Data Logger for the
34-Meter Vertical Axis
Wind Turbine Test Bed

Mark E. Ralph

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and
Livermore, California 94550
for the United States Department of Energy
under Contract DE-AC04-76DP00789
Contents

Introduction .................................................. 1
Purpose ......................................................... 1
Requirements .................................................. 2
Hardware ......................................................... 3
Software ......................................................... 4
Summary ......................................................... 8
References ....................................................... 8
Appendix A Data Logger Displays and Plots ................. A-1
Appendix B Data Logger Operator Notes ..................... B-1
Appendix C Wind Direction Averaging ....................... C-1
Appendix D FORTRAN Source Code ......................... D-1
Appendix E System Generation Answer File and Welcome File ................................................... E-1

Figures and Tables

Analog to Digital Converter Specifications .................. 3
Data Logger Flow Chart ....................................... 5
File Header Structure ........................................ 7
Introduction

Sandia National Laboratories, as the lead laboratory for vertical axis wind turbine (VAWT) technology development in the United States, has built a 34-m-diameter research-oriented VAWT, called the Test Bed, at Bushland, Texas. The acquisition and reduction of various types of data are obviously very important for such a machine. Based on our experiences on previous machines, we decided to use a mini-computer-controlled data acquisition and analysis system (DAAS) to obtain detailed short-term turbine performance and response data (Akins 1978, Akins et al. 1987, Stiefeld 1978). The DAAS is described in Berg et al. 1988. However, we also decided to develop a minicomputer-controlled data logger to obtain long-term data to characterize the wind at the site of the turbine, record performance data of the control system, obtain a continuous record of events at the test site, consolidate displays for the test engineer and provide a display of current information for visitors to the site.

The data logger is implemented on a Hewlett Packard 11P 1000 A600 mini-computer. Software programs collect data from 35 channels, display the collected data, and record the data on a hard disk. The system has been in operation at the site since March of 1986.

This report discusses the purpose and the requirements that were established for the data logger. Both the hardware and software that make up the data logger are also described, and operator instructions, operating system commands, and procedure files are appended to it.

Purpose

At the test site, the data logger collects data to characterize the wind, validates long-term control system performance, provides a continuous record of events, displays pertinent real-time data for the test engineer, and displays data for visitors to the site.

The energy input to a wind turbine is defined by the speed and density of the air entering the turbine's swept area. The density of the air is a function of the barometric pressure and air temperature. To characterize the wind at the Test Bed site, the data logger monitors nine wind-speed and seven wind-direction sensors, and two ambient-temperature sensors. Most of the atmospheric instrumentation is located on an existing 48-m meteorological tower located 150 meters from the turbine. The meteorological tower supports wind speed and direction sensors at 10 meters, 20 meters, 30 meters, 40 meters and 48 meters ambient temperature sensors at 10 meters and 48 meters. The five stations were selected to provide data on the wind shear at the site. Four additional wind speed and direction sensors are located on anemometer towers at turbine equator height (30 meters) two diameters upwind from the turbine in the two most predominant wind directions. Data from these anemometer towers will provide wind speed data that is highly correlated with the actual wind that the turbine is seeing. A barometric pressure sensor is also located at the site.

The data logger will record wind speed, turbine rpm, and power produced. The average wind speed and the corresponding maximum energy available can then be calculated from this stored data. The performance of the turbine control system can then be evaluated by comparing the actual energy produced with the available energy over a given time period.
The data logger records data from the site and turbine instrumentation whenever the instrumentation is operating. Daily reports are printed showing the average wind speed and direction for each sensor. Unreasonable discrepancies between the instruments can indicate the degradation of an instrument's accuracy. This record allows the test engineer to make informed decisions on when an instrument needs to be recalibrated or repaired.

Previous Sandia test installations have used various displays to provide the test engineer with real-time information about test conditions. Each of these displays had to be calibrated independent of the calibration of the data being recorded by the DAAS. Because of this, the information was not presented in a clear and concise manner and extra time was required to calibrate the displays. The data logger provides a real-time display of the wind and turbine's status. The display is updated every three seconds. A typical display screen is shown in Appendix A. The data logger's display can be easily modified to reflect changing test requirements. New instruments can be added to the display as test requirements require new measurements. Other instruments can be deleted when they are no longer needed. Since the data logger is calibrated in the same manner as the DAAS, calibration time has been reduced. The calibration of the DAAS and data logger only require changing values stored in a calibration table on the computer.

A menu-driven display program is available for researchers and visitors to the site. This display program lists current average ambient and turbine data in System International and British Engineering units. This display is updated every 10 seconds. Typical display screens are shown in Appendix A. Also available are plots of the same data showing the performance and conditions over the last 3-minutes, 18-minutes, 1-hour, 24-hour, and one-week period. The researcher can select one to three channels to be displayed from menus. Typical plots available from the data logger are also shown in Appendix A.

Requirements

The data logger's primary requirement is to sample all 35 channels every 500 milliseconds. All of these channels are displayed on the system and display consoles. Twenty-five are stored on the hard disk. Since the data logger operates continuously except during routine maintenance, the amount of data collected would be impractical to store without reducing the data first. One-minute time intervals were selected as being a reasonable compromise for data storage. The one-minute time intervals provide sufficient resolution for the data analysis that is being undertaken.

The 25 channels that are logged are averaged, and the standard deviation is calculated over a one-minute time period. The collected one-minute averages and standard deviations are stored on a daily basis as integers. A calibration table for converting the data to engineering units is appended as a header to each days’ record. Results stored on disk as integer values reduce the amount of storage space by almost half with respect to storing floating point values. This storage format was compatible with the existing data reduction programs utilized to analyze the data collected by the DAAS.

It is necessary to sample the data as quickly as possible in order to obtain an accurate average value and standard deviation. The sample rate also has to allow time for the data logger to complete the other tasks that it has to accomplish. A sample rate of reading all 25 channels every 500 milliseconds (2 Hz) has proven adequate to obtain the resolution required and leave sufficient time to compute averages and standard deviations, to store and display data.
Hardware

The data logger hardware consists of a mini computer with a hard disk, 40-channel analog-to-digital converter (ADC), two monochrome terminals, and two printers.

The computer is a Hewlett Packard (HP) 1000-series A600 mini-computer with 1024 kilobytes (512 kilowords) of random access memory. There are three input/output boards in the computer: an eight-channel RS232 communication port multiplexer for connecting the system console and display terminal to the computer, and two HPIB (IEEE 488) cards for communicating with the hard disk and the system printer. The computer also contains the analog-to-digital converter and a 25-kHz power supply that is used by the ADC. The A600 was chosen as the least expensive computer that could perform the required task and easily maintain compatibility with the existing DAAS.

The storage device is a HP 7942A 24 megabyte disk drive with a 1/4-inch cartridge tape. The disk drive is divided into three logical units (LU). The first LU is utilized for the operating system as scratch disk space. The second LU contains the operating system and application programs. The third LU is utilized to store the data collected by the data logger. Once a month the data stored on the hard disk are transferred to streaming tape for archiving the data and for transferring the data to other computers for data analysis. The backed-up data are then deleted from the disk drive to provide room for another month's worth of data.

An HP 12060B eight-channel analog differential input card and an HP 12061A 32-channel analog differential input expansion card are used to digitize the analog data. These cards provide up to 40 channels of data with 12-bit resolution. The gain of the analog input card can be set for signals with the following ranges and specifications:

<table>
<thead>
<tr>
<th>ADC GAIN</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>FULL SCALE RANGE</td>
<td>+/-.10.24V</td>
<td>+/-.5.12V</td>
<td>+/-.2.56V</td>
<td>+/-.1.28V</td>
</tr>
<tr>
<td>RESOLUTION</td>
<td>5mV</td>
<td>2.5mV</td>
<td>1.25mV</td>
<td>0.625mV</td>
</tr>
<tr>
<td>TEMPERATURE COEF</td>
<td>+/-0.38mV/C</td>
<td>+/-0.38mV/C</td>
<td>+/-0.095mV/C</td>
<td>+/-0.048mV/C</td>
</tr>
</tbody>
</table>

All of the input channels have been set to a gain of 2, which accepts analog inputs from -5.12 to +5.12 volts. A 5.12-V input signal is represented in the computer as an integer of 32000, and a -5.12-V input is represented as -32000. Each 2.5-mV change in an analog input signal will correspond to a change of 1 in the integer value represented in the computer. The input signal will drift less than 0.38 mV for any temperature change of 1 degree Celsius in the ADC.

The system console and display terminal are both HP 150 terminals. The terminals are monochrome graphics terminals with screen resolution of 390 X 512 pixels, allowing 500 data points to be plotted.
Software

The HP 1000 mini-computer is controlled by the Real Time Executive (RTE-A) operating system. RTE-A is a real-time operating system, allowing for software programs to be scheduled to run at specified times and with the ability to give programs differing priorities. All of the programs were written in FORTRAN 77, using HP 1000 extensions for scheduling programs and handling input/output operations. The HP 1000 Device-independent Graphics Library (DGL) is used for plotting the data on the display screen (Hewlett-Packard 1984).

The data logger software controls the computer in performing the following tasks: collecting data from the A/D card, reducing the collected data by averaging the points over time, storing the reduced data, and displaying the data on the two terminals. The real-time nature of the operating system allows for scheduling various programs to run at specific time intervals as well as at specific times of day. The program GETDA is scheduled to run every 500 milliseconds. This program collects data from the analog to digital converter (ADC) and reduces the data into averages and standard deviations. To store the data, the program DAILY runs at 2400 hours every night. To prevent storing from interrupting the collection of data, the GETDA program is run at a higher priority than the DAILY program. Two programs, MENU and DISPL, display data for the test engineer and visitors to the site. These two display programs run at even lower priorities than the GETDA and DAILY program. Other programs on the data logger include INITIAL, which sets up the data arrays and schedules the programs GETDA, DAILY, and DISPL; program PLOT, which retrieves and plots previously stored data; and program ADCAL, which is used to update calibration factors for the analog inputs.

The programs INITIAL, GETDA, DAILY, DISPL, and MENU communicate with each other by utilizing system common for program and data parameters and sharable extended memory area (EMA) for large data arrays. EMA allows data arrays larger than 512 words (1/2 kiloword) to be used. The data logger requires more than 200 kilowords of memory for storing the reduced data. Since the EMA arrays are sharable between programs, the various programs can all access the same data arrays. Figure 1 shows graphically how the programs interact with the data in memory. Following are more detailed descriptions of the programs.

Program INITIAL is scheduled to run by the system whenever the computer is booted, or may be run by the test engineer to re-initialize the data logger. This program sets the Sharable EMA arrays used by the program DISPL to zero. Next, the program reads the system's time and date and checks to see if data have already been stored on the hard disk for the present day. If data for the present day already exist, the data are recovered, or if none exist, the data arrays for the one-minute averages and standard deviations are set to zero. The program then reads the calibration table, which is stored on the hard disk.

The calibration table is used to convert the voltages read by the ADC into engineering units. Next, the program GETDA is scheduled to start in one to two minutes, depending on the present time, and to run every 500 milliseconds thereafter. Program DAILY is then scheduled to run at 2400 hours, and program DISPL is scheduled to run immediately.
Figure 1

DATA LOGGER FLOW CHART

- System Common
  - Indexes Status
  - Program Display program
  - Terminal LU 31 keyboard crt printer

- Analog Digit Converter
  - PROGRAM INITIAL startup program

- PROGRAM GETDA reads A/DC
  - PROGRAM DAILY stores data

- System Printer LU 6

- Calibration Table Disk File
  - Calibration ADCAL calibration program
  - Utility TFT tape transfer program

- Daily Data Disk Files
  - Daily Data Disk Files
  - Program PLOT output program

- Shareable EMA
  - Scales & Zeros
    - 0.5 sec Averages
    - 3 sec Averages
    - 10 sec Averages
    - 4 min Averages
    - 28 min Averages
    - 1 min Averages
    - 1 min Standard Deviations

- Terminal LU 1 keyboard crt printer
Program GETDA reads the last 35 channels (channels 6 thru 40 of the ADC) from the ADC and computes 3-second, 10-second, 1-minute, 4-minute, and 28-minute averages and 1-minute standard deviations. The program then checks to see if the store-data status bit has been set by the operator. If the bit is set, the one-minute averages and standard deviations are stored on the hard disk, and a message is printed on the system console telling the operator to abort the programs. If the bit is not set, the program is made dormant, saving computer resources until it is scheduled to run again in 500 milliseconds.

Program DAILY stores the one-minute averages and standard deviations in a temporary array. The one-minute arrays are then set equal to zero for the next day of data. The program then updates the calibration table values by reading the table stored on the hard disk. The file header is assembled and the header and one-minute averages and standard deviations are written to the hard disk. The file header includes the calibration data and units for each channel. The format for the file header is shown in Table 2. A daily report is then assembled and printed on the system printer. The program then reschedules itself to run at 2400 hours the next day.

Program DISPL is a menu-driven program that displays the current data (10-second average) in System International or British Engineering units. The last three minutes (data points sampled every 500 milliseconds), 18 minutes (3-second averages), 1 hour (10-second averages), 24 hours (4-minute averages) or week (28-minute averages) of data can be plotted on the screen. The data and plots can be copied to the printer connected to the terminal.

Program MENU is a menu-driven program that allows the test engineer to control the other programs and to display the current data (3-second average) on the system console. During instrument calibration, the test engineer can instruct the GETDA program to set the one-minute average data to zero so that erroneous data are not stored. The test engineer can also instruct the GETDA program to save the current data and suspend operation if the computer needs to be shut down.

Program ADCAL is a menu-driven program that is used to create and update the calibration table stored on the hard disk. This program was written for the DAAS. To keep the data logger compatible with new versions of ADCAL that may be written for the DAAS, some features of ADCAL that are not utilized on the data logger are ignored.

Program PLOT is used to retrieve and plot stored data. The program prompts the operator for the date of the file that is to be read. The file is then read and the plotting options are then displayed in menus. Up to three channels can be plotted on one plot with each covering 8-, 16-, or the entire 24-hour period of the file.

