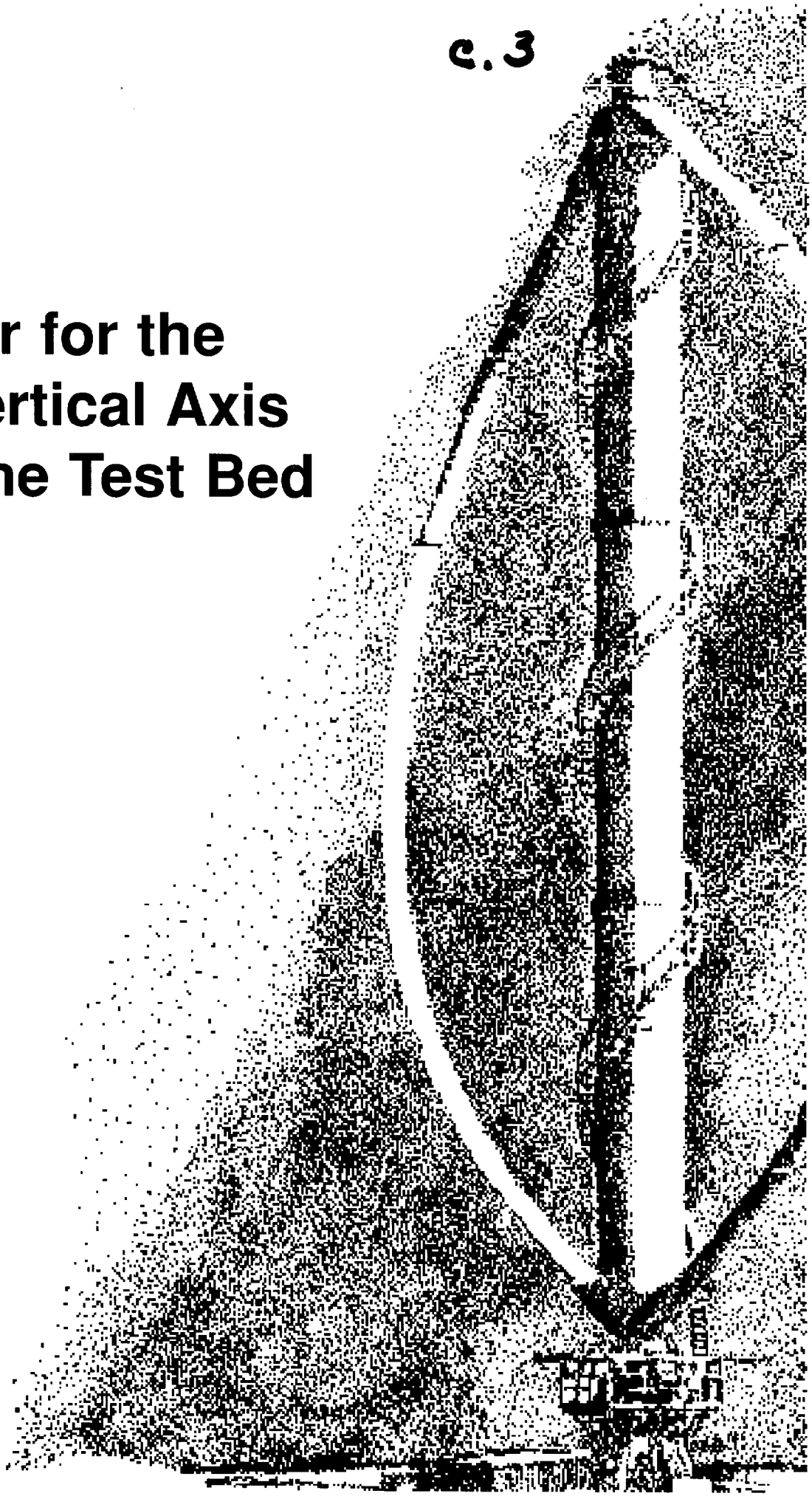
SAND90-0116
Unlimited Release
Printed April 1990

c.3

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



SNLA LIBRARY
* 8 2 1 3 7 3 6 *
SAND90-0116
0003
UNCLASSIFIED
04/90
63P STAC

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-ACO4-76DP00789

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from
Office of Scientific and Technical Information
PO Box 62
Oak Ridge, TN 37831

Prices available from (615) 576-8401, FTS 626-8401

Available to the public from
National Technical Information Service
US Department of Commerce
5285 Port Royal Rd
Springfield, VA 22161

NTIS price codes
Printed copy: A02
Microfiche copy: A01

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 HP 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 and 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 1 ANALOG-TO-DIGITAL CONVERTER SPECIFICATIONS

ADC GAIN	1	2	3	4
FULL SCALE RANGE	+/-10.24V	+/-5.12V	+/-2.56V	+/-1.28V
RESOLUTION	5mV	2.5mV	1.25mV	0.625mV
TEMPERATURE COEF	+/-0.38mV/C	+/-0.38mV/C	+/-0.095mV/C	+/-0.048mV/C

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.

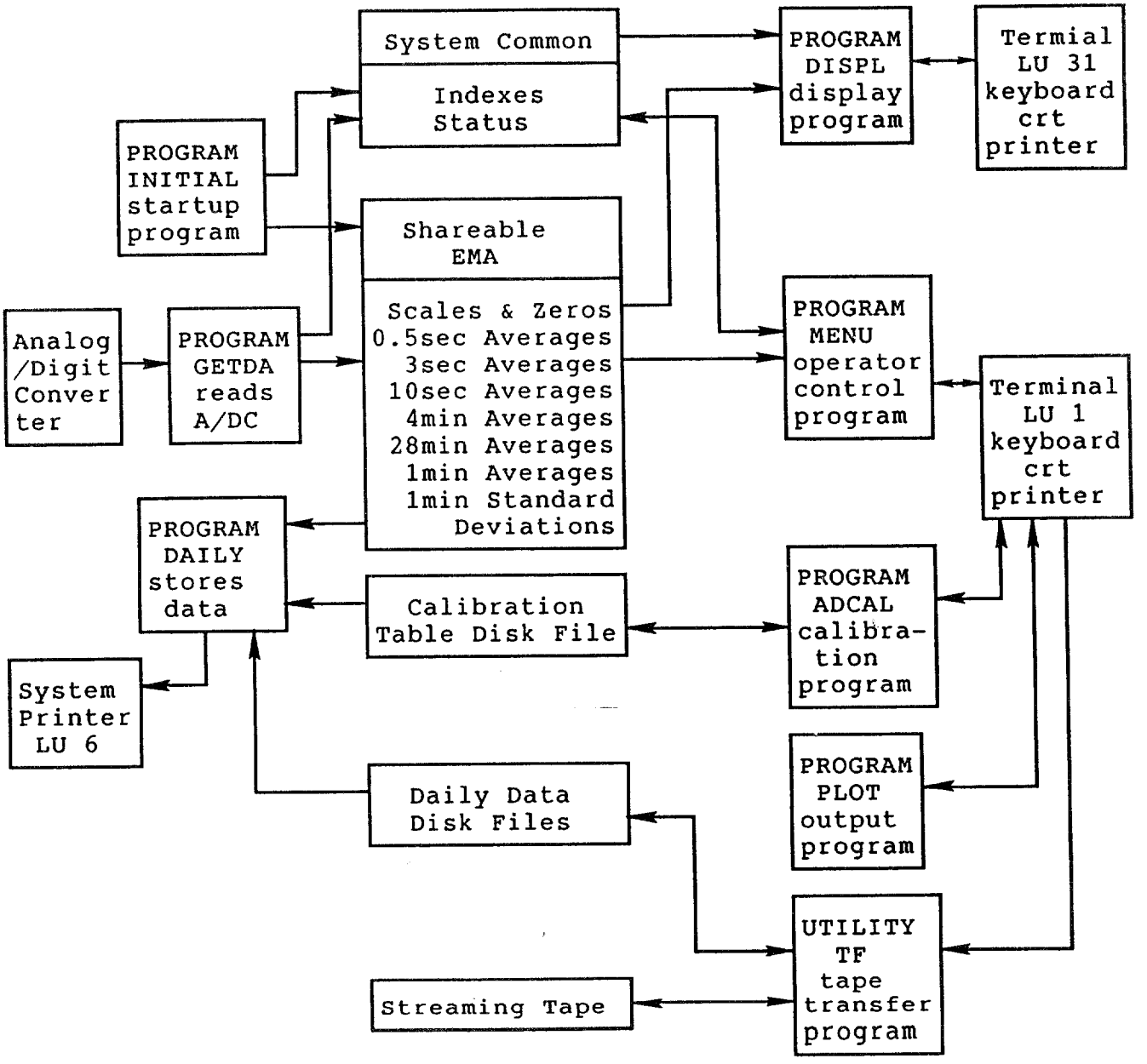
The programs INITIAL, GETDA, DAILY, DISPL, and MENU communicate with each other by utilizing system common for program and data parameters and sharable extended memory area (EMA) for large data arrays. EMA allows data arrays larger than 512 words (1/2 kiloword) to be used. The data logger requires more 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



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 2 FILE HEADER STRUCTURE

Integer Address	Real Address	
1	1	Zero offsets for channel 1
2		stored as real number
3	2	Zero offsets for channel 2
4		stored as real number
.	.	.
.	.	.
.	.	.
97	49	Zero offsets for channel 49
98		stored as real number
99	50	Zero offsets for channel 50
100		stored as real number
101	51	
.	.	
.	.	Not used
.	.	
129	65	Scale factor for channel 1
130		stored as real number
131	66	Scale factor for channel 2
132		stored as real number
.	.	.
.	.	.
.	.	.
225	113	Scale factor for channel 49
226		stored as real number
227	114	Scale factor for channel 50
228		stored as real number
229	115	
.	.	
.	.	Not used
.	.	
257	129	Units for channel 1 stored
258		as 2 integers
259	130	Units for channel 2 stored
260		as 2 integers
.	.	.
.	.	.
.	.	.
353	177	Units for channel 49 stored
354		as 2 integers
355	178	Units for channel 50 stored
356		as 2 integers

Integer Address	Real Address	
357	179	
358		
.	.	Not used
.	.	
.	.	
385	193	Month - stored as integer
386		Day - stored as integer
387	194	Year - stored as an integer
388		Location - stored as an integer
389	195	
.		Not used
.	.	
.	.	
405	203	Length of record in sectors (integer)
406		Number of channels (50 integer)
407	204	1st channel stored (1 integer)
408		2nd channel stored (2 integer)
409	205	3rd channel stored (3 integer)
.		.
.	.	.
.	.	.
430	216	24th channel stored (24 integer)
431		25th channel stored (25 integer)
432	217	26th channel stored (ch 1 std dev)
433		27th channel stored (ch 2 std dev)
.	.	.
.	.	.
.	.	.
455		49th channel stored (ch 24 std dev)
456	229	50th channel stored (ch 25 std dev)
458		
.	.	Not Used
.	.	
.	.	
471	236	Sample rate (2 Hz)
472		stored as real
473	237	Length of record in seconds
474		(86400 - real)
.	.	
.	.	
.	.	Not Used
511	256	
512		

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

Akins, R.E., 1978, *Wind Characteristics at the VAWT Test Facility*, SAND78-0760, Sandia National Laboratories, Albuquerque, N.M., September 1978.

Akins, R.E., D.E. Berg, W.T. Cyrus, 1987, *Measurements and Calculations of Aerodynamic Torques for a Vertical Axis Wind Turbine*, SAND86-2164, Sandia National Laboratories, Albuquerque, N.M., October 1987.

Berg, D.E., M.A. Rumsey, L.R. Gallo, D.P. Burwinkle, 1988, "Data Acquisition and Analysis System for the Sandia 34-meter Vertical Axis Wind Turbine," *Seventh ASME Wind Energy Symposium*, The American Society of Mechanical Engineers, January 1988.

Hewlett-Packard Co., 1984, Data Systems Division, *RTE-A Operating System Manual Set*, Hewlett-Packard Company, Cupertino, Calif., February 1984.

Hewlett-Packard Co., 1985, Networks Division, *Specification Guide*, Hewlett-Packard Company, Roseville, Calif., August 1985.

Stephenson, W.A., 1986, *Test Plan for the 34 Meter Vertical Axis Wind Turbine Test Bed*, SAND86-1623, Sandia National Laboratories, Albuquerque, N.M., December 1986.

Stiefeld, B., 1978, *Wind Turbine Data Acquisition and Analysis Systems*, SAND77-1164, Sandia National Laboratories, Albuquerque, N.M., July 1978.

Appendix A

DATA LOGGER DISPLAYS AND PLOTS

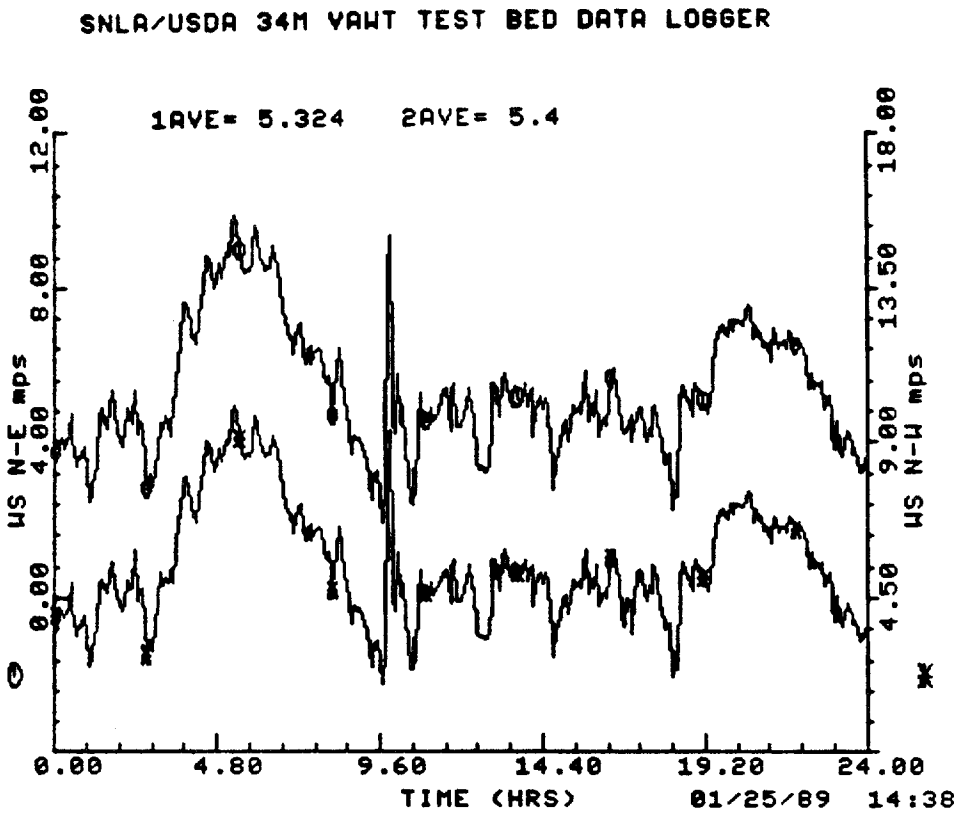
Typical data display screens:

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

WIND        Speed     Direction                COEF OF PERFORMANCE
 10 m      7.4  mps 258.5  deg W               Equator:   Aero          System
 30 m      7.5  mps 253.5  deg W               10 metre   Aero          System
 NE        8.4  mps 247.6  deg SW
 NW       12.3  mps                            POWER
 SE       12.5  mps 185.6  deg S               Electrical    189.6 kW   185.4 kVAr
 SW       12.7  mps                            Rotor         228.2 kW

CONTROLLER                                     TURBINE
4.18 Volts    .75 Volts                        Speed          22.4 rpm
                                               Torque         97.5 kNm
                                               Tiedown Cable 574.1 kN

TranAxialVibra        .17 in/s    .19
StandVibration        .09 in/s    .09          AMBIENT CONDITIONS
strainGauge1AML     -2.64 MPa    4.00          Temperature 10 metre 299.3  C
straingauge1AMf    -29.16 MPa   31.45          Temperature 48 metre  -2.5  C
straingauge1QML    -17.71 MPa   16.94          Barometric Pressure   127.1 kPa
StrainGauge1QMF     19.31 MPa   26.43           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

WIND        Speed     Direction                COEF OF PERFORMANCE
 10 m     13.2  mps 187.2  deg S               Equator:   Aero          System
 30 m     14.4  mps 191.1  deg S               10 metre   Aero          System
 NE       12.9  mps 183.5  deg S
 NW       12.9  mps                            POWER
 SE       14.6  mps 189.0  deg S               Electrical    494.4 kW   511.8 kVAr
 SW       14.7  mps                            Rotor         478.6 kW

CONTROLLER                                     TURBINE
 .40 Volts  1.91 Volts                         Speed          34.2 rpm
                                               Torque        133.5 kNm
                                               Tiedown Cable 868.0 kN

TranAxialVibra        .21 in/s    .01
StandVibration        .01 in/s    .00          AMBIENT CONDITIONS
strainGauge1AML     -7.53 MPa    3.58          Temperature 10 metre  -2.5  C
straingauge1AMf    -41.60 MPa    7.26          Temperature 48 metre  -2.9  C
straingauge1QML     -7.51 MPa    2.50          Barometric Pressure   89.39 kPa
StrainGauge1QMF     31.27 MPa    9.22           9:06 AM  THU., 23  FEB., 1989
PRESS ANY KEY TO RETURN TO THE MAIN MENU
```

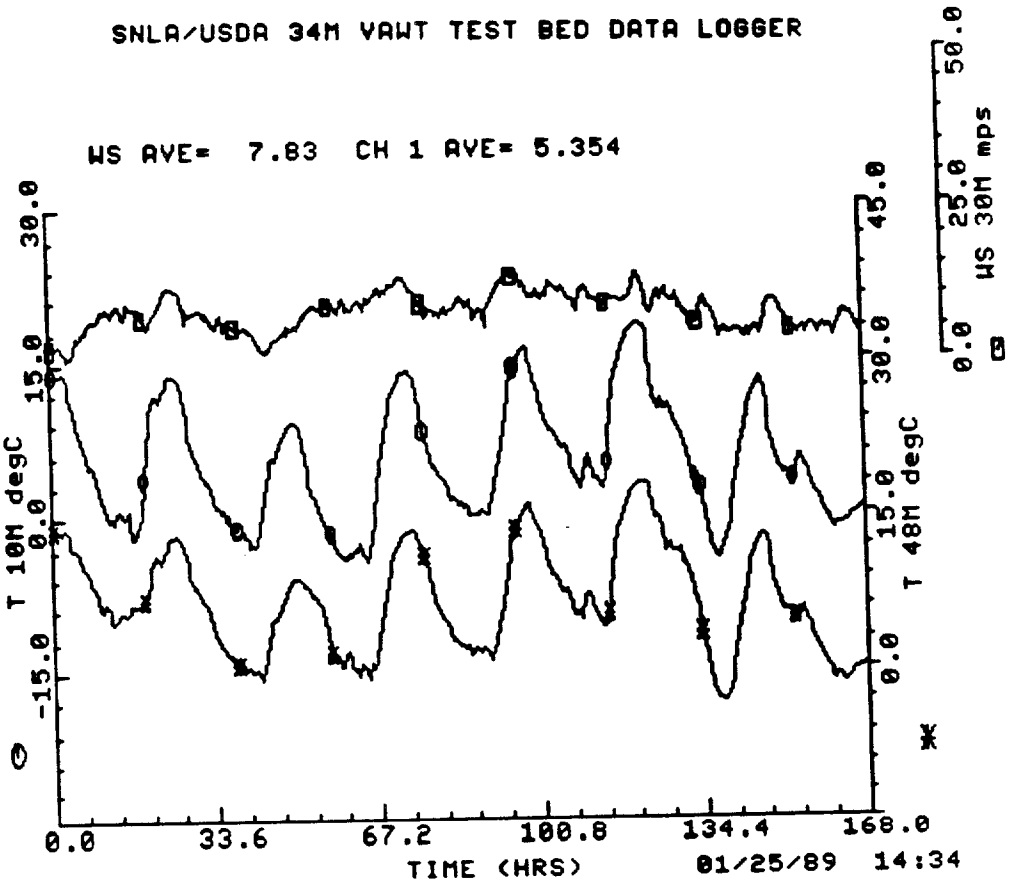
DATA LOGGER DISPLAYS AND PLOTS

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.

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

lu	device name	select code	hpib address	device status
1	terminal	23		busy with class request
2	not assigned			
3	not assigned			
4	not assigned			
5	not assigned			
6	printer	31	1	up
7	not assigned			
8	not assigned			
9	not assigned			
10	device type = 45b	33		up
11	not assigned			
12	not assigned			
13	not assigned			
14	not assigned			
15	not assigned			
16	CS 80 disc	27	2	up
17	CS 80 disc	27	2	up
18	not assigned			
19	not assigned			
20	CS 80 disc	27	2	up
21	floppy disc	27	3	up
22	floppy disc	27	3	up
23	floppy disc	27	3	up
24	CS 80 tape drive	27	2	up
25	not assigned			
26	not assigned			
27	not assigned			
28	not assigned			
29	not assigned			
30	not assigned			
31	terminal	23		up
32	terminal	23		up
33	terminal	23		up
34	terminal	23		up
35	terminal	23		up
36	terminal	23		up
37	terminal	23		up

DATA LOGGER OPERATOR NOTES

Analog Input Card Wiring Details

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

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
```

