UILU-ENG-82-2502

A E R O N O M Y R E P O R T

N O. 102

A PREPROCESSOR FOR THE URBANA COHERENT-SCATTER RADAR

by

F. T. Zendt
S. A. Bowhill

March 1, 1982

Supported by

National Aeronautics and Space Administration
Grant NSG 7506
National Science Foundation
Grant ATM-78 21765

Aeronomy Laboratory
Department of Electrical Engineering
University of Illinois
Urbana, Illinois

ABSTRACT

A preprocessor built for the Urbana coherent-scatter radar system is described. The object of the preprocessor is to increase the altitude and temporal resolution of the present coherent-scatter system. This system upgrade requires an increase in the data collection rate. Replacing the present, relatively slow, ADC with two high speed ADCs achieves the increased echo sampling rate desired. To stay within the capabilities of the main computer's I/O and processing rate the data must be reduced before transfer to the main computer. Thus the preprocessor also coherently integrates the data before transfer. This report describes the design, interfacing, testing, and operation of the preprocessor.

TABLE OF CONTENTS

LIST OF TABLES

Page

LIST OF FIGURES

1. INTRODUCTION

1.1 *Backscattering Theory*

When an electromagnetic wave propagates through a dielectric medium the wave is reflected or attenuated according to the properties of the medium. For a transparent medium, most of the signal power is transmitted through the medium, only a small amount is reflected at interfaces or converted to a different form of energy (attenuated). In any dielectric however, some of the power in the incident wave is scattered in all directions. This scattering is due to induced oscillations of electrons, bound or free. Changes in the electron density change the dielectric constant of the medium and hence affect the scattering properties.

In the atmosphere there are two principal processes which produce the electron density fluctuations. Thermodynamic fluctuations of ions produce fluctuations in electron density that give what is commonly called Thomson or incoherent scattering. Backscattered radio waves due to Thomson scattering have been used for a number of years to investigate the ionosphere (EVANS 1969).

Radar echoes received from the stratosphere and mesosphere are often greatly enhanced over the thermal scattering level. This indicates that a second mechanism of electron density fluctuation must exist. WOODMAN and GUILLEN (1974) have proposed that the enhancement of the dielectric fluctuations in the stratosphere is due to clear air turbulence. VILLARS and WEISSKOPF (1955) have discussed how turbulence in the neutral atmosphere affects radio wave scattering. More recently this subject was considered by RASTOGI and BOWHILL (1976a).

1.2 *Turbulent Scatter Theory*

When considering the radio wave scattering from the mesosphere one

starts with the continuity equation for electron density which, in the absence of diffusion, is

$$\frac{\partial N}{\partial t} = q - L - \nabla \cdot (N\bar{v}) \tag{1.1}$$

The terms on the right-hand side represent production, loss and transport respectively. The time constants for the production and loss terms are assumed long compared to the time constant for transport. Since the production and loss processes nearly balance each other, $q - L = 0$ one has $\frac{\partial N}{\partial t} = -\nabla \cdot (N\bar{v})$. The right-hand side can be factored using a vector identity, yielding

$$\frac{\partial N}{\partial t} = -\bar{v} \cdot \nabla N - N(\nabla \cdot \bar{v}) \tag{1.2}$$

In the mesosphere there are frequent collisions between electrons and neutral particles, hence the velocity $\bar{v}$ in equation (1.2) is the velocity of the neutral medium, which is assumed to be incompressible. Under this assumption equation (1.2) becomes

$$\frac{\partial N}{\partial t} = -\bar{v} \cdot \nabla N \tag{1.3}$$

Generally the electron density gradient, ∇N, is vertical. Since the vertical velocities are usually very small, $\frac{\partial N}{\partial t} \approx 0$. If there is a horizontal velocity component $\bar{u}$, however, then a $u \cdot \nabla N$ transport term is produced. A small fluctuating $\bar{u}$ will result in fluctuations in the electron density.

Let l be a characteristic length scale associated with wavenumber $k = l^{-1}$ and U_l and T_l a characteristic velocity and time respectively. Assuming the electron density gradients ∇N at scales larger than l are known then fluctuations δN_l at scale l are given by

$$< \delta N_l^2 > \alpha \nabla N^2 U_l^2 T_l^2 \tag{1.4}$$

Next assume that the length l is small and the turbulence is homogeneous and isotropic at this scale. RASTOGI and BOWHILL (1976a) then go on to apply

several results from the statistical theory of turbulence and work by Booker to equation (1.4) to show the scattering cross section

$$\sigma \propto r_e^2 \, \overline{\nabla N}^2 \, k_0^{-2} \, E(k_0) \, T^2 \, (k_0)$$

where r_e is the standard electron radius, $E(k_0)$ is the energy spectrum of the turbulence, $T(k_0)$ equals the characteristic time T_l and k_0 is the scattering or Bragg vector $(k_i - k_r)$. k_i and k_r are the propagation vectors of the incident wave and scattering wave respectively. This result relates the measured radar signal returns to the turbulence by which they are produced.

1.3 *Mesospheric VHF Backscatter*

Measurements of backscattered VHF radio waves from stratospheric and mesospheric turbulence were first made by WOODMAN and GUILLEN (1974) at Jicamarca. Line of sight velocity measurements were taken and the signal power, mean frequency shift, and frequency spectrum width were computed. It was concluded that one or several regions of turbulence approximately 100 m thick would explain the power levels received and the frequency spectrum width. In this experiment it was also noted that the correlation time of the backscattered signals is approximately 1 sec, much larger than the 700 μsec inter-pulse period. This fact allows the signal-to-noise ratio to be improved by adding together, for a given altitude, samples from many successive transmitter pulses. The signal, being correlated, tends to add while the noise, which is uncorrelated between transmitter pulses, tends to cancel.

RASTOGI and WOODMAN (1974) further pursued the work in mesospheric VHF backscatter. Their results show an echo power which varied over a 30 dB range, changing as much as 20 dB over 1 min, with a median strength of 4 dB above the incoherent-scatter level. It is suggested that the intermittency of these echoes is an indication of turbulence. RASTOGI and BOWHILL (1976b)

show that the scattered power depends on ε, the rate at which energy is dissipated per unit mass by turbulence. Changes in signal power are a consequence of the intermittent behavior of ε.

1.4 *Objective Of This Report*

The purpose of this report is to describe the design of a preprocessor for the Urbana coherent-scatter radar system. The object of the preprocessor is to increase the altitude and temporal resolution of the present coherent-scatter system. This system upgrade requires an increase in the data collection rate. Consequently the preprocessor also coherently integrates the collected data before transferring it to the main computer to reduce the I/O rate.

Chapter two briefly describes the present coherent-scatter system. The last section discusses more specificly the purpose of the preprocessor and how it changes the present system.

Chapter three is a detailed description of how the preprocessor works. The means used to increase the data acquisition rate and perform the coherent integration are discussed. A description of the microprogram controlling the preprocessor is also given.

The electronic circuits are presented in Chapter four. Circuit diagrams and details of all the preprocessor hardware are given. The decoding and execution of each microinstruction is explained.

Chapter five discusses the software used by the main computer to which the preprocessor is interfaced. The programs are used to transfer data from the preprocessor to the main computer and for processing of the data.

Conclusions and suggestions for future work appear in Chapter six.

2. THE URBANA COHERENT-SCATTER RADAR SYSTEM

2.1 *Introduction*

Interest in coherent scatter from the mesosphere began in the early 70's. At this time a multistatic meteor-radar system was under construction by the University of Illinois' Aeronomy Lab (HESS and GELLER, 1976). It was hoped that the same transmitter and receiving system could be employed for coherent scatter observations also. Consequently a large dipole array antenna was constructed (ALLMAN and BOWHILL, 1976) and necessary modifications to the meteor-radar system were undertaken.

During July, 1976 P. K. Rastogi made the first coherent scatter observations using the Urbana radar, the data output was in the form of oscilloscope A-traces and chart recorder outputs however. The use of a computer to perform coherent integration and complex autocorrelation of real time data was first attempted in November of 1977. No substantive data were collected however due to the poor performance of sections of the meteor-radar system. Significant improvements were then made in the system hardware to bring it to its present condition. Detailed descriptions of the Urbana radar system appear elsewhere (GIBBS and BOWHILL, 1979; ALLMAN and BOWHILL, 1976; HESS and GELLER, 1976) and will not be repeated here except to present the important operating parameters.

2.2 *Radar Hardware*

2.2.1 *Transmitter.* A single transmitter is utilized by both the coherent scatter radar and the meteor radar at Urbana. The transmitter operates at a frequency of 40.92 MHz. It has a 4 MW peak pulse output power rating with nominal average output power of 20 kW. The pulse width can be varied from 3 μs to 100 μs. The system is presently operated with a 20 μs pulse at a pulse repetition frequency of 400 Hz.

2.2.2 *Antenna and transmit/receive switch.* A coherent scatter system requires a high gain narrow beam antenna. The Urbana antenna is an array of 1008 halfwave dipoles. Characteristics of this dipole array are given in Table 2.1. Open-wire transmission line transformers are used to match the antenna and feed system. A balanced coaxial line runs from the transmitter to the antenna.

2.2.3 *Radar director.* The radar director consists of a RF section and a timing and pulse generation section. The RF unit contains crystal oscillators at the transmitter frequency, 40.92 MHz, and the receiver local oscillator frequency of 35.42 MHz. Mixing these two frequencies and phase shifting by ±45° yields quadrature reference signals at the intermediate frequency of 5.5 MHz.

The timing section of the radar director is driven by a 100-kHz external master clock derived from a 5-MHz reference. Every pulse interval is a multiple of the base 10-μs period. The transmitter, analog-to-digital converter and blanker are all driven by various pulses from the radar director. The radar-director circuitry is described by HESS and GELLER (1976); GIBBS and BOWHILL (1979) provide a timing diagram and explanation of the various controls and the resultant pulse trains. For the purposes of this report only one of the radar director signals, the echo sample window (ESW), is of import. The ESW is a negative-going pulse train, the pulse width and time between pulses are both 5 μs. The number of sample pulses and when, with respect to the radar pulse, they occur is selected by the operator.

2.2.4 *Receiving system.* The receiving system front end is a blanker/preamplifier unit. This unit is located near the T/R switch. The blanker isolates the receiving system from the antenna feed line during transmit

Table 2.1 Antenna parameters for the Urbana coherent-scatter radar.

Aperture illumination efficiency (η_i)	0.69, -1.6 dB
Antenna efficiency (η_a)	0.17, -7.6 dB
Radiation efficiency ($\eta_r = \eta_a/\eta_i$)	0.25, -6.0 dB
Physical aperture (A_o)	1100 m^2
Effective aperture ($\eta_a A_o$)	1870 m^2
Directivity ($g_o = 4\pi A_o \eta_i/\lambda^2$)	1800, 33 dB
Power gain ($g_p = \eta_r g_o$)	450, 27 dB

pulses. The low noise preamplifier is located immediately following the blanker in order to minimize system noise-figure degradation.

The receiver is single conversion with a bandwidth of 230 kHz centered around 40.92 MHz. A toggle-switch attenuator is located between the mixer and the IF section to provide signal level control at the 5.5 MHz IF frequency. The receiver output is fed to two four-quadrant multiplier chips which also receive the appropriate quadrature phase references at 5.5 MHz generated by the radar director. The quadrature-detected signals are filtered to reduce the bandwidth to 75 kHz. For coherent scatter additional filtering is added to bring the bandwidth down to 40 kHz. A detailed description of the receiver and phase detector is given by HESS and GELLER (1976).

2.3 *Data Processing*

2.3.1 *Theory.* The purpose of the coherent scatter system is to obtain wind velocity vs. altitude and time measurements. This is derived from echo power and velocity measurements at the desired heights. The data is processed in the time domain, this requires autocorrelating the digitized phase detector outputs. The echo samples, and hence phase detector outputs, consists of both noise and signal components. The noise and signal are uncorrelated however, so the autocorrelation of the sum of noise and signal is the sum of the autocorrelation of the components. In addition, the signal energy is concentrated about the Doppler frequency of the returned signal whereas the noise has a band-limited spectrum. We have, consequently, a signal component which is correlated for times much longer than the noise component. Also, as WOODMAN and GUILLEN (1974) showed, the signal is correlated for times longer than the interpulse period. The process of coherent integration, where successive samples are added together with the signal

component tending to add and the noise component tending to cancel, can be used to increase the signal-to-noise ratio (SNR). The length of the coherent integration, i.e., the number of samples summed, is chosen to maximize the increase in SNR while still maintaining a good time resolution in the subsequent autocorrelation process.

The power in the returned signal is calculated by the zeroth lag of the signal autocorrelation. The definition of the autocorrelation of a complex random process $x(t)$ is

$$R(\tau) = E\{x(t + \tau)\ x^{*}\ (t)\} \tag{2.1}$$

where the expected value of the product of two random processes is defined as

$$E\{\ x(t_1)\ x(t_2)\} = \int_{-\infty}^{\infty} x_1 x_2\ f(x_1,\ x_2;\ t_1,\ t_2)\ dx_1\ dx_2$$

The function being integrated on the right-hand side is the joint probability density function of the two random variables x_1 and x_2.

At $\tau = 0$, the zeroth lag,

$$R(0) = E\{x(t)\ x^{*}(t)\} = E\{|x(t)|^{2}\} \tag{2.2}$$

which is an expression for the power.

For coherent scatter the discrete random process $x(t)$ is the ensemble of values (radar echo samples) for a given altitude. The time between samples is the radar interpulse period, 2500 μs. Each sample is the sum of a noise and signal value. Since the noise has such a short correlation time it contributes to the total autocorrelation primarily at the zeroth lag. Assuming that the noise is constant over the sampled altitude region and varies slowly, then changes in the power calculated from the zeroth lag of the autocorrelation will be due to fluctuations in the signal power. Hence,

relative fluctuations in power over altitude and time are measured.

Motion of the scattering region is measured by the mean Doppler shift of the returned signal. If the echo sample has a Doppler shifted frequency f_d then the radial velocity V_r of the scattering region can be computed from

$$V_r = \frac{\lambda f_d}{2} \tag{2.3}$$

where λ is the radar wavelength. The radial velocity is related to the time rate of change of the phase of the autocorrelation function $\frac{d\phi}{d\tau}$ through the radian Doppler frequency ω_d as

$$V_r = \frac{\lambda f_d}{2} = \frac{\lambda \omega_d}{4\pi} = \frac{\lambda}{4\pi} \frac{d\phi}{d\tau} \tag{2.4}$$

Since the zeroth lag of the autocorrelation is real, i.e., $\phi(0) = 0$, the estimate of $\frac{d\phi}{d\tau}$ becomes

$$\frac{d\phi}{d\tau} = \frac{\phi(\tau_2) - \phi(\tau_1)}{\tau_2 - \tau_1} = \frac{\phi(\tau_2)}{\tau_2}\Bigg|_{\tau_1 = 0} \tag{2.5}$$

The coherent scatter system uses a weighted average of the first three lags to calculate the velocity.

2.3.2 *Software.* The present software performs four functions: data input, coherent integration, autocorrelation, and averaging. All of these functions are done in real time. The functions are interlaced, with the data input operating on an interrupt basis. The distance to the scattering region from which a radar pulse is reflected is computed from

$$R = \frac{1}{2} c t \tag{2.6}$$

where R is the range, c is the speed of light, and t is the elapsed time between transmitting and sampling the pulse. The distance between two consecutive samples taken at times t_1 and t_2 is then

$$R = \frac{1}{2}c\ (t_2 - t_1) \tag{2.7}$$

The analog-to-digital converter (ADC) digitizes a sample every ten microseconds, hence the distance between samples is 1.5 km. Twenty samples are taken from the 30 km region of the atmosphere (60 to 90 km) under investigation.

The two phase detector channels, cosine and sine, are sampled on alternate radar pulses. Consequently, two radar pulses are required to form one complex sample at each of the 20 heights. Since the interpulse period is 2.5 ms, twenty complex samples are produced every 5 ms. The input buffer holds twenty-five sets of complex samples. Double-buffering the input allows data collection to continue, using alternate buffers every 1/8 second, while data processing occurs.

The first step of the data processing is coherent integration. The corresponding height and phase values of the twenty-five sets of samples in a buffer are added together. This reduces the twenty-five sets of complex samples to a single set of twenty complex values.

Next the most recent coherent integration interval is correlated with data from the previous intervals. As a data set, i.e. the twenty coherently integrated complex samples, becomes available it is pushed on the top of a stack. The oldest data set is lost out the bottom. The computer now performs a complex autocorrelation on the top data set and the other data sets in the stack. The speed of the computer allows the autocorrelation to be performed out to lag 12 (1.5 seconds) in the 1/8 second coherent integration time. The result of the complex autocorrelation is a real value at lag zero and real and imaginary values at lags 1 through 12. Twenty-five numbers are thus produced for each of the twenty heights every 1/8 second.

The autocorrelation functions are now averaged for one minute by add-

ing. The one-minute averages are saved on disk for post processing. GIBBS and BOWHILL (1979) present a more complete discussion of the coherent scatter software.

2.3.3 *Data processing hardware.* Information from the phase detectors is converted to a 10-bit digital word by a Hewlett-Packard 5610A analog-to-digital converter (ADC). The ADC operates at 100 kHz with an accuracy of $\pm\frac{1}{2}$LSB. Data transfer to the main computer is through a custom built interface.

The main computer is a Digital Equipment Corporation PDP-15. Presently this machine coherently integrates, correlates and averages the data in real time. The averages are stored on disk until data collection terminates. At this time the data can be written to DECtape for temporary storage or processed immediately. The processed data is punched onto paper tape. The paper tape, which contains the useful scientific information, is read by a Hewlett-Packard 9830A desktop computer which stores the data on cassette tape. The HP9830A and its associated plotter are used to produce the three types of plots obtained from the data; velocity vs. time, power vs. altitude at a given time, and power vs. time at a given altitude.

2.4 *Purpose of a Preprocessor*

The relatively slow rate at which the present ADC and computer operate severely limits the coherent scatter system's data acquisition. In particular, the ADC's 10-μs conversion time results in an altitude range resolution of only 1.5 km. It has been shown (WOODMAN and GUILLEN (1974), WOODMAN et al. (1979)) that turbulence in the atmosphere occurs at scales as fine as 100 m. Also, since only one ADC is used, two radar pulses are required to obtain one complex sample, the cosine phase detector channel is sampled during one pulse and the sine channel during the next. This decreases the

temporal resolution. To achieve a range resolution of even 150 m requires a sampling rate of 1 μs. This increases the data input rate to the computer by a factor of ten, or twenty if two ADC's are used so that both phase detector channels can be sampled each pulse. A 2 MHz data input and processing rate is beyond the capabilities of the Urbana radar facilities' present computer, hence a method of data reduction before transfer to the computer is required. This is the purpose of the preprocessor, to increase the echo sampling rate and yet decrease the data handling demands on the computer.

An increase in the echo sample rate is achieved by replacing the Hewlett-Packard HP5610A ADC with two high speed ADCs. The data input rate to the computer is decreased by coherently integrating the data before transfer to the computer.

3. THE PREPROCESSOR

3.1 *Introduction*

The purpose of the preprocessor is to sample data and perform a real time coherent integration. This requires sampling a returned radar pulse, converting the analog signal to a binary word and adding this word to a running sum of previous samples corresponding to the same altitude and phase. When a coherent integration is complete the data must be transferred to the main computer.

The preprocessor uses two ADCs so that both phase detector channels can be sampled simultaneously. The Micro Networks MN5101 ADC chosen performs an eight bit conversion in 850 ns with an accuracy of $\pm\frac{1}{2}$ LSB. Approximately 50 ns are used to sample then hold the analog line to the ADC input. Another 100 ns are used to latch the 8-bit ADC output. This results in a 1 μs sample interval. Six hundred samples are taken each radar pulse resulting in 1200 eight bit words (600 per channel x two channels) to be saved.

To reduce the necessary amount of data to be transferred to the computer, the preprocessor also performs a coherent integration. The corresponding altitude and phase values of N successive radar pulses (N is a user selectable number between ten and 256) are summed. A 16-bit running sum of each altitude-phase value is stored by the preprocessor until the N pulses have been sampled. The 1200 16-bit words are then transferred to the computer. The preprocessor's memory is double buffered so that data acquisition and coherent integration can occur continuously. Data transfer is interleaved with the data acquisition operations. Since the preprocessor performs a 16-bit add, up to 255 8-bit samples can be summed without overflow.

3.2 *Data Acquisition*

Figure 3.1 is a block digram showing the data flow through the preprocessor. A Datel-Intersil SHM-HU sample-and-hold (S/H), Micro Networks MN 5101 analog-to-digital converter (ADC) and 8-bit latch (ADC LATCH) are required for each analog channel. The purpose of the S/H is to present a constant analog voltage to the ADC input. When the control circuitry issues a sample command the time-varying analog signal at the S/H input is sampled. This value is then held at the S/H output until the next sample command.

A START CONVERT command must be issued to reset the ADC. The SAMPLE command occurs during the START CONVERT so that when the ADC is reset a new sample is ready and conversion begins immediately. The analog signal at the ADC input is then converted to an 8-bit digital word. The EOC (End of Convert) signal enables the ADC LATCH. The ADC data word is latched so that another conversion can begin immediately. Figure 3.2 shows the timing of the data acquisition signals. Each phase detector channel is sampled, converted to a binary data word and latched once every μsec. The running sums/coherent integration for analog channel #1 are stored in odd memory locations, analog channel #2 values are stored in even memory locations.

3.3 *Coherent Integration*

A memory read, ALU add and memory write can be done in 300 nsec, hence one memory and ALU can process both channels in the 1 μs data acquisition time. The data mux presents one of the two new data words (ADC LATCH output) to one input of the ALU. The feedback buffer presents the running sum stored in the memory at the other ALU input. The ALU adds the new data word to the running sum and the new sum is written back into the same memory location where the previous running sum was stored. Figure 3.3 shows the

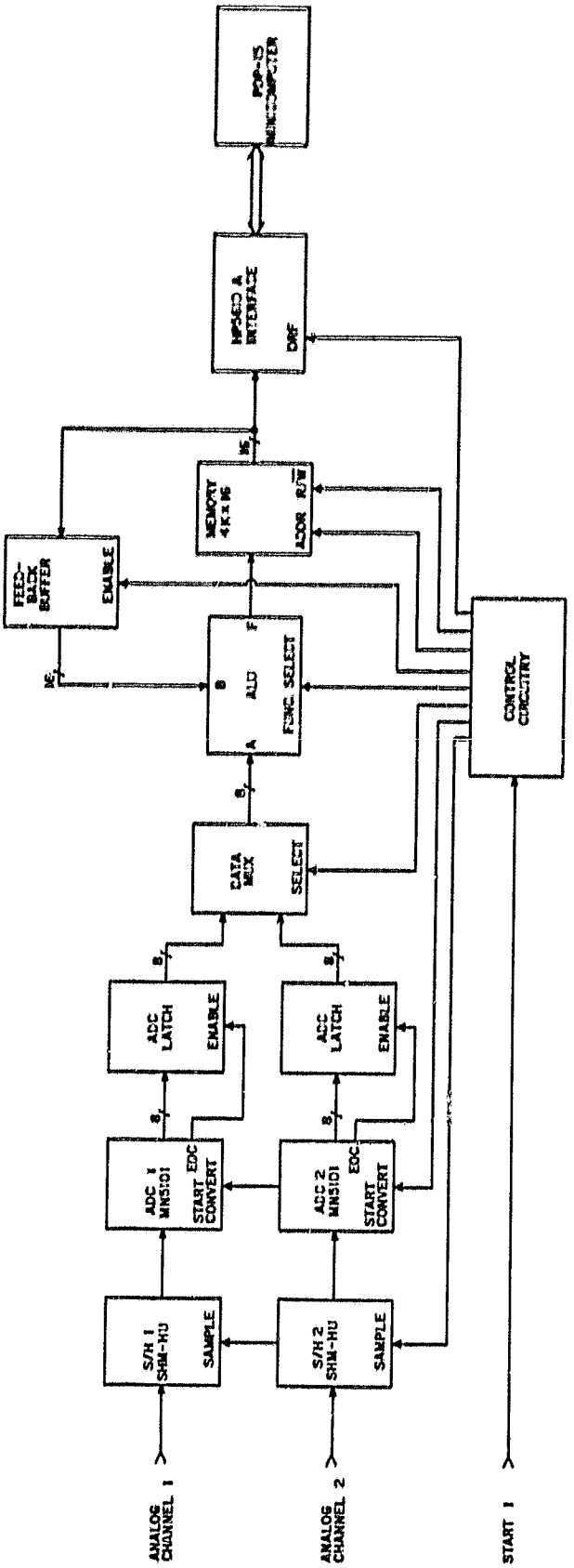

Figure 3.1 Preprocessor block diagram.

