3DB point of 20.9 degrees or a 5.8% time delay of the original signal.

Figure 2.5 shows the digital and analog filter's step responses. The effective analog response of the digital filter is shown as a phase shifted analog response in Figure 2.5 also.

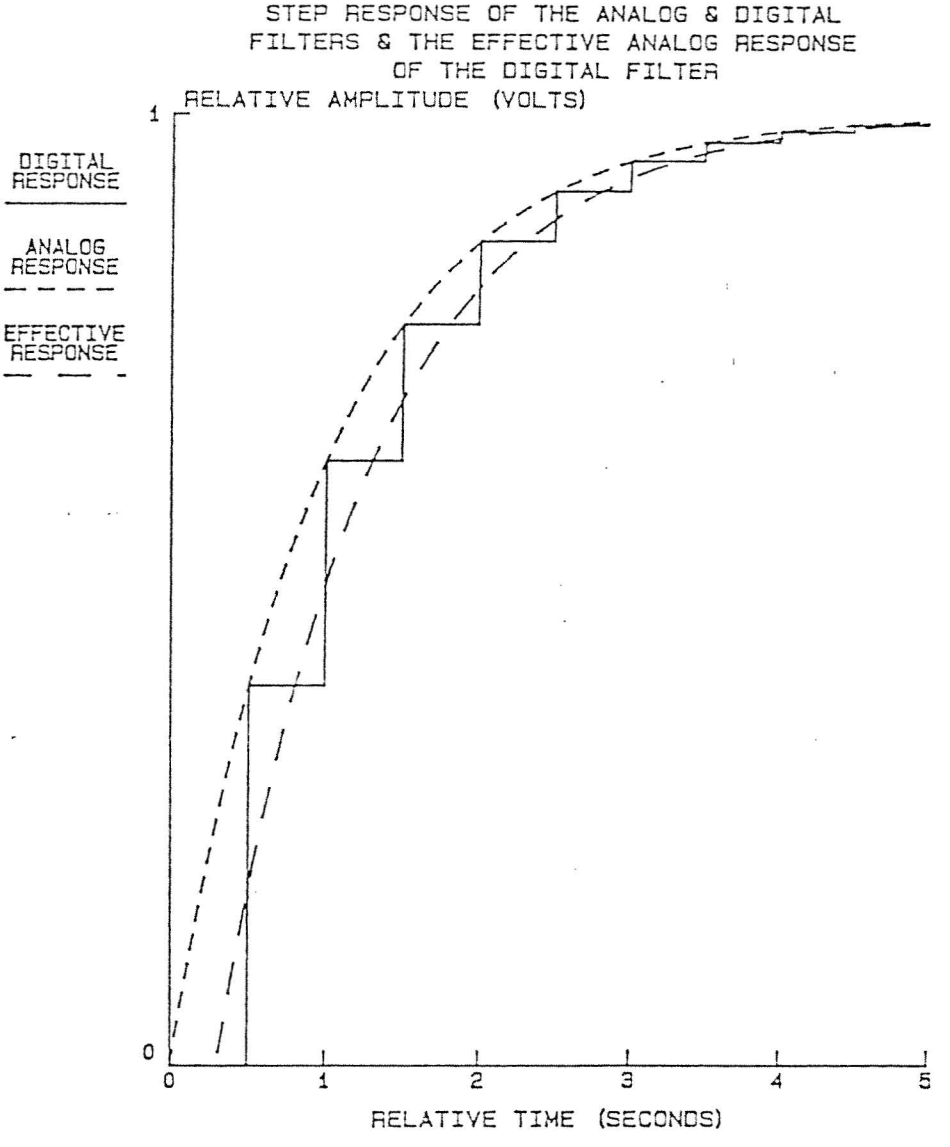

Figure 2.5
Step Response of Analog and
Digital Filters

2.2 Intelligent Analog I/O Modules

Most analog I/O modules to data are limited to 12 bits of resolution for the converted signals[21]. The through-put rate of these modules varies from 9.5 to 50.0 kilosamples per second. The number of channels range from eight differential to thirty-two single ended inputs. The majority of the modules interface to an eight bit data bus in a synchronous fashion. Two reads commands are typically required to access all twelve bits of converted data. In addition, the external system generally synchronously controls the sample rate and channel selection.

The intelligence of an analog I/O board is defined as the ability of the module to independently control onboard functions without external commands except for initialization and the accessing of converted data. Module intelligence would include the programmable gain amplifier (PGA) function for amplifying input signals on a per channel basis.[23] Many modules provide this function at discrete gains, i.e. 8X, 2X, 12024X. This gain usually is changed via the reinitialization of the module.[24] Certain modules are capable of being programmed to flag the external system in a particular manner upon completion of a conversion. The flagging of the system is called scan control and includes four modes.[22] The modes are a CPU interrupt, holding data until its read, polling CPU when data is ready, and holding data and polling CPU when data is ready. Other intelligent functions[20] include the option to select between serial or parallel interfacing, short cycles with or without auto gain, self test capabilities, and trigger control for starting conversions. All of

the aforementioned modules generally require a halting of data conversion when the external processor accesses the module memory. It would be unfair to say that no analog I/O modules provide the intelligence needed for DCS applications. However, few if any have been designed with DCS application needs as the chief emphasis.

2.3 A/D Conversion Time vs Bit Resolution

There is a definite limitation in using a successive approximation (SA) 16 bit A/D converter as opposed to using the more popular SA 12 bit A/D converter. That limitation is conversion or aperature speed. Conversion speed (CS) is defined as the amount of time the converter needs to make a single conversion to the full scale resolution of the device.[1] The typical SA 12 bit A/D converter has a conversion speed of from 1 to 50 microseconds. The typical SA 16 bit A/D converter has a conversion speed of from 17 to 170 microseconds. The ideal SA 12 bit A/D converter is 16 times faster than the ideal SA 16 bit A/D converter.

However, this is not the entire picture. The ideal SA 16 bit A/D converter is 16 times more accurate than the ideal SA 12 bit A/D converter due to the resolution difference. The amount of error[1] associated with the converter is equal to

$$e = \frac{1}{2^n} \qquad (2.16)$$

where "n" is the bit size of the converter, i.e. 12 or 16 bits. The

error (nonlinearity) of all parts that process the analog signal before it reaches the A/D converter should be less than the error associated with the converter.

There is another consideration that is related to converter selection. The ideal SA A/D converter requires[1] a sample and hold to reduce its effective CS if the signal being converted has a period greater than

$$t_{max} = 2\pi(CS)2n \qquad (2.17)$$

For example, a 12 bit converter with a CS of 1 microsecond needs a sample and hold if the frequency of the input signal is greater than 38.85 Hz or if the period is faster than 25.74 milliseconds. A 16 bit converter with the same 1 microsecond CS would require a sample and hold if the input signal is greater than 2.428 Hz or faster than 411.7 milliseconds.

2.4 Optimum Usage of A/D in Multichannel Configuration

The optimum usage of an A/D converter in a multichannel configuration[8] requires the designer to weigh several items of concern. First, can the A/D be shared between several channels or should certain channels have dedicated A/D converters? Will all input channels have the same dynamic range or will certain channels require greater input amplitudes? What will the rate of change of the signals on these channels be and how quickly must the conversion take place? How much can be spent on parts and what is the power limitation? These

questions and more must be addressed by the designer in an attempt to define what multichannel configuration should be used in their particular application.

The limitations placed on the design of a multichannel system can be divided into the following groups; cost, power, size, application speed, accuracy, and module intelligence. The first three limitations force the designer to take inventory of their wallet, their power consumption limits and the amount of real-estate (board space) that they have available. The application in this particular case was to design a multichannel module that would interface to a digital control system. Thus, it became necessary to focus in on certain performance requirements (see Section 2.1) that probably would not be important in a data acquisition application that is not control related. Speed and accuracy are inversely related to one another (see Section 2.3). The designer should do a trade-off study between the rate of change of the input signals and the amount of resolution required for his application. The final parameter of interest is the module intelligence (see Section 2.2). Certain applications will require no onboard intelligence. The digital control application of the analog I/O module would generally require either a fast interface between the external processor or an independent control of the sampling and conversion performed on the module. Thus, an optimum design is relative to the aforementioned limitations and cannot be qualitatively defined.

PART III

3.0 DEDICATED MULTICHANNEL A/D CONVERTER MODULE HARDWARE

Part three is divided into seven sections and deals with the dedicated multichannel A/D converter hardware. This part covers both the philosophy of design and the design itself. Although every detail of the actual module is not explained in this part. The reader should be able to understand the detailed schematics in section seven after reading part three.

Section one of part three describes the sample timing controller circuit. This circuit provides the independent control over the channel selection and the rate of selection after the module has been initialized. Section two explains how the nonlinear gain amplifier, a cousin to the programmable gain amplifier, functions. This section also describes how hysteresis around the thresholds of the nonlinear gain amplifier was designed. Section three of part three explains, in a general sense, the 12 bit A/D converter path. While section four describes the 16 bit A/D converter path. The hardware needed to interface to Intel's MULTIBUS is explained in section five. Section six deals with the 16 by 16 dual port SRAMs and write lockout logic. The final section explains the module buffering, signal decoding and programmable clock circuits.

3.1 Sample Timing Controller Circuit

The heart of the Dedicated Multichannel A/D Converter Module is the Sample Timing Controller (STC). The controller, one for both converter paths, is divided into three sections; the programmable clock (PC), the window logic (WL) and the sequence memory (SM). These three subsections can be looked at as a programmable length FIFO. The FIFO can be programmed to be from 1 to 256 nibbles long and 4 bits wide.

The PC shown in Figure 3.1 generates the clock for the converter path via the use of a AM9513 IC. The AM9513 contains five independent 16 bit counters which can be configured in various ways. The PC consists of two independent 32 bit counters which divide the 6.0MHz clock by the value loaded into the 32 bit counter. The count value is loaded into the 32 bit counter is in negative 2's complement notation and up counted to its terminal count. (See section 3.7 for more information.)

The WL shown in Figure 3.2 is an eight bit synchronous counter which controls the address lines for the sequence memory. The WL is clocked by either a delayed active write during initialization or the inverted end of conversion from the converter during normal operation. The clear line of the eight bit counter is asynchronously cleared upon initialization by writing to address 10H for the 16 bit A/D counter clock or address 11H for the 12 bit A/D counter clock. The counter is cleared during normal operation when both the 4th bit in the sequence memory and the inverted end of conversion from the A/D converter are active (a logic one). The clearing of the counter is inhibited during the loading of the sequence memory. At the last sequence address the

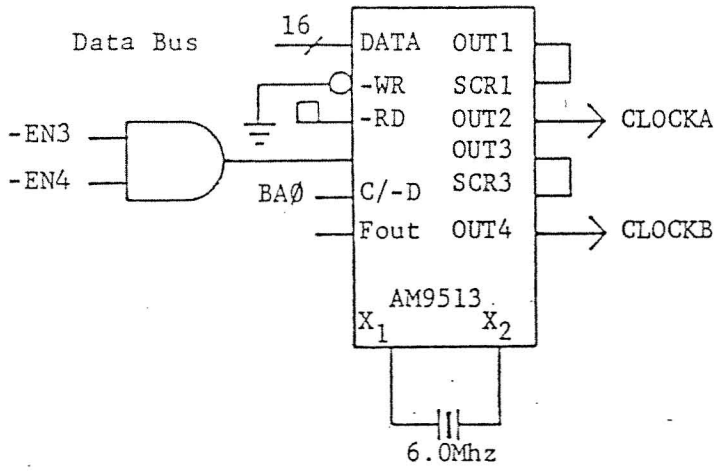

Figure 3.1
Programmable Clocks

4th bit is set for clearing the counter back to its initial count value. (See part five concerning initialization.)

The SM shown in Figure 3.2 is a volatile 256 by 4 memory space which is loaded as a single external address, 14H for the 16 bit A/D clock and 15H for the 12 bit A/D clock. The first three bits in the sequence memory represent the address of the analog input channel that will be addressed and the location in the dual port SRAM where the converted signal from that channel will be stored. The fourth bit in the SM is used to clear the WL at the final nibble.

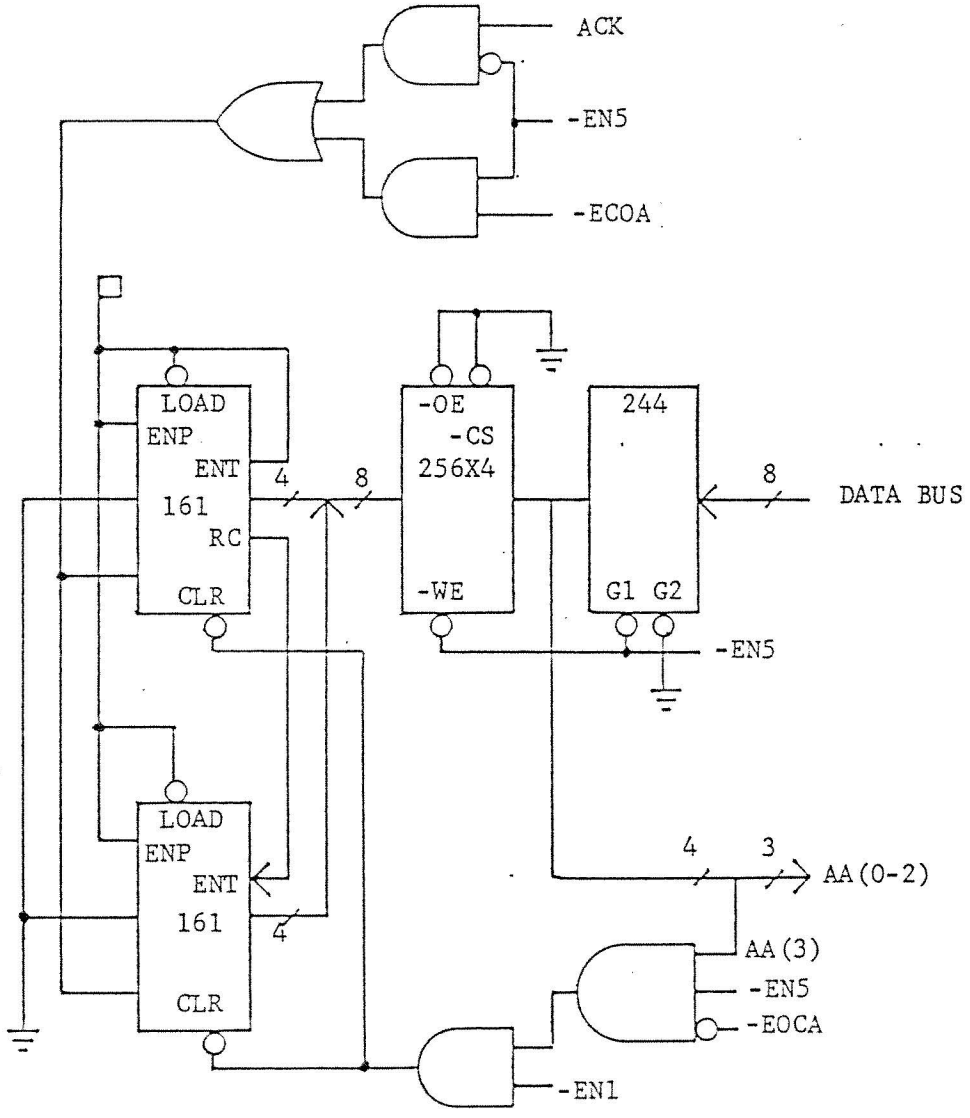

Figure 3.2
Window Logic and Sequence Memory

3.2 Nonlinear Gain Amplifier and Hysteresis Circuits

The NGA (nonlinear gain amplifier) shown in Figure 3.3 performs the function of extending the dynamic range of the SA 12 bit A/D converter. This is accomplished via the amplification of the absolute value of the input signal by a factor of eight when it is smaller in amplitude than the absolute value of the threshold voltage. A gain of eight was selected because an external processor can easily divide by eight without using floating point techniques. The thirteenth bit of the 12 bit converter word is set when the input signal has been amplified. This bit is supplied to the external processor for scaling. As the absolute value of the input signal exceeds the absolute value of the threshold voltage the NGA is forced to unity gain and the thirteenth bit of the 12 bit converter word is reset. Figure 3.4 shows the output response of the NGA section.

