EXPERIMENTAL APPARATUS FOR MEASURING THE STATISTICS OF THE RECEIVED INTENSITY FOR OPTICALLY SPATIALLY FILTERED LASER RADIATION PERTURBED BY SPECKLE AND TURBULENCE

Todd Lewis Cloninger
A.B., Warren Wilson College, 1984

A thesis submitted to the faculty
of the Oregon Graduate Center
in partial fulfillment of the
requirements for the degree
Master of Science
in
Applied Physics

March 1989

The thesis "Experimental Apparatus for Measuring the Statistics of the Received Intensity for Optically Spatially Filtered Laser Radiation Perturbed by Speckle and Turbulence," by Todd Lewis Cloninger has been examined and approved by the following Examination Committee:

J. Fred Holmes, Advisor
Professor

Richard A. Elliott
Professor

Paul R. Davis
Professor

ACKNOWLEDGMENTS

My foremost thanks belong to God for His help in completing this thesis and the associated research. His sustaining presence has been an indispensable source of encouragement during the entire process. To God be the glory!

I am grateful to my advisor, Dr. J. Fred Holmes, for his guidance and dedication. His experience and knowledge of the empirical results of prior research was particularly important in testing and troubleshooting the system. I was glad to be able to take my questions to Dr. Holmes because of his ability to provide clear, concise explanations.

John Hunt provided technical guidance, primarily in electronics design and construction. He designed and executed most of the construction of the driver circuitry for the acousto-optic modulator. Some of the diagrams in this paper are edited copies of his drawings. He also wrote the machine language routines that controlled the analog-to-digital converter. It was a pleasure to learn from him.

Libo Sun did much of the preliminary work on this project. He also recorded and processed data before compiling the results. His Ph.D. dissertation describes how theory involving turbulence was used to accurately predict the empirical results.

I am greatly indebted to Beverly Kyler for her efforts in preparing this paper. In my absence, she keyed my work into the department's computer, allowing me to take advantage of its photo-typesetting capabilities.

I also would like to thank my fellow students who provided help on a number of matters along the way. Finally, I cannot forget the support provided by my family and friends.

Table Of Contents

List of Figures

ABSTRACT

EXPERIMENTAL APPARATUS FOR MEASURING THE STATISTICS OF THE RECEIVED INTENSITY FOR OPTICALLY SPATIALLY FILTERED LASER RADIATION PERTURBED BY SPECKLE AND TURBULENCE

Todd Lewis Cloninger

Oregon Graduate Center, 1989

Supervising Professor: J. Fred Holmes

Speckle-turbulence interaction has the potential for allowing single ended remote sensing of the path averaged strength of turbulence (structure constant) along the line of sight to a remote object. If a laser transmitter is used to illuminate a diffuse object in the atmosphere, the resultant speckle field is randomly perturbed by the atmospheric turbulence as it propagates back to the location of the transmitter-receiver. Consequently, the variance of the received intensity contains information about the strength of atmospheric turbulence. Unfortunately, the fluctuations due to speckle and those due to the turbulence cannot be separated; and therefore the variance of the received intensity cannot be used very effectively to remotely sense the strength of turbulence. However, by util-

izing optical spatial filtering before measuring the received intensity, the strength of turbulence can be determined over three or more orders of magnitude. As part of a larger research project, the work for this thesis involved the design, construction and test of an appropriate laser transmitter; the construction and test of an optical/electronic receiver; and the development of interactive computer software for aligning the system, and automatically collecting and processing data. Good experimental results were obtained with the system.

I. INTRODUCTION

Uneven local heating of terrestrial surfaces by the sun produces thermal gradients. These thermal variations and the associated turbulent mixing of the air constitute fluctuations in the air's index of refraction. For more than 20 years scientists have known that the speckle (or intensity scintillations) seen in laser light that has traversed such a medium is a function of the wind speed and turbulence level [1]. Much effort has been devoted to the construction of devices that use laser light to measure wind velocity and turbulence [2]. Unfortunately, placing the laser transmitter and receiver at opposite ends of a long optical path is often inconvenient and sometimes impossible. In addition, thermal gradients in the atmosphere (via the associated gradients in refractive index) bend the laser beam. As changing meteorological conditions alter these thermal gradients and the amount of bending, the orientation of the laser transmitter must be adjusted so that it continues to point at the receiver.

One approach to overcoming these problems is to fold the optical path using a diffuse reflector. Placing the transmitter and receiver at the same location is generally more convenient and sometimes essential. Since the light will travel over the same path to and from the diffuse target, the bending effects of thermal gradients will cancel.

Because coherent light reflected from a diffuse target produces a speckle pattern associated with the roughness of the surface, another problem arises. Any movement of the beam across the target produces a changing speckle pattern that adds to the speckle resulting from turbulence. This thesis describes an experimental optical transmitter-receiver system used to evaluate the effects of placing a high-pass spatial filter in the receiver focal plane. The spatial filter is expected to remove the speckle generated by the beam wander since it contains lower spatial frequencies than the speckle resulting from turbulence. The spatial filter also blocks the light which was not deflected by turbulence. For zero turbulence conditions, no signal should reach the detector. Consequently, instead of measuring small changes in a relatively large signal, the system measures an offset from zero.

II. DESIGN OF APPARATUS FOR DATA COLLECTION

Rather than creating a complete data collection system, this work involved the evaluation and modification of existing hardware. Because emphasis was given to using as much of the existing system as practical, a chronological account of its design and construction would not provide a clear and concise description of the design challenges. Instead, this chapter attempts to provide a systematic explanation of the apparatus in terms of three component subsystems: Transmitter, Target, and Receiver. Figure 1 on the following page shows the transmitter and target.

A. Transmitter

A 5 mW helium-neon (He-Ne) laser (having a wavelength of 632.8 nm) is the source used to illuminate a spot on the target. The laser light is pulsed at 100 kHz in order to distinguish it from background light. The pulsing of the laser is accomplished by redirecting the beam with an acousto-optic modulator (AOM) while the laser operates in a continuous mode. When no power is supplied to the AOM, the beam from the laser is blocked before entering the transmitting telescope. With a 40 MHz signal applied to the AOM, a large fraction of the light is deflected so that it passes through the telescope and is focused on the target. Diagrams of the circuits that generate the 40 MHz signal modulated at 100 kHz are contained in the first three figures in Appendix A.

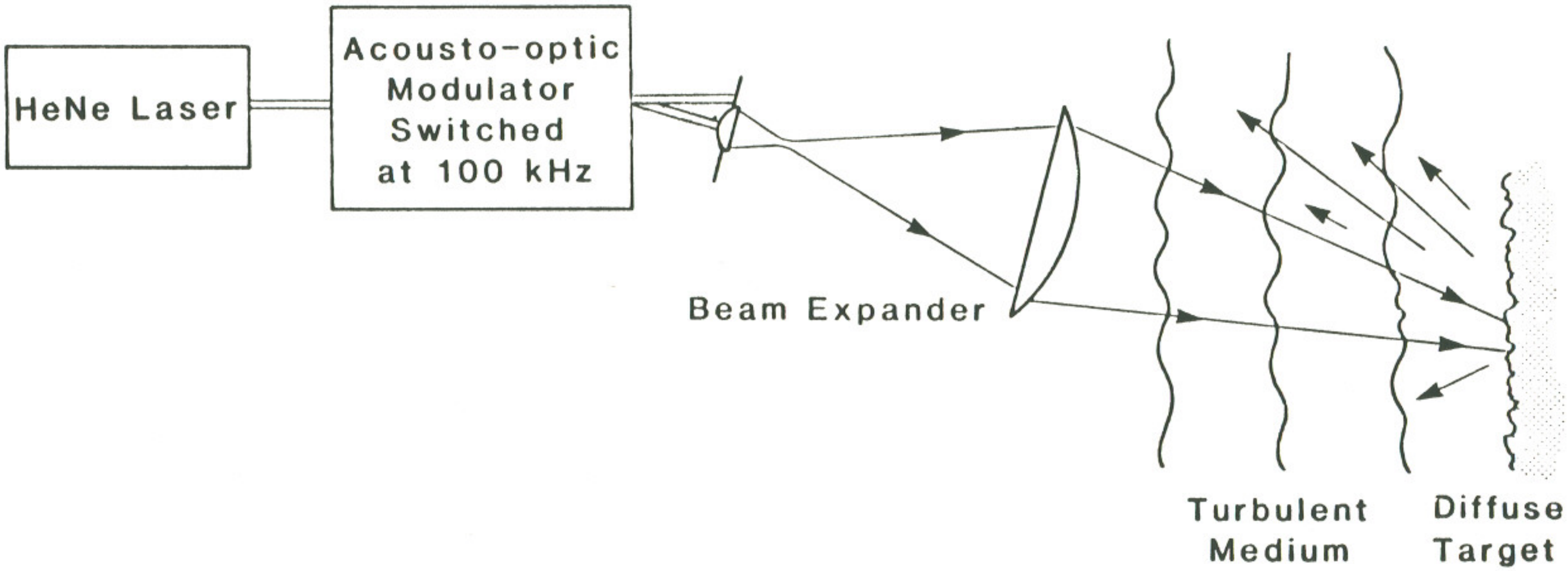


Figure 1. Transmitter Diagram

Since the laser beam has an output diameter of 0.8 mm and a beam divergence of 1 mrad, a telescope must be used to produce a small spot on the target which is located hundreds of meters away. A Keplerian telescope was used because it produces a real focus, which is a necessity if the beam requires spatial filtering. The expression for the radius of a Gaussian beam as a function of the propagation distance is written:

$$w(z) = w_0 \sqrt{1 + \left(\lambda z \,/\, \pi w_0^2\right)^2} \tag{1}$$

where w(z) is the radius of the beam a distance z from the beam waist, w_0 is the radius of the beam waist, and λ is the wavelength of the light (632.8 nm for the He-Ne laser). By assuming that the beam is focused on the target, the waist is then located at the target and it is possible to estimate the size of the beam required at the transmitter. Solving equation 1 for w_0 yields:

$$w_0^2 = \frac{w^2}{2} \pm \left[\frac{w^4}{4} - \left(\frac{\lambda z}{\pi}\right)^2\right]^{\frac{1}{2}} \tag{2}$$

Requiring that w_0 have a real value for its radius places restrictions upon w at the transmitter.

$$w \geq \left(\frac{2\lambda z}{\pi}\right)^{\frac{1}{2}} \tag{3}$$

With the target at 290 meters, w ≥ 10.8 mm.

In order for an aperture to transmit 99% of the light in a Gaussian beam, its diameter must be a little more than three times the radius estimated above. Consequently, the transmitting telescope required an output lens with a diameter of at least 33 mm. For a laser with a beam radius of 0.409 mm, the telescope must expand the beam by a factor of at least 26.4 to produce an output beam with a radius of 10.8 mm or more.

From geometrical optics, the focal length of the second lens is the product of the beam expansion factor and the focal length of the first lens. These considerations together with component availability and pricing information prompted the selection of a 20X microscope objective with a 9.0 mm effective focal length as the first lens and a 50 mm diameter plano-convex lens with a 302 mm focal length as the second lens. The second lens had a surface accuracy of $\lambda/8$ and a 40-20 scratch-dig surface quality. Experimental tests revealed that poorer quality lenses introduced beam distortions that were unacceptable for the long path lengths involved.

Some equations developed by Sidney A. Self [3] provide a method for approximating the characteristics of a Gaussian beam as it propagates through a lens. (Diffraction effects arising from the lens aperture are not considered.) The relationship between the location of the input and output beam waists is given by:

$$\frac{1}{s + z_R^2 / (s - f)} + \frac{1}{s'} = \frac{1}{f} \tag{4}$$

Where s is the distance from the input waist to the lens, s' is the distance from the lens to the output waist, f is the focal length of the lens, and z_R is the Rayleigh length: $z_R \equiv \pi w_0^2/\lambda$. Equation 4 can be rearranged to give an expression for s'.

$$s' = f \, \frac{s^2 - sf + z_R^2}{(s - f)^2 + z_R^2} \tag{4a}$$

Self also provides an equation for the magnification (m) produced by the lens.

$$m = \frac{w'_0}{w_0} = \frac{1}{\sqrt{[(s/f) - 1]^2 + (z_R/f)^2}} \tag{5}$$

The parameters w_0 and w'_0 are the radii of the input and output beam waists respectively. Equation 5 can be rearranged to provide a formula for the output beam waist.

$$w'_0 = \frac{fw_0}{\sqrt{[s - f]^2 + z_R^2}} \tag{5a}$$

With a 0.409 mm beam waist (w_0) at the laser's output mirror, the 20X microscope objective (f = 9.0 mm) located 420 mm from the laser will produce an intermediate waist with a radius of 3.97 μm located 9.04 mm from the microscope objective.

By adjusting the position of the output lens along the optic axis, the size of the spot at the target may be minimized. The distance from the intermediate waist to the output lens must be known before the size of the waist at the target can be calculated. Rearranging equation 4 and using the quadratic equation to solve for s produces:

$$s = f + \frac{f^2}{2(s' - f)} \pm \left[\left(\frac{f^2}{2(s' - f)} \right)^2 - z_R^2 \right]^{1/2} \tag{6}$$

Notice that there are two values of s that will produce the same value of s'. This means that there are two positions for the output lens that will produce a waist at the target. With s' = 290 m, f = 302 mm, and w_0 (used to calculate z_R) = 3.97 μm; s may be either 302.29 mm or 302.02 mm. Using s = 302.29 mm in equation 5a leads to the conclusion that the waist at the target will be 3.94 mm. (The other value of s produces a larger waist at the target.)

B. Target

A target covered with Scotchlite (a material manufactured by 3M) was positioned approximately 290 m from the transmitter and 1.7 m (5 to 6 feet) above the ground. Scotchlite is composed of a layer of tiny glass spheres with a reflective backing that directs much of the incident light back toward its source while destroying the spatial coherence of the beam. The retro-reflective properties of the Scotchlite produce an effective gain in light intensity at the receiver of about 1000 over ordinary diffuse surfaces.

Because of the relatively short path length used to test this system for measuring turbulence, the beam did not wander across the target enough to produce a fully developed speckle pattern at the receiver. The problem was eliminated by using a motor to rotate a disk covered with Scotchlite. A surface speed of roughly 1 cm/second for the beam on the rotating target compensated for the insufficient beam wander.

C. Receiver

Since the light returning from the target is not spatially coherent, geometrical optics was applied in the design of the receiver subsystem illustrated on page 9. A Newtonian telescope collected the light; its parabolic

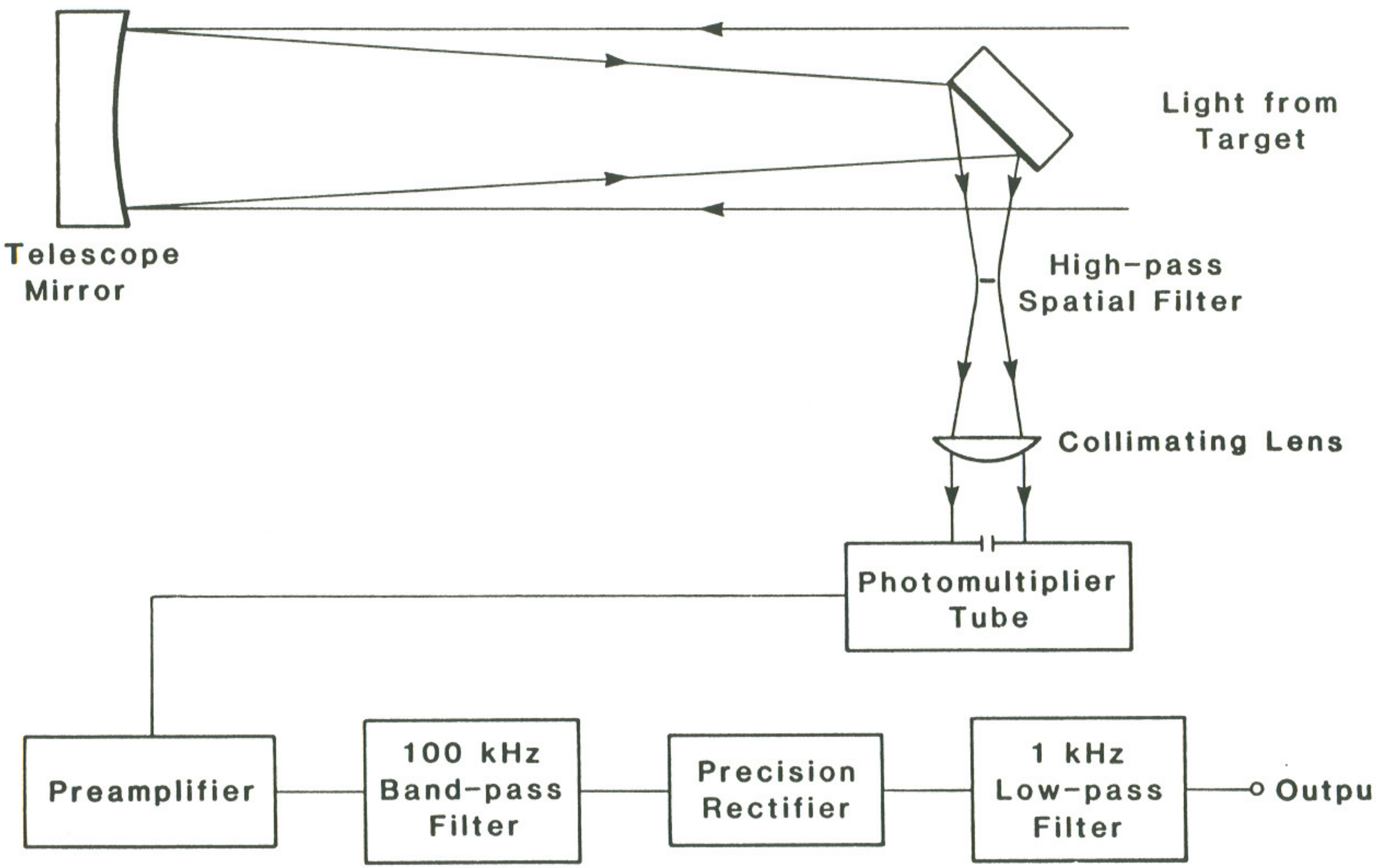


Figure 2. Receiver Diagram

primary mirror was 43 cm (17 inches) in diameter with a 2 meter focal length. The magnification resulting from the mirror is given by

$$M = \frac{f_m}{L_o - f_m} \tag{7}$$

With an object distance (L_o) of 290 m and a focal length (f_m) of 2 m, a magnification of 0.0069 results. The 12 mm diameter spot on the target containing 99% of the signal power produces a circular region 83 μm in diameter in the image plane. During experimentation, a high-pass spatial filter was positioned in the image plane. The spatial filters were circular pieces of anti-reflection coated optical glass with an opaque metallic spot centered on the input surface. The opaque disks ranged from 50 to 300 μm in diameter for different spatial filters. A 3 axis translation stage helped position the spot on the spatial filter to cover the image of the laser spot on the target. Only light deflected by turbulence could pass the spatial filter and reach the detector. A circular stop was also placed in the image plane to reduce background light. The opening in the stop had to be large enough to permit the deflected laser light to pass through.

