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

Unclas

CSCL 14B G3/35 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 Digitizer 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 ......... 14
    *2.2.6.2.1.3. Fluctuating data disk file naming** ..... 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
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 \( \frac{\partial \log e}{\partial \log u} \rho, T_0, T_w \)

\( S_\rho \)  
density sensitivity \( \frac{\partial \log e}{\partial \log \rho} u, T_0, T_w \)

\( S_{T_0} \)  
temperature sensitivity \( \frac{\partial \log e}{\partial \log T_0} u, \rho, T_w \)

\( T_0 \)  
mean total temperature

\( T_w \)  
mean temperature of heated wire

\( u \)  
mean velocity

\( \rho \)  
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. 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 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.).

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 Digitizer 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 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 not 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. Computer Peripherals

2.1.5.1. Disk storage

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. 55Mb hard disk

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. **20Mb hard disk**

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

2.1.5.2. **Tape storage**

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

2.1.5.3. **Display**

A color CRT is the system console and data display.

2.1.5.4. **Plotter**

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. **Printer**

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. **SOFTWARE**

2.2.1. **Software environment**
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 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. 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.

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 shown¹:

\[
\log E = A_1 + A_2 \log u + A_3 \log \rho + A_4 \log T_0 \\
+ A_5 \log u \log \rho + A_6 \log u \log T_0 \\
+ A_7 \log \rho \log T_0 \\
+ A_8 \log u \log \rho \log T_0
\]  

(1)

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.

\[
S_u = A_2 + A_5 \log \rho + A_6 \log T_0 \\
+ A_8 \log \rho \log T_0 \\
\]

\[
S_\rho = A_3 + A_5 \log u + A_7 \log T_0 \\
+ A_8 \log u \log T_0 \\
\]

\[
S_{T_0} = A_4 + A_6 \log u + A_7 \log \rho \\
+ A_8 \log u \log \rho \\
\]

(2a) \hspace{1cm} (2b) \hspace{1cm} (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 sensitivities 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:

\[
\begin{align*}
\begin{bmatrix}
\varepsilon' \\
\varepsilon' \\
\varepsilon'
\end{bmatrix}
& =
\begin{bmatrix}
S_{u_1} \frac{u'}{U} + S_{\rho_1} \frac{\rho'}{\rho} + S_{T_0} \frac{T_0'}{T_0} \\
S_{u_2} \frac{u'}{U} + S_{\rho_2} \frac{\rho'}{\rho} + S_{T_0} \frac{T_0'}{T_0} \\
S_{u_3} \frac{u'}{U} + S_{\rho_3} \frac{\rho'}{\rho} + S_{T_0} \frac{T_0'}{T_0}
\end{bmatrix}
\end{align*}
\]

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

\[
\begin{bmatrix}
\varepsilon' \\
\varepsilon' \\
\varepsilon'
\end{bmatrix}
= 
\begin{bmatrix}
S_{u_1} & S_{\rho_1} & S_{T_0} \\
S_{u_2} & S_{\rho_2} & S_{T_0} \\
S_{u_3} & S_{\rho_3} & S_{T_0}
\end{bmatrix}
\begin{bmatrix}
\frac{u'}{U} \\
\frac{\rho'}{\rho} \\
\frac{T_0'}{T_0}
\end{bmatrix}
\]
By rearranging again, which involves inverting the sensitivity matrix, solve for the three unknowns:

\[
\begin{bmatrix}
\frac{u'}{U} \\
\frac{\rho'}{\rho} \\
\frac{T_0'}{T_0}
\end{bmatrix}
= \begin{bmatrix}
S_{u_1} & S_{\rho_1} & S_{T_0_1} \\
S_{u_2} & S_{\rho_2} & S_{T_0_2} \\
S_{u_3} & S_{\rho_3} & S_{T_0_3}
\end{bmatrix}^{-1}
\begin{bmatrix}
\frac{e'}{E} \cdot \frac{1}{G_w}
\end{bmatrix}
\]

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, \rho, 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:

\[
\frac{e'}{E} \cdot \frac{1}{G_w}
\]

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

compute the RMS values of \( \frac{u'}{U}, \frac{\rho'}{\rho}, \frac{T_0'}{T_0} \).

store the RMS values of velocity, density, and temperature:

\[
\begin{bmatrix}
\frac{u'}{U} & \frac{\rho'}{\rho} & \frac{T_0'}{T_0}
\end{bmatrix}
\]

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
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. ACQUIRE installation

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. System variables
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, \( \rho u \), (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. Operating system

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. Programming language

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:


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


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

Figure 1. DDAS System Block Diagram
Figure 2. Plot: hotwire voltage vs. mass flow
### Figure 3. Observation Report

#### Tunnel Conditions

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mach</td>
<td>0.8001725</td>
</tr>
<tr>
<td>Reynolds No.</td>
<td>9.7948923333</td>
</tr>
<tr>
<td>Pt</td>
<td>709.55</td>
</tr>
<tr>
<td>Ps</td>
<td>465.4</td>
</tr>
<tr>
<td>Tt</td>
<td>80.3028133333</td>
</tr>
<tr>
<td>Velocity</td>
<td>850.406598442</td>
</tr>
<tr>
<td>Density</td>
<td>0.018230240804</td>
</tr>
<tr>
<td>LOG(Rhoe)</td>
<td>1.18876833942</td>
</tr>
</tbody>
</table>

Mean (HW1)

<table>
<thead>
<tr>
<th>Mean</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEAN(HW1)</td>
<td>4.82403933333</td>
</tr>
<tr>
<td>MEAN(HW2)</td>
<td>4.67869933333</td>
</tr>
<tr>
<td>MEAN(HW3)</td>
<td>5.40758666667</td>
</tr>
</tbody>
</table>

Mean (HW1)

<table>
<thead>
<tr>
<th>Mean (HW1)</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>S(U)</td>
<td>0.0801895340403</td>
</tr>
<tr>
<td>S(Rhoe)</td>
<td>2.46631039249</td>
</tr>
<tr>
<td>S(Tt)</td>
<td>-0.72168324665</td>
</tr>
<tr>
<td>S(U)</td>
<td>0.0815796208448</td>
</tr>
<tr>
<td>S(Rhoe)</td>
<td>2.2379374977</td>
</tr>
<tr>
<td>S(Tt)</td>
<td>-0.79872035258</td>
</tr>
<tr>
<td>S(U)</td>
<td>0.0757205995812</td>
</tr>
<tr>
<td>S(Rhoe)</td>
<td>2.31703194152</td>
</tr>
<tr>
<td>S(Tt)</td>
<td>-0.52354462725</td>
</tr>
</tbody>
</table>

u'/U (rms)

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0132928317592</td>
</tr>
</tbody>
</table>

p'/P (rms)

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00400499662361</td>
</tr>
</tbody>
</table>

t'/T (rms)

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000224062285135</td>
</tr>
</tbody>
</table>

R(Rhoe)

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.99747275712</td>
</tr>
</tbody>
</table>

R(U0)

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.872662983427</td>
</tr>
</tbody>
</table>

R(RhoCT0)

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.875779642389</td>
</tr>
</tbody>
</table>

M'/M

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00930062427607</td>
</tr>
</tbody>
</table>

P'/P

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00729524085695</td>
</tr>
</tbody>
</table>

TOTAL:

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.75614</td>
</tr>
</tbody>
</table>

**Figure 3. Observation Report**
Figure 4. Plot: Waveforms - Floor strut
Figure 5. Plot: Waveforms - Wall strut
Ps
tunnel static pressure (psf)

Pt
tunnel total pressure (psf)

Tt
tunnel total temperature (deg F)

Mach number

Reynolds number (per chord foot)

PtS1
Strut 1 (Wall) total pressure (psf)

PsS1
static pressure (psf)

TtS1
total temperature (deg F)

PtS2
Strut 2 (Floor) total pressure (psf)

PsS2
static pressure (psf)

TtS2
total temperature (deg F)

PtS3
Strut 3 (Unused)

PsS3

TtS3

P1HW1
Strut 1 Hot wire 1 mean voltage

2 mean voltage

3 mean voltage

4 mean voltage

P2HW1
Strut 2 Hot wire 1 mean voltage

2 mean voltage

3 mean voltage

P3HW1
Strut 3 Hot wire 1 mean voltage

P4HW1
Strut 4 Hot wire 1 mean voltage

P5HW1
Strut 5 Hot wire 1 mean voltage

Kulitel
Microphone RMS voltage

Gain code representing instrument gain

Wires 5, 6, and 7 are on probe 2.

Gain code representing instrument gain

<ETX>  (ignored)
Fluctuating Data File Name Format

<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>Real time digitized data</td>
</tr>
<tr>
<td>P</td>
<td>Playback (FM tape) digitized data</td>
</tr>
<tr>
<td>V</td>
<td>Velocity ratio - computed data</td>
</tr>
<tr>
<td>D</td>
<td>Density ratio - computed data</td>
</tr>
<tr>
<td>T</td>
<td>Temperature ratio - computed data</td>
</tr>
</tbody>
</table>

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)

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 not 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.
This program becomes a part of "ACQUIRE", and provides additional utility functions. As the need for additional functions increases, so will the functions implemented in this program.

26055 OPTION BASE 1
26056 COM /Arr/ INTEGER Arrowar(*)
26057 COM /Cross/ Crval(*),INTEGER Crvar(*),Crtab(*),Crmprtr
26058 COM /Curs/ Caddrx(*),INTEGER Cincr(*),Cpos(*),Scursor(*),Cactive,Cpixmin (*),Cpixmax(*),Cflag
26059 COM /Error/ INTEGER Errf,Errtype
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$(*)*,INTEGER X(*),Isw(*),Memlenb(*),Memlenu(*)
26065 COM /Menu/ Menulab$*,Mvar$(*)*,Mlit$(*)*,INTEGER Mivar(*),Menutab(*),
26066 COM /Param/ Mval$,Mval(*),INTEGER Hstack(*),Hvallist(*),Mstackptr,Mvalptr,
26067 COM /Plot1/ INTEGER Titlexcoor,Titleycoor,Titlesize,Ntrace,T_chan(*),Lty pe(*),Secondc(*),Titlep,Plotbh,Plname$,Plotstring$
26068 COM /Plot2/ REAL Xstart(*),Xend(*),Ystart(*),Yend(*),Xorigin(*),Yorigin(*)
26069 COM /Plot3/ INTEGER Xlabelp(*),Xlabelpc(*),Ylabelp(*),Ylabelpc(*),Commentp(16,8),Commentpc(16,8),Commentsize(*),Npoint(*),Labelsize(*)
26070 COM /Plot4/ INTEGER Comcount(*),Tabcount(*),Eventbit(*),Tabval(*),Tabvalpc(*)
26071 COM /Plot5/ INTEGER Created,Noofpen,Plotdev,Plgrid(*),Tlabp(*),Tablpc(*)
26072 Usercom:
m_obs_plotted
26088 COM /Usr1/ From_disk$(10),To_disk$(10),File_grp$(10),File_name$(10)
26089 COM /Usr2/ Sfn$(1),REAL Hwsens$(1),Sensinv$(1),Mean$(1),Mean_param$(1),Norm$(1),Stddev_param$(1),Max_param$(1),Min_param$(1),Vo_param$(1),Vs_param$(1)
26090 COM /Usr2/ Gain_code$(14),Gain_code$(57)
26091 COM /Usr2/ C_names$(1),P_c$(1)
26092 COM /Usr2/ Log_fn$,Data_set_title$,Logged_var_name$(1),Obs_rec$(1),Pkt_sc,Pkt$(1),Pkt_avg$(1),Num_avg$(1),Max_vars$(1),Rcvd_vars$(1)
26093 COM /Usr2/ Subfile_names$(1),Subfile_chartst$(1),Initial_obs,Ending_obs,Num_obs_rec$(1),Num_obs_printed,Max_obs_rec$(1),Tag_pkt$(1)
26094 COM /Usr2/ Initial_probe,Ending_probe
26095 COM /Usr2/ A_fn$,A_set_title$,A_var_name$(1),A_rec$(1)
26096 COM /Usr2/ A_subfile_names$(1),A_sub_chartst$(1)
26097 COM /Usr2/ B_fn$,B_set_title$,B_var_name$(1),B_rec$(1)
26098 COM /Usr2/ B_subfile_names$(1),B_sub_chartst$(1)
26099 COM /Usr2/ C_fn$,C_set_title$,C_var_name$(1),C_rec$(1)
26100 COM /Usr2/ C_subfile_names$(1),C_sub_chartst$(1)
26101 COM /Usr2/ Hw_rms$(1)
26102 INTEGER V(3)
26103 INTEGER I
26104 DIM VS$(30)
26105 SELECT Routine
26106 CASE 1 ! init pass!
26107 RESTORE Menulist
26108 LOOP
26109 READ VS$(1)
26110 EXIT IF VS$="***"
26111 READ V$(1)
26112 CALL Chkmitem(V$(1),VS$)
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(2),V(2))
26123 END LOOP
26124 !
26125 Menulist: !
26126 ! "LABEL,Function,Flag1,Flag2"
26127 DATA "UTILITY",4000,0,6
26128 DATA "PLOT UTILITIES",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 UTILITIES",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

