Hanford Meteorological Station Computer Codes

Volume 8 - The REVIEW Computer Code

**G. L. Andrews
K. W. Burk**

August 1988

**Prepared for the U.S. Department of Energy
under Contract DE-AC06-76RLO 1830**

**Pacific Northwest Laboratory
Operated for the U.S. Department of Energy
by Battelle Memorial Institute**

DISCLAIMER

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 Battelle Memorial Institute, nor any of their employees, makes any warranty, expressed 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 or any agency thereof, or Battelle Memorial Institute. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof, or Battelle Memorial Institute.

PACIFIC NORTHWEST LABORATORY
operated by
BATTELLE MEMORIAL INSTITUTE
for the
UNITED STATES DEPARTMENT OF ENERGY
under Contract DE-AC06-76RLO 1830

Printed in the United States of America
Available from
National Technical Information Service
United States Department of Commerce
5285 Port Royal Road
Springfield, Virginia 22161

NTIS Price Codes
Microfiche A01

Printed Copy

Pages	Price Codes
001-025	A02
026-050	A03
051-075	A04
076-100	A05
101-125	A06
126-150	A07
151-175	A08
176-200	A09
201-225	A010
226-250	A011
251-275	A012
276-300	A013

HANFORD METEOROLOGICAL STATION
COMPUTER CODES

Volume 8 - The REVIEW Computer Code

G. L. Andrews
K. W. Burk

August 1988

Prepared for
the U.S. Department of Energy
under Contract DE-AC06-76RLO 1830

Pacific Northwest Laboratory
Richland, Washington 99352

PREFACE

This volume is the eighth in a series of volumes that describe computer programs routinely used at the Hanford Meteorological Station (HMS) (e.g., data entry programs). Each volume, which documents one computer code, consists of a program overview, user's guide, description of input, a detailed program description and flowchart, and a source listing of the code.

The first seven volumes present the data entry programs used at the HMS for archiving and checking various types of data. All of these programs use a data entry form to facilitate data input, and each code calls the same series of subroutines that control and display the data entry form at the user's terminal. The subroutines were generated from the program FORMEDIT, which was developed by John Wiberg of the Computer Science staff in the Engineering Physics Department at Pacific Northwest Laboratory. An informal user's guide to the FORMEDIT program that describes how to create, modify, and eliminate data entry forms is available from the HMS system manager.

SUMMARY

The Hanford Meteorological Station (HMS) routinely collects meteorological data from sources on and off the Hanford Site. The data are averaged over both 15 minutes and 1 hour and are maintained in separate databases on the Digital Equipment Corporation (DEC) VAX 11/750 at the HMS. The databases are transferred to the Emergency Management System (EMS) DEC VAX 11/750 computer. The EMS is part of the Unified Dose Assessment Center, which is located on the ground-level floor of the Federal Building in Richland and operated by Pacific Northwest Laboratory. The computer program REVIEW is used to display meteorological data in graphical and alphanumeric form from either the 15-minute or hourly database. The code is available on the HMS and EMS computer. The REVIEW program helps maintain a high level of quality assurance on the instruments that collect the data and provides a convenient mechanism for analyzing meteorological data on a routine basis and during emergency response situations.

ACKNOWLEDGMENTS

The authors would like to thank the staff at the Hanford Meteorological Station for their assistance during the preparation of this document. Special thanks to the UDAC Technical Manager Dave McCormack for his support during REVIEW program development and HMS Project Manager Dana Hoitink for his support during documentation of the REVIEW program. Ora Gifford provided helpful suggestions and ideas during the peer review.

CONTENTS

PREFACE	iii
SUMMARY	v
ACKNOWLEDGMENTS	vii
INTRODUCTION	1
THE REVIEW COMPUTER CODE	5
PROGRAM OVERVIEW	5
USER'S GUIDE FOR THE REVIEW COMPUTER CODE	6
DESCRIPTION OF INPUT	10
LIMITATIONS TO THE PROGRAM, DATABASE, AND COMPUTER SYSTEM	14
TROUBLE-SHOOTING THE REVIEW COMPUTER CODE	17
TERMINAL PROBLEMS/FAILURES	17
SYSTEM PROBLEMS/FAILURES	17
PROGRAM PROBLEMS/FAILURES	18
APPENDIX A - DESCRIPTION AND FLOWCHART OF THE REVIEW PROGRAM	A.1
APPENDIX B - DESCRIPTIONS AND FLOWCHARTS OF REVIEW SUBROUTINES	B.1
APPENDIX C - REVIEW PROGRAM AND SUBROUTINE SOURCE CODE LISTINGS	C.1

FIGURES

1	Location of the Data Collection Components on the Hanford Site . .	2
2	Display of 15-minute Meteorological Data: Area Variables . . .	3
3	An Example of a Variable (Dry Bulb Temperature) Plot from the REVIEW Computer Code	4
4	Example of the Data Entry Form Used by the REVIEW Computer Code .	6
5	Completed Data Entry Form from Example Session	9

INTRODUCTION

The Hanford Meteorological Station (HMS) was established in 1944 on the Hanford Site to collect and archive meteorological data and provide weather forecasts and related services for Hanford project activities. The HMS is located in the northwest quadrant of the Hanford Site approximately 1/2 mi east of the 200 West Area, and is operated by the Pacific Northwest Laboratory (PNL) for the U.S. Department of Energy. Meteorological data are collected from various sensors and equipment located on and off the Hanford Site (Fig. 1). The 15-minute and hourly average for the collected data are stored in two databases on the Digital Equipment Corporation (DEC) VAX 11/750 at the HMS (hereafter referred to as the HMS computer). Files from those databases are routinely transferred to the Emergency Management System (EMS) computer at the Unified Dose Assessment Center (UDAC).

The REVIEW computer code displays the 15-minute or hourly meteorological data in graphical and alphanumeric form (Fig. 2 and 3). This code provides a method for routinely analyzing meteorological data and facilitating access to the data during emergency response situations. In addition, the code helps maintain a high level of quality assurance on the meteorological sensors located throughout the Hanford Site.

The following sections in this volume describe the implementation and operation of the REVIEW computer code. Appendix A contains a description and flowchart of the REVIEW computer code, Appendix B contains brief descriptions and flowcharts of supporting subroutines, and Appendix C contains a complete source listing of the main program and support subroutines.

- Telemetry Locations
- △ SODAR Locations
- 200-ft Tower Locations
- ⊕ 410-ft Tower and
Surface Observation
Location

FIGURE 1. Location of the Data Collection Components on the Hanford Site

VARIABLES FOR 400 AREA

DATE: 88-05-09

TIME: 10:15

15-MINUTE DATA

Numeric ID.	Alphanumeric ID.	Mean Parameter Value
105	AT-400A-2M	67.2
104	AT-400A-10M	65.2
100	AT-400A-60M	62.4
103	WD-400A-10M	200
99	WS-400A-10M	18
102	WD-400A-25M	210
98	WS-400A-25M	20
101	WD-400A-60M	200
97	WS-400A-60M	21
113	DP-400A	41.0
112	DT-400A	-2.79

FIGURE 2. Display of 15-minute Meteorological Data: Area Variables

DRY BULB TEMP FOR HANFORD (F)

DATE: 88-05-09

TIME: 10:15

15-MINUTE DATA

Numeric ID.	Alphanumeric ID.	Mean Parameter Value
105	AT-FFTF-2M	67.2
140	AT-100A-2M	65.4
143	AT-300A-2M	67.8
233	AT-6T-3F	66.0
239	AT-RMTN-2M	52.2
241	AT-EDC-2M	64.8
246	AT-RSPG-2M	69.4
278	AT-YAKB-2M	67.8
281	AT-EDNA-2M	68.6
282	AT-200W-2M	999.9
283	AT-WAHL-2M	69.6
284	AT-WYEB-2M	66.2
285	AT-GABL-2M	63.0
286	AT-WPPS-2M	67.6
289	AT-SAGE-2M	64.2
297	AT-ARMY-2M	67.7
309	AT-FRNK-2M	64.3
310	AT-RING-2M	66.5
342	AT-200E-2M	66.8
346	AT-PROS-2M	68.2
347	AT-RICH-2M	65.3
348	AT-GBLW-2M	66.7
349	AT-100F-2M	66.5
380	AT-PASC-2M	66.3
391	AT-VERN-2M	66.1

FIGURE 3. An Example of a Variable (Dry Bulb Temperature) Plot from the REVIEW Computer Code

THE REVIEW COMPUTER CODE

The REVIEW computer program provides graphical and alphanumeric displays by plotting data obtained from the 15-minute and hourly meteorological databases maintained on the HMS and EMS computers. For each database you can plot either hourly or 15-minute data for an area [e.g., 400 Area (see Fig. 2)] or for a specific meteorological parameter [e.g., dry bulb temperature (see Fig. 3)].

PROGRAM OVERVIEW

The REVIEW computer program can plot the following variables using a Hanford Site map as background for the plot:

- dry bulb temperature
- dew point temperature
- delta temperature
- precipitation amounts
- telemetry wind network
- wind speed and direction at the 25-m level
- wind speed and direction at the 60-m level
- dry bulb temperature at the 10-m level
- dry bulb temperature at the 60-m level
- potential temperature.

Information from the meteorological towers located at the 100N Area, HMS, 400 Area, and 300 Area is also available to plot.

User input for the REVIEW code is facilitated by a data entry form that consists of several input fields (Fig. 4). A prompt for each field appears at the bottom of the entry form when you move the cursor to that field. You complete the field by typing your selection and depressing the RETURN key.

After exiting the data entry form, a graphical representation of the selected data is plotted on your terminal along with corresponding alphanumeric data. Appropriate headings indicate the date, time, and source of the selected data. Detailed information on how to execute and interact with the REVIEW computer code is presented in the next section.

PROGRAM REVIEW	
TYPE OF DATA	
Hourly: X	
15-Minute: X	
VARIABLE	AREA
Dry Bulb Temperature: X	100N: X
Dew Point Temperature: X	300A: X
Delta-Temperature: X	400A: X
Precipitation Amounts: X	622R: X
Telemetry Wind Network: X	
Wind Speed and Direction (25 m): X	
Wind Speed and Direction (60 m): X	
Dry Bulb Temperature (10 m): X	
Dry Bulb Temperature (60 m): X	
Potential Temperature: X	
DATE: XXXXXX	TIME: XXXX

FIGURE 4. Example of the Data Entry Form Used by the REVIEW Computer Code

USER'S GUIDE FOR THE REVIEW COMPUTER CODE

This section describes how to invoke the REVIEW computer code on the HMS and EMS computer systems and how to respond to the various prompts associated with the code. We assume that you know how to access the FCST account on the HMS computer or on the EMS computer at UDAC or at other Hanford emergency centers. The REVIEW computer code works only on Tektronix terminals (e.g., 4107, 4105). The terminal must be in the DEC VT100 terminal mode to use the data entry form and must have a page length of 24 lines. The EMS version of REVIEW automatically sets the terminal to VT100 and the page length to 24. To determine the terminal status, simply type SHOW TERMINAL and depress the RETURN key. A list of terminal characteristics appears on the screen. Check to make sure the terminal type is set to VT100 and the page length is set at 24 (top of screen). If the terminal is not in the VT100 mode, type SET TERMINAL/VT100 and depress the RETURN key. If the terminal

does not have a page length of 24, type SET TERMINAL/PAGE=24 and depress the RETURN key.