An anti-reflection coated plano-convex lens collimated the light before it reached the detector. Using a collimating lens with a focal length of 76 mm (3.0 inches), the telescope provided a beam reduction factor of 26. The detector sampled a small portion of the speckle field roughly 76 mm from the collimating lens.

The photomultiplier tube, used as the detector, was mounted in a black light-tight box. The size of the opening for admitting light to the detector could be changed by attaching different apertures to the front of the box. The apertures were made by drilling holes (ranging from 0.1 mm to 3.2 mm in diameter) in aluminum strips and then painting the strips flat black. An optical filter with 3 nanometer pass band was mounted inside of the box to reduce the amount of background light reaching the photomultiplier tube. The photomultiplier tube was a type R446 manufactured by Hamamatsu. Using a tube with a high quantum efficiency at 632.8 nm resulted in a relatively high photocathode current. Since shot noise in the first stage of the photomultiplier tube was a major source of noise at low light levels, the increased current improved the signal to noise ratio. A diagram of the detector circuit is located in Appendix A.

The signal conditioning circuitry is diagrammed in Figure A-3.1 of Appendix A. The preamplifier stage contains two operational amplifiers (op-amps). The first is in a transimpedance configuration to convert the current signal provided by the photomultiplier tube into a voltage signal for further processing. The second op-amp in the preamplifier stage produces a signal gain of 20. An op-amp configured as a voltage follower buffers the signal and provides a way to attach an oscilloscope for monitoring the signal. The 100 kHz bandpass filter employs 3 op-amps in an active filter design to realize a bandwidth of 5 kHz. Since the laser light is modulated at 100 kHz, the signal passes through the filter without attenuation while most of the background noise is removed. This signal is also buffered and made available for monitoring. The precision rectifier stage incorporates an a.c. coupled op-amp for introducing a signal gain of 20. The rectifier uses two op-amps and two Schottky diodes to rectify the high frequency signal even at low voltages. The final stage of the signal processing circuitry is an op-amp configured as a low-pass filter. This filter attenuates frequencies greater than 1 kHz since they are not important to this experimental investigation.

The output of the signal processing circuitry can be connected to an oscilloscope or an analog-to-digital converter (ADC) in a computer. In this case, a Micro PDP 11/73 with a 12 bit ADC was used to record the signal. Since no clock card was available for this computer, data conversion was triggered by an external pulse generator. The ADC's multiple input channels enabled it to digitize signals from more than one source during data acquisition. Campbell Scientific's CA-9 Space Averaging Anemometer generated reference values for comparison with the experimental results. The CA-9 supplied the computer with analog signals corresponding to the wind velocity (across the path of the beam) and the standard deviation of the log of the field amplitude (σ_x) which is related to the turbulence. A 4 milliwatt He-Ne laser in the field near the target (it was kept 2 meters away to prevent it from interfering with the experimental system) was aimed at the CA-9 to provide a light source for its operation.

III. RECORDING DATA

Recording experimental data was a fairly complex process because of the large number of components in the system as a whole. Failure of any element produces meaningless data. In response to this problem, the data acquisition software contains many prompts to the user to carry out the correct procedure. Prior to use of the program, the following steps need to be carried out:

1. Supply power to computer, terminal, and tape drive.
2. Turn on transmitter and allow laser to warm up.
3. Turn on oscilloscope (used for monitoring signals).
4. Turn on clock pulse generator.
5. Turn on signal conditioning circuitry.
6. Block photomultiplier tube aperture.
7. Turn on high voltage supply for photomultiplier tube.
8. Place laser used with CA-9 unit in the field near the target.
9. Turn on laser and aim it at the CA-9 unit.
10. Turn on rotating target.
11. Check alignment of the laser transmitter and receiving telescope.

With these steps complete, the software may be used.

The user should be logged in to a privileged account from the system console to use this software. The flowcharts on the following three pages outline the operation of the data acquisition software. To initiate operation type @RECORD. The instructions in the indirect command file RECORD.CMD direct the PDP 11/73 to set the necessary memory partition and prompt the user to insert the tape for storing the data. Details of the format used in storing data on tape are provided at the beginning of Appendix B. (Appendix B also contains listings of the software used for data acquisition and data processing.) Once the tape is successfully loaded, control is passed to a FORTRAN program named RECDAT. If the tape being used already contains some data, the user must respond that it is necessary to advance the tape. In that case, the tape will be forwarded until it reaches the end of data block; otherwise, data will be written beginning at the first of the tape, erasing any preexisting data.

There are numerous questions which must be answered before any data can be recorded. This information is stored in the header block at the beginning of each file for use in processing the data or analyzing system performance under a variety of conditions. The questions are generally self-explanatory with default responses appearing in parentheses. The user is permitted to enter a text comment which will be stored in the header block; this comment should contain any information that may affect the system's operation, for example weather conditions. If the automatic naming feature of the data processing program is to be used, then the first line of this comment should contain the name to be used for the file. Upon completion of

Flow Chart For Data Acquisition

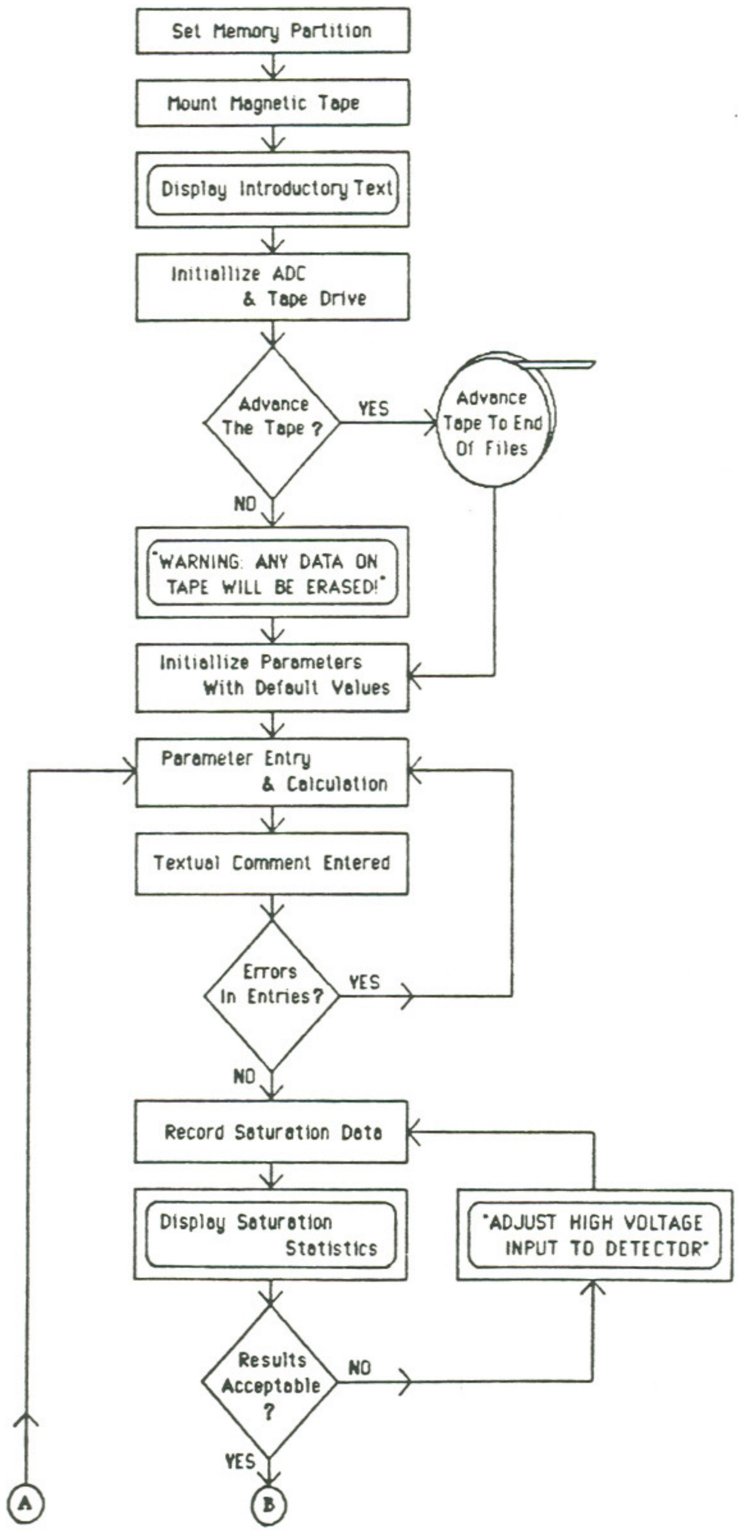


Figure 3.1

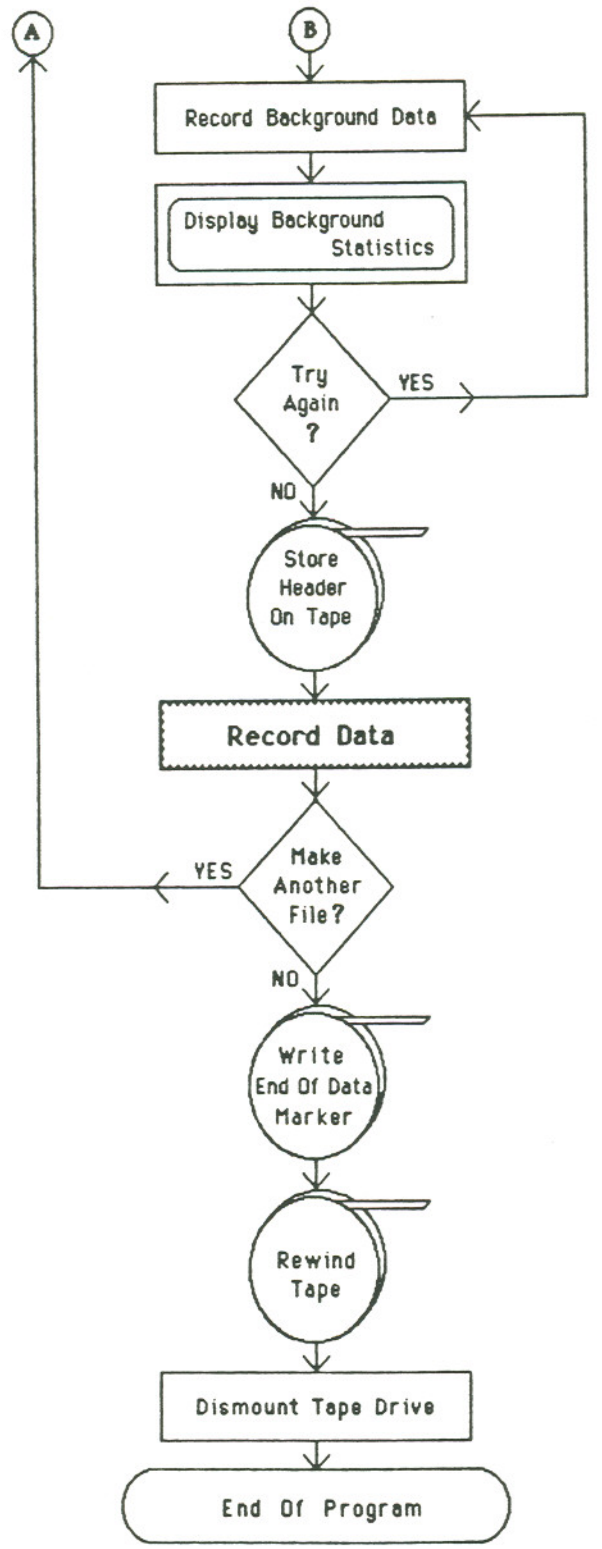
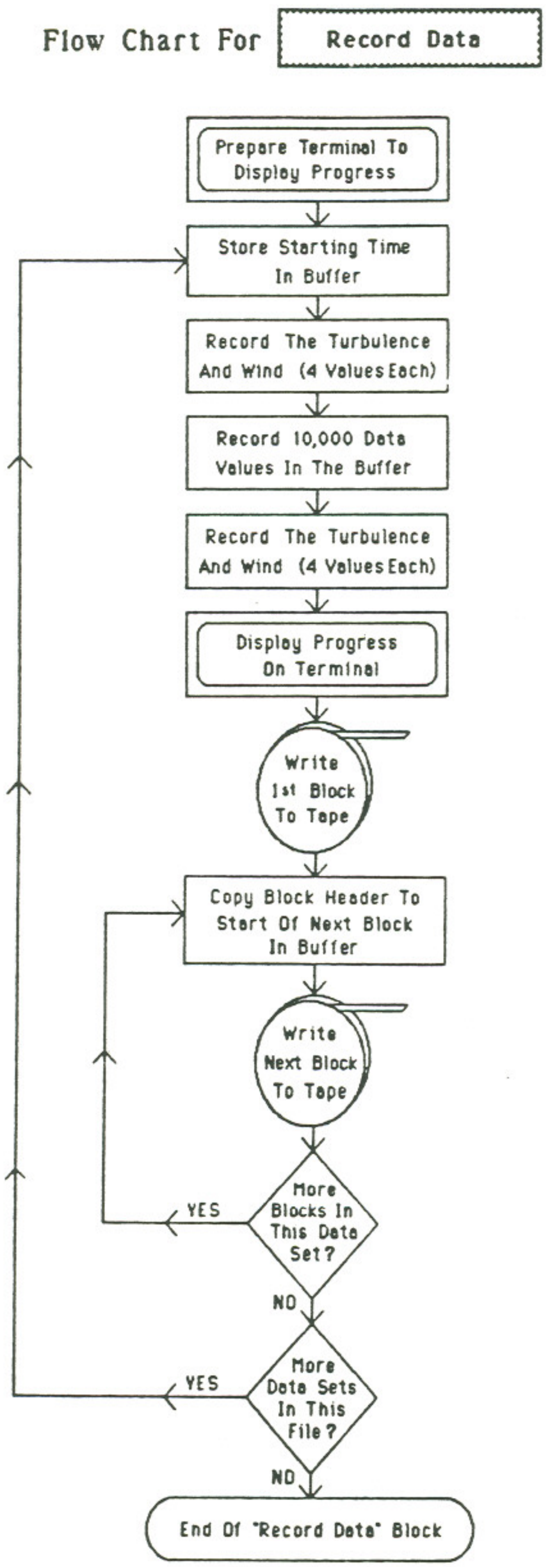

A
B
Record Background Data
Display Background Statistics
Try Again ?
YES
NO
Store Header On Tape
Record Data
Make Another File?
YES
NO
Write End Of Data Marker
Rewind Tape
Dismount Tape Drive
End Of Program

Figure 3.2



Flow Chart For
Record Data
Prepare Terminal To Display Progress
Store Starting Time In Buffer
Record The Turbulence And Wind (4 Values Each)
Record 10,000 Data Values In The Buffer
Record The Turbulence And Wind (4 Values Each)
Display Progress On Terminal
Write 1st Block To Tape
Copy Block Header To Start Of Next Block In Buffer
Write Next Block To Tape
More Blocks In This Data Set?
YES
NO
More Data Sets In This File?
YES
NO
End Of "Record Data" Block

Figure 3.3

the comment entry, the user is given the option to correct any errors made in entering the parameters. Background and saturation statistics must also be stored in the header block. Saturation is measured with the system operating normally, with the exception that the spatial filter is moved to one side enough to allow most of the laser light to reach the detector. These conditions produce the maximum signal that the detector will see. By using the oscilloscope and computer generated statistics the high voltage to the photomultiplier tube is adjusted to achieve the desired gain. The voltage should be gradually increased from a relatively low initial value since excessive current can damage the detector. The circuitry was designed to operate in the range of 0 to 5 volts. If the gain is too high, the signal will be clipped. There is a trade off since lower gain means that less of full range of the ADC is used and decreased resolution results. For background readings the spatial filter is returned to its normal position and the transmitting laser is blocked. The user may repeat the background recording if there was a problem. At this point the completed header block is copied to the tape.

The program prompts the user to unblock the laser before the actual recording of the data begins. Every file contains an integer number of data sets, each of which contains ten data blocks. Each data block stores 1000 values from the digitized signal. The digitized values are loaded into memory by a machine language routine (listed in Appendix B). Ten blocks of 1000 values can be loaded into memory simultaneously. The time is recorded prior to recording each data set. Four values for both the wind velocity and standard deviation signals from the CA-9 are recorded before and after each data set is loaded into memory. The computer stops recording data while a data set is transferred to tape; this fact has implications for data processing since there are time intervals missing from the signal between adjacent data sets.

The computer displays its progress as it records data. Movement should be kept to a minimum during data acquisition since vibrations will have an effect upon the optical equipment. Once the computer has completed recording the data, it offers the user the option of recording another file of data. If no more files are requested, a special "end of data" block (described in Appendix B) is recorded on the tape.

IV. PROCESSING DATA

The data processing program (named stat) was written in FORTRAN on a MicroVAX II. The data tape should be loaded into the computer before the program is started. This program is menu driven; the flow chart in Figure 4.1 on the following page shows the options available from the menu. Figure 4.2 on page 22 shows details of how the 'db' option reads and displays one block of data from the tape. The 'rw' option rewinds the tape. The 'sk' option advances the tape the number of blocks specified by the number entered with the option (default of one). The tape is advanced by reading individual blocks. A special command exists for forwarding or reversing the tape, but it did not operate correctly in the version of the operating system that was installed. The 'nf' options operates in a similar fashion; it searches for a header block, indicating the beginning of the next file. The 'td' option transfers blocks of data from the tape to the computer's hard disk without changing the format of the information. The 'tg' option converts the data to a format used for graphing data.

