NASA Contractor Report 181758

A High Speed Data Acquisition and Analysis System for Transonic Velocity, Density, and Total Temperature Fluctuations

Steven J. Clukey

Vigyan Research Associates, Inc.

Hampton, VA 23666

Contract NAS1-17919

December 1988

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665

(NASA-CR-181758) A HIGH SPEED DATA ACQUISITION AND ANALYSIS SYSTEM FOR TRANSONIC VELOCITY, DENSITY, AND TOTAL TEMPERATURE FLUCTUATIONS (Vigyan Research Associates) 143 p

N89-15381

CSCL 14B G3/35

Unclas
0187917

CONTENTS

1. INTRODUCTION 2

2. SYSTEM DESCRIPTION 4

2.1. HARDWARE 4
2.1.1. Amplifier and Filter Subsystem 5
2.1.2. High Speed Digitzer Subsystem 5
2.1.3. Low speed digitizer 6
2.1.4. Computer link to tunnel computer 6
2.1.5. Computer Peripherals 7
2.1.5.1. Disk storage 7
2.1.5.1.1. 55Mb hard disk 7
2.1.5.1.2. 20Mb hard disk 8
2.1.5.2. Tape storage 8
2.1.5.3. Display 8
2.1.5.4. Plotter 8
2.1.5.5. Printer 8

2.2. SOFTWARE 8
2.2.1. Software environment 8
2.2.2. Baseline software 9
2.2.3. Hot wire application software 9
2.2.4. Configuration files 10
2.2.5. Sequence program files 10
2.2.6. Hot wire data acquisition 10
2.2.6.1. Calibration 11
2.2.6.1.1. Observation files 12
2.2.6.2. Dynamic data 13
2.2.6.2.1. Fluctuating data 13
2.2.6.2.1.1. Fluctuating data volume 14
2.2.6.2.1.2. Fluctuating data disk file naming convention ... 14
2.2.7. Coefficient calculations 15
2.2.7.1. Sensitivity calculations 16
2.2.7.2. Calculating velocity, density, and temperature fluctuations ... 17
2.2.8. Precision filter system control 19
2.2.9. Utility Functions 20

3. OPERATION 20
3.1. System setup - hardware 20
3.2. System setup - software 21
3.3. ACQUIRE installation 21
3.4. System variables 21
3.5. Binary switches - digitizer configuration - MULTITRAP ... 22
3.6. Plot setup 22
configuration is saved to disk. 22

3.7. Sequence program - initialization 22
3.8. Sequence program - acquisition 23
3.9. File transfer to PC 23

4. SYSTEM STRENGTHS 24
4.1. ACQUIRE 24
4.2. Data logging 25
4.3. Computer link 25
4.3.1. DDAS to tunnel computer 25
4.3.2. DDAS to PC 25

5. LIMITATIONS 25
5.1. Uncalibrated wires 25
5.2. Data storage 26
5.3. Compute speed 27
5.3.1. Hardware - central processing unit (CPU) 27
5.3.2. Data structure 27
5.3.3. Operating system 28
5.3.4. Programming language 28

6. RECOMMENDATIONS 29
6.1. Improvement options 29
6.1.1. Array processor hardware 29
6.1.2. Faster CPU 29
6.1.3. Utilize UNIX operating system 29
6.1.4. Abandon ACQUIRE 30
6.1.5. Remote processing 30
6.2. Preferred solution 30

References: 32

Figure 1. System Block Diagram 33

Figure 2. Plot: hot wire voltage vs. mass flow 34

Figure 3. Observation Report 35

Figure 4. Plot: Waveforms - Floor strut 36

Figure 5. Plot: Waveforms - Wall strut 37

Table 1. Fluctuating Data File Name Format 38

Table 2. Fluctuating Data File Name Format 39

APPENDIX A. Program Listings 41

APPENDIX B. Function definitions 88

APPENDIX C. Sequence Programs 132

Abstract

This report describes the high speed Dynamic Data Acquisition System (DDAS) which provides the capability for the simultaneous measurement of velocity, density, and total temperature fluctuations. The system of hardware and software is described in context of the wind tunnel environment.

The DDAS replaces both a recording mechanism and a separate data processing system. The data acquisition and data reduction process has been combined within DDAS. DDAS receives input from hot wires and anemometers, amplifies and filters the signals with computer controlled modules, and converts the analog signals to digital with real-time simultaneous digitization followed by digital recording on disk or tape. Automatic acquisition (either from a computer link to an existing wind tunnel acquisition system, or from data acquisition facilities within DDAS) collects necessary calibration and environment data. The generation of hot wire sensitivities is done in DDAS, as is the application of sensitivities to the hot wire data to generate turbulence quantities. The presentation of the raw and processed data, in terms of root mean square values of velocity, density and temperature, and the processing of the spectral data is accomplished on demand in near-real-time with DDAS.

This paper describes the interface to DDAS and the internal mechanisms of DDAS. A summary of operations relevant to the use of the DDAS is also provided.

Symbols

A_1-A_8	Constants in equation (1)
E	mean voltage across wire
e'	instantaneous voltage across wire (less the mean)
G_w	instrumentation amplifier scalar
S_u	velocity sensitivity $\left.\frac{\partial \log e}{\partial \log u}\right\rvert_{\rho, T_0, T_w}$
S_ρ	density sensitivity $\left.\frac{\partial \log e}{\partial \log \rho}\right\rvert_{u, T_0, T_w}$
S_{T_0}	temperature sensitivity $\left.\frac{\partial \log e}{\partial \log T_0}\right\rvert_{u, \rho, T_w}$
T_0	mean total temperature
T_w	mean temperature of heated wire
u	mean velocity
ρ	mean density

1. INTRODUCTION

Recent advancements have been made in hot wire anemometry techniques which allow a three wire probe to separate three components of the perturbations in the flow field. Velocity, density and total temperature fluctuations can be determined as a function of three parallel hot wires, since at subsonic and transonic speeds it is generally conceded that the voltage measured across a heated wire mounted normal to the flow and operated with a constant temperature anemometer is a function of velocity, density and total temperature.[1] Under these conditions, a single equation is obtained for the fluctuating voltage across a single wire which is a function of the three variables - velocity, density, and total temperature. Quantitative measurements for the three fluctuations in the flow variables have used probes with three wires mounted normal to the flow and operated at three different "overheats".

The development of a dedicated hardware and software system to support hot wire anemometry at NASA Langley Research Center in the Fluid Dynamics Branch of the Transonic Aeronautics Division was precipitated by the necessity to process simultaneous hot wire data from three wire probes more rapidly than previously possible.

Prior to the development of the DDAS, all data was acquired on FM tape, and all processing was done in an off-line batch mode. This method delayed recognition of faulty or incomplete data, and test results were often delayed several months.

During a flow diagnostics test in the 8 Foot Transonic Pressure Tunnel (8'TPT) at NASA Langley Research Center in January of 1988[4] the DDAS was connected in parallel to the existing test instrumentation systems to provide an initial test bed for the new system. See Figure 1. The DDAS was not designated as a primary data acquisition or reduction system, but it soon became apparent that the data logging capabilities would be especially helpful in collecting the hot wire calibration data in an easily manageable format. The hot wire calibration data and the generation of hot wire sensitivities were processed only by the DDAS and the calibration data and sensitivities were used both by DDAS and by other data processing facilities. As a test of the digitization and recording capability, dynamic data was routinely digitized in parallel with the FM tape recordings. As soon as adequate calibration data was collected, the DDAS processed some of the data, and provided velocity, density, and total temperature turbulence measurements. These results compared favorably with subsequent off-line batch data processing.

A second test was supported to compare hot wire techniques and laser velocimetry techniques in the Basic Aerodynamic Research Facility (B.A.R.F.).[3,5]

The DDAS provides the processes necessary to:

1) acquire the hot wire calibration data
2) acquire the dynamic hot wire data
3) generate hot wire coefficients and sensitivities
4) compute velocity, density, and total temperature fluctuations
5) compute other statistical relationships
6) provide spectral analysis
7) manage data
8) produce reports and plots

2. SYSTEM DESCRIPTION

DDAS is a system of hardware and software based on systems purchased from Data Laboratories, Ltd., Precision Filters, Inc., and Hewlett Packard Corporation. Modifications and enhancements to the software and hardware have converted a waveform recorder into a hot wire anemometry acquisition and processing system specifically tailored to the three wire technique that yields separate velocity, density and total temperature components of turbulence.

2.1. HARDWARE

The system (fig. 1) is divided into an analog front end, and a computer-based processing and display section. The analog front end consists of a filter/amplifier subsystem, a high speed digitizer, and a low speed digitizer (or a data link to another acquisition computer). All are fully computer controlled. The processing and display subsystem controls the analog subsystems, and receives the digitized data, processes the data, displays the data, and stores the data in a permanent file.

2.1.1. Amplifier and Filter Subsystem

The analog signals from the hot wire anemometers are first routed to the Precision Filters, Inc. precision amplifier and filter subsystem. This subsystem is currently configured for four channels, providing support for only one three wire probe. Each channel successively passes the anemometer signal through a pre amplifier, high pass filter, low pass filter, and a post amplifier. The full bandwidth capability of each channel is .1Hz to 200KHz, but the high pass and low pass filters usually provide a narrower bandwidth (1Hz to 5KHz). The high pass filter acts as the anti-aliasing filter for the high speed digitizer.

2.1.2. High Speed Digitzer Subsystem

A high speed digitizer, called a Multitrap modular waveform recorder by Data Laboratories, Ltd., is configured to digitize up to 14 channels of fluctuating hot wire data (three channels are required for each 3-wire probe) at rates of up to one million (1M) samples per second (6 channels at up to 1M samples/sec, and 8 channels at up up to 256 thousand (256K) samples/second. This data is stored temporarily in the Multitrap buffer memories (up to 256,000 samples per channel), and then transferred to the

HP 9000/330 computer at about 100,000 samples per second - one channel memory at a time - via a dedicated 16 bit parallel bus (GPIO).

2.1.3. Low speed digitizer

This subsystem is not currently implemented, and an existing wind tunnel data acquisition system provided DDAS the functions of a low speed digitizer subsystem. However, the optional low speed digitizer subsystem would consist of a multiplexer and digitizer selected for collection of mean values, not fluctuating values. This subsystem would be used to collect calibration data and tunnel parameter data, which would be logged for further processing of the calibration and fluctuating data.

2.1.4. Computer link to tunnel computer

The General Purpose Interface Bus (GPIB), an IEEE488 standard bus, is used to receive static data from the existing tunnel data acquisition system computer. The tunnel computer transmits a packet of data relevant to the tunnel conditions and hot wire calibration data. This link was selected because of its availability in both the tunnel computer and in the DDAS computer. It provides an 82,000 byte per second transfer rate, which is more than adequate to receive as many as four complete ASCII data packets per second. The tunnel computer actually sent only one packet per second.

Use of the existing tunnel data acquisition system to collect the tunnel conditions and the additional mean values related to the hot wire calibration data eliminates the need for a parallel hardware system (the low speed digitizer), and the need to develop instrument calibration software and hardware. It does, however, provide additional work for the tunnel computer personnel to configure their acquisition setup to handle the additional channels, and to provide the GPIB software to generate the data packets for the DDAS.

The link is configured with the DDAS end as <u>not</u> system controller, and as device 01. This was accomplished by setting switches on the HP 98624A HP-IB Interface Card inserted into the computer specifically for the link. The select code was set to 8; interrupts are not relevant, since they are not used. The wind tunnel computer providing the data packet is configured as system controller, and outputs ASCII data packets at a rate set by the wind tunnel computer.

The packet is read into a DDAS packet buffer with one program statement in the subroutine Get_packet in module MODUSR2: ENTER Pkt_sc;Pkt$(*), where Pkt_sc is equal to 8, and the Pkt$ array was sized for 47 each 80 character strings.

The packet format is shown in Table 1.

2.1.5. <u>Computer Peripherals</u>

2.1.5.1. <u>Disk storage</u>

75Mb of non removable disk storage is available for programs and data. In addition, a 1.2Mb removable disk drive is available for program development and hot wire calibration data.

2.1.5.1.1. <u>55Mb hard disk</u>

The computer then transfers the data to a 55Mb hard disc at about the same 100,000 samples per second - one channel at a time.

2.1.5.1.2. <u>20Mb hard disk</u>

Programs and support software is stored on the 20Mb hard disk.

2.1.5.2. <u>Tape storage</u>

Once the data disc is full, the data is copied to a 67Mb tape cartridge for permanent storage.

2.1.5.3. <u>Display</u>

A color CRT is the system console and data display.

2.1.5.4. <u>Plotter</u>

An 8 pen autoload flatbed plotter is available for plot generation, and is used to display dynamic data and hot wire calibration data.

2.1.5.5. <u>Printer</u>

A dot matrix printer is available for data display. It can produce screen dumps, but is used primarily to generate a record of the hot wire calibration data, and, as data processing is accomplished, the results of the processing are printed.

2.2. <u>SOFTWARE</u>

2.2.1. <u>Software environment</u>

All DDAS programs operate under a BASIC operating system, in an interpretive BASIC language. Several compiled subroutines are a part of the ACQUIRE software system to enhance computational speed in some parts of the software.

2.2.2. Baseline software

The ACQUIRE[5] software system, provided by Data Laboratories, Ltd. is the basis for the Dynamic Data Acquisition System (DDAS) used for the acquisition of hot wire anemometry data.

Since the ACQUIRE package is an off-the-shelf product, no attempt to describe its full capability will be made.

ACQUIRE has been slightly modified in only one area - the addition of a sequence number to a dynamic data disk file was inhibited if it was not necessary to discriminate between two files with the same name. See Section 2.2.6.2.1.2.for further discussion on the requirement for the modification.

2.2.3. Hot wire application software

Major additions were made to ACQUIRE in the form of two sub-programs: MODUSR1 and MODUSR2. These user written modules are configured according to guidelines provided by ACQUIRE, so that they will be automatically included in ACQUIRE. Appendix A contains the full program listings of these two modules.

The functions implemented in both MODUSR1 and MODUSR2 are listed in Appendix B.

2.2.4. Configuration files

Configuration files used by ACQUIRE for the 8'TPT test include acquisition setup parameters, hardware configuration parameters, and default display parameters. They are set, saved and stored by a variety of ACQUIRE functions.

2.2.5. Sequence program files

The sequence program functions of ACQUIRE provide a mechanism for specifying a series of functions to be accomplished. Both the initialization and acquisition sequence program files used to tailor the DDAS for support of the 8'TPT test are listed in Appendix C.

Although the configuration files and sequence files are not the easiest to configure, the result is a system that is literally a "turnkey" system. Turn on the hardware, allow the hardware and software to be configured, and press a button to simultaneously acquire calibration and dynamic data.

2.2.6. Hot wire data acquisition

The ACQUIRE software system was augmented to support the specific requirements of hot wire anemometry systems currently in use at NASA Langley Research Center in the Fluid Dynamics Branch of the Transonic Aeronautics Division. Software design and implementation followed both form and style of the supplied ACQUIRE software, maintaining the appearance of a seamless environment within ACQUIRE. This feature resulted in a software system for an instrument that has evolved from a waveform recorder to a hot wire anemometry system designed to acquire and process both mean hot wire calibration data and fluctuating hot wire data.

The end result has been an easily used flow diagnostics instrument that minimizes the researchers workload.

With hot wires, there are most often two concurrent tasks: calibration of the wires, and acquisition of the dynamic, or fluctuating data. This system is designed to calibrate and process three wire probe data. The current implementation supports two three wire probes, and acquires, processes, and stores them separately.

Once the instrument is configured, each data point is acquired with the push of a single button -- one button recording. Data processing does require a few more button sequences, but only because the researcher provides more direction in the data reduction process.

For the 8'TPT flow diagnostics test in January of 1988, the hot wires had not been previously calibrated, so concurrent calibration data and dynamic data acquisition was necessary.

2.2.6.1. Calibration

Calibration of three wire probes require a complete data system to acquire mean (static) conditions of both the operational environment and of the mean hot wire values.

It is assumed that the hot wires are sensitive to velocity, density, and total temperature.[1] To determine what the sensitivity is to each variable, the mean voltage output of each hot wire must be measured at each combination of velocity, density, and temperature. The tunnel run schedule was configured to assure that adequate data points are taken to provide a realistic profile of sensitivities.

The run schedule was also selected to expose the hot wire probe to the highest dynamic pressures first, so that if a wire is going to break, then the least amount of tunnel time will be lost.

Each of the three parallel hot wires on a probe are operated at different overheats, to encourage a wide separation of sensitivity between each hot wire.[1]

The calibration data consists of mean values, which do not require the high sampling rates normally invoked to digitize the dynamic data, so the calibration data is not acquired through the high speed digitizers, An existing wind tunnel data acquisition system collected the data, and then transferred it through a GPIB link to the DDAS computer. Several data packets of data are averaged and then stored in a formatted record on disk.

2.2.6.1.1. Observation files

This calibration data is stored in a file called an observation file. Other related mean data is also stored in the observation file:

- tunnel conditions,
- test identification parameters,
- auxillary data (such as RMS microphone readings, and amplifier gain settings), and
- simple calculated data (such as density, velocity, static temperature, the logs of a variety of data, and the products of the logs of a variety of data.

The observation "file" is really several files. The first is a file containing the data received from a packet sent from the MODCOMP via the HP-IB link. (Actually, several packets are averaged, and the average is stores as an observation in the first observation file.) The second and third files each contain data related only to a specific three wire probe. These files also contain tunnel condition and test identification data as well, but only the tunnel computer data and simple, computed values related to a specific probe is contained in these files.

The internal format of the observation files was carefully selected to conform to the format specifications of an existing statistics package. The Basic Data Statistics package from HP has historically been the statistics package used to reduce the hot wire calibration data, so the file format was made compatible with that package.

2.2.6.2. Dynamic data

The acquisition of the dynamic hot wire data is entirely accomplished by the off-the-shelf ACQUIRE software. ACQUIRE has all the mechanisms necessary to configure the actual data acquisition hardware and the capability to manage the data once it has been digitized and buffered by the high speed digitizer hardware, which includes the transfer from the buffer to computer memory, and the transfer of the data to disk. These functional modules are "strung together" in a sequence program mechanism, which is also an inherent part of ACQUIRE.

2.2.6.2.1. Fluctuating data

Fluctuating data is either digitized data from the hot wire anemometers, or it is calculated data from the process of computing velocity, density, and temperature fluctuations, which is discussed later. In either case, the result is a time varying array of samples of data.

2.2.6.2.1.1. Fluctuating data volume

The sheer volume of fluctuating data is worth note: since each channel can handle 256K samples (512K bytes) at a time, and there are 3 channels per probe, and the same amount of results exist, 256K X 2 X 3 X 2 = 31457K bytes per observation. For 117 observations (8' Transonic Pressure Tunnel Test 934), 368M bytes of data storage becomes necessary.

2.2.6.2.1.2. Fluctuating data disk file naming convention

DDAS collects and generates multiple channels of fluctuating data files for each test condition or observation. These files are related to a specific record in an observation file, which contains non-fluctuating scalar data related to the observation. The relationship of the fluctuating data file names to the test condition and to the observation file and record within the observation file is specifically defined by convention. The use of a naming convention allows data reduction programs to associate all data files necessary for data reduction and for naming resultant fluctuating data files. Table 2 details the naming format.

The dynamic data samples digitized by the MULTITRAP digitizer hardware are stored by the waveform recorder function of ACQUIRE on a channel-per-file basis. Fluctuating data names are generated whenever a hot wire calibration observation is logged, so that a naming convention is followed - should a request to digitize hot wire fluctuating data be processed.

2.2.7. Coefficient calculations

The process of providing the sensitivities necessary to convert three hot wire data arrays into velocity, density and temperature fluctuation arrays first requires that the calibration data be processed by multiple linear regression techniques to produce a set of coefficients for each of the three hot wires. Since the relationship of the performance of the hot wire is highly nonlinear in relationship to the velocity, density, and temperature, up to 10 coefficients are required (Eight are in use, as shown[1]:

$$\begin{aligned}\log E = {} & A_1 + A_2 \log u + A_3 \log \rho + A_4 \log_0 T_0 \\ & + A_5 \log u \log \rho + A_6 \log u \log T_0 \\ & + A_7 \log \rho \log T_0 \\ & + A_8 \log u \log \rho \log T_0 \qquad (1)\end{aligned}$$

Since velocity, density, and temperature are all known for each observation (as collected by the DDAS form the tunnel data acquisition system - in the form of P_T, P_S, and T_T), the most direct solution is through multiple linear regression.

Whenever requested by the operator, a Multiple Linear Regression routine (which is a specifically modified version of the Hewlett Packard routine MLR which was purchased as part of a statistics package) is invoked, which calculates coefficients for each hot wire on each probe. These coefficients can then be stored in a coefficient disk file related to each probe. (This internal MLR routine is not currently implemented.)

Alternatively, coefficients calculated in a separate multiple linear regression package may be read from a disk file generated by that package, or the coefficients may be manually entered through the keyboard.

2.2.7.1. Sensitivity calculations

The coefficients, which represent the hot wire relationship to velocity, density, and temperature, are combined with specific test conditions, which have been stored in an observation record of the observation file.

$$S_u = A_2 + A_5 \log \rho + A_6 \log T_0 + A_8 \log \rho \log T_0 \qquad (2a)$$

$$S_\rho = A_3 + A_5 \log u + A_7 \log T_0 + A_8 \log u \log T_0 \qquad (2b)$$

$$S_{T_0} = A_4 + A_6 \log u + A_7 \log \rho + A_8 \log u \log \rho \qquad (2c)$$

New ACQUIRE functions created in MODUSR2 allow the appropriate calibration file to be specified, and the beginning and ending observations and beginning and ending probes to be selected for the computations. For each observation and each probe, the log values of velocity, density and temperature are retrieved from the appropriate record in the observation file. Once the computation is completed for each probe, the resultant senstivitys are inserted into existing, but as yet unused variables in the previously recorded observation.

2.2.7.2. Calculating velocity, density, and temperature fluctuations

Once the sensitivities have been calculated, the operator may request that the dynamic data for a given set of observations and probes be processed in a way that yields dynamic waveforms representing fluctuating velocity, density and temperature (instead of 3 fluctuating voltages) and with turbulence figures and other statistical performance characteristics.

The equation that defines the relationship of voltages to turbulence parameters is[1]:

$$\left[\frac{e'}{E} \right]_1 = S_{u_1} \frac{u'}{U} + S_{\rho 1} \frac{\rho'}{\rho} + S_{T_{0_1}} \frac{T_0'}{T_0}$$

$$\left[\frac{e'}{E} \right]_2 = S_{u_2} \frac{u'}{U} + S_{\rho 2} \frac{\rho'}{\rho} + S_{T_{0_2}} \frac{T_0'}{T_0}$$

$$\left[\frac{e'}{E} \right]_3 = S_{u_3} \frac{u'}{U} + S_{\rho 3} \frac{\rho'}{\rho} + S_{T_0} \frac{T_0'}{T_0}$$

To solve for the three unknowns ($\frac{u'}{U}$, $\frac{\rho'}{\rho}$, and $\frac{T_0'}{T_0}$) in the three equations, rearrange, and organize for a matrix operation:

$$\begin{bmatrix} \left[\frac{e'}{E} \frac{1}{G_w}\right]_1 \\ \left[\frac{e'}{E} \frac{1}{G_w}\right]_2 \\ \left[\frac{e'}{E} \frac{1}{G_w}\right]_3 \end{bmatrix} = \begin{bmatrix} S_{u_1} & S_{\rho_1} & S_{T_{0_1}} \\ S_{u_2} & S_{\rho_2} & S_{T_{0_2}} \\ S_{u_3} & S_{\rho_3} & S_{T_{0_3}} \end{bmatrix} \times \begin{bmatrix} \frac{u'}{U} \\ \frac{\rho'}{\rho} \\ \frac{T_0'}{T_0} \end{bmatrix}$$

By rearranging again, which involves inverting the sensitivity matrix, solve for the three unknowns:

$$\begin{bmatrix} \left[\frac{u'}{U} \right] \\ \left[\frac{\rho'}{\rho} \right] \\ \left[\frac{T_0'}{T_0} \right] \end{bmatrix} = \begin{bmatrix} S_{u_1} & S_{\rho_1} & S_{T_{0_1}} \\ S_{u_2} & S_{\rho_2} & S_{T_{0_2}} \\ S_{u_3} & S_{\rho_3} & S_{T_{0_3}} \end{bmatrix}^{-1} X \begin{bmatrix} \left[\frac{e'}{E} \frac{1}{G_w} \right]_1 \\ \left[\frac{e'}{E} \frac{1}{G_w} \right]_2 \\ \left[\frac{e'}{E} \frac{1}{G_w} \right]_3 \end{bmatrix}$$

The computation of the instantaneous velocity, density, and temperature is accomplished as shown:

for each probe and each observation to be processed,
  for each of the three hot wires
    the mean hot wire voltage (E) is retrieved
    the gain (G) for the fluctuating hot wire voltage is retrieved
    the three sensitivities (u,ρ,T_0) are retrieved and placed in the sensitivity matrix
    the dynamic data file is retrieved from disk, and placed in memory
  the sensitivity matrix is inverted
  for each of the instantaneous samples
    for each of the three hot wires
      compute:

$$\frac{e'}{E} \frac{1}{G_w}$$

      and store in the independent variable matrix

matrix multiply the inverted sensitivity array by the independent variable array, and place the instantaneous turbulence ratios ($\frac{u'}{U}$, $\frac{\rho'}{\rho}$, and $\frac{T_0'}{T_0}$) in memory

compute the RMS values of $\frac{u'}{U}$, $\frac{\rho'}{\rho}$, and $\frac{T_0'}{T_0}$

store the RMS values of velocity, density, and temperature $\left[\frac{\tilde{u}'}{U} , \frac{\tilde{\rho}'}{\rho} , \frac{\tilde{T}_0'}{T_0} \right]$ for each observation into existing, but as yet unused variables in the previously recorded observation

store the fluctuating velocity, density, and temperature waveforms in disk files for later spectral investigations, utilizing existing functions of ACQUIRE

2.2.8. Precision filter system control

The computer control of the filters and amplifiers allows adaptive processing of various hot wire signals, which are dependent upon a variety of operational parameters. The software interface allows full control of each functional module within the Precision Filter system - including the calibration module, and also allows the interrogation of all status and condition data - including the calibration module. The modules are connected to provide a full calibration sequence, and to allow full operator control, semiautomatic or automatic operation. (This feature not implemented as of April 88).

2.2.9. Utility Functions

Other utility functions were implemented to enhance operational characteristics of the system. The ability to eject plots, and the ability to plot "special" hot wire calibration data, are "plot utilitys". The ability to list categories of files on the disk, to purge extraneous files - or groups of files, the ability to copy or move files - or groups of files - to another disk (or tape) are "file utilitys".

Function LOGFILLE TO PC was used to transfer observation files containing logged and computed data to another system. A GPIB bus connects DDAS to the PC, where a GPIB card (National Instruments or HP) is installed. Appendix D contains the PC BASIC program used with the HP GPIB card to receive and store the data on the PC disk.

3. OPERATION

Operation of DDAS begins prior to the tunnel operation. Configuration of both the ACQUIRE software and the high speed digitizer is accomplished from within the software. Configuration files of various types are generated by the software and are recallable either at power on time or from within a sequence program.

3.1. System setup - hardware

The initial configuration of hardware is accomplished only once. The assignment of device addresses is as follows:

GPIB devices:

printer	701
20Mb program disk	703,0

67Mb data tape	703,1
plotter	705
high speed digitizer (control link)	708
computer link	801
55Mb data disk	1400

GPIO (16 bit parallel) devices:

high speed digitizer (data link)	12

3.2. <u>System setup - software</u>

The ACQUIRE operating system was configured to operate within the memory constraints of 4Mbytes, and to be configured for 14 channels of digitizers, and 14 channels of data memory. Refer to the ACQUIRE operations manual for further details.

3.3. <u>ACQUIRE installation</u>

Upon receipt of the software, the installation procedure defined by the manufacturer allows the software to be configured for existing hardware, including amount of computer memory, number of digitizers in the digitizer chassis, and the maximum number of channels in the computer memory at any one time.

3.4. <u>System variables</u>

A file structure is maintained in the ACQUIRE software for containing a wide variety of currently selected operating parameters. This mechanism allows the operator to interactively select preferred operational conditions, and then store the "sysvars" on disk for later retrieval. These parameters include, but are not limited to, display format, memory length for each channel, waveform file names, waveform channel selection, system variables file name, binary switch name, plot file name, and sequence file name. To recall a specific set of system variables automatically at power on time, the system variables are stored in a file called "AUTOVARS".