The storage of data is an important task for the software. The data are stored on the hard disk on a daily basis. First a file name is created of the form BUDLOGJAN0188 with a file extension of DAT. The BU represents the site code for Bushland, Texas. The DLOG indicates that the file is a data logger file and the JAN0188 is the date. A file of the appropriate size is then created on the hard disk with the given name. Next, the file header is assembled. The file header contains integers and real numbers. Rather than creating two headers, a real header and an integer header, a combined real and integer header is created. Since real numbers require the same storage space as two integers, an integer array of 512 words and a real array of 256 words require the same storage area on the computer.
<table>
<thead>
<tr>
<th>Integer Address</th>
<th>Real Address</th>
<th>Integer Address</th>
<th>Real Address</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>357</td>
<td>179</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>358</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>385</td>
<td>193</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>386</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>387</td>
<td>194</td>
</tr>
<tr>
<td>97</td>
<td>49</td>
<td>388</td>
<td></td>
</tr>
<tr>
<td>98</td>
<td>50</td>
<td>389</td>
<td>195</td>
</tr>
<tr>
<td>100</td>
<td>51</td>
<td>405</td>
<td>203</td>
</tr>
<tr>
<td></td>
<td></td>
<td>406</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>407</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>408</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>409</td>
<td></td>
</tr>
<tr>
<td>129</td>
<td>65</td>
<td>410</td>
<td></td>
</tr>
<tr>
<td>130</td>
<td>66</td>
<td>411</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>412</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>413</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>414</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>415</td>
<td></td>
</tr>
<tr>
<td>225</td>
<td>113</td>
<td>416</td>
<td></td>
</tr>
<tr>
<td>226</td>
<td>114</td>
<td>417</td>
<td></td>
</tr>
<tr>
<td>227</td>
<td>115</td>
<td>418</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>419</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>420</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>421</td>
<td></td>
</tr>
<tr>
<td>257</td>
<td>129</td>
<td>422</td>
<td></td>
</tr>
<tr>
<td>258</td>
<td></td>
<td>423</td>
<td></td>
</tr>
<tr>
<td>259</td>
<td>130</td>
<td>424</td>
<td></td>
</tr>
<tr>
<td>260</td>
<td></td>
<td>425</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>426</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>427</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>428</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>429</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>430</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>431</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>432</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>433</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>434</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>435</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>436</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>437</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>438</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>439</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>440</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>441</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>442</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>443</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>444</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>445</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>446</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>447</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>448</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>449</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>450</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>451</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>452</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>453</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>454</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>455</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>456</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>457</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>458</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>459</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>460</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>461</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>462</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>463</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>464</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>465</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>466</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>467</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>468</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>469</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>470</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>471</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>472</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>473</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>474</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>475</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>476</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>477</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>478</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>479</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>480</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>481</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>482</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>483</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>484</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>485</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>486</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>487</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>488</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>489</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>490</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>491</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>492</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>493</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>494</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>495</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>496</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>497</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>498</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>499</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>500</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>501</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>502</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>503</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>504</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>505</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>506</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>507</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>508</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>509</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>510</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>511</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>512</td>
<td></td>
</tr>
</tbody>
</table>

Table 2  FILE HEADER STRUCTURE

| Zero offsets for channel 1 stored as real number |
| Zero offsets for channel 2 stored as real number |
| Zero offsets for channel 49 stored as real number |
| Zero offsets for channel 50 stored as real number |
| Scale factor for channel 1 stored as real number |
| Scale factor for channel 2 stored as real number |
| Scale factor for channel 49 stored as real number |
| Scale factor for channel 50 stored as real number |
| Units for channel 1 stored as 2 integers |
| Units for channel 2 stored as 2 integers |
| Units for channel 49 stored as 2 integers |
| Units for channel 50 stored as 2 integers |

- Not used
- Length of record in sectors (integer)
- Number of channels (50 integer)
- 1st channel stored (1 integer)
- 2nd channel stored (2 integer)
- 3rd channel stored (3 integer)
- 24th channel stored (24 integer)
- 25th channel stored (25 integer)
- 26th channel stored (ch 1 std dev)
- 27th channel stored (ch 2 std dev)
- 49th channel stored (ch 24 std dev)
- 50th channel stored (ch 25 std dev)
- Sample rate (2 Hz)
- Length of record in seconds (86400 - real)
- Not Used
To simplify the handling of the file header, the integer and real arrays are declared as 512 integer words and 256 real words and then equivalenced. Values are then assigned to either the real or integer array as appropriate, and the integer array is written to the disk. The one minute averages and standard deviations are written to the hard disk as integer arrays. Table 2 shows the array structure for the header with both the integer and real indexes.

Summary

The data logger has been in operation for over three years with only minor hardware problems. Over this period, test requirements have changed as different phases of the Test Plan have been completed. The new test requirements have been easily implemented by minor modifications to the software programs. The data logger has met and is still meeting all of the requirements of the testing program.

References


## Appendix A

### DATA LOGGER DISPLAYS AND PLOTS

Typical data display screens:

**SNLA/USDA 34 METRE VAWT TEST BED DATA LOGGER**

<table>
<thead>
<tr>
<th>WIND</th>
<th>Speed</th>
<th>Direction</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 m</td>
<td>7.4 mps</td>
<td>258.5 deg W</td>
</tr>
<tr>
<td>30 m</td>
<td>7.5 mps</td>
<td>253.5 deg W</td>
</tr>
<tr>
<td>NE</td>
<td>8.4 mps</td>
<td>247.6 deg SW</td>
</tr>
<tr>
<td>NW</td>
<td>12.3 mps</td>
<td></td>
</tr>
<tr>
<td>SE</td>
<td>12.5 mps</td>
<td>185.6 deg S</td>
</tr>
<tr>
<td>SW</td>
<td>12.7 mps</td>
<td></td>
</tr>
</tbody>
</table>

**CONTROLLER**

- 4.18 Volts .75 Volts

- TranAxialVibra .17 in/s
- StandVibration .09 in/s
- strainGaugeAML -2.64 MPa
- strainGaugeAMf -29.16 MPa
- strainGaugeQML -17.71 MPa
- strainGaugeQHF -19.31 MPa

**COEF OF PERFORMANCE**

- Equator: Aero System
- 10 metre Aero System
- POWER
- Electrical 189.6 kW 185.4 kVAR
- Rotor 226.2 kW

**TURBINE**

- Speed 22.4 rpm
- Torque 97.5 kNm
- Tiedown Cable 574.1 kN

**AMBIENT CONDITIONS**

- Temperature 10 metre 299.3 °C
- Temperature 48 metre -2.5 °C
- Barometric Pressure 127.1 kPa

9:22 AM THU., 23 FEB., 1989

**PRESS ANY KEY TO RETURN TO THE MAIN MENU**

**SNLA/USDA 34 METRE VAWT TEST BED DATA LOGGER**

<table>
<thead>
<tr>
<th>WIND</th>
<th>Speed</th>
<th>Direction</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 m</td>
<td>13.2 mps</td>
<td>187.2 deg S</td>
</tr>
<tr>
<td>30 m</td>
<td>14.4 mps</td>
<td>191.1 deg S</td>
</tr>
<tr>
<td>NE</td>
<td>12.9 mps</td>
<td>183.5 deg S</td>
</tr>
<tr>
<td>NW</td>
<td>12.9 mps</td>
<td></td>
</tr>
<tr>
<td>SE</td>
<td>14.6 mps</td>
<td>189.0 deg S</td>
</tr>
<tr>
<td>SW</td>
<td>14.7 mps</td>
<td></td>
</tr>
</tbody>
</table>

**CONTROLLER**

- .40 Volts 1.91 Volts

- TranAxialVibra .21 in/s
- StandVibration .01 in/s
- strainGaugeAML -7.53 MPa
- strainGaugeAMf -41.60 MPa
- strainGaugeQML -7.51 MPa
- strainGaugeQHF -31.27 MPa

**COEF OF PERFORMANCE**

- Equator: Aero System
- 10 metre Aero System
- POWER
- Electrical 494.4 kW 511.8 kVAR
- Rotor 478.6 kW

**TURBINE**

- Speed 34.2 rpm
- Torque 133.5 kNm
- Tiedown Cable 868.0 kN

**AMBIENT CONDITIONS**

- Temperature 10 metre -2.5 °C
- Temperature 48 metre -2.9 °C
- Barometric Pressure 89.39 kPa

9:06 AM THU., 23 FEB., 1989

**PRESS ANY KEY TO RETURN TO THE MAIN MENU**
DATA LOGGER DISPLAYS AND PLOTS

Typical data plots of current data:
DATA LOGGER DISPLAYS AND PLOTS

Typical data plots of current data:
Appendix B

DATA LOGGER OPERATOR NOTES

System Prompts

VCP > Virtual Control Panel Indicates that the computer's operating system has not been loaded. The operating system must be loaded before any other action can be taken. Respond with "%BDC" <CR> to load the primary operating system (boot the computer).

CI > Command Interpreter Normal system prompt. Enter desired command or program.

CM > Command Interpreter Secondary system prompt. This prompt indicates that the primary Command Interpreter program is busy. Entering <CR> will allow the primary CI program to continue.

RTE > Real Time Executive Tertiary system prompt. This prompt indicates that the primary and secondary Command Interpreter programs are busy. Entering <CR> will allow the primary, CI and secondary, CM programs to continue.
DATA LOGGER OPERATOR NOTES

Useful Commands and Programs

**ADCAL**  
Program to display and update calibration factors including zero offsets, scale factors and engineering units.

**CLEAR31**  
Program to clear the screen of the display terminal (LU31) and turn off the function keys.

**INITIAL**  
Program to initiate the taking of data. This program schedules the other programs.

**IO**  
Command for listing the status of all Input/Output devices.

**LU**  
Logical Unit. The number assigned to I/O devices.

**MENU**  
Program to control the taking of data and display information for the Test Engineer.

**OF**  
Command for aborting the program running on the terminal.

**OFF**  
Program for aborting the other programs: GETDA, DISPL, and DAILY.

**PU,file**  
Command for purging a file.

**PU,BUDLOGJAN--86.DAT::DATA,OK**  
Command for purging Data Logger files. Use this command after transferring the files to streaming tape to make room on the hard disk for new files.

**TF**  
Utility program for transferring files between the hard disk and the streaming tape.

**TF,CO,BUDLOGJAN--86.DAT::DATA,24**  
Utility program for transferring Data Logger files from the hard disk to the streaming tape. This command will transfer all Data Logger files for January, 1986 to the streaming tape. Use this form for new tapes.

**TF,CO,BUDLOGJAN--86.DAT::DATA,24.A**  
Utility program for transferring Data Logger files from the hard disk to the streaming tape. This command is the same as the previous command except the files will be appended to the tape. Use this form when files already exist on the tape.

**TF,DL,24**  
Utility program for listing the directory of a tape.

B-2
Useful Commands and Programs (Continued)

**TF,LH,24**  Utility program for listing the header of a tape.

**UP,lu**  Command for resetting an I/O device that has been downed (made unavailable) by the system.

**XQ,DISPL**  Command for running a program, DISPL, without the terminal having to wait for the program to complete, i.e. run the program in the background.

**WH,AL**  Command for displaying the system status.
## DATA LOGGER OPERATOR NOTES

### Input / Output Device Logical Unit Number Assignments

<table>
<thead>
<tr>
<th>LU</th>
<th>Device Name</th>
<th>Select Code</th>
<th>HPIB Address</th>
<th>Device Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>terminal</td>
<td>23</td>
<td></td>
<td>busy with class request</td>
</tr>
<tr>
<td>2</td>
<td>not assigned</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>not assigned</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>not assigned</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>not assigned</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>printer</td>
<td>31</td>
<td>1</td>
<td>up</td>
</tr>
<tr>
<td>7</td>
<td>not assigned</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>not assigned</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>not assigned</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>device type = 45b</td>
<td>33</td>
<td></td>
<td>up</td>
</tr>
<tr>
<td>11</td>
<td>not assigned</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>not assigned</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>not assigned</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>not assigned</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>not assigned</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>CS 80 disc</td>
<td>27</td>
<td>2</td>
<td>up</td>
</tr>
<tr>
<td>17</td>
<td>CS 80 disc</td>
<td>27</td>
<td>2</td>
<td>up</td>
</tr>
<tr>
<td>18</td>
<td>not assigned</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>not assigned</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>CS 80 disc</td>
<td>27</td>
<td>2</td>
<td>up</td>
</tr>
<tr>
<td>21</td>
<td>floppy disc</td>
<td>27</td>
<td>3</td>
<td>up</td>
</tr>
<tr>
<td>22</td>
<td>floppy disc</td>
<td>27</td>
<td>3</td>
<td>up</td>
</tr>
<tr>
<td>23</td>
<td>floppy disc</td>
<td>27</td>
<td>3</td>
<td>up</td>
</tr>
<tr>
<td>24</td>
<td>CS 80 tape drive</td>
<td>27</td>
<td>2</td>
<td>up</td>
</tr>
<tr>
<td>25</td>
<td>not assigned</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>not assigned</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>not assigned</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>not assigned</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>not assigned</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>not assigned</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>terminal</td>
<td>23</td>
<td></td>
<td>up</td>
</tr>
<tr>
<td>32</td>
<td>terminal</td>
<td>23</td>
<td></td>
<td>up</td>
</tr>
<tr>
<td>33</td>
<td>terminal</td>
<td>23</td>
<td></td>
<td>up</td>
</tr>
<tr>
<td>34</td>
<td>terminal</td>
<td>23</td>
<td></td>
<td>up</td>
</tr>
<tr>
<td>35</td>
<td>terminal</td>
<td>23</td>
<td></td>
<td>up</td>
</tr>
<tr>
<td>36</td>
<td>terminal</td>
<td>23</td>
<td></td>
<td>up</td>
</tr>
<tr>
<td>37</td>
<td>terminal</td>
<td>23</td>
<td></td>
<td>up</td>
</tr>
</tbody>
</table>
DATA LOGGER OPERATOR NOTES

Analog Input Card Wiring Details

<table>
<thead>
<tr>
<th>A/D Mux Card Hardware Channel Number</th>
<th>Cable Color Code</th>
<th>Software Array Code</th>
<th>Channel Description</th>
<th>Signal Line Patch Panel Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>red/wht</td>
<td>1</td>
<td>10mWS</td>
<td>83</td>
</tr>
<tr>
<td>16</td>
<td>red/grn</td>
<td>2</td>
<td>20mWS</td>
<td>82</td>
</tr>
<tr>
<td>17</td>
<td>red/blu</td>
<td>3</td>
<td>30mWS</td>
<td>81</td>
</tr>
<tr>
<td>18</td>
<td>red/yel</td>
<td>4</td>
<td>40mWS</td>
<td>80</td>
</tr>
<tr>
<td>19</td>
<td>red/bln</td>
<td>5</td>
<td>48mWS</td>
<td>79</td>
</tr>
<tr>
<td>20</td>
<td>red/orn</td>
<td>6</td>
<td>NE WS</td>
<td>31</td>
</tr>
<tr>
<td>21</td>
<td>grn/wht</td>
<td>7</td>
<td>NW WS</td>
<td>32</td>
</tr>
<tr>
<td>22</td>
<td>grn/blu</td>
<td>8</td>
<td>SE WS</td>
<td>33</td>
</tr>
<tr>
<td>23</td>
<td>grn/yel</td>
<td>9</td>
<td>SW WS</td>
<td>34</td>
</tr>
<tr>
<td>24</td>
<td>grn/bln</td>
<td>10</td>
<td>10mWD</td>
<td>88</td>
</tr>
<tr>
<td>25</td>
<td>grn/orn</td>
<td>11</td>
<td>20mWD</td>
<td>87</td>
</tr>
<tr>
<td>26</td>
<td>wht/blu</td>
<td>12</td>
<td>30mWD</td>
<td>86</td>
</tr>
<tr>
<td>27</td>
<td>wht/yel</td>
<td>13</td>
<td>40mWD</td>
<td>85</td>
</tr>
<tr>
<td>28</td>
<td>wht/bln</td>
<td>14</td>
<td>48mWD</td>
<td>84</td>
</tr>
<tr>
<td>29</td>
<td>wht/orn</td>
<td>15</td>
<td>N WD</td>
<td>35</td>
</tr>
<tr>
<td>30</td>
<td>blu/yel</td>
<td>16</td>
<td>S WD</td>
<td>36</td>
</tr>
<tr>
<td>31</td>
<td>blu/blu</td>
<td>17</td>
<td>kW</td>
<td>43</td>
</tr>
<tr>
<td>32</td>
<td>blu/orn</td>
<td>18</td>
<td>kVAR</td>
<td>41</td>
</tr>
<tr>
<td>33</td>
<td>brn/yel</td>
<td>19</td>
<td>Torque</td>
<td>51</td>
</tr>
<tr>
<td>34</td>
<td>brn/orn</td>
<td>20</td>
<td>Tiedown</td>
<td>51</td>
</tr>
<tr>
<td>35</td>
<td>orn/yel</td>
<td>21</td>
<td>10mTp</td>
<td>38</td>
</tr>
<tr>
<td>36</td>
<td>vio/orn</td>
<td>22</td>
<td>48mTp</td>
<td>39</td>
</tr>
<tr>
<td>37</td>
<td>vio/red</td>
<td>23</td>
<td>BaroPr</td>
<td>37</td>
</tr>
<tr>
<td>38</td>
<td>vio/wht</td>
<td>24</td>
<td>RPM</td>
<td>40</td>
</tr>
<tr>
<td>39</td>
<td>vio/grn</td>
<td>25</td>
<td>PLC</td>
<td>69</td>
</tr>
<tr>
<td>8</td>
<td>blk/red</td>
<td>26</td>
<td>#1</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>blk/wht</td>
<td>27</td>
<td>#2</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>blk/grn</td>
<td>28</td>
<td>#3</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>blk/blu</td>
<td>29</td>
<td>#4</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>blk/yel</td>
<td>30</td>
<td>#5</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>blk/bln</td>
<td>31</td>
<td>#6</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>blk/orn</td>
<td>32</td>
<td>#7</td>
<td></td>
</tr>
</tbody>
</table>
Appendix C