Figure 4.3 contains the flow chart of the 'mv' option. This option produces three disk files: a text file containing the statistical values calculated, a graph file containing the log of the turbulence (as measured by the CA-9) versus the log of the normalized mean of the data, and a graph file containing the log of the turbulence versus the log of the normalized variance of the data. These files are given the same name, but different suffixes: ".val",

Flow Chart For Data Processing

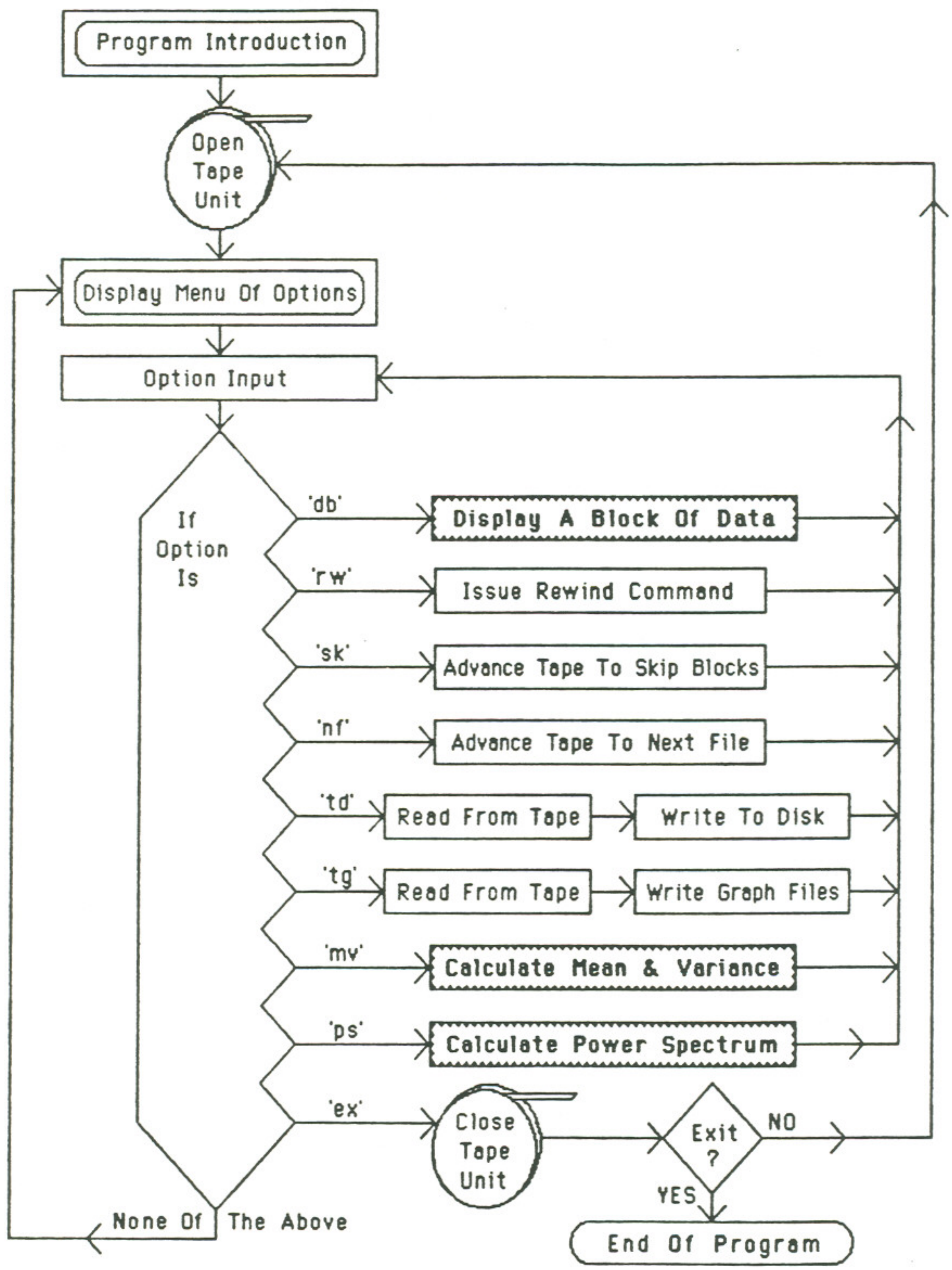


Figure 4.1

Flow Chart For Display A Block Of Data

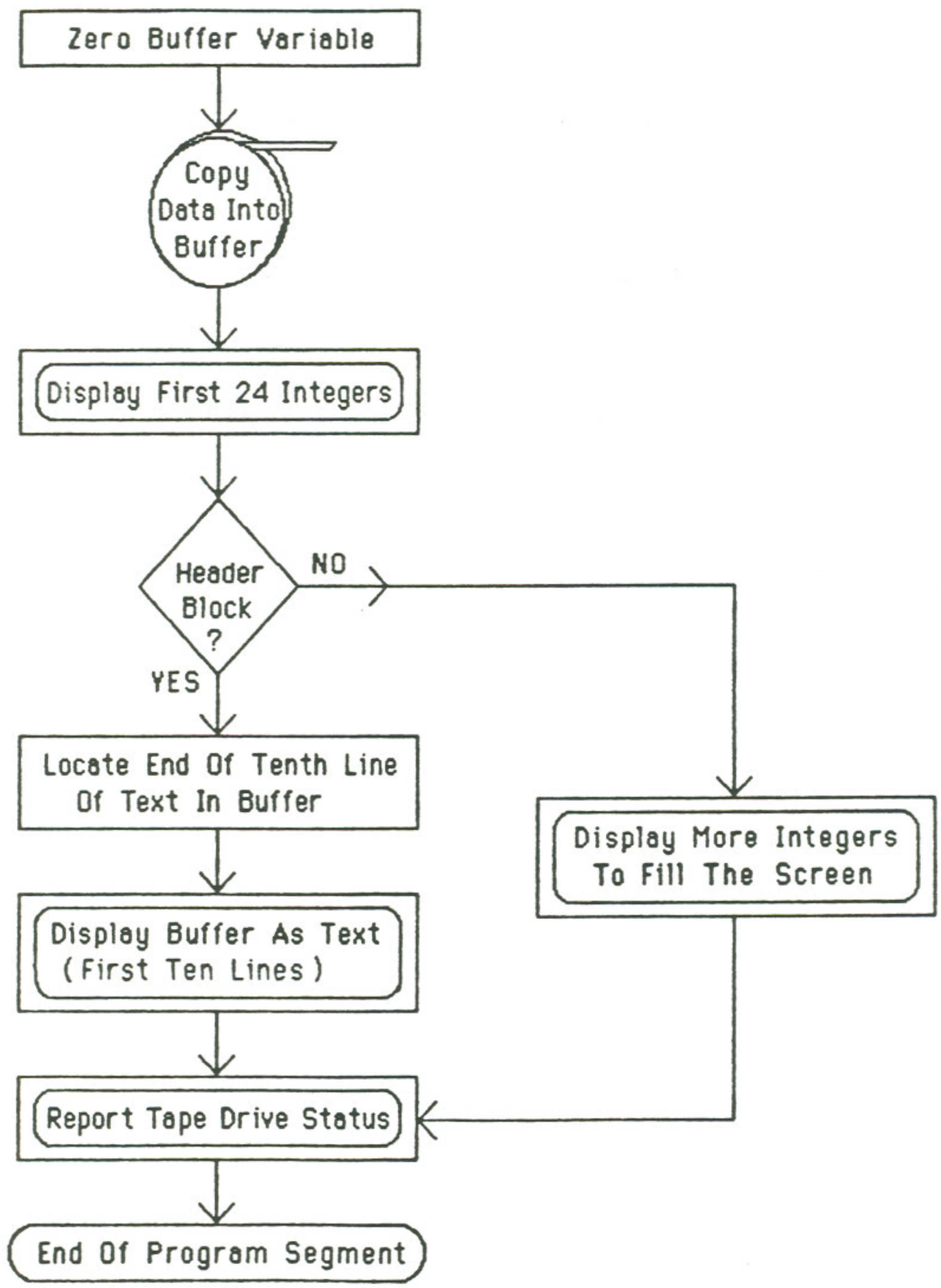


Figure 4.2

Flow Chart For Calculate Mean & Variance

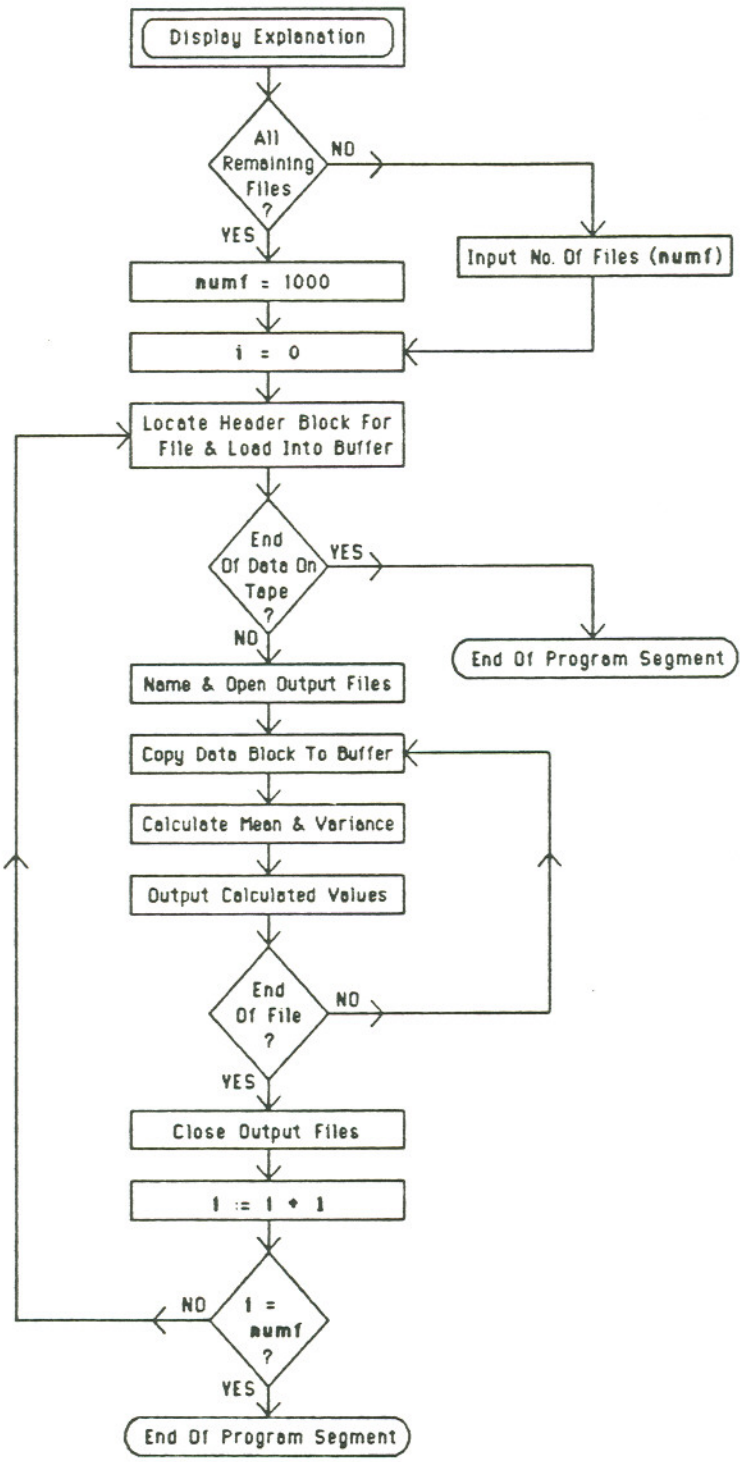


Figure 4.3

Flow Chart For Calculate Power Spectrum

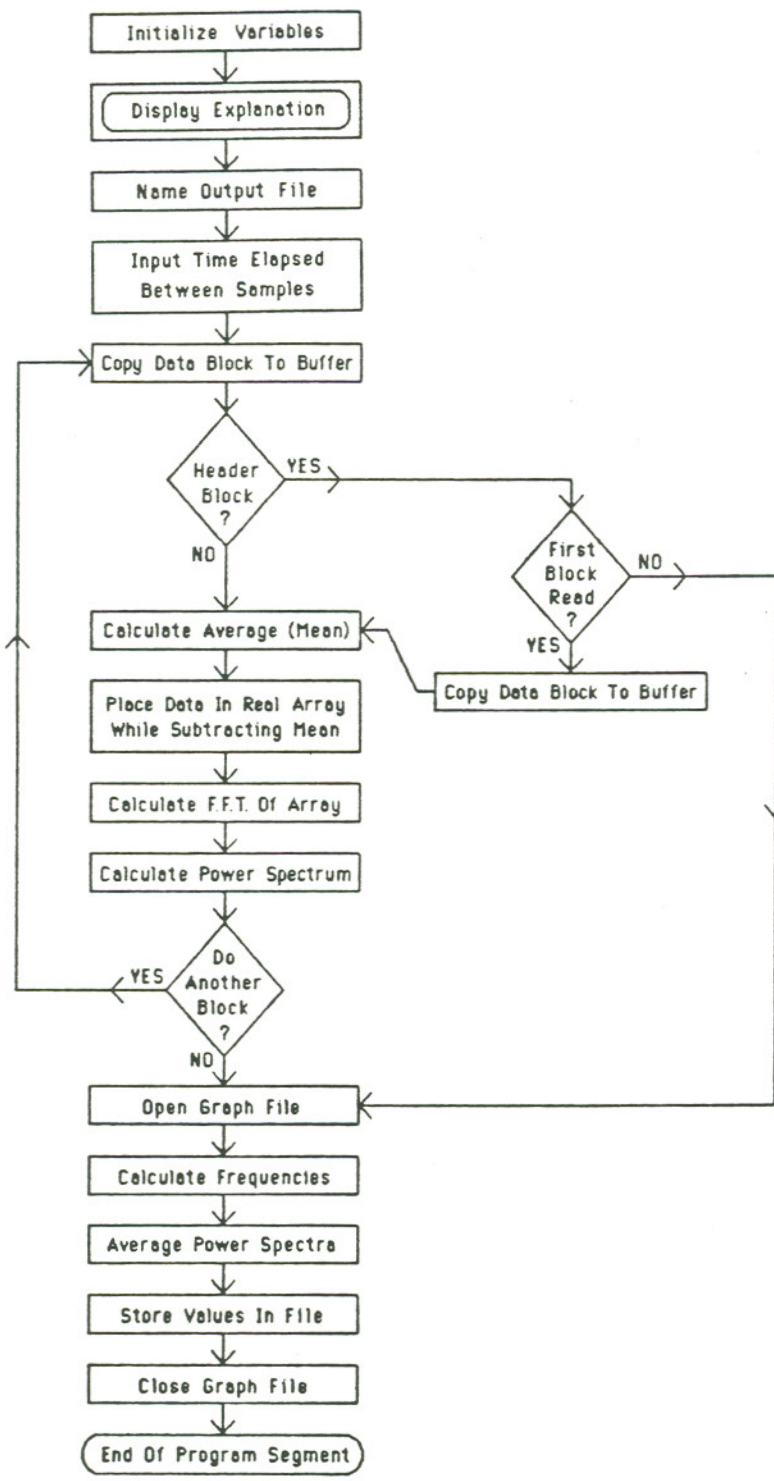


Figure 4.4

".lnm", and ".nv" rspectively. The normalization is carried out as follows.

$$\text{Mean}_{\text{Norm.}} = \frac{\text{Mean} - \text{Mean}_{\textbf{Background}}}{\text{Mean}_{\textbf{Saturation}} - \text{Mean}_{\textbf{Background}}} \tag{8}$$

$$\text{Variance}_{\text{Norm.}} = \frac{\text{Variance} - \text{Variance}_{\textbf{Background}}}{(\text{Mean} - \text{Mean}_{\textbf{Background}})^2} \tag{9}$$

These normalized values are calculated for each block of data within a data set. The ten values are then averaged to produce a number for the data set as a whole.

The 'ps' option calculates the power spectrum of the data by using a FFT routing in the NAG library. The spectrum is generated for individual blocks and then the spectra are averaged together for a user specified number of blocks. This option is useful in determining the required sampling rate.

A utility program named TAPDIR helps keep track of data collected on the tapes. This FORTRAN program generates directory listings for the data tapes. The brief listings are useful to store with the tape; the full listings provide more detailed information that may be kept in a notebook. The listings are written to disk files which must be printed by the user. Appendix B also contains a listing of this program beginning on page 74.

V. RESULTS AND RECOMMENDATIONS

Before beginning experimentation, a test was conducted to insure that the equipment produced results consistent with the results of prior research. The test consisted of recording and processing data when the system was operating without the spatial filter in place. If the system operates correctly, the normalized variance [4,5] should be very near unity, indicating a saturated speckle pattern. Low values for the normalized variance may be corrected by increasing the path length or rotating the target. The problem might also be signal averaging which results from using too large of an aperture on the detector. It is undesirable to make the aperture any smaller than necessary, since the signal reaching the detector would be reduced unnecessarily.

The time interval between samples should be as large as possible without losing information important to the results. More time between samples means that for given amounts of sampling time there will be less data to store and process. Figure 5 on the next page shows a typical power spectrum of a data file recorded with the system. Since there is little of interest taking place at frequencies greater than 500 Hz, a sampling rate of 1000 samples per second was used in these experiments. Further investigation may show that an even narrower frequency range gives acceptable results.

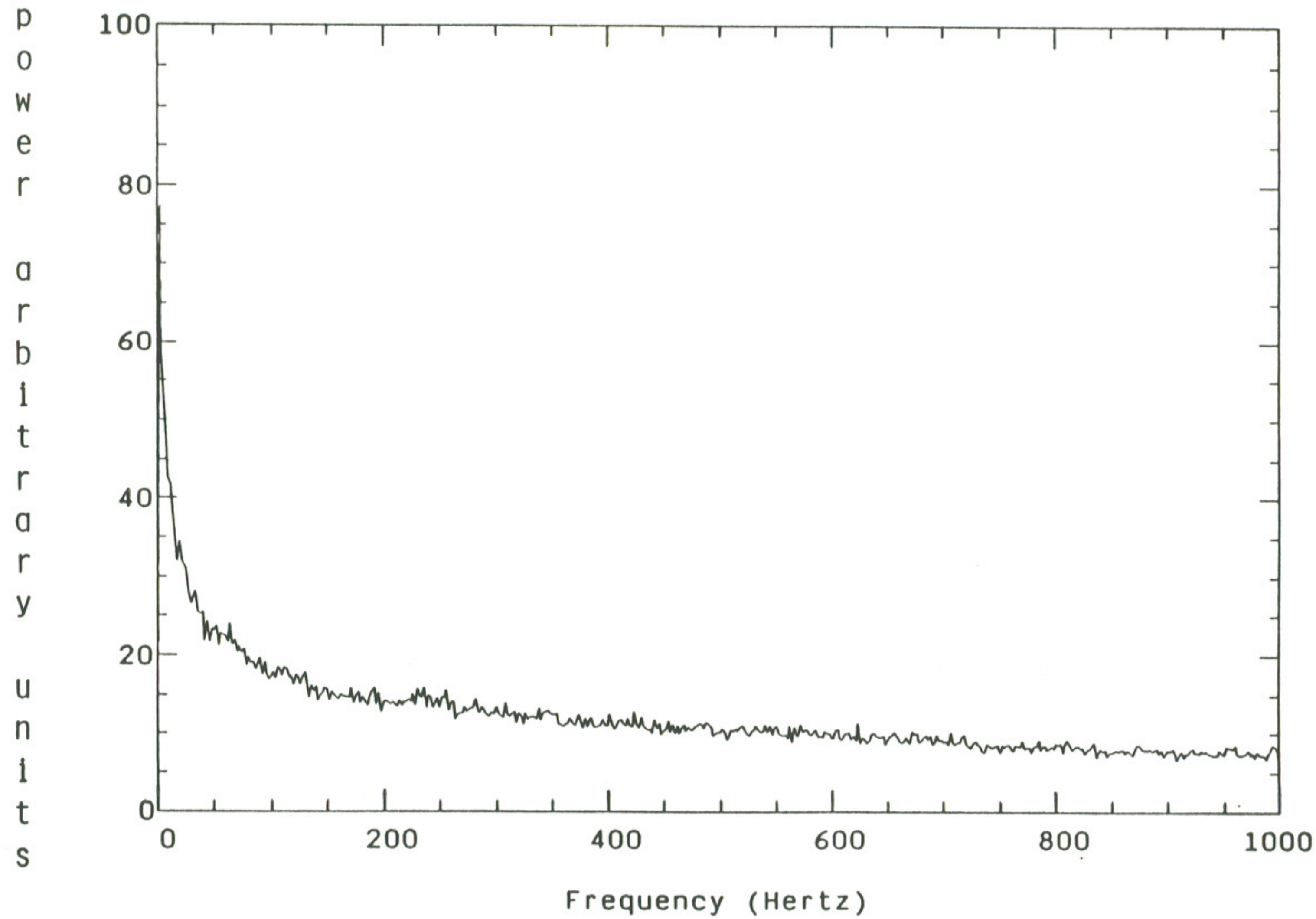


Figure 5. A Typical Power Spectrum

The actual experimentation involved recording data under a variety of conditions in order to provide data for a wide range of turbulence levels. The lowest turbulence levels occur a little after dusk and around dawn. Fog and rain interfere with operation of the system because of their effect on the laser beam. On one occasion a gentle breeze carried the fine mist from a sprinkler ten meters or more into the path of the beam. Although the mist was virtually invisible without the laser light, it appeared as a cloud under laser illumination and had a major impact upon the system's operation. The graph in Figure 6 contains a summary of the data collected. The horizontal axis measures the log of the variance supplied by the CA-9 anemometer; the vertical axis measures the log of the normalized sample mean which indicates what fraction of the light reaches the detector. Each dot represents an average of 10,000 data values recorded over a 10 second time interval. The theoretical curve shown on the graph was generated by Libo Sun [6]. There is a definite correlation between the results predicted theoretically and those achieved experimentally. The discrepancies indicate that either the theory or the experimental technique and equipment need refinement. Because of the random nature of the experiment, the scatter of the points could be reduced by increasing the length of time for which the data was averaged.