3.5. Binary switches - digitizer configuration - MULTITRAP

The high speed digitizer is configured utilizing an interactive session to select sampling rates, gains, trigger modes, data block size, etc., and then the configuration of the binary switches within the digitizer are saved in a binary switch configuration file. To recall a specific configuration for the digitizer automatically at power on time, the binary switches are stored in a file called "AUTOSW".

3.6. Plot setup

The format of a plot is defined interactively and may then be saved on a plot file. The actual data is not saved in the file. Once the waveform channel in memory is selected, the position, scaling, and labeling of the axis is defined, and waveform labeling is determined. Once all channels are positioned and defined, the plot title is defined, and the plot configuration is saved to disk.

3.7. Sequence program - initialization

An initialization sequence program is interactively generated, which will determine a sequence of functions to be performed to set up DDAS to a configuration relevant to a specific wind tunnel test. See Appendix C for a listing of the initialization sequence used for the 8'TPT test. To recall the initialization sequence program at power on time, the sequence is stored in a file called "AUTOSEQ".

3.8. Sequence program - acquisition

A run sequence program is interactively generated, which will determine a sequence of functions to be performed to:

log calibration data

digitize and store fluctuating data

plot hot wire mean voltages vs. mass flow, ρu, (see Fig. 2)

print a report displaying many parameters of the current observation

whenever the operator presses a single button. See Appendix C for a listing of the run sequence used for the 8'TPT test. The run sequence program is automatically loaded by the initialization sequence program, so that once all configurations are defined, powering on the system, and pressing a button is all that is necessary to simultaneously acquire both calibration and fluctuating hot wire data.

3.9. File transfer to PC

The probe log data files - one or both - can be transferred to a PC via a dedicated GPIB cabled between DDAS and a PC. The PC BASIC program "XFR.HP" (see Appendix D) should be started first, and then, before providing the requested file name, invoke the DDAS function LOGFILE TO PC. Refer to the relevant function sheet in Appendix B for details on proper configuration prior to starting the transfer. When the file name is then entered into the PC, which defines where the data is to be stored, the transfer will begin.

4. SYSTEM STRENGTHS

4.1. ACQUIRE

ACQUIRE, in combination with the hardware is a very versatile waveform recorder:

It controls all the hardware associated with the system.
It manages hardware and software configuration - via files.
It manages process, or "sequence" files.
It manages dynamic data files and internal arrays of data.
It provides a choice of operator dialogue techniques, including: cursor, menu, and command line entry.
It provides data display management.
It provides the waveform plotting capabilities.
A Digital Signal Processing package is included which provides:

Fast Fourier Transforms
filters
power spectrum
transfer functions

The acquisition of the dynamic data, and the storing of the dynamic data is a very significant strength of ACQUIRE. But most importantly, the internal design allowed application routines to be written into ACQUIRE, which produces a set of software that appears to the user to be a single entity, without seams, and fully integrated.

4.2. Data logging

The internal log file format allows direct access by a commercially available statistics package, which includes a multiple linear regression analysis capability necessary to generate coefficients used in creating hot wire sensitivities.

4.3. Computer link

4.3.1. DDAS to tunnel computer

A software/hardware link is currently used with the MODCOMP data acquisition computer to receive mean data values, but a self-contained, accurate and reliable static data acquisition subsystem could be integrated, making the DDAS self-contained. The use of an existing data acquisition system for the collection of mean values transfers the instrument calibration requirements for those values to another system.

4.3.2. DDAS to PC

The LOGFILE TO PC function to transmit the logged and computed parameters to another system, where the data is reformatted and imported to a spreadsheet program (Lotus Symphony) for further analysis and data presentation.

5. LIMITATIONS

5.1. Uncalibrated wires

Hot wire calibration currently consumes the major portion of the tunnel operation time. Although not a limitation of the DDAS, the process of calibrating hot wire probes relative to temperature, density and pressure is currently the most expensive part of the three wire technique.

Pre-calibrated wires would allow real time processing of the voltages from the three wires into the velocity, density, and temperature components of turbulence. The facility would be much less expensive to construct and operate than the wind tunnel to be supported, since the size could be much smaller, and the tolerable turbulence levels could be higher, since only the mean values of velocity, density, and temperature are used in determining hot wire sensitivities.

For wind tunnels not capable of independently controlling velocity, density (or total pressure) and temperature, the three wire technique requires that the wires be pre-calibrated, since the sensitivities could not be properly determined in such a wind tunnel.

Although a hot wire calibration tunnel has been partially constructed, it is not yet operational due to manpower and funding constraints. A data logging program module developed for DDAS is available as a module for eventual integration with an instrumentation system expressly for the hot wire calibration facility.

5.2. Data storage

The acquisition of 2.5 seconds (50KHz bandwidth) of fluctuating data representing a single 3-wire probe hot wire output requires the rapid digitization, processing, display and storage, of 1.5Mb of data. 150Mb of data could easily be collected in 8 hours of transonic wind tunnel testing. The hardware originally purchased with ACQUIRE can adequately digitize 14 hot wire channels of the dynamic components. Modifications have been made to rapidly transfer the digitized data to the computer. adequate hardware exists to transfer the data to permanent storage. But only 55Mb of conventional disk space is available for data. A Write Once, Read Many (WORM) laser disk drive ($14K) would dramatically improve the storage capability, since WORM drives can typically store 600-800Mb of data per disc.

5.3. Compute speed

5.3.1. Hardware - central processing unit (CPU)

The real limitation of this system was - and is - in the processing speed of the CPU. The original HP 9000/310 CPU was about as fast as an IBM PC/AT, and often took minutes to perform a simple evaluation of a few thousand points of dynamic data from a single channel. The upgrade to an HP 9000/330 (for $13K) in the beginning of 1988 improved the processing performance somewhat, but the array processing problem is still not being met head on.

5.3.2. Data structure

For each computation the array structure requires an indexing algorithm to access each sample. Through the indexing mechanism, and by representing sample values in an integer format, at least a four-fold savings of computer memory and disk space is realized. But the saving of space (memory) has become an unnecessary and unacceptable tradeoff. All generated data had been simply rescaled (as ratios) existing data, so the linear coefficients were easily determined for the new integer data arrays. However, the solution for three unknowns in three simultaneous equations does not allow for a simple determination for the linear coefficients to scale the integer data. To compute instantaneous turbulence fluctuation, each instantaneous voltage must be translated to floating point by applying first order coefficients. Then the floating point computations (a floating point matrix multiply operation is in itself not a fast operation) are accomplished for each instant in time. But the answers are in a floating point format and no linear coefficients have been determined to convert the floating point answers to a range of integer values. Therefore, the hot wire computations are accomplished twice - once to determine linear coefficients, and once to store the instantaneous turbulence fluctuation in an integer format. Both the integer format and the index algorithm produce excessively slow computing processes.

5.3.3. <u>Operating system</u>

The existing BASIC operating system is a single user, single task executive. It cannot support high speed communication via Ethernet. It cannot support concurrent operations; program development, data acquisition, data processing, and data communications cannot all be executing concurrently.

5.3.4. <u>Programming language</u>

The BASIC language is an interpreter, rather than a compiler, which trades off execution speed for ease of program development and maintenance. Although the ACQUIRE software takes advantage of some compiled and assembled subroutines - for speed - all of the application software is still interpreted.

6. RECOMMENDATIONS

Several solutions exist - they all require much larger investments of time and money than a mere doubling of financial resources.

6.1. Improvement options

6.1.1. Array processor hardware

A dedicated array processor is available from Analogic Corp ($27K) which is designed to interface to both the HP 9000/330 and the HP software. Modifications to the ACQUIRE Digital Signal Processing software (DSP), or development of a user-provided DSP routine to replace the ACQUIRE DSP software (3 man-months, est.) would be required

6.1.2. Faster CPU

A larger HP 9000/350 CPU ($29K) would quadruple the processing speed, and still be able to run the existing ACQUIRE software.

A combination of the Analogic array processor and the HP 9000/350 would provide the best performance possible - without abandoning the ACQUIRE software.

6.1.3. Utilize UNIX operating system

Provide the multitasking, multiuser environment necessary to support concurrent operations, Ethernet (TCP/IP) communications, a choice of programming languages - including interpretive and compiled, and a wider marketplace for software and hardware solutions like nine track magnetic tape support, laser printer support, graphics and statistics support, and data management support.

6.1.4. Abandon ACQUIRE

Abandon the ACQUIRE software system, and actively search for a software system that operates in the UNIX environment, has the potential for supporting the high speed digitizer, can acquire, process, display, and save both the mean data and fluctuating data more rapidly than ACQUIRE.

6.1.5. Remote processing

Processing the data elsewhere: ACD, MODCOMP or other larger computer resource. The solution is suggested by the existence of other computational resources that may be made available, including the tunnel computer, and would provide parallel processing of the DDAS data once the hot wire sensitivities are available, and once the instantaneous data has been acquired and transferred to the other resource. This approach assumes that a viable communications link like Ethernet is available. At NASA LaRC, this capability is called LaRCNET. Although this link is proposed for the East Area of LaRC - where 8'TPT is located, its presence is still about 2 years distant. LaRCNET also implies - by its very existence - that the ACD computational resources will be in great demand. The MODCOMP connection, however, proposes a much closer solution. Although not an array processor machine, and not yet capable of communicating via LaRCNET, the access via a local Ethernet (to eventually be a part of LaRCNET) is scheduled for the forth quarter of 1988.

6.2. Preferred solution

The preferred solution is: abandon ACQUIRE for a UNIX compatible set of software, translate existing hot wire programs to the UNIX environment, purchase new statistical software, and purchase an array processor and a faster CPU (the HP 9000/350). About 1 man-month would be required for conversion of the hot wire software, and about 3 man-months would be required to integrate all the various software and hardware modules. This solution minimizes the engineering integration risks attendant in any system of this complexity.

References:

1. P. C. Stainback, C. B. Johnson, et al; Preliminary Measurements of Velocity, Density and Total Temperature Fluctuations in Compressible Subsonic Flow; AIAA-83-0384

2. P. C. Stainback; Some Influences of Approximate Values for Velocity, Density and Total Temperature Sensitivities on Hot Wire Anemometer Results; AIAA-86-0506

3. Bobbitt, Percy J.: Instrumentation Advances for Transonic Testing. Presented at the Transonic Symposium, NASA Langley Research Center, Hampton, Virginia, April 19-21, 1988. (To be published in NASA CP-3020, 1989.)

4. G. S. Jones, P. C. Stainback; A New Look At Wind Tunnel Flow Quality for Transonic Flows; SAE-88-1452

5. ACQUIRE 1.2, issue 2, modification 0, March 23, 1988; System Operating Manual OM0022 System Reference Manual OM0023 Issue A (September 1986, with Addendum February 1988); Data Laboratories Limited, 28 Wates Way, Mitcham Surrey. CR4 4HR England. Telephone 01-640-5321.

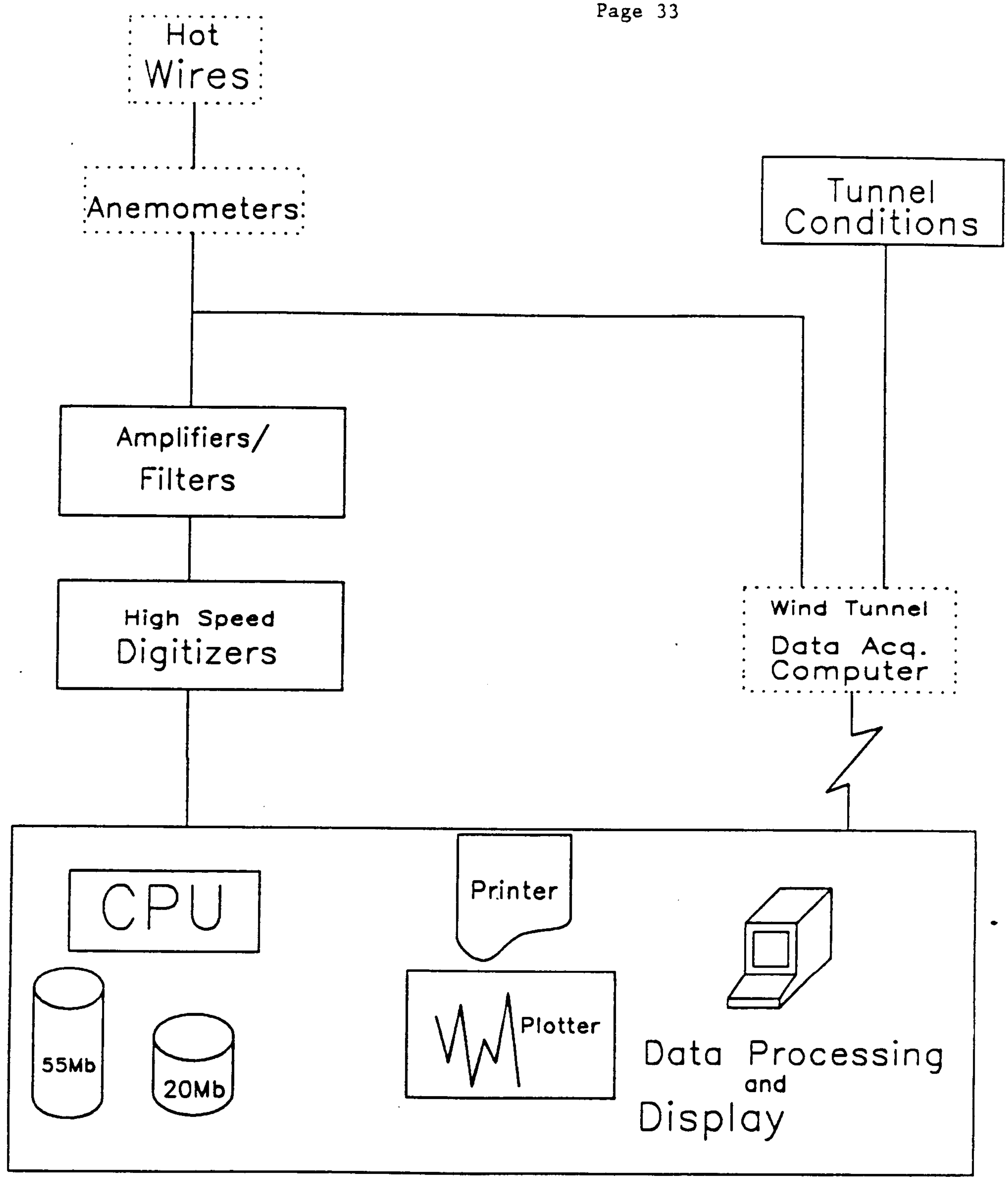

Figure 1. DDAS System Block Diagram

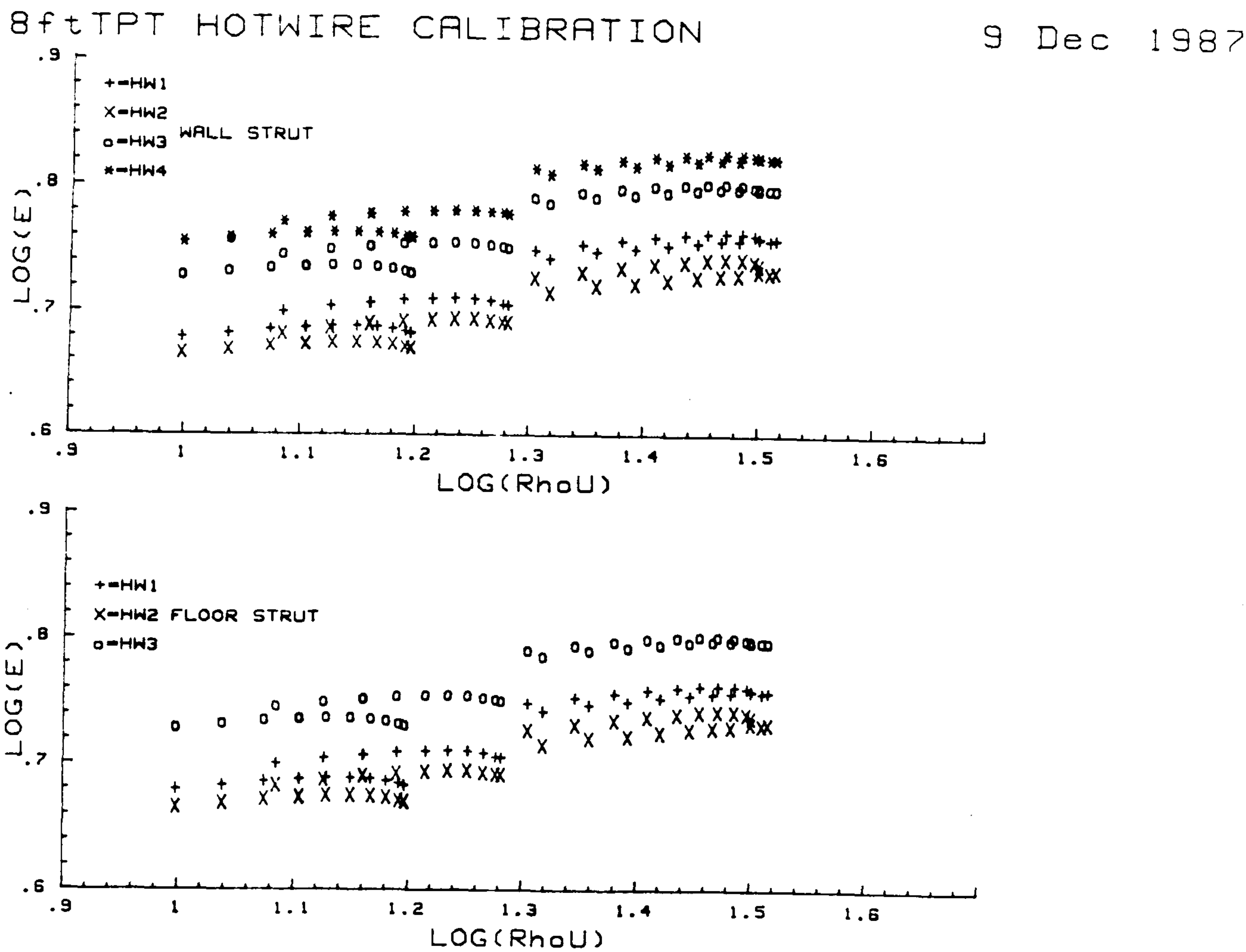

Figure 2. Plot: hotwire voltage vs. mass flow

ORIGINAL PAGE IS
OF POOR QUALITY

8FT TEST 934 - DATA REDUCTION - OBSERVATION # 11

TEST 934
RUN 17 LOG FILE 934REDUCE
POINT 8

	TUNNEL CONDITIONS	LOCAL WALL PROBE CONDITIONS	LOCAL FLOOR PROBE CONDITIONS
Mach	.8001725		
Reynolds No.	9.79489233333		
Pt	709.55	708.603433333	710.075833333
Ps	465.4	477.4245	475.0303
Tt	80.3028133333	540.302813333	540.302813333
Velocity	858.406598442	832.201491334	839.235011328
Density	.018230240804	.0185583708825	.0185028169039
LOG(RhoU)		1.18876833942	1.19112144477
MEAN(HW1)		4.82403933333	5.75614
MEAN(HW2)		4.67863933333	4.93762666667
MEAN(HW3)		5.40758666667	5.3251
S(U) (HW1)		.0801895340403	.0711717036965
S(Rho)(HW1)		.246631035249	.151745706072
S(To) (HW1)		-.371686324654	-.217170651926
S(U) (HW2)		.0815796208448	.0114987894883
S(Rho)(HW2)		.22379374977	.179632925078
S(To) (HW2)		-.798782032582	-.724845767689
S(U) (HW3)		.0757205995021	.0298907458578
S(Rho)(HW3)		.231703194152	.194507190167
S(To) (HW3)		-.523546487225	-.427212086868
u'/U (rms)		.0132928317592	-9.99999999999E+6
p'/P (rms)		.00400499662361	-9.99999999999E+6
to'/To (rms)		.000224062285135	-9.99999999999E+6
R(RhoU)		-.997767275712	-9.99999999999E+6
R(UTO)		.872062983427	-9.99999999999E+6
R(RhoTO)		-.875770964388	-9.99999999999E+6
M'/M		.00930062427607	-9.99999999999E+6
P'/P		.00725524205695	-9.99999999999E+6

Figure 3. Observation Report

ORIGINAL PAGE IS
OF POOR QUALITY

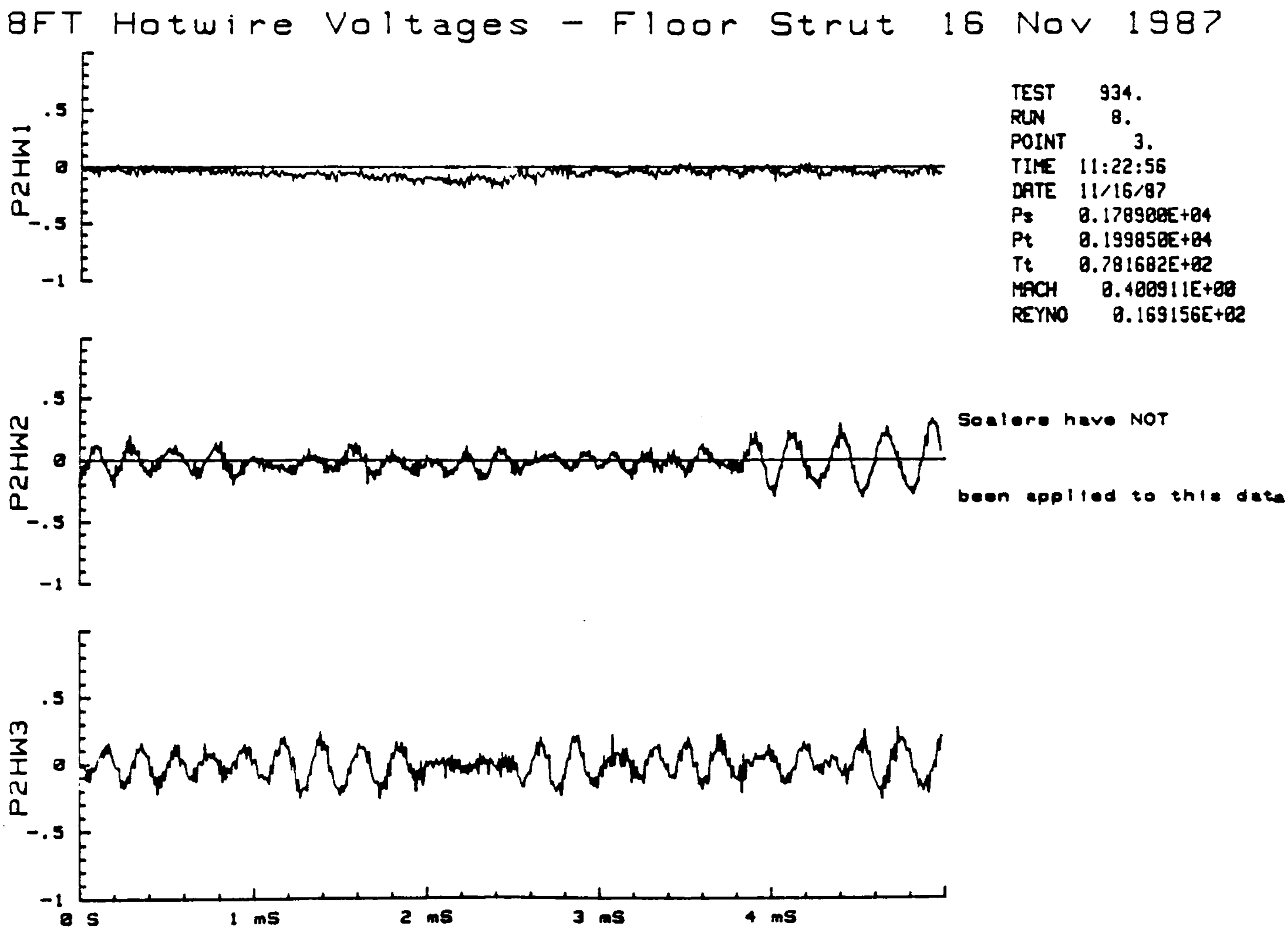

Figure 4. Plot: Waveforms - Floor strut

ORIGINAL PAGE IS
OF POOR QUALITY

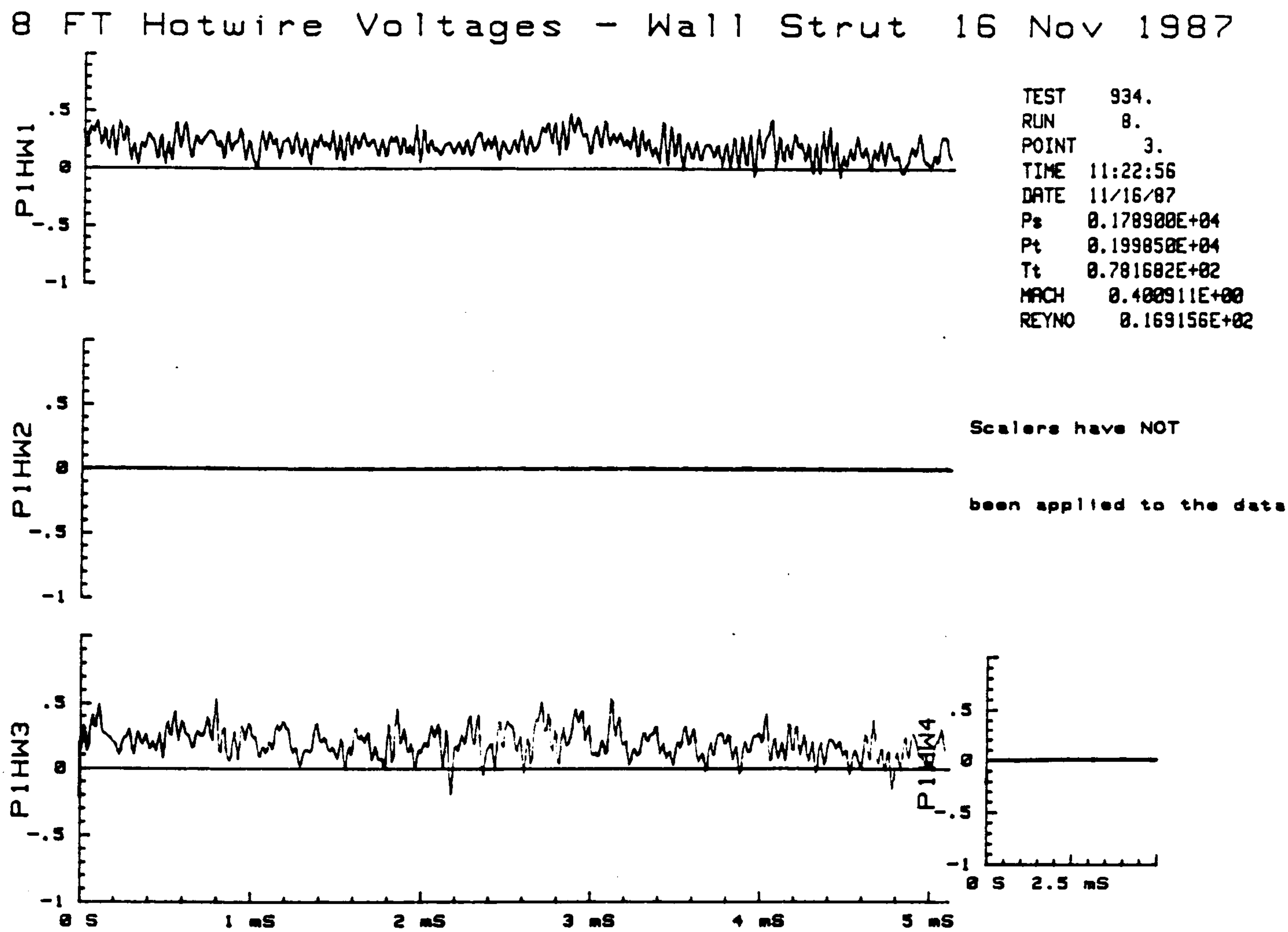

Figure 5. Plot: Waveforms - Wall strut

Pkt$(1)	<stx>	(ignored)	
(2)	TEST		
(3)	RUN		
(4)	POINT		
(5)	TIME		
(6)	DATE		
(7)	Ps	tunnel static pressure (psf)	
(8)	Pt	tunnel total pressure (psf)	
(9)	Tt	Tunnel total temperature (deg F)	
(10)	Mach number		
(11)	Reynolds number (per chord foot)		
(12)	PtS1	Strut 1 (Wall)	total pressure (psf)
(13)	PsS1		static pressure (psf)
(14)	TtS1		total temperature (deg F)
(15)	PtS2	Strut 2 (Floor)	total pressure (psf)
(16)	PsS2		static pressure (psf)
(17)	TtS2		total temperature (deg F)
(18)	PtS3	Strut 3 (Unused)	
(19)	PsS3		
(20)	TtS3		
(21)	P1HW1	Strut 1 Hot wire	1 mean voltage
(22)	P1HW2		2 mean voltage
(23)	P1HW3		3 mean voltage
(24)	P1HW4		4 mean voltage
(25)	P2HW1	Strut 2 Hot wire	1 mean voltage
(26)	P2HW2		2 mean voltage
(27)	P2HW3		3 mean voltage
(28)	P3HW1	Strut 3 Hot wire	1 mean voltage
(29)	P4HW1	Strut 4 Hot wire	1 mean voltage
(30)	P5HW1	Strut 5 Hot wire	1 mean voltage
(31)	Kulite1	Microphone RMS voltage	
(32)	2		
(33)	3		
(34)	4		
(35)	5		
(36)	6		
(37)	HW1-4GAIN	Gain code representing instrument gain	
(38)	HW5GAIN		
(39)	HW6GAIN	Wires 5, 6, and 7 are on probe 2,	
(40)	HW7GAIN	wires 1, 2 and 3	
(41)	Kulite1GAIN	Gain code representing instrument gain	
(42)	2		
(43)	3		
(44)	4		
(45)	5		
(46)	6		
(47)	<ETX>	(ignored)	

Table 1. Fluctuating Data File Name Format

Fluctuating Data File Name Format

character	1	
R	Real time digitized data	
P	Playback (FM tape) digitized data	
V	Velocity ratio - computed data	
D	Density ratio - computed data	
T	Temperature ratio - computed data	
character	2	
A	"A" 3-wire probe (wall strut - test 934)	
B	"B" 3-wire probe (floor strut - test 934)	
character	34	
rr	run number (00-99)	
character	56	
pp	point number(00-99)	
character	78	
cc	channel number (00-07)	
1		
character	90	
ss	sequence number - if any assigned	
example:	RA171203	real time, probe A, run 17, point 12, no sequence number assigned

The ACQUIRE software module MODUSR2 has been written to utilize the fluctuating data file naming conventions described above.