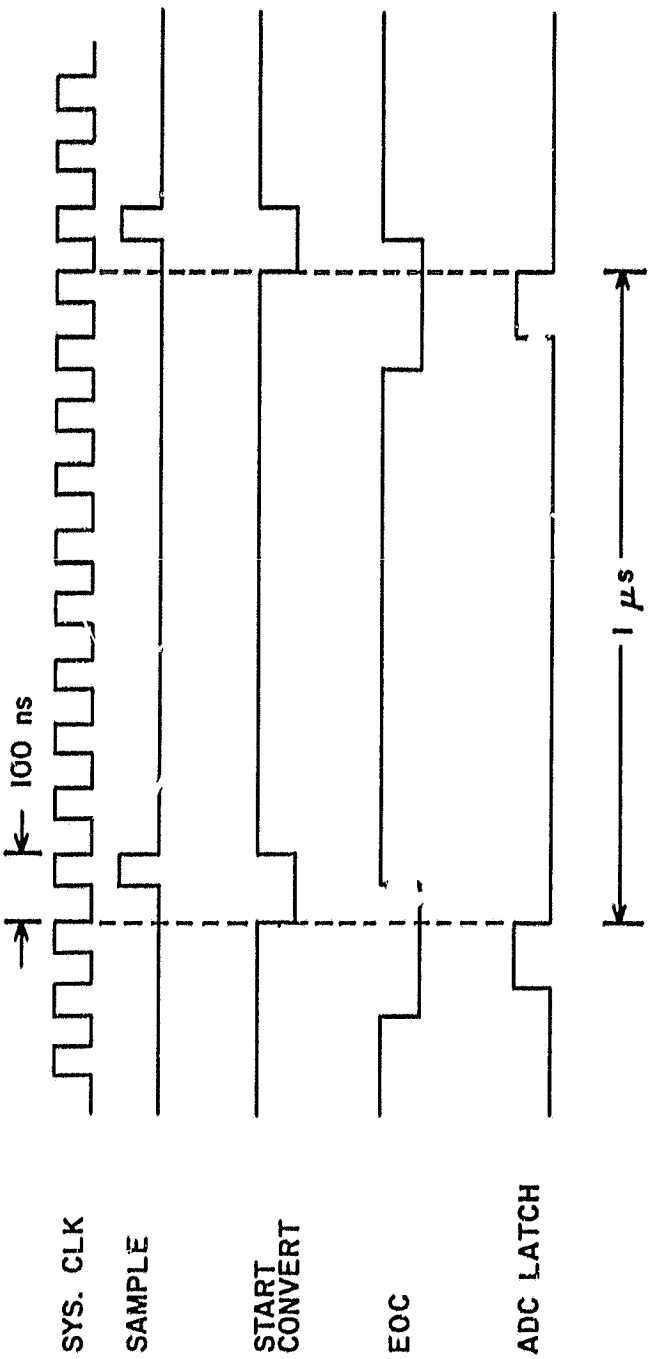

Figure 3.2 Data acquisition timing diagram.

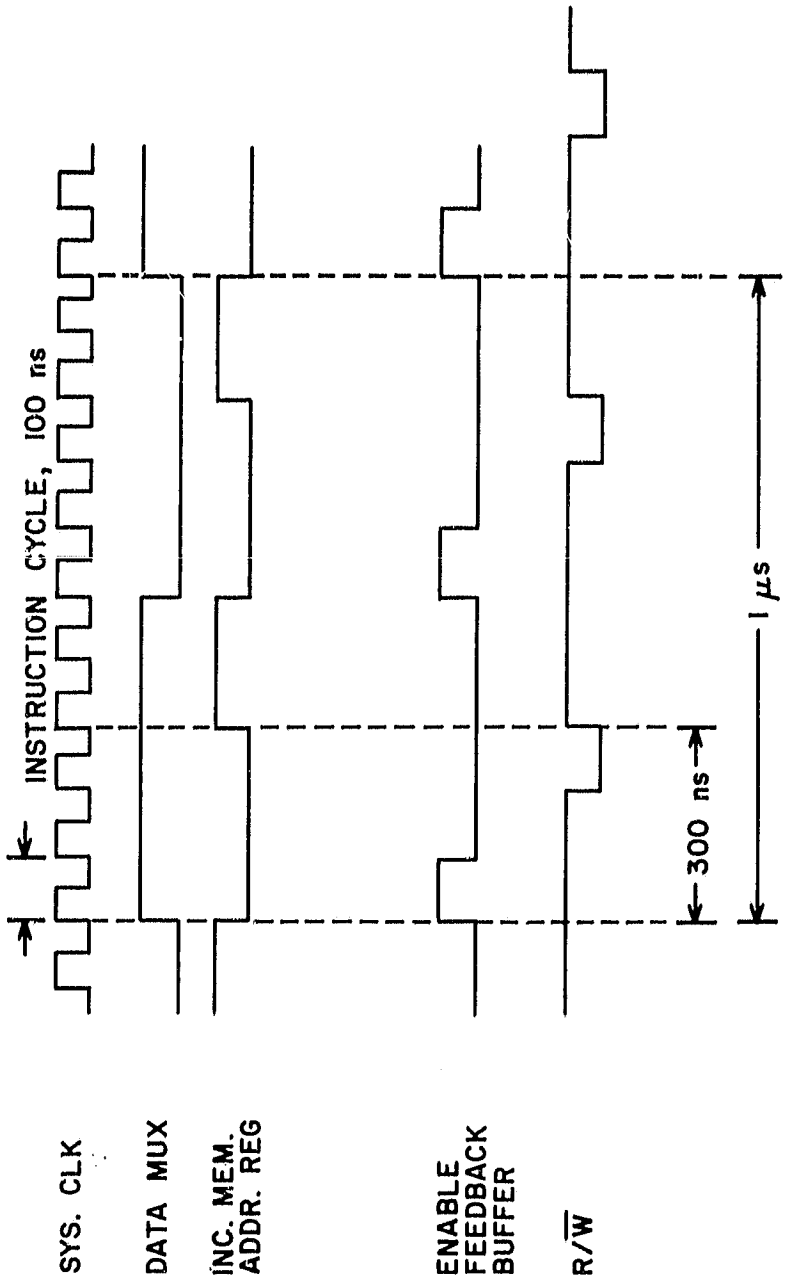

Figure 3.3 The read-add-write loop timing diagram.

timing for a read-add-write cycle. During the first instruction cycle, the data mux selects one of the input channels, the memory address register is incremented and the feedback buffer input reads the running sum stored in this location. On the second instruction cycle the feedback buffer is latched and the ALU presents a valid sum to the memory input. The third instruction cycle writes the new running sum into memory. Two instruction cycles are required for the microprogram to execute a jump (see Section 4.5) after which another read-add-write cycle is executed. Thus, the data on both input channels is processed in 1 μsec.

For the first pulse of a coherent integration there is no running sum to which the new data word is to be added. Consequently, this pulse must be treated as a special case. Rather than filling the memory with zeros and having the ALU perform the usual addition, a different approach was used. In the control circuitry a flip-flop (PULSE.1 flip-flop) was included which is set during the first pulse of a coherent integration. Pulse one flip-flop controls which arithmetic operation the ALU performs. When pulse one flip-flop is set, the output, F, equals the value on the A input, the new data word. After all 600 samples of the first pulse have been taken and written directly into the memory, PULSE.1 flip-flop is cleared. With PULSE.1 flip-flop cleared the ALU is instructed to add inputs A and B, F = A plus B.

3.4 *The Memory*

The memory is twice the size required for one coherent integration and is logically broken into two halves, 2K x 16 bits each. This allows continuous data collection to occur. At any given time, one-half of the memory stores the running sums of the coherent integration presently being collected while the other half contains the values of the previous coherent

integration. Transfer of data to the main computer is interleaved with data collection and integration (Figure 3.4).

3.5 *Interface to the Main Computer*

The preprocessor is interfaced to the PDP-15 minicomputer through an interface custom-built for a Hewlett-Packard 5610A ADC. The interface is connected to the PDP-15 I/O BUS and handles all communication with the PDP-15. To effect a data transfer only requires the preprocessor to place the desired data word on its data bus and send a data ready flag, DRF, to the interface. Data is transferred at a rate of 75 KHz.

3.6 *The Microprogram*

The preprocessor is a self-contained operating unit; it does not receive instructions from any other device. This requires that the control circuitry completely direct all operations. The control circuitry therefore consists of a control store containing a microprogram and all the hardware necessary to execute the microinstructions. Since the preprocessor is a dedicated piece of equipment, no programming flexibility is required. The control circuitry was designed to perform one set of instructions only. Figure 3.5 is a flowchart of the microprogram, while Figure 3.6 is a listing of the microprogram itself. It will be helpful to refer to these throughout the following discussion.

Instruction 0 initializes the preprocessor, there are four specific events which occur at this time. The memory full flip-flop (MEM.FULL) is set when there is data in the preprocessor memory to be transferred to the main computer. Since no coherent integration has been completed yet, there is no data to transfer. Hence the memory full flip-flop is initially cleared.

The memory half flip-flop (MEM.HALF) contains the most significant bit

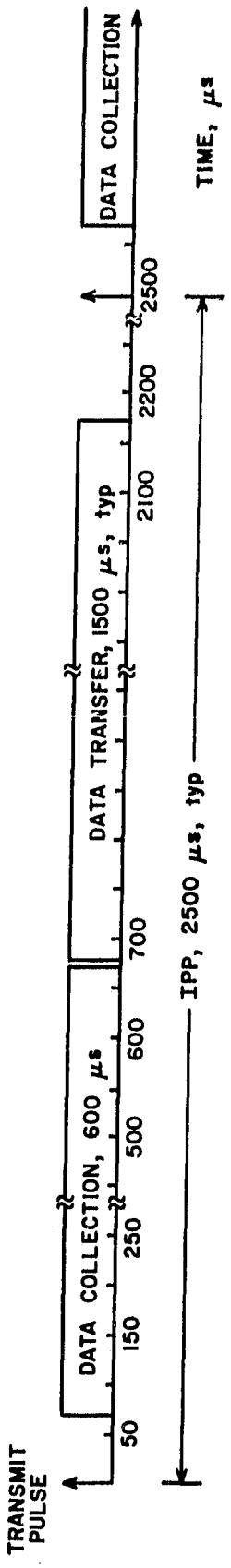

Figure 3.4 Timing of interleaving of data collection and data transfer to main computer.

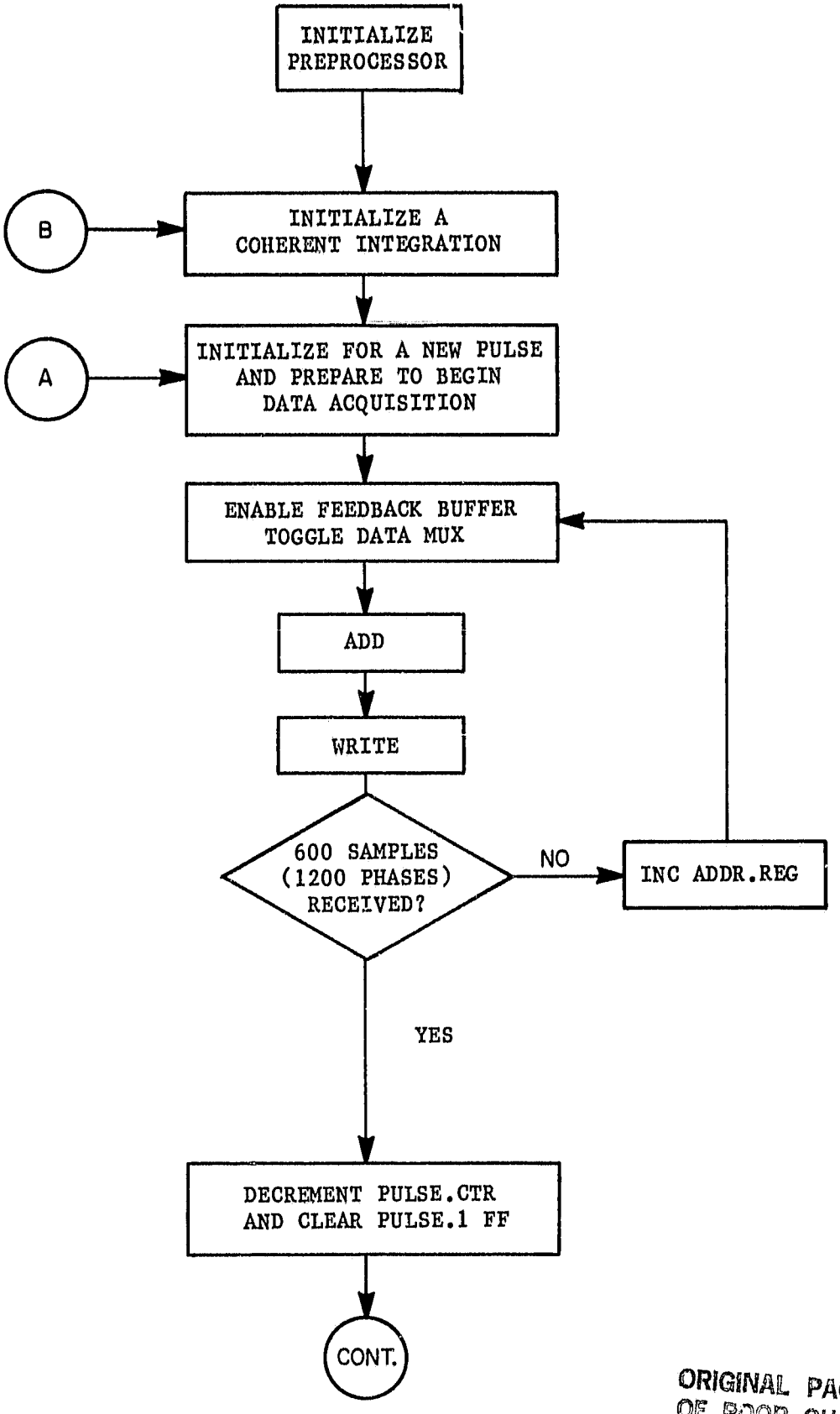

Figure 3.5 Microprogram flowchart

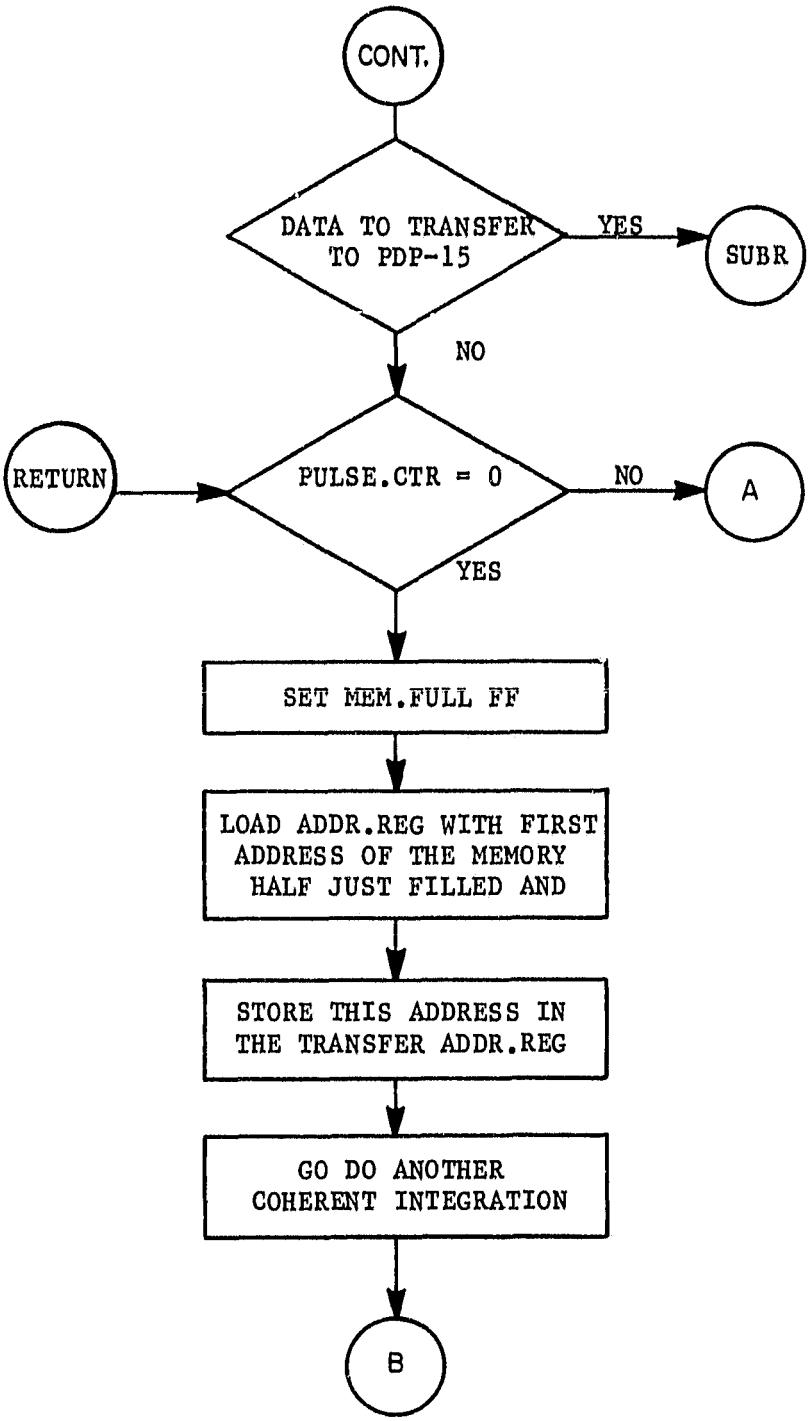

Figure 3.5 (Cont.)

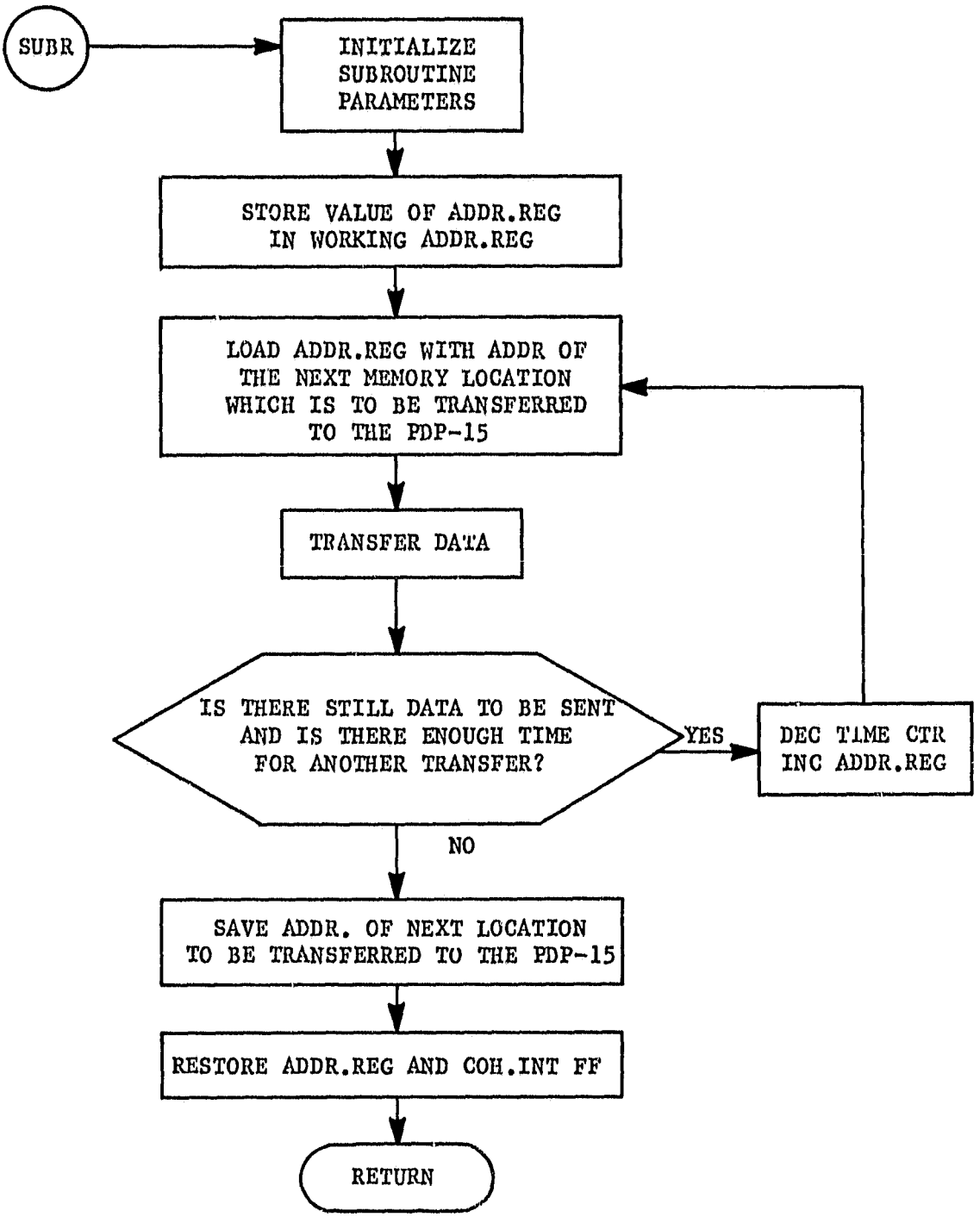

Figure 3.5 (cont.)

μ INSTR. ADDR:		μ INSTR		
BINARY	DECIMAL	MSW	LSW	
0 0000	0	7F	08	CLR MEM.FULL FF
				SET MEM.HALF FF
				SET JMP.INST FF
				SET COH.INT FF
0 0001	1	BF	08	LOAD PULSE.CTR
				SET PULSE.1 FF
				TOGGLE MEM.HALF FF
0 0010	2	FF	48	SET INPUT DATA MUX FF
				LOAD ADDR.REG (with MEM.HALF ADDR)
0 0011	3	3F	48	WAIT
0 0100	4	0F	18	TOGGLE DATA MUX FF
				ENABLE FEED.BUF
0 0101	5	0F	48	LATCH FEED.BUF
0 0110	6	0F	40	MEM.WRITE
0 0111	7	AD	44	JDA on $Z_{\phi\#}$ to address 4 and
				INC ADDR.REG
0 1000	8	AD	44	
0 1001	9	1F	48	DEC. PULSE.CTR
				CLK (HENCE CLR.) PULSE.1 FF
0 1010	10	A7	53	JMF on $Z_{MEM.FULL}$ to address 19
0 1011	11	A7	53	
0 1100	12	AE	42	JCI on Z_{PULSE} to address 2
0 1101	13	AE	42	
0 1110	14	4F	48	SET MEM.FULL FF

Figure 3.6 Microprogram listing

0 1111	15	FF	48	LOAD ADDR.REG (with MEM.HALF ADDR.)
1 0000	16	6F	48	LOAD TRANSFER REG (with ADDR.REG)
1 0001	17	8F	41	JMP to address 1
1 0010	18	8F	41	
1 0011	19	CF	48	LOAD TIME.CTR
1 0100	20	5F	48	WORKING ADDR.REG ← ADDR.REG
1 0101	21	EF	48	ADDR.REG ← TRANSFER ADDR.REG
1 0110	22	2F	48	ISSUE DATA READY FLAG (DRF) DEC TIME.CTR CLR COH.INT FF INC ADDR.REG
1 0111	23	9F	48	NOP
1 1000	24	AB	56	JDT on $Z_{TRANS} \wedge Z_{TIME}$ to address 22
1 1001	25	AB	56	
1 1010	26	6F	48	TRANSFER ADDR.REG ← ADDR.REG
1 1011	27	DF	48	ADDR.REG ← WORKING ADDR.REG SET COH.INT FF
1 1100	28	8F	4C	JMP to address 12
1 1101	29	8F	4C	

Figure 3.6 (cont.)

of the memory address. It determines which half of the memory is used. During initialization, the memory half flip-flop is set. The flip-flop is toggled whenever a new coherent integration begins, thus using alternate memory halves each coherent integration.

For sequential program execution, the jump instruction flip-flop (JMP. INSTR.) is set. Only when a jump is to occur (an unconditional or a conditional jump) is the jump instruction flip-flop cleared.

The coherent integration flip-flop (COH.INT) determines whether the phase counter flag, $Z_{\phi\#}$, or the data transfer complete flag, Z_{TRANS}, is active. While in the main body of the microprogram, as is the case initially, the coherent integration flip-flop is set. The phase counter flag is examined to determine if all 1200 phases of a radar pulse have been sampled and integrated. In the subroutine the coherent integration flip-flop is cleared. Here the data transfer complete flag is active and examined to determine if all 1200 phases of a completed coherent integration have been transferred to the main computer.

Instruction 1 makes the proper initializations each time a new coherent integration is to begin. The pulse one flip-flop is set indicating that it is the first pulse of a coherent integration, hence the ALU passes its A inputs directly to the output.

The pulse counter (PULSE.CTR) is loaded with the number of pulses which are to be added together to form a coherent integration. This number is user selected by eight toggle switches on the front panel. In order to allow enough time for all 1200 data words to be transferred to the main computer at least ten transmit pulses must be coherently integrated (for an interpulse period of 2500 μs).

The memory half flip-flop (MEM.HALF) is clocked. This toggles the

flip-flop so that the other half (i.e., the half not used just previously) of the memory is now addressed.

Instruction 2 makes the proper initializations for a new radar pulse to be received. The input data mux flip-flop is set hence the SELECT input of the data mux (Figure 4.7) is high. The address register is loaded with the address specified by the memory half flip-flop, either 001_{HEX} for the bottom half of memory, or 801_{HEX} for the top half.

Instruction 3, DIS, holds the system clock low for an unspecified amount of time. Microprogram execution is halted while the system clock is inhibited. The DIS instruction also clears the decade counter used to generate a 1 MHz clock.

At this point in the microprogram, all initializations are completed and the preprocessor is ready to start collecting data. In order to be in synchronization with the coherent scatter system, however, the preprocessor waits until it receives a start pulse, START1 (Figure 4.1), from the radar director before program execution continues and the data collection portion of the microprogram is entered.

The 1 MHz clock is used to generate the START CONVERT signal for the ADC, and SAMPLE signal for the sample-and-holds. In order to keep these signals in synchronization with the coherent scatter system, the 1 MHz clock is also disabled during each DIS instruction and restarted when data collection is to begin.

Instruction 4 is the first instruction of a read-add-write loop which is executed each time a data word is acquired. The feedback buffer is enabled and the running sum stored in the memory is read into the buffer. Also the data mux flip-flop is toggled so the data word from the channel not used the previous time through the loop is selected.