Basically, there are six steps to executing the REVIEW computer code, filling in the data entry form, and terminating the program on the HMS computer. The EMS version of REVIEW differs only in the first and second step. These steps are listed below, and each step is discussed in detail in the following paragraphs:

- Log in to the HMS computer (need the FCST account password), or log in to the EMS computer (need the local password).
- Type RUN REVIEW for the HMS version, then depress the RETURN key. For the EMS version, select R (Meteorological Parameter Display) from the menu.
- Fill in the data entry form.
- Exit the form.
- Answer "yes" or "no" to "would you care for another plot" prompt.
- Terminate program.

To log in to the HMS computer system, depress the RETURN key a couple of times. The HMS computer responds with the prompt USERNAME:. Enter FCST and depress the RETURN key. The computer then responds with the prompt PASSWORD. You must know the password to the FCST account before logging in to the HMS computer. Enter the correct password. If you fail to answer either of these prompts correctly, the computer repeats the appropriate prompt. You are given three chances to enter the correct response. When you successfully log in to the FCST account, you are placed in the [FCST] directory and can immediately invoke the REVIEW computer code. You must follow the same procedure to log in to the EMS system, although you will need the local USERNAME: and PASSWORD.

On the HMS computer, you invoke the REVIEW code by typing RUN REVIEW and depressing the RETURN key. On the EMS computer, you invoke the REVIEW code by selecting "R" or "r" from the EMS menu and depressing the RETURN key. When this is done, the screen clears and the data entry form appears on the terminal. You can now select the parameters you would like to plot. If

problems occur when executing the code, refer to the "Trouble-Shooting" section for help.

The data entry form appears on the screen as shown in Figure 4 with the cursor in field one (HOURLY). Adjacent to each field is a brief description of that field. When the cursor is moved to a new field, a prompt for the field appears on the screen below the data entry form. Included with each prompt is a default value and a valid range or list of matching values from which the input must be selected. The fields under the headings VARIABLE and AREA contain default values of "n." To enter input into a field, simply type in the requested response and depress the RETURN key. The default value may be entered by depressing the ENTER key or the SELECT DEFAULT key, which is found on some Tektronix terminals. The input value appears in the field (assuming a valid value or default value was entered) while the cursor moves from left to right into the next field. After data are entered in the rightmost field, the cursor moves down one line to the leftmost field. If corrections are needed use the up, down, right, and left arrow keys (e.g., ↑, ↓, →, ←) located at the top of the keyboard to move within the form. Once you are satisfied the entered values are correct, depress the EXIT key (the uppermost right key) to exit the form. You may exit the data entry form from any field. In the sample session given below, we assume you have logged in to the HMS or EMS computer system and have invoked the program:

- Under the heading TYPE OF DATA, select 15-MINUTE data by typing "n" or "N" in the HOURLY field and depressing the RETURN key. Next, move the cursor to the 15-MINUTE field by using the down arrow key (e.g., ↓). Type "y" or "Y" and depress the RETURN key.
- Under the VARIABLE heading, select dry bulb temperature data by moving the cursor to the DRY BULB TEMPERATURE field. Type "Y" or "y" and depress the RETURN key.
- Select the desired date by moving the cursor to the DATE field (use the down ↓ or up ↑ arrow key). Type the desired date (e.g., 880504 for May 4, 1988) and depress the RETURN key.

- Select the desired time by moving the cursor to the TIME field (use the right → arrow key). Type the desired time (e.g., 1400 for 2:00 p.m.) and depress the RETURN key. The completed form is shown in Figure 5.
- To exit the form and begin plotting, depress the EXIT key.

The REVIEW program provides two phases of quality assurance to the input data. The first phase is applied as you enter the input data for each field. An error message appears at the bottom of the form if the input value is not within the valid range or valid list of matching numbers, and you are then given the opportunity to enter another value for that field. The second phase occurs when you exit the form. As soon as you exit, the program will run a series of error checks. If an error is detected an appropriate message will be sent to your terminal. The program will then pause for approximately 10 seconds so you can read the message, then will return to the data entry form so you can correct the error. The arrow keys are used to move the cursor from field to field within the form.

PROGRAM REVIEW	
TYPE OF DATA	
Hourly: n	
15-Minute: y	
VARIABLE	AREA
Dry Bulb Temperature: y	100N: n
Dew Point Temperature: n	300A: n
Delta-Temperature: n	400A: n
Precipitation Amounts: n	622R: n
Telemetry Wind Network: n	
Wind Speed and Direction (25m): n	
Wind Speed and Direction (60m): n	
Dry Bulb Temperature (10m): n	
Dry Bulb Temperature (60m): n	
Potential Temperature: n	
DATE: 880504	TIME: 1400

FIGURE 5. Completed Data Entry Form from Example Session

After you exit the form, the program begins reading the selected data from an external disk file and then displays the data graphically and numerically. Plotting begins approximately 10 to 15 seconds after you depress the EXIT key.

The REVIEW program is terminated by answering "n" or "N" to the "would you like another plot" prompt. The message FORTRAN STOP is sent to your terminal. Additional information about each input field is discussed in the following section.

DESCRIPTION OF INPUT

This section gives detailed information about each field in the data entry form (see Fig. 4). The data entry form consists of 18 fields. You must provide an "n," "N," "y," or "Y" for the fields under the headings TYPE OF DATA, VARIABLE, and AREA. Only one field under the VARIABLE and AREA headings can contain a "y" or "Y" selection because plots of more than one variable or area, or combination of variable and area, are not allowed. In addition, either hourly or 15-minute data must be chosen under the TYPE OF DATA heading; you cannot choose both. Integer values are required for the DATE and TIME fields. When you move the cursor from field to field, a prompt appears at the bottom of the screen. The prompt consists of a brief description of the field, the valid range or list of matching numbers from which the input value must be selected, and a default value (refer to the "USER'S GUIDE" section for information on how to enter the default value). If you enter an invalid value for a field, an error message appears at the bottom of the form and the cursor remains in the same location so you can correct the value.

The following list describes each input field, the associated range or list of matching values from which the input must be selected, and the default value.

- 1) HOURLY - The input value for this field determines whether or not hourly data are used. The valid list of input values is "n," "N," "y," or "Y." If you desire hourly data, enter "y" or "Y" and depress the RETURN key. If you do not desire hourly data, enter "n" or "N" and depress the RETURN key. The default value for this field is "n."

- 2) 15-MINUTE - The input value for this field determines whether or not 15-minute data are used. The valid list of input values is "n," "N," "y," or "Y." If you desire 15-minute data, enter "y" or "Y" and depress the RETURN key. If you do not, enter "n" or "N" and depress the RETURN key. The default value for this field is "n."
- 3) DRY BULB TEMPERATURE - The input value for this field determines whether or not the dry bulb temperature ($^{\circ}\text{F}$) at 2-m (7 ft) locations is plotted. Dry bulb temperatures at 2-m can be obtained from each wind telemetry station (see Fig. 1). The valid list of input values is "n," "N," "y," or "Y." If you desire a plot of the dry bulb temperature data, enter "y" or "Y" and depress the RETURN key. If you do not, enter "n" or "N" and depress the RETURN key. The default value for this field is "n."
- 4) DEW POINT TEMPERATURE - The input value for this field determines whether or not the dew point temperature ($^{\circ}\text{F}$) at 2-m locations is plotted. Dew point temperature can be obtained from the 100N, 400, and 300 Areas. The data type, valid list of input values, default value, and method of selection are identical to that for field 3.
- 5) DELTA-TEMPERATURE - The input value for this field determines whether or not the delta-temperature ($^{\circ}\text{F}$) is plotted. Delta-temperature sensors are located at the 100N, 400, and 300 areas. The delta-temperature at the HMS is calculated from the 10-m (33-ft) and 60-m (197-ft) temperatures. The data type, valid list of input values, default value, and method of selection are identical to that for field 3.
- 6) PRECIPITATION AMOUNTS - The input value for this field determines whether or not precipitation amounts (inches) are plotted. Precipitation data are available from the wind telemetry stations at Rattlesnake Springs, Yakima Barricade, Pasco, and Richland. The data type, valid list of input values, default value, and method of selection are identical to that for field 3.
- 7) TELEMETRY WIND NETWORK - The input value for this field determines whether or not the wind speed (mi/hr) and direction (degrees from north) from the 10-m wind telemetry station network are plotted

- (see Fig. 1). The data type, valid list of input values, default value, and method of selection are identical to that for field 3.
- 8) WIND SPEED AND DIRECTION (25 m) - The input value for this field determines whether or not the wind speed (mi/hr) and direction (degrees from north) at the 25-m (82-ft) level are plotted. Wind speeds and direction at this level can be obtained from the 100N, 400, and 300 Areas and the HMS. The data type, valid list of input values, default value, and method of selection are identical to that for field 3.
 - 9) WIND SPEED AND DIRECTION (60 m) - The input value for this field determines whether or not the wind speed (mi/hr) and direction (degrees from north) at the 60-m level are plotted. Wind speeds and direction at this level can be obtained from the 100N, 400, and 300 Areas and the HMS. The data type, valid list of input values, default value, and method of selection are identical to that for field 3.
 - 10) DRY BULB TEMPERATURE (10 m) - The input value for this field determines whether or not the dry bulb temperatures at the 10-m level are plotted. Dry bulb temperatures at this level can be obtained from the 100N, 400, and 300 Areas and the HMS. The data type, valid list of input values, default value, and method of selection are identical to that for field 3.
 - 11) DRY BULB TEMPERATURE (60 m) - The input value for this field determines whether or not the dry bulb temperatures at the 60-m level are plotted. Dry bulb temperatures at this level can be obtained from the 100N, 400, and 300 Areas and the HMS. The data type, valid list of input values, default value, and method of selection is the same as that for field 3.
 - 12) POTENTIAL TEMPERATURE (2 m) - The input value for this field determines whether or not the potential temperature at the 2-m level is plotted. Potential temperatures at this level are calculated from the 2-m dry bulb temperature obtained from all the wind telemetry locations. Potential temperature can also be calculated from the station pressure at the HMS. The data type, valid list of input values, default value, and method of selection are identical to that for field 3.

- 13) 100N - The input value for this field determines whether or not data from the 200-ft tower at the 100N Area are plotted. The tower provides the following data:

- dry bulb temperature (°F) at the 2-, 10-, and 60-m level
- wind speed (mi/hr) and direction (degrees from north) at the 10-, 25-, and 60-m level
- dew point temperature (°F) at the 2-m level
- delta-temperature (°F) from the 60- and 10-m levels.

The data type, valid list of input values, default value, and method of selection are identical to that for field 3.

