NASA-CR-191316

IN-35-CR
135402
P-102

NASA Contractor Report

# Development of a Fiber Optic High Temperature Strain Sensor

E. O. Rausch, K. E. Murphy and S. P. Brookshire

GEORGIA TECH RESEARCH INSTITUTE
GEORGIA INSTITUTE OF TECHNOLOGY
ATLANTA, GA 30332-0800

GRANT NAG8-869
NOVEMBER 1992

National Aeronautics and
Space Administration

**Marshall Space Flight Center**
Huntsville, AL 35812

(NASA-CR-191316) DEVELOPMENT OF A FIBER OPTIC HIGH TEMPERATURE STRAIN SENSOR Progress Report, 1 Apr. 1991 - 31 Aug. 1992 (Georgia Inst. of Tech.) 102 p

N93-14810

Unclas

G3/35 0135402

## Preface

This report describes work performed during the period 1 April 1991 through 31 August 1992 by the Georgia Tech Research Institute (GTRI) under NASA Marshall Grant NAG8-869. Ms. Sheila Nash-Stevenson was the technical monitor for NASA Marshall. Dr. E. O. Rausch was the principal investigator at the Georgia Tech Research Institute.

## Contents

## List of Figures

# 1. Introduction

## 1.1 Objectives

From 1 April 1991 to 31 August 1992, the Georgia Tech Research Institute, working under NASA Marshall Grant NAG8-869, conducted a research program to develop a high temperature fiber optic strain sensor as part of a measurement program for the space shuttle booster rocket motor. The major objectives of this program were divided into four tasks. Under Task 1, the literature on high-temperature fiber optic strain sensors was reviewed. Task 2 addressed the design and fabrication of the strain sensor. Tests and calibration were conducted under Task 3, and Task 4 was to generate recommendations for a follow-on study of a distributed strain sensor. Task 4 was submitted to NASA as a separate proposal.

## 1.2 Background

Development of sensors for the measurement of strain at temperatures exceeding 1,200°F (649°C) has been in progress for at least 25 years. Between 1961 and 1975, research and development (R&D) in this field was high because of the combined needs of the Department of Defense (DoD) and NASA. Between the mid-1970s and the mid-1980s, however, this work virtually stopped. Now there is a sudden upsurge in sensor research to measure strains at temperatures approaching 1,800°F (982°C) and higher. Hot structures in rocket engines will be exposed to such temperatures, and their strain response during ground testing as well as in flight must be determined. Finite element methods for stress prediction also need experimental validation.

There are no commercially available sensors capable of valid strain measurements above 1,500°F (815°C). Research and development is in progress at numerous institutions to develop sensors for static or quasi-static strain measurements at temperatures approaching 2,000°F (1,093°C). These include electric resistance strain gauges (both wire and thin film), capacitance gauges, and optical strain gauge systems.

For the past five years, researchers at GTRI have been developing strain sensors involving the use of optical fibers.[1] Some characteristics advantages of fiber optic sensors are:

1. Non-electrical,
2. Explosion-proof,
3. Remotable,
4. Small size and weight,
5. Access into normally inaccessible areas,
6. Potentially easy to install,
7. Immune to electromagnetic interference (EMI),
8. Solid-state reliability,
9. High temperature, and
10. Easily interfaced with data communication systems.

Fiber optic sensors can be divided into four basic categories: phase-modulated, intensity-modulated, polarization-modulated, and special sensors. A literature survey of fiber optic strain sensors was completed during the early phases of this project. A condensed version of this review is attached in Appendix A. Conclusions drawn from this task are as follows:

1. Phase-modulated sensors compare the phase of light in a sensing fiber to that of a reference fiber. Phase differences can be measured with high sensitivity. Thus, phase-modulated sensors are much more accurate than intensity-modulated sensors and can be used over a much larger dynamic range. Phase-modulated sensors, however, are usually sensitive to vibration.

2. Intensity-modulated (or amplitude) sensors generally are associated with displacement or some other physical perturbation that interacts with the fiber. The perturbation which is a function of the phenomenon being measured, causes a change in received light intensity.

3. Polarization-modulated sensors detect changes in optical polarization due to strain within the sensing fiber. Polarization-modulated sensors are generally limited in dynamic range, because it is difficult to isolate the sensor polarization from the input polarization by more than 20 dB.

4. Finally, there exists a class of special sensors that utilize Bragg gratings. Bragg gratings are imprinted within the fiber core as a periodic variation in the index of refraction using an excimer laser. Under normal conditions, the Bragg or diffraction grating returns a particular frequency from the broad light emitting

diode (LED) spectrum to the detector. A variation in the spacing between the index peaks due to tensile or compressive strain will change the frequency of the returned optical energy. Thus, the returned optical frequency is directly proportional to strain or temperature. Unfortunately, at temperatures above 900°C, the Bragg grating disappears, because the structural defects created by the excimer laser are annealed and removed from the lattice.[2]

The intensity modulation technique was selected for this project for the following reasons:

1. It offered a relatively large dynamic range (> 20 dB) on the basis of preliminary measurements.

2. It appeared to be vibration insensitive. This is an important issue, because the proposed application of this sensor is to measure the strain in a booster rocket nozzle during a ground or in-flight test.

3. The intensity modulated sensor concept can be implemented with a simple gap between the sensing fiber and a mirror. This gap sensor is easily fabricated for low temperature applications. But even at high temperatures, the sensor components, such as high temperature optical fibers and platinum alignment tubes, are available on the commercial market.

Optical fibers commonly used for communication purposes are made of $SiO_2$ (i.e., silica). Although silica melts around 1,600°C, almost all communications fibers produced commercially are coated with a polymer material to exclude water vapor in the atmosphere and preserve the integrity of the fiber over many years. At high temperatures, the polymer will burn off. Water molecules can then settle in the microcracks, present in any fiber, and accelerate crack propagation by binding to the oxygen in the silica. For this reason, fibers used in high temperature operation, for example to monitor the curing cycle of graphite epoxy composite, are hermetically sealed by jacketing the fibers with gold instead of a polymer. Even then, the maximum recommended continuous operating temperature is only 750°C, although the fiber can withstand higher temperatures (~ 900°C) for a short period of time (~ minutes). When the continuous operating temperature exceeds 900°C, the fiber must be coated with other types of metals, e.g., platinum. These fibers are currently in development and are not yet available on the commercial market.

Although silica melts at 1,600°C, the material becomes plastic above 1,000°C; therefore, it may not be suitable at temperatures in excess of 1,000°C. A material that can withstand temperatures up to 1,500°C is sapphire. Sapphire fibers are available with a minimum diameter of 250 micrometers. It has a higher index of refraction (1.76) than that of air (1.0) and is capable of guiding light by internal reflection but with an attenuation of about 1 to 2 decibels per foot. Lower attenuation coefficients are not necessary at the present time, because the sapphire fiber would be short (~ 1 foot length) and would be used only in the immediate vicinity of the high temperature region. Outside the critical temperature region, the fiber must be transitioned to silica.

The minimum diameter of the sapphire fiber that was obtained from Saphikon, Inc., was 250 microns. This fiber has no guiding core, and can not be easily transitioned to a silica fiber that has a 50 micron core diameter and a 125 micron outside diameter. It is possible to purchase a 125 to 250 micron silica transition and to connect the silica to the sapphire fiber. In that case, however, the light returned from the mirror would be dispersed throughout the 250 micron sapphire fiber, and only a small fraction would be coupled back into the 50 micron core of the silica fiber, resulting in excessive losses and extremely low signal-to-noise ratios. For this reason, the gap sensor developed for this project utilized only silica fibers. This sensor is presented in the following sections. The test system is discussed first in Section 2, and the sensor system is discussed in Section 3. The conclusions for this project are given in Section 4, and the user's guide of the gap sensor is presented in Section 5.

# 2 Test System

## 2.1 Introduction

The test system was designed and developed to meet three objectives.

1. It must be capable of collecting dynamic strain, static strain, and vibration data in a relatively short period of time (~ 1 hour).
2. The interface between the operator and the test system should be computerized and placed within the framework of a menu environment. This feature ensures a user friendly operating system and fast data collection.
3. The test system must operate with the sensor head located in a high temperature environment.

The test system that satisfies all objectives outlined above is illustrated in Figure 1. It consists of a furnace, a furnace controller, a computer with the necessary signal input/output interface, and a piezoelectric transducer that produces the strain. This system is described in greater detail in this section in terms of the hardware and associated software.

## 2.2 Hardware

High temperature tests were conducted in a cylindrical furnace Model 3320 purchased from Applied Test Systems, Inc. The heated and overall lengths of the furnace were 4 and 14 inches, respectively. Maximum operating temperature was 2,800°F (1,538°C). The sample to be tested was fastened to the ends of two ceramic fixtures. In turn, each fixture was mounted to a ceramic rod made of alumina ($Al_2O_3$). The upper rod was mounted to a rigid immovable structure. The lower rod was attached to a piezo transducer that was purchased from Polyoptronics, the U.S. representative of the German transducer manufacturer, also known as Physik Instrument. Desirable properties of this transducer (Model P-841.60) included a built-in strain gauge with direct readout in microns, expansion up to 90 microns, and a maximum pushing force of 800 newtons (~ 180 pounds). All signals to and from the piezo transducer were routed through the piezo-controller unit. This unit had three modules: a display indicating transducer displacement in microns, a driver that controls the transducer, and a module that interfaces the strain gage sensor output with the data acquisition system. The piezo driver required an input of 0 to 10 volts and provided a proportional voltage output that adjusted the transducer over a range of 0 to 90 microns.

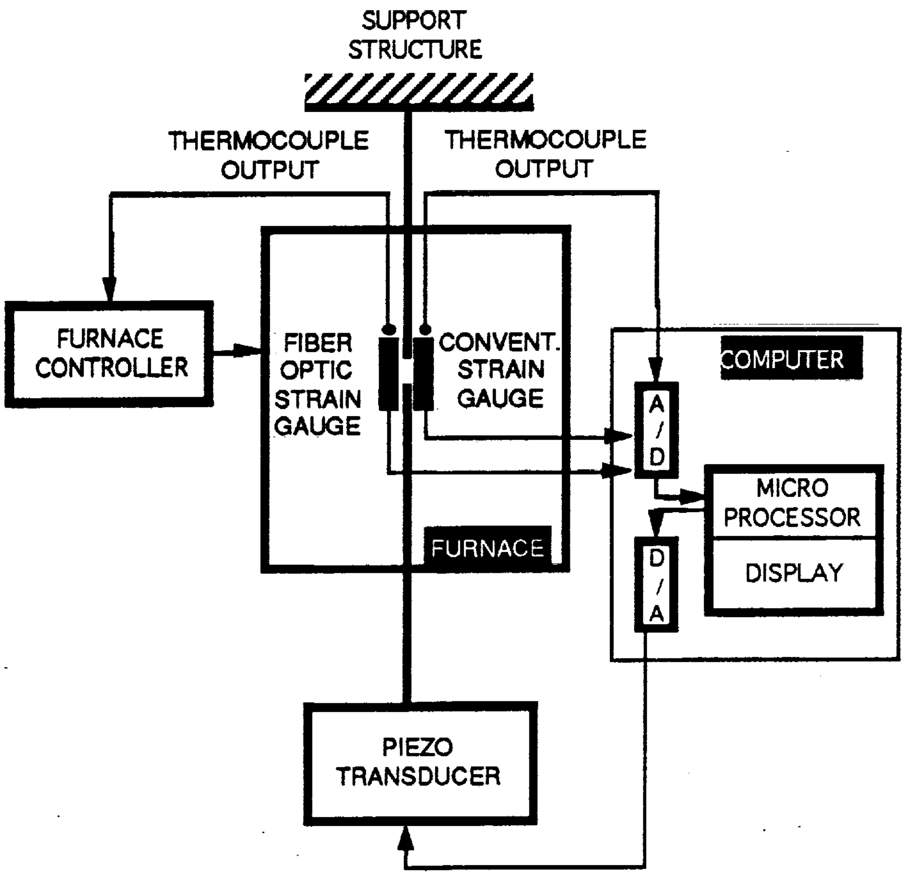

Figure 1. Block Diagram of High Temperature Fiber Optic Sensor Test Set.

The transducer was operated in the pushing mode as shown in Figure 2.

The displacement produced by the transducer was transferred to the sensor head by means of an alumina rod. The sensor head was fastened to machinable ceramic clamps which, in turn, were mounted to the alumina as shown in Figure 3. A photograph of the high temperature test fixture is shown in Figure 4. A Platinum-10% Rhodium-Platinum thermocouple monitored the temperature at the center of the furnace near the sensor head. Both the thermocouple output and the sensor were routed to detection systems that conditioned the electrical output voltage within the 0 to 10 V range necessary for analog-to-digital conversion.

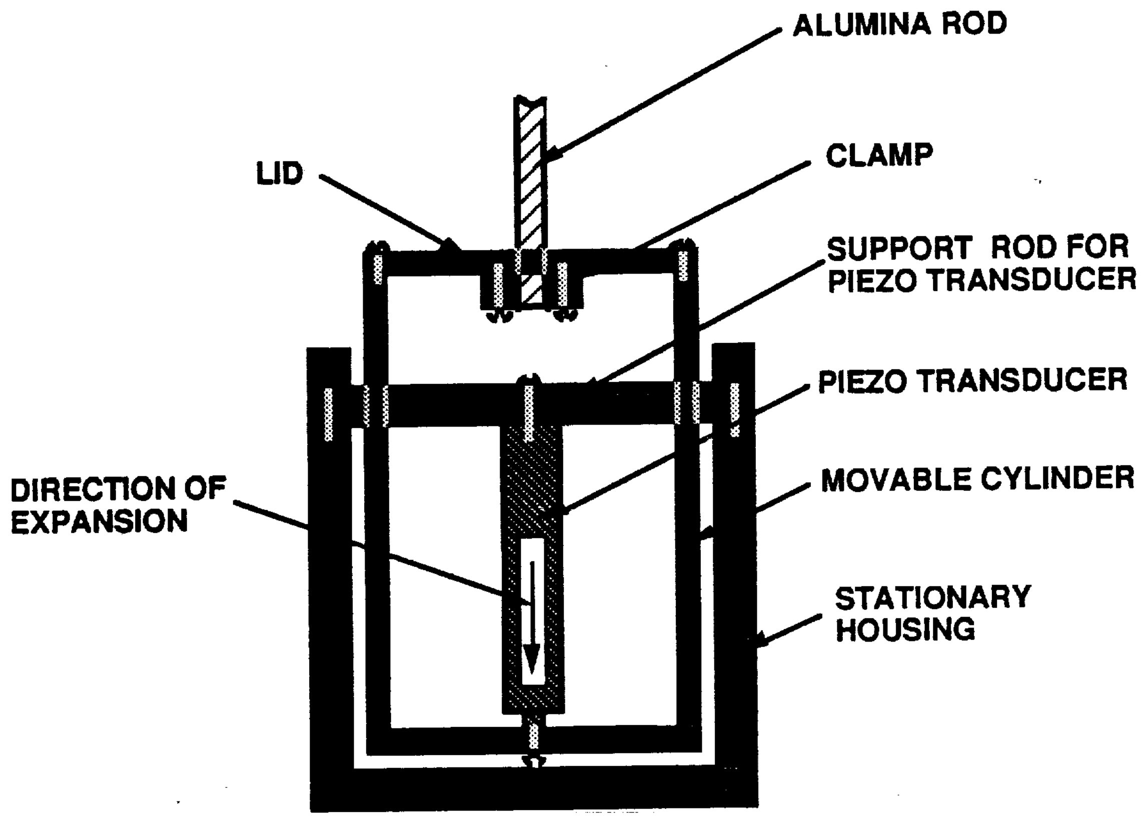

Figure 2. Cross-Sectional View of Piezo Transducer Structure.

A second thermocouple, identical in nature to the first unit, served as a feedback to the furnace controller to regulate the specified temperature. This controller thermocouple was also located at the midpoint of the furnace but opposite the monitor thermocouple. The furnace controller was an LFE model 2012 that allowed the furnace temperature to be varied according to a specified profile, with various set points and delays. The software in the data acquisition system followed these set points to allow unsupervised testing at various temperatures. Each set point required a temperature, $T_i$, and a time duration, $\Delta t_j$; the furnace temperature was automatically increased to $T_i$ by the controller and maintained at $T_i$ for a time period $\Delta t_j$. A typical operating profile is illustrated in Figure 5.

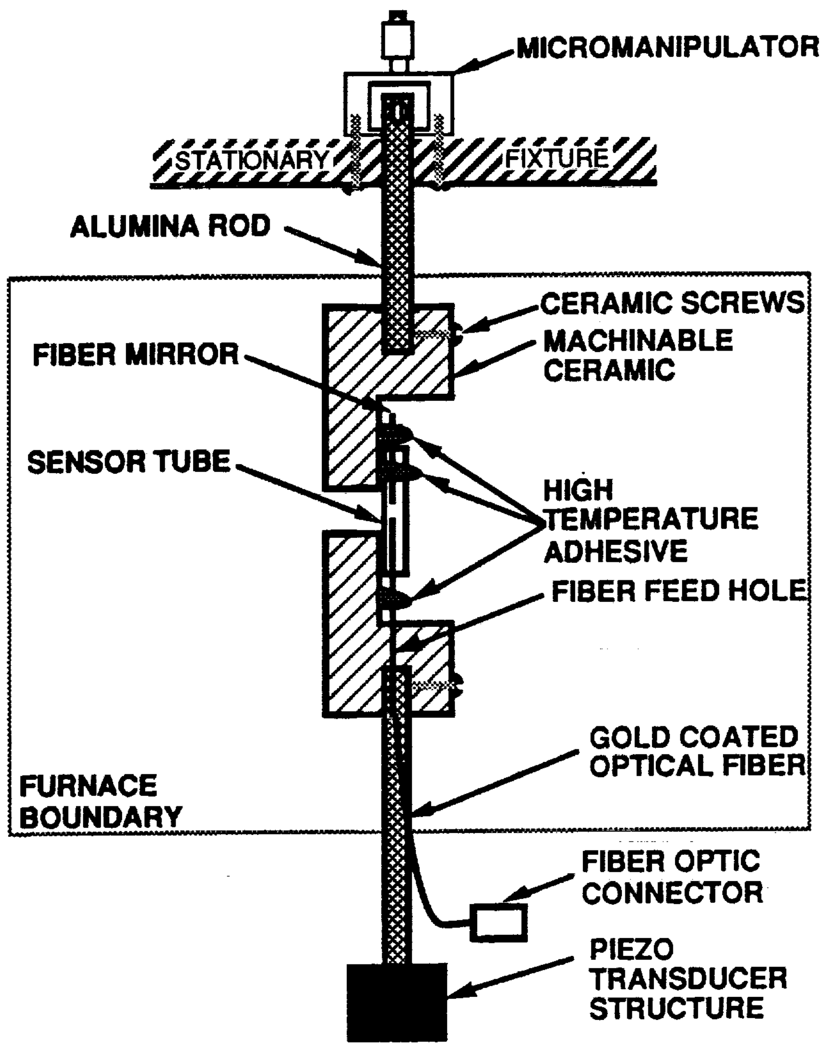

Figure 3. Cross-Sectional View of Mounted Sensor Head in Test Fixture.

Both sensor and thermocouple outputs were converted to digital format and processed by a 386 SX personal computer. The 386 SX had the following features:

| | |
|---|---|
| Hard Disk Size: | 40 Megabytes |
| RAM Memory: | 4 1-Megabyte SIMMS |
| Clock Speed: | 16 MHz |
| Floppy Disk Drive: | One 3.5-inch High Density Drive |

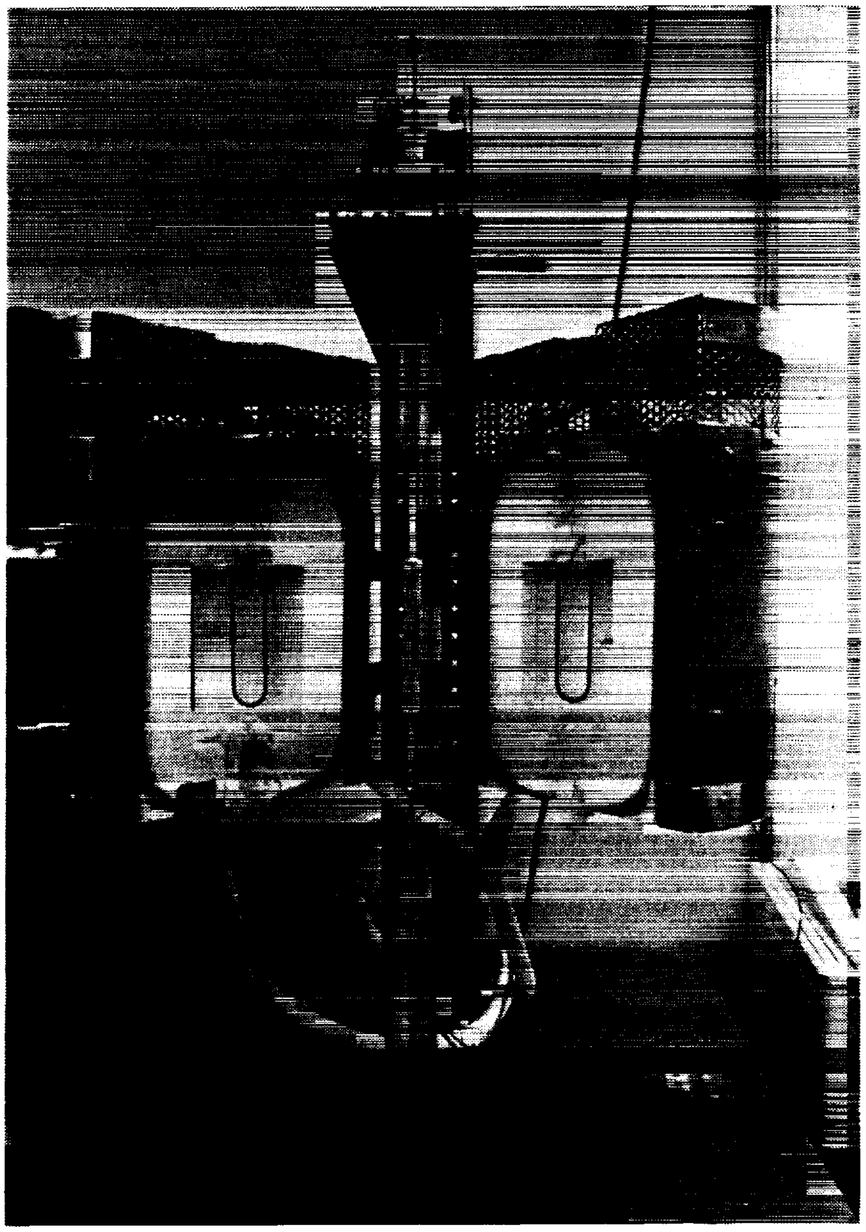

Figure 4. Photograph of High Temperature Test Fixture.

ORIGINAL PAGE
BLACK AND WHITE PHOTOGRAPH

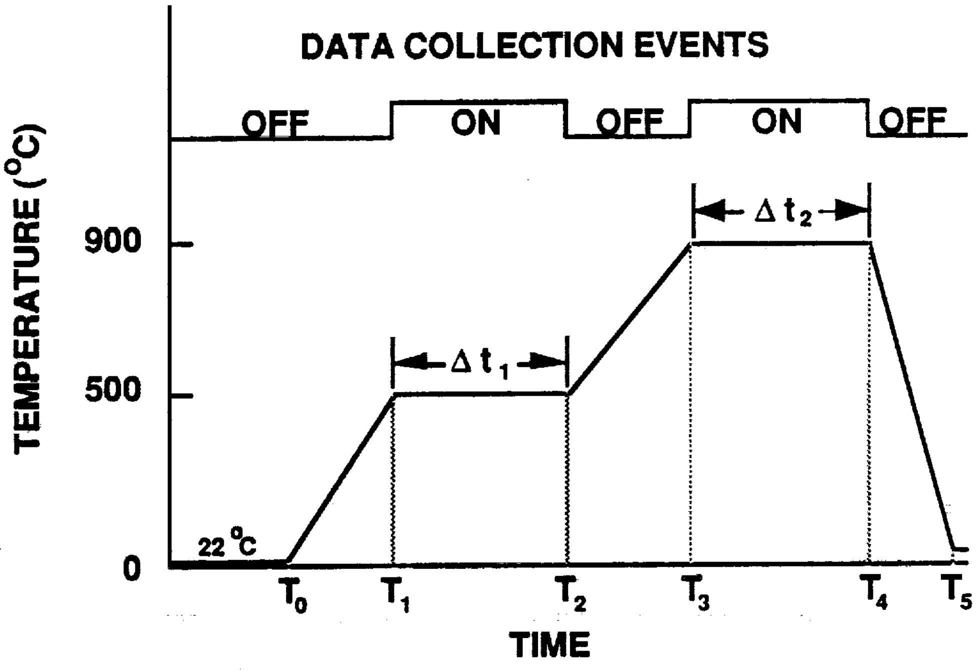

Figure 5. Typical Operating Profile for the Furnace Controller.

In addition, the 386 SX contained an input/output (I/O) board, Model CIO-AD16, manufactured by Computer Boards, Inc. All signals were routed through this I/O board. Figure 6 is a block diagram illustrating the interconnection of the various components.

The complete data acquisition system consisted of three basic elements: the OMEGA Signal Conditioning Rack, the Intel 386-SX based microcomputer with an I/O card, and an extension board. The OMEGA, Inc., rack was used for conditioning the thermocouple and standard strain sensor inputs, and also to tie the various inputs and outputs together in one system. There were two input modules on the rack: one for the thermocouple, and one for the standard strain sensor. They converted the input voltages to a standard 0 to 10 volts used by the I/O card. Also mounted on the rack was a small connector board. This board connected the fiber optic strain sensor, the transducer input and output, and the references from the power supply to the I/O card.

Two 26-pin connectors (one from the connector board and one from the rack) interfaced to a single 37-pin connector on the I/O card. Figure 7 shows the wiring.

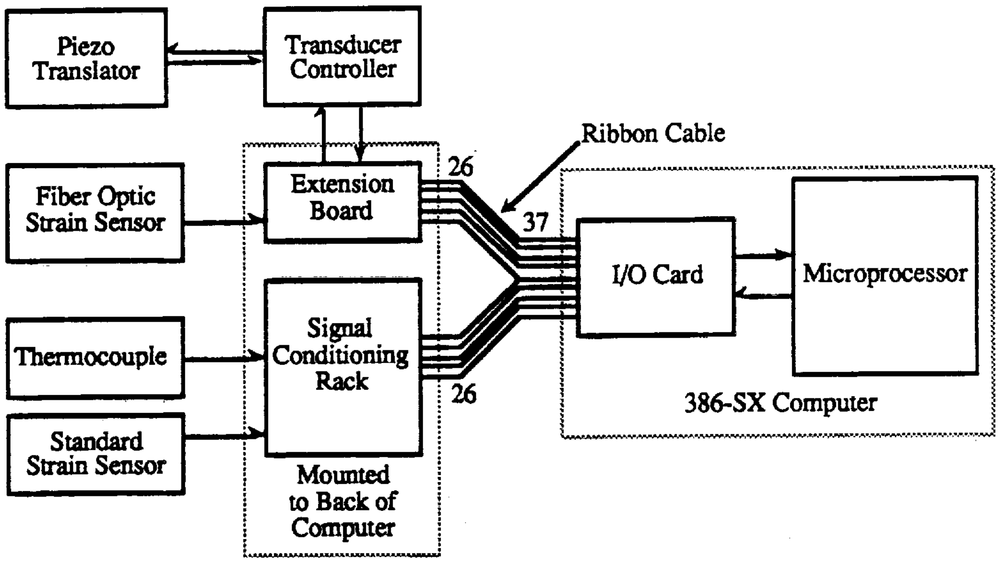

Figure 6. Block Diagram of Component Interconnections.

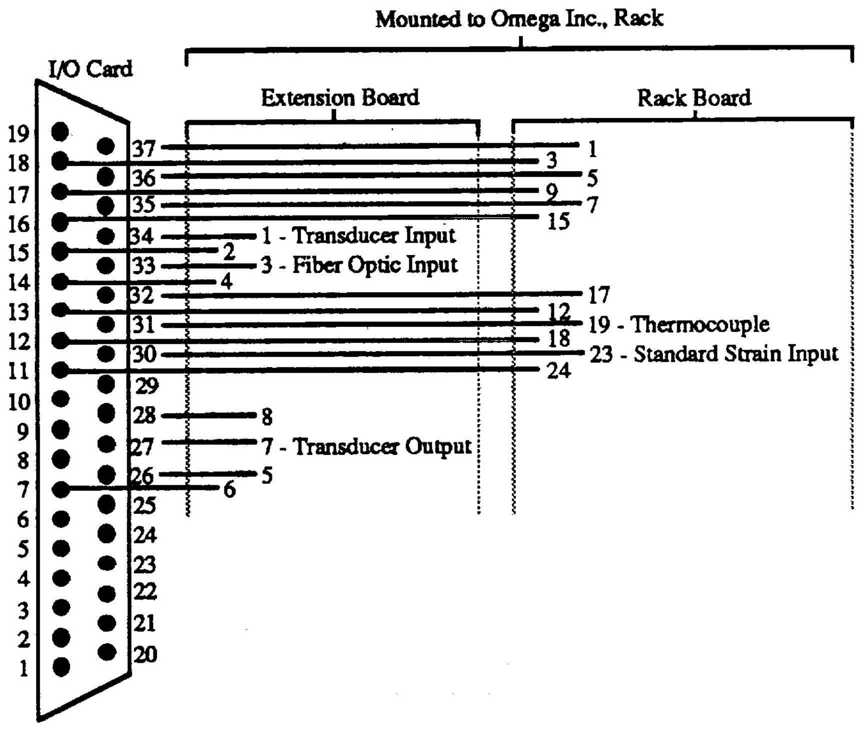

Figure 7. Pin to Pin Interface Diagram.

At the heart of the data acquisition system is the 386-SX microcomputer. The CI0-AD16 I/O card was plugged into an expansion slot on this system. It provided the interface with the rest of the experimental system. Control software, written specifically for this project, provided the user with a means of guiding the experiment, while the computer took over the task of transducer control and data acquisition and interpretation.

## 2.3. Software

The software used in this system had two main functions: data acquisition and data interpretation. The data acquisition portion of the software allowed user control of the experimental apparatus and collection of data. The data interpretation portion allowed the user to store data, view charts and graphs, and perform Fourier transforms. There are two primary modes associated with the data acquisition software. The first mode facilitated static testing of the sensor. It took the sensor from zero strain to a full extension of 90 microns while collecting data on the temperature, transducer displacement, and fiber optic strain. The resulting data were saved to hard disk. The second mode collected data on the dynamic performance of the sensor. It oscillated the sensor with an amplitude of 90 microns at user-defined frequencies ranging from 1 to 100 Hertz. The remaining software was devoted to the interpretation of the collected data. The output consisted of two-dimensional graphs with user-defined axes and fast Fourier transforms (FFTs) of the dynamic test data. The FFT section was used to provide a frequency response curve of the sensor over a range of frequencies within the 1 to 100 Hertz range, which could be plotted using the graph function.