Instruction 5 latches the running sum into the feedback buffer. During this instruction cycle, both inputs to the ALU settle and the ALU output becomes valid.

Instruction 6 writes the new running sum, at the ALU outputs, into memory. The previous running sum is overwritten.

Instructions 7 and 8 are a conditional jump. A check is made to see if 1200 phase samples (corresponding to 600 pulses) have been accepted. If the phase counter flag $Z_{\phi\#}$ is high, the microprogram jumps to instruction 4, the beginning of the read-add-write loop, to accept another data word. The address register is incremented to address the next consecutive location in memory. If $Z_{\phi\#}$ is low, then 1200 phases have been received and sequential program execution continues.

Instruction 9 updates the pulse information. The pulse counter is decremented and the pulse one flip-flop is clocked. If previously set (indicating that it was the first pulse of a coherent integration) the pulse one flip-flop will be cleared, if already clear it will remain so.

Instructions 10 and 11 check to see if there is data to transfer to the main computer. If the memory full flip-flop is set ($Z_{MEM.FULL} = 1$), the conditional jump is true and the microprogram jumps to the data transfer subroutine. For $Z_{MEM.FULL} = 0$, there is no data to transfer, hence sequential program execution continues.

Instructions 12 and 13 check whether a coherent integration is complete. If the pulse counter has not been decremented to zero, Z_{PULSE} will be high and the conditional jump will occur. This takes the microprogram back to instruction 2 where initialization for another pulse occurs. When Z_{PULSE} is low, a coherent integration is complete, the conditional jump condition is not met, hence sequential program execution

continues.

Instruction 14 sets the memory full flip-flop. A coherent integration has just been completed, hence there is data to transfer to the main computer.

Instruction 15 loads the address register with the beginning address of the memory half just used.

Instruction 16 stores this address in the transfer address register.

Instruction 17 jumps to instruction 1 where another coherent is initialized.

The microprogram subroutine is executed in the time between data collection (Figure 3.4), its purpose is to transfer the preprocessor data to the main computer, a PDP-15. The computer interface data transfer rate is 75 kHz, (13.2 μs per data word), approximately 16000 μs are required to transfer all the data of one coherent integration. There are approximately 1800 μs from the end of data collection until the next transmit pulse, consequently the subroutine must be entered several times to completely transfer the data. Since the PRF may be varied, the amount of time spend in the subroutine cannot be fixed. Hence, a time counter (TIME.CTR) is loaded with the number of data words to be transferred per subroutine call each time the subroutine is entered. The time counter is decremented each time a DRF is issued so the amount of time spent in the subroutine is approximately (time counter) x 13.2 μs, since the DRF instruction is in a loop which requires 13.2 μs to execute. The user selects the value loaded into the time counter via twelve toggle switches on the front panel.

The subroutine is exited when the time counter flag, Z_{TIME}, goes low or when all 1200 data words have been transferred, indicated by the data transfer flag, Z_{TRANS}, going low. After being flagged, the preprocessor will

require $\leq$1.7 μs of overhead time before it is again ready to begin data acquisition.

The subroutine is lines 19 through 29 of the microprogram. Instruction 19 loads the time counter.

Instruction 20 saves the present address register value in the working address register.

Instruction 21 loads the address register with the address stored in the transfer address register. This is the address of the next data word to be transferred.

Instructions 22 and 23 initiate the data transfer. The data ready flag, DRF, is issued to the Hewlett-Packard HP5601A interface which handles the data transfer synchronization with the PDP-15. The preprocessor's system clock is disabled for 12.8 μs to allow the data transfer to occur. Also, the coherent integration flip-flop is cleared, thus enabling the Z_{TRANS} flag and disabling the $Z_{\phi\#}$ flag, and the time counter is decremented. When the data transfer is complete the address register is incremented and the system clock is restarted.

Instructions 24 and 25 are a conditional jump. If all the data has not been transferred, $Z_{TRANS} = 1$, and if there is still enough time to transfer another data word, $Z_{TIME} = 1$, then the program jumps back to instruction 22. If $Z_{TRANS} = 0$ or $Z_{TIME} = 0$, sequential program execution continues.

In preparing to return to the main program, instruction 26 stores the address of the next data word to be transferred in the transfer address register.

Instruction 27 reloads the address register with the value it contained when the subroutine was entered. Also the coherent integration flip-flop is set so that $Z_{\phi\#}$ is enabled and Z_{TRANS} is disabled.
Instructions 28 and 29 jump to the exit point of the main program.

4. THE ELECTRONIC CIRCUITS

4.1 *Introduction*

The preprocessor was specified to sample and store, in the form of a coherent integration, both channels of the coherent scatter receiver once every μsec. Using a single memory and adder requires a memory read, add and memory write cycle to be performed twice for each radar pulse sampled, i.e. twice every μsecond. An instruction cycle of 100 nsec performs the two read-add-write cycles in 600 nsec, leaving 400 nsec for overhead to keep within the 1 μsec limit. Executing one instruction cycle each clock period led to the selection of a 10 MHz system clock.

All digital circuity is TTL logic, Schottky series is used where available in order to decrease propagation delay times. Logic signal propagation delay was the primary consideration when designing the control circuitry. The number of gate levels was minimized in order to insure that each instruction is executed within the 100 nsec instruction cycle.

4.2 *Preprocessor Clock*

The master clock (MSTR CLK) is provided by a Motorola K1091A 10.000 MHz oscillator. This is the clock signal from which all other timing signals are generated as shown in Figure 4.1.

The MSTR CLK can be synchronously gated on and off by microinstructions DIS and DRF and by the single step circuitry. The gated master clock output is the system clock (SYS CLK). The system clock and logical combinations of the delayed system clock provide the timing signals for an instruction fetch, Figure 4.2. A 74H04 hex inverter is used to delay the system clock. The 74H04 has a typical propagation delay time of 6 nsec hence by combining the appropriately delayed and/or inverted system clock phases the required timing signals are generated.

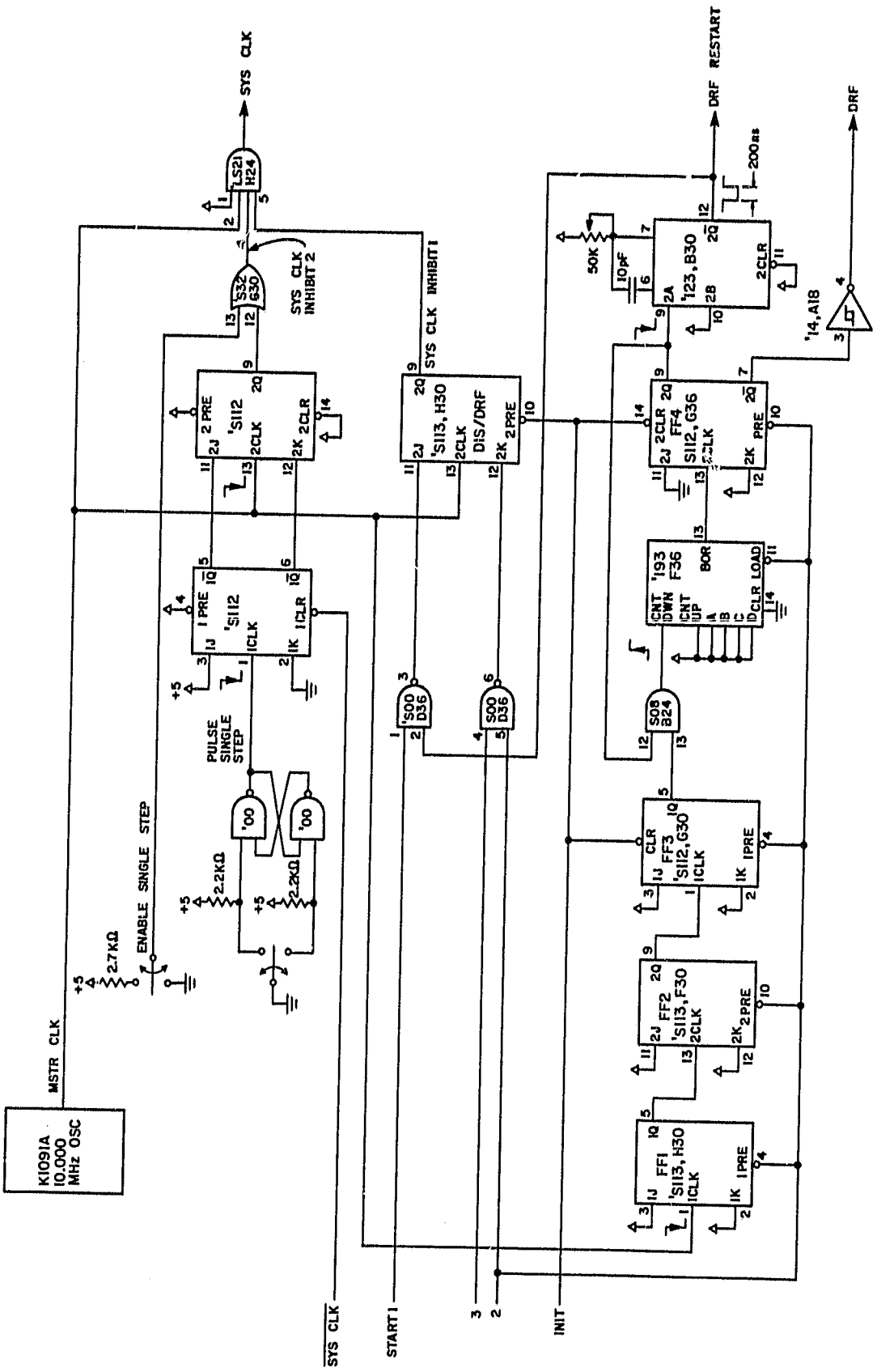

Figure 4.1 System clock generation and control circuitry. The flip-flops, counter, and monostable along the bottom of the figure are the data-ready flag circuitry.

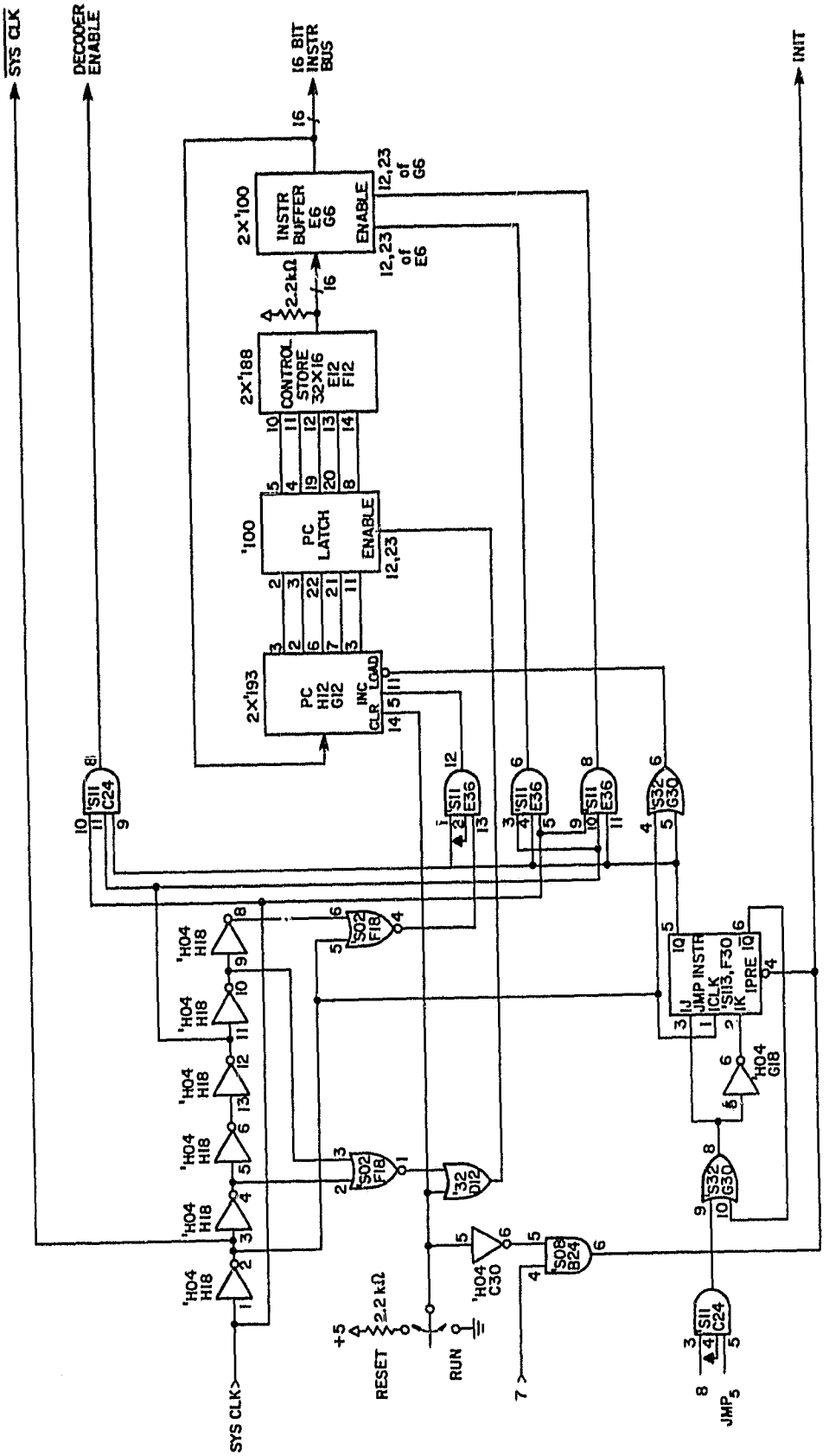

Figure 4.2 The instruction fetch circuitry.

4.3 *Instruction Fetch*

An instruction fetch requires incrementing the program counter (PC) to the next address, for sequential program execution, or loading the PC with the desired address in the case of a jump. The program counter latch (PC LATCH) then holds the new address. The contents of this address, the microinstruction, is read by the control store (CS). The microinstruction is then held by the instruction buffer (INSTR BUFFER).

Figure 4.3 shows the timing of the instruction fetch signals. Increment program counter (INC PC) is a 30 ns pulse which initiates a new instruction. When the program counter has settled to its new value (47 ns MAX) the program counter enable pulse (ENABLE PC LATCH) latches the new program counter value. The next 30 ns (40 ns MAX) are used by the control store to read the new microinstruction which is then latched by the instruction buffer enable pulse (ENABLE INSTR BUFFER). From the time the program counter is incremented until the corresponding instruction is read and latched in the instruction buffer (INSTR BUFFER) is $\simeq$ 150 ns. However, latching both the program counter and the microinstruction allows the instruction cycle to be pipelined. This means the program counter is incremented before the present program counter address, stored in the program counter latch, has been read by the control store and the microinstruction latched. Hence a new control store address is read and the microinstruction latched every 100 ns.

4.4 *The Microinstruction Word*

There are twenty-one instructions in the preprocessor's instruction set, Table 4.1. The instructions are decoded from the sixteen bit microinstruction word. The microinstruction word consists of three fields; a four bit opcode field, a four bit conditional jump instruction field, and an

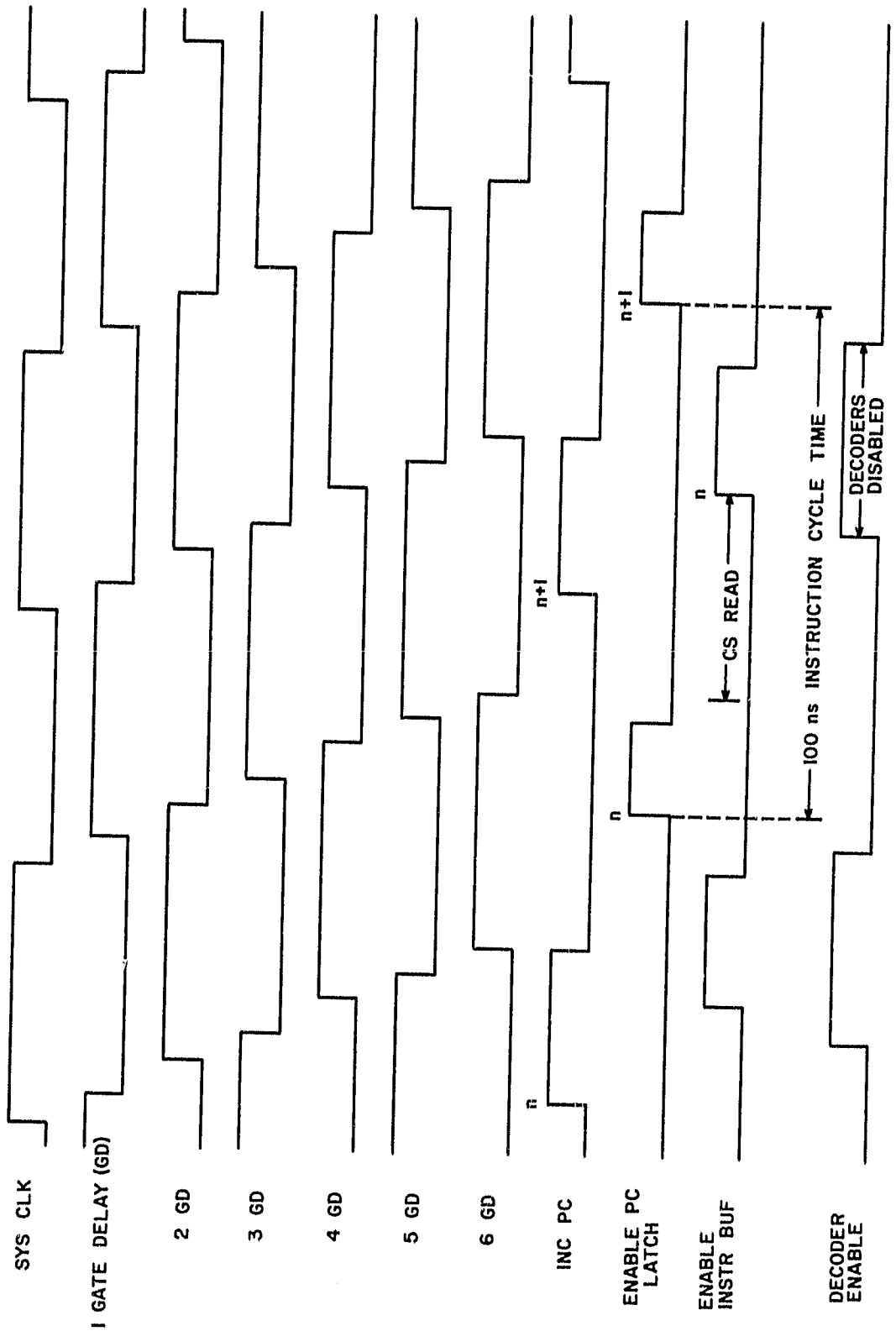

Figure 4.3 Timing diagram for an instruction fetch. The letter "n" refers to any instruction address.

Table 4.1 Microinstruction codes, mnemonics, and functions.

MICROINSTRUCTION CODE		INSTRUCTION	
HEX	BINARY, BITS 15-0	MNEMONIC	FUNCTION
0F18	0000111100011000	SAI	Set ALU inputs: Toggle data mux flip-flop Enable the feedback buffer
0F40	0000111101000000	MWR	Memory write
0F48	0000111101001000	WAC	Wait for addition complete
1F48	0001111101001000	DPC	Decrement the pulse counter
2F48	0010111101001000	DRF	Issue the data-ready flag
3F48	0011111101001000	DIS	Disable the system clock
4F48	0100111101001000	SMF	Set the memory full flip-flop
5F48	0101111101001000	WWR	Write the contents of the address register to the working address register
6F48	0110111101001000	WTR	Write the contents of the address register to the transfer address register
7F08	0111111100001000	INIT	Initialize the preprocessor
8F40+A	100011110l0XXXXX	JMP	Unconditional jump to address A
9F48	1001111101001000	NOP	No operation
A740+A	101001110l0XXXXX	JMF	Jump to A on memory full
AB40+A	101010110l0XXXXX	JDT	Jump to A on data transfer
AD40+A	101011010l0XXXXX	JDA	Jump to A on data acquisition
AE40+A	101011100l0XXXXX	JCI	Jump to A to continue collecting data for the present coherent integration
BF08	1011111100001000	ICI	Initialize the preprocessor to begin a new coherent integration
CF48	1100111101001000	LTC	Load the time counter
DF48	1101111101001000	RWR	Read the working address register contents into the address register
EF48	1110111101001000	RTR	Read the transfer address register contents into the address register
FF48	1111111101001000	LAR	Load the address register

eight bit auxiliary information field. The opcode field is bits 15-12 of the microinstruction. Two 74S138 3-to-8 line decoders decode bits 14-12 (Figure 4.4). Bit 15 enables one decoder and disables the other.

For a conditional jump bits 11-8, the conditional jump instruction field, are examined to decode the proper microinstruction. One of bits 11-8 is low, sele .ing JMF, JDT, JDA, or JCI respectively. For the conditional and unconditional jump bits 5-0 carry the jump address.

During the data acquisition instruction bits 6, 4, and 3, are examined. In this case one of instructions SAI, MWR, or WAC, is decoded as shown in Table 4.1.

4.5 *Microinstruction Decoding*

Figure 4.4 shows the two 74S138 3-to-8 line decoders which decode the opcode field of the microinstruction word. Each decoder has two active-low and one active-high enable inputs; G2A, G2B, and G1 respectively. When disabled all output lines are at a logic high, in the enabled state the selected output line goes low. The DECODER ENABLE line is to insure that the decoders are disabled while the instruction buffer is enabled, as shown in Fig. 4.3. This prevents the decoders from issuing instructions while the instruction bus word is not valid. The most significant bit, MSB, of the instruction word is used to select one of the two decoders. When bit 15 is high decoder 1 is enabled and decoder 2 is disabled; for bit 15 low decoder 1 is disabled and decoder 2 enabled. The enabled decoder reads bits 14, 13, and 12 to select the appropriate output, 0 through F (the decoder outputs are labeled with the hexidecimal representation of the microinstruction opcode field, 0-F).

Six OR gates and one AND gate are used to determine the decoding of the four conditional jump instructions and SAI, MWR, and WAC. For a conditional

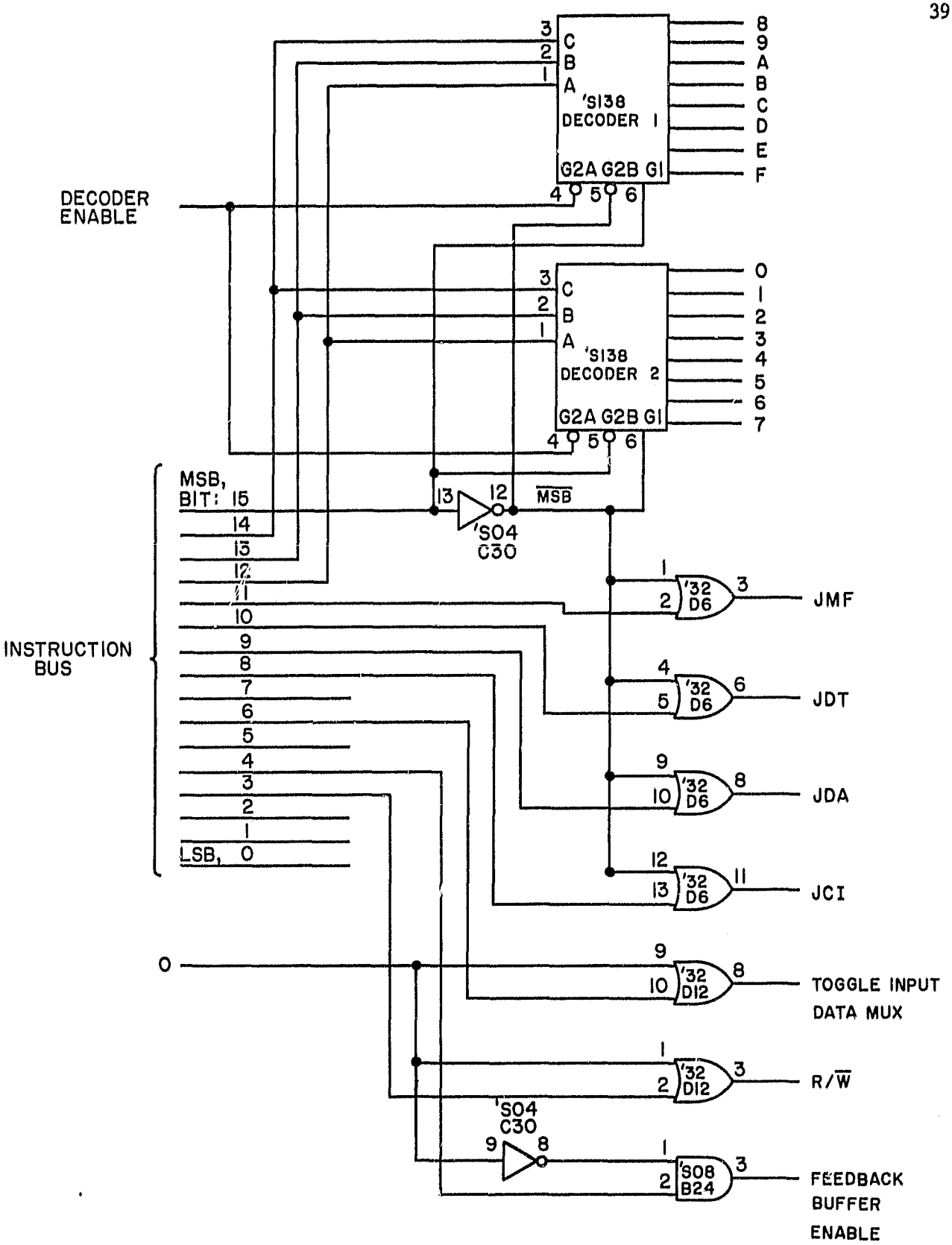

Figure 4.4 Microinstruction decoding circuitry.

jump MSB is low and ORed with each of the conditional jump instruction field bits. One of bits 11-8 low takes the selected conditional jump line low. During a data acquisition instruction decoder output 0 is low and one of three instructions are possible, as selected by bits 6, 4, and 3. Bits 6 and 3 are active low, bit 4 is active high. During the SAI instruction bit 6 is low, toggling the data mux flip-flop, and bit 4 is high, enabling the feedback buffer, bit 3 is inactive. For a memory write, MWR, bit 3 is low and bits 6 and 4 are inactive. A wait for addition complete, WAC, has all three bits inactive.