- 14) 300A - The input value for this field determines whether or not data from the 200-ft tower at the 300 Area are plotted. This tower provides the same information as the 100N 200-ft tower. The data type, valid list of input values, default value, and method of selection are identical to that for field 3.
- 15) 400A - The input value for this field determines whether or not data from the 200-ft tower at the 400 Area are plotted. This tower provides the same information as the 100N 200-ft tower. The data type, valid list of input values, default value, and method of selection are identical to that for field 3.
- 16) 622R - The input value for this field determines whether or not data from the 622R 410-ft tower are plotted. Although this tower provides several more levels of data than the 200-ft towers, the 622R plot displays the same information as the 100N, 300A, and 400A plots. The data type, valid list of input values, default value, and method of selection are identical to that for field 3.
- 17) DATE - The input value for this field is the desired date of the data to be displayed. The date format is YYMMDD: where YY is the last two digits of the current year (e.g., 87 for 1987), MM is the numerical equivalent of the month (e.g., 04 represents April), and DD is the day of the month. An example of a date is 870421, which is April 21, 1987. The range of the date is 860101 (i.e., January 1, 1986) through 990101 (i.e., January 1, 1999). The default value

for the date is 999999. (Refer to the User's Guide Section for details on entering default values.)

- 18) TIME - The input value for this field is the hour and minute of the desired day. The range of the hour is 1 through 24. The minute is listed as 00, 15, 30, or 45. When requesting hourly data a minute of 00 should always be entered. For example, an entry of 1400 indicates a desired time of 2:00 p.m.; that is, display information averaged over the time 1:00 p.m. through 2:00 p.m. For 15-minute data a minute of 00, 15, 30, or 45 can be entered. For example, an entry of 1415 indicates a desired time of 2:15 p.m.; that is, display information averaged over the time 2:00 p.m. through 2:15 p.m. A default value of 1300 can be used.

LIMITATIONS TO THE PROGRAM, DATABASE, AND COMPUTER SYSTEM

The REVIEW computer program does contain some limitations. The program plots information only from the EMS database, which is located on the HMS and EMS computer systems. The REVIEW computer program is capable of producing only one plot at a time (e.g., hourly data for a variable at a specific hour, 15-minute data for a variable at a specific 15-minute period, hourly data for an area at a specific hour, 15-minute data for an area at a specific 15-minute period).

In addition, the temporal availability of data is limited. The HMS and EMS computer systems keep 15-minute data for approximately 10 days, and hourly data for approximately 1 month. Hourly data for the most recently completed hour is not available until 20 minutes after the current hour. For example, 1400 hourly data (i.e., data averaged from 1300 through 1400) are not available until 1420. Fifteen-minute data for the most recently completed 15-minute period are not available until 20 minutes after that period. For example, 1415 15-minute data (i.e., data averaged from 1400 through 1415) are not available until 1435.

Although the REVIEW code filters much of the erroneous data, some spurious values may appear in the plots. A value that appears to be inconsistent with corresponding data should be ignored.

Finally, the HMS and EMS computer systems are time-sharing computers, and as the number of users increases the execution time for your process decreases.

1
2
3
4
5

6

7

8

9
10

11

TROUBLE-SHOOTING THE REVIEW COMPUTER CODE

This section provides information that may help resolve problems that occur when operating the REVIEW computer code. We assume that you are familiar with the VAX EDT editor and the VAX operating system commands. Basically, there are three types of failure or problem areas: terminal (i.e., CRT), system, and/or program. The following sections will discuss problems or failures that may occur when using the REVIEW computer code and the suggested recovery procedures. If a series of recovery actions are suggested, you should try them in sequence until the problems or failures are resolved. If you are using the EMS version of REVIEW and are experiencing problems, call the EMS system manager for help. The following problems and recovery procedures are for the HMS version of REVIEW.

TERMINAL PROBLEMS/FAILURES

Problem: Terminal does not respond.

Recovery: Determine if terminal is on. On Tektronix terminals the on/off switch is located in the front and to the right.

Determine if the dimmer switch is turned all the way down. On Tektronix terminals the dimmer switch is located in the front and to the right. Turn the switch clockwise to increase the light intensity on the screen.

If the above actions fail to get a response from the terminal, ask the instrument specialist (the HMS Room 125) to check the terminal.

SYSTEM PROBLEMS/FAILURES

Problem: System does not respond to operating system commands or carriage returns.

Recovery: Check with the EMS computer manager (Kelly Feuerbacher, 376-1107, or Ora Gifford, 373-3216) to see if the HMS computer system is down for maintenance or repair.

If the EMS computer manager indicates the HMS computer system is operating, turn the terminal off, wait approximately 15 to 20 seconds, and turn the terminal back on. Depress the RETURN key several times to get the attention of the computer system.

If the above actions fail ask an instrument specialist to examine the computer and notify the EMS computer manager.

PROGRAM PROBLEMS/FAILURES

Problem: Terminal not correctly displaying data entry form.

Recovery: Depress the CTRL key and the Y key simultaneously to get to the dollar sign prompt. Type SET TERMINAL/VT100 and depress RETURN. Try running the program again.

Depress the SETUP key at the top of the keyboard. The terminal should respond with an asterisk. Type CODE ANSI and depress RETURN, then depress the SETUP key. Try running the program again.

Problem: Terminal displaying garbled data.

Recovery: Depress the SETUP key at the top of the keyboard. The terminal should respond with an asterisk. Type CODE ANSI, depress RETURN, then depress the SETUP key.

Problem: Program REVIEW.EXE cannot be found or program has been "trashed" in some way.

Recovery: Copy backup version to the FCST directory. Type COPY [FCST.REVIEW]REVIEW.EXE [FCST]REVIEW.EXE. Try running the program again.

APPENDIX A

DESCRIPTION AND FLOWCHART OF THE REVIEW PROGRAM

APPENDIX A

DESCRIPTION AND FLOWCHART FOR THE REVIEW PROGRAM

REVIEW.FOR

REVIEW is able to pass control to the appropriate subroutines until the program is terminated by the user. The source codes are located in the [ERDSSYS.REVIEW] directory on the EMS system. At the HMS, the source codes are in [FCST.REVIEW], with a copy of the executable file located in [FCST]. Figure A.1 shows a flowchart for the REVIEW program.

The program starts by calling routines that create the data entry form. These routines are in a special library called ERDSLIB that can be found only on the two VAX 11/750 computers on the EMS system. The data entry form was created for use in REVIEW only.

After control is returned from the forms routines, the entered data are passed to ERROR_CHECK, which checks the data for errors. When control is returned to REVIEW from ERROR_CHECK, the error flag is checked. If there is an error, ERROR_CHECK displays error messages and REVIEW will interrupt execution by making a call to WAIT. The program then returns control to the data entry form. This process continues until errors are eliminated from the data.

When the user-entered data are error free, control is passed to a routine that will open and read from the appropriate parameter identification (ID) file. The ID files are located in the same directory as all the source codes ([ERDSSYS.REVIEW] on EMS, and [FCST.REVIEW] on HMS). There is one data file for each variable and area. Each file contains the EMS parameter IDs that pertain to the specific plot (e.g., TEMP_IDS.DAT contains all the parameter IDs for data in the EMS database that will be used in the dry bulb temperature plot). The ID files also contain coordinates for each station that are used when plotting symbols and data on the screen. All the parameter IDs and coordinates are stored in arrays for later use.

The parameter ID files are listed below:

DELTAT_IDS.DAT	Delta temperatures
DEWPT_IDS.DAT	Dew point temperatures
POTEMP_IDS.DAT	Potential temperatures (HMS only)
PRECIP_IDS.DAT	Precipitation amounts
TELEM_IDS.DAT	Wind speeds and directions for telemetry network
TEMP_IDS.DAT	Dry bulb temperatures
TEMP10M_IDS.DAT	Temperatures at 10 m
TEMP60M_IDS.DAT	Temperatures at 60 m
WND25M_IDS.DAT	Wind speeds and directions at 25 m
WND60M_IDS.DAT	Wind speeds and directions at 60 m
100N_IDS.DAT	100 Area information
300_IDS.DAT	300 Area information
400_IDS.DAT	400 Area information (FFTF)
622R_IDS.DAT	HMS information

Before actual meteorological data can be retrieved from the EMS database, the desired hour may require reformatting. The EMS database uses the 0 through 23 format for the hour and the date. The user will provide the desired hour in the format 1 through 24. The only hour value that differs between the two formats is 24. Control is passed to REFORM_HOUR only if the desired hour is 24. When control is returned to REVIEW, the hour and date have been reformatted appropriately to the EMS format.

At this point, the program can read data from the EMS database. All the pertinent information is sent to GET_EMS_DATA, where the database is searched for the correct date and the data are read. If an error tries to read from the database, a message is sent to the screen informing the user and control is returned to the beginning of the program. GET_EMS_DATA returns an array of all the desired data when control is returned, without an error, to REVIEW.

After all the desired data are received, the program is ready to pass control to a plotting routine. If a variable is chosen for the plot, control is passed to PLOT_VARIABLE. Otherwise, control is passed to PLOT_AREA. Within these two routines special cases are handled automatically (e.g., different types of data and different plotting formats).

When the desired plot is finished, control is returned to REVIEW and the user is then asked if another plot is desired. If the answer is "Y" ("yes"), the program returns control to the data entry form routines. If the answer is "N" ("no"), the program simply stops execution and the system prompt (\$) becomes visible.

The EMS version of REVIEW differs from the HMS version in that the potential temperatures cannot be plotted, and an error message will appear if plotting is attempted. Atmospheric pressure is needed to calculate potential temperatures, and this value is only found in the hourly surface observation data files. The EMS system cannot access those files; however, the HMS version can access the surface observation data because the data are stored on the HMS computer.

FIGURE A.1. Flowchart for the START.REVIEW Subroutine

APPENDIX B

DESCRIPTIONS AND FLOWCHARTS OF REVIEW SUBROUTINES

APPENDIX B

DESCRIPTIONS AND FLOWCHARTS OF REVIEW SUBROUTINES

This appendix describes the subroutines used in the REVIEW computer program. The hierarchy of the REVIEW subroutines is shown in Figure B.1. The subroutines described here are presented in alphabetical order. Each description states the purpose of the subroutine, describes the general sequence of the code, and states the result of the subroutine.

Subroutines associated with the TRPLOT and ERDSLIB libraries will not be discussed. The TRPLOT graphics software has been documented in TRPLOT Graphics Library User's Guide (TRPLOT Version 1.0, December 1984). Subroutines associated with the ERDSLIB library are not documented.

ERROR_CHECK

The main purpose of this subroutine is to check all the information the user enters on the data entry form. A flowchart for the ERROR_CHECK subroutine is shown in Figure B.2.

Extensive error checking prevents errors from occurring during further execution. The date and time are checked for correctness and validity. The subroutine also ensures that only one area or variable was chosen to plot and that only one type of data were selected (i.e., either hourly or 15-minute data). Descriptive error messages are displayed on the screen if there is an error, and the user can correct the information.

The character form of the date and time, a variable indicating which area or variable to plot, and another variable signifying whether to use 15-minute or hourly data are all returned to REVIEW at the completion of this subroutine.

FIGURE B.1. Hierarchy of Subroutines Within the REVIEW Computer Code

FIGURE B.2. Flowchart for the ERROR_CHECK Subroutine

GET_EMS_DATA

The objective of this subroutine is to obtain EMS data from either the 15-minute or hourly data base. A flowchart for the GET_EMS_DATA subroutine is shown in Figure B.3.