ORIGINAL PAGE IS OF POOR QUALITY
DATA "MOVE FILES",4030,8704,394
DATA "***"
!
Keylist: !
DATA 0,4000
DATA 100,4001
DATA 110,4002
DATA 120,4003
DATA 130,4004
DATA 140,4005
DATA 150,4006
DATA 200,4020
DATA 210,4021
DATA 220,4022
DATA 221,4023
DATA 222,4024
DATA 240,4025
DATA 241,4026
DATA 242,4027
DATA 243,4028
DATA 245,4030
DATA 247,4029
DATA -1,-1
!
CASE 3 !RUN TIME INITIALIZATION
Fil_nam$=""
Fil_grp$=""
RESTORE Symbols
READ Sym_cbl$(*)
Symbols: !
DATA "+
DATA "X"
DATA "0"
DATA "*
CASE 4 !Power on initializations:
Num_obs_plotte=0
!
CASE ELSE
Usercode:!
SELECT Code
CASE 4002 !PLOT EJECT
SELECT Routine
CASE 31
OUTPUT Plotdev;"PG"
Num_obs_plotte=0 !RESET OBSERVATIONS POINTER for Function 4005
Num_obs_printed=0 !RESET OBSERVATIONS PRINTED for Function 4107
END SELECT
CASE 4003,4004 !PLOT TAG  PICTURE TAG
SELECT Routine
CASE 31
Errf=0
IF Errf=0 AND Code=4003 THEN PLOTTER IS Plotdev,"HPGL"
IF Errf=0 THEN
OFF TIMEOUT 7
Yminoff=0
Yoff=1
X_gdu_min=0
X_gdu_max=100*RATIO
Y_gdu_min=0
Y_gdu_max=100

DEG

PEN 1

IF Titlesize<-.8 THEN Titlesize=4

CSIZE .65*Titlesize*Ymoff,.65*RATIO*.5

LORG 3

LDIR 0

VIEWPORT X_gdu_min,X_gdu_max,Y_gdu_min*Ymoff+Yminoff,Y_gdu_max*Ymoff+Yminoff

WINDOW X_gdu_min,X_gdu_max,Y_gdu_min,Y_gdu_max

MOVE X_gdu_max-(.20*X_gdu_max),Y_gdu_max-10

FOR I=1 TO 10

SELECT I

CASE 1,2,5!TEST, RUN, DATE only

LABEL Logged_var_name$(I) & " & Tag_pkt$(I)

END SELECT

NEXT I

FOR Probe=Initial_probe TO Ending_probe

SELECT Probe

CASE 1

PEN 1

LABEL "WALL PROBE"

PEN 2

LABEL "FLOOR PROBE"

CASE 2

END SELECT

NEXT Probe

END IF

PENUP

VIEWPORT X_gdu_min,X_gdu_max,Y_gdu_min,Y_gdu_max

WINDOW X_gdu_min,X_gdu_max,Y_gdu_min,Y_gdu_max

MOVE X_gdu_max,Y_gdu_max

PLOTTER IS CRT,"INTERNAL"

END SELECT

PLOTTER IS CRT, "INTERNAL"

CASE 4005,4006 !PLOT LOG(E) vs LOG(Rho*U)

SELECT Routine

CASE 31

IF Num_obs_plotted=0 THEN !Get the axis plotted

IF Code=4005 THEN OUTPUT Plotdev; "PG"!Eject old plot

Mvar$(3)="RhoU"

CALL Routine(22,618) !PLOT NAME=RhoU

CALL Routine(31,616) !LOAD PLOT

IF Code=4005 THEN

CALL Routine(31,615)!PLOT PICTURE

ELSE

CALL Routine(31,612)!REDRAW PICTURE

END IF

END IF

IF Code=4005 THEN

PLOTTER IS Plotdev,"HPGL"

OFF TIMEOUT 7

ELSE

PLOTTER IS CRT,"INTERNAL"

END IF

Yminoff=0

Ymoff=1

IF Titlesize<-.8 THEN Titlesize=4
FOR Probe=Initial_probe TO Ending_probe
  PEN Probe
  FOR This_obs=Initial_obs TO Ending_obs
    SELECT Probe
    CASE 1
      Num_wires=4
    CASE 2
      Num_wires=3
    CASE ELSE
      Num_wires=1
    END SELECT
    VIEWPORT Xstart(1),Xend(1),Ystart(1)*Ymoff+Yminoff,Yend(1)*Ymoff+Yminoff
    WINDOW Xmin(1),Xmax(1),Ymin(1),Ymax(1)
    !######_ FOR Wire=1 TO Num_wires
    FOR Wire=1 TO 1
      SELECT Probe
      CASE 1
        Pressure=A_rec(4,This_obs)
        X_val=MAX(-E6,A_rec(13,This_obs))
        Y_val=MAX(-E6,A_rec(21+Wire,This_obs))
      CASE 2
        Pressure=B_rec(4,This_obs)
        X_val=MAX(-E6,B_rec(13,This_obs))
        Y_val=MAX(-E6,B_rec(21+Wire,This_obs))
      END SELECT
      MOVE X_val,Y_val
      SELECT Pressure
      CASE 700. TO 720.
        LABEL Sym_tbl$(1)
      CASE 850. TO 880.
        LABEL Sym_tbl$(2)
      CASE 1400. TO 1500.
        LABEL Sym_tbl$(3)
      CASE 1700. TO 1800.
        LABEL Sym_tbl$(4)
      END SELECT
      NEXT Wire
    NEXT This_obs
    IF Probe=1 THEN
      MOVE 1.,6639
      DRAW 1.54,.77832
      LINE TYPE 4
      PEN Probe+3
      MOVE 1.,67889
      DRAW 1.58,.72076
      LINE TYPE 1
    END IF
    NEXT Probe
  END SELECT
  PLOTTER IS CRT,"INTERNAL"
END SELECT
CASE 4021 !CAT A SELECT GROUP OF FILES
SELECT Routine
CASE 21
  Mvar$(3)=FIL_grp$
CASE 22
  FIL_grp$=Mvar$(3)
CASE 31
   CAT; SELECT Fil_grp$
CASE 21
   Mvar$(3)=Fil_grp$
CASE 22
   Fil_grp$=Mvar$(3)
CASE 31
   CAT TO Cat_array$(*) ; SELECT Fil_grp$, NO HEADER, COUNT Num_in_grp
   Fil_grp$=""
   FOR I=1 TO Num_in_grp
   Fil_nam$=Cat_array$(I)[1;10]
   PURGE Fil_nam$
   NEXT I
CASE 4023
   !PURGE A SELECT GROUP OF FILES
   SELECT Routine
   CASE 21
   Mvar$(3)=Fil_grp$
   CASE 22
   Fil_grp$=Mvar$(3)
   CASE 31
   CAT TO Cat_array$(*) ; SELECT Fil_grp$, NO HEADER, COUNT Num_in_grp
   Fil_grp$=""
   FOR I=1 TO Num_in_grp
   Fil_nam$=Cat_array$(I)[1;10]
   PURGE Fil_nam$
   NEXT I
CASE 4024
   !PURGE A FILE
   SELECT Routine
   CASE 21
   Mvar$(3)=Fil_nam$
   CASE 22
   Fil_nam$=Mvar$(3)
   CASE 31
   IF LEN(Fil_nam$)>0 THEN PURGE Fil_nam$
   CALL Pline(0,"FILE ":Fil_nam$" HAS BEEN PURGED")
CASE 4026
   !FROM DISK
   SELECT Routine
   CASE 21
   Mvar$(3)=From_disk$
   CASE 22
   From_disk$=Mvar$(3)
   END SELECT
CASE 4027
   !TO DISK
   SELECT Routine
   CASE 21
   Mvar$(3)=To_disk$
   CASE 22
   To_disk$=Mvar$(3)
   END SELECT
CASE 4028
   !FILE GROUP
   SELECT Routine
   CASE 21
   Mvar$(3)=File_grp$
   CASE 22
   File_grp$=Mvar$(3)
   END SELECT
CASE 4029,4030
   !COPY OR MOVE A SELECT GROUP OF FILES
   !( Copy leaves the files on both from and to disks.)
   !( Move leaves the files only on the to disk, by purging
   ! the files successfully copied from the 'from' disk.)
   SELECT Routine
   CASE 21
   Mvar$(3)=File_grp$
   CASE 22
   File_grp$=Mvar$(3)
   CASE 31
   CALL Pline(0,"Selecting files. Please wait....")
   CAT From_disk$ TO Cat_array$(*) ; SELECT File_grp$, NO HEADER, COUNT
T Num_in_grp
26382                        PRINTER IS PRT;WIDTH 108
26383                        ! Form Feed
26384                        PRINT CHR$(12); " files have been selected for copying."
26385                        PRINT
26386                        PRINT FROM;TAB(30);"TO"
26387                        PRINT From_disk$;TAB(30);To_disk$
26388                        PRINT
26389                        FOR I=1 TO Num_in_grp
26390                        PRINT TAB(20);Cat_array$(I)[1,10]
26391                        NEXT I
26392                        PRINT
26393                        PRINT
26394                        PRINT
26395                        PRINT
26396                        PRINTER IS CRT
26397                        !
26398                        Number_copied=0
26399                        PRINTER IS PRT
26400                        PRINT "The following files"&CHR$(27)&"&D"&" HAVE BEEN COPIED "
26401                        &CHR$(27)"&D"&"to "&To_disk$
26402                        PRINTER IS CRT
26403                        FOR I=1 TO Num_in_grp
26404                        ON ERROR GOTO 26414
26405                        File_name$=Cat_array$(I)[1,10]
26406                        COPY File_name$&From_disk$ TO File_name$&To_disk$
26407                        CALL Pline(0,"FILE"&CHR$(129)" "&File_name$&" "&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_name$
26411                        PRINT "FILE"&CHR$(129)" "&File_name$&" "&CHR$(128)&"HAS BEEN COPIED FROM DISK "&From_disk$" TO DISK "&To_disk$"
26412                        PRINTER IS CRT
26413                        GOTO 26420
26414                        PRINT TAB(10);"File "&File_name$&" was NOT successfully copied..."
26415                        PRINT ERRM$
26416                        PRINTER IS PRT
26417                        PRINT TAB(10);"File "&File_name$&" was NOT successfully copied..."
26418                        PRINT ERRM$
26419                        PRINTER IS CRT
26420                        NEXT I
26421                        ON ERROR CALL Error
26422                        CALL Pline(0," FILE COPYS COMPLETE &CHR$(128)&"
26423                        !
26424                        !
26425                        IF Code=4030 THEN ! If a MOVE, then purge the original file
26426                        PRINTER IS PRT;WIDTH 108
26427                        PRINT
26428                        PRINT Number_copied;" FILES HAVE BEEN COPIED; PURGING ORIGINALES"
26429                        ON ERROR GOTO 26432
26430                        FOR I=1 TO Number_copied
26431                        PURGE Cat_array$(I)[1,10]&From_disk$
26432                        NEXT I
26433                        ON ERROR CALL Error
PRINT "MOVE FILES function is complete"

PRINT

PRINTER IS CRT

END IF

END SELECT

END SELECT

END SELECT

END SELECT

SUBEND

!

!
This routine logs mean value data received from another CPU via the HP-IB bus. This bus is SC-8, and the primary address is 01. This end is NOT System Controller!

The variables are logged in a disc file that contains a header record, a variable names record, and 100 observation records. It is an ASCII file.

This routine also computes coefficients of calibration for multiple wire probes thru the use of multiple linear regression. Alternatively, the coefficients can be entered either by reading a coefficient file, or by manually keying in the coefficients -

With the calibration coefficients, this routine can generate sensitivities for the designated probe for each observation previously logged.

This routine can then apply the sensitivities to dynamic HOTWIRE data!

in trace and returns:

1 HOTWIRE 1 fluctuating velocity \( u \) in trace 4
2 HOTWIRE 2 fluctuating density \( p \) in trace 5
3 HOTWIRE 3 fluctuating temperature \( T_o \) in trace 6

These calculations are performed only between cursor positions - if - both cursors of trace 1 are active; or from the only cursor to the end of the memory. Otherwise, sensitivity coefficients are applied to the entire sample.