4.6 *Jump Instruction Circuitry*

4.6.1 *Jump Instruction Decoding Circuitry.* The preprocessor executes conditional and unconditional jump instructions. The unconditional jump microinstruction causes a jump in the microprogram to occur regardless of the condition of any flags or register. The conditional jumps require a particular flag to be true in order for the jump to be executed otherwise sequential program execution continues. The unconditional jump is straightforward, when this microinstruction is decoded, output 8 goes low. This immediately initiates the jump instruction execution sequence (Section 4.6.2).

When a conditional jump occurs, JMP_5 goes low. JMP_5 is the output of a 74S64 (4-2-3-2 AND-OR-INVERT gates); Table 4.2 shows JMP_5 for the possible conditional jump conditions. A flag is tested by each conditional jump instruction to determine if the jump is to be executed. JMF jumps on the condition that there are data to transfer from the preprocessor memory to the main computer. This condition is indicated by $Z_{Mem.Full}$ high. As Figure 4.5 shows, $Z_{Mem.Full}$ is the output of the memory full flip-flop (MEM.FULL). This flip-flop is cleared by INIT and set by SMF.

TABLE 4.2 Signal JMP_5 for possible conditional jump states.
X means irrelevant.

JMF	JDT	JDA	JCI	$Z_{MEM.\ FULL}$	Z_{TRANS} V Z_{TIME}	$Z_{\phi\#}$	Z_{PULSE}	JMP_5
0	1	1	1	1	X	X	X	0
0	1	1	1	0	X	X	X	1
1	0	1	1	X	1	X	X	0
1	0	1	1	X	0	X	X	1
1	1	0	1	X	X	1	X	0
1	1	0	1	X	X	0	X	1
1	1	1	0	X	X	X	1	0
1	1	1	0	X	X	X	0	1

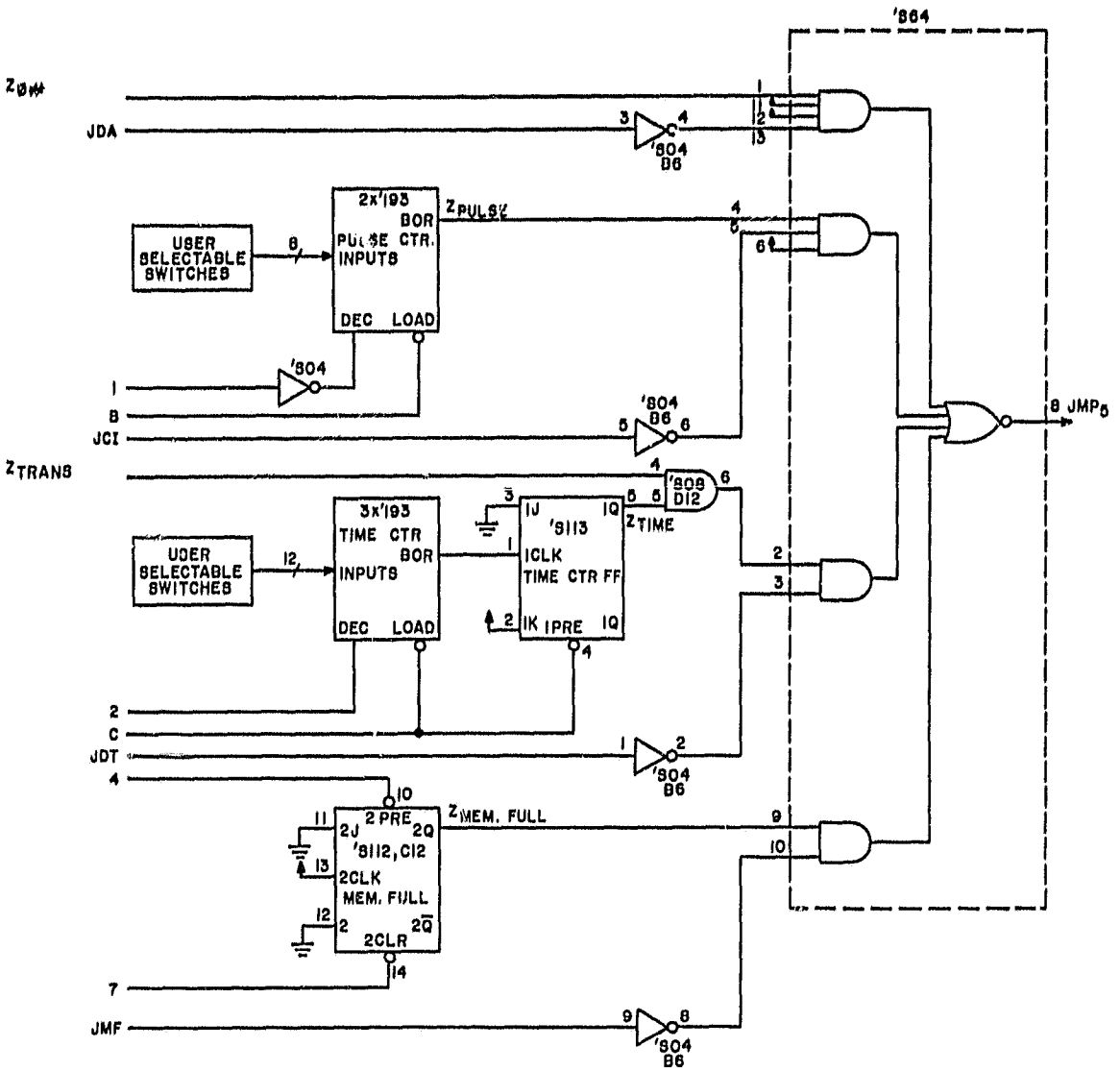

Figure 4.5 Conditional jump instruction circuitry.

The JDT instruction examines two flags, Z_{TRANS} and Z_{TIME}. Both flags must be high to satisfy the jump condition. Z_{TRANS} is low only when the memory address is 1201_{10} and the coherent integration flip-flop (COH.INT) is cleared. This flag [illegible]dicates that the microprogram is in the data transfer subroutine (coherent integration flip-flop clear) and all 1200_{10} words of the preprocessor memory have been transferred (lower 11 bits of memory address set to 1201_{10}). Z_{TIME} low flags the microprogram data transfer subroutine that there is not enough time (before the next radar pulse) to transfer another word to the main computer. Z_{TIME} is the output of a flip-flop which is clocked by the time counter (TIME.CTR). When the data transfer subroutine is entered instruction LTC loads the time counter with the user selected number of data transfers to occur per subroutine and presets the time counter flip-flop (setting Z_{TIME} high). When the DRF instruction decrements the time counter to zero the borrow line goes low, clocking the time counter flip-flop, hence resetting the output, Z_{TIME}.

The condition for JDA is that the phase counter flag, $Z_{\phi\#}$ be high. $Z_{\phi\#}$ operates similarly to Z_{TRANS}; it examines the lower 11 bits of the memory address for 1200_{10} (JDA occurs before incrementing memory address, hence $Z_{\phi\#}$ decodes 1200_{10}, whereas JDT occurs after incrementing the memory address, so it decodes 1201_{10}), but will not go low unless the coherent integration flip-flop is set by instructions INIT and RWR and hence is set whenever the main body of the microprogram is being executed. The purpose of $Z_{\phi\#}$ is to flag when 1200 data words (600 altitudes times two phases per altitude) have been acquired. The $Z_{\phi\#}$ flag low prevents the microprogram from jumping back to the beginning of the read-add-write loop when 1200 data words have been acquired.

The JCI instruction returns the preprocessor to the beginning of the

data acquisition loop provided the Z_{PULSE} flag is high. When the preprocessor is initialized to begin a new coherent integration by instruction ICI, the pulse counter (PULSE.CTR) is loaded with the number of radar pulses whose return samples are to be summed. After the 1200_{10} samples from a radar pulse are received and stored instruction DPC decrements the pulse counter. When the selected number of radar pulses are counted the pulse counter borrow line, Z_{PULSE}, goes low. Z_{PULSE} low indicates that the requested number of radar pulses has been coherently integrated so the microprogram should not return to collect samples for the present coherent integration.

4.6.2 *Jump Instruction Execution Circuitry.* In order to execute a jump instruction two instruction cycles are required, Figure 4.6. On the first instruction cycle the program counter latch (PC LATCH) is loaded with the jump address in the jump instruction. During the second instruction cycle the increment program counter line is inhibited and the load program counter line goes low, loading the program counter with the address on the lower five bits of the instruction word. Also during this instruction cycle the enable instruction buffer and DECODER ENABLE lines remain inhibited (low) so the jump instruction remains decoded during all of both instruction cycles. On the next instruction cycle the jump address is latched and the new instruction is decoded and latched, hence normal operation resumes.

Figure 4.2 shows the negative edge triggered flip-flop, JMP INSTR flip-flop, responsible for gating the normal instruction cycle signals to the program counter and instruction buffer or selecting the special jump instruction sequence. Normally the flip flop inputs are in a set configuration, J=1 and K=0, so the Q output is high, thus holding the program counter load high and gating the increment program counter, enable instruction buf-

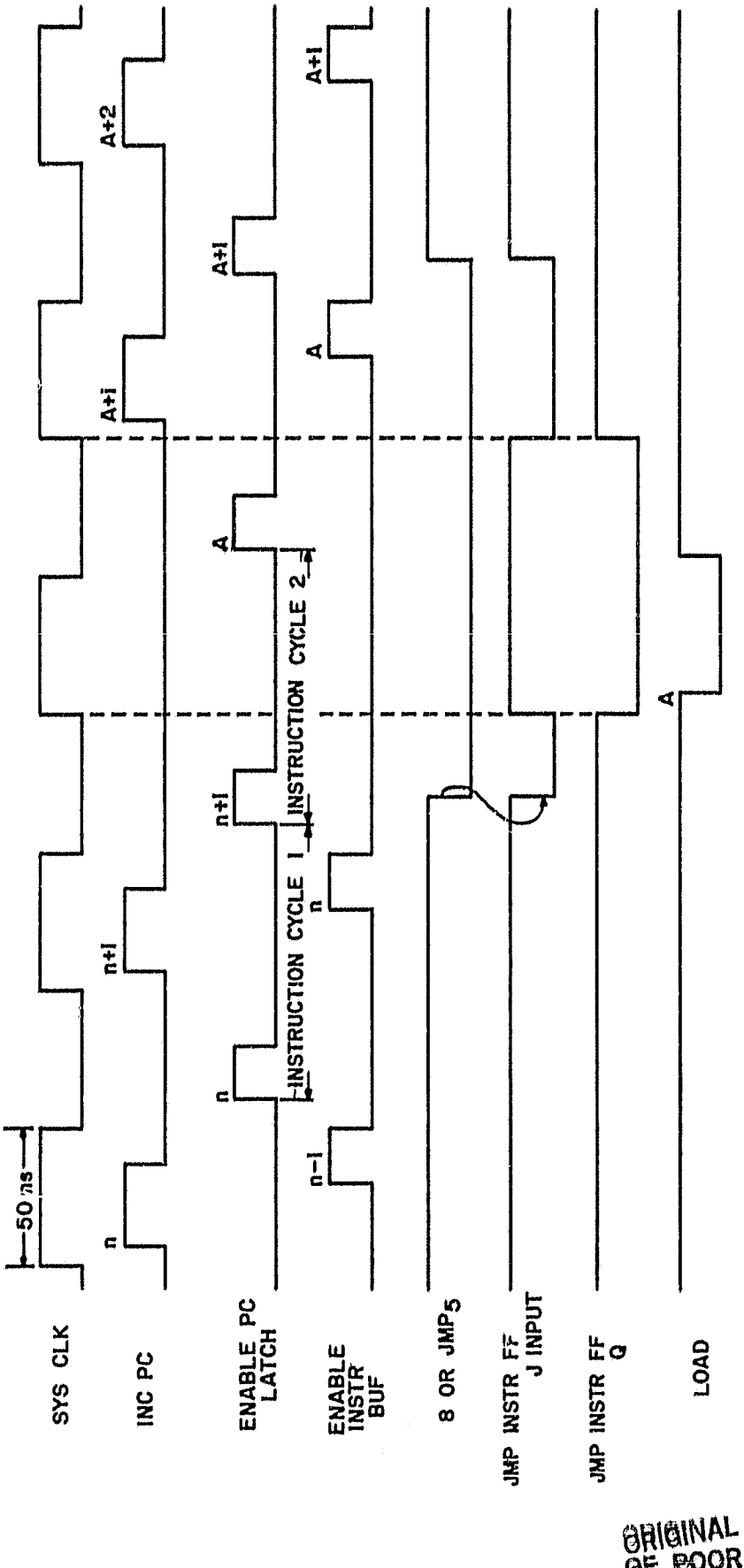

Figure 4.6 Jump instruction timing diagram. "A" is the jump address.

fer, and DECODER ENABLE signals through the AND gates. If a conditional or unconditional jump is to be executed (instruction *n* in Figure 4.6) the JMP or decoder output 0 line respectively will go low thus resetting the jump instruction flip-flop. At the beginning of the next instruction cycle the flip-flop is clocked and the Q output goes low thus holding increment program counter, enable instruction buffer, and DECODER ENABLE lines low and enabling the load program counter line. Now instead of being incremented the program counter is loaded with the address, A in Figure 4.6, on its input. Jump instruction flip-flop $\overline{Q}$ is now high hence the following instruction cycle sets the flip-flop and a normal instruction cycle occurs. Since a new instruction is now in the instruction buffer the JMP_5 or decoder output 0 line will go high (assuming the new instruction is not another jump command) and the flip-flop remains in the set mode.

4.7 *Data Acquisition Circuits*

Figure 4.7 is a block diagram of the data acquisition circuits. There are two identical channels, the pin level circuit diagram of one channel is shown in Figure 4.8. There are three high speed analog devices in each channel; the Datel-Intersil SHM-HU Sample-Hold (S/H), the National Semiconductor LH0033 Fast Buffer Amplifier, and the Micro Networks MN5101 8-bit ADC. Due to the wide bandwidths of these devices, each channel is on a separate board to prevent capacitive coupling. Particular care is taken with the board layout and construction in order to insure optimum operation.

As recommended each power supply of the S/H is bypassed with a 0.1 μF capacitor. In addition, a 100 pF capacitor is on each BIAS2 pin to help prevent oscillations. A 33 pF polystyrene capacitor is added on the output. This increases the S/H hold capacitor value which in turn increases the acquisition time, but decreases the hold mode droop and sample-to-hold

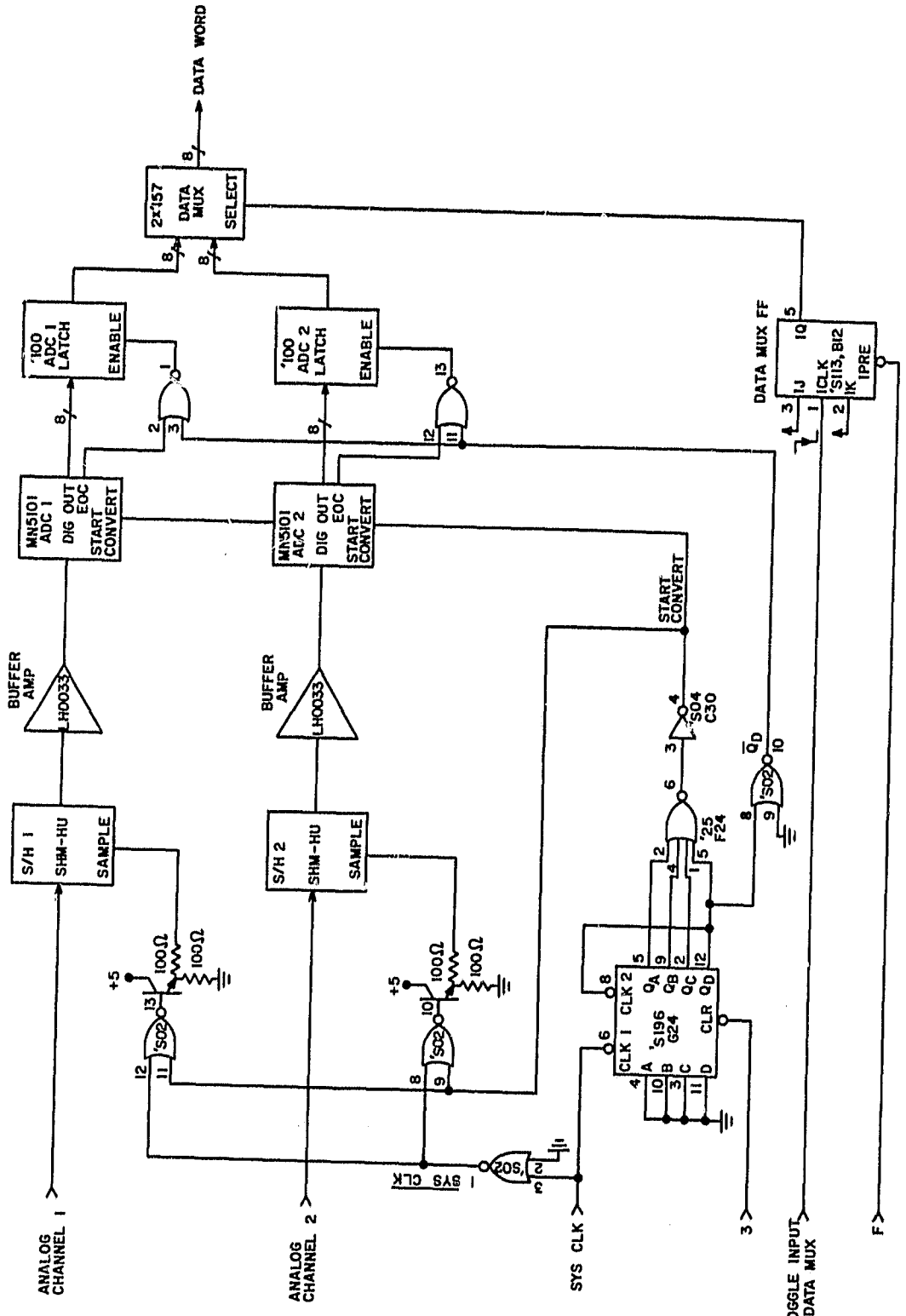

Figure 4.7 Data acquisition circuit block diagram.

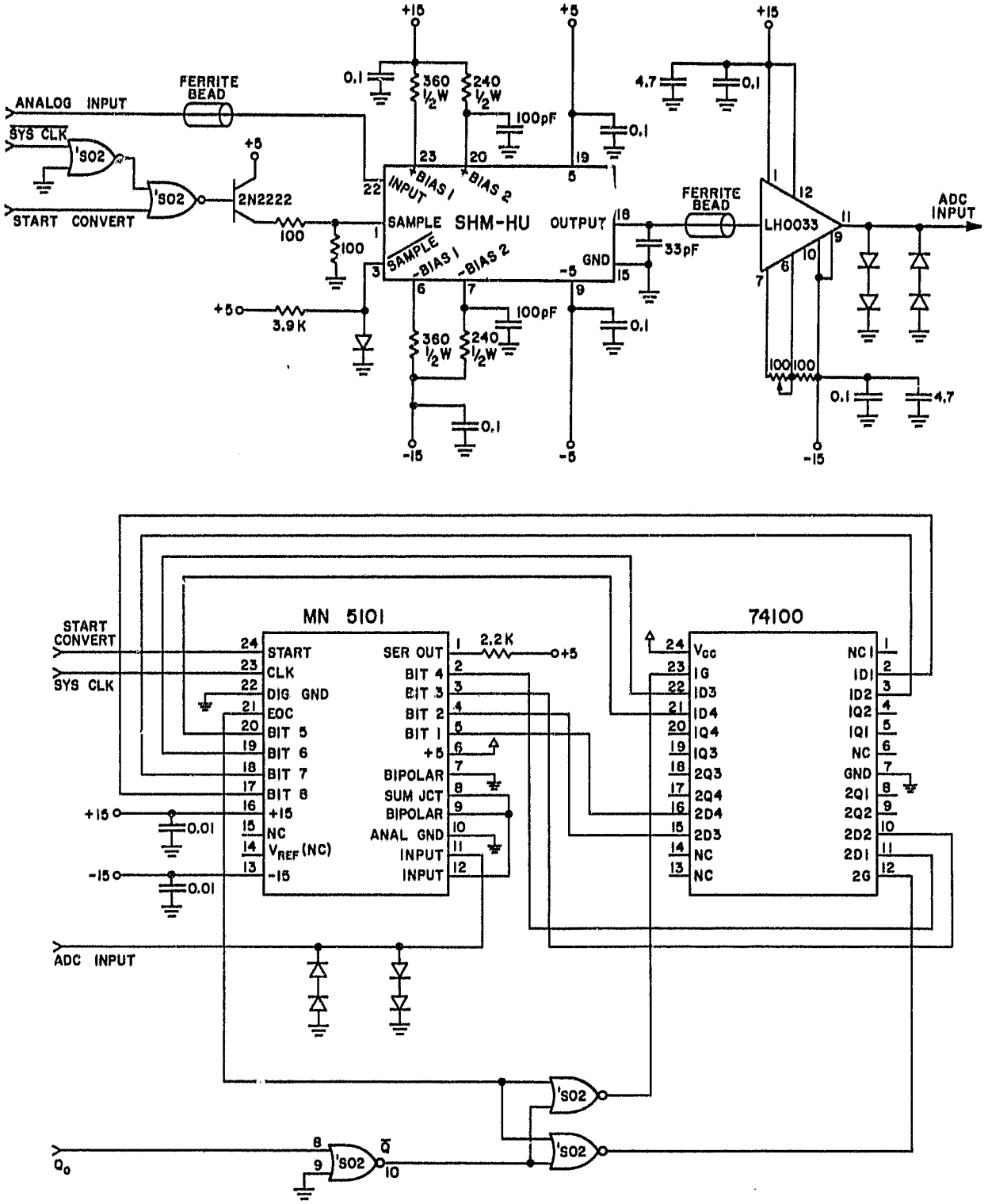

Figure 4.8 Data acquisition circuit pin diagram. All resistors are in ohms, all capacitors are in μF except where noted. The 33 pF capacitor is a polystyrene type. All diodes are 1N270. The 74100 (ADC latch) outputs, pins 1Q1-1Q4 and 2Q1-2Q4 go to the printed circuit board and edge fingers, fingers C-L for channel 1 and fingers N-W for channel 2 (see Figures I.4 and I.5).

offset error. The ferrite beads on the input and output lines act as high frequency chokes; low frequencies (below ~50 MHz) on the lines are unaffected but high frequencies which appear as ripple on the output signal are attenuated.

The National Semiconductor LH0033 Fast Buffer Amplifier (FBA) is required by the sample-hold. The S/H has no output amplifier hence an external voltage follower current amp is necessary. The FBA has a maximum input bias current of 5 nA and is capable of supplying 10 mA into a 1KΩ load at a slew rate of 1500 V/μs. Each power supply is bypassed with a 0.1 μF ceramic disc capacitor in parallel with a 4.7 μF solid tantalum capacitor to prevent oscillation. A 1.5 KΩ resistor to ground is mounted on each board next to the output of the buffer amp. This resistor can be used as a dummy load when testing the S/H FBA combination without the ADC installed. The FBA is not short circuit proof; also running it with no load may cause destructive oscillation.

The ADC has nine possible input voltage ranges, selected by properly connecting the bipolar, summing junction, analog ground and analog input pins. The configuration used converts analog inputs between ±2.5 volts, see Table 4.3. The FBA has an output voltage swing equal to its power supply voltages, ±15V; diodes on the ADC analog input line will clamp the input at ±1.3V. This is to protect the ADC in case the sample-hold or buffer amp should fail. It is not necessary to allow the ADC input to vary over its full range, ±2.5V, because the phase detector output, which is the S/H analog input, does not exceed ±1.0V.