The desired date and time, along with an array of desired parameter IDs (see Fig. 1 in the main text), are passed into this subroutine. A variable that indicates which type of data to read is also passed to the subroutine. First, GET_EMS_DATA makes a call to either GET_HLG_FILENAME or GET_QLG_FILENAME, which will construct the name of the file to be opened and read. While reading from that file, the data are stored in an array called DESIRED_DATA.

When control is returned to REVIEW at the completion of this subroutine, DESIRED_DATA will contain all the data required for the chosen plot.

GET_HLG_FILENAME AND GET_QLG_FILENAME

Both of these subroutines are called from GET_EMS_DATA and are used to construct the appropriate filename from which to read (e.g., hourly or 15-minute data). A flowchart for the GET_HLG_FILENAME and GET_QLG_FILENAME subroutines is shown in Figure B.4.

GET_HLG_FILENAME is called when the user requests hourly data from the database, and GET_QLG_FILENAME is called when 15-minute data are requested. The desired date and time are passed into the subroutines when called. Only one of these subroutines can be called for each plot performed.

When these subroutines are completed, control is returned to GET_EMS_DATA, along with the appropriate file name.

GET_MONTH

This subroutine simply returns a three-character abbreviation for the month that the user identified. A flowchart for the GET_MONTH subroutine is shown in Figure B.5.

FIGURE B.3. Flowchart for the GET_EMS_DATA.FOR Subroutine

FIGURE B.4. Flowchart for the GET_QLG_FILENAME.FOR/GET_HLG_FILENAME.FOR Subroutine

FIGURE B.5. Flowchart for the GET_MONTH.FOR Subroutine

This subroutine is called from GET_HLG_FILENAME and GET_QLG_FILENAME to confirm the integer value of the month that the user identified. The character abbreviation that the subroutine returns is used in constructing a data file name.

When the subroutine is completed, control is returned to the appropriate calling routine (i.e., either GET_HLG_FILENAME or GET_QLG_FILENAME).

GET_STABILITY

This subroutine determines the atmospheric stability using the delta temperature scheme developed by the Nuclear Regulatory Commission(a). A flowchart for the GET_STABILITY subroutine is shown in Figure B.6.

Delta temperature is passed to this subroutine as a real number from the routine PLOT_VAR. A series of checks are made and the appropriate integer

(a) Snell. 1982. Nuclear Regulatory Commission Staff Computer Programs for Use with Meteorological Data. NUREG-0917, U.S. Nuclear Regulatory Commission, Washington, D.C.

FIGURE B.6. Flowchart for the GET_STABILITY.FOR Subroutine

and character forms of the stability are assigned to the variables. The integer and character stability codes are returned to PLOT_VAR, as well as control of program execution, when the subroutine is complete.

GET_THETA

This subroutine is designed to calculate potential temperatures from the dry bulb temperature at various locations (i.e., telemetry stations) on the Hanford Site. A flowchart for the GET_THETA subroutine is shown in Figure B.7.

The desired date and the temperatures for that date are passed to the subroutine. The appropriate surface observation file is opened and the surface pressure is read. The potential temperatures are then calculated.

At the completion of this subroutine, an array containing all the potential temperatures is returned to PLOT_VAR, where execution proceeds.

OPEN_FILE

OPEN_FILE opens and reads the numeric ID, alphanumeric ID, and the map coordinates from the desired ID file. A flowchart for the OPEN_FILE subroutine is shown in Figure B.8.

The identity of the file that must be opened and read is passed to this subroutine from REVIEW. The information that we read is stored in arrays that are then returned to REVIEW.

The main program will now be able to retrieve the desired information when control is returned to REVIEW.

PLOT_AREA

The objective of this subroutine is to plot the meteorological information collected from the 200-ft tower for an area (e.g., 400, 300, 100, or 622R) selected by the user. A flowchart for the PLOT_AREA subroutine is shown in Figure B.9.

FIGURE B.7. Flowchart for the GET_THETA.FOR Subroutine

FIGURE B.8. Flowchart for the OPEN_FILE.FOR Subroutine

FIGURE B.9. Flowchart for the PLOT_AREA.FOR Subroutine

The subroutine begins by plotting a title, the date and time of the data, and other pertinent labels that precisely describe the plot. A two-dimensional plot of the temperature, wind direction vectors, and columns of raw data is also plotted on the screen.

When the subroutine is completed, the full area plot is on the screen, and control is returned to REVIEW.

PLOT_NORTH

The purpose of this subroutine is to plot a north indicator near any wind direction vectors not located on a background map. A flowchart for the PLOT_NORTH subroutine is shown in Figure B.10.

The only information this subroutine needs is the scale and the x and y coordinates. These data are located in the data file NORTH.DAT. PLOT_AREA opens and reads this information before passing it to PLOT_NORTH. The x and y coordinates, NORTHX and NORTHY, are the actual position on the screen in

FIGURE B.10. Flowchart for the PLOT_NORTH.FOR Subroutine

inches. The scale, S, is the size of the indicator in inches. An N, with an arrow intercept, is plotted at position NORTHX, NORTHY and of size S.

At the completion of this plotting subroutine, the north indicator appears on the screen and control returns to PLOT_AREA.

PLOT_TEMPS

This subroutine is used to plot a vertical line graph of the three tower temperatures for the chosen area. A flowchart for the PLOT_TEMPS subroutine is shown in Figure B.11.

The code automatically provides a scale and label for the x axis, given the temperatures that are passed in (i.e., either TEMP1, TEMP2, and TEMP3). The y-axis coordinates for the points on the graph are constant, while the x-axis coordinates are calculated. Symbols are then plotted at the calculated positions, and a line is plotted to connect the three symbols.

After completing the graph, control is returned to PLOT_AREA, the calling subroutine.

PLOT_VAR

The purpose of this subroutine is to plot the desired meteorological parameter (e.g., 3-m temperatures, 60-m wind speeds and direction, etc.) on a background map of the Hanford Area. A flowchart for the PLOT_VAR subroutine is shown in Figure B.12.

When REVIEW calls this subroutines, two arrays are passed: one that contains the desired data and another with the coordinates for the station that recorded the data. Descriptive headings and labels are plotted on the screen.

When all the data are plotted, the execution of this subroutine is complete, and control is returned to REVIEW.

FIGURE B.11. Flowchart for the PLOT_TEMPS.FOR Subroutine

FIGURE B.12. Flowchart for the PLOT_VAR.FOR Subroutine

PLOT_VECTORS

This subroutine is called from both PLOT_VAR and PLOT_AREA, and is used to plot wind direction vectors at any of the specified locations on the screen. A flowchart for the PLOT_VECTORS subroutine is shown in Figure B.13.

The calling subroutine passes an array containing the wind directions and speeds. Another array that is passed to PLOT_VECTOR contains the coordinates for the location of the vectors on the screen. Using the wind speeds, the magnitude of the vector is calculated. An arrow is then plotted that originates at the station coordinates and points in the direction that the wind is blowing.

When the subroutine is complete, three vectors will be plotted for the area plots, or twenty-five vectors for the variable plot. Control is then returned to the calling subroutine, either PLOT_VAR or PLOT_AREA.

PLT_GRAPHICS

The purpose of this subroutine is to read digitized information for the Hanford Area Map background used in all variable plots. A flowchart for the PLT_GRAPHICS subroutine is shown in Figure B.14.

The input file, MESOI_GRAPHICS.DAT, contains the information to plot roads, rivers, and the Hanford boundary. Coordinates and pen actions are read and passed back to the calling routine, where the information is then plotted.

The map information will be read in only once per execution of REVIEW. Control is returned to PLOT_VARIABLE upon completion.

REFORM_HOUR

This subroutine is used to reformat the desired hour from a 1 to 24 format to a 0 to 23 format. A flowchart for the REFORM_HOUR subroutine is shown in Figure B.15.

FIGURE B.13. Flowchart for the PLOT_VECTORS.FOR Subroutine

FIGURE B.14. Flowchart for the PLT_GRAPHICS.FOR Subroutine

FIGURE B.15. Flowchart for the REFORM_HOUR.FOR Subroutine

The desired hour is passed to the subroutine from REVIEW if the hour is equal to 24, which is the only hour that needs to be reformatted. Appropriate checks for leap year, end of year, and end of month are performed.

When the subroutine is complete, the hour is reformatted and control is returned to REVIEW.

WAIT

This subroutine provides a method to suspend execution of the program so error messages on the screen can be read. A flowchart for the WAIT subroutine is shown in Figure B.16.

The subroutine gets the system time by making a call to TIME, a Fortran routine. TIME returns the current hour, minute, and seconds. The delta time, or amount of time that has passed, is calculated by calling another Fortran subroutine, SECNDS. This routine passes the current time and automatically returns a delta time. After the delta time is calculated, it is then compared to the desired waiting time.

If delta time is greater than WAIT_TIME, execution of the calling routine, either REVIEW or GET_THETA, proceeds. Otherwise, delta time is recalculated and continually checked until it is large enough.

FIGURE B.16. Flowchart for the WAIT.FOR Subroutine

APPENDIX C

REVIEW PROGRAM AND SUBROUTINE SOURCE CODE LISTINGS

```

0001  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002  C
0003  C      REVIEW.FOR
0004  C
0005  C      WRITTEN BY:  KEN BURK      378-5913
0006  C                  GREGG ANDREWS 378-6164
0007  C
0008  C      PURPOSE:  THIS PROGRAM IS GOING TO BE USED TO PLOT
0009  C                  HOURLY AND 15 MINUTE DATA.  THE USER CAN
0010  C                  CHOOSE BETWEEN FOUR DIFFERENT AREAS FOR
0011  C                  AN AREA PLOT.  WHEN DOING AN AREA PLOT,
0012  C                  TEMPERATURES, WIND SPEEDS, AND WIND DIRECTIONS
0013  C                  ARE GRAPHED AND PLOTTED.  THE WIND DATA WILL
0014  C                  BE IN THE FORM OF VECTORS.  THE USER WILL ALSO
0015  C                  BE ABLE TO CHOOSE BETWEEN SEVERAL DIFFERENT METEOROLOGICAL
0016  C                  VARIABLES TO DO VARIABLE PLOTS ON THE BACKGROUND
0017  C                  MAP.  ON BOTH OF THESE PLOTS, THE ACTUAL PARAMETER
0018  C                  VALUES ARE DISPLAYED, AS WELL AS THE GRAPHIC
0019  C                  REPRESENTATION.
0020  C
0021  C      ALGORITHM :  DECLARE VARIABLES
0022  C                  PUT DATA-ENTRY FORM ON SCREEN
0023  C                  DO ERROR CHECKING ON USER INPUT
0024  C                  OPEN AND READ APPROPRIATE ID FILE
0025  C                  REFORMAT THE HOUR IF NEEDED
0026  C                  GO TO THE EMS DATA BASE AND GET DESIRED DATA
0027  C                  PLOT THE DATA
0028  C                  ASK USER IF ANOTHER PLOT IS NEEDED
0029  C                  END ROUTINE
0030  C
0031  C
0032  C      CALLS TO :  ERROR CHECK.FOR
0033  C                  OPEN FILE.FOR
0034  C                  REFORM HOUR.FOR
0035  C                  GET EMS DATA.FOR
0036  C                  PLOT AREA.FOR
0037  C                  PLOT VAR.FOR
0038  C                  WAIT.FOR
0039  C                  POT.FOR
0040  C                  PST_ADJUSTMENT
0041  C
0042  C      VERSION :   1.0
0043  C                  1.1 MODIFIED TO ADJUST FOR PACIFIC STANDARD TIME,
0044  C                      ADD NEW GRAPHICS DATA POINTS TO SPEED UP, AND
0045  C                      ELIMINATE THE PAUSE IN THE GRAPHICS ROUTINES
0046  C                      PLOT VAR, PLOT AREA, PLT WNDPLT, AND
0047  C                      PLT_GRAPHICS.  G. ANDREWS 8/2/88
0048  C
0049  C      DATE :  FEBRUARY 1988
0050  C
0051  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0052  C
0053  C
0054  C      INCLUDE      'FORMFLAGS.INC'
0055  C
0056  C      VARIABLE DECLARATIONS
0057  C
0058  C      TYPE | VARIABLE | COMMENTS
0059  C      -----|-----|-----
0060  C
0061  C      INTEGER*2  xdate(3)      ! Integer form of desired date
0062  C      INTEGER*2  xtime(2)      ! Integer form of desired hour
0063  C      INTEGER*2  data flag      ! Flag for hourly or 15-minute data
0064  C      INTEGER*2  number_of_parameters ! The number of parameters for the chosen plot