There are a number of hardware changes that might reduce the noise in the system. First, the laser power output was not constant; during a few minutes it would change by more than ten percent. With these fluctuations taking place, the saturation measurements are not constant. Beside the option of purchasing a more stable laser, the power output of the laser could

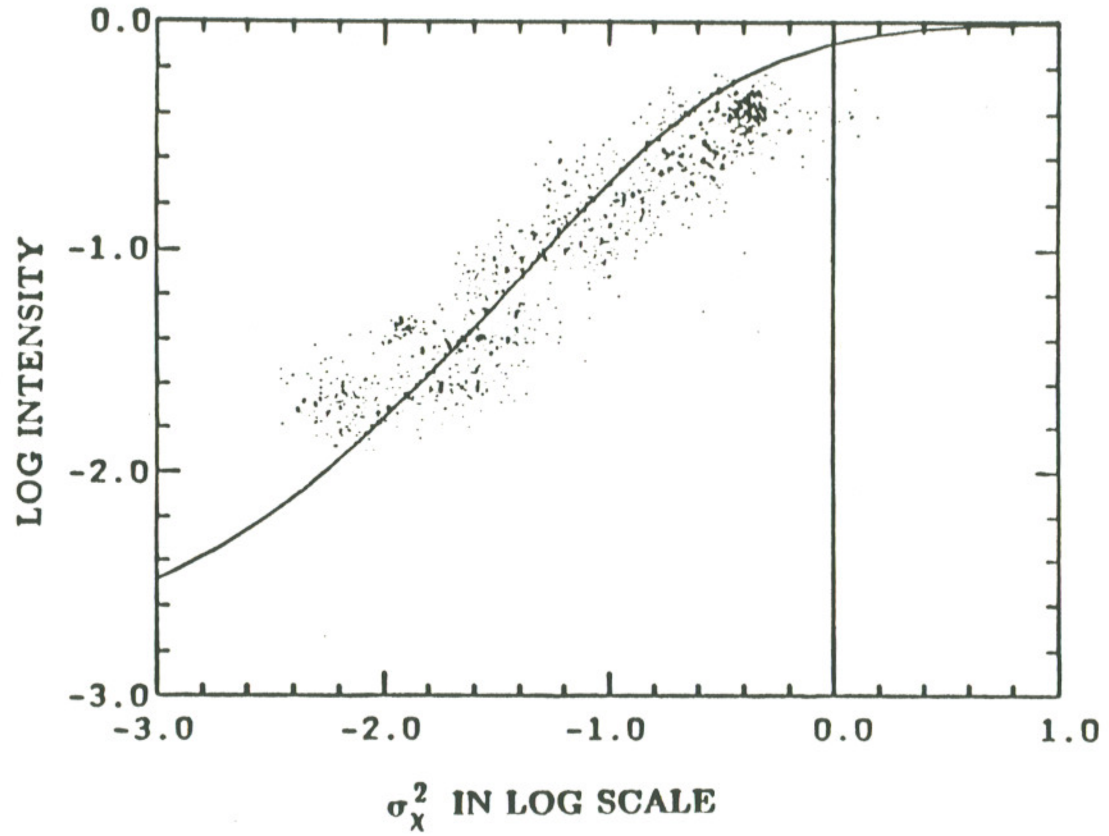


Figure 6. Graph Of Results

be recorded along with the data to allow for corrections later. Devices are available to regulate the output of lasers by measuring the power and attenuating the beam to produce the desired level. Using a spectrum analyzer to examine the output of detectors illuminated by the laser light reveals that there are noise spikes shifting up and down the frequency spectrum. When a component of the noise has a frequency of 100 kHz, it passes through the band-pass filter in the signal processing circuit. This noise, which seems to be the result of mode competition within the laser, is particularly severe while the laser is warming up. Using a laser with an etalon or mode locking should eliminate this source of noise. Replacing the He-Ne laser with a semiconductor laser could also solve this problem and simplify the transmitter design, since the output of a semiconductor laser can be modulated by modulating its input current.

Within the receiver subsystem, the largest source of noise is shot noise in the first stage of the photomultiplier tube. Because shot noise is a function of the square root of the current, the signal to noise ratio (current not power) increases as the square root of the photocathode current. In other words, increasing the current by a factor of four would double the signal to noise ratio. Selecting a photomultiplier tube with a higher quantum efficiency (ratio of electrons produced to number of incident photons) could increase the signal to noise ratio by about 40 percent. If a shorter wavelength of light were used, even greater gains could be achieved--higher quantum efficiencies are available in that spectral region. Other options include increasing the transmitter power or using a larger aperture on the photomultiplier tube, but these possibilities are limited by safety considerations and aperture averaging, respectively.

Although the experimental apparatus described in this thesis did not exhibit optimal performance, it adequately fulfilled its purpose by experimentally verifying the results predicted by theory. This system, however, cannot establish the performance limitations imposed by theory and current technology. A practical (rugged and easy to operate) device employing these principles cannot be produced without significant hardware modifications.

VI. REFERENCES

1. D. H. Hohn, "Effects of Atmospheric Turbulence on the Transmission of a Laser Beam at 6328Å," Appl. Opt., **5,** No. 9, September 1966, pp. 1427-1436.

2. R. S. Lawrence, G. R. Ochs, and S. F. Clifford, "Use of Scintillations to Measure Average wind Across a Light Beam," Appl. Opt., **11,** No. 2, February 1972, pp. 239-243.

3. Sidney A. Self, "Focusing of Spherical Gaussian Beams," Appl. Opt., **22,** No. 5, March 1983, pp. 658-661.

4. J. Fred Holmes, Myung Hun Lee, and J. Richard Kerr, "Effect of the log-amplitude covariance function on the statistics of speckle propagation through the turbulent atmosphere," J. Opt. Soc. Am., **70,** April 1980.

5. Myung Hun Lee, J. Fred Holmes, and J. Richard Kerr, "Statistics of speckle propagation through the turbulent atmosphere," J. Opt. Soc. Am., **66,** November 1976.

6. Libo Sun, Ph.D. dissertation, "The Effect of Optical Spatial Filtering on the Statistics of Laser Radiation Propagating Through the Turbulent Atmosphere," Oregon Graduate Center, Beaverton, Oregon, September 1988.

APPENDIX A

CIRCUIT DIAGRAMS

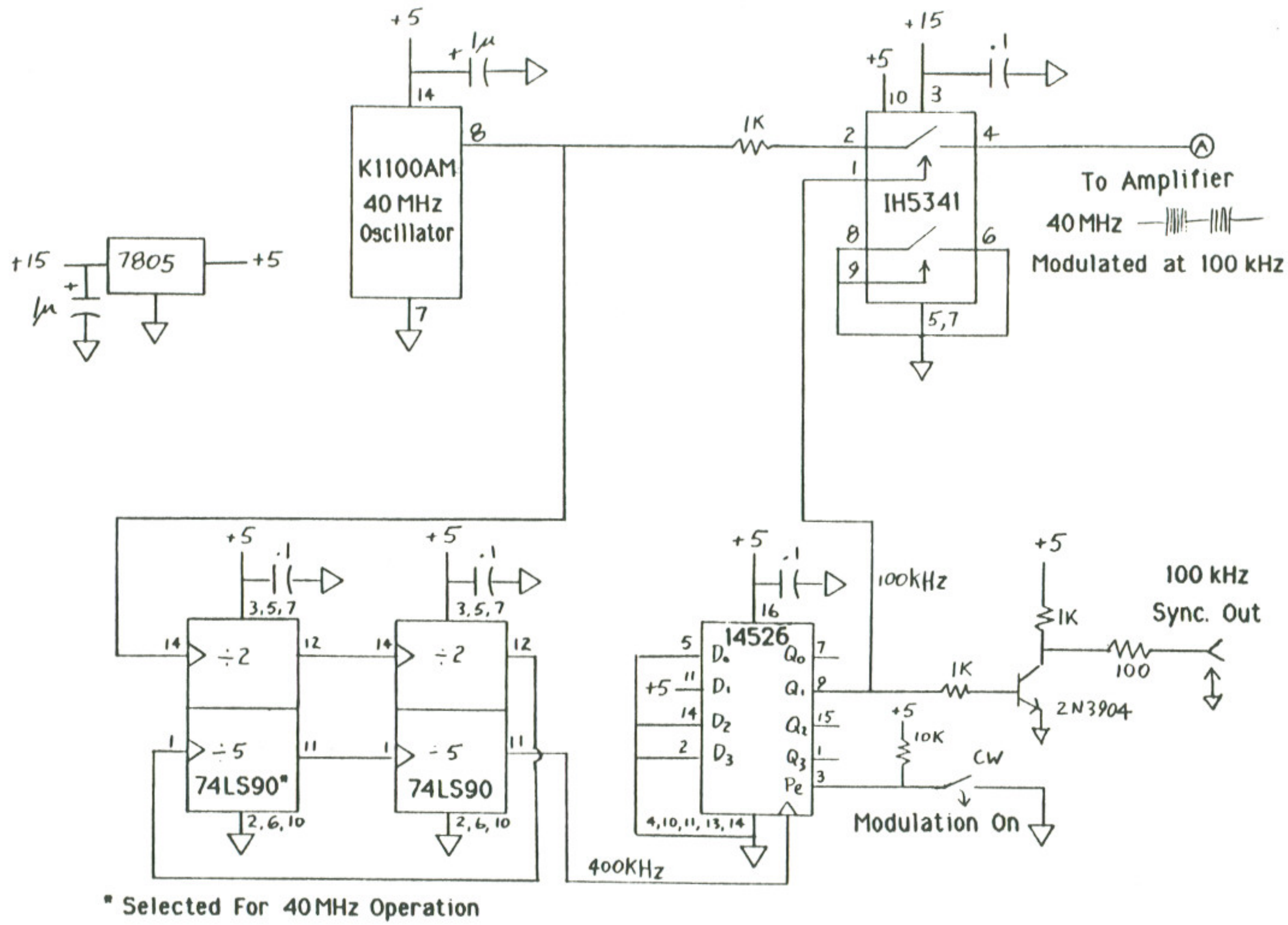
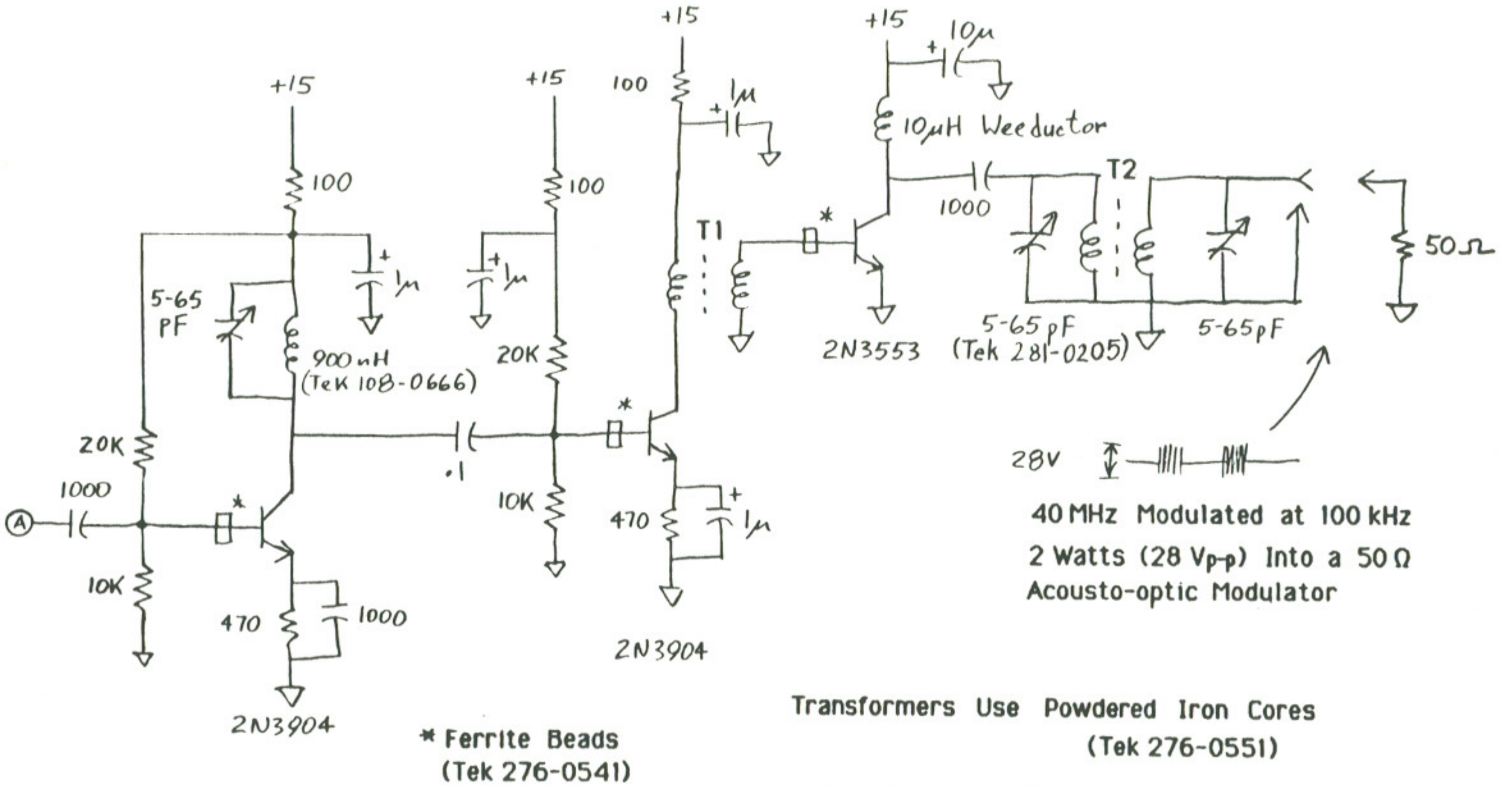


Figure A-1.1 Driver Circuit For AOM



Transformers Use Powdered Iron Cores
(Tek 276-0551)