The test software for the sensors was written in Turbo Pascal 4.0™ with full Turbo-Vision™ support, and is, henceforth, referred to as the Manager. This software consists of a menu skeleton and data acquisition and processing components as shown in Figure 8. The menu environment is created through a Turbo Pascal utility known as Turbo-Vision. Data acquisition routines use Das-16 Turbo Pascal Units from Quinn-Curtis, Inc. The Manager allows the user to collect data from three input channels (fiber optic sensor, strain gauge, and thermocouple) for any time interval up to 120 seconds (the limit imposed by base memory constraints). In addition, it allows the analysis of data through either amplitude versus time graphs or amplitude versus frequency (i.e., FFT) graphs. The following is a description of the modules embedded within the Manager.

The Manager begins by reading its location and the location of the sensor calibration file from the configuration file created during installation. The DOS command file is then located for use in external operations. Initialization of the Turbo-Vision skeleton is accomplished in the

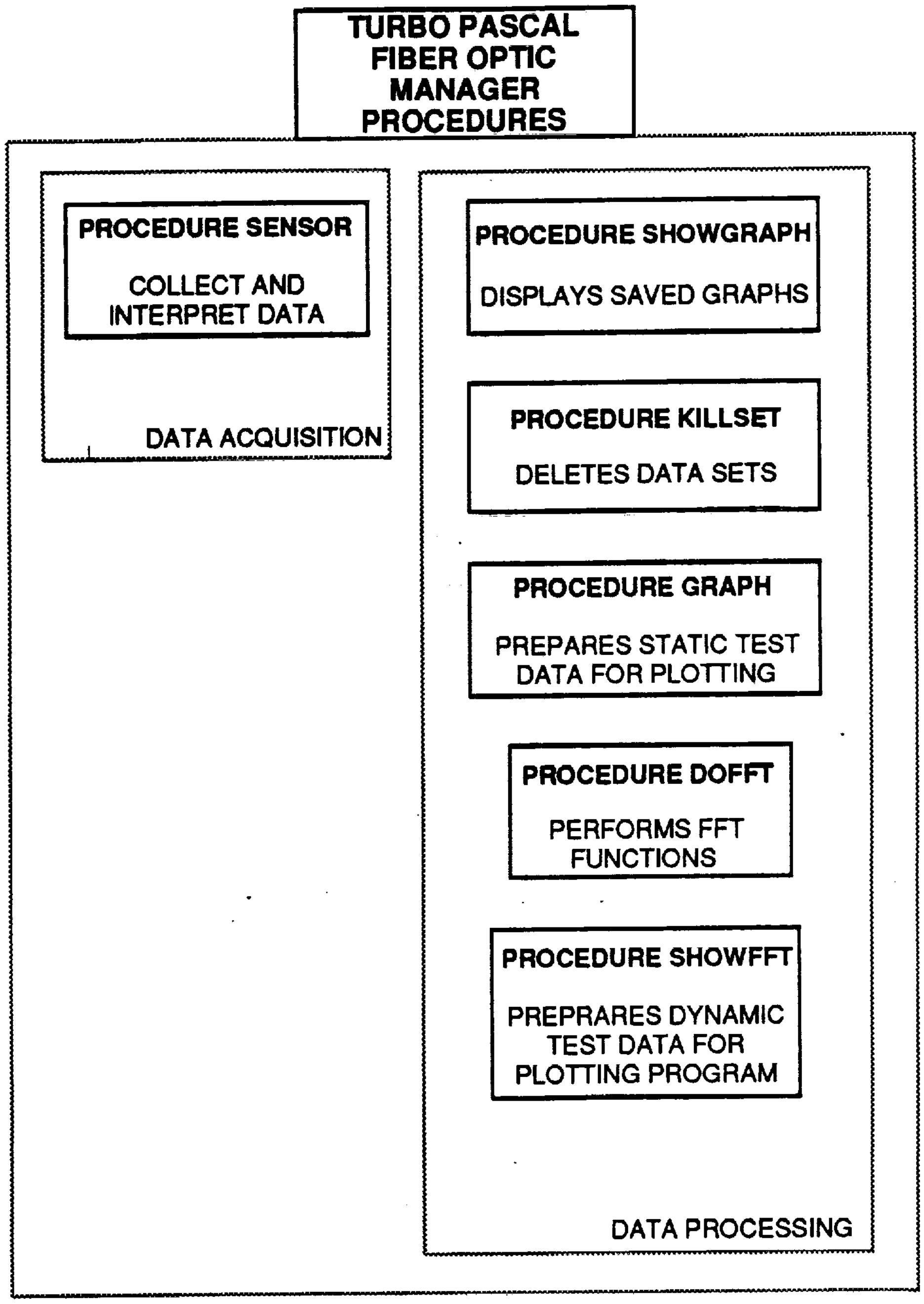

Figure 8. Software Components of Turbo Pascal Fiber Optic Manager.

constructor TFOMAN.Init and its accompanying functions and procedures. TFOMAN is an acronym for (Turbo pascal Fiber Optic Manager) and is the object variable which represents the window background. The color scheme for the background is read using function

TFOMAN.GetPalette. The Menu Bar at the top of the screen is created by procedure TFOMAN.InitMenuBar, while the status line at the bottom of the screen is defined by procedure TFOMAN.InitStatusLine.

Also part of the Turbo-Vision Skeleton are the procedures which control the software's operation. Procedure TFOMAN.ViewFile displays text files on the background and is used to display the help files and the start-up background. Procedure TFOMAN.Idle monitors the software while it is idle. Procedure TFOMAN.GetEvent determines that an event has occurred, and procedure TFOMAN.HandleEvent performs the operation appropriate to the event.

All of the primary operations of the software are embedded in the subprocedures of TFOMAN.HandleEvent. Program development information can be displayed using procedure ABOUT. The Manager may relegate primary processor or video control to DOS with procedure SHUTDOWN and regain control using procedure REINSTATE. The Boolean operator VIDEO in these two routines indicates whether or not to surrender video as well as processor control to DOS.

All data acquisition routines are found within procedure SENSOR shown in Figure 9. Procedure SENSOR begins by initializing the D/A board with procedure INITIALIZE and acquiring a directory name (DIRNAME) in which to store data from procedure DIRECTORY. The Boolean variable SUCCESS indicates whether a name was acquired or the routine was canceled. If a directory was not acquired, the procedure is ended. A dialog box then informs the user of the 120 second time constraint and asks for how long the user wishes to take data. This box remains until either a value (TIME) is given within the acceptable range or the user presses the <ESC> key or the Cancel button. If the user aborts, the rest of the procedure is skipped; otherwise, a message box is created which simply asks the user to press OK when ready to begin. After this is done, processor (but not video) control is relegated to DOS in order to create the subdirectory DIRNAME in the Data directory, along with a dummy file for use in set selection in later routines. Control is then resumed, and procedure SENSOR transfers to subprocedure TAKE_DATA.

Subprocedure TAKE_DATA takes readings for TIME seconds, spools the data to a temporary file (RAWDATA.PRE), and returns to procedure SENSOR the number of readings it has taken. It first creates a dialog box to inform the user that data collection is in progress. Then, the data acquisition rate is set, and the number of data points to be taken is calculated. SETS determines the number of times the Manager will need to transfer data from the A/D buffer

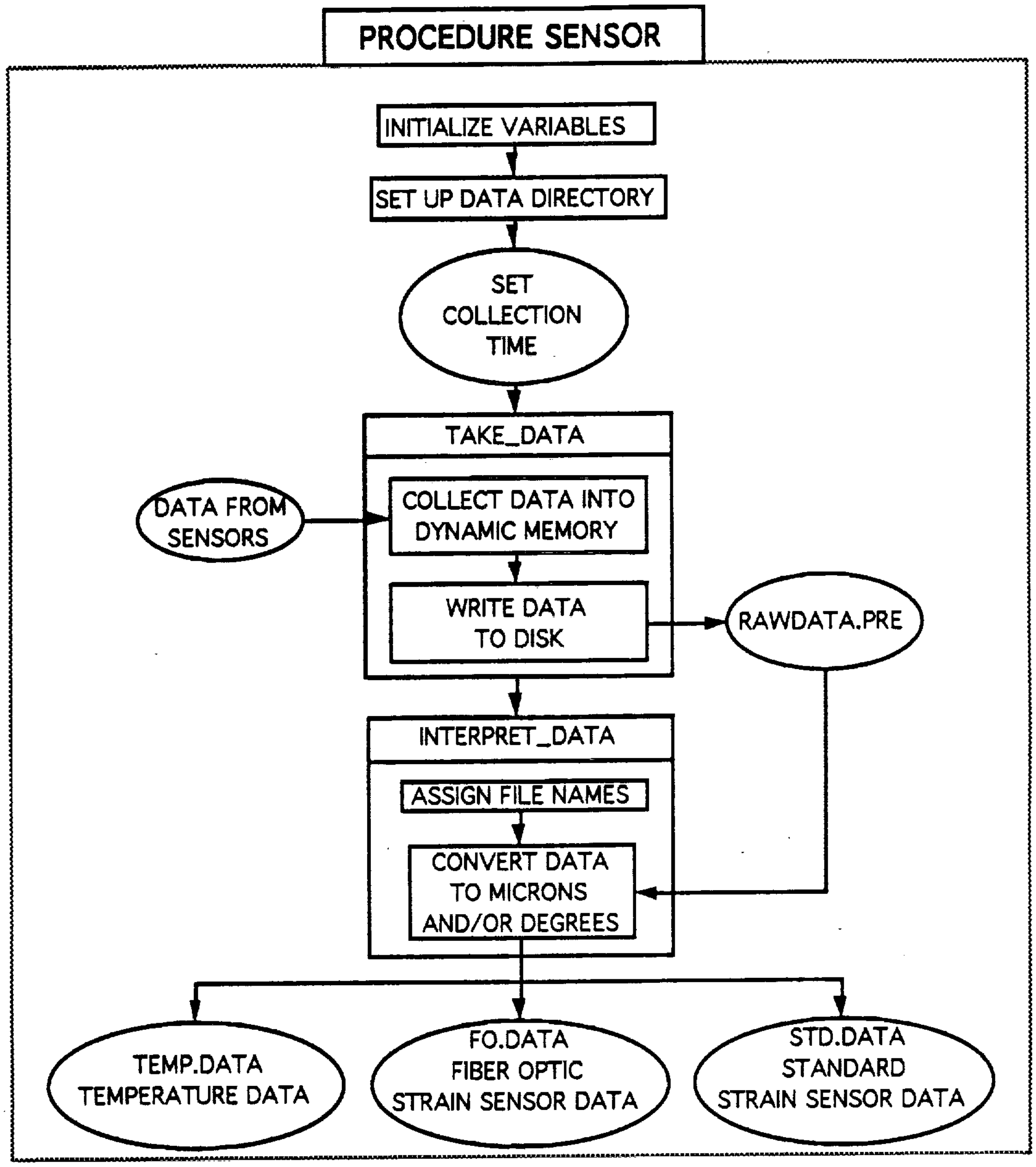
PROCEDURE SENSOR
INITIALIZE VARIABLES
SET UP DATA DIRECTORY
SET COLLECTION TIME
TAKE_DATA
DATA FROM SENSORS
COLLECT DATA INTO DYNAMIC MEMORY
WRITE DATA TO DISK
RAWDATA.PRE
INTERPRET_DATA
ASSIGN FILE NAMES
CONVERT DATA TO MICRONS AND/OR DEGREES
TEMP.DATA TEMPERATURE DATA
FO.DATA FIBER OPTIC STRAIN SENSOR DATA
STD.DATA STANDARD STRAIN SENSOR DATA

Figure 9. Software Components of Procedure Sensor.

to memory due to the memory limit of the buffer, and FINISH determines the number of readings which must be taken after all complete sets have been collected. The Manager then initiates the data collection sequence through procedure SCAN, spooling data into the dynamic memory each time the A/D buffer is filled. Procedure SCAN scans channels 4 through 7 on the A/D board at RATE samples per second, spooling the values into the A/D buffer until

SAMPLES data points have been taken, and returns to procedure TAKE_DATA. If a shortened set is still required to complete data collection, this is also taken through procedure SCAN.

After all data have been collected, the original dialog box is closed and replaced with a box that indicates data are being saved and gives an estimated time to completion. The Manager then opens a temporary file (RAWDATA.PRE) and spools all data in memory to it. The file is then closed, dynamic memory is cleared, and the procedure ends.

After procedure TAKE_DATA ends and procedure SENSOR resumes, procedure SENSOR transfers to subprocedure INTERPRET_DATA. Procedure INTERPRET_DATA first reads the calibration curve of the present fiber optic sensor into an array and then creates a window to inform the user of the interpretation process. Temporary file RAWDATA.PRE is opened for reading, and three files are opened to receive converted data (STD.DAT, FO.DAT, and TMP.DAT for standard sensor, fiber optic sensor, and thermocouple data, respectively). A '1' followed by NUMBER (the number of readings taken in procedure TAKE_DATA) is written to each of the new files to accommodate the plotting software. The message box is then updated, and four values are read from RAWDATA.PRE, converted to their respective values in microns and degrees, and written to each of the output files (with channel 5 being skipped). This continues until all data have been interpreted and stored. Procedure INTERPRET_DATA then closes all files and dialog boxes and returns to procedure SENSOR. Upon completion of procedure INTERPRET_DATA, procedure SENSOR deletes the temporary file RAWDATA.PRE and ends.

All graphs which have been previously generated and saved with the plotting software can be displayed using procedure SHOWGRAPH. This procedure first creates a dialog box to ask the user to select a data set. It then changes to the appropriate directory and transfers processor and video control to the gallery program contained in the plotting software. Control is resumed when the user exits the gallery program.

Data sets may be deleted using procedure KILLSET. This procedure first creates a dialog box to ask the user to select a data set. It then transfers processor control to DOS, deletes the chosen directory (along with all files contained therein), and ends.

FFTs may be performed upon previously collected sets of data using procedure DOFFT. Procedure DOFFT creates a dialog box to ask the user to select a data set. When this has been done, a message box is created to indicate an operation in progress, and procedure DOFFT transfers to subprocedure SEPARATE. Procedure SEPARATE separates the data files FO.DAT and STD.DAT into bins of BINSIZE (400) points (labeled FO1.PFT, FO2.PFT, or STD1.PFT, STD2.PFT, etc.) First, the number of bins which will need to be created are calculated by dividing the BINSIZE into the number of data points in the original data files. For each bin, a file is created of the form shown above containing a '1' and a number of data points from the original file equal to BINSIZE. This operation is repeated until no more data points are available, and the last bin is padded with zeros until it reaches full size. Procedure SEPARATE then transfers back to procedure DOFFT which relegates processor control to DOS and uses procedure CALCULATE to perform the FFT calculations.

Procedure CALCULATE loads the FFT software into the expanded memory block using the SPAWN procedure (located in SpawnEMS.TPU). This is necessary due to the size limitations of base memory. An FFT is performed upon each PFT file to create FFT files, and then all PFT files are deleted. The procedure then transfers back to procedure DOFFT which resumes primary processor control and ends.

FFT files are plotted using procedure SHOWFFT. This procedure creates a dialog box to ask the user to select a data set and then creates another to ask the user which bin to plot. The user selects the bin by selecting the appropriate file from the listing of the FFT files in the chosen directory. Once this has been done, primary control is transferred to DOS. From DOS, the SPAWN procedure is used to load the plotting software into an expanded memory block. Primary control is resumed, and the procedure is ended when the user exits the plotting software.

Time versus data graphs are created using procedure GRAPH. First, a dialog box is created that asks the user to select a data set. Another is then created to enable the user to select the input channel (fiber optic sensor, standard sensor, or thermocouple) to plot on the Y-axis. This operation is performed in the BOX subprocedure. If the user has not aborted by this point, the header file is modified by subprocedure HEADER in accordance with the Y-axis chosen.

Procedure HEADER first determines the appropriate header file based upon the Y-axis selected. Both the original header file and a new header file in the chosen subdirectory are opened. The appropriate title and axis names are written to the new file, and all line labels are deleted. Both files are then closed, and procedure GRAPH is resumed. Procedure GRAPH then transfers to procedure AVERAGE. Procedure AVERAGE simply averages each 400 points in the selected data file and creates a plot file from the averaged data before returning to procedure GRAPH. Primary control is then transferred to DOS, and the SPAWN procedure is used to load the plotting software and allow the user to plot the graph. When the user exits the plotting software, primary control is reasserted, and procedure GRAPH ends. Details of the manager software are given in Appendix B.

# 3 Sensor System

## 3.1 Introduction

Light loss in amplitude-modulated sensors can be induced with transmission, reflection, microbending, absorption, scattering, and fluorescence techniques. The reflective concept is especially attractive for sensor applications due to simplicity and low cost. This concept is shown in Figure 10.

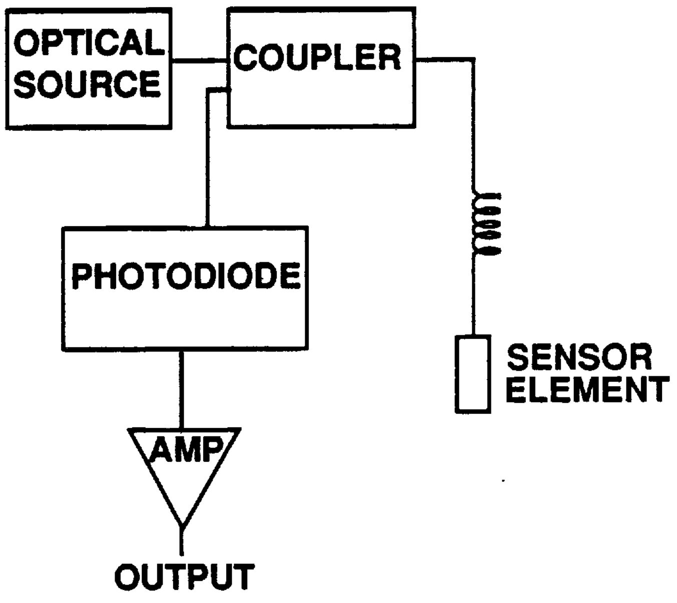

Figure 10. Simplified Diagram of an Amplitude-Modulated Fiber Optic Strain Sensor.

An optical source, for example an LED, supplies the light to the sensing element by means of an optical fiber. Light, modulated by the strain, is returned by the same optical fiber and routed through a coupler to a photodiode. The output of the photodiode is an electrical signal that is amplified and processed by digital means.

The sensing element itself consists of a metal jacket or tube that contains the optical fiber and a plug separated by a gap of about 50 micrometers. As shown in Figure 11, the front face of the plug has a 100% reflective metallic coating and, thus, acts a mirror. The fiber and the plug are bonded to the substrate to be tested. Any expansion or contraction of the substrate is transferred to the sensor element. In turn, the expansion will change the gap width between the fibers and, hence, the amount of reflected light captured by the input/output fiber of the sensing element.

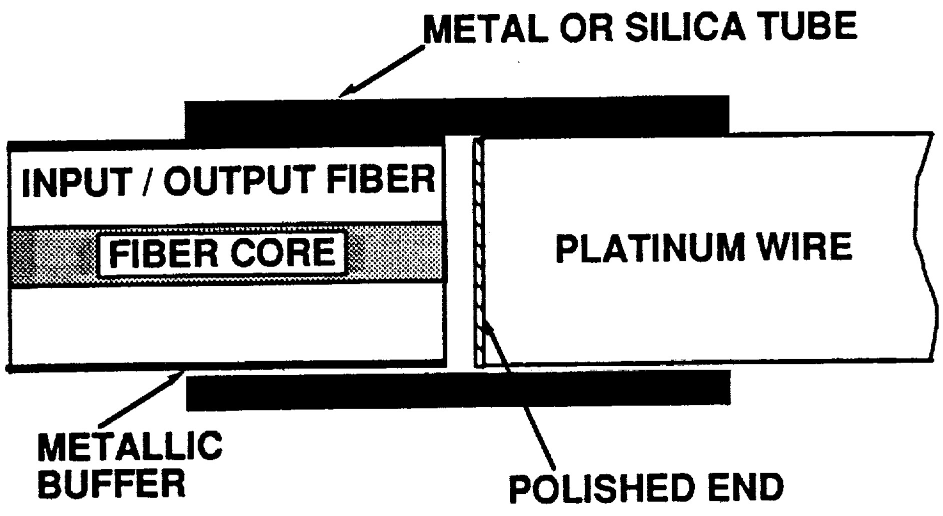

Figure 11. Sensing Element of Amplitude-Modulated Point Strain Sensor.

Unfortunately, the light intensity of an optical LED is not stable over a long period of time. Thus, changes in light intensity at the source will be interpreted as strain changes at the sensing element. It was thought that this problem could be solved with an optical wheatstone bridge that can be fabricated from an assembly of 1 by 2 optical couplers. Figure 12 is a diagram that shows how the wheatstone bridge was incorporated into the point strain sensor. This new configuration requires two LED sources. The LEDs were modulated at different frequencies, for example, with 1.5 kHz and 15 kHz signals, to differentiate between the two sources. The signals were submitted to the wheatstone bridge which is a combination of four optical couplers as shown in Figure 12. The outputs were routed to two photodiodes and recovered through electronic filters as signals $A_{1.5}$, $A_{15}$, $B_{1.5}$, and $B_{15}$, where the subscripts refer to the signal modulation frequencies. An actual output signal from the sensor was obtained by forming the ratio

$$S = \frac{A_{1.5} \bullet B_{15}}{A_{15} \bullet B_{1.5}} \tag{1}$$

Now assume that the LED source modulated with the 1.5 kHz signal has drifted in intensity by a factor of f. At the same time, the applied strain at the sensor head was not changed. Thus, the

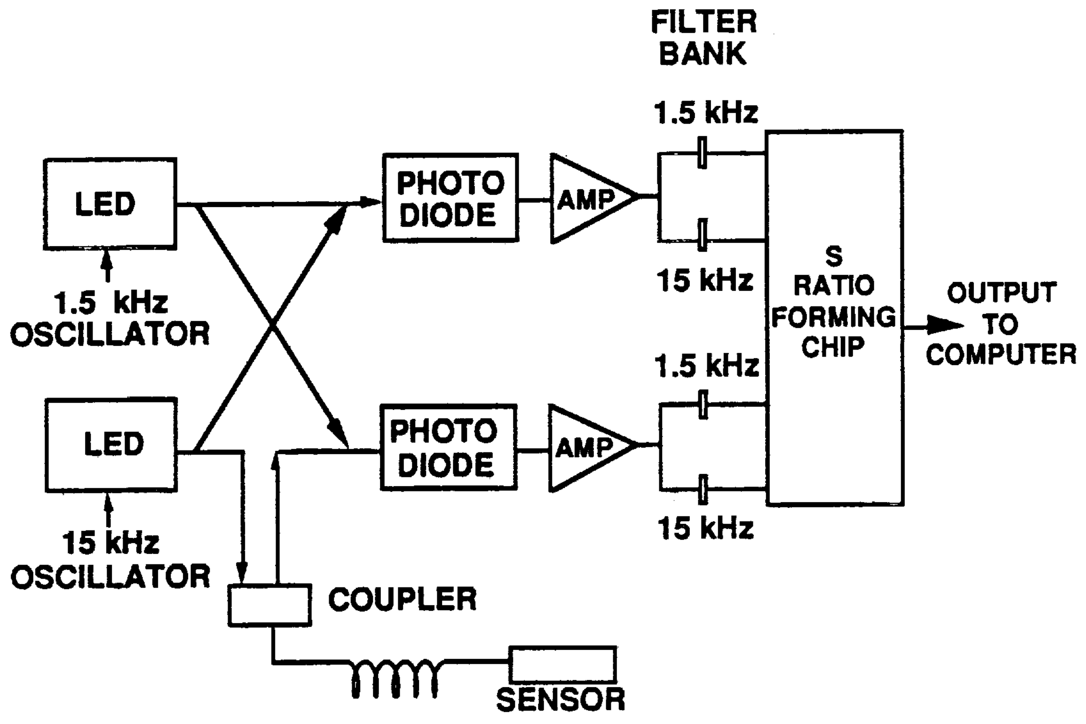

Figure 12. Point Strain Sensor with Optical Wheatstone Bridge.

outputs $A_{1.5}$ and $B_{1.5}$ are $fA_{1.5}$, and $fB_{1.5}$ and, hence, S remains unchanged as shown in Equation (2).

$$S = \frac{fA_{1.5} \bullet B_{15}}{A_{15} \bullet fB_{1.5}} = \frac{A_{1.5} \bullet B_{15}}{A_{15} \bullet B_{1.5}} \tag{2}$$

Similarly, if either receiver drifts in gain, such that the outputs are $g_1\ B_{1.5}$, $g_1\ B_{15}$, $g_2\ A_{1.5}$, and $g_2\ A_{15}$, S will still remain unchanged because

$$S = \frac{g_2A_{1.5} \bullet g_1B_{15}}{g_2A_{15} \bullet g_1B_{1.5}} = \frac{A_{1.5} \bullet B_{15}}{A_{15} \bullet B_{1.5}} \tag{3}$$

Changes in strain due to the sensor head will only appear in $B_{15}$ and, thus, will not be canceled. A single sensor system contained five printed circuit boards, consisting of two transmitter boards, two receiver boards, and one signal generator board. The single sensor output was digitized and processed by the 386 SX computer in real time.

Figure 13 shows experimental measurements made with a preliminary laboratory model of the strain sensor. No actual strain was applied, and an optical wheatstone bridge was not used. Only the gap width between an optical fiber and a mirror was changed over a range of 200 microns. But normal sensor operation covers only a gap distance of about 100 microns, for example, from 50 to 150 microns. The initial sensor output voltage at 50 microns is zeroed to maintain sensitivity and to utilize the full dynamic range of the analog-to-digital converter. Any difference between the initial reading and the current sensor reading is amplified such that a 100 micron extension produces an output of nearly 10 volts. This value represents the upper limit of the analog-to-digital converter. Details of the electronic circuit associated with the signal detection system are described next.

## 3.2 Detector

The basic function of the detector system is to transmit optical energy to the sensor head, receive the sensor modulated signal, and then pass this signal to the computer to be translated into strain. This function requires a steady transmitter signal, low circuit noise, and amplification of small changes in the sensor signal to take advantage of the full dynamic range of the analog-to-digital converters. A circuit with these characteristics was designed, constructed, and tested. This circuit contained the optical wheatstone bridge shown in Figure 12. Preliminary tests indicated a signal power that was lower by approximately a factor of at least four compared to a system without the bridge. Noise voltage fluctuations of about $\pm 1$ volt, corresponding to about $\pm 10$ microns were also observed with the bridge. Without the bridge, noise fluctuations were minimized, but at the same time the system was exposed to LED drift. Nevertheless, the strain sensitivity of a circuit without a bridge, which included LED drift, was far better than the sensitivity of a circuit with an optical wheatstone bridge. Some attempts were made to detect the noise and drift characteristics with the second receiver board. This signal could have been used as feedback to the LED driver chip to cancel LED drift and low frequency noise. Unfortunately, the noise characteristics in the two receiver channels were uncorrelated; therefore, the feedback experience was terminated.

A simplified block diagram of the circuit in its present form is shown in Figure 14. Photographs of the generator, transmitter, and receiver boards are presented in Figures 15, 16, and 17. A photograph of the assembled detector system is presented in Figure 18. Each printed circuit board is discussed in detail in Appendix C.

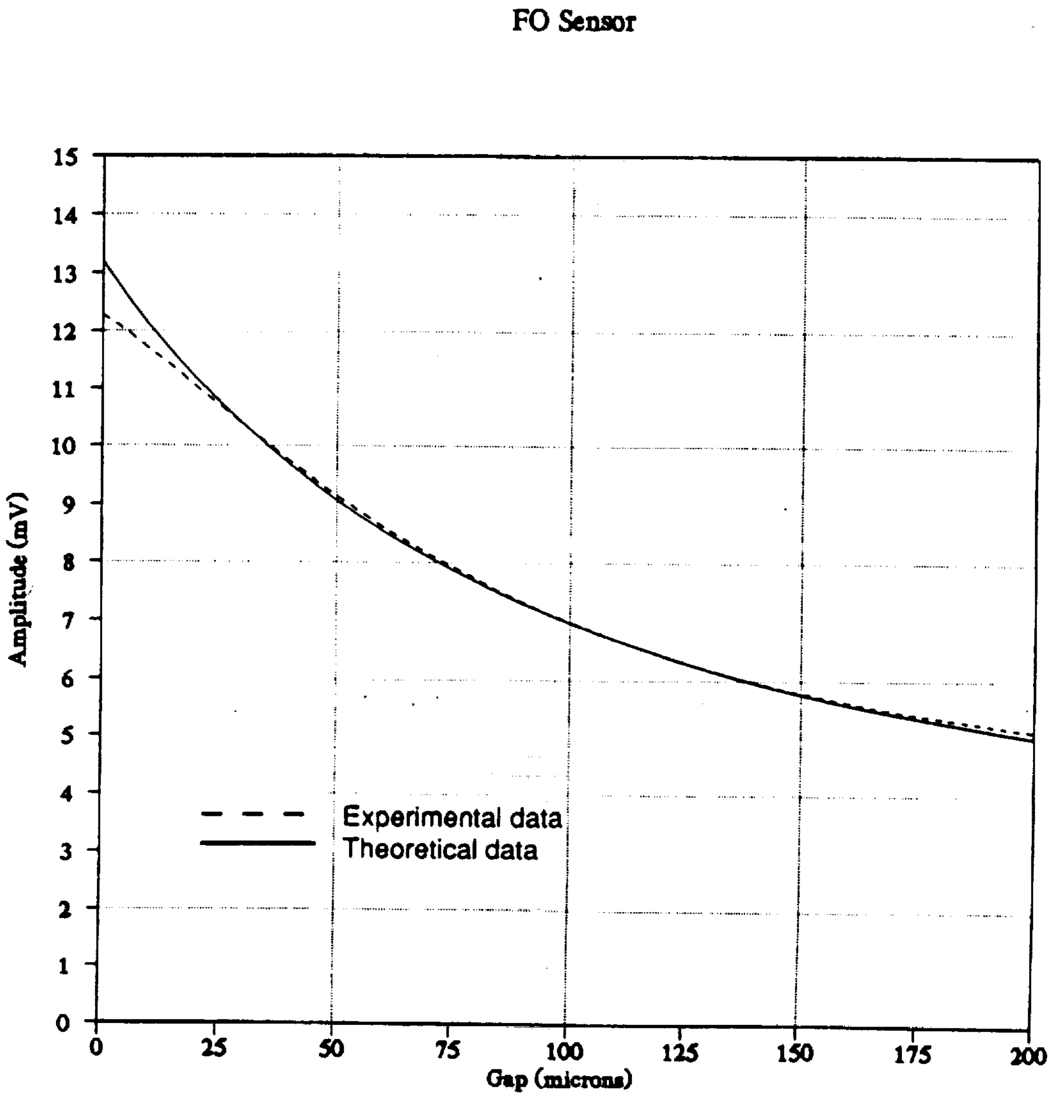

Figure 13. Theoretical and Experimental Amplitude Modulated Sensor Data.