The ACQUIRE software module MODGEN (provided by Data Laboratories) was modified to <u>not</u> automatically add a sequence number in column 9 if not necessary to differentiate between two files with the same name (in columns one through eight).

Table 2. Fluctuating Data File Name Format

APPENDICES

APPENDIX A. Program Listings

This appendix contains the DDAS program listings for all the application specific code to acquire, process, display, store and transmit the hot wire data.

The program is separated into two modules: MODUSR1 provides general utility functions. MODUSR2 provides all hot wire specific functions.

```
26042 Modusrl:SUB Modusrl(INTEGER Routine,Code,OPTIONAL REAL Rvar,Rvar$)
26043   ! filename MODUSR1
26044   ! issue 1
26045   ! mod 0
26046   ! date 01 Oct 1987
26047   ! mod 1
26048   ! date 22 Dec 1987
26049   ! programmer S. CLUKEY, Vigyan Research Assoc.
26050   !
26051   ! This program becomes a part of "ACQUIRE", and provides additional
26052   ! utility functions. As the need for additional functions increases,
26053   ! so will the functions implemented in this program.
26054   !
26055 OPTION BASE 1
26056   COM /Arr/ INTEGER Arrowvar(*)
26057   COM /Cross/ Crval(*),INTEGER Crvar(*),Crtab(*),Crmptr
26058   COM /Curs/ Caddrx(*),INTEGER Cincr(*),Cpos(*),Scursor(*),Cactive,Cpixmin
(*),Cpixmax(*),Cflag
26059   COM /Error/ INTEGER Errf,Errtype
26060   COM /Inp/ Inpstr$,INTEGER Ilinex,Iliney,Ilinefd,Insertf,Inptype,Inpstr_p
os,Inpx,Inpy,Inpfd
26061   COM /Input/ Tinput$,INTEGER Kposx,Kposy,Quitcode
26062   COM /Keys/ Keylab$(*),INTEGER Keymap(*),Keymenu,Keyincr,Okeytype,Okeymen
u
26063   COM /Localvar/ Lvar$
26064   COM /Mem/ Bsw$(*),Sw$(*),Sw(*),INTEGER X(*),Isw(*),Memlenb(*),Memlenu(*)
,Memstartb(*),Memstartu(*),Nummem,Maxnmem,Memmaxl,Tnmem
26065   COM /Menut/ Menulab$,Mvar$(*),Mlit$,Mvar(*),INTEGER Mivar(*),Menutab(*),
Keycode(*),Sysmod(*),Nummitems,Nummods,Numkeys,Menuptr,Xindex,Yindex
26066   COM /Param/ Mval$,Mval(*),INTEGER Mstack(*),Mvallist(*),Mstackptr,Mvalpt
r,Mvalstrptr
26067   COM /Rcl/ Rclstr$,INTEGER Rclstr_ptr(*),Rclptr,Rclnum
26068   COM /Screen/ INTEGER Garray(*),Ctextx(*),Ctexty,Ctextn,Ctextw,Ocpos(*),O
smptr,Ostype,Updatetype
26069   COM /Scrtab/ Cmd_exec$,INTEGER Morefl,Smtab(*),Smptr,Smnum,Smvptr
26070   COM /Scrvars/ INTEGER Crtvar(*),Pwidth,Endline,Nextl,Topline
26071   COM /State/ Status$,INTEGER Statx,Staty,Statfd,Statusp,Conffl
26072   COM /Sysvar/ INTEGER Stype,Schg,Sysrec,Sysinit,Seqrunfl,Sysflags,Prdev,G
rtype
26073   COM /Trvars/ Trval(*),INTEGER Trmem(*),Trmembit(*),Trcrt(*),Tryytr(*),Tr
active,Tron,Troverlay,Trflags,Trlabel,Strace,Numtr
26074 !
26075 !
26076 !
26077 !
26078   COM /Plot1/ INTEGER Titlexcoor,Titleycoor,Titlesize,Ntrace,T_chan(*),Lty
pe(*),Secondc(*),Titlep,Titlepc,Ploth,Plname$,Plotstring$
26079   COM /Plot2/ REAL Xstart(*),Xend(*),Ystart(*),Yend(*),Xorigin(*),Yorigin(
*),Xmin(*),Xmax(*),Ymin(*),Ymax(*),Xtic(*),Ytic(*)
26080   COM /Plot3/ INTEGER Xlabelp(*),Xlabelpc(*),Ylabelp(*),Ylabelpc(*),Commen
tp(16,8),Commentpc(16,8),Commentsize(*),Npoint(*),Labelsize(*)
26081   COM /Plot4/ INTEGER Comcount(*),Tabcount(*),Eventbit(*),Tabvalp(*),Tabva
lpc(*),REAL Tabvalx(*),Comxcoor(*),Comycoor(*)
26082   COM /Plot5/ INTEGER Created,Noofpen,Plotdev,Plgrid(*),Tlabp(*),Tlabpc(*)
,Cur_trace,Plzref(*),Xtype(*),Ytype(*)
26083 !
26084 !
26085 !
26086 Usercom:      !
26087 COM /Usr1/ Cat_array$(800)[80],Fil_nam$[10],Fil_grp$[10],Sym_tbl$(4)[1],Nu
```

```
m_obs_plotted
26088 COM /Usr1/ From_disk$[10],To_disk$[10],File_grp$[10],File_nam$[10]
26089 COM /Usr2/ Sfn$,REAL Hwsens(*),Sensinv(*),Mean(*),Mean_param(*),Enorm(*),Stddev_param(*),Max_param(*),Min_param(*),Vo_param(*),Vs_param(*)
26090 COM /Usr2/ Gain_code_14(*),Gain_code_57(*)
26091 COM /Usr2/ C_names$(*),P_c(*)
26092 COM /Usr2/ Log_fn$,Data_set_title$,Logged_var_name$(*),Obs_rec(*),Pkt_sc,Pkt$(*),Pkt_avg(*),Num_avgs,Max_vars,Rcvd_vars
26093 COM /Usr2/ Subfile_names$(*),Subfile_chartst(*),Initial_obs,Ending_obs,Num_obs_recd,Num_obs_printed,Max_obs_rec,Tag_pkt$(*)
26094 COM /Usr2/ Initial_probe,Ending_probe
26095 COM /Usr2/ A_fn$,A_set_title$,A_var_name$(*),A_rec(*)
26096 COM /Usr2/ A_subfile_names$(*),A_sub_chartst(*)
26097 COM /Usr2/ B_fn$,B_set_title$,B_var_name$(*),B_rec(*)
26098 COM /Usr2/ B_subfile_names$(*),B_sub_chartst(*)
26099 COM /Usr2/ C_fn$,C_set_title$,C_var_name$(*),C_rec(*)
26100 COM /Usr2/ C_subfile_names$(*),C_sub_chartst(*)
26101 COM /Usr2/ Hw_rms(*)
26102 INTEGER V(3)
26103 INTEGER I
26104 DIM V$[30]
26105 SELECT Routine
26106 CASE 1  ! init pass !
26107    RESTORE Menulist
26108    LOOP
26109        READ V$
26110    EXIT IF V$="***"
26111        READ V(*)
26112        CALL Chkmitem(V(*),V$)
26113    END LOOP
26114 CASE 2   ! init pass 2
26115    RESTORE Keylist
26116    LOOP
26117        Errfl=0
26118        Errtype=1
26119        READ V(1)
26120    EXIT IF V(1)<0
26121        READ V(2)
26122        CALL Chkkey(V(1),V(2))
26123    END LOOP
26124  !
26125 Menulist: !
26126     !       LABEL,Function,Flag1,Flag2
26127    DATA "UTILITY",4000,0,6
26128    DATA "PLOT UTILITYS",4001,0,6
26129    DATA "PLOT EJECT",4002,0,262
26130    DATA "TAG PLOT",4003,0,262
26131    DATA "TAG PICTURE",4004,0,262
26132    DATA "PLOT LOG_E",4005,0,262
26133    DATA "PICTURE LOG_E",4006,0,262
26134    DATA "FILE UTILITYS",4020,0,6
26135    DATA "CAT GROUP",4021,8704,260
26136    DATA "PURGE",4022,0,6
26137    DATA "PURGE GROUP",4023,8704,390
26138    DATA "PURGE FILE",4024,8704,390
26139    DATA "FILE COPY",4025,0,6
26140    DATA "FROM DISK",4026,8704,22
26141    DATA "TO DISK",4027,8704,22
26142    DATA "FILE GROUP",4028,8704,22
26143    DATA "COPY FILES",4029,8704,394
```

```
26144     DATA "MOVE FILES",4030,8704,394
26145     DATA "***"
26146  !
26147 Keylist: !
26148     DATA 0,4000
26149     DATA 100,4001
26150     DATA 110,4002
26151     DATA 120,4003
26152     DATA 130,4004
26153     DATA 140,4005
26154     DATA 150,4006
26155     DATA 200,4020
26156     DATA 210,4021
26157     DATA 220,4022
26158     DATA 221,4023
26159     DATA 222,4024
26160     DATA 240,4025
26161     DATA 241,4026
26162     DATA 242,4027
26163     DATA 243,4028
26164     DATA 245,4030
26165     DATA 247,4029
26166     DATA -1,-1
26167  !
26168 CASE 3      !RUN TIME INITIALIZATION
26169     Fil_nam$=""
26170     Fil_grp$=""
26171     RESTORE Symbols
26172     READ Sym_tbl$(*)
26173 Symbols:        !
26174     DATA "+"
26175     DATA "X"
26176     DATA "o"
26177     DATA "*"
26178  !
26179 CASE 4          ! Power on initializations:
26180     Num_obs_plotted=0
26181  !
26182 CASE ELSE
26183 Usercode:!
26184     SELECT Code
26185     CASE 4002                           !PLOT EJECT
26186         SELECT Routine
26187         CASE 31
26188             OUTPUT Plotdev;"PG"
26189             Num_obs_plotted=0 !RESET OBSERVATIONS POINTER for Function 4005
26190             Num_obs_printed=0 !RESET OBSERVATIONS PRINTED for Function 4107
26191         END SELECT
26192     CASE 4003,4004                      !PLOT TAG   PICTURE TAG
26193         SELECT Routine
26194         CASE 31
26195             Errf=0
26196             IF Errf=0 AND Code=4003 THEN PLOTTER IS Plotdev,"HPGL"
26197             IF Errf=0 THEN
26198                 OFF TIMEOUT 7
26199                 Yminoff=0
26200                 Ymoff=1
26201                 X_gdu_min=0
26202                 X_gdu_max=100*RATIO
26203                 Y_gdu_min=0
```

```
26204                   Y_gdu_max=100
26205                   DEG
26206                   PEN 1
26207                   IF Titlesize<=.8 THEN Titlesize=4
26208                   CSIZE .65*Titlesize*Ymoff,.65*RATIO*.5
26209                   LORG 3
26210                   LDIR 0
26211                   VIEWPORT X_gdu_min,X_gdu_max,Y_gdu_min*Ymoff+Yminoff,Y_gdu_
max*Ymoff+Yminoff
26212                   WINDOW X_gdu_min,X_gdu_max,Y_gdu_min,Y_gdu_max
26213                   MOVE X_gdu_max-(.20*X_gdu_max),Y_gdu_max-10
26214                   FOR I=1 TO 10
26215                       SELECT I
26216                       CASE 1,2,5!TEST, RUN, DATE only
26217             !#####        LABEL Logged_var_name$(I)&"  "&Tag_pkt$(I)
26218                       END SELECT
26219                   NEXT I
26220                   FOR Probe=Initial_probe TO Ending_probe
26221                       SELECT Probe
26222                       CASE 1
26223                           PEN 1
26224                           LABEL "WALL PROBE"
26225                       CASE 2
26226                           PEN 2
26227                           LABEL "FLOOR PROBE"
26228                       END SELECT
26229                   NEXT Probe
26230               END IF
26231               PENUP
26232               VIEWPORT X_gdu_min,X_gdu_max,Y_gdu_min,Y_gdu_max
26233               WINDOW X_gdu_min,X_gdu_max,Y_gdu_min,Y_gdu_max
26234               MOVE X_gdu_max,Y_gdu_max
26235               PLOTTER IS CRT,"INTERNAL"
26236           END SELECT
26237 !
26238 !
26239 Plot_log_e: !
26240     CASE 4005,4006                          !PLOT LOG(E) vs LOG(Rho*U)
26241         SELECT Routine
26242         CASE 31
26243             IF Num_obs_plotted=0 THEN  !Get the axis plotted
26244                 IF Code=4005 THEN OUTPUT Plotdev;"PG"!Eject old plot
26245                 Mvar$(3)="RhoU"
26246                 CALL Routine(22,618)   !PLOT NAME=RhoU
26247                 CALL Routine(31,616)   !LOAD PLOT
26248                 IF Code=4005 THEN
26249                     CALL Routine(31,615)!PLOT PICTURE
26250                 ELSE
26251                     CALL Routine(31,612)!REDRAW PICTURE
26252                 END IF
26253             END IF
26254             IF Code=4005 THEN
26255                 PLOTTER IS Plotdev,"HPGL"
26256                 OFF TIMEOUT 7
26257             ELSE
26258                 PLOTTER IS CRT,"INTERNAL"
26259             END IF
26260             Yminoff=0
26261             Ymoff=1
26262             IF Titlesize<=.8 THEN Titlesize=4
```

```
26263           CSIZE .65*Titlesize*Ymoff,.65*RATIO*.5
26264           LORG 5                          ! Center the symbol
26265 !
26266           FOR Probe=Initial_probe TO Ending_probe
26267               PEN Probe
26268               FOR This_obs=Initial_obs TO Ending_obs
26269                   SELECT Probe
26270                   CASE 1
26271                       Num_wires=4
26272                   CASE 2
26273                       Num_wires=3
26274                   CASE ELSE
26275                       Num_wires=1
26276                   END SELECT
26277                   VIEWPORT Xstart(1),Xend(1),Ystart(1)*Ymoff+Yminoff,Yend
(1)*Ymoff+Yminoff
26278                   WINDOW Xmin(1),Xmax(1),Ymin(1),Ymax(1)
26279       !######     FOR Wire=1 TO Num_wires
26280                   FOR Wire=1 TO 1
26281                       SELECT Probe
26282                       CASE 1
26283                           Pressure=A_rec(4,This_obs)
26284                           X_val=MAX(-E6,A_rec(13,This_obs))
26285                           Y_val=MAX(-E6,A_rec(21+Wire,This_obs))
26286                       CASE 2
26287                           Pressure=B_rec(4,This_obs)
26288                           X_val=MAX(-E6,B_rec(13,This_obs))
26289                           Y_val=MAX(-E6,B_rec(21+Wire,This_obs))
26290                       END SELECT
26291                       MOVE X_val,Y_val
26292                       SELECT Pressure
26293                       CASE 700. TO 720.
26294                           LABEL Sym_tbl$(1)
26295                       CASE 850. TO 880.
26296                           LABEL Sym_tbl$(2)
26297                       CASE 1400. TO 1500.
26298                           LABEL Sym_tbl$(3)
26299                       CASE 1700. TO 1800.
26300                           LABEL Sym_tbl$(4)
26301                       END SELECT
26302                   NEXT Wire
26303               NEXT This_obs
26304               IF Probe=1 THEN
26305                   MOVE 1.,.6639
26306                   DRAW 1.54,.77832
26307                   LINE TYPE 4
26308                   PEN Probe+3
26309                   MOVE 1.,.67889
26310                   DRAW 1.58,.72076
26311                   LINE TYPE 1
26312               END IF
26313           NEXT Probe
26314           PLOTTER IS CRT,"INTERNAL"
26315       END SELECT
26316   CASE 4021       !CAT A SELECT GROUP OF FILES
26317       SELECT Routine
26318       CASE 21
26319           Mvar$(3)=Fil_grp$
26320       CASE 22
26321           Fil_grp$=Mvar$(3)
```

```
26322         CASE 31
26323             CAT;SELECT Fil_grp$
26324         END SELECT
26325     CASE 4023      !PURGE A SELECT GROUP OF FILES
26326         SELECT Routine
26327         CASE 21
26328             Mvar$(3)=Fil_grp$
26329         CASE 22
26330             Fil_grp$=Mvar$(3)
26331         CASE 31
26332             CAT TO Cat_array$(*);SELECT Fil_grp$,NO HEADER,COUNT Num_in_grp
26333             Fil_grp$=""
26334             FOR I=1 TO Num_in_grp
26335                 Fil_nam$=Cat_array$(I)[1;10]
26336                 PURGE Fil_nam$
26337             NEXT I
26338         END SELECT
26339     CASE 4024      !PURGE A FILE
26340         SELECT Routine
26341         CASE 21
26342             Mvar$(3)=Fil_nam$
26343         CASE 22
26344             Fil_nam$=Mvar$(3)
26345         CASE 31
26346             IF LEN(Fil_nam$)>0 THEN PURGE Fil_nam$
26347             CALL Pline(0,"FILE "&Fil_nam$&" HAS BEEN PURGED")
26348         END SELECT
26349     CASE 4026      !FROM DISK
26350         SELECT Routine
26351         CASE 21
26352             Mvar$(3)=From_disk$
26353         CASE 22
26354             From_disk$=Mvar$(3)
26355         END SELECT
26356     CASE 4027      !TO DISK
26357         SELECT Routine
26358         CASE 21
26359             Mvar$(3)=To_disk$
26360         CASE 22
26361             To_disk$=Mvar$(3)
26362         END SELECT
26363     CASE 4028      !FILE GROUP
26364         SELECT Routine
26365         CASE 21
26366             Mvar$(3)=File_grp$
26367         CASE 22
26368             File_grp$=Mvar$(3)
26369         END SELECT
26370     CASE 4029,4030 !COPY OR MOVE A SELECT GROUP OF FILES
26371                    !( Copy leaves the files on both from and to disks.)
26372                    !( Move leaves the files only on the to disk, by purging
26373                    !  the files successfully copied from the 'from' disk.)
26374         SELECT Routine
26375         CASE 21
26376             Mvar$(3)=File_grp$
26377         CASE 22
26378             File_grp$=Mvar$(3)
26379         CASE 31
26380             CALL Pline(0,"Selecting files. Please wait....")
26381             CAT From_disk$ TO Cat_array$(*);SELECT File_grp$,NO HEADER,COUN
```

```
T Num_in_grp
26382  !
26383           PRINTER IS PRT;WIDTH 108
26384           PRINT CHR$(12)          ! Form Feed
26385           PRINT Num_in_grp;" files have been selected for copying."
26386           PRINT
26387           PRINT " FROM";TAB(30);"TO"
26388           PRINT From_disk$;TAB(30);To_disk$
26389           PRINT
26390           FOR I-1 TO Num_in_grp
26391               PRINT TAB(20);Cat_array$(I)[1,10]
26392           NEXT I
26393           PRINT
26394           PRINT
26395           PRINT
26396           PRINTER IS CRT
26397  !
26398           Number_copied-0
26399           PRINTER IS PRT
26400           PRINT "The following files"&CHR$(27)&"&dD"&" HAVE BEEN COPIED "
&CHR$(27)&"&d@"&"to "&To_disk$
26401           PRINT
26402           PRINTER IS CRT
26403           FOR I-1 TO Num_in_grp
26404               ON ERROR GOTO 26414
26405               File_nam$-Cat_array$(I)[1;10]
26406               COPY File_nam$&From_disk$ TO File_nam$&To_disk$
26407               CALL Pline(0,"FILE"&CHR$(129)&" "&File_nam$&" "&CHR$(128)&"
HAS BEEN COPIED FROM DISK "&From_disk$&" TO DISK "&To_disk$)
26408               PRINTER IS PRT;WIDTH 108
26409               Number_copied-Number_copied+1
26410               Cat_array$(Number_copied)[1,10]-File_nam$
26411               PRINT "FILE"&CHR$(129)&" "&File_nam$&" "&CHR$(128)&"HAS BEE
N COPIED FROM DISK "&From_disk$&" TO DISK "&To_disk$
26412               PRINTER IS CRT
26413               GOTO 26420
26414               PRINT TAB(10);"File "&File_name$&" was NOT successfully cop
ied..."
26415               PRINT ERRM$
26416               PRINTER IS PRT
26417               PRINT TAB(10);"File "&File_name$&" was NOT successfully cop
ied..."
26418               PRINT ERRM$
26419               PRINTER IS CRT
26420           NEXT I
26421           ON ERROR CALL Error
26422           CALL Pline(0,"                "&CHR$(129)&" FILE COPYS COMPLETE
 "&CHR$(128)&"                  ")
26423  !
26424  !
26425           IF Code-4030 THEN  ! If a MOVE, then purge the origional file
26426               PRINTER IS PRT;WIDTH 108
26427               PRINT
26428               PRINT Number_copied;" FILES HAVE BEEN COPIED; PURGING ORIGI
ONALS"
26429               ON ERROR GOTO 26432
26430               FOR I-1 TO Number_copied
26431                   PURGE Cat_array$(I)[1,10]&From_disk$
26432               NEXT I
26433               ON ERROR CALL Error
```

```
26434                PRINT "    MOVE FILES  function is complete"
26435                PRINT
26436                PRINTER IS CRT
26437             END IF
26438          END SELECT
26439       END SELECT
26440 END SELECT
26441 SUBEND
26442 !
26443 !
```

```
24000 Modusr2:SUB Modusr2(INTEGER Routine,Code,OPTIONAL REAL R
var,Rvar$)
24002   ! filename MODUSR2
24004   ! issue 1
24006   ! mod 1
24007   ! date 28 Sept 1987
24008   !
24012   ! mod 6
24013   ! date 8 Apr 1988
24014   !
24016   ! programmer S. CLUKEY, Vigyan Research Assoc.
24017   !
24018   ! This routine logs mean value data received from another CPU
24019   ! via the HP-IB bus. This bus is SC=8, and the primary address is 01.
24020   ! This end is NOT System Controller!
24021   ! The variables are logged in a disc file that contains a header
24022   ! record, a variable names record, and 100 observation records.
24023   ! It is an ASCII file.
24024   !
24025 !!! This routine also computes coefficients of calibration for multiple
24026   ! 3 wire probes thru the use of multiple linear regression.
24027   ! Alternatively, the coefficients can be entered either by reading
24028   ! a coefficient file, or by manually keying in the coefficients -
24030   !
24031 !!! With the calibration coefficients, this routine can generate
24032   ! sensitivities for the designated probe for each observation
24033   ! previously logged.
24034   !
24035 !!! This routine can then apply the sensitivities to dynamic HOTWIRE data!
24036   ! which is automatically loaded:
24037   !
24038   ! in trace                   and returns:
24039   !          1  HOTWIRE 1          fluctuating velocity    (u)  in trace 4
24040   !          2  HOTWIRE 2          fluctuating density     (p)  in trace 5
24041   !          3  HOTWIRE 3          fluctuating temperature (To) in trace 6
24042   !
24043   !  These calculations are performed only between cursor positions
24044   !  - if - both cursors of trace 1  are active; or from the
24045   !   only cursor to the end of the memory.
24046   !  Otherwise, sensitivity coefficients are applied to the entire sample.
24047   !
24048   ! Traces 1, 2, and 3 are automatically loaded from disk files that were
24049   !  previously recorded using a naming convention defined here:
24050   !   If the log filename is of the format "XXX"
24051   !                   [where XXX is the test number]
24052   !      then the hotwire data disk file names would have the format:
24053   !                                          "MPXXYYZZnn"
24054   !                 [where     M  is the mode as follows:
24055   !                                  R - Real time digitization
24056   !                                  P - Playback  digitization
24057   !                            P  is the probe selection as follows:
24058   !                                  A - probe A
24059   !                                  B - probe B
24061   !                           XX  is the RUN number
24062   !                           YY  is the POINT number
24063   !                                  of the data most recently logged,
24064   !                                  and therefore most likely to define
24065   !                                  related mean conditions, and will
24066   !                                  contain calculated sensitivities,
24067   !                                  gains of the fluctuating voltages, etc
```

```
24069   !                           ZZ  is the digitizer channel number (this
24070   !                                  naming convention assumes a 1-to-1
24071   !                                  relationship:
24072   !                                                    probe "A"
24073   !                                              chan 1 - hotwire 1
24074   !                                                   2           2
24075   !                                                   3           3
24076   !                                                     probe "B"
24077   !                                                   5           1
24078   !                                                   6           2
24079   !                                                   7           3
24080   !                           nn is the "serial number" automatically
24081   !                                 applied (unfortunately) by ACQUIRE,
24082   !                                 and must be ignored.
24083   !
24084   !  NOTE:  the hotwire data disk file name is defined by ACQUIRE function
24085   !         "FILENAME".
24086   !
24090   !
24091 !!!  Using the sensitivities, mean voltages, and gains from the "A" or "B"
 probe file,
24092   !  traces 4, 5, and 6 are scaled and created to represent the ratios:
24093   !   trace 4 - velocity fluctuations / mean velocity
24094   !   trace 5 - density fluctuations / mean density
24095   !   trace 6 - temperature fluctuations / mean temperature
24096   !
24097   !   The mean value is removed from these ratios, and an rms value of
24098   !     the three ratios is stored as variables 47, 48, and 49 of the
24099   !     appropriate probe file - "A" or "B".
24100   !
24101 !!!  Additionally, the ratios above can be retrieved and used to compute
24102   !     correlations between velocity, density, and temperature fluctuatio
ns,
24103   !     and then, using these correlations, go on and compute massflow and
24104   !     pressure fluctuations.  These results are stored much as the ratio
s
24105   !     above are stored.
24106   !
24107 OPTION BASE 1
24137   COM /Arr/ INTEGER Arrowvar(*)
24138   COM /Cross/ Crval(*),INTEGER Crvar(*),Crtab(*),Crmptr
24139   COM /Curs/ Caddrx(*),INTEGER Cincr(*),Cpos(*),Scursor(*),Cactive,Cpixmin
(*),Cpixmax(*),Cflag
24140   COM /Error/ INTEGER Errf,Errtype
24141   COM /Genvar/ Filename$(*),Discdev$,INTEGER Discmap(*),Recordnum
24143   COM /Inp/ Inpstr$,INTEGER Ilinex,Iliney,Ilinefd,Insertf,Inptype,Inpstr_p
os,Inpx,Inpy,Inpfd
24144   COM /Input/ Tinput$,INTEGER Kposx,Kposy,Quitcode
24145   COM /Keys/ Keylab$(*),INTEGER Keymap(*),Keymenu,Keyincr,Okeytype,Okeymen
u
24146   COM /Localvar/ Lvar$
24147   COM /Mem/ Bsw$(*),Sw$(*),Sw(*),INTEGER X(*),Isw(*),Memlenb(*),Memlenu(*)
,Memstartb(*),Memstartu(*),Nummem,Maxnmem,Memmax1,Tnmem
24148   COM /Menut/ Menulab$,Mvar$(*),Mlit$,Mvar(*),INTEGER Mivar(*),Menutab(*),
Keycode(*),Sysmod(*),Nummitems,Nummods,Numkeys,Menuptr,Xindex,Yindex
24149   COM /Param/ Mval$,Mval(*),INTEGER Mstack(*),Mvallist(*),Mstackptr,Mvalpt
r,Mvalstrptr
24150   COM /Rcl/ Rclstr$,INTEGER Rclstr_ptr(*),Rclptr,Rclnum
24151   COM /Screen/ INTEGER Garray(*),Ctextx(*),Ctexty,Ctextn,Ctextw,Ocpos(*),O
smptr,Ostype,Updatetype
```

```
24152   COM /Scrtab/ Cmd_exec$,INTEGER Morefl,Smtab(*),Smptr,Smnum,Smvptr
24153   COM /Scrvars/ INTEGER Crtvar(*),Pwidth,Endline,Nextl,Topline
24154   COM /State/ Status$,INTEGER Statx,Staty,Statfd,Statusp,Conffl
24155   COM /Sysvar/ INTEGER Stype,Schg,Sysrec,Sysinit,Seqrunfl,Sysflags,Prdev,G
rtype
24156   COM /Trvars/ Trval(*),INTEGER Trmem(*),Trmembit(*),Trcrt(*),Tryytr(*),Tr
active,Tron,Troverlay,Trflags,Trlabel,Strace,Numtr
24157 !
24158 !
24159 !
24160 Usercom:COM /Usr2/ Sfn$[20],REAL Hwsens(3,3),Sensinv(3,3),Mean(3),Mean_par
am(3),Enorm(3),Stddev_param(3),Max_param(6),Min_param(6),Vo_param(6),Vs_param(6)
24161 COM /Usr2/ Gain_code_14(17),Gain_code_57(8)
24162 COM /Usr2/ C_names$(10)[10],P_c(3,10) ! P_c is the coefficient file
24163 COM /Usr2/ Log_fn$[20],Data_set_title$[80],Logged_var_name$(50)[10],Obs_re
c(50,300),Pkt_sc,Pkt$(47)[80],Pkt_avg(45),Num_avgs,Max_vars,Rcvd_vars
24164 COM /Usr2/ Subfile_names$(20)[10],Subfile_chartst(20),Initial_obs,Ending_o
bs,Num_obs_recd,Num_obs_printed,Max_obs_rec,Tag_pkt$(10)[80]
24165 COM /Usr2/ Initial_probe,Ending_probe