Traces 1, 2, and 3 are automatically loaded from disk files that were previously recorded using a naming convention defined here:

If the log filename is of the format "XXX"

[where XXX is the test number]
then the hotwire data disk file names would have the format:

"MPXXYYZZmn"

[where M is the mode as follows:
R - Real time digitization
P - Playback digitzation
P is the probe selection as follows:
A = probe A
B = probe B
XX is the RUN number
YY is the POINT number
of the data most recently logged, and therefore most likely to define related mean conditions, and will contain calculated sensitivities, gains of the fluctuating voltages, etc
ZZ is the digitizer channel number (this naming convention assumes a 1-to-1 relationship:

probe "A"
chan 1 - hotwire 1
2
2
3
3

probe "B"
5
1
6
2
7
3

nn is the "serial number" automatically applied (unfortunately) by ACQUIRE, and must be ignored.

NOTE: the hotwire data disk file name is defined by ACQUIRE function "FILENAME".

Using the sensitivities, mean voltages, and gains from the "A" or "B" probe file, traces 4, 5, and 6 are scaled and created to represent the ratios:

trace 4 = velocity fluctuations / mean velocity
trace 5 = density fluctuations / mean density
trace 6 = temperature fluctuations / mean temperature

The mean value is removed from these ratios, and an rms value of the three ratios is stored as variables 47, 48, and 49 of the appropriate probe file - "A" or "B".

Additionally, the ratios above can be retrieved and used to compute correlations between velocity, density, and temperature fluctuations, and then, using these correlations, go on and compute massflow and pressure fluctuations. These results are stored much as the ratios above are stored.

OPTION BASE 1

COM /Arr/ INTEGER Arrowvar(*)
COM /Cross/ Crval(*),INTEGER Crtab(*),Crmptr
COM /Curs/ Caddm(*),INTEGER Cincr(*),Cpos(*),Scursor(*),Cactive,Cpixmin(*),Cpixmax(*),Cflag
COM /Error/ INTEGER Errf,Errtype
COM /Genvar/ Filename$(*),Discdev$,INTEGER Discmap(*),Recordnum
COM /Input/ Inpstr$,INTEGER Kposx,Kposy,Quitcode
COM /Keys/ Keylab$(*),INTEGER Keymap(*),Keymenu,Keyincr,Okeytype,Okeymenu
COM /Localvar/ Lvar$
COM /Mem/ Bsw$(*),Sw$(*),Sw(*),INTEGER X(*),Isw(*),Memlenb(*),Memlenu(*),Memstartb(*),Memstartu(*),Maxmem,Memmaxl,Tmem
COM /Memut/ Menulab$,Mvar$(*),Mlit$,Mvar(*),INTEGER Mivar(*),Menutab(*),Keycode(*),Numitems,Nummods,Numkeys,Menuptr,Xindex,Yindex
COM /Param/ Mval$,Mval(*),INTEGER Mstack(*),Mvallist(*),Mstackptr,Mvalptr,Mvallstrptr
COM /Rcl/ Rclstr$,INTEGER Rclstr_ptr(*),Rclptr,Rclnum
COM /Screen/ INTEGER Garray(*),Ctextx(*),Ctexty,Ctextn,Ctextw,Opco(*),Opco(*)
The A, B, and C files below contain the "calculated" variables related to: A) wall strut, B) floor strut, and C) Kulites (and 'other' 'big end' wires)

usercom:com /usr2/ Sfn$[20],REAL Hwsens(3,3),Sensinv(3,3),Mean(3),Mean parameter(3),Enorm(3),StdddevParam(3),MaxParam(6),MinParam(6),VoParam(6),VsParam(6)

COM /usr2/ Gain_code_14(17),Gain_code_57(8)

COM /usr2/ C_names$(10)[10],P_c(3,10) ! P_c is the coefficient file

COM /usr2/ Log_fn$[20],Data_set_title$[80],Logged_var_name$(50)[10],Obs_rec(50,300),Pkt_sc,Pkt$(47)[80],Pkt_avg(45),Num_avg,Max_vars,Rcvd_vars

COM /usr2/ Subfile_names$(20)[10],Subfile_chartst(20),Initial_obs,Ending_obs,Num_obs_rec,Num_obs_printed,Max_obs_rec,Tag_pkt$(10)[80]

COM /usr2/ Initial_probe,Ending_probe

usercom:COM /usr2/ A_fn$[20],A_set_title$[80],A_var_name$(50)[10],A_rec(50,300)

COM /usr2/ A_subfile_names$(20)[10],A_sub_chartst(20)

COM /usr2/ B_fn$[20],B_set_title$[80],B_var_name$(50)[10],B_rec(50,300)

COM /usr2/ B_subfile_names$(20)[10],B_sub_chartst(20)

COM /usr2/ C_fn$[20],C_set_title$[80],C_var_name$(50)[10],C_rec(50,300)

COM /usr2/ C_subfile_names$(20)[10],C_sub_chartst(20)

COM /usr2/ Hw_rms(3)

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

INTEGER Routinl

REAL Temp,Wmark(6,2),Vo(6),Vs(6),Sum_param(3),SumSq_param(3),Vdp(3),Wire_gain(4)

INTEGER V(3)

DIM V$[30],File_comments$[80]

Routinl=Routinl

SELECT Routine

CASE 1 ! init pass !

RESTORE Menulist

LOOP

READ V$

EXIT IF V$="***"

READ V(*)

CALL Chkitem(V$)

END LOOP

CASE 2 ! init pass 2

RESTORE Keylist

LOOP

Errfl=0

Errtype=1

READ V(1)

EXIT IF V(1)<0
READ V(2)
CALL Chkkey(V(l),V(2))
END LOOP

Menulist:

LABEL, Function, Flag1, Flag2
DATA "HOTWIRE MENU", 4100, 0, 6
DATA "COEF FILENAME", 4101, 8704, 22
DATA "LOAD COEFS", 4102, 0, 262
DATA "ENTER COEFS", 4103, 0, 388
DATA "CALC VEL etc", 4104, 0, 262
DATA "LOG FILENAME", 4105, 8704, 22
DATA "LOAD LOGFILE", 4106, 0, 262
DATA "PRNT LOGFILE", 4107, 0, 262
DATA "LOG DATA POINT", 4108, 0, 390
DATA "LOG DATA", 4109, 0, 6
DATA "SAMPLES TO AVG", 4110, 24608, 22
DATA "COMPUTE SENS.", 4111, 0, 390
DATA "INITIAL OBS", 4112, 24608, 22
DATA "ENDING OBS", 4113, 24608, 22
DATA "HOTWIRE CALC", 4120, 0, 6
DATA "GET COEF", 4122, 0, 6
DATA "COMPUTE COEFS", 4123, 0, 390
DATA "STORE COEFS", 4124, 0, 262
DATA "CODE TO GAINS", 4125, 0, 388
DATA "FILE TRANSFERS", 4126, 0, 6
DATA "LOGFILE TO PC", 4127, 0, 388
DATA "Remake Probe", 4128, 0, 388
DATA "SELECTOR", 4129, 0, 6
DATA "INITIAL PROBE", 4130, 24608, 22
DATA "ENDING PROBE", 4131, 24608, 22
DATA "COMPUTE R etc", 4132, 0, 388
DATA "EDIT COEFS", 4133, 0, 388
DATA "***"

Keylist:

DATA 0, 4100, !HOTWIRE MENU
DATA 300, 4129, !SELECTOR
DATA 320, 4112, !INITIAL OBS
DATA 330, 4113, !ENDING OBS
DATA 340, 4130, !INITIAL PROBE
DATA 350, 4131, !ENDING PROBE
DATA 370, 4110, !SAMPLES TO AVG
DATA 400, 4120, !HOTWIRE CALC
DATA 420, 4122, !GET COEFS
DATA 421, 4101, !COEF FILENAME
DATA 422, 4102, !LOAD COEFS
DATA 423, 4103, !ENTER COEFS
DATA 424, 4133, !EDIT COEFS
DATA 425, 4123, !COMPUTE COEFS
DATA 427, 4124, !STORE COEFS
DATA 430, 4111, !COMPUTE SENS
DATA 440, 4104, !CALC VEL etc
DATA 450, 4132, !COMPUTE E etc
DATA 460, 4125, !XLATE GAIN CODES
DATA 470, 4128, !Remake Probe Data
DATA 500, 4109, !LOG DATA
DATA 510, 4105, !LOG FILENAME
DATA 520, 4106, !LOAD LOGFILE
DATA 530, 4107, !PRNT LOGFILE
Page 54