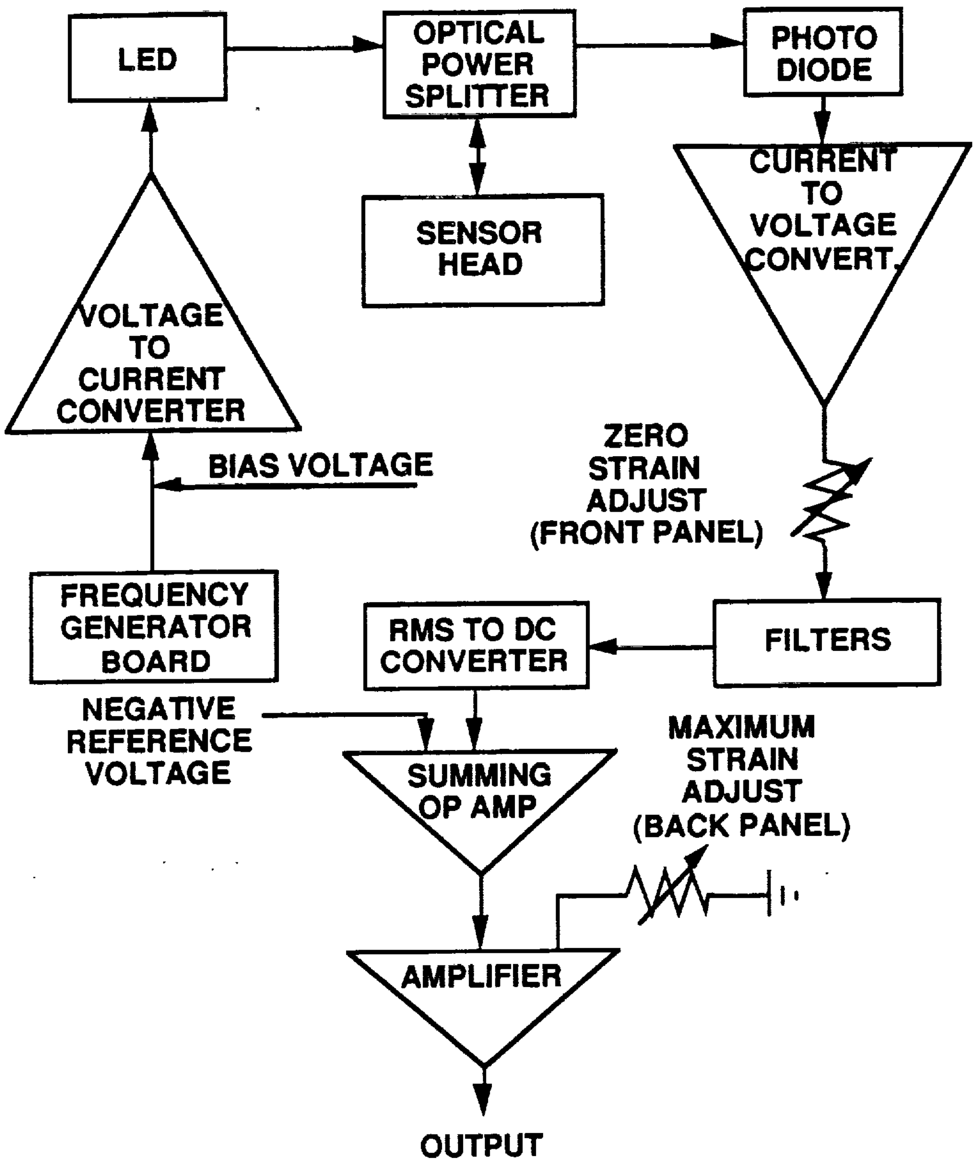
LED
OPTICAL POWER SPLITTER
PHOTO DIODE
SENSOR HEAD
CURRENT TO VOLTAGE CONVERT.
VOLTAGE TO CURRENT CONVERTER
BIAS VOLTAGE
ZERO STRAIN ADJUST (FRONT PANEL)
FREQUENCY GENERATOR BOARD
RMS TO DC CONVERTER
FILTERS
NEGATIVE REFERENCE VOLTAGE
SUMMING OP AMP
MAXIMUM STRAIN ADJUST (BACK PANEL)
AMPLIFIER
OUTPUT

Figure 14. Block Diagram of Simplified Strain Sensor.

## 3.3 Sensor Head

The sensor head designed for this project was conceptually simple but difficult to implement due to many engineering problems. Reliability, adverse environment survivability,

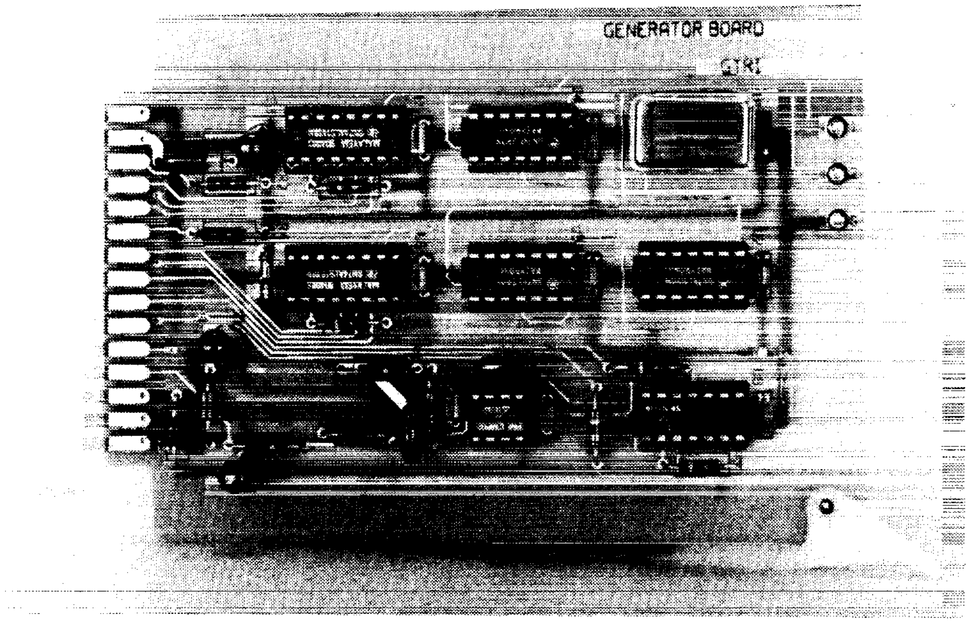

Figure 15. Photograph of Signal Generator Board.

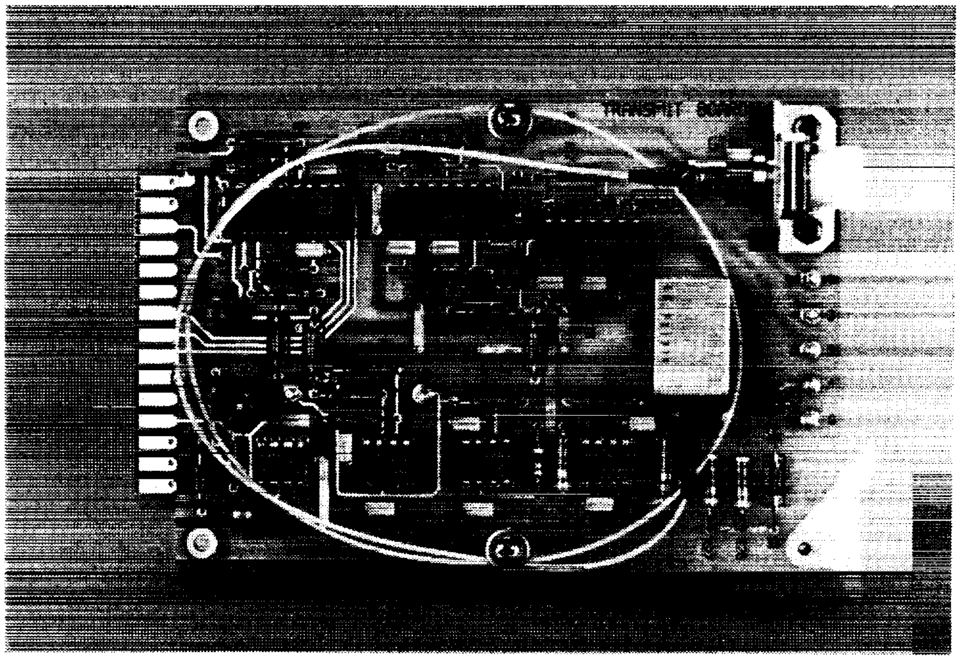

Figure 16. Photograph of Transmitter Board.

ORIGINAL PAGE
BLACK AND WHITE PHOTOGRAPH

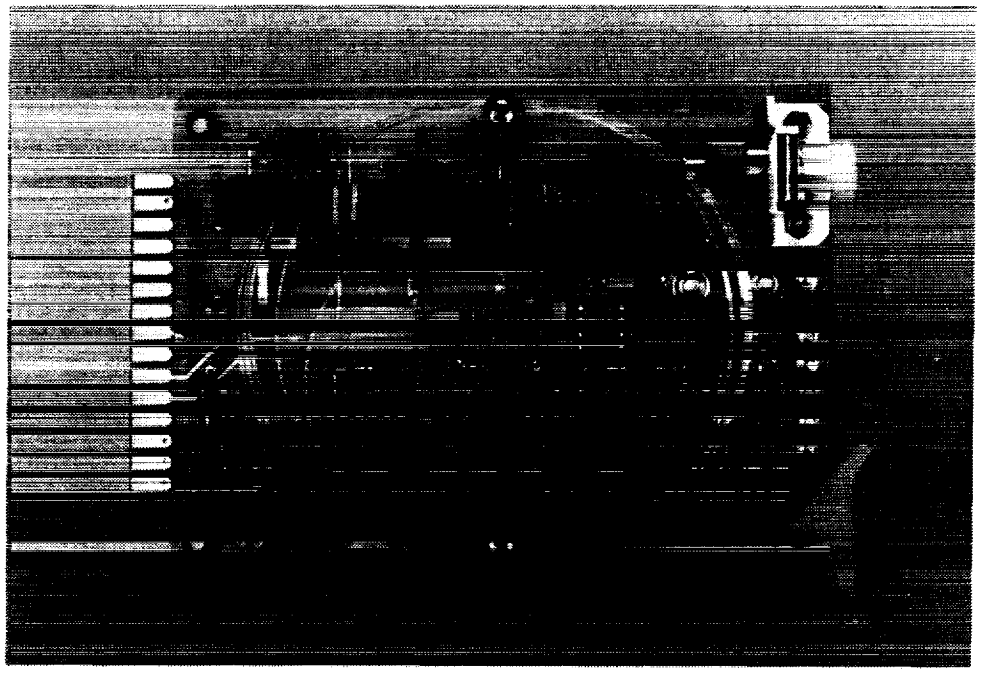

Figure 17. Photograph of Receiver Board.

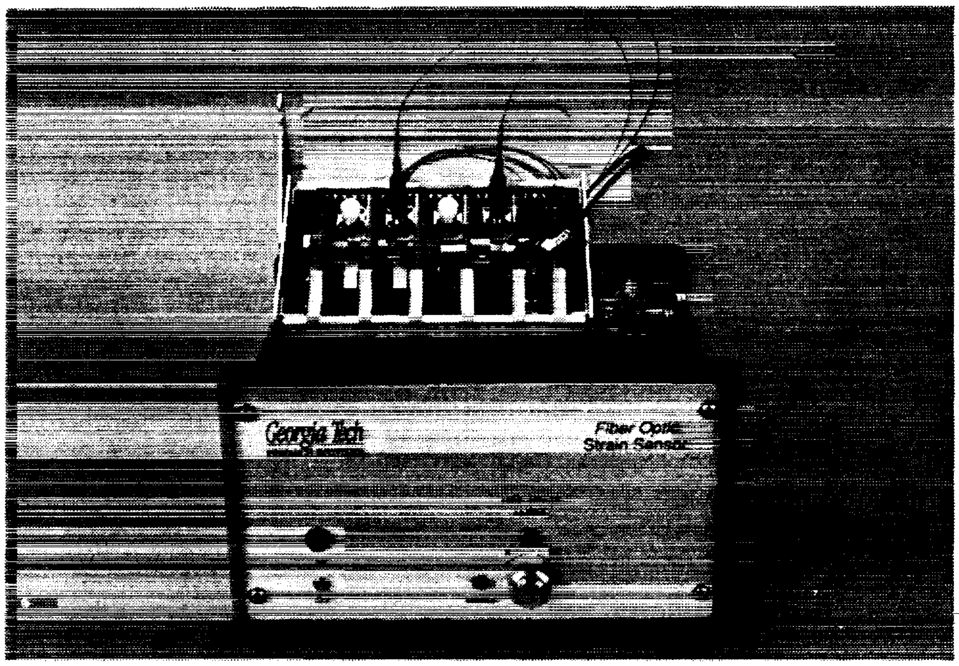

Figure 18. Photograph of Assembled Detector System.

and user friendliness were objectives included in the design criteria of this project. The main objective of survivability at temperatures up to 1,000°C was the driving force for the selection of the sensor material. Most fibers produced for the telecommunications market or for monitoring aircraft-composite materials are made of silica ($SiO_2$) material surrounded by a protective buffer. Teflon is selected to withstand the curing temperatures (~ 400°F) encountered in graphite-epoxy composites. Gold coated fibers, purchased from Fiber Guide Industries, are rated at a continuous operating temperature of 750°C but may be used at 900°C for short time periods (few minutes). The melting point of gold is 1,060°C. The gold softens at 900°C, but that will not deteriorate the performance of the sensor head. The sensor, as engineered at the present time, was designed for one-time use and for a maximum temperature of about 900°C. Design details of the head are illustrated in Figure 19.

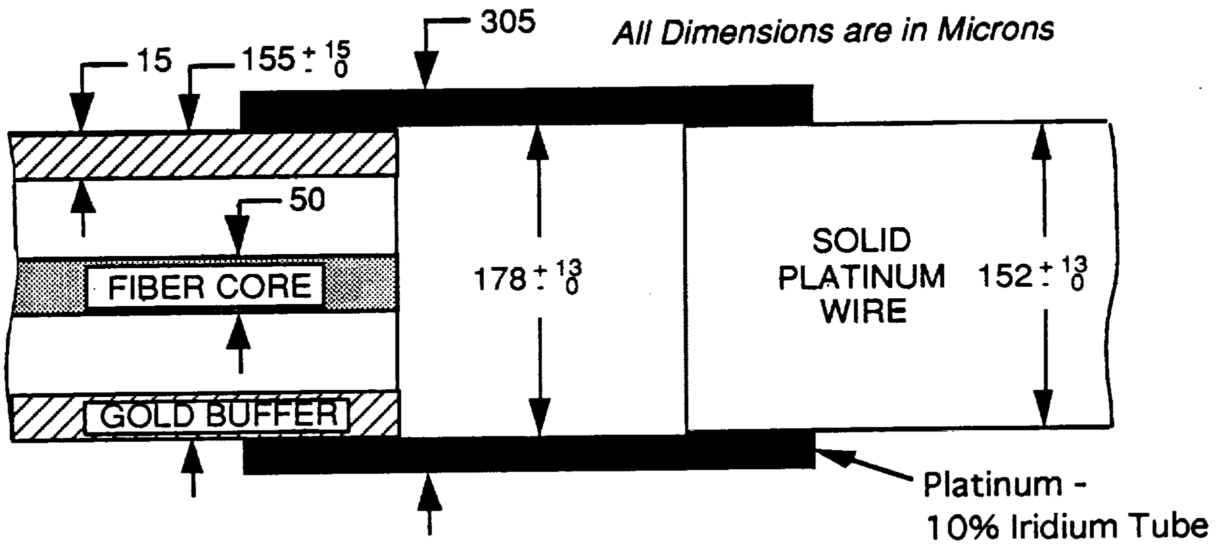

Figure 19. Sensor Head Dimensions.

The gold-coated multimode fiber was cleaved, polished, and inserted into a 1 cm tube made of platinum-10% Iridium alloy. The iridium provided additional strength against bending. The tube had an inside diameter of 0.007 + 0.0005/-0 inches (178 +13/-0 microns), whereas the fiber outside diameter was 155 +15/-0 microns. A mirror was inserted into the other end of the platinum tube to provide the necessary optical reflection. Most mirror designs failed to provide adequate performance at high temperatures. Designs that were implemented included a gold coated optical fiber. One end of the fiber was polished and coated with various sputtered metallic layers, for example chromium or chromium-platinum. In all cases, the sputtered layer either oxidized or exhibited pits where desorptive gases erupted and, thereby, destroyed the reflective quality of the mirror surface.

In the past, both the mirror and the fiber were soldered to the platinum tube with a fiber optic fusion splicer. Although the time and current of the electric arc were controlled, the excess heat produced by the arc would invariably reduce the mirror qualities in an unpredictable manner. These problems were eliminated by allowing the fiber and mirror components to move freely within the tube and by fabricating the mirror from a solid platinum wire. The wire was cut and polished like an optical fiber. All three component were then assembled under a microscope. The mirror was moved with a micromanipulator until the reflected optical power reached a preset value of 1 microwatt. The ends were then fixed with quick-set epoxy to maintain the preset reflected power value. At that point, the sensor was ready for delivery. The value of 1 microwatt is near the top of the power range that can be measured by the sensor head. When stress is applied, the power decreases until it reaches a minimum value that is beyond the measuring capability of the detector. This power level difference corresponds to a change in gap width of 100 microns. Since the sensor head is 1 cm long, the 100 micron displacement results in a maximum strain of $10^2/10^4$ microns, or 0.01, or 1 percent. The maximum strain value may be increased by means of an amplifier gain adjustment located on the rear panel of the detector system.

When mounted to a substrate, all three components (the fiber, tube, and mirror) must be attached separately to the substrate with high temperature cement, such as 904 Zirconia from Cotronix Corporation. After the cement is cured, the epoxy can be removed with a heat gun. The sensor head is then ready for measurements. This procedure was not followed in the laboratory, because the sensor head was not mounted to a single uniform substrate; it was mounted between two test fixtures. At elevated temperatures, the test fixtures expanded, causing the gap between the mirror and the fiber to decrease. The resultant change in gap width between room temperature and 900°C was greater than 1.5 mm. In general, the sensor head was destroyed, unless a gap width in excess of 1.5 mm was introduced prior to furnace start-up; however, with a 1.5 mm preset gap, the desired output at the detector could not be obtained easily at the maximum temperature of 900°C. This problem was solved by mounting a micromanipulator at the top of the test fixture and by attaching the formerly fixed alumina rod to the micromanipulator as shown in Figure 4.

This test system allowed us to dial in an initial gap width of 2 mm and to adjust the reflected optical power to the value desired prior to strain testing.

The overall performance of the sensor was satisfactory for a period of about one hour. Fiber breakage occurred when the sensor head was exposed to high temperatures for longer time periods. The cause of fiber and tube breakage is probably due to a combination of effects that include small distortions of the alumina rods during furnace closure and sticking of the fiber in the silica or platinum tube. Distortions in the alumina rod could introduce small bends in the optical fiber which could have caused the fiber to stick. In turn, fiber sticking caused the fiber to bow during the heating cycle as the alumina rods moved closer together. Extreme bowing resulted in high fiber stresses and, ultimately, in fiber breakage.

## 3.4 Test Results

A thorough performance check of the high temperature strain sensor included static and dynamic strain tests as well as vibration tests. During static tests, the sensor head was mounted vertically between the high temperature test fixtures at the center of the furnace. Initial tests were conducted at room temperature with the sensor in an unstrained condition to determine the apparent sensor displacements in microns as a function of time due to drift in LED output power. Data from several consecutive 300 second measurements at different times were superimposed to clearly delineate the extent of the LED drift. The results, shown in Figure 20, indicate a maximum apparent displacement of the order of $\pm$ 0.75 micron. If the extent of the test region to be strained is 1 cm, then the corresponding apparent strain due to LED drift is $\pm$ 7.5 x $10^{-5}$. In addition, vibration tests indicate a maximum peak-to-peak amplitude of $\pm$ 0.20 microns. Representative plots at frequencies of 30, 60 and 90 Hz are shown in Figures 21, 22, and 23. These values must be added to the general LED drift to determine an overall apparent displacement of approximately $\pm$ 1.0 microns. Thus, the actual displacement, due to strain, must be greater than 1 micron (39.4 microinches). The minimum strain sensitivity of this sensor system at room temperature is 1 micron/1 cm or 100 microinches per inch. The strain sensitivity at 900°C was determined in a similar manner, i.e., the vibrational displacements measured at that temperature ($\pm$ 0.1 micron) were added to the LED drift. The resultant high temperature strain sensitivity is 6 x $10^{-5}$, or $\pm$ 60 microinches/inch. High temperature vibrational data are shown in Figures 24, 25, and 26. Fourier transforms of the 30 Hz vibration data at room temperature and 905°C are displayed in Figures 27 and 28. The spectra indicate prominent peaks at the driving frequencies as expected; however, other frequencies are excited as well, although usually at a lower amplitude level.

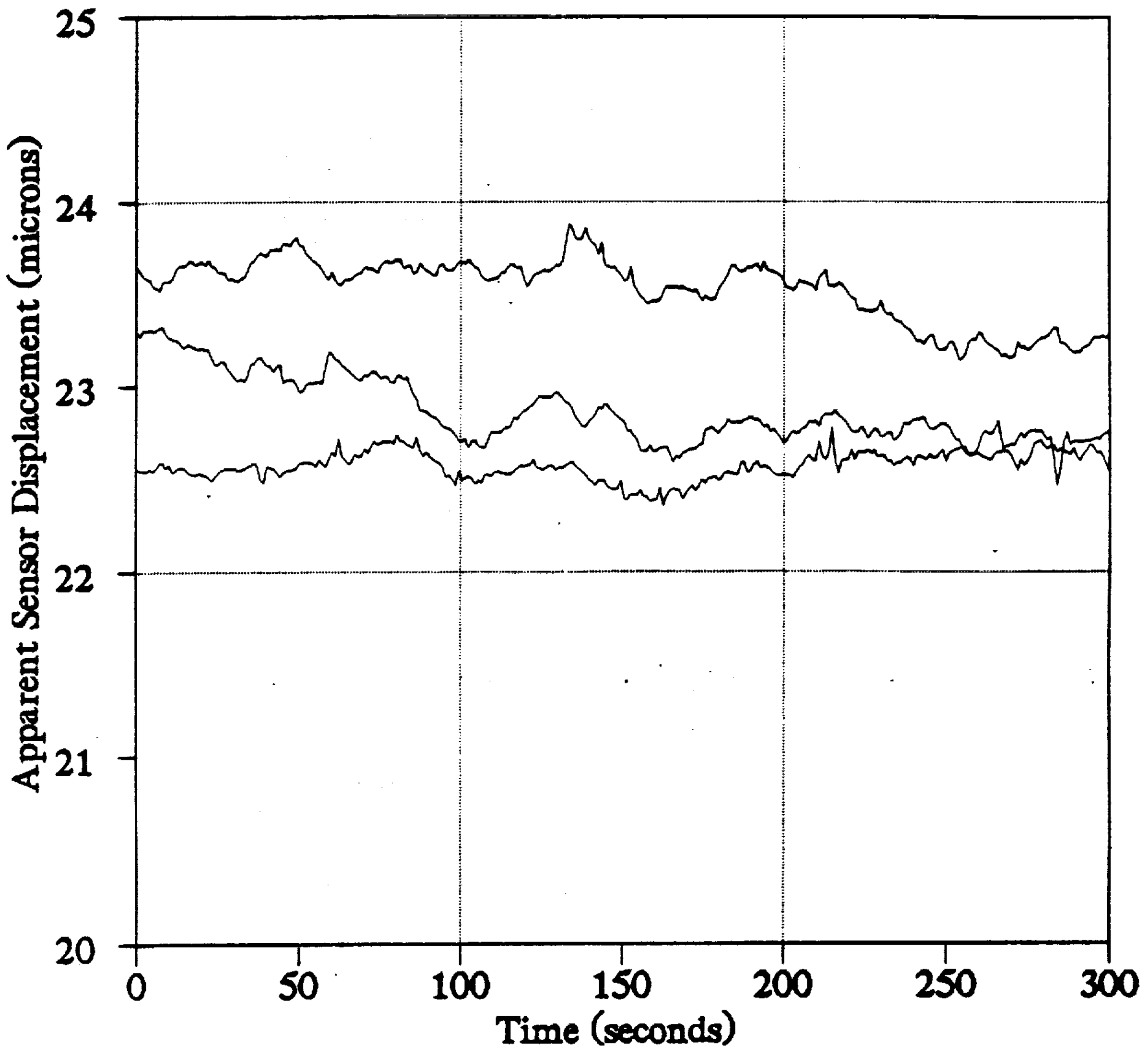

Figure 20. Apparent Displacement as a Function of Time due to LED Drift.

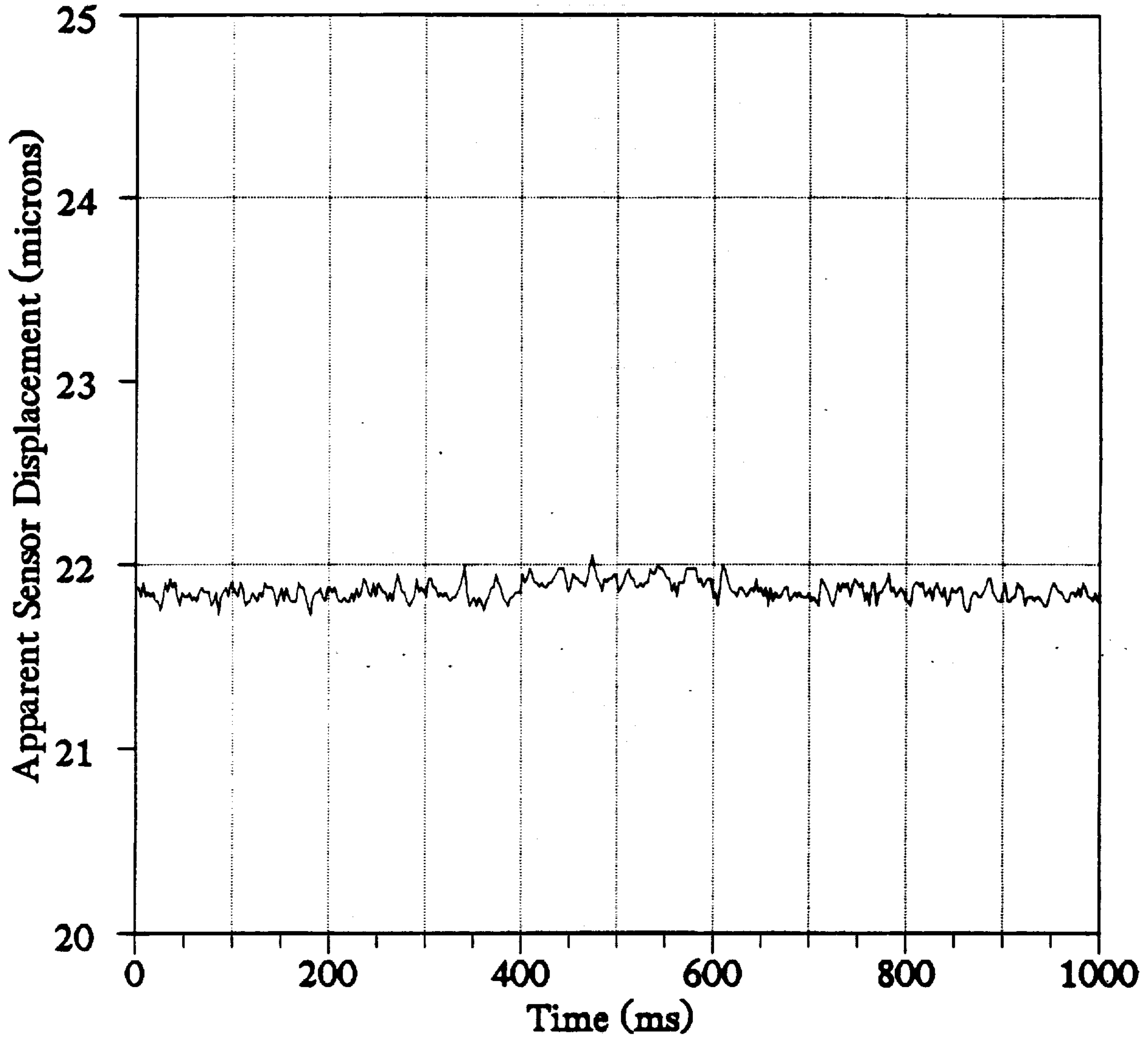

Figure 21. 30 Hz Vibration Data at Room Temperature.

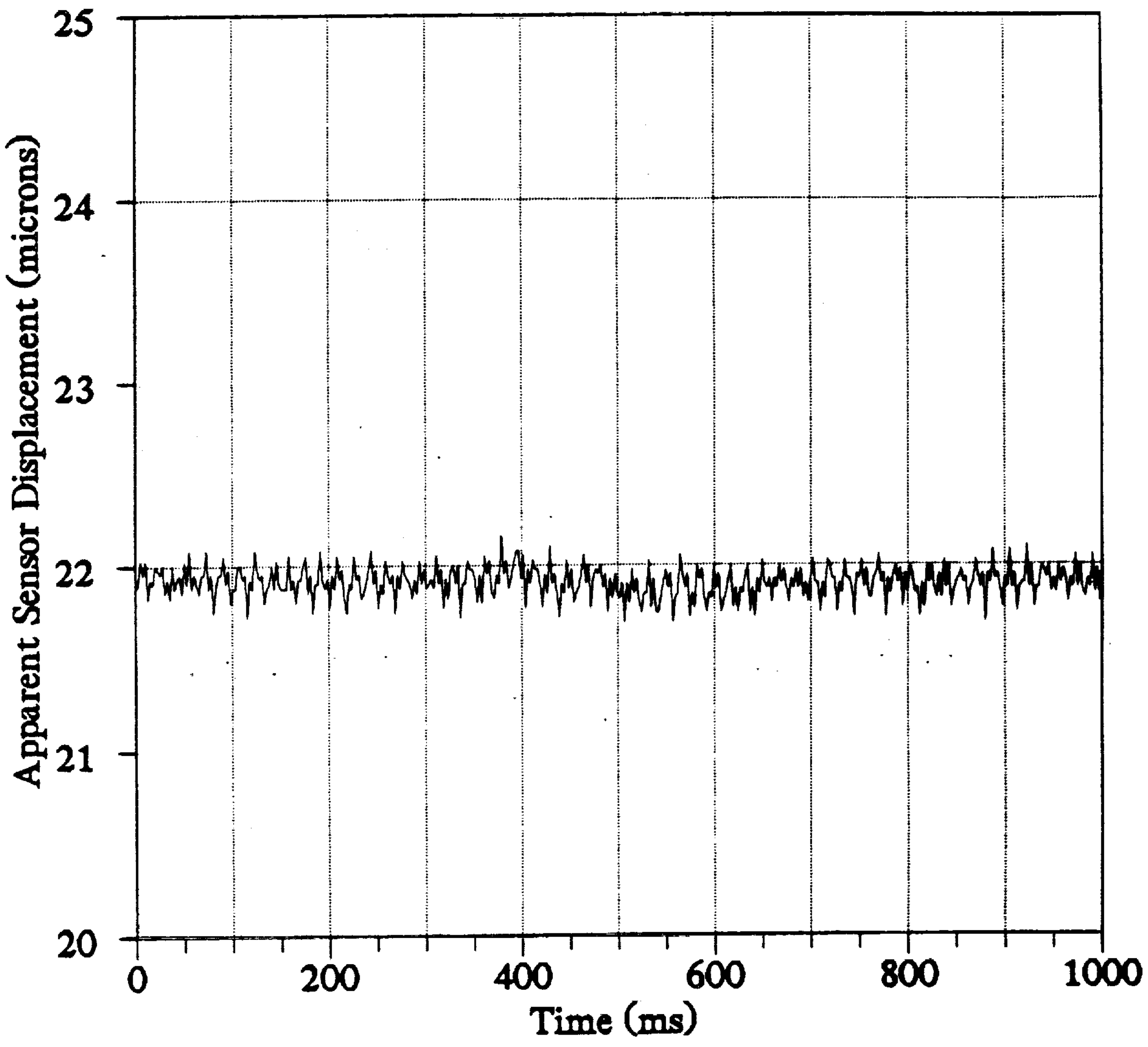

Figure 22. 60 Hz Vibration Data at Room Temperature.