Timing signals for data acquisition are shown in Figure 4.9. The system clock clocks a 74S196 decade counter, the outputs of which are ORed together to form the ADC START CONVERT signal. In order to reset the ADC,

Table 4.3 Output coding for the Micro Networks MN5101 analog to digital converter

ANALOG INPUT ±2.5V	DIGITAL OUTPUT MSB LSB
+2.5	0000 0000
+2.481	0000 0001
+0.019	0111 1111
0.000	1000 0000
-0.019	1000 0001
-2.461	1111 1110
-2.481	1111 1111

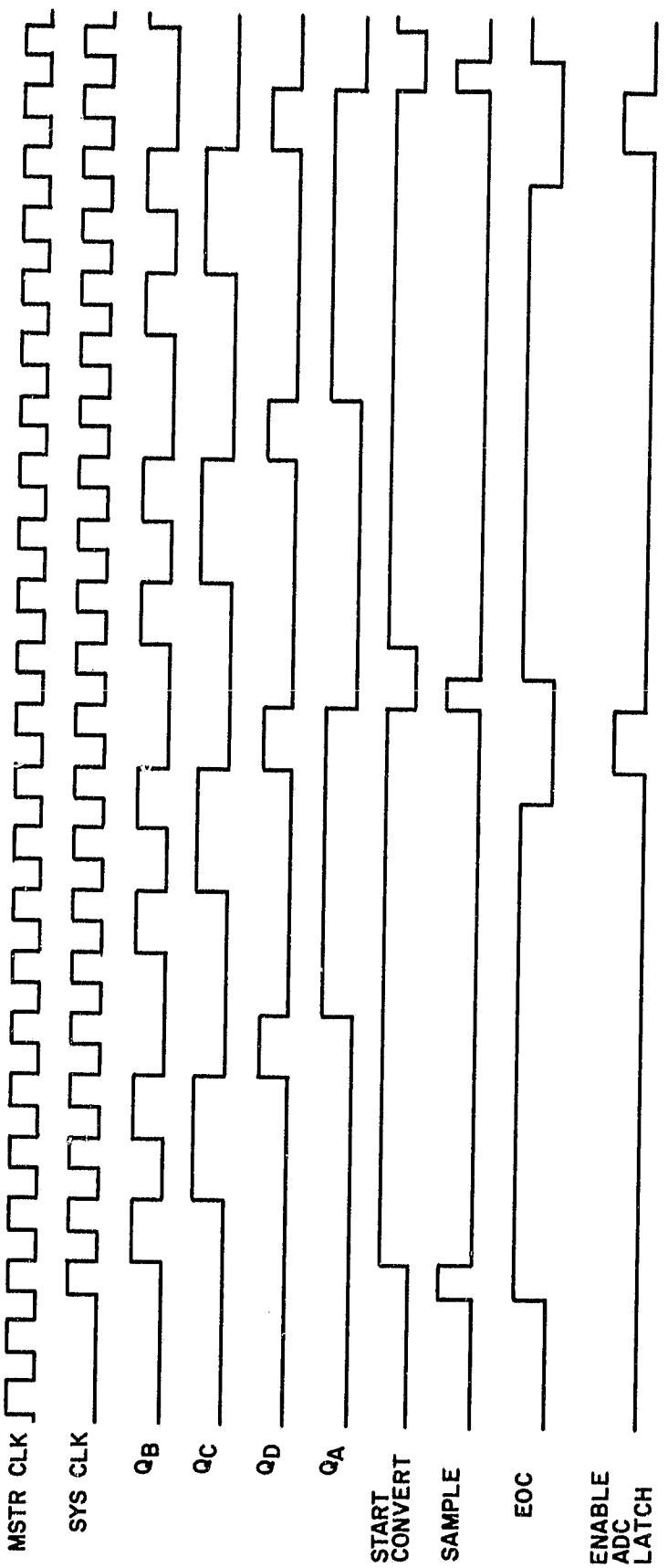

Figure 4.9 Data acquisition circuit timing diagram.

the START CONVERT line must be held low during a low to high transition of the ADC clock input and the START CONVERT line must be low for a minimum of 25 ns prior to this clock transition. Conversion begins on the next rising edge of the clock signal.

The S/H samples when the SAMPLE line is high. The SAMPLE line, START CONVERT NORed with the inverted SYS.CLK, is high for 50 ns during the time the ADC is being reset. The S/H has a 25-ns acquisition time, hence a 50-ns sample pulse will assure a correct sample value.

The ADC latch enable goes high for one clock period during the time the EOC (End Of Convert) line is low. This latches the valid conversion. The ADC latch then holds the DATA WORD for one acquisition cycle, 1 ns for processing while another conversion takes place.

4.8 *The Read-Add-Write Loop*

Figures 4.7, 4.10 and 4.11 show the logic used to implement the read-add-write loop instructions. The first instruction of this loop, SAI, reads a word from memory and sets it up at the FEEDBACK BUFFER inputs while the enable feedback buffer line is high, and also toggles the data mux flip-flop. By toggling the data mux flip-flop each time through the read-add-write loop the data mux alternately selects the two input channels. On the second instruction, WAC, the enable feedback buffer line goes low, latching in the memory word, the ALU B inputs are valid within 30 ns. The ALU completes the addition with 70 ns. The third instruction, MWR, takes the R/W line low, writing the ALU output into the memory. When the JDA condition is true the microprogram jumps to the beginning of the read-add-write loop and increments the address register, see Fig. 4.13.

The function implemented by the ALU is assigned by the pulse one flip-flop, (PULSE.1 FF), Fig 4.10. When the microprogram initializes the pre-

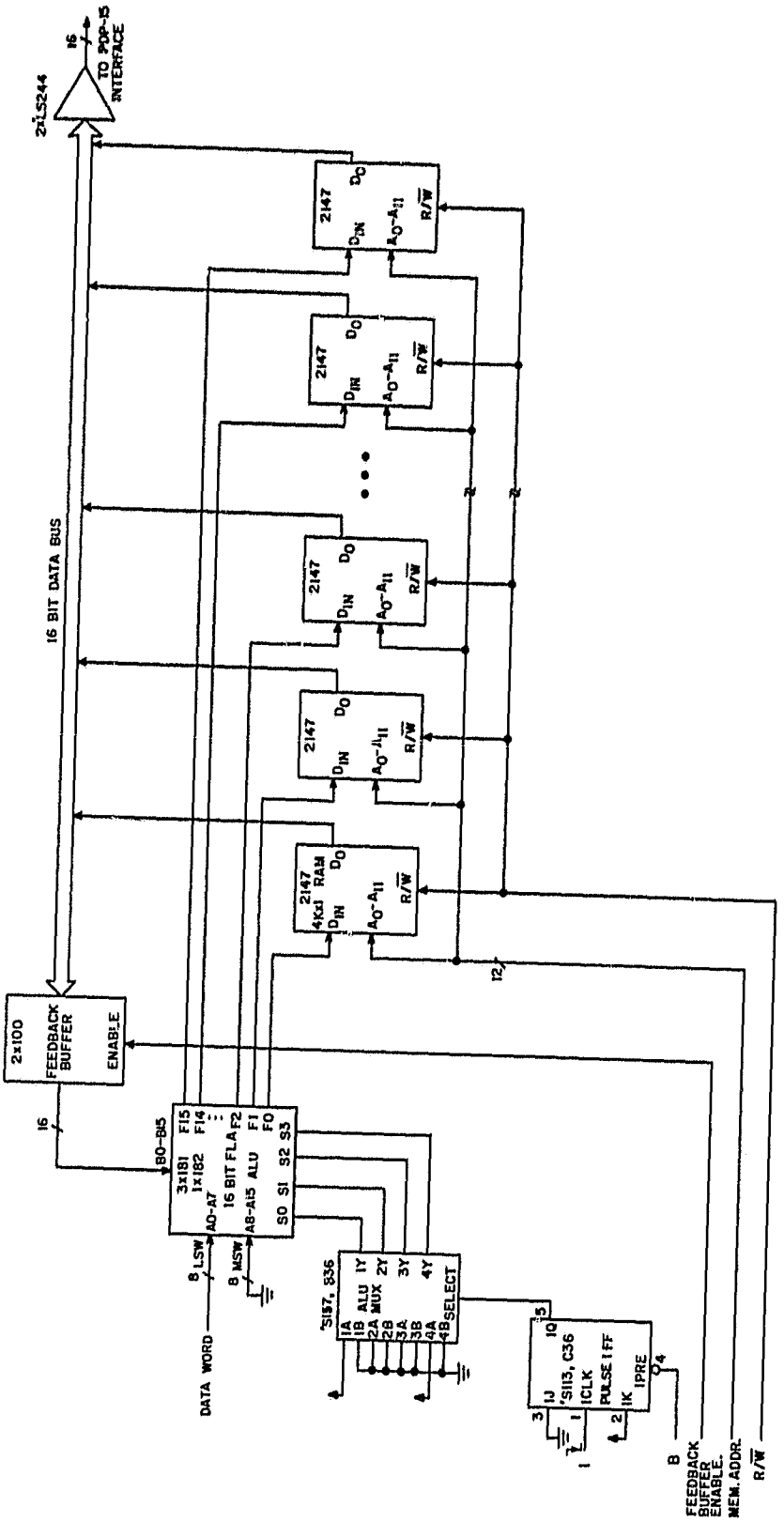

Figure 4.10 ALU and memory block diagram.

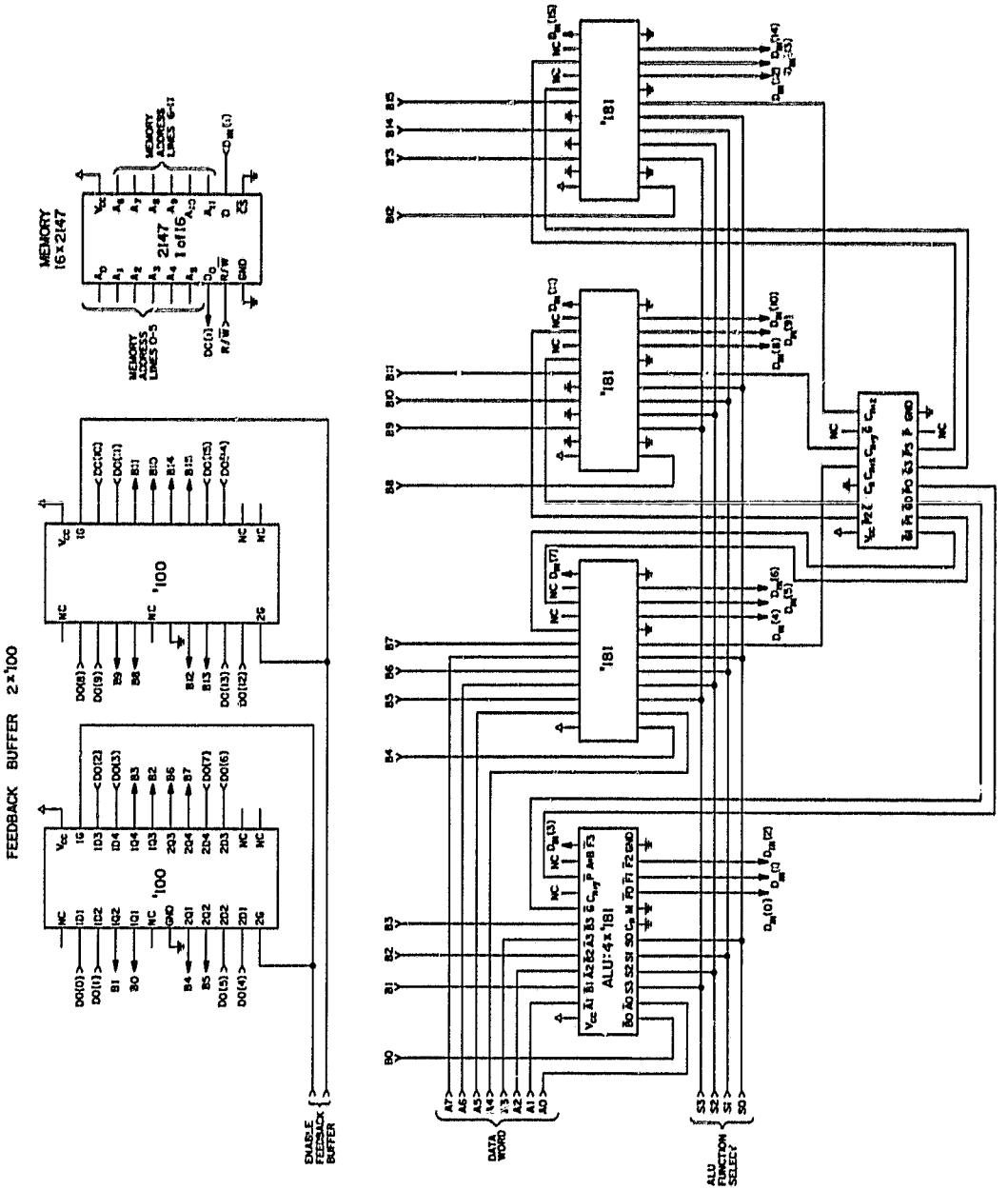

Figure 4.11 ALU and memory pin diagram.

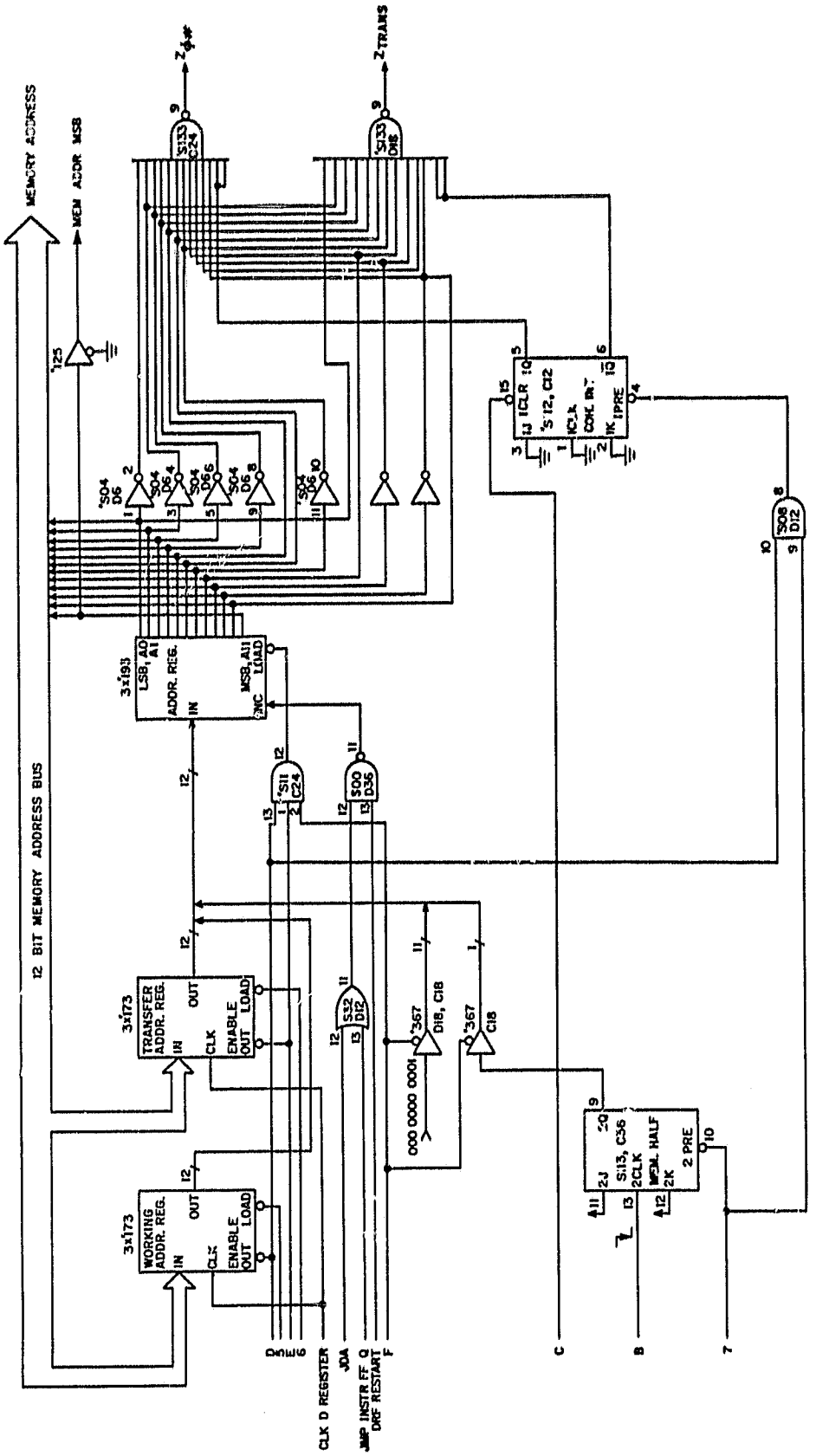

Figure 4.13 Address register and memory address bus.

processor for a new coherent integration, instruction ICI, the pulse one flip-flop is set. With pulse one flip-flop set the ALU MUX selects its B inputs, thus the function select inputs, S0-S3 of the ALU are all low. The ALU function implemented is simply set the output F, equal to the A input. Hence during the first pulse the ALU allows the DATA WORD to be directly written to memory.

After all the samples from a radar pulse are taken, instruction DPC clocks the pulse one flip-flop, resetting its Q output. The ALU now selects its A inputs hence setting up a 1001 at the ALU function select inputs S0, S1, S2 and S3 respectively. The ALU now adds its A and B inputs.

4.9 *DIS and DRF Circuits*

There are two instructions which require the preprocessor to temporarily stop program execution, DIS and DRF. The DIS instruction disables the system clock until a signal from the radar director, PULSE1, is received. The DRF disables the system clock for 12 μs, after which time the preprocessor restarts itself. Both instructions operate in a similar manner, they cause the disable/data ready flag (DIS/DRF) flip-flop, initially preset, to take the SYS CLK INHIBIT1 line low (Figure 4.1).

A DIS instruction takes the DIS/DRF flip-flop K input high (Figure 4.12a). The next falling edge of the master clock then clocks the DIS/DRF flip-flop, this clears the flip-flop hence SYS CLK INHIBIT1 goes low. The system clock will now remain low until the SYS CLK INHIBIT1 line goes high. Restarting the system clock is initiated by a 5 μs low pulse on START1, issued by the radar director. (START1 must remain low for a minimum of 150 ns to insure that the clock is restarted). START1 low takes the DIS/DRF flip-flop J input high, the flip-flop is now in the set mode so SYS CLK INHIBIT1 is clocked high by the master clock and the system clock is re-

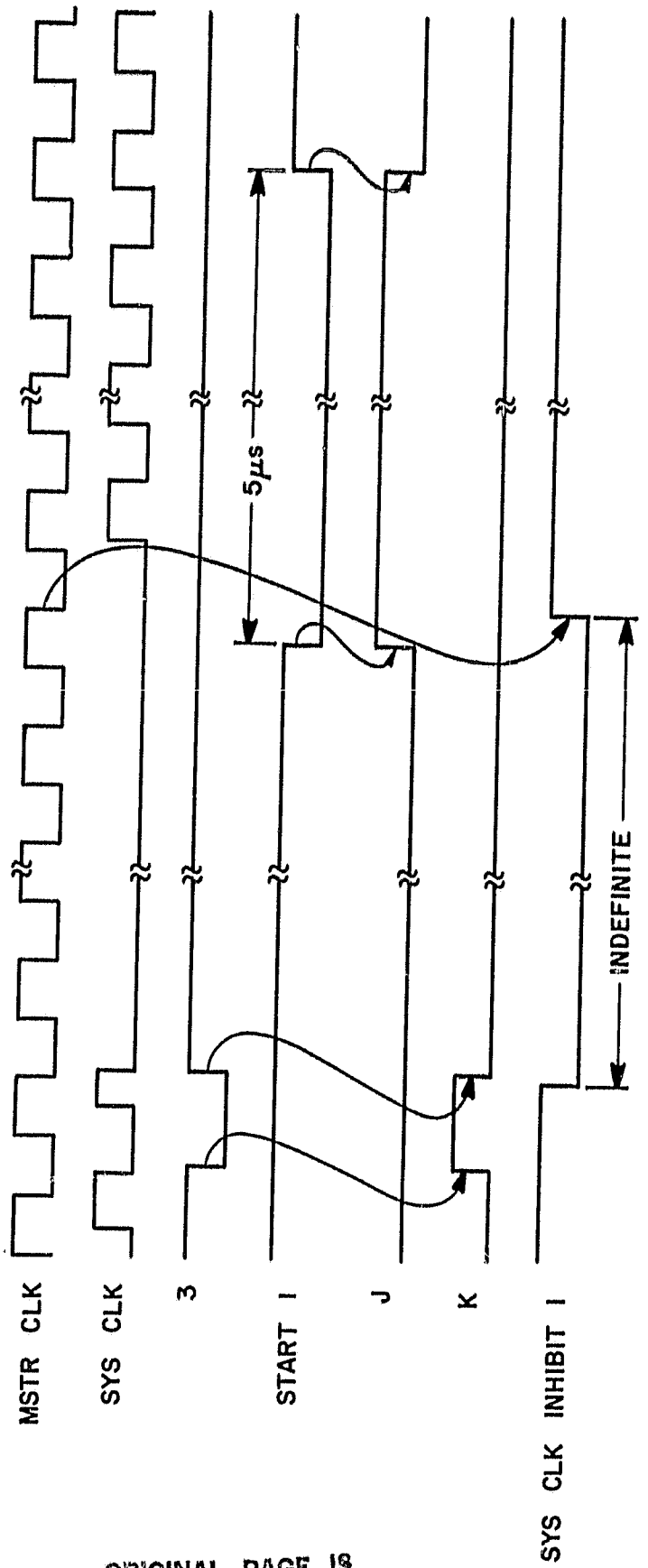

Figure 4.12(a) Disable instruction, DIS, timing diagram.

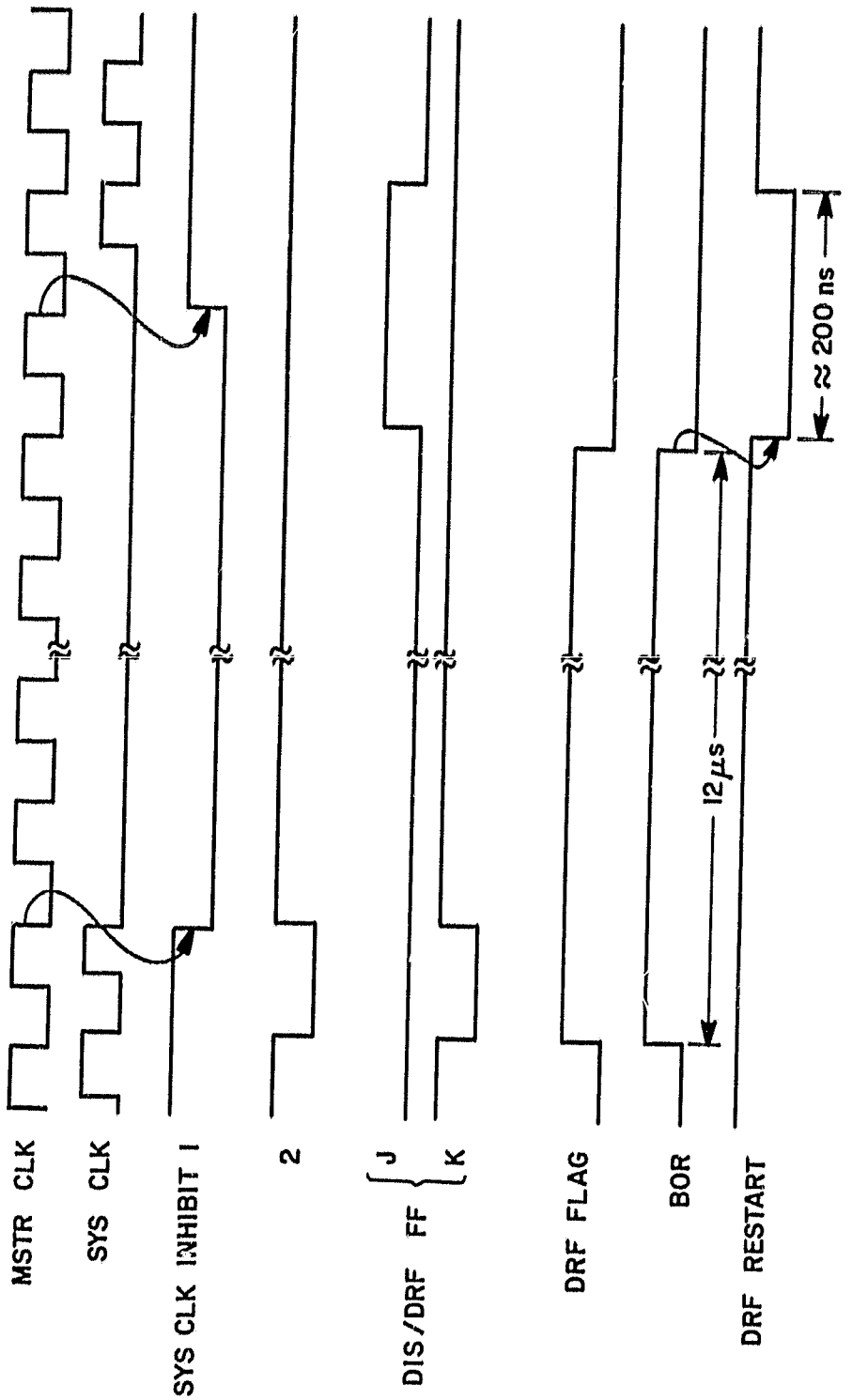

Figure 4.12(b) Data-ready flag instruction DRF, timing diagram.

started. The DIS/DRF flip-flop remains in a set mode while START1 is low. When START1 goes high the DIS/DRF flip-flop changes to the hold mode. SYS CLK INHIBIT1 remains high.

The data ready flag instruction is used to issue a DRF to the PDP-15 interface when data is to be transferred to the main computer. A minimum of 10 μs are required to transfer each data word, hence after issuing the data ready flag, the preprocessor must wait before continuing program execution. The DRF instruction presets the four flip-flops and loads the counter used in the DRF circuitry and takes the DIS/DRF flip-flop input high. Again the master clock clocks the SYS CLK INHIBIT1 low. When flip-flop 4, FF4, is preset its $2\overline{Q}$ output goes low taking the DRF line high and thus beginning a data transfer cycle (Figure 4.12b). Flip-flops 1, 2, and 3 divide the master clock down to a 1.25 MHz signal which is used to clock the counter. When the counter has counted down to zero its borrow output goes low clocking FF4 into a reset state. This takes the DRF line high, inhibits the clock to the counter input and triggers a monostable. The monostable output in a 200 ns negative-going pulse which increments the address register and takes the DIS/DRF flip-flop J input high to restart the system clock.