The hysteresis circuitry shown in Figure 3.3 was included within the NGA section to prevent oscillations around the absolute value of the threshold offset. The hysteresis circuit was designed to produce symmetrical hysteresis at the positive and negative threshold offsets. The absolute value maximum of the hysteresis (MH) must be less than or equal to the absolute value of the quotient of the full scale (FS) deflection and the gain (G) of the NGA or

$$| MH | < \frac{| FS |}{| G |}. \qquad (3.1)$$

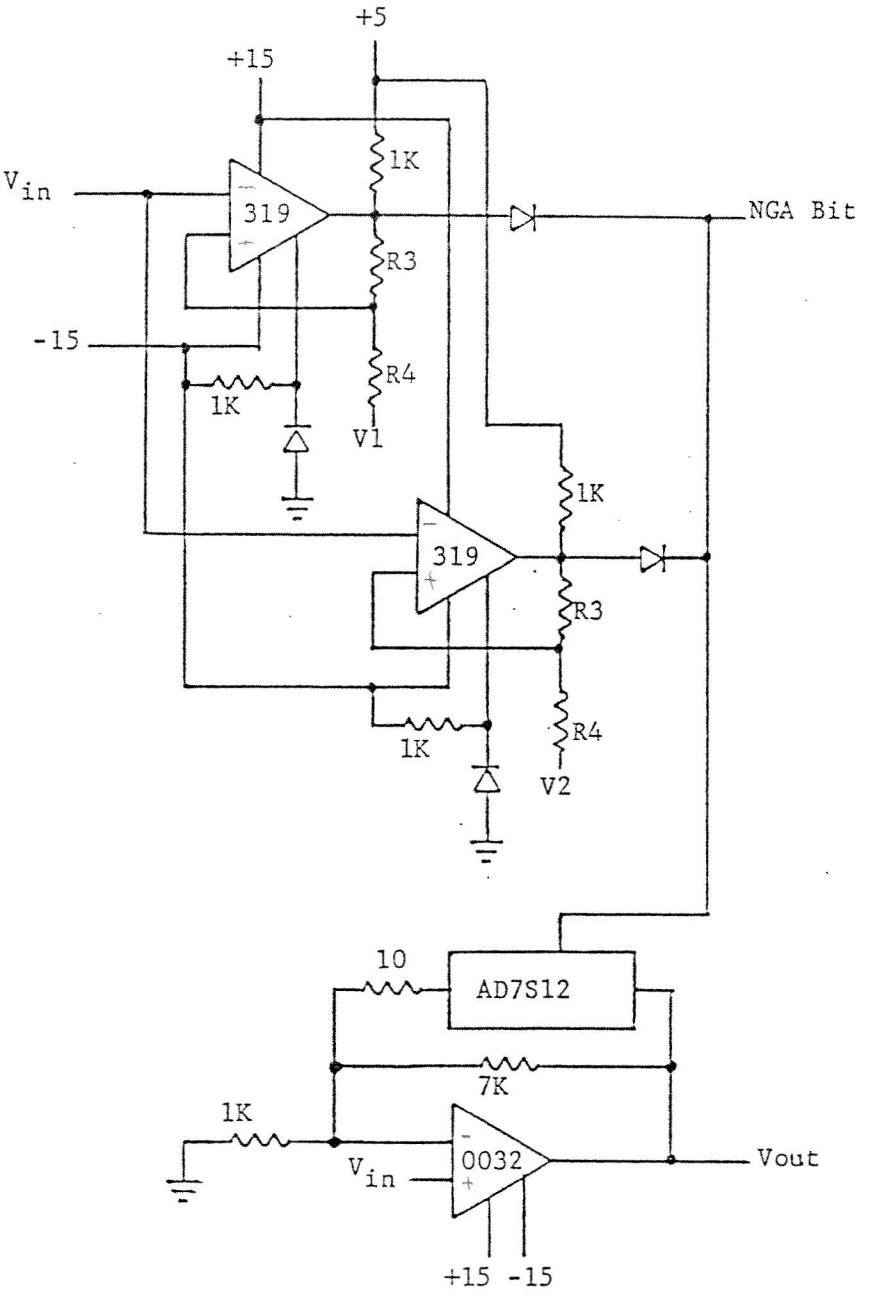

Figure 3.3
NGA and Hysteresis Circuit

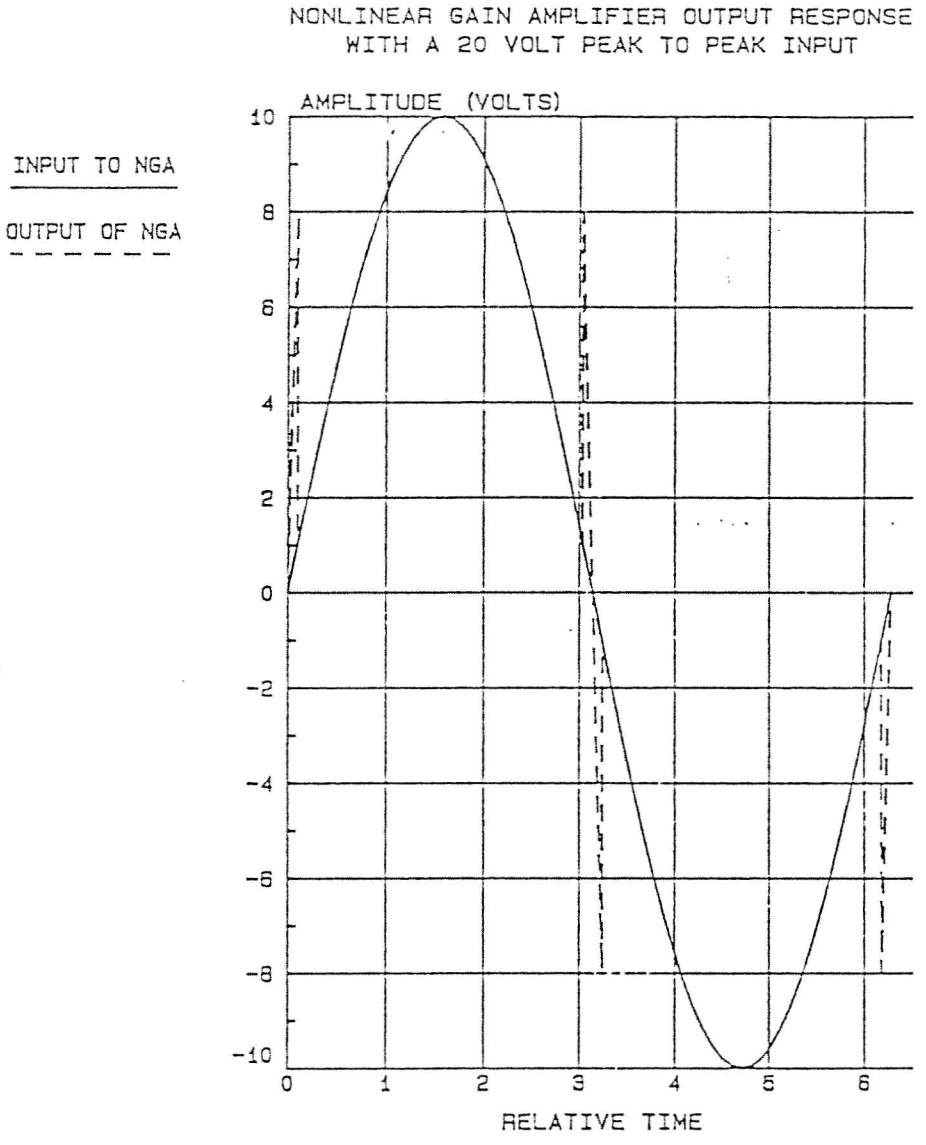

Figure 3.4
Output Response of NGA

The threshold or point where the gain changes is dependent upon the direction of change of the input signal. For design purposes a family of curves, shown in Figure 3.5, was generated so that the designer could select the amount of desired hysteresis versus the values of R3 and R4.

Suppose, 0.2 volts of hysteresis are needed with the threshold offers of plus and minus 0.8 volts. What are the values of R3, R4, V1 and V2 shown in Figure 3.3? The values of R3 and R4 are independent of the threshold offsets or

$$R3 = 24(R4). \tag{3.2}$$

From Figure 3.4 and equation 3.2 the values of R3 and R4 are

$$R3 = 4.8k\Omega \tag{3.3}$$

and

$$R4 = 200\Omega \tag{3.4}$$

Next, it is necessary to determine V1, the positive threshold offset. The value of V1 is equal to

$$V1 = \frac{0.7(R3 + R4)}{R3} \tag{3.5}$$

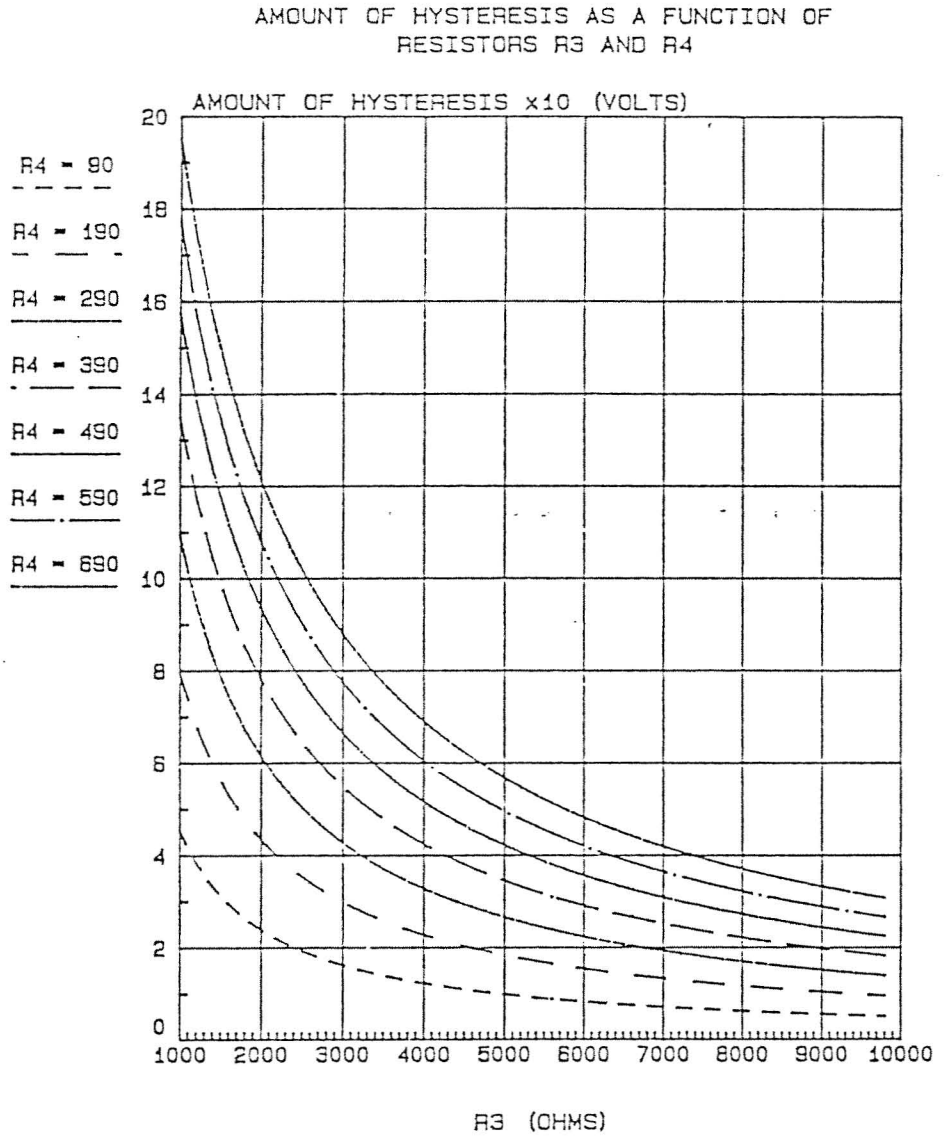

Figure 3.5
R3 and R4 Values Versus
the Amount of Hysteresis

or

$$V1 = 0.73 \text{ volts.} \tag{3.6}$$

The maximum positive threshold (MPT) is the minimum positive threshold plus the amount of hysteresis or in this case MPT is 0.9 volts. The value of the negative threshold offset is

$$V2 = \frac{-0.9(R3 + R4)}{R3} \tag{3.7}$$

or

$$V2 = -0.9375 \text{ volts} \tag{3.8}$$

and the least negative threshold is -0.7 volts. Thus, the hysteresis around the threshold offsets of plus and minus 0.8 volts are established as plus or minus 0.1 volts, respectively.

3.3 12 Bit A/D Converter Path

The 12 bit A/D converter path shown in Figure 3.6 is divided into six sections; the analog multiplexer, the nonlinear gain amplifier, the hysteresis circuitry, the sample and hold, the 12 bit A/D converter and the 8 by 16 dual port memory (SRAM). The timing for the 12 bit A/D converter path is controlled by its own programmable clock (CLOCKB). The sequencing of the channels and the addressing of the dual port SRAM is controlled by an independent sample timing controller.

The analog multiplexer selects one of eight channels based on the three bit code applied to its address inputs from the sequence memory.

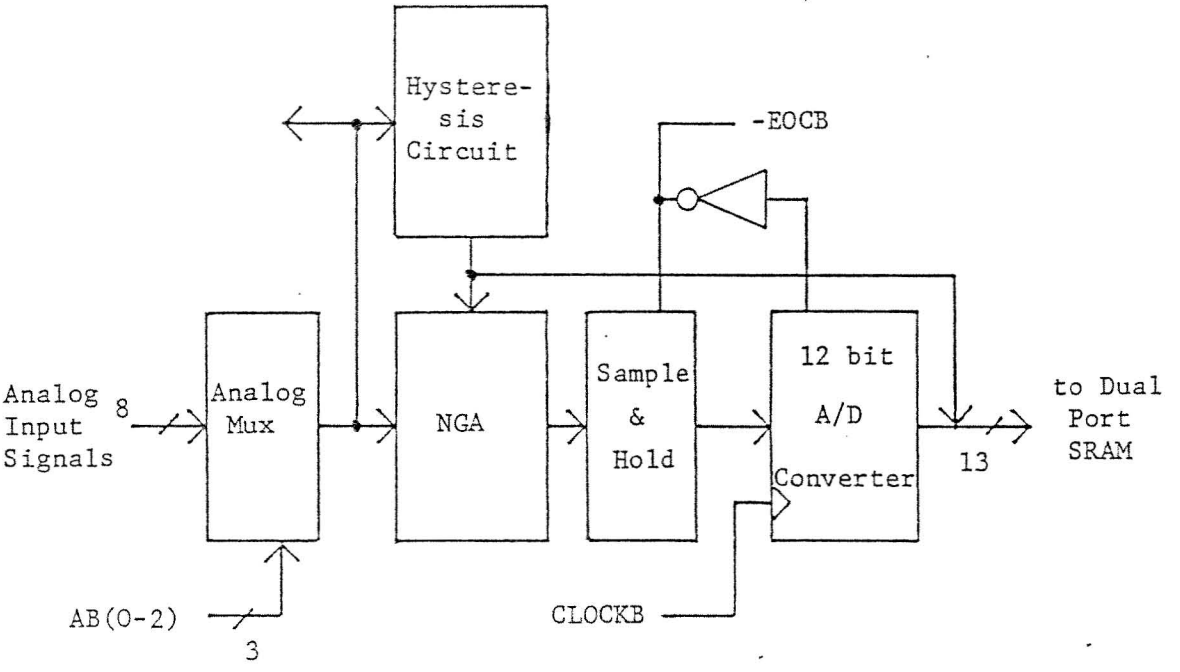

Figure 3.6
12 Bit A/D Converter Path

Channels one through seven are preconditioned bipolar 20 volt maximum peak to peak input signals whose total maximum rate of change must be less than 2.12 volts per microsecond or a through-put rate of 80KHz. Channel eight of the analog multiplexer is the buffered output of a 12 bit DAC which is used to test and debug the 12 bit A/D converter path.