Appendix D

FORTRAN SOURCE CODES

```
C*************************************************************
C
$EMA/DATA/
      SUBROUTINE CHECKWS
      INTEGER IHOME
      CHARACTER BELL
C
      INCLUDE EMA&SC
C
      IHOME=15400B+150B
      BELL=CHAR(7)
C
      WRITE(1,101) IHOME
      CALL CHCKLU(6)                    ! PRINTER
      CALL CHCKLU(10)                   ! A/D
      CALL CHCKLU(20)                   ! DISK
C
      WSNE=X10S(110S-1,6)
      WSNW=X10S(110S-1,7)
      WSSE=X10S(110S-1,8)

      WSSW=X10S(110S-1,9)
C
      IF( ABS(WSNE-WSNW).GT.1.5 )THEN
          WRITE(1,110)
          COUNTN=COUNTN+1
      ELSE
          WRITE(1,130)
          COUNTN=COUNTN-1
          IF( COUNTN.LT.0 )THEN
              COUNTN=0
              END IF
          END IF
C
      IF( ABS(WSSE-WSSW).GT.1.5 )THEN
          WRITE(1,120)
          COUNTS=COUNTS+1
      ELSE
          WRITE(1,130)
          COUNTS=COUNTS-1
          IF( COUNTS.LT.0 )THEN
              COUNTS=0
              END IF
          END IF
      IF( COUNTN.EQ.15 )THEN
          WRITE(6,110)
          END IF
      IF( COUNTS.EQ.15 )THEN
          WRITE(6,120)
          END IF
      RETURN
  101 FORMAT(A2,/)
  110 FORMAT('     North anemometers do not agree          ')
  120 FORMAT('     South anemometers do not agree          ')
  130 FORMAT('                                             ')
      END
C
C*************************************************************
C
      SUBROUTINE CHCKLU(LUC)
C
      INTEGER LUC,STAT1,STAT,BIT14
C
      BIT14=40000B
      CALL EXEC(13,LUC,STAT1)
      STAT=STAT1.AND.BIT14
      IF( STAT.NE.0 )THEN
          WRITE(1,100) LUC,LUC
          END IF
      RETURN
  100 FORMAT('LU',I2,' Down, Try UP,',I2,' at CI> promt')
      END
```

```
isting DAILY.FTN:::4:6:39

EMA/DATA/
      PROGRAM DAILY
C
C      LOCAL VARIABLES
C
      REAL RHEADER(256),AVE(20)
      REAL RBLOCK(64,10),ZEROS(128),SCALES(128)
      REAL TEMPR
      REAL PER                          ! CONSTANT FOR CONVERTING # PTS TO %
      INTEGER*2 HEIGHT(2)               ! HEIGHTS OF METERS FOR PRINTOUT
      INTEGER WDCH(4)                   ! WIND DIRECTION CHANNEL NUMBERS
      INTEGER DCB(144)                  ! DCB CONTROL BLOCK FOR Fmp VMA FILES
      INTEGER DAILY(3)                  ! PROGRAM NAME
      INTEGER IBLOCK(128,10)            ! CALIBRATION TABLE
      INTEGER LEN                       ! LENGTH OF FILE READ
      INTEGER FmpOpen,FmpRead           ! Fmp ROUTINES CALLED AS INTEGER FUN.
      INTEGER BUF(15)                   ! BUFFER FOR RETREVING DATE
      REAL    DIRECTION(7,8)            ! WIND DIREC.,# PTS:N,NE,E,SE,S,SW,W,NW
      INTEGER JULIAN(5)                 ! JULIAN(5) JULIAN DAY OF YEAR
      INTEGER FORMFEED                  ! FORM FEED
      CHARACTER*64 CALTBL               ! DISK FILE NAME FOR CALIBRATION FILE

C
      INCLUDE EMA&SC
C
      EQUIVALENCE (IBLOCK,RBLOCK)
      EQUIVALENCE (IBLOCK(1,1),ZEROS(1))
      EQUIVALENCE (IBLOCK(1,3),SCALES(1))
C
      DATA DAILY /'DAILY '/
      DATA HEIGHT/2H10,2H30/
      DATA WDCH/10,12,15,16/
C
      LUP = 6                                       ! LU FOR PRINTOUT
      FORMFEED = 006000B
      WRITE(LUP,*) 'DAILY REPORT, ',NAME
C
C***************** STORE DATA IN TEMP ARRAYS ************************
      DO 10 K=1,1440
          DO 20 J=1,20
              ITP1(K,J)=IDATA(K,J)
              ITP2(K,J)=IDATA(K,J+20)
              IDATA(K,J)=0
              IDATA(K,J+20)=0
   20     CONTINUE
          DO J=1,10
              ITP3(K,J)=IDATA(K,J+40)
              IDATA(K,J+40)=0
          END DO
   10 CONTINUE
      NUMBER = COUNT
      COUNT = 0
C
C***************** WRITE FILE TO DISK ******************************
      CALL BLDHDER(NAME,IHEADER,NUMBER)
      CALL FmpOpen(DCB,IERR,FILENAME,'WC',1)
      IF (IERR .LT. 0) THEN
          CALL FmpOpen(DCB,IERR1,FILENAME,'WO',1)
          IF (IERR1 .LT. 0) THEN
             CALL FmpReportError(IERR1,DAILY)
             WRITE(LUP,*) 'DATA FILE NOT OPENed'
             GOTO 999
          END IF
      END IF
C
C***************** WRITE DATA TO DISK ******************************
      CALL FmpWrite(DCB,IERR,IHEADER,2*512)
      IF (IERR .LT. 0) THEN
          WRITE(LUP,*) 'HEADER NOT WRITTEN'
          CALL FmpReportError(IERR,DAILY)
          GOTO 999
      END IF
      CALL VWrit(DCB,IERR,ITP1,28800)
      CALL VWrit(DCB,IERR,ITP2,28800)
      CALL VWrit(DCB,IERR,ITP3,14400)
      IF (IERR .LT. 0) THEN
          WRITE(LUP,*) 'DATA NOT WRITTEN'
          CALL FmpReportError(IERR,DAILY)
          GOTO 999
      END IF
      CALL FmpClose(DCB,IERR)
      IF (IERR .LT. 0) THEN
          WRITE(LUP,*) 'ERROR IN CLOSING DATA FILE'
          CALL FmpReportError(IERR,DAILY)
      END IF
  999 CONTINUE
C
C*****************  DAILY REPORT ************************************
      IF (NUMBER.EQ.0) GOTO 160
      DO JT=1,20
         AVE(JT)=0.0
         END DO
      DO JC=1,4
          DO JT=1,8
             DIRECTION(JC,JT)=0.0
             END DO
          END DO
```

```
      DO JI=1,360
          AVE(1)=AVE(1)+X4M(JI,1)/NUMBER        ! WS
          AVE(2)=AVE(2)+X4M(JI,3)/NUMBER        ! WS
          AVE(3)=AVE(3)+X4M(JI,6)/NUMBER        ! WS
          AVE(4)=AVE(4)+X4M(JI,7)/NUMBER        ! WS
          AVE(5)=AVE(5)+X4M(JI,8)/NUMBER        ! WS
          AVE(6)=AVE(6)+X4M(JI,9)/NUMBER        ! WS
          AVE(13)=AVE(13)+X4M(JI,21)/NUMBER       ! TEMPO 10M
          AVE(14)=AVE(14)+X4M(JI,22)/NUMBER       ! TEMPO 48M
          AVE(15)=AVE(15)+X4M(JI,13)/NUMBER       ! BARO PRESS
          PWR=X4M(JI,17)*24/NUMBER
          IF(PWR.LT.0.1) THEN
              PWR=0.0
              END IF
          AVE(16)=AVE(16)+PWR                     ! DAILY AVE kW * 24 hr
          DO JC=1,4
              JT=WDCH(JC)
              IF (X4M(JI,JT).LE.23.OR.X4M(JI,JT).GT.338)THEN
                  DIRECTION(JC,1)=DIRECTION(JC,1)+1
              ELSEIF (X4M(JI,JT).LE.68.0) THEN
                  DIRECTION(JC,2)=DIRECTION(JC,2)+1
              ELSEIF (X4M(JI,JT).LE.113.0) THEN
                  DIRECTION(JC,3)=DIRECTION(JC,3)+1
              ELSEIF (X4M(JI,JT).LE.158.0) THEN
                  DIRECTION(JC,4)=DIRECTION(JC,4)+1
              ELSEIF (X4M(JI,JT).LE.203.0) THEN
                  DIRECTION(JC,5)=DIRECTION(JC,5)+1
              ELSEIF (X4M(JI,JT).LE.248.0) THEN
                  DIRECTION(JC,6)=DIRECTION(JC,6)+1
              ELSEIF (X4M(JI,JT).LE.293.0) THEN
                  DIRECTION(JC,7)=DIRECTION(JC,7)+1
              ELSE
                  DIRECTION(JC,8)=DIRECTION(JC,8)+1
                  ENDIF
              END DO
          END DO
      DO JC=1,4
          DIRECTION(JC,1)=DIRECTION(JC,1)-(360-NUMBER)
          END DO
      PER=100.0/FLOAT(NUMBER)
      DO JC=1,4
          DO JT=1,8
              TEMPR=DIRECTION(JC,JT)
              DIRECTION(JC,JT)=TEMPR*PER
C                 WRITE(6,52) JC,JT,PER,TEMPR,DIRECTION(JC,JT)
C  52             FORMAT('JC=',I2,' JT=',I2,' PER=',F10.4,' TEMPR=',
C    1                  F7.2,' DIRECTION =',F7.2)
              END DO
          END DO
      WRITE (LUP,100) (HEIGHT(JT),AVE(JT),JT=1,2)
  100 FORMAT('Average wind speed ',A2,'m   = ',F4.1,'mps')
      WRITE (LUP,110) AVE(3),AVE(4),AVE(5),AVE(6)
  110 FORMAT('Average wind speed North = ',2(F4.1,'mps  ')/,
     1       'Average wind speed South = ',2(F4.1,'mps  '))
      WRITE (LUP,120)((DIRECTION(JC,JT),JC=1,4),JT=1,8)
  120 FORMAT('Wind direction:  10metre  30metre   North    South'/,
     2       '    North        ',4(F5.1,'%   ')/,
     3       '    Northeast    ',4(F5.1,'%   ')/,
     4       '    East         ',4(F5.1,'%   ')/,
     5       '    Southeast    ',4(F5.1,'%   ')/,
     6       '    South        ',4(F5.1,'%   ')/,
     7       '    Southwest    ',4(F5.1,'%   ')/,
     8       '    West         ',4(F5.1,'%   ')/,
     9       '    Northwest    ',4(F5.1,'%   '))
      WRITE (LUP,130) AVE(13),AVE(14),AVE(15),AVE(16)
  130 FORMAT('Average temperature 10m = ',F4.1,'deg C'/
     1       '                    48m = ',F4.1,'deg C'/
     1       'Average barometric pressure = ',F4.1,'kPa'/
     3       'Electric power generated = ',F4.1,'kWh')
  160 IF(NUMBER.LT.360) THEN
          WRITE(LUP,140) NUMBER
  140     FORMAT('Only ',I3,' four minute averages were taken'/,
     1          '360 four minute averages available')
      END IF
      WRITE(LUP,150)
  150 FORMAT(//)
C
C***************** UPDATE CALIBRATION FACTORS ************************
      CALL FmpBuildName(CALTBL,'CALTBL','DAT',0,'DATA')
      IF( FmpOpen(DCB,IERR,CALTBL,'ROQ',1) .LT.0) THEN
          IF( IERR .EQ. -6 ) THEN
              WRITE(LUP,*) 'CALIBRATION TABLE DOES NOT EXIST'
          ELSE
              CALL FmpReportError(IERR,CALTBL)
              END IF
          WRITE(LUP,*) 'CALIBRATION FACTORS NOT UPDATED'
          GOTO 9999
          END IF
      LEN = FmpRead(DCB,IERR,IBLOCK,128*10*2)
      IF( IERR .LT. 0) THEN
          CALL FmpReportError( IERR,CALTBL)
          WRITE(LUP,*) 'CALIBRATION FACTORS NOT UPDATED'
          GOTO 9999
      ELSE IF( LEN .NE. 128*10*2 ) THEN
           WRITE(LUP,*) 'COULD NOT READ EXPECTED AMOUNT OF DATA'
           WRITE(LUP,*) 'FROM CALIBRATION TABLE.   PROGRAM STOPPED!'
           GOTO 9999
           END IF
      DO 70 K=1,25
          A(K)=SCALES(K)
```

```
          B(K)=ZEROS(K)
   70     CONTINUE

      CALL EXEC (11,JULIAN)
      IJULIAN = (JULIAN(5)/2)*2
      IF(JULIAN(5).NE.IJULIAN) THEN
          WRITE(LUP,5) FORMFEED
    5     FORMAT(A2)
          END IF
C
C***************** UPDATE DATE AND RESCHEDULE PROGRAM ****************
 9999 CALL DFILEN(FILENAME,NAME)  ! UPDATE FILENAMES FOR NEXT DAY
      CALL EXEC(12,DAILY,1,0,0,0)
      STOP
      END
```

```
I32> II,@.FTN

Listing DFILEN.FTN:::4:18:35

      SUBROUTINE DFILEN(FILEN,NAME)
C     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 FTIME(IBUF)
C***************** BUILD DATA FILE NAME ****************************
      NAME(1:6)='BUDLOG'
      NAME(7:9)=CBUF(21:23)
      IF( CBUF(17:17) .EQ. ' ') CBUF(17:17) = '0'
      NAME(10:11)=CBUF(17:18)
      NAME(12:13)=CBUF(29:30)
      SIZE=4+564
      CALL FmpBuildName(FILEN,NAME,'DAT',0,'DATA',1,SIZE)
      RETURN
      END
```

```
EMA/DATA/
      PROGRAM DISPL
C
C     LOCAL VARIABLES
C
      REAL PARRAY(363,3)                    ! DATA ARRAYS FOR PLOTTING
      REAL SI                               ! SECONDS PER POINT
      INTEGER IUN1(5)                       ! AXIS HEADER FOR WS
      INTEGER IUN2(5)                       ! AXIS HEADER FOR CH 2
      INTEGER IUN3(5)                       ! AXIS HEADER FOR CH 3
      INTEGER IUN(5,2)                      ! AXIS HEADER FOR CH 3
      INTEGER ITIM(5)                       ! AXIS HEADER FOR TIME
      INTEGER CNB(3)                        ! CHANNEL NUMBERS
      INTEGER J1CH(4)                       ! W DIRC CHANNEL NUMBERS
      INTEGER NCH                           ! NUMBER OF CH (1 OR 2)
      INTEGER WS                            ! WS >0 THEN PLOT WS
      INTEGER CLNB                          ! CLASS NUMBER
      INTEGER PARM1,PARM2                   ! DUMMY PARM
      INTEGER WD(7)                         ! WIND DIRECTION DESIGNATION
      INTEGER BUF(15)
      INTEGER INDX,IST(5),STDE

      INTEGER LUDD                          ! DISPLAY LU NOT RESETABLE BY MENU
      REAL SCALE(32),ARRAY(32),DV(32),CP(4)
      CHARACTER BELL
      CHARACTER*4 UWS,UTP,UBP,USG
      CHARACTER*6 UTQ
C
      INCLUDE EMA&SC
C
      DATA IUN1 /2HWS,2H  ,2H  ,2H m,2Hps/
      DATA ITIM /2HTI,2HME,2H (,2H  ,2H  /
      DATA J1CH /10,12,15,16/
C
      BELL  = CHAR(7)
      LUDD  = 31
      IMEN1 = 15400B+046B            ! TURN OFF MENU PT1  Esc &
      IMEN2 = 65000B+100B            ! TURN OFF MENU PT2  j @
      IFUN1 = 15400B+046B            ! DISABLE FUNCTION KEYS PT1  Esc &
      IFUN2 = 65000B+123B            ! DISABLE FUNCTION KEYS PT2  j S
      IHOME = 15400B+150B            ! HOME ALPHA CURSER  Esc h
      IRASE = 15400B+112B            ! CLEAR ALPHA DISPL  Esc J
      IGRP1 = 15400B+052B            ! TURN OFF GRAPH DISPL PT1  Esc *
      IGRP2 = 62000B+144B            ! TURN OFF GRAPH DISPL PT2  e d
      STDE  = 2                      ! SELECT 10sec STD DEV
C
      DO 15 J=1,9
         ARRAY(J)= 2.24                 ! mph/mps
   15    CONTINUE
      DO J=10,32
         ARRAY(J)=1.0
         END DO
      ARRAY(13)=0.296                ! inHg/kPa
      ARRAY(19)=0.7376               ! ft-lb/Nm
      ARRAY(20)=0.2248               ! lb/N
      CLNB=0
      CALL CLRQ(1,CLNB)
    5 CONTINUE
      WRITE(LUD,7) IHOME,IRASE
      WRITE(LUD,7) IGRP1,IGRP2
      IF( LUDD.EQ.LUD ) THEN
         WRITE(LUDD,7) IMEN1,IMEN2
         WRITE(LUDD,7) IFUN1,IFUN2
         END IF
    7 FORMAT(2A2)
   10 WRITE(LUD,200) IHOME
  200 FORMAT(//A2,
     1    17X,'        SANDIA NATIONAL LABORATORIES          '/
     2    17X,'       34 M VAWT TEST BED DATA LOGGER         ',///,