WIND DIRECTION AVERAGING

Averaging wind direction signals presents a problem since a discontinuity exists between zero and 360 degrees. If the two points zero and 360 are averaged, the answer would be 180 degrees while the correct answer is either zero or 360 degrees. To correct this problem, an offset is introduced to move the mean wind direction toward the continuous part of the range. Points that are far away from the offset are wrapped around to keep the points in the correct range.

The offset is selected to be the average wind direction of the last time period. The initial offset is selected to be the first wind direction measured, since the average wind direction of the previous period is unknown.

The algorithm used for calculating the average wind direction is

If first time period then
   Offset = first wind direction point measured
Else
   Offset = last average wind direction
   End If
WindDirectionAve = 0
For I = 1 to NumberOfPoints
   WindDirectionMeasured = WindDirection(I)
   If WindDirectionMeasured > (360 - Offset) Then
      WindDirectionScaled = WindDirectionMeasured - 360 + Offset
   Else If WindDirectionMeasured < - Offset Then
      WindDirectionScaled = WindDirectionMeasured + 360 + Offset
   Else
      WindDirectionScaled = WindDirectionMeasured + Offset
   End If
   WindDirectionAve = WindDirectionAve + WindDirectionScaled
End For
WindDirectionAve = WindDirectionAve / NumberOfPoints - Offset
LISTING CHECKWS: 4:18:34

C***************************************************************************
C       SEMA/DATA/
C       SUBROUTINE CHECKWS
C       INTEGER HOME
C       CHARACTER BELL
C***************************************************************************
C       INCLUDE EMASC
C       HOME=(15400B+150B)
C       BELL=CHAR(7)
C
C       WRITE(1,101) HOME
C       CALL CHECKLU(6)  I PRINTER
C       CALL CHECKLU(10) I A/D
C       CALL CHECKLU(20) I DISK
C
C       WSNE=X10S(110S-1,6)
C       WSNW=X10S(110S-1,7)
C       WSSE=X10S(110S-1,8)
C       WSSW=X10S(110S-1,9)
C
C       IF(ABS(WSNE-WSNW).GT.1.5)THEN
C          WRITE(1,110)
C          COUNTN=COUNTN+1
C       ELSE
C          WRITE(1,130)
C          COUNTN=COUNTN-1
C          IF(COUNTN.LT.0)THEN
C             COUNTN=0
C             END IF
C       END IF
C
C       IF(ABS(WSSE-WSSW).GT.1.5)THEN
C          WRITE(1,120)
C          COUNTS=COUNTS+1
C       ELSE
C          WRITE(1,130)
C          COUNTS=COUNTS-1
C          IF(COUNTS.LT.0)THEN
C             COUNTS=0
C             END IF
C       END IF
C
C       IF(COUNTN.EQ.15)THEN
C          WRITE(6,110)
C       END IF
C
C       IF(COUNTS.EQ.15)THEN
C          WRITE(6,120)
C       END IF
C
C       RETURN
C
C       101 FORMAT(A2,/
C       110 FORMAT(' North anemometers do not agree ')
C       120 FORMAT(' South anemometers do not agree ')
C       130 FORMAT(' ')
C       END

C***************************************************************************
C       SUBROUTINE CHECKLU(LUC)
C***************************************************************************
C       INTEGER LUC,STAT1,STAT,BIT14
C
C       BIT14=40000B
C       CALL EXEC(13,LUC,STAT1)
C       STAT=STAT1 .AND. BIT14
C       IF(STAT.NE.0)THEN
C          WRITE(1,100) LUC,LUC
C       END IF
C       RETURN
C
C       100 FORMAT('LU',I2,' Down, Try UP,',I2,' at Cl> prompt')
C       END

D-2
PROGRAM DAILY

LOCAL VARIABLES
REAL RHEADER(256), AVE(20)
REAL RBLK(64,10), ZEROS(128), SCALES(128)
REAL TEMPR
INTEGER HEIGHT(2)
INTEGER IWC(6)
INTEGER DCB(144)
INTEGER DAILY(3)
INTEGER IBLOCK(128,10)
INTEGER LEN
INTEGER FmOpen, FmRead
INTEGER BLTU(15)
REAL DIRECTION(7,8)
INTEGER JULIAN(5)
INTEGER FORMFEED
CHARACTER*64 CALTBL

INCLUDE EMASC

EQUIVALENCE (IBLOCK, RBLK)
EQUIVALENCE (IBLOCK(1,1), ZEROS(1))
DATA DAILY /'DAILY '/
DATA HEIGHT/2H10 2H30/
DATA DAILY /'DAILY '/
DATA HEIGTHT/2H10 2H30/
DATA MDC/10,12,15,16/
LUP = 6
FORMFEED = 0060008
WRITE(LUP,*) 'DAILY REPORT, ', NAME

WRITE DATA TO DISK
CALL BLVDER(NAME, HEADER, NUMBER)
CALL FmOpen,(DCB, IERR, FILENAME, 'WC',1)
IF (IERR .LT. 0) THEN
CALL FmOpen,(DCB, IERR, FILENAME, 'WO',1)
ELSE IF (IERR .LT. 0) THEN
CALL ReportError(IERR, DAILY)
END IF
END IF

WRITE DATA TO DISK
CALL FmWrite(DCB, IERR, HEADER, 2*512)
IF (IERR .LT. 0) THEN
WRITE(LUP,*) 'HEADER NOT WRITEN'
CALL ReportError(IERR, DAILY)
GOTO 999
END IF

CALL VWRITE(DCB, IERR, ITP1, 28800)
CALL VWRITE(DCB, IERR, ITP2, 28800)
CALL VWRITE(DCB, IERR, ITP3, 14400)
IF (IERR .LT. 0) THEN
WRITE(LUP,*) 'DATA NOT WRITEN'
CALL ReportError(IERR, DAILY)
GOTO 999
END IF

CALL FmClose(DCB, IERR)
IF (IERR .LT. 0) THEN
WRITE(LUP,*) 'ERROR IN CLOSING DATA FILE'
CALL ReportError(IERR, DAILY)
END IF

999 CONTINUE

DAILY REPORT
IF (NUMBER .EQ. 0) GOTO 160
DO JT=1,20
AVE(JT)=0.0
END DO

160 CONTINUE
DO JC=1,4
DO JT=1,8
DIRECTION(JC, JT)=0.0
END DO
END DO

D-3
DO J=1,360
AVE(1)=AVE(1)+X*4M(J,1)/NUMBER 1 US
AVE(2)=AVE(2)+X*4M(J,2)/NUMBER 1 US
AVE(3)=AVE(3)+X*4M(J,3)/NUMBER 1 US
AVE(4)=AVE(4)+X*4M(J,4)/NUMBER 1 US
AVE(5)=AVE(5)+X*4M(J,5)/NUMBER 1 US
AVE(6)=AVE(6)+X*4M(J,6)/NUMBER 1 US
AVE(13)=AVE(13)+X*4M(J,21)/NUMBER 1 TEMPO 10M
AVE(14)=AVE(14)+X*4M(J,22)/NUMBER 1 TEMPO 48M
AVE(15)=AVE(15)+X*4M(J,15)/NUMBER 1 BARO PRESS
PWR=X*4M(J,17)/24 NUMBER
IF(PWR.LT.0.1) THEN
PWR=0.0
END IF
AVE(16)=AVE(16)+PWR
DO JC=1,4
JT=DIRECTION(JC)
IF (X*4M(JT).LE.23 OR X*4M(JT).GT.338) THEN
DIRECTION(JC,1)=DIRECTION(JC,1)+1
ELSEIF (X*4M(JT).LE.68.0) THEN
DIRECTION(JC,2)=DIRECTION(JC,2)+1
ELSEIF (X*4M(JT).LE.113.0) THEN
DIRECTION(JC,3)=DIRECTION(JC,3)+1
ELSEIF (X*4M(JT).LE.158.0) THEN
DIRECTION(JC,4)=DIRECTION(JC,4)+1
ELSEIF (X*4M(JT).LE.203.0) THEN
DIRECTION(JC,5)=DIRECTION(JC,5)+1
ELSEIF (X*4M(JT).LE.248.0) THEN
DIRECTION(JC,6)=DIRECTION(JC,6)+1
ELSEIF (X*4M(JT).LE.293.0) THEN
DIRECTION(JC,7)=DIRECTION(JC,7)+1
ELSE
DIRECTION(JC,8)=DIRECTION(JC,8)+1
END IF
END DO
END DO
END DO
DO JC=1,4
DO JT=1,8
TEMPS=DIRECTION(JC, JT)
DIRECTION(JC, JT)=TEMPS PER
WRITE(6,52) JC, JT, PER, TEMPS DIRECT ION(JC, JT)
C 52 FORMAT(JC=' ', JT=' ', PER=.' ', TEMPS=' ', DIRECTIONS=' ')
END DO
END DO
END DO
WRITE(LUP, 100) (HEIGHT(JT), AVE(JT), JT=1,2)
100 FORMAT('Average wind speed 1 A2, 'm = 'E4.1, 'mps ')
WRITE(LUP,110) AVE(3), AVE(4), AVE(5), AVE(6)
110 FORMAT('Average wind speed South = 'F4.1, 'mps ')
WRITE(LUP,120) DIRECTION(JC, JT), JC=1,4, JT=1,8
120 FORMAT('Wind direction: 10 metre 30 metre North South ')
WRITE(LUP,130) AVE(13), AVE(14), AVE(15), AVE(16)
130 FORMAT('Average temperature 10m = 'E4.1, 'deg C ')
140 FORMAT('Average barometric pressure = 'E4.1, 'kPa ')
150 FORMAT('Electric power generated = 'E4.1, 'kW ')
160 IF (NUMBER.LT.360) THEN
WRITE(LUP,140) NUMBER
140 FORMAT('Only 1,13, four minute averages were taken ')
150 FORMAT('360 four minute averages available ')
END IF
WRITE(LUP,150)
150 FORMAT('/)
C************* UPDATE CALIBRATION FACTORS ***************
CALL FmbBuildName(CALTBL, 'CALTBL', 'DAT', 'DATA')
IF (FmbOpen(DCB, IERR, CALTBL, 'ROQ', '1').LT.0) THEN
IF (IERR .EQ. -6) THEN
WRITE(LUP,*) 'CALIBRATION TABLE DOES NOT EXIST'
ELSE
CALL FmbReportError(IERR, CALTBL)
END IF
WRITE(LUP,*) 'CALIBRATION FACTORS NOT UPDATED'
GOTO 9999
END IF
LEN = FmbRead(DCB, IERR, IBLOCK, 128*10*2)
IF (IERR .LT. 0) THEN
CALL FmbReportError(IERR, CALTBL)
WRITE(LUP,*) 'CALIBRATION FACTORS NOT UPDATED'
GOTO 9999
ELSE IF (LEN .NE. 128*10*2) THEN
WRITE(LUP,*) 'CALIBRATION FACTORS NOT UPDATED'
GOTO 9999
END IF
WRITE(LUP, 1) 'SUMMARY REPORT OF DATA COMPLETED'
GOTO 9999
END IF
DO J=1,25
A(K)=SCALES(K)
D-4
CALL EXEC (11, J)

JULIAN = (JULAN(5)/2)*2

IF(JULIAN(5), NE, JULIAN) THEN
  WRITE(LUP, 5) FORMFEED
  FORMAT(A2)
END IF

C*************** UPDATE DATE AND RESCHEDULE PROGRAM ***************
9999 CALL DFILEM(FILENAME, NAME) I UPDATE FILENAMES FOR NEXT DAY
    CALL EXEC(12,DAILY,1,0,0,0)
    STOP
END
SUBROUTINE DFILN(FILEN, NAME)

SUBROUTINE RETURNS CURRENT DAY'S FILENAME

IMPLICIT NONE

INTEGER IBUF(15), SIZE
CHARACTER*64 FILEN       ! DISK FILE NAME FOR DATA FILE
CHARACTER*30 CBUF
CHARACTER*13 NAME        ! FILE NAME FOR DATA FILE
EQUIVALENCE(CBUF, IBUF)

CALL FILENAME(IBUF)

*************** BUILD DATA FILE NAME ***********************

NAME(1:6) = 'BUFILE'
NAME(7:9) = CBUF(1:3)
NAME(10:11) = CBUF(7:9) EQ. ' ' CBUF(17:17) = 'O'
NAME(10:11) = CBUF(17:17)
NAME(12:13) = CBUF(29:30)
SIZE = 4 + 566

CALL fmpBuildName(FILEN, NAME, 'DAT', 0, 'DATA', 1, SIZE)

RETURN
END
LOCAL VARIABLES

REAL PARRAY(363, 3)
REAL S1
INTEGER IUN1(5)
INTEGER IUN2(5)
INTEGER IUN3(5)
INTEGER IUN(5, 2)
INTEGER ITIM(5)
INTEGER CNI(5)
INTEGER W(7)
INTEGER W(15)
INTEGER INDX, IST(5, 15)
INTEGER IND(1, 15, 16)

DATA IUN1 /2 HUS, 2H, 2H, 2H, 2Hps/>
DATA ITIM /2HTI, 2HTM, 2H, 2H/  
DATA JICH /10, 12, 15, 16/  
DATA ATCH /10, 13, 15, 16/  