The nonlinear gain amplifier provides gain control over the input signal based on its amplitude. For low signal amplitudes inside the threshold, the gain of the nonlinear amplifier is set equal to eight. In addition, the most significant bit of the 12 bit converter word, bit 13, is set. For high level inputs signals, outside the threshold, the gain is set equal to one. In addition, the most significant bit of the 12 bit converter word is reset. (See section 3.2 for more information.)

The hysteresis circuit provides control of the threshold so that noise around the threshold will not cause the nonlinear gain amplifier to change gains due to noise. (See section 3.2 for more information.)

The sample and hold is required to effectively reduce the aperture or conversion time of the successive approximation 12 bit A/D converter. An offset adjust is provided so that the reference level of the output of the sample and hold can be adjusted. The control line input of the sample and hold is controlled directly by the inverted end of conversion or status signal from the 12 bit A/D converter.

The 12 bit A/D converter is a high speed successive approximation converter that generates a 12 bit offset binary word that represents the signal applied at its input. Adjusts for the bipolar offset and reference are supplied for fine tuning the converter. The conversion stars on the falling edge of CLOCKB and continues until completed. On

the rising edge of -ECOB the converted data is stored in the 8 by 16 SRAM at the address defined by the sequence memory.

The 8 by 16 dual port SRAM provides the temporary storage for the 13 bits of binary information generated by the 12 bit A/D converter and the hysteresis circuit. This data is clocked into the dual port memory on the rising edge of -ECOB. The address lines of the dual port SRAM are the outputs of the sequence memory. (See Section 7.7 for more information.)

3.4 16 Bit A/D Converter Path

The 16 bit A/D converter path shown in Figure 3.7 is divided into five sections; the analog multiplexer, the high speed buffer, the sample and hold, the 16 bit A/D converter and the 8 by 16 dual port memory (SRAM). The timing for the 16 bit A/D converter path is controlled by its own programmable clock (CLOCKA). The sequencing of the channels and the addressing of the dual port SRAM is controlled by an independent sample timing controller.

The analog multiplexer selects one of eight channels based on the three bit code applied to its address inputs from the sequence memory. Channels one through seven are preconditioned bipolar 20 volt maximum peak to peak input signals whose total maximum rate of change must be less than 0.95 volts per microsecond or a through-put rate of 47KHz. Channel eight of the analog multiplexer is the buffered output of a 12 bit DAC which is used to test and debug the 16 bit A/D converter path.

The high speed buffer is used to match the input impedance of the sample and hold with the output impedance of the analog multiplexer.

The sample and hold is required to effectively reduce the aperature

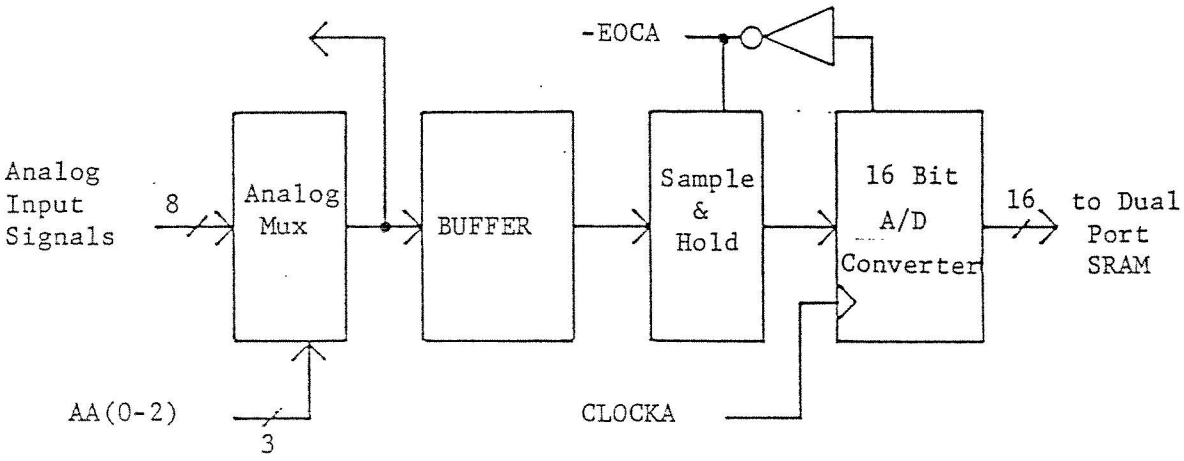

Figure 3.7
16 Bit A/D Converter Path

or conversion time of the successive approximation 16 bit A/D converter. An offset adjust is provided so that the reference level of the output of the sample and hold can be adjusted. The control line input of the sample and hold is controlled directly by the inverted end of conversion or status signal from the 16 bit A/D converter.

The 16 bit A/D converter is a high speed successive approximation converter that generates a 16 bit complemented offset binary word that represents the signal applied at its input. Adjust for the bipolar offset and reference are supplied for fine tuning the converter. The conversion starts on the falling edge of CLOCKA and continues until completed. On the rising edge of -ECOA the converted data is stored in the 8 by 16 SRAM at the address defined by the sequence memory.

The 8 by 16 dual port SRAM provides the temporary storage for the 16 bits of binary information generated by the converter. This data is clocked into the memory on the rising edge of -ECOA. The address lines of the dual port SRAM are the outputs of the sequence memory. (See section 7.7 for more information.)

3.5 MULTIBUS Interface Logic

There are two fundamental requirements placed upon modules interfacing to Intel's MULTIBUS. First, all address and data lines have been inverted in this bus scheme. Second, the module must generate a transfer acknowledge signal (-XACK) for the master processor.

To compensate for the inverted data and address bus all address lines are inverted on the DMA/DCM via three 74LS240's. All data lines are inverted in software via the development system. The control lines

are not inverted but rather are buffered through a single 74LS244.

The control signal -XACK is generated by clocking two flip flops with the asynchronous clock -BCLK. When either -MWTC or -MRDC, memory commands, goes inactive the two flip flops are asynchronously cleared. The clearing of the flip flops forces -XACK inactive. Figure 3.8 shows the -XACK generating circuit and Figure 3.9 shows the associated timing of that circuit. It should be noted that the -XACK goes active from 100 to 200 nanoseconds after an active read or write to the module. Then the -XACK signal will go inactive within 65 nanoseconds after the active read or write goes inactive.

3.6 16 by 16 Dual Port SRAM with Write Lockout

The 16 by 16 dual port memory is divided into two functionally identical 8 by 16 dual port memory sections. The lower 8 by 16 memory shown in Figure 3.10 is used to store the 16 bit output of the 16 bit A/D converter. The eight analog input signals connected to the analog multiplexer of the 16 bit path each map into its own 16 bit memory location. The upper 8 by 16 memory shown in Figure 3.11 is used to store the 12 bit output code of the 12 bit A/D converter and the gain control bit of the nonlinear gain amplifier. Likewise, the eight analog input channels connected to the analog multiplexer of the 12 bit path each map into its own 16 bit memory location.

The write lockout signals, -INHA for the 16 bit A/D converter path and -INHB for the 12 bit A/D converter path, are generated to inhibit the writing of new data to a memory location that is being addressed and read by the external processor. The lockout signals become active

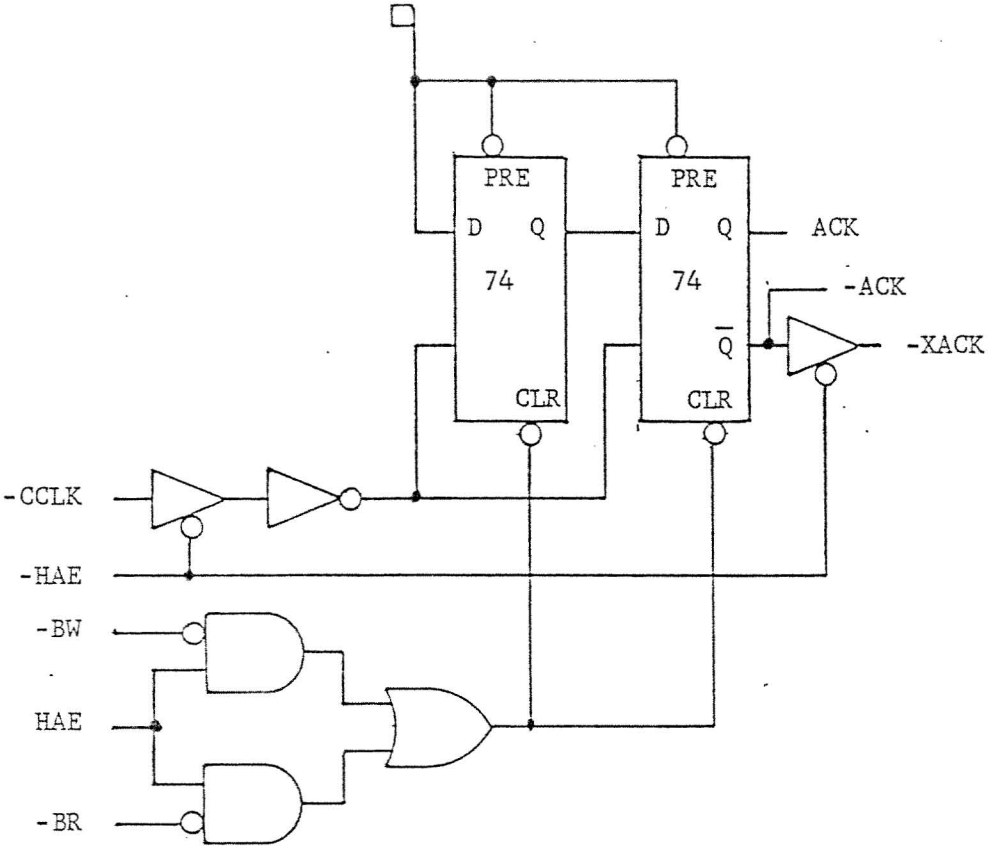

Figure 3.8
-XACK Generation Circuit

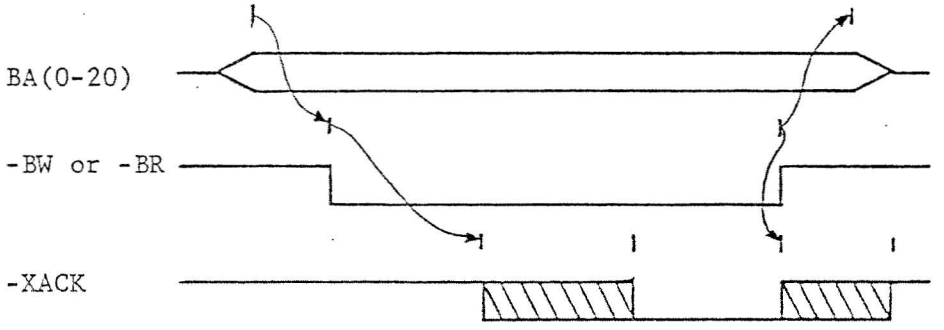

Figure 3.9
Timing For -XACK Generation

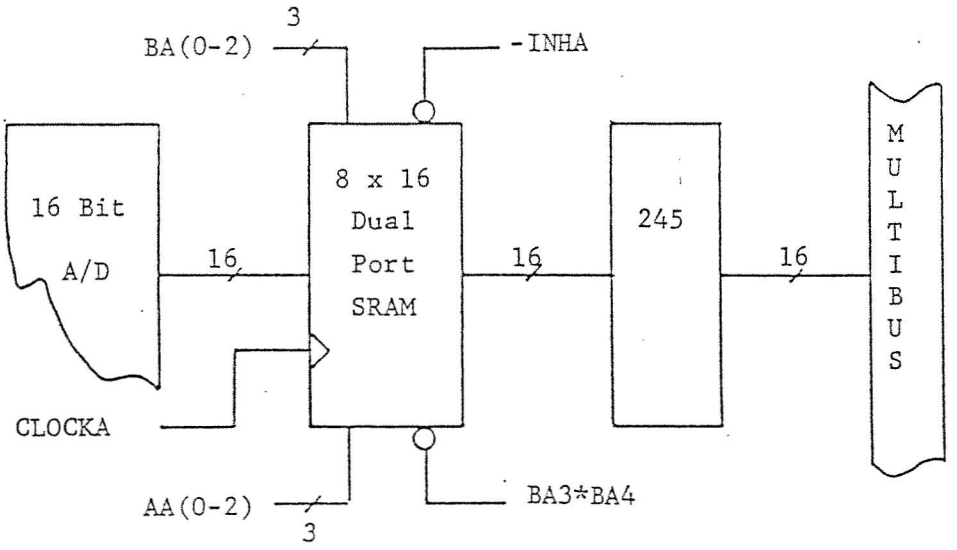

Figure 3.10
16 Bit A/D Converter
8 by 16 Dual Port SRAM

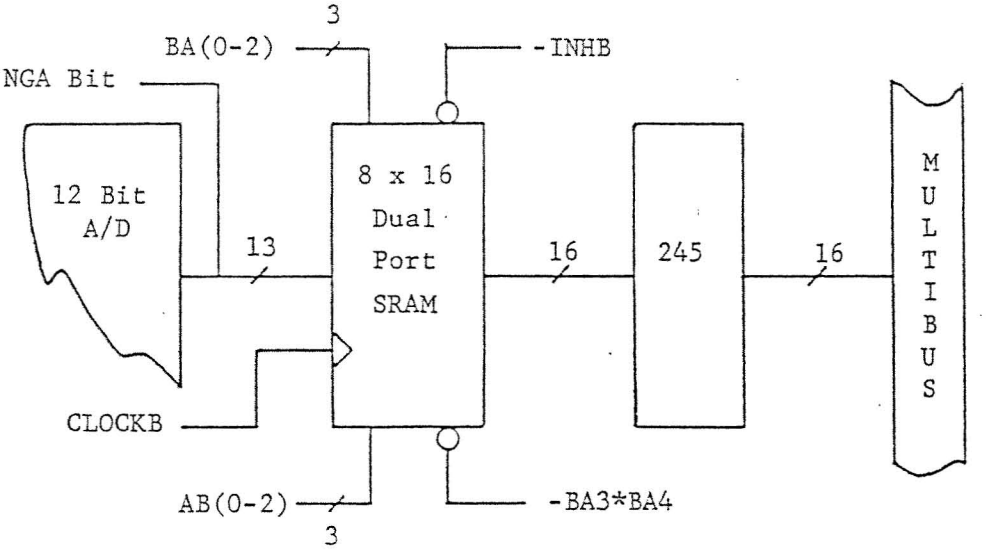

Figure 3.11
12 Bit A/D Converter
8 by 16 Dual Port SRAM

from 100 to 200 nanoseconds after an active read to the selected module occurs. If the address lines on from the sequence memory equals the address lines from the external processor then the writing to the dual port SRAM at that address is inhibited during the entire read cycle. Any new information generated for that memory location is lost. However, the most recently converted data for that address should exist at that address due to the 100 to 200 nanosecond delay. The signals, -INHA and -INHB, are the exclusive oring of the external inverted address lines with the sequence memory outputs (dual port memory address lines) ored with the read, module select and transfer acknowledge signals. Thus, the Booleans are:

$$-INHA = BA0 + -AA0 + -BA0 + AA0 + BA1 + -AA1 + -BA1 + AA1 + BA2 + -AA2 + -BA2 + AA2 + BA3 + -BR + -HAE + -XACK \quad (3.9)$$

and

$$-INHB = BA0 + -AB0 + -BA0 + AB0 + BA1 + -AB1 + -BA1 + AB1 + BA2 + -AB2 + -BA2 + AB2 + -BA3 + -BR + -HAE + -XACK. \quad (3.10)$$

These Booleans are generated by the PAL16H2 and a 74LS32 ICs. (See section 7.4 for more details.)