     3    20X,'1   DISPLAY TURBINE STATUS   S.I. UNITS',//,
     4    20X,'2   DISPLAY TURBINE STATUS   B.E. UNITS',//,
     5    20X,'3   PLOT DATA',//////
     6    20X, 'ENTER NUMBER     ')
      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                    ! TEMP CHANNELS
            ELSE
               SCALE(J)=1.0
               ENDIF
            END DO
         UWS=' mps'
         UTQ='kNm   '
         UTP='C   '
         UBP='kPa '
         USG='kN  '
         WRITE(LUD,7) IHOME,IRASE
         IC=0
         GOTO 1000
      ELSEIF (INPUT.EQ.2) THEN
         DO J=1,32
            SCALE(J)=ARRAY(J)
            END DO
         UWS=' mph'
         UTQ='kft-lb'
```

```
          UTP='F   '
          UBP='inHg'
          USG='klb '
          WRITE(LUD,7) IHOME,IRASE
          IC=0
          GOTO 1000
      ELSEIF (INPUT.EQ.3) THEN
          GOTO 2000
          END IF
   20 CONTINUE
      WRITE(LUD,400) BELL
  400 FORMAT(A2,'  INVALID ENTRY  TRY AGAIN')
      GOTO 10
 1000 CONTINUE
      INPUT=0
      INDEX=I10S-1
      IF( IC.GE.10 ) THEN
          WRITE(LUD,7) IHOME,IRASE
          IC=0
      ELSE
          IC=IC+1
      END IF
      WRITE(LUD,1001) IHOME
 1001 FORMAT(A2,' ',15X,'SNLA/USDA 34 METRE VAWT TEST BED DATA ',
     1    'LOGGER',17X,//)
      DO J=1,32
          IF(J.EQ.21.OR.J.EQ.22) THEN
              DV(J)=X10S(INDEX,J)+SCALE(J)*(0.8*X10S(INDEX,J)+32)
          ELSE
              DV(J)=X10S(INDEX,J)*SCALE(J)
              ENDIF
          END DO
C*****************************************************************
      LUS=LUD
      INCLUDE SCREEN
C
      WRITE(LUD,1006)
 1006 FORMAT('PRESS ANY KEY TO RETURN TO THE MAIN MENU  ')
      CALL EXEC(17,100400B+LUD,INPUT,-1,PARM1,PARM2,CLNB+100000B)
 1100 CONTINUE
      CALL EXEC(21,CLNB+120000B,INPUT,-1)
      IF(INPUT.NE.0) THEN
            CALL CLRQ(3,CLNB,LUD)
          GOTO 5
      END IF
      IF(I10S-1 .NE. INDEX) THEN
          CALL CLRQ(3,CLNB,LUD)
          GOTO 1000
      END IF
      IBK=IFBRK()
      IF(IBK.NE.-1) GOTO 1100
      GOTO 9000
C****************** PLOT DATA ****************************************
 2000 CONTINUE
      CALL CHANNEL(CNB,NCH,WS,IUN1,IUN,LUD,CHNM,CHUNIT)
      DO JK =1,5
          IUN2(JK)=IUN(JK,1)
          IUN3(JK)=IUN(JK,2)
          END DO
      CALL GETINDEX( IDX,SI,LUD )
      IST(1)=IHS+17
      IST(2)=I3S+18
      IST(3)=I10S+18
      IST(4)=I4M+18
      IST(5)=I28M+18
      ISTART=IST(IDX)
      DO J3=1,NCH+1
         IF( CNB(J3).LE.0 ) THEN
            CONTINUE                          ! WS NOT TO BE PLOTTED
         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)=XHS(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)=X3S(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)=X10S(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)=X4M(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)=X28M(J2,CNB(J3))
                  END DO
```

```
          ELSE
            WRITE(LUD,2003) IHOME,IRASE
2003        FORMAT(2A2,'ERROR  INVALID TIME INDEX NUMBER'/
     1                  '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+ISTART
                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+ISTART
                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+ISTART
                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+ISTART
                IF( J2.GT.380 ) J2=J2-380
                PARRAY(J1,J3)=X4M(J2,19)*X4M(J2,24)*0.1047
                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)=X28M(J2,19)*X28M(J2,24)*0.1047
                END DO
          ELSE
            WRITE(LUD,2003) IHOME,IRASE
            CALL INPUTVAL(IDUM,LUD)
            GOTO 2000
            END IF
        ELSE
          WRITE(LUD,2004) IHOME,IRASE
2004      FORMAT(2A2,'ERROR  INVALID CHANNEL NUMBER'/
     1                'PRESS ANY KEY TO CONTINUE')
          CALL INPUTVAL(IDUM,LUD)
          GOTO 2000
          END IF
        END DO
      ITIM(4) = 2HSE
      ITIM(5) = 2HC)
C
      CALL PLUT(PARRAY,LUD,WS,NCH,IUN1,IUN2,IUN3,ITIM,CHB,SI)
      GOTO 5
C************************** PROGRAM STOPPED **************************
 9000 CONTINUE
      WRITE(LUD,9010) IHOME,IRASE
 9010 FORMAT(2A2)
      STOP
  999 END
C
C******************* SUBROUTINE INPUTVAL ************************
C   SUBROUTINE RETURNS AN INTEGER ENTERED FROM THE KEY BOARD
C
      SUBROUTINE INPUTVAL(R,LUD)
      CHARACTER SEL
      INTEGER C,R
C
      ILU=LUD+100B
      CALL EXEC( 1, ILU , C , -1 )
      SEL = CHAR( C/256 )
      R = ICHAR(SEL)-ICHAR('0')
      RETURN
      END
C
C **************** SUBROUTINE GETINDEX ************************
C  SUBROUTINE PROMPTS THE TERMINAL FOR AN INDEX FOR PLOTTING
C
      SUBROUTINE GETINDEX( INDX,SI,LUD )
C
      CHARACTER BELL,ESC
      CHARACTER*2 HOME,CLS
      INTEGER INDX
C
      BELL=CHAR(7)
      ESC=CHAR(27)
      HOME=ESC // 'h'
      CLS=ESC // 'J'
C
      WRITE(LUD,*) HOME,CLS
 2001 FORMAT(2A2)
      WRITE(LUD,2002)
 2002 FORMAT(18X,'        SELECT LENGTH OF PLOT      '///
     1       20X,'1      LAST  3 MINUTES OF DATA'//
     2       20X,'2      LAST 18 MINUTES OF DATA'//
     3       20X,'3      LAST HOUR OF DATA'//
     4       20X,'4      LAST DAY OF DATA'//
     5       20X,'5      LAST WEEK OF DATA'////
```

```
    6       20X,'ENTER NUMBER  _')
 2100 CONTINUE
      CALL INPUTVAL( INDX ,LUD )
      IF(INDX.EQ.1)THEN
          SI = 0.5
      ELSE IF(INDX.EQ.2)THEN
          SI = 3.0
      ELSE IF(INDX.EQ.3)THEN
          SI = 10.0
      ELSE IF(INDX.EQ.4)THEN
          SI = 240
      ELSE IF(INDX.EQ.5)THEN
          SI = 1680
      ELSE
          WRITE(LUD,2003) BELL
 2003     FORMAT( A2,/,20X,'INVALID ENTRY    TRY AGAIN  _')
          GOTO 2100
      END IF
      RETURN
      END

C***************** SUBROUTINE CHANNEL *************************
      SUBROUTINE CHANNEL(CNB,NCH,WS,IUN1,IUN,LUD,CHNM,CHUNIT)
C
      INTEGER CNB(3),NCH,NU(2),WS,IUN1(5),IUN(5,2)
      INTEGER*4 CHNM(7,4),CHUNIT(7)
C
      CHARACTER BELL,ESC
      CHARACTER*2 HOME,CLS
      CHARACTER*4 CHART
      INTEGER INDX,IST(5)
C
      BELL=CHAR(7)
      ESC=CHAR(27)
      HOME=ESC // 'h'
      CLS=ESC // 'J'
C
      WRITE(LUD,80) HOME,CLS
   80 FORMAT(2A2,//////,
     1        'DO YOU WANT TO PLOT WIND SPEED ? (Y OR N  CR>) _')
      READ(LUD,81) IWPLT
   81 FORMAT(A1)
      IF( IWPLT.EQ. 1HY .OR. IWPLT.EQ. 1Hy )THEN
          WS = 1
          WRITE(LUD,82) HOME,CLS
   82     FORMAT(2A2)
   84     WRITE(LUD,85) HOME
   85     FORMAT ( A2,/18X,
     1      '      SELECT WIND SPEED VARIABLE TO PLOT       '//
     2        20X,'1    WIND SPEED - 10M'//                    ! 1
     4        20X,'2    WIND SPEED - 30M,EQUATOR'//            ! 3
     7        20X,'3    WIND SPEED - NORTH-E'//                ! 6
     8        20X,'4    WIND SPEED - NORTH-W'//                ! 7
     9        20X,'5    WIND SPEED - SOUTH-E'//                ! 8
     1        20X,'6    WIND SPEED - SOUTH-W'//)               ! 9
          WRITE(LUD,88)
   88     FORMAT(/20X,'ENTER NUMBER  _')
          CALL INPUTVAL( NUB,LUD )
          IF( NUB.EQ.1 ) THEN
              CNB(1) = 1
              IUN1(2)=2H 1
              IUN1(3)=2HOM
          ELSE IF( NUB.EQ.2 ) THEN
              CNB(1) = 3
              IUN1(2)=2H 3
              IUN1(3)=2HOM
          ELSE IF( NUB.EQ.3 ) THEN
              CNB(1) =  6
              IUN1(2)=2H N
              IUN1(3)=2H-E
          ELSE IF( NUB.EQ.4 ) THEN
              CNB(1) = 7
              IUN1(2)=2H N
              IUN1(3)=2H-W
          ELSE IF( NUB.EQ.5 ) THEN
              CNB(1) = 8
              IUN1(2)=2H S
              IUN1(3)=2H-E
          ELSE IF( NUB.EQ.6 ) THEN
              CNB(1) = 9
              IUN1(2)=2H S
              IUN1(3)=2H-W
          ELSE
              WRITE(LUD,*) 'INVALID INPUT   TRY AGAIN'
              GOTO 84
              END IF
      ELSE
          WS = 0
          CNB(1) = 0
      END IF
      WRITE(LUD,90) HOME,CLS
   90 FORMAT(2A2,////,'PLOT 1 OR 2 ADDITIONAL CHANNELS ?_')
      CALL INPUTVAL(NCH,LUD)
      IF( NCH.NE.2 ) NCH = 1
      WRITE(LUD,95) HOME,CLS
   95 FORMAT(2A2)
  100 WRITE(LUD,101) HOME
  101 FORMAT(A2,4X,
     1   '  ' 8X,'        SELECT VARIABLE(S) TO PLOT',26X,' '//!CNB
     1   6X,'1  Power - Rotor      ',16X,'17  Wind Speed - 10m    '/!40,1
```

```
   2   6X,'2  Power - Electrical ',16X,'18  Wind Speed - 30m    '/!17,3
   3   6X,'3  Power - Reactive   ',16X,'19  Wind Speed - North-E'/!18,6
   4   6X,'4  Rotor Speed        ',16X,'20  Wind Speed - North-W'/!24,7
   5   6X,'5  Rotor Torque       ',16X,'21  Wind Speed - South-E'/!19,8
   6   6X,'6  Temperature - 10M  ',16X,'22  Wind Speed - South-W'/!21,9
   7   6X,'7  Temperature - 48M  ',16X,'23  Wind Direction - 10m'/!22,10
   8   6X,'8  Barometric Pressure',16X,'24  Wind Direction - 30m'/!13,12
   9   6X,'9  TieDown Number 1   ',16X,'25  Wind Direction-North')!20,15
     WRITE(LUD,102) ((CHNM(IJK,IJL),IJL=1,4),IJK=1,7)
102  FORMAT(5X,
   1      '10  ',4A,                  19X,'26  Wind Direction-South'/!26,16
   2  5X,'11  ',4A,                   19X,/                          !27
   3  5X,'12  ',4A,                   19X,/                          !28
   4  5X,'13  ',4A,                   19X,/                          !29
   5  5X,'14  ',4A,                   19X,/                          !30
   6  5X,'15  ',4A,                   19X,/                          !31
   7  5X,'16  ',4A,                   19X)                           !32
    DO J2=1,NCH
        WRITE(LUD,105)
105     FORMAT(16X,'ENTER CHANNEL NUMBER FOLLOWED BY CR>  _')
        READ(LUD,106,ERR=104) NU(J2)
106     FORMAT(I2)
    END DO
    DO 99 J2=2,NCH+1
        NUB=NU(J2-1)
        IF(NUB.EQ.1) THEN
            CNB(J2)=40
            IUN(1,J2-1)=2HRo
            IUN(2,J2-1)=2Hto
            IUN(3,J2-1)=2Hr
            IUN(4,J2-1)=2HkW
            IUN(5,J2-1)=2H
        ELSE IF(NUB.EQ.2) THEN
            CNB(J2)=17
            IUN(1,J2-1)=2HPo
            IUN(2,J2-1)=2Hwe
            IUN(3,J2-1)=2Hr
            IUN(4,J2-1)=2HkW
            IUN(5,J2-1)=2He
        ELSE IF(NUB.EQ.3) THEN
            CNB(J2)=18
            IUN(1,J2-1)=2HPo
            IUN(2,J2-1)=2Hwe
            IUN(3,J2-1)=2Hr
            IUN(4,J2-1)=2HkV
            IUN(5,J2-1)=2HAr
        ELSE IF(NUB.EQ.4) THEN
            CNB(J2)=24
            IUN(1,J2-1)=2HRo
            IUN(2,J2-1)=2Hto
            IUN(3,J2-1)=2Hr
            IUN(4,J2-1)=2HRP
            IUN(5,J2-1)=2HM
        ELSE IF(NUB.EQ.5) THEN
            CNB(J2)=19
            IUN(1,J2-1)=2HRo
            IUN(2,J2-1)=2Hto
            IUN(3,J2-1)=2Hr
            IUN(4,J2-1)=2HkN
            IUN(5,J2-1)=2Hm
        ELSE IF(NUB.EQ.6) THEN
            CNB(J2)=21
            IUN(1,J2-1)=2HT
            IUN(2,J2-1)=2H10
            IUN(3,J2-1)=2HM
            IUN(4,J2-1)=2Hde
            IUN(5,J2-1)=2HgC
        ELSE IF(NUB.EQ.7) THEN
            CNB(J2)=22
            IUN(1,J2-1)=2HT
            IUN(2,J2-1)=2H48
            IUN(3,J2-1)=2HM
            IUN(4,J2-1)=2Hde
            IUN(5,J2-1)=2HgC
        ELSE IF(NUB.EQ.8) THEN
            CNB(J2)=13
            IUN(1,J2-1)=2HBa
            IUN(2,J2-1)=2Hro
            IUN(3,J2-1)=2H P
            IUN(4,J2-1)=2H k
            IUN(5,J2-1)=2HPa
        ELSE IF(NUB.EQ.9) THEN
            CNB(J2)=20
            IUN(1,J2-1)=2HTi
            IUN(2,J2-1)=2HeD
            IUN(3,J2-1)=2How
            IUN(4,J2-1)=2H
            IUN(5,J2-1)=2HkN
        ELSE IF(NUB.EQ.10) THEN
            CNB(J2)=26
            CHART=CHAR(CHNM(1,1))
            IUN(1,J2-1)=ICHAR(CHART(1:2))
            IUN(2,J2-1)=ICHAR(CHART(3:4))
            IUN(3,J2-1)=2H
            IUN(4,J2-1)=2H
            IUN(5,J2-1)=2H
        ELSE IF(NUB.EQ.11) THEN
            CNB(J2)=27
            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.12) THEN
      CNB(J2)=28
      IUN(1,J2-1)=2H
      IUN(2,J2-1)=2H
      IUN(3,J2-1)=2H
      IUN(4,J2-1)=2H28
      IUN(5,J2-1)=2H
 ELSE IF(NUB.EQ.13) THEN
      CNB(J2)=29
      IUN(1,J2-1)=2H29
      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
      CNB(J2)=30
      IUN(1,J2-1)=2H30
      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
      CNB(J2)=31
      IUN(1,J2-1)=2H31
      IUN(2,J2-1)=2H
      IUN(3,J2-1)=2H
      IUN(4,J2-1)=2HO
      IUN(5,J2-1)=2HEQ
ELSE IF(NUB.EQ.16) THEN
      CNB(J2)=32
      IUN(1,J2-1)=2H32
      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
      CNB(J2)=1
      IUN(1,J2-1)=2HWS
      IUN(2,J2-1)=2H 1
      IUN(3,J2-1)=2HOM
      IUN(4,J2-1)=2H m
      IUN(5,J2-1)=2Hps
ELSE IF(NUB.EQ.18) THEN
      CNB(J2)=3
      IUN(1,J2-1)=2HWS
      IUN(2,J2-1)=2H 3
      IUN(3,J2-1)=2HOM
      IUN(4,J2-1)=2H m
      IUN(5,J2-1)=2Hps
ELSE IF(NUB.EQ.19) THEN
      CNB(J2)=6
      IUN(1,J2-1)=2HWS
      IUN(2,J2-1)=2H N
      IUN(3,J2-1)=2H-E
      IUN(4,J2-1)=2H m
      IUN(5,J2-1)=2Hps