T1: Primary 9 Turns #32
Secondary 3 Turns #26

T2: Primary 9 Turns #32
Secondary 9 Turns #32

Figure A-1.2 Amplifier Circuit For AOM

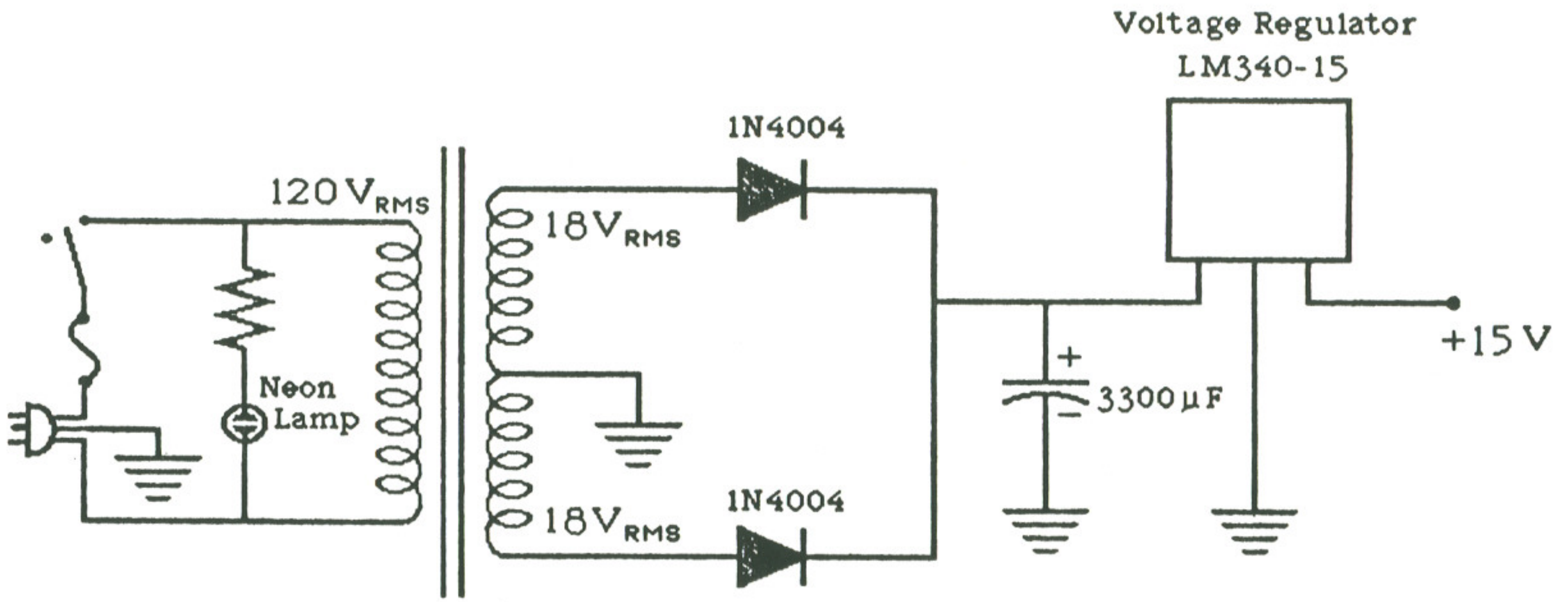


Figure A-1.3 Power Circuit For AOM

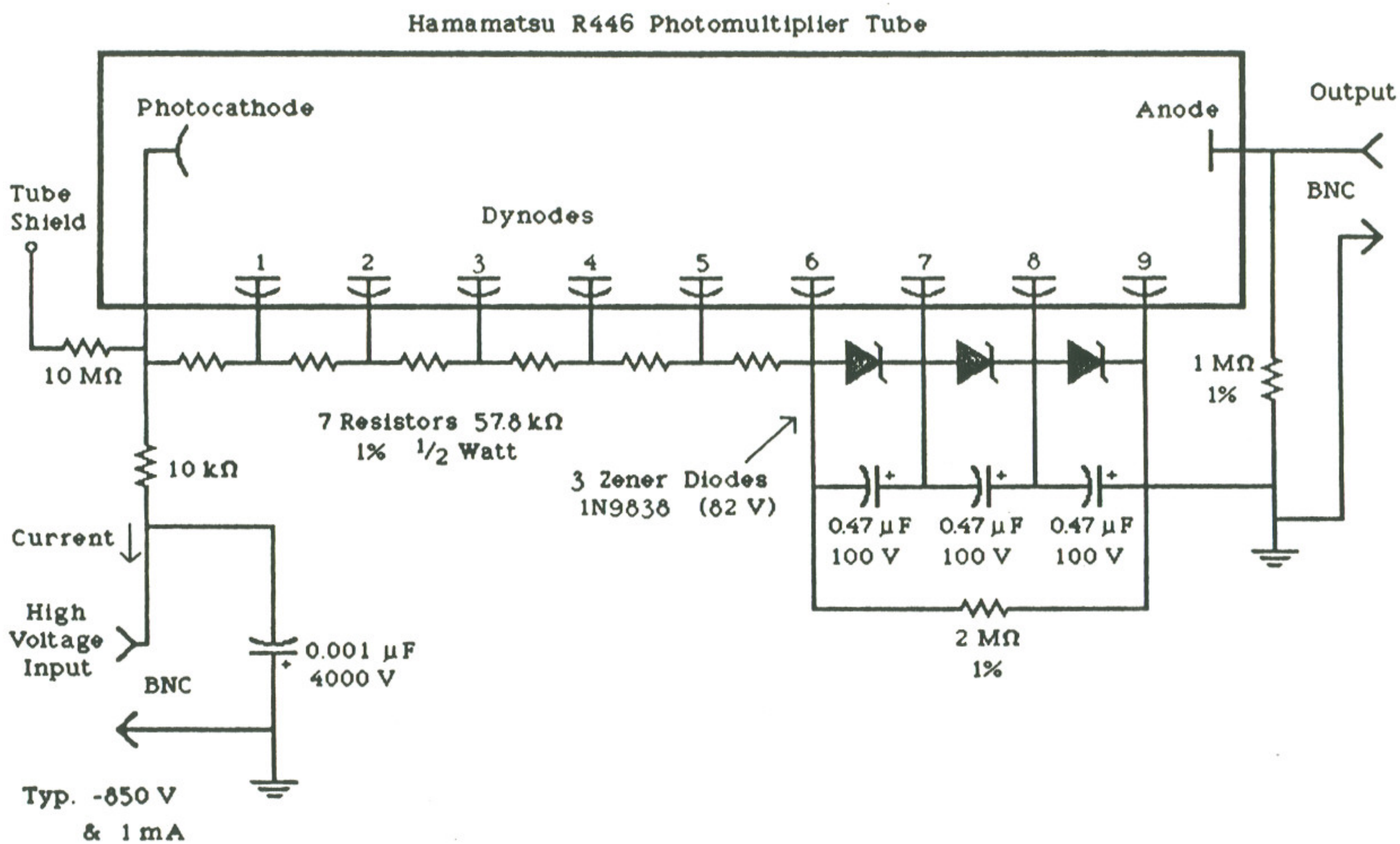


Figure A-2. Detector Circuit

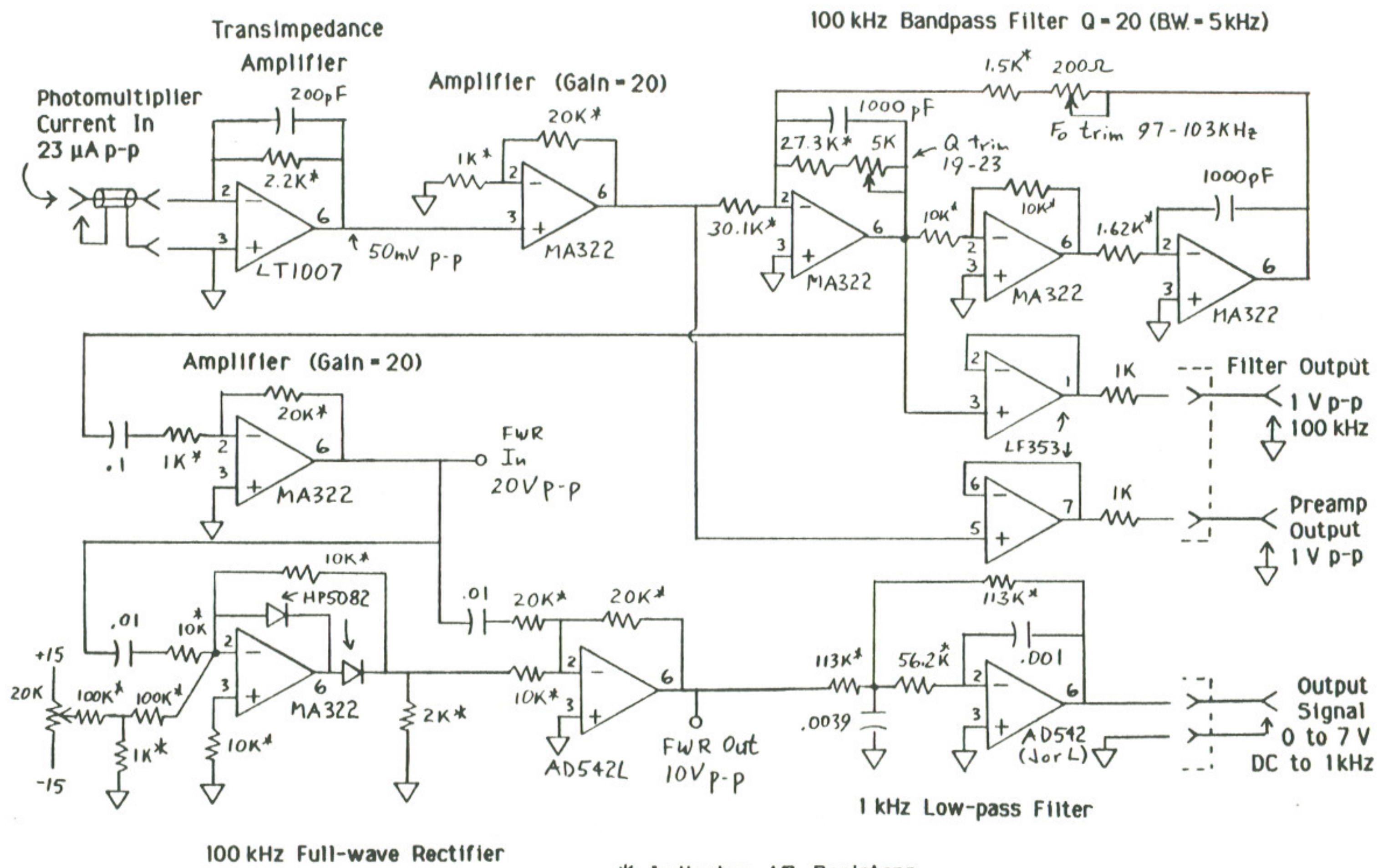
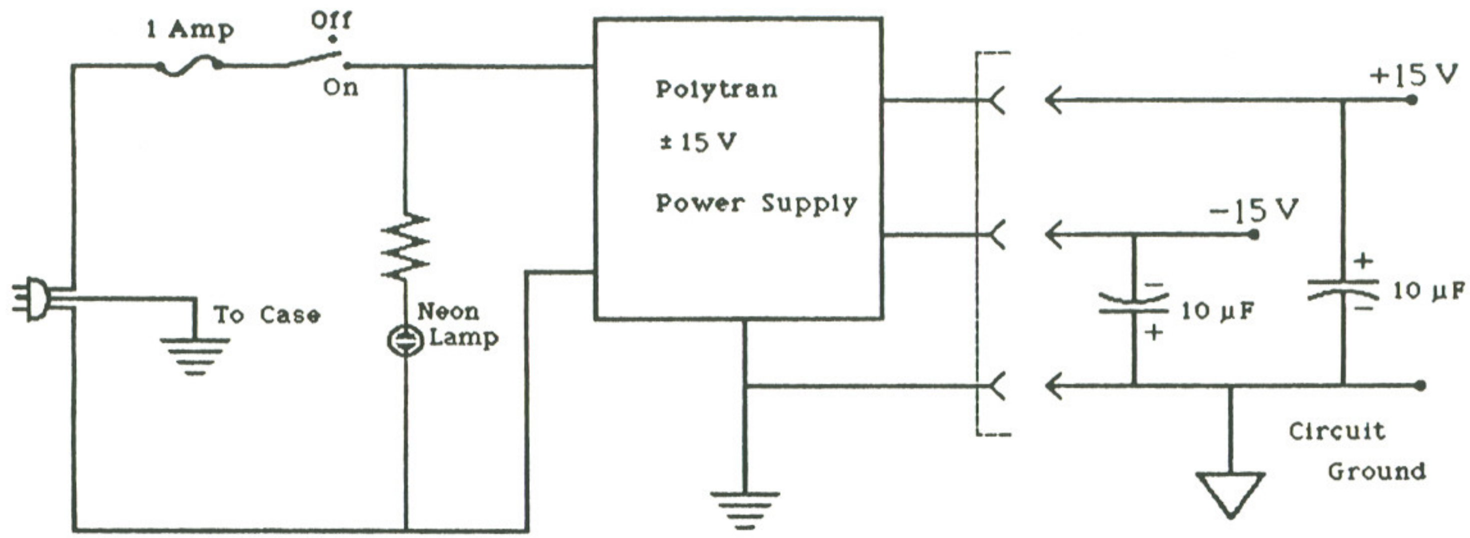
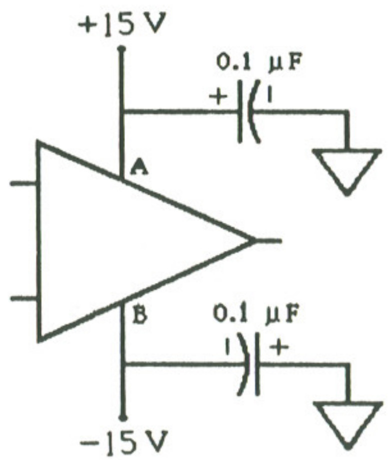


Figure A-3.1 Signal Processing Circuit





Op-Amp Power Connections

Type	Pin Number A	Pin Number B
LF 353	4	8
LT 1007	4	7
MA 322	4	7

Figure A-3.2 Power Circuit For Receiver

APPENDIX B

SOFTWARE LISTINGS

FMAT.TXT

Description of format RECDAT.FTN uses to store data from a/d on TK50 tape.

The program RECDAT.FTN uses an Analog to Digital Converter (ADC) and an external timing circuit to accomplish real time data acquisition. The data is stored on digital magnetic tape using a TK50 tape drive. The computer prompts the user to enter important parameters and to make necessary equipment adjustments. The user specifies the amount of data to be recorded and enters comment text to identify the files.

Blocks written on the TK50 are 2048 bytes (1024 words) long. There are three types of blocks: header, data, and end of data.

The header is written before data is taken. It shows the sample period (number of microseconds between sampes), blocks/data set, number of data sets, date and time, and the comment text. After the parameters have been entered, RECDAT records data under saturation (spatial filter moved out of the brightest spot) and background (laser blocked) conditions. Only the mean and variance of these two groups of data are stored in the header block. to the TK50 and begins to take data. Data is collected in data sets. First the insitu data is collected (8 readings from channel 2, followed by 8 readings from channel 1). Then the requested number of readings are taken from channel 0. Typically a 1000 microsecond sampling period will be used (1000 samples/second). One data set will contain 10 seconds of data since there are 1000 readings in a block and 10 blocks per data set.

The data blocks hold 8 words of parameters (data set number, block number, sample period, time, a/d board error word) followed by 8 words of channel 1 (wind) data, 8 words of channel 2 (turbulence) data and 1000 words of channel 0 data.

While a data set is being taken, the ch 0 sampling rate is continuous; it is specified by the sample period. If a data point is missed, the a/d routine returns an error word (see a/d program for the description). Normally the error word = 0. It is copied to word 8 in the data block. This word should be checked when the data is processed to insure that the block of data is valid. The hardware status word from the a/d board is recorded.

Note that the data is continuous over a data set even though the data set is recorded over several blocks on the tape. The beginning of the data block on the tape indicates the data set and block. If multiple blocks are needed, the same insitu data is recorded on each block of the data set.

The header of each tape file includes the time and date when the header block was written. The time at which data collection begins for each data set is recorded in all data blocks which comprise that data set.

TK50 Magtape Format

The data format is similar to that used in the CA system. It is not structured as normal DEC files. Nor is it structured as normal ANSI magtape files. It is structured to conserve tape and for potentially rapid data transfer. It should be readable on other systems since the block size (2048 bytes) is the maximum size allowed for the ANSI standard.

HEADER Block Contents:

Word number (1 word = 2 bytes)	Description
1	data set number (always 0 for header)
2	block number (always 0 for header)

3	time between samples (in microseconds)
4	number of blocks per data set
5	number of data sets in this "file"
6	high voltage level to PMT (in Volts)
7	laser power transmitted (in microwatts)
8	number of samples per 1000 that were too large when recording saturation data
9	year (00 - 99)
10	month (1-12)
11	day of month (1-31)
12	hours (1 - 24)
13	minutes (0 - 59)
14	seconds (0 - 59)
15	tics (0 - 59) 60 per second
16	Campbell Unit (CA-9) parameters 256 * Range + Time Constant
	The next 4 entries are real values transferred to integer variables via an equivalence statement
17 - 18	mean value of saturation data
19 - 20	variance of saturation data
21 - 22	mean value of background data
23 - 24	variance of background data
25 - 1024	comment text (ascii)

DATA Block Contents:

1	data set number (starts with 1)
2	block number in this data set (starts with 1)
3	a/d hardware status word
4	hours (1 - 24)
5	minutes (0 - 59)
6	seconds (0 - 59)
7	tics (0 - 59) fraction of a second
8	a/d error code (0 means no error)
9 - 12	4 readings from channel 1 before recording data
13 - 16	4 readings from channel 1 after recording data
17 - 20	4 readings from channel 2 before recording data
21 - 24	4 readings from channel 2 after recording data
25 - 1024	data: 1000 readings from channel 0

END OF DATA Block Contents:

1	0 (to match header format)
2	0 (to match header format)
3	arbitrary value (0 is used)
4	-1 (an impossible value for the number of blocks per data set, indicating no more data follows)
5 - 1024	arbitrary values (0 is used)

Analog to Digital Conversion with Andromeda ADC11 board

digital value	input voltage
0 (minimum)	-5.000 V (-FS)
2048	0.00 V
4095 (maximum)	+4.9976 V (+FS - 1 LSB)