3.7 Decoders, Buffers and Programmable Clocks

All address and control lines that are applied to the module from the MULTIBUS are buffered. The address lines -ADR(0-23) are inverted

via several 74LS240 buffers to form the module address lines BA0 to BA23. The five least significant address lines, BA0 to BA4, are used on the module to address the thirty-two read/write locations. The three most significant address lines are inverted but not used. Address -ADR(20) is inverted, tied high and applied to the PAL20C1 decoder IC along with the remaining address lines, BA(5-19). The PAL20C1 generate an active module select signals, -HAE and HAE, when the MULTIBUS address is between 90000H and 9001FH.

The control lines -MRDC, -MWTC and -CCLK are buffered but not inverted through a 74LS244 buffer to form -BR, -BW and -BCLK, respectively. The control signal -XACK is generated by the module and applied to the same 74LS244. One of the enable lines for this buffer is controlled by the module select -HAE. When the module is selected the control lines will be enabled.

The -XACK signal generated by the module provides a 100 to 200 nanosecond delay from the leading edge of an active read or write signal. The -XACK goes inactive synchronous with an inactive going read or write signal. (See section 3.5 for more information.)

The data from the MULTIBUS is buffered on the module via two 74LS245 transceivers. The outputs of the transceivers are enabled when the module has been selected. The direction of data flow through the transceiver is determined via the anding of the module select, HAE, and the read line (-BR).

There are only eight decoded write locations on the module. A single 74LS138 three to eight decoder is used to generate the eight write enables, -EN1 to -EN8, for the module.

There are two programmable clocks on the module that are generated via the programmable AM9513 timing controller IC. Two of the five 16 bit counters of the controller are cascaded together to produce a single thirty-two bit counter. The clocks CLOCKA and CLOCKB are produced by dividing the 6.0Mhz base clock by some integer value placed in the load registers in the AM9513. Figure 17 shows the available clock frequencies based on the integer division of the 6.0Mhz clock.

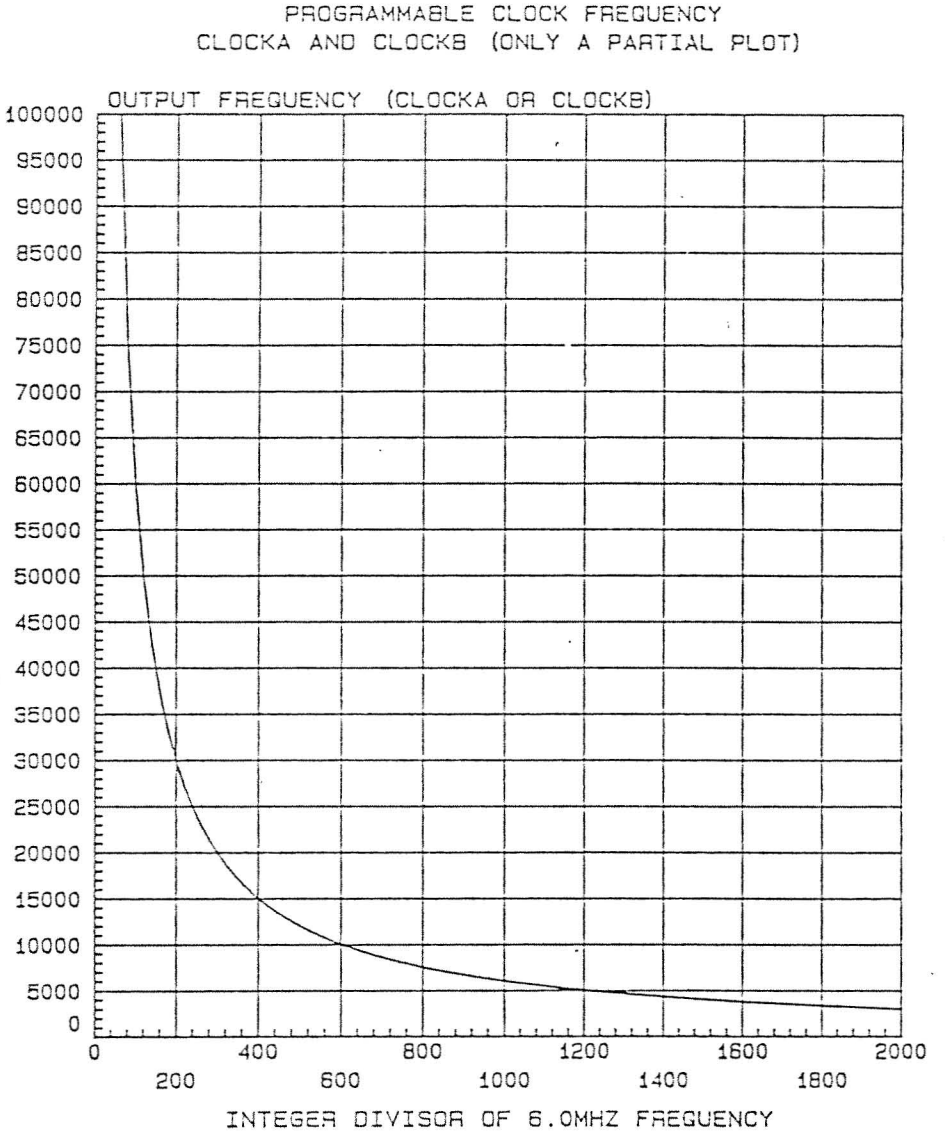

Figure 3.12
Available Clock Frequencies From AM9513

PART IV

4.0 DEDICATED MULTICHANNEL A/D CONVERTER MODULE SOFTWARE

Part four deals with the software need to initialize the module and the memory map of the module. Section one describes the 32 read/write location on the module. This section will explain how the module is partitioned within the total memory. Section two describes how the two programmable clocks are programmed. The final section, section three, explains the support software that is used to determine how the sequence memory should be loaded to produce ten uniform sample slots for each sampled signal.

4.1 Memory Map

The read (-BR) and write (-BW) commands are mutually exclusive. That is, only one command can be active at a time. Table 4.1 shows how the module is mapped into its 32 memory locations. It should be noted that certain address locations cannot be written to or read from and other locations are undefined. However, half of the read locations are defined as the outputs of the dual port SRAMs. These locations contain the A/D converter information for each channel. No other location on the module is readable. The write locations deal with the initialization of the module. At address 10H and 11H on the module the sample timing controllers may be cleared to their initial values by writing anything to these locations. Addresses 12H and 13H are used to program the programmable clocks. The remaining two write locations are 14H and 15H on the module. These address locations are the currently addressed sequence memory locations. Of the 32 read/write location only 50% of the read locations and 18.75% of the write locations are used. Thus, if extra decoding logic is used the unused memory locations could be accessed.

TABLE 4.1
MEMORY MAP OF MODULE

ADDRESS	BA4	BA3	BA2	BA1	BAO	/BW	/BR	CHANNEL DEFINITION
90000H	0	0	0	0	0	1	0	Channel #1 of 16 bit converter
90001H	0	0	0	0	1	1	0	Channel #2 of 16 bit converter
90002H	0	0	0	1	0	1	0	Channel #3 of 16 bit converter
90003H	0	0	0	1	1	1	0	Channel #4 of 16 bit converter
90004H	0	0	1	0	0	1	0	Channel #5 of 16 bit converter
90005H	0	0	1	0	1	1	0	Channel #6 of 16 bit converter
90006H	0	0	1	1	0	1	0	Channel #7 of 16 bit converter
90007H	0	0	1	1	1	1	0	Channel #8 of 16 bit converter
90008H	0	1	0	0	0	1	0	Channel #1 of 12 bit converter
90009H	0	1	0	0	1	1	0	Channel #2 of 12 bit converter
9000AH	0	1	0	1	0	1	0	Channel #3 of 12 bit converter
9000BH	0	1	0	1	1	1	0	Channel #4 of 12 bit converter
9000CH	0	1	1	0	0	1	0	Channel #5 of 12 bit converter
9000DH	0	1	1	0	1	1	0	Channel #6 of 12 bit converter
9000EH	0	1	1	1	0	1	0	Channel #7 of 12 bit converter
9000FH	0	1	1	1	1	1	0	Channel #8 of 12 bit converter
90010H	1	0	0	0	0	0	1	16 bit window width
90011H	1	0	0	0	1	0	1	12 bit window width
90012H	1	0	0	1	0	0	1	Data location for conv. clock
90013H	1	0	0	1	1	0	1	Com. location for conv. clock
90014H	1	0	1	0	0	0	1	Load 16 bit sequ. memory
90015H	1	0	1	0	1	0	1	Load 12 bit sequ. memory

The remaining address locations are spares, (90017H to 9001FH).

4.2 Programmable Clocks

Both CLOCKA and CLOCKB are programmable clocks that are produced by the AM9513[25]. This device is programmed as two independent 32 bit up counter with an active high terminal count. A negative two's complement 32 bit value is loaded into the load registers and up counted by the counter registers to an active high terminal count. Then the value that was placed in the load registers is used to reinitialize the counter registers and the up count continues in the same manner.

For example, suppose it is desired to program the AM9513 to produce a 50KHz clock. Then it is necessary to divide the 6.0MHz clock by 120. To do this the load registers must be load with the value FFFFFF88H. The difference between the terminal count and the loaded initial values when counting up is 120. Thus, the 6.0MHz clock will be divided by 120.

The AM9513 contains a command and date register. The command register is located at 13H and is written to when -BW and -HAE are active. Likewise, the data registers located at address 12H is written to in the same manner.

To program the AM9513 as two 32 bit counters it is first necessary to write to the command register to select the master mode register. This is accomplished when -HAE and -BW are active and the word FF17H is written to address 13H of the module. Next, the word 21B0H is written to the master mode register. This command disables the compares and time of day modes and switches the AM9513 to the 16 bit wide data bus inputs. The Fout is turned on but not used on the

module. This master mode command also will force the internal data pointer to increment through the element cycle. The first step in the element cycle is to write FF01H at address 13H on the module. This command selects the mode #1 register, to count the 6.0MHz clock on its rising edge. In addition, all gating control will be turned off. The counter will repetitively count up to an active high terminal count. The next step in the element cycle is to write the least significant word (LSW) into the load #1 register at address 12H. The value of the LSW is expressed in negative two's complement notation. The next register to be addressed is the hold #1 register. This register is not used and FFFFH is loaded into it. The second counter or the counter #2 mode register is loaded with the command 0129H. This forces the counter #2 register to count on the rising edge of SRC1 or the terminal count of counter #1. In addition, all gating control will be turned off for counter #2. The counter will repetitively count up to an active high terminal count. Next, at address 12H on the module the most significant word (MSW) is loaded into the load #2 register. The value of the MSW is expressed in negative two's complement notation. The hold #2 register is addressed next. This register is not used and FFFFH is loaded into it. The LSW in counter #1 and the MSW in counter #2 form the 32 bit counter that generates CLOCKA for the 16 bit A/D converter path. At address 12H on the module the word 0B29H is written for programming the counter #3 mode register. This command will force counter #3 to count the 6.0MHz clock on its rising edge. In addition, all gating control will be turned off for this counter. The counter will repetitively count up to an active high terminal count. Next, at

address 12H on the module the least significant word (LSW) is loaded into the load #3 register. The value of the LSW is expressed in negative two's complement notation. The load #3 register is addressed next. This register is not used and FFFFH is loaded into it. The counter #4 mode register is loaded next with the command 0329H. This forces the counter #4 register to count on the rising edge SRC3 or the terminal count of counter #3. In addition, all gating control will be turned off. The counter will repetitively count up to an active high terminal count. Next, at address 12H on the module the most significant word (MSW) is loaded into the load #4 register. The value of the MSW is expressed in negative two's complement notation. The next register to be addressed is the hold #4 register. This register is not used and FFFFH is loaded into it. The LSW in counter #3 and the MSW in counter #4 form the 32 bit counter that generates CLOCKB for the 12 bit A/D converter path. The remaining three registers are not used and are programmed as follows. First, at address 12H on the module the command word 0B00H is written to the counter #5 mode register. This command forces the output low and disables all gating. This command is a dummy commmand for the fifth counter. Next. the load #5 register is loaded with the value FFFFH. The last register to be loaded is the hold #5 register. This register is also loaded with FFFFH. The master mode register has incremented through the element cycle Finally, the Arm and Load command FF6FH is written to address 12H to start CLOCKA and CLOCKB. To disable the clocks simply write FF8FH to address 12H. Table 4.2 shows what has been explained here in an abbreviated form.

TABLE 4.2

PROGRAM SEQUENCE FOR AM 9513

DATA	C/-D	-HAE	-BW	-BR	COMMAND DESCRIPTION
FF17H	1	0	0	1	Write to command register to select MM register.
21B0H	0	0	0	1	Write to MM register.
FF01H	1	0	0	1	Write to command register to select mode register #1.
0B29H	0	0	0	1	Set up counter #1 to count clock.
XXXXH	0	0	0	1	Load LSW of count value for 16 bit converter clock.
FFFFH	0	0	0	1	Load hold #1 register (register not used).
0129H	0	0	0	1	Set up counter #2 to count counter #1 TC.
XXXXH	0	0	0	1	Load MSW of count value for 16 bit converter clock.
FFFFH	0	0	0	1	Load hold #2 register (register not used).
0B29H	0	0	0	1	Set up counter #3 to count clock.
XXXXH	0	0	0	1	Load LSW of count value for 12 bit converter clock.
FFFFH	0	0	0	1	Load hold #3 register (register not used).
0329H	0	0	0	1	Set up counter #4 to count counter #3 TC.

(Continued - TABLE 4.2)

DATA	C/-D	-HAE	-BW	-BR	COMMAND DESCRIPTION
XXXXH	0	0	0	1	Load MSW of count value for 12 bit converter clock.
FFFFH	0	0	0	1	Load hold #4 register (register not used).
OBOOH	0	0	0	1	Set up counter #5 (not used).
FFFFH	0	0	0	1	Set up counter #5 not to count (register not used).
FFFFH	0	0	0	1	Load hold #5 register (register not used).
FF6FH	0	0	0	1	Arm and load counters.
FF8FH	1	0	0	1	Disarm and save counter.

4.3 Sampling Algorithm Software

A sampling sequence must be established which is of minimal length and cyclic over a period that is equal to one tenth of the period of the slowest signal of interest. The main objective is to store the least amount of data in the sequence memory and to uniformly sample the input signals. For digital control applications all signals should be sampled at least ten times within their respective periods. Since the input signals are multiplexed they must be meshed and uniformly spaced in time.