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$(*)
24273  Coef_names:  ! THESE ARE THE ORDER IN WHICH THE COEFFICIENTS ARE PROC
24274  DATA "CONSTANT"
24275  DATA "L(U)"
24276  DATA "L(Rho)"
24277  DATA "L(TO)"
24278  DATA "L(R)L(TO)"
24279  DATA "L(R)L(U)"
24280  DATA "L(U)L(TO)"
24281  DATA "LULRLTO"
24282  DATA "(L(U))**2"
24283  DATA "unused"
24284  !
24285  RESTORE Code_14
24286  READ Gain_code_14(*) !GAIN CODE CONVERSION TO GAINS FOR HOTWIRES 1-4
24287  Code_14:  !
24288  DATA 0  ! CODE 0  off; not defined
24289  DATA .25  ! 1
24290  DATA .5  ! 2
24291  DATA 1.
24292  DATA 1.99
24293  DATA 3.98
24294  DATA 7.84
24295  DATA 15.8
24296  DATA 31.6
24297  DATA 63.
24298  DATA 125.
24299  DATA 251.
24300  DATA 501.
24301  DATA 1000.
24302  DATA 1995.
24303  DATA 3981.
24304  DATA 7943.
24305  RESTORE Code_57
24306  READ Gain_code_57(*) !GAIN CODE CONVERSION TO GAINS FOR HOTWIRES 5-7
24307  Code_57:  !
24308  DATA 0  ! CODE 0
24309  DATA 1.  ! CODE 1
24310  DATA 2.
24311  DATA 5.
24312  DATA 10.
24313  DATA 20.
24314  DATA 50.
24315  DATA 100.
24316  Max_obs_rec=300
24317  Max_vars=50
24318  Rcvd_vars=45
24319  CASE 4  ! POWER ON VARIABLE INITIALIZATION
24320  !
24321  Sfn$="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 ! CASE 4101 !enter HW COEFFICIENT FILE NAME 
24336 SELECT Routine 
24337 CASE 21 !get old data, set up parameters 
24338 Mvar$(3)=Sfn$ !present setting 
24339 CASE 22 !store new data 
24340 Sfn$=Mvar$(3) 
24341 END SELECT 
24342 ! 
24343 ! CASE 4111 ! COMPUTE SENSITIVITIES 
24344 SELECT Routine 
24345 CASE 31 
24346 FOR Probe=Initial_probe TO Ending_probe 
24347 FOR R=Initial_obs TO Ending_obs 
24348 FOR W=1 TO 3 
24349 CASE Probe 
24350 ! S(U) =A2 +A6 *Log(Rho) +A7 *Log(T0) +A8 *Log(Rho)*Log(T0)+A9*2.*Log(U) 
24351 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) 
24352 ! S(Rho) =A3 +A5 *Log(T0)+A6 *Log(U) 
24353 +A8 *Log(U)*Log(T0) 
24354 ! S(T0) =A4 +A5 *Log(Rho)+A7 *Log(U) 
24355 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,7)*A_rec(10,R)+P_c(W,8)*A_rec(16,R) 
24356 ! S(Rho) =A3 +A5 *Log(T0)+A6 *Log(U) 
24357 ! S(T0) =A4 +A5 *Log(Rho)+A7 *Log(U) 
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) 
24363 ! S(T0) =A4 +A5 *Log(Rho)+A7 *Log(U) 
24364 ! S(Rho) =A3 +A5 *Log(T0)+A6 *Log(U) 
24365 ! S(T0) =A4 +A5 *Log(Rho)+A7 *Log(U) 
24366 END SELECT 
24367 NEXT W 
24368 NEXT R 
24369 NEXT Probe 
24370 GOSUB Log_vars ! Update the disk files 
24371 END SELECT
CASE 4123    ! COMPUTE COEFFicients
  SELECT Routine CASE 31
  CALL Mlr    !Perform Multiple Linear Regression
      ! for the probes selected
END SELECT
CASE 4124    ! STORE COEFFicient file
  SELECT Routine CASE 31
  Errf=0
  CREATE BDAT Sfn$,1,256
  ASSIGN @Disk TO Sfn$
  IF Errf=0 THEN OUTPUT @Disk;P_c(*)
  ASSIGN @Disk TO *
END SELECT
CASE 4102    ! LOAD COEFFicient file
  SELECT Routine CASE 31
  Errf=0
  ASSIGN @Disk TO Sfn$
  IF Errf=0 THEN ENTER @Disk;P_e(*)
  ASSIGN @Disk TO *
PRINTER IS PRT
PRINT CHR$(12)
PRINT "THESE ARE THE COEFFICIENTS FROM COEFFICIENT FILE ";Sfn$
PRINT
PRINT
PRINT
PRINT PRINT TAB(5);"WIRE 1";TAB(25);"WIRE 2";TAB(45);"WIRE 3"
PRINT
FOR I=1 TO 10
  PRINT P_c(I,1);TAB(20);P_c(I,2);TAB(40);P_c(I,3)
NEXT I
PRINTER IS CRT
END
SELECT
SELECT Routine CASE 31
CASE 4103,4133    ! ENTER COEFF file; EDIT COEFF file
  IF Code=4133 THEN GOTO Edit_coefs
  FOR I=1 TO 3
  BEEP
  FOR J=1 TO 10
  CALL Pline(0,"Enter the "&C_names$(J)&" (A"&VAL$(J)&") coefficient for wire ";" Andersen probe");Ans$=FNInput$(2,"NO")
CALL Pline(17,"
"
SELECT Ans$
CASE "YES","YE","Y"
CASE ELSE
CALL Pline(15,"Enter the wire # (1-3) whose coefficients need changing:"
I=VAL(FNInput$(3,"1"))
IF I<1 OR I>3 THEN GOTO 24433
CALL Pline(15,"Enter the coefficient number (A#) which is to be entered:"
J=VAL(FNInput$(3,"1"))
IF J<1 OR J>10 THEN GOTO 24436
CALL Pline(15,"Enter the "%C_names$(J)" (A"&VAL$(J)" and coefficient for wire "&VAL$(I)"):
ON ERROR RECOVER 24443
Temp$=FNInput$(2)
P_c(I,J)=VAL(Temp$
ON ERROR CALL Error
CALL Pline(2+J,VAL$(P_c(1,J))&"&VAL$(P-c(3,J)))
UNTIL Ans$="Y" OR Ans$="YES" OR Ans$="YE"
END SELECT
END SELECT
CASE 4125 !XLATE GAIN CODES TO GAINS
SELECT Routine
CASE 31
FOR This_obs=Initial_obs TO Ending_obs
FOR Probe=Initial_probe TO Ending_probe
! GET MEAN VALUE & GAIN FROM THE LOG FILE
SELECT Probe
CASE 1
FOR Wire=1 TO 4
!####
SELECT Obs_rec(36,This_obs)
SELECT 1
CASE 1 TO 16
!####
Wire_gain(Wire)=Gain_code_14(Obs_rec(36,This_obs)+1)
wire_gain(Wire)=Obs_rec(36,This_obs)
A_rec(41+Wire,This_obs)=Wire_gain(Wire)
A_var_name$(41+Wire)="GAIN "&VAL$(Wire)
END SELECT
NEXT Wire
CASE 2
FOR Wire=1 TO 3
!####
SELECT Obs_rec(36+Wire,This_obs)
SELECT 1
CASE 1 TO 7
!####
Wire_gain(Wire)=Gain_code_57(Obs_rec(36+Wire,This_obs)+1)
wire_gain(Wire)=Obs_rec(36+Wire,This_obs)
B_rec(41+Wire,This_obs)=Wire_gain(Wire)
B var_name$(41+Wire)="GAIN "&VAL$(Wire)
CASE ELSE
B_rec(41+Wire,This_obs)=0!Default gain = 0
B var_name$(41+Wire)="GAIN "&VAL$(Wire)
END SELECT
NEXT Wire
END SELECT
NEXT Probe
NEXT This_obs
GOSUB Log_vars
END SELECT
GOSUB Tun_vars
FOR Probe=Initial_probe TO Ending_probe
SELECT Probe
CASE 1
CALL Pline(Staty,"Recalculating probe A for observation "," &VAL$(This_obs))
GOSUB Wall_vars
CASE 2
CALL Pline(Staty,"Recalculating probe B for observation "," &VAL$(This_obs))
GOSUB Floor_vars
END SELECT
NEXT Probe
CALL Pline(Staty,
NEXT This_obs
GOSUB Log_vars
END SELECT
CASE 4128 !REMAKE PROBE DATA
SELECT Routin1
CASE 31
FOR This_obs=Initial_obs TO Ending_obs
CALL Pline(Staty,"Recalculating Tunnel parameters for observation "," &VAL$(This_obs))
GOSUB Tun_vars
FOR Probe=Initial_probe TO Ending_probe
SELECT Probe
CASE 1
CALL Pline(Staty,"Recalculating probe A for observation "," &VAL$(This_obs))
GOSUB Wall_vars
CASE 2
CALL Pline(Staty,"Recalculating probe B for observation "," &VAL$(This_obs))
GOSUB Floor_vars
END SELECT
NEXT Probe
CALL Pline(Staty,
NEXT This_obs
GOSUB Log_vars
END SELECT
CASE 4132 !COMPUTE R [CORRELATIONS] etc
SELECT Routin1
CASE 31
CLEAR ALL DATA TRANSFERS TO START
Mvar(14)=0
Mvar(2)=0
CALL Routine(22,239) !DATA FILE MAP
LOAD ONLY CHANNELS 2,3,4
FOR I=2 TO 4
Mvar(2)=1
Mvar(14)=I
CALL Routine(22,239) !DATA FILE MAP
NEXT I
FOR Probe=Initial_probe TO Ending_probe
FOR This_obs=Initial_obs TO Ending_obs
DISP "PROBE "," &VAL$(Probe)"; OBS "," &VAL$(This_obs)"; ";
SPECIFY FILENAMES:
FOR I=2 TO 4
Filename$(I)=" 
SELECT I
CASE 2
CASE 3
Filename$(I)[1,1]="D"
Filename$(I)[7,8]="05"
END SELECT

CASE 4
Filename$(I)[1,1]="T"
Filename$(I)[7,8]="06"

SELECT INT(Obs_rec(2,This_obs))
CASE 0 TO 9
Filename$(I)[3,4]="0"&VAL$(Obs_rec(2,This_obs))
!Run
CASE 10 TO 99
Filename$(I)[3,4]=VAL$(Obs_rec(2,This_obs))
END SELECT
SELECT INT(Obs_rec(3,This_obs))
CASE 0 TO 9
Filename$(I)[5,6]="0"&VAL$(Obs_rec(3,This_obs))
!Point
CASE 10 TO 99
Filename$(I)[5,6]=VAL$(Obs_rec(3,This_obs))
END SELECT
Mivar(14)=1
CALL Routine(21,232) !DISPLAY THE FILENAME

NEXT I
LOAD THE THREE CHANNELS INTO MEMORIES 2,3,4
(2=V 3=D 4=T):
CALL Routine(31,235) !LOAD DATA

FOR Xp=1 TO 3
DISP "CORRELATION ";
Mivar(14)=1
SELECT Xp
CASE 1
COPY MEMORY 3 TO MEMORY 1, leaving D in 1 and V in 2:
Mvar(2)=3
CALL Routine(22,280)!MEMORY COPY V TO 1
CASE 2
COPY MEMORY 4 TO MEMORY 1, leaving T in 1 and V in 2:
Mvar(2)=4
CALL Routine(22,280)!MEMORY COPY T TO 1
CASE 3
COPY MEMORY 3 TO MEMORY 1, and
COPY MEMORY 4 TO MEMORY 2, leaving D in 1 and T in 2:
Mvar(2)=3
CALL Routine(22,280)!MEMORY COPY D TO 1
Mivar(14)=2
Mvar(2)=4
CALL Routine(22,280)!MEMORY COPY T TO 2
END SELECT

MULTIPLY MEMORY 1 BY MEMORY 2:
Mivar(14)=0
CALL Routine(31,533)!MEMORY MULTIPLY

FIND MEAN OF MEMORY 1
Mivar(14)=1
CALL Routine(31,511)!MEAN
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
   SELECT Xp
   CASE 1!D*V
      Temp_rms=A_rec(48,This_obs)*A_rec(47,This_obs)
   CASE 2!V*T
      Temp_rms=A_rec(47,This_obs)*A_rec(49,This_obs)
   CASE 3!D*T
      Temp_rms=A_rec(48,This_obs)*A_rec(49,This_obs)
24607
24608
24609
24610
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
   Temp_rms=B_rec(48,This_obs)*B_rec(47,This_obs)
   CASE 2!V*T
   Temp_rms=B_rec(47,This_obs)*B_rec(49,This_obs)
   CASE 3!D*T
   Temp_rms=B_rec(48,This_obs)*B_rec(49,This_obs)
24620
24621
24622 END SELECT
24623 B_rec(25+Xp,This_obs)=Temp_mean/Temp rms
24624
24625 !
24626 SELECT Probe
24627 CASE 1
   A_var_name$(26)="R(RhoU)"
   A_var_name$(27)="R(UTO)"
   A_var_name$(28)="R(RhoTO)"
   CASE 2
   B_var_name$(26)="R(RhoU)"
   B_var_name$(27)="R(UTO)"
   B_var_name$(28)="R(RhoTO)"
24630 END SELECT
24631 GOSUB Log_vars
24632 NEXT Xp
24633 NEXT This_obs
24634 NEXT Probe
24635
24636 Etc4132: ! COMPUTE M'/M, P'/P
24639 FOR Probe=Initial_probe TO Ending_probe
24640 FOR This_obs=Initial_obs TO Ending_obs
24641 Mach=Obs_rec(9,This_obs)
24642 M2=Mach*Mach
24643 SELECT Probe
24644 CASE 1
   U="A_rec(47,This_obs)
   Rho=U="/A_rec(48,This_obs)
   TO=U="/A_rec(49,This_obs)
   CASE 2
   U="B_rec(47,This_obs)
   Rho=U="/A_rec(48,This_obs)
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 Routin1
CASE 31
!perform action
END SELECT

FOR Probe=Initial_probe TO Ending_probe
  FOR This_obs=Initial_obs TO Ending_obs
    FOR Wire=1 TO 3
      SELECT Probe! GET MEAN VALUE & GAIN FROM THE LOG FI
CASE 1
  Mean(Wire)=A_rec(17+Wire,This_obs)
  Wire_gain(Wire)=A_rec(41+Wire,This_obs)

CASE 2
  Mean(Wire)=B_rec(17+Wire,Initial_obs)
  Wire_gain(Wire)=B_rec(41+Wire,This_obs)

END SELECT

FOR Sens=1 TO 3! GET THE SENSITIVITIES FROM THE LOG F

SELECT Probe
CASE 1
  Hwsens(Wire,Sens)=A_rec(25+(Sens*4)+Wire,This_obs)

CASE 2
  Hwsens(Wire,Sens)=B_rec(25+(Sens*4)+Wire,This_obs)

CASE 3
  Hwsens(Wire,Sens)=C_rec(25+(Sens*4)+Wire,This_obs)

END SELECT

NEXT Sens

SET UP DATA FILE MAP FOR TRACES 1,2,3 ONLY

Mivar(14)=Wire
Mvar(2)=1
CALL Routine(22,239)! TURN ON data files 1,2, or 3
Mivar(14)=3+Wire
Mvar(2)=0
CALL Routine(22,239)! TURN OFF data files 4,5, or 6

Temp$[1]="R"
Temp$[2]=CHRS(64+Probe)
SELECT INT(Obs_rec(2,This_obs))
CASE 0 TO 9
  Temp$[3,4]="0"&VAL$(INT(Obs_rec(2,This_obs)))!RUN
UN

CASE 10 TO 99
  Temp$[3,4]=VAL$(INT(Obs_rec(2,This_obs)))!RUN
END SELECT

SELECT INT(Obs_rec(3,This_obs))
CASE 0 TO 9
  Temp$[5,6]="0"&VAL$(INT(Obs_rec(3,This_obs)))!POINT
UN

CASE 10 TO 99
  Temp$[5,6]=VAL$(INT(Obs_rec(3,This_obs)))!POINT
END SELECT

SELECT Probe
CASE 1
  Temp$[7,8]="0"&VAL$(Wire)
CASE 2
  Temp$[7,8]="0"&VAL$(Wire+4)
END SELECT

Temp$[9,9]="1"

DISP Temp$
WAIT 1

Filename$(Wire)=Temp$
Mivar(14)=Wire
CALL Routine(21,232)! DISPLAY THE FILENAME
TRANSFER CHANNEL CHARACTERISTICS
TO COMPUTED CHANNELS

FOR I=1 TO 15
    Sw(Wire+3,I)=Sw(Wire,I)
NEXT I
NEXT Wire

CALL Routine(43,235) ! LOAD DATA
CALL Routine(31,235)
FOR I=1 TO 3 ! For each channel
    CALL Routine(43,235)
    CLEAR MEMORY TAGS FOR "REMOVE MEAN" FUNCTION
    FOR J=0 TO 2
        Sw$(I,13+J)="
    NEXT J
Mivar(14)=I
CALL Routine(31,524)! REMOVE MEAN
CALL Routine(31,513)! FIND RMS
Hw_rms(I)=Mvar(3)
NEXT I