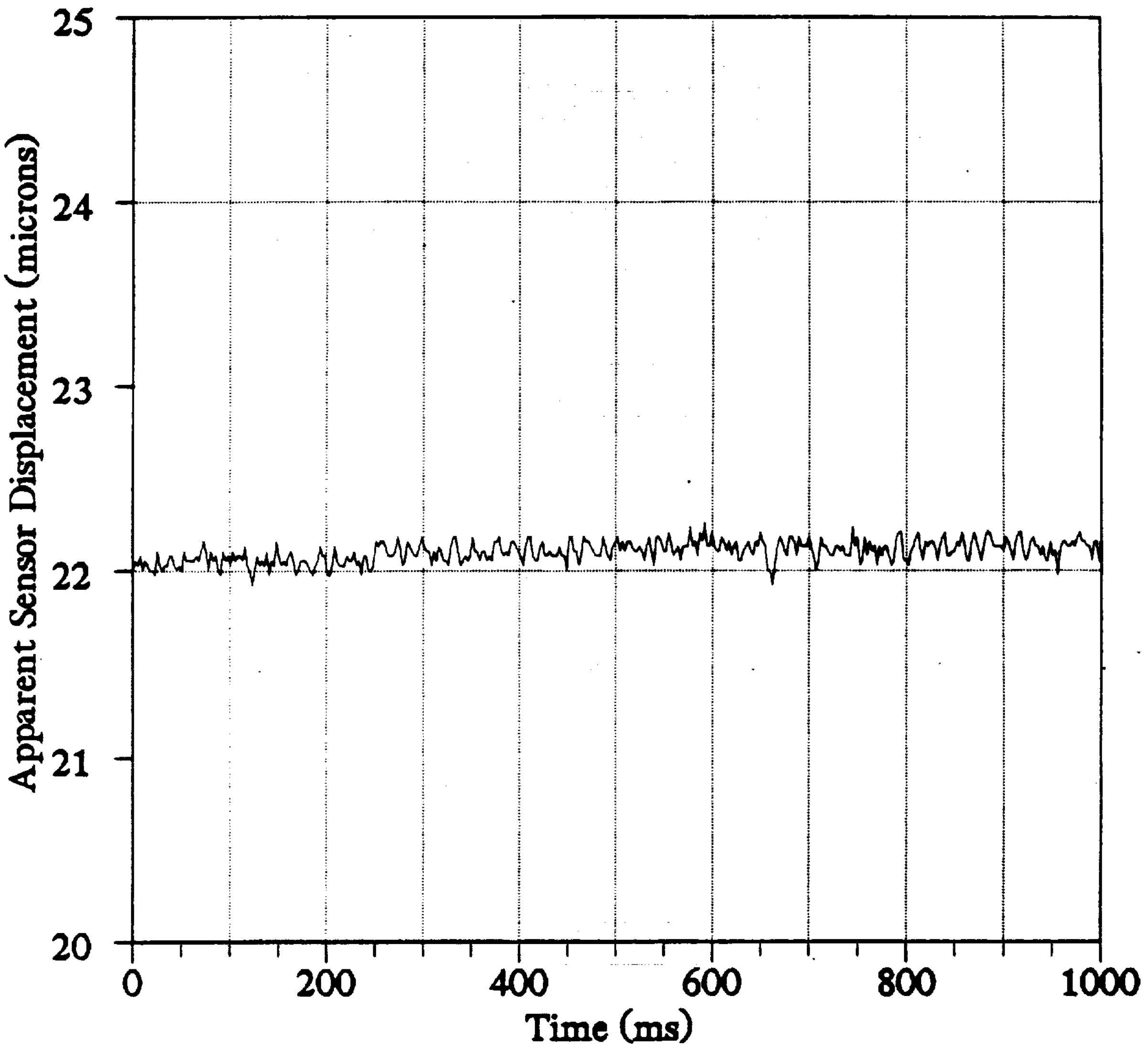

Figure 23. 90 Hz Vibration Data at Room Temperature.

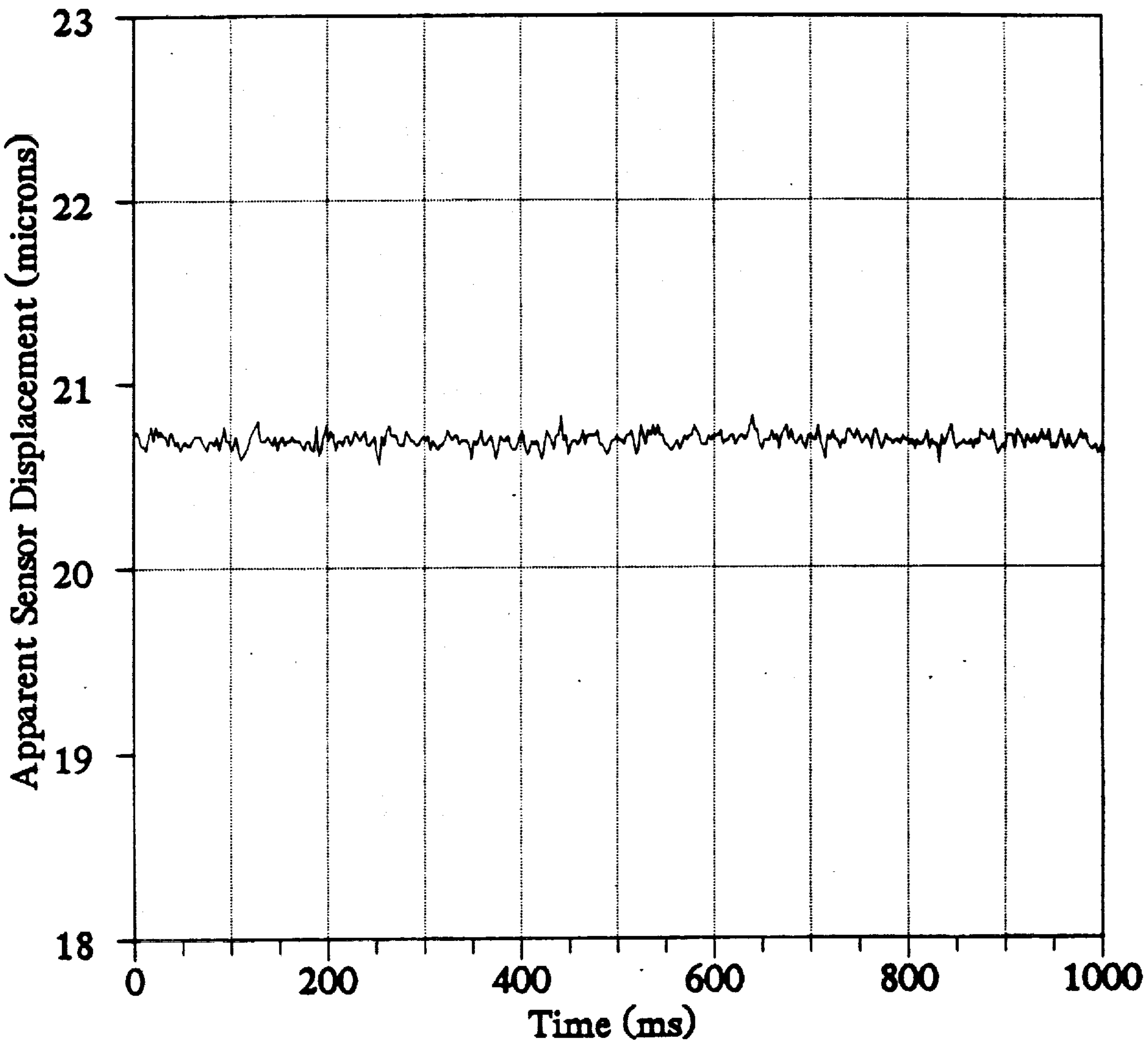

Figure 24. 30 Hz Vibration Data at 905°C.

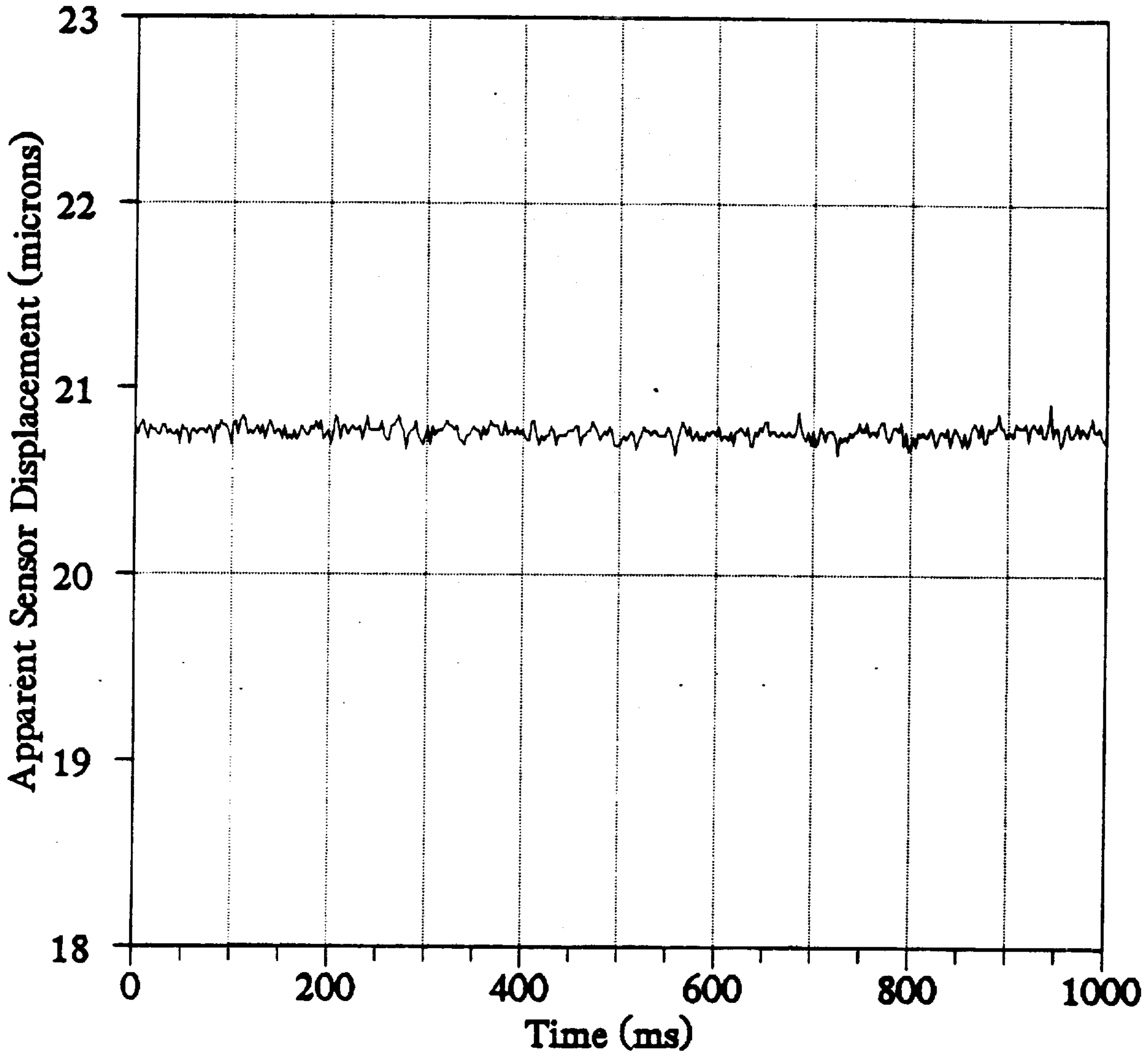

Figure 25. 60 Hz Vibration Data at 905°C.

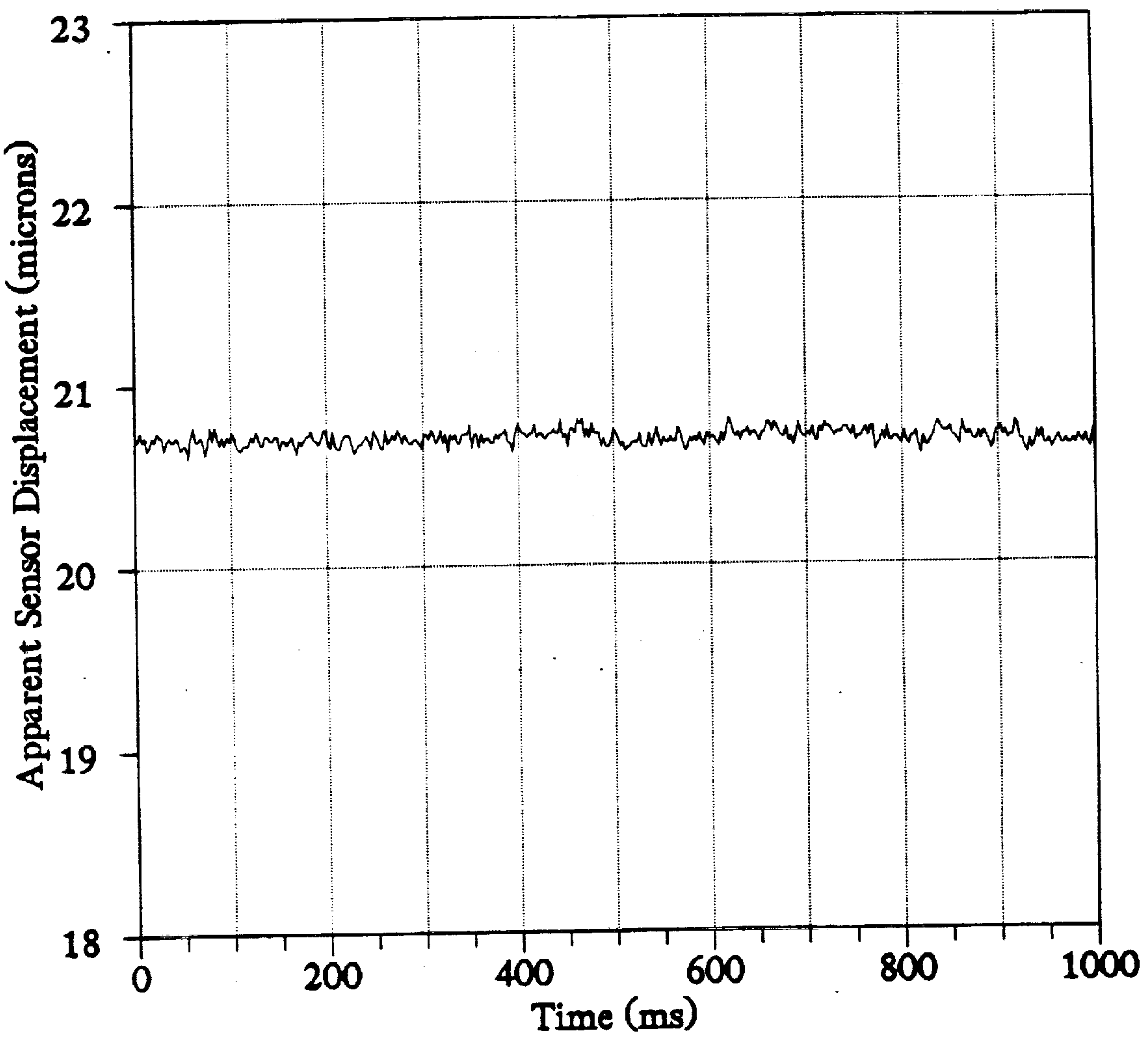

Figure 26. 90 Hz Vibration Data at 905°C.

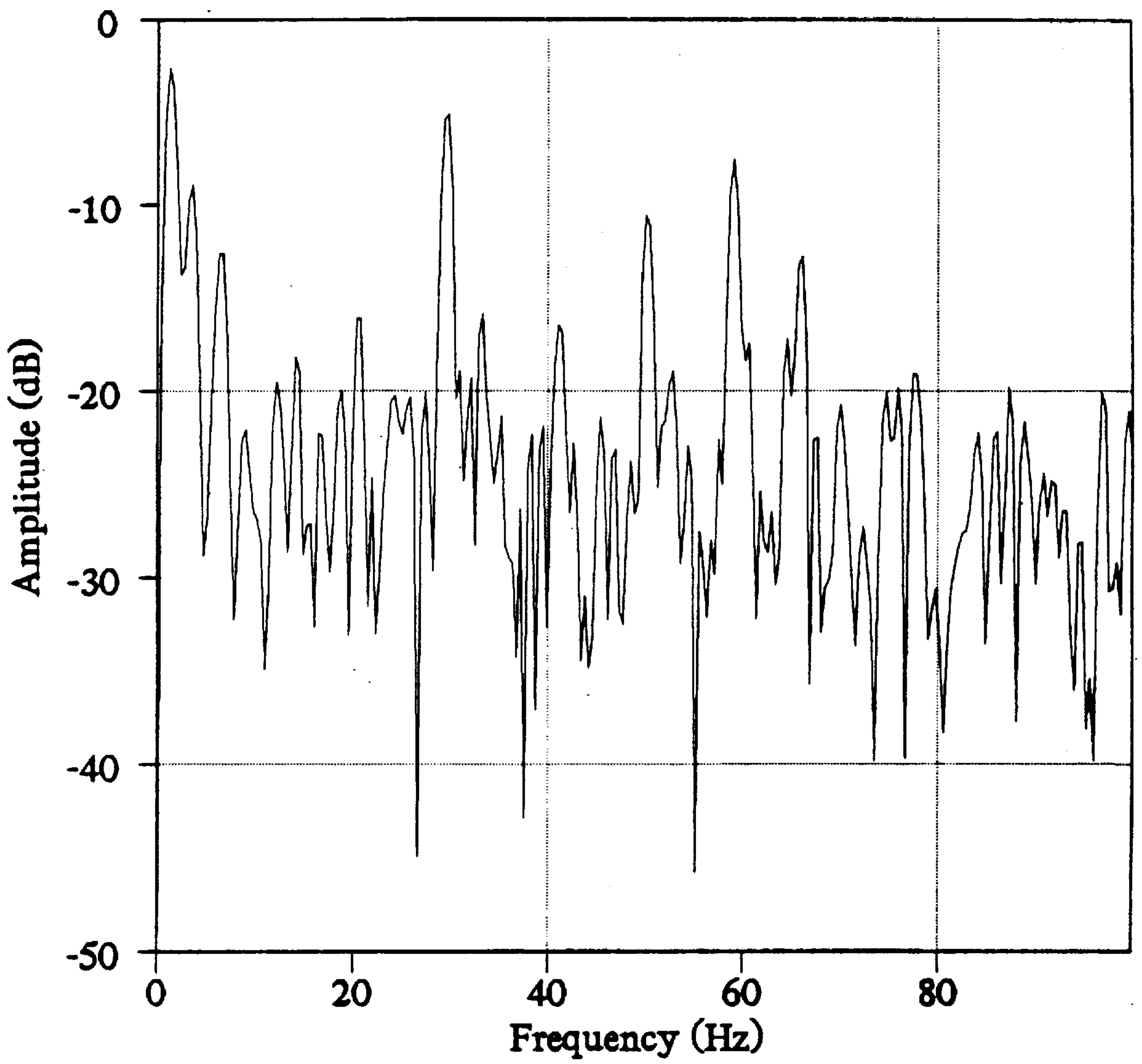

Figure 27. Fourier Transform of Figure 21.

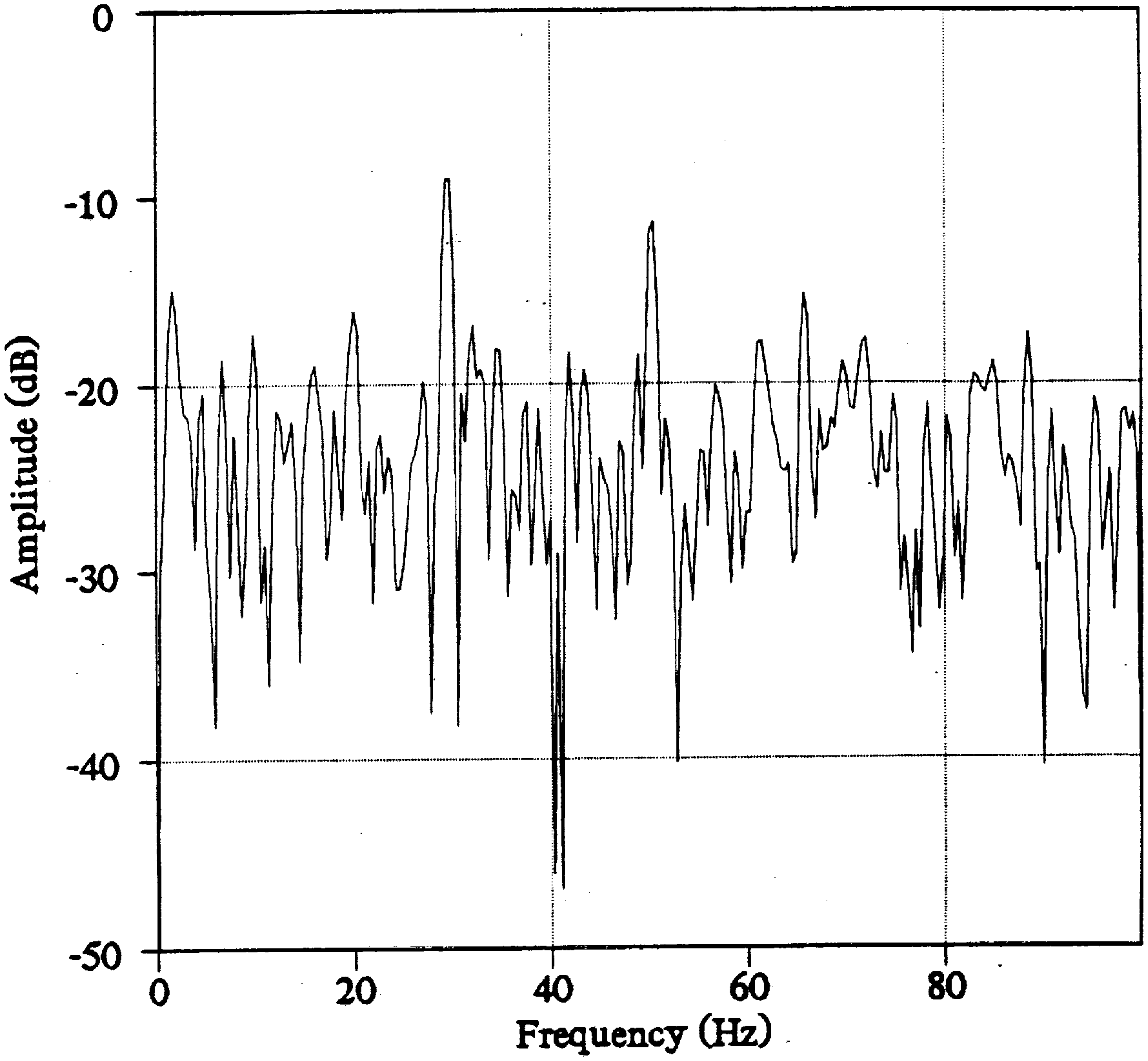

Figure 28. Fourier Transform of Figure 24.

The sensor output in volts must be converted to displacement, $\Delta L$, by means of a calibration curve. At room temperature, the calibration curve is relatively linear as shown in Figure 29. Data collected during the expansion cycle of the piezo transducer were superimposed onto data taken during the contraction cycle. Minor deviations in data between the two cycles can be attributed to the LED drift. At high temperatures (see Figure 30), deviations in the data from linearity are pronounced, because the sensor output voltage is very sensitive to temperature deviations and the furnace could not be controlled within the narrow temperature range required to prevent large fluctuations in output voltage. These fluctuations are an artifact of the furnace controller and are not expected to be present during actual booster strain tests.

Burn rates within the booster nozzle may vary slightly. These variations could translate into nozzle "breathing" modes covering a range of low acoustic frequencies. The point strain sensor can determine the frequency context of these "breathing" modes, because its response is linear within $\pm$ 2 dB from DC to 100 Hz. Figure 31 shows the sensor's response to a typical 30 Hz sinusoidal waveform at room temperature. The Fourier transform of Figure 31 exhibits the fundamental 30 Hz driving frequency plus 60 and 90 Hz harmonics of low amplitude as shown in Figure 32. The system's frequency response at room temperature was obtained by sampling the peak of all fundamental frequencies between 1 and 100 Hz. This response is given in Figure 33. The response at high temperatures (905°C) is shown in Figure 34. Its characteristics are similar to the response at room temperature.

All strain data must be corrected for thermal expansion of the platinum mirror and tube. Figure 35 shows the gap changes for a typical sensor head as a function increasing temperature. The data were collected with the sensor head being free and unattached to any fixture inside the furnace. The negative displacement values indicate a shrinking gap. To compensate strain data for temperature variations the apparent displacements ($D_A$) in Figure 35 must be subtracted from the displacements derived under strain ($D_S$) at the relevant temperature, such that $S(T_i) = D_A(T_i) - D_S(T_i)$, where $S(T_i)$ is the actual strain at temperature $T_i$. The $D_A$ data and corresponding temperature values are stored on hard disk in file TCAL.DAT under the directory name DYNAMIC.

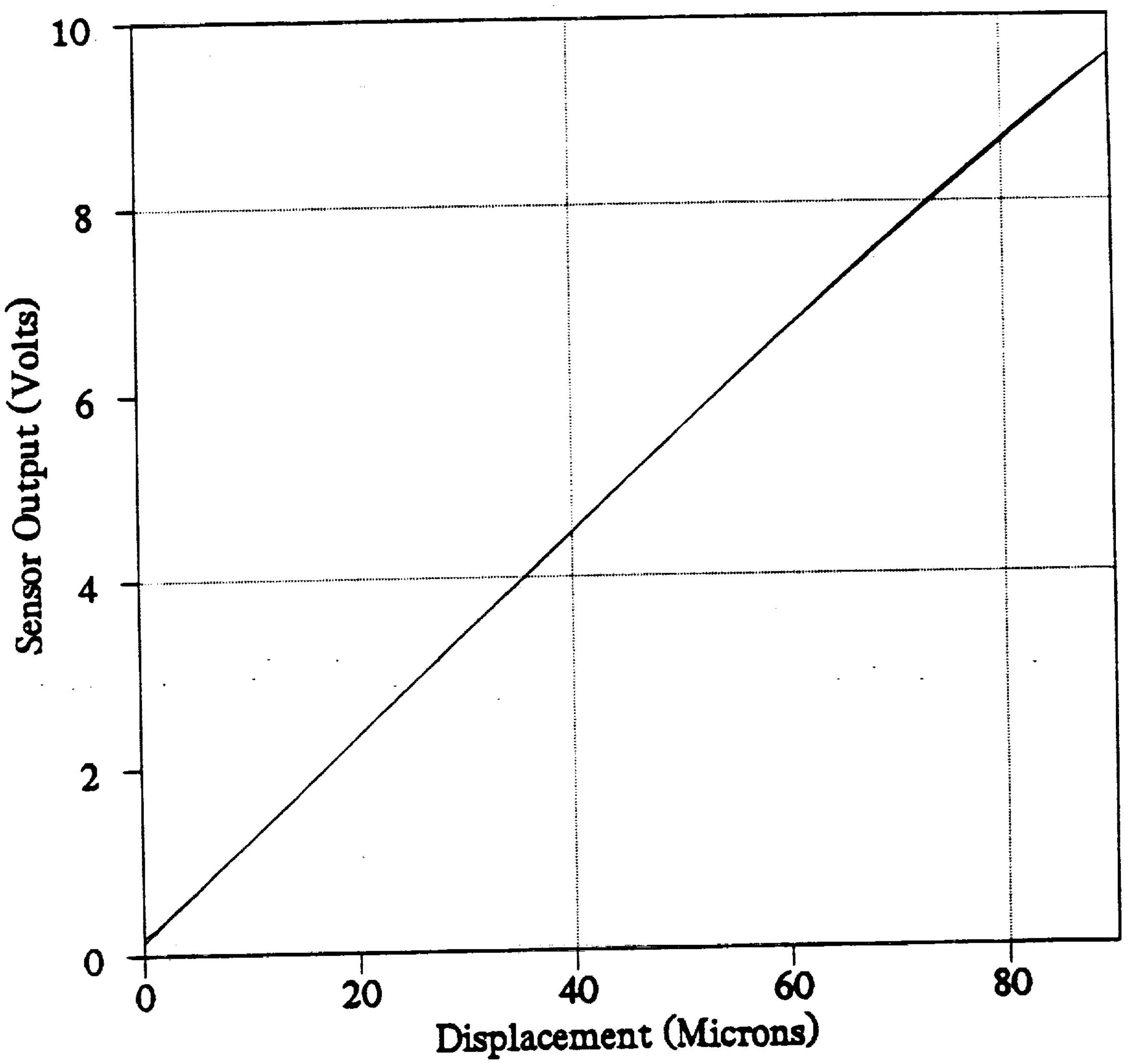

Figure 29. Calibration Curve of Fiber Optic Strain Sensor at Room Temperature.