4.10 *The Address Register*

The 12-bit address for the preprocessor's random access memory is provided by the address register (Figure 4.13). This register is made from three 74193 sychronous 4-bit up/down counters (Figure 4.14). These counters have a count-up input which increments the register value on a rising pulse edge. They are also equipped with a carry output which goes low when the register value reaches 15. On the next count pulse the carry line goes high and the register output is reset to zero. By using the carry of one counter as the count up input of another a 4n bit counter can be built. The address

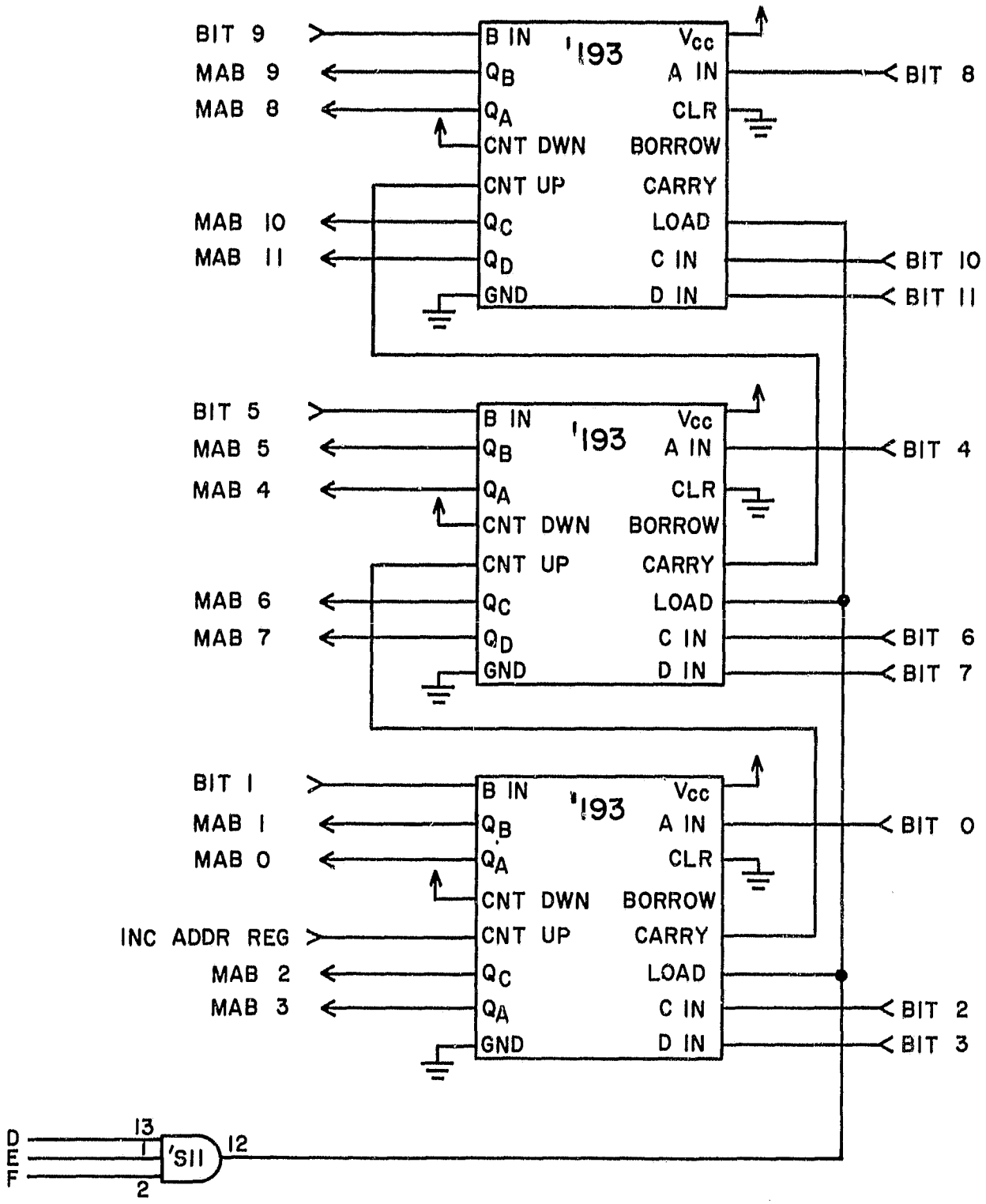

Figure 4.14 Address register pin diagram. MAB i is the memory address bus bit i. Bits 0-11 are the address register inputs.

register is incremented by instruction DRF (DRF RESTART line going low) or a jump on data acquisition, JDA.

The address register can be programmed to any value by entering the desired data at the data inputs while the load line is held low. Three different register outputs can be selected and loaded into the address register. The working address register, transfer address register, and memory half word all have three state outputs which are connected to the address register inputs. When any of the enable lines D, E, or F, are taken low by instruction RWR, RTR, or LAR respectively, the address register load line is taken low and the address register is loaded with the data on its input.

The memory half word is the output of two 74367 hex bus drivers. The input to eleven of the drivers is hardwired while the input of the twelfth driver, the most significant bit of the address register input, is tied to the memory half flip-flop. When instruction LAR enables these drivers one of two words is loaded into the address register, depending upon the memory half flip-flop, MSB 000 0000 0001.

The transfer address register and working address register are used to temporarily store the address register value. Both registers are three 74173s, 4-bit D-type registers with 3-state outputs (Figure 4.15). The data on the MEMORY ADDRESS BUS is loaded into a register on the rising edge of its clock pulse, CLK D-REGISTER, when instruction WWR or WTR takes the load line low. When the enable out line is high the register outputs are in the high impedance state, this does not affect operations of the register however.

4.11 *Memory Half Flip-Flop*

The memory half flip-flop (MEM.HALF) is a 74S113 J-K negative-edge triggered flip-flop (Figure 4.13). It determines whether the MSB of the

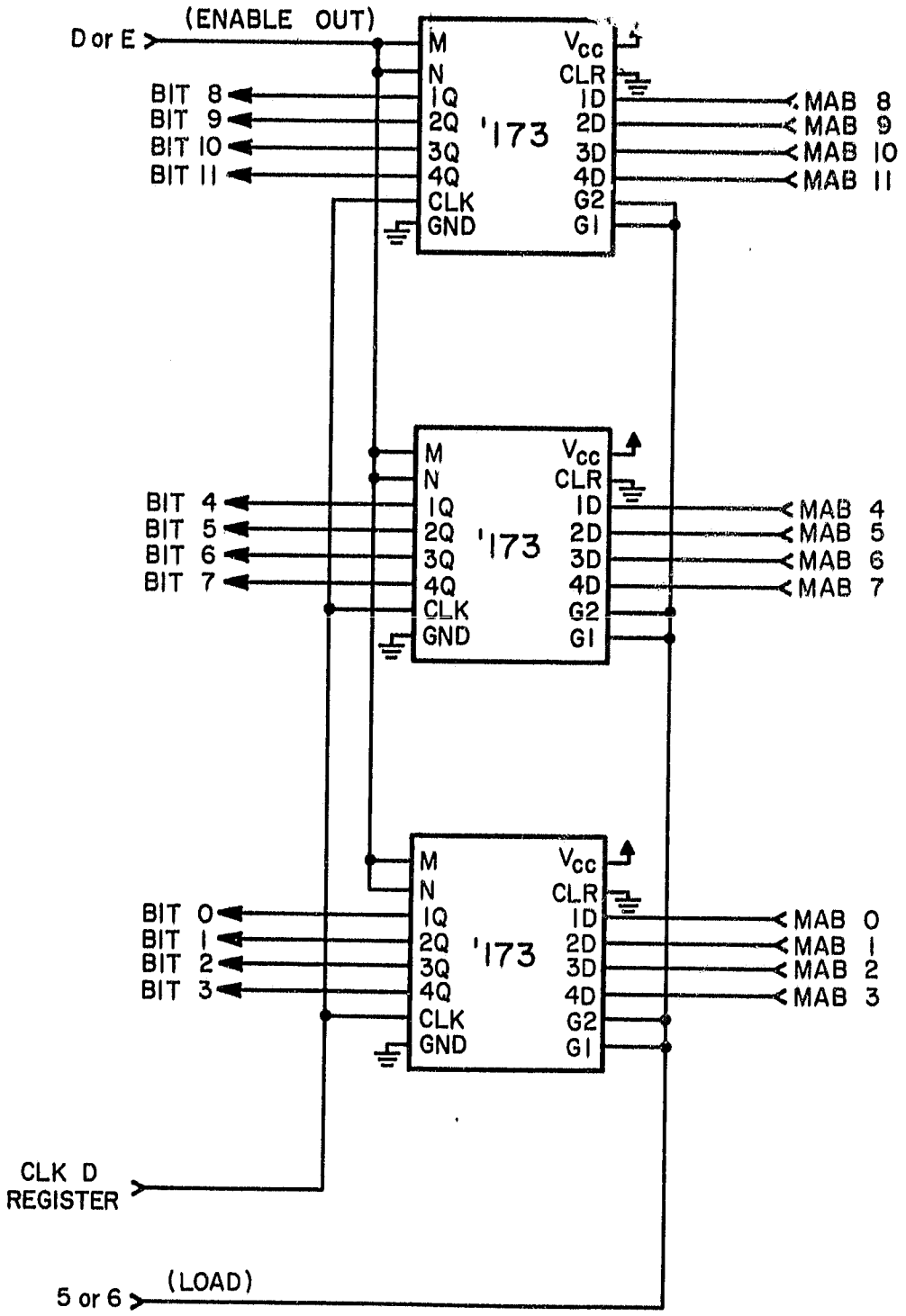

Figure 4.15 Timing and working address register pin diagram. MAB i is the memory address bus bit i. Bits 0-11 are the address register inputs.

memory half word is high or low. The flip-flop has its J and K inputs tied high hence it is in the toggle mode, instruction ICI toggles the output. The flip-flop is preset by the INIT instruction.

4.12 *Reset*

When the preprocessor is turned on its initial state is not known. In order to start operation the program counter (PC) must be cleared, the program counter latch enabled and then latched and the jump instruction and disable/data ready flag flip-flops must be preset. A single pole double throw (SPDT) switch on the front panel activates these initializations (Figure 4.2). In the RESET position this switch pulls the jump instruction and DIS/DRF flip-flop preset lines low, thus setting these flip-flops. RESET also takes the clear program counter and enable program counter latch high. This sets the program counter to zero which is latched by the program counter latch when the reset switch goes low.

4.13 *Single Step*

The single step circuitry inhibits the system clock. When pulsed the signle step circuitry allows a single master clock period onto the system clock line. This is an extremely helpful debugging tool since it allows instructions to be executed singly and the logic state of the preprocessor examined before executing the next instruction. The single step is implemented with two 74S112 J-K negative edge triggered flip-flops and an OR gate (Figure 4.1). The two control lines are connected to SPDT toggle switches on the preprocessor front panel, the PULSE SINGLE STEP switch is debounced. When the ENABLE SINGLE STEP line is low (enabled) the SYS CLK INHIBIT2 line is determined by the J-K flip-flop 2Q output. A single clock pulse is gated onto the system clock line by synchronously raising the 2Q output high for one master clock period. This is initiated by a falling edge on the PULSE

SINGLE STEP line which clocks 1Q high and $1\overline{Q}$ low (the flip-flop is initially clear (Figure 4.16)). The next falling edge of the master clock then clocks 2Q high, uninhibiting the system clock. The first falling edge of the inverted system clock ($\overline{\text{SYS CLK}}$) clears the first flip-flop hence the falling edge of the corresponding master clock pulse resets the second flip-flop. 2Q is now low and the system clock once again inhibited.

4.14 *Power Supplies*

Three power supplies are used to provide the required power levels needed, a POWER-ONE HA-5, a POWER-ONE AA15-8 and an ADTECH POWER APS 5-10. The HA-5 has a floating output and is connected as a -5V, 1.2 A supply. The AA15-8 is a dual ±15V, 0.8 A supply. The APS5-10 is a +5V, 10 A supply. The power supplies are mounted on a panel separate from the preprocessor. The AC input to each supply is fused, a single switch turns on all three supplies (Figure 4.17). Each DC output has an LED indicator lamp.

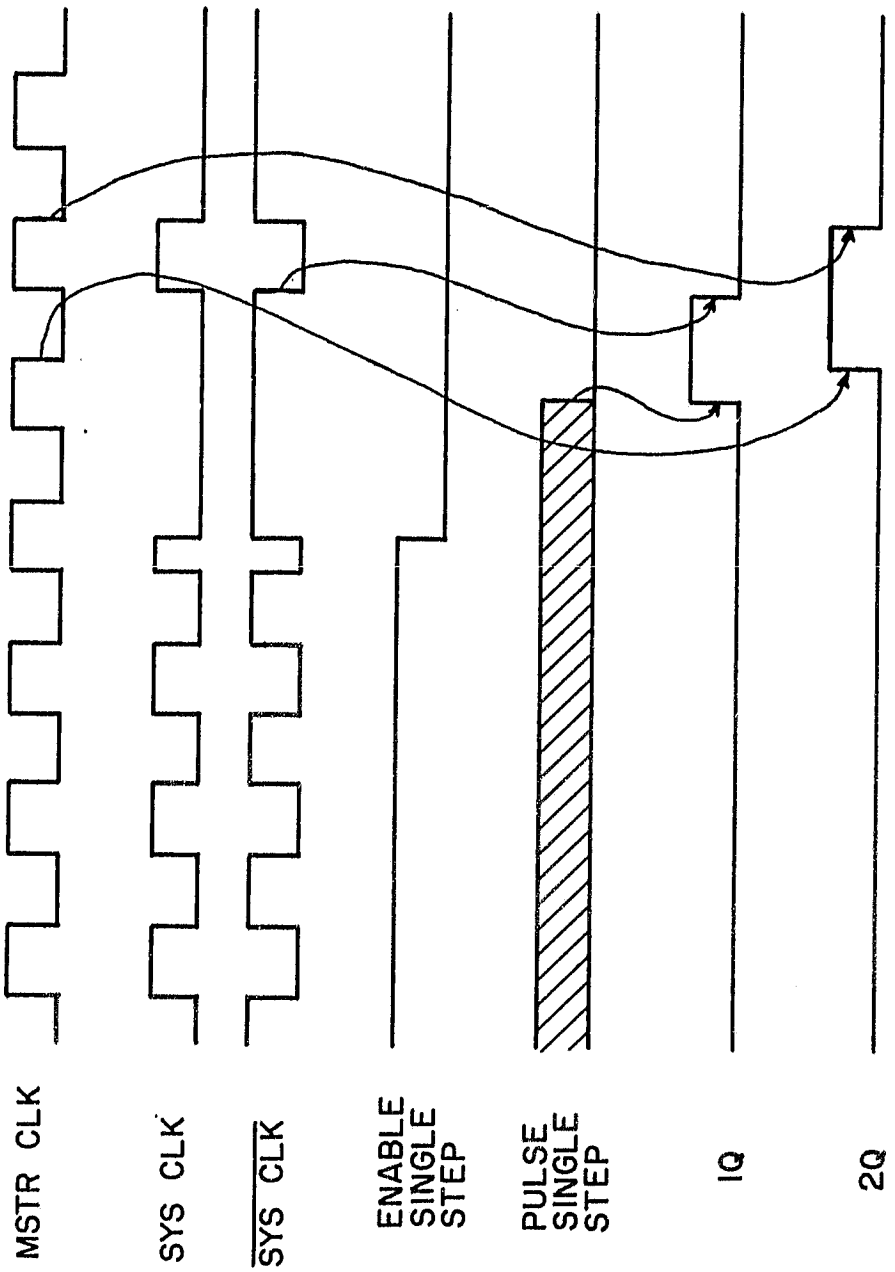

Figure 4.16 Single step timing diagram.

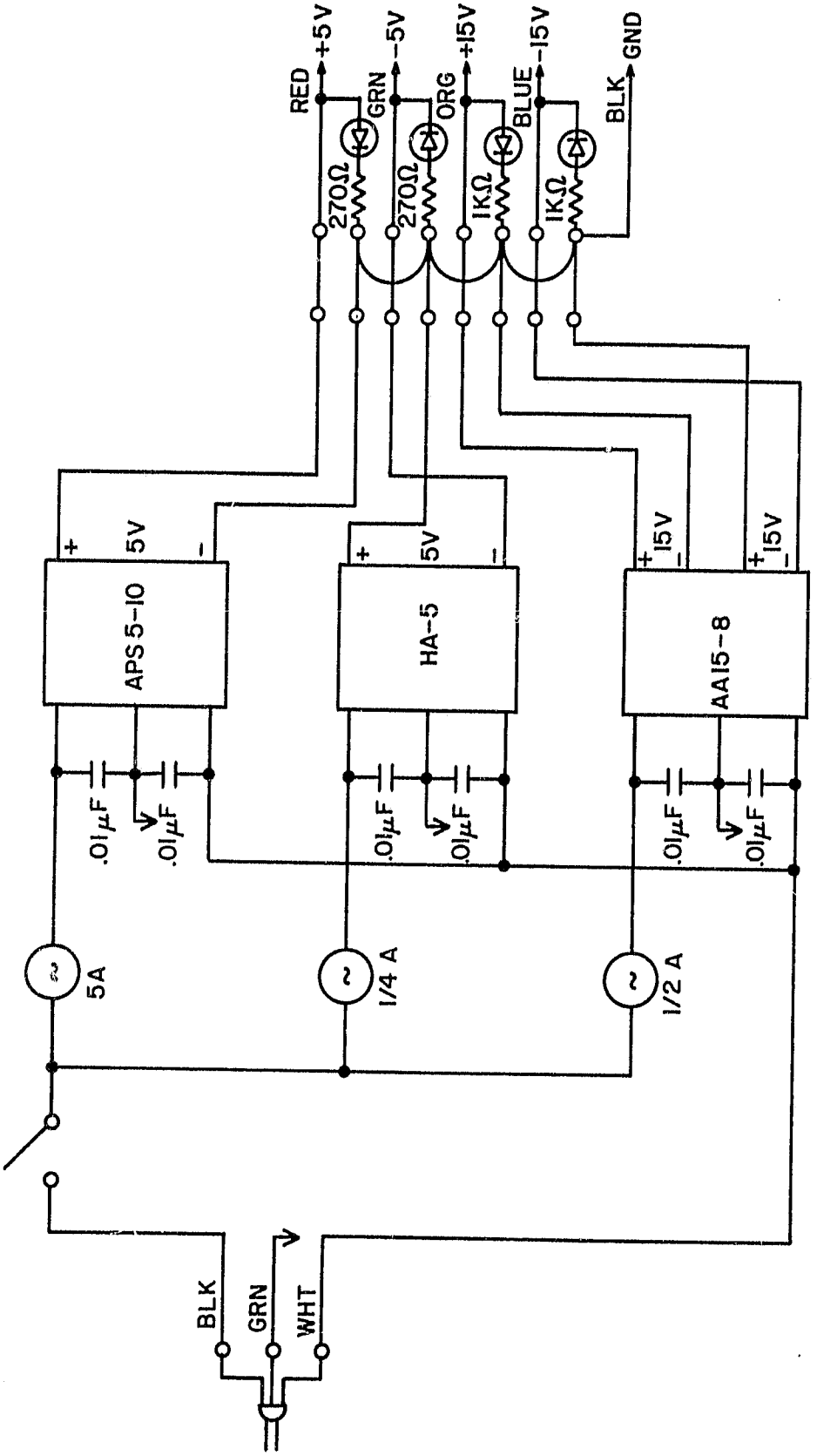

Figure 4.17 Power supplies wiring.

5. FUNCTIONAL VERIFICATION

5.1 *Data Transfer*

The main computer, a PDP-15, uses several programs to input data from the preprocessor and process it. These programs are written in FORTRAN or PDP-15 assembly language and appear in Appendix II. The first program, INPAD, permits input of any specified number of samples from the A/D interface into a core buffer. INPAD is used as a subroutine in the main data processing program. When a subroutine call to this program is made, the array name where the samples are to be stored, the number of samples to be transferred, and the completion flag name must be specified. The subroutine call clears the completion flag. After the subroutine call, INPAD immediately begins inputting data. When the specified number of samples has been transferred (1200 for the preprocessor since that is the number of samples in a coherent integration), the completion flag is set to one. In the case of a data timing error the completion flag is set to -512. Any data collection or processing program being executed by the PDP-15 runs completely independently of the preprocessor. Hence the data collection program may call INPAD at any point during a coherent integration. If INPAD is called while a set of coherently integrated data is being transferred then some of the data collected by INPAD will be from one set of coherently integrated data and the remaining data will come from the subsequent set of coherently integrated data. Since the preprocessor's data transfer subroutine and INPAD run asynchronously a software means of synchronization is implemented.

Program PSYNC is used to synchronize the transfer of data from the preprocessor to the PDP-15. When PSYNC is called the array containing the data transferred from the preprocessor and the error flag must be passed. The 18-

bit words transferred from the preprocessor consist of a sixteen-bit data word and the most the most significant bit of its memory address. Since all 1200 words of one set of coherently integrated data are stored in the top or bottom half of the preprocessor's memory all 1200 words have the same memory address MSB. PSYNC checks the memory address MSB of the first and last (1200^{th}) words transferred during one call to INPAD. If the memory address MSB is the same for both words, and hence all 1200 of the words just transferred, then these 1200 words all come from the same half of the preprocessor memory and must comprise a set of coherently integrated data. The data transfer is in synchronization so the error flag is set to one. If the memory address MSB of the first and 1200^{th} words transferred are different then these words are from two different halves of the preprocessor memory. This means the data just transferred came from two different sets of coherently integrated data. The data transfer is out of synchronization so the error flag is set to zero. Program INPAD is called again and the new array of input data is chucked by PSYNC. This procedure is repeated until synchronization is achieved. Generally only one or two calls to INPAD are required since the time required to transfer a set of coherently integrated data is small compared to the integration time.

5.2 *Data Reformatting*

Since the preprocessor uses a 16-bit data word and the PDP-15 uses an 18-bit data word the preprocessor's data must be reformatted before they can be used by the PDP-15. Program CNVRT2 is used to remove the preprocessor's memory address bit from the word transferred and convert the 16-bit numbers to 18-bit numbers. The ADCs used do not convert the analog input voltage to a two's complement digital word (see Table 4.3), hence CNVRT2 also performs

the necessary operations to convert the preprocessor data to two's complement numbers. The memory address bit is removed by ANDing the data word with an 177777_8. The result is an 18-bit word with the two most significant bits equal to zero followed by the 16-bit preprocessor data word. This is then converted to an 18-bit two's complement word by subtracting 80_{16} times the number of radar pulses coherently integrated (ie. the number of samples summed to form a coherent integration), to remove an "offset" on each word from the ADC. The remainder is then two's complemented and the correct 18-bit PDP-15 format word results.

5.3 *Testing of data*

Two programs were used on the PDP-15 to test the preprocessor data. The first program is RWVOLT. This is a FORTRAN program which collects one coherent integration (1200 words), converts the coherent integration numbers to their corresponding analog input voltage value and prints these values. Using a function generator, a sinusoidal input with a DC offset was placed on the preprocessor analog inputs. This simulates the condition which will occur during actual data collection, a small, relatively constant signal value with a larger, independent noise value added. The purpose of the preprocessor is to coherently integrate out the AC or noise value superimposed on the DC or very slowly changing signal value. When RWVOLT was run, constant values equal to the DC offset were obtained, (see Table 5.1). This indicates that the preprocessor is indeed coherently integrating out the AC noise component.

Another testing program, AVTST2, collects and processes coherent scatter data. This test program analyzes data collected by the coherent scatter system with the preprocessor in operation. When one minute of data is collected the autocorrelation function at lags zero and one are calculated.

Table 5.1 RWVOLT program sample output. The first 80 numbers are from the preprocessor memory locations 1-80. The last 80 numbers are from the preprocessor's memory locations 1121-1200. The first three pairs of values may not be correct as it takes the preprocessor three sample pulses to clear the S/H, ADC pipeline. The input for this sample was: odd memory locations, -0.26 VDC, ±0.1 VAC @1KHz; even memory locations, +0.5 VDC


```
* AND THE DATA IS =
-0.257    0.508   -0.275   -0.152   -0.258    0.518   -0.260    0.526
-0.259    0.523   -0.258    0.519   -0.258    0.518   -0.258    0.524
-0.258    0.498   -0.258    0.521   -0.258    0.522   -0.258    0.524
-0.259    0.515   -0.258    0.521   -0.259    0.517   -0.258    0.524
-0.260    0.496   -0.259    0.521   -0.261    0.523   -0.259    0.523
-0.260    0.516   -0.259    0.521   -0.260    0.518   -0.258    0.524
-0.259    0.495   -0.258    0.521   -0.259    0.519   -0.259    0.524
-0.258    0.515   -0.257    0.521   -0.258    0.524   -0.258    0.524
-0.259    0.502   -0.259    0.521   -0.260    0.526   -0.260    0.524
-0.259    0.516   -0.260    0.521   -0.260    0.523   -0.260    0.524
-0.259    0.500   -0.258    0.521   -0.258    0.524   -0.259    0.522
-0.259    0.513   -0.260    0.521   -0.259    0.522   -0.260    0.522
-0.260    0.502   -0.259    0.521   -0.260    0.528   -0.259    0.524
-0.260    0.515   -0.258    0.521   -0.259    0.512   -0.259    0.522
-0.259    0.509   -0.258    0.521   -0.258    0.523   -0.260    0.524
-0.259    0.515   -0.259    0.521   -0.260    0.528   -0.259    0.524
-0.260    0.506   -0.258    0.520   -0.260    0.523   -0.260    0.525
-0.261    0.513   -0.258    0.518   -0.260    0.524   -0.259    0.522
-0.260    0.508   -0.258    0.520   -0.259    0.526   -0.259    0.525
-0.259    0.514   -0.258    0.520   -0.258    0.524   -0.259    0.525
```

From the autocorrelation functions the returned signal power, line of sight velocity of the scattering volume, and ratio of magnitude of lag one to lag zero are computed. Using AVTST2 the preprocessor was tested for three different input values; a target generator, sky noise collected by the coherent scatter receiver, and data from the coherent scatter radar system.