Full Scale (FS) is 5.00 V.
1 LSB = (10 V / 4096) = 2.4414 mV

```
        .; file: RECORD.CMD
        .;  This indirect command file prepares the system and runs
        .; the FORTRAN program RECDAT.
        .;
        .;  See chapter 9 of Micro/RSX User's Guide Volume 1 on
        .; The Indirect Command Processor.
        .;
        .; To Use This File Type:   @RECORD
        .;
.ENABLE SUBSTITUTION
.DISABLE DISPLAY
.ENABLE QUIET
CLR             ! Clear The Screen
        .; Set Memory Partition, if it hasn't been done yet.
.TESTPARTITION ATOD
.PARSE <EXSTRI> "," S1 S2 S3 S4 S5
.IF S4 EQ "NSP" set par ATOD/dev 177704 1
.DISABLE QUIET
```

```
.100:   ;
        ; Load your magnetic tape into the TK50 tape drive.
        ; Wait Until The Tape Drive Is Ready, Before You Continue.
        ;
        .ASKS DUMY Both lights should be on, before you press <RETURN>.
        ;
        .ENABLE QUIET
        MOUNT/FOREIGN MU:
                .;
                .; Check To Make Sure Mounting Was Successful.
        .TESTDEVICE MU:
        .PARSE <EXSTRI> ":," JUNK GOOD
.200:   .PARSE GOOD "," JUNK GOOD
        .IF JUNK EQ "MTD" .GOTO 300
        .IF GOOD NE "" .GOTO 200        ! More Of String To Be Checked
        .DISABLE QUIET
        ;    Attempt to access tape was UNSUCCESSFUL!
        ;         Please Prepare To Try Again.
        .GOTO 100                       ! Ask User To Try Again
                .;
                .; When Mounting Is Successful Run Program.
.300:   RUN RECDAT
        DISMOUNT MU:
        CLR             ! Clear The Screen
        .DISABLE QUIET
        .EXIT
```

```
C     RECDAT.VAR contains parameter and common statements for RECDAT.FTN

C       ibeep = ascii bell character
C       ibpds = number of 1000-point data Blocks Per Data Set
C       itime = time between data values in micro seconds
        parameter(ibeep = 7, ibpds = 10, itime = 1000)

        common ibuf(10024)
        common iaderr, istat
```

```
      Program RECDAT

C      RECDAT.FTN by Todd L. Cloninger
C       based upon ADTK.FTN  by  John Hunt
C       to run on a Micro PDP11/73 with Micro RSX
C      Last Modified on August 12, 1987

C       Calls adrun.

C       Incorporate file containing parameter and common declarations
        include 'RECDAT.VAR'

        integer*2 adcw, addat, adsr
        external adcw, addat, adsr

        integer*4 ovflo

        integer*2 itor(2)
        integer*2 ica
        real*4 rtoi
        real*4 bgmen, bgvar, satmen, satvar
        character*1 YorN, Dumy

C      To allow real values to be moved into an integer array.
        equivalence(itor(1),rtoi)

 1001 format (x,2a,$)
        print 1001, 27,'[2J'    ! clear screen
        print 1001, 27,'[0;0H'  ! move cursor to home position
        print *, ' RECDAT.FTN       Last Modified: August 12, 1987'
        print *
        print *, '  This program uses an Analog To Digital ',
     *   'Converter to collect data from the'
        print *, 'laser receiver and store it on a TK50 tape.'
        print *, '  A file consists of a header block followed ',
     *   'by a group of 10-block data sets.'
        print *, 'Each block is 2048 bytes long and contains ',
     *   '1000 data values.'
        print 1002, 'NOTE: Parentheses enclose default responces.'
        print *

C       Clear ADC to prevent system lock-up
        if (adsr(0) .ne. 0) then
           i = adcw(0)
           i = adcw(1)
           i = i+1       ! delay to be sure A to D conversion is finished
25         if (adsr(0) .ne. 0) then
              print *, 'ADC Status Register =',adsr(0)
              i = addat(0)
              print *, 'ADC Status Register =',adsr(0)
              goto 25
            endif
         endif

        print 1002, 'Rewinding Tape!'
        call tkcmd(1280,1)       ! rewind command

 1002 format (/x,a)
 1003 format (a1)
 1004 format (x,a)
 1005 format (x,a,$)

30      print *
        print *, 'Is it necessary to advance the tape before ',
     *   'writing data on it?'
```

```
        print 1005, ' [Y/N]  -->'
        read(5,1003)YorN
        print *
        if ((YorN .eq. 'Y').or.(YorN .eq. 'y')) then
           call advanc(ibu)  ! Advance Tape To The End Of Its Last File
         elseif ((YorN .eq. 'N').or.(YorN .eq. 'n')) then
           print *, 'WARNING: Any Data On The Tape Will Be Written ',
     *       'Over And Lost!'
         else
           goto 30       ! ask again
         endif
        print *

C       -----***** Default Parameters *****-----
        ihvs = 750       ! abs(default High Voltage Setting for PMT)
        idnds = 12       ! default # of data sets in a file


C  ------****** Main Loop For Creating Entire Files ******------
40      print *
        print 1001, 'Enter the transmitted laser power ',
     *   '(in micro Watts) --> '
        read(5,1010,err=40) ilpwr
 1010 format(i4)

50      print *
        print 1001, 'Enter the Campbell Unit CA-9 Range ',
     *   '[5, 10, 20] m/sec (5)--> '
        read(5,1010,err=50) i
        if (i.eq.0) i = 5
        if ((i.ne.5) .and. (i.ne.10) .and. (i.ne.20)) goto 50
        ica = 256 * i

60      print *
        print 1001, 'Enter the CA-9 Time Constant ',
     *   '[1, 10, 100] seconds (1)--> '
        read(5,1010,err=60) i
        if (i.eq.0) i = 1
        if ((i.ne.1) .and. (i.ne.10) .and. (i.ne.100)) goto 60
        ica = ica + i

        call highv(ihvs,ihva)

        print 1002, ' '
        print 1015, 'Sampling Period =',itime,' micro-sec;     ',
     *  'Sampling Frequency =',1000000/itime,' Hertz'
 1015 format (x,a,i5,a,a,i5,a)

        print 1002,'The sampling rate will be:'
        print 1020,1000.0/itime
 1020 format(21x,f5.2,' blocks/second')

        tpds = itime * 1.0e-3 * ibpds   ! time per data set (seconds)

        print 1030, 1/tpds       !data sets per second
 1030 format(2x,'or',16x,f6.2,' data sets/second')

        print 1035, tpds, tpds/60
 1035 format(2x,'or',16x, f6.2, ' seconds/data set  (',f5.3,
     *  ' minutes/data set)')

150     print 1002,'How many data sets do you want recorded in this file?'
        print 1040,'[1 to 360]  (',idnds,') -->'
 1040 format (x,a,i2,a$)
        read(5,1045,err=150)inds
 1045 format(i3)
```

```
        if (inds .eq. 0) inds = idnds
        if ((inds .lt. 0).or.(inds .gt. 360)) then
          print 1005,ibeep
          print *,'Your entry may not be < 1 or > 360; please try again!'
          goto 150
         endif
        idnds = inds

        print *
        ttime = tpds*inds
        print 1050, 'Sampling will require: ', ttime, ttime/60
 1050 format(x,a,f7.2,' seconds or ',f6.2, ' minutes')
        print 1002,' Writing data to tape requires additional time.'
        ttime = ttime + 2.8*inds
        print 1050, 'Elapsed time will be about: ', ttime, ttime/60


C    -----***** Read Comments Into ibuf As Part Of Header *****-----
        print 1002,'Enter text for comment block in magtape header.'
        print *,' (Terminate with a blank line.)'

C       Fill ibuf with text, line by line,  starting with ibuf(25)
        iopntr = 25

310     ipntr = iopntr + 39
        if (ipntr .gt. 1024) then
          ipntr = 1024
          print *,' **** Out of Space Make Last Entry (',
     *      2*(1+ipntr-iopntr), ' characters) ****'
         endif

315     read(5,1055,err=315) (ibuf(i), i=iopntr,ipntr)
 1055 format(40a2,$)
        print 1060, (ibuf(i), i=iopntr,ipntr)   ! echo the entered line
 1060 format(h+,40a2)

C       Determine where text from this line ends
320     if(ibuf(ipntr) .eq. '2020'x) then
          ipntr = ipntr - 1
          goto 320
         endif

        ipntr = ipntr + 1
        if (ipntr .gt. iopntr) then
C         --Add carriage return and line feed to end of line
          ibuf(ipntr) = '0d0a'x
          ipntr = ipntr + 1
          iopntr = ipntr
C         --If there is space, get another line
          if (ipntr .le. 1024) goto 310
         endif

C       Fill the rest of the block with zeros
        do 330 j=ipntr,1024
           ibuf(j)=0
330     continue
C    -----***** Finished Reading Comments *****-----


350     print *
        print *,'Do you need to change the parameters or reenter the text?'
        print 1005,'  [Y/N] -->'
        read(5,1003) YorN
        if ((YorN .eq. 'Y').or.(YorN .eq. 'y')) goto 40
        if ((YorN .ne. 'N').and.(YorN .ne. 'n')) then
          print 1005,ibeep
```

```
          goto 350
         endif

C  *************************************************************
C       Write Header Block For Tape File

520     ibuf(1) = 0              ! data set number (0 for header)
        ibuf(2) = 0              ! block number (0 for header)
        ibuf(3) = itime          ! time between samples (in microseconds)
        ibuf(4) = ibpds          ! blocks per data set
        ibuf(5) = inds           ! number of data sets to be taken
        ibuf(6) = ihva           ! high voltage level to PMT (in Volts)
        ibuf(7) = ilpwr          ! laser power transmitted (micro Watts)

        print 1002, ' Adjust spatial filter for MINIMUM signal.'
        print *, ' THEN press <RETURN> to continue.'
        read(5,1003) Dumy

        print *
        print *, 'Rotate horizontal transverse micrometer one turn ',
     *   '(0.5 mm) clockwise'
        print *, '  to move spot out of beam center.'
        print *, 'Press <RETURN> to record SATURATION data.'
550     read(5,1003) Dumy
        print *, 'Sampling will take about 65 seconds.'

        call dstats(60000,0,satmen,satvar,ovflo)

570     print 1004,ibeep
        print *, 'SATURATION:  Mean =',satmen -2048
        print *, 'Standard Dev. =',sqrt(satvar),'    Variance =',satvar
        print *, ovflo/600.0, '% of the data values were too large!'
        print 1005, 'Are these results acceptable? [Y/N]  -->'
        read(5,1003)YorN
        print *
        if ((YorN .eq. 'N').or.(YorN .eq. 'n')) then
           print *, 'Adjust the High Voltage to the P.M.T.'
           call highv(ihvs,ihva)
           ibuf(6) = ihva        ! high voltage level to PMT (in Volts)
           print *, 'Press <RETURN> to record SATURATION data.'
           goto 550
         elseif ((YorN .ne. 'Y').and.(YorN .ne. 'y')) then
           goto 570      ! ask again
         endif

        ibuf(8) = nint(ovflo/60.0)   ! Sat. Overflow (parts / thousand)

        rtoi = satmen            ! mean value of saturation data
        ibuf(17) = itor(1)
        ibuf(18) = itor(2)
        rtoi = satvar            ! variance of saturation data
        ibuf(19) = itor(1)
        ibuf(20) = itor(2)

        print 1002, ' Adjust spatial filter for MINIMUM signal.'
        print *, ' THEN press <RETURN> to continue.'
        read(5,1003) Dumy

580     print 1002, '***--- BLOCK THE LASER! ---***'
        print *, 'Press <RETURN> to record BACKGROUND data.'
        read(5,1003) Dumy
        print *, 'Sampling will take about 20 seconds.'

        call dstats(20000,0,bgmen,bgvar,ovflo)
590     print 1004,ibeep
        print *, 'BACKGROUND:  Mean =',bgmen -2048,'    Variance =',bgvar
```

```
        print 1005, 'Do you want to record background again? [Y/N]  -->'
        read(5,1003)YorN
        print *
        if ((YorN .eq. 'Y').or.(YorN .eq. 'y')) then
           goto 580
         elseif ((YorN .ne. 'N').and.(YorN .ne. 'n')) then
           goto 590      ! ask again
         endif

C       Move real values into an integer array.
        rtoi = bgmen              ! mean value of background data
        ibuf(21) = itor(1)
        ibuf(22) = itor(2)
        rtoi = bgvar              ! variance of background data
        ibuf(23) = itor(1)
        ibuf(24) = itor(2)


C       Executive call to get current time (ibuf(9) to ibuf(16))
        call GETTIM(ibuf(9))

        ibuf(16) = ica            ! CA-9 parameters

        call tkcmd(256,1)         ! store header block on tape


        ibadds = 0                          ! number of bad data sets
        ids = 0                             ! data set number

        print 1002, ' UNBLOCK LASER!'
        print *, 'Press <RETURN> to begin recording data.'
        read(5,1003) Dumy

        print 1001, 27,'[2J'     ! clear screen
        print 1001, 27,'[0;0H'   ! move cursor to home position
        print 1001, 27,'[5;r'     ! Set Top and Bottom Margins (DECSTBM)

        print 1065, 'Time','Data Sets','Bad Blocks'
 1065 format (16x,a,26x,a,11x,a)
        print *, '  -------------------------------       -------',
     *   '------------      ----------'
        print *, '  minutes    seconds   tics(1/60)      in file',
     *   '   remaining       Total'
        print *

600     ids = ids + 1                       ! next data set

C       Executive call to get current time (ibuf(1) to ibuf(8))
        call GETTIM(ibuf(1))

C       overwrite year, month and day
        ibuf(1) = ids             ! data set number
        ibuf(2) = 1               ! 1st block of data set

        call adcall(4,1,9)        ! get 4 wind readings from channel 1

        call adcall(4,2,17)       ! get 4 turbulence readings from channel 2

        call adcall(1000*ibpds,0,25)           ! record data (channel 0)

        ibuf(3) = istat           ! a/d hardware status word
        ibuf(8) = iaderr          ! stuff error word from a/d calls
                                  ! iaderr = 0 if no errors occurred
                                  ! adrun ierr parameter is stuffed
                                  ! into iaderr if a/d error occurs
```

```
        if (iaderr .ne. 0) ibadds = ibadds + 1  ! count bad data sets

        call adcall(4,1,13)      ! get 4 wind readings (channel 1)

        call adcall(4,2,21)      ! get 4 turbulence readings (channel 2)

        print 1070,ibuf(5),ibuf(6),ibuf(7),ids,inds-ids,ibadds
 1070 format (3(8x,i2),9x,i6,4x,i6,9x,i6)

        ihead = 1                ! index into ibuf for head of next block

        call tkcmd(256,ihead)    ! write first block (2048 bytes) to tape

        do 790 iblock = 2, ibpds
           ibuf(2) = iblock

           ihead = ihead + 1000

C          Copy data block header to beginning of next block of data.
           j = ihead
           do 710 i=1,24
              ibuf(j) = ibuf(i)
              j = j+1
710         continue

C          Write 2048-byte data block to TK50
           call tkcmd(256,ihead)

790      continue
C    -----***** End Block Loop For This Data Set *****-----

C       Take next data set unless this was the last one in this file.
        if (ids .lt. inds) goto 600
C   ------****** End Data Set Loop For This File ******------

        print 1005,ibeep
        print *,'Finished recording data file.'


820     print *, 'Would you like to record another file of data?'
        print 1005, ' [Y/N]  -->'
        read(5,1003)YorN
        print 1001, 27,'[2J'    ! clear screen
        print 1001, 27,'[;r'    ! Reset Top and Bottom Margins (DECSTBM)
        if ((YorN .eq. 'Y').or.(YorN .eq. 'y')) then
           goto 40
         elseif ((YorN .ne. 'N').and.(YorN .ne. 'n')) then
           print 1005,ibeep
           goto 820      ! ask again
         endif
        print *

C       --Put special block on tape to indicate end of last file on tape.
980     continue
        do 990 j=1,1024
           ibuf(j)=0
990     continue
        ibuf(4) = -1             ! bogus # of blocks per data set

        call tkcmd(256,1)        ! store block on tape

        print *, 'Rewinding tape!'
        call tkcmd(1280,1)       ! rewind command

       end
```

```
C ******************************************************************
C
      subroutine advanc(nbu)
C
C     Advance Tape To The End Of Its Last File
C     returns:   nbu = # of Blocks of tape already Used
C *****-------------------------------------------------------*****

C       Incorporate file containing parameter and common declarations
        include 'RECDAT.VAR'

        integer*2 llen, nskip

          nbu = 0        ! # of Blocks of tape Used
          print *, 'Please be patient; this may take several minutes!'
          print *
          print *, 'Advancing tape so that files will be appended.'
          print *
200       call tkcmd(512,1)      ! read 1 block from TK50
          if ((ibuf(1).ne.0) .or. (ibuf(2).ne.0) .or.
     *     ((ibuf(4).ne.-1) .and. (ibuf(4).ne.ibpds))) then
             print *
             print *, 'Error in the file format used on this tape ',
     *         'after block #',nbu - nskip
             print *, '**--- ABORT PROGRAM ---**'
             call tkcmd(1280,1) ! rewind command
             call EXIT          ! system call to terminate program
           endif
          if (ibuf(4).eq.ibpds) then
             if (ibuf(25).ne.0) then
                llen = 25
C               --Determine length of first line of the header comment
C               -- by checking for carriage return and line feed
250             llen = llen + 1
                if (ibuf(llen).ne.'0d0a'x) goto 250

                do 260 i=llen,64
                   ibuf(i) = 0
260              continue
                print 1105, (ibuf(i), i=25,64)
 1105 format(6x,40a2)
              else
                print *, 'skip a file'
              endif
             nskip = ibuf(5) * ibpds     ! calc # of blocks in file
             call tkcmd(1296,nskip)      ! skip to next header block
             nbu = nbu + 1 + nskip
             goto 200
           endif

          print *
          print *, 'Skipped the first',nbu,' blocks on the tape.'

          call tkcmd(1296,-1)    ! wind tape back 1 block

900     return
999     end


C ******************************************************************
C
      subroutine highv(ihvs,ihva)
C
C     Input & Calculation Of High Voltage Used For P.M.T.
C      ihvs = the default high voltage setting
```

```
C       ihva = the actual high voltage (calculated)
C *****-----------------------------------------------------------*****

C       Incorporate file containing parameter and common declarations
        include 'RECDAT.VAR'

        idhvs = ihvs
240     print *
        print *,'The High Voltage to the PhotoMultiplier Tube may be ',
     *     'set from -500V to -1110V.'
 1120 format (x,a,i5,a$)
        print 1120,'Enter the voltage setting. (',-idhvs,' V) -->'
 1125 format(i5)
        read(5,1125,err=240) ihvs
        if (ihvs .eq. 0) then
          ihvs = idhvs
         else
          ihvs = abs(ihvs)
          idhvs = ihvs
         endif
C       For our power supply the measured V differs from the V setting.
        ihva = nint(1.0244 * ihvs -0.75)        ! calc actual H.V.
 1130 format (x,a,i5,a,i5,a)
        print 1130, 'High Voltage:  setting =',-ihvs,
     *  ' V      actual value =',-ihva,' V'

900     return
999     end


C *****************************************************************
C
        subroutine dstats(ipts,ichan,mean,vary,ovflo)
C
C       Use the ADC to measure inum data values from channel # ichan
C       Return mean and variance in the real variables:  mean, vary
C *****-----------------------------------------------------------*****


C       Incorporate file containing parameter and common declarations
        include 'RECDAT.VAR'

        integer*4 ipts, ovflo, icnt, lsum
        integer*2 i, inum, itbuf
        real*4 mean, rbuf, vary
        real*8 ssum

        icnt = ipts
        lsum = 0
        ssum = 0.0
        ovflo = 0

300     if (icnt .gt. 9000) then
           inum = 9000