BEL = CHAR(77)
LUO = 31
IMEN1 = 15400B+046B
IMEN2 = 65000B+100B
IFUN1 = 15400B+046B
IFUN2 = 65000B+123B
HOME = 15400B+150B
ICRP1 = 15600B+052B
ICRP2 = 62000B+144B
STDE = 2

DO 15 J=1,9
ARRAY(J)=2.24
DO J=10,32
ARRAY(J)=1.0
END DO

DATA SCALE (32), ARRAY (32), DV (32), CP (4)

CHARACTER BELL
CHARACTER*4 UWS, UTP, UBP, USG
CHARACTER*8 UTO

CALL EMNASSC

DO 15 J=1,9
ARRAY(J)=2.24
DO J=10,32
ARRAY(J)=1.0
END DO

ARRAY(13)=0.296
ARRAY(19)=0.735
ARRAY(20)=0.224
CALL CLRO(1, CLNB)

WRITE(LUD, 7) HOME, Irase
WRITE(LUD, 7) ICRP1, ICRP2
IF(LUD.EQ. LUD) THEN
  WRITE(LUD, 7) IMEN1, IMEN2
  WRITE(LUD, 7) IFUN1, IFUN2
END IF

7 FORMAT (2A2)
10 WRITE(LUD, 200) HOME
200 FORMAT (2A2, 
1 17X, '/
2 17X: "SANDIA NATIONAL LABORATORIES" '/
3 17X: '34 M VANT TEST BED DATA LOGGER', '/
4 20X: 'DISPLAY TURBINE STATUS S.I. UNITS', '/
5 20X: 'DISPLAY TURBINE STATUS B.E. UNITS', '/
6 20X: 'ENTER STATUS', '/
IF (STATUS.EQ. 2 OR. STATUS.EQ. 3) GOTO 9000
CALL INPUTVAL (INPUT, LUD)
IF (INPUT.EQ. 1) THEN
  DO J=1, 32
    IF(J.EQ. 21 OR J.EQ. 22) THEN
      SCALE(J)=0.0
    ELSE
      SCALE(J)=1.0
    END IF
  END DO
ELSE IF (INPUT.EQ. 2) THEN
  SCALE(J)=ARRAY(J)
END IF

UWS=' mps'
UTO=' ft-lbm'
UWS=' mps'
UTO=' ft-lbm'
GOTO 9000
ELSE IF (INPUT.EQ. 2) THEN
  SCALE(J)=ARRAY(J)
END DO

UWS=' mps'
UTO=' ft-lbm'

D-7
C
INCLUDE SCREEN

WRITE(LUD,1006)
1006 FORMAT(1X,'PRESS ANY KEY TO RETURN TO THE MAIN MENU ')!
CALL EXEC(17,100400B,LUD,INPUT,-1,PARK1,PARK2,CLNB=100000B)
1100 CONTINUE
CALL EXEC(21,CLNB=120000B,INPUT,-1)
IF(INPUT.W.E.0) THEN
CALL CLRO(3,CLNB,LUD)
GOTO 5
END IF
IF(110S-1.W.E.INDEX) THEN
CALL CLRO(3,CLNB,LUD)
GOTO 1000
END IF
IF(IBK=IFBK(1))
IF(IBK.W.E.-1) GOTO 1100
GOTO 9000
C**************************************************************************
C PLOT DATA ***************************************************************
2000 CONTINUE
CALL CHANNEL(CNB,NCH,WS,1UN1,1UN,LUD,CHNM,CHUNIT)
DO JK=1,5
1UN2(JK)=1UN(JK,1)
1UN3(JK)=1UN(JK,2)
END DO
CALL GETINDEX( IDX,S1,LUD )
IST(1)=IS+17
IST(2)=13S+18
IST(3)=110S+18
IST(4)=14M+18
IST(5)=128M+18
ISTART=IST(IDX)
DO J3=1,NCH+l
IF(CNB(J3).LE.0) THEN I WS NOT TO BE PLOTTED
CONTINUE
ELSE IF(CNB(J3).LE.32) THEN
IF(IDX.EQ.1) THEN
DO J1=1,361
J2=J1*ISTART
IF( J2.GT.380 ) J2=J2-380
PARRAY(J1,J3)=XMS(J2,CNB(J3))
END DO
ELSE IF(IDX.EQ.2) THEN
DO J1=1,361
J2=J1*ISTART
IF( J2.GT.380 ) J2=J2-380
PARRAY(J1,J3)=XMS(J2,CNB(J3))
END DO
ELSE IF(IDX.EQ.3) THEN
DO J1=1,361
J2=J1*ISTART
IF( J2.GT.380 ) J2=J2-380
PARRAY(J1,J3)=XMS(J2,CNB(J3))
END DO
ELSE IF(IDX.EQ.4) THEN
DO J1=1,361
J2=J1*ISTART
IF( J2.GT.380 ) J2=J2-380
PARRAY(J1,J3)=XMS(J2,CNB(J3))
END DO
ELSE IF(IDX.EQ.5) THEN
DO J1=1,361
J2=J1*ISTART
IF( J2.GT.380 ) J2=J2-380
PARRAY(J1,J3)=XMS(J2,CNB(J3))
END DO
ENDIF
ELSE
WRITE(LUD, 2003) INOME, IRASE
FORMAT(2A2, 'ERROR INVALID TIME INDEX NUMBER'/'
'PRESS ANY KEY TO CONTINUE')
CALL INPUTVAL(IDUM, LUD)
GOTO 2000
END IF
ELSE IF(CNB(J3).EQ.40) THEN
IF(IDX.EQ.1) THEN
DO J1=1,361
J2=J1+IISTART
IF(J2.GT.380) J2=J2-380
PARRAY(J1,J3)=XHS(J2,19)*XHS(J2,24)*0.1047
END DO
ELSE IF(IDX.EQ.2) THEN
DO J1=1,361
J2=J1+IISTART
IF(J2.GT.380) J2=J2-380
PARRAY(J1,J3)=X3S(J2,19)*X3S(J2,24)*0.1047
END DO
ELSE IF(IDX.EQ.3) THEN
DO J1=1,361
J2=J1+IISTART
IF(J2.GT.380) J2=J2-380
PARRAY(J1,J3)=X10S(J2,19)*X10S(J2,24)*0.1047
END DO
ELSE IF(IDX.EQ.4) THEN
DO J1=1,361
J2=J1+IISTART
IF(J2.GT.380) J2=J2-380
PARRAY(J1,J3)=X4F4(J2,19)*X4M(J2,24)*0.1047
END DO
ELSE IF(IDX.EQ.5) THEN
DO J1=1,361
J2=J1+IISTART
IF(J2.GT.380) J2=J2-380
PARRAY(J1,J3)=X28N(J2,19)*X2EIM(J2,24)*0.1047
END DO
ELSE
WRITE(LUD, 2003) INOME, IRASE
CALL INPUTVAL(IDUM, LUD)
GOTO 2000
END IF
ELSE
WRITE(LUD, 2004) INOME, IRASE
FORMAT(2A2, 'ERROR INVALID CHANNEL NUMBER'/'
'PRESS ANY KEY TO CONTINUE')
CALL INPUTVAL(IDUM, LUD)
GOTO 2000
END IF
ITIM(4) = 2MSE
ITIM(5) = 2HC
CALL PLUT(PARRAY, LUD, NCH, IUN1, IUN2, IUN3, ITIM, CHB, SI)
CONTINUE
9000 CONTINUE
WRITE(LUD, 9010) INOME, IRASE
9010 FORMAT(2A2)
STOP
999 END
C ************** SUBROUTINE INPUTVAL ***************
C SUBROUTINE RETURNS AN INTEGER ENTERED FROM THE KEY BOARD
C
C SUBROUTINE INPUTVAL(R, LUD)
CHARACTER SEL
INTEGER C, R
ILU=LUD+1000
CALL EXEC(1, ILU, C, -1)
SEL = CHAR(C/256)
R = ICHAR(SEL)-ICHAR('0')
RETURN
END
C ************** SUBROUTINE GETINDEX ***************
C SUBROUTINE PROMPTS THE TERMINAL FOR AN INDEX FOR PLOTTING
C
C SUBROUTINE GETINDEX( IND, SI, LUD )
C
CHARACTER BELL, ESC
CHARACTER*2 HOME, CLS
INTEGER INDEX
BEL = CHAR(7)
ESC = CHAR(27)
HOME = ESC // 'H'
CLS = ESC // 'J'
C
WRITE(LUD, *) HOME, CLS
2001 FORMAT(2A2)
WRITE(LUD, 2002)
2002 FORMAT(B1X, 3/1, 1L3) SELECT LENGTH OF PLOT '/''
1 20K,' '1 LAST 3 MINUTES OF DATA''/
2 20K,' '2 LAST 18 MINUTES OF DATA''/
3 20K,' '3 LAST HOUR OF DATA''/
4 20K,' '4 LAST DAY OF DATA''/
5 20K,' '5 LAST WEEK OF DATA''/
2100 CALL INPUTVAL(INHX, LW)

IF(INHX.EQ.1) THEN
  SI = 0.5
ELSE IF(INHX.EQ.2) THEN
  SI = 5.0
ELSE IF(INHX.EQ.3) THEN
  SI = 10.0
ELSE IF(INHX.EQ.4) THEN
  SI = 24.0
ELSE IF(INHX.EQ.5) THEN
  SI = 1680
ELSE
  WRITE(LU0, 2003) BELL
  2003 FORMAT(A20,/,20X,'INVALID ENTRY TRY AGAIN _')
  GO TO 2100
END IF
RETURN

C************************* SUBROUTINE CHANNEL *************************

SUBROUTINE CHANNEL(CNB, NCH, WS, IUN1, IUN, LWM, CHNM, CHUNIT)

INTEGER CNB(3), NCH, WS, IUN1, IUN, LWM, CHNM, CHUNIT

CHARACTER BELL, ESC
CHARACTER*2 HOME, CLS
CHARACTER*4 CHARF

INTEGER INOX, IST(5).

BELL=CHAR(7)
ESC=CHAR(27)
HOME=ESC // 'h'
CLS=ESC // 'J'

WRITE(LW, 80) HOME, CLS

80 FORMAT(A20,///)
  1 'DO YOU WANT TO PLOT WIND SPEED? (Y OR N CR>) _'
READ(LW, 81) IMPLT

81 FORMAT(A1)
IF( IMPLT.EQ. 'Y' .OR. IMPLT.EQ. 'y' ) THEN
  WS = 1
  WRITE(LU0, 82) HOME, CLS
  82 FORMAT(A20)
  84 WRITE(LU0, 85) HOME

85 FORMAT(A20,///)
  1 SELECT WIND SPEED VARIABLE TO PLOT //
  2 '20X,1 WIND SPEED - 10m//' // 1
  4 '20X,2 WIND SPEED - 30m, EQUATOR//' // 1 3
  7 '20X,3 WIND SPEED - NORTH-E//' // 1 9
  8 '20X,4 WIND SPEED - NORTH-W//' // 1 9
  9 '20X,5 WIND SPEED - SOUTH-E//' // 1 8
  1 '20X,6 WIND SPEED - SOUTH-W//' // 1 9

WRITE(LU0, 88) HOME, CLS

88 FORMAT(A20, 'ENTER NUMBER _')
CALL INPUTVAL(NUB, LW)

IF( NUB.EQ.1 ) THEN
  CNB(1) = 1
  IUN1(2)=2H1
  IUN1(3)=2HOM
ELSE IF( NUB.EQ.2 ) THEN
  CNB(1) = 3
  IUN1(2)=2H3
  IUN1(3)=2HOM
ELSE IF( NUB.EQ.3 ) THEN
  CNB(1) = 6
  IUN1(2)=2HN
  IUN1(3)=2HE
ELSE IF( NUB.EQ.4 ) THEN
  CNB(1) = 7
  IUN1(2)=2HN
  IUN1(3)=2HW
ELSE IF( NUB.EQ.5 ) THEN
  CNB(1) = 8
  IUN1(2)=2HS
  IUN1(3)=2HE
ELSE IF( NUB.EQ.6 ) THEN
  CNB(1) = 9
  IUN1(2)=2HS
  IUN1(3)=2HW
ELSE
  WRITE(LU0,*) 'INVALID INPUT TRY AGAIN'
  GOTO 84
END IF
ELSE
  WS = 0
  CNB(1) = 0
END IF
WRITE(LU0, 90) HOME, CLS

90 FORMAT(A20,///,'PLOT 1 OR 2 ADDITIONAL CHANNELS _'
CALL INPUTVAL(NCH, LW)

IF( NCH.EQ.1 ) THEN
  NCH = 1
WRITE(LU0, 95) HOME, CLS

95 FORMAT(A20)

100 WRITE(LU0, 101) HOME

101 FORMAT(A20, ///)
1 ' 8X _ SELECT VARIABLE(S) TO PLOT _26X _ '/CHNB
1 6X _ Power - Rotor _16X _ '17 Wind Speed - 10m _ '/140,1
<table>
<thead>
<tr>
<th>No.</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Power - Electrical</td>
<td>16X, 18</td>
</tr>
<tr>
<td>2</td>
<td>Wind Speed - 30m</td>
<td>/19,17</td>
</tr>
<tr>
<td>3</td>
<td>Power - Reactive</td>
<td>16X, 20</td>
</tr>
<tr>
<td>4</td>
<td>Wind Speed - North-E</td>
<td>/18, 6</td>
</tr>
<tr>
<td>5</td>
<td>Rotor Speed</td>
<td>16X, 21</td>
</tr>
<tr>
<td>6</td>
<td>Wind Speed - South-E</td>
<td>/19, 8</td>
</tr>
<tr>
<td>7</td>
<td>Temperature - 10M</td>
<td>16X, 22</td>
</tr>
<tr>
<td>8</td>
<td>Wind Speed - South-E</td>
<td>/12, 9</td>
</tr>
<tr>
<td>9</td>
<td>Temperature - 48M</td>
<td>16X, 23</td>
</tr>
<tr>
<td>10</td>
<td>Wind Direction - 10m</td>
<td>/22, 10</td>
</tr>
<tr>
<td>11</td>
<td>Barometric Pressure</td>
<td>16X, 24</td>
</tr>
<tr>
<td>12</td>
<td>Wind Direction - 30m</td>
<td>/13, 12</td>
</tr>
<tr>
<td>13</td>
<td>Wind Direction - North-E</td>
<td>/20, 15</td>
</tr>
</tbody>
</table>

**DO J2 = 1, NCH**

**DO J2 = 2, NCH+1**

NBU = NU(J2-1)