TRACE 1, 2, AND 3 POINT TO THE MEMORIES CONTAINING THE FLUCTUATING COMPONENT ONLY

SET UP DATA FILE MAP FOR RESULTS: 4, 5, 6 ONLY
FOR Results=4 TO 6
    Mivar(14)=Results-3
    Mvar(2)=0
    CALL Routine(22,239)! TURN OFF data files 1, 2, 3
    CALL Routine(31,524)! REMOVE MEAN
    Mivar(14)=Results
    Mvar(2)=1
    CALL Routine(31,513)! FIND RMS
    Hw_rms(I)=Mvar(3)
NEXT I

SELECT Results
CASE 4 ! Velocity
    Temp$[1]="V"
CASE 5 ! Density
    Temp$[1]="D"
CASE 6 ! Temperature
    Temp$[1]="T"
END SELECT

SELECT Probe
CASE 1
    Temp$[5,6]="0"&VAL$(Results)
CASE 2
    END SELECT
24824
Temp$[7,8]="0"&VAL$(Results+4)
END SELECT
Filename$(Results)=Temp$
Mivar(l4)=Results
CALL Routine(21,232)! DISPLAY THE FILENAME
NEXT Results
!
FILENAME SET UP TO STORE DATA FILES
!
MAT Sensinv= INV(Hwsens)
!
DETERMINE SCALING (from a sampling of the data)
!
Wtr=1
Numb_samp=Sw(Wtr,2)
SELECT Numb_samp
CASE <100
Numb_sub=Numb_samp
CASE 100 TO 10000
Numb_sub=100
CASE >10000
Numb_sub=Numb_samp/100
END SELECT
!
DETERMINE Mean, Standard deviation, Min, Max, range, Offset, Scale of velocity, density, and temperature.
!
Value=0
FOR Wtr=1 TO 6!pick up trace number
Wmem(Wtr)=Trmem(Wtr)!pick up the memory number
IF (BIT(Cactive,Wtr*2-1)) THEN ! ACTIVE 1st CURSOR
Wmark(Wtr,1)=Caddrx(Wtr*2-1)!get the position of first cursor
ELSE ! IN-ACTIVE 1st CURSOR
Wmark(Wtr,1)=0
END IF
!
ELSE ! IN-ACTIVE 2nd CURSOR
IF (BIT(Cactive,Wtr*2)) THEN ! ACTIVE 2nd CURSOR
Wmark(Wtr,2)=Caddrx(Wtr*2)
ELSE ! IN-ACTIVE 2nd CURSOR
Wmark(Wtr,2)=Sw(Wtr,2)!no second cursor; use
END IF
!
END IF
!
Wmark(Wtr,2)>Wmark(Wtr,1) THEN ! assure starting position is <=
Temp=Wmark(Wtr,1)! to ending position
Wmark(Wtr,1)=Wmark(Wtr,2)
Wmark(Wtr,2)=Temp
END IF
!
Xch(Wtr)=((FNMs(1,Wmem(Wtr))+Wmark(Wtr,1)) DIV 10
Xpt(Wtr)=((FNMs(1,Wmem(Wtr))+Wmark(Wtr,1)) MOD 10
!
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  ! FOR THE SAMPLE POINTS:
24884  FOR I=Strtpt TO Stoppt STEP Numb Samp/Numb Sub
24885  FOR Wire=1 TO 3
24886  IF FNMemms(1,Trmem(Wire))+I<=Sw(Wire,2) THEN
24887  Xsch(Wire)=(((FNMemms(1,Trmem(Wire))+I)+I) DIV 1
24888  024)+1
24889  Xspt(Wire)=(((FNMemms(1,Trmem(Wire))+I) MOD 1
24890  024)+1
24891  Temp=X(Xsch(Wire),Xspt(Wire))
24892  Value=((Temp/65536)-Vo(Wire))*Vs(Wire)
24893  Enorm(Wire)=Value/(Mean(Wire)*Wire*gain(Wir
e))
24894  IF Idiag<9 AND Wire=1 AND I<900 THEN
24895  PRINT I;"th Sample -(Wire";Wire;")- X: 
24896  "Temp;" X(volts): ";Value;" Enorm:";Enorm(Wire)
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
24910  p+3)=Vdp(Ip)
24911  IF Vdp(Ip)<Min_param(Ip+3) THEN Min_param(I
24912  p+3)=Vdp(Ip)
24913  Sum_param(Ip)=Sum_param(Ip)+Vdp(Ip)
24914  SumsqParam(Ip)=Sumsq_param(Ip)+(Vdp(Ip)*Vd
24915  p(Ip))
24916  IF Idiag AND Ip=3 THEN
24917  PRINTER IS PRT
24918  PRINT "TEST VALUES ";Vdp(*)
24919  PRINT "MAX ";Max_param(*)
24920  PRINT "MIN ";Min_param(*)
24921  PRINTER IS CRT
24922  END IF
24923  END SELECT
24924  NEXT Ip
24925  NEXT I
24926  CALCULATE THE SLOPE (Vs_param) AND INTERCEPT (Vo_param)
24927  FOR Ip=1 TO 3
24928  Mean_param(Ip)=Sum_param(Ip)/Numb Sub
IF ((Sumsq_param(Ip)-(Numb_sub*Mean_param(Ip)*Mean_param(Ip)))/(Mean_param(Ip)-1))>0 THEN
Stddev_param(Ip)=SQR((Sumsq_param(Ip)-(Numb_sub*Mean_param(Ip)*Mean_param(Ip)))/(Mean_param(Ip)-1))
ELSE
Stddev_param(Ip)=0
END IF

Vo_param(Ip+3)=((Max_param(Ip+3)+Min_param(Ip+3))/2).
IF Max_param(Ip+3)-Vo_param(Ip+3)>0 THEN
Vs_param(Ip+3)=10*((Max_param(Ip+3)-Vo_param(Ip+3)))/32767)
ELSE
Vs_param(Ip+3)=10.
END IF

STORE OFFSETS AND SLOPES IN THE MEMORIES POINTED TO BY TRACES 4, 5, 6
Sw(Ip+3,28)=Vo_param(Ip+3)
Sw(Ip+3,27)=Vs_param(Ip+3)

IDENTIFY DISPLAY AS USER UNITS
Isw(Ip+3,13)=1
SELECT Ip
CASE 1
Sw$(I+3,11)="u'/U"
CASE 2
Sw$(I+3,11)="p'/P"
CASE 3
Sw$(I+3,11)="t'/T"
END SELECT
NEXT Ip

COMPUTE u, p, T
FOR I=Strtpt TO Stoppt
FOR Wire=1 TO 3
Temp=X(Xch(Wire),Xpt(Wire))
Value=((Temp/65536)-Vo(Wire))*Vs(Wire)
Enorm(Wire)=Value/(Mean(Wire)*Wire_gain(Wire))
Xpt(Wire)=Xpt(Wire)+1
IF Xpt(Wire)>1024 THEN
Xpt(Wire)=1
Xch(Wire)=Xch(Wire)+1
END IF
NEXT Wire
MAT Vdp= Sensinv*Enorm
IF Idiag THEN
PRINTER IS PRT;WIDTH 134
IF I=Strtpt THEN
PRINT "MAX_P: ";Max_param(*)
PRINT "MIN_P: ";Min_param(*)
PRINT "SLOPE: ";Vs_param(*)
PRINT "INTERCEPT: ";Vo_param(*)
PRINT "Means: ";Mean(*)
PRINT "Gains: ";Wire_gain(*)
PRINT
END IF
PRINTER IS CRT
END IF
FOR Ip=4 TO 6! PUT u, p, TO IN MEMORIES UNDER TRAC
Value = INT((Vdp(Ip-3) - Vo_param(Ip))/Vs_param(Ip))

IF Idiag AND Ip=6 THEN
  IF I MOD 100=0 AND I<1500 THEN
    PRINT "PRI: WIDTH 134"
    PRINT "SAMPLE: ";I;"TRACE: ";Ip;"Vdp: ";Vdp(Ip-3);"Value: ";Value
  END IF
END IF

SELECT Value! Limit the range to the 16 bit int

CASE -32768 TO 32767
  X(Xch(Ip),Xpt(Ip))=Value
CASE >32767
  X(Xch(Ip),Xpt(Ip))=32767
CASE <32768
  X(Xch(Ip),Xpt(Ip))=-32768
END SELECT

IF Xpt(Ip)>1024 THEN
  X(Xch(Ip),Xpt(Ip))=-32768
  Xpt(Ip)=1
  Xch(Ip)=Xch(Ip)+1
END IF

NEXT Ip

DISP I

NEXT I

FOR Ip=1 TO 3! For each ratio (u'/U,p'/P, t0'/TO)

FOR I=4 TO 6! CLEAR MEMORY TAGS FOR "REMOVE MEAN" F

FUNCTION

FOR J=0 TO 2
  Sw$(I,13+J)=""
  NEXT J
END FUNCTION

NEXT I

Mivar(I4)=Ip+3
CALL Routine(31,524)! REMOVE MEAN
CALL Routine(31,513)! FIND RMS
Value=Mvar(3)
SELECT Probe

CASE 1
  A_rec(46+Ip,This_obs)=Value
CASE 2
  B_rec(46+Ip,This_obs)=Value
END SELECT

NEXT Ip

CALL Routine(43,234)! STORE FLUCTUATING DATA
CALL Routine(31,234)

RESTORE A_names
READ A_var_name$(*)
RESTORE B_names
READ B_var_name$(*)

GOSUB Log_vars ! Save the computed values on disc
PRINT "TEST ";obs_rec(1,This_obs)
PRINT "RUN ";obs_rec(2,This_obs)
PRINT "POINT ";obs_rec(3,This_obs)
PRINT "P_TOTAL ";obs_rec(7,This_obs)
PRINT "P_STATIC ";obs_rec(6,This_obs)
PRINT "T_TOTAL ";obs_rec(8,This_obs)
PRINT "MACH ";obs_rec(9,This_obs)
PRINT "V_mean";TAB(20);Mean(l);TAB(40);Mean(2);TAB(60);Mean(3)
PRINT "V_rms";TAB(20);Hw_rms(1);TAB(40);Hw_rms(2);TAB(60);Hw_rms(3)
PRINT "GAIN";TAB(20);Wire_gain(l);TAB(40);Wire_gain(2)
PRINT "SENSITIVITY";TAB(20);Hwsens(l,1);TAB(40);Hwsens(2,1)
PRINT TAB(20);Hwsens(l,2);TAB(40);Hwsens(2,2);TAB(60);Hwsens(3,1)
PRINT TAB(20);Hwsens(l,3);TAB(40);Hwsens(2,3);TAB(60);Hwsens(3,3)
SELECT Probe
CASE 1
Vel_f=A_rec(47,This_obs)
Dens_f=A_rec(48,This_obs)
Temp_f=A_rec(49,This_obs)
CASE 2
Vel_f=B_rec(47,This_obs)
Dens_f=B_rec(48,This_obs)
Temp_f=B_rec(49,This_obs)
END SELECT
PRINT "u'/U_rms";TAB(20);Vel_f
PRINT "p'/P_rms";TAB(20);Dens_f
PRINT "t'/T_rms";TAB(20);Temp_f
PRINT}
NEXT This_obs
NEXT Probe
END SELECT
Schg=BINIOR(Schg,16384) !SET BIT 14 TO SAY WAVEFORMS CHANGED
Log_file_menu: !
CASE 405 !enter LOG FILE NAME
CASE 21 !get old data, set up parameters
Mvar$(3)=Log_fn$ !present setting
CASE 22 !store new data
CASE 4106  ! LOAD LOG FILE
SELECT Routin1
CASE 31
ON ERROR GOTO No_file
ASSIGN @Disk TO Log_fn$
ON ERROR CALL Error
ENTER @Disk,1;Data_set_title$,Max_obs_rec,Max_vars,Logged_var_names(*),Numsubfile,Subfile_names$(*) ,Subfile_chartst(*)
Num_obs_rec=Subfile_chartst(1)
REDIM Obs_rec(Max_vars,Max_obs_rec),A_rec(Max_vars,Max_obs_rec),B_rec(Max_vars,Max_obs_rec)
! Enter wall strut file
ASSIGN @Diska TO Log-fn$&"A"
ENTER @Diska,1;A_set_title$,Dummy,Dummy,A_var_names$(*) ,A_subfile_names$(*) ,A_sub_chartst(*)
ENTER @Diska,2
ENTER @Diska;A_rec(*)
Set up floor strut file
ASSIGN @Diskb TO Log-fn$&"B"
ENTER @Diskb,1;B_set_title$,Dummy,Dummy,B_var_names$(*) ,B_subfile_names$(*) ,B_sub_chartst(*)
ENTER @Diskb,2
ENTER @Diskb;B_rec(*)
Set up other strut file
ASSIGN @Diskc TO Log-fn$&"C"
ENTER @Diskc,1;C_set_title$,Dummy,Dummy,C_var_names$(*) ,C_subfile_names$(*) ,C_sub_chartst(*)
ENTER @Diskc,2
ENTER @Diskc;C_rec(*)
GOTO Got_it
No_file:  ! NO FILE.  ! OK, SO CREATE THE FILE
ON ERROR CALL Error
Errf=0
CREATE BDAT Log_fn$,INT(8*Max_vars*Max_obs_rec/1280)+2,1280
ASSIGN @Disk TO Log_fn$
CREATE BDAT Log_fn$&"A",INT(8*Max_vars*Max_obs_rec/1280)+2,1280
ASSIGN @Diska TO Log_fn$&"A"
CREATE BDAT Log_fn$&"B",INT(8*Max_vars*Max_obs_rec/1280)+2,1280
ASSIGN @Diskb TO Log_fn$&"B"
CREATE BDAT Log_fn$&"C",INT(8*Max_vars*Max_obs_rec/1280)+2,1280
ASSIGN @Diskc TO Log_fn$&"C"
File_comments$=""
DISP "ENTER THE LOG FILE DATA SET TITLE - Return IF NONE";
INPUT ",File_comments$
IF LEN(File_comments$)>0 THEN
Data_set_title$=File_comments$
A_set_title$="WALL STRUT :::"&File_comments$
B_set_title$="FLOOR STRUT :::"&File_comments$
C_set_title$="KULITE, MISC :::"&File_comments$
ELSE
Data_set_title$=""
A_set_title$=""
B_set_title$=""
C_set_title$=""
END IF
Num_obs_rec'd=0
Numsubfile=0
MAT Subfile_names$=""
MAT A_subfile_names$=""
MAT B_subfile_names$=""
MAT C_subfile_names$=""
MAT Subfile_chartst=0
MAT A_sub_chartst=0
MAT B_sub_chartst=0
MAT C_sub_chartst=0
MAT Obs_rec= (-999999.9999)
MAT A_rec= (-999999.9999)
MAT B_rec= (-999999.9999)
MAT C_rec= (-999999.9999)