ELSE IF(NUB.EQ.20) THEN
      CNB(J2)=7
      IUN(1,J2-1)=2HWS
      IUN(2,J2-1)=2H N
      IUN(3,J2-1)=2H-W
      IUN(4,J2-1)=2H m
      IUN(5,J2-1)=2Hps
ELSE IF(NUB.EQ.21) THEN
      CNB(J2)=8
      IUN(1,J2-1)=2HWS
      IUN(2,J2-1)=2H S
      IUN(3,J2-1)=2H-E
      IUN(4,J2-1)=2H m
      IUN(5,J2-1)=2Hps
ELSE IF(NUB.EQ.22) THEN
      CNB(J2)=9
      IUN(1,J2-1)=2HWS
      IUN(2,J2-1)=2H S
      IUN(3,J2-1)=2H-W
      IUN(4,J2-1)=2H m
      IUN(5,J2-1)=2Hps
ELSE IF(NUB.EQ.23) THEN
      CNB(J2)=10
      IUN(1,J2-1)=2HWD
      IUN(2,J2-1)=2H 1
      IUN(3,J2-1)=2HOM
      IUN(4,J2-1)=2H d
      IUN(5,J2-1)=2Heg
ELSE IF(NUB.EQ.24) THEN
      CNB(J2)=12
      IUN(1,J2-1)=2HWD
      IUN(2,J2-1)=2H 3
      IUN(3,J2-1)=2HOM
      IUN(4,J2-1)=2H d
      IUN(5,J2-1)=2Heg
ELSE IF(NUB.EQ.25) THEN
      CNB(J2)=15
      IUN(1,J2-1)=2HWD
      IUN(2,J2-1)=2HNo
      IUN(3,J2-1)=2Hrt
      IUN(4,J2-1)=2HhD
```

```
            IUN(5,J2-1)=2Heg
        ELSE IF(NUB.EQ.26) THEN
            CNB(J2)=16
            IUN(1,J2-1)=2HWD
            IUN(2,J2-1)=2HSo
            IUN(3,J2-1)=2Hut
            IUN(4,J2-1)=2HhD
            IUN(5,J2-1)=2Heg
        ELSE
104         WRITE(LUD,103) BELL
103         FORMAT(A2,35X,'INVALID ENTRY          TRY AGAIN')
            GOTO 100
        END IF
 99 CONTINUE
    RETURN
    END
```

```
isting GETDA.FTN:::4:3:36

EMA/DATA/
      PROGRAM GETDA
C
      INCLUDE EMA&SC
C
C***************** LOCAL VARIABLES **********************************
      INTEGER DAILY(3),GETDA(3),DISPL(3) ! PROGRAM NAMES
      INTEGER IX(40)                     ! DATA READ FROM A/D
      INTEGER DCB(144)                   ! DCB BUFFER
      INTEGER IHEADER(512)               ! DATA FILE HEADER
      INTEGER ERROR                      ! ERROR RETURN
      INTEGER FmpRpProgram               ! FMP CALLED AS INTEGER
      INTEGER ECODE12                    ! ECODE FOR EXEC 12 CALL
      INTEGER BUF(15)                    ! BUF FOR READING SYSTEM DATE
      REAL X                             ! CURRENT INPUT
      REAL XTEMP
      REAL SUM(32)                       ! SUM OF INPUTS FOR 1MIN
      REAL SUMSQ(25)                     ! SUM OF SQUARES FOR STD DEV
      REAL BUF4(4,32)                    ! BUFFER FOR FOUR MIN AVE
      REAL SQ3S(7),SQ10S(7)              ! SUM OF SQUARES FOR 3s& 10s
C                                          STD DEV FOR NON STORED

C
      DATA DAILY,GETDA,DISPL /'DAILY ','GETDA ','DISPL '/
      LUP = 6                                        ! LOGGING PRINTER
      DO J=1,25
          SUM(J)=0.0
          SUMSQ(J)=0.0
          END DO
      DO J=26,32
          SUM(J)=0.0
          END DO
          DO J=1,7
              SQ3S(J)=0.0
              SQ10S(J)=0.0
          END DO
C
C***************** READ DATA FROM A/D BOARD ****************
      ICWD=41000B+8              !      + 1ST WORD TO READ
      CALL EXEC(1,10,IX,32,ICWD)
      DO J=10,16
         OFFSET1M(J)=16000.-FLOAT(IX(J+6))
      ENDDO
    5 CONTINUE
C         1=READ,LU:10,ARRAY IX,32 WORDS READ, START CHANNEL 8
      CALL EXEC(1,10,IX,32,ICWD)
C***************** CALIBRATE DATA AND SUM ******************
      DO 10 J=1,25
          IXT=IX(J+6)
C         IF(STATUS.EQ.0) IXT=0
          XE=FLOAT(IXT)
          IF(J.GE.10.AND.J.LE.16) THEN
             IF(XE.GT.32000.-OFFSET1M(J))THEN
                  VAL=XE-32000.+OFFSET1M(J)
             ELSE IF(XE.LT.-OFFSET1M(J))THEN
                  VAL=XE+32000.+OFFSET1M(J)
             ELSE
                  VAL=XE+OFFSET1M(J)
                  END IF
          ELSE
             VAL=XE
             END IF
          SUM(J)=SUM(J)+VAL
          SUMSQ(J)=SUMSQ(J)+VAL**2
          XHS(IHS,J)=A(J)*XE+B(J)
   10     CONTINUE
      DO 20 J=2,7
          IXT=IX(J-1)
          XE=C(J)*FLOAT(IXT)+D(J)
          XHS(IHS,J+25)=XE
          SUM(J+25)=SUM(J+25)+XE
          SQ3S(J)=SQ3S(J)+XE**2
          SQ10S(J)=SQ10S(J)+XE**2
   20     CONTINUE
C            WRITE(31,11) (IX(J),J=1,7),IX(32)
C  11        FORMAT (8(I6,2X))
C            IXT=IX(1)
C            XE=FLOAT(IXT)
C            XHS(IHS,26)=C(1)*XE*D(1)
             XHS(IHS,26)=0
             SUM(26)=0
C
C***************** X3S = (SUM OF LAST 6 0.5 SEC VALUES)/6.0 *****
      IF (MOD(I,6).EQ.0) THEN
          IF(IHS.LT.6) THEN
              DO 30 J=1,32
                  XT=0.0
                  OFFSET=0
                  IF(J.GE.10.AND.J.LE.16) OFFSET=180-XHS(1,J)
                  DO 35 K=1,IHS
                     VAL=XHS(K,J)
                     CALL MAP(VAL,OFFSET)
                     XT=XT+VAL
   35                CONTINUE
                  DO 37 K=375+IHS,380
                     VAL=XHS(K,J)
                     CALL MAP(VAL,OFFSET)
```

```
                XT=XT+VAL
 37             CONTINUE
            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
 30         CONTINUE
      ELSE
         DO 40 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=IHS-5,IHS
               VAL=XHS(K,J)
               CALL MAP(VAL,OFFSET)
               XT=XT+VAL
 45            CONTINUE
            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
 40         CONTINUE
         END IF
      DO J=2,7
         XTEMP=SQ3S(J)/6.0-X3S(I3S,J+25)**2
         STDDEV(1,J)=(ABS(XTEMP))**0.5
         SQ3S(J)=0.0
         ENDDO
      I3S=I3S+1
      IF(I3S.EQ.381) I3S=1
      ENDIF
C
C***************** X10S = (SUM OF LAST 20 0.5 SEC VALUES)/20.0 ******
   IF(MOD(I,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 60 K=IHS-19,IHS
            VAL=XHS(K,J)
            CALL MAP(VAL,OFFSET)
            XT=XT+VAL
 60         CONTINUE
         X10S(I10S,J)=XT/20.0-OFFSET
         IF(J.GE.10.AND.J.LE.16) THEN
             X=X10S(I10S,J)
             CALL CHECK(X)
             X10S(I10S,J)=X
             END IF
 50      CONTINUE
      DO J=2,7
         XTEMP=SQ10S(J)/20.0-X10S(I10S,J+25)**2
         STDDEV(2,J)=(ABS(XTEMP))**0.5
         SQ10S(J)=0.0
         ENDDO
      I10S=I10S+1
      IF(I10S.EQ.381) I10S=1
C
C***************** 1MA = (SUM OF LAST 120 IX)/120 *********************
C***************** 1MSD = SQRT((SUM OF LAST 120 IX**2)/120-1MA**2) ***
      IF (I.EQ.120) THEN
         COUNT1=COUNT1+1
         DO 70 J=1,25
            SUMTP=SUM(J)/120.0-OFFSET1M(J)
            IF (SUMTP .GT. 32000) THEN
                SUMTP=0
c               SUMTP=SUMTP-32000
                END IF
            IF (SUMTP .LT. -32000) THEN
                SUMTP=0
c               SUMTP=SUMTP+32000
                END IF
            IF(STATUS.EQ.0) SUMTP=0
            IDATA(I1M,J)=INT(SUMTP)
            BUF4(COUNT1,J)=SUMTP
            X=SUMSQ(J)/120.0-SUMTP**2
            IF(X.LT.0) X=0.0
            X=SQRT(X)
            IF(STATUS.EQ.0) X=0
            IDATA(I1M,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
 70         CONTINUE
         DO J=26,32
            BUF4(COUNT1,J)=SUM(J)/120.0
            SUM(J)=0.0
            END DO
         I=0
         IF(COUNT1.GE.4) THEN
C***************** X4M = (SUM OF LAST 4 1MA)/4.0 ******** D-15 *
             DO 940 J=1,32
                XT=0.0
                IF(J.GE.10.AND.J.LE.16) THEN
```

```
                    ELSE
                        OFFSET=0
                        END IF
                    DO 950 K=1,4
                       VAL=BUF4(K,J)
                       CALL MAP(VAL,OFFSET)
                       XT=XT+VAL
 950                CONTINUE
                    XT=XT/4.0-OFFSET
                    X4M(I4M,J)=A(J)*XT+B(J)
                    IF(J.GE.10.AND.J.LE.16) THEN
                        X=X4M(I4M,J)
                        CALL CHECK(X)
                        X4M(I4M,J)=X
                       ENDIF
 940                CONTINUE
                COUNT=COUNT+1
                COUNT4=COUNT4+1
                COUNT1=0
C
C************* X28M = (SUM OF LAST 7 X4M)/7.0 *******************
                IF(COUNT4.GE.7) THEN
                    IF(I4M.LT.7) THEN
                        DO 960 J=1,32
                           XT=0.0
                           IF(J.GE.10.AND.J.LE.16) THEN
                               OFFSET=180-X4M(1,J)
                           ELSE
                               OFFSET=0
                               END IF
                           DO 970 K=1,I4M
                               VAL=X4M(K,J)
                               CALL MAP(VAL,OFFSET)
                               XT=XT+VAL
 970                           CONTINUE
                           DO 980 K=374+I4M,380
                               VAL=X4M(K,J)
                               CALL MAP(VAL,OFFSET)
                               XT=XT+VAL
 980                           CONTINUE
                           X28M(I28M,J)=XT/7.0-OFFSET
                           IF(J.GE.10.AND.J.LE.16) THEN
                               X=X28M(I28M,J)
                               CALL CHECK(X)
                               X28M(I28M,J)=X
                               ENDIF
 960                       CONTINUE
                    ELSE
                        DO 9100 J=1,32
                           XT=0.0
                           IF(J.GE.10.AND.J.LE.16) THEN
                               OFFSET=180-X4M(I4M-3,J)
                           ELSE
                               OFFSET=0
                               END IF
                           DO 990 K=I4M-6,I4M
                              VAL=X4M(K,J)
                              CALL MAP(VAL,OFFSET)
                              XT=XT+VAL
 990                          CONTINUE
                           X28M(I28M,J)=XT/7.0-OFFSET
                           X28M(I28M,J)=XT/7.0-OFFSET
                           IF(J.GE.10.AND.J.LE.16) THEN
                               X=X28M(I28M,J)
                               CALL CHECK(X)
                               X28M(I28M,J)=X
                               ENDIF
 9100                      CONTINUE
                        END IF
                    I28M=I28M+1
                    IF(I28M.EQ.381) I28M=1
                    COUNT4=0
                    ENDIF
                I4M=I4M+1
                IF(I4M.EQ.381) I4M=1
                END IF
            I1M=I1M+1
            IF(I1M .GT. 1440) I1M = 1
            ENDIF
        ENDIF
    IHS=IHS+1
    IF(IHS.EQ.381) IHS=1
    I=I+1
C
C********** MAKE PROGRAM DORMANT - SAVE RESOURCES UNTIL RESCHEDULED ****
    CALL EXEC(6,0,1)
C
C***************** CHECK STATUS ****************************************
    IF(STATUS.EQ.2) THEN                  ! STORE DATA AND STOP
        GOTO 90
    ELSE IF(STATUS.EQ.3) THEN             ! STOP PROGRAMS WITHOUT STO
        GOTO 999
        END IF
C
C***************** CHECK FOR BREAK ***************************
    IBK=IFBRK ()
    IF(IBK.NE.-1) GOTO 5              ! IF NO BREAK, READ NEW DATA
 90 CALL FTIME(BUF)
    IF(IBK.EQ.-1) THEN
```

```
          WRITE(LUP,80) (BUF(JI),JI=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 BLDHDER(NAME,IHEADER,COUNT)
      CALL FmpOpen(DCB,IERR,FILENAME,'WC',1)
      IF (IERR .LT. 0) THEN
          CALL FmpOpen(DCB,IERR1,FILENAME,'WO',1)
          IF (IERR1 .LT. 0) THEN
             CALL FmpReportError(IERR1,GETDA)
             WRITE(LUP,*) 'DATA FILE NOT OPENed'
             GOTO 999
              END IF
          END IF
C
C***************** WRITE DATA TO DISK ******************************
      CALL FmpWrite(DCB,IERR,IHEADER,2*512)
      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
          END DO
      CALL VWrit(DCB,IERR,ITP1,28800)
      CALL VWrit(DCB,IERR,ITP2,28800)
      CALL VWrit(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
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
          END IF
      RETURN
      END
C
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
```

```
isting HEADER.FTN:::4:3:34

      SUBROUTINE BLDHDER(NAME,IHDR,COUNT)

C         SUBROUTINE BUILDS HEADER, NAME IS PASSED TO THE
C                 SUBROUTINE AND RHDER IS RETURNED
C
      INTEGER IHEADER(512),DCB(16),IERR(10),IBLOCK(128,10),IHDR(512)
      INTEGER UNITS(2,128)
      REAL RHEADER(256),RBLOCK(64,10),ZEROS(128),SCALES(128)
      CHARACTER*16 NAME,CNAME
      INTEGER INAME(8)
      EQUIVALENCE (INAME,CNAME)
      EQUIVALENCE (IHEADER,RHEADER)
      EQUIVALENCE (IBLOCK,RBLOCK)
      EQUIVALENCE (IBLOCK(1,1),ZEROS(1))
      EQUIVALENCE (IBLOCK(1,3),SCALES(1))
      EQUIVALENCE (IBLOCK(1,5),UNITS(1,1))
      CALL FmpOpen(DCB,IERR(1),'CALTBL.DAT::DATA','ROQ',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)=ZEROS(L)              ! ZERO ARRAY
          K=L+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=L+407
          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(236)=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(406)=2*25                    ! # OF CHANNELS
      DO 20 L=1,16,2
          CNAME(L:L+1)=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
```

```
isting INITIAL.FTN:::4:18:38

EMA/DATA/