```

```

0103      INTEGER*2      i                ! Array and loop index
0104      INTEGER*2      data_type         ! The variable or area chosen by user
0105      INTEGER*4      parameter_id(300) ! The EMS internal ids for the desired data
0106      INTEGER*4      save_mon          ! desired month used in pdt
0107      INTEGER*4      save_day          ! desired day used in pdt
0108      INTEGER*4      save_hour         ! desired hour used in pdt
0109
0110      REAL*4          xsta(60)          ! X coordinate of a station
0111      REAL*4          ysta(60)          ! Y coordinate of a station
0112      REAL*4          xdesired_data(300) ! The actual data read from EMS
0113
0114      CHARACTER*1      answer           ! Question answer Used for data checks
0115      CHARACTER*1      more            ! Answer used for more plots
0116      CHARACTER*10     alpha_id(200)   ! Alphanumeric ID for the EMS parameter
0117      CHARACTER*2      cdate(3)        ! Character form of the date
0118      CHARACTER*2      ctme(2)         ! Character form of the time
0119
0120      LOGICAL          read_data        ! True if map data has already been read in, saves time
0121      LOGICAL          readerr         ! Boolean for an error during the EMS file read
0122      LOGICAL          dataerr         ! Error in data read in from data entry form.
0123      LOGICAL          flag            ! Flag to indicate if pdt
0124
0125      C
0126      C      DECLARATIONS FOR THE SPECIAL VARIABLES IN THE DATA-ENTRY FORM
0127      C
0128
0129
0130      Bytes          Form_Interface_Record( 24)
0131      Integer*4      Iflag( 18)        ! Form flags
0132      CHARACTER*1    c_var001          ! Enter y or Y for EMS Hourly Data
0133      CHARACTER*1    c_var002          ! Enter y or Y for EMS 15-Minute Data
0134      CHARACTER*1    c_var003          ! Enter y or Y for Dry Bulb Temperature
0135      CHARACTER*1    c_var004          ! Enter y or Y for Dew Point Temperature
0136      CHARACTER*1    c_var005          ! Enter y or Y for Delta-Temperature
0137      CHARACTER*1    c_var006          ! Enter y or Y for Precipitation Amounts
0138      CHARACTER*1    c_var007          ! Enter y or Y for Telemetry Wind Fields
0139      CHARACTER*1    c_var008          ! Enter y or Y for Wind Speed and Direction (25m)
0140      CHARACTER*1    c_var009          ! Enter y or Y for Wind Speed and Direction (60m)
0141      CHARACTER*1    c_var010          ! Enter y or Y for Dry Bulb Temperature (10m)
0142      CHARACTER*1    c_var011          ! Enter y or Y for Dry Bulb Temperature (60m)
0143      CHARACTER*1    c_var012          ! Enter y or Y for 100N Data
0144      CHARACTER*1    c_var013          ! Enter y or Y for 300A Data
0145      CHARACTER*1    c_var014          ! Enter y or Y for 400A Data
0146      CHARACTER*1    c_var015          ! Enter y or Y for 822R Data
0147      CHARACTER*1    c_var016          ! Enter y or Y for Potential Temperatures
0148      INTEGER*4      i_var001          ! Enter the Desired Date
0149      INTEGER*4      i_var002          ! Enter Desired Time
0150
0151      EQUIVALENCE (Form_Interface_Record, c_var001)
0152
0153      COMMON /Form_Record/ c_var001, c_var002, c_var003, c_var004,
0154      .                   c_var005, c_var006, c_var007, c_var008,
0155      .                   c_var009, c_var010, c_var011, c_var012,
0156      .                   c_var013, c_var014, c_var015, i_var001,
0157      .                   i_var002, c_var016
0158
0159      C
0160      C      PUT DATA-ENTRY FORM ON THE SCREEN FOR USER TO ENTER DATA
0161      C
0162      C
0163
0164      1      DO i = 1,15                ! SET ALL ANSWERS FOR VARIABLES
0165      .       form_interface_record(i) = 'n' ! ON DATA-ENTRY FORM, TO NO='n'
0166      .     ENDDO
0167      .     form_interface_record(24) = 'n'

```

```

0168
0169      iflags(1) = FMS_FORCE_START      ! FORCES CURSOR TO FIRST POSITION ON FORM
0170
0171      CALL Subprocess_Open_IO
0172
0173      CALL Select_ANSI
0174
0175      CALL FM_Form_Open (33, 'HMSFORMS.FRM', IError)
0176
0177      CALL FM_Form_Input ('REVIEW', Form_Interface_Record,
0178                        iflags, Ierror, User_Check, *999)
0179
0180      CALL FM_Form_Close (IError)
0181
0182      CALL Subprocess_Close_IO
0183
0184
0185      C
0186      C      DO ERROR CHECKING ON DATA FROM DATA-ENTRY FORM
0187      C
0188
0189      dataerr = .FALSE.
0190
0191      CALL error_check(xtime,xdate,ctime,cdate,data_type,data_flag,dataerr)
0192
0193      IF (dataerr) THEN
0194          CALL wait(7.0)      ! WAIT SO USER CAN READ ERROR MESSAGES
0195          GOTO 1
0196      ENDIF
0197
0198      C
0199      C      OPEN AND READ EMS ID's FROM APPROPRIATE DATA FILE
0200      C
0201
0202      CALL open_file(data_type,parameter_id,alpha_id,
0203                    +      number_of_parameters,xata,yata)
0204
0205
0206      C
0207      C      CONVERT THE HOUR FORMAT FROM 1-24 TO 0-23
0208      C      THE USER WILL USE THE 1-24 FORMAT, BUT THE EMS
0209      C      DATABASE EXPECTS 0-23 FORMAT. THE ONLY HOUR THAT
0210      C      CHANGES IS 24.
0211      C
0212
0213      c      IF(xtime(1) .EQ. 24) THEN      ! EMS VERSION STAYS WITH 0-23 HOUR FORMAT
0214      c          xtime(1) = 0
0215      c          CALL reform_hour(xdate)
0216      c      ENDIF
0217
0218      C
0219      C      GO TO THE EMS DATABASE AND GET THE DESIRED DATA
0220      C
0221
0222      TYPE*, ' '
0223      TYPE*, ' Begin Reading Data . . . '
0224
0225      DO i = 1,300
0226          xdesired_data(i) = 0.0      ! INITIALIZE DATA ARRAY
0227      ENDDO
0228
0229      readerr = .FALSE.
0230      C
0231      C..... FOLLOWING CODE ADDED TO COVERT FRO POT TO PST IF NECESSARY;
0232      C..... PREVIOUS VALUES OF MONTH, DAY, AND HOUR ARE SAVED