**IF(NUB.EQ.1)** THEN

CNB(J2) = 40

IUN(1, J2-1) = 2H0

IUN(2, J2-1) = 2H1

IUN(3, J2-1) = 2H2

IUN(4, J2-1) = 2H3

IUN(5, J2-1) = 2H4

ELSE IF(NUB.EQ.2)** THEN

CNB(J2) = 17

IUN(1, J2-1) = 2H5

IUN(2, J2-1) = 2H6

IUN(3, J2-1) = 2H7

IUN(4, J2-1) = 2H8

IUN(5, J2-1) = 2H9

ELSE IF(NUB.EQ.3)** THEN

CNB(J2) = 18

IUN(1, J2-1) = 2H10

IUN(2, J2-1) = 2H11

IUN(3, J2-1) = 2H12

IUN(4, J2-1) = 2H13

IUN(5, J2-1) = 2H14

ELSE IF(NUB.EQ.4)** THEN

CNB(J2) = 24

IUN(1, J2-1) = 2H15

IUN(2, J2-1) = 2H16

IUN(3, J2-1) = 2H17

IUN(4, J2-1) = 2H18

IUN(5, J2-1) = 2H19

ELSE IF(NUB.EQ.5)** THEN

CNB(J2) = 19

IUN(1, J2-1) = 2H20

IUN(2, J2-1) = 2H21

IUN(3, J2-1) = 2H22

IUN(4, J2-1) = 2H23

IUN(5, J2-1) = 2H24

ELSE IF(NUB.EQ.6)** THEN

CNB(J2) = 31

IUN(1, J2-1) = 2H25

IUN(2, J2-1) = 2H26

IUN(3, J2-1) = 2H27

IUN(4, J2-1) = 2H28

IUN(5, J2-1) = 2H29

ELSE IF(NUB.EQ.7)** THEN

CNB(J2) = 22

IUN(1, J2-1) = 2H30

IUN(2, J2-1) = 2H31

IUN(3, J2-1) = 2H32

IUN(4, J2-1) = 2H33

IUN(5, J2-1) = 2H34

ELSE IF(NUB.EQ.8)** THEN

CNB(J2) = 13

IUN(1, J2-1) = 2H35

IUN(2, J2-1) = 2H36

IUN(3, J2-1) = 2H37

IUN(4, J2-1) = 2H38

IUN(5, J2-1) = 2H39

ELSE IF(NUB.EQ.9)** THEN

CNB(J2) = 20

IUN(1, J2-1) = 2H40

IUN(2, J2-1) = 2H41

IUN(3, J2-1) = 2H42

IUN(4, J2-1) = 2H43

IUN(5, J2-1) = 2H44

ELSE IF(NUB.EQ.10)** THEN

CNB(J2) = 26

IUN(1, J2-1) = 2H45

IUN(2, J2-1) = 2H46

IUN(3, J2-1) = 2H47

IUN(4, J2-1) = 2H48

IUN(5, J2-1) = 2H49

D-11
ELSE IF (NUB.EQ.12) THEN
CWB(J2)=28
IUN(1, J2-1)=2H
IUN(2, J2-1)=2H
IUN(3, J2-1)=2H
IUN(4, J2-1)=2H
IUN(5, J2-1)=2H

ELSE IF (NUB.EQ.13) THEN
CWB(J2)=29
IUN(1, J2-1)=2H
IUN(2, J2-1)=2H
IUN(3, J2-1)=2H
IUN(4, J2-1)=2H
IUN(5, J2-1)=2H

ELSE IF (NUB.EQ.14) THEN
CWB(J2)=30
IUN(1, J2-1)=2H
IUN(2, J2-1)=2H
IUN(3, J2-1)=2H
IUN(4, J2-1)=2H
IUN(5, J2-1)=2H

ELSE IF (NUB.EQ.15) THEN
CWB(J2)=31
IUN(1, J2-1)=2H
IUN(2, J2-1)=2H
IUN(3, J2-1)=2H
IUN(4, J2-1)=2H
IUN(5, J2-1)=2H

ELSE IF (NUB.EQ.16) THEN
CWB(J2)=32
IUN(1, J2-1)=2H
IUN(2, J2-1)=2H
IUN(3, J2-1)=2H
IUN(4, J2-1)=2H
IUN(5, J2-1)=2H

ELSE IF (NUB.EQ.17) THEN
CWB(J2)=33
IUN(1, J2-1)=2H
IUN(2, J2-1)=2H
IUN(3, J2-1)=2H
IUN(4, J2-1)=2H
IUN(5, J2-1)=2H

ELSE IF (NUB.EQ.18) THEN
CWB(J2)=34
IUN(1, J2-1)=2H
IUN(2, J2-1)=2H
IUN(3, J2-1)=2H
IUN(4, J2-1)=2H
IUN(5, J2-1)=2H

ELSE IF (NUB.EQ.19) THEN
CWB(J2)=35
IUN(1, J2-1)=2H
IUN(2, J2-1)=2H
IUN(3, J2-1)=2H
IUN(4, J2-1)=2H
IUN(5, J2-1)=2H

ELSE IF (NUB.EQ.20) THEN
CWB(J2)=36
IUN(1, J2-1)=2H
IUN(2, J2-1)=2H
IUN(3, J2-1)=2H
IUN(4, J2-1)=2H
IUN(5, J2-1)=2H

ELSE IF (NUB.EQ.21) THEN
CWB(J2)=37
IUN(1, J2-1)=2H
IUN(2, J2-1)=2H
IUN(3, J2-1)=2H
IUN(4, J2-1)=2H
IUN(5, J2-1)=2H

ELSE IF (NUB.EQ.22) THEN
CWB(J2)=38
IUN(1, J2-1)=2H
IUN(2, J2-1)=2H
IUN(3, J2-1)=2H
IUN(4, J2-1)=2H
IUN(5, J2-1)=2H

ELSE IF (NUB.EQ.23) THEN
CWB(J2)=39
IUN(1, J2-1)=2H
IUN(2, J2-1)=2H
IUN(3, J2-1)=2H
IUN(4, J2-1)=2H
IUN(5, J2-1)=2H

ELSE IF (NUB.EQ.24) THEN
CWB(J2)=40
IUN(1, J2-1)=2H
IUN(2, J2-1)=2H
IUN(3, J2-1)=2H
IUN(4, J2-1)=2H
IUN(5, J2-1)=2H

ELSE IF (NUB.EQ.25) THEN
CWB(J2)=41
IUN(1, J2-1)=2H
IUN(2, J2-1)=2H
IUN(3, J2-1)=2H
IUN(4, J2-1)=2H

D-12
ELSE IF(NUL.EQ.26) THEN
CMB(J2)=26
IUM(1,J2-1)=2HAD
IUM(2,J2-1)=2HSO
IUM(3,J2-1)=2HUT
IUM(4,J2-1)=2HNO
IUM(5,J2-1)=2Heg
END IF

WRITE(LU1,103) BELL
103 FORMAT(A2,35X,'INVALID ENTRY TRY AGAIN')
GOTO 100
END IF

CONTINUE
RETURN
END
**PROGRAM NAME:** GETDA

**LOCAL VARIABLES:**

- INTEGER DAILY(3), GETDA(3), DISPL(3)
- INTEGER I(X, 40)
- INTEGER DCB(144)
- INTEGER IHEADER(512)
- INTEGER ERROR
- INTEGER EXEC(32)
- INTEGER BUF(15)
- REAL BUF6(432)
- REAL SQ3S(7), SQ1OS(7)
- REAL X, XTEMP
- REAL BLM(32), SUFKO(25)
- LOGGING PRINTER

**DATA READ FROM A/D BOARD:***

- ICM=10008+8
- CALL EXEC(1,10,IX,32,ICM)
- DO J=10,16
- OFFSET(J)=16000.0 - FLOAT(IX(J)+6)
- END DO

**READ DATA FROM A/D BOARD:**

- CALL EXEC(1,10,IX,32,ICM)
- CALL EXEC(10,IX,32,ICM)

**CALIBRATION AND SUM:**

- DO 20 J=1,25
- IXT=IX(J)+6
- IF (STATUS.EQ.0) IXT=0
- XE=FLOAT(IXT)
- IF (J.GE.10.AND.J.LE.16) THEN
- IF (XG.GT.32000.-OFFSET(J)) THEN
- VAL=IXE+32000.*OFFSET(J)
- ELSE IF (XG.LT.-OFFSET(J)) THEN
- VAL=IXE+32000.*OFFSET(J)
- ELSE
- VAL=IXE
- END IF
- END IF
- SLMD(J)=SLMD(J)+VAL
- SUMSQ(J)=SUMSQ(J)+VAL**2
- XHS(J)=X(x(J),A(J))
- CONTINUE
- DO 20 J=26,32
- IXT=IX(J-1)
- XE=C(J)*FLOAT(IXT)+D(J)
- XHS(J)=X(J,25)+XE
- SLMD(J)=SLMD(J)+VAL
- SUMSQ(J)=SUMSQ(J)+VAL**2
- CONTINUE

**SUM OF SQUARES FOR 60 SEC AVG:**

- IF (NOD(J).EQ.0) THEN
- IF (J.GE.10.AND.J.LE.16) THEN
- DO 35 K=1,18
- XH=H$(K, J)
- CALL MAP(XH, OFFSET)
- WRITE(I1,11) (IX(J), J=1,7), IX(32)
- CONTINUE
- FORMAT (8(16,2X))
- IXT=IX(T)
- XE=FLOAT(IXT)
- XHS(J)=X(J,25)+C(1)*XE*D(1)
- XH=H$(J, N, S, 6, 6)
- SUM(SQ)=0
- CONTINUE

**SUM OF LAST 6.0 SEC VALUES:**

- IF (MOD(16, 6).EQ.0) THEN
- IF (INS.LT.6) THEN
- DO 30 J=1,32
- XH=H$(K, J)
- CALL MAP(XH, OFFSET)
- XH=H$(K, J)
- CALL MAP(XH, OFFSET)
- CONTINUE
- DO 35 K=1,18
- XH=H$(K, J)
- CALL MAP(XH, OFFSET)
- XH=H$(K, J)
- CALL MAP(XH, OFFSET)
- CONTINUE
- DO 35 K=1,18
- XH=H$(K, J)
- CALL MAP(XH, OFFSET)
- XH=H$(K, J)
- CALL MAP(XH, OFFSET)

**SUM OF SQUARES FOR 30 MIN:**

- STD DEV FOR 30 MIN STORED

---

**DATA DAILY, GETDA, DISPL,'DAILY', GETDA', DISPL'/**

**LOGGING PRINTER**

**READ FROM A/D**

- READ, 10, ARRAY 1X, 32 WORDS READ, START CHANNEL 8

**CALIBRATE DATA AND SUM:**

- WRITE(31,11) (IX(J), J=1,7), IX(32)

**FORMAT (8(16,2X))**

- IXT=IX(I)
- XE=FLOAT(IXT)
- XHS(J)=X(J,25)+C(1)*XE*D(1)
- XH=H$(J, N, S, 6, 6)
- SUM(SQ)=0

---

**SUM OF LAST 6.0 SEC VALUES:**

- IF (MOD(16, 6).EQ.0) THEN
- IF (INS.LT.6) THEN
- DO 30 J=1,32
- XH=H$(K, J)
- CALL MAP(XH, OFFSET)
- XH=H$(K, J)
- CALL MAP(XH, OFFSET)
- CONTINUE
- DO 35 K=1,18
- XH=H$(K, J)
- CALL MAP(XH, OFFSET)
- XH=H$(K, J)
- CALL MAP(XH, OFFSET)