      PROGRAM INITIAL
C
C     LOCAL VARIABLES
C
      REAL RBLOCK(64,10),ZEROS(128),SCALES(128)
      INTEGER IERR(10),DCB(144),GETDA(3),DAILY(3),DISPL(3)
      INTEGER IHEADER(512)              ! FILE HEADER
      INTEGER IBLOCK(128,10)            ! CALIBRATION TABLE
      INTEGER LEN                       ! LENGTH OF FILE READ
      INTEGER FmpOpen,FmpRead           ! Fmp ROUTINES CALLED AS INTEGER FUN.
      INTEGER BUF(15)                   ! BUFFER FOR RETREVING DATE
      INTEGER IDCB(144)                 ! Data Control Block for FMP CALLS
      INTEGER*4 IBK(5,7,2)              ! Integer data block for FMP CAL
      REAL    RBK(2,7)                  ! Real data block for FMP CALLS
      CHARACTER*64 CALTBL               ! DISK FILE NAME FOR CALIBRATION FILE
C
      INCLUDE EMA&SC
C
      EQUIVALENCE (IBLOCK,RBLOCK)
      EQUIVALENCE (IBLOCK(1,1),ZEROS(1))

      EQUIVALENCE (IBLOCK(1,3),SCALES(1))
      EQUIVALENCE (IBK(1,1,2),RBK(1,1))
      DATA GETDA,DAILY,DISPL /'GETDA ','DAILY ','DISPL '/
      LU = 1
      LUD = 31
      LUP = 6
      DO JK=1,380
         DO JL=1,32
            XHS(JK,JL)=0.0
            X3S(JK,JL)=0.0
            X10S(JK,JL)=0.0
            X4M(JK,JL)=0.0
            X28M(JK,JL)=0.0
            END DO
         END DO
      DO JL=1,25
          OFFSET1M(JL)=0.0
          END DO
      I=1
      IHS=1
      I3S=1
      I10S=1
      I4M=1
      I28M=1
      COUNT=0
      COUNT1=0
      COUNT4=0
      COUNTN=0
      COUNTS=0
      STATUS=1                                        ! TAKE DATA
C
C***************** Extra Chaanel Data ******************************
C
      IDUMY=FmpOpen(IDCB,IERR,'/DATA/CHANNEL.DAT','RO',1)
      IF(IERR.LT.0) THEN
           CALL FmpReportError(IERR,INITIAL)
           END IF
      WRITE(1,*) '     Reading extra channal data.'
      CALL FmpRewind(IDCB,IERR)
      IDUMY=FmpRead(IDCB,IERR,IBK,280)
      IF (IERR.LT.0) THEN
           WRITE(1,*) 'ERROR IN READING    PRESS <CR>'
           CALL FmpReportError(IERR,initial)
           END IF
      DO 230 I=1,7
           DO 240 J=1,4
  240           CHNM(I,J)=IBK(J,I,1)
           CHUNIT(I)=IBK(5,I,1)
           C(I)=RBK(1,I)
  230      D(I)=RBK(2,I)
C
C***************** Rp PROGRAMS TO BE SCHEDULED **********************
      CALL FmpRpProgram('GETDA.RUN::USER','GETDA','P',IERR(1))
      CALL FmpRpProgram('DAILY.RUN::USER','DAILY','P',IERR(2))
      CALL FmpRpProgram('DISPL.RUN::USER','DISPL',' ',IERR(3))
      IF (IERR(1).LT.0) THEN
         CALL FmpReportError(IERR(1),GETDA)
         WRITE(LUP,*) 'GETDA.RUN      PROGRAM INITIAL STOPPED'
         STOP
      END IF
      IF (IERR(2).LT.0) THEN
         CALL FmpReportError(IERR(3),DAILY)
         WRITE(LUP,*) 'DAILY.RUN      PROGRAM INITIAL STOPPED !'
         STOP
      END IF
      IF (IERR(3).LT.0) THEN
         CALL FmpReportError(IERR(4),DISPL)
         WRITE(LUP,*) 'DISPL.RUN      PROGRAM INITIAL STOPPED !'
         STOP
      END IF
C
C***************** LOCK SHAREABLE EMA PARTITION **************
      CALL LKEMA
C
C***************** 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
      ENDIF
      TIME(2)=0
      TIME(1)=0
      I1M=TIME(4)*60+TIME(3)+1          ! SET I1M TO STARTING TIME
C
      CALL FTIME(BUF)
      CALL DFILEN(FILENAME,NAME)
      WRITE(LUP,10) (BUF(JI),JI=1,15)
   10 FORMAT(//,'****  34M VAWT DATA LOGGER ',15A2,'  ****')
      LEN = FmpOpen(DCB,IERR(10),FILENAME,'ROO',1)
      IF(IERR(10).GE.0) THEN            ! IF DATA FILE ALREADY EXIST
          WRITE(LUP,*) 'READING PREVIOUS DATA FROM DISK'
          LEN = FmpRead(DCB,IERR(6),IHEADER,512)
          CALL VMAREAD(DCB,IERR,IDATA,1800)
          CALL VMAREAD(DCB,IERR,IDATA,1800)
          CALL VMAREAD(DCB,IERR,IDATA,1800)
          CALL VMAREAD(DCB,IERR,IDATA,1800)
          COUNT = IHEADER(394)
          IF (IERR .LT. 0) GOTO 30
       ELSE                             ! SET DATA FILE TO ZERO
30        WRITE(LUP,*) 'INITIALIZE 1 MIN ARRAYS'
          DO 50 J=1,50
              DO 60 K=1,1440
                  IDATA(K,J)=0
   60         CONTINUE
   50     CONTINUE
      ENDIF
C
C***************** READ CALIBRATION FACTORS FROM DISK ***************
      CALL FmpBuildName(CALTBL,'CALTBL','DAT',0,'DATA')
      IF( FmpOpen(DCB,IERR(7),CALTBL,'ROO',1) .LT.0) THEN
          IF( IERR(7) .EQ. -6 ) THEN
              WRITE(LUP,*) 'CALIBRATION TABLE DOES NOT EXIST'
          ELSE
              CALL FmpReportError(IERR(7),CALTBL)
          END IF
              WRITE(LUP,*) 'PROGRAM STOPPED !'
              STOP
      END IF
      LEN = FmpRead(DCB,IERR(8),IBLOCK,128*10*2)
      IF( IERR(8) .LT. 0) THEN
          CALL FmpReportError( 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
C
C***************** SCHEDULE PROGRAMS ********************************
      CALL EXEC(12,GETDA,1,50,TIME(4),TIME(3))   ! SCD @ T4:T3 & EVERY 0.1S
      WRITE(LUP,105) TIME(4),TIME(3)
  105 FORMAT(' PROGRAM GETDA SCHEDULED TO START AT ',I2,':',I2)
      CALL EXEC(12,DAILY,1,0,0,0)                ! SCD @ 00:00:00
      WRITE(LUP,*) 'PROGRAM DAILY SCHEDULED'
      CALL EXEC(10,DISPL)                        ! SCD IMMEDIATELY
      WRITE(LUP,106)
  106 FORMAT('****  PROGRAM INITIAL COMPLETE  ****'////)
      STOP
      END
```

```
isting MENU.FTN:::4:5:39

$EMA/DATA/
      PROGRAM MENU
C
C     LOCAL VARIBLES
C
      INTEGER CLAS                          ! CLASS NUMBER
      INTEGER WD(7)                         ! WIND DIRECTION DESIGNATION
      INTEGER ANS
      INTEGER IDCB(144)                     ! Data Control Block for FMP CALLS
      INTEGER*4 IBK(5,7,2)                  ! Integer data block for FMP CAL
      REAL    RBK(2,7)                      ! Real data block for FMP CALLS
      INTEGER CHN
      INTEGER J1CH(4)                       ! WIND DIRECTION CHANNEL NUMBERS
      INTEGER*4 NCHNM(4)
      INTEGER BUF(15)
      INTEGER FORMFEED,IHOME,IRASE
      INTEGER INDEX2
      INTEGER INDX,IST(5),STDE
      INTEGER LUDD                          ! DISPLAY LU NOT RESETABLE BY MENU
      REAL SCALE(32),DV(32),CP(4)
      CHARACTER BELL

      CHARACTER*4 UWS,UTP,UBP,USG
      CHARACTER*6 UTQ
C
      INCLUDE EMA&SC
C
C
      EQUIVALENCE (IBK(1,1,2),RBK(1,1))
      DATA J1CH/10,12,15,16/
      DATA SCALE/32*1.0/
      BELL  = CHAR(7)
      LUDD  = 1
      IHOME = 15400B+150B            ! HOME ALPHA CURSER  Esc h
      IRASE = 15400B+112B            ! CLEAR ALPHA DISPL  Esc J
      FORMFEED = 006000B
      STDE = 1                       ! Select 3 sec Std dev
      CLAS  = 0
      CALL CLRQ(1,CLAS)
      CALL FmpOpen(IDCB,IERR,'/DATA/CHANNEL.DAT','WRO',1)
      IF (IERR.LT.0) THEN
          CALL FmpReportError(IERR,CHANNEL)
          CALL INPUTVAL(IWAIT)
          END IF
C
   10 WRITE(1,5) IHOME,IRASE
    5 FORMAT(2A2)
   15 CALL FTIME(BUF)
      WRITE(1,30) IHOME,(BUF(JI),JI=1,15),STATUS
   30 FORMAT(A2,/,21X,'SANDIA NATIONAL LABORATORIES '/
     1    20X,'34 METRE TEST BED DATA LOGGER'/,20X,15A2/,
     2    20X,'   Current STATUS =',I2/,
     3    20X,'0  Set current DATA to ZERO'/,
     4    20X,'1  Return/Continue taking DATA'/,
     5    20X,'2  Store current DATA and STOP taking DATA'/,
     6    20X,'3  STOP taking DATA'/,
     7    20X,'4  Display Data')
      WRITE(1,31)
   31 FORMAT(20X,'5  Input/Change Extra Channel data'/,
     1    20X,'6  Change LU for DISPL program '/,
     2    20X,'7  Return to CI  ',//,
     3    20X,'   Enter Number _')
      CALL INPUTVAL(NSTATUS)
      IF(NSTATUS.EQ.0) THEN
          CALL CHECK(ANS)
          IF(ANS.EQ.1) STATUS = 0                    !SET DATA TO ZERO
      ELSE IF(NSTATUS.EQ.1) THEN
          STATUS = 1                                 !TAKE DATA
      ELSE IF(NSTATUS.EQ.2) THEN
          CALL CHECK(ANS)
          IF(ANS.EQ.1) THEN
              STATUS = 2                         !STORE DATA AND STOP
              GOTO 900
          END IF
      ELSE IF(NSTATUS.EQ.3) THEN
          CALL CHECK(ANS)
          IF(ANS.EQ.1) THEN
              STATUS = 3                         !STOP WITH OUT STORE
              GOTO 900
              END IF
      ELSE IF(NSTATUS.EQ.7) THEN
          WRITE(1,90)
   90     FORMAT(///)
          GOTO 900                               !EXIT PROGRAM
      ELSE IF(NSTATUS.EQ.4) THEN
          GOTO 100
      ELSE IF(NSTATUS.EQ.5) THEN
          GOTO 200
      ELSE IF(NSTATUS.EQ.6) THEN
          WRITE(1,40)                            !CHANGE LU FOR DISPL
   40     FORMAT(//'INPUT LU NUMBER FOR DISPL PROGRAM (31,32,1) _')
          READ(1,45) LUD
   45     FORMAT(I2)
      ELSE
          WRITE(1,*) BELL
   60     FORMAT(A2,'INVALID RESPONSE')
```

```
          DO 50 JKL= 1,1000
   50         CONTINUE
          END IF
      GOTO 10
  900 STOP
C
C ******************** DISPLAY DATA ***********************************
C
  100 CONTINUE
      WRITE(1,5) IHOME,IRASE
      WRITE(LUDD,5) IHOME,IRASE
          UWS=' mps'
          UTQ='kNm   '
          UTP='C   '
          UBP='kPa '
          USG='kN  '
          IC=0
 1000 INPUT=0
      INDEX2=I3S-1
      IF( IC.GE.10 ) THEN
          WRITE(LUDD,5) IHOME,IRASE
          IC=0
      ELSE
          IC=IC+1
          END IF
      WRITE(LUDD,1001) IHOME
 1001 FORMAT(A2,' ',15X,'   SNLA 34 METRE VAWT TEST BED DATA ',
     1    'LOGGER',20X,' '//)
      CALL CHECKWS
      DO J=1,32
          DV(J)=X3S(INDEX2,J)
          END DO
C*********************************************************************
      LUS=1
      INCLUDE SCREEN
C
      WRITE(LUDD,1006) DV(25)
 1006 FORMAT('PLC=',F5.2,' V                    ',
     1         'PRESS ANY KEY TO RETURN TO THE MAIN MENU  ')
      CALL EXEC(17,100400B+LUDD,INPUT,-1,PARM1,PARM2,CLAS+100000B)
 1100 CONTINUE
      CALL EXEC(21,CLAS+120000B,INPUT,-1)
      IF(INPUT.NE.0) THEN
          CALL CLRQ(3,CLAS,1)
          GOTO 10
          END IF
      IF(I3S-1 .NE. INDEX2) THEN
          CALL CLRQ(3,CLAS,1)
          GOTO 1000
          END IF
      GOTO 1100
C******************** INPUT CHANNEL DATA ***************************************
  200 CONTINUE
      WRITE(1,2000) IHOME,IRASE
      WRITE(1,2005) (J,(CHNM(J,JK),JK=1,4),CHUNIT(J),C(J),D(J),J=1,7)
  210 WRITE(1,2010)
      CALL INPUTVAL(CHN)
      IF (CHN.LT.0.OR.CHN.GT.10) THEN
          GOTO 210
      ELSE IF (CHN.EQ.0) THEN
          WRITE(6,2005) (J,(CHNM(J,JK),JK=1,4),CHUNIT(J),
     1                  C(J),D(J),J=1,7)
          WRITE(6,2000) FORMFEED
          GOTO 200
      ELSE IF (CHN.EQ.8) THEN              ! Return to MAIN MENU
          GOTO 10
      ELSE IF (CHN.EQ.9) THEN              ! Save values on the disk
          WRITE(1,*) '    WRITE DATA '
          DO 250 I=1,7
             DO 260 J=1,4
  260           IBK(J,I,1)=CHNM(I,J)
             IBK(5,I,1)=CHUNIT(I)
             RBK(1,I)=C(I)
  250        RBK(2,I)=D(I)
          CALL FmpRewind(IDCB,IERR)
          CALL FmpWrite(IDCB,IERR,IBK,280)
          IF (IERR.LT.0) THEN
              WRITE(1,*) 'ERROR IN WRITING DATA     PRESS <CR>'
              CALL FmpReportError(IERR,CHANNEL)
              CALL INPUTVAL(IWAIT)
              END IF
          GOTO 200
      ELSE IF (CHN.EQ.10)THEN              ! Read values from the disk
          WRITE(1,*) '     READING DATA'
          CALL FmpRewind(IDCB,IERR)
          CALL FmpRead(IDCB,IERR,IBK,280)
          IF (IERR.LT.0) THEN
              WRITE(1,*) 'ERROR IN READING    PRESS <CR>'
              CALL FmpReportError(IERR,CHANNEL)
              CALL INPUTVAL(IWAIT)
              END IF
          DO 230 I=1,7
             DO 240 J=1,4
  240           CHNM(I,J)=IBK(J,I,1)
             CHUNIT(I)=IBK(5,I,1)
             C(I)=RBK(1,I)
  230        D(I)=RBK(2,I)
          GOTO 200
          END IF
      WRITE(1,2025) CHN
```

```
      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) C(CHN)
      WRITE(1,2050)
      READ(1,2045) D(CHN)
      GOTO 200
 2000 FORMAT(2A2)
 2005 FORMAT('Channel   Discription       Units   Scale     Zero'/,
     1        7(3X,I2,6X,4A,2X,A4,4X,F8.6,2X,F8.6,/))
 2010 FORMAT(/,'Input channel number to change: 1-7, or'/,
     1        'O to print data, or R to READ values from the disk ',/,
     2        'S to SAVE values on the disk, ',
     3        'or M to return to MAIN MENU  _')
 2025 FORMAT(' For channel ',I2,':',/,
     1        '    CR> saves current discription/units, sets A/B=0'/,
     2        '    Input channel discription, up to 16 characters, _')
 2030 FORMAT(4A4)
 2035 FORMAT('    Input channel units, up to 4 characters, _')
 2040 FORMAT('    Input channel scale,  Y=Ax+B   A=_')
 2045 FORMAT(F8.0)
 2050 FORMAT('    Input channel offset, Y=Ax+B   B=_')
      END
C******************** SUBROUTINE CHECK ****************************
C     SUBROUTINE RETURN A 1 IF OK TO CONTINUE
C
      SUBROUTINE CHECK(ANS)
      INTEGER ANS,CH
C
   10 WRITE(1,100)
  100 FORMAT(//'ARE YOU SURE (N)?    ENTER Y OR N _')
      ILU=1+100B
      CALL INPUTVAL(CH)
      CH=CH+ICHAR('0')