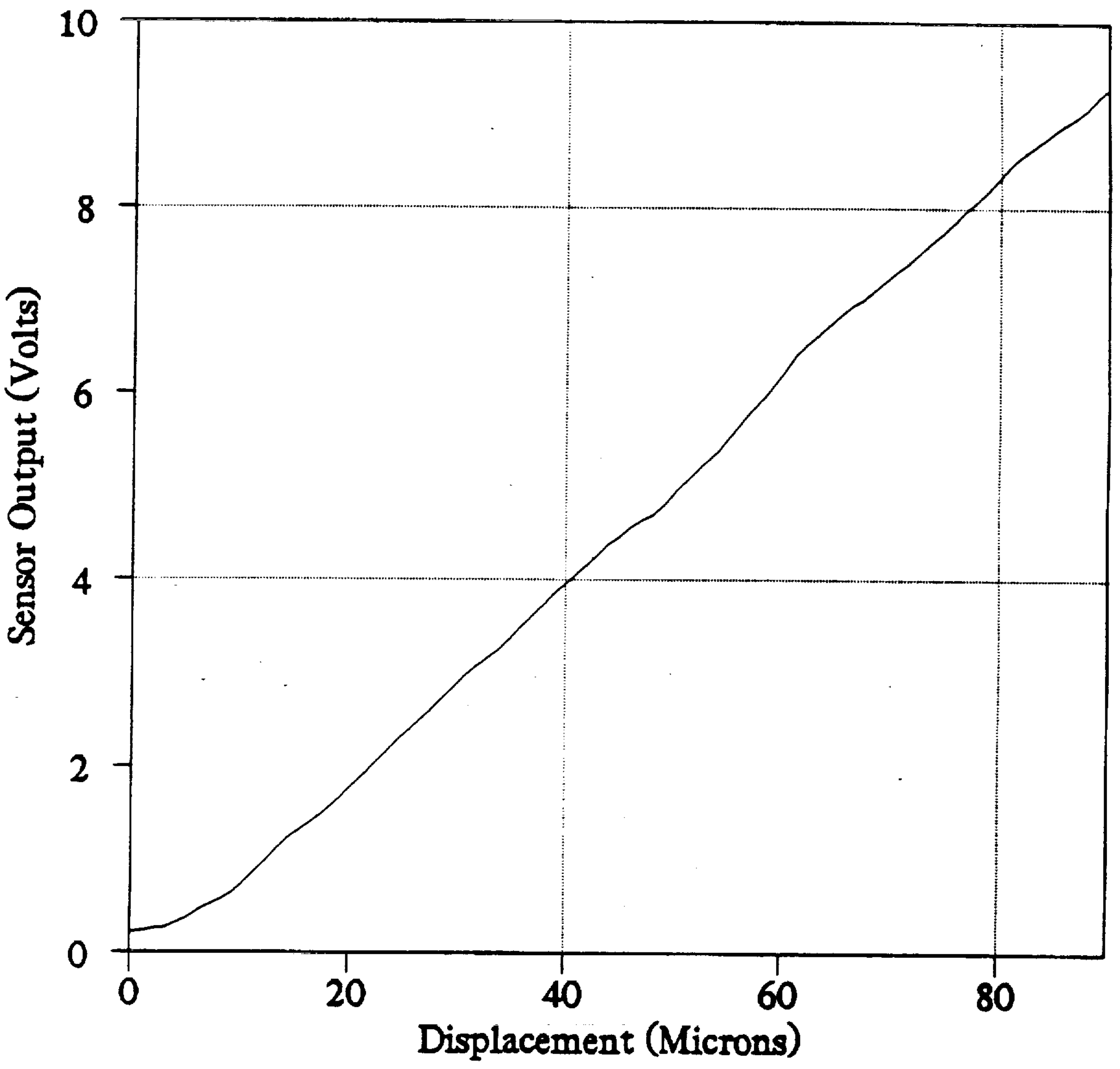

Figure 30. Static Strain Data at 905° C.

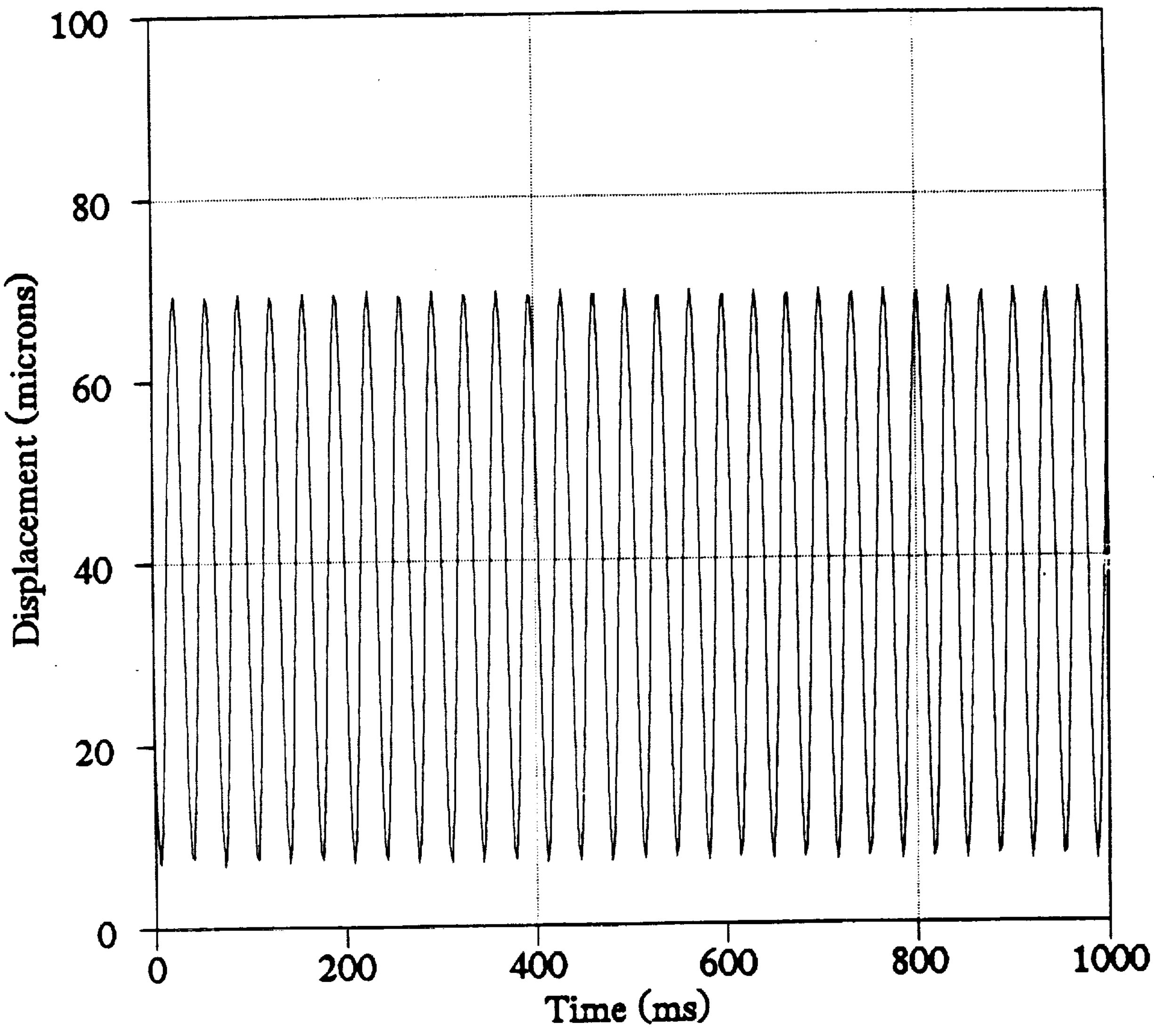

Figure 31. Sensor Response to 30 Hz Sinusoidal Waveform at Room Temperature.

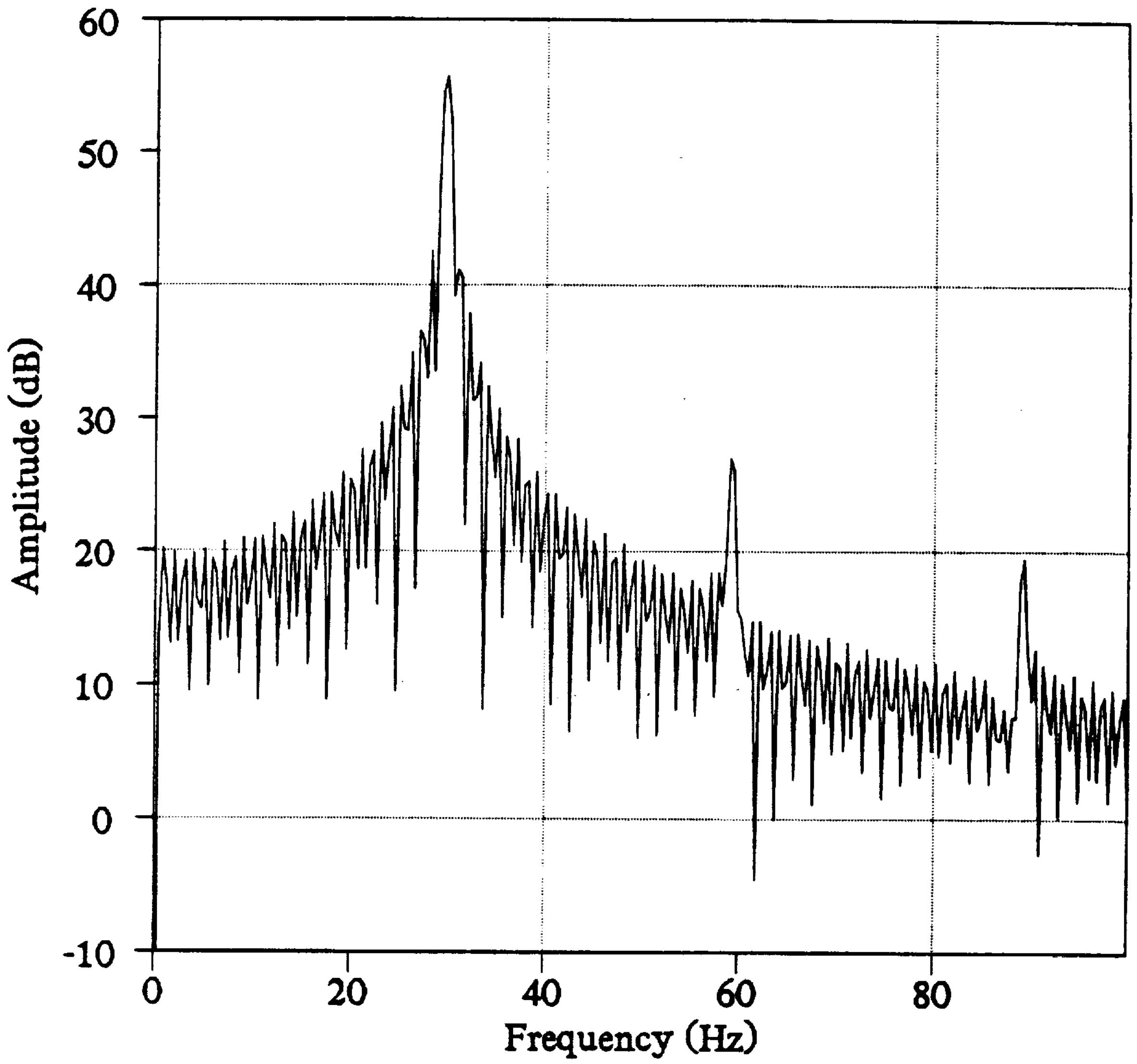

Figure 32. Fourier Transform of Figure 31.

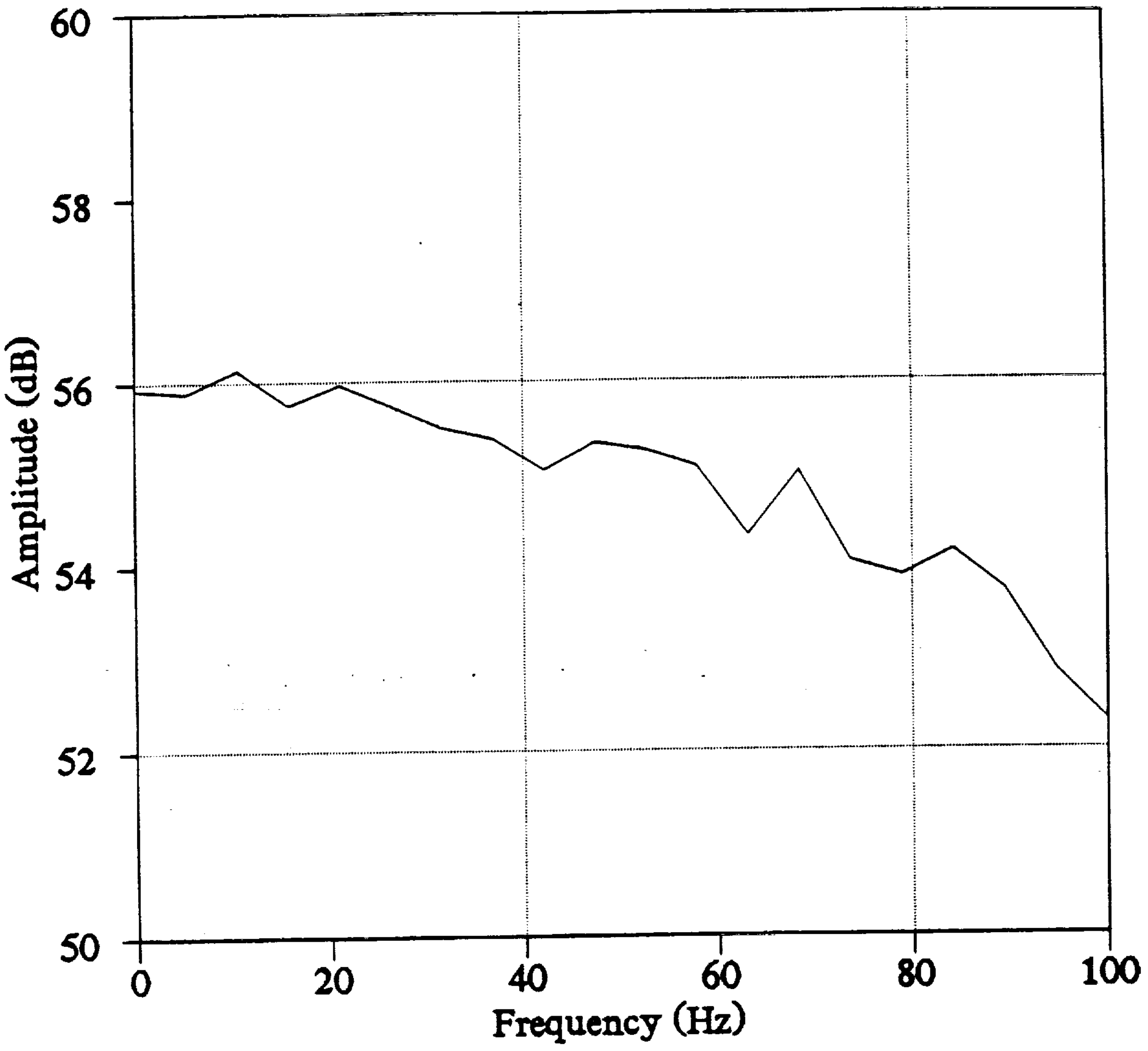

Figure 33. Frequency Response of Fiber Optic Strain Sensor at Room Temperature.

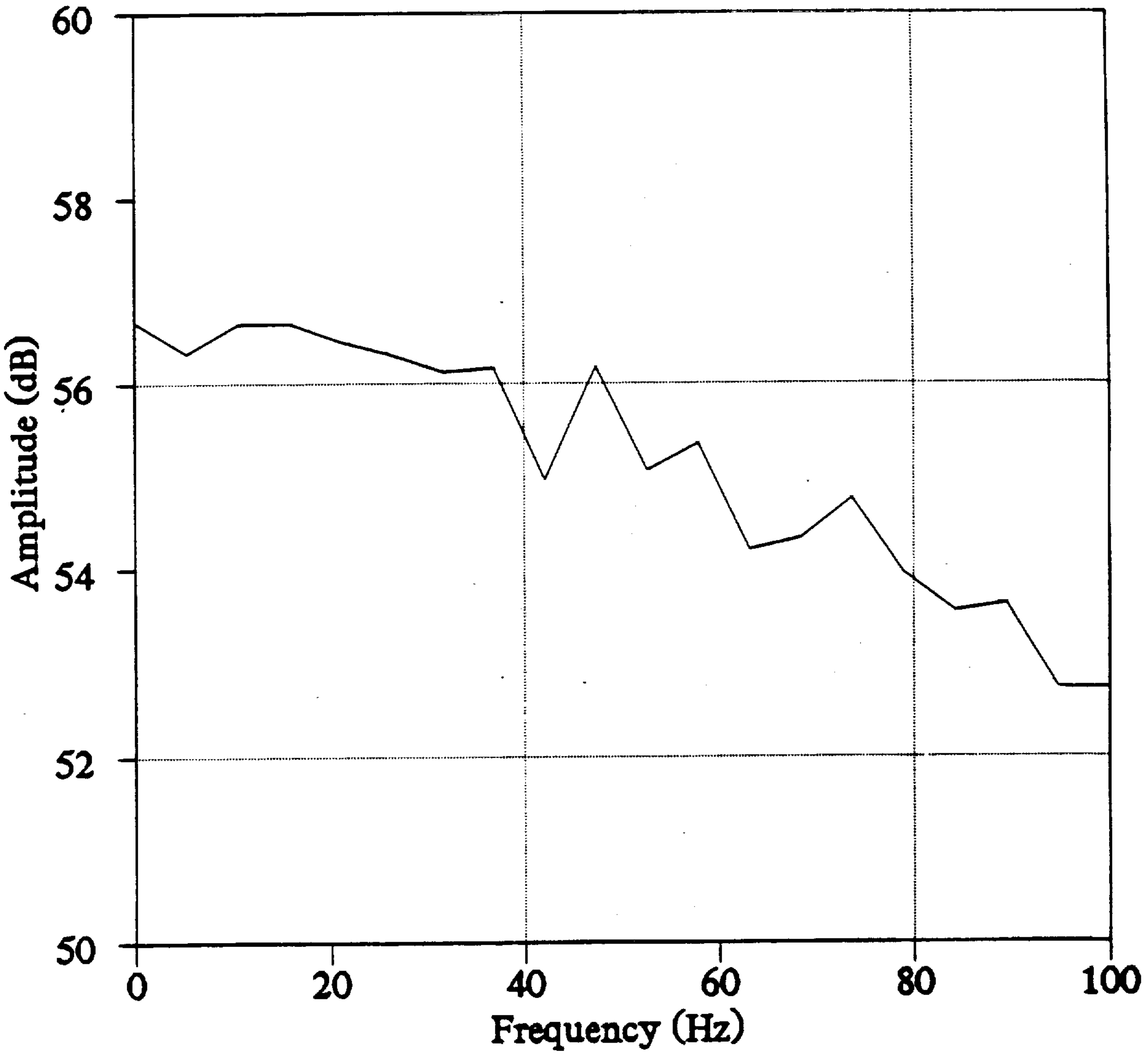

Figure 34. Frequency Response of Fiber Optic Strain Sensor at 905° C.

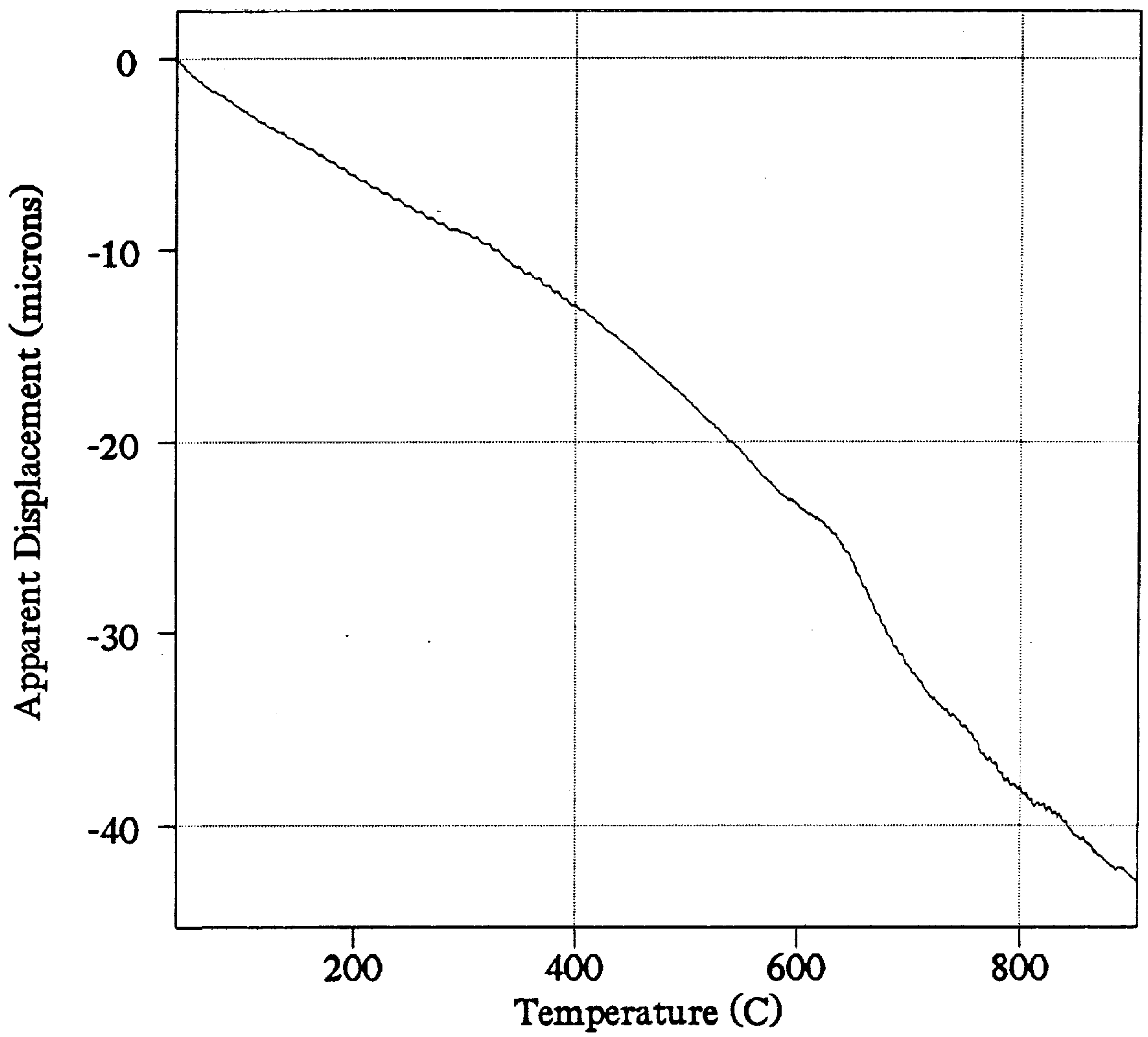

Figure 35. Apparent Displacement Due to Temperature.

# 4 Conclusions

A fiber optic point strain sensor consisting of a sensor head and a signal detection system was tested at 22°C and 905°C. Sensor heads were constructed using silica ($SiO_2$) or platinum tubes. Platinum wire, polished to a mirror finish, was inserted into one end of the tube. A gold coated optical fiber was inserted into the other end of the tube. This fiber transmitted optical energy to the mirror and carried the reflected signal back to the detection system. The sensor head was cemented to the substrate under test. Stress or strain experienced by the substrate was transferred to the sensor head as a change in gap width between the mirror and the fiber. In turn, changes in gap width affected the optical intensity that was reflected by the mirror and detected by the sensor system. The minimum strain sensitivity was measured to be 100 microinches/inch at 905°C. The maximum strain that could be measured with the point strain sensor was greater than $10^4$ microinches/inches, or > 1%, at all temperatures between 22°C and 905°C. Sensitivity was limited by drift in the LED's output intensity. An optical wheatstone bridge was incorporated into the detection system, but it was later removed because the signal-to-noise ratio of the detection system was greatly reduced.

Another way to reduce LED drift is to incorporate a reference detector that monitors the LED output directly. The detected reference signal is channeled through a feedback loop to the negative input on the operational amplifier that drives the LED. Two prerequisites required to obtain a high performance feedback loop are:

1. The gain and bandwidth of the feedback loop must be adjusted to prevent circuit oscillations, and

2. The output signals for the sensing and reference channels under zero strain condition must be identical.

Preliminary inspection of the two channels indicated different noise characteristics that made it difficult to discern the signal drift characteristics. Ultimately, the feedback effort was abandoned because the noise characteristics were dissimilar, and the remaining time on the project was devoted toward system tests without the wheatstone bridge.

The main disadvantage of this sensor system is that each sensor head can be used only one time. Once the sensor is cemented to a substrate, it can not be removed without damage to the sensor head. The high cost (> $300) per sensor head combined with a one-time use makes this sensor unattractive for frequent applications.

An ideal strain sensor would consist of a single, low cost, optical fiber. A short section of this fiber (~ 1 cm) is bonded to the substrate and serves as the sensor head. After each application, the fiber is cleaved near the bond and is immediately ready for the next application. This sensor head is easily generated and applied and could also be used for distributed sensing by bonding long fiber lengths ($\geq$ 10 m) to the substrate. Under these conditions, the strain may be determined at every point in the fiber. A design of this distributed sensor was presented to NASA Marshall under a seperate proposal.

# 5 User Guide

## 5.1 Introduction

The fiber optic Manager is a utility program which allows data to be taken from three input channels (Fiber Optic Sensor, Standard Sensor, and Thermocouple) for a specified amount of time. The data may then be analyzed and viewed in the form of either time graphs or Fourier transforms. The manager is written in standard Turbo-Vision format, so all window functions will perform exactly as in Turbo Pascal.

## 5.2 System Requirements

To function at full capacity, the fiber optic Manager software requires the following minimum system configuration:

| | | |
|---|---|---|
| Monitor | : | EGA/VGA |
| Processor | : | 286/16 MHz or faster |
| Memory | : | 2 megabytes, with at least 256K XMS or EMS memory and 590K free base memory. |
| Hard Disk | : | At least 320K free for programs, and enough additional space to accommodate a 30K per second data acquisition rate. |
| A/D Board | : | Designed for a Computer Boards, Inc., CIO-AD16 data board. Other Boards may not function properly with this program. |

## 5.3 Manager Installation, Setup and Commands

To install the Manager on a hard disk, type A: to transfer to the floppy drive and type Install. When asked to give a directory, simply type a name (NASA), not a path (c:\NASA). The Manager will be installed in the directory you indicate, with a batch file in the root directory to enable the program to be run at any time by typing FO.

Two commands are also available once the Manager has been installed. Type MC + from the directory containing the manager to convert to the version of the Manager which requires a math coprocessor. Type MC - to return to standard format that does not require a math coprocessor. You may also type FOCAL X to calibrate the Manager to a new sensor, with X being the letter designation of the sensor which has been installed. After installation, use the <TAB> key to cycle through the buttons and input lines in a pop-up window, or use the mouse. The following commands should be observed if the tab keys are employed. To access the menu bar, press either <F10> or <ALT> and the highlighted letter of the menu item you wish to select. You may then use the arrow keys to move around the menu panel and to move from one panel to the next. Several "hot-keys" are also available :

| | | | |
|---|---|---|---|
| | <F1> | : | Collect Data |
| <ALT> - | <F1> | : | Delete a Data Set |
| | <F2> | : | Perform FFT calculations |
| <CTRL> - | <F2> | : | Plot Graph of FFT function |
| <ALT> - | <F2> | : | Plot Time Graph |
| <ALT> - | <F3> | : | Close a text window |
| | <F4> | : | Display currently saved Graphs |
| | <F7> | : | Quick Help |
| <ALT> - | <F7> | : | Help |
| <ALT> - | X | : | Exit to DOS |

## 5.4 Fiber Optic Sensor Preparation

To prepare a fiber optic sensor for testing, place the sensor onto the test surface. Use three small patches of ceramic cement to attach the platinum mirror, the gold wire, and the guiding tube separately to the surface. Cotronics 904 Zirconia Adhesive (for use up to 2,200°C) is recommended for test surfaces with low expansion coefficients, but other adhesives can be recommended for test surfaces with higher expansion coefficients. Once the sensor has been mounted, connect the sensor to the detection system with standard 50/125 fiber optic cable. Additional gold-coated fiber may be purchased from Fiberguide Industries (Sterling, New Jersey) if the sensor filament is not long enough to extend out of the high temperature area. Once the sensor has been connected to the detector, turn the detector on and press the small red button on the front. Hold this button down while turning the calibration knob until the exact point at which the LED changes color is reached. Finally, type FOCAL X (X being the letter designation of the sensor) from the Manager directory on the computer to select the appropriate calibration file for the data acquisition routines.

## 5.5 Data Collection

To collect data, start the Manager by typing FO. When the menu screen appears, select COLLECT DATA as detailed in the introduction. You will first be asked to provide a name for the data set. The data set is the subdirectory in which all data collected at this time will be stored. When you have provided a name for the data set, you will be asked how long you wish to take data. This should be no longer than 120 seconds due to the available base memory. After the amount of time to gather data has been entered, the Manager will ask for confirmation before beginning data collection. This will allow time for any external setup which might be necessary. When the collection has been confirmed, the Manager will begin to collect data. After the given time expires, a window will appear which states SAVING DATA, followed by an estimated time to completion. This should take no longer than the time for which data were taken. The window will remain while the Manager transfers all data from memory to the hard disk. When this is completed, another window will appear which states INTERPRETING XXXX READINGS, with the Xs being the number of readings which were taken. Underneath this line will be a small box in which the Manager will display its progress until all data have been converted to microns, degrees, etc. This could take up to 30 minutes depending on the processor speed of the computer and the amount of data which has been taken. The data collection process may be aborted at any time prior to the collection stage by simply pressing <ESC>. If you should wish to delete a data set at a later time, simply select Delete Data Set from the Data menu.

## 5.6 Data Analysis

After a set of data has been collected, the Manager will allow you to perform FFTs and display FFT and time graphs to visualize the data. To perform an FFT, first select CREATE FFT FILES from the analysis menu. The Manager will then divide the data into sets of 400 and calculate the FFTs. The FFT calculation may take hours on a slow machine with a large set of data. After the FFTs have been performed, you may graph an FFT by selecting PLOT DYNAMIC STRAIN from the analysis menu. You will be asked to select a data set, and then you will be shown a list of FFT files. A file name beginning with FO is Fiber Optic Sensor Data, while a file name beginning with STD is Standard Sensor Data. A file name ending with a 1 indicates bin number 1. Once you have selected an FFT file, you will be transferred to a plotting program developed by J.M. Baden at GTRI. From this software, either press <F9> to plot the graph using default settings or press <F1> for help in changing the plot settings. Press <ALT>-X to exit the plotter and return to the Manager.

To plot a graph of data versus time, select PLOT STATIC STRAIN from the analysis menu. You will be asked to select a data set and an input channel (Fiber Optic Sensor, Standard Sensor, or Thermocouple). Every 400 points of data will be averaged to reduce noise. Plotting is accomplished through Mike Baden's software as described above.

## 6 References

[1] E. O. Rausch and P.B. Ruffin, "Fiber Optic Strain Sensing with RF interferometric techniques," SPIE Proc., Vol. 1170, pp. 440-450, 1989.

[2] W. W. Morey, J.R. Dunphy, and G. Meltz, "Multiplexing Fiber Bragg Grating Sensors," SPIE Proc., Vol. 1586, pp. 22-30, 1991.