24166 !
24167 !
24168 !       The A, B, and C files below contain the "calculated" variables
24169 !           related to: A) wall strut, B) floor strut, and C) Kulites
24170 !           (and 'other' 'big end' wires)
24171 !
24172 COM /Usr2/ A_fn$[20],A_set_title$[80],A_var_name$(50)[10],A_rec(50,300)
24173 COM /Usr2/ A_subfile_names$(20)[10],A_sub_chartst(20)
24174 !
24175 COM /Usr2/ B_fn$[20],B_set_title$[80],B_var_name$(50)[10],B_rec(50,300)
24176 COM /Usr2/ B_subfile_names$(20)[10],B_sub_chartst(20)
24177 !
24178 COM /Usr2/ C_fn$[20],C_set_title$[80],C_var_name$(50)[10],C_rec(50,300)
24179 COM /Usr2/ C_subfile_names$(20)[10],C_sub_chartst(20)
24180 !
24181 COM /Usr2/ Hw_rms(3)
24182 !
24183 INTEGER Xch(6),Xpt(6),Nch(6),Npt(6),Xchmax,Xptmax,Ymax,Xchmin,Xptmin,Ymin,
J,Wtr,Wmem(6),Xsch(3),Xspt(3),Ip,Wire
24184 INTEGER Routinl
24185 REAL Temp,Wmark(6,2),Vo(6),Vs(6),Sum_param(3),Sumsq_param(3),Vdp(3),Wire_g
ain(4)
24186 INTEGER V(3)
24187 DIM V$[30],File_comments$[80]
24188 Routinl=Routine
24189 SELECT Routine
24190 CASE 1  ! init pass !
24191    RESTORE Menulist
24192    LOOP
24193        READ V$
24194    EXIT IF V$="***"
24195        READ V(*)
24196        CALL Chkmitem(V(*),V$)
24197    END LOOP
24198 CASE 2   ! init pass 2
24199    RESTORE Keylist
24200    LOOP
24201        Errfl=0
24202        Errtype=1
24203        READ V(1)
24204    EXIT IF V(1)<0
```

```
24205         READ V(2)
24206         CALL Chkkey(V(1),V(2))
24207     END LOOP
24208   !
24209 Menulist: !
24210      !        LABEL,Function,Flag1,Flag2
24211     DATA "HOTWIRE MENU",4100,0,6
24212     DATA "COEF FILENAME",4101,8704,22
24213     DATA "LOAD COEFS",4102,0,262
24214     DATA "ENTER COEFS",4103,0,388
24215     DATA "CALC VEL etc",4104,0,262
24216     DATA "LOG FILENAME",4105,8704,22
24217     DATA "LOAD LOGFILE",4106,0,262
24218     DATA "PRNT LOGFILE",4107,0,262
24219     DATA "LOG DATA POINT",4108,0,390
24220     DATA "LOG DATA",4109,0,6
24221     DATA "SAMPLES TO AVG",4110,24608,22
24222     DATA "COMPUTE SENS.",4111,0,390
24223     DATA "INITIAL OBS",4112,24608,22
24224     DATA "ENDING OBS",4113,24608,22
24225     DATA "HOTWIRE CALC",4120,0,6
24226     DATA "GET COEF",4122,0,6
24227     DATA "COMPUTE COEFS",4123,0,390
24228     DATA "STORE COEFS",4124,0,262
24229     DATA "CODE TO GAINS",4125,0,388
24230     DATA "FILE TRANSFERS",4126,0,6
24231     DATA "LOGFILE TO PC",4127,0,388
24232     DATA "Remake Probe",4128,0,388
24233     DATA "SELECTOR",4129,0,6
24234     DATA "INITIAL PROBE",4130,24608,22
24235     DATA "ENDING PROBE",4131,24608,22
24236     DATA "COMPUTE R etc",4132,0,388
24237     DATA "EDIT COEFS",4133,0,388
24238     DATA "***"
24239   !
24240 Keylist: !
24241     DATA 0,4100         !HOTWIRE MENU
24242     DATA 300,4129       !SELECTOR
24243     DATA 320,4112       !INITIAL OBS
24244     DATA 330,4113       !ENDING  OBS
24245     DATA 340,4130       !INITIAL PROBE
24246     DATA 350,4131       !ENDING PROBE
24247     DATA 370,4110       !SAMPLES TO AVG
24248     DATA 400,4120       !HOTWIRE CALC
24249     DATA 420,4122       !GET COEFS
24250     DATA 421,4101       !COEF FILENAME
24251     DATA 422,4102       !LOAD COEFS
24252     DATA 423,4103       !ENTER COEFS
24253     DATA 424,4133       !EDIT  COEFS
24254     DATA 425,4123       !COMPUTE COEFS
24255     DATA 427,4124       !STORE COEFS
24256     DATA 430,4111       !COMPUTE SENS
24257     DATA 440,4104       !CALC VEL etc
24258     DATA 450,4132       !COMPUTE E etc
24259     DATA 460,4125       !XLATE GAIN CODES
24260     DATA 470,4128       !Remake Probe Data
24261     DATA 500,4109       !LOG DATA
24262     DATA 510,4105       !LOG FILENAME
24263     DATA 520,4106       !LOAD LOGFILE
24264     DATA 530,4107       !PRNT LOGFILE
```

```
24265     DATA 570,4108      !LOG DATA POINT
24266     DATA 600,4126      !FILE TRANSFERS
24267     DATA 610,4127      !LOGFILE TO PC
24268     DATA -1,-1
24269  !
24270 CASE 3   ! RUN TIME VARIABLE INITIALIZATION
24271     RESTORE Coef_names
24272     READ C_names$(*)
24273 Coef_names:         ! THESE ARE THE ORDER IN WHICH THE COEFFICIENTS ARE PROC
ESSED
24274     DATA "CONSTANT"
24275     DATA "L(U)"
24276     DATA "L(Rho)"
24277     DATA "L(TO)"
24278     DATA "L(R)L(TO)"
24279     DATA "L(R)L(U)"
24280     DATA "L(U)L(TO)"
24281     DATA "LULRLTO"
24282     DATA "(L(U))**2"
24283     DATA "unused"
24284 !
24285     RESTORE Code_14
24286     READ Gain_code_14(*)  !GAIN CODE CONVERSION TO GAINS FOR HOTWIRES 1-4
24287 Code_14:    !
24288     DATA 0          ! CODE  0     off; not defined
24289     DATA .25        !       1
24290     DATA .5         !       2
24291     DATA 1.
24292     DATA 1.99
24293     DATA 3.98
24294     DATA 7.84
24295     DATA 15.8
24296     DATA 31.6
24297     DATA 63.
24298     DATA 125.
24299     DATA 251.
24300     DATA 501.
24301     DATA 1000.
24302     DATA 1995.
24303     DATA 3981.
24304     DATA 7943.
24305     RESTORE Code_57
24306     READ Gain_code_57(*)  !GAIN CODE CONVERSION TO GAINS FOR HOTWIRES 5-7
24307 Code_57:    !
24308     DATA 0          ! CODE  0
24309     DATA 1.         ! CODE  1
24310     DATA 2.
24311     DATA 5.
24312     DATA 10.
24313     DATA 20.
24314     DATA 50.
24315     DATA 100.
24316     Max_obs_rec=300
24317     Max_vars=50
24318     Rcvd_vars=45
24319 CASE 4   ! POWER ON VARIABLE INITIALIZATION
24320  !
24321     Sfn$="C2"        !COEFFICIENT FILENAME
24322     Initial_probe=1
24323     Ending_probe=2
```

```
24324     MAT P_c= (0)
24325     Num_avgs=5
24326     Log_fn$="LOGFILE"
24327     Initial_obs=0
24328     Ending_obs=0
24329     Num_obs_printed=0
24330     MAT Pkt$= ("0")
24331 CASE ELSE
24332 Usercode:!
24333     SELECT Code
24334 !
24335 !
24336     CASE 4101                         !enter HW COEFFICIENT FILE NAME
24337         SELECT Routinl
24338         CASE 21           !get old data, set up parameters
24339             Mvar$(3)=Sfn$              !present setting
24340         CASE 22             !store new data
24341             Sfn$=Mvar$(3)
24342         END SELECT
24343 !
24344 !
24345     CASE 4111                         ! COMPUTE SENSitivities
24346         SELECT Routine
24347         CASE 31
24348             FOR Probe=Initial_probe TO Ending_probe
24349                 FOR R=Initial_obs TO Ending_obs
24350                     FOR W=1 TO 3
24351                         SELECT Probe
24352                         CASE 1
24353                     ! S(U)        =A2      +A6      *Log(Rho)   +A7
*Log(T0)    +A8     *Log(Rho)Log(T0)+A9*2.*Log(U)
24354                            A_rec(29+W,R)=P_c(W,2)+P_c(W,6)*A_rec(11,R)+P_c
(W,7)*A_rec(12,R)+P_c(W,8)*A_rec(14,R)+P_c(W,9)*2.*A_rec(10,R)
24355                     ! S(Rho)      =A3      +A5      *Log(T0)     +A6
*Log(U)     +A8     *Log(U)Log(T0)
24356                            A_rec(33+W,R)=P_c(W,3)+P_c(W,5)*A_rec(12,R)+P_c
(W,6)*A_rec(10,R)+P_c(W,8)*A_rec(16,R)
24357                     ! S(T0)       =A4      +A5      *Log(Rho)   +A7
*Log(U)     +A8     *Log(U)Log(Rho)
24358                            A_rec(37+W,R)=P_c(W,4)+P_c(W,5)*A_rec(11,R)+P_c
(W,7)*A_rec(10,R)+P_c(W,8)*A_rec(15,R)
24359                         CASE 2
24360                     ! S(U)        =A2      +A6      *Log(Rho)   +A7
*Log(T0)    +A8     *Log(Rho)Log(T0)+A9*2.*Log(U)
24361                            B_rec(29+W,R)=P_c(W,2)+P_c(W,6)*B_rec(11,R)+P_c
(W,7)*B_rec(12,R)+P_c(W,8)*B_rec(14,R)+P_c(W,9)*2.*B_rec(10,R)
24362                     ! S(Rho)      =A3      +A5      *Log(T0)     +A6
*Log(U)     +A8     *Log(U)Log(T0)
24363                            B_rec(33+W,R)=P_c(W,3)+P_c(W,5)*B_rec(12,R)+P_c
(W,6)*B_rec(10,R)+P_c(W,8)*B_rec(16,R)
24364                     ! S(T0)       =A4      +A5      *Log(Rho)   +A7
*Log(U)     +A8     *Log(U)Log(Rho)
24365                            B_rec(37+W,R)=P_c(W,4)+P_c(W,5)*B_rec(11,R)+P_c
(W,7)*B_rec(10,R)+P_c(W,8)*B_rec(15,R)
24366                         END SELECT
24367                     NEXT W
24368                 NEXT R
24369             NEXT Probe
24370             GOSUB Log_vars ! Update the disk files
24371         END SELECT
```

```
24372     CASE 4123                              ! COMPUTE COEFficients
24373         SELECT Routinl
24374         CASE 31
24375             CALL Mlr         !Perform Multiple Linear Regression
24376                              ! for the probes selected
24377         END SELECT
24378     CASE 4124                              ! STORE COEFficient file
24379         SELECT Routinl
24380         CASE 31
24381             Errf=0
24382             CREATE BDAT Sfn$,1,256
24383             ASSIGN @Disk TO Sfn$
24384             IF Errf=0 THEN OUTPUT @Disk;P_c(*)
24385             ASSIGN @Disk TO *
24386         END SELECT
24387     CASE 4102                              ! LOAD COEFficient file
24388         SELECT Routinl
24389         CASE 31
24390             Errf=0
24391             ASSIGN @Disk TO Sfn$
24392             IF Errf=0 THEN ENTER @Disk;P_c(*)
24393             ASSIGN @Disk TO *
24394             PRINTER IS PRT
24395             PRINT CHR$(12)
24396             PRINT "THESE ARE THE COEFFICIENTS FROM COEFFICIENT FILE ";Sfn$
24397             PRINT
24398             PRINT
24399             PRINT
24400             PRINT TAB(5);"WIRE 1";TAB(25);"WIRE 2";TAB(45);"WIRE 3"
24401             PRINT
24402             FOR I=1 TO 10
24403                 PRINT P_c(1,I);TAB(20);P_c(2,I);TAB(40);P_c(3,I)
24404             NEXT I
24405             PRINTER IS CRT
24406         END SELECT
24407     CASE 4103,4133                         ! ENTER COEF file; EDIT COEF file
24408         SELECT Routinl
24409         CASE 31
24410            IF Code=4133 THEN GOTO Edit_coefs
24411             FOR I=1 TO 3
24412                 BEEP
24413                 FOR J=1 TO 10
24414                     CALL Pline(0,"Enter the "&C_names$(J)&" (A"&VAL$(J)&")
coefficient for wire "&VAL$(I)&":                          ")
24415                     ON ERROR RECOVER 24418
24416                     Temp$=FNInput$(3)
24417                     P_c(I,J)=VAL(Temp$)
24418                     ON ERROR CALL Error
24419                 NEXT J
24420             NEXT I
24421 Edit_coefs: !
24422             REPEAT
24423                 CALL Pline(2,"wirel                 wire2                wire3")
24424                 FOR Coef_num=1 TO 10
24425                     CALL Pline(2+Coef_num,VAL$(P_c(1,Coef_num))&"
"&VAL$(P_c(2,Coef_num))&"           "&VAL$(P_c(3,Coef_num)))
24426                 NEXT Coef_num
24427                 CALL Pline(17,"ARE THESE COEFFICIENTS CORRECT FOR PROBE "&V
AL$(Initial_probe)&"?                     ")
24428                 Ans$=FNInput$(2,"NO")
```

```
24429                 CALL Pline(17,"
    ")
24430                 SELECT Ans$
24431                 CASE "YES","YE","Y"
24432                 CASE ELSE
24433                     CALL Pline(15,"Enter the wire # (1-3) whose coeffficien
ts need changing:                                ")
24434                     I=VAL(FNInput$(3,"1"))
24435                     IF I<1 OR I>3 THEN GOTO 24433
24436                     CALL Pline(15,"Enter the coefficient number (A#) whcih
is to be entered:                                ")
24437                     J=VAL(FNInput$(3,"1"))
24438                     IF J<1 OR J>10 THEN GOTO 24436
24439                     CALL Pline(15,"Enter the "&C_names$(J)&" (A"&VAL$(J)&")
 coefficient for wire "&VAL$(I)&":               ")
24440                     ON ERROR RECOVER 24443
24441                     Temp$=FNInput$(2)
24442                     P_c(I,J)=VAL(Temp$)
24443                     ON ERROR CALL Error
24444                     CALL Pline(2+J,VAL$(P_c(1,J))&"         "&VAL$(P_c(2,J
))&"         "&VAL$(P_c(3,J)))
24445                 END SELECT
24446             UNTIL Ans$="Y" OR Ans$="YES" OR Ans$="YE"
24447             Schg=BINIOR(Schg,2^6+2^7)!CLEAR SCREEN+CLEAR GRAPHICS
24448         END SELECT
24449 !
24450 !
24451 !
24452 !
24453     CASE 4125                     !XLATE GAIN CODES TO GAINS
24454         SELECT Routinl
24455         CASE 31
24456             FOR This_obs=Initial_obs TO Ending_obs
24457                 FOR Probe=Initial_probe TO Ending_probe
24458                   ! GET MEAN VALUE & GAIN FROM THE LOG FILE
24459                     SELECT Probe
24460                     CASE 1
24461                         FOR Wire=1 TO 4
24462         !####                SELECT Obs_rec(36,This_obs)
24463                             SELECT 1
24464                             CASE 1 TO 16
24465         !#####                   Wire_gain(Wire)=Gain_code_14(Obs_rec(36,Thi
s_obs)+1)
24466                                 Wire_gain(Wire)=Obs_rec(36,This_obs)
24467                                 A_rec(41+Wire,This_obs)=Wire_gain(Wire)
24468                                 A_var_name$(41+Wire)="GAIN "&VAL$(Wire)
24469                             CASE ELSE
24470                                 Obs_rec(36,This_obs)=0!DEFAULT GAIN
24471                                 A_var_name$(41+Wire)="GAIN "&VAL$(Wire)
24472                             END SELECT
24473                         NEXT Wire
24474                     CASE 2
24475                         FOR Wire=1 TO 3
24476            !####             SELECT Obs_rec(36+Wire,This_obs)
24477                             SELECT 1
24478                             CASE 1 TO 7
24479            !#####                Wire_gain(Wire)=Gain_code_57(Obs_rec(36+Wir
e,This_obs)+1)
24480                                 Wire_gain(Wire)=Obs_rec(36+Wire,This_obs)
24481                                 B_rec(41+Wire,This_obs)=Wire_gain(Wire)
```

```
24482                               B_var_name$(41+Wire)="GAIN "&VAL$(Wire)
24483                           CASE ELSE
24484                               B_rec(41+Wire,This_obs)=0!Default gain = 0
24485                               B_var_name$(41+Wire)="GAIN "&VAL$(Wire)
24486                           END SELECT
24487                       NEXT Wire
24488                   END SELECT
24489               NEXT Probe
24490           NEXT This_obs
24491           GOSUB Log_vars
24492       END SELECT
24493 !
24494 !
24495   CASE 4128                   !REMAKE PROBE DATA
24496       SELECT Routinl
24497       CASE 31
24498           FOR This_obs=Initial_obs TO Ending_obs
24499               CALL Pline(Staty,"Recalculating Tunnel parameters for obser
vation "&VAL$(This_obs))
24500               GOSUB Tun_vars
24501               FOR Probe=Initial_probe TO Ending_probe
24502                   SELECT Probe
24503                   CASE 1
24504 !####                 CALL Pline(Staty,"Recalculating probe A for observa
tion "&VAL$(This_obs)&"         ")
24505                       GOSUB Wall_vars
24506                   CASE 2
24507 !####                 CALL Pline(Staty,"Recalculating probe B for observa
tion "&VAL$(This_obs)&"         ")
24508                       GOSUB Floor_vars
24509                   END SELECT
24510               NEXT Probe
24511               CALL Pline(Staty,"
    ")
24512           NEXT This_obs
24513           GOSUB Log_vars
24514       END SELECT
24515 !
24516 !
24517   CASE 4132                   !COMPUTE R [CORRELATIONS] etc
24518       SELECT Routinl
24519       CASE 31
24520 !             CLEAR ALL DATA TRANSFERS TO START
24521           Mivar(14)=0
24522           Mvar(2)=0
24523           CALL Routine(22,239)        !DATA FILE MAP
24524 !             LOAD ONLY CHANNELS 2,3,4
24525           FOR I=2 TO 4
24526               Mvar(2)=1
24527               Mivar(14)=I
24528               CALL Routine(22,239)        !DATA FILE MAP
24529           NEXT I
24530           FOR Probe=Initial_probe TO Ending_probe
24531               FOR This_obs=Initial_obs TO Ending_obs
24532                   DISP "PROBE "&VAL$(Probe)&"; OBS "&VAL$(This_obs)&"; ";
24533 ! SPECIFY FILENAMES:
24534                   FOR I=2 TO 4
24535                       Filename$(I)="          "
24536                       SELECT I
24537                       CASE 2
```

```
24538                           Filename$(I)[1,1]="V"
24539                           Filename$(I)[7,8]="04"
24540                       CASE 3
24541                           Filename$(I)[1,1]="D"
24542                           Filename$(I)[7,8]="05"
24543                       CASE 4
24544                           Filename$(I)[1,1]="T"
24545                           Filename$(I)[7,8]="06"
24546                       END SELECT
24547                       Filename$(I)[2,2]=CHR$(64+Probe)!"A" for Probe 1, e
tc
24548                       SELECT INT(Obs_rec(2,This_obs))
24549                       CASE 0 TO 9
24550                           Filename$(I)[3,4]="0"&VAL$(Obs_rec(2,This_obs))
!Run
24551                       CASE 10 TO 99
24552                           Filename$(I)[3,4]=VAL$(Obs_rec(2,This_obs))
24553                       END SELECT
24554                       SELECT INT(Obs_rec(3,This_obs))
24555                       CASE 0 TO 9
24556                           Filename$(I)[5,6]="0"&VAL$(Obs_rec(3,This_obs))
!Point
24557                       CASE 10 TO 99
24558                           Filename$(I)[5,6]=VAL$(Obs_rec(3,This_obs))
24559                       END SELECT
24566                       Mivar(14)=I
24567                       CALL Routine(21,232) !DISPLAY THE FILENAME
24568                   NEXT I
24569 ! LOAD THE THREE CHANNELS INTO MEMORYS 2,3,4
24570 !                 (2=V 3=D 4=T):
24571                   CALL Routine(31,235)     !LOAD DATA
24572 !
24573                   FOR Xp=1 TO 3
24574                       DISP "CORRELATION "&VAL$(Xp)&" ";
24575                       Mivar(14)=1
24576                       SELECT Xp
24577                       CASE 1
24578 ! COPY MEMORY 3 TO MEMORY 1, leaving D in 1 and V in 2:
24579                           Mvar(2)=3
24580                           CALL Routine(22,280)!MEMORY COPY V TO 1
24581                       CASE 2
24582 ! COPY MEMORY 4 TO MEMORY 1, leaving T in 1 and V in 2:
24583                           Mvar(2)=4
24584                           CALL Routine(22,280)!MEMORY COPY T TO 1
24585                       CASE 3
24586 ! COPY MEMORY 3 TO MEMORY 1, and
24587 ! COPY MEMORY 4 TO MEMORY 2, leaving D in 1 and T in 2:
24588                           Mvar(2)=3
24589                           CALL Routine(22,280)!MEMORY COPY D TO 1
24590                           Mivar(14)=2
24591                           Mvar(2)=4
24592                           CALL Routine(22,280)!MEMORY COPY T TO 2
24593                       END SELECT
24594 ! MULTIPLY MEMORY 1 BY MEMORY 2:
24595                       Mivar(14)=0
24596                       CALL Routine(31,533)!MEMORY MULTIPLY
24597 ! FIND MEAN OF MEMORY 1
24598                       Mivar(14)=1
24599                       CALL Routine(31,511)!MEAN
24600                       Temp_mean=Mvar(3)
```

```
24601 ! STORE MEAN INTO DATA BASE AS D*V, V*T, or D*T CORRELATION
24602                       SELECT Probe
24603                       CASE 1
24604                           SELECT Xp
24605                           CASE 1!D*V
24606                               Temp_rms=A_rec(48,This_obs)*A_rec(47,This_o
bs)!(d'/D)*(v'/V)
24607                           CASE 2!V*T
24608                               Temp_rms=A_rec(47,This_obs)*A_rec(49,This_o
bs)!(v'/V)*(t'/T)
24609                           CASE 3!D*T
24610                               Temp_rms=A_rec(48,This_obs)*A_rec(49,This_o
bs)!(d'/D)*(t'/T)
24611                           END SELECT
24612                           A_rec(25+Xp,This_obs)=Temp_mean/Temp_rms
24613                       CASE 2
24614                           SELECT Xp
24615                           CASE 1!D*V
24616                               Temp_rms=B_rec(48,This_obs)*B_rec(47,This_o
bs)!(d'/D)*(v'/V)
24617                           CASE 2!V*T
24618                               Temp_rms=B_rec(47,This_obs)*B_rec(49,This_o
bs)!(v'/V)*(t'/T)
24619                           CASE 3!D*T
24620                               Temp_rms=B_rec(48,This_obs)*B_rec(49,This_o
bs)!(d'/D)*(t'/T)
24621                           END SELECT
24622                           B_rec(25+Xp,This_obs)=Temp_mean/Temp_rms
24623                       END SELECT
24624 !
24625                       SELECT Probe
24626                       CASE 1
24627                           A_var_name$(26)="R(RhoU)"
24628                           A_var_name$(27)="R(UTO)"
24629                           A_var_name$(28)="R(RhoTO)"
24630                       CASE 2
24631                           B_var_name$(26)="R(RhoU)"
24632                           B_var_name$(27)="R(UTO)"
24633                           B_var_name$(28)="R(RhoTO)"
24634                       END SELECT
24635                       GOSUB Log_vars
24636                   NEXT Xp
24637               NEXT This_obs
24638           NEXT Probe
24639 Etc4132: !                     COMPUTE M'/M, P'/P
24640           FOR Probe=Initial_probe TO Ending_probe
24641               FOR This_obs=Initial_obs TO Ending_obs
24642                   Mach=Obs_rec(9,This_obs)
24643                   M2=Mach*Mach
24644                   SELECT Probe
24645                   CASE 1
24646                       U=A_rec(47,This_obs)
24647                       Rho=A_rec(48,This_obs)
24648                       TO=A_rec(49,This_obs)
24649                       R_urho=A_rec(26,This_obs)
24650                       R_trho=A_rec(28,This_obs)
24651                       R_ut=A_rec(27,This_obs)
24652                   CASE 2
24653                       U=B_rec(47,This_obs)
24654                       Rho=B_rec(48,This_obs)
```

```
                        T0=B_rec(49,This_obs)
                        R_urho=B_rec(26,This_obs)
                        R_trho=B_rec(28,This_obs)
                        R_ut=B_rec(27,This_obs)
                    END SELECT
                                !MASSFL is M'/M
                    Massfl=SQR(U*U+2*R_urho*U*Rho+Rho*Rho)
                                !PRESS  is P'/P
                    Press=Rho*Rho+M2*M2*(1.44*T0*T0+.16*U*U)
                    Press=Press+M2*(2.4*R_trho*Rho*T0-.8*R_urho*U*Rho)
                    Press=Press+M2*M2*(-.96)*R_ut*U*T0
                    Press=SQR(Press)
                    SELECT Probe
                    CASE 1
                        A_rec(41,This_obs)=Massfl
                        A_rec(46,This_obs)=Press
                    CASE 2
                        B_rec(41,This_obs)=Massfl
                        B_rec(46,This_obs)=Press
                    END SELECT
                    GOSUB Log_vars
                NEXT This_obs
                SELECT Probe
                CASE 1
                    A_var_name$(41)="M'/M"
                    A_var_name$(46)="P'/P"
                CASE 2
                    B_var_name$(41)="M'/M"
                    B_var_name$(46)="P'/P"
                END SELECT
            NEXT Probe
        END SELECT
!
!
                                !CALC VEL etc
!
    CASE 4104                   !Perform translation from voltages to
                                ! velocity, density, temperature
        SELECT Routinl
        CASE 31                 !perform action
            Idiag=0
            FOR Probe=Initial_probe TO Ending_probe
                FOR This_obs=Initial_obs TO Ending_obs! Do one observation at a time
!
!           FOR ALL SAMPLES IN THE MEMORIES - AS DEFINED BY THE CURSORS OF
!           OF TRACE 1, CALCULATE THE EQUIVALENT INSTANTANEOUS :
!
!                   VELOCITY,   DENSITY,    TEMPERATURE
!
!
!
                    Mivar(14)=-1
                    Mvar(2)=0
                    CALL Routine(21,239)!Turn off all data files (maps)
!
!
                    FOR Wire=1 TO 3
                        SELECT Probe! GET MEAN VALUE & GAIN FROM THE LOG FILE
```

```
24713                         CASE 1
24714                             Mean(Wire)=A_rec(17+Wire,This_obs)
24715                             Wire_gain(Wire)=A_rec(41+Wire,This_obs)
24716                         CASE 2
24717                             Mean(Wire)=B_rec(17+Wire,Initial_obs)
24718                             Wire_gain(Wire)=B_rec(41+Wire,This_obs)
24719                         END SELECT
24720                         FOR Sens=1 TO 3! GET THE SENSITVITYS FROM THE LOG F
ILE
24721                             SELECT Probe
24722                             CASE 1
24723                                 Hwsens(Wire,Sens)=A_rec(25+(Sens*4)+Wire,Th
is_obs)
24724                             CASE 2
24725                                 Hwsens(Wire,Sens)=B_rec(25+(Sens*4)+Wire,Th
is_obs)
24726                             CASE 3
24727                                 Hwsens(Wire,Sens)=C_rec(25+(Sens*4)+Wire,Th
is_obs)
24728                             END SELECT
24729                         NEXT Sens
24730!