The target generator is a piece of test equipment which will generate a sinusoidal waveform which is frequency shifted a selectable amount from the radar receiver's phase detector reference signal. Using this signal as the coherent scatter radar receiver input the receiver outputs are a sine and cosine wave at the frequency which is the difference of the phase detector reference signal and the target generator output. AVTST2 interprets this difference frequency as it appears in the preprocessor data, as a doppler frequency and calculates the corresponding line of sight velocity using the algorithm of equation 2.4. Since the doppler frequency is constant with a constant magnitude the power, velocity, and ratio of lags should all be constant over altitude. The velocity is determined by the doppler frequency. The ratio of lags should be approximately one since the doppler frequency is so small. Table 5.2 shows the values computed for an input doppler frequency of $\sim$ 2 Hz.

When the coherent scatter radar receiver is operating with no transmitted radar pulse, AVTST2 processes the background sky noise collected by the coherent scatter receiving system. In this case the signal power level should be approximately constant over altitude, the velocity should be random, and the ratio of lag one to lag zero should be small (less than 0.1). Table 5.3 shows sky noise power, velocity, and lag one to lag zero ratio values for the 70 to 80 km region. As expected the power is approximately equal and the ratio is less that 0.1 at all altitudes. When the ratio is

Table 5.2 AVTST2 program sample output. The target generator was used to generate an input doppler frequency of approximately two hertz. Column four is the ratio of the magnitude of the autocorrelation function of lag one to lag zero. Column five is the ratio of the real part of lag one to the magnitude of lag zero. Columns six and seven are the real and imaginary parts of the autocorrelation function at lag one.

ONE MINUTE AVERAGES

ALTITUDE (KM)	POWER	VELOCITY (M/S)				
70.00	0.176E+09	-7.69	0.94	-0.07	-0.127E+08	-0.165E+09
70.15	0.175E+09	-7.69	0.94	-0.07	-0.128E+08	-0.164E+09
70.30	0.175E+09	-7.69	0.94	-0.07	-0.128E+08	-0.164E+09
70.45	0.175E+09	-7.69	0.94	-0.07	-0.128E+08	-0.164E+09
70.60	0.176E+09	-7.69	0.94	-0.07	-0.128E+08	-0.165E+09
70.75	0.175E+09	-7.70	0.94	-0.07	-0.128E+08	-0.164E+09
70.90	0.175E+09	-7.69	0.94	-0.07	-0.128E+08	-0.164E+09
71.05	0.175E+09	-7.69	0.94	-0.07	-0.128E+08	-0.164E+09
71.20	0.176E+09	-7.69	0.94	-0.07	-0.128E+08	-0.165E+09
71.35	0.175E+09	-7.69	0.94	-0.07	-0.128E+08	-0.164E+09
71.50	0.175E+09	-7.69	0.94	-0.07	-0.128E+08	-0.164E+09
71.65	0.175E+09	-7.69	0.94	-0.07	-0.127E+08	-0.164E+09
71.80	0.176E+09	-7.69	0.94	-0.07	-0.128E+08	-0.165E+09
71.95	0.175E+09	-7.69	0.94	-0.07	-0.128E+08	-0.164E+09
72.10	0.175E+09	-7.69	0.94	-0.07	-0.127E+08	-0.164E+09
72.25	0.175E+09	-7.69	0.94	-0.07	-0.128E+08	-0.164E+09
72.40	0.176E+09	-7.69	0.94	-0.07	-0.128E+08	-0.164E+09
72.55	0.175E+09	-7.69	0.94	-0.07	-0.127E+08	-0.164E+09
72.70	0.175E+09	-7.69	0.94	-0.07	-0.128E+08	-0.164E+09
72.85	0.175E+09	-7.69	0.94	-0.07	-0.128E+08	-0.164E+09
73.00	0.175E+09	-7.69	0.94	-0.07	-0.126E+08	-0.164E+09
73.15	0.174E+09	-7.69	0.94	-0.07	-0.126E+08	-0.163E+09
73.30	0.175E+09	-7.69	0.94	-0.07	-0.127E+08	-0.164E+09
73.45	0.175E+09	-7.69	0.94	-0.07	-0.127E+08	-0.164E+09
73.60	0.176E+09	-7.69	0.94	-0.07	-0.128E+08	-0.165E+09
73.75	0.175E+09	-7.69	0.94	-0.07	-0.126E+08	-0.164E+09
73.90	0.175E+09	-7.69	0.94	-0.07	-0.127E+08	-0.164E+09
74.05	0.175E+09	-7.70	0.94	-0.07	-0.129E+08	-0.164E+09

Table 5.3 AVTST2 program sample output. The input is sky noise collected by the coherent-scatter receiving system. Column four is the ratio of the magnitude of the autocorrelation function of lag one to lag zero. Column five is the ratio of the real part of lag one to the magnitude of lag zero. Columns six and seven are the real and imaginary parts of the autocorrelation function at lag one.

ONE MINUTE AVERAGES

ALTITUDE (KM)	POWER	VELOCITY (M/S)				
70.00	0.168E+07	0.00	0.05	-0.01	-0.118E+05	0.780E+05
70.15	0.173E+07	0.00	0.05	-0.03	-0.584E+05	0.640E+05
70.30	0.180E+07	0.00	0.04	-0.03	-0.611E+05	-0.391E+05
70.45	0.182E+07	0.00	0.02	-0.01	-0.257E+05	-0.323E+05
70.60	0.195E+07	0.00	0.03	0.01	0.150E+05	0.473E+05
70.75	0.193E+07	0.00	0.04	0.03	0.553E+05	0.405E+05
70.90	0.194E+07	0.00	0.01	0.01	0.236E+05	-0.605E+03
71.05	0.198E+07	0.00	0.02	0.02	0.318E+05	-0.370E+05
71.20	0.199E+07	0.00	0.06	-0.02	-0.362E+05	-0.112E+06
71.35	0.196E+07	0.00	0.03	-0.01	-0.148E+05	-0.551E+05
71.50	0.185E+07	0.00	0.07	-0.06	-0.115E+06	-0.468E+05
71.65	0.177E+07	0.00	0.06	-0.06	-0.101E+06	0.162E+05
71.80	0.178E+07	0.00	0.07	-0.06	-0.115E+06	-0.164E+05
71.95	0.182E+07	0.00	0.03	-0.03	-0.456E+05	-0.351E+05
72.10	0.168E+07	0.00	0.02	0.01	0.229E+05	-0.270E+05
72.25	0.170E+07	0.00	0.05	0.05	0.773E+05	-0.345E+05
72.40	0.164E+07	0.00	0.04	0.01	0.229E+05	-0.598E+05
72.55	0.168E+07	0.00	0.05	-0.00	-0.178E+04	-0.790E+05
72.70	0.171E+07	0.00	0.03	0.01	0.149E+05	-0.419E+05
72.85	0.184E+07	0.00	0.02	-0.01	-0.107E+05	0.444E+05
73.00	0.195E+07	0.00	0.02	-0.00	-0.433E+04	0.345E+05
73.15	0.192E+07	0.00	0.05	-0.04	-0.718E+05	0.681E+05
73.30	0.188E+07	0.00	0.04	-0.04	-0.809E+05	0.144E+05
73.45	0.186E+07	0.00	0.04	0.01	0.126E+05	-0.661E+05
73.60	0.185E+07	0.00	0.03	-0.02	-0.423E+05	-0.199E+05
73.75	0.179E+07	0.00	0.04	0.00	0.267E+04	0.693E+05
73.90	0.174E+07	0.00	0.06	0.05	0.864E+05	0.656E+05
74.05	0.168E+07	0.00	0.06	0.06	0.102E+06	0.295E+05

less than 0.1 the velocity is not computed, the default value is zero.

With the transmitter operating, AVTST2 collects coherent scatter data. The coherent scatter data in Table 5.4 shows the ratio is significantly greater than 0.1, velocity values are on the order of meters per second and the power goes as much as one or two orders of magnitude greater than the background sky noise. These are the same results seen with the present coherent scatter system used at the University of Illinois, (Gibbs and Bowhill, 1979), and again indicates the proper functioning of the preprocessor. AVTST2 is a fortran program and requires approximately ten minutes to process one minute of data. Although AVTST2 runs too slowly to gather data for scientific analysis, as a test program it shows that the data collected by the preprocessor are similar to those from the present coherent scatter system indicating the preprocessor is operating correctly.

Table 5.4 AVTST2 program sample output. These results show coherent-scatter data collected the morning of February 8, 1982. Column four is the ratio of the magnitude of the autocorrelation function at lag one to lag zero. Column five is the ratio of the real part of lag one to the magnitude of lag zero. Columns six and seven are the real and imaginary parts of the autocorrelation function at lag one.

ONE MINUTE AVERAGES

ALTITUDE (KM)	POWER	VELOCITY (M/S)				
70.00	0.293E+08	1.92	0.88	0.80	0.234E+08	0.103E+08
70.15	0.294E+08	1.93	0.88	0.80	0.235E+08	0.104E+08
70.30	0.294E+08	1.95	0.87	0.80	0.234E+08	0.104E+08
70.45	0.289E+08	1.94	0.87	0.80	0.231E+08	0.102E+08
70.60	0.289E+08	1.92	0.87	0.80	0.231E+08	0.101E+08
70.75	0.282E+08	1.87	0.87	0.80	0.225E+08	0.953E+07
70.90	0.273E+08	1.81	0.86	0.80	0.218E+08	0.889E+07
71.05	0.264E+08	1.71	0.86	0.80	0.212E+08	0.812E+07
71.20	0.256E+08	1.64	0.86	0.80	0.206E+08	0.756E+07
71.35	0.254E+08	1.53	0.85	0.81	0.205E+08	0.700E+07
71.50	0.253E+08	1.45	0.85	0.81	0.205E+08	0.656E+07
71.65	0.253E+08	1.44	0.85	0.81	0.204E+08	0.648E+07
71.80	0.249E+08	1.41	0.83	0.80	0.198E+08	0.619E+07
71.95	0.241E+08	1.41	0.83	0.79	0.191E+08	0.595E+07
72.10	0.232E+08	1.41	0.83	0.79	0.184E+08	0.572E+07
72.25	0.222E+08	1.42	0.83	0.79	0.176E+08	0.551E+07
72.40	0.216E+08	1.44	0.82	0.78	0.169E+08	0.539E+07
72.55	0.217E+08	1.46	0.82	0.78	0.169E+08	0.549E+07
72.70	0.219E+08	1.43	0.82	0.78	0.171E+08	0.539E+07
72.85	0.222E+08	1.42	0.83	0.79	0.176E+08	0.551E+07
73.00	0.228E+08	1.39	0.84	0.80	0.182E+08	0.559E+07
73.15	0.237E+08	1.36	0.84	0.80	0.190E+08	0.568E+07
73.30	0.251E+08	1.36	0.84	0.81	0.203E+08	0.608E+07
73.45	0.264E+08	1.38	0.84	0.81	0.213E+08	0.648E+07
73.60	0.268E+08	1.41	0.84	0.81	0.216E+08	0.675E+07
73.75	0.276E+08	1.44	0.85	0.81	0.222E+08	0.706E+07
73.90	0.283E+08	1.45	0.85	0.81	0.228E+08	0.732E+07
74.05	0.286E+08	1.40	0.85	0.81	0.231E+08	0.714E+07

6.CONCLUSIONS AND SUGGESTIONS FOR FUTURE WORK

The data collection test program indicates that the preprocessor is performing well. Returned signal power and line of sight velocity data computed from preprocessor data are very similar to data from the present Urbana coherent-scatter radar system. For a given radar pulse width and pulse repetition frequency the signal-to-noise ratio of the preprocessor data is double that of the present system. This is a consequence of sampling the two phase detector channels simultaneously rather than alternately as in the present system.

To take advantage of the preprocessor requires implementing a real-time data collection program which uses preprocessor data. Since the coherent integration is performed by the preprocessor only computing the autocorrelation functions and then averaging need be done by the main computer. The PDP-15 minicomputer presently used for data collection is not capable of processing in real time all the preprocessor data, which cover a 90 km range. However, only a 30 km region of the mesosphere is of active interest presently. The amount of data needed to be processed can be further reduced. The radar pulse width is 20 μs so the preprocessor's 1 μs sampling interval is tremendously oversampling the return signal. Processing every fourth preprocess or altitude sample will still result in oversampling by a factor of five but should reduce the amount of data to within the PDP-15's limit.

Pulse coding the radar would take advantage of the preprocessor's sampling rate. To code the radar the phase of the transmitter is varied during the pulse according to a code word. The minimum time between phase shifts determines the range resolution. The returned signal is decoded by corre-

lating the returned signal with the transmitted code. Upgrades to the Urbana transmitter necessary for phase coding are presently being investigated.

APPENDIX I

BACKPLANE AND RIBBON CABLE SIGNAL NAMES.

The tables in this section identify the signals on the preprocessor's backplane and ribbon cables. Tables I.1-I.6 show the signals on the six wire-wrap board backplane connectors. The B1, B2,..., B6 next to each signal name is the board or boards to which this signal is connected, FP front panel. Tables I.7-I.15 show signal names on the ribbon cables.

BOARD #1

	+5V	A	1	+5V
		B	2	
CHANNEL 1:	BIT 0, B4	C	3	B2, S0
	BIT 1, B4	D	4	B2, S1
	BIT 2, B4	E	5	B2, S2
	BIT 3, B4	F	6	B2, S3
	BIT 4, B4	H	7	
	BIT 5, B4	J	8	
	BIT 6, B4	K	9	
	BIT 7, B4	L	10	B2, R/W
		M	11	B2, ENABLE FEED.BUF
CHANNEL 2:	BIT 0, B5	N	12	
	BIT 1, B5	P	13	B2, DATA MUX SELECT
	BIT 2, B5	R	14	B2, INC ADDR REG
	BIT 3, B5	S	15	B2, LOAD ADDR REG
	BIT 4, B5	T	16	
	BIT 5, B5	U	17	
	BIT 6, B5	V	18	
	BIT 7, B5	W	19	
		X	20	
		Y	21	
	GND	Z	22	GND

BOARD #2

+5V	A	1	+5V
SYS.CLK (5 G.D.), B3	B	2	
D, B3	C	3	B1, S0
5, B3	D	4	B1, S1
E, B3	E	5	B1, S2
6, B3	F	6	B1, SB
7, B3	H	7	START1 (FROM RADAR DIRECTOR)
3, B3	J	8	
SYS.CLK, B4,B5	K	9	
JCI, B3	L	10	B1, R/W
JDA, B3	M	11	B1, ENABLE FEED BUF
JDT, B3	N	12	
JMF, B3	P	13	B1, DATA MUX SELECT
C, B3	R	14	B1, INC ADDR REG
2, B3	S	15	B1, F
B, B3	T	16	B4,B5 START CONVERT
1, B3	U	17	B4,B5 Q (FROM 196)
	V	18	
	W	19	B4, PULSE SINGLE STEP
JMP_5, B3	X	20	B4, ENABLE SINGLE STEP
	Y	21	B4, RESET
GND	Z	22	GND

BOARD #3

+5V	A	1	+5V
$\overline{\text{SYS CLK}}$ (5 G.D.), B2	B	2	
D, B2	C	3	
5, B2	D	4	
E, B2	E	5	
6, B2	F	6	
7, B2	H	7	
3, B2	J	8	
	K	9	
JCI, B2	L	10	
JDA, B2	M	11	
JDT, B2	N	12	
JMF, B2	P	13	
C, B2	R	14	
2, B2	S	15	
B, B2	T	16	
1, B2	U	17	
	V	18	
	W	19	
JMP_5, B2	X	20	
	Y	21	
GND		22	GND

BOARD #4

	+5V	A	1	+5V
	SYS CLK, B2	B	2	
		C	3	SPARE SWITCH #1 (ON FP)
		D	4	SPARE SWITCH #2 (ON FP)
		E	5	-5V
		F	6	
		H	7	B2, PULSE SINGLE STEP (FP SWITCH)
		J	8	B2, ENABLE SINGLE STEP (FP SWITCH)
		K	9	B2, RESET (FP SWITCH)
		L	10	+15V
		M	11	
CHANNEL 2:	BIT 0, B1	N	12	-15V
	BIT 1, B1	P	13	
	BIT 2, B1	R	14	
	BIT 3, B1	S	15	
	BIT 4, B1	T	16	B2, START CONVERT
	BIT 5, B1	U	17	B2, Q (FROM 196)
	BIT 6, B1	V	18	
	BIT 7, B1	W	19	
	GND	X	20	
	GND	Y	21	
	GND	Z	22	GND

BOARD #5

+5V	A	1	+5V
SYS CLK, B2	B	2	
CHANNEL 1: BIT 0, B1	C	3	
BIT 1, B1	D	4	
BIT 2, B1	E	5	-5V
BIT 3, B1	F	6	
BIT 4, B1	H	7	
BIT 5, B1	J	8	
BIT 6, B1	K	9	
BIT 7, B1	L	10	+15V
	M	11	
	N	12	-15V
	P	13	
	R	14	
	S	15	
	T	16	B2, START CONVERT
	U	17	B2, Q_D (FROM 196)
	V	18	
	W	19	
GND	X	20	
GND	Y	21	
GND	Z	22	GND

BOARD #6

DIG GND	A	1	DO 9/DO 9
EXT IN/DO 12	B	2	
DO 8/DO 8	C	3	DO 5/DO 5
EXT IN/DO 13	D	4	DO 6/DO 6
DRF/DRF	E	5	DO 7/DO 7
	F	6	DO 4/DO 4
DO 3/DO 3	H	7	DO 2/DO 2
EXT IN/DO 10	J	8	DO 1/DO 1
	K	9	DO 0/DO 0
EXT IN/DO 11	L	10	CHAN ID 0/DO 14
CHAN ID 1/DO 15	M	11	CHAN ID 2/GND
	N	12	CHAN ID 3/MEM MSB
ENCODE COMMAND	P	13	RAC #1
	R	14	RAC #3
FREE RUN-ENCODE	S	15	RAC #0
	T	16	RAC #2
INT-EXT SEQ	U	17	EXT SEQ PULSE
	V	18	RAM-SEQ MODE

NOTE 1. PINS A-M AND 1-12 each have one of two possible signals, signal A / signal B; signal A originates in the HP5610A ADC, signal B originates in the preprocessor. Toggle switches on the preprocessor front panel determine which signal is selected.

BOARD #1 - 26 PIN RIBBON CABLE CONNECTOR

		25	26	
ADDR.REG INPUT	A_{10}	23	24	A_{11}
	A_8	21	22	A_9
	A_6	19	20	A_7
	A_4	17	18	A_5
	A_2	15	16	A_3
	A_0	13	14	A_1
ADDR.REG OUTPUT	A_{10}	11	12	A_{11}
	A_8	9	10	A_9
	A_6	7	8	A_7
	A_4	5	6	A_5
	A_2	3	4	A_3
	A_0	1	2	A_1

BOARD #1 - 20 PIN RIBBON CABLE CONNECTOR

	19	20	
	17	18	MEM MSB
DATA OUT 14	15	16	DATA OUT 15
DATA OUT 12	13	14	DATA OUT 13
DATA OUT 10	11	12	DATA OUT 11
DATA OUT 8	9	10	DATA OUT 9
DATA OUT 6	7	8	DATA OUT 7
DATA OUT 4	5	6	DATA OUT 5
DATA OUT 2	3	4	DATA OUT 3
DATA OUT 0	1	2	DATA OUT 1

BOARD #2 - 34 PIN RIBBON CABLE CONNECTOR

8	33	34	8
9	31	32	9
10	29	30	10
11	27	28	11
12	25	26	12
13	23	24	13
14	21	22	14
15	19	20	15
	17	18	
0	15	16	0
1	13	15	1
2	11	12	2
3	9	10	3
4	7	8	4
5	5	6	5
6	3	4	6
7	1	2	7
TO INSTR BUFFER INPUTS			FROM CS OUTPUTS

NOTE 1. For operation, short pin 1 to pin 2, pin 3 to pin 4, etc.

2. For debugging, the CS can be removed and instructions externally placed on the INSTR BUFFER INPUTS.

BOARD #2 - 20 PIN RIBBON CABLE CONNECTOR

		19	20	
		17	18	
		15	16	
		13	14	
ADDR REG INPUTS (from hardwired MEM.HALF)	A_{10}	11	12	A_{11}
	A_8	9	10	A_9
	A_6	7	8	A_7
	A_4	5	6	A_5
	A_2	3	4	A_3
	A_0	1	2	A_1

C-2

BOARD #3 - 50 PIN RIBBON CABLE CONNECTOR

		49	50	
		47	48	
		45	46	
TO TIME.CTR	A_6	43	44	A_7
INPUTS FROM	A_4	41	42	A_5
FRONT	A_2	39	40	A_3
PANEL	A_0	37	38	A_1
TO PULSE. CTR	A_{10}	35	36	A_{11}
INPUTS FROM	A_8	33	34	A_9
FRONT	A_6	31	32	A_7
PANEL	A_4	29	30	A_5
	A_2	27	28	A_3
	A_0	25	26	A_1
FROM	A_{10}	23	24	A_{11}
WORKING ADDR.REG	A_8	21	22	A_9
AND TRANSFER ADDR	A_6	19	20	A_7
REG OUTPUTS TO	A_4	17	18	A_5
ADDR. REG INPUT	A_2	15	16	A_3
	A_0	13	14	A_1
FROM ADDR.REG	A_{10}	11	12	A_{11}
OUTPUT TO	A_8	9	10	A_9
WORKING ADDR.REG	A_6	7	8	A_7
AND TRANSFER	A_4	5	6	A_5
ADDR REG INPUTS	A_2	3	4	A_3
	A_0	1	2	A_1

BOARD #5 - 26 PIN RIBBON CABLE CONNECTOR

FREE RUN-ENCODE	25	26	INT-EXT SFQ
RAM-SEQ MODE	23	24	ENCODE COMMAND
RAC #2	21	22	EXT SEQ PULSE
RAC #3	19	20	RAC #0
DIG GND	17	18	RAC #1
DO 3	15	16	DRF
DO 8	13	14	CHAN ID 1
CHAN ID 2	11	12	CHAN ID 3
DO 0	9	10	CHAN ID 0
DO 2	7	8	DO 1
DO 7	5	6	DO 4
DO 5	3	4	DO 6
DO 9	1	2	

NOTE 1. Pins 17-26 go directly to the 34 pin ribbon cable connector on BOARD #6.

2. Pins 1-16 go to the 36 pin ribbon cable connector on the front panel.

BOARD #6 - 34 PIN RIBBON CABLE CONNECTOR

	33	34	
	31	32	
FREE RUN-ENCODE	29	30	INT-EXT SEQ
RAM-SEC MODE	27	28	ENCODE COMMAND
RAC #2	25	26	EXT SEQ PULSE
RAC #3	23	24	RAC #0
DIG GND	21	22	RAC #1
	19	20	
EXT INPUT/DO 12	17	18	EXT INPUT/DO 13
EXT INPUT/DO 10	15	16	EXT INPUT/DO 11
CHAN ID 2/GND	13	14	CHAN ID 3/MEM MSB
CHAN ID 0/DO 14	11	12	CHAN ID 1/DO 15
DO 8/DO 8	9	10	DO 9/DO 9
DO 6/DO 6	7	8	DO 7/DO 7
DO 4/DO 4	5	6	DO 5/DO 5
DO 2/DO 2	3	4	DO 3/DO 3
DO 0/DO 0	1	2	DO 1/DO 1

NOTE 1. Signals on lines 21-30 come directly from the HP5610A ADC.

2. Pins 1-18 each have one of two possible signals, SIGNAL A/SIGNAL B; signal A originates in the HP5610A ADC, signal B originates in the preprocessor. Toggle switches on preprocessor front panel determine whcih signal is selected.

FRONT PANEL - 34 PIN RIBBON CABLE CONNECTOR

	33	34	MEM MSB
DO 14	31	32	DO 15
DO 12	29	30	DO 13
DO 10	27	28	DO 11
DO 8	25	26	DO 9
DO 6	23	24	DO 7
DO 4	21	22	DO 5
DO 2	19	20	DO 3
DO 0	17	18	DO 1
DO 3	15	16	DRF
DO 8	13	14	CHAN ID 1
CHAN ID 2	11	12	CHAN ID 3
DO 0	9	10	CHAN ID 0
DO 2	7	8	DO 1
DO 7	5	6	DO 4
DO 5	3	4	DO 6
DO 9	1	2	

NOTE 1. Pins 1-16 come from the 26 pin ribbon cable connector on BOARD #5.

2. Pins 17-34 come from the 20 pin ribbon cable connector on BOARD #5.

FRONT PANEL - 20 PIN RIBBON CABLE CONNECTOR

	19	20	
EXT INPUT/DO 12	17	18	EXT INPUT/DO 13
EXT INPUT/DO 10	15	16	EXT INPUT/DO 11
CHAN ID 2/GND	13	14	CHAN ID 3/MEM MSB
CHAN ID 0/DO 14	11	12	CHAN ID 1/DO 15
DO 8/DO 8	9	10	DO 9/DO 9
DO 6/DO 6	7	8	DO 7/DO 7
DO 4/DO 4	5	6	DO 5/DO 5
DO 2/DO 2	3	4	DO 3/DO 3
DO 0/DO 0	1	2	DO 1/DO 1

NOTE 1. EXT INPUT - BNC connectors on the back of the preprocessor which allow external signals to be placed on the interface lines.