A sampling algorithm was developed so that the sequence memory can be loaded with from 1 to 256 addresses. These addresses will be used to control the address lines of the dual port SRAMs and the analog input multiplexer. There are only 256 locations in the sequence memory.

To illustrate this algorithm, suppose four input signals

$$f1 = 2KHz \tag{4.1}$$

$$f2 = 1KHz \tag{4.2}$$

$$f3 = 500Hz \tag{4.3}$$

and

$$f4 = 50Hz \tag{4.4}$$

are to be sampled. The minimal sampling frequency (Fm) is

$$Fm = 10\sum_{n=1}^{4} fn \tag{4.5}$$

or

$$Fm = 35{,}500 \text{ samples/second.} \tag{4.6}$$

This translates into a sampling period of 18.169 microseconds per sample. The meshing of the samples due to multiplexing of the four channels would require nonuniform sampling as shown in Figure 4.1 of the input signals. Thus, for this example, the uniform sampling frequency (Fu) should be

$$Fu = 20f1 \tag{4.7}$$

or

$$Fu = 40{,}000 \text{ samples/second.} \tag{4.8}$$

This translates into a sampling period of 25.0 microseconds per sample.

Since forty periods of f1, twenty periods of f2 and ten periods of f3 exist in one period of f4 it is possible to express the sequenced output of the analog input multiplexer (S) as

$$S = f1\sum_{n=o}^{399} \delta(t1-50n) + f2\sum_{n=o}^{199} \delta(t2-100n) + f3\sum_{n=o}^{99} \delta(t3-200n) + f4\sum_{n=o}^{9} \delta(t4-2000n). \tag{4.9}$$

The variable t1, t2, t3, and t4 are four time delays associated with

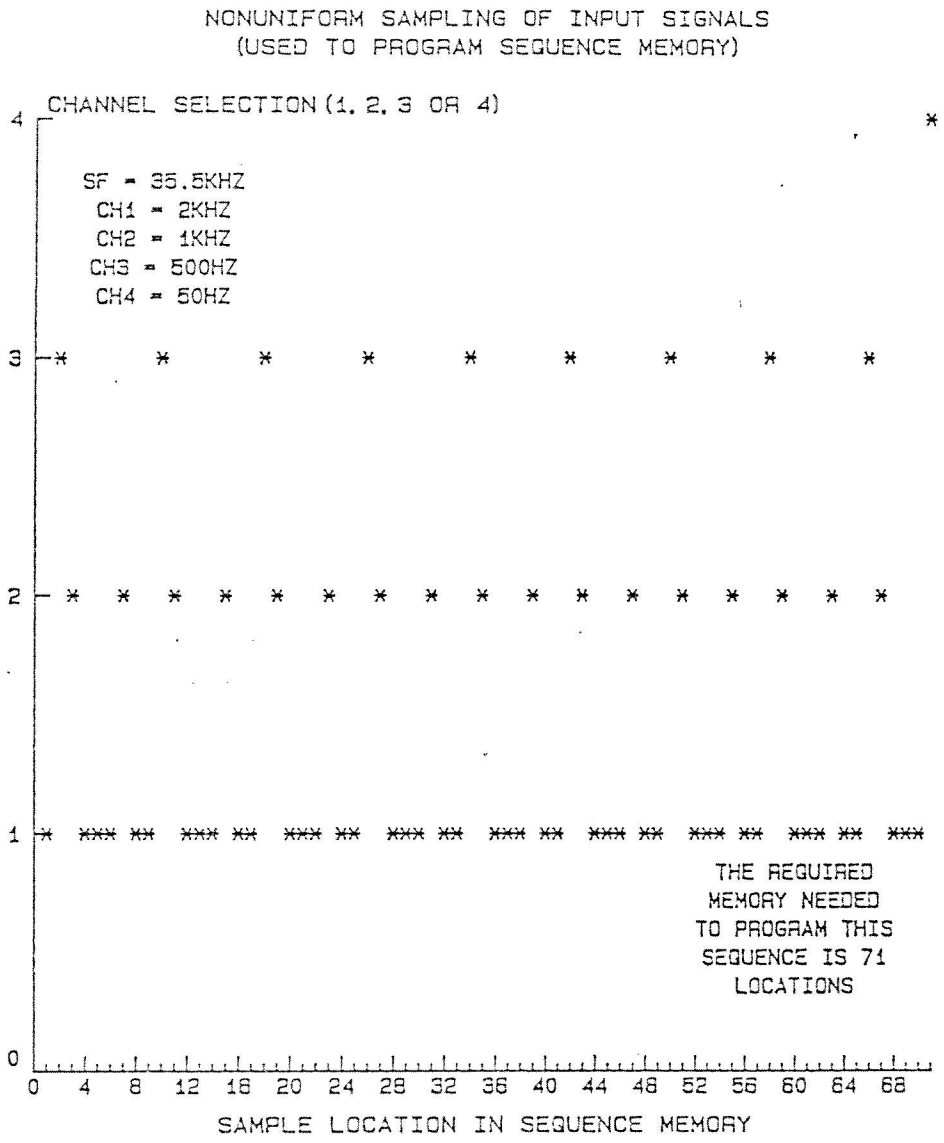

Figure 4.1
Nonuniform Sampling of Input Signals

the offsets of the input signals. There are 710 samples within one period of the slowest rate of change signal or 71 samples in one tenth of that period. However, the ratio between the sampling frequence and the slowest rate of range signal is 800 samples or 80 samples in one tenth of that period. Thus, there are 9 extra sample slots that could be used for sampling a test. Care must be taken so that the sum of the ratios between the minimum rate of change signal with the other signals does not exceed 256. The meshing of the four uniformly sampled input signals is shown in Figure 4.2.

The algorithm, used in program /MASTER/SAMPLE, performs five basic functions. First, the periods of the input signals to be sampled are determined. The lowest frequency signal established the sampling period at one tenth of its period. Second, the highest frequency signal is slotted. If all sample slots are used for sampling the highest frequency signal, then nonuniform sampling will result. Nonuniform sampling results in slotting slower frequencies over the higher frequencies. If all sample slots are not used or uniform sampling results, then the next highest frequency is slotted until all samples are slotted. Finally, spare slots can be filled and the length of the sequence is established and addresses stored in sequence memory.

There are three fundamental items to keep in mind when using this algorithm. First, determine the maximum rate of change of the input signals. Ideally the maximum rates should be minimum maximums because extra bandwidth may limit the sampling of the other signals. Second, try to make the ratio between the sampling frequency and the maximum rate of change for each signal an integer ratio. This will aid in

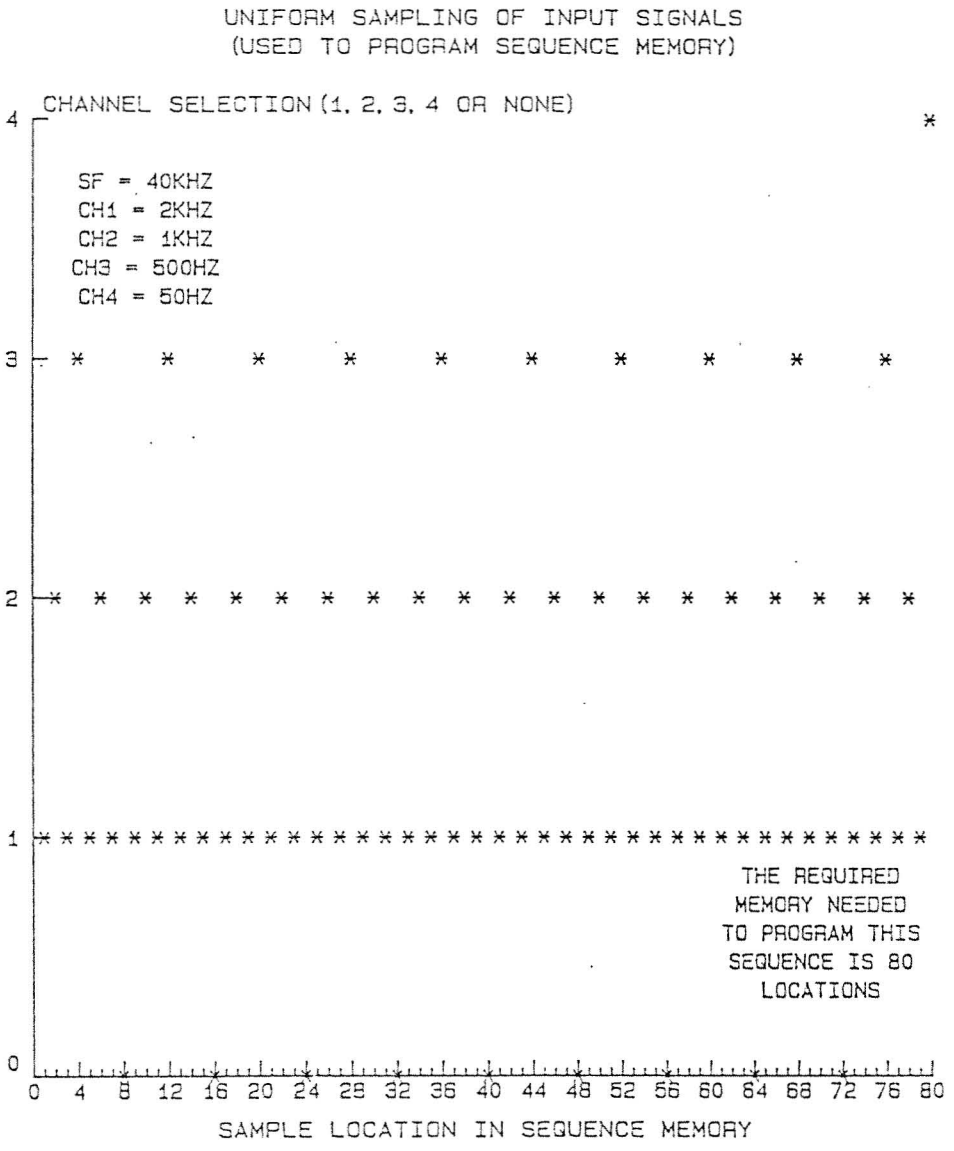

Figure 4.2
Uniform Sampling of Input Signals

uniform sampling. Next, make sure that the sum of the ratios between the slowest rate of change signal with the other signals is less than 256. This will insure that all addresses will fit within the 256 address locations in the sequence memory.

PART V

5. DISCUSSION AND CONCLUSION

The module is initiated on a write to address 90010H, the 16 bit window width location. A write to this location clears the WL and forces the 8 bit counter to its initial count. Next, at address 90014H a write command is issued whose four least significant data bits contain the sequence address and WL clear bit. The sequence memory is loaded one nibble at a time until the entire sequence is loaded into address 90014H. Each time a write is issued to this address the sequence memories address lines are incremented to the next address by the window logic. At the last sequence address the 4th bit is set and the window logic is cleared to its initial count.

The sequence memory and window logic of the 12 bit A/D converter path are programmed in the same manner except its window logic is at 90011H and its sequence memory is at address 90015H.

The two programmable clocks, CLOCKA and CLOCKB, are programmed next to produce two clock frequencies which are integer divisions of the 6.0 MHz clock.

The external process then generates the ARM and LOAD command for the programmable clocks which will start the modules independent operation. The clocks are disabled by issuing a Disarm and Save command.

The "free run" of the module allows for two independent conversion paths to occur at the same time. The sequence memory addresses the channel and dual port memory locations synchronously with the

programmable clocks. Conversion begins on the following edge of the clock and continues until completed. At which time the converter issues an end of conversion command. This command is inverted and fed to the sample and hold which takes another sample. The command is also anded with the clear bit in the sequence memory. If both bits are set the window logic is cleared and the sequence memory is forced back to its initial address. On the rising edge of the inverted end of conversion signal, converted data is latched into the dual port SRAMs before the address is allowed to change. The "free run" continues until halted by the external processor.

The accessing of data from the external process through the MULTIBUS occurs in an asynchronous fashion. (The 6.0MHz clock is not synchronized to the system clock.) The external processor issues a read command to one of the sixteen read locations on the module. The locations represent the most current conversion data placed in that dual port memory location. At the addressed location the data will be frozen for the entire read command. However, all other locations will continue to be loaded by the onboard hardware with conversion data.

There are several points of interest that should be addressed now to conclude this DMA/DCM. First, why not use an onboard microprocessor to control the sampling and interfacing? This raises a very good point because some analog I/O modules[20] do just that. This would allow the sampling software to be placed on the module. However, an onboard microprocessor would be taxed if it were to perform all at the functions currently performed by the sequence memory, window logic and programmable clock. The microprocessor would also need volatile and

nonvolatile memory. Thus, flexibility could be gained if the microprocessor could perform all of its required tasks in time.

The next major point of interest is interfacing. There are numerous ways the external processor can be notified that a conversion process has been completed. The DMA/DCM ignores the external process during the free run except when the processor is accessing data. No flagging is issued and the external system only knows that the data on the dual port SRAMs is the newest. (A maximum of one tenth of period of that channels highest frequency of interest old.) Flagging could possibly provide quicker accessing.

Signal preconditioning has not been handled at all on the DMA/DCM except to say that the signals have been preconditioned. This would require the input signals to be amplified and filtered to get the maximum use out of the A/D converters. Some analog I/O modules allow the user to define the amount of amplification via programming a PGA. This amplifier could be programmed via software or hardware.

Finally, it should be noted that the best design doesn't always require the costliest parts, or the fastest, or most accurate. In fact, using such parts may lead to degraded performance. The best design is the design that meets the required specifications with the least cost, complications and ease of implementation.

PART VI

REFERENCE BOOKS

1.) "Data Acquisition and Conversion Handbook" by Eugene L. Zuch, published by Datel/Intersil, 1979

2.) "Theory of Sampled-Data Control Systems" by David P. Lindorff, published by John Wiley & Sons, Inc., 1965

3.) "Sampled-Data Control Systems" by John R. Ragazzini and Gene F. Franklin, published by McGraw-Hill, 1958

4.) "Digital Control of Dynamic Systems" by Gene F. Franklin and J. David Powell, published by Addison-Wesley, 1980

5.) "Digital & Sampled-Data Control Systems" by Julius T. Tou, published by McGraw-Hill, 1959

6.) "Modern Control Theory & Applications" by Stanley Shinner, published by Addison-Wesley, 1978

7.) "Information, Transmission, Modulation and Noise" by Mischa Schwartz, published by McGraw-Hill, 1980

8.) "Analog Systems for Microprocessors and Minicomputers" by Patrick H. Garrett, published by Reston Publishing Company, Inc., 1978

9.) "Introduction to Operational Amplifier Theory and Application" by John V. Wait, Lawrence P. Huelsman and Granino A. Korn, published by McGraw Hill, 1975

10.) "OEM Systems Handbook" by Intel, published by Intel Corporation, 1984

11.) "PAL Handbook", Third Edition, by Monolithic Memories, published by Monolithic Memories, Inc., 1983

12.) "Linear Databook" by National Semiconductor Corporation, published by National Semiconductor Corporation, 1982

13.) "Data-Acquisition Databook 1984, Volume 1 Integrated Circuits" by Analog Devices, published by Analog Devices, Inc., 1984

14.) "MOS Microprocessor and Peripherals Data Book" by Advance Micro Devices, published by Advanced Micro Devices, Inc. 1983

15.) "Hitachi IC Memories Data Book" by Hitachi,
published by Hitachi, #M11

16.) "IC Master", Volume II,
published by Schweber Electronics, 1983

17.) "Product Data Book",
published by Burr-Brown, Inc., July 1984

18.) "Memory Databook",
published by National Semiconductor, Inc., 1980

19.) "The TTL Data Book For Design Engineers",
published by Texas Instruments, Inc., 1981

20.) "16/32-Bit Microcomputer System Components", M68RAD1 Module,
published by Motorola, Inc., 1984

21.) "Instruments and Systems Handbook", ST-6832 Module,
published by Datel/Intersil, 1980

22.) "New Data Update8", CIM-411 Module,
published by National Semiconductor, Corp.., Feb.. 1983

23.) "Module Subsystems", RPI-00 Module,
published by Analog Devices, Inc., 1984

24.) "Module Subsystems", RPI-230 Module,
published by Analog Devices, Inc., 1984

25.) "MOS Microprocessors and Peripherals",
published by Advance Micro Devices, Inc., 1983

PART VII

7. APPENDIX

Part seven contains seven sections used to support the description of how the DMA/DCM functions. Section one is Table 7.1 or the LIST OF VARIABLES AND ACRONYMS. Section two contains the computer programs used to generate some of the figures. Section three contains the detailed schematics. In section four the parts list, Table 7.2 is given. Section five contains the PAL programs used for coding the PAL20C1 and PAL16H2. Section six shows Figure 7.6, or the module layout. The final section shows the module timing.