# Appendix A. Fiber Optic Strain Sensor Survey

## A.1 Introduction

This appendix is the result of an extensive literature search on the topic of fiber optic strain sensors. Although numerous variations exist, the types of fiber optic strain sensors tend to fall into the following five general categories: amplitude sensors, interferometric sensors, polarimetric sensors, Bragg grating sensors, and Raman scattering sensors. Each of these categories is summarized below, with an analysis of the advantages and disadvantages of the particular strain sensor type. A complete bibliography is included at the end of this appendix.

## A.2 Amplitude Sensors

Compact amplitude strain sensors have a gap interrupting the fiber path. The sensor relies on the length of the interrupting gap and on the properties of the air-glass interface.[1] The gap is usually created by axially aligning the fiber and a reflector of some kind in a small diameter tube (see Figure A-1). The sensor is mounted such that the length of the air gap changes as the material stretches. As the length of the gap changes, the intensity at the output leg of the fiber optic coupler will change due to attenuation of the light travelling through the air gap. This property can be monitored, and calculations can be made to get the displacement and strain values.

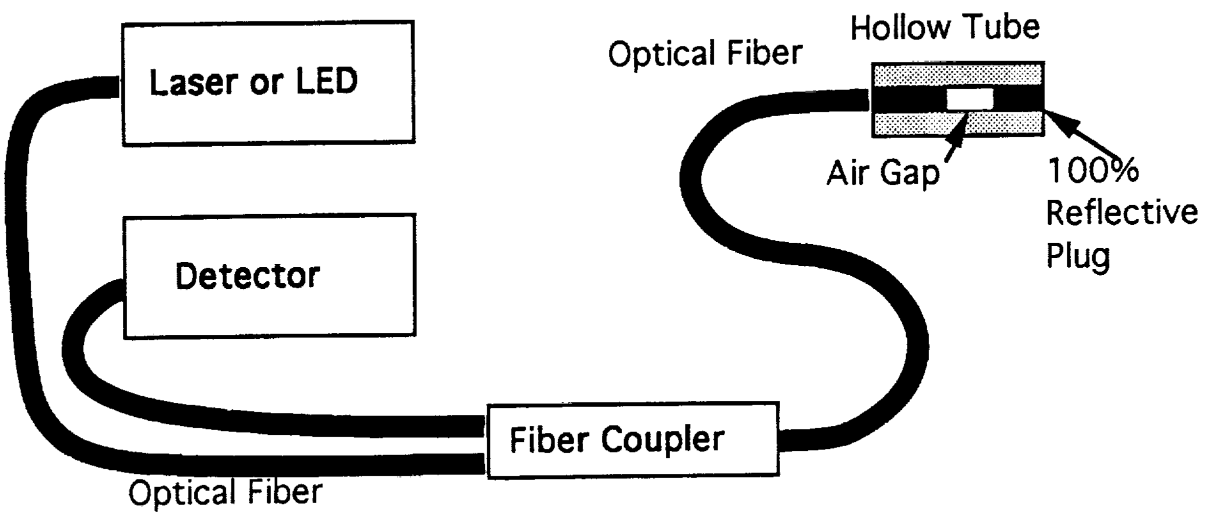

Figure A-1. Amplitude Fiber-Optic Strain Sensor.

The advantages of the amplitude sensor are twofold. First, it is relatively simple. No special fibers or construction techniques are required. Second, it is an extrinsic sensor. That is, the fiber is a conducting medium; it is not part of the sensor itself. This allows for remote testing of strain over small areas.

The disadvantage is that the amplitude sensor does not allow for continuous sensing along the entire length of the fiber.

## A.3 Interferometric Sensors

Interferometric sensors include the family of sensors which use the techniques of interferometry to measure sensor changes.[2,3] Interferometric sensors require two fiber optic signal paths. One path is used as a reference path, and the other is used as the sensing path. The strain sensor is located somewhere along the sensor path. As the strain increases, the path length of the signal will change. This results in a phase difference between the outputs of the reference and sensing fibers. The intensity output, $I_i$, at any interfering position will be

$$I_i = 2\sin\left(\frac{\Delta\phi_i}{2}\right)\sin\phi_i$$

where $\Delta\phi_i = \phi_{1i} - \phi_{2i}$, which is the differential phase delay at position i. This output may be projected onto a screen (see Figure A-2). An interference pattern will be seen as the light sources add constructively and destructively. A change in the strain will result in a change in the interference fringes. A fringe counter or other sensing circuit can count the fringes and calculate a value for the strain.

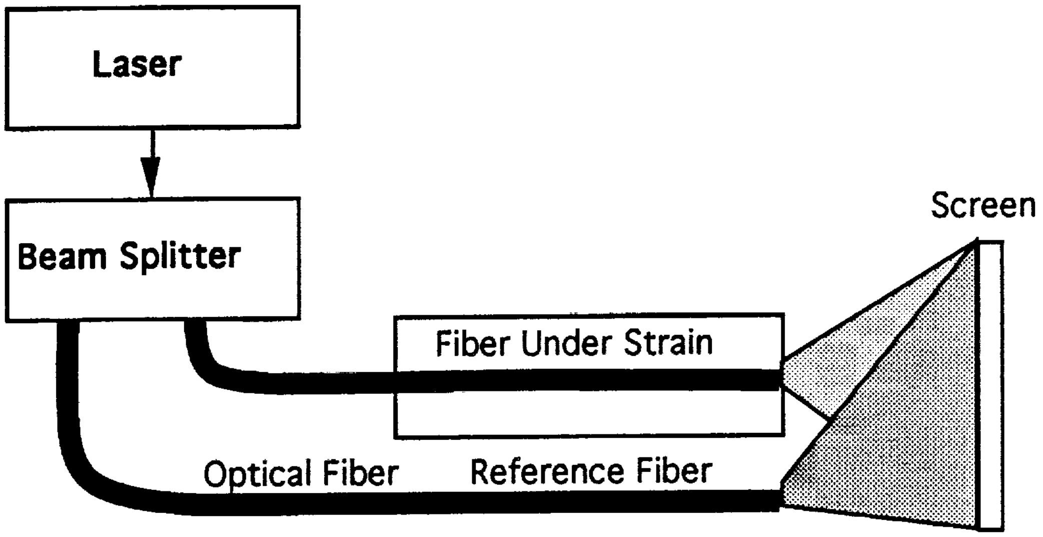

Figure A-2. Interferometric Fiber-Optic Strain Sensor.

The interferometric sensor can be very sensitive because the fringe pattern will be a function of the frequency spectrum of the light used.

The interferometric sensor requires two fiber paths. Also, additional circuitry will be required to count the fringes and convert the count to a strain value.

## A.4 Polarimetric Sensors

Polarimetric sensors use birefringence optical fibers and polarization techniques in order to set up two independent optical paths within one fiber. A section of the fiber is attached with a 45° splice to isolate it as the sensing region. This can either be in line with 45° splices on each end or at the end of the fiber with one 45° splice and a mirrored termination (see Figure A-3).

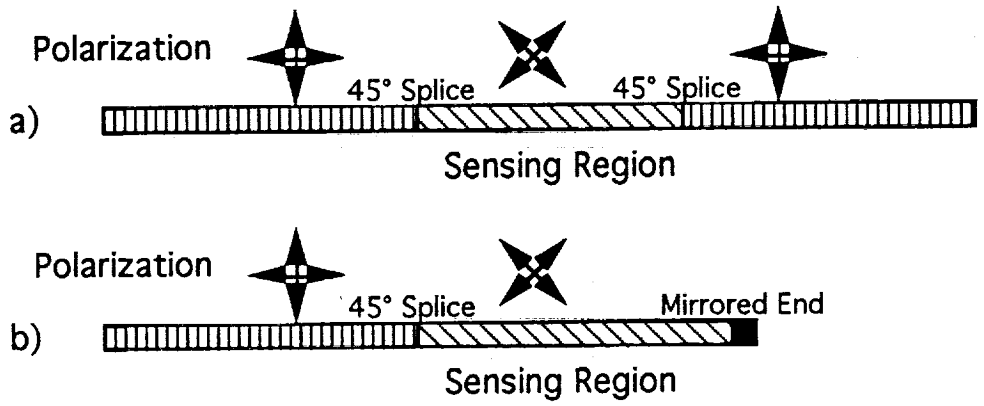

Figure A-3. Polarimetric Fiber-Optic Strain Sensing Fiber, (a) In Line Configuration and (b) End Configuration.[4]

The sensing region is firmly mounted to the specimen. The optical source and the detection optics are aligned with the axes of the birefringence fiber, and the two orthogonal paths are split using a polarizing beamsplitter (see Figure A-4).[5,6] As the sensing region is stretched, the polarization difference between the two paths will change. Interferometric techniques are then used to measure the indicated strain.

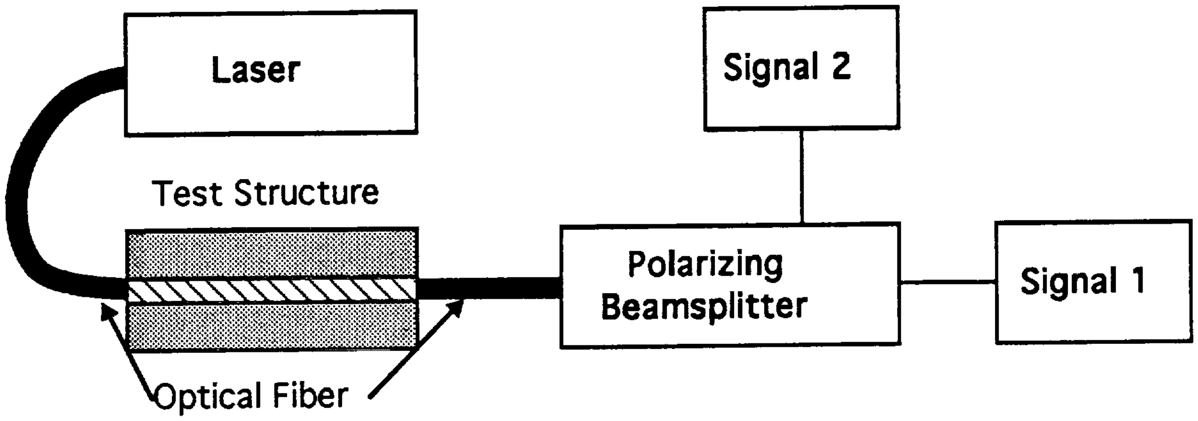

Figure A-4. Polarimetric Fiber-Optic Strain Sensor.

Polarimetric strain sensors have much better lead-in/out insensitivity than their interferometric counterparts. Sensor lengths of several meters can be constructed.

Polarimetric sensors are not as sensitive as interferometric sensors. Also, more equipment is required in order to align the optics with the fiber axes. Splicing is required for the 45° splices on each of the sensing sections.

## A.5 Bragg Grating Sensors

Bragg grating strain sensors provide a means for distributed strain sensing.[7] Gratings can be holographically generated at any position along the length of the fiber.[8,9] A short section of the core is doped with Germania and exposed from the side to an interference pattern of coherent ultraviolet light. This interference pattern sets up a permanent grating in the fiber. Each grating is designed to reflect certain frequencies. The grating wavelength is sensitive to changes in temperature and strain. The shift in the Bragg wavelength can be observed and related to these quantities. The fractional shift in the Bragg wavelength, $\Delta\lambda_b/\lambda_b$, is given by

$$\frac{\Delta\lambda_b}{\lambda_b} = (1 - p_e)\varepsilon$$

where $\varepsilon$ is the longitudinal strain, and $p_e$ is the effective photoelastic constant. Thus, the strain between the optical source and any particular grating can be measured, and the strain between any two gratings can be calculated.[10]

The gratings are easy to generate and require no mechanical alteration of the fiber or splicing. Gratings can be generated anywhere along the length of the fiber, thus providing a very flexible sensor system (see Figure A-5).

Bragg gratings are sensitive to both temperature and strain, so it may be difficult to isolate only one of the variables.

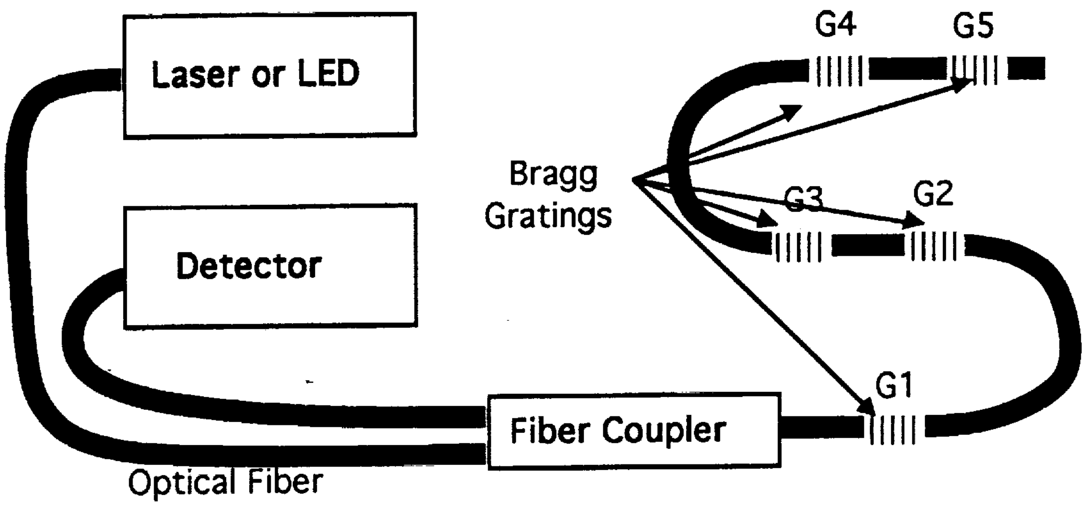

Figure A-5. Bragg Grating Fiber-Optic Strain Sensor.

## A.6 Raman Sensors

Raman strain sensors use the ratio of the anti-Stokes and Stokes backscatter signals to get a measure of the temperature or strain along the entire length of the fiber. The local stress can be estimated nondestructively by measurements of the Raman backscattering spectrum.[11,12] Brillouin scattering can similarly be used to measure temperature or strain.[13]

Local stress or strain measurements can be made along the entire length of the fiber (see Figure A-6). Raman scattering can also be used as a distributed temperature sensor.

Raman scattering techniques require a significant amount of equipment, including filters, amplifiers, and delay units, in order to make the measurements.

Each of the strain measuring methods has its own set of advantages and disadvantages. The application will determine which method is best. For simple localized strain measurements, the amplitude sensor should provide adequate performance without requiring excessive equipment or preparation.

Interferometric sensors provide more sensitivity, but they are more susceptible to vibration and require two signal paths.

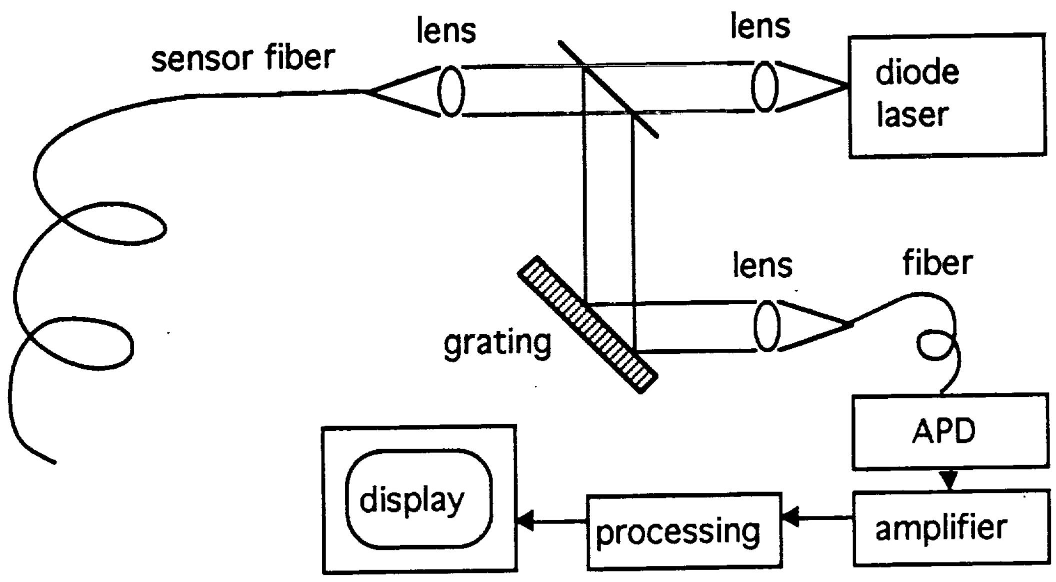

Figure A-6. Raman Fiber-Optic Strain/Temperature Sensor.[14]

## A.7 Conclusions

Polarimetric sensors provide a partial solution to the two-path problem of interferometric sensors by using berefringent fibers and orthogonally polarized paths. Polarimetric sensors also allow for sensors of various lengths, but the construction and alignment challenges need to be considered.

Bragg gratings are easily generated in optical fibers and provide a means for distributed strain sensing. This sensor is sensitive to temperature as well as strain. Separation of the signals may be a problem.

The Raman based sensors are very effective as completely distributed temperature sensors. Strain can be related to the change in the Raman spectrum. This operation requires more equipment than most other methods. Brillouin scattering can be used to measure temperature along the fiber and to make even more accurate strain measurements.

Significant work has been done in the area of fiber-optic strain sensing. The above methods represent the majority of work being done, but there are several variations and specifics that were not included in this appendix. For additional information, see References [15] through [23].

## A.8 Bibliography

1. R. S. Rogowski, J. S. Heyman, and M. S. Holben, Jr., "An Amplitude Modulated Laser System for Distance and Displacement Measurement," Proceedings of the SPIE, Vol. 663, 1986.

2. E. O. Rausch and P. B. Ruffin, "Fiber Optic Strain Sensing with RF Interferometric Techniques," SPIE Vol. 1170, Fiber Optic Smart Structures and Skins II, pp. 440-450, 1989.

3. J. S. Sirkis and C. E. Taylor, "Interferometric-Fiber-Optic Strain Sensor," Experimental Mechanics, pp. 170-176, June 1988.

4. W. D. Hogg, R. D. Turner, and R. M. Measures, "Polarimetric Fiber Optic Structural Strain Sensor Characterization," SPIE Vol. 1170, Fiber Optic Smart Structures and Skins II, pp. 542-550, 1989.

5. W. J. Bock and T. R. Wolinski, "Temperature-compensated Fiber-optic Strain Sensor Based on Polarization-rotated Reflection," SPIE Vol. 1370, Fiber Optic Smart Structures and Skins III, pp. 189-196, 1990.

6. S. Chen and I. P. Giles, "Optical Coherence Domain Polarimetry: Intensity and Interferometric Type for Quasi-distributed Optical Fibre Sensors," SPIE Vol. 1370, Fiber Optic Smart Structures and Skins III, pp. 217-225, 1990.

7. W. W. Morey, "Distributed Fiber Grating Sensors," Proceedings, 7th Optical Fiber Sensors Conference, December 2-6, 1990, Sydney, New South Wales, 1990.

8. W. W. Morey, G. Meltz, and W. H. Glenn, "Holographically Generated Gratings in Optical Fibers," Optics and Photonics News, Vol. 1, No. 7, pp. 14-16, July 1990.

9. G. Meltz, W. W. Morey, and W. H. Glenn, "Formation of Bragg Gratings in Optical Fibers by a Transverse Holographic Method," Optics Letters, Vol. 14, No. 15, pp. 823-825, August 1, 1989.

10. W. W. Morey, G. Meltz, and W. H. Glenn, "Fiber Optic Bragg Grating Sensors," SPIE Vol. 1169, Fiber Optic and Laser Sensors VII, pp. 98-107, 1989.

11. T. Valis, R. D. Turner, and R. M. Measures, "Distributed Fiber Optic Sensing Based on Counterpropagating Waves," Applied Optics, Vol. 28, No. 11, pp. 1984-1990, 1 June, 1989.

12. H. Sakata, G. Dresselhaus, and J. S. Dresselhaus, "Effect of Uniaxial Stress on the Raman Spectra of Graphite Fibers," Journal of Applied Physics, Vol. 64, No. 8, pp. 2769-2772, 15 April, 1988.

13. B. G. Gorshkov, I. E, Gorbatov, Y. K. Danileiko, and A. V. Sidorin, "Luminescence, Scattering, and Absorption of Light in Quartz Optical Fibers and Prospective Use of These Properties in Distributed Waveguide Sensors," Soviet Journal of Quantum Eletronics, Vol. 20, No. 3, pp. 283-288, March 1990.

14. J. P. Dakin, D. J. Pratt, G. W. Bibby, and J. N. Ross, "Distributed Optical Fiber Raman Temperature Sensor Using a Semiconductor Light Source and Detector," Electronics Letters, Vol. 21, No. 13, pp. 569-570, 20 June, 1985.

15. J. K. A. Everard and R. Thomas, "Distributed Optical Fiber Temperature Sensor Using Spread-spectrum Techniques," Electronics Letters, Vol. 25, No. 2, pp. 140-142, 19 January, 1989.

16. Z. V. Nesterova, I. V. Aleksandrov, V. V. Zhakhov, and L. G. Karpov, "Induced Raman Scatter of Light in Fiber Lightguides," Fizika i Khimiya Stekla, Vol. 12, No. 4, pp. 443-447, July-August, 1986.

17. J. R. Dunphy, G. Meltz, R. M. Elkow, "Distributed Strain Sensing with a Twin-Core Fiber Optic Sensor," ISA Transactions, Vol. 26, No. 1, pp. 7-10, 1987.

18. C. F. Fan and S. L. Hsu, "Compressive Behavior of the Reinforcement Fiber and the Buildup of Thermal Residual Stress," Journal of Polymer Science: Part B: Polymer Physics, Vol. 27, pp. 337-353, 1989.

19. K. A. Murphy, C. E. Koob, A. J. Plante, S. Desu, and R. O Claus, "High Temperature Sensing Applications of Silica and Sapphire Optical Fibers," SPIE Vol. 1370, Fiber Optic Smart Structures and Skins III, pp. 169-178, 1990.

20. Z. J. Lu and F. A. Blaha, "A Two-mode Fiber Optic Strain Sensor System for Smart Structures and Skins," SPIE Vol. 1370, Fiber Optic Smart Structures and Skins III, pp. 180-188, 1990.

21. Y. Murakami, K. Hoguchi, N. Uesugi, K. Ishihara, and Y. Negishi, "Optical Fiber Loss Increase in the Infrared Wavelength Region due to Hydrogen Molecules Induced by Electrolysis," The Transactions of the IECE of Japan, Vol. E68, No. 2, pp. 65-70, 1984.

22. S. G. Kosinski and A. J. Bruce, "Vibrational Spectra of Oxide Doped Fluorozirconate Glasses," Material Research Society Symposium Proceedings, Vol. 88, pp. 163-168, 1987.

23. R. M. Measures, "Fiber Optics Smart Structures Program at UTIAS," SPIE Vol. 1370, Fiber Optic Smart Structures and Skins III, pp. 92-108, 1990.

24. Z. J. Lu and F. A. Blaha, "A Fiber Optic Strain and Impact Sensor System for Composite Materials," SPIE Vol. 1170, Fiber Optic Smart Structures and Skins II, pp. 239-248, 1989.

25. R. S. Rogowski, M. S. Holben, Jr., J. S. Heyman, D. W. DeHart, and S. Margulies, "Thermal Effects on Fiber Optic Strain Sensors Embedded in Graphite-epoxy Composites," SPIE Vol. 1170, Fiber Optic Smart Structures and Skins II, pp. 435-439, 1989.