2. Pins 1-18 each have one or two possible signals, signal A/ signal B; signal A originates in the HP5610A ADC, signal B originates in the preprocessor. Toggle switches on the pre-processor fron tpanel determine which signal is selected.

APPENDIX II

PDP-15 DATA INPUT AND PROCESSING PROGRAMS

The following programs are used by the PDP-15 minicomputer to input and test the date from the preprocessor. A brief explanation of the purpose of each program is given in Chapter five.

```
        .TITLE  A/D CONVERTER SERVICE ROUTINES FOR 8K.-FO.
/       DEK415 V3A SERVICE ROUTINES FOR THE HP 5610A A TO D
/  CONVERTER.  THESE ROUTINES PERMIT INPUT OF ANY SPECIFIED
/  NUMBER OF SAMPLES INTO A CORE BUFFER.  INPUT MAY BE OVER-

/  LAPPED WITH PROGRAM EXCUTION, AND CONTROL MAY BE RELINQUISHED
/  TO LOWER PRIORITY  PROGRAMS WHILE DATA TRANSFER TAKES PLACE.
/    MACRO-15 CALLING SEQUENCE:
/        JMS     INPAD
/        NUMBER OF SAMPLES REQUIRED
/        BUFFER ADDRESS
/        COMPLETION FLAG ADDRESS
/        REAL-TIME SUBROUTINE ADDRESS, PRIORITY LEVEL IN BITS 0-2
/  (EXAMPLE: 500000+RTSUBA)
/  (RETURNS HERE IMMEDIATELY)
/        IF THE 4TH WORD AFTER THE JMS IS 0, NO REAL-TIME SUBROUTINE
/  WILL BE ACTIVATED.  NOTE:  THE PRIORITY CODE FOR MAINSTREAM IS 1
/        THE COMPLETION FLAG IS CLEARED BY THE CALL TO INPAD,
/  AND SET TO +1 FOR NORMAL COMPLETION OR -1001 IF A DATA
/  TIMING ERROR OCCURS.
/
/
ADWCR=26                        /A-D WORD COUNT
ADCAR=ADWCR+1                   /AND CURRENT ADDRESS REGISTERS
.SCOM=100                       /MONITOR'S COMMUNICATION AREA
ADWI=703724                     /A-D CONVERTER WRITE INITIALIZE
ADSO=703701                     /SKIP ON WORD COUNT OVERFLOW
ADST=703721                     /SKIP ON DATA TIMING ERROR
ADCO=703704                     /CLEAR OVERFLOW FLAG
ADCT=703744                     /CLEAR TIMING FLAG
/
/  ENTRY POINT FOR A-D INTERFACE INITIALIZATION
/
        .GLOBL  INPAD,.DA
INPAD   0
        JMS*    .DA
        JMP     .+4
INAR    0
INWC    0
INFLAG  0
INR     JMP     INSET           /REPLACED BY "LAC*      INWC"
        TCA
        DAC*    (ADWCR)         /SET WORD COUNT
        LAW     -1
        TAD*    INAR            /BUFFER ADDRESS -1
        DAC*    (ADCAR)         /   TO CURRENT ADDRESS REG.
        DZM*    INFLAG          /CLEAR FLAG
        DZM     INSUB#          /CLEAR REAL-TIME SUBROUTINE
        ADWI                    /INITIALIZE INTERFACE
        JMP*    INPAD           /RETURN
/
/   THE FOLLOWING CODE IS EXECUTED ONLY ONCE
INSET   LAC*    (.SCOM+55)      /GET ENTRY POINT ADDERSS OF .SETUP
ADSVA   DAC     .
        JMS*    .-1             /CALL .SETUP TO CONNECT ADINT TO API
        ADSO
        ADINT
        DZM*    (204
        LAC     (LAC*   INWC
        DAC     INR             /MODIFY INSTRUCTION
        JMP     INR             / AND JUMP TO IT
/
/
/INTERRUPT SERVICE ROUTINE.  EXECUTED IMMEDIATELY AFTER COMPLETION
/  OF DATA TRANSFER.  DETERMINES STATUS OF A-D INTERFACE, SETS
/  COMPLETION FLAG AND ACTIVATES REAL-TIME SUBROUTINE.
/  RUNS AT API LEVEL 0.
/
/
ADINT   0
        DBA                     /PAGE ADDRESSING MODE
        DAC     ADSVA           /SAVE AC
        ADST                    /TIMING ERROR?
        SKP!CLA!IAC             /NO,+1 TO AC
        LAW     -1001           /YES, ERROR CODE
        DAC*    INFLAG          /SET FLAG
        ADCO                    /CLEAR
        ADCT                    /   INTERFACE FLAGS
ADXIT   LAC     ADSVA           /RESTORE AC
        DBR                     /SET TO LEAVE HARDWARE API LEVEL
        JMP*    ADINT           / RETURN TO INTERRUPTED PROGRAM
        .END
```

PSYNC program

```
        .GLOBL PSYNC, .DA
/  CHECK FOR SYNCHRONIZATION BETWEEN PREPROCESSOR AND INPAD
PSYNC   0
        JMS*    .DA
        JMP     .+4
STOR    0
IMP     0
IFLAG   0
        DZM*    IFLAG   /SET IFLAG TO ZERO
        LAC*    STOR    /GET ADDRESS OF ARRAY
        DAC     STOR    /AND STORE IT
        LAC*    STOR    /GET FIRST SAMPLE
        AND     (400000 /MASK OUT DATA WORD
        DAC     CH1     /STORE MEMORY MSB
        LAW     -1      /LOAD MINUS ONE AND
        TAD*    IMP     /  ADD TO NUMBER OF SAMPLES
        TAD     STOR    /GET ADDRESS OF LAST SAMPLE
        DAC     STOR    / AND STORE IT
        LAC*    STOR    /GET LAST SAMPLE
        AND     (400000 /MASK OUT DATA WORD
        SAD     CH1     /SKIP IF MEMORY MSM DIFFERENT
        ISZ*    IFLAG   /SET IFLAG IF MEMORY MSB IS THE SAME
        JMP*    PSYNC   /RETURN
CH1     0
        .END
```

CNVRT2 program

```
        .GLOBL CNVRT2, .DA
/  PROGRAM TO MASK PREPROCESSOR MEM MSB, SUBTRACT THE ADC OFFSET
/  AND CONVERT NUMBERS TO PDP-15 (18 BIT) FORMAT
CNVRT2  0
        JMS*    .DA
        JMP     .+3
RD      0
ADCOFF  0
        LAC*    ADCOFF  /GET VALUE OF ADC OFFSET
        TCA             /NEGATE THE ADC OFFSET
        DAC     ADCOFF  /SAVE THE NEGATIVE VALUE
        LAW     -2260   /SET COUNTER; 1200(DEC) = 2260(OCTAL)
        DAC     REP     / STORE IT
        LAC*    RD      /GET ADDRESS OF ARRAY
        DAC     RD      / AND STORE
ML      LAC*    RD      /GET SAMPLE
        AND     (177777 /REMOVE THE UPPER TWO BITS
        TAD     ADCOFF  /SUBTRACT THE ADC OFFSET
        TCA             /  TWO'S COMPLIMENT THE RESULT
        DAC*    RD      /AND STORE
        ISZ     REP     /DONE?
        JMP     .+2     /NO, CONTINUE
        JMP*    CNVRT2  /YES, RETURN
        ISZ     RD      /INCREMENT ARRAY POINTER
        JMP     ML      / AND DO NEXT SAMPLE
REP     0
        .END
```

```
      REAL DCVOLT(1200)
      INTEGER COHEN(1200), ADCOFF, CNT, CONT
      NSAMP=1200
400   FORMAT(7H *HELLO)
      WRITE(6,400)
6000  FORMAT(43H HOW MANY PULSES WERE COHERENTLY INTEGRATED)
500   WRITE(6,6000)
6100  FORMAT(I3)
      READ(6,6100) NPULSE
      ADCOFF=NPULSE*128
6200  FORMAT(14H NOT IN SYNC  ,I2)
      CNT=0
90    CONTINUE
      CNT=CNT+1
      WRITE(6,6200) CNT
      CALL INPAD(COHEN,NSAMP,ICOM)
25    CONTINUE
      IF(ICOM.EQ.-513) GO TO 90
      IF(ICOM.NE.1) GO TO 25
      CALL PSYNC(COHEN,NSAMP,IERROR)
      IF(IERROR.NE.1) GO TO 90
      CALL INPAD(COHEN,NSAMP,ICOM)
26    CONTINUE
      IF(ICOM.EQ.-513) GO TO 90
      IF(ICOM.NE.1) GO TO 26
      CALL CNVRT2(COHEN,ADCOFF)
      DO 40 I=1,1200
      DCVOLT(I)=FLOAT(COHEN(I))/FLOAT(NPULSE)*0.019
40    CONTINUE
```

RWVOLT program (cont.)

```
2500    FORMAT(20H * AND THE DATA IS =)
        WRITE(6,2500)
        DO 850 K=1,150
        J1=8*K-7
        J8=8*K
        WRITE(6,450) (DCVOLT(J), J=J1,J8)
        IF(K.EQ.10) K=140
850     CONTINUE
450     FORMAT(1X,8(F6.3,3X))
        WRITE(6,900)
900     FORMAT(2X,47H TYPE 1 TO RERUN PROGRAM, ANY OTHER KEY TO STOP)
        READ(6,6100) CONT
        IF(CONT.EQ.1) GO TO 500
        STOP
        END
```

```
C TEST PROGRAM TO PROCESS COHERENT SCATTER DATA. AVTST2 READS IN KSAMP
C  COHERENT INTEGRATIONS AND THEN AVERAGES THE AUTOCORRELATION FUNCTIONS
C  (LAGS 0 AND 1 ) OVER THE KSAMP SAMPLES. THE PROGRAM THEN CONTINUES
C  TO READ IN BLOCKS OF KSAMP COHERENT INTEGRATIONS AND
C  AVERAGING ALL THE AUTOCORRELATIONS TOGETHER UNTIL ONE MINUTE OF
C  DATA IS AVERAGED.
      REAL AVEPOW(67), AVEVEL(67), RK, IK, RK1, IK1
      REAL RKCHK,IKCHK, RK1CHK, IK1CHK
      REAL RDC(67), IDC(67), RLAGS
      INTEGER ADCOFF, RDATA(68,110), IDATA(68,110), COHEN(1200)
      REAL POWER(67), RACF(67), IACF(67), VEL(67)
      COMMON /A/RDATA,IDATA, COHEN
      COMMON /B/AVEPOW, AVEVEL
      WRITE(6,6000)
      READ(6,6100) NPULSE
      ADCOFF=NPULSE*128
      PULSE=FLOAT(NPULSE)
      FUDGE=7.33/(4.0*3.141596*0.0025*FLOAT(NPULSE))
      WRITE(6,7000) FUDGE
      WRITE(6,8000)
      READ(6,8100) KSAMP
      WRITE(6,8500)
      READ(6,6100) NUMBER
      KF=KSAMP-1
      FKF=FLOAT(KF)
C   MINUTE=(60 SEC * PRF ) / (NUMBER OF PULSES PER COH INT * NUMBER OF
C                COH INT SAMPLED EACH TIME THRU THE LOOP )
      MINUTE=IFIX(24000.0/FLOAT(NPULSE*KSAMP))
      WRITE(6,6100) MINUTE
      DO 1300 NUM=1,NUMBER
      DO 6 I=1,67
      RDC(I)=0.0
      IDC(I)=0.0
      AVEPOW(I)=0.0
```

```
      AVEVEL(I)=0.0
      RACF(I)=0.0
      IACF(I)=0.0
 6    CONTINUE
      DO 1100 MIN=1,MINUTE
      ICOM=100
 10   IF(ICOM.EQ.-513) WRITE(6,9100)
 15   WRITE(6,9000)
C  GET 1200 VALUES FROM THE PREPROCESSOR, CHECKING FOR TIMING ERROR
      CALL INPAD(COHEN,1200,ICOM)
 20   CONTINUE
      IF(ICOM.EQ.-513) GO TO 10
      IF(ICOM.NE.1) GO TO 20
C  CHECK FOR SYNCHRONIZATION, IF OUT OF SYNC TRY AGAIN.
      CALL PSYNC(COHEN,1200,IERROR)
      IF(IERROR.NE.1) GO TO 15
      WRITE(6,9300)
C  CALL INPAD AND PROCESS COHERENTLY INTEGRATED DATA KSAMP NUMBER OF TIMES
      DO 500 K=1,KSAMP
      CALL INPAD(COHEN,1200,ICOM)
 30   CONTINUE
      IF(ICOM.EQ.-513) GO TO 10
      IF(ICOM.NE.1) GO TO 30
C  REMOVE THE MEM MSB USED FOR SYNCING AND SUBTRACT THE ADC OFFSET
      CALL CNVRT2(COHEN,ADCOFF)
C  SAVE THE 136 COH. INT. VALUES CORRESPONDING TO 70 TO 80 KM IN THE
C  ARRAY PDATA. INDEX I = HEIGHT, K = SAMPLE
      DO 300 I=1,68
      J=2*I+731
      J1=J+1
      RDATA(I,K)=COHEN(J)
      IDATA(I,K)=COHEN(J1)
 300  CONTINUE
 500  CONTINUE
```

```
      DO 700 K=1,KF
      DO 600 I=1,67
      RDC(I)=RDC(I)+FLOAT(RDATA(I,K))
      IDC(I)=IDC(I)+FLOAT(IDATA(I,K))
600   CONTINUE
700   CONTINUE
C  POWER = SQR( SUM OF SQURES ); EACH ALTITUDE IS AVERAGED OVER KSAMP SAMPLES
C  AVERAGE OVER THE KSAMP SAMPLES:
C    RACF(I)=REAL PART OF THE AVERAGE CROSS CORRELATION, LAG 1
C    IACF(I)=IMAGINARY PART OF THE AVE CROSS CORRELATION, LAG 1
750   DO 900 K=1,KF
      DO 800 I=1,67
      K1=K+1
      RK=FLOAT(RDATA(I,K))-(RDC(I)/FKF)
      RKCHK=ABS(RK/PULSE*0.019)
      IF(RKCHK.LE.0.005) RK=0.0
      IK=FLOAT(IDATA(I,K))-(IDC(I)/FKF)
      IKCHK=ABS(IK/PULSE*0.019)
      IF(IKCHK.LE.0.005) IK=0.0
      RK1=FLOAT(RDATA(I,K1))-(RDC(I)/FKF)
      RK1CHK=ABS(RK1/PULSE*0.019)
      IF(RK1CHK.LE.0.005) RK1=0.0
      IK1=FLOAT(IDATA(I,K1))-(IDC(I)/FKF)
      IK1CHK=ABS(IK1/PULSE*0.019)
      IF(IK1CHK.LE.0.005) IK1=0.0
      AVEPOW(I)=AVEPOW(I)+RK**2+IK**2
      RACF(I)=RACF(I)+RK*RK1+IK*IK1
      IACF(I)=IACF(I)+RK*IK1-RK1*IK
800   CONTINUE
900   CONTINUE
      DO 1000 I=1,67
      RDC(I)=0.0
      IDC(I)=0.0
1000  CONTINUE
1100  CONTINUE
```

```
C  VELOCITY =  ( LAMBDA/4*PI) / (1/8) ) * PHASE AT LAG 1 . K GIBBS' THESIS P.21
C  AVEVEL = AVERAGE VELOCITY AT A GIVEN ALTITUDE.
      WRITE(6,9800)
      WRITE(6,9810)
      DO 1200 I=1,67
      RATIO=SQRT((IACF(I))**2+(RACF(I))**2)/AVEPOW(I)
      RLAGS=RACF(I)/AVEPOW(I)
      IF(RATIO.LE.0.1) AVEV=0.0
      IF(RATIO.GT.0.1) AVEV=FUDGE*ATAN2(IACF(I),RACF(I))
      ALT=69.85+(FLOAT(I)*0.15)
      WRITE(6,9900) ALT,AVEPOW(I),AVEV, RATIO, RLAGS, RACF(I), IACF(I)
      RDC(I)=0.0
      IDC(I)=0.0
 1200 CONTINUE
 1300 CONTINUE
 2000 FORMAT(2X,13HALTITUDE (KM),6X,5HPOWER,6X,14HVELOCITY (M/S))
 3000 FORMAT(2X,F6.2,9X,F9.1,5X,F6.2)
 6000 FORMAT(42HHOW MANY PULSES WERE COHERENTLY INTEGRATED)
 6100 FORMAT(I3)
 7000 FORMAT(2X,9H FUDGE = ,F5.2)
 7100 FORMAT(F5.2)
 8000 FORMAT(42H HOW MANY COHERENT INTEGRATIONS TO AVERAGE)
 8100 FORMAT(I3)
 8500 FORMAT(35H HOW MANY ONE MINUTE AVERAGES TO DO)
 9000 FORMAT(42H SYNCHRONIZING THE PREPROCESSOR WITH INPAD)
 9100 FORMAT(13H TIMING ERROR)
 9300 FORMAT(12H NOW IN SYNC)
 9800 FORMAT(20H ONE MINUTE AVERAGES)
 9810 FORMAT(2X,13HALTITUDE (KM),6X,5HPOWER,6X,14HVELOCITY (M/S))
 9900 FORMAT(2X,F6.2,9X,E10.3,5X,F5.2,4X,F5.2,2X,F5.2,2(3X,E10.3))
      STOP
      END
```

APPENDIX III

PREPROCESSOR USER GUIDE

III.1 *Introduction*

The preprocessor is designed to operate as independently of the coherent-scatter radar system as possible. Only one control signal from the radar director is required. Consequently two preprocessor operating parameters must be set by the operator.

Section two of this appendix describes the preprocessor's inputs and outputs. Also described is how to compute the operating parameters and operation of the control switches.

Section three gives a step by step procedure for operating the preprocessor in the Urbana coherent-scatter radar system.

III.2.1 *Analog Input Signals*

There are two identical analog input lines on the preprocessor, one for each of the coherent-scatter phase detector outputs. The lines are RG-174U coaxial cable. Each input line is tied directly to the input of a DATEL-INTERSIL SHM-HU sample-and-hold. The S/H requires a source impedence of 51Ω or less. The phase detector output impedence is several ohms hence it can be connected directly to the S/H inputs. If a filter is to be used on the phase detector output however, this impedence matching requirement must be considered. The S/H has a minimum input voltage range of ±2.5V.

III.2.2 *START1*

START1 is a TTL level digital input signal. The input port is a coaxial cable connector on the backplane of the preprocessor. The signal is used to synchronize the preprocessor data collection with the radar direc-

tor. When the preprocessor microprogram has initialized the preprocessor to begin a new coherent integration, the preprocessor system clock, (SYS CLK), is disabled and a wait state is entered. When the START1 line goes low the system-clock is restarted and data acquisition begins immediately. (START1 must be low for a minimum of 150 ns to insure the SYS CLK is restarted). For preprocessor operation connect the START1 input to the Echo Sample Window, ESW, output of the radar director. The ESW is a gated 100 khz square wave. Two thumbwheel switches on the radar director set the beginning and ending time in tens of microseconds of the ESW pulse train. To generate just one negative-going pulse, as required by the preprocessor, the thumbwheel switches should be set to consecutive times. The beginning time determines at what altidue the preprocessor begins sampling. A setting of 10 (100 s), the lowest possible, corresponds to 15 km ($d = \frac{1}{2}ct$). The preprocessor will then take 600 altitude samples, spaced every 0.15 km, a 90 km altitude region. This 90 km window of samples can be moved by changing the time at which the ESW pulse occurs.

III.2.3 *Front Panel Switches*

III.2.3.1 Z_{TIME}. The top row of twelve single-pole double-throw toggle switch set the binary value loaded into the time counter, TIME CTR. This determines the number of data words which will be transferred to the main computer each time the microprogram's subroutine is entered. For an Interpulse Period (IPP) of 2500 μs there is approximately 1800 μs between the end of data collection for one pulse and the next transmit pulse (see Figure 3.4). Each data word requires 12.3μs to transmit hence no more than 146 data words can be transmitted each time the subroutine is entered. If the IPP is shortened the maximum number of data word transfers per IPP is

reduced also.

III.2.3.2 *Z_{PULSE}.* The eight single-pole double-throw toggle switches labeled Z_{PULSE} load the pulse counter, PULSE CTR., with the binary number of transmit pulses which will to be coherently integrated. The coherent integration time, Z_{PULSE} times the interpulse period, is determined by this setting. Any number between ten and 255 may be selected. At least ten pulses must be coherently integrated so as to allow enough time for the preprocessor to transfer the coherently integrated values to the main computer.

III.2.3.3 *Control Switches.* There are three single-pole double-throw toggle switches on the lower left of the front panel labeled; PULSE SINGLE STEP, FREE RUN/SINGLE STEP, and RESET/RUN. The FREE RUN/SINGLE STEP switch controls the preprocessor's system clock. In the FREE RUN position the system clock is free running; the preprocessor's state is determined by its microprogram. When switched to the SINGLE STEP position the system clock is inhibited. A single clock period can now be gated onto the system clock line by toggling the PULSE SINGLE STEP switch high.

The RESET/RUN switch initializes the preprocessor. When turned on the preprocessor comes up in a random state. RESET sets the microprogram counter to zero and properly initializes the necessary flip-flops.

III.2.3.4 *Preprocessor Data / HP5610A ADC Selection Switches.* The six toggle switches on the lower right of the front panel control the nineteen lines connected to the PDP-15 interface. When these switches are up the ten data lines, four channel identification lines and data ready flag from the HP6510A are connected to the PDP-15. These switches must be up when any experiment using the Hewlett-Packard ADC is running. The four remaining lines are user accessible at BNC connectors on the preprocessor

backplane. When the six switches are down all nineteen lines carry preprocessor data (sixteen data line, memory address MSB, data ready flag and one unused) to the interface.

III.3 *Preprocessor Operation Outline*

This outline presents instruction on how to connect and operate the preprocessor using the Urbana coherent-scatter radar system. It is assumed the coherent-scatter radar system is running with an interpulse period (IPP) of 2500 μs and a one-eight second coherent integration time is desired.

1. Connect the phase detector outputs to the preprocessor analog input. Analog channel one is stored in odd memory locations, analog channel two is stored in even memory locations. Which phase detector output is connected to which preprocessor input depends upon the velocity algorithum of the collection program. For AVTST2 connect the COS phase detector output to channel one and the SIN phase detector output to channel two.
2. On the radar director, set the echo sample window (ESW) thumbwheel switches to 10 and 11.
3. Connect the ESW output to the preprocessor START1 input (coaxial connector on preprocessor's backplane).
4. Set the PDP-15 interface switches down, connecting the preprocessor lines to the interface.
5. Set the Z_{TIME} switches to 080_{HEX}.
6. Set the Z_{PULSE} switches to 32_{HEX}.
7. Turn on the preprocessor's power supplies. One AC switch is used for all the DC supplies. Make sure the LED indicator light for each of the four DC power levels lights.
8. Set the RUN/SINGLE STEP switch to SINGLE STEP.

9. Set the PULSE SINGLE STEP switch low.
10. Toggle the RESET/RUN switch to RESET, then leave in the RUN position.
11. The four microinstruction address LEDs should be off indicating the microprogram counter is set to zero.
12. If the microinstruction being executed when the preprocessor system clock was disabled (part 5) is a jump instruction (1010 or 1000 appears on microinstruction LED display) then the microinstruction address will not get reset to zero by step 7. In this case toggle the PULSE SINGLE STEP switch (once or twice) until the microinstruction changes. Now toggle the RESET/RUN switch again. The microinstruction address should be set to zero.
13. Set the RUN/SINGLE STEP switch to run. The preprocessor is now operating.

REFERENCES

Allman, M. E., and S. A. Bowhill (1976), Feed system design for the Urbana incoherent-scatter radar antenna, *Aeron. Rep. No. 71*, Aeron. Lab., Dept. Elec. Eng., Univ. Ill., Urbana-Champaign.

Evans, J. V., (1969), Theory and practice of ionospheric study by Thomson scatter radar, *Proc. IEEE 57*, 496-530.

Gibbs, K. P., and S. A. Bowhill (1979), The Urbana cohernet-scatter radar: synthesis and first results, *Aeron. Rep. No. 90*, Aeron. Lab Dept. Elec. Eng., Univ. Ill., Urbana-Champaign.

Hess, G. C., and M. A. Geller (1976), The Urbana meteor-radar system - Design development and first observations, *Aeron. Rep. No. 74*, Aeron. Lab., Dept. Elec. Eng., Univ. Ill., Urbana-Champaign.

Rastogi, P. K., and S. A. Bowhill (1976a),, Scattering of radio waves from the mesosphere - I. Theory and observations, *J. Atmos. Terr. Phys., 38*, 399-411.

Rastogi, P. K., and S. A. Bowhill (1976b), Scattering of radio waves from the mesosphere - II. Evidence for intermittent mesospheric turbulence, *J. Atmos. Terr. Phys., 38*, 449-462.

Rastogi, P. K., and R. F. Woodman (1974), Mesospheric studies using the Jicamarca incoherent-scatter radar, *J. Atmos. Terr. Phys., 36*, 1217-1231.

Villars, R., and V. F. Weiskopf (1955), On the scattering of radio waves by turbulent fluctuations of the atmosphere, *Proce. IRE 43*, 1232-1239.

Woodman, R. F., and A. Guillen (1974), Radar observations of the winds and turbulence in the stratosphere and mesosphere, *J. Atmos. Sci., 31*, 493-503.

Woodman, R. F., R. P. Kugel, and J. Rottger, A coherent Integrator - Decoder Preprocessor for the SOUSY-VHF Radar Arecibo Observatory, 1979.