24731!                          SET UP DATA FILE MAP FOR TRACES 1,2,3 ONLY
24732                         Mivar(14)=Wire
24733                         Mvar(2)=1
24734                         CALL Routine(22,239)!TURN ON data files 1,2,or3
24735                         Mivar(14)=3+Wire
24736                         Mvar(2)=0
24737                         CALL Routine(22,239)!TURN OFF data files 4,5,or6
24738!
24739                         Temp$[1]="R"
24740                         Temp$[2]=CHR$(64+Probe)
24741                         SELECT INT(Obs_rec(2,This_obs))
24742                         CASE 0 TO 9
24743                             Temp$[3,4]="0"&VAL$(INT(Obs_rec(2,This_obs)))!R
UN
24744                         CASE 10 TO 99
24745                             Temp$[3,4]=VAL$(INT(Obs_rec(2,This_obs)))!RUN
24746                         END SELECT
24747                         SELECT INT(Obs_rec(3,This_obs))
24748                         CASE 0 TO 9
24749                             Temp$[5,6]="0"&VAL$(INT(Obs_rec(3,This_obs)))!P
OINT
24750                         CASE 10 TO 99
24751                             Temp$[5,6]=VAL$(INT(Obs_rec(3,This_obs)))!POINT
24752                         END SELECT
24753                         SELECT Probe
24754                         CASE 1
24755                             Temp$[7,8]="0"&VAL$(Wire)
24756                         CASE 2
24757                             Temp$[7,8]="0"&VAL$(Wire+4)
24758                         END SELECT
24759                         Temp$[9,9]="1"
24760           DISP Temp$
24761           WAIT 1
24762                         Filename$(Wire)=Temp$
24763                         Mivar(14)=Wire
24764                         CALL Routine(21,232)! DISPLAY THE FILENAME
24765!
24766!
```

```
24767                               !TRANSFER CHANNEL CHARACTERISTICS
24768                               ! TO COMPUTED CHANNELS
24769           FOR I=1 TO 15
24770               Sw(Wire+3,I)=Sw(Wire,I)
24771           NEXT I
24772       NEXT Wire
24773!
24774       CALL Routine(43,235) ! LOAD DATA
24775       CALL Routine(31,235)
24776       FOR I=1 TO 3         ! For each channel
24777                ! CLEAR MEMORY TAGS FOR "REMOVE MEAN" FUNCTION
24778           FOR J=0 TO 2
24779               Sw$(I,13+J)=""
24780           NEXT J
24781       Mivar(14)=I
24782       CALL Routine(31,524)! REMOVE MEAN
24783       CALL Routine(31,513)! FIND RMS
24784       Hw_rms(I)=Mvar(3)
24785       NEXT I
24786!
24787!         TRACE 1, 2, AND 3 POINT TO THE MEMORIES
24788!               CONTAINING THE FLUCTUATING COMPONENT ONLY
24789!
24790 !
24791!            SET UP DATA FILE MAP FOR RESULTS: 4,5,6 ONLY
24792       FOR Results=4 TO 6
24793           Mivar(14)=Results-3
24794           Mvar(2)=0
24795           CALL Routine(22,239)!TURN OFF data files 1,2,3
24796           Mivar(14)=Results
24797           Mvar(2)=1
24798           CALL Routine(22,239)!TURN ON data files 4,5,6
24799!
24800           SELECT Results
24801           CASE 4  ! Velocity
24802               Temp$[1]="V"
24803           CASE 5  ! Density
24804               Temp$[1]="D"
24805           CASE 6  ! Temperature
24806               Temp$[1]="T"
24807           END SELECT
24808           Temp$[2]=CHR$(64+Probe)
24809           SELECT INT(Obs_rec(2,This_obs))
24810           CASE 0 TO 9
24811               Temp$[3,4]="0"&VAL$(INT(Obs_rec(2,This_obs)))!R
UN
24812           CASE 10 TO 99
24813               Temp$[3,4]=VAL$(INT(Obs_rec(2,This_obs)))!RUN
24814           END SELECT
24815           SELECT INT(Obs_rec(3,This_obs))
24816           CASE 0 TO 9
24817               Temp$[5,6]="0"&VAL$(INT(Obs_rec(3,This_obs)))!P
OINT
24818           CASE 10 TO 99
24819               Temp$[5,6]=VAL$(INT(Obs_rec(3,This_obs)))!POINT
24820           END SELECT
24821           SELECT Probe
24822           CASE 1
24823               Temp$[7,8]="0"&VAL$(Results)
24824           CASE 2
```

```
24825                           Temp$[7,8]="0"&VAL$(Results+4)
24826                       END SELECT
24827                       Filename$(Results)=Temp$
24828                       Mivar(14)=Results
24829                       CALL Routine(21,232)! DISPLAY THE FILENAME
24830                   NEXT Results
24831   !
24832   !    FILENAMES SET UP TO STORE DATA FILES
24833   !
24834                   MAT Sensinv= INV(Hwsens)
24835!
24836!  D E T E R M I N E   S C A L I N G    (from a sampling of the data)
24837!
24838                   Wtr=1
24839                   Numb_samp=Sw(Wtr,2)
24840                   SELECT Numb_samp
24841                   CASE <100
24842                       Numb_sub=Numb_samp
24843                   CASE 100 TO 10000
24844                       Numb_sub=100
24845                   CASE >10000
24846                       Numb_sub=Numb_samp/100
24847                   END SELECT
24848!  DETERMINE Mean, Standard deviation, Min, Max, range, Offset, Scale
24849!           of velocity, density, and temperature.
24850!
24851                   Value=0
24852                   FOR Wtr=1 TO 6!pick up trace number
24853                       Wmem(Wtr)=Trmem(Wtr)!pick up the memory number
24854                       IF (BIT(Cactive,Wtr*2-1)) THEN ! ACTIVE 1st CURSOR
24855                           Wmark(Wtr,1)=Caddrx(Wtr*2-1)!get the position o
f first cursor
24856                           IF (BIT(Cactive,Wtr*2)) THEN !  and ACTIVE 2nd
CURSOR
24857                               Wmark(Wtr,2)=Caddrx(Wtr*2)
24858                           ELSE                  !  and IN-ACTIVE 2nd CURSOR
24859                               Wmark(Wtr,2)=Sw(Wtr,2)!no second cursor; us
e last point in memory
24860                           END IF
24861                       ELSE                      ! IN-ACTIVE 1st CURSOR
24862                           IF (BIT(Cactive,Wtr*2)) THEN ! ACTIVE 2nd CURSO
R
24863                               Wmark(Wtr,1)=Caddrx(Wtr*2)!get the position
 of second cursor
24864                           ELSE
24865                               Wmark(Wtr,1)=0
24866                           END IF
24867                           Wmark(Wtr,2)=Sw(Wtr,2)
24868                       END IF
24869                       IF Wmark(Wtr,1)>Wmark(Wtr,2) THEN !assure starting
position is <=
24870                           Temp=Wmark(Wtr,1)! to ending position
24871                           Wmark(Wtr,1)=Wmark(Wtr,2)
24872                           Wmark(Wtr,2)=Temp
24873                       END IF
24874                       Xch(Wtr)=((FNMems(1,Wmem(Wtr))+Wmark(Wtr,1)) DIV 10
24)+1!starting row address
24875                       Xpt(Wtr)=((FNMems(1,Wmem(Wtr))+Wmark(Wtr,1)) MOD 10
24)+1!starting column address
24876                       Nch(Wtr)=(Wmark(Wtr,2)-Wmark(Wtr,1)) DIV 1024!numbe
```

```
r of rows
24877                   Npt(Wtr)-(Wmark(Wtr,2)-Wmark(Wtr,1)) MOD 1024!numbe
r of columns in last row
24878                   Vo(Wtr)-Sw(Trmem(Wtr),22)/100!y OFFSET      offset
24879                   Vs(Wtr)-Sw(Trmem(Wtr),21)!y GAIN  (VOLTS) sensitivi
ty
24880               NEXT Wtr
24881               Strtpt-MAX(Wmark(1,1),Wmark(2,1),Wmark(3,1))
24882               Stoppt-MIN(Wmark(1,2),Wmark(2,2),Wmark(3,2))
24883 !
24884 !  FOR  THE SAMPLE POINTS:
24885               FOR I-Strtpt TO Stoppt STEP Numb_samp/Numb_sub
24886                   FOR Wire-1 TO 3
24887                       IF FNMems(1,Trmem(Wire))+I<-Sw(Wire,2) THEN
24888                           Xsch(Wire)-((FNMems(1,Trmem(Wire))+I) DIV 1
024)+1
24889                           Xspt(Wire)-((FNMems(1,Trmem(Wire))+I) MOD 1
024)+1
24890                           Temp-X(Xsch(Wire),Xspt(Wire))
24891                           Value-((Temp/65536)-Vo(Wire))*Vs(Wire)
24892                           Enorm(Wire)-Value/(Mean(Wire)*Wire_gain(Wir
e))
24893                           IF Idiag-9 AND Wire-1 AND I<900 THEN
24894                               PRINTER IS PRT
24895                               PRINT I;"th Sample -(Wire";Wire;")-  X:
";Temp;" X(volts): ";Value;" Enorm:";Enorm(Wire)
24896                               PRINTER IS CRT
24897                           END IF
24898                       END IF
24899                   NEXT Wire
24900                   MAT Vdp- Sensinv*Enorm
24901                   FOR Ip-1 TO 3
24902                       SELECT I
24903                       CASE Strtpt
24904                           Max_param(Ip+3)-Vdp(Ip)
24905                           Min_param(Ip+3)-Vdp(Ip)
24906                           Sum_param(Ip)-Vdp(Ip)
24907                           Sumsq_param(Ip)-Vdp(Ip)*Vdp(Ip)
24908                       CASE ELSE
24909                           IF Vdp(Ip)>Max_param(Ip+3) THEN Max_param(I
p+3)-Vdp(Ip)
24910                           IF Vdp(Ip)<Min_param(Ip+3) THEN Min_param(I
p+3)-Vdp(Ip)
24911                           Sum_param(Ip)-Sum_param(Ip)+Vdp(Ip)
24912                           Sumsq_param(Ip)-Sumsq_param(Ip)+(Vdp(Ip)*Vd
p(Ip))
24913                           IF Idiag AND Ip-3 THEN
24914                               PRINTER IS PRT
24915                               PRINT "TEST VALUES  ";Vdp(*)
24916                               PRINT "MAX          ";Max_param(*)
24917                               PRINT "MIN          ";Min_param(*)
24918                               PRINTER IS CRT
24919                           END IF
24920                       END SELECT
24921                   NEXT Ip
24922               NEXT I
24923!
24924!      CALCULATE THE SLOPE (Vs_param) AND INTERCEPT (Vo_param)
24925               FOR Ip-1 TO 3
24926                   Mean_param(Ip)-Sum_param(Ip)/Numb_sub
```

```
24927                       IF ((Sumsq_param(Ip)-(Numb_sub*Mean_param(Ip)*Mean_
param(Ip)))/(Mean_param(Ip)-1))>=0 THEN
24928                           Stddev_param(Ip)=SQR((Sumsq_param(Ip)-(Numb_sub
*Mean_param(Ip)*Mean_param(Ip)))/(Mean_param(Ip)-1))
24929                       ELSE
24930                           Stddev_param(Ip)=0
24931                       END IF
24932                       Vo_param(Ip+3)=(Max_param(Ip+3)+Min_param(Ip+3))/2.
24933                       IF Max_param(Ip+3)-Vo_param(Ip+3)<>0 THEN
24934                           Vs_param(Ip+3)=10*((Max_param(Ip+3)-Vo_param(Ip
+3))/32767)
24935                       ELSE
24936                           Vs_param(Ip+3)=10.
24937                       END IF
24938! STORE OFFSETS AND SLOPES IN THE MEMORIES POINTED TO BY TRACES 4,5,6
24939                       Sw(Ip+3,28)=Vo_param(Ip+3)
24940                       Sw(Ip+3,27)=Vs_param(Ip+3)
24941! IDENTIFY DISPLAY AS USER UNITS
24942                       Isw(Ip+3,13)=1
24943                       SELECT Ip
24944                       CASE 1
24945                           Sw$(Ip+3,11)="u'/U"
24946                       CASE 2
24947                           Sw$(Ip+3,11)="p'/P"
24948                       CASE 3
24949                           Sw$(Ip+3,11)="t'/T"
24950                       END SELECT
24951                   NEXT Ip
24952!
24953!   C O M P U T E             u, p, To
24954!
24955                   FOR I=Strtpt TO Stoppt
24956                       FOR Wire=1 TO 3
24957                           Temp=X(Xch(Wire),Xpt(Wire))
24958                           Value=((Temp/65536)-Vo(Wire))*Vs(Wire)
24959                           Enorm(Wire)=Value/(Mean(Wire)*Wire_gain(Wire))
24960                           Xpt(Wire)=Xpt(Wire)+1
24961                           IF Xpt(Wire)>1024 THEN
24962                               Xpt(Wire)=1
24963                               Xch(Wire)=Xch(Wire)+1
24964                           END IF
24965                       NEXT Wire
24966!
24967                       MAT Vdp= Sensinv*Enorm
24968                       IF Idiag THEN
24969                           PRINTER IS PRT;WIDTH 134
24970                           IF I=Strtpt THEN
24971                               PRINT "MAX_P:  ";Max_param(*)
24972                               PRINT "MIN_P:  ";Min_param(*)
24973                               PRINT
24974                               PRINT "SLOPE:     ";Vs_param(*)
24975                               PRINT "INTERCEPT: ";Vo_param(*)
24976                               PRINT
24977                               PRINT "Means: ";Mean(*)
24978                               PRINT "Gains: ";Wire_gain(*)
24979                               PRINT
24980                           END IF
24981                           PRINTER IS CRT
24982                       END IF
24983                       FOR Ip=4 TO 6! PUT u, p, To IN  MEMORIES UNDER TRAC
```

```
ES 4, 5, 6
24984                          Value=INT((Vdp(Ip-3)-Vo_param(Ip))/Vs_param(Ip)
)
24985                          IF Idiag AND Ip=6 THEN
24986                             IF I MOD 100=0 AND I<1500 THEN
24987                                PRINTER IS PRT;WIDTH 134
24988                                PRINT "SAMPLE: ";I;"TRACE: ";Ip,"Vdp: "
;Vdp(Ip-3)," Value: ";Value
24989                                PRINTER IS CRT
24990                             END IF
24991                          END IF
24992                          SELECT Value! Limit the range to the 16 bit int
eger
24993                          CASE -32768 TO 32767
24994                             X(Xch(Ip),Xpt(Ip))=Value
24995                          CASE >32767
24996                             X(Xch(Ip),Xpt(Ip))=32767
24997                          CASE <32768
24998                             X(Xch(Ip),Xpt(Ip))=-32768
24999                          END SELECT
25000                          Xpt(Ip)=Xpt(Ip)+1
25001                          IF Xpt(Ip)>1024 THEN
25002                             Xpt(Ip)=1
25003                             Xch(Ip)=Xch(Ip)+1
25004                          END IF
25005                       NEXT Ip
25006     !                 DISP I
25007                    NEXT I
25008                    FOR Ip=1 TO 3! For each ratio (u'/U,p'/P,t0'/T0)
25009                       FOR I=4 TO 6! CLEAR MEMORY TAGS FOR "REMOVE MEAN" F
UNCTION
25010                          FOR J=0 TO 2
25011                             Sw$(I,13+J)=""
25012                          NEXT J
25013                       NEXT I
25014                       Mivar(14)=Ip+3
25015                       CALL Routine(31,524)! REMOVE MEAN
25016                       CALL Routine(31,513)! FIND RMS
25017                       Value=Mvar(3)
25018                       SELECT Probe
25019                       CASE 1
25020                          A_rec(46+Ip,This_obs)=Value
25021                       CASE 2
25022                          B_rec(46+Ip,This_obs)=Value
25023                       END SELECT
25024                    NEXT Ip
25025!
25026!
25027                    CALL Routine(43,234)!STORE FLUCTUATING DATA
25028                    CALL Routine(31,234)
25029!
25030!
25031  !!!!!!!
25032                    RESTORE A_names
25033                    READ A_var_name$(*)
25034                    RESTORE B_names
25035                    READ B_var_name$(*)
25036  !!!!!!!
25037                    GOSUB Log_vars  ! Save the computed values on disc
25038  !
```

```
25039                    PRINTER IS PRT
25040                    PRINT CHR$(12) !FORM FEED
25041                    PRINT Data_set_title$;TAB(40);"PROBE ";Probe;TAB(60);"O
BSERVATION # ";This_obs
25042                    PRINT
25043                    PRINT "TEST     ";Obs_rec(1,This_obs)
25044                    PRINT "RUN      ";Obs_rec(2,This_obs)
25045                    PRINT "POINT    ";Obs_rec(3,This_obs)
25046                    PRINT "P_TOTAL  ";Obs_rec(7,This_obs)
25047                    PRINT "P_STATIC ";Obs_rec(6,This_obs)
25048                    PRINT "T_TOTAL  ";Obs_rec(8,This_obs)
25049                    PRINT "MACH #   ";Obs_rec(9,This_obs)
25050                    PRINT
25051                    PRINT TAB(20);"WIRE 1";TAB(40);"WIRE 2";TAB(60);"WIRE 3
"
25052                    PRINT "V_mean";TAB(20);Mean(1);TAB(40);Mean(2);TAB(60);
Mean(3)
25053                    PRINT "V_rms";TAB(20);Hw_rms(1);TAB(40);Hw_rms(2);TAB(6
0);Hw_rms(3)
25054                    PRINT "GAIN";TAB(20);Wire_gain(1);TAB(40);Wire_gain(2);
TAB(60);Wire_gain(3)
25055                    PRINT "SENSITIVITY";TAB(20);Hwsens(1,1);TAB(40);Hwsens(
2,1);TAB(60);Hwsens(3,1)
25056                    PRINT TAB(20);Hwsens(1,2);TAB(40);Hwsens(2,2);TAB(60);H
wsens(3,2)
25057                    PRINT TAB(20);Hwsens(1,3);TAB(40);Hwsens(2,3);TAB(60);H
wsens(3,3)
25058                    SELECT Probe
25059                    CASE 1
25060                      Vel_f=A_rec(47,This_obs)
25061                      Dens_f=A_rec(48,This_obs)
25062                      Temp_f=A_rec(49,This_obs)
25063                    CASE 2
25064                      Vel_f=B_rec(47,This_obs)
25065                      Dens_f=B_rec(48,This_obs)
25066                      Temp_f=B_rec(49,This_obs)
25067                    END SELECT
25068                    PRINT
25069                    PRINT "u'/U_rms";TAB(20);Vel_f
25070                    PRINT "p'/P_rms";TAB(20);Dens_f
25071                    PRINT "t'/T_rms";TAB(20);Temp_f
25072                    PRINT
25073                    PRINTER IS CRT
25074                NEXT This_obs
25075  !
25076  !
25077  !
25078            NEXT Probe
25079  !
25080        END SELECT
25081        Schg=BINIOR(Schg,16384) !SET BIT 14 TO SAY WAVEFORMS CHANGED
25082  !
25083  !
25084  !
25085 Log_file_menu: !
25086    CASE 4105                         !enter LOG FILE NAME
25087        SELECT Routinl
25088        CASE 21            !get old data, set up parameters
25089            Mvar$(3)=Log_fn$                !present setting
25090        CASE 22              !store new data
```

```
25091             Log_fn$=Mvar$(3)
25092         END SELECT
25093     CASE 4106                        ! LOAD LOG FILE
25094         SELECT Routinl
25095         CASE 31
25096             ON ERROR GOTO No_file
25097             ASSIGN @Disk TO Log_fn$
25098             ON ERROR CALL Error
25099             ENTER @Disk,1;Data_set_title$,Max_obs_rec,Max_vars,Logged_var_n
ame$(*),Numsubfile,Subfile_names$(*),Subfile_chartst(*)
25100             Num_obs_recd=Subfile_chartst(1)
25101             !
25102             REDIM Obs_rec(Max_vars,Max_obs_rec),A_rec(Max_vars,Max_obs_rec)
,B_rec(Max_vars,Max_obs_rec)
25103             !
25104             ENTER @Disk,2
25105             ENTER @Disk;Obs_rec(*)
25106 !                                     Set up WALL strut file
25107             ASSIGN @Diska TO Log_fn$&"A"
25108             ENTER @Diska,1;A_set_title$,Dummy,Dummy,A_var_name$(*),Dummy,A_
subfile_names$(*),A_sub_chartst(*)
25109             ENTER @Diska,2
25110             ENTER @Diska;A_rec(*)
25111 !                                     Set up FLOOR strut file
25112             ASSIGN @Diskb TO Log_fn$&"B"
25113             ENTER @Diskb,1;B_set_title$,Dummy,Dummy,B_var_name$(*),Dummy,B_
subfile_names$(*),B_sub_chartst(*)
25114             ENTER @Diskb,2
25115             ENTER @Diskb;B_rec(*)
25116 !                                     Set up OTHER strut file
25117 !!!!         ASSIGN @Diskc TO Log_fn$&"C"
25118 !!!!         ENTER @Diskc,1;C_set_title$,Dummy,Dummy,C_var_name$(*),Dummy,C
_subfile_names$(*),C_sub_chartst(*)
25119 !!!!         ENTER @Diskc,2
25120 !!!!         ENTER @Diskc;C_rec(*)
25121             GOTO Got_it
25122 No_file:    !NO FILE.       !OK, SO CREATE THE FILE
25123             ON ERROR CALL Error
25124             Errf=0
25125             CREATE BDAT Log_fn$,INT(8*Max_vars*Max_obs_rec/1280)+2,1280
25126             ASSIGN @Disk TO Log_fn$
25127             CREATE BDAT Log_fn$&"A",INT(8*Max_vars*Max_obs_rec/1280)+2,1280
25128             ASSIGN @Diska TO Log_fn$&"A"
25129             CREATE BDAT Log_fn$&"B",INT(8*Max_vars*Max_obs_rec/1280)+2,1280
25130             ASSIGN @Diskb TO Log_fn$&"B"
25131 !!!!         CREATE BDAT Log_fn$&"C",INT(8*Max_vars*Max_obs_rec/1280)+2,128
0
25132 !!!!         ASSIGN @Diskc TO Log_fn$&"C"
25133             File_comments$=""
25134             DISP "ENTER THE LOG FILE DATA SET TITLE - Return IF NONE";
25135             INPUT "",File_comments$
25136             IF LEN(File_comments$)>0 THEN
25137                 Data_set_title$=File_comments$
25138                 A_set_title$="WALL STRUT :::"&File_comments$
25139                 B_set_title$="FLOOR STRUT :::"&File_comments$
25140                 C_set_title$="KULITE, MISC :::"&File_comments$
25141             ELSE
25142                 Data_set_title$=""
25143                 A_set_title$=""
25144                 B_set_title$=""
```

```
25145                C_set_title$=""
25146             END IF
25147             Num_obs_recd=0
25148             Numsubfile=0
25149             MAT Subfile_names$= ("")
25150             MAT A_subfile_names$= ("")
25151             MAT B_subfile_names$= ("")
25152             MAT C_subfile_names$= ("")
25153             MAT Subfile_chartst= (0)
25154             MAT A_sub_chartst= (0)
25155             MAT B_sub_chartst= (0)
25156             MAT C_sub_chartst= (0)
25157             MAT Obs_rec= (-9999999.99999)
25158             MAT A_rec= (-9999999.99999)
25159             MAT B_rec= (-9999999.99999)
25160             MAT C_rec= (-9999999.99999)
25161 !
25162 !
25163             RESTORE Var_names   !   NOTE:
25164                                 ! IN FUTURE LINKS, THESE NAMES WILL PRECEDE
25165                                 ! THE VALUES IN THE DATA PACKET RECEIVED.
25166             READ Logged_var_name$(*)
25167 Var_names:! THE VARIABLE NAMES THAT EACH OBSERVATION RECORD CONTAINS:
25168             DATA "TEST"  ! 1
25169             DATA "RUN"   ! 2
25170             DATA "POINT" ! 3
25171             DATA "TIME"  ! 4
25172             DATA "DATE"  ! 5
25173             DATA "Ps"    ! 6
25174             DATA "Pt"    ! 7
25175             DATA "Tt"    ! 8
25176             DATA "MACH"  ! 9
25177             DATA "REYNO" !10
25178             DATA "PtS1"
25179             DATA "PsS1"
25180             DATA "TtS1"
25181             DATA "PtS2"
25182             DATA "PsS2"
25183             DATA "TtS2"
25184             DATA "PtS3"
25185             DATA "PsS3"
25186             DATA "TtS3"
25187             DATA "P1HW1"
25188             DATA "P1HW2"
25189             DATA "P1HW3"
25190             DATA "P1HW4"
25191             DATA "P2HW1"
25192             DATA "P2HW2"
25193             DATA "P2HW3"
25194             DATA "P3HW1"
25195             DATA "P4HW1"
25196             DATA "P5HW1"
25197             DATA "KULITE1" !30
25198             DATA "KULITE2" !31
25199             DATA "KULITE3" !32
25200             DATA "KULITE4" !33
25201             DATA "KULITE5" !34
25202             DATA "KULITE6" !35
25203             DATA "HW1-4 GAIN"
25204             DATA "P2HW1GAIN"
```

```
25205           DATA "P2HW2GAIN"
25206           DATA "P2HW3GAIN"
25207           DATA "K1 GAIN"
25208           DATA "K2 GAIN"
25209           DATA "K3 GAIN"
25210           DATA "K4 GAIN"
25211           DATA "K5 GAIN"
25212           DATA "K6 GAIN"
25213           DATA ""
25214           DATA ""
25215           DATA ""
25216           DATA ""
25217           DATA ""
25218   !
25219   !
25220           Subfile_chartst(1)=Num_obs_recd
25221           OUTPUT @Disk,1;Data_set_title$,Max_obs_rec,Max_vars,Logged_var_
name$(*),Numsubfile,Subfile_names$(*),Subfile_chartst(*)
25222           STATUS @Disk,3;Norecs,Nobpr
25223           CONTROL @Disk,7;Norecs,Nobpr
25224           ENTER @Disk,2
25225           OUTPUT @Disk;Obs_rec(*)
25226   !
25227           RESTORE A_names  !   NOTE:
25228                               ! THESE ARE THE VARIABLE NAMES FOR THE
25229                               ! WALL STRUT DATA FILE "A"
25230           READ A_var_name$(*)
25231 A_names:! THE VARIABLE NAMES THAT EACH "A" DATA RECORD CONTAINS:
25232           DATA "TEST"  ! 1
25233           DATA "RUN"   ! 2
25234           DATA "POINT" ! 3
25235           DATA "PtS1"
25236           DATA "PsS1"
25237           DATA "TtS1"
25238           DATA "VELOCITY"
25239           DATA "DENSITY"
25240           DATA "Ts"
25241           DATA "L(U)"
25242           DATA "L(Rho)"
25243           DATA "L(TO)"
25244           DATA "L(RhoU)"
25245           DATA "L(R)L(TO)"
25246           DATA "L(R)L(U)"
25247           DATA "L(U)L(TO)"
25248           DATA "LULRLTO"
25249           DATA "P1HW1"       !18
25250           DATA "P1HW2"
25251           DATA "P1HW3"
25252           DATA "P1HW4"
25253           DATA "LOG(P1HW1)"  !22
25254           DATA "LOG(P1HW2)"
25255           DATA "LOG(P1HW3)"
25256           DATA "LOG(P1HW4)"
25257           DATA "R(RhoU)"        !26
25258           DATA "R(UTO)"
25259           DATA "R(RhoTO)"
25260           DATA ""
25261           DATA "S(U)1"       !30
25262           DATA "S(U)2"
25263           DATA "S(U)3"
```