## Appendix B. Fiber Optic Manager Software

```
{$A+,B-,D+,E+,F-,G-,I+,L+,N-,O-,R+,S+,V+,X+}
{$M 55520,100000,655360}
program FOMAN;

uses
  App,       Crt,       DemoCmds, DemoHelp, Dialogs, Dos,
  Drivers,   FViewer,   Gadgets,  HelpFile, Memory,  Menus,
  Objects,   SpawnEMS,  FDlg,     Tp4Tclin, Tp4D16,  Views;

const
  BINSIZE      = 400   ;  { -- Size of FFT Bins                      -- }
  BOARD_NUM    = 0     ;  { -- A/D BOARD     Number                  -- }
  CMSENSOR     = 8001  ;  { -- Command Code                          -- }
  CMDOFFT      = 8002  ;  { --    ""      ""                         -- }
  CMSHOWFFT    = 8003  ;  { --    ""      ""                         -- }
  CMKILL       = 8004  ;  { --    ""      ""                         -- }
  CMGRAPH      = 8005  ;  { --    ""      ""                         -- }
  CMSHOW       = 8006  ;  { --    ""      ""                         -- }
  CMHELPME     = 8007  ;  { --    ""      ""                         -- }
  CMQHELP      = 8010  ;  { --    ""      ""                         -- }
  FOPTICHAN    = 4     ;  { -- Fiber Optic Strain Gauge Channel-- }
  STDSTCHAN    = 7     ;  { -- Standard Strain Gauge Channel    -- }
  TEMPRCHAN    = 6     ;  { -- ThermoCouple Channel              -- }
  TCTYPE       = 's'   ;  { -- ThermoCouple Type                 -- }
  TEMPGAIN     = 584.5;   { -- ThermoCouple Gain                 -- }
  TEMPOFFSET   = 2     ;  { -- ThermoCouple Offset               -- }
  STDGAIN      = 1     ;  { -- Standard Gauge Gain               -- }
  STDOFFSET    = 0     ;  { -- Standard Gauge Offset             -- }

Type

  FLOAT = real;

  { TFOMAN }
  PFOMAN = ^TFOMAN;
  TFOMAN = object (TApplication)
    Clock : PClockView;
    Heap  : PHeapView ;
    constructor Init  ;
    procedure   GetEvent      (var Event  : TEvent); virtual;
    function    GetPalette    : PPalette           ; virtual;
    procedure   HandleEvent   (var Event  : TEvent); virtual;
    procedure   Idle          ; virtual  ;
    procedure   InitMenuBar   ; virtual  ;
    procedure   InitStatusLine; virtual  ;
    procedure   ViewFile      (FILENAME   : PathStr);
  end;  {TFOMAN}

  PDIALOG = ^TDialog;
  TDemoDialog =  object (TDialog)
    end;  {TDemoDialog}

var
  B              : PView   ;     { -- Turbo Vision Constructor    -- }
  COMMANDCOM     : string  ;     { -- Location of Command.Com     -- }
  CONTROL        : word    ;     { -- DIALOG box Commands         -- }
  DATAVAL        : integer;      { -- D/A acquisition variable    -- }
  DIALOG         : PDIALOG;      { -- Represents DIALOG Box       -- }
  DIRNAME        : PathStr;      { -- Data Set directory          -- }
  DNAME          : PathStr;      { --  ""     ""     ""           -- }
  ERR_CODE       : integer;      { -- Tracks A/D BOARD errors     -- }
  F              : text    ;     { -- Text File                   -- }
  FDIALOG        : PFileDialog;  { -- File Selection Box          -- }
  FILENAME       : PathStr;      { -- Used to store File Names    -- }
  FOCAL          : PathStr;      { -- Sensor Calibration Curve    -- }
  INNER          : longint;      { -- Counts in FOR loops         -- }
  MAIN           : string  ;     { -- Stores Manager Directory    -- }
  MDIR           : PathStr;      { -- Main Program Directory      -- }
  OUTER          : longint;      { -- Counts in FOR loops         -- }
  R              : TRect   ;     { -- Turbo Vision Constructor    -- }
  SUCCESS        : boolean;      { -- Operation Success           -- }
  TEMP           : string  ;     { -- Temporary string variables-- }
  VOLTS          : FLOAT   ;     { -- A/D input voltage           -- }
  W              : PWindow;      { -- Turbo-Vision constructor    -- }

{______________________________________________________________________}
{______________________________________________________________________}
{ TFOMAN }
constructor TFOMAN.Init;
  { -- Initializes and Builds Manager                                -- }

  begin {TFOMAN.Init}
    TApplication.Init;
    RegisterObjects  ;
    RegisterViews    ;
    RegisterMenus    ;
    RegisterDialogs  ;
    RegisterApp      ;
    RegisterFViewer  ;

    { -- Display current time in upper right corner                  -- }
    GetExtent (R)      ;
    R.A.X    := R.B.X - 9;
    R.B.Y    := R.A.Y + 1;
    Clock    := New (PClockView, Init (R));
    Insert     (Clock);

    { -- Ensure appropriate directory                                -- }
    ChDir    (MAIN);

    { -- Load Background file                                    -- }
    ViewFile ('Startup');

  end;  {TFOMAN.Init}

{_______________________________________________________________}
{_______________________________________________________________}

function TFOMAN.GetPalette: PPalette;
  { -- Loads color palette for Manager                          -- }

  const
    CNewColor      = CColor      + CHelpColor      ;
    CNewBlackWhite = CBlackWhite + CHelpBlackWhite;
    CNewMonochrome = CMonochrome + CHelpMonochrome;

    P: array [apColor..apMonochrome] of string [Length
      (CNewColor)] = (CNewColor, CNewBlackWhite, CNewMonochrome);

  begin {function TFOMAN.GetPalette}
    GetPalette := @P [AppPalette];
  end;  {function TFOMAN.GetPalette}

{_______________________________________________________________}
{_______________________________________________________________}

procedure TFOMAN.InitMenuBar;
  { -- Initializes and creates Menu Bar                         -- }

  begin {procedure TFOMAN.InitMenuBar}
    GetExtent (R);
    R.B.Y   := R.A.Y + 1;
    MenuBar := New (PMenuBar, Init  (R, NewMenu (
      NewSubMenu ('~S~oftware         ',  hcSystem,     NewMenu (
        NewItem  ('~A~bout Software', '', kbNoKey,      cmAbout,
                                          hcNoContext,
        NewItem  ( 'E~x~it to DOS',      'Alt-X',      kbAltX,
                                          cmQuit,       hcFExit,
      NIL))),
      NewSubMenu ('~D~ata               ',  hcSystem, NewMenu (
        NewItem  ('~C~ollect Data      ',  'F1',     kbF1,
                                          CMSENSOR, hcNoContext,
        NewItem  ('~D~elete  Data Set',  'Alt-F1',  kbAltF1,
                                          CMKILL,   hcNoContext,
      NIL))),
      NewSubMenu ('~A~nalysis           ',  hcNoContext, NewMenu (
        NewItem  ('~C~reate FFT Files   ','F2',        kbF2,
                                                CMDOFFT,   hcNoContext,
        NewItem   ('Plot ~D~ynamic Strain','Ctrl-F2',  kbCtrlF2,
                                                CMSHOWFFT, hcNoContext,
        NewItem   ('Plot ~S~tatic  Strain','Alt-F2',   kbAltF2,
                                                CMGRAPH,   hcNoContext,
        NewLine   (
        NewItem   ('Display ~P~ictures   ','F4',    kbF4,
                                                 CMSHOW, hcNoContext,
      NIL)))))),
      NewSubMenu ('~H~elp', hcNoContext, NewMenu (
        NewItem   ('~H~elp',         'Alt-F7',  kbAltF7,
                                         CMHELPME, hcNoContext,
        NewItem   ('~Q~uick help', 'F7',       kbF7,
                                         CMQHELP,  hcNoContext,
        NIL))),
    NIL)))))));
  end;  {procedure TFOMAN.InitMenuBar}

{______________________________________________________________}
{______________________________________________________________}

procedure TFOMAN.InitStatusLine;
  { -- Initializes and creates Status Line                    -- }

  begin {procedure TFOMAN.InitStatusLine}
    GetExtent (R);
    R.A.Y := R.B.Y - 1;
    StatusLine := New (PStatusLine, Init (R,
      NewStatusDef   (0, $FFFF,
        NewStatusKey ('~F1~ Collect Data ',   kbF1,     CMSENSOR,
        NewStatusKey ('~F7~ Quick Help ',     kbF7,     CMQHELP,
        NewStatusKey ('~Alt-F3~ Close Window',kbAltF3,  CMCLOSE,
        NewStatusKey ('~Alt-X~  Exit Manager',kbAltX,   CMQUIT,
        NewStatusKey ('',                     kbAltF1,  CMKILL,
        NewStatusKey ('',                     kbF2,     CMDOFFT,
        NewStatusKey ('',                     kbAltF2,  CMGRAPH,
        NewStatusKey ('',                     kbCtrlF2,CMSHOWFFT,
        NewStatusKey ('',                     kbF4,     CMSHOW,
        NewStatusKey ('',                     kbAltF7,  CMHELPME,
        NewStatusKey ('',                     kbF10,    CMMENU,
      NIL))))))))))),
    NIL)));
  end;  {procedure TFOMAN.InitStatusLine}

{______________________________________________________________}
{______________________________________________________________}

procedure TFOMAN.ViewFile (FILENAME : PathStr);
  { -- Opens a window and displays FILENAME                   -- }
  begin
    W            := New (PFileWindow, Init (FILENAME));
    W^.HelpCtx := hcViewer;
    if ValidView (W) <> NIL then
      DeskTop^.Insert (W);
  end;  {procedure TFOMAN.ViewFile}

{_______________________________________________________________}
{_______________________________________________________________}

procedure TFOMAN.Idle;
  { -- Monitors DeskTop while idle                           -- }

  begin {procedure TFOMAN.Idle}
    TApplication.Idle;
    Clock^.Update;
  end;  {procedure TFOMAN.Idle}

{_______________________________________________________________}
{_______________________________________________________________}

procedure TFOMAN.GetEvent (var Event: TEvent);
  { -- Discerns what event has taken place                   -- }

  begin {TFOMAN.GetEvent}
    TApplication.GetEvent (Event);
    case   Event.What of
      evMouseDown:
        if Event.Buttons <> 1 then
           Event.What := evNothing;
      end;  {case}
  end;  {TFOMAN.GetEvent}

{_______________________________________________________________}
{_______________________________________________________________}

procedure TFOMAN.HandleEvent (var Event: TEvent);
  { -- Runs procedure linked to event which has taken place  -- }

{_______________________________________________________________}

  procedure About;
    { -- Displays programmer development information         -- }

    begin {procedure About}
      R.Assign (0, 0, 40, 18);
      DIALOG := New (PDIALOG, Init (R, 'About'));
      with DIALOG^ do
```

```
        begin {with}
          Options := Options or ofCentered;
          R.Grow  (-1   , -1);
          Dec     (R.B.Y,  3);
          Insert  (New (PStaticText, Init (R,
            #13 +
            ^C'Fiber Optic Manager'#13 +
            #13 +
            ^C'programming by'#13 +
            #13 +
            ^C'Kevin E. Murphy and Joel Webber'#13 +
            #13 +
            ^C'under supervision of'#13 +
            ^C'Dr. E. O. Rausch, PhD.'#13 +
            #13 +
            ^C'Georgia Tech Research Institute')));
          R.Assign (15, 15, 25, 17);
          Insert   (New (PButton, Init (R, 'O~K', CMOK,
                         bfDefault)));
        end;  {with}
      if ValidView (DIALOG) <> NIL then
        begin {if}
          DeskTop^.ExecView (DIALOG);
          Dispose (DIALOG, Done);
        end;  {if}
    end;  {procedure About}

{_______________________________________________________________}

procedure ShutDown (Video : boolean);
  { -- Shuts Down Manager in preparation for DOS routines     -- }

  begin {procedure ShutDown}
    if Video then
      DoneVideo;
    DoneSysError;
    DoneEvents;
    DoneMemory;
    SetMemTop (HeapPtr);
  end;  {procedure ShutDown}

{_______________________________________________________________}

procedure ReInState (Video : boolean);
  { -- ReInstates Manager after DOS routines                  -- }

  begin {procedure ReInState}
    SwapVectors;
    SetMemTop (HeapEnd);
    InitMemory;
    InitEvents;
    InitSysError;
    if Video then
      InitVideo;
    Redraw;
  end;  {procedure ReInState}

{_______________________________________________________________}
{_______________________________________________________________}

procedure Sensor;
{ -- Gathers, Interprets, and Sorts Data into disk files      -- }

  var
    Code      : integer    ;  { -- Used in VAR operations      -- }
    Limit     : integer    ;  { -- Maximum Time to take data   -- }
    Number    : longint    ;  { -- Number of readings taken    -- }
    Point     : longint    ;  { -- Specific data point         -- }
    SText     : PStaticText;  { -- TurboVision constructor     -- }
    Time      : longint    ;  { -- Time to take Data           -- }

 { ------------------------------------------------------------ }

  procedure Initialize;
  { -- Setup I/O BOARD_NUM                                    -- }

    const
      Base_addr = $300;
      DMA_level = 1;
      Int_level = 5;

    begin {procedure Initialize}
      D16_Init (BOARD_NUM, Base_addr, Int_level, DMA_level,
                ERR_CODE);
    end;  {procedure Initialize}

 { ------------------------------------------------------------}

    procedure Take_Data (Time       : integer;
                     var Number     : longint);
      { -- Take Readings from sensors for TIME seconds        -- }

      const
        Samples = 32760;

      type
        DataPtr = ^DataRec;
        DataRec  = record
      Data : array [1..Samples] of integer;
      Next : DataPtr;
    end;

  var
    DataFile   : text    ;  { -- File to write to              -- }
    DataHead   : DataPtr;   { -- Head of Dynamic variable      -- }
    Finish     : longint;   { -- Leftover Samples after
                               -- complete Sets                 -- }
    Original   : pointer;   { -- Marks original Heap           -- }
    Present    : DataPtr;   { -- Dynamic variable position -- }
    Rate       : real    ;  { -- Rate of Data Acquisition      -- }
    RecSize    : longint;   { -- Dynamic record size           -- }
    Sets       : longint;   { -- # Complete Sets of Samples-- }

{ ------------------------------------------------------------ }

  procedure Scan   (SCount     : integer;
                var SRate      : real    ;
                var Data_Pntr  : integer);
  { -- Scan input channels from ChanLo to ChanHi at SRate--
    -- samples per second until Scount samples are taken -- }

    const
      ChanHi   = STDSTCHAN; { -- First input channel       -- }
      ChanLo   = FOPTICHAN; { -- Last  input channel       -- }

    var
     c1dat, c2dat : word;  { -- A/D Setup variables        -- }

    begin
      d16_set_mux (BOARD_NUM, chanlo, chanhi, err_code);
      d16_calc_timer_values (SRate, c1dat, c2dat);
      d16_set_pit_ratio (BOARD_NUM, c1dat, c2dat, err_code);
      d16_ain_direct (OARD_NUM, SCount, 1, data_pntr,
                      err_code);
    end;

{ ------------------------------------------------------------}

  begin {procedure Take_Data}

    { -- Inform User of new operation                        -- }
    R.Assign (10, 6, 55, 10);
    DIALOG   := New (PDIALOG, Init (R, ''));
    with DIALOG^ do
      begin {with}
        Options  := Options or ofCentered;
        R.Grow   (-1    ,    -1);
    Dec      (R.B.Y,    3);
    R.Assign (5, 1, 40, 3);
    Insert   (New (PStaticText, Init (R,'Taking Data')));
  end; {with}
DeskTop^.Insert (DIALOG);

{ -- Initialize Data partition variables                 -- }
Rate    := 1600.0;
Number  := round (Time * Rate);
Sets    := trunc (Number / Samples);
Finish  := Sets * Samples;
Finish  := Number - Finish;

{ -- Create Dynamic variable for Data storage            -- }
Mark       (Original);
RecSize := SizeOf(DataRec);
GetMem     (DataHead, RecSize);
Present := DataHead;

{ -- Read all sets of Data into dynamic memory           -- }
for OUTER := 1 to Sets do
  begin
    Scan      (SAMPLES, Rate, Present^.Data[1]);
    GetMem    (Present^.Next, RecSize);
    Present := Present^.Next;
  end;

{ -- Read any remaining Data into shortened array        -- }
if Finish > 0 then
  Scan (Finish, Rate, Present^.Data[1]);

Dispose (DIALOG, Done);

{ -- Inform User of saving operation                     -- }
R.Assign (10, 6, 55, 10);
DIALOG   := New (PDIALOG, Init (R, ''));
with DIALOG^ do
  begin {with}
    Options  := Options or ofCentered;
    R.Grow   (-1   ,   -1);
    Dec      (R.B.Y,    3);
    R.Assign (5, 1, 40, 3);
    OUTER  := round(Time / 1.3333333333333);
    Str      (OUTER, TEMP);
    Insert   (New (PStaticText, Init (R,
                  'Saving Data : ' +
                   TEMP + ' seconds')));
  end; {with}
DeskTop^.Insert (DIALOG);
```

```
    { -- Transfer Data from dynamic memory to hard disk  -- }
    assign    (DataFile, MDIR + 'DATA\RawData.Pre');
    rewrite   (DataFile);
    Present   := DataHead;
    for OUTER := 1 to Sets do
      begin
        for INNER := 1 to Samples do
          writeln (DataFile, Present^.Data[INNER]);
        Present := Present^.Next;
      end;
    for OUTER := 1 to Finish do
      writeln (DataFile, Present^.Data[OUTER]);

    { -- Close window and data file                      -- }
    close   (DataFile);
    Dispose (DIALOG, Done);

    Release (Original);
    Number := round (Number / 4);

  end; {procedure Take_Data}

{ . . . . . . . . . . . . . . . . . . . . .}

  procedure Interpret_Data (DIRNAME : Pathstr;
                            Number  : longint );
  { -- Converts Raw Data to meaningful values             -- }

    type
      { -- Calibration Curve                              -- }
      Caltype = array [0..90] of FLOAT;

      DataRec = record                  { -- Window record  -- }
          Reading : string[10];
        end;

    var
      CalCurve  : CalType; { -- Sensor Calibration Curve    -- }
      BxData    : DataRec; { -- Window update variable      -- }
      FOFile    : text   ; { -- Fiber Optic Sensor Data File-- }
      InFile    : text   ; { -- File from which to read     -- }
      Microns   : FLOAT  ; { -- Microns Sensor expansion    -- }
      NumStr    : String ; { -- Interpretation Time         -- }
      StdFile   : text   ; { -- Standard Sensor Data File   -- }
      Total     : integer; { -- Points in calibration curve -- }
      TmpFile   : text   ; { -- Temperature Data File       -- }
      Volt      : FLOAT  ; { -- Used in Raw Data conversion -- }
```

```
begin  {procedure Interpret_Data}

  { -- Read Calibration Curve from Disk                      -- }
  assign   (F, FOCAL);
  reset    (F);
  readln   (F, Total);
  readln   (F, Total);
  for OUTER := 1 to Total do
    readln (F, CalCurve[OUTER]);
 close     (F);
 CalCurve[0]          := 0;
 CalCurve[OUTER + 1] := 11;

 { -- Create monitor window for operation                    -- }
 Str      (Number, NumStr);
 NumStr   := 'Interpreting ' + NumStr + ' Readings';
 R.Assign  (10, 6, 55, 10);
 DIALOG    := New (PDIALOG, Init (R, ''));
 with DIALOG^ do
   begin {with}
     Options  := Options or ofCentered;
     R.Grow   (-1   , -1);
     Dec      (R.B.Y,  3);
     R.Assign (5, 1, 40, 2);
     Insert   (New (PStaticText, Init (R, NumStr)));
     R.Assign (18, 2, 28, 3);
     B := New (PInputLine, Init (R, 10));
     Insert   (B);
   end; {with}
 DeskTop^.Insert (DIALOG);
 BxData.Reading := ' 0        ';
 Dialog^.SetData(BxData);

 { -- Assign and prepare appropriate file names              -- }
 FILENAME := MDIR + 'DATA\RawData.Pre';
 assign  (InFile,  FILENAME);
 assign  (StdFile, DIRNAME + '\Std.Dat');
 assign  (FOFile,  DIRNAME + '\FO.Dat' );
 assign  (TmpFile, DIRNAME + '\Tmp.Dat');
 reset   (InFile);
 rewrite (FOFile);
 rewrite (StdFile);
 rewrite (TmpFile);

 { -- Write Data interpretation info to output files  -- }
 writeln (FOFile, '1');
 writeln (FOFile,  Number);
 writeln (StdFile, '1');
 writeln (StdFile, Number);
```

```
    writeln (TmpFile, '1');
    writeln (TmpFile, Number);

    { -- Read Data from InFile, convertto appropriate units,
      -- and write to Outfile                                -- }
    for INNER := 1 to Number do
      begin {for}

        Str    (INNER, BxData.Reading);
        Dialog^.SetData(BxData);

        { -- Read Fiber Optic Data and convert to microns-- }
        readln (InFile , Volt);
        Volt   := Volt / 409.5;
        OUTER  := 0;
        repeat
          OUTER := OUTER + 1;
        until (CalCurve[OUTER] >= Volt) and
              (CalCurve[OUTER - 1] <= Volt);
        Microns := OUTER + ((Volt  - CalCurve[OUTER - 1])
                /  (CalCurve[OUTER] - CalCurve[OUTER - 1]));
        writeln   (FOFile, Microns);

        { -- Skip Dummy Channel                              -- }
        readln  (InFile , volt);

        { -- Read Thermocouple data, convert to degrees      -- }
        readln  (InFile , volt);
        Volt    := volt / 409.5;
        Microns := tclin (Volt / TEMPGAIN, TCTYPE);
        writeln (TmpFile, Microns - TEMPOFFSET);

        { -- Read Standard Sensor value,
             convert to microns                              -- }
        readln  (InFile , volt);
        Volt := volt / 409.5;
        writeln (StdFile, Volt * STDGAIN + STDOFFSET);
      end;  {for}

    { -- Close all files                                     -- }
    close   (InFile);
    close   (TmpFile);
    close   (StdFile);
    close   (FOFile);
    Dispose (DIALOG, Done);
  end; {procedure Interpret_Data}

{ -------------------------------------------------------------}
```

```
procedure Directory (var SUCCESS : Boolean;
                     var DIRNAME : Pathstr);
  { -- Create DIALOG box to determine Save directory        -- }

  begin {procedure Directory}
    SUCCESS  := TRUE;

    { -- Create DIALOG Box                                   -- }
    R.Assign (20, 5, 59, 15);
    DIALOG   := New (PDIALOG, Init (R, ''));
    with DIALOG^ do
      begin {with}
        Options := Options or ofCentered;
        R.Grow  (-1   , -1);
        Dec     (R.B.Y,  3);
        R.Assign (5, 1, 35, 3);
        Insert   (New (PStaticText, Init (R,
        #13 + 'Enter a name for the Data Set')));
        R.Assign (15, 4, 25, 5);
        B := New (PInputLine,   Init (R, 10));
        Insert   (B);
        R.Assign (25, 6, 35, 8);
        Insert   (New (PButton, Init (R, ' ~O~K '    , CMOK,
                      bfDefault)));
        R.Assign (5, 6, 15, 8);
        Insert   (New (PButton, Init (R, '~C~ancel ', CMCANCEL,
                      bfNormal)));
      end; {with}
    CONTROL := DeskTop^.ExecView (DIALOG);

    { -- Perform tasks appropriate to selection              -- }
    if (CONTROL = CMCANCEL) then
      SUCCESS  := FALSE;
    if (CONTROL = CMOK) then
      DIALOG^.GetData (DIRNAME);

    Dispose (DIALOG, Done);
  end; {procedure Directory}

 { -------------------------------------------------------------}

begin {procedure Sensor}

  { -- Initialize variables and determine Directory name     -- }
  Initialize;
  Point := 1;
  Directory (SUCCESS, DIRNAME);

  { -- Continue if a Directory was chosen                    -- }
```

```
if SUCCESS then
  begin {if}

    { -- Inform user of maximum collection time              -- }
    while Time > 120 do
      begin {while}
        R.Assign  (10, 5, 68, 15);
        DIALOG    := New (PDIALOG, Init (R, ''));
        SUCCESS   := true;
        with DIALOG^ do
          begin {with}
            Options := Options or ofCentered;
            R.Grow  (-1    , -1);
            Dec     (R.B.Y,  3);
            R.Assign (5, 1, 55, 4);
            Insert   (New (PStaticText, Init (R,
              'For how many seconds do you wish to take Data ?'
            + #13
            + 'Warning : Due to Memory constraints'
            + #13
            + 'this should not be greater than 120 seconds')));
            R.Assign (20, 5, 30, 6);
            B := New (PInputLine, Init (R, 10));
            Insert   (B);
            R.Assign (35, 7, 45, 9);
            Insert   (New (PButton,Init(R,'~O~K ',    CMOK,
                           bfDefault)));
            R.Assign (5, 7, 15, 9);
            Insert   (New (PButton,Init(R,'~C~ancel ',CMCANCEL,
                           bfNormal )));
          end; {with}
        CONTROL := DeskTop^.ExecView (DIALOG);
        if (CONTROL = CMCANCEL) then
          begin {if}
            SUCCESS  := FALSE;
            Time     := 0;
          end;  {if}
        if (CONTROL = CMOK) then
          DIALOG^.GetData (TEMP);
        val (TEMP, Time, code);
        Dispose   (DIALOG, Done);
      end;  {while}

    if SUCCESS then
      begin {if}
        { -- Wait for User to begin data acquisition        -- }
        R.Assign (21, 7, 56, 14);
        DIALOG := New (PDIALOG, Init (R, 'Ready'));
        with DIALOG^ do
```

```
          begin {with}
            Options := Options or ofCentered;
            R.Grow  (-1   , -1);
            Dec     (R.B.Y,  3);
            R.Assign (1, 1, 34, 4);
            Insert (New (PstaticText, Init (R,
            #13 +
           '   Press OK when ready to begin')));
            R.Assign (13, 4, 23, 6);
            Insert   (new (Pbutton, Init (R, '~O~K', CMOK,
                                bfDefault)));
          end;  {with}
        if ValidView (DIALOG) <> NIL then
          begin {if}
            DeskTop^.Execview (DIALOG);
            Dispose (DIALOG, Done);
          end;  {if}

        ShutDown  (FALSE);

        { -- Create Desired Directory and Dummy Menu File  -- }
        ChDir      (MDIR + 'DATA');
        exec       (COMMANDCOM, '/c md ' + DIRNAME);
        ReInState  (FALSE);
        assign     (F, MDIR + 'DATA\' + DIRNAME + '.Set');
        rewrite    (F);
        writeln    (F, ' ');
        close      (F);
        DIRNAME    := MDIR + 'DATA\' + DIRNAME;

        Take_Data  (Time, Number);

        Interpret_Data  (DIRNAME, Number);

        ShutDown  (FALSE);

        { -- Delete temporary data storage file               -- }
        ChDir      (MDIR + 'DATA');
        exec       (COMMANDCOM, '/c del ' + MDIR +
                                'DATA\RawData.Pre');

        ReInState (FALSE);
      end;  {if}
    end; {if}
end;  {procedure Sensor}

{_______________________________________________________________}
{_______________________________________________________________}
```

```
procedure ShowGraph;
  { -- Runs Gallery program in PlotDirectory                   -- }

  begin {procedure ShowGraph}

    { -- Allow user to select a Data Set                        -- }
    FDIALOG := New (PFileDialog, Init (MDIR + 'DATA\*.Set',
    'Choose Data Set', '~D~ata Set', fdOpenButton, 100));
    if ValidView (FDIALOG) <> NIL then
      if DeskTop^.ExecView (FDIALOG) <> CMCANCEL then
        begin {if}
          FDIALOG^.GetFileName (DIRNAME);
          Dispose   (FDIALOG, Done);
          if DIRNAME[17] = '\' then
            DNAME   := MDIR
          else
            DNAME   := MDIR + 'DATA\';
          for OUTER := 13 to (length(DIRNAME)-4) do
            DNAME   := DNAME + DIRNAME[OUTER];

          { -- Copy Pics to c:\Plot and run Gallery             -- }
          ShutDown (TRUE);
          Chdir ('\Plot');
          exec  (COMMANDCOM, '/c Copy ' + DNAME + '*.Pic');
          exec  (COMMANDCOM, '/c Copy ' + MDIR  + '*.Pic');
          spawn (COMMANDCOM, '/c Gallery', 1, $ffff, FALSE);
          exec  (COMMANDCOM, '/c del *.Pic');
          ChDir (MAIN);
          exec  (COMMANDCOM, '/c exit'     );

          ReInState (TRUE);
        end;  {if}
   end;  {procedure ShowGraph}

{_________________________________________________________________}

procedure KillSet;

  begin {procedure KillSet}

    { -- Allow user to select a Data Set                        -- }
    FDIALOG := New (PFileDialog, Init (MDIR + 'DATA\*.Set',
    'Choose Data Set', '~D~ata Set', fdOpenButton, 100));
    if ValidView (FDIALOG) <> NIL then
      if DeskTop^.ExecView (FDIALOG) <> CMCANCEL then
        begin {if}
          FDIALOG^.GetFileName (DIRNAME);
          Dispose   (FDIALOG, Done);
          if DIRNAME[17] = '\' then
          DNAME    := MDIR
        else
          DNAME    := MDIR + 'DATA\';
        for OUTER := 13 to (length(DIRNAME)-4) do
          DNAME    := DNAME + DIRNAME[OUTER];

        ShutDown  (FALSE);

        { -- Delete all pertinent files and directories     -- }
        Chdir     (MDIR + 'DATA');
        exec      (COMMANDCOM, '/c del ' + DNAME + '.Set');
        Chdir     (DNAME);
        exec      (COMMANDCOM, '/c del *.??t');
        exec      (COMMANDCOM, '/c del *.pic');
        exec      (COMMANDCOM, '/c del *.dsk');
        ChDir     ('..');
        exec      (COMMANDCOM, '/c rd '  + DNAME);
        exec      (COMMANDCOM, '/c exit'     );

        ReInState (FALSE);
      end;  {if}
  end;  {procedure KillSet}

{_______________________________________________________________}

procedure DoFFT;
{ -- Calculate FFT                                           -- }

  var
    Bins      : longint;  { -- Number of Bins to create      -- }
    BinStr    : string ;  { -- Used in string conversions    -- }
    Cancelled : boolean;  { -- Indicates User cancellation   -- }

{ -----------------------------------------------------------------}

  procedure Separate (DIRNAME  : PathStr;
                      Input    : string ;
                  var Bins     : longint);
    { -- Separate data files into Bins of selected size      -- }

    var
      Bin     : longint; { -- Present Bin in FOR loop        -- }
      Dummy   : FLOAT  ; { -- Dummy variable used in I/O     -- }
      InFile  : text   ; { -- Input File                     -- }
      OutFile : text   ; { -- Output File                    -- }
      Point   : longint; { -- Point in FOR loop              -- }
      Points  : longint; { -- Number of total Data points    -- }

    begin {procedure Separate}
    { -- Open Input file                                          -- }
    FILENAME := DIRNAME + '\' + Input + '.Dat';
    assign  (InFile, FILENAME);
    reset   (InFile);
    readln  (InFile,  Points);
    readln  (InFile,  Points);

    { -- Calculate number of Bins at selectied Bin Size           -- }
    Bins    := trunc (Points  / BINSIZE);

    if Bins < 1 then
       Bins := 1;

    { -- Divide Data into appropriate Bins                        -- }
    for Bin := 1 to Bins do
      begin {for}
        str     (Bin, BinStr);
        assign  (OutFile, DIRNAME + '\' + Input + BinStr +
                          '.PFT');
        rewrite (OutFile);
        writeln (OutFile, '1');
        writeln (OutFile, BINSIZE);
        for Point := 1 to BINSIZE do
          begin {for}
            if Point <= Points then
              readln  (InFile,  Dummy)
            else
              Dummy := 0;
            writeln (OutFile, Dummy);
          end;  {for}
        close (OutFile);
      end;  {for}

    close (InFile);
  end;  {procedure Separate}

{ -------------------------------------------------------------}

 procedure Calculate (FILENAME : PathStr;
                      Bins     : longint);
   { -- Perform FFT's for all Bin files                           -- }

   begin {procedure Calculate}
     for OUTER := 1 to Bins do
       begin {for}
         str (OUTER, BinStr);
         spawn   (COMMANDCOM, '/c fft ' + FILENAME + BinStr +
                 '.PFT /dc /db20 /r /R /z=1024',
                1, $ffff, FALSE);
        exec    (COMMANDCOM, '/c del ' + FILENAME + BinStr +
                                     '.PFT');
      end;  {for}
  end;  {procedure Calculate}

{ -------------------------------------------------------------}

 begin {procedure DoFFT}

   { -- Allow user to select a Data Set                      -- }
   FDIALOG := New (PFileDialog, Init (MDIR + 'DATA\*.Set',
   'Choose Data Set', '~D~ata Set', fdOpenButton, 100));
   if ValidView (FDIALOG) <> NIL then
     if DeskTop^.ExecView (FDIALOG) <> CMCANCEL then
       begin {if}
         FDIALOG^.GetFileName (DIRNAME);
         Dispose   (FDIALOG, Done);
         if DIRNAME[17] = '\' then
           DNAME   := MDIR
         else
           DNAME   := MDIR + 'DATA\';
         for OUTER := 13 to (length(DIRNAME)-4) do
           DNAME   := DNAME + DIRNAME[OUTER];

         { -- Indicate ongoing operation                     -- }
         R.Assign  (10, 6, 55, 10);
         DIALOG    := New (PDIALOG, Init (R, ''));
         with DIALOG^ do
           begin {with}
             Options := Options or ofCentered;
             R.Grow  (-1   , -1);
             Dec     (R.B.Y,  3);
             R.Assign (5, 1, 40, 3);
             Insert   (New (PStaticText, Init (R,
             'Organizing Data Files')));
           end; {with}
         DeskTop^.Insert (DIALOG);
         Separate  (DNAME, 'FO',  Bins);
         Separate  (DNAME, 'Std', Bins);
         Dispose   (DIALOG, Done);

         { -- Perform Fourier Transforms                     -- }
         ShutDown  (TRUE);
         clrscr;
         writeln   ('       Fiber Optic Strain Sensor Input');
         writeln;
         ChDir     (MAIN);
         Calculate (DNAME + '\' + 'FO',  Bins);
        clrscr;
        writeln   ('          Standard Strain Sensor Input');
        Calculate (DNAME + '\' + 'Std', Bins);
        exec      (COMMANDCOM, '/c exit');

        ReInState (TRUE);

      end;  {if}
  end;  {procedure DoFFT}

{________________________________________________________________}

procedure ShowFFT;
{ -- Display FFT Graph                                        -- }

  begin {procedure ShowFFT}

    { -- Allow User to select a Data Set                      -- }
      FDIALOG := New (PFileDialog, Init (MDIR + 'DATA\*.Set',
      'Choose Data Set', '~D~ata Set', fdOpenButton, 100));
      if ValidView (FDIALOG) <> NIL then
        if DeskTop^.ExecView (FDIALOG) <> CMCANCEL then
          begin {if}
            FDIALOG^.GetFileName (DIRNAME);
            Dispose   (FDIALOG, Done);
            if DIRNAME[17] = '\' then
              DNAME   := MDIR
            else
              DNAME   := MDIR + 'DATA\';
            for OUTER := 13 to (length(DIRNAME)-4) do
              DNAME   := DNAME + DIRNAME[OUTER];

          { -- Allow User to select which Bin to display      -- }
          FDIALOG := New (PFileDialog, Init (DNAME + '\*.fft',
                          'Choose Bin', '~B~in',
                           fdOpenButton, 100));
          if ValidView (FDIALOG) <> NIL then
            begin {if}
              if DeskTop^.ExecView (FDIALOG) <> CMCANCEL then
                begin {if}
                  FDIALOG^.GetFileName (FILENAME);
                  for OUTER := 12 to (length(FILENAME)) do
                    DNAME   := DNAME + FILENAME[OUTER];
                  FILENAME  := DNAME;
                  Dispose   (FDIALOG, Done);

                  { -- Log appropriate file into plotter      -- }
                  ShutDown  (TRUE);
                  ChDir     (MAIN);
                  spawn     (COMMANDCOM, '/c lg fft ' +
                             FILENAME, 1, $ffff, FALSE);
                  exec      (COMMANDCOM, '/c exit');
                  ReInState (TRUE);

                end;  {if}
            end;  {if}
        end;  {if}
  end;  {procedure ShowFFT}

  {_________________________________________________________________}

  procedure Graph;
    { -- Creates Time vs Input Graphs                           -- }

    var
      Axis       : integer;  { -- Indicates Y Axis of graph     -- }
      Cancelled  : boolean;  { -- Cancellation Indicator        -- }
      Header_In  : text   ;  { -- Builds Plot File Headers      -- }
      Header_Out: text    ;  { --    ""              ""         -- }
      Title      : string ;  { -- Title of Graph                -- }
      X_Axis     : string ;  { -- Title of X Axis               -- }
      Y_Axis     : string ;  { -- Title of Y Axis               -- }

{ ----------------------------------------------------------------------}

    procedure Box (var Cancelled : boolean;
                   var FILENAME  : Pathstr;
                   var Axis      : integer);
    { -- Create DIALOG box for user to chooses axes             -- }

      begin {procedure Box}
        Axis := 0;

        { -- Create DIALOG Box                                  -- }
        R.Assign      (10, 2, 65, 15);
        DIALOG := New (PDIALOG, Init (R, 'Make Graph'));
        with DIALOG^ do
          begin {with}
            Options := Options or ofCentered;
            R.Grow  (-1   , -1);
            Dec     (R.B.Y,  3);
            R.Assign (15, 3, 35, 6);
            B := New (PRadioButtons, Init (R,
                 NewSItem ('~F~OSensor',
                 NewSItem ('~S~tdSensor',
                 NewSItem ('~T~emperature',
                 NIL)))
                 ));
        Insert   (B);
        R.Assign (5 , 2, 25, 3);
        Insert   (New (PLabel, Init (R, 'Y - Axis', B)));
        R.Assign (11, 10, 21, 12);
        Insert   (New (PButton, Init (R, '~O~k', CMOK,
                      bfDefault)));
        R.Assign (34, 10, 44, 12);
        Insert   (New (PButton, Init (R, '~C~ancel', CMCANCEL,
                      bfNormal)));
      end;  {with}

      { -- Initialize input settings                           -- }
      DIALOG^.SetData (Axis);

      { -- Get Command from box                                -- }
      CONTROL := DeskTop^.ExecView (DIALOG);

      { -- If OK was pushed then continue                      -- }
      if (CONTROL = CMOK) then
        begin {if}
          DIALOG^.GetData (Axis);
        end;  {if}

      { -- If ESC was pushed, close box and set CANCEL         -- }
      if CONTROL = CMCANCEL then
        Cancelled := TRUE;

      Dispose (DIALOG, Done);
      Case Axis of
        0 : FILENAME := 'FO.Dat';
        1 : FILENAME := 'Std.Dat';
        2 : FILENAME := 'Tmp.Dat';
      end;  {Case}
    end;  {procedure Box}

{ ------------------------------------------------------------}

  procedure Average (var FILENAME : PathStr);
    { -- Average each second of Data and create a plot file -- }

    type
      BxRec = record
          Number : string[10];
        end;

    var
      BxData   : BxRec   ; { -- Record stores Dialog Box Data -- }
      Data     : FLOAT   ; { -- Stores Data read from InFile  -- }
      InFile   : text    ; { -- Data Input File               -- }
  NumStr   : string ; { -- String to build Dialog Box      -- }
  OutFile  : text   ; { -- Output File for averaged Data -- }
  Tempr    : longint; { -- Data acquisition variable      -- }
  Total    : FLOAT  ; { -- Used to Average input Data     -- }

begin {procedure Average}

  { -- Set Up input and output files                       -- }
  assign  (InFile,   FileName);
  reset   (InFile);
  assign  (OutFile, 'C:\FOSensor\Plot.Dat');
  rewrite (OutFile);
  readln  (InFile,  Tempr);
  writeln (OutFile, Tempr);
  readln  (InFile,  Tempr);
  writeln (OutFile, trunc(Tempr / BINSIZE));

  { -- Create monitor window for operation                 -- }
  Str       (trunc(Tempr/BINSIZE), NumStr);
  NumStr    := 'Averaging ' + NumStr + ' Seconds';
  R.Assign  (10, 6, 45, 10);
  DIALOG    := New (PDIALOG, Init (R, ''));
  with DIALOG^ do
    begin {with}
      Options  := Options or ofCentered;
      R.Grow   (-1   , -1);
      Dec      (R.B.Y,  3);
      R.Assign (5, 1, 30, 2);
      Insert   (New (PStaticText, Init (R, NumStr)));
      R.Assign (14, 2, 19, 3);
      B := New (PInputLine, Init (R, 10));
      Insert   (B);
    end; {with}
  DeskTop^.Insert (DIALOG);
  BxData.Number := ' 0         ';
  Dialog^.SetData(BxData);

  for OUTER := 1 to trunc(Tempr / BINSIZE) do
    begin {for}
      Total := 0;

      { -- Update Monitor Box                              -- }
      Str    (OUTER,   BxData.Number);
      Dialog^.SetData (BxData);

      { -- Average BINSIZE Numbers                         -- }
      for INNER := 1 to BINSIZE do
        begin
          readln (InFile,   Data);
          Total  := Total + Data;
         end;
        Total := Total / BINSIZE;

        { -- Write Average to OutFile                                  -- }
        writeln (OutFile, Total);
       end;  {for}

      { -- Close Files and Windows                                     -- }
      close   ( InFile);
      close   (OutFile);
      Dispose (Dialog, Done);

      FileName := 'C:\FOSensor\Plot.Dat';

     end;  {procedure Average}
{ -----------------------------------------------------------------------}

 begin {procedure Graph}

   { -- Allow User to select a Data Set                                -- }
   Cancelled := FALSE;
   FDIALOG := New (PFileDialog, Init (MDIR + 'DATA\*.Set',
   'Choose Data Set', '~D~ata Set', fdOpenButton, 100));
   if ValidView (FDIALOG) <> NIL then
     if DeskTop^.ExecView (FDIALOG) <> CMCANCEL then
       begin {if}
         FDIALOG^.GetFileName (DIRNAME);
         Dispose   (FDIALOG, Done);
         if DIRNAME[17] = '\' then
           DNAME   := MDIR
         else
           DNAME   := MDIR + 'DATA\';
         for OUTER := 13 to (length(DIRNAME)-4) do
           DNAME   := DNAME + DIRNAME[OUTER];

         { -- Allow User to select Y - Axis                            -- }
         Box (Cancelled, FILENAME, Axis);
         FILENAME := DNAME + '\' + FILENAME;

         if not Cancelled then
           begin {if}

             Average (FILENAME);

             { -- Plot chosen file                                     -- }
             ShutDown (TRUE);
             ChDir    (MAIN);
             Case Axis Of
```

```
                  0 : spawn (COMMANDCOM, '/c lg FO '  + FILENAME,
                                         1, $ffff, FALSE);
                  1 : spawn (COMMANDCOM, '/c lg Strn '+ FILENAME,
                                         1, $ffff, FALSE);
                  2 : spawn (COMMANDCOM, '/c lg tmp ' + FILENAME,
                                         1, $ffff, FALSE)
                end;
                exec      (COMMANDCOM, '/c exit');
                ReInState (TRUE);

              end; {if}
          end;  {if}
    end;  {procedure Graph}

  {____________________________________________________________}

    begin {procedure HandleEvent}
      TApplication.HandleEvent (Event);

      { -- Perform procedure linked to present Event           -- }
      case Event.What of
        evCommand:
          begin {evCommand}
            case Event.Command of
              cmAbout    : About;
              CMHELPME   :
                begin {Help}
                  ChDir    (MAIN);
                  ViewFile ('Help');
                end;  {Help}
              CMQHELP    :
                begin {Quick help}
                  ChDir    (MAIN);
                  ViewFile ('QHelp');
                end;  {Quick help}
              CMSENSOR     : Sensor    ;
              CMGRAPH      : Graph     ;
              CMKILL       : KillSet   ;
              CMDOFFT      : DoFFT     ;
              CMSHOWFFT    : ShowFFT   ;
              CMSHOW       : ShowGraph;
            else
              Exit;
            end;  {case}
            ClearEvent (Event);
          end;  {EvCommand}
      end;  {Case}
    end;  {procedure HandleEvent}
```

```
{_______________________________________________________________}
{_______________________________________________________________}

var
  FO  : TFOMAN;  { -- represents backboard                    -- }

begin
  assign  (F, 'c:\FO.cfg');
  reset   (F);
  readln  (F,  MAIN);
  readln  (F,  FOCAL);
  close   (F);
  FOCAL := 'c:\' + MAIN + '\' + FOCAL + 'CAL.DAT';
  MDIR  := 'c:\' + MAIN + '\';
  MAIN  := '\'   + MAIN;

  { -- Locate DOS Command File                                -- }
  COMMANDCOM    := getenv ('ComSpec');
  if COMMANDCOM  = '' then
     COMMANDCOM := '\Command.Com';
```