           icnt = icnt - 9000
         else
           inum = icnt
           icnt = 0
         end if

        call adcall(inum,ichan,1025)            ! take ch0 data

        do 400 i = 1025,1024+inum
           itbuf = ibuf(i) - 2048
C          --Check for gain (High Voltage) set too high
           if (itbuf.ge.2046) ovflo = ovflo + 1
```

```
            lsum = lsum + itbuf
            rbuf = float(itbuf)
            ssum = ssum + rbuf ** 2
400       continue

        if (icnt .gt. 0) goto 300

        mean = float(lsum) / ipts
        vary = (ssum - mean * lsum)/ (ipts -1)  ! sample variance
        mean = mean + 2048

900     return
999     end

C *****************************************************************
C
      subroutine tkcmd(icmd,indx)
C
C      Send command to TK50 tape drive and check for error
C       icmd = the command number
C       indx = a parameter associated with the particular command
C *****-------------------------------------------------------*****

C       Incorporate file containing parameter and common declarations
        include 'RECDAT.VAR'

        dimension isb(2),iprl(2)
        character*1 YorN

        if (icmd .eq. 1296) then                 ! skip blocks
           iprl(1) = indx
         else
           call GETADR(iprl(1),ibuf(indx))      ! Executive call
         endif

        iprl(2) = 2048          ! read 2048 byte blocks

        call wtqio(icmd,1,1,0,isb,iprl,ids)

        if(ids .eq. 1) goto 800 ! good directive status (qio call ok)

        write(6,1220)ids
 1220 format(' Problem in wtqio call for tape:  ids= ',i6)

800     if (isb(1) .eq. 1) goto 900     ! good status from tape

        print *, 'Bad status returned from tape command.'
        print 1225
 1225 format(8x,'decimal  octal  hex')
        print 1230, 1,isb(1),isb(1),isb(1)
        print 1230, 2,isb(2),isb(2),isb(2)
 1230 format(x,'isb(',i1,')  ', i6,2x, o6.6,2x, z4.4)

        print *
850     print 1235, 'Do you want to exit the program? [Y/N]  -->'
 1235 format (x,a,$)
        read(5,1240) YorN
 1240 format (a1)
        print *
        if ((YorN .eq. 'Y').or.(YorN .eq. 'y')) then
           print *, '**--- ABORT PROGRAM ---**'
           call EXIT            ! system call to terminate program
         elseif ((YorN .eq. 'N').or.(YorN .eq. 'n')) then
           print *, 'Atempting to proceed!'
         else
```

```
            goto 850      ! ask again
          endif
        print *

900     return
999     end

C ****************************************************************
C
      subroutine adcall(inum,ichan,index)
C
C   Read inum values from chanel # ichan of Analog to Digital Converter
C   Store the values in the ibuf array, beginning with ibuf(index)
C *****------------------------------------------------------*****

C       Incorporate file containing parameter and common declarations
        include 'RECDAT.VAR'

        inumc = inum

        iclk = 11           ! 1 mHz pacer clock
        ibpr = -itime       ! a/d trigger interval (2s complement, in microseconds)

        istat = 0
        iaderr = 0

        call adrun(ibuf(index),inumc,ichan,iclk,ibpr,0,istat,iaderr)

        if(iaderr .ne. 0) then
           print 1320,iaderr
 1320 format(' Error Returned By adcall ************ error = ',i6)
           print 1321,ibeep
 1321 format(x,a2, 'beep')
          endif

900     return
       end

C ****************************************************

C       format of adrun call:
C       call adrun(iread(1),inum,ichan,iclk,ibpr,iseq,istat,ierr)
C            iread(1) is start of data array for a/d data
C            inum is number of a/d readings to take
C            ichan selects channel (range 0 to 7) to take data from
C            iclk selects pacer clock (11 = 1 mHz)
C            ibpr is the pacer clock buffer/preset register.
C               the a/d is triggered when the bpr rolls over.
C               it is set to the 2s complement of the desired count.
C            iseq selects single channel or sequential channel mode
C               iseq = 0 for single channel
C               iseq > 0 causes the channel to increment after each
C               conversion.  The channel number sequences to the number
C               selected in iseq, then rolls over to channel 0 and
C               repeats the sequence process.
C            istat is a/d board status word after final conversion
C            ierr  is error word returned by adrun
C               ierr = 0 means no error
C
C ****************************************************
```

```
; Type:
;       link @recdat
;  to link object codes of ANDRUN1.MAC and RECDAT.FTN to create RECDAT.TSK
;
;  To generate a map file, replace line below with the following line.
;recdat.tsk/mm/pr:0, recdat.map/-sp/-wi/ma=recdat.obj,andrun1.obj,[1,1]f77fcs/lb
;
recdat.tsk/mm/pr:0  =recdat.obj,andrun1.obj, [1,1]f77fcs/lb
[1,54]rsx11m.stb/ss
/
common=atod:rw
clstr=f77fcs,fcsres:ro
;       par=gen:0:40000
asg=mu:1
asg=ti0:5:6
pri=250
//
```

```
;
;       andrun1.mac
;
;       driver for andromeda adc11 a/d board
;       provides non-interrupt driven adrun subroutine callable from fortran
;       pacer clock commands disabled in this version
;       Also has bonus utility commands for controlling a/d board
;
;       4-17-87 jh
;
        .psect  andrun
        .even
        .list ttm


; ***********************************************
;format of adrun call:
;                       2     4     6     10    12    14    16    20
;       call adrun(iread(1),inum,ichan,iclk,ibpr,iseq,istat,ierr)
;
;       inum is requested conversions
;       istat is a/d status read after final conversion
;       ierr is error word from adrun call:
;               0       no error
;               1       a/d status word has error bit set
;               10      a/d timed out
;               100     illegal number of conversions (< 1) requested

adrun:: mov     2(r5),r1        ;address of fortran data array
        mov     @4(r5),r2       ;conversions to take
        ble     err1            ;exit if 0 or negative conversions requested
        add     @4(r5),r2       ;add twice (2 bytes/conversion)
        add     r1,r2           ;address of end of buffer (+2)

;       mov     @10(r5),iclk    ;pacer clock setting
;       mov     @12(r5),ibpr    ;pacer clock buffer/preset register
;       clr     $kwvcsr         ;turn off kwv11-c pacer clock
        clr     $adcsr          ;clear a/d status register

;       clr     @22(r5)         ;error word to return for testing


1$:     mov     $addbr,r0       ;dummy reading to clear ad done bit
;       inc     @22(r5)         ;stuff number of fifo clears for testing
        mov     $adcsr,r0       ;read done bit
        bne     1$              ;read until fifo buffer empty

        tst     @14(r5)         ;test iseq
        beq     setch           ;branch for single channel operation

        bis     #20002,$adcsr   ;enable sequencer and truncation
        nop
        nop
        mov     @6(r5),$addbr   ;load truncation channel
        br      go              ;start conversion from channel 0


setch:  mov     @6(r5),r0       ;a/d channel to be used
        bic     #77760,r0       ;clear all but lower 4 bits
        ash     #8.,r0          ;shift up for adcsr
        mov     r0,$adcsr       ;send channel to adcsr
        nop

go:     bis     #20,$adcsr      ;enable trigger from external input
```

```
eoc:    clr     r3
eocl:   inc     r3
        beq     err2            ;a/d timed out
        tstb    $adcsr          ;check a/d done bit
        bpl     eocl            ;branch if a/d not done

        mov     $addbr,(r1)+    ;stuff data in output buffer

        cmp     r1,r2           ;final data reading?
        blo     eoc             ;branch for more data

        bic     #20,$adcsr      ;disable external a/d trigger
        br      exit


errl:   mov #100.,@20(r5)       ;set wrong requested count error
        br      exit

err2:   mov #10.,@20(r5)        ;set a/d timeout error

exit:   mov     $adcsr,@16(r5)  ;get a/d status to return
        bpl     exitl           ;a/d error bit not set
        inc     @20(r5)         ;a/d error, so set istat

exitl:  return                  ;return to fortran

;
; ************************************************************
;
;       bonus fortran callable routines
;
;       (not needed for adrun call)
;

adsr::  mov     $adcsr,r0       ;read a/d csr
        rts     pc              ;return to fortran

adcw::  mov     @2(r5),r0       ;get desired value from fortran
        mov     r0,$adcsr       ;stuff it in csr
        rts     pc              ;return to fortran

addat:: mov     $addbr,r0       ;get data from a/d buffer
        rts     pc              ;return to fortran

adtrw:: mov     @2(r5),r0       ;get value from fortran
        mov     r0,$addbr       ;stuff it in truncation register
        rts     pc

adgo::  incb    $adcsr  ;increment adcsr to set the go bit
        mov     $adcsr,r0       ;read a/d csr
        rts     pc              ;return to fortran

;       format of call to fill:
;       call fill(iarray, inum)

fill::  mov     #1,r3           ;status flag
        mov     2(r5),r0        ;address of array to fill
        mov     @4(r5),r1       ;number of elements to fill
        clr     r2
        inc     r3

loop:   mov     r2,(r0)+
        inc     r2
        cmp     r2,r1
```

```
bne     loop
rts     pc              ;return to fortran

.end
```

```
c     file: stat.var
c
c     Contains Parameter & Common Statements for stat.f
c     by Todd Cloninger
c     **********************************************************
c
c     ibufsiz = length in bytes of character buffer that the
c                data on tape is read into
c     nbyt    = number of bytes required for 1 block of data
c     tlu     = tape logic unit [0 - 3]
c
      integer tlu
      parameter(ibufsiz=20480,nbyt=2048,tlu=2)
      character*(ibufsiz) B
      common B
```

```
c***************************************************************
c      Program stat.f
c      Written by Todd Cloninger
c      Last Modified: August 12, 1987
c***************************************************************
c
c       Written for use on a MicroVAX II
c       To compile type:   f77 stat.f -lnag

c       Common Variables & Parameters Defined In Separate File
       include 'stat.var'

       integer tcsr,fn,rn
       logical eoff,errf,eotf
       character op*2,yorn*1

        print *
        print *,'Program stat   by Todd Cloninger  August 12, 1987'
        print *
        print *, '  This program processes data stored on a TK50 ',
      *  'tape with the same format'
        print *, 'as used by RECDAT.FTN on the Micro PDP11/73.'
        print 150, nbyt
  150 format(x,'The data is stored in blocks of',i5,' bytes.')

2000    continue

c       Open TK50 Tape File
        ito=topen (tlu,'/dev/nrmt0',.false.)
        ist=tstate (tlu,fn,rn,errf,eoff,eotf,tcsr)
        print*,'ito=',ito,' tcsr=',tcsr
        print*,'errf=',errf,'eoff=',eoff,'eotf=',eotf

1500    print 1
    1 format(/,
     -10x'***************************************************'/,
     -10x'*                COMMAND LIST                     *'/,
     -10x'*                                                 *'/,
     -10x'*    db:  display a block of data                 *'/,
     -10x'*    rw:  rewind tape                             *'/,
     -10x'*    sk:  skip "nb" blocks                        *'/,
     -10x'*    nf:  skip to beginning of next file          *'/,
     -10x'*    td:  copy "nb" blocks to your data file      *'/,
     -10x'*    tg:  copy "nb" blocks to your graph file     *'/,
     -10x'*    mv:  calculate mean and variance of data     *'/,
     -10x'*    ps:  calculate mean power spectrum           *'/,
     -10x'*    ex:  exit program                            *'/,
     -10x'***************************************************')

1000    continue
        print 32, 'Input op(db,rw,sk,nf,td,tg,mv,ex),nb -> '
   32  format(/a$)
        read(5,33,err=1000) op,nb
   33  format(a2,i5)
```

```
      print *
      if (op.eq.'db') then
         call db
       else if (op.eq.'rw') then
         call rw
       else if (op.eq.'sk') then
         call sk (nb)
       else if (op.eq.'nf') then
         call nf
       else if (op.eq.'td') then
         call td (nb)
       else if (op.eq.'tg') then
         call tg (nb)
       else if (op.eq.'mv') then
         call mv
       else if (op.eq.'ps') then
         call ps (nb)
       else if (op.eq.'ex') then
         goto 800
       else
         goto 1500
       end if

      goto 1000

800   ics=tclose(tlu)

818  format(/'Do you want to exit this program? [y or n] '$)
850   print 818
      read(*,*,err=850) yorn
      if ((yorn.eq.'n').or.(yorn.eq.'N')) goto 2000
      if ((yorn.ne.'y').and.(yorn.ne.'Y')) goto 850
      print *
      print *, 'Program Terminated!'
      print *
     stop
     end

c    ************************************************************

     subroutine sk (nb)

     include 'stat.var'

     integer fn,rn
     integer tcsr
     logical eoff,errf,eotf

      if (nb.eq.0) nb = 1
      do 134 i=1,nb
         ird=tread(tlu,B(1:nbyt))
         ist=tstate (tlu,fn,rn,errf,eoff,eotf,tcsr)
         if (errf) then
            call tperr
```

```
          return
        end if
 134   continue
      print*,'fn =',fn,' Record # =',rn,
     *     ', errf=',errf,', eoff=',eoff,', eotf=',eotf
      return
      end

c     ************************************************************

      subroutine nf

      include 'stat.var'

      integer fn,rn,tcsr,i,j
      logical eoff,errf,eotf
      integer*2 iorc, iorc2
      character cori*1, cori2*2
      equivalence (cori,iorc)
      equivalence (cori2,iorc2)

       iorc2 = 0
 134   ird=tread(tlu,B(1:nbyt))
       ist=tstate (tlu,fn,rn,errf,eoff,eotf,tcsr)
       if (errf) then
          call tperr
          return
        end if
       if ((B(1:2).ne.cori2).or.(B(1:2).ne.cori2)) goto 134

       cori2 = B(7:8)
       if (iorc2.lt.0) then
          print*,'There is no more data on this tape.'
          call rw
        else
c         Print Up To 10 Lines From Header Block
          iorc = 10
          i = 0
          j = 48
 170      j = index(B(j+1:nbyt),cori) + j
          i = i + 1
          if (i.lt.10) goto 170

          print 2003, B(49:j)
 2003 format(x,a$)
        end if
      return
      end

c     ************************************************************

      subroutine rw

      include 'stat.var'
```

```
      integer tcsr,fn,rn
      logical eoff,errf,eotf

       print *, 'Rewinding the tape.'
       irw=trewin (tlu)
       ist=tstate (tlu,fn,rn,errf,eoff,eotf,tcsr)
       print*,'irw=',irw,' errf=',errf,' tcsr=',tcsr
       print*,'fn =',fn,' Record # =',rn
      return
      end

c     **********************************************************

      subroutine db

      include 'stat.var'

      logical eoff,errf,eotf
      integer tcsr,fn,rn,i,j
      integer*2 iorc, iorc2
      character cori*1, cori2*2
      equivalence (cori,iorc)
      equivalence (cori2,iorc2)

       iorc2=0
       do 67 kp=1,ibufsiz,2
          B(kp:kp+1)=cori2
   67   continue

       print*,'This option displays 1 block of data from the tape.'

       ird=tread(2,B(1:nbyt))
       write(*,2002) (B(j:j+1),j=1,16,2)
       write(*,2002) (B(16+j:j+17),j=1,16,2)
       write(*,2002) (B(32+j:j+33),j=1,16,2)
 2002 format(x,8i6)

       if ((B(1:2).eq.cori2).and.(B(3:4).eq.cori2)) then

c         Print Up To 10 Lines From Header Block
          print*
          iorc = 10
          i = 0
          j = 48
   70     j = index(B(j+1:nbyt),cori) + j
          i = i + 1
          if (i.lt.10) goto 70

          write(*,2003) B(49:j)
 2003 format(x,a$)
          print*
        else

c         Print 17 Lines Of Numbers
          do 77 i=48, 17*16+48, 16
```

```
            write(*,2002) (B(i+j:i+j+1),j=1,16,2)
 77       continue
        end if

       ist=tstate (tlu,fn,rn,errf,eoff,eotf,tcsr)
       print*,' Record # = ',rn,', errf=',errf,',eoff=',eoff
      return
      end

c     ***********************************************************

      subroutine td (nb)

      include 'stat.var'

      integer tcsr,fn,rn,inul,ifc,i,j,k
      logical eoff,errf,eotf
      character ofilnam*15
      character cti*2, cnul*1
      equivalence (cti,ifc)
      equivalence (cnul,inul)

      if (nb.eq.0) nb = 1
      inul=0
      do 67 kp=1,ibufsiz
      B(kp:kp)=cnul
   67 continue

      print*,'This option copies data from tape to your directory'
 330  write(*,*)

      print*,'Input filename for writing (< 16 char.)'
      read(*,1201) ofilnam
 1201 format(a15)
      print*,' Copying from tape to: ',ofilnam
      open (9,file=ofilnam,status='unknown')
      rewind 9

      do 34 i=1,nb
      ird=tread(2,B(1:nbyt))

      do 17 j=1,nbyt,16
 2003 format(8(x,i6))
       write(9,2003) (B(j+k:j+k+1),k=0,15,2)
*      write(*,2003) (B(j+k:j+k+1),k=0,15,2)
  17  continue
  34  continue
      ist=tstate (tlu,fn,rn,errf,eoff,eotf,tcsr)
      print*,' block address is ',rn,', errf=',errf,',eoff=',eoff
      close (9)
      return
      end

c     ***********************************************************
```

```
      subroutine tg (nb)

      include 'stat.var'

      integer tcsr,fn,rn,inul,ifc,offset
      real dt,t
      logical eoff,errf,eotf
      character ofilnam*15
      character cti*2, cnul*1
      equivalence (cti,ifc)
      equivalence (cnul,inul)

      if (nb.eq.0) nb = 1
      inul=0
      do 67 kp=1,ibufsiz
      B(kp:kp)=cnul
   67 continue

      print*,'This option copies data from tape to your directory.'
      print*,'It produces a file that has the format for graphing.'
      write(*,*)

      write(*,*) 'Enter offset: '
      read(*,*) offset

      write(*,*)
      print*,'Input filename for writing (< 16 char.)'
      read(*,1201) ofilnam
 1201 format(a15)
      print*,' Copying from tape to: ',ofilnam
      open (9,file=ofilnam,status='unknown')
      rewind 9

      t = 0.0
      do 34 i=1,nb
        ird=tread(2,B(1:nbyt))

c        check block # to find out if beginning a new data set
        cti=B(3:4)
        if ((ifc.eq.1).and.(t.gt.0.0)) then
c           skip a time interval before starting new data set
           t = t + 20*dt*nb
           write(9,*) '"'
         end if

c        get sampling interval (usec)
        cti=B(5:6)
c        convert to millisec
        dt=ifc/1000.0

        do 17 j=49,nbyt,2
           cti = B(j:j+1)
           write(9,2002) t,ifc + offset
*          write(*,2002) t,B(j:j+1)
           t=t+dt
```

```
 17     continue

 34   continue
 2002 format(x,f8.2,x,i4)
      ist=tstate (tlu,fn,rn,errf,eoff,eotf,tcsr)
      print*,' block address is ',rn,', errf=',errf,',eoff=',eoff
      close (9)
      return
      end


c     ************************************************************

      subroutine mv

      include 'stat.var'

      integer i, numf
      integer tcsr,fn,rn
      logical eoff,errf,eotf, auton
      character yorn*1, cnul2*2
      integer*2 iorc2
      character cori2*2
      equivalence (cori2,iorc2)

  205 format(a)
  207 format(a$)
  210 format(a,x,a)
       print 210, 'This option produces files containing the mean',
     *   '& variance along with'
       print 205, 'the output of the Campbell Unit Anemometer.'
       print 210,'Results are generated for each block of data',