PRECEDING PAGE BLANK NOT FILMED

```
25264           DATA ""
25265           DATA "S(Rho)1"     !34
25266           DATA "S(Rho)2"
25267           DATA "S(Rho)3"
25268           DATA ""
25269           DATA "S(TO)1"      !38
25270           DATA "S(TO)2"
25271           DATA "S(TO)3"
25272           DATA "M'/M"           !41
25273           DATA "GAIN 1"      !42
25274           DATA "GAIN 2"
25275           DATA "GAIN 3"
25276           DATA "GAIN 4"
25277           DATA "P'/P"           !46
25278           DATA "u'/U"           !47
25279           DATA "p'/P"
25280           DATA "to'/To"
25281           DATA ""
25282  !
25283  !
25284           OUTPUT @Diska,1;A_set_title$,Max_obs_rec,Max_vars,A_var_name$(*
),Numsubfile,A_subfile_names$(*),Subfile_chartst(*)
25285           STATUS @Diska,3;Norecs,Nobpr
25286           CONTROL @Diska,7;Norecs,Nobpr
25287           ENTER @Diska,2
25288           OUTPUT @Diska;A_rec(*)
25289  !
25290           RESTORE B_names  !   NOTE:
25291                               ! THESE ARE THE VARIABLE NAMES FOR THE
25292                               ! FLOOR STRUT DATA FILE "B"
25293           READ B_var_name$(*)
25294 B_names:! THE VARIABLE NAMES THAT EACH "B" DATA RECORD CONTAINS:
25295           DATA "TEST"  ! 1
25296           DATA "RUN"   ! 2
25297           DATA "POINT" ! 3
25298           DATA "PtS2"
25299           DATA "PsS2"
25300           DATA "TtS2"
25301           DATA "VELOCITY"
25302           DATA "DENSITY"
25303           DATA "Ts"
25304           DATA "L(U)"
25305           DATA "L(Rho)"
25306           DATA "L(TO)"
25307           DATA "L(RhoU)"
25308           DATA "L(R)L(TO)"
25309           DATA "L(R)L(U)"
25310           DATA "L(U)L(TO)"
25311           DATA "LULRLTO"
25312           DATA "P2HW1"          !18
25313           DATA "P2HW2"
25314           DATA "P2HW3"
25315           DATA ""
25316           DATA "LOG(P2HW1)"     !22
25317           DATA "LOG(P2HW2)"
25318           DATA "LOG(P2HW3)"
25319           DATA ""
25320           DATA "R(RhoU)"           !26
25321           DATA "R(UTO)"
25322           DATA "R(RhoTO)"
```

```
25323           DATA ""
25324           DATA "S(U)1"            !30
25325           DATA "S(U)2"
25326           DATA "S(U)3"
25327           DATA ""
25328           DATA "S(Rho)1"          !34
25329           DATA "S(Rho)2"
25330           DATA "S(Rho)3"
25331           DATA ""
25332           DATA "S(TO)1"           !38
25333           DATA "S(TO)2"
25334           DATA "S(TO)3"
25335           DATA "M'/M"                !41
25336           DATA "GAIN 1"              !42
25337           DATA "GAIN 2"
25338           DATA "GAIN 3"
25339           DATA ""
25340           DATA "P'/P"                !46
25341           DATA "u'/U"                !47
25342           DATA "p'/P"
25343           DATA "to'/To"
25344           DATA ""
25345   !
25346   !
25347           OUTPUT @Diskb,1;B_set_title$,Max_obs_rec,Max_vars,B_var_name$(*
),Numsubfile,B_subfile_names$(*),Subfile_chartst(*)
25348           STATUS @Diskb,3;Norecs,Nobpr
25349           CONTROL @Diskb,7;Norecs,Nobpr
25350           ENTER @Diskb,2
25351           OUTPUT @Diskb;B_rec(*)
25352   !
25353   !     RESTORE C_names        !   NOTE:
25354                                 ! THESE ARE THE VARIABLE NAMES FOR THE
25355                                 ! 'OTHER' STRUT DATA FILE "C"
25356   !     READ C_var_name$(*)
25357 C_names:! THE VARIABLE NAMES THAT EACH "C" DATA RECORD CONTAINS:
25358           DATA "TEST"   ! 1
25359           DATA "RUN"    ! 2
25360           DATA "POINT"  ! 3
25361           DATA "PtS3"
25362           DATA "PsS3"
25363           DATA "TtS3"
25364           DATA "VELOCITY"
25365           DATA "DENSITY"
25366           DATA "Ts"
25367           DATA "L(U)"
25368           DATA "L(Rho)"
25369           DATA "L(TO)"
25370           DATA "L(RhoU)"
25371           DATA "L(R)L(TO)"
25372           DATA "L(R)L(U)"
25373           DATA "L(U)L(TO)"
25374           DATA "LULRLTO"
25375           DATA "P3HW1"
25376           DATA "P4HW1"
25377           DATA "P5HW1"
25378           DATA ""
25379           DATA "LOG(P3HW1)"       !22
25380           DATA "LOG(P4HW1)"
25381           DATA "LOG(P5HW1)"
```

```
25382          DATA ""
25383          DATA ""
25384          DATA ""
25385          DATA ""
25386          DATA ""
25387          DATA "S(U)1"          !30
25388          DATA "S(U)2"
25389          DATA "S(U)3"
25390          DATA ""
25391          DATA "S(Rho)1"        !34
25392          DATA "S(Rho)2"
25393          DATA "S(Rho)3"
25394          DATA ""
25395          DATA "S(TO)1"         !38
25396          DATA "S(TO)2"
25397          DATA "S(TO)3"
25398          DATA ""
25399          DATA ""
25400          DATA ""
25401          DATA ""
25402          DATA ""
25403          DATA ""
25404          DATA "u'/U"
25405          DATA "p'/P"
25406          DATA "to'/To"
25407          DATA ""
25408  !
25409  !
25410 !!!!       OUTPUT @Diskc,1;C_set_title$,Max_obs_rec,Max_vars,C_var_name$(
*),Numsubfile,C_subfile_names$(*),Subfile_chartst(*)
25411 !!!!       STATUS @Diskc,3;Norecs,Nobpr
25412 !!!!       CONTROL @Diskc,7;Norecs,Nobpr
25413 !!!!       ENTER @Diskc,2
25414 !!!!       OUTPUT @Diskc;C_rec(*)
25415  !
25416 Got_it: !
25417          ASSIGN @Disk TO *
25418          ASSIGN @Diska TO *
25419          ASSIGN @Diskb TO *
25420 !!!!     ASSIGN @Diskc TO *
25421          Num_obs_plotted=0
25422          Num_obs_printed=0
25423          Initial_obs=Num_obs_recd+1
25424          Ending_obs=Initial_obs
25425        END SELECT
25426     CASE 4107                        ! PRINT LOG FILE INFO
25427        SELECT Routinl
25428        CASE 31
25429          Errf=0
25430  !
25431          Errorf=0
25432          ASSIGN @Disk TO Log_fn$
25433          IF Errorf=0 THEN ENTER @Disk,1;Data_set_title$,Max_obs_rec,Max_
vars,Logged_var_name$(*),Numsubfile,Subfile_names$(*),Subfile_chartst(*)
25434          Num_obs_recd=Subfile_chartst(1)
25435         !
25436          REDIM Obs_rec(Max_vars,Max_obs_rec),A_rec(Max_vars,Max_obs_rec)
,B_rec(Max_vars,Max_obs_rec)
25437         !
25438          ENTER @Disk,2
```

```
25439              IF Errf=0 AND Num_obs_recd>=1 AND Num_obs_recd<=Max_obs_rec THE
N ENTER @Disk;Obs_rec(*)
25440              ASSIGN @Disk TO *
25441  !
25442              Errorf=0
25443              ASSIGN @Diska TO Log_fn$&"A"
25444              IF Errorf=0 THEN ENTER @Diska,1;A_set_title$,Dummy,Dummy,A_var_
name$(*),Dummy,A_subfile_names$(*),A_sub_chartst(*)
25445              ENTER @Diska,2
25446              IF Errf=0 AND Num_obs_recd>=1 AND Num_obs_recd<=Max_obs_rec THE
N ENTER @Diska;A_rec(*)
25447              ASSIGN @Disk TO *
25448  !
25449              Errorf=0
25450              ASSIGN @Diskb TO Log_fn$&"B"
25451              IF Errorf=0 THEN ENTER @Diskb,1;B_set_title$,Dummy,Dummy,B_var_
name$(*),Dummy,B_subfile_names$(*),B_sub_chartst(*)
25452              ENTER @Diskb,2
25453              IF Errf=0 AND Num_obs_recd>=1 AND Num_obs_recd<=Max_obs_rec THE
N ENTER @Diskb;B_rec(*)
25454              ASSIGN @Disk TO *
25455  !
25456              Errorf=0
25457 !!!!       ASSIGN @Diskc TO Log_fn$&"C"
25458 !!!!       IF Errorf=0 THEN ENTER @Diskc,1;C_set_title$,Dummy,Dummy,C_var_n
ame$(*),Dummy,C_subfile_names$(*),C_sub_chartst(*)
25459 !!!!       ENTER @Diskc,2
25460 !!!!       IF Errf=0 AND Num_obs_recd>=1 AND Num_obs_recd<=Max_obs_rec THEN
 ENTER @Diskc;C_rec(*)
25461 !!!!       ASSIGN @Disk TO *
25462  !
25463              GOSUB Print_log_data
25464              GOTO End_4107
25465  !
25466  !
25467 Print_log_data:  !Subroutine to print a formatted report.
25468  !
25469  !
25470              IF Num_obs_printed=0 THEN Num_obs_printed=Initial_obs-1
25471              WHILE Num_obs_printed<Ending_obs
25472                  Num_obs_printed=Num_obs_printed+1
25473                  PRINTER IS PRT;WIDTH 108
25474                  PRINT CHR$(12)      !Form feed
25475                  PRINT Data_set_title$;TAB(60);"OBSERVATION # ";Num_obs_prin
ted
25476                  IF Num_obs_printed>0 AND Num_obs_printed<=Max_obs_rec THEN
25477                      PRINT
25478                      PRINT "TEST  ";Obs_rec(1,Num_obs_printed)
25479                      PRINT "RUN   ";Obs_rec(2,Num_obs_printed);TAB(18);"LOG
FILE  ";Log_fn$
25480                      PRINT "POINT ";Obs_rec(3,Num_obs_printed);TAB(46);"LOCA
L";TAB(69);"LOCAL"
25481                      PRINT TAB(46);"WALL";TAB(69);"FLOOR"
25482                      PRINT TAB(23);"TUNNEL";TAB(46);"PROBE";TAB(69);"PROBE"
25483                      U_$=CHR$(27)&"&dD"!Underline
25484                      Nu_$=CHR$(27)&"&d@"!No underline
25485                      PRINT TAB(23);U_$&"CONDITIONS"&Nu_$;TAB(54);U_$&"CONDIT
IONS"&Nu_$;TAB(85);U_$&"CONDITIONS"&Nu_$
25486                      PRINT
25487                      PRINT "Mach";TAB(23);Obs_rec(9,Num_obs_printed)
```

```
25488            PRINT "Reynolds No.";TAB(23);Obs_rec(10,Num_obs_printed)
25489            PRINT "Pt";TAB(23);Obs_rec(7,Num_obs_printed);TAB(46);A_rec(4,Num_obs_printed);TAB(69);B_rec(4,Num_obs_printed)
25490            PRINT "Ps";TAB(23);Obs_rec(6,Num_obs_printed);TAB(46);A_rec(5,Num_obs_printed);TAB(69);B_rec(5,Num_obs_printed)
25491            PRINT "Tt";TAB(23);Obs_rec(8,Num_obs_printed);TAB(46);A_rec(6,Num_obs_printed);TAB(69);B_rec(6,Num_obs_printed)
25492            PRINT "Velocity";TAB(23);Obs_rec(46,Num_obs_printed);TAB(46);A_rec(7,Num_obs_printed);TAB(69);B_rec(7,Num_obs_printed)
25493            PRINT "Density";TAB(23);Obs_rec(47,Num_obs_printed);TAB(46);A_rec(8,Num_obs_printed);TAB(69);B_rec(8,Num_obs_printed)
25494            PRINT "LOG(RhoU)";TAB(46);A_rec(13,Num_obs_printed);TAB(69);B_rec(13,Num_obs_printed)
25495            PRINT
25496            PRINT "MEAN(HW1)";TAB(46);A_rec(18,Num_obs_printed);TAB(69);B_rec(18,Num_obs_printed)
25497            PRINT "MEAN(HW2)";TAB(46);A_rec(19,Num_obs_printed);TAB(69);B_rec(19,Num_obs_printed)
25498            PRINT "MEAN(HW3)";TAB(46);A_rec(20,Num_obs_printed);TAB(69);B_rec(20,Num_obs_printed)
25499            PRINT
25500            PRINT "S(U)  (HW1)";TAB(46);A_rec(30,Num_obs_printed);TAB(69);B_rec(30,Num_obs_printed)
25501            PRINT "S(Rho)(HW1)";TAB(46);A_rec(34,Num_obs_printed);TAB(69);B_rec(34,Num_obs_printed)
25502            PRINT "S(To) (HW1)";TAB(46);A_rec(38,Num_obs_printed);TAB(69);B_rec(38,Num_obs_printed)
25503            PRINT "S(U)  (HW2)";TAB(46);A_rec(31,Num_obs_printed);TAB(69);B_rec(31,Num_obs_printed)
25504            PRINT "S(Rho)(HW2)";TAB(46);A_rec(35,Num_obs_printed);TAB(69);B_rec(35,Num_obs_printed)
25505            PRINT "S(To) (HW2)";TAB(46);A_rec(39,Num_obs_printed);TAB(69);B_rec(39,Num_obs_printed)
25506            PRINT "S(U)  (HW3)";TAB(46);A_rec(32,Num_obs_printed);TAB(69);B_rec(32,Num_obs_printed)
25507            PRINT "S(Rho)(HW3)";TAB(46);A_rec(36,Num_obs_printed);TAB(69);B_rec(36,Num_obs_printed)
25508            PRINT "S(To) (HW3)";TAB(46);A_rec(40,Num_obs_printed);TAB(69);B_rec(40,Num_obs_printed)
25509            PRINT
25510            PRINT "u'/U (rms)";TAB(46);A_rec(47,Num_obs_printed);TAB(69);B_rec(47,Num_obs_printed)
25511            PRINT "p'/P (rms)";TAB(46);A_rec(48,Num_obs_printed);TAB(69);B_rec(48,Num_obs_printed)
25512            PRINT "to'/To (rms)";TAB(46);A_rec(49,Num_obs_printed);TAB(69);B_rec(49,Num_obs_printed)
25513            PRINT
25514            PRINT "R(RhoU)";TAB(46);A_rec(26,Num_obs_printed);TAB(69);B_rec(26,Num_obs_printed)
25515            PRINT "R(UTO)";TAB(46);A_rec(27,Num_obs_printed);TAB(69);B_rec(27,Num_obs_printed)
25516            PRINT "R(RhoTO)";TAB(46);A_rec(28,Num_obs_printed);TAB(69);B_rec(28,Num_obs_printed)
25517            PRINT
25518            PRINT "M'/M";TAB(46);A_rec(41,Num_obs_printed);TAB(69);B_rec(41,Num_obs_printed)
25519            PRINT "P'/P";TAB(46);A_rec(46,Num_obs_printed);TAB(69);B_rec(46,Num_obs_printed)
25520  !
```

```
25521                 END IF
25522  !
25523                 PRINTER IS CRT
25524             END WHILE
25525  !
25526             RETURN
25527  !
25528  !
25529 End_4107:END SELECT
25530  !
25531  !
25532    CASE 4108
25533        SELECT Routinl
25534        CASE 31
25535 Load_data: !
25536 ! THIS ROUTINE IS THE ACQUISITION AND DATA LOGGING PORTION OF THE HOTWIRE
25537 ! DYNAMIC DATA SYSTEM.
25538 !
25539 ! STEVE CLUKEY, VIGYAN RESEARCH ASSOCIATES, SEPTEMBER, 1987
25540 !
25541 ! DATA IS RECEIVED FROM ANOTHER COMPUTER (MODCOMP) VIA HP-IB IN ASCII.
25542 ! UPON REQUEST OF THE OPERATOR, THIS DATA IS AVERAGED, AND STORED (LOGGED)
25543 ! ON DISK AS AN OBSERVATION IN AN X-Y ARRAY. SOME VALUES ARE CALCULATED
25544 ! AND STORED ALONG WITH THE DATA RECEIVED FROM THE OTHER COMPUTER.
25545 ! THESE DATA OBSERVATIONS WILL BE USED TO COMPUTE HOTWIRE SENSITIVITIES,
25546 ! AND THEREFORE THE FORMAT OF THE DISK FILE SHOULD BE COMPATIBLE WITH
25547 ! THE COMPUTATIONAL PROGRAMS.
25548 !
25549 ! FOR EACH OBSERVATION, "Max_vars" VARIABLES ARE STORED:
25550 !
25551 !
25552 !   THE PROCESS IS:
25553 !
25554 !1. CREATE AN OBSERVATION FILE, OR DECLARE AN EXISTING FILE, AND SET UP
25555 !   TO APPEND (WITH ADDITIONAL RECORDS) ADDITIONAL OBSERVATIONS.
25556 !2. WHEN REQUESTED BY THE OPERATOR, DATA IS READ FROM THE OTHER COMPUTER
25557 !   VIA HP-IB. BOTH THE DATA HOTWIRE DATA AND THE TUNNEL PARAMETER DATA
25558 !   ARE AVERAGED, AND STORED IN THE NEXT AVAILABLE RECORD.
25559 !3. WHEN REQUESTED BY THE OPERATOR, THE FILE IS CLOSED.
25560 !
25561 ! AT SOME LATER TIME, THE FILE WILL BE PROCESSED TO PROVIDE A HOTWIRE
25562 ! SENSITIVITY FILE.
25563 !
25564 !
25565 !
25566             CALL Pline(0,"OBSERVATION LOGGING STARTED")
25567             WAIT 1
25568 !
25569             Pkt_sc=8
25570             MAT Pkt_avg= (0)
25571             I=0
25572             FOR I=1 TO Num_avgs
25573                 GOSUB Enable_link
25574                 GOSUB Get_packet
25575                 IF I=1 THEN !Don't average test,run,point......
25576                     FOR J=1 TO 3
25577                         Pkt_avg(J)=VAL(Pkt$(J+1))*Num_avgs
25578                     NEXT J
25579                     FOR J=4 TO 5
25580                         Pkt_avg(J)=0! THIS WAS DATE AND TIME
```

```
25581                       NEXT J
25582                       FOR J=1 TO 10! SAVE THE FIRST 10 FOR "TAG PLOT,PICTURE"
25583                           Tag_pkt$(J)=Pkt$(J+1)
25584                       NEXT J
25585                   END IF
25586                   FOR J=6 TO Rcvd_vars
25587                       ON ERROR GOTO 25589!Ignore error if not a good VAL
25588                       Pkt_avg(J)=Pkt_avg(J)+VAL(Pkt$(J+1))
25589                       ON ERROR CALL Error
25590                   NEXT J
25591               NEXT I
25592 !!!!!         GOSUB Disable_link
25593               MAT Pkt_avg= Pkt_avg/(Num_avgs)! FIND THE AVERAGE
25594 !
25595 ! Get ready to log data - by opening files
25596               ASSIGN @Disk TO Log_fn$
25597               ASSIGN @Diska TO Log_fn$&"A"
25598               ASSIGN @Diskb TO Log_fn$&"B"
25599 !!!!                          ASSIGN @Diskc TO Log_fn$&"C"
25600 !
25601 ! Find out where we are - retrieve number of observations recorded
25602 !
25603               ENTER @Disk,1;Data_set_title$,Max_obs_rec,Max_vars,Logged_var_n
ame$(*),Numsubfile,Subfile_names$(*),Subfile_chartst(*)
25604               Num_obs_recd=Subfile_chartst(1)
25605               This_obs=MAX(Num_obs_recd+1,1)! bump the observations pointer
25606               IF This_obs<=Max_obs_rec THEN ! move the data to the observatio
n record
25607                   FOR I=1 TO Rcvd_vars
25608                       Obs_rec(I,This_obs)=Pkt_avg(I)
25609                   NEXT I
25610                   Num_obs_recd=This_obs! prepare to save the observation poin
ter
25611                   Ending_obs=Num_obs_recd
25612                   Subfile_chartst(1)=Num_obs_recd
25613 !
25614                   FOR I=1 TO Nummem
25615!                          SET UP DATA FILE MAP FOR TRACES 1 THRU NUMMEM
25616                       Mivar(14)=I
25617                       Mvar(2)=1
25618                       CALL Routine(22,239)   !TURN ON data files
25619!
25620                       Temp$[1]="R"
25621                       SELECT I
25622                       CASE 1 TO 4
25623                           Temp$[2]="A"
25624                       CASE 5 TO 7
25625                           Temp$[2]="B"
25626                       CASE 8 TO 99
25627                           Temp$[2]="C"
25628                       END SELECT
25629                       SELECT INT(Obs_rec(2,This_obs))
25630                       CASE 0 TO 9
25631                           Temp$[3,4]="0"&VAL$(INT(Obs_rec(2,This_obs)))!RUN
25632                       CASE 10 TO 99
25633                           Temp$[3,4]=VAL$(INT(Obs_rec(2,This_obs)))!RUN
25634                       END SELECT
25635                       SELECT INT(Obs_rec(3,This_obs))
25636                       CASE 0 TO 9
25637                           Temp$[5,6]="0"&VAL$(INT(Obs_rec(3,This_obs)))!POINT
```

```
25638                    CASE 10 TO 99
25639                        Temp$[5,6]=VAL$(INT(Obs_rec(3,This_obs)))!POINT
25640                    END SELECT
25641                    SELECT I
25642                    CASE 1 TO 9
25643                        Temp$[7,8]="0"&VAL$(I)
25644                    CASE 10 TO 99
25645                        Temp$[7,8]=VAL$(I)
25646                    END SELECT
25647                    Filename$(I)=Temp$
25648                    Mivar(14)=I
25649                    CALL Routine(21,232) ! DISPLAY THE FILENAME
25650  !!!              PRINTER IS PRT
25651  !!!!             PRINT "FOR OBS # ",Recnum;", THE FILENAME IS: ";Filename
$(I)
25652  !!!!             PRINTER IS CRT
25653                NEXT I
25654                GOSUB Compute_vars
25655 !
25656                GOSUB Log_vars
25657                Ending_obs=Num_obs_recd
25658            ELSE
25659                CALL Pline(0," LOG FILE FULL ")
25660                WAIT 1
25661                CALL Pline(0,"                            ")
25662            END IF
25663 !
25664            CALL Pline(0,"OBSERVATION LOGGING COMPLETE         ")
25665            WAIT 1
25666            CALL Pline(0,"                            ")
25667            GOTO End_4108
25668 !
25669 !
25670 !
25671 Enable_link:! SEND A PACKET TO THE OTHER CPU WHICH CONTAINS THE "GO" WORD
25672 !
25673 !
25674            Go_word$="GO"
25675            OUTPUT Pkt_sc;Go_word$
25676            RETURN
25677 !
25678 !
25679 !
25680 Disable_link:! SEND A PACKET TO THE OTHER CPU WHICH CONTAINS THE "STOP" WO
RD
25681 !
25682            Go_word$="STOP"
25683            OUTPUT Pkt_sc;Go_word$
25684            RETURN
25685 !
25686 !
25687 !
25688 !
25689 !
25690 !
25691 Get_packet: !
25692            ENTER Pkt_sc;Pkt$(*)
25693 !
25694 !!!  PRINTER IS PRT
25695            CALL Pline(0,"RECEIVED PACKET FOR SAMPLE "&VAL$(I)&"   ")
```

```
25696 !!!  FOR Vars_ctr=1 TO RCVD_vars
25697 !!!   PRINT Pkt$(Vars_ctr+1)
25698 !!!  NEXT Vars_ctr
25699 !!!  PRINTER IS CRT
25700           RETURN
25701 !
25702 !
25703 !
25704 !
25705 !
25706 Compute_vars:  ! compute values associated with mean logged_vars
25707           GOSUB Tun_vars
25708           FOR Probe=Initial_probe TO Ending_probe
25709               SELECT Probe
25710               CASE 1
25711                   GOSUB Wall_vars
25712               CASE 2
25713                   GOSUB Floor_vars
25714               CASE 3
25715        ! GOSUB OTHER_VARS
25716               END SELECT
25717           NEXT Probe
25718           RETURN
25719 !
25720 Tun_vars: !
25721 !Construct variables for the tunnel conditions
25722           Strut$="TUNNEL"
25723           GOSUB Compute_u_rho_m!Compute velocity, density, massflow
25724           Obs_rec(46,This_obs)=Local_velocity
25725           Obs_rec(47,This_obs)=Local_density
25726           RETURN
25727 !
25728 Wall_vars: !
25729 ! Construct variables for the wall strut computed variables logfile (A)
25730           A_rec(1,This_obs)=Obs_rec(1,This_obs)!TEST
25731           A_rec(2,This_obs)=Obs_rec(2,This_obs)!RUN
25732           A_rec(3,This_obs)=Obs_rec(3,This_obs)!POINT
25733           A_rec(4,This_obs)=Obs_rec(11,This_obs)!P_TOTAL
25734           A_rec(5,This_obs)=Obs_rec(12,This_obs)!P_STATIC
25735           A_rec(6,This_obs)=Obs_rec(8,This_obs)+460  !T_TOTAL (local) !Us
e TUNNEL total
25736 !
25737           Strut$="WALL"
25738           GOSUB Compute_u_rho_m!Compute velocity,density,mass flow
25739           A_rec(7,This_obs)=Local_velocity
25740           A_rec(8,This_obs)=Local_density
25741           A_rec(9,This_obs)=Ts
25742           IF A_rec(7,This_obs)>0 THEN
25743               A_rec(10,This_obs)=LGT(A_rec(7,This_obs))    !Log(U)
25744           END IF
25745           IF A_rec(8,This_obs)>0 THEN
25746               A_rec(11,This_obs)=LGT(A_rec(8,This_obs))    !Log(Rho)
25747           END IF
25748           IF A_rec(6,This_obs)>0 THEN
25749               A_rec(12,This_obs)=LGT(A_rec(6,This_obs))    !Log(T0)
25750           END IF
25751           IF Local_mass_flow>0 THEN
25752               A_rec(13,This_obs)=LGT(Local_mass_flow)     !Log(Rho_inf*U)
25753           END IF
25754           A_rec(14,This_obs)=A_rec(11,This_obs)*A_rec(12,This_obs)!Log(Rh
```

```
o)*Log(T0)
25755          A_rec(15,This_obs)=A_rec(10,This_obs)*A_rec(11,This_obs)!Log(U)
*Log(Rho)
25756          A_rec(16,This_obs)=A_rec(10,This_obs)*A_rec(12,This_obs)!Log(U)
*Log(T0)
25757          A_rec(17,This_obs)=A_rec(15,This_obs)*A_rec(12,This_obs)!Log(U)
*Log(Rho)*Log(T0)
25758          FOR Wire=1 TO 4
25759              A_rec(17+Wire,This_obs)=Obs_rec(19+Wire,This_obs)
25760              IF A_rec(17+Wire,This_obs)>0 THEN
25761                  A_rec(21+Wire,This_obs)=LGT(A_rec(17+Wire,This_obs))!Lo
g(E)
25762              END IF
25763              SELECT Obs_rec(36,This_obs)
25764              CASE 1 TO 16
25765                  A_rec(41+Wire,This_obs)=Gain_code_14(Obs_rec(36,This_ob
s)+1)
25766              CASE ELSE
25767                  A_rec(41+Wire,This_obs)=0!Default gain=0
25768              END SELECT
25769          NEXT Wire
25770          RETURN
25771 !
25772 Floor_vars: !
25773 ! Construct variables for the floor strut computed variables logfile (B)
25774          B_rec(1,This_obs)=Obs_rec(1,This_obs)!TEST
25775          B_rec(2,This_obs)=Obs_rec(2,This_obs)!RUN
25776          B_rec(3,This_obs)=Obs_rec(3,This_obs)!POINT
25777          B_rec(4,This_obs)=Obs_rec(14,This_obs)!P_TOTAL (local)
25778          B_rec(5,This_obs)=Obs_rec(15,This_obs)!P_STATIC(local)
25779          B_rec(6,This_obs)=Obs_rec(8,This_obs)+460    !T_TOTAL (local)   !
Use TUNNEL total
25780 !
25781          Strut$="FLOOR"
25782          GOSUB Compute_u_rho_m!Compute velocity,density,mass flow
25783          B_rec(7,This_obs)=Local_velocity
25784          B_rec(8,This_obs)=Local_density
25785          B_rec(9,This_obs)=Ts
25786          IF B_rec(7,This_obs)>0 THEN
25787              B_rec(10,This_obs)=LGT(B_rec(7,This_obs))    !Log(U)
25788          END IF
25789          IF B_rec(8,This_obs)>0 THEN
25790              B_rec(11,This_obs)=LGT(B_rec(8,This_obs))    !Log(Rho)
25791          END IF
25792          IF B_rec(6,This_obs)>0 THEN
25793              B_rec(12,This_obs)=LGT(B_rec(6,This_obs))    !Log(T0)
25794          END IF
25795          IF Local_mass_flow>0 THEN
25796              B_rec(13,This_obs)=LGT(Local_mass_flow)     !Log(Rho_inf*U)
25797          END IF
25798          B_rec(14,This_obs)=B_rec(11,This_obs)*B_rec(12,This_obs)!Log(Rh
o)*Log(T0)
25799          B_rec(15,This_obs)=B_rec(10,This_obs)*B_rec(11,This_obs)!Log(U)
*Log(Rho)
25800          B_rec(16,This_obs)=B_rec(10,This_obs)*B_rec(12,This_obs)!Log(U)
*Log(T0)
25801          B_rec(17,This_obs)=B_rec(15,This_obs)*B_rec(12,This_obs)!Log(U)
*Log(Rho)*Log(T0)
25802          FOR Wire=1 TO 3
25803              B_rec(17+Wire,This_obs)=Obs_rec(23+Wire,This_obs)
```

```
                IF B_rec(17+Wire,This_obs)>0 THEN
                    B_rec(21+Wire,This_obs)=LGT(B_rec(17+Wire,This_obs))
                END IF
                SELECT Obs_rec(36+Wire,This_obs)
                CASE 1 TO 7
                    B_rec(41+Wire,This_obs)=Gain_code_57(Obs_rec(36+Wire,This_obs)+1)
                CASE ELSE
                    B_rec(41+Wire,This_obs)=0!Default gain=0
                END SELECT
            NEXT Wire
            RETURN
!
Other_vars: !
! Construct variables for the 'other' strut computed variables logfile (C)
            C_rec(1,This_obs)=Obs_rec(1,This_obs)!TEST
            C_rec(2,This_obs)=Obs_rec(2,This_obs)!RUN
            C_rec(3,This_obs)=Obs_rec(3,This_obs)!POINT
            C_rec(4,This_obs)=Obs_rec(17,This_obs)!P_TOTAL (local)
            C_rec(5,This_obs)=Obs_rec(18,This_obs)!P_STATIC(local)
            C_rec(6,This_obs)=Obs_rec(19,This_obs)!T_TOTAL (local)
!
            Strut$="OTHER"
            GOSUB Compute_u_rho_m!Compute velocity,density,mass flow
            C_rec(7,This_obs)=Local_velocity
            C_rec(8,This_obs)=Local_density
            C_rec(9,This_obs)=Ts
            IF C_rec(7,This_obs)>0 THEN
                C_rec(10,This_obs)=LGT(C_rec(7,This_obs))    !Log(U)
            END IF
            IF C_rec(8,This_obs)>0 THEN
                C_rec(11,This_obs)=LGT(C_rec(8,This_obs))    !Log(Rho)
            END IF
            IF C_rec(6,This_obs)>0 THEN
                C_rec(12,This_obs)=LGT(C_rec(6,This_obs)+460)    !Log(T0)
            END IF
            IF Local_mass_flow>0 THEN
                C_rec(13,This_obs)=LGT(Local_mass_flow)    !Log(Rho_inf*U)
            END IF
            C_rec(14,This_obs)=C_rec(11,This_obs)*C_rec(12,This_obs)!Log(Rho)*Log(T0)
            C_rec(15,This_obs)=C_rec(10,This_obs)*C_rec(11,This_obs)!Log(U)*Log(Rho)
            C_rec(16,This_obs)=C_rec(10,This_obs)*C_rec(12,This_obs)!Log(U)*Log(T0)
            C_rec(17,This_obs)=C_rec(15,This_obs)*C_rec(12,This_obs)!Log(U)*Log(Rho)*Log(T0)
            FOR Wire=1 TO 3
                C_rec(17+Wire,This_obs)=Obs_rec(23+Wire,This_obs)
                IF C_rec(17+Wire,This_obs)>0 THEN
                    C_rec(21+Wire,This_obs)=LGT(C_rec(17+Wire,This_obs))
                END IF
            NEXT Wire
            RETURN
!
!
!
!
Compute_u_rho_m:   !COMPUTE LOCAL VELOCITY,DENSITY,MASS FLOW
            SELECT Strut$
```

```
25859              CASE "TUNNEL"
25860                  Pt=Obs_rec(7,This_obs)
25861                  Ps=Obs_rec(6,This_obs)
25862                  Tt=Obs_rec(8,This_obs)+460
25863              CASE "WALL"
25864                  Pt=A_rec(4,This_obs)
25865                  Ps=A_rec(5,This_obs)
25866                  Tt=A_rec(6,This_obs)
25867              CASE "FLOOR"
25868                  Pt=B_rec(4,This_obs)
25869                  Ps=B_rec(5,This_obs)
25870                  Tt=B_rec(6,This_obs)
25871              CASE "OTHER"
25872                  Pt=C_rec(4,This_obs)
25873                  Ps=C_rec(5,This_obs)
25874                  Tt=C_rec(6,This_obs)
25875              END SELECT
25876 !
25877              IF Ps=0 THEN
25878                  Prat=0
25879                  GOTO 25883
25880              END IF
25881              P_rat=Ps/Pt
25882              IF P_rat>1 THEN P_rat=1
25883              IF P_rat<=0 THEN
25884                  Ts=MAX(.1,Tt)
25885                  Local_velocity=0.
25886                  Local_density=Ps/(53.3*Ts)
25887                  Local_mass_flow=0.
25888              ELSE
25889                  Ts=(Tt)/(P_rat^(-.285714285))
25890                  IF P_rat=1 THEN
25891                      Local_velocity=0.
25892                  ELSE
25893                      Sound=SQR(2402.764*Ts)
25894                      Local_mach=SQR(5*(P_rat^(-.285714285)-1))
25895                      Local_velocity=Local_mach*Sound! U
25896                  END IF
25897                  Local_density=Ps/(53.3*Ts)      ! Rho
25898                  Local_mass_flow=Local_velocity*Local_density! M
25899              END IF
25900              RETURN
25901 !
25902 !
25903 !
25904 Log_vars: ! LOG THE CURRENT OBSERVED VARIABLES
25905 !
25906 !
25907              ASSIGN @Disk TO Log_fn$
25908 !
25909              OUTPUT @Disk,1;Data_set_title$,Max_obs_rec,Max_vars,Logged_var_
name$(*),Numsubfile,Subfile_names$(*),Subfile_chartst(*)
25910              ENTER @Disk,2
25911              OUTPUT @Disk;Obs_rec(*)
25912              ASSIGN @Disk TO *
25913 !
25914              ASSIGN @Diska TO Log_fn$&"A"
25915 !
25916              OUTPUT @Diska,1;A_set_title$,Max_obs_rec,Max_vars,A_var_name$(*
),Numsubfile,A_subfile_names$(*),A_sub_chartst(*)
```

```
25917            ENTER @Diska,2
25918            OUTPUT @Diska;A_rec(*)
25919            ASSIGN @Diska TO *
25920 !
25921            ASSIGN @Diskb TO Log_fn$&"B"
25922 !
25923            OUTPUT @Diskb,1;B_set_title$,Max_obs_rec,Max_vars,B_var_name$(*
),Numsubfile,B_subfile_names$(*),B_sub_chartst(*)
25924            ENTER @Diskb,2
25925            OUTPUT @Diskb;B_rec(*)
25926            ASSIGN @Diskb TO *
25927 !
25928 !!!!               ASSIGN @DISKC TO LOG_FN$&"C"
25929 !
25930 !!!!               OUTPUT @Diskc,1;C_set_title$,Max_obs_rec,Max_vars,C_var
_name$(*),Numsubfile,C_subfile_names$(*),C_sub_chartst(*)
25931 !!!!               ENTER @Diskc,2
25932 !!!!               OUTPUT @Diskc;C_rec(*)
25933 !!!!               ASSIGN @Diskc TO *
25934 !
25935 !
25936            RETURN
25937 !
25938 !
25939 !
25940 End_4108: !
25941         END SELECT
25942      CASE 4110    !            "SAMPLES TO AVG"
25943         SELECT Routinl
25944         CASE 21
25945            Mvar(2)-Num_avgs
25946            Mvar(4)-1
25947            Mvar(5)-1.
25948            Mvar(6)-300.
25949            Mivar(9)-0
25950         CASE 22
25951            Num_avgs-Mvar(2)
25952         END SELECT
25953      CASE 4112    !            "INITIAL OBS"
25954         SELECT Routinl
25955         CASE 21
25956            Mvar(2)-Initial_obs
25957            Mvar(4)-1
25958            Mvar(5)-1.
25959            Mvar(6)-300.
25960            Mivar(9)-0
25961         CASE 22
25962            Initial_obs-Mvar(2)
25963         END SELECT
25964      CASE 4113    !            "ENDING OBS"
25965         SELECT Routinl
25966         CASE 21
25967            Mvar(2)-Ending_obs
25968            Mvar(4)-1
25969            Mvar(5)-1.
25970            Mvar(6)-300.
25971            Mivar(9)-0
25972         CASE 22
25973            Ending_obs-Mvar(2)
25974         END SELECT
```

```
25975 !
25976 !
25977 !
25978    CASE 4130   !            "INITIAL PROBE"
25979        SELECT Routinl
25980        CASE 21
25981            Mvar(2)-Initial_probe
25982            Mvar(4)-1
25983            Mvar(5)-1.
25984            Mvar(6)-2.
25985            Mivar(9)-0
25986        CASE 22
25987            Initial_probe-Mvar(2)
25988        END SELECT
25989    CASE 4131   !            "ENDING PROBE"
25990        SELECT Routinl
25991        CASE 21
25992            Mvar(2)-Ending_probe
25993            Mvar(4)-1
25994            Mvar(5)-1.
25995            Mvar(6)-2.
25996            Mivar(9)-0
25997        CASE 22
25998            Ending_probe-Mvar(2)
25999        END SELECT
26000 !
26001 !
26002 !
26003    CASE 4127   !            "LOGFILE TO PC"
26004        SELECT Routinl
26005        CASE 31
26006            Pc_device-8       ! Send data out thru HP-IB bus 8
26007                              ! bus 8 is NOT system controller
26008          FOR Probe-Initial_probe TO Ending_probe
26009            SELECT Probe
26010            CASE 1
26011                OUTPUT Pc_device;VAL$(Ending_obs-Initial_obs+1),VAL$(Max_va
rs)
26012                FOR I-1 TO Max_vars
26013                    DISP "NAME("&VAL$(I)&") IS: "&A_var_name$(I)
26014                    OUTPUT Pc_device;A_var_name$(I)[1,10]
26015                NEXT I
26016                FOR This_obs-Initial_obs TO Ending_obs
26017                    FOR I-1 TO Max_vars
26018                        OUTPUT Pc_device;A_rec(I,This_obs)
26019                    NEXT I
26020                NEXT This_obs
26021            CASE 2
26022                OUTPUT Pc_device;VAL$(Ending_obs-Initial_obs+1),VAL$(Max_va
rs)
26023                FOR I-1 TO Max_vars
26024                    DISP "NAME("&VAL$(I)&") IS: "&B_var_name$(I)
26025                    OUTPUT Pc_device;B_var_name$(I)[1,10]
26026                NEXT I
26027                FOR This_obs-Initial_obs TO Ending_obs
26028                    FOR I-1 TO Max_vars
26029                        OUTPUT Pc_device;B_rec(I,This_obs)
26030                    NEXT I
26031                NEXT This_obs
26032            END SELECT
```

```
26033          NEXT Probe
26034        END SELECT
26035    END SELECT
26036 END SELECT
26037 SUBEND
26038 !
26039 !
26040 !
26041 !
```