! RESTORE Var_names ! NOTE:
! IN FUTURE LINKS, THESE NAMES WILL PRECEDE
! THE VALUES IN THE DATA PACKET RECEIVED.

READ Logged_var_name$(*)

Var_names: THE VARIABLE NAMES THAT EACH OBSERVATION RECORD CONTAINS:
DATA "TEST" !1
DATA "RUN" !2
DATA "POINT" !3
DATA "TIME" !4
DATA "DATE" !5
DATA "Ps" !6
DATA "Pt" !7
DATA "Tt" !8
DATA "MACH" !9
DATA "REYN" !10
DATA "PsS1"
DATA "PsS2"
DATA "PsS3"
DATA "PsS4"
DATA "PsS5"
DATA "PsS6"
DATA "TtS1"
DATA "TtS2"
DATA "TtS3"
DATA "TtS4"
DATA "TtS5"
DATA "TtS6"
DATA "P1HW1"
DATA "P1HW2"
DATA "P1HW3"
DATA "P1HW4"
DATA "P2HW1"
DATA "P2HW2"
DATA "P2HW3"
DATA "P3HW1"
DATA "P4HW1"
DATA "P5HW1"
DATA "KULITE1" !30
DATA "KULITE2" !31
DATA "KULITE3" !32
DATA "KULITE4" !33
DATA "KULITE5" !34
DATA "KULITE6" !35
DATA "HW1-4 GAIN"
DATA "P2HW1GAIN"
DATA "P2HW2GAIN"
DATA "P2HW3GAIN"
DATA "K1 GAIN"
DATA "K2 GAIN"
DATA "K3 GAIN"
DATA "K4 GAIN"
DATA "K5 GAIN"
DATA "K6 GAIN"
DATA ""
DATA ""
DATA ""
DATA ""
DATA ""
DATA ""
DATA "P2HW2GAIN"
DATA "P2HW3GAIN"
DATA "K1 GAIN"
DATA "K2 GAIN"
DATA "K3 GAIN"
DATA "K4 GAIN"
DATA "K5 GAIN"
DATA "K6 GAIN"
DATA ""
DATA ""
DATA ""
DATA ""
DATA ""        !34
DATA "S(Rho)1"        !38
DATA "S(Rho)2"        !38
DATA "S(Rho)3"        !38
DATA ""            !41
DATA "GAIN 1"        !42
DATA "GAIN 2"        !42
DATA "GAIN 3"        !42
DATA "GAIN 4"        !42
DATA "P'/P"        !46
DATA "u'/U"        !47
DATA "p'/Pn"        !47
DATA "to'/To"        !47
DATA ""            !47
DATA "GAIN 1"        !42
DATA "GAIN 2"        !42
DATA "GAIN 3"        !42
DATA "GAIN 4"        !42
DATA "P'/P"        !46
DATA "u'/U"        !47
DATA "p'/Pn"        !47
DATA "to'/To"        !47
DATA ""            !47

OUTPUT @Diska,1;A_set_title$,Max_obs_rec,Max_vars,A_var_name$(*
Numsubfile,A_subfile_names$(*),Subfile_chartst(*)
STATUS @Diska,3;Norecs,Nobpr
CONTROL @Diska,7;Norecs,Nobpr
ENTER @Diska,2
OUTPUT @Diska;A_rec(*)

RESTORE B_names    !  NOTE:
! THESE ARE THE VARIABLE NAMES FOR THE
! FLOOR STRUT DATA FILE "B"

READ B_var_name$(*)
B_names:$ THE VARIABLE NAMES THAT EACH "B" DATA RECORD CONTAINS:
DATA "TEST"     ! 1
DATA "RUN"        ! 2
DATA "POINT"     ! 3
DATA "PtS2"
DATA "PsS2"
DATA "TcS2"
DATA "VELOCITY"
DATA "DENSITY"
DATA "Ts"
DATA "L(U)"
DATA "L(Rho)"
DATA "L(TO)"
DATA "L(RhoU)"
DATA "L(R) L(TO)"
DATA "L(R) L(U)"
DATA "L(U) L(TO)"
DATA "LULRLTO"
DATA "P2HW1"        !18
DATA "P2HW2"
DATA "P2HW3"
DATA ""        !18
DATA ""        !18
DATA ""        !18
DATA ""        !18
DATA ""        !18
DATA "R(Rho)U"
DATA "R(UTO)"
DATA "R(RhoTO)"
DATA "" 25323
DATA "S(U)1" !30 25324
DATA "S(U)2" 25325
DATA "S(U)3" 25326
DATA "" 25327
DATA "S(Rho)1" !34 25328
DATA "S(Rho)2" 25329
DATA "S(Rho)3" 25330
DATA "" 25331
DATA "S(T0)1" !38 25332
DATA "S(T0)2" 25333
DATA "S(T0)3" 25334
DATA "M'/M" !41 25335
DATA "GAIN 1" !42 25336
DATA "GAIN 2" 25337
DATA "GAIN 3" 25338
DATA "" 25339
DATA "P'/P" !46 25340
DATA "u'/U" !47 25341
DATA "p'/P" 25342
DATA "to'/To" 25343
DATA "" 25344
DATA "" 25345
OUTPUT @Diskb,1;B_set_title$,Max_obs_rec,Max_vars,B_var_name$(*),Numsubfile,B_subfile_names$(*),Subfile_chartst(*)
STATUS @Diskb,3;Norecs,Nobpr 25348
CONTROL @Diskb,7;Norecs,Nobpr 25349
ENTER @Diskb,2 25350
OUTPUT @Diskb;B_rec(*) 25351
RESTORE C names ! NOTE:
! THESE ARE THE VARIABLE NAMES FOR THE
! 'OTHER' STRUT DATA FILE "C"
READ C var_name$(*)
C_names:! THE VARIABLE NAMES THAT EACH "C" DATA RECORD CONTAINS:
DATA "TEST" ! 1 25358
DATA "RUN" ! 2 25359
DATA "POINT" ! 3 25360
DATA "PsS3" 25361
DATA "PsS3" 25362
DATA "TtS3" 25363
DATA "VELOCITY" 25364
DATA "DENSITY" 25365
DATA "Ts" 25366
DATA "L(U)" 25367
DATA "L(Rho)" 25368
DATA "L(TO)" 25369
DATA "L(RhoU)" 25370
DATA "L(RhoL(TO))" 25371
DATA "L(R)" 25372
DATA "L(R)" 25373
DATA "L(U)" 25374
DATA "LULRLO" 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(T0)1"   !38 
25396   DATA "S(T0)2" 
25397   DATA "S(T0)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$(*),Subfile_chartst(*) 
25434   Num_obs_recd=Subfile_chartst(1) 
25435   ! 
25436   REDIM Obs_rec(Max_vars,Max_obs_rec),A_rec(Max_vars,Max_obs_rec),B_rec(Max_vars,Max_obs_rec) 
25437   ! 
25438   ENTER @Disk,2
25439          IF Errf=0 AND Num_obs_rec>1 AND Num_obs_rec<Max_obs_rec THEN
25440          ENTER @Disk;Obs_rec(*)
25441          ASSIGN @Disk TO *
25442          !
25443          Errorf=0
25444          ASSIGN @Disk TO Log_fn$&"A"
25445          IF Errorf=0 THEN ENTER @Diska,1;A_set_title$,Dummy,Dummy,A_var_name$(*),Dummy,A_subfile_names$(*),A_sub_chartst(*)
25446          ENTER @Diska,2
25447          IF Errorf=0 AND Num_obs_rec>1 AND Num_obs_rec<Max_obs_rec THEN
25448          ASSIGN @Disk TO *
25449          !
25450          Errorf=0
25451          ASSIGN @Diskb TO Log_fn$&"B"
25452          IF Errorf=0 THEN ENTER @Diskb,1;B_set_title$,Dummy,Dummy,B_var_name$(*),Dummy,B_subfile_names$(*),B_sub_chartst(*)
25453          ENTER @Diskb,2
25454          IF Errorf=0 AND Num_obs_rec>1 AND Num_obs_rec<Max_obs_rec THEN
25455          ASSIGN @Disk TO *
25456          !
25457          Errorf=0
25458          ASSIGN @Diskc TO Log_fn$&"C"
25459          IF Errorf=0 THEN ENTER @Diskc,1;C_set_title$,Dummy,Dummy,C_var_name$(*),Dummy,C_subfile_names$(*),C_sub_chartst(*)
25460          ENTER @Diskc,2
25461          IF Errorf=0 AND Num_obs_rec>1 AND Num_obs_rec<Max_obs_rec THEN
25462          ASSIGN @Disk TO *
25463          GOSUB Print_log_data
25464          GOTO End_4107
25465          !
25466          !
25467          Print_log_data: !Subroutine to print a formatted report.
25468          !
25469          !
25470          IF Num_obs_printed=0 THEN Num_obs_printed=Initial_obs-1
25471          WHILE Num_obs_printed<Ending_obs
25472          Num_obs_printed=Num_obs_printed+1
25473          PRINTER IS PRT;WIDTH 108
25474          PRINT CHR$(12) !Form feed
25475          PRINT Data_set_title$;TAB(60);"OBSERVATION # ";Num_obs_printed
25476          IF Num_obs_printed=0 AND Num_obs_printed<Max_obs_rec THEN
25477          PRINT "FILE ";Log_fn$
25478          PRINT "TEST ";Obs_rec(1,Num_obs_printed)
25479          PRINT "RUN ";Obs_rec(2,Num_obs_printed);TAB(18);"LOG"
25480          PRINT 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)"&dDn!Underline
25484          Nu$_$=CHR$(27)"&d@"No underline
25485          PRINT TAB(23);U_$&"CONDITIONS"&Nu$_$;TAB(54);U_$&"CON DiO
25486          PRINT "Mach":TAB(23);Obs_rec(9,Num_obs_printed)
25487          PRINT "Mach":TAB(23);Obs_rec(9,Num_obs_printed)
PRINT "Reynolds No."
PRINT "Pt"
PRINT "Ps"
PRINT "Tc"
PRINT "Velocity"
PRINT "MEAN(HW1)"
PRINT "MEAN(HW2)"
PRINT "MEAN(HW3)"
PRINT "S(U) (HW1)"
PRINT "S(Rho)(HW1)"
PRINT "S(To) (HW1)"
PRINT "S(U) (HW2)"
PRINT "S(Rho)(HW2)"
PRINT "S(To) (HW2)"
PRINT "S(U) (HW3)"
PRINT "S(Rho)(HW3)"
PRINT "S(To) (HW3)"
PRINT "R(RhoU)"
PRINT "R(UTO)"
PRINT "R(RhoTO)"
PRINT "ttM'/M"
PRINT "p'/Pn"
END IF

PRINTER IS CRT

END WHILE

RETURN

End 4107:END SELECT

CASE 4108

SELECT Routin1

CASE 31

This routine is the acquisition and data logging portion of the hotwire
dynamic data system.

Steve Clukey, Vigyan Research Associates, September, 1987

Data is received from another computer (Modcomp) via HP-IB in ASCII.
Upon request of the operator, this data is averaged, and stored (logged)
on disk as an observation in an X-Y array. Some values are calculated
and stored along with the data received from the other computer.
These data observations will be used to compute hotwire sensitivities,
and therefore the format of the disk file should be compatible with
the computational programs.