```

```

0233 C      CALL pdt(xdate(2), xdate(3), flag)
0234      IF(flag) THEN
0235          save_mon = xdate(2)
0236          save_day = xdate(3)
0237          save_hour = xtime(1)
0238          CALL pat_adjustment(xdate(2), xdate(3), xtime(1))
0239      ENDIF
0240
0241      CALL get_eme_data(xdate, xtime, data_flag, parameter_id,
0242                      number_of_parameters, xdesired_data, readerr)
0243
0244      xdate(2) = save_mon
0245      xdate(3) = save_day
0246      xtime(1) = save_hour
0247
0248
0249
0250
0251      IF(readerr) THEN
0252          type = 'Read error or, file not found'
0253          CALL wait(5.0)
0254          goto 1
0255      ENDIF
0256
0257 C
0258 C      PLOT THE DESIRED DATA USING TRIPLOT SOFTWARE.
0259 C      TRIPLOT GRAPHICS LIBRARY, VERSION 1.0.
0260 C
0261
0262
0263      IF(( data_type .LT. 10 ) .OR. ( data_type .GT. 13 )) THEN
0264          CALL plot_var(read_data, data_type, xdesired_data, alpha_id, parameter_id,
0265                      number_of_parameters, xsta, ysta, cdate, ctime, data_flag)
0266      ELSE
0267          CALL plot_area(data_type, xdesired_data, alpha_id, parameter_id,
0268                      number_of_parameters, cdate, ctime, data_flag)
0269      ENDIF
0270
0271 990 TYPE *, ' '
0272 TYPE *, ' '
0273 WRITE(0,200)
0274 200 FORMAT(3X, ' Do Another Plot ? [ Y or N ] --> ', $)
0275 READ(*, '(A1)') more
0276
0277 IF((more .EQ. 'Y') .OR. (more .EQ. 'y')) GOTO 1
0278
0279 STOP
0280 END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	527	PIC COM REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	130	PIC COM REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	6636	PIC COM REL LCL NOSHR NOEXE RD WRT LONG
3 FORM_RECORD	24	PIC OVR REL GBL SHR NOEXE RD WRT LONG
Total Space Allocated	7317	

ENTRY POINTS

Address	Type	Name	References
0-00000000		REVIEW\$MAIN	

VARIABLES

Address	Type	Name	Attributes	References
**	CHAR	ANSWER		114
3-00000000	CHAR	C_VAR001	COMM EQUIV	132 151 153
3-00000001	CHAR	C_VAR002	COMM EQUIV	133 153
3-00000002	CHAR	C_VAR003	COMM EQUIV	134 153
3-00000003	CHAR	C_VAR004	COMM EQUIV	135 153
3-00000004	CHAR	C_VAR005	COMM EQUIV	136 153
3-00000005	CHAR	C_VAR006	COMM EQUIV	137 153
3-00000006	CHAR	C_VAR007	COMM EQUIV	138 153
3-00000007	CHAR	C_VAR008	COMM EQUIV	139 153
3-00000008	CHAR	C_VAR009	COMM EQUIV	140 153
3-00000009	CHAR	C_VAR010	COMM EQUIV	141 153
3-0000000A	CHAR	C_VAR011	COMM EQUIV	142 153
3-0000000B	CHAR	C_VAR012	COMM EQUIV	143 153
3-0000000C	CHAR	C_VAR013	COMM EQUIV	144 153
3-0000000D	CHAR	C_VAR014	COMM EQUIV	145 153
3-0000000E	CHAR	C_VAR015	COMM EQUIV	146 153
3-00000017	CHAR	C_VAR016	COMM EQUIV	147 153
2-00001838	L*4	DATAERR		122 189= 191A 193
2-0000181E	I*2	DATA_FLAG		101 191A 242A 264A 267A
2-00001822	I*2	DATA_TYPE		104 191A 202A 263(2) 264A
267A				
2-0000183C	L*4	FLAG		123 234A 235
**	I*2	I		103 104= 165 225= 226
2-00001844	I*4	IERROR		177A
2-00001840	I*4	IDERROR		175A 180A
3-0000000F	I*4	I_VAR001	COMM EQUIV	148 153
3-00000013	I*4	I_VAR002	COMM EQUIV	149 153
2-0000181C	CHAR	MORE		115 276= 277(2)
2-00001820	I*2	NUMBER_OF_PARAMETERS		102 202A 242A 264A 267A
2-00001834	L*4	READERR		121 229= 242A 251
2-00001830	L*4	READ_DATA		120 264A
2-00001828	I*4	SAVE_DAY		107 237= 248
2-0000182C	I*4	SAVE_HOUR		108 238= 247
2-00001824	I*4	SAVE_MON		106 236= 245
2-00001848	R*4	USER_CHECK		177A

ARRAYS

Address	Type	Name	Attributes	Bytes	Dimensions	References
2-00000092	CHAR	ALPHA_ID		3200	(200)	116 202A
264A						
2-00001812	CHAR	CDATE		6	(3)	117 191A
264A						
2-00001818	CHAR	CTIME		4	(2)	118 191A

264A

3-00000000 L+1 FORM_INTERFACE_RECORD
165=

2-00000040 I+4 IFLAGS

177A

2-00000000 I+4 PARAMETER_ID

242A

2-00000000 I+2 XDATE

234(2)A

2-00000000 R+4 XDESIRED_DATA

242A

2-00000040 R+4 XSTA

264A

2-00000000 I+2 XTIME

238

267A

COMM EQUIV

24 (24)

130

151

167= 177A

72 (18)

131

169=

1200 (300)

105

202A

264A 267A

8 (3)

99

191A

235

237

239(2)A 242A

245=

245=

1200 (300)

112

226=

264A 267A

240 (60)

110

202A

4 (2)

100

191A

239A

242A

247=

REVIEW\$MAIN

Page 7

13-Jun-1988 13:59:54

VAX FORTRAN V4.7-271

3-Jun-1988 10:33:48

SYS\$SYSDEVICE:[BURK.LIS]REVIEW.FOR;102

2-00000050 R+4 YSTA

264A

240 (60)

111

202A

PARAMETER CONSTANTS

Type Name

References

I+4 FMS FIELD BOLD

78

82

I+4 FMS FIELD UNDERLINE

78

82

I+4 FMS FORCE START

78

82

169

I+4 FMS NO ECHO

78

82

I+4 FMS NO SEE

78

82

I+4 FMS NO WRITE

78

82

I+4 FMS SET ERROR

78

82

I+4 FMS SET EXCEPTION

78

82

I+4 FMS USER CHECK

78

82

LABELS

Address	Label	References
0-00000014	1	164# 195 264 277
1-00000057	200'	273 274#
0-00000185	990	177A 271#

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	References
	ERROR CHECK	191
	FW FORM CLOSE	180
	FW FORM INPUT	177
	FW FORM OPEN	175
	GET_EMS_DATA	242
	OPEN FILE	282
	PDT	234
	PLOT AREA	287
	PLOT VAR	284
	PST ADJUSTMENT	239
	SELECT ANSI	173
	SUBPROCESS CLOSE IO	182
	SUBPROCESS OPEN IO	171
	WAIT	194 263

KEY TO REFERENCE FLAGS	
=	- Value Modified
#	- Defining Reference
A	- Actual Argument, possibly modified
D	- Data Initialization
(n)	- Number of occurrences on line

COMMAND QUALIFIERS

FOR/LIS/EXTEND/CROSS/CHECK=ALL REVIEW

/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)
 /DEBUG=(NOSYMBOLS,TRACEBACK)
 /STANDARD=(NOSYNTAX,NOSOURCE FORM)
 /SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP,NODICTIONARY,SINGLE)
 /WARNINGS=(GENERAL,NODECLARATIONS,NOLTRIX)
 /CONTINUATIONS=19 /CROSS REFERENCE /NOO LINES /EXTEND SOURCE /F77
 /NOQ_FLOATING /I4 /NOMACHINE_CODE /OPTIMIZE /NOANALYSIS

COMPILATION STATISTICS

Run Time: 4.98 seconds
 Elapsed Time: 5.92 seconds
 Page Faults: 857
 Dynamic Memory: 386 pages

```

0001  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002  C
0003  C      ERROR_CHECK.FOR
0004  C
0005  C      WRITTEN BY:  K.W. BURK
0006  C                  G.L. ANDREWS
0007  C
0008  C      PURPOSE :  TO DO SOME ERROR CHECKING ON THE DATA
0009  C                  THAT IS INPUT TO THE DATA ENTRY FORM.
0010  C                  THIS ROUTINE WILL ALSO RETURN :
0011  C                      - THE CHARACTER FORM OF THE DATE AND TIME
0012  C                      - WHICH VARIABLE OR AREA TO GET DATA FOR
0013  C                      - WHAT KIND OF DATA, HOURLY OR 15-MINUTE
0014  C
0015  C      ALGORITHM :  DECLARE VARIABLES
0016  C                      CHECK HOURLY AND 15-MINUTE FLAGS NOT EQUAL
0017  C                      ASSIGN DATA FLAG
0018  C                      CONVERT CHARACTER FLAGS TO INTEGER ARRAY
0019  C                      Y OR y = 1, ELSE =0
0020  C                      CHECK THAT ONLY ONE ELEMENT =1
0021  C                      ASSIGN DATA TYPE TO THAT ONE
0022  C                      CHECK DATE AND TIME FOR ERRORS
0023  C                      CALCULATE CHARACTER FORM OF DATE AND TIME
0024  C                      END OF SUBROUTINE
0025  C
0026  C      VERSION :  1.0
0027  C
0028  C      DATE :    FEBRUARY 1988
0029  C
0030  C      CALLED FROM :  MAIN PROGRAM  REVIEW.FOR
0031  C
0032  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0033  C
0034  C      SUBROUTINE error_check(hrmin,yyradd,chrmin,cyyradd,which_var,
0035  C                          hr15,formerr)
0036  C
0037  C      IMPLICIT NONE
0038  C
0039  C
0040  C      VARIABLE DECLARATIONS
0041  C
0042  C      -----
0043  C      TYPE          |  VARIABLE          |  COMMENTS
0044  C      -----
0045  C      CHARACTER*1    c_var001      ! Enter y or Y for EMS Hourly Data
0046  C      CHARACTER*1    c_var002      ! Enter y or Y for EMS 15-Minute Data
0047  C      CHARACTER*1    c_var003      ! Enter y or Y for Dry Bulb Temperature
0048  C      CHARACTER*1    c_var004      ! Enter y or Y for Dew Point Temperature
0049  C      CHARACTER*1    c_var005      ! Enter y or Y for Delta-Temperature
0050  C      CHARACTER*1    c_var006      ! Enter y or Y for Precipitation Amounts
0051  C      CHARACTER*1    c_var007      ! Enter y or Y for Telemetry Wind Data
0052  C      CHARACTER*1    c_var008      ! Enter y or Y for Wind Speed and Direction
0053  C      CHARACTER*1    c_var009      ! Enter y or Y for Wind Speed and Direction
0054  C      CHARACTER*1    c_var010      ! Enter y or Y for Dry Bulb Temperature
0055  C      CHARACTER*1    c_var011      ! Enter y or Y for Dry Bulb Temperature
0056  C      CHARACTER*1    c_var012      ! Enter y or Y for 100W Data
0057  C      CHARACTER*1    c_var013      ! Enter y or Y for 300A Data
0058  C      CHARACTER*1    c_var014      ! Enter y or Y for 400A Data
0059  C      CHARACTER*1    c_var015      ! Enter y or Y for 822R Data
0060  C      CHARACTER*1    c_var016      ! Enter y or Y for Potential Temperatures
0061  C      CHARACTER*2    chrmin(2)     ! Character array form of the time
0062  C      CHARACTER*2    cyyradd(3)    ! Character array form of the date
0063  C      CHARACTER*2    syshour       ! Systems current hour
0064  C      CHARACTER*2    sysmin        ! Systems current minute
0065  C      CHARACTER*4    alltime       ! Four byte variable for the time

```

```

0066 CHARACTER*6 alldate      ! Six byte variable for the date
0067 CHARACTER*6 systime     ! Current system time
0068
0069 INTEGER*2 yyymmdd(3)     ! Integer form of date
0070 INTEGER*2 hrmn(2)        ! Integer form of time
0071 INTEGER*2 which_var      ! Which variable or area
0072 INTEGER*2 hr15           ! Hourly or 15-minute data
0073 INTEGER*2 i              ! Array index
0074 INTEGER*2 temp(14)       ! Temporary array
0075 INTEGER*2 lastday(12)    ! Last day of each month
0076 INTEGER*2 sysyear       ! Current system year
0077 INTEGER*2 sysmonth      ! Current system month
0078 INTEGER*2 sysday        ! Current system day
0079 INTEGER*4 i_var001      ! Enter the Desired Date
0080 INTEGER*4 i_var002      ! Enter Desired Time
0081
0082 LOGICAL formerr          ! Boolean for error in the data
0083
0084
0085 COMMON /Form_Record/ c_var001, c_var002, c_var003, c_var004,
0086 +                   c_var005, c_var006, c_var007, c_var008,
0087 +                   c_var009, c_var010, c_var011, c_var012,
0088 +                   c_var013, c_var014, c_var015, i_var001,
0089 +                   i_var002, c_var016
0090
0091 DATA lastday /31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 /
0092
0093 formerr = .FALSE.
0094
0095 C
0096 C CHECK HOURLY AND 15-MINUTE FLAGS
0097 C
0098
0099 IF(((c_var001 .EQ. 'y') .OR. (c_var001 .EQ. 'Y')) .AND.      ! BOTH CANNOT
0100 + ((c_var002 .EQ. 'y') .OR. (c_var002 .EQ. 'Y'))) THEN      ! EQUAL YES
0101   formerr = .TRUE.
0102   TYPE *, ' ** ERROR, YOU CHOSE BOTH HOURLY AND 15-MINUTE DATA **'
0103 ELSE
0104   IF((c_var001 .EQ. 'y') .OR. (c_var001 .EQ. 'Y')) hr15 = 1 ! HOURLY DATA
0105   IF((c_var002 .EQ. 'y') .OR. (c_var002 .EQ. 'Y')) hr15 = 0 ! 15-MINUTE DATA
0106 ENDIF
0107
0108 C
0109 C CHECK VARIABLE FLAGS
0110 C
0111
0112 DO i = 1,14
0113   temp(i) = 0
0114 ENDDO
0115
0116 IF((c_var003 .EQ. 'y') .OR. (c_var003 .EQ. 'Y')) temp(1) = 1
0117 IF((c_var004 .EQ. 'y') .OR. (c_var004 .EQ. 'Y')) temp(2) = 1
0118 IF((c_var005 .EQ. 'y') .OR. (c_var005 .EQ. 'Y')) temp(3) = 1
0119 IF((c_var006 .EQ. 'y') .OR. (c_var006 .EQ. 'Y')) temp(4) = 1
0120 IF((c_var007 .EQ. 'y') .OR. (c_var007 .EQ. 'Y')) temp(5) = 1
0121 IF((c_var008 .EQ. 'y') .OR. (c_var008 .EQ. 'Y')) temp(6) = 1
0122 IF((c_var009 .EQ. 'y') .OR. (c_var009 .EQ. 'Y')) temp(7) = 1
0123 IF((c_var010 .EQ. 'y') .OR. (c_var010 .EQ. 'Y')) temp(8) = 1
0124 IF((c_var011 .EQ. 'y') .OR. (c_var011 .EQ. 'Y')) temp(9) = 1
0125 IF((c_var012 .EQ. 'y') .OR. (c_var012 .EQ. 'Y')) temp(10) = 1
0126 IF((c_var013 .EQ. 'y') .OR. (c_var013 .EQ. 'Y')) temp(11) = 1
0127 IF((c_var014 .EQ. 'y') .OR. (c_var014 .EQ. 'Y')) temp(12) = 1
0128 IF((c_var015 .EQ. 'y') .OR. (c_var015 .EQ. 'Y')) temp(13) = 1
0129 IF((c_var016 .EQ. 'y') .OR. (c_var016 .EQ. 'Y')) THEN
0130   formerr = .TRUE.

```