---

**SUM OF SQUARES FOR 30 MIN:**

- STD DEV FOR 30 MIN STORED

---

**SUM OF LAST 6.0 SEC VALUES:**

- IF (MOD(16, 6).EQ.0) THEN
- IF (INS.LT.6) THEN
- DO 30 J=1,32
- XH=H$(K, J)
- CALL MAP(XH, OFFSET)
- XH=H$(K, J)
- CALL MAP(XH, OFFSET)
- CONTINUE
- DO 35 K=1,18
- XH=H$(K, J)
- CALL MAP(XH, OFFSET)
- XH=H$(K, J)
- CALL MAP(XH, OFFSET)
X3S(I3S,J) = XT/6.0-OFFSET
IF(J.GE.10.AN D.J.LE.16) THEN
X = X3S(I3S,J)
CALL CHECK(X)
X3S(I3S,J) = X
END IF
CONTINUE
ELSE
DO 60 J = 1, 32
XT = 0.0
OFFSET = 0
IF(J.GE.10.AND.J.LE.16) OFFSET = 180-XHS(IHS-3,J)
DO 45 K = 15, IHS
VAL = XHS(K,J)
CALL MAP(VAL,OFFSET)
XT = XT + VAL
CONTINUE
45
X3S(I3S,J) = XT/6.0-OFFSET
IF(J.GE.10.AND.J.LE.16) THEN
X = X3S(I3S,J)
CALL CHECK(X)
X3S(I3S,J) = X
END IF
CONTINUE
60
END IF
DO J = 2, 7
XTEMP = SQS(J)/6.0-X3S(I3S,J)*25
STODEV(1,J) = ABS(XTEMP)**0.5
STQ(S) = 0.0
ENDDO
I3S = I3S + 1
IF(I3S.EQ.381) I3S = 1
END IF
C *********************************************** X10S = (SUM OF LAST 20 0.5 SEC VALUES)/20.0 ****** IF MOD(1.20).EQ.0. THEN
DO 50 J = 1, 32
XT = 0.0
OFFSET = 0
IF(J.GE.10.AND.J.LE.16) OFFSET = 180-XHS(IHS-10,J)
DO 45 K = 19, IHS
VAL = XHS(K,J)
CALL MAP(VAL,OFFSET)
XT = XT + VAL
CONTINUE
45
X10S(I10S,J) = XT/20.0-OFFSET
IF(J.GE.16.AN D.J.LE.16) THEN
X = X10S(I10S,J)
CALL CHECK(X)
X10S(I10S,J) = X
END IF
CONTINUE
50
END IF
DO J = 2, 7
XTEMP = SQ10S(J)/20.0-X10S(I10S,J)*2
STODEV(2,J) = ABS(XTEMP)**0.5
STQ(I10S) = 0.0
ENDDO
I10S = I10S + 1
IF(I10S.EQ.381) I10S = 1
C *********************************************** 11MA = (SUM OF LAST 120 IX)/120 ************ C *********************************************** 11MS = SORT((SUM OF LAST 120 IX**2)/120.11MA**2) **** IF(I.EQ.120) THEN COUNT = COUNT + 1
DO 70 J = 1, 25
SUMTP = SUM(J)/120.0-OFFSET1M(J)
IF(SUMTP.LT.32000) THEN
SUMTP = 0
SUMTP = SUMTP - 32000
END IF
IF(SUMTP.GT.32000) THEN
SUMTP = 0
SUMTP = SUMTP + 32000
END IF
IF(STATUS.EQ.0) SUMTP = 0
DATA(11M,J) = INT(SUMTP)
BUF(COUNT,J) = SUMTP
X = SUMSQ(J)/120.0-SUMTP**2
IF(X.LT.0.0) X = 0.0
X = SORT(X)
IF(STATUS.EQ.0) X = 0
DATA(11M,J+25) = INT(X)
SUM(J) = 0.0
SUMSQ(J) = 0.0
IF(J.GE.10.AND.J.LE.16) THEN
OFFSET1M(J) = 16000.-SUMTP
END IF
CONTINUE
70
END IF
COUNT = COUNT + 1
IF(COUNT.EQ.4) THEN
K = 0
END IF
C *********************************************** X4M = (SUM OF LAST 4 11MA)/4.0 ******* D-15
**ERROR:** Document contains programming code with syntax errors and unclear flow. Recoding and debugging are required for readability.

```
ELSE
OFFSET=0
END IF
DO 950 K=1,14M
VAL=BUF(K, J)
CALL MAP(VAL, OFFSET)
XT=XT+VAL
CONTINUE
X=XT/7.0-OFFSET
IF(J.GE.10.AND.J.LE.16) THEN
  X2BM(J)=X
  CALL CHECK(X)
END IF
950
CONTINUE
COUNT=COUNT+1
COUNT4=COUNT4+1
COUNT1=0

C--------- X2BM = (SUM OF LAST 7 X4M)/7.0 ---------------

IF(COUNT4.GE.7) THEN
  IF(14M.GT.7) THEN
    DO 960 J=1,32
      XT=0.0
      IF(J.GE.10.AND.J.LE.16) THEN
        OFFSET=180-X4M(J)
      ELSE
        OFFSET=0
      END IF
      DO 970 K=1,14M
        VAL=X4M(K, J)
        CALL MAP(VAL, OFFSET)
        XT=XT+VAL
        CONTINUE
      DO 980 K=1,14M,80
        VAL=X4M(K, J)
        CALL MAP(VAL, OFFSET)
        XT=XT+VAL
        CONTINUE
      X2BM(J)=XT/7.0-OFFSET
      IF(J.GE.10.AND.J.LE.16) THEN
        X2BM(J)=X
        CALL CHECK(X)
        X2BM(J)=X
        END IF
      END IF
    END DO
    X28M(128M, J)=X2BM(J)
    CALL CHECK(X2BM(J))
    COUNT1=COUNT1+1
    END IF
  ELSE
    CONTINUE
  END IF
  960
  ELSE
    CONTINUE
  970
  ELSE
    CONTINUE
  980
  END IF

990

C--------- MAKE PROGRAM DORMANT - SAVE RESOURCES UNTIL RESCHEDULED ****
CALL EXEC(6,0,1)
C--------- CHECK STATUS **********************************************
IF(STATUS.EQ.2) THEN
  STORE DATA AND STOP
ELSE IF(STATUS.EQ.3) THEN
  STOP PROGRAMS WITHOUT STOP
GOTO 999
END IF
C--------- CHECK FOR BREAK ********************************************
IBK=IFBRK()
IF(IBK.EQ.-1) GOTO 5
CALL TIME(BUF)
IF(IBK.EQ.-1) THEN
```
WRITE(LUP,80) (BUF(JI),J I=1,15) ELSE WRITE(LUP,85) (BUF(JI),JI=1,15) END IF 80 FORMAT///'BREAK DETECTED BY PROGRAM GETDA',15A2) 85 FORMAT///'PROGRAMS STOPPED',15A2) CALL BLDNDER(NAME THE HEADER COUNT) CALL FmpDopen(DCB, IERR,FILNAM,MC',1) IF (IERR.LT.0) THEN CALL FmpOpen(DCB, IERR,FILNAM, 'Ao',1) IF (IERR.LT.0) THEN WRITE(LUP,*) 'DATA FILE NOT OPENED' GOTO 999 END IF END IF
C*************** WRITE DATA TO DISK ***********************
CALL FmpWrite(DCB, IERR, THE HEADER COUNT) IF (IERR.LT.0) THEN WRITE(LUP,*) 'HEADER NOT WRITTEN' CALL FmpReportError(IERR, GETDA) GOTO 999 END IF DO JI=1,1440 DO JK=1,20 ITP1(JI,JK)=IDATA(JI,JK) ITP2(JI,JK)=IDATA(JI,JK+20) END DO DO JK=1,10 ITP3(JI,JK)=IDATA(JI,JK+40) END DO CALL Write(DCB, IERR, ITP1,28800) CALL Write(DCB, IERR, ITP2,28800) CALL Write(DCB, IERR, ITP3,14400) IF (IERR.LT.0) THEN WRITE(LUP,*) 'DATA NOT WRITTEN' CALL FmpReportError(IERR, GETDA) GOTO 999 END IF CALL FmpClose(DCB, IERR) IF (IERR.LT.0) THEN WRITE(LUP,*) 'ERROR IN CLOSING DATA FILE' CALL FmpReportError(IERR, GETDA) ELSE WRITE(LUP,*) 'DATA WRITTEN TO DISK' END IF 999 CONTINUE CALL EXEC(12, GETDA,0) CALL EXEC(12,DAILY,0) STOP END C******************** SUBROUTINE MAP ***********************
SUBROUTINE MAP(VAL OFFSET) IF(OFFSET.EQ.0) RETURN IF(VAL.GT.360-OFFSET) THEN VAL=VAL-360+OFFSET ELSE IF(VAL.LT.-OFFSET) THEN VAL=VAL+360+OFFSET ELSE VAL=VAL+OFFSET RETURN END IF END C******************** SUBROUTINE CHECK ***********************
SUBROUTINE CHECK(X) IF(X.LT.0.0) THEN X=X+360.0 ELSE IF(X.GT.360.0) THEN X=X-360.0 END IF RETURN END
SUBROUTINE BLDHEADER(NAME, IHDR, COUNT)

SUBROUTINE BLDHEADER, NAME IS PASSED TO THE SUBROUTINE AND HEADER IS RETURNED

INTEGER IHEADER(512), DCB(16), IERR(10), IBLOCK(128), IHDR(512)
INTEGER UNITS(2, 128)
REAL RHEADER(256), RBLOCK(64, 10), ZROS(128), SCALES(128)
CHARACTER*16 NAME, CNAME
INTEGER NNAME(0)
EQUIVALENCE (NAME, CNAME)
EQUIVALENCE (IHEADER, RHEADER)
EQUIVALENCE (IBLOCK, ABLOCK)
EQUIVALENCE (IBLOCK(1, 1), ZROS(1))
EQUIVALENCE (IBLOCK(1, 3), SCALES(1))
EQUIVALENCE (IBLOCK(1, 5), UNITS(1, 1))
CALL FmpRead(DCB, IERR(1), 'CALBL.DAT:DATA', 'ROO', 1)
IF (IERR(1), LT, 0) THEN
  CALL FmpReportError(IERR(1), 'BUILDHEADER')
  WRITE(1, *) 'PROGRAM STOPPED !'
  STOP
END IF
CALL FmpRead(DCB, IERR(2), IBLOCK, 128*10)
DO 10 L=1, 25
  RHEADER(L)=ZROS(L) ! ZERO ARRAY
  N=64
  RHEADER(K)=SCALES(L) ! SLOPE ARRAY
  M=2*L+255
  IHEADER(M)=UNITS(1, M) ! UNIT ARRAY
  IHEADER(M+1)=UNITS(2, M) ! UNIT ARRAY
  J=4*L+1
  IHEADER(J)=L ! CHANNELS RECORDED AVE
  IHEADER(J+25)=L+25 ! CHANNELS RECORDED STD
10 CONTINUE
CALL FmpClose(DCB, IERR(3))
IF (IERR(3), LT, 0) THEN
  CALL FmpReportError(IERR(3), 'BLDHEADER')
  RETURN
END IF
RHEADER(64)=11 ! FILE TYPE
RHEADER(136)=60.0 ! SAMPLE RATE IN SEC
RHEADER(237)=86400. ! LENGTH OF RECORD IN SEC
IHEADER(394)=COUNT ! # OF 4MIN AVE TAKEN
IHEADER(405)=563 ! # OF BLOCKS
IHEADER(400)=25 ! # OF CHANNELS
DO 20 L=1, 16, 2
  NAME(L:L)=NAME(L:L+1)
20 CONTINUE
DO 30 L=1, 8
  J=L+387
  IHEADER(J)=INAME(L)
30 CONTINUE
DO L=1, 512
  IHDR(L)=IHEADER(L)
END DO
RETURN
END
PROGRAM INITIAL

LOCAL VARIABLES

REAL RBLK(64,10),ZEROS(128),SCALES(128)
INTEGER INTRC(10),DCB(144),GETDA(3),DAILY(3),DISPL(3)
INTEGER INHEADER(512)   ! FILE HEADER
INTEGER ITBLK(120,10)   ! CALIBRATION TABLE
INTEGER ILBN(15)        ! LENGTH OF FILE READ
INTEGER FmOpen,FmRead   ! Fmp ROUTINES CALLED AS INTEGER Func.
INTEGER BUF(144)        ! BUFFER FOR RETRIEVING DATE
INTEGER*4 IBK(5,7,2)    ! Integer data block for FMP CALLS
REAL RBLK(2,7)          ! Real data block for FMP CALLS
CHARACTER*64 CALTBL     ! DISK FILE NAME FOR CALIBRATION FILE

INCLUDE EMASC

EQUIVALENCE (IBLOCK, RBLK)
EQUIVALENCE (IBLOCK(1,1),ZEROS(1))

EQUIVALENCE (IBLOCK(1,3),SCALES(1))
EQUIVALENCE (IBLOCK(1,2),ABK(1,1))
DATA GETDA,DAILY,DISPL,'GETDA','DAILY','DISPL'
LU = 1
LUP = 6
DO JK=1,380
  DO JL=1,32
    XMS(JK,JL)=0.0
    X3S(JK,JL)=0.0
    X10S(JK,JL)=0.0
    X4M(JK,JL)=0.0
    X20M(JK,JL)=0.0
  END DO
END DO
DO JL=1,25
  OFSET(JL)=0.0
END DO
    ! TAKE DATA

************ Extra Channel Data **********

IDUMT=FmpOpen(IDC8,IERR,'/DATA/CHANNEL.DAT','RO',1)
IF (IERR.LT.0) THEN
  CALL FmReportError(IERR,INITIAL)
  END IF
WRITE(1,*), 'Reading extra channel data.'
CALL FmRewind(IDC8,IERR)
IDUMT=FmpRead(IDC8,IERR,IBK(280))
IF (IERR.LT.0) THEN
  WRITE(1,*), 'ERROR IN READING, PRESS <CR>.'
  CALL FmReportError(IERR,INITIAL)
END IF
DO 230 I=1,7
  DO 240 J=1,14
  CHNM(J,J)=IBK(J,1,1)
  CHNM(1,J)=IBK(5,1,1)
  C(J)=RBLK(J,1)
  D(J)=RBLK(2,1)
240 CONTINUE
230 CONTINUE

******************************************************************

CALL FmpProgram('GETDA.RUN::USER1','GETDA',IP,IERR(1))
CALL FmpProgram('DAILY.RUN::USER1','DAILY',IP,IERR(2))
CALL FmpProgram('DISPL.RUN::USER1','DISPL',IP,IERR(3))
IF (IERR(3).LT.0) THEN
  CALL FmpReportError(IERR(3),GETDA)
  WRITE(LUP,*) 'GETDA.RUN PROGRAM INITIAL STOPPED'
END IF
IF (IERR(2).LT.0) THEN
  CALL FmpReportError(IERR(2),DAILY)
  WRITE(LUP,*) 'DAILY.RUN PROGRAM INITIAL STOPPED '
END IF
IF (IERR(1).LT.0) THEN
  CALL FmpReportError(IERR(1),DISPL)
  WRITE(LUP,*) 'DISPL.RUN PROGRAM INITIAL STOPPED '
END IF

******************************************************************

CALL LKEMA

******************************************************************

READ SYSTEM TIME ***************
CALL EXEC(11, TIME)
TIME(3)=TIME(3)+2  ! INCREMENT STARTING TIME 2 MINS
IF (TIME(3).GE.59) THEN  ! THEN START 1 MIN PAST HOUR
  TIME(4)=TIME(4)+1
  TIME(3)=1
END IF
TIME(2)=0
TIME(1)=0
11=TIME(4)*60+TIME(3)+1  ! SET 1 MIN TO STARTING TIME

CALL TIME(BUF)
CALL DFIREN(FILENAME, NAME), WRITE(LUP,10) (BUF(J)), J=1,15
10 FORMAT(//, '**** 34M VAUT DATA LOGGER ' , 15A2, ' ****)
LEN = FmOpen(DCB, IERR(10), FILENAME, 'RD', 1)
IF (IERR(10), GE, 0) THEN  ! IF DATA FILE ALREADY
  WRITE(LUP,*) 'READING PREVIOUS DATA FROM DISK'
  LEN = FmRead(DCB, IERR(6), IHEADER,512)
  CALL VNREAD(DCB, IERR, IDATA, 1800)
  CALL VNREAD(DCB, IERR, IDATA, 1800)
  CALL VNREAD(DCB, IERR, IDATA, 1800)
  COUNT = IHEADER(394)
  IF (IERR .LT. 0) GOTO 30
ELSE  ! SET DATA FILE TO ZERO
  WRITE(LUP,*) 'INITIALIZE 1 MIN ARRAYS'
  DO 50 J=1,50
    DO 60 K=1,1440
      IDATA(K,J)=0
  CONTINUE
  60 CONTINUE
END IF

********* READ CALIBRATION FACTORS FROM DISK ************
CALL FmHasForeignKey(CALTBL, 'CALTBL', 'DAT', '0', 'DATA')
IF( FmOpen(DCB, IERR(7), CALTBL, 'RD', 1), LT, 0) THEN
  WRITE(LUP,*) 'CALIBRATION TABLE DOES NOT EXIST'
  ELSE
    CALL FmReportError(IERR(7), CALTBL)
  END IF
  WRITE(LUP,*) 'PROGRAM STOPPED !
END IF
LEN = FmRead(DCB, IERR(8), IBLOCK, 128*10*2)
IF( IERR(8), LT, 0) THEN
  CALL FmReportError(IERR(8), CALTBL)
  WRITE(LUP,*) 'PROGRAM STOPPED !
  STOP
ELSE IF( LEN .NE. 128*10*2 ) THEN
  WRITE(LUP,*) 'COULD NOT READ EXPECTED AMOUNT OF DATA'
  WRITE(LUP,*) 'FROM CALIBRATION TABLE. PROGRAM STOPPED!
  STOP
ELSE
  CONTINUE
END IF
DO 70 K=1,25
  A(K)=SCALES(K)
  B(K)=ZEROS(K)
70 CONTINUE

********* SCHEDULE PROGRAMS **********************
CALL EXEC(12, GETDA 1, 1.50, TIME(4), TIME(3))  ! SCD @ T4:15 & EVERY 0.15
  TIME(4)=TIME(4)+1
  IF (TIME(3).GE.59) THEN
    TIME(4)=TIME(4)+1
    TIME(3)=1
  END IF
105 FORMAT( 'PROGAM GETDA SCHEDULED TO START AT ' , 12, ' : ', 12)
CALL EXEC(10, DISPL)  ! SCD IMMEDIATELY
WRITE(LUP, *) 'PROGRAM INITIALLY COMPLETE

D-20
listing MENU.FTN:::4:5:39

LOCAL VARIABLES

INTEGER CLAS
INTEGER WD(7)
INTEGER ANS
INTEGER IDCB(14)
INTEGER*4, BK(3,2)
REAL RBK(2,7)
INTEGER CNN
INTEGER JICH(4)
INTEGER*4, NCNM(4)
INTEGER BUF(15)
INTEGER, FORMFEED, HOME, IRASE
INTEGER IDX, IST(5), STOE
INTEGER LUD
REAL SCALE(32), DV(32), CP(4)