FOR each observation, "Max_vars" variables are stored:

THE PROCESS IS:

1. Create an observation file, or declare an existing file, and set up
   to append (with additional records) additional observations.

2. When requested by the operator, data is read from the other computer
   via HP-IB. Both the data hotwire data and the tunnel parameter data
   are averaged, and stored in the next available record.

3. When requested by the operator, the file is closed.

At some later time, the file will be processed to provide a hotwire
sensitivity file.

CALL Pline(0,"Observation logging started")

WAIT 1

Pkt_sc=8

MAT Pkt_avg= (0)

I=0

FOR I=1 TO Num_avgs

GOsub Enable_link

GOsub Get_packet

IF I=1 THEN !Don't average test, run, point......

FOR J=1 TO 3

    Pkt_avg(J)=VAL(Pkt$(J+1))*Num_avgs

NEXT J

FOR J=4 TO 5

    Pkt_avg(J)=0! This was date and time
NEXT J
FOR J-1 TO 10! SAVE THE FIRST 10 FOR "TAG PLOT, PICTURE"
Tag_pkt$(J)=Pkt$(J+1)
NEXT J
END IF
FOR J=6 TO Rcvd_vars
ON ERROR GOTO 25589! Ignore error if not a good VAL
Pkt_avg(J)=Pkt_avg(J)+VAL(Pkt$(J+1))
ON ERROR CALL Error
NEXT J

!!!!!
GOSUB Disable_link
MAT Pkt_avg= Pkt_avg/(Num_avgs)! FIND THE AVERAGE

! Get ready to log data - by opening files
ASSIGN @Disk TO Log_fn$
ASSIGN @Diska TO Log_fn$&"A"
ASSIGN @Diskb TO Log_fn$&"B"

!!!!
ASSIGN @Diskc TO Log_fn$&"C"

! Find out where we are - retrieve number of observations recorded
ENTER @Disk,1;Data_set_title$,Max_obs_rec,Max_vars,Logged_var_name$(*) Numsubfile,Subfile_names$(*) Subfile_chartst(*)
Num_obs_rec=Subfile_chartst(1)
This_obs=MAX(Num_obs_rec+1,1)! bump the observations pointer
IF This_obs<Max_obs_rec THEN! move the data to the observation record
FOR I=1 TO Rcvd_vars
Obs_rec(I,This_obs)=Pkt_avg(I)
NEXT I

Num_obs_rec=This_obs! prepare to save the observation pointer
Ending_obs=Num_obs_rec
Subfile_chartst(1)=Num_obs_rec

! SET UP DATA FILE MAP FOR TRACES 1 THRU NUMMEM
Mivar(14)=I
Mvar(2)=1
CALL Routine(22,239)! TURN ON data files

Temp$[1]="R"
SELECT I
CASE 1 TO 4
Temp$[2]="A"
CASE 5 TO 7
Temp$[2]="B"
CASE 8 TO 99
Temp$[2]="C"
END SELECT
SELECT INT(Obs_rec(2,This_obs))
CASE 0 TO 9
Temp$[3,4]="0"&VAL$(INT(Obs_rec(2,This_obs)))! RUN
CASE 10 TO 99
Temp$[3,4]=VAL$(INT(Obs_rec(2,This_obs)))! RUN
END SELECT
SELECT INT(Obs_rec(3,This_obs))
CASE 0 TO 9
Temp$[5,6]="0"&VAL$(INT(Obs_rec(3,This_obs)))! POINT
CASE 10 TO 99
   Temp$[5,6]=VAL$(INT(Obs_rec(3,This_obs)))!POINT
END SELECT
SELECT I
CASE 1 TO 9
   Temp$[7,8]="0"&VAL$(I)
CASE 10 TO 99
   Temp$[7,8]=VAL$(I)
END SELECT
Filename$(I)=Temp$
Mivar(14)=I
CALL Routine(21,232) ! DISPLAY THE FILENAME
PRINT "FOR OBS # ",Recnum "; THE FILENAME IS: ";Filename$(I)
NEXT I
GOSUB Compute_vars
GOSUB Log_vars
Ending_obs=Num_obs_rec
ELSE
   CALL Pline(0," LOG FILE FULL ")
   WAIT 1
   CALL Pline(0,"")
END IF
CALL Pline(0,"OBSERVATION LOGGING COMPLETE ")
WAIT 1
GOTO End_4108
Enable_link: ! SEND A PACKET TO THE OTHER CPU WHICH CONTAINS THE "GO" WORD
Go_word$="GO"
OUTPUT Pkt_sc;Go_word$
RETURN
Disable_link: ! SEND A PACKET TO THE OTHER CPU WHICH CONTAINS THE "STOP" WORD
Go_word$="STOP"
OUTPUT Pkt_sc;Go_word$
RETURN
Get_packet: !
ENTER Pkt_sc;Pkt$(*)
! PRINTER IS PRT
CALL Pline(0,"RECEIVED PACKET FOR SAMPLE ",&VAL$(I)&")
FOR Vars_ctr=1 TO RCVD_vars
PRINT Pkt$(Vars_ctr+1)
NEXT Vars_ctr
PRINTER IS CRT
RETURN
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!...
*Log(U)*

25755 \( A_{\text{rec}}(15, \text{This}_{\text{obs}}) = A_{\text{rec}}(10, \text{This}_{\text{obs}}) \times A_{\text{rec}}(11, \text{This}_{\text{obs}}) \) !Log(U)

*Log(Rho)*

25756 \( A_{\text{rec}}(16, \text{This}_{\text{obs}}) = A_{\text{rec}}(10, \text{This}_{\text{obs}}) \times A_{\text{rec}}(12, \text{This}_{\text{obs}}) \) !Log(U)

*Log(TO)*

25757 \( A_{\text{rec}}(17, \text{This}_{\text{obs}}) = A_{\text{rec}}(15, \text{This}_{\text{obs}}) \times A_{\text{rec}}(12, \text{This}_{\text{obs}}) \) !Log(U)

*Log(Rho)\*Log(TO)*

25758 FOR Wire=1 TO 4

25759 \( A_{\text{rec}}(17+\text{Wire}, \text{This}_{\text{obs}}) = \text{Obs}_{\text{rec}}(19+\text{Wire}, \text{This}_{\text{obs}}) \)

25760 IF \( A_{\text{rec}}(17+\text{Wire}, \text{This}_{\text{obs}}) > 0 \) THEN

25761 \( A_{\text{rec}}(21+\text{Wire}, \text{This}_{\text{obs}}) = \text{LGT}(A_{\text{rec}}(17+\text{Wire}, \text{This}_{\text{obs}})) \) !Log(E)

25762 END IF

25763 SELECT Obs_{\text{rec}}(36, \text{This}_{\text{obs}})

25764 CASE 1 TO 16

25765 \( A_{\text{rec}}(41+\text{Wire}, \text{This}_{\text{obs}}) = \text{Gain}_{\text{code}}_{14}(\text{Obs}_{\text{rec}}(36, \text{This}_{\text{obs}}) + 1) \)

25766 CASE ELSE

25767 \( A_{\text{rec}}(41+\text{Wire}, \text{This}_{\text{obs}}) = 0 \) !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 \( B_{\text{rec}}(1, \text{This}_{\text{obs}}) = \text{Obs}_{\text{rec}}(1, \text{This}_{\text{obs}}) !\text{TEST} \)

25775 \( B_{\text{rec}}(2, \text{This}_{\text{obs}}) = \text{Obs}_{\text{rec}}(2, \text{This}_{\text{obs}}) !\text{RUN} \)

25776 \( B_{\text{rec}}(3, \text{This}_{\text{obs}}) = \text{Obs}_{\text{rec}}(3, \text{This}_{\text{obs}}) !\text{POINT} \)

25777 \( B_{\text{rec}}(4, \text{This}_{\text{obs}}) = \text{Obs}_{\text{rec}}(14, \text{This}_{\text{obs}}) !\text{P\_TOTAL} (\text{local}) \)

25778 \( B_{\text{rec}}(5, \text{This}_{\text{obs}}) = \text{Obs}_{\text{rec}}(15, \text{This}_{\text{obs}}) !\text{P\_STATIC} (\text{local}) \)

25779 \( B_{\text{rec}}(6, \text{This}_{\text{obs}}) = \text{Obs}_{\text{rec}}(8, \text{This}_{\text{obs}}) + 460 !\text{T\_TOTAL} (\text{local}) !\)

Use TUNNEL total

25780 !

25781 Strut$="FLOOR"

25782 GOSUB Compute_u_rho_m!Compute velocity,density,mass flow

25783 \( B_{\text{rec}}(7, \text{This}_{\text{obs}}) = \text{Local}_{\text{velocity}} \)

25784 \( B_{\text{rec}}(8, \text{This}_{\text{obs}}) = \text{Local}_{\text{density}} \)

25785 \( B_{\text{rec}}(9, \text{This}_{\text{obs}}) = \text{T}_s \)

25786 IF \( B_{\text{rec}}(7, \text{This}_{\text{obs}}) > 0 \) THEN

25787 \( B_{\text{rec}}(10, \text{This}_{\text{obs}}) = \text{LGT}(B_{\text{rec}}(7, \text{This}_{\text{obs}})) \) !Log(U)

25788 END IF

25789 IF \( B_{\text{rec}}(8, \text{This}_{\text{obs}}) > 0 \) THEN

25790 \( B_{\text{rec}}(11, \text{This}_{\text{obs}}) = \text{LGT}(B_{\text{rec}}(8, \text{This}_{\text{obs}})) \) !Log(Rho)

25791 END IF

25792 IF \( B_{\text{rec}}(6, \text{This}_{\text{obs}}) > 0 \) THEN

25793 \( B_{\text{rec}}(12, \text{This}_{\text{obs}}) = \text{LGT}(B_{\text{rec}}(6, \text{This}_{\text{obs}})) \) !Log(TO)

25794 END IF

25795 IF \( \text{Local}_{\text{mass\_flow}} > 0 \) THEN

25796 \( B_{\text{rec}}(13, \text{This}_{\text{obs}}) = \text{LGT}(\text{Local}_{\text{mass\_flow}}) \) !Log(Rho_{\text{inf}}*U)

25797 END IF

25798 \( B_{\text{rec}}(14, \text{This}_{\text{obs}}) = B_{\text{rec}}(11, \text{This}_{\text{obs}}) \times B_{\text{rec}}(12, \text{This}_{\text{obs}}) \) !Log(Rho)

25799 \( B_{\text{rec}}(15, \text{This}_{\text{obs}}) = B_{\text{rec}}(10, \text{This}_{\text{obs}}) \times B_{\text{rec}}(11, \text{This}_{\text{obs}}) \) !Log(U)

25800 \( B_{\text{rec}}(16, \text{This}_{\text{obs}}) = B_{\text{rec}}(10, \text{This}_{\text{obs}}) \times B_{\text{rec}}(12, \text{This}_{\text{obs}}) \) !Log(U)

25801 \( B_{\text{rec}}(17, \text{This}_{\text{obs}}) = B_{\text{rec}}(15, \text{This}_{\text{obs}}) \times B_{\text{rec}}(12, \text{This}_{\text{obs}}) \) !Log(U)

25802 FOR Wire=1 TO 3

25803 \( B_{\text{rec}}(17+\text{Wire}, \text{This}_{\text{obs}}) = \text{Obs}_{\text{rec}}(23+\text{Wire}, \text{This}_{\text{obs}}) \)
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)=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(TO)
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(TO)
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(TO)
C_rec(17,This_obs)=C_rec(15,This_obs)*C_rec(12,This_obs)!Log(U)*Log(Rho)*Log(TO)
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
CASE "TUNNEL"
    Pt-Obs_rec(7, This_obs)
    Ps-Obs_rec(6, This_obs)
    Tt-Obs_rec(8, This_obs)+460
CASE "WALL"
    Pt-A_rec(4, This_obs)
    Ps-A_rec(5, This_obs)
    Tt-A_rec(6, This_obs)
CASE "FLOOR"
    Pt-B_rec(4, This_obs)
    Ps-B_rec(5, This_obs)
    Tt-B_rec(6, This_obs)
CASE "OTHER"
    Pt-C_rec(4, This_obs)
    Ps-C_rec(5, This_obs)
    Tt-C_rec(6, This_obs)
END SELECT

IF Ps=0 THEN
    Prat=0
    GOTO 25883
END IF

P_rat=Ps/Pt
IF P_rat>1 THEN P_rat=1
IF P_rat<0 THEN
    Ts=MAX(.1, Tt)
    Local_velocity=0.
    Local_density=Ps/(53.3*Ts)
    Local_mass_flow=0.
ELSE
    Ts=(Tt)/(P_rat^(-.285714285))
    IF P_rat=1 THEN
        Local_velocity=0.
    ELSE
        Sound=SQR(2402.764*Ts)
        Local_mach=SQR(5*(P_rat^(-.285714285)-1))
        Local_velocity=Local_mach*Sound! U
    END IF
    Local_density=Ps/(53.3*Ts) ! Rho
    Local_mass_flow=Local_velocity*Local_density! M
END IF