7.1 Table of Variables and Acronyms

The variables used in this report are defined in this section in Table 7.1

TABLE 7.1

TABLE OF VARIABLES AND ACRONYMS

Dedicated Multichannel A/D Converter Module	DMA/DCM
Digital to Analog Converter	DAC
Nonlinear Gain Amplifier	NGA
Transfer Acknowledge Signal	-XACK
Megahertz	MHz
Hertz	Hz
First In First Out	FIFO
Digital Control System	DCS
Analog Control System	ACS
Minimal Acceptable Sampling Frequency	MASF
Normalized Sampling Frequency	NSF
Normalized Duty Cycle	NDC
Frequency at the Crossover	ω_c
Sampling Frequency	ω_s
Normalized Sample Rate	T
Zero Order Hold	ZOH
Sample Timing Controller	STC
Programmable Clock	PC
Window Logic	WL
Sequence Memory	SM
Static Random Access Memory	SRAM
Successive Approximation	DA
Full Scale	FS

(Continued -- TABLE 7.1)

Analog to Digital Converter.....A/D

Input/Output.....I/O

Gain of Digital Filter.....K

Conversion Speed.....CS

Absolute Maximum Value of Hysteresis.....MH

Gain of NGA.....G

Maximum Positive Threshold.....MPT

7.2 Support Software

The following eight programs were used to generate the plots used in Parts II, III, and IV.

```
C* RESA /MASTER/SPEC
C
 INTEGER C1,C2,C3,C4,C5,C6,C7
C
C
C*********************************************************
C THIS PROGRAM CALCULATES THE AMOUNT OF SPECTRAL OVERLAP*
C AS A FUNCTION OF THE NSF AND NDC SHOWN IN FIGURE 2.1. *
C*********************************************************
C
C
 CALL ATTACH(10,"/HOME/DATA;",2,0,I1,)
C
C***** INITILIZE CONSTANCES *****
C
 DATA TPI,SF,SW,SA,DW/6.283185,1.0,0.02,0.0,0.1/
 DATA AR,OV/0.0,0.0/
C
C*****         OUTER LOOP          *****
C***** INCREMENT SAMPLING FREQUENCY *****
C
 DO 30 J=1,20
C
C*****      MIDDLE LOOP      *****
C***** INCREMENT PULSE WIDTH *****
C
  DO 35 K=1,5
   AR=0.0
   OV=0.0
   C1=0
   C2=0
   C3=0
   C4=0
   C5=0
   C6=0
   C7=0
   LAP(K)=0.0
   SR=1.0/SF
   SA=SR/(TPI*SW)
   CONS=((TPI*SW*SA)/(SR))
   WN=TPI/SR
   WS=((WN*SW)/2.0)
C
C*****        INNER LOOP           *****
C***** CALCULATE AMOUNT OF OVERLAP *****
C
    DO 20 I=1,5001
     F0=CONS*(SIN(WS))/WS
     F1=CONS*(SIN(WS-WN))/(WS-WN)
     F2=F0**2
```

```
      F3=F1**2
       IF(C1.GE.1) GO TO 21
       IF(F0.LT.0.0) C1=1
        21 IF(C2.GE.1) GO TO 22
           IF(C1.LT.1) GO TO 22
           IF(F0.GT.0.0) C2=1
        22 IF(C3.GE.1) GO TO 23
           IF(C2.LT.1) GO TO 23
           IF(F0.LT.0.0) C3=1
        23 IF(C4.GE.1) GO TO 24
           IF(C3.LT.1) GO TO 24
           IF(F0.GT.0.0) C4=1
        24 IF(C5.GE.1) GO TO 25
           IF(C4.LT.1) GO TO 25
           IF(F0.LT.0.0) C5=1
        25 IF(C6.GE.1) GO TO 26
           IF(C5.LT.1) GO TO 26
           IF(F0.GT.0.0) C6=1
        26 CONTINUE
           C7=C1+C2+C3+C4+C5+C6
           IF(C7.GE.6) GOTO 28
           AR=AR+F2*DW
           OV=OV+F3*DW
           WS=WS+DW
    20 CONTINUE
C
   28 CONTINUE
   LAP(K)=100.0*(OV/AR)
   SW=SW+0.02
  35 CONTINUE
C
  SW=0.02
   WRITE(10,10)SF,LAP(1),LAP(2),LAP(3),LAP(4),LAP(5)
   SF=SF+1.0
 30 CONTINUE
C
C***** FORMAT STATEMENT *****
C
 10 FORMAT(1X,F10.5,1X,F10.5,1X,F10.5,1X,F10.5,1X,F10.5,1X,F10.5)
 STOP;END
```

```
C*RESA /MASTER/M3DB
C
 REAL PI,K,PZ5,PA,PE
 CALL ATTACH(10,"/HOME/DATA;",2,0,I1,)
 DATA PI/3.1415927/
C
C
C*********************************************************
C THIS PROGRAM CALCULATES THE ANALOG AND DIGITAL        *
C PHASES AT THE -3DB POINT. IN ADDITION, THE APPROXIMATE*
C PHASE AT THE -3DB POINT FOR THE DIGITAL PHASE IS      *
C DETERMINED AND USED TO GENERATE FIGURE 2.3.           *
C*********************************************************
C
C
C
C***** CALCULATE ANALOG PHASE AND SET CONSTANTS *****
C
 K=180.0/PI
 PA=ATAN(-1.0)*K
 F=1.0
C
C***** CALCULATE DIGITAL AND APPROXIMATE PHASES *****
C*****        AS FUNCTIONS OF THE NSF           *****
C
 DO 30 I=1,101
  PE=PA-(1.0/F)*K
  PZ5=(ATAN(-SIN(1.0/F)/((COS(1.0/F))-EXP(-1.0/F))))*K
   WRITE(10,15)F,PA,PZ5,PE
  F=F+1.0
 30 CONTINUE
C
C*** FORMAT STATEMENT ***
C
 15 FORMAT(1X,F10.5,1X,F10.5,1X,F10.5,1X,F10.5)
 STOP;END
```

```
C*RESA /MASTER/HOLDS
C
 REAL PI,K,PD,ZOH,FOH
 CALL ATTACH(10,"/HOME/DATA;",2,0,I1,)
 DATA PI,DW,W/3.1415927,0.01,0.0/
C
C
C*********************************************************
C THIS PROGRAM CALCULATES THE ADDED PHASE DUE TO A ZERO*
C AND FIRST ORDER HOLD AND THE ADDED PHASE DUE TO A    *
C PURE DELAY. THIS INFORMATION IS PLOTED IN FIGURE 2.4.*
C*********************************************************
C
C
C
C***** DETERMINE CONSTANT *****
 K=180.0/PI
C
C
C***** DETERMINE PHASE OF ZOH, FOH AND PURE DELAY *****
C*****        AS FUNCTIONS OF THE NSF             *****
C
 DO 20 J=1,101
  ZOH=-PI*W*K
  PD=2.0*ZOH
  FOH=((ATAN(2.0*PI*W))-2*PI*W)*K
   WRITE(10,15)W,ZOH,FOH,PD
  W=W+DW
 20 CONTINUE
C
C*** FORMAT STATEMENT ***
C
 15 FORMAT(F5.2,1X,F6.1,1X,F6.1,1X,F6.1)
 STOP;END
```

```
C*RESA /MASTER/STEP
C
 REAL Y(1100),X(1100),Z(1100),LIM
 CALL ATTACH(10,"/HOME/DATA;",2,0,I1,)
 DATA T1,T2,DT,OFF,LIM,CON,ST/0.0,0.0,0.01,0.0,0.5,0.0,0.0/
C
C
C*************************************************************
C THIS PROGRAM DETERMINES THE ANALOG, DIGITAL AND EFFECTIVE*
C STEP RESPONSES AS FUNCTIONS OF TIME. THE DATA FROM THIS  *
C PROGRAM IS USED TO GENERATE FIGURE 2.5.                  *
C*************************************************************
C
C
C
C***** DETERMINE CONSTANT *****
C
 DELAY=LIM/2.0
C
C***** DETERMINE THE ANALOG, DIGITAL AND EFFECTIVE *****
C*****    STEP RESPONSES AS A FUNCTION OF TIME     *****
C
 DO 20 I=1,501
  FL=1.0
   Y(I)=1.0-EXP(-T1)
    IF(OFF.LT.LIM) X(I)=CON
    IF(OFF.GE.LIM) CON=Y(I)
    IF(OFF.GE.LIM) X(I)=CON
    IF(OFF.GE.LIM) FL=0.0
    IF(FL.LE.0.0) X(I)=X(I-1)
    IF(OFF.GE.LIM) OFF=0.0
    IF(T1.LT.DELAY) Z(I)=0.0
    IF(T1.LT.DELAY) ST=0.0
    IF(T1.GE.DELAY) ST=1.0
    IF(T1.GE.DELAY) Z(I)=1.0-EXP(-T2)
   WRITE(10,10)T1,X(I),Y(I),Z(I),FL
  T1=T1+DT*FL
  T2=T2+DT*ST*FL
  OFF=OFF+DT*FL
 20 CONTINUE
C
C*** FORMAT STATEMENTS ***
C
 10 FORMAT(1X,F10.5,1X,F10.5,1X,F10.5,1X,F10.5,1X,F10.5)
 STOP;END
```

```
C*RESA /MASTER/NGA
C
 REAL A(1205),B(1205),PI
 CALL ATTACH(10,"/HOME/DATA;",2,0,I1,)
 DATA PI,W,C,THP,THN/3.1415927,0.0,8.0,0.1,-0.1/
C
C
C***********************************************
C THIS PROGRAM DETERMINES THE OUTPUT RESPONSE OF*
C THE NGA AS A FUNCTION OF THE INPUT AMPLITUDE. *
C THE DATA FROM THIS PROGRAM IS USED TO GENERATE*
C FIGURE 3.4.                                   *
C***********************************************
C
C
C
C***** DETERMINE CONSTANTS *****
C
 DW=2.0*PI/1200.0
C
C***** CALCULATE THE OUTPUT RESPONSE OF THE NGA *****
C
 DO 20 J=1,1201
  A(J)=SIN(W)
   IF(A(J).GT.THP) C=1.0
   IF(A(J).LE.THP) C=8.0
   IF(A(J).LT.THN) C=1.0
  B(J)=C*SIN(W)
    WRITE(10,10)W,10.0*A(J),10.0*B(J)
   W=W+DW
 20 CONTINUE
C
C*** FORMAT STATEMENT ***
 10 FORMAT(F10.5,1X,F10.5,1X,F10.5)
 STOP;END
```

```
C*RESA /MASTER/HYST
C
 DIMENSION A(110,12),B(110,12),C(110,12)
 CALL ATTACH(10,"/HOME/DATA;",2,0,I1,)
 DATA R3,R4,V1,DR/1000.0,100.0,0.5,90.0/
C
C
C*************************************************
C THIS PROGRAM CALCULATES THE AMOUNT OF HYSTERESIS*
C AS A FUNCTION OF R3 AND R4. THE OUTPUT OF THIS  *
C PROGRAM IS USED TO GENERATE FIGURE 3.5.         *
C*************************************************
C
C
C
C***** OUTER LOOP FOR INCREMENTING R3 *****
C
 DO 30 K=1,11
  R3=1000.0
C
C***** INNER LOOP FOR INCREMENTING R4 *****
C
   DO 20 J=1,100
    A(J,K)=(V1*R3)/(R3+R4)
    B(J,K)=(5.0*R4+(V1*R3))/(R3+R4)
    C(J,K)=B(J,K)-A(J,K)
     R3=R3+DR
   20 CONTINUE
C
  R4=R4+DR
 30 CONTINUE
C
  R3=1000.0
C
C***** WRITE TO DATA FILE *****
C
 DO 40 I=1,100
  WRITE(10,10)R3,C(I,1),C(I,2),C(I,3),C(I,4),C(I,5),C(I,6),C(I,7)
   R3=R3+DR
 40 CONTINUE
C
C*** FORMAT STATEMENTS ***
 10 FORMAT(1X,F8.3,F8.3,F8.3,F8.3,F8.3,F8.3,F8.3,F8.3,F8.3)
 STOP;END
```

```
C*RESA /MASTER/CLOCK
C
 DIMENSION X(1191)
 CALL ATTACH(10,"/HOME/DATA;",2,0,I1,)
 DATA F,DF,DS/6.0E6,1.0,60.0/
C
C
C*******************************************************
C THIS PROGRAM USED TO GENERATE FIGURE 3.12 WHICH    *
C SHOWS THE AVAILABLE PROGRAMMABLE CLOCK FREQUENCIES.*
C*******************************************************
C
C
C***** DETERMINE CONSTANTS *****
C
 AMP=F/(DF+DS)
C
C*****   CALCULATE AVAILABLE CLOCK FREQUENCIES    *****
C***** AS A FUNCTION OF DIVIDING THE 6.0MHZ CLOCK *****
C*****            BY AN INTEGER VALUE             *****
C
 DO 20 I=1,1191
  X(I)=F/(DF+DS)
   IF(AMP-X(I).LT.1000) GOTO22
   DELTA=AMP-X(I)
   AMP=X(I)
   WRITE(10,10)DS,AMP,DELTA
  22 DS=DS+DF
 20 CONTINUE
C
C*** FORMAT STATEMENT ***
 10 FORMAT(1X,F12.5,1X,F12.5,1X,F12.5)
 STOP;END
```

```
C*RESA /MASTER/SAMPLE
C
 INTEGER IFLAG,ICOUNT
 REAL SF,FLAG,A(512),DY(8)
 REAL F(8),FO(8),B(256)
 CALL ATTACH(10,"/HOME/DATA;",2,0,I1,)
 DATA T,A(1),X,FT,J/0.0,0.0,0.0,0.0,0/
C
C
C*******************************************************
C THIS PROGRAM CALCULATES THE DATA THAT SHOULD BE     *
C PLACED IN THE SEQUENCE MEMORY FOR A GIVEN SAMPLING  *
C AND GIVEN CHANNEL FREQUENCIES. THIS PROGRAM WAS USED*
C TO GENERATE FIGURES 4.1 AND 4.2.                    *
C*******************************************************
C
C
C***** QUESTION PROMPTER *****
C
 52 CONTINUE
   FT=0.0
  WRITE(5,99)
  READ(6,90)SR
  WRITE(5,98)
  READ(6,91)N
   DO 20 I=1,N
    WRITE(5,97)I
    READ(6,90)F(I)
     FT=FT+F(I)
 20 CONTINUE
  IF(SR.LT.10.0*FT) WRITE(5,94)
  IF(SR.LT.10.0*FT) GOTO52
C
C***** ORDERING ALGORITHIM *****
C
 DO 24 K=1,8
   FLAG=50.0
  DO 22 I=1,8
    IF(F(I).GT.FLAG) IFLAG=I
    IF(F(I).GT.FLAG) FLAG=F(I)
  22 CONTINUE
    FO(K)=FLAG
    F(IFLAG)=0.0
 24 CONTINUE
  FLAG=5000.0
C
C***** DETERMINE SAMPLING PERIOD *****
C
 DX=1.0/SR
C
```

```
C***** SET SERIES TO BE CALCULATED *****
C
 ICOUNT=512
C
C***** DETERMINE DELTAS BETWEEN SAMPLES *****
C
 WRITE(5,87)
 READ(6,90)C
  DO 26 I=1,N
   DY(I)=1.0/(C*FO(I))
  26 CONTINUE
 IF((DY(1)/DX)-1.0.LT.1.0) WRITE(5,96)
C
C***** DETERMINE FASTS SIGNAL LOCATIONS*****
C
 DO 28 I=1,ICOUNT
  IF(X.GT.DY(1)-DX) X=0.0
   IF(X.LT.DX) A(I)=1.0
    IF(A(I).GE.FLOAT(N).AND.L.GT.0.AND.M.LE.0) M=I
    IF(A(I).GE.FLOAT(N).AND.L.LE.0) L=I
   IF(X.GE.DX) A(I)=0.0
     X=X+DX
 28 CONTINUE
 IF(N.LE.1) GOTO 48
   K=2
C
C***** DETERMINE NEXT AVAILABLE SAMPLE SPACE *****
C
  IFLAG=0
 38 CONTINUE
   IF(IFLAG.GE.1) GOTO33
   X=0.0
  DO 30 I=1,ICOUNT
   IF(A(I).LT.0.9) J=I
   IF(A(I).LT.0.9) GOTO32
  30 CONTINUE
 32 IF(J.LE.0.AND.IFLAG.LE.0) IFLAG=1
C
C***** ESTABLISH NONUNIFORM SAMPLING OFFSETS *****
C
  IF(J.LE.0.AND.IFLAG.GE.1) DY(2)=DY(2)+DX
  IF(J.LE.0.AND.IFLAG.GE.1) DY(3)=DY(3)+DX
  IF(J.LE.0.AND.IFLAG.GE.1) DY(4)=DY(4)+DX
  IF(J.LE.0.AND.IFLAG.GE.1) DY(5)=DY(5)+DX
  IF(J.LE.0.AND.IFLAG.GE.1) DY(6)=DY(6)+DX
  IF(J.LE.0.AND.IFLAG.GE.1) DY(7)=DY(7)+DX
  IF(J.LE.0.AND.IFLAG.GE.1) DY(8)=DY(8)+DX
C
C***** DETERMINE NEXT FASTEST SIGNAL LOCATION *****
C
```

```
 33 CONTINUE
  DO 34 I=J,ICOUNT
   IF(X.GT.DY(K)-DX) X=0.0
   IF(IFLAG.LT.1.AND.X.LE.DX.AND.A(I).LT.0.9) A(I)=FLOAT(K)
   IF(IFLAG.GE.1.AND.X.LT.DX) A(I)=FLOAT(K)
   IF(A(I).GE.FLOAT(N).AND.L.GT.0.AND.M.LE.0) M=I
   IF(A(I).GE.FLOAT(N).AND.L.LE.0) L=I
    X=X+DX
  34 CONTINUE
   IF(K.LE.N) K=K+1
   IF(K.LE.N) GOTO 38
 48 CONTINUE
C
C***** FILL SPARE SLOTS PROMPTER *****
C
  IF(IFLAG.LE.0) WRITE(5,93)
  IF(IFLAG.LE.0) READ(6,91)N
  WRITE(5,95)
C
C***** WRITE DATA *****
C
  L=L+1
  K=1
 DO 50 I=L,M
  IF(A(I).LT.0.9.AND.N.GE.1) A(I)=1.0
   WRITE(10,88)K,A(I)
   WRITE(5,88)K,A(I)
  K=K+1
 50 CONTINUE
C
C ***** FORMAT STATEMENTS *****
C
 86 FORMAT(1X,'SIGNAL #',I2,' PERIOD IS',E10.3,' SECONDS')
 87 FORMAT(1X,'NUMBER OF SAMPLES PER SAMPLE PERIOD?')
 88 FORMAT(1X,I5,5X,F15.10)
 90 FORMAT(F10.5)
 91 FORMAT(I3)
 92 FORMAT(1X,'INSUFFICIENT MEMORY!!!')
 93 FORMAT(1X,'USE EXTRA SPACES FOR FASTS SIGNAL? (YES=1)',I2)
 94 FORMAT(1X,'SAMPLE FREQUENCY TO SLOW!!!')
 96 FORMAT(1X,'NONUNIFORM SAMPLING WARNING!!!')
 95 FORMAT(1X,'ADDRESS        CHANNEL')
 97 FORMAT(1X,'THE MAXIMUM FREQUENCY ON CHANNEL # ',I2,' IS ?')
 98 FORMAT(1X,'HOW MANY CHANNELS ?')
 99 FORMAT(1X'WHAT IS THE SAMPLE FREQUENCY ( A REAL NUMBER ) ?')
 STOP;END
```