CHARACTER BELL
CHARACTER*4 UWS, UTP, UBP, USG
CHARACTER*6 UTQ

EQUIVALENCE (IBK(1,1,2), RBK(1,1))

DATA JICH(10, 12, 15, 16)/
DATA SCALE/2*1.0/

CALL CLRO(CLASS)
CALL FmpOpen(IDCBl, ERR, '/DATA/CHANNEL.DAT', 'WRO', 1)
IF (ERR. LT. 0) THEN
  CALL FmpReport Error(ERR, CHANNEL)
  CALL INPUTVAL(WAIT)
END IF

10 WRITE(1,5) HOME, IRASE
5 FORMAT(2A2)
15 CALL FIME(BUF)
WRITE(1, 30) INCOME, (BUF(JJ), JJ = 1, 2, STATUS)
30 FORMAT(2A2, '2X, 'SANDBIA NATIONAL LABORATORIES' ,
1 '2DX', 'METRE TEST BED DATA LOGGER', '2DX', '15A2/',
2 '2DX', 'Current STATUS = ', 12/,
3 '2DX', 'Set current DATA to ZERO', /
4 '2DX', 'Return/Continue taking DATA', /
5 '2DX', 'Stop current DATA and STOP taking DATA', /
6 '2DX', 'Display Display Data', /
7 '2DX', 'Display Display Data', /
8 '2DX', 'Input/Change Extra Channel data', /
9 '2DX', 'Enter Number _', )
CALL INPUTVAL(STATUS)
IF (STATUS .EQ. 0) THEN
  CALL CHECK(ANS)
  IF (ANS .EQ. 1) STATUS = 0
  ELSE IF (ANS .EQ. 1) THEN
    STATUS = 1
  ELSE IF (ANS .EQ. 2) THEN
    CALL CHECK(ANS)
    IF (ANS .EQ. 1) THEN
      STATUS = 2
      GOTO 900
    END IF
  ELSE IF (ANS .EQ. 3) THEN
    CALL CHECK(ANS)
    IF (ANS .EQ. 1) THEN
      STATUS = 3
      GOTO 900
    END IF
  ELSE IF (ANS .EQ. 7) THEN
    WRITE(1, 90) GOTO 900
90 FORMAT(//)
100 EXIT PROGRAM

ELSE IF (STATUS .EQ. 4) THEN
  GOTO 100
ELSE IF (STATUS .EQ. 5) THEN
  GOTO 200
ELSE IF (STATUS .EQ. 6) THEN
  WRITE(1, 40) GOTO 200
40 FORMAT(////) 'CHANGE LU FOR DISPL

READ(1,45) LUD
45 FORMAT(12)
ELSE

WRITE(1,*) BELL
60 FORMAT(12, 'INVALID RESPONSE')
DO 50 JKL= 1,1000
CONTINUE
END IF
GOTO 10
900 STOP

C******************************************************************************
C ******************************************************************************
C 100 CONTINUE
C******************************************************************************
C 100 CONTINUE
WRITE(1,5) INHOME, IRASE
WRITE(LUDD,5) INHOME, IRASE
UMS=' mps'
UTD='kmh'
UPD='C'
USD='KN'
ICD=0

1000 INPUT=0
INDEX2=105-1
IF (IC.GE.10) THEN
WRITE(LUDD,5) INHOME, IRASE
IC=0
ELSE
IC=IC+1
END IF
WRITE(LUDD,1001) INHOME
1001 FORMAT(A2,'ISX', 'SNLA 34 METRE VAWT TEST BED DATA',
1 'LOGGER', '20X', '//' )
CALL CHECKWS
DO J=1,32
DIV(J)=X35(INDEX2,J)
END DO

C******************************************************************************
C ******************************************************************************
C ******************************************************************************
C 200 CONTINUE
C******************************************************************************
C 200 CONTINUE
WRITE(1,2000) INHOME, IRASE
WRITE(LUDD,2000) FORMFA/ED
GOTO 200

C******************************************************************************
C ******************************************************************************
C******************************************************************************
C******************************************************************************
C******************************************************************************
C******************************************************************************
C******************************************************************************
READ(1,2030) (NCHNM(J),J=1,4)
IF (NCHNM(1).NE.4H) THEN
  DO J = 1,4
    CHNM(CHN,J) = NCHNM(J)
  END DO
END IF
WRITE(1,2035)
READ(1,2030) NCHNM(1)
IF (NCHNM(1).NE.4H) THEN
  CHUNIT(CHN) = NCHNM(1)
END IF
WRITE(1,2040)
READ(1,2045) DCHNM
GOTO 200
2000 FORMAT(2A2)
2005 FORMAT('Channel Description Units Scale Zero',/,
     (3X,'6A,2X,4A,6X,2X,4X,FB,6X,2X,FB,6)())
2010 FORMAT('Input channel number to change: 1-4 or /
     '0' to print data, or R to READ values from the disk' ,/,
     'S to SAVE values on the disk',/,
     'or M to return to MAIN MENU.',/)!
2025 FORMAT('For channel: ',/,'/.
     'C* saves current description/units, sets A/B=0'/,
     'I: input channel description, up to 16 characters,' ,/)
2030 FORMAT(4(A))
2035 FORMAT('Input channel units, up to 4 characters, _')
2040 FORMAT('Input channel scale, Y=Ax+B A=_')
2050 FORMAT('Input channel offset, Y=Ax+B B=_')
END
C******************** SUBROUTINE RETURN A 1 IF OK TO CONTINUE
C SUBROUTINE CHECK(ANS)
C    INTEGER ANS, CH
C 10 WRITE(1,100)
100 FORMAT('ARE YOU SURE (Y)? ENTER Y OR N _')
   ILU=1+100B
   CALL INPUTVAL(CH)
   CH=CH+CHAR('0')
   IF(CH.EQ.69 .OR. CH.EQ.121) THEN
     Y OR y
     ANS=1
   ELSE
     ANS=2
   END IF
   RETURN
END
C******************** SUBROUTINE INPUTVAL ***********************
C SUBROUTINE INPUTVAL(INT)
C CHARACTER SEL
C INTEGER C, INT, ISEL
C INTEGER LUINPUT, ILU
C LUINPUT=1
C ILU=LUINPUT+100B
C CALL EXEC(1,ILU,C-1)
C SEL = CHAR( (C/255) )
C ISEL = ICHAR(SEL)
C IF (ISEL.EQ.77 .OR. ISEL.EQ.109) THEN
C  1 IF M
C   INT = B
C ELSEIF (ISEL.EQ.83 .OR. ISEL.EQ.115) THEN
C  1 IF S
C   INT = 9
C ELSEIF (ISEL.EQ.82 .OR. ISEL.EQ.114) THEN
C  1 IF R
C   INT = 10
C ELSE
C   INT = ISEL-ICHAR('0')
C END IF
C RETURN
END
C INCLUDE CHECKWS
SUBROUTINE PLUT(PARRAY, LUD, WS, NCH, IUN1, IUN2, IUN3, ITIM, ICH, 

REAL PARRAY(363, 3)  ! DATA ARRAYS
REAL PLOTAR(363)   ! ARRAY TO PLOT
REAL DATO(363)     ! TIME AXIS ARRAY
REAL DEL(3)        !
REAL FP(3)         !
REAL TOTA, TOT2, TOT3  ! CHANNEL TOTALS FOR AVE
REAL AVG2, AVG3  ! CHANNEL AVERAGES
REAL DMAX(3)  ! CHANNEL MAX VALUE
REAL DMIN(3)  ! CHANNEL MIN VALUE
REAL DR(3)  ! CHANNEL RANGE
REAL SI  ! SECONDS PER POINT
REAL S1  !
REAL CH(3)  ! WORKING BUFFER FOR HEADERS
INTEGER IUN1(5) ! AXIS HEADER FOR WS
INTEGER IUN2(5) ! AXIS HEADER FOR CH 2
INTEGER IUN3(5) ! AXIS HEADER FOR CH 3
INTEGER ITIM(5) ! AXIS HEADER FOR TIME
INTEGER NCH  ! NUMBER OF CH (1 OR 2)
INTEGER LW  ! NUMBER OF OUTPUT DEVICE
INTEGER US  ! WS >0 THEN PLOT WS

DATA MFN /'SNLA/USDA 34 MVA NT TEST BED DATA LOGGER'/

IN HOME = 154 OB = 150  ! HOME CURSOR
IRASE = 154008 + 1128  ! CLEAR ALPH
IREM = 154006 + 1500  ! RING BELL
IND = 361  ! NUMBER OF POINTS TO PLOT ?
NRD = 361  ! NUMBER OF POINTS TO PLOT ?
NNN = 1  ! PLOT EVERY POINT

DO J = 1, 361
  DATO(J) = J - 1
END DO

FIND DATA'S MAX & MIN AMPLITUDES

10 DMAX(2) = 1.E+30
DMIN(2) = 1.E+30
DMAX(3) = -1.E+30
DMIN(3) = -1.E+30
DO I = 1, IND
  IF( PARRAY(1, 2) .GT. DMAX(2) ) DMAX(2) = PARRAY(1, 2)
  IF( -PARRAY(1, 2) .LT. DMIN(2) ) DMIN(2) = -PARRAY(1, 2)
  IF( PARRAY(1, 3) .GT. DMAX(3) ) DMAX(3) = PARRAY(1, 3)
  IF( -PARRAY(1, 3) .LT. DMIN(3) ) DMIN(3) = -PARRAY(1, 3)
END DO
DR(2) = DMAX(2) - DMIN(2)
DR(3) = DMAX(3) - DMIN(3)

AUTO SCALE ON TWO ADDITIONAL CHANNELS

Determine the 1ST POINT AND DELTA TO PLOT

30 DELCH = 0.01
DO 70 I = 2, 3
  IF( DR(I) .LE. 0.0 ) THEN
    DEL( I ) = 0.01
    FP( I ) = DMAX( I ) - 2.0 * DEL( I )
    GOTO 70
  END IF
70  CONTINUE
35 FLAST = 0.02
DALL = 0.02
DELL1 = 0.1
40 IF( DR(1) .LE. FLAST ) GOTO 60
  IF( FLAST .GE. DELL1 ) THEN
    DALL = DELL1
    DELL1 = DELL1 + 10.0
  END IF
50 FLAST = FLAST + DALL
GOTO 60
60 DEL(1) = FLAST / 2.0
IFLAS1 = IFIX( DMIN(1) / 10000.0 / DEL(1) )
VARCHK = DMIN(1) - FLOAT( IFLAS1 ) * 10000.0 / DEL(1)
IFLAS = VARCHK / DEL(1)
IF( DMIN(1) .LE. 0.0 ) IFLAS = IFLAS - 1
IF( IFLAS .EQ. 2 ) IFLAS = IFLAS - 1
FPI(1) = (FLOAT(IFLAS1) * 10000.0 + FLOAT(IFLAS)) * DEL(1)
1 CONTINUE
80 FP1 = 0.0
DELT = 25.0

FIX TIME SCALE PARAMETERS

TT = SI * (NRD - 1.0) * WNN
BT = DATO(1)
DELT = TT / 5.0
IF( TT > 3600.0 ) THEN
  TT = TT / 3600.0
1 CONTINUE
ITIM = 1
DELT = 1
IF( ITIM > 3600.0 ) THEN
  TT = TT / 3600.0
1 CONTINUE
24
DELT = DELT / 3600.0
DO 700 I=1, MRD
  DATO( I ) = DATO( I ) / 3600.0
700 CONTINUE
IF(IT .GT. 200) THEN
  BT = BT / 60.0
  DELT = DELT / 60.0
DO 710 I=1, MRD
  DATO( I ) = DATO( I ) / 60.0
710 CONTINUE
END IF
WRITE(LUJ,581) INOME
WRITE(LUJ,581) INOME
581 FORMAT(9A2)

CALL PLOTS(LUJ)
CALL PLOTS(ARRAY,1,LUJ)

1ST POINT AND DELTA STORED FOR LATER USE
FP2 = FP( 2 )
FP3 = FP( 3 )
DEL2 = DEL( 2 )
DEL3 = DEL( 3 )
PARRAY( IND + 1, 2 ) = FP( 2 )
PARRAY( IND + 1, 3 ) = FP( 3 )
PARRAY( IND + 1, 1 ) = 0
PARRAY( IND + 2, 1 ) = DEL1
DATA( IND + 1 ) = BT
DATA( IND + 2 ) = DELT / 5I

TOTA = 0.0
TOT2 = 0.0
TOT3 = 0.0
DO 750 I=1, IND
  TOTA = TOTA + PARRAY( I, 1 )
  TOT2 = TOT2 + PARRAY( I, 2 )
  TOT3 = TOT3 + PARRAY( I, 3 )
750 CONTINUE
AVGA = TOTA / IND
AVG2 = TOT2 / IND
AVG3 = TOT3 / IND

START PLOTTING
CALL ZNEW
CALL ZMOVE( 0.0 , 5.85 )

WRITE(GBUFF,560) MFW(1:50)
560 FORMAT(1X,A)
CALL PRINTG(GBUFF)
IF(WS.GT.0) THEN
  IF(NCH.EQ.1) THEN
    WRITE(GBUFF,570) AVGA,1,AVG2
  ELSE IF(NCH.EQ.2) THEN
    WRITE(GBUFF,570) AVGA,AVG2,2,AVG3
  END IF
ELSE IF(NCH.EQ.1) THEN
  WRITE(GBUFF,575) AVGA,1,AVG2
  ELSE IF(NCH.EQ.2) THEN
    WRITE(GBUFF,575) AVGA,AVG2,2,AVG3
  END IF
575 FORMAT(1X,'CHANNELS',2(12,'AVE=','F6.3,2X))
END IF
CALL PRINTG(GBUFF)

FILL LINE ARRAYS
DO 760 I=1, IND
  DATA( I ) = DATO( I ) - BT
  PARRAY( I, 1 ) = PARRAY( I, 1 ) + 3.0 * DEL1
760 CONTINUE

DRAW AXISES
CALL PPLLOT( 0.35 , 0.4 , -3 )
NCL = MRD / 100
NCL = NCL * 10 + 10
TIME = BT
CALL AXIS( 0.0 , 0.0 , ITIM , -10 , 5.0 , 0.0 , TIME , DELT )
CALL AXIS( 0.0 , 0.0 , INUM , 10 , -4.0 , 90.0 , FP( 2 ) , 
  1 , DEL( 2 ) )
IF( NCH.EQ.2 ) THEN
  CALL AXIS( 5.0 , 0.0 , INUM , -10 , -4.0 , 90.0 , 
  1 , FP( 3 ) , DEL( 3 ) )
END IF
IF(WS.GT.0) THEN
  CALL AXIS( 5.5 , 3.0 , INUM , -10 , 2.0 , 90.0 , 0.0 , 25.0)
  DO JT=1,IND+2
    PLOTAR(JT)=PARRAY(JT,1)
END DO
CALL ELINE(DATO, PLOTAR, IND, 1, NCL, 0)
CALL SYMB(5.85, 3.0, 0.14, 0.0, 0.0, -1)
END IF
DO JT=1, IND+2
   PLOTAR(JT) = PARRAY(JT, 2)
END DO
CALL ELINE(DATO, PLOTAR, IND, 1, NCL, 1)
CALL SYMB(-0.25, 0.5, 0.14, 1, 0.0, -1)
IF( NCH .EQ. 2 ) THEN
   DO JT=1, IND+2
      PLOTAR(JT) = PARRAY(JT, 3)
   END DO
   CALL ELINE(DATO, PLOTAR, IND, 1, NCL, 11)
   CALL SYMB(5.35, 0.5, 0.14, 11, 0.0, -1)
END IF
330 CONTINUE
CALL ZANG(0.0)
CALL EDATE(GBUFF)
CALL PPLOT(3.9, -0.4, 3)
CALL PRINTG(GBUFF)
CALL AIMG6
WRITE(LUD, 580) IBELL
580 FORMAT(//////, A2)
WRITE(LUD, 590)
590 FORMAT(///)
   1  63X:'PRESS CR+/',
   2  63X:'TO MAIN MENU/',
   3  63X:'TO COPY PLOT _'
   4  63X:'TO GRAPHICS/',
   5  63X:'TO KEY BOARD/',
READ(LUD, 335) IDUMMY
335 FORMAT(A2)
340 CALL EDITM(LUD)
RETURN
END
COMMON VARIABLE DEFINITIONS