      IF(CH.EQ.89.OR.CH.EQ.121) THEN            !Y OR y
         ANS=1
      ELSE
         ANS=2
      END IF
      RETURN
      END
C
C******************** SUBROUTINE INPUTVAL ************************
C     SUBROUTINE RETURNS AN INTEGER ENTERED FROM THE KEY BOARD
C
      SUBROUTINE INPUTVAL(INT)
      CHARACTER SEL
      INTEGER C,INT,ISEL
      INTEGER LUINPUT,ILU
C
      LUINPUT=1
      ILU=LUINPUT+100B
      CALL EXEC(1,ILU,C,-1)
      SEL = CHAR( C/256 )
      ISEL = ICHAR(SEL)
      IF     (ISEL.EQ.77 .OR. ISEL.EQ.109) THEN       ! IF M
         INT =  8
      ELSEIF (ISEL.EQ.83 .OR. ISEL.EQ.115) THEN       ! IF S
         INT =  9
      ELSEIF (ISEL.EQ.82 .OR. ISEL.EQ.114) THEN       ! IF R
         INT =  10
      ELSE
         INT = ISEL-ICHAR('0')
         END IF
      RETURN
      END
C
      INCLUDE CHECKWS
C
```

```
*********************************************************************
C
      SUBROUTINE PLUT(PARRAY,LUD,WS,NCH,IUN1,IUN2,IUN3,ITIM,
     1                ICH,SI)
C
      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 AVGA, 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
      CHARACTER MFN*50                      ! TimeSeries File Name
      CHARACTER GBUFF*102                   ! 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 ICH(3)                        ! CHANNEL NUMBERS
      INTEGER NCH                           ! NUMBER OF CH (1 OR 2)
      INTEGER LUD                           ! LU # OF OUTPUT DEVICE
      INTEGER WS                            ! WS >0 THEN PLOT WS
C
      DATA MFN /'SNLA/USDA 34M VAWT TEST BED DATA LOGGER'/
C
      IHOME = 15400B+150B                   ! HOME CURSOR   ESC h
      IRASE = 15400B+112B                   ! CLEAR ALPH DISPL   ESC R
      IBELL = 003400B                       ! RING BELL    CHAR(7)
      IND = 361                             ! NUMBER OF POINTS TO PLOT ?
      NRD = 361                             ! NUMBER OF POINTS TO PLOT ?
      NNN = 1                               ! PLOT EVERY POINT
C
      DO J=1,361
         DATO(J)=J-1
      END DO
C
C
C     FIND DATA'S MAX & MIN AMPLITUDES
C
   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( I,2 ) .GT. DMAX( 2 ) ) DMAX( 2 ) = PARRAY( I,2 )
         IF( PARRAY( I,2 ) .LT. DMIN( 2 ) ) DMIN( 2 ) = PARRAY( I,2 )
         IF( PARRAY( I,3 ) .GT. DMAX( 3 ) ) DMAX( 3 ) = PARRAY( I,3 )
         IF( PARRAY( I,3 ) .LT. DMIN( 3 ) ) DMIN( 3 ) = PARRAY( I,3 )
         END DO
      DR( 2 ) = DMAX( 2 ) - DMIN( 2 )
      DR( 3 ) = DMAX( 3 ) - DMIN( 3 )
C
C  AUTO SCALE ON TWO ADDITIONAL CHANNELS
C     DETERMINE THE 1ST POINT AND DELTA TO PLOT
C
   30 DELCHK = 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
   35    FLAST = 0.02
         DALL = 0.02
         DELL1 = 0.1
   40    IF( DR( I ) .LE. FLAST ) GOTO 60
         IF( FLAST .GE. DELL1 ) THEN
            DALL = DELL1
            DELL1 = DELL1 * 10.0
            END IF
   50    FLAST = FLAST + DALL
         GOTO 40
   60    DEL( I ) = FLAST / 2.0
         IFLAS1 = IFIX( DMIN( I ) / 10000.0 / DEL( I ) )
         VARCHK = DMIN( I ) - FLOAT( IFLAS1 ) * 10000.0 * DEL( I )
         IFLAS = VARCHK / DEL( I )
         IF( DMIN( I ) .LT. 0 ) IFLAS = IFLAS - 1
         IF( I .EQ. 2 ) IFLAS = IFLAS - 1
         FP( I ) = (FLOAT( IFLAS1 ) * 10000.0 + FLOAT( IFLAS )) *
     1             DEL( I )
   70    CONTINUE
   80 FP1 = 0.0
      DEL1 = 25.0
C
C  FIX TIME SCALE PARAMETERS
C
      TT = SI *(NRD - 1.0) * NNN                ! TOTAL TIME
      BT = DATO( 1 )
      DELT = TT / 5.0                           ! TOTAL TIME / 5 INCH
      IF( TT .GT. 3600 ) THEN                   ! MORE THAN 1 HOUR
          BT = BT / 3600.0
```

```
          DELT = DELT / 3600.0
          DO 700 I=1 , NRD
              DATO( I ) = DATO( I ) / 3600.0
  700     CONTINUE
          ITIM( 4 ) = 2HHR
          ITIM( 5 ) = 2HS)
      ELSE IF( TT .GT. 200 ) THEN              ! MORE THAN 3 MIN
          BT = BT / 60.0
          DELT = DELT / 60.0
          DO 710 I=1 , NRD
              DATO( I ) = DATO( I ) / 60.0
  710     CONTINUE
          ITIM( 4 ) = 2HMI
          ITIM( 5 ) = 2HN)
      END IF
   85 WRITE(LUD,581) IHOME
  581 FORMAT(5A2)
      WRITE(LUD,581) IRASE
C
      CALL PLOTS( LUD )                        ! INITIALIZE GRAPHICS
C     CALL PLOTS(IARRAY,1,LUD)                 ! INITIALIZE GRAPHICS
C
C     1ST POINT AND DELTA STORED FOR LATER USE
C
      FP2 = FP( 2 )
      FP3 = FP( 3 )
      DEL2 = DEL( 2 )
      DEL3 = DEL( 3 )
      PARRAY( IND + 1, 2 ) = FP( 2 )
      PARRAY( IND + 2, 2 ) = DEL( 2 )
      PARRAY( IND + 1, 3 ) = FP( 3 )
      PARRAY( IND + 2, 3 ) = DEL( 3 )
      PARRAY( IND + 1, 1 ) = 0
      PARRAY( IND + 2, 1 ) = DEL1
      DATO( IND + 1 ) = BT                          ! CHANGED BP TO BT
      DATO( IND + 2 ) = DELT / SI                   ! ADDED (/ SI)
C
C     COMPUTE AVERAGES.
C
      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
C
C     START PLOTTING
C
      CALL ZNEWF
      CALL ZMOVE( 0.0 , 5.85 )
C
      WRITE(GBUFF,560) MFN(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,1,AVG2,2,AVG3
          END IF
  570     FORMAT(1X,'WS AVE= ',F5.2,2('  CH',I2,' AVE=',F6.3,2X))
      ELSE
          IF(NCH.EQ.1) THEN
              WRITE(GBUFF,575) 1,AVG2
          ELSE IF(NCH.EQ.2) THEN
              WRITE(GBUFF,575) 1,AVG2,2,AVG3
          END IF
  575     FORMAT(1X,'CHANNELS',2(I2,'AVE=',F6.3,2X))
      END IF
      CALL PRINTG( GBUFF )
C
C  FILL LINE ARRAYS
C
  290 DO 760 I=1 , IND
          DATO( I ) = DATO( I ) - BT
          PARRAY( I,1 ) = PARRAY( I,1 ) + 3.0 * DEL1
  760     CONTINUE
C
C     DRAW AXISES
C
      CALL PPLOT( 0.35 , 0.4 , -3 )
      NCL = NRD / 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 , IUN2 ,  10 , -4.0 , 90.0 , FP( 2 ) ,
     1           DEL( 2 ) )
      IF( NCH .EQ. 2 ) THEN
          CALL AXIS( 5.0 , 0.0 , IUN3 , -10 , -4.0 , 90.0 ,
     1           FP( 3 ) , DEL ( 3 ) )
          END IF
      IF( WS .GT. 0 ) THEN
          CALL AXIS( 5.5 , 3.0 , IUN1 , -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 , -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 GTDAT( GBUFF )                  ! PUT DATE AND TIME ON PLOT
   CALL PPLOT( 3.9, -0.4, 3 )
   CALL PRINTG( GBUFF )
   CALL ANMDE
   WRITE(LUD,580) IBELL
580 FORMAT(//////////,A2)
   WRITE(LUD,590)
590 FORMAT(/////,
  1        63X,'PRESS CR>'/,
  1        63X,'TO RETURN'/,
  2        63X,'TO MAIN MENU'//,
  3        63X,'PRESS "ENTER"'/,
  4        63X,'ON GRAPHICS'/,
  4        63X,'KEY BOARD'/,
  5        63X,'TO COPY PLOT _')
   READ(LUD,335) IDUMMY
335 FORMAT(A2)
340 CALL EDITM(LUD)
   RETURN
   END
```

```
C     SYSTEM COMMON VARIABLE DEFINITIONS
C
      INTEGER I                  !  SAMPLE INDEX                  1-120
      INTEGER IHS                ! 0.5 SEC INDEX                  1-380
      INTEGER I3S                !  3  SEC INDEX                  1-380
      INTEGER I10S               ! 10  SEC INDEX                  1-380
      INTEGER I4M                !  4  MIN INDEX                  1-380
      INTEGER I28M               ! 28  MIN INDEX                  1-380
      INTEGER I1M                !  1  MIN INDEX                  1-1440
      INTEGER COUNT              !  COUNT OF 4MIN AVE             0-360
      INTEGER COUNT1             !  COUNT OF 1MIN AVE             0-3
      INTEGER COUNT4             !  COUNT OF 4MIN AVE             0-7
      INTEGER LUD                !  LU FOR DISPL
      INTEGER COUNTN,COUNTS      !  COUNT OF TIMES NORTH/SOUTH ANEMOM<>
      INTEGER*4 CHNM(7,4)        !  NAMES OF NON STRORED CHANNELS
      INTEGER*4 CHUNIT(7)        !  UNITS OF NON STRORED CHANNELS
      INTEGER STATUS             !  STATUS: 0=SET NEW DATA TO ZERO,
C                                   1=TAKE DATA, 2=STORE DATA AND STOP
C                                   3=STOP WITHOUT SAVING DATA
      REAL OFFSET1M(25)          !  OFFSET ARRAY FOR CORRECTING WD

      REAL STDDEV(2,7)           !  Standard Deviation of non stored
C                                   channels, 3 sec/ 10 sec
C
      INTEGER TIME(5)            ! DAY OF YEAR,HOUR,MIN,SEC,10mSEC
      CHARACTER*64 FILENAME      ! DATA STORAGE FILE DESCRIPTOR
      CHARACTER*16 NAME          ! DATA STORAGE FILE NAME
C
      INTEGER LU
C
      COMMON//I,IHS,I3S,I10S,I4M,I28M,I1M,TIME,NAME,FILENAME,LU,
     1        COUNT,COUNT1,COUNT4,COUNTN,COUNTS,CHNM,CHUNIT,
     2        STATUS,OFFSET1M,LUD,STDDEV
C
C
C     SHARABLE EMA VARIABLE DEFINITIONS
C
      COMMON/DATA/A(25),B(25),  !CALIBRATION FACTORS--SLOPE & ZERO OFFSET
C        C(7),D(7),             !CALIBRATION FACTORS--CHANGEABLE CHANNELS
C        XHS(380,32),           ! 0.5 SEC VALUES        LAST 3 MIN
C        X3S(380,32),           !  3  SEC AVERAGES      LAST 18 MINS
C        X10S(380,32),          ! 10  SEC AVERAGES      LAST HOUR
C        X4M(380,32),           !  4  MIN AVERAGES      LAST DAY
C        X28M(380,32)           ! 28  MIN AVERAGES      LAST WEEK
C        IDATA(1440,50),        !  1  MIN AVE & STD DEVIATIONS
C        ITP1(1440,20),         !  1  MIN AVE & SD TEMP ARRAY FOR VWrit
C        ITP2(1440,20),         !  1  MIN AVE & SD TEMP ARRAY FOR VWrit
C        ITP3(1440,10)          !  1  MIN AVE & SD TEMP ARRAY FOR VWrit
C
```

```
      DO J=1,4
          J1=J1CH(J)
          IF(DV(J1).LE.23.0.OR.DV(J1).GT.338) THEN
               WD(J)=(2HN )
          ELSEIF(DV(J1).LE.68) THEN
               WD(J)=(2HNE)
          ELSEIF(DV(J1).LE.113) THEN
               WD(J)=(2HE )
          ELSEIF(DV(J1).LE.158) THEN
               WD(J)=(2HSE)
          ELSEIF(DV(J1).LE.203) THEN
               WD(J)=(2HS )
          ELSEIF(DV(J1).LE.248) THEN
               WD(J)=(2HSW)
          ELSEIF(DV(J1).LE.293) THEN
               WD(J)=(2HW )
          ELSE
               WD(J)=(2HNW)
             END IF
          END DO
C           RotorPower=Torque*Rpm*2Pi/60

      RP=DV(19)*DV(24)*0.1047/SCALE(19)
C           CP=POWER/(0.5*row    *AREA*WS**3)
C           CP=POWER/(0.5*1.293 *955 *WS**3)
      CP(1)=RP/(617.4*DV(2)**3)
      CP(2)=DV(17)/(617.4*DV(2)**3)
      CP(3)=RP/(617.4*DV(1)**3)
      CP(4)=DV(17)/(617.4*DV(2)**1)
      WRITE(LUS,1010)
      WRITE(LUS,1011) DV(1),UWS,DV(10),WD(1),CP(1),CP(2)
      WRITE(LUS,1012) DV(3),UWS,DV(12),WD(2),CP(3),CP(4)
      WRITE(LUS,1013) DV(6),UWS,DV(15),WD(3)
      WRITE(LUS,1014) DV(7),UWS
      WRITE(LUS,1015) DV(8),UWS,DV(16),WD(4),DV(17),DV(18)
      WRITE(LUS,1016) DV(9),UWS,RP
      WRITE(LUS,1017)
      WRITE(LUS,1018)
      WRITE(LUS,1019) DV(4),DV(5),DV(24)
      WRITE(LUS,1020) DV(19),UTQ
      WRITE(LUS,1021) (CHNM(1,M),M=1,4),DV(26),CHUNIT(1),
     1                STDDEV(STDE,1),DV(20),USG
      WRITE(LUS,1022) (CHNM(2,M),M=1,4),DV(27),CHUNIT(2),
     1                STDDEV(STDE,2)
      WRITE(LUS,1023) (CHNM(3,M),M=1,4),DV(28),CHUNIT(3),
     1                STDDEV(STDE,3)
      WRITE(LUS,1024) (CHNM(4,M),M=1,4),DV(29),CHUNIT(4),
     1                STDDEV(STDE,4),DV(21),UTP
      WRITE(LUS,1025) (CHNM(5,M),M=1,4),DV(30),CHUNIT(5),
     1                STDDEV(STDE,5),DV(22),UTP
      WRITE(LUS,1026) (CHNM(6,M),M=1,4),DV(31),CHUNIT(6),
     1                STDDEV(STDE,6),DV(13),UBP
      CALL FTIME(BUF)
      WRITE(LUS,1027) (CHNM(7,M),M=1,4),DV(32),CHUNIT(7),
     1                STDDEV(STDE,7),(BUF(J),J=1,15)
 1010 FORMAT('WIND        Speed    Direction    ',10X,
     1       'COEF OF PERFORMANCE                  ')
 1011 FORMAT(' 10 m    ',F4.1,A5,F6.1,'  deg ',A2,10X,
     1       'Equator:   Aero ',F5.3,'    System ',F5.3)
 1012 FORMAT(' 30 m    ',F4.1,A5,F6.1,'  deg ',A2,10X,
     1       '10 metre   Aero ',F5.3,'    System ',F5.3)
 1013 FORMAT(' NE      ',F4.1,A5,F6.1,'  deg ',A2,'            ')
 1014 FORMAT(' NW      ',F4.1,A5,'                ',10X,
     1       'POWER                                ')
 1015 FORMAT(' SE      ',F4.1,A5,F6.1,'  deg ',A2,10X,
     1       'Electrical    ',F6.1,' kW',2X,F6.1,' kVAr')
 1016 FORMAT(' SW      ',F4.1,A5,'                ',10X,
     1       'Rotor',8X,F6.1,' kW')
 1017 FORMAT('                                    ')
 1018 FORMAT('CONTROLLER                          ',10X,
     1       'TURBINE                              ')
 1019 FORMAT(F4.2,' Volts  ',F4.2,' Volts',20X,
     1       'Speed',10X,F4.1,' rpm')
 1020 FORMAT('                                    ',10X,
     1       'Torque',7X,F6.1,A7)
 1021 FORMAT(4A,1X,F7.2,1X,A4,1X,F6.2,6X,
     1       'Tiedown Cable ',F5.1,A5)
 1022 FORMAT(4A,1X,F7.2,1X,A4,1X,F6.2,'                          ')
 1023 FORMAT(4A,1X,F7.2,1X,A4,1X,F6.2,6X,
     1       'AMBIENT CONDITIONS                   ')
 1024 FORMAT(4A,1X,F7.2,1X,A4,1X,F6.2,6X,
     1       'Temperature 10 metre ',F5.1,A6)
 1025 FORMAT(4A,1X,F7.2,1X,A4,1X,F6.2,6X,
     1       'Temperature 48 metre ',F5.1,A6)
 1026 FORMAT(4A,1X,F7.2,1X,A4,1X,F6.2,6X,
     1       'Barometric Pressure',3X,F5.2,A5)
 1027 FORMAT(4A,1X,F7.2,1X,A4,1X,F6.2,6X,15A2,'  ')
C
```