7.3 Circuit Schematic

The detailed schematics for the DMA/DCM are drawn in this section. Figures 7.1 to 7.4 represent the four detailed schematics.

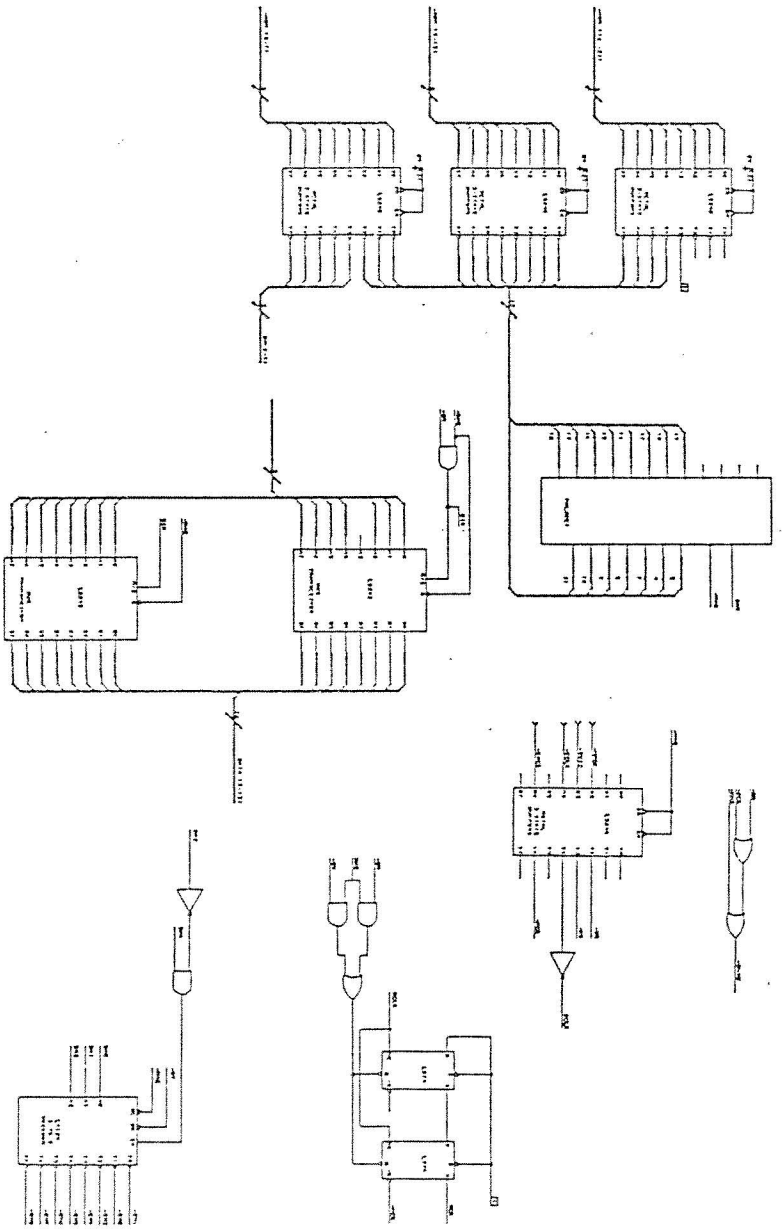

Figure 7.1
Address and Decode Logic Schematic

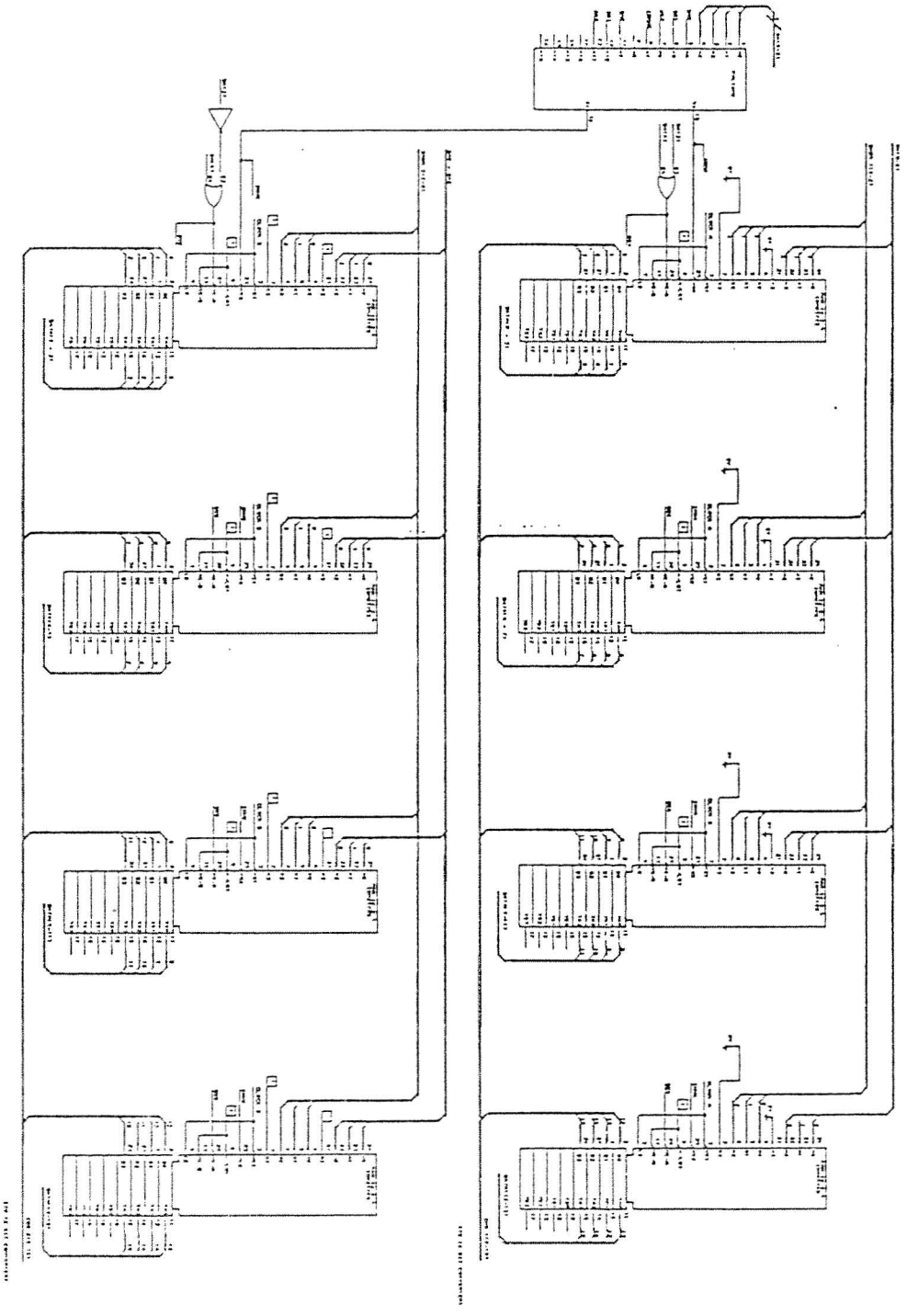

Figure 7.2
Dual Port 16 by 16 SRAM Schematic

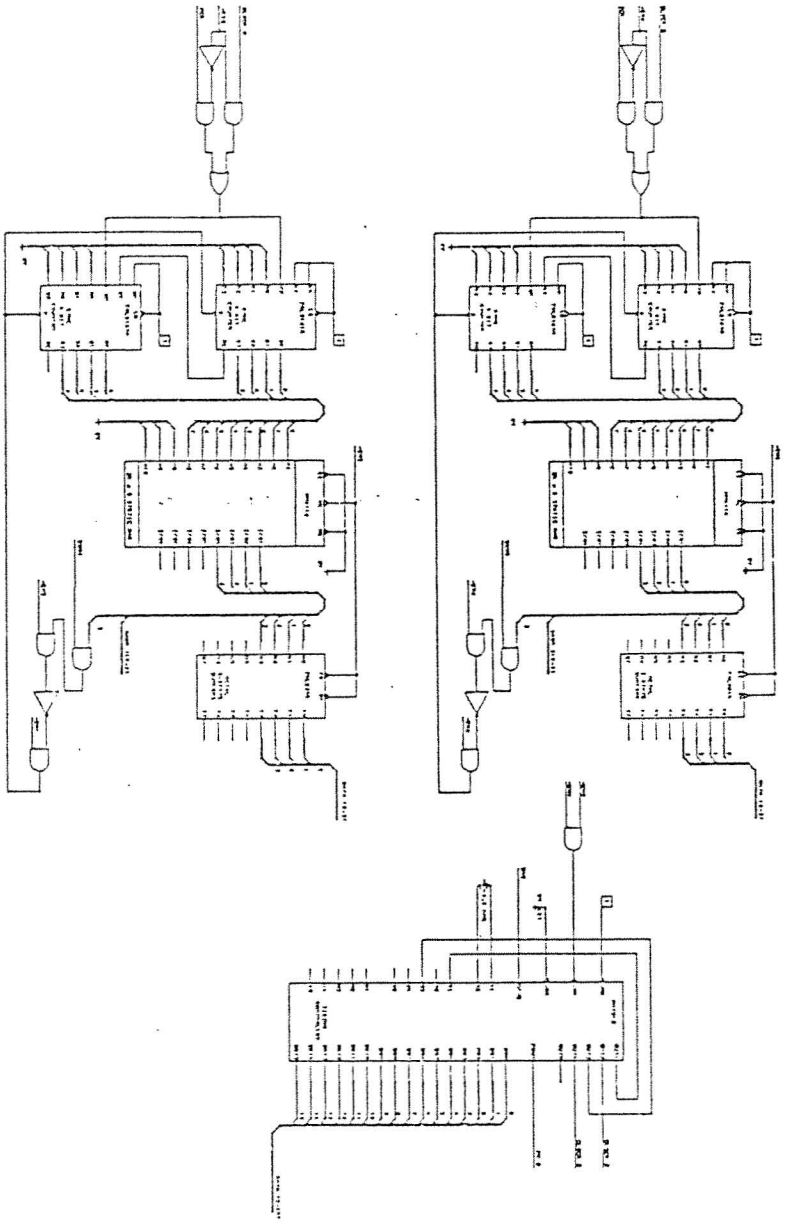

Figure 7.3
Sample Timing Controller Schematic

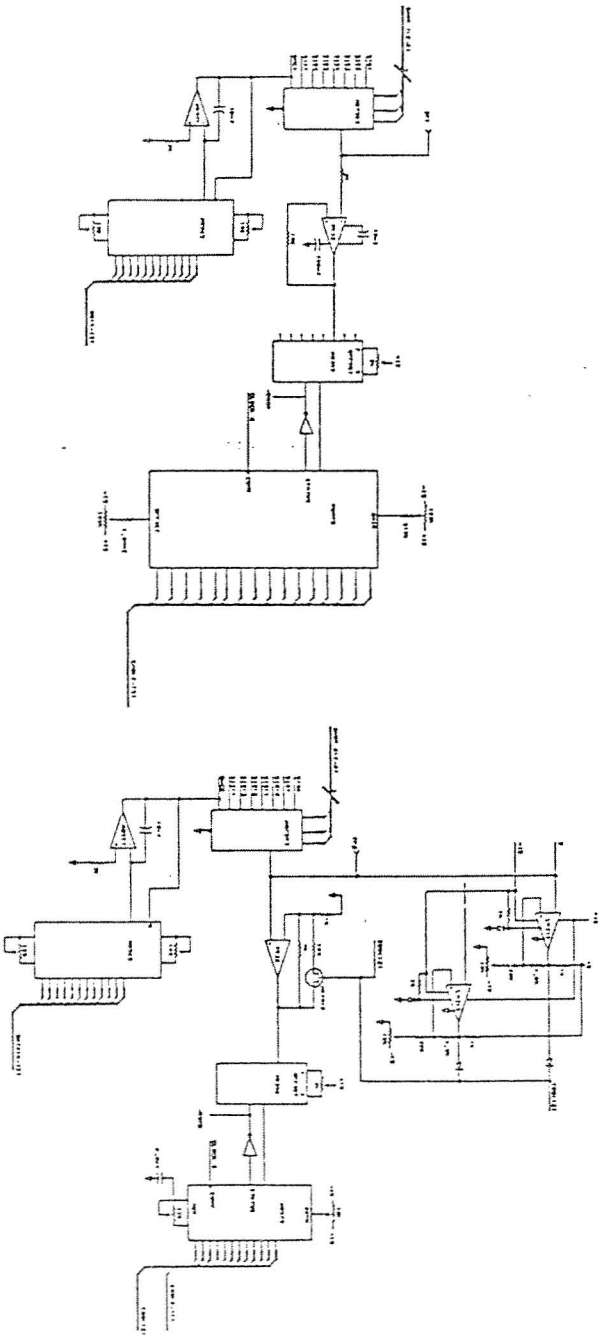

Figure 7.4
A/D Converter Paths Schematic

7.4 Parts List

Section four of part seven contains the detailed parts list for the DMA/DCM. Table 7.2 shows a listing of all the parts used.

TABLE 7.2

PARTS LIST

DESCRIPTION	MANUFACTURE	DEVICE	QUANITY
A/D 16 BIT CONVERTER	BURR-BROWN	PCM75KG	1
A/D 12 BIT CONVERTER	ANALOG DEVICES	AD578KN	1
SAMPLE & HOLD	ANALOG DEVICES	AD346JD	1
SAMPLE & HOLD	ANALOG DEVICES	AD389JD	1
8 TO 1 ANALOG MULTIPLEXER	ANALOG DEVICES	AD7503JN	2
FAST OP AMP	ANALOG DEVICES	ADLH0032CG	2
D/A 12 BIT CONVERTER	ANALOG DEVICES	AD565AJD	2
ANALOG SWITCH	ANALOG DEVICES	AD7512	1
OP AMP	ANALOG DEVICES	AD509JH	2
COMPARATOR	NATIONAL	LM319N	1
DUAL PORT SRAM	NATIONAL	IDM29705JC	8
TIMER	AMD	AM9513PC	1
HEX INVINVERTER	T.I.	SN74LS04N	2
QUAD AND	T.I.	SN74LS08N	3
QUAD OR	T.I.	SN74LS32N	3
FLIP FLOP	T.I.	SN74LS74AN	1
3 TO 8 DECODER	T.I.	SN74LS138N	1
COUNTER	T.I.	SN74LS161AN	4
NONINVERTING BUFFER	T.I.	SN74LS244N	3
INVERTING BUFFER	T.I.	SN74LS240N	3

(TABLE 7.2 - CONTINUED)

DESCRIPTION	MANUFACTURE	DEVICE	QUANITY
TRANCEIVER	T.I.	SN74LS245N	2
2 BY 8 SRAM	HITACHI	HM6116	2
DECODER	MONOLITHIC	PAL20C1CN	1
DECODER	MONOLITHIC	PAL16H2CN	1

In addition, a 6.0MHz crystal, resostors and are capacitors are used.

7.5 PAL Programs

The two PALs used on the DMA/DCM are defined by the two PAL programs that were generated by the program PALASM. Table 7.3 shows the fuse table for the PAL20C1 and Table 7.4 shows the fuse table for the PAL16H2.

```
PAL20C1                          PAL DESIGN SPECIFICATION
DECODE PAL 2                     FNB/AGB 3/14/85
C-COURSE PROJECT
GEOS

NC NC NC BA20 BA19 BA18 BA17 BA16 BA15 BA14 BA13 GND
BA12 BA11 BA10 BA09 BA08 NHAE HAE BA07 BA06 BA05 NC VCC

HAE=/BA20 + /BA19 + BA18 + BA17 + /BA16 + BA15 + BA14 + BA13 + BA12
     + BA10 + BA09 + BA08 + BA07 + BA06 + BA05 .                    (7.1)

DESCRIPTION
```

TABLE 7.3

PAL20C1 FUSE TABLE

C-COURSE PROJECT

```
              11 1111 1111 2222 2222 2233 3333 3333
   0123 4567 8901 2345 6789 0123 4567 8901 2345 6789

32 ---- ---- -X-- ---- ---- ---- ---- ---- ---- ---- /BA20
33 ---- ---- ---- -X-- ---- ---- ---- ---- ---- ---- /BA19
34 ---- ---- ---- ---- X--- ---- ---- ---- ---- ---- BA18
35 ---- ---- ---- ---- ---- X--- ---- ---- ---- ---- BA17
36 ---- ---- ---- ---- ---- ---- -X-- ---- ---- ---- /BA16
37 ---- ---- ---- ---- ---- ---- ---- X--- ---- ---- BA15
38 ---- ---- ---- ---- ---- ---- ---- ---- X--- ---- BA14
39 ---- ---- ---- ---- ---- ---- ---- ---- ---- X--- BA13

40 ---- ---- ---- ---- ---- ---- ---- ---- ---- --X- BA12