INTEGER 1, 1H5, 13S, 1T0S, 14M, 12BM, 11M, TIME NAME, FILENAME, LU,
COUNT, COUNT1, COUNT2, COUNT3, COUNT4, CHNM, CHUNIT,
2 STATUS, OFFSET1M, LUD, STDDEV

COMMON/1, 1H5, 13S, 1T0S, 14M, 12BM, 11M, TIME NAME, FILENAME, LU,
COUNT, COUNT1, COUNT2, COUNT3, COUNT4, CHNM, CHUNIT,
2 STATUS, OFFSET1M, LUD, STDDEV

SHAREABLE EMA VARIABLE DEFINITIONS

COMMON/DATA/A(25), B(25), CALIBRATION FACTORS -- SLOPE & ZERO OFFSET
C C(7), D(7), CALIBRATION FACTORS -- CHANGEABLE CHANNELS
C X(30, 32), 1 0.5 SEC VALUES LAST 3 MIN
C X3S(380, 32), 1 3 SEC AVERAGES LAST 18 MINS
C X10S(380, 32), 1 10 SEC AVERAGES LAST HOUR
C X4M(380, 32), 1 4 MIN AVERAGES LAST DAY
C X2BM(380, 32), 1 20 MIN AVERAGES LAST WEEK
C IDATA(1440, 50), 1 1 MIN AVE & STD DEVIATIONS
C ITP1(1440, 20), 1 1 MIN AVE & SD TEMP ARRAY FOR WRIT
C ITP2(1440, 10), 1 1 MIN AVE & SD TEMP ARRAY FOR WRIT
C ITP3(1440, 10), 1 1 MIN AVE & SD TEMP ARRAY FOR WRIT

REAL STDDEV(2, 7) 1 Standard Deviation of non stored channels, 3 sec/10 sec
REAL OFFSET1M(25) 1 OFFSET ARRAY FOR CORRECTING WD

CHARACTER*64 FILENAME 1 DATA STORAGE FILE DESCRIPTOR
CHARACTER*16 NAME 1 DATA STORAGE FILE NAME

INTEGER LU
COMMON/LU, 1H5, 13S, 1T0S, 14M, 12BM, 11M, NAME, FILENAME, LU,
COUNT, COUNT1, COUNT2, COUNT3, COUNT4, CHNM, CHUNIT,
2 STATUS, OFFSET1M, LUD, STDDEV
listing SCREEN::4:24:32

DO J=1,4
  J=J+1CH(J)
  IF(DV(J1).LE.23.0.OR.DV(J1).GT.338) THEN
    W(J)=(2HN)
  ELSEIF(DV(J1).LE.68) THEN
    W(J)=(2NE)
  ELSEIF(DV(J1).LE.113) THEN
    W(J)=(2ME)
  ELSEIF(DV(J1).LE.158) THEN
    W(J)=(2NEE)
  ELSEIF(DV(J1).LE.203) THEN
    W(J)=(2NS)
  ELSEIF(DV(J1).LE.248) THEN
    W(J)=(2NSW)
  ELSEIF(DV(J1).LE.293) THEN
    W(J)=(2NW)
  ELSE
    W(J)=(2HNN)
  END IF
END DO

RotorPower=Torque*Rpm/Pi/60

CP=POWER/(0.5+Cos)*AREA**2*3
CP(1)=CP(1)/101.4
CP(2)=CP(2)/101.4
CP(3)=CP(3)/101.4
CP(4)=CP(4)/101.4

WRITE(LUS,1010)
WRITE(LUS,1011)
WRITE(LUS,1012)
WRITE(LUS,1013)
WRITE(LUS,1014)
WRITE(LUS,1015)
WRITE(LUS,1016)
WRITE(LUS,1017)
WRITE(LUS,1018)
WRITE(LUS,1019)
WRITE(LUS,1020)
WRITE(LUS,1021)
WRITE(LUS,1022)
WRITE(LUS,1023)
WRITE(LUS,1024)
WRITE(LUS,1025)
WRITE(LUS,1026)
WRITE(LUS,1027)

CALL FTIME(BUF)

WRITE(LUS,1027)

1010 FORMAT('10 m',10x,'F4.1,A5,F6.1,1,deg','A2.10x,1')
1011 FORMAT('Equator:','Aero','F5.3','System','F5.3,1')
1012 FORMAT('10 metre','Aero','F6.1,1,deg','A2.10x,1')
1013 FORMAT('10 metre','F4.1,A5,1')
1014 FORMAT('10 metre','F4.1,A5,1')
1015 FORMAT('10 metre','F4.1,A5,1')
1016 FORMAT('10 metre','F4.1,A5,1')
1017 FORMAT('10 metre','F4.1,A5,1')
1018 FORMAT('10 metre','F4.1,A5,1')
1019 FORMAT('10 metre','F4.1,A5,1')
1020 FORMAT('10 metre','F4.1,A5,1')

1021 FORMAT('10 metre','F4.1,A5,1')
1022 FORMAT('10 metre','F4.1,A5,1')
1023 FORMAT('10 metre','F4.1,A5,1')
1024 FORMAT('10 metre','F4.1,A5,1')
1025 FORMAT('10 metre','F4.1,A5,1')
1026 FORMAT('10 metre','F4.1,A5,1')
1027 FORMAT('10 metre','F4.1,A5,1')
Appendix E

HP 1000
SYSTEM GENERATION ANSWER FILE
AND
WELCOME FILE

E-1
RTE: A.3 Primary System Generation Answer File (Non VC+)

REVISION 02-22-89 M E RALPH

* links,cp,, use current page links

***************
* System Relocation
***************

re SVCTR::RTE

re XEXEC::RTE
re XMEMRY::RTE

re XPL60::RTE, , A-600 no CDS no double precision floating point

re XSAM::RTE
re XTIME::RTE
re XCHED::RTE
re XSTRNG::RTE
re XLOCK::RTE
re XERLOG::RTE
re XSTAT::RTE
re XLOAD::RTE

re XRTIA::RTE
re XIDMOD::RTE
re X/RTE/XPERR
re XCLASS::RTE

***************
* SEARCH LIBRARIES
***************

re SYSA::RTE
re XMAIN::RTE
re SYSLB::RTE

***************
* OS PARTITIONS
***************

***************
* Driver partitions
***************

re XDD.33::RTE,, CS 80 DISK

re %ID.00::RTE, , ASYNC SERIAL

re XDD.35::RTE,, CS 80 DISK

re XDD.36::RTE,, DISK 9133

re %ID.37::RTE, , ASYNC MUX

re %IDMOD::RTE, , TERMINAL

re XDD.00::RTE,, TERMINAL

re ALIGN

re %ID.50, , A/D DRIVER

***************
* Table Generation -- configure LU tables
***************

* ASYNC MUX LU'S

***************
* IFI, %IDMOD::RTE,SC:238,TX:20


* DVT, %IDMOD::RTE,M266X,LU:34,QU:FI,TX:57,DP:1:200048,
A. J. Eggers, Jr.
RANN, Inc.
260 Sheridan Ave., Suite 414
Palo Alto, CA 94306

John Ereaux
RR No. 2
Woodbridge, Ontario L4L 1A6
CANADA

Dr. R. A. Galbraith
Dept. of Aerospace Engineering
James Watt Building
University of Glasgow
Glasgow G12 8QG
Scotland

A. D. Garrad
Garrad Hasson
10 Northampton Square
London EC1M 5PA
UNITED KINGDOM

P. R. Goldman
Wind/Oceans Technologies Division
U.S. Department of Energy
1000 Independence Avenue
Washington, DC 20585

Dr. I. J. Graham
Dept. of Mechanical Engineering
Southern University
P.O. Box 9445
Baton Rouge, LA 70813-9445

Professor G. Gregorek
Aeronautical & Astronautical Dept.
Ohio State University
2300 West Case Road
Columbus, OH 43220

Professor N. D. Ham
Aero/Astro Dept.
Massachusetts Institute of Technology
77 Massachusetts Avenue
Cambridge, MA 02139

W. L. Harris
Aero/Astro Dept.
Massachusetts Institute of Technology
77 Massachusetts Avenue
Cambridge, MA 02139

T. Hillesland
Pacific Gas and Electric Co.
3400 Crow Canyon Road
San Ramon, CA 94583

Eric N. Hinrichsen
Power Technologies, Inc.
P.O. Box 1058
Schenectady, NY 12301-1058

W. E. Holley
US WindPower
Suite 3050
400 West Cummings Park
Woburn, MA 01801

J. J. Iannucci
Pacific Gas and Electric Co.
3400 Crow Canyon Road
San Ramon, CA 94583

K. Jackson
Dynamic Design
123 C Street
Davis, CA 95616

O. Krauss
Division of Engineering Research
Michigan State University
East Lansing, MI 48825

V. Lacey
Indal Technologies, Inc.
3570 Hawkestone Road
Mississauga, Ontario L5C 2V8
CANADA

A. Laneville
Faculty of Applied Science
University of Sherbrooke
Sherbrooke, Quebec J1K 2R1
CANADA

Dist-2
G. G. Leigh
New Mexico Engineering Research Institute
Campus P.O. Box 25
Albuquerque, NM 87131

L. K. Liljergren
120 East Penn Street
San Dimas, CA 71773

Robert Lynette
R. Lynette & Assoc., Inc.
15042 NE 40th Street
Suite 206
Redmond, WA 98052

Peter Hauge Madsen
Riso National Laboratory
Postbox 49
DK-4000 Roskilde
DENMARK

David Malcolm
Lavalin Engineers, Inc.
Atria North - Phase 2
2235 Sheppard Avenue East
Willowdale, Ontario M2J 5A6
CANADA

Bernard Masse
Institut de Recherche d’Hydro-Quebec
1800, Montée Ste-Julie
Varennes, Quebec JOL 2P0
CANADA

G. M. McNerney
US WindPower
Suite 3050
400 West Cummings Park
Woburn, MA 01801

Brian McNiff
Engineering Consulting Services
55 Brattle Street
So. Berwick, ME 03908

R. N. Meroney
Dept. of Civil Engineering
Colorado State University
Fort Collins, CO 80521

Alan H. Miller
10013 Tepopa Drive
Oakdale, CA 95361

D. Morrison
New Mexico Engineering Research Institute
Campus P.O. Box 25
Albuquerque, NM 87131

V. Nelson
Department of Physics
West Texas State University
P.O. Box 248
Canyon, TX 79016

J. W. Oler
Mechanical Engineering Dept.
Texas Tech University
P.O. Box 4289
Lubbock, TX 79409

Debby Oscar
MIT Branch
P.O. Box 313
Cambridge, MA 02139

Dr. D. I. Page
Energy Technology Support Unit
B 156.7 Harwell Laboratory
Oxfordshire, OX11 ORA
UNITED KINGDOM

Ion Paraschivoiu
Dept. of Mechanical Engineering
Ecole Polytechnique
CP 6079
Succursale A
Montreal, Quebec H3C 3A7
CANADA

Troels Friis Pedersen
Riso National Laboratory
Postbox 49
DK-4000 Roskilde
DENMARK

Helge Petersen
Riso National Laboratory
Postbox 49
DK-4000 Roskilde
DENMARK
Walter V. Thompson  
410 Ericwood Court  
Manteca, CA 95336

R. W. Thresher  
Solar Energy Research Institute  
1617 Cole Boulevard  
Golden, CO 80401

K. J. Touryan  
P.O. Box 713  
Indian Hills, CO 80454

W. A. Vachon  
W. A. Vachon & Associates  
P.O. Box 149  
Manchester, MA 01944

P. Vittecoq  
Faculty of Applied Science  
University of Sherbrooke  
Sherbrooke, Quebec J1K 2R1  
CANADA

T. Watson  
Canadian Standards Association  
178 Rexdale Boulevard  
Rexdale, Ontario M9W 1R3  
CANADA

L. Wendell  
Battelle-Pacific Northwest Laboratory  
P.O. Box 999  
Richland, WA 99352

W. Wentz  
Aero Engineering Department  
Wichita State University  
Wichita, KS 67208

R. E. Wilson  
Mechanical Engineering Dept.  
Oregon State University  
Corvallis, OR 97331

R. C. Maydew
L. W. Davison
R. C. Reuter, Jr.
D. W. Lobitz
E. D. Reedy
J. H. Biffle
C. R. Dohrmann
D. R. Martinez
C. W. Peterson
J. H. Strickland
G. F. Homicz
S. A. Landenberger (5)
W. I. Klein (3)
C. L. Ward (8)

P. S. Wilson
D. L. Hartley
V. L. Dugan
D. G. Schueler
H. M. Dodd (50)
T. D. Ashwill
D. E. Berg
M. E. Ralph
D. C. Reda
M. A. Rumsey
L. L. Schluter
W. A. Stephenson
H. J. Sutherland
P. S. Veers
T. G. Carrie
J. R. Wackerly
Data Logger for the 34-Meter
Vertical Axis Wind Turbine Test Bed

Mark E. Ralph
Wind Energy Research Division
Sandia National Laboratories
Albuquerque, NM 87185

ABSTRACT
This report discusses the purpose and requirements that were established for the data logger at the 34-m diameter, research-oriented vertical axis wind turbine, the Test Bed, which Sandia National Laboratories built at Bushland, Texas. The data logger is a minicomputer-based system that collects data from 35 channels, displays the collected data, and records them on a hard disc. Both the hardware and software that make up the data logger are also described, and the operator's instructions and the operating system commands and procedure files are appended. The data logger is used to obtain long-term data to characterize the wind at the site of the turbine, record the performance data of the control system, obtain a continuous record of events at the test site, consolidate displays for the test engineer, and provide a display of current information for visitors to the site.
ACKNOWLEDGMENTS

I would like to thank Emil Kadlec for defining this project, Matt Mattison for help in selecting the hardware, Tait Cyrus (NMERI) for very useful programming suggestions, Paul Klimas, Bill Stephenson, and Ron Davis (USDA) for help in debugging and improving the system, and Anne Poore and Dale Berg for help with editing this report.