Appendix E

HP 1000
SYSTEM GENERATION ANSWER FILE
AND
WELCOME FILE

```
I32> LI,ANS.4
 ANS     92077-17197   REV.2440 <890222.1804>
 RTE-A.3  Primary System Generation Answer File (Non VC+)

*REVISED 02-22-89  M E RALPH
*
*
*
links,cp,,          use current page links
*
********************
* System Relocation
********************
*
er
re %VCTR::RTE
*
*
re %EXEC::RTE
re %MEMRY::RTE
*
re %RPL60::RTE,,     A-600   no CDS    no double precison floating point

re %SAM::RTE
re %TIME::RTE
re %SCHED::RTE
re %STRNG::RTE
re %LOCK::RTE
re %ERLOG::RTE
RE,/RTE/%VEMA
re %OPMSG::RTE
RE,/RTE/%XCMND
re %SYCOM::RTE
RE,/RTE/%STAT
re %LOAD::RTE
*
re %RTIOA::RTE
re %IOMOD::RTE
RE,/RTE/%PERR
re %CLASS::RTE
*
********************
*  SEARCH LIBRARIES
********************
*
ms $SYSA::RTE
se $MATH::RTE
se $SYSLB::RTE
end
******************
*   OS PARTITIONS
******************
*
**********************
*   Driver partitions
**********************
*
re %DD.33::RTE,, CS 80 DISK
end
*
* re %ID.00::RTE,,  ASYNC SERIAL
* end
*
re %ID.37::RTE,, HPIB
RE,/RTE/%DD.30,, DISK 9133
end
*
re %IDM00::RTE,, ASYNC MUX
end
*
re %DD.00::RTE,, TERMINAL
ALIGN
RE,/RTE/%ID.50,,  A/D DRIVER
end
*
*    end driver partition
*
end
*
****************************************
* Table Generation -- configure LU tables
****************************************
*
*
*********************************************************************
* ASYNC MUX LU'S                                      select code = 23b
*                                                        LU 1 & 31-37
*********************************************************************
*
IFT,%idm00::RTE,SC:23B,TX:20
*
DVT,%DD.00::RTE,M26XX,LU:1,QU:FI,TX:57,DP:1:20004B,-
    DP:5:CI:20040B:20040B:0,DP:9:CM:20040B:20040B:CM
DVT,%DD.00::RTE,M26XX,LU:31,QU:FI,TX:57,DP:1:20004B,-
    DP:5:CI:20040B:20040B:0,DP:9:CM:20040B:20040B:CM
DVT,%DD.00::RTE,M26XX,LU:32,QU:FI,TX:57,DP:1:20004B,-
    DP:5:CI:20040B:20040B:0,DP:9:CM:20040B:20040B:CM
DVT,%DD.00::RTE,M26XX,LU:33,QU:FI,TX:57,DP:1:20004B,-
    DP:5:CI:20040B:20040B:0,DP:9:CM:20040B:20040B:CM
DVT,%DD.00::RTE,M26XX,LU:34,QU:FI,TX:57,DP:1:20004B,-
```

```
    DP:5:CI:200408:200408:0,DP:9:CM:200408:200408:CM
DVT,%DD.00::RTE,M26XX,LU:35,QU:FI,TX:57,DP:1:200048,-
    DP:5:CI:200408:200408:0,DP:9:CM:200408:200408:CM
DVT,%DD.00::RTE,M26XX,LU:36,QU:FI,TX:57,DP:1:200048,-
    DP:5:CI:200408:200408:0,DP:9:CM:200408:200408:CM
DVT,%DD.00::RTE,M26XX,LU:37,QU:FI,TX:57,DP:1:200048,-
    DP:5:CI:200408:200408:0,DP:9:CM:200408:200408:CM
*
*****************************************************************
* HP-IB #1 -- discs and magnetic tape              select code = 27b
*****************************************************************q
*
IFT,%ID.37::RTE,SC:27B
*
*
*
* 7942 CS-80 DISC                                    HP-IB address 2
*                                                       LU 16,17 & 20
*
DVT,%DD.33::RTE,M7912_LF:0,LU:16,DP:1:2:0:0:0:0,DP:6:400:48:0
DVT,%DD.33::RTE,M7912_LF:1,LU:17,DP:1:2:0:0:0:19200,DP:6:943:48:0
DVT,%DD.33::RTE,M7912_LF:2,LU:20,DP:1:2:0:0:0:64750,DP:6:587:48:0
*
* CS-80 cartridge tape cache                            LU 24
DVT,%DD.33::RTE,MTAPE,LU:24,DP:1:2:4008:1000008:0:64464,DP:6:0:0:0
*
*
* 9133XV SNLA Org 7521 Aux disk for SYSTEM updates   HP-IB address 3
*                                                       LU 21-23
DVT,/RTE/%DD.30,M9134L:0,LU:21,DP:1:3,DP:4:0:0:300:31:6
DVT,/RTE/%DD.30,M9134L:1,LU:22,DP:1:3,DP:4:50:0:720:31:6
DVT,/RTE/%DD.30,M9134L:2,LU:23,DP:1:3,DP:4:170:0:810:31:6
*
*
* HP-IB tape drive 7970E                             HP-IB address 4
*   ****** RELOCATE DRIVER                              LU 8
*DVT,%DD.23::RTE,M7970E:0,LU:8,DP:1:4,PR:1
*
*****************************************************************
* ANALOG INPUT CARD                              SELECT CODE 33B
*****************************************************************
*
IFT,%ID.50::RTE,SC:33B
*                                                   LU 10
DVT,,,LU:10,TO:500,DT:45B,DX:2,DP:1:1:2
*
*
*****************************************************************
* HP-IB #2 --  Printers and slow devices           select code = 31b
*****************************************************************
*
IFT,%ID.37::RTE,SC:31B
*
*
* THINKJET                                           HP-IB address 1
*                                                       LU 6
DVT,,,LU:6,TO:500,DT:12B,DX:1,DP:1:1
*
*
*
end
*
end
*
********************
* Define Node Lists
********************
*
* ASYNC MUX
*NODE,1,31,32,33,34,35,36,37
*
*  7942 CS-80 DISC DRIVE
node,16,17,20,24
*
* 5.25" Fixed Disc LU's (9133/4 XV)      Aux disk for updates
node,21,22,23
*
*
end,,,,node list
*
************************
end,,,,interrupt table
************************
*
*********************
* Memory Allocation
*********************
*
clas 10
resn 10
id 20
rs 0
sam 4096
sl 0,0
bg 30
qu 300 50
sp 0
mb 500,,          For DS!, mb should be about 500
us 3
lb
```

```
*
****************
* System Common
****************
*
*
end,,,,LABELED SYSTEM COMMON
*
***************
* BLANK COMMON
***************
*
COM, 1024
*
* System Messages
*
re %MSGTB::RTE
en
re %BM000::RTE
en
en
*
* System Libraries
*
lib $FNDLB
lib $BIGLB
*
*
end
```

```
LT,WELCOME1.CMD
* welcome1.cmd          2-22-89
WD /USER
*
* ENABLING MUX TERMINALS
CN 31 30B 152331B
CN 32 30B 152332B
* Schedule primary program
* CN 31 20B C131
CN 32 20B C132
* Schedule secondary program
*CN 1 40B What_?
*CN 31 40B What_?
*CN 32 40B What_?
*
RU,CLEAR1
RU,DATE
RU,CLEAR31
RU,INITIAL
* ru,MENU
EX
```

DISTRIBUTION:

D. K. Ai
Alcoa Technical Center
Aluminum Company of America
Alcoa Center, PA 15069

Dr. R. E. Akins
Washington & Lee University
P.O. Box 735
Lexington, VA 24450

The American Wind Energy Association
1730 N. Lynn Street, #610
Arlington, VA 22209

Dr. Mike Anderson
VAWT, Ltd.
1 St. Albans
Hemel Hempstead
Herts HP2 4TA
United Kingdom

Dr. M. P. Ansell
School of Material Science
University of Bath
Claverton Down
Bath BA2 7AY
Avon
UNITED KINGDOM

Holt Ashley
Dept. of Aeronautics and
Astronautics Mechanical Engr.
Stanford University
Stanford, CA 94305

K. Bergey
University of Oklahoma
Aero Engineering Department
Norman, OK 73069

Ir. Jos Beurskens
Programme Manager for
Renewable Energies
Netherlands Energy Research
Foundation ECN
Westerduinweg 3
P.O. Box 1
1755 ZG Petten (NH)
The Netherlands

J. R. Birk
Electric Power Research Institute
3412 Hillview Avenue
Palo Alto, CA 94304

N. Butler
Bonneville Power Administration
P.O. Box 3621
Portland, OR 97208

Monique Carpentier
Energy, Mines and Resources
Renewable Energy Branch
460 O'Connor Street
Ottawa, Ontario K1A OE4
CANADA

Dr. R. N. Clark
USDA
Agricultural Research Service
Southwest Great Plains Research
Center
Bushland, TX 79012

Otto de Vries
National Aerospace Laboratory
Anthony Fokkerweg 2
Amsterdam 1017
THE NETHERLANDS

E. A. DeMeo
Electric Power Research Institute
3412 Hillview Avenue
Palo Alto, CA 94304

C. W. Dodd
Universal Data Systems
5000 Bradford Drive
Huntsville, AL 35805

J. B. Dragt
Physics Department
Nederlands Energy Research
Foundation
(E.C.N.)
Westerduinweg 3 Petten (NH)
THE NETHERLANDS

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

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 M25 5A6
CANADA

Bernard Masse
Institut de Recherche d'Hydro-Quebec
1800, Montee Ste-Julie
Varennes, Quebec JOL 2PO
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 Polytecnique
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

Dr. R. Ganesh Rajagopalan
Assistant Professor
Aerospace Engineering Department
Iowa State University
404 Town Engineering Bldg.
Ames, IA 50011

R. Rangi
Low Speed Aerodynamics Laboratory
NRC-National Aeronautical
Establishment
Montreal Road
Ottawa, Ontario K1A OR6
CANADA

Markus G. Real, President
Alpha Real Ag
Feldeggstrasse 89
CH 8008 Zurich
Switzerland

L. J. Rogers
Wind/Oceans Technologies Division
U.S. Department of Energy
1000 Independence Avenue
Washington, DC 20585

R. L. Scheffler
Research and Development Dept.
Room 497
Southern California Edison
P.O. Box 800
Rosemead, CA 91770

L. Schienbein
FloWind Corporation
1183 Quarry Lane
Pleasanton, CA 94566

Gwen Schreiner
Librarian
National Atomic Museum
Albuquerque, NM 87185

Thomas Schweizer
Science Applications International
Corp.
4300 King Street, Suite 310
Alexandria, VA 22302

David Sharpe
Dept. of Aeronautical Engineering
Queen Mary College
Mile End Road
London, E1 4NS
UNITED KINGDOM

J. Sladky, Jr.
Kinetics Group, Inc.
P.O. Box 1071
Mercer Island, WA 98040

M. Snyder
Aero Engineering Department
Wichita State University
Wichita, KS 67208

L. H. Soderholm
Agricultural Engineering
Room 213
Iowa State University
Ames, IA 50010

Peter South
ADECON
6535 Millcreek Dr., Unit 67
Mississauga, Ontario L5N 2M2
CANADA

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

Forrest S. Stoddard
West Texas State University
Alternative Energy Institute
WT Box 248
Canyon, Texas 79016

Derek Taylor
Alternative Energy Group
Walton Hall
Open University
Milton Keynes MK7 6AA
UNITED KINGDOM

G. P. Tennyson
DOE/AL/ETWMD
Albuquerque, NM 87115

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

400	R. C. Maydew
1520	L. W. Davison
1522	R. C. Reuter, Jr.
1522	D. W. Lobitz
1522	E. D. Reedy
1523	J. H. Biffle
1524	C. R. Dohrmann
1524	D. R. Martinez
1550	C. W. Peterson
1552	J. H. Strickland
1556	G. F. Homicz
3141	S. A. Landenberger (5)
3151	W. I. Klein (3)
3154-1	C. L. Ward (8) For DOE/OSTI (Unlimited Release)
3161	P. S. Wilson
6000	D. L. Hartley
6200	V. L. Dugan
6220	D. G. Schueler
6225	H. M. Dodd (50)
6225	T. D. Ashwill
6225	D. E. Berg
6225	M. E. Ralph
6225	D. C. Reda
6225	M. A. Rumsey
6225	L. L. Schluter
6225	W. A. Stephenson
6225	H. J. Sutherland
6225	P. S. Veers
7543	R. Rodeman
7543	T. G. Carne
7543	J. Lauffer
8524	J. R. Wackerly

Distribution
Category UC-261

SAND90-0116
Unlimited Release
Printed April 1990

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.

RECEIVED
JUN 01 1990
3141