     *  '(1000 values).'
  230 format(/'Do you want to process all files to the end of ',
     *   'this tape? [y/n] '$)
       numf = 1000
2020   write(*,230)
       read(*,*,err=2020) yorn
       if ((yorn.eq.'n').or.(yorn.eq.'N')) then
2030      write(*,207) 'How many files to be processed? ->'
          read(*,*,err=2030) numf
          if (numf.eq.0) return
        else if ((yorn.ne.'y').and.(yorn.ne.'Y')) then
          goto 2020
        end if
  240 format(/'Do you want the files to be named ',
     *   'automatically? [y/n] '$)
2035   write(*,240)
       read(*,*,err=2035) yorn
       auton = ((yorn.eq.'y').or.(yorn.eq.'Y'))
       if (.not.auton) then
          if ((yorn.ne.'n').and.(yorn.ne.'N')) goto 2035
        end if
```

```
       i = 0
       iorc2 = 0
       cnul2 = cori2
       ist=tstate (tlu,fn,rn,errf,eoff,eotf,tcsr)
       if (rn.eq.0) goto 2041

c      Get Header Block Into Buffer If It Isn't There
2040   if (B(1:2).eq.cnul2) goto 2060
2041   ird=tread(tlu,B(1:nbyt))
       ist=tstate (tlu,fn,rn,errf,eoff,eotf,tcsr)
       if (errf) then
          call tperr
          return
        end if
       goto 2040

c      Check For End Of Data Block
2060   cori2 = B(7:8)
       if (iorc2.lt.0) then
          print *, '**-- End Of Data On Tape! --**'
          print *
          call rw
          return
        end if

c      Process One File
       call mv1f (auton)
       i = i + 1

       if (i.lt.numf) goto 2040

      return
      end

c     **********************************************************

      subroutine mv1f (auton)

      parameter(nppb = 1000, ibeep = 7)
c      nppb is the Number of Points Per Block

      include 'stat.var'

      real bkgmen, satmen, bkgvar, satvar, difmen, sqdm
      real turb, scwind, wind, men, var, snmen, snvar
      integer i, j, k, m, isum, nbpds, nds, raset
      integer fn,rn
      logical eoff,errf,eotf, auton
      character filnam*20, fullfn*23
      integer*2 iorc, iorc2
      real*4 rorc
      character cori*1, cori2*2, corr*4
      equivalence (cori,iorc)
      equivalence (cori2,iorc2)
```

```
      equivalence (corr,rorc)

  205 format(a)
  206 format(/a)
  207 format(a$)
  208 format(a/)
  210 format(a,x,a)

c      Display Up To 5 Lines From Header Block
       iorc = 10
       i = 0
       j = 48
2040   j = index(B(j+1:nbyt),cori) + j
       i = i + 1
       if (i.lt.5) goto 2040

       print 230, B(49:j)
  230 format(/x,a$)

       j = index(B(49:69),cori)
c      Fill char string with spaces
       do 2045 i = 1,20
          filnam(i:i) = ' '
2045    continue
       if (auton) then
          filnam(1:j) = B(49:47+j)
        else
          print 206,'Please enter output file name (< 21 char.)'
          read(*,240) filnam
  240 format(a20)
        end if
       print*,' File will be named: ',filnam

       cori = B(32:32)
       raset = iorc
       if (raset.gt.0) goto 2060

2050   print 206,'Enter the Range setting of the Campbell unit.'
       print 207, '   [5, 10, 20] --> '
       read(*,250,err=2050) raset
  250 format(i2)
2060   if ((raset.ne.5).and.(raset.ne.10).and.(raset.ne.20))
     *    goto 2050
c       Generate Scaling Factor Used To Calculate Wind Speed
       scwind = raset / (8*409.6)

       cori2 = B(7:8)
       nbpds = iorc2
       cori2 = B(9:10)
       nds = iorc2
       corr = B(33:36)
       satmen = rorc
       corr = B(37:40)
       satvar = rorc
```

```
      corr = B(41:44)
      bkgmen = rorc
      corr = B(45:48)
      bkgvar = rorc
      difmen = satmen - bkgmen
      sqdm = difmen**2

c      m Tells How Many Non-Space Characters Begin String
      m = lnblnk(filnam)
c     -Open file to contain a list of all the calculated values
      fullfn = filnam(1:m)//'.val'
      open (8,file=fullfn,status='unknown')
c     -graph file plotting log(turbulence) vs log(norm. mean)
      fullfn = filnam(1:m)//'.lnm'
      open (9,file=fullfn,status='unknown')
c     -graph file plotting log(turbulence) vs log(norm. variance)
      fullfn = filnam(1:m)//'.nv'
      open (10,file=fullfn,status='unknown')
      rewind 8
      rewind 9
      rewind 10
      write(8,*)'Background Mean =',bkgmen
      write(8,*)'Background Variance =',bkgvar
      write(8,*)'Saturation Mean =',satmen
      write(8,*)'Saturation Variance =',satvar
      write(8,*)'Normalized Saturation Variance =',satvar/sqdm
      write(8,*)'Normalized Background Variance =',bkgvar/sqdm
      write(8,210) '--------- Campbell Unit ---------',
     *    ' ------ Experimental System ------'
      write(8,260) 'Wind (m/sec)','Turbulence (Cn^2)',
     *    'Norm. Mean','Norm. Variance'
  260 format(a14,2x,a17,x,a14,3x,a16)
      print *, 'Data Processing Is In Progress!'

      do 2300 i=1,nds
         ird=tread(tlu,B(1:nbyt))
         ist=tstate(tlu,fn,rn,errf,eoff,eotf,tcsr)
         if (errf) then
            call tperr
            goto 2400
          end if
         isum = 0
         do 2110 k =17,32,2
            cori2 = B(k:k+1)
            isum = isum + iorc2
2110      continue
         wind = scwind * (isum - 8*2048)
         isum = 0
         do 2120 k =33,48,2
            cori2 = B(k:k+1)
            isum = isum + iorc2
2120      continue
         turb = (isum - 8*2048.0)**2 * 2.020e-17 / 64

         snmen = 0.0
```

```
          snvar = 0.0
          do 2200 j=1,nbpds
c            Do Statistics On 1 Block
             call statlb(men,var)
             snmen = snmen + (men - bkgmen)/difmen
             snvar = snvar + (var - bkgvar)/(men - bkgmen)**2
*         print 270, wind,turb,men,var
  270 format(2x,e12.5,5x,e12.6,6x,e12.6,5x,e12.6)

             if (j.lt.nbpds) then
                ird=tread(tlu,B(1:nbyt))
                ist=tstate (tlu,fn,rn,errf,eoff,eotf,tcsr)
                if (errf) then
                   call tperr
                   goto 2400
                 end if
              end if
2200       continue
          write(9,280) log10(turb), log10(snmen/nbpds)
          write(10,280) log10(turb), snvar/nbpds
  280 format(x,f9.5,2x,f9.5)
          write(8,270) wind,turb,snmen/nbpds,snvar/nbpds
*         print 270, wind,turb,snmen/nbpds,snvar/nbpds
2300     continue
2400    continue
        close (8)
        close (9)
        close (10)

        print 207, ibeep

       return
       end

c      ***********************************************************

       subroutine statlb (men, var)
c      Calculate Mean & Variance for 1 Block of Data

       parameter(nppb = 1000)
c       nppb is the Number of Points Per Block

       include 'stat.var'

       real men, var, dif, rsum
       integer i
       integer*4 isum
       integer*2 iorc2
       character cori2*2
       equivalence (cori2,iorc2)

c       Loop To Calculate Mean
        isum = 0
        do 2020 i =49,nbyt,2
           cori2 = B(i:i+1)
```

```
         isum = isum + iorc2
2020    continue
       men = float(isum) / nppb

       rsum = 0.0
       do 2040 i =49,nbyt,2
          cori2 = B(i:i+1)
          dif = iorc2 - men
          rsum = rsum + dif**2
2040    continue
       var = rsum / (nppb - 1)

      return
      end


c     ***********************************************************

      subroutine ps (nb)

      integer Pnps, Pnpsd2
      parameter(Pnps = 1000, Pnpsd2 = Pnps/2, ibeep = 7)
c      Pnps is the Parameter for the Number of Points used to find
c              the Spectrum (power spectral density)

      include 'stat.var'

      real psd(Pnpsd2+1)
      real dfreq
      double precision x(Pnps), s(Pnps)
      integer ix(Pnps)
      integer i, j, k, ix, ifail
      integer tcsr,fn,rn,inul,ifc,offset
      logical eoff,errf,eotf
      character ofilnam*15
      character cti*2, cnul*1
      equivalence (cti,ifc)
      equivalence (cnul,inul)


       if (nb.eq.0) nb = 1
       inul=0
       do 3010 i=1,ibufsiz
          B(i:i)=cnul
3010    continue

       do 3020 i=1,Pnpsd2+1
          psd(i) = 0
3020    continue

       print 210,' This option reads data from tape & ',
     * 'calculates the power spectrum.'
  210 format(a,x,a)
       print 210,'The sample mean is subtracted before using',
     * 'the FFT routine in the NAg library.'
```

```
      print 210,'The work is done in groups of 1000 data',
     *  'values.'
      print 210,'The spectrum is generated for each block',
     *  '(1000 values) of data.'
      print 210,'A file for graphing is produced by',
     *  'averaging these spectrums.'
      print *

      print *
      print*,'Input filename for writing (15 char. or less)'
      read(*,240) ofilnam
  240 format(a15)
      print*,' File will be named: ',ofilnam

c       get sampling interval (usec)
      print*,'Enter the sampling interval (micro sec)'
      read(*,245) ifc
  245 format(i5)
      dfreq = 1 / (ifc * 1.0e-6 * Pnps)
      print*,'Data Processing in Progress!'

      do 3100 i=1,nb
         ird=tread(2,B(1:nbyt))

c        check data set # for beginning of a new file
         cti=B(1:2)
         if (ifc.eq.0) then
            if (i.eq.1) then
               ird=tread(2,B(1:nbyt))
             else
               goto 3200
             endif
          endif

         k = 1
         offset = 0
         do 3030 j=49,nbyt,2
            cti = B(j:j+1)
            ix(k) = ifc
            offset = offset + ifc
            k = k + 1
3030      continue
         offset = -offset / Pnps

         do 3040 j=1,Pnps
            x(j) = dfloat(ix(j) + offset)
3040      continue

         j=Pnps
         ifail =0

         call c06faf(x,j,s,ifail)
c       c06faf is faster than c06eaf, but requires more memory

c       Calculate Power Spectral Density
```

```
          psd(1) = psd(1) + abs(x(1))
          do 3060 j=2, Pnpsd2
             psd(j) = psd(j) + sqrt(x(j)**2 + x(Pnps+2-j)**2)
3060      continue
          psd(Pnpsd2+1) = psd(Pnpsd2+1) + abs(x(Pnpsd2+1))

3100    continue

3200   open (9,file=ofilnam,status='unknown')
       rewind 9
       do 3250 j=0,Pnpsd2
          write(9,270)j*dfreq,psd(j+1)/(i-1)
3250    continue
  270  format(x,f6.1,x,f9.4)
       close (9)

       print 280, ibeep
  280  format(a1$)

       ist=tstate (tlu,fn,rn,errf,eoff,eotf,tcsr)
       print*,' block address is ',rn,', errf=',errf,',eoff=',eoff

      return
      end

c     ************************************************************

      subroutine tperr

      include 'stat.var'

      integer fn,rn
      integer tcsr
      logical eoff,errf,eotf

c     Error Processing Is Limited To Closing File & Reopening It
       itc=tclose(tlu)
       print*,'**-- ERROR IN READING TAPE! --**'
       ist=tstate (tlu,fn,rn,errf,eoff,eotf,tcsr)
       print*,'itc=',itc,' tcsr=',tcsr
       print*,'errf=',errf,' eoff=',eoff,' eotf=',eotf
       ito=topen (tlu,'/dev/nrmt0',.false.)
       ist=tstate (tlu,fn,rn,errf,eoff,eotf,tcsr)
       print*,'ito=',ito,' tcsr=',tcsr
       print*,'fn =',fn,' Record # =',rn,
     *    ', errf=',errf,', eoff=',eoff,', eotf=',eotf
       print*
      return
      end
```

```
c     file: tapdir.var
c
c     Contains Parameter & Common Statements for tapdir.f
c     by Todd Cloninger
c     **********************************************************
c
c     ibufsiz = length in bytes of character buffer that the
c               data on tape is read into
c     nbyt    = number of bytes required for 1 block of data
c     tlu     = tape logic unit [0 - 3]
c     dflu    = disk file logic unit
c
      integer dflu, tlu
      parameter(ibufsiz=2048,nbyt=2048,tlu=2,dflu=9)
      character*(ibufsiz) B
      character*3 month(12)
      common ibn, B, month
```

```
c
      program tapdir
c
c     **********************************************************
c     Written by Todd Cloninger
c     Last Modified: July 31, 1987
c     **********************************************************
c
      include 'tapdir.var'

      integer i,inul,ifn,tcsr,fn,rn
      logical eoff,errf,eotf,full,valid
      character cnul*1,yorn*1
      character devname*15, fname*40
      equivalence (cnul,inul)

    1 format(a)
    2 format(a,a)
    4 format(/a)
    5 format(/a,a)
    6 format(/a/)
    7 format(a$)
    8 format(/a$)
   10 format(a,i2,a)
   15 format(/i2,a,i5,a)

      month(1) = 'Jan'
      month(2) = 'Feb'
      month(3) = 'Mar'
      month(4) = 'Apr'
      month(5) = 'May'
      month(6) = 'Jun'
      month(7) = 'Jul'
      month(8) = 'Aug'
      month(9) = 'Sep'
      month(10) = 'Oct'
      month(11) = 'Nov'
      month(12) = 'Dec'

      inul=0
      do 1010 i=1,ibufsiz
         B(i:i)=cnul
1010   continue

      print 2, '  This program generates a file in your current ',
     * 'directory'
      print 1, 'listing the contents of a TK50 tape.'
1000  print 4, 'Enter the name for the file.'
      print 7, ' (default = ''tape.dir'') -->'
      read(*,1) fname

      if (lnblnk(fname).eq.0) fname = 'tape.dir'
      print 5, 'The file will be named:  ',fname

1020  print 4, 'You may choose either a brief or full listing.'
```

```
      print 7, ' Would you prefer a brief listing? [y / n] —>'
      read(*,1) yorn
      if ((yorn.eq.'y').or.(yorn.eq.'Y')) then
         full = .false.
       else if ((yorn.eq.'n').or.(yorn.eq.'N')) then
         full = .true.
       else
         go to 1020
       endif

      print 5, 'Make sure the tape is properly loaded ',
     *  'and press the <RETURN> key.'
      read(*,*)
C      device name assigned to tape drive
      devname='/dev/nrmt0'
      ito=topen (tlu,devname,.false.)
      if (ito.lt.0) then
         print 6, 'ERROR IN OPENING TAPE!'
         ist=tstate (tlu,fn,rn,errf,eoff,eotf,tcsr)
         print*,'ito=',ito
         print*,'tcsr=',tcsr
         print*,'errf=',errf,'eoff=',eoff,'eotf=',eotf
       endif

C      Rewind Tape
      irw=trewin (tlu)
      if (irw.lt.0) print 6, 'ERROR IN REWINDING TAPE!'
      ibn = 1
      ifn = 0

      print 6, 'This process will take several minutes.'

      open (dflu,file=fname,status='unknown')
      rewind dflu

1100  call onefil(valid,full)
      if (valid) then
         ifn = ifn + 1
         print 10, 'File ',ifn,' done.'
         go to 1100
       endif

      write(dflu,15) ifn,' files on this tape using ',
     *  ibn,' blocks.'

      close (dflu)

C      Rewind Tape
      print 6, 'Rewinding The Tape.'
      irw=trewin (tlu)
      if (irw.lt.0) print 6, 'ERROR IN REWINDING TAPE!'

      ics=tclose(tlu)

1150  print 8, 'Do you want to process another tape? (y or n) '
```

```
      read(*,*,err=1150) yorn
      if ((yorn.eq.'y').or.(yorn.eq.'Y')) go to 1000
      if ((yorn.ne.'n').and.(yorn.ne.'N')) go to 1150

      print 6, 'Program Completed & Terminated! '

      end

c     **********************************************************

      subroutine onefil (valid,full)

      include 'tapdir.var'

      logical full, valid
      integer dt, lpwr, nblks, nds, nbpds
      integer*2 iorc, iorc2
      character cori*1, cori2*2, cnul*1, corr*4
      real*4 rorc, avg, var
      equivalence (cori,iorc)
      equivalence (cori2,iorc2)
      equivalence (corr,rorc)

  207 format(x,a$)
       iorc = 0
       cnul = cori
       valid = .true.

       ird=tread(tlu,B(1:nbyt))

       cori2 = B(1:2)
       ids = iorc2
       cori2 = B(3:4)
       iblk = iorc2
       if (ird.lt.0) then
          print *, 'ERROR IN READING BLOCK FROM TAPE!'
          valid = .false.
        else if ((ids.ne.0).or.(iblk.ne.0)) then
          print *, 'ERROR IN HEADER BLOCK FORMAT!'
          valid = .false.
        else
          cori2 = B(7:8)
          nbpds = iorc2
          if (nbpds.eq.-1) then
             print *, 'End Of Data On Tape!'
             valid = .false.
           else
             cori2 = B(9:10)
             nds = iorc2
             nblks = nbpds * nds
             if (full) then
                write(dflu,207) B(49:nbyt)
                cori2 = B(19:20)
  210 format(x,'19',i2,'-',a3,'-',i2.2,5x,i2,':',i2.2)
```

```
             write(dflu,210) B(17:18),month(iorc2),B(21:22),
     *         B(23:24),B(25:26)
             cori2 = B(13:14)
             lpwr = iorc2
215 format(x,a,f5.3,a)
             write(dflu,215) 'Transmitted laser power = ',
     *         lpwr*1e-3,' mW'
220 format(x,a,i4,a)
             write(dflu,220) '-',B(11:12),
     *         ' Volts supplied to Photomultiplier Tube'
             cori2 = B(5:6)
             dt = iorc2
225 format(x,i4,a,7x,a,i4,x,a)
             write(dflu,225) dt,' microsec between samples',
     *         '(sampling frequency = ',nint(1.0e6/dt),'Hz)'

             cori2 = B(15:16)
230 format(x,a,x,f4.1,a)
             write(dflu,230)'When recording saturation data,',
     *         iorc2*1.0e-1,' % of values were too large'
             corr = B(33:36)
             avg = rorc
             corr = B(37:40)
             var = rorc
235 format(x,a,f6.1,7x,a,e9.4)
             write(dflu,235) 'Saturation:   mean = ',avg,
     *         'variance = ',var
             corr = B(41:44)
             avg = rorc
             corr = B(45:48)
             var = rorc
             write(dflu,235) 'Background:   mean = ',avg,
     *         'variance = ',var

240 format(x,a,i2,a,i2,a)
             write(dflu,240) 'There are ',nds,
     *         ' data sets with ',nbpds,' blocks / data set'
245 format(x,i3,x,a,a,i5/)
             write(dflu,245) nblks+1,'blocks long ',
     *         'beginning at block ',ibn
           else
             iorc = 10
             write(dflu,207) B(49:index(B(49:nbyt),cori)+48)
             cori2 = B(19:20)
             write(dflu,210) B(17:18),month(iorc2),B(21:22),
     *         B(23:24),B(25:26)
             write(dflu,245) nblks+1,'blocks long ',
     *         'beginning at block ',ibn
           endif
          ibn=ibn+1
          call skipb(nblks)
        endif
      endif

2999  return
```

```
      end

c     ************************************************************

      subroutine skipb (nblks)

      include 'tapdir.var'

      integer nblks
      integer i,ird

       do 2100 i=1,nblks
          ird=tread(tlu,B(1:nbyt))
          if (ird.lt.0) print *,
     *            'ERROR IN READING DATA BLOCK FROM TAPE!'
          ibn = ibn + 1
2100    continue

2999  return
      end
```

VII. BIOGRAPHICAL NOTE

Todd Lewis Cloninger was born on August 8, 1962 in Gastonia, North Carolina, a few miles from his home in Dallas. He became a member of Philadelphia Lutheran Church of Dallas. Graduating second in his class from North Gaston High School in 1980, he received the President's Honor Scholarship at Warren Wilson College (Swannanoa, N.C.). His study included considerable work involving computerized data acquisition and control. In 1984 he completed his Bachelor of Arts degree with a major in Applied Math/Physics and a major in Chemistry.

At Oregon Graduate Center (Beaverton, Oregon) he worked toward a Master of Science degree in Applied Physics with Optics as the area of concentration. He assisted in researching the possibility of using a laser to weld wax patterns used in investment casting. A patent was issued as a result of this research. The work is described in the October 1987 issue of *Modern Casting*.

Having completed the required course work and research for his thesis, Todd L. Cloninger was granted a leave of absence by Oregon Graduate Center in September of 1987. He returned to his home (Dallas, N.C.) and completed his thesis. As of March 1989, he was working as a computer technician.