41 ---- ---- ---- ---- ---- ---- ---- --X- ---- ---- BA10
42 ---- ---- ---- ---- ---- ---- --X- ---- ---- ---- BA09
43 ---- ---- ---- ---- ---- --X- ---- ---- ---- ---- BA08
44 ---- ---- ---- ---- --X- ---- ---- ---- ---- ---- BA07
45 ---- ---- ---- --X- ---- ---- ---- ---- ---- ---- BA06
46 ---- ---- --X- ---- ---- ---- ---- ---- ---- ---- BA05

LEGEND:  X : FUSE NOT BLOWN (L,N,0)  - : FUSE BLOWN  (H,P,1)
NUMBER OF FUSES BLOW =  585
```

```
PAL16H2                          PAL DESIGN SPECIFICATION
DECODE PAL                       FNB/AGB 3/13/85
C-COURSE PROJECT
GEOS

BA0 BA1 BA2 BA3 AA0 AA1 AA2 /BHAE NC GND
AB0 AB1 AB2 NC /INHA /INHB NC NC NC VCC

/INHA=/BA0*AA0 + BA0*/AA0 + /BA1*AA1 + BA1*/AA1 + /BA2*AA2
      + BA2*/AA2 + BA3 + /BHAE                                    (7.2)

/INHB=/BA0*AB0 + BA0*/AB0 + /BA1*AB1 + BA1*/AB1 + /BA2*AB2
      + BA2*/AB2 + /BA3 + /BHAE                                   (7.3)

DESCRIPTION
```

TABLE 7.4

PAL16H2 FUSE TABLE

C-COURSE PROJECT

```
              11 1111 1111 2222 2222 2233
   0123 4567 8901 2345 6789 0123 4567 8901

24 ---X ---- ---- ---- ---- ---- ---- --X- /BA0*AB0
25 --X- ---- ---- ---- ---- ---- ---- ---X BA0*/AB0
25 -X-- ---- ---- ---- ---- ---- --X- ---- /BA1*AB1
27 X--- ---- ---- ---- ---- ---- ---X ---- BA1*/AB1
28 ---- -X-- ---- ---- ---- --X- ---- ---- /BA2*AB2
29 ---- X--- ---- ---- ---- ---X ---- ---- BA2*/AB2
30 ---- ---- -X-- ---- ---- ---- ---- ---- /BA3
31 ---- ---- ---- ---- ---- ---- X--- ---- /BHAE

32 ---X ---- ---- X--- ---- ---- ---- ---- /BA0*AA0
33 --X- ---- ---- -X-- ---- ---- ---- ---- BA0*/AA0
34 -X-- ---- ---- ---- X--- ---- ---- ---- /BA1*AA1
35 X--- ---- ---- ---- -X-- ---- ---- ---- BA1*/AA1
36 ---- -X-- ---- ---- ---- X--- ---- ---- /BA2*AA2
37 ---- X--- ---- ---- ---- -X-- ---- ---- BA2*/AA2
38 ---- ---- X--- ---- ---- ---- ---- ---- BA3
39 ---- ---- ---- ---- ---- ---- X--- ---- /BHAE

LEGEND:  X : FUSE NOT BLOWN (L,N,0)  - : FUSE BLOWN  (H,P,1)
NUMBER OF FUSES BLOW =  585
```

7.6 Module Layout

In section six of part seven Figure 7.5 shows in a general sense how the module was layed out on a MULTIBUS circuit card.

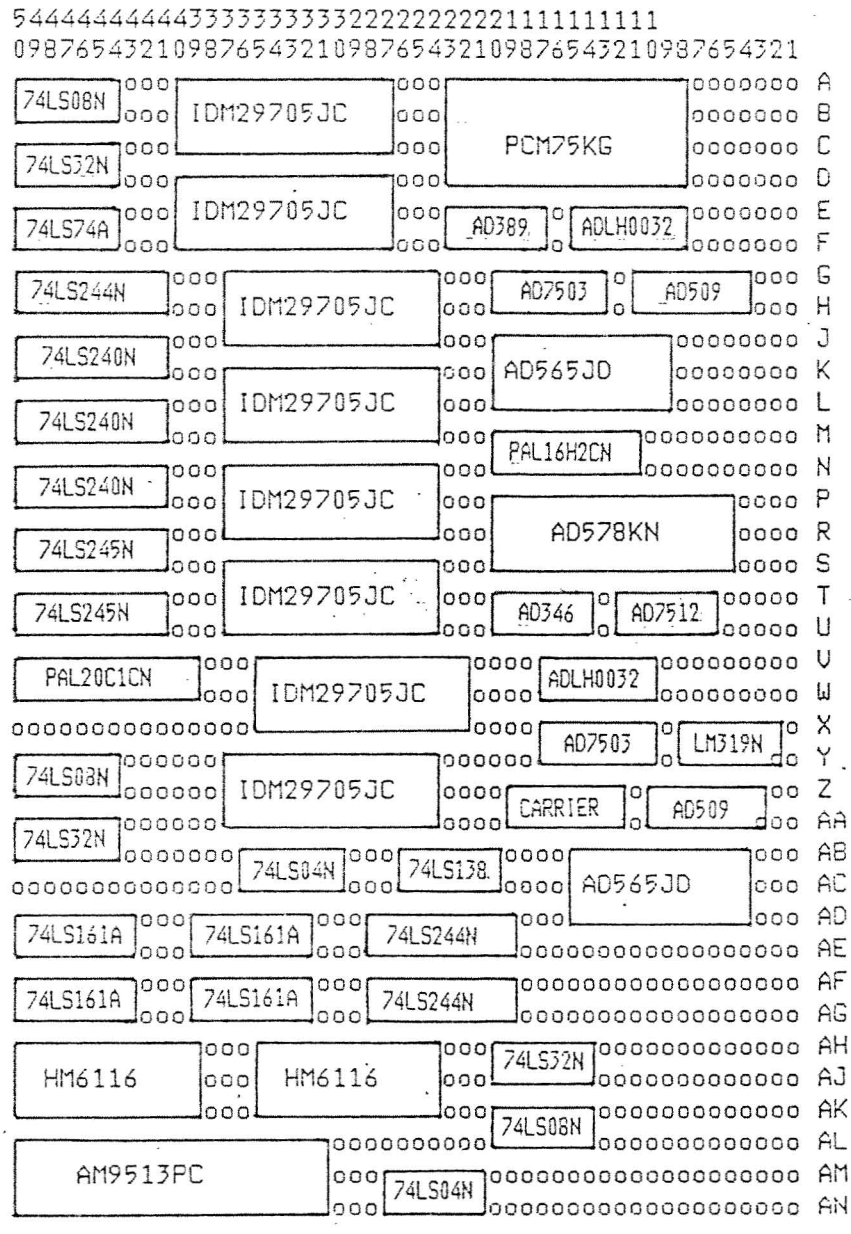

Figure 7.5
Module Layout

7.7 Module Timing

Figure 7.6 shows the typical timing for the module. The reason for this figure is to show when events occur relative to other events. There are five events of importance. First, on the falling edge of CLOCKA the A/D converter is commanded to start a conversion. The status or end of conversion line will remain high, at a logic one level, until the conversion is completed. When the end of conversion line goes low the dual port SRAM is commanded to clock in the data from the A/D converter. Next, the window logic is clocked and the sequence memory forces the address lines of the analog input multiplexer and dual port SRAM to the next address. Finally, the sample and hold circit is commanded to acquire another sample and the events continue.

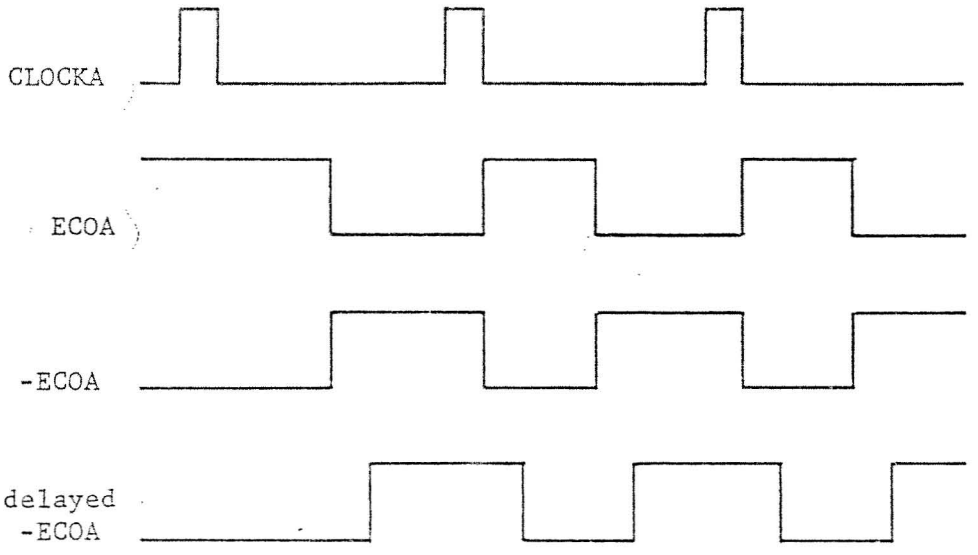

Figure 7.6
Module Timing