APPENDIX B. Function definitions

This appendix contains the DDAS functions added to ACQUIRE by the user to provide the necessary functionality to acquire, process, display, store and transmit the hot wire data.

The function definition sheets provide essential definition data for each function. Full documentation for each function is contained within the program source code listings.

Function Name: CALC VEL etc
Function Number: 4104
Module: MODUSR2*

This function computes velocity, density and temperature fluctuations as ratios of the fluctuating quantitys to the mean quantitys:

u'/U p'/P t'/T

The log file for each observation in the range of INITIAL OBS to ENDING OBS is processed for each probe in the range of INITIAL PROBE to ENDING PROBE.
For each observation, and each probe, the three fluctuating data files related to the Run and Point (variables 2 and 3 in the observation record) are loaded into traces (and memories) 1,2, and 3.

The computations retrieve mean values, sensitivities, and gains related to the data in channels 1, 2 and 3.
For each simultaneous sample in each of the three traces (the beginning and ending samples are defined by the cursors on trace 1) the instantaneous value is divided by the equivalent mean value and the gain. then the three ratios are matrix multiplied by the nmatrix inversion of the sensitivities. This process essentially solves a set of simultaneous equations for three unknowns: u'/U, p'/P, and t'/T.

The solutions are placed in traces (memorys) 4, 5 and 6. The mean value is removed, and the RMS (root mean square) value of each of the answers is stored in the logfile for the appropriate observation and probe. Traces 4, 5, and 6 are then stored in separate disk files.

[CALC VEL etc]----------------|

*. This routine - MODUSR2 - was written by Steven J. Clukey of Vigyan Research Associates, Inc. for NASA LaRC TAD/FDB under contract NAS1-17919, Task 36. This work was done beginning in October, 1986 and continues thru April,1988.

Function Name: CAT GROUP

Function Number: 4021

Module: MODUSR1*

This function catalogs all selected files. The selection criteria is defined in detail in the HP BASIC language refernce manual for HP function CAT (SELECT).

```
[ CAT GROUP ]----------------------------|
                |                        |
                |                        |
                |--[ = ]--[ name ]--|
```

Item	Description	Range
name	string expression	any valid characters that are allowed in a file name

*. This routine - MODUSR1 - was written by Steven J. Clukey of Vigyan Research Associates, Inc. for NASA LaRC TAD/FDB under contract NAS1-17919, Task 36. This work was done beginning in October, 1986 and continues thru April,1988.

Function Name: CODE TO GAINS
Function Number: 4125
Module: MODUSR2*

This function computes actual voltage gains fom gain codes. The fluctuating data is gained to provide adaquate voltage for filtering an d digitization, and the gain codes are sent in a data packet from the static data computer during the logging of a data point. For the probes in the range INITIAL PLROBE to ENDING PROBE, and for observations in the range INITIAL OBS to ENDING OBS, the gain codes are retrieved from the observation file, and thru a table lookup algorithm, the actual gains are retrieved, and stored back into the appropriate place in the logfile for ht eprobe and observation being processed.

[CODE TO GAINS]---------------|

*. This routine - MODUSR2 - was written by Steven J. Clukey of Vigyan Research Associates, Inc. for NASA LaRC TAD/FDB under contract NAS1-17919, Task 36. This work was done beginning in October, 1986 and continues thru April,1988.

Function Name: COEF FILENAME
Function Number: 4101
Module: MODUSR2*

This function selects the file name relaven to the hotwire coeficient file being processed. See functions LOAD COEFS and STORE COEFS.

[COEF FILENAME]---------------|

*. This routine - MODUSR2 - was written by Steven J. Clukey of Vigyan Research Associates, Inc. for NASA LaRC TAD/FDB under contract NAS1-17919, Task 36. This work was done beginning in October, 1986 and continues thru April,1988.

Function Name: COMPUTE COEFS
Function Number: 4123
Module: MODUSR2

This function computes hotwire coefficients for the probe previously selected - by function INITIAL PROBE - utilizing a custom multiple linear regression rouutine that generates up to 10 coefficients based on calibration data already in the logfile. These coefficients are then stored - by function STORE COEFS - in a coefficient file whose name has been previously defined - by function COEF FILENAME. Function COMPUTE SENS utilizes these coefficients to generate senstivities necessary to compute velocity, density and temperature turbulence ratios -by function CALC VAL etc.

[COMP COEFS]---------------|

This routine - MODUSR2 - was written by Steven J. Clukey of Vigyan Research Associates, Inc. for NASA LaRC TAD/FDB under contract NAS1-17919, Task 36. This work was done beginning in October, 1986 and continues thru April,1988.

Function Name: COMPUTE R etc
Function Number: 4132
Module: MODUSR2

This function computes the correlations between velocity, density and temperature fluctuations and then computes mass flow fluctuation [m'/M(rms)] and pressure fluctuation [p'/P(rms)] using the correlation between velocity, density, and temperature fluctuations.

The log file for each observation in the range of INITIAL OBS to ENDING OBS is processed for each probe in the range of INITIAL PROBE to ENDING PROBE.

For each observation, and each probe, the three fluctuating data files related to the Run and Point (variables 2 and 3 in the observation record) are loaded into traces (and memories) 2, 3, and 4.

As each pair of traces is multiplied together, the resulting trace ends up in trace 1. The correlation of the two traces multiplied is the ratio of the rms value of the first trace multiplied by the rms value of the second trace to the mean value of trace 1. The rms values of the first and second traces have been previously calculated by function CALC VEL etc, and were called u'/U, p'/P, and t'/T.

The correlation between these three components of turbulence are stored in the appropriate observation record for the observation and probe being processed as R(RhoU), R(UTO), and R(RhoTO).

Massflow and pressure fluctuations are then computed from the correlations just computed, and these are also stored in the appropriate logfile as M'/M and P'/P.

[COMPUTE R etc]----------------|

This routine - MODUSR2 - was written by Steven J. Clukey of Vigyan Research Associates, Inc. for NASA LaRC TAD/FDB under contract NAS1-17919, Task 36. This work was done beginning in October, 1986 and continues thru April,1988.

Function Name: COMPUTE SENS

Function Number: 4111
Module: MODUSR2

This function computes hotwire sensitivities for the probes in the range INITIAL PROBE to ENDING PROBE for all observations in the range INITIAL OBS to ENDING OBS. The hotwire coefficients previously defined - see functions GET COEFS, COEF FILENAME, LOAD COEFS, ENTER COEFS, EDIT COEFS, STORE COEFS, AND COMPUTE COEFS.

The computed sensitivities are stored in the appropriate probe file for each appropriate observation.

These senstivities are read from the logfile - by function CALC VAL etc - to compute velocity, density and temperature turbulence ratios.

[COMPUTE SENS]---------------|

This routine - MODUSR2 - was written by Steven J. Clukey of Vigyan Research Associates, Inc. for NASA LaRC TAD/FDB under contract NAS1-17919, Task 36. This work was done beginning in October, 1986 and continues thru April,1988.

Function Name: EDIT COEFS
Function Number: 4133
Module: MODUSR2*

This function allows the operator to manually edit hot wire calibration coefficients (up to 10) for three wires. These coefficients have been previously generated. The operator views a list of the entered coefficients, and, by responding to prompts, select the coefficient to be edited. After each coefficient is edited, the operator may accept the coefficients, or be prompted to select another coefficient to be edited. The operator should then invoke function STORE COEFS.

[EDIT COEFS]----------------|

NOTE: This function is highly interactive, and is <u>not recommended</u> for inclusion in a SEQUENCE PROGRAM.

*. This routine - MODUSR2 - was written by Steven J. Clukey of Vigyan Research Associates, Inc. for NASA LaRC TAD/FDB under contract NAS1-17919, Task 36. This work was done beginning in October, 1986 and continues thru April,1988.

Function Name: ENDING OBS
Function Number: 4113
Module: MODUSR2*

This function declares the ending observation to be processed by other MODUSR2 functions - see function INITIAL OBS. Functions utilizing this feature to define the range of observations to be processed are: COMPUTE SENS, LOG DATA POINT, CODE TO GAINS, CALC VEL etc, COMPUTE R etc, Remake probe and LOGFILE TO PC.

```
[ ENDING OBS ]-----------------------------|
                  |                       |
                  |                       |
                  |--[ = ]--[value]--|
```

Item	Description	Range
value	numeric integer	1 to 300, the max number of observations

*. This routine - MODUSR2 - was written by Steven J. Clukey of Vigyan Research Associates, Inc. for NASA LaRC TAD/FDB under contract NAS1-17919, Task 36. This work was done beginning in October, 1986 and continues thru April,1988.

Function Name: ENTER COEFS
Function Number: 4103
Module: MODUSR2*

This function allows the operator to manually enter hot wire calibration coefficients (up to 10) for three wires. These coefficients have been previously generated elsewhere, and are not available for entry via disc (see function LOAD COEFS). The operator is prompted for each coefficient by wire, number, and name. Once all 30 coefficients are entered (unused coefficients should be set to 0.0), the operator views a list of the entered coefficients, and chooses to accept or reject the coefficients. If they aree accepted, the function is complete. The operator should then invoke function STORE COEFS. If the coefficients are not accepted - because they are not correct, this function <u>automatically</u> enters the EDIT COEFS function.

[ENTER COEFS]---------------|

NOTE: This function is highly interactive, and is <u>not recommended</u> for inclusion in a SEQUENCE PROGRAM.

*. This routine - MODUSR2 - was written by Steven J. Clukey of Vigyan Research Associates, Inc. for NASA LaRC TAD/FDB under contract NAS1-17919, Task 36. This work was done beginning in October, 1986 and continues thru April,1988.

Function Name: FILE COPY
Function Number: 4025
Module: MODUSR1*

This function serves merely as a label for the path for accessing other related functions via the softkeys.

```
[ FILE COPY ]----------------|
```

*. This routine - MODUSR1 - was written by Steven J. Clukey of Vigyan Research Associates, Inc. for NASA LaRC TAD/FDB under contract NAS1-17919, Task 36. This work was done beginning in October, 1986 and continues thru April,1988.

Function Name: FILE GROUP
Function Number: 4028
Module: MODUSR1*

This function defines the files to be selected for copying - function COPY FILES - or moving - function MOVE FILES. the files selected will begin with, or be equal to the character(s) defined by this function. For example, if this function defines "ABC", then all of the files in the TO DISK device that begin with "ABC" would be selected for copying or moving. Note that this function only defines the character(s): no selection is done until COPY FILES or MOVE FILES is invoked.

```
[ FILE GROUP ]-----------------------------|
                      |                     |
                      |                     |
                      |--[ = ]--[ name ]--|
```

Item	Description	Range
name	string expression	any valid characters that are allowed in a file name

*. This routine - MODUSR1 - was written by Steven J. Clukey of Vigyan Research Associates, Inc. for NASA LaRC TAD/FDB under contract NAS1-17919, Task 36. This work was done beginning in October, 1986 and continues thru April,1988.

Function Name: FILE TRANSFERS
Function Number: 4126
Module: MODUSR2*

This function serves merely as a label for the path for accessing other related functions via the softkeys.

[FILE TRANSFERS]---------------|

*. This routine - MODUSR2 - was written by Steven J. Clukey of Vigyan Research Associates, Inc. for NASA LaRC TAD/FDB under contract NAS1-17919, Task 36. This work was done beginning in October, 1986 and continues thru April,1988.

Function Name: FILE UTILITYS
Function Number: 4020
Module: MODUSR1*

This function serves merely as a label for the path for accessing other related functions via the softkeys.

```
[ FILE UTILITYS ]----------------|
```

*. This routine - MODUSR1 - was written by Steven J. Clukey of Vigyan Research Associates, Inc. for NASA LaRC TAD/FDB under contract NAS1-17919, Task 36. This work was done beginning in October, 1986 and continues thru April,1988.

Function Name: FROM DISK
Function Number: 4026
Module: MODUSR1*

This function defines the mass storage device **from** which the files will be copied - function COPY FILES - or moved - function MOVE FILES.

```
[ FROM DISK    ]------------------------------|
                 |                   |
                 |                   |
                 |--[ = ]--[ MSI]--|
```

Item	Description	Range
value	MSI	any valid HP storage device

*. This routine - MODUSR1 - was written by Steven J. Clukey of Vigyan Research Associates, Inc. for NASA LaRC TAD/FDB under contract NAS1-17919, Task 36. This work was done beginning in October, 1986 and continues thru April,1988.

Function Name: GET COEF
Function Number: 4122
Module: MODUSR2*

This function serves merely as a label for the path for accessing other related functions via the softkeys.

[GET COEF]----------------|

*. This routine - MODUSR2 - was written by Steven J. Clukey of Vigyan Research Associates, Inc. for NASA LaRC TAD/FDB under contract NAS1-17919, Task 36. This work was done beginning in October, 1986 and continues thru April,1988.

Function Name: HOTWIRE MENU
Function Number: 4100
Module: MODUSR2*

This function serves merely as a label for the path for accessing other related functions via the softkeys.

[HOTWIRE MENU]---------------|

*. This routine - MODUSR2 - was written by Steven J. Clukey of Vigyan Research Associates, Inc. for NASA LaRC TAD/FDB under contract NAS1-17919, Task 36. This work was done beginning in October, 1986 and continues thru April,1988.

Function Name: HOTWIRE CALC
Function Number: 4100
Module: MODUSR2*

This function serves merely as a label for the path for accessing other related functions via the softkeys.

```
[ HOTWIRE CALC ]----------------|
```

*. This routine - MODUSR2 - was written by Steven J. Clukey of Vigyan Research Associates, Inc. for NASA LaRC TAD/FDB under contract NAS1-17919, Task 36. This work was done beginning in October, 1986 and continues thru April,1988.

Function Name: INITIAL OBS
Function Number: 4112
Module: MODUSR2*

This function declares the beginning observation to be processed by other MODUSR2 functions - see function ENDING OBS. Functions utilizing this feature to define the range of observations to be processed are: COMPUTE SENS, LOG DATA POINT, CODE TO GAINS, CALC VEL etc, COMPUTE R etc, Remake probe and LOGFILE TO PC.

```
[ INITIAL OBS ]-----------------------------|
               |                        |
               |                        |
               |--[ = ]--[value]--|
```

Item	Description	Range
value	numeric integer	1 to 300, the max number of observations

*. This routine - MODUSR2 - was written by Steven J. Clukey of Vigyan Research Associates, Inc. for NASA LaRC TAD/FDB under contract NAS1-17919, Task 36. This work was done beginning in October, 1986 and continues thru April,1988.

Function Name: INITIAL PROBE
Function Number: 4130
Module: MODUSR2*

This function declares the <u>beginning</u> probe to be processed by other MODUSR2 functions - see function ENDING PROBE. Functions utilizing this feature to define the range of probes to be processed are: COMPUTE SENS, LOG DATA POINT, CODE TO GAINS, CALC VEL etc, COMPUTE R etc, Remake Probe and LOGFILE TO PC.

```
[ INITIAL PROBE ]-----------------------------|
                 |                   |
                 |                   |
                 |--[ = ]--[value]--|
```

Item	Description	Range
value	numeric integer	1 to 3, the max number of probes

*. This routine - MODUSR2 - was written by Steven J. Clukey of Vigyan Research Associates, Inc. for NASA LaRC TAD/FDB under contract NAS1-17919, Task 36. This work was done beginning in October, 1986 and continues thru April,1988.