# Appendix C. Description of Detection System

## C.1 Introduction

The strain sensor was originally designed to work with a fiber optic bridge that would provide compensation for both LED and photodiode drift. This was to be accomplished by having two LEDs modulated with different frequencies.

LED modulation frequencies of 1.5 kHz and 15 kHz were chosen. The lower 1.5 kHz modulation frequency was selected to be above any frequencies expected from the strain sensor, and the 15 kHz modulation was selected to be well above the 1.5 kHz so that the energy in each of these two frequencies at each photodiode output could be more easily separated.

## C.2 Modulation Waveform Generation

U1 on the Frequency Generation Board is a TTL output 6 MHz crystal oscillator (refer to Figure C-1). A crystal oscillator was chosen so that very stable precise modulation frequencies could be generated. U2 is a 74LS90 counter. It was designed to divide the 6 MHz oscillator output by 5 to give a 1.2 MHz output. U3, another 74LS90, divides this output by 10 to give an output of 120 kHz. This output can be seen at TP-2 and is also sent to U6 and U4. U4, another 74LS90, divides the 120 kHz by 10 to give 12 kHz. U6 is a 74ALS163 counter which was used to divide the 120 kHz output from U3 by 2, 4, and 8 to give 60 kHz, 30 kHz, and 15 kHz square waves on its QA, QB, and QC outputs, respectively. In a similar manner U5, another ALS163 counter, generated outputs of 6 kHz, 3 kHz, and 1.5 kHz.

All frequency generation, using TTL digital counters, was performed on a separate board from the linear circuits to keep interference due to switching transients to a minium. In the final configuration, only the outputs from U6 on the Frequency Generation Board were used. These outputs were sent to U3, a HI-508 multiplexer on the Transmitter Board.

The use of a fiber optic bridge circuit required that the low frequency modulated 1.5 kHz LED produced low harmonics near the modulation frequency of the 15 kHz LED. Also, since a high degree of sensitivity is required to detect small changes in strain, the LED drive should be very stable. Therefore, care was used to generate a low distortion sine wave with a very stable amplitude.

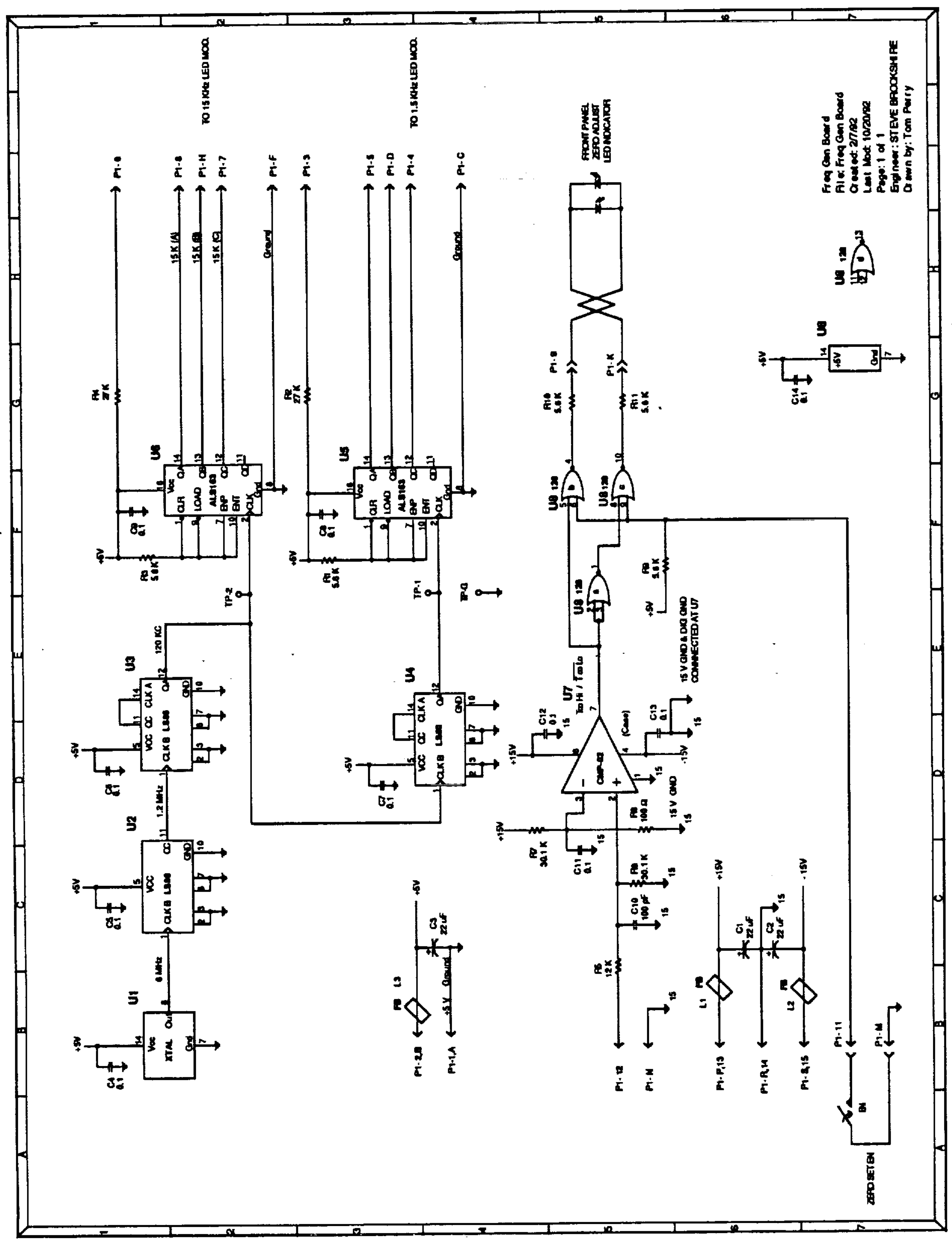

Figure C1. Schematic Diagram of Frequency Generation Board.

On the Transmitter Board a REF-01 precision voltage reference, U1, provided + 10 volts with excellent temperature stability. U3, a HA-5177 precision operational amplifier used in an inverting configuration with a gain of 1, uses refrence + 10 volts from U1 as an input to provide a stable -10 volt source. Both of these reference voltages were sent to voltage dividers R4 through R6 and R7 through R9 which provided inputs to the HI-508 multiplexer, U3. The selected input was determined by digital code at the A0, A1, and A2 inputs to U3 from the 74ALS163 counter, U6, on the Frequency Generation Board as discussed previously. The combination of this digital input address code and the highly stable input voltages derived from the voltage references and proper divider ratios provided an 8 times oversampled staircase approximation of a sine wave which is available at test point 1, i.e., TP1. Compared with a square wave, the 8 times oversampled waveform greatly simplifies the filter requirements for generating a low distortion sine wave because the first -12 dB harmonic occurs at a frequency that is seven times the fundamental, but at that frequency (10.5 kHz) the harmonics were attenuated > 60 dB by the filter.

The staircase approximation sine wave is smoothed using U4 and U5, UAF-42 universal active filters. Each uses state-variable architecture to form a time-continuous filter

A bi-quad configuration was also used for each filter. This resulted in a four pole bandpass filter with a center frequency of 15 kHz and a 10 percent bandwidth. That signal appears at test point 3, i.e., TP3, and is also sent to the LED driver.

## C.3 LED Driver

A HA-5177 precision operational amplifier, U6, HA-5002 current buffer, U7, and other passive components on the Transmitter Board are used to form a current amplifier to drive the LED as shown in Figure C-2. U6 provides the voltage necessary to drive the U7 current buffer which is capable of driving up to + 200 mA of output current. The LED current is biased at 40 mA and increases or decreases about this value when modulated. Resistors R26 and R27 actually supply the nominal 40 mA to relax the drive requirements of U7 and therefore improve the lifetime of this device. Only the current required to increase or decrease this nominal current to modulate the LED is supplied by U7. In addition, resistors R24 and R25 lower the voltage drop across the output stage of U7 to lower its power dissipation and therefore aid in low failure rate. Diode D1 is placed in series with the LED to ensure that a negative voltage is not imposed across the LED.

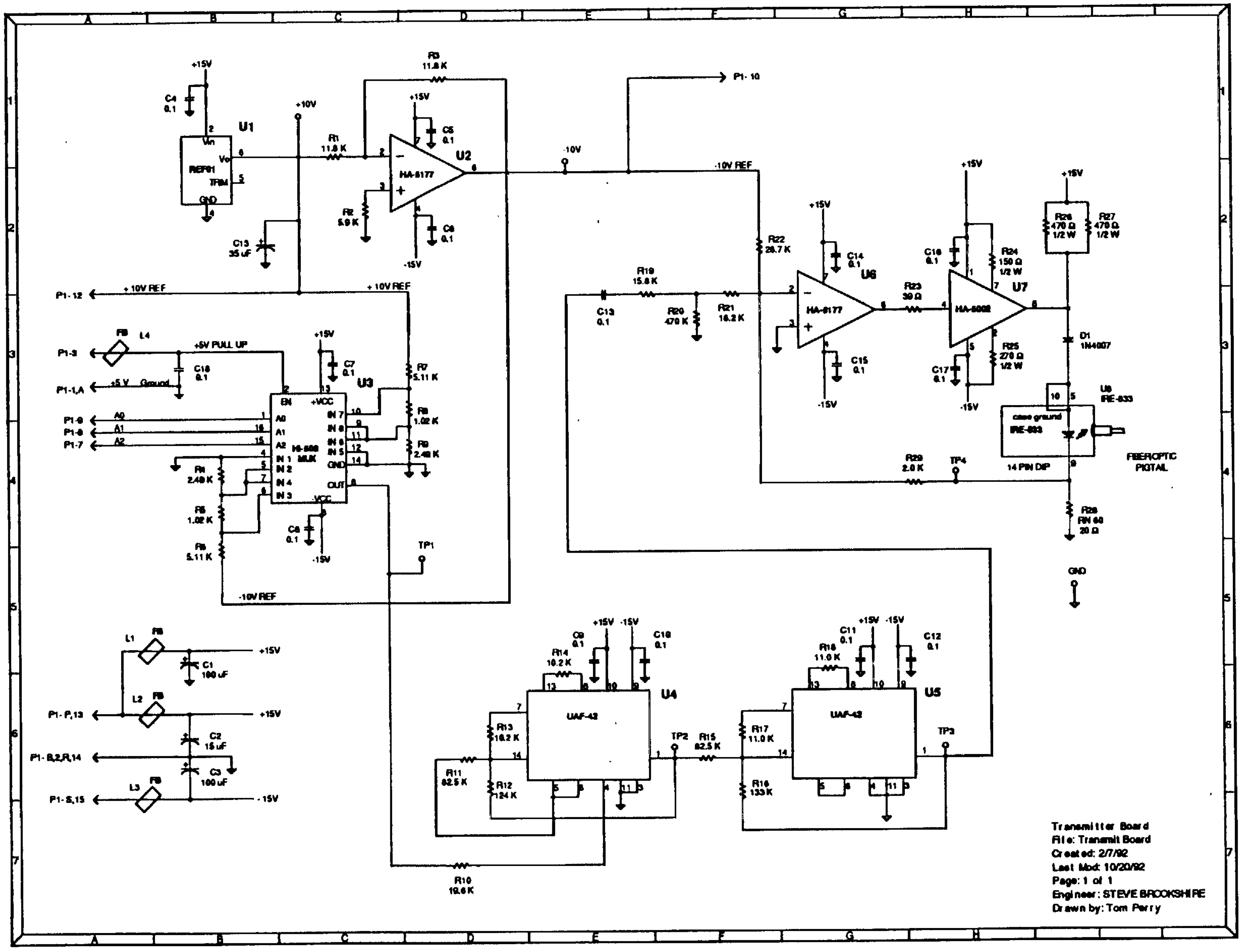

Figure C2. Schematic Diagram of Transmitter Board.

Most of the LED drive current flows through resistor R28 to ground. The voltage across R28 is sampled by R29, which provides a feedback signal for the LED current. The precise amount of unmodulted LED current is determined by the -10 reference voltage and resistor R22 that tries to remove current from the U6 summing junction, pin 2. This current causes the output of U6 to rise and drives the U7 input. U7 supplies any current necessary to maintian nearly the same output voltage, and the current in the LED increases until the voltage across R28, and therefore also across R29, is sufficient so that the current through R29 precisely balances the current through R22. The low distortion sine wave seen at TP3 is AC-coupled by C13, divided by resistors R19, 20, and 21, and applied to the U6 summing junction to provide the modulation. The remaining circuit reacts in a similar manner to match the modulation current for the LED drive current.

## C.4 Receiver Board

The Receiver Board is used to detect the light energy, separate the energy in each modulation frequency when the bridge configuration is used, and convert each modulation frequency to a DC voltage. The schematic diagram of the Receiver Board is shown in Figure C-3.

A photodiode, D1, converts the modulated light energy from the LED that is reflected from the high temperature strain sensor to electrical energy. U1, an ultralow-noise precision operational amplifier converts the photodiode current to a voltage. The voltage across the photodiode is maintained at zero and its current output is linearly related to the incident light energy by connecting the photodiode directly to the operational amplifier inputs. . Because the operational amplifier has high input impedance, the diode current only flows through the feedback resistor, R4, and the resultant voltage output developed is the diode current times the feedback resistor.

In this design, the LED is modulated. Thus, the modulation amplitude, not the DC voltage value, determines the strain. This feature reduced the effect of LED or photodiode drift.

The photodiode amplifier output was split into two legs. The top leg, which includes U2, U3, and U4, detected the 1.5 kHz modulation when the optical bridge configuration was used. The output to the bottom leg passed through a 100K-ohm ten-turn precision potentiometer on the front panel which compensated for any difference in light reflectivity between different sensors at zero strain. Next, a 4-pole bandpass filter, U5 and U6, selects only the modulation at 15 kHz.

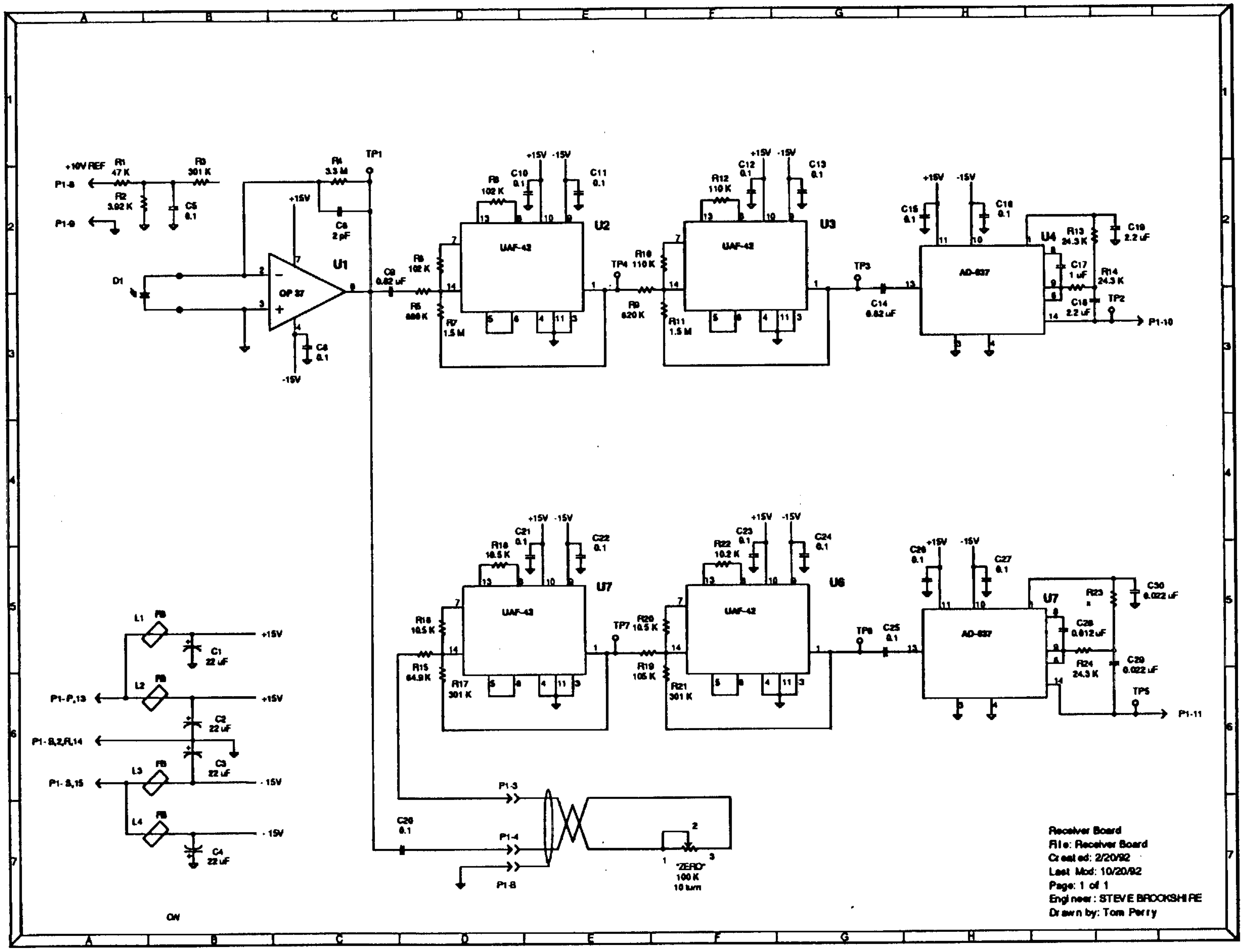

Figure C3. Schematic Diagram of Receiver Board.

This filter stage is very similar to the filter used on the transmitter board to generate a low harmonic sine wave for the LED modulation. The 15 kHz filter output can be observed at test point 6, i.e., TP6. The amplitude of the 15 kHz modulation is detected with U7, an AD-637 high precision RMS to DC converter. The DC output of U7 is filtered to help remove any ripple or noise and is sent to the Signal Conditioning Board.

## C.5 Signal Conditioning Board

The signal conditioning board (see Figure C-4) was originally designed to process the optical bridge signals to further compensate for both LED and photodiode drift and to additionally amplify and offset the response due to sensor strain so that the output will vary from 0 to + 10 volts when the sensor gap is increased by 100 microns.

Originally, both U1 and U2 were AD-538 real-time computational units. The 1.5 kHz outputs from both the reference photodiode, A1.5, and from the sensor photodiode, B1.5, were used as inputs to U1 to form the ratio $10\left(\frac{A1.5}{B1.5}\right)$ which appeared at TP1 and was also sent to U2. Since the system gains were adjusted so that A1.5 and B1.5 were normally equal in amplitude, the output of U1 was very close to +10 volts. This signal did not change with a variation in the 1.5 kHz modulated LED output; it changed only if the photodiodes drifted with respect to each other.

The second AD-538, U2, used the reference photodiode output A15, the sensor photodiode output B15, and the U1 output to compute

$$10\left(\frac{A1.5}{B1.5}\right)\left(\frac{B15}{A15}\right)$$

Again, this value was close to +10 volts with zero strain, since system gains were adjusted so that B15 and A15 were nearly equal at this condition. This output should remain stable with drift in the 15 kHz modulated LED due to $\left(\frac{B15}{A15}\right)$, and either photodiode drift should be compensated for due to the $\left(\frac{A1.5}{B1.5}\right)$ and $\left(\frac{B15}{A1.5}\right)$ ratios.

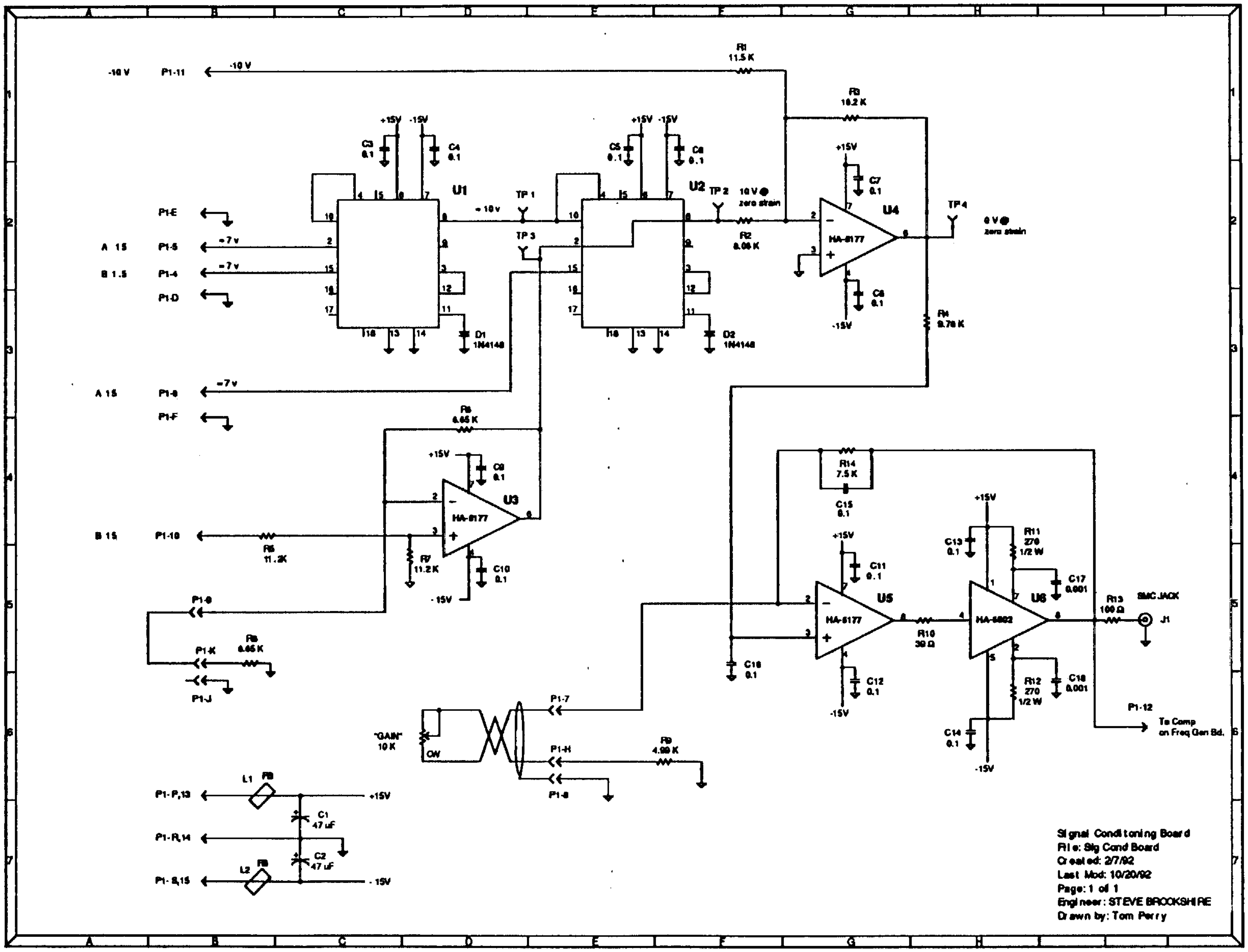

Figure C4. Schematic Diagram of Signal Conditioning Board.

At present, the optical bridge is not used, so neither U1 or U2 is in the active circuit. Only the detected 15 kHz sensor signal is being processed. This signal is buffered by U3 which was originally used as an adjustable gain stage but is now fixed. The output of U3 can be observed at TP2 and TP3 and should be close to +7 volts at zero strain.

U4 was used as an inverting amplifier to sum this +7 volts through 8.06K-ohm resistor R2 with a -10 volt reference through 11.5K-ohm resistor R1 to give approximately zero volts at TP4 when the strain was zero. As the strain was increased, the light energy from the sensor decreased, the detected sensor signal decreased, and the signal at TP4 became more positive. U5 was used to further amplify this signal. The 10K-ohm potentiometer on the rear panel compensated for different sensitivity sensors and was normally adjusted so that the output was +10 volts when the sensor gap increased to 100 microns. U6 provided a low impedance output to ensure that the output is not affected under a variety of different loads.

The output of this final gain/driver stage was sent to a BNC chassis output jack and to a zero comparator and bipolar LED driver stage on the Frequency Generation Board. The "Zero Set Enable" pushbutton switch on the front panel enables the LED driver so that the output level relative to zero can be observed. The output could be adjusted to near zero volts when the LED started to flicker between red and green by pressing the enable switch and turning the "Zero Set" (actually a gain control as described in the Receiver Board Section) in the direction indicated by the LED color.