```

0131      TYPE*, ' *****'
0132      TYPE*, ' ** POTENTIAL TEMPERATURES CANNOT BE PLOTTED **'
0133      TYPE*, ' ** ON ERDS7A, ONLY ON ERDS6A. **'
0134      TYPE*, ' *****'
0135      temp(14) = 1
0136    ENDIF
0137
0138    which_var = 0
0139
0140    DO i = 1,14
0141      IF(temp(i) .EQ. 1) THEN
0142        IF(which_var .EQ. 0) THEN
0143          which_var = i      ! ASSIGN NUMBER OF VARIABLE OR AREA
0144        ELSE
0145          formerr = .TRUE.    ! MORE THAN ONE FLAG EQUALS YES
0146          TYPE*, ' ** ERROR, YOU CHOSE MORE THAN ONE AREA OR VARIABLE **'
0147        ENDIF
0148      ENDIF
0149    ENDDO
0150
0151    IF(which_var .EQ. 0) THEN      ! NO FLAG EQUALS YES
0152      formerr = .TRUE.
0153      TYPE*, ' ** ERROR, FORGOT TO CHOOSE AN AREA OR VARIABLE **'
0154    ENDIF
0155
0156    C
0157    C CHECK DATE TIME, AND CREATE CHARACTER EQUIVALENCES
0158    C
0159
0160    WRITE(alldate,'(i6)')i var#01      ! CONVERT INTEGER TO CHARACTER.
0161    READ(alldate(1:2),'(i2)')yyymmdd(1) ! SEPERATE DATE INTO THREE
0162    READ(alldate(3:4),'(i2)')yyymmdd(2) ! INTEGER PARTS.
0163    READ(alldate(5:6),'(i2)')yyymmdd(3)
0164    cyyymmdd(1) = alldate(1:2)          ! SEPERATE DATE INTO THREE
0165    cyyymmdd(2) = alldate(3:4)          ! CHARACTER PARTS.
0166    cyyymmdd(3) = alldate(5:6)
0167
0168    WRITE(alltime,'(i4)')i var#02      ! CONVERT INGEGER TO CHARACTER.
0169    READ(alltime(1:2),'(i2)')hrmin(1)  ! SEPERATE TIME INTO TWO
0170    READ(alltime(3:4),'(i2)')hrmin(2)  ! INTEGER PARTS.
0171    chrmin(1) = alltime(1:2)           ! SEPERATE TIME INTO TWO
0172    chrmin(2) = alltime(3:4)           ! CHARACTER PARTS.
0173
0174    IF(IMOD(yyymmdd(1),4) .EQ. 0) lastday(2) = 29      ! LEAP YEAR
0175
0176    C
0177    C CHECK ENTERED DATE WITH CURRENT SYSTEM DATE
0178    C
0179
0180    CALL IDATE(sysmonth,sysday,sysyear)
0181
0182    IF(yyymmdd(1) .GT. sysyear) THEN
0183      formerr = .TRUE.
0184      TYPE*, ' ** ERROR IN ENTERED YEAR, GREATER THAN CURRENT YEAR **'
0185    ENDIF
0186
0187    IF((yyymmdd(1) .EQ. sysyear) .AND. (yyymmdd(2) .GT. sysmonth)) THEN
0188      formerr = .TRUE.
0189      TYPE*, ' ** ERROR IN ENTERED MONTH, GREATER THAN CURRENT MONTH **'
0190    ENDIF
0191
0192    IF((yyymmdd(2) .EQ. sysmonth) .AND. (yyymmdd(3) .GT. sysday)) THEN
0193      formerr = .TRUE.
0194      TYPE*, ' ** ERROR IN ENTERED DAY, GREATER THAN CURRENT DAY **'
0195    ENDIF

```

```

0196
0197
0198 C
0199 C CHECK ENTERED TIME WITH CURRENT SYSTEM TIME, IF ENTERED CURRENT DAY
0200 C
0201
0202 CALL TIME(systime)
0203
0204 syshour = systime(1:2)
0205 sysmin = systime(4:5)
0206 IF(yymmdd(3) .EQ. sysday) THEN
0207     IF(chrmin(1) .EQ. syshour) THEN
0208         IF(chrmin(2) .GT. sysmin) THEN
0209             formerr = .TRUE.
0210             TYPE*, ' ** ERROR IN ENTERED TIME, MINUTES ARE TOO LARGE **'
0211         ENDIF
0212     ELSE
0213         IF(chrmin(1) .GT. syshour) THEN
0214             formerr = .TRUE.
0215             TYPE*, ' ** ERROR IN ENTERED TIME, HOUR IS TOO LARGE **'
0216         ENDIF
0217     ENDIF
0218 ENDIF
0219
0220 C
0221 C CHECK FOR GROSS ERRORS IN ENTERED DATA
0222 C
0223
0224 IF((yymmdd(2) .GT. 12) .OR. (yymmdd(2) .LT. 1)) THEN
0225     formerr = .TRUE.
0226     TYPE*, ' ** ERROR IN ENTERED MONTH **'
0227 ENDIF
0228
0229 IF(formerr) GOTO 888
0230
0231 IF((yymmdd(3) .GT. lastday(yymmdd(2))) .OR. (yymmdd(3) .LT. 0)) THEN
0232     formerr = .TRUE.
0233     TYPE*, ' ** ERROR IN ENTERED DAY **'
0234 ENDIF
0235
0236 888 IF((hrmin(2) .GT. 45) .OR. (hrmin(2) .LT. 0)) THEN
0237     formerr = .TRUE.
0238     TYPE*, ' ** ERROR IN ENTERED MINUTE **'
0239 ENDIF
0240
0241 IF((hr15 .EQ. 1) .AND. (hrmin(2) .NE. 0)) THEN
0242     formerr = .TRUE.
0243     TYPE*, ' ** ERROR IN ENTERED MINUTE FOR HOURLY DATA **'
0244 ENDIF
0245
0246 IF((hr15 .EQ. 0) .AND. (IMOD(hrmin(2),15) .NE. 0)) THEN
0247     formerr = .TRUE.
0248     TYPE*, ' ** ERROR IN ENTERED MINUTE FOR 15-MINUTE DATA **'
0249 ENDIF
0250
0251
0252 RETURN
0253 END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
------	-------	------------

0	SCODE	1670	PIC	CON	REL	LCL	SHR	EXE	RD	NOWRT	LONG
1	SPDATA	784	PIC	CON	REL	LCL	SHR	NOEXE	RD	NOWRT	LONG
2	SLOCAL	388	PIC	CON	REL	LCL	NOSHR	NOEXE	RD	WRT	LONG
3	FORM_RECORD	24	PIC	OVR	REL	GSL	SHR	NOEXE	RD	WRT	LONG

Total Space Allocated	2848
-----------------------	------

ENTRY POINTS

Address	Type	Name	References
0-00000000		ERROR_CHECK	341

VARIABLES

Address	Type	Name	Attributes	References
2-0000003C	CHAR	ALLDATE		66 166 161 162 163 164 165
2-00000038	CHAR	ALLTIME		65 160 169 170 171 172
3-00000000	CHAR	C VAR001	COMM	46 85 99(2) 164(2)
3-00000001	CHAR	C VAR002	COMM	46 85 99(2) 165(2)
3-00000002	CHAR	C VAR003	COMM	47 85 116(2)
3-00000003	CHAR	C VAR004	COMM	48 85 117(2)
3-00000004	CHAR	C VAR005	COMM	49 85 118(2)
3-00000005	CHAR	C VAR006	COMM	50 85 119(2)
3-00000006	CHAR	C VAR007	COMM	51 85 120(2)
3-00000007	CHAR	C VAR008	COMM	52 85 121(2)
3-00000008	CHAR	C VAR009	COMM	53 85 122(2)
3-00000009	CHAR	C VAR010	COMM	54 85 123(2)
3-0000000A	CHAR	C VAR011	COMM	55 85 124(2)
3-0000000B	CHAR	C VAR012	COMM	56 85 125(2)
3-0000000C	CHAR	C VAR013	COMM	57 85 126(2)
3-0000000D	CHAR	C VAR014	COMM	58 85 127(2)
3-0000000E	CHAR	C VAR015	COMM	59 85 128(2)
3-0000000F	CHAR	C VAR016	COMM	60 85 129(2)
AP-0000001C	I*4	FORMERR		34 82 93= 161= 130= 145= 152=
183=				
				188= 193= 209= 214= 225= 229 232= 237=
242=				
247=				
AP-00000018	I*2	HR15		34 72 164= 165= 241 246
**	I*2	I		79 112= 119 146= 141 143
3-0000000F	I*4	I VAR001	COMM	79 85 160
3-00000013	I*4	I VAR002	COMM	80 85 166
2-0000004E	I*2	SYSDAY		78 180A 192 266
2-00000034	CHAR	SYS HOUR		63 264= 267 213
2-00000036	CHAR	SYS MIN		64 265= 268
2-0000004C	I*2	SYS MONTH		77 180A 187 192
2-00000042	CHAR	SYS TIME		67 262A 264 265
2-00000044	I*2	SYS YEAR		76 180A 182 187
AP-00000014	I*2	WHICH_VAR		34 71 138= 142 143= 151

ARRAYS

Address	Type	Name	Attributes	Bytes	Dimensions	References
---------	------	------	------------	-------	------------	------------