RETURN

Log_vars: ! LOG THE CURRENT OBSERVED VARIABLES

ASSIGN @Disk TO Log_fn$

OUTPUT @Disk,1;Data_set_title$,Max_obs_rec,Max_vars,Logged_var_name$(*) ,Numsubfile,Subfile_names$(*) ,Subfile_chartst(*)
ENTER @Disk,2
OUTPUT @Disk;Obs_rec(*)(
ASSIGN @Disk TO *
ASSIGN @Diska TO Log_fn$&"A"
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 RETURN  
25936  
25937  
25938  
25939  
25940 END_4108:  
25941 END SELECT  
25942 CASE 4110 "SAMPLES TO AVG"  
25943 SELECT Routin1  
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 Routin1  
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 Routin1  
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
CASE 4130  ! "INITIAL PROBE"
SELECT Routinl
CASE 21
   Mvar(2)-Initial_probe
   Mvar(4)=1
   Mvar(5)=1.
   Mvar(6)=2.
   Mivar(9)=0
CASE 22
   Initial_probe=Mvar(2)
END SELECT
CASE 4131  ! "ENDING PROBE"
SELECT Routinl
CASE 21
   Mvar(2)-Ending_probe
   Mvar(4)=1
   Mvar(5)=1.
   Mvar(6)=2.
   Mivar(9)=0
CASE 22
   Ending_probe=Mvar(2)
END SELECT

CASE 4127  ! "LOGFILE TO PC"
SELECT Routinl
CASE 31
   Pc_device=8   ! Send data out thru HP-IB bus 8
   bus 8 is NOT system controller
FOR Probe-Initial_probe TO Ending_probe
SELECT Probe
CASE 1
   OUTPUT Pc_device;VAL$(Ending_obs-Initial_obs+1),VAL$(Max_vars)
   FOR I=1 TO Max_vars
      DISP "NAME("&VAL$(I)&") IS: ",&A_var_name$(I)
      OUTPUT Pc_device;A_var_name$(I)[1,10]
   NEXT I
   FOR This_obs-Initial_obs TO Ending_obs
      FOR I=1 TO Max_vars
         OUTPUT Pc_device;A_rec(I,This_obs)
      NEXT I
   NEXT This_obs
CASE 2
   OUTPUT Pc_device;VAL$(Ending_obs-Initial_obs+1),VAL$(Max_vars)
   FOR I=1 TO Max_vars
      DISP "NAME("&VAL$(I)&") IS: ",&B_var_name$(I)
      OUTPUT Pc_device;B_var_name$(I)[1,10]
   NEXT I
   FOR This_obs-Initial_obs TO Ending_obs
      FOR I=1 TO Max_vars
         OUTPUT Pc_device;B_rec(I,This_obs)
      NEXT I
   NEXT This_obs
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 quantities to the mean quantities:

\[ \frac{u'}{U}, \frac{p'}{P}, \frac{t'}{T} \]

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: \[ \frac{u'}{U}, \frac{p'}{P}, \text{and} \frac{t'}{T} \].

The solutions are placed in traces (memories) 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.
This function catalogs all selected files. The selection criteria is defined in detail in the HP BASIC language reference manual for HP function CAT (SELECT).

[ CAT GROUP ]-----------------------------
|             |             |
|             |             |
| ~-[- ]-~[ name ]--|

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string expression</td>
<td>any valid characters that are allowed in a file name</td>
</tr>
</tbody>
</table>

* 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 from gain codes. The fluctuating data is gained to provide adequate voltage for filtering and 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 PROBE 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 the probe 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 relevant to the hotwire coefficient 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. 
This function computes hotwire coefficients for the probe previously selected - by function INITIAL PROBE - utilizing a custom multiple linear regression routine 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 sensitivities necessary to compute velocity, density and temperature turbulence ratios - by function CALC VAL 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.
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 \(R(\text{RhoU})\), \(R(\text{UTO})\), and \(R(\text{RhoT0})\).

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 \(P'/P\).

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.
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 sensitivities 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.
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.

NOTE: This function is highly interactive, and is not recommended 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 are 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 automatically enters the EDIT COEFS function.

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

NOTE: This function is highly interactive, and is not recommended 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.

[* 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
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.

* 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 UTILITIES
Function Number: 4020
Module: MODUSR1*

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

* This routine - MODUSR1 - was written by Steven J. Clukey of Vigyan Research Associates, Inc. for NASA LaRC TAD/FDB under contract NAS1-17919, Task 35. This work was done beginning in October, 1986 and continues thru April, 1988.
Function Name: FROM DISK
Function Number: 4026
Module: MODUSRI

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

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>MSI</td>
<td>any valid HP storage device</td>
</tr>
</tbody>
</table>

*. This routine - MODUSRI - 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.

* 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.

* 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]-- |

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>numeric integer</td>
<td>1 to 300, the max number of observations</td>
</tr>
</tbody>
</table>

* 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 beginning 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]--|

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>numeric integer</td>
<td>1 to 3, the max number of probes</td>
</tr>
</tbody>
</table>

* 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.
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.

NOTE:

This function must be performed before 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.

NOTE: This function must be performed before 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.

* 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 compatibility 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.
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.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string expression</td>
<td>any valid characters that are allowed in a file name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>see FILE GROUP for details</td>
</tr>
</tbody>
</table>

*. 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 NASl-17919, Task 36. This work was done beginning in October, 1986 and continues thru April, 1988.
This function generates x-y plots on the Plotter which represent the Log(\(\rho U\)) vs. Log(E) for each hotwire. The axes, titles, etc are internally generated using a plot file called "\(\rho U\)" - see function PLOT NAME. (It should be noted that this '\(\rho U\) 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.

* 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 UTILITIES
Function Number: 4001
Module: MODUSR1*

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

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

[ PURGE ]----------|
**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 ]--|
```

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string expression</td>
<td>any valid characters that are allowed in a file name</td>
</tr>
</tbody>
</table>

* 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 reference manual for HP function PURGE (SELECT).

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string expression</td>
<td>any valid characters</td>
</tr>
<tr>
<td></td>
<td></td>
<td>that are allowed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>in a file name</td>
</tr>
</tbody>
</table>

* 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.
Remake Probe

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 ]-----------------|
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.

* 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.
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.

* 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: MODUSRI

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

Item | Description | Range
--- | --- | ---
value | MSI | any valid HP
 | | storage
 | | device

* This routine - MODUSRI - 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.

* 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
APPENDIX C.1. Initialization Sequence Program - "8FTINI"
PLOT EJLT
INITIAL OBS=49
PLOT LOG_E
PRINT LOGFILE
SEQUENCE MENU
LET I=0
PRINT "PRESS ENTER WHEN READY TO LOG DATA"
WAIT 3
ARM
LOG DATA POINT
PRINT LOGFILE
PRINT "POINT IN PROGRESS"
PRINT LOG_E
IF I=0 THEN TAG PLOT
I=1
WAIT *
PRINT "TRANSFERRING DATA TO MEMORY"
TF FROM REC
PRINT "RECORDING DATA"
STORE DATA
PRINT "POINT COMPLETE"
GOTO 1C

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 ,&HFEOO
130 I-&HFEOO
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$ = ENVIRONS("PCIB")
260 I$ = PCIB.DIR$ + "\PCIBILC.BLD"
270 BLOAD I$,&HFEOO
280 CALL I(PCIB.DIR$, 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$,PCIB.GUERR)
540 PCIB.BASERR = 255
550 ON ERROR GOTO 870
560
570 J= -1
580 I$=PCIB.DIR$+"\HPIB.SYN"
CALL O.S(I$)
IF PCIB.ERR<>0 THEN ERROR PCIB.BASERR

' Determine entry points for HP-IB Library routines

I = 0
CALL I.V(I, IOABORT, IOCLEAR, IOCONTROL, IOENTER)
IF PCIB.ERR<>0 THEN ERROR PCIB.BASERR
CALL I.V(I, IOENTERA, IOENTERS, IOEOI, IOEOL)
IF PCIB.ERR<>0 THEN ERROR PCIB.BASERR
CALL I.V(I, IOGETTERM, IOLLOCKOUT, ILOLOCAL, IOMATCH)
IF PCIB.ERR<>0 THEN ERROR PCIB.BASERR
CALL I.V(I, IOOUTPUT, IOOUTPUTA, IOOUTPUTS, IOPPOLL)
IF PCIB.ERR<>0 THEN ERROR PCIB.BASERR
CALL I.V(I, IOPPOLLC, IOPPOLLU, IORLYOTE, IORESET)
IF PCIB.ERR<>0 THEN ERROR PCIB.BASERR
CALL I.V(I, IOTRIGGER, J, J, J)
IF PCIB.ERR<>0 THEN ERROR PCIB.BASERR
CALL C.S
I$ = PCIB.DIR$ + "\HPIB.PLD"
CALL L.P(I$)
IF PCIB.ERR<>0 THEN ERROR PCIB.BASERR
GOTO 1000

Error handling routine

IF ERR=PCIB.BASERR THEN GOTO 900
PRINT "BASIC error ";ERR;" occurred in line ";ERL
STOP
TMPERR = PCIB.ERR
IF TMPERR = 0 THEN TMPERR = PCIB.GLBERR
PRINT "PC Instrument error ";TMPERR;" detected at line ";ERL
PRINT "Error: ";PCIB.ERRS
STOP

COMMON declarations are needed if your program is going to chain
to other programs. When chaining, be sure to call DEF.ERR as
well upon entering the chained-to program

COMMON PCIB.DIR$, PCIB.SEG
COMMON LD.FILE, GET.MEM, PANELS, DEF.ERR
COMMON PCIB.BASERR, PCIB.ERR$, PCIB.ERRS, PCIB.NAME$, PCIB.GLBERR
COMMON IOABORT, IOCLEAR, IOCONTROL, IOENTER, IOENTERA, IOENTERS, IOEOI, IOEOL, IOGETTERM, IOLLOCKOUT, ILOLOCAL, IOMATCH, IOOUTPUT, IOOUTPUTA, IOOUTPUTS, IOPPOLL, IOPPOLLC, IOPPOLLU

FALSE = 0
TRUE = NOT FALSE
NOERR = 0
EUNKNOWN = 100001!
ESEL = 100002!
ERANGE = 100003!
ETIME = 100004!
ECTRL = 100005!
EPASS = 100006!
ENUM = 100007!
EADDR = 100008!
COMMON FALSE, TRUE, NOERR, EUNKNOWN, ESEL, ERANGE, ETIME, ECTRL, EPASS, EN
END PROGRAM

User program can begin anywhere past this point

Program for a system to receive data from the Dynamic Data Acquisition System.

Set up HP-IB addressing and initialize system

OPTION BASE 1

MAX.VARIABLES= 50

DIM NAMES$ (MAX.VARIABLES)

DIM X(3)

ACT.VARIABLES= 0

NAMES$ = SPACE$(50)

Set up HP-IB addressing and initialize system

ISC-7

DEV=1

DEV = ISC * 100 + DEV

CALL IORESET (ISC)

IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR

TIMEOUT = 5

CALL IOTIMEOUT (ISC, TIMEOUT)

IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR

CALL IOCLEAR (ISC)

IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR

CALL IOEOI (ISC, FALSE)

IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR

PRINT "DO YOU WISH TO RECEIVE A FILE FROM THE HP COMPUTER? (Y or N)"

INPUT ANS$

IF ANS$="N" THEN GOTO 1730

IF ANS$<>"Y" THEN GOTO 1491

PRINT "ENTER THE NAME OF THE FILE TO RECEIVE TRANSFERED DATA: "

INPUT FILS.

FOR J=1 TO 2

FOR J=1 TO NOBS

OPEN FILS FOR OUTPUT AS #1

FOR J=1 TO 2

MM=6 : AA=0

T$$(J)=SPACE$(MM)

CALL IOENTERS (DEV,T$(J),MM,AA)

IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR

N(J)=VAL(LEFT$(T$(J),AA-2))

PRINT N(J)

NEXT J

NOBS=N(1)+1 : NVARS=N(2)

FOR J=1 TO NOBS

FOR I=1 TO NVARS

M=20 : A=0

TEMP$=SPACE$(M)

CALL IOENTERS (DEV,TEMP$,M,A)

IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR

NAMES$(I)=LEFT$(TEMP$,A-2)

PRINT "Record ",J", item ".(I),"= " NAMES$(I)

IF J=1 THEN PRINT #1,NAMES$(I) ELSE PRINT #1,VAL(NAMES$(I))

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

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.