Function Name: LOAD COEFS
Function Number: 4102
Module: MODUSR2*

This function loads coefficients previously stored by function STORE COEFS.

The coefficient filename must have been previously defined - see function COEF FILENAME.

[LOAD COEFS]----------------|

*. This routine - MODUSR2 - was written by Steven J. Clukey of Vigyan Research Associates, Inc. for NASA LaRC TAD/FDB under contract NAS1-17919, Task 36. This work was done beginning in October, 1986 and continues thru April,1988.

Function Name: LOAD LOGFILE
Function Number: 4106
Module: MODUSR2*

This function loads the files used to log hotwire calibration observation data. The function LOG FILENAME declares the file name, and function DISC DEVICE declares the Mass Storage Identifier. See LOG FILENAME for a description of the files loaded into memory.

[LOAD LOGFILE]----------------|

NOTE:

This function <u>must</u> be performed <u>before</u> the function LOG DATA POINT can be invoked.

*. This routine - MODUSR2 - was written by Steven J. Clukey of Vigyan Research Associates, Inc. for NASA LaRC TAD/FDB under contract NAS1-17919, Task 36. This work was done beginning in October, 1986 and continues thru April,1988.

Function Name: LOAD LOGFILE
Function Number: 4106
Module: MODUSR2*

This function loads the files used to log hotwire calibration observation data. The function LOG FILENAME declares the file name, and function DISC DEVICE declares the Mass Storage Identifier. See LOG FILENAME for a description of the files loaded into memory.

```
[ LOAD LOGFILE ]----------------|
```

NOTE:

This function <u>must</u> be performed <u>before</u> the function LOG DATA POINT can be invoked.

*. This routine - MODUSR2 - was written by Steven J. Clukey of Vigyan Research Associates, Inc. for NASA LaRC TAD/FDB under contract NAS1-17919, Task 36. This work was done beginning in October, 1986 and continues thru April,1988.

Function Name: LOG DATA
Function Number: 4109
Module: MODUSR2*

This function serves merely as a label for the path for accessing other related functions via the softkeys.

[LOG DATA]----------------|

*. This routine - MODUSR2 - was written by Steven J. Clukey of Vigyan Research Associates, Inc. for NASA LaRC TAD/FDB under contract NAS1-17919, Task 36. This work was done beginning in October, 1986 and continues thru April,1988.

Function Name: LOG DATA POINT
Function Number: 4108
Module: MODUSR2*

This function logs the hotwire calibration observation data into the logfile(s). The function LOG FILENAME declares the file name, and function DISC DEVICE declares the Mass Storage Identifier. See LOG FILENAME for a description of the files which receive the data.

The hotwire calibration data is received from a static data acquisition system - MODCOMP, other HP, etc - thru a GPIB interface in the form of an ASCII data packet. The static data system is regularly sending about 1 packet per second, and it contains the necessary data in engineering units, including test conditions and test identification.

This function also calculates various data items for inclusion in some of the data files.

This function also generates fluctuating data file names base on the RUN, POINT and channel of the data. These names relate to the data being logged, and allow storing of the fluctuating data in appropriately named files. See functions STORE or TF/STORE ALL.

Example: "RA031701"
R =Realtime digitization
A =probe A
03=RUN
17=POINT
01=data channel 1

[LOG DATA POINT]----------------|

*. This routine - MODUSR2 - was written by Steven J. Clukey of Vigyan Research Associates, Inc. for NASA LaRC TAD/FDB under contract NAS1-17919, Task 36. This work was done beginning in October, 1986 and continues thru April, 1988.

Function Name: LOGFILE TO PC
Function Number: 4127
Module: MODUSR2*

This function transfers logfile data to another computer via a GPIB. The information transmitted is all in ASCII to assure compatability between systems.

For each probe in the range INITIAL PROBE to ENDING PROBE and for observations in the range INITIAL OBS to ENDING OBS, the appropriate observations records are transmitted. Prior to transmitting the set of each probes observations, the names of the variables contained in the observation record are transmitted.

[LOGFILE TO PC]----------------|

*. This routine - MODUSR2 - was written by Steven J. Clukey of Vigyan Research Associates, Inc. for NASA LaRC TAD/FDB under contract NAS1-17919, Task 36. This work was done beginning in October, 1986 and continues thru April,1988.

Function Name: MOVE FILES
Function Number: 4030
Module: MODUSR1*

This function defines the files to be selected for moving, and then copy the selected files **from** a disk device **to** a disk device using the HP COPY command. when the files are selected, a report is generated on the printer which declares the number of files selected, the **from** device, the **to** device, and the files selected for copying. As each file is actually copied, a message is displayed on the CRT, and printed on the printer. Once all selected files have been copied, all successfully copied files are purged from the **from** device, completing the "move". Functions FROM DISK, TO DISK, AND FILE GROUP all effect the results of this function. Selection of files by indicating a group name with this function overrides the group name selection previously made by function FILE GROUP.

```
[ MOVE FILES ]-----------------------------|
                   |                  |
                   |                  |
                   |--[ = ]--[ name ]--|
```

Item	Description	Range
name	string expression	any valid characters that are allowed in a file name see FILE GROUP for details

*. This routine - MODUSR1 - was written by Steven J. Clukey of Vigyan Research Associates, Inc. for NASA LaRC TAD/FDB under contract NAS1-17919, Task 36. This work was done beginning in October, 1986 and continues thru April,1988.

Function Name: PICTURE LOG E
Function Number: 4006
Module: MODUSR1*

This function generates x-y plots on the CRT which represent the Log(RhoU) vs. Log(E) for each hotwire. The axes, titles, etc are internally generated using a plot file called "RhoU" - see function PLOT NAME. (It should be noted that this 'RhoU' picture is actually generated for dynamic channels 8 and 9 - which are assumed to be 1 point long, - and offscale as well.)

This function represents data from all observations in the range INITIAL OBS to ENDING OBS for all probes in the range INITIAL PROBE TO ENDING PROBE.

[PICTURE LOG E]------------|

*. This routine - MODUSR1 - was written by Steven J. Clukey of Vigyan Research Associates, Inc. for NASA LaRC TAD/FDB under contract NAS1-17919, Task 36. This work was done beginning in October, 1986 and continues thru April,1988.

Function Name: PLOT EJECT
Function Number: 4002
Module: MODUSR1*

This function sends a "PG" command to the plot device (if the plot device is not the CRT device.

This function also resets the 'number of observations plotted' pointer, the number of observations printed' pointer, and the 'ending observations' pointer, which affects the operation of functions ENDING OBS, PRNT LOGFILE, PLOT LOG_E and PICTURE LOG_E.

[PLOT EJECT]----------------|

*. This routine - MODUSR1 - was written by Steven J. Clukey of Vigyan Research Associates, Inc. for NASA LaRC TAD/FDB under contract NAS1-17919, Task 36. This work was done beginning in October, 1986 and continues thru April,1988.

Function Name: PLOT LOG E
Function Number: 4005
Module: MODUSR1*

This function generates x-y plots on the Plotter which represent the Log(RhoU) vs. Log(E) for each hotwire. The axes, titles, etc are internally generated using a plot file called "RhoU" - see function PLOT NAME. (It should be noted that this 'RhoU' picture is actually generated for dynamic channels 8 and 9 - which are assumed to be 1 point long, - and offscale as well.)

This function represents data from all observations in the range INITIAL OBS to ENDING OBS for all probes in the range INITIAL PROBE TO ENDING PROBE.

[PLOT LOG E]-------------|

*. This routine - MODUSR1 - was written by Steven J. Clukey of Vigyan Research Associates, Inc. for NASA LaRC TAD/FDB under contract NAS1-17919, Task 36. This work was done beginning in October, 1986 and continues thru April,1988.

Function Name: PLOT UTILITYS
Function Number: 4001
Module: MODUSR1*

This function serves merely as a label for the path for accessing other related functions via the softkeys.

[PLOT UTILITYS]----------------|

*. This routine - MODUSR1 - was written by Steven J. Clukey of Vigyan Research Associates, Inc. for NASA LaRC TAD/FDB under contract NAS1-17919, Task 36. This work was done beginning in October, 1986 and continues thru April,1988.

Function Name: PRNT LOGFILE
Function Number: 4107
Module: MODUSR2*

This function prints the data hotwire calibration observation data currently in the logfile(s). The function LOG FILENAME declares the file name, and function DISC DEVICE declares the Mass Storage Identifier. See LOG FILENAME for a description of the files which contain the data printed. The format of the printout is customized to best demonstrate the hotwire calibration data. This function prints all observations beginning with INITIAL OBS, and ending with ENDING OBS unless previously printed. When the program first starts, and when the PLOT EJECT function is invoked, the number of observations printed is reset to zero, causing all observations already logged to be printed when PRNT LOGFILE is invoked. This technique allows the easy implementation of a sequence program loop including both function LOG DATA POINT and PRNT LOGFILE without having to manipulate observation pointers.

```
[ PRNT LOGFILE ]---------------|
```

*. This routine - MODUSR2 - was written by Steven J. Clukey of Vigyan Research Associates, Inc. for NASA LaRC TAD/FDB under contract NAS1-17919, Task 36. This work was done beginning in October, 1986 and continues thru April,1988.

Function Name: PURGE
Function Number: 4022
Module: MODUSR1*

This function serves merely as a label for the path for accessing other related functions via the softkeys.

[PURGE]----------------|

*. This routine - MODUSR1 - was written by Steven J. Clukey of Vigyan Research Associates, Inc. for NASA LaRC TAD/FDB under contract NAS1-17919, Task 36. This work was done beginning in October, 1986 and continues thru April,1988.

Function Name: PURGE FILE
Function Number: 4024
Module: MODUSR1*

This function defines the file to be selected for purging(deleting) from the disk, and then purges the selected file.

```
[ PURGE FILE ]-----------------------------|
                 |                       |
                 |                       |
                 |--[ = ]--[ name ]--|
```

Item	Description	Range
name	string expression	any valid characters that are allowed in a file name

*. This routine - MODUSR1 - was written by Steven J. Clukey of Vigyan Research Associates, Inc. for NASA LaRC TAD/FDB under contract NAS1-17919, Task 36. This work was done beginning in October, 1986 and continues thru April,1988.

Function Name: PURGE GROUP

Function Number: 4023

Module: MODUSR1*

This function purges all selected files. The selection criteria is defined in detail in the HP BASIC language refernce manual for HP function PURGE (SELECT).

```
[ PURGE GROUP ]----------------------------|
                 |                   |
                 |                   |
                 |--[ = ]--[ name ]--|
```

Item	Description	Range
name	string expression	any valid characters that are allowed in a file name

*. This routine - MODUSR1 - was written by Steven J. Clukey of Vigyan Research Associates, Inc. for NASA LaRC TAD/FDB under contract NAS1-17919, Task 36. This work was done beginning in October, 1986 and continues thru April,1988.

Function Name: Remake Probe
Function Number: 4128
Module: MODUSR2*

This function reproduces the computations normally accomplished during the execution of the function LOG DATA POINT, but without actually acquiring any new data. The purpose is to recompute data should modifications to the computations become necessary.

This function performs these calculations for all probes in the range INITIAL PROBE to ENDING PROBE, and for all observations in the range INITIAL OBS to ENDING OBS.

NOTE:

This function permanently overwrites previous data in the logfiles.

[LOG DATA POINT]----------------|

*. This routine - MODUSR2 - was written by Steven J. Clukey of Vigyan Research Associates, Inc. for NASA LaRC TAD/FDB under contract NAS1-17919, Task 36. This work was done beginning in October, 1986 and continues thru April,1988.

Function Name: SAMPLES TO AVG
Function Number: 4110
Module: MODUSR2*

This function declares the number of data samples to be included in an average of the hotwire calibration data. See LOG DATA POINT for a description of the actual data acquisition process.

[SAMPLES TO AVG]---------------|

*. This routine - MODUSR2 - was written by Steven J. Clukey of Vigyan Research Associates, Inc. for NASA LaRC TAD/FDB under contract NAS1-17919, Task 36. This work was done beginning in October, 1986 and continues thru April,1988.

Function Name: STORE COEFS
Function Number: 4124
Module: MODUSR2

This function stores coefficients previously defined by functions ENTER COEFS, LOAD COEFS, COMPUTE COEFS, etc.

The coefficient filename must have been previously defined - see function COEF FILENAME.

[STORE COEFS]----------------|

This routine - MODUSR2 - was written by Steven J. Clukey of Vigyan Research Associates, Inc. for NASA LaRC TAD/FDB under contract NAS1-17919, Task 36. This work was done beginning in October, 1986 and continues thru April,1988.

Function Name: TAG PICTURE
Function Number: 4004
Module: MODUSR1*

This function "tags" the picture with the first few parameter names and values from the current observation. These few values are intended to identify the environment from which the "picture" was taken. This function is therefore intended to be invoked just after function REDRAW PICTURE or PICTURE LOG E.

[TAG PICTURE]-------------|

*. This routine - MODUSR1 - was written by Steven J. Clukey of Vigyan Research Associates, Inc. for NASA LaRC TAD/FDB under contract NAS1-17919, Task 36. This work was done beginning in October, 1986 and continues thru April,1988.

Function Name: SELECTOR
Function Number: 4129
Module: MODUSR2*

This function serves merely as a label for the path for accessing other related functions via the softkeys.

[SELECTOR]----------------|

*. This routine - MODUSR2 - was written by Steven J. Clukey of Vigyan Research Associates, Inc. for NASA LaRC TAD/FDB under contract NAS1-17919, Task 36. This work was done beginning in October, 1986 and continues thru April,1988.

Function Name: TAG PLOT
Function Number: 4003
Module: MODUSR1*

This function "tags" the plot with the first few parameter names and values from the current observation. These few values are intended to identify the environment from which the plot was taken. This function is therefore intended to be invoked just after function REDRAW PLOT or PLOT LOG E.

```
[ TAG PLOT ]------------|
```

*. This routine - MODUSR1 - was written by Steven J. Clukey of Vigyan Research Associates, Inc. for NASA LaRC TAD/FDB under contract NAS1-17919, Task 36. This work was done beginning in October, 1986 and continues thru April,1988.

Function Name: TO DISK
Function Number: 4027
Module: MODUSR1*

This function defines the mass storage device to which the files will be copied - function COPY FILES - or moved - function MOVE FILES.

```
[ TO DISK    ]-------------------------------|
                 |                          |
                 |                          |
                 |--[ = ]--[ MSI ]--|
```

Item	Description	Range
value	MSI | any valid HP	storage device

*. This routine - MODUSR1 - was written by Steven J. Clukey of Vigyan Research Associates, Inc. for NASA LaRC TAD/FDB under contract NAS1-17919, Task 36. This work was done beginning in October, 1986 and continues thru April,1988.

Function Name: UTILITY
Function Number: 4000
Module: MODUSR1*

This function serves merely as a label for the path for accessing other related functions via the softkeys.

```
[ UTILITY ]----------------|
```

*. This routine - MODUSR1 - was written by Steven J. Clukey of Vigyan Research Associates, Inc. for NASA LaRC TAD/FDB under contract NAS1-17919, Task 36. This work was done beginning in October, 1986 and continues thru April,1988.

APPENDIX C. Sequence Programs

```
5    SYS VARS NAME=AUTOVARS
6    LOAD SYS VARS
7    LOG FILENAME=934
8    LOAD LOGFILE
10   PLOT EJECT
15   GRAPH OFF
20   TRACE ACTIVE=1,2,3,4,5,6,7
30   C=ALL
40   C TO WHOLE
50   X CALIB TYPE=TIME
60   Y CALIB TYPE=VOLTS
65   BIN SW NAME=8FTSW
66   LOAD BIN SW
70   SEQ PROG NAME = 8FTRSEQ
71   DATA DISC DEV = :,1400
72   FILENAME(1)=T934__0100
73   FILENAME(2)=T934__0200
74   FILENAME(3)=T934__0300
75   FILENAME(4)=T934__0400
76   FILENAME(5)=T934__0500
77   FILENAME(6)=T934__0600
78   FILENAME(7)=T934__0700
90   DATA FILE MAP(1)=YES
91   DATA FILE MAP(2)=YES
92   DATA FILE MAP(3)=YES
93   DATA FILE MAP(4)=YES
94   DATA FILE MAP(5)=YES
95   DATA FILE MAP(6)=YES
96   DATA FILE MAP(7)=YES
97   DATA FILE MAP(8)=NO
98   DATA FILE MAP(9)=NO
99   DATA FILE MAP(10)=NO
100  DATA FILE MAP(11)= NO
101  DATA FILE MAP(12)=NO
102  DATA FILE MAP(13)=NO
103  DATA FILE MAP(14)=NO
110  MEM LENGTH=64K
111  MEM LENGTH(1)=64K
112  MEM START(1)=0
119  MEM START(8)=1791K
121  TF MAP CHAN(1)=1
122  TF MAP CHAN(2)=2
123  TF MAP CHAN(3)=3
124  TF MAP CHAN(4)=4
125  TF MAP CHAN(5)=5
126  TF MAP CHAN(6)=6
127  TF MAP CHAN(7)=7
128  TF MAP CHAN(8)=0
129  TF MAP CHAN(9)=0
130  TF MAP CHAN(10)=0
131  TF MAP CHAN(11)=0
132  TF MAP CHAN(12)=0
133  TF MAP CHAN(13)=0
134  TF MAP CHAN(14)=0
200  LOAD SEQ PROG
210  RUN SEQ PROG
```

APPENDIX C.1. Initialization Sequence Program - "8FTINI"

```
3   PLOT EJECT
4   INITIAL OBS=49
5   PLOT LOG_E
6   PRNT LOGFILE
7   SEQUENCE MENU
8   LET I=0
10  PRINT "PRESS ENTER WHEN READY TO LOG DATA"
20  WAIT ?
70  ARM
80  LOG DATA POINT
85  PRNT LOGFILE
90  PRINT "POINT IN PROGRESS"
95  PLOT LOG_E
96  IF I=0 THEN TAG PLOT
97  I=1
100 WAIT *
101 PRINT "TRANSFERRING DATA TO MEMORY"
105 TF FROM REC
106 PRINT "RECORDING DATA"
107 STORE DATA
110 PRINT "POINT COMPLETE   "
800 GOTO 10
```

APPENDIX C.2. Run Sequence Program - "8FTRSEQ"

APPENDIX D. PC BASIC Program - "XFR.HP"

```
LIST
1 LIST
10    '   PROGRAM  XFR.HP     S. J. Clukey, Vigyan Research Associates
20    '
30    '
40    'Initialization from "EXAMPLE.BAS" of the HP-IB Command Library
50    'Copyright Hewlett-Packard 1984, 1985
60   '
70   '  Set up program for MS-DOS HP-IB I/O Library
80   '  For use independent of the PC instrument bus system
90 CLS
100   '
110   DEF SEG
120   CLEAR  ,&HFE00
130   I=&HFE00
140   '
150   '  PCIB.DIR$ represents the directory where the library files
160   '     are located
170   '  PCIB is an environment variable which should be set from MS-DOS
180   '     i.e.  A:> SET PCIB=A:\LIB
190  '
200  '     If there is insufficient environment space a direct assignment
210  '     can be made here, i.e
220  '         PCIB.DIR$ = "A:\LIB"
230  '     Using the environment variable is the preferred method
240  '
250  PCIB.DIR$ = ENVIRON$("PCIB")
260  I$ = PCIB.DIR$ + "\PCIBILC.BLD"
270  BLOAD I$,&HFE00
280  CALL I(PCIB.DIR$, I%, J%)
290  PCIB.SEG = I%
300  IF J%=0 THEN GOTO 370
310  PRINT "Unable to load.";
320  PRINT "   (Error #";J%;")"
330  STOP
340  '
350  '  Define entry points for setup routines
360  '
370  DEF SEG = PCIB.SEG
380  O.S      = 5
390  C.S      = 10
400  I.V      = 15
410  I.C      = 20
420  L.P      = 25
430  LD.FILE = 30
440  GET.MEM = 35
450  L.S      = 40
460  PANELS  = 45
470  '
480  '  Establish error variables and ON ERROR branching
490  '
500  DEF.ERR = 50
510  PCIB.ERR$ = STRING$(64,32)
520  PCIB.NAME$ = STRING$(16,32)
530  CALL DEF.ERR(PCIB.ERR,PCIB.ERR$,PCIB.NAME$,PCIB.GLBERR)
540  PCIB.BASERR = 255
550  ON ERROR GOTO 870
560  '
570  J=-1
580  I$=PCIB.DIR$+"\HPIB.SYN"
```

```
590  CALL O.S(I$)
600  IF PCIB.ERR<>0 THEN ERROR PCIB.BASERR
610  '
620  '  Determine entry points for HP-IB Library routines
630  '
640  I=0
650  CALL I.V(I,IOABORT,IOCLEAR,IOCONTROL,IOENTER)
660  IF PCIB.ERR<>0 THEN ERROR PCIB.BASERR
670  CALL I.V(I,IOENTERA,IOENTERS,IOEOI,IOEOL)
680  IF PCIB.ERR<>0 THEN ERROR PCIB.BASERR
690  CALL I.V(I,IOGETTERM,IOLLOCKOUT,IOLOCAL,IOMATCH)
700  IF PCIB.ERR<>0 THEN ERROR PCIB.BASERR
710  CALL I.V(I,IOOUTPUT,IOOUTPUTA,IOOUTPUTS,IOPPOLL)
720  IF PCIB.ERR<>0 THEN ERROR PCIB.BASERR
730  CALL I.V(I,IOPPOLLC,IOPPOLLU,IOREMOTE,IORESET)
740  IF PCIB.ERR<>0 THEN ERROR PCIB.BASERR
750  CALL I.V(I,IOSEND,IOSPOLL,IOSTATUS,IOTIMEOUT)
760  IF PCIB.ERR<>0 THEN ERROR PCIB.BASERR
770  CALL I.V(I,IOTRIGGER,J,J,J)
780  IF PCIB.ERR<>0 THEN ERROR PCIB.BASERR
790  CALL C.S
800  I$=PCIB.DIR$+"\HPIB.PLD"
810  CALL L.P(I$)
820  IF PCIB.ERR<>0 THEN ERROR PCIB.BASERR
830  GOTO 1000
840  '
850  '  Error handling routine
860  '
870  IF ERR=PCIB.BASERR THEN GOTO 900
880  PRINT "BASIC error #";ERR;" occurred in line ";ERL
890  STOP
900  TMPERR = PCIB.ERR
910  IF TMPERR = 0 THEN TMPERR = PCIB.GLBERR
920  PRINT "PC Instrument error #";TMPERR;" detected at line ";ERL
930  PRINT "Error: ";PCIB.ERR$
940  STOP
950  '
960  '  COMMON declarations are needed if your program is going to chain
970  '     to other programs.  When chaining, be sure to call DEF.ERR as
980  '     well upon entering the chained-to program
990  '
1000  COMMON PCIB.DIR$,PCIB.SEG
1010  COMMON LD.FILE,GET.MEM,PANELS,DEF.ERR
1020  COMMON PCIB.BASERR,PCIB.ERR,PCIB.ERR$,PCIB.NAME$,PCIB.GLBERR
1030  COMMON IOABORT,IOCLEAR,IOCONTROL,IOENTER,IOENTERA,IOENTERS,IOEOI,IOEOL,IOG
ETTERM,IOLLOCKOUT,IOLOCAL,IOMATCH,IOOUTPUT,IOOUTPUTA,IOOUTPUTS,IOPPOLL,IOPPOLLC,
IOPPOL
1040  '
1050  FALSE    = 0
1060  TRUE     = NOT FALSE
1070  NOERR    = 0
1080  EUNKNOWN = 100001!
1090  ESEL     = 100002!
1100  ERANGE   = 100003!
1110  ETIME    = 100004!
1120  ECTRL    = 100005!
1130  EPASS    = 100006!
1140  ENUM     = 100007!
1150  EADDR    = 100008!
1160  COMMON FALSE, TRUE, NOERR, EUNKNOWN, ESEL, ERANGE, ETIME, ECTRL, EPASS, EN
```

```
M, EADDR
1170  '
1180  ' End Program Set-up
1190  ' User program can begin anywhere past this point
1200  ' Program for a system to receive data from the Dynamic Data
1210  ' Acquisition System.
1220  '
1230  '
1240  OPTION BASE 1
1250  MAX.VARIABLES= 50
1260  DIM NAMES$ (MAX.VARIABLES)
1270 DIM X(3)
1280  ACT.VARIABLES= 0
1290  NAMES$ = SPACE$(50)
1300  '
1310  ' Set up HP-IB addressing and initialize system
1320  '
1330  ISC=7
1340  DEV=1
1350  DEV = ISC * 100 + DEV
1360  CALL IORESET (ISC)
1370  IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR
1380  TIMEOUT = 5
1390  CALL IOTIMEOUT (ISC, TIMEOUT)
1400  IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR
1410  CALL IOCLEAR (ISC)
1420  IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR
1430  '
1440  '
1450  '
1460  CALL IOEOI (ISC, FALSE)
1470  IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR
1480 '
1490 '
1491 PRINT "DO YOU WISH TO RECEIVE A FILE FROM THE HP COMPUTER? (Y or N)"
1492 INPUT ANS$
1493 IF ANS$="N" THEN GOTO 1730
1494 IF ANS$<>"Y" THEN GOTO 1491
1500 PRINT "ENTER THE NAME OF THE FILE TO RECEIVE TRANSFERED DATA:  "
1510 INPUT FIL$
1520 OPEN FIL$ FOR OUTPUT AS #1
1530 FOR J=1 TO 2
1540   MM=6 : AA=0
1550   T$(J)=SPACE$(MM)
1560   CALL IOENTERS (DEV,T$(J),MM,AA)
1570   IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR
1580   N(J)=VAL(LEFT$(T$(J),AA-2))
1590   PRINT N(J)
1600 NEXT J
1610 NOBS=N(1)+1 : NVARS=N(2)
1620 FOR J=1 TO NOBS
1630   FOR I=1 TO NVARS
1640     M=20 : A=0
1650     TEMP$=SPACE$(M)
1660     CALL IOENTERS (DEV,TEMP$,M,A)
1670     IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR
1680     NAMES$(I)=LEFT$(TEMP$,A-2)
1690 '   PRINT "Record ",J,", item ",(I),"= " NAMES$(I)
1701     IF J=1 THEN PRINT #1,NAMES$(I) ELSE PRINT #1,VAL(NAMES$(I))
1710   NEXT I
```

```
1720 NEXT J
1721 CLOSE #1
1722 GOTO 1490
1730 END
Ok
```

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA CR-181758		

4. Title and Subtitle	5. Report Date
A High Speed Data Acquisition and Analysis System for Transonic Velocity, Density, and Total Temperature Fluctuations	December 1988
	6. Performing Organization Code
7. Author(s)	8. Performing Organization Report No.
Steven J. Clukey	
	10. Work Unit No. 505-60-21-06
9. Performing Organization Name and Address	11. Contract or Grant No.
Vigyan Research Associates, Inc. 30 Research Drive Hampton, VA 23666-1325	NAS1-17919
	13. Type of Report and Period Covered
12. Sponsoring Agency Name and Address	Contractor Report
National Aeronautics and Space Administration Langley Research Center Hampton, VA 23665-5225	14. Sponsoring Agency Code

15. Supplementary Notes

Langley Technical Monitor: Gregory S. Jones

16. Abstract

This report describes the high speed Dynamic Data Acquisition System (DDAS) which provides the capability for the simultaneous measurement of velocity, density, and total temperature fluctuations. The system of hardware and software is described in context of the wind tunnel environment.

The DDAS replaces both a recording mechanism and a separate data processing system. The data acquisition and data reduction process has been combined within DDAS. DDAS receives input from hot wires and anemometers, amplifies and filters the signals with computer controlled modules, and converts the analog signals to digital with real-time simultanteous digitization followed by digital recording on disk or tape. Automatic acquisition (either from a computer link to an existing wind tunnel acquisition system, or from data acquisition facilities within DDAS) collects necessary calibration and environment data. The generation of hot wire sensitivities is done in DDAS, as is the application of sensitivities to the hot wire data to generate turbulence quantities. The presentation of the raw and processed data, in terms of root mean square values of velocity, density and temperature, and the processing of the spectral data is accomplished on demand in near-real-time with DDAS.

A comprehensive description of the interface to the DDAS and of the internal mechanisms will be presented. A summary of operations relevant to the use of the DDAS will be provided.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Hot Wire, Anemometry, Fluctuating, Transonic, Data Acquisition, Three-Wire	Unclassified - Unlimited Subject Category 35

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price
Unclassified	Unclassified	142	A07

NASA FORM 1626 OCT 86