AP-000000C0 CHAR CHRMIN 207	4 (2)	34	61	171=	172=
AP-00000100 CHAR CYMDDO 166=	208 213 6 (3)	34	62	164=	165=
AP-00000040 I+2 HRMIN 236(2)	4 (2)	34	70	169=	170=
2-0000001C I+2 LASTDAY	241 246 24 (12)	75	91D	174=	231
2-00000000 I+2 TEMP 118=	28 (14)	74	113=	116=	117=
	119= 120= 121= 122= 123=				
	124= 125= 126= 127= 128=				
	136= 141				
AP-00000000 I+2 YYMDDO 163=	6 (3)	34	60	161=	162=
	174 182 187(2) 192(2) 206				
	224(2) 231(3)				

LABELS

Address	Label	References
0-000005F7	888	229 236#

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	References
	FOR\$DATE	180
	FOR\$TIME_T_DS	202

KEY TO REFERENCE FLAGS	
=	- Value Modified
#	- Defining Reference
A	- Actual Argument, possibly modified
D	- Data Initialization
(n)	- Number of occurrences on line

COMMAND QUALIFIERS

```

FOR/LIS/EXTEND/CROSS/CHECK=ALL ERROR_CHECK

/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)
/DEBUG=(NOSYMBOLS,TRACEBACK)
/STANDARD=(NOSYNTAX,NOSOURCE FORM)
/SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP,NODICTIONARY,SINGLE)
/WARNINGS=(GENERAL,NODECLARATIONS,NOULTRIX)
/CONTINUATIONS=19 /CROSS REFERENCE /NOD LINES /EXTEND SOURCE /F77
/NOQ_FLOATING /I4 /NOMACHINE_CODE /OPTIMIZE /NOANALYSIS

```

COMPILATION STATISTICS

```

Run Time:      9.67 seconds
Elapsed Time:  10.90 seconds
Page Faults:   768
Dynamic Memory: 601 pages

```



```

0001  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002  C
0003  C      OPEN_FILE.FOR
0004  C
0005  C      WRITTEN BY : K.W. BURK
0006  C
0007  C      PURPOSE : TO OPEN THE CORRECT ID. FILE AND READ THE
0008  C                  INFORMATION. FOR A VARIABLE FILE THE DATA IS
0009  C                  THE NUMERIC ID, THE ALPHA-NUMERIC ID, AND ITS
0010  C                  MAP COORDINATES. FOR AN AREA FILE THE DATA IS
0011  C                  ONLY THE NUMERIC, AND ALPHANUMERIC ID.
0012  C                  THE NUMERIC ID'S WILL THEN BE USED TO ACCESS
0013  C                  METEOROLOGICAL DATA IN THE EMS DATABASE.
0014  C
0015  C      ALGORITHM : DECLARE VARIABLES
0016  C                  LOAD FILENAME INTO ARRAY WITH DATA STATEMENT
0017  C                  OPEN FILE
0018  C                  READ NUMBER OF IDS
0019  C                  READ DATA
0020  C                  CLOSE FILE
0021  C                  END SUBROUTINE
0022  C
0023  C      VERSION : 1.0
0024  C
0025  C      DATE : FEBRUARY 1988
0026  C
0027  C      CALLED FROM : MAIN PROGRAM REVIEW.FOR
0028  C
0029  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0030  C
0031  C      SUBROUTINE open_file(file_num, number_id, letter_id, total_ids,
0032  C      +                    xstation, ystation)
0033  C
0034  C      IMPLICIT NONE
0035  C
0036  C
0037  C      VARIABLE DECLARATION
0038  C
0039  C      -----
0040  C      TYPE      | VARIABLE      | COMMENTS
0041  C      -----
0042  C      INTEGER*2  total_ids      | Number of internal ids
0043  C      INTEGER*2  file_num       | Represents the data type or file to be read
0044  C      INTEGER*2  i              | Loop index
0045  C      INTEGER*4  number_id(300) | Numeric id for a variable
0046  C
0047  C      REAL*4      xstation(80)   | The x coordinates for a station
0048  C      REAL*4      ystation(80)   | The y coordinates for a station
0049  C
0050  C      CHARACTER*16 letter_id(300) | Alpha numeric id for a variable
0051  C      CHARACTER*15 file_names(14) | Array of file names
0052  C
0053  C
0054  C      DATA file_names / 'temp_ids.dat ', 'dewpt_ids.dat ',
0055  C      +                  'deltat_ids.dat ', 'precip_ids.dat ', 'talem_ids.dat ',
0056  C      +                  'wnd25m_ids.dat ', 'wnd60m_ids.dat ', 'temp10m_ids.dat ',
0057  C      +                  'temp60m_ids.dat ', '100h_ids.dat ', '300_ids.dat ',
0058  C      +                  '400_ids.dat ', '622r_ids.dat ', 'potemp_ids.dat ' /
0059  C
0060  C
0061  C      OPEN(UNIT=1, FILE='[andrews.review]//file_names(file_num),
0062  C      +      STATUS='OLD', READONLY, SHARED)
0063  C
0064  C      READ(1,100) total_ids
0065  C      100 FORMAT(1x,i3)

```

```

0066
0067      DO i = 1,total_ids
0068
0069          READ(1,101)number_id(i),letter_id(i),xstation(i),ystation(i)
0070      101      FORMAT(1x,i3,6x,a10,1x,f5.2,1x,f5.2)
0071
0072      ENDDO
0073
0074      RETURN
0075      END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	305	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	40	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	344	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	689	

ENTRY POINTS

Address	Type	Name	References
0-00000000		OPEN_FILE	31#

VARIABLES

Address	Type	Name	Attributes	References
AP-000000040	I*2	FILE_NUM		31 43 01
**	I*2	I		44 07= 09(4)
AP-000000100	I*2	TOTAL_IDS		31 42 04= 07

ARRAYS

Address	Type	Name	Attributes	Bytes	Dimensions	References
2-00000000	CHAR	FILE NAMES		210	(14)	51 54D 01
AP-0000000C0	CHAR	LETTER ID		4000	(300)	31 50 09=
AP-000000000	I*4	NUMBER ID		1200	(300)	31 45 09=
AP-000000140	R*4	XSTATION		240	(60)	31 47 09=
AP-000000180	R*4	YSTATION		240	(60)	31 48 09=

LABELS

Address	Label	References
1-00000000	100'	04 05#
1-00000005	101'	09 70#

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	References
------	------	------------

#001

KEY TO REFERENCE FLAGS	
=	- Value Modified
#	- Defining Reference
A	- Actual Argument, possibly modified
D	- Data Initialization
(n)	- Number of occurrences on line

COMMAND QUALIFIERS

FOR/LIS/EXTEND/CROSS/CHECK=ALL OPEN_FILE

/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)

/DEBUG=(NOSYMBOLS,TRACEBACK)

/STANDARD=(NOSYNTAX,NOSOURCE FORM)

/SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP,NODICTIONARY,SINGLE)

/WARNINGS=(GENERAL,NODECLARATIONS,NOLTRIX)

/CONTINUATIONS=19 /CROSS REFERENCE /NOD LINES /EXTEND SOURCE /F77

/NOQ_FLOATING /I4 /NOMACHINE_CODE /OPTIMIZE /NOANALYSIS

COMPILATION STATISTICS

Run Time: 2.83 seconds

Elapsed Time: 2.93 seconds

Page Faults: 587

Dynamic Memory: 339 pages

```

0001  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002  C
0003  C      REFORM_HOUR.FOR
0004  C
0005  C      WRITTEN BY : K. W. BURK
0006  C                  Q. L. ANDREWS
0007  C
0008  C      PURPOSE : TO REFORMAT THE HOUR FROM A
0009  C                  1-24 FORMAT TO A 0-23 FORMAT FOR ACCESSING
0010  C                  THE EMS DATABASE.
0011  C
0012  C      VERSION : 1.0
0013  C
0014  C      DATE : FEBRUARY 1988
0015  C
0016  C      CALLED FROM : MAIN PROGRAM REVIEW.FOR
0017  C
0018  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0019
0020      SUBROUTINE REFORM_HOUR(fdate)
0021
0022      IMPLICIT NONE
0023
0024  C
0025  C      VARIABLE DECLARATIONS
0026  C-----
0027  C      TYPE      | VARIABLE      | COMMENTS
0028  C-----
0029  C
0030
0031      INTEGER*2      fdate(3)      ! Array contains YY,MM,DD
0032      INTEGER*2      i              ! Array and loop index
0033      INTEGER*2      last_day(12)   ! Last day of each month
0034
0035
0036
0037      DATA last_day /31, 28, 31, 30, 31, 30,
0038      *              31, 31, 30, 31, 30, 31 /
0039
0040      IF(IMOD(fdate(1),4) .EQ. 0) THEN      ! CHECK FOR LEAP YEAR
0041          last_day(2) = last_day(2) + 1
0042      ENDIF
0043
0044      IF((fdate(3) .EQ. 31) .AND. (fdate(2) .EQ. 12)) THEN ! END OF YEAR
0045
0046          fdate(3) = 1
0047          fdate(2) = 1
0048          fdate(1) = fdate(1) + 1
0049
0050      ELSE
0051          IF(fdate(3) .EQ. last_day(fdate(2))) THEN      ! END OF MONTH
0052
0053              fdate(3) = 1
0054              fdate(2) = fdate(2) + 1
0055
0056          ELSE
0057
0058              fdate(3) = fdate(3) + 1
0059
0060          ENDIF
0061
0062      ENDIF
0063
0064      RETURN
0065      END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	184	PIC CON REL LCL SHR EXE RD NOWRT LONG
2 \$LOCAL	44	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	148	

ENTRY POINTS

Address	Type	Name	References
0-00000000		REFORM_HOUR	20#

VARIABLES

Address	Type	Name	Attributes	References
**	I*2	I		32

ARRAYS

Address	Type	Name	Attributes	Bytes	Dimensions	References
AP-00000000	I*2	FDATE		8	(3)	2# 31 4# 44(2)
46=				47=	48(2)=	51(2) 53= 54(2)=
2-00000000	I*2	LAST_DAY		58(2)=	24 (12)	33 37D 41(2)= 51

```

+-----+
|          KEY TO REFERENCE FLAGS          |
|  =  - Value Modified                     |
|  #  - Defining Reference                 |
|  A  - Actual Argument, possibly modified |
|  D  - Data Initialization                |
|  (n) - Number of occurrences on line     |
+-----+

```

COMMAND QUALIFIERS

```

FOR/LIS/EXTEND/CROSS/CHECK=ALL REFORM_HOUR

/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)
/DEBUG=(NOSYMBOLS,TRACEBACK)
/STANDARD=(NOSYNTAX,NOSOURCE FORM)
/SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP,NODICTIONARY,SINGLE)
/WARNINGS=(GENERAL,NODECLARATIONS,NOULTRIX)
/CONTINUATIONS=10 /CROSS REFERENCE /MOD LINES /EXTEND SOURCE /F77
/NOG_FLOATING /I4 /NOMACHINE_CODE /OPTIMIZE /NOANALYSIS

```

COMPILATION STATISTICS

```

Run Time:      1.85 seconds
Elapsed Time:  2.58 seconds
Page Faults:   512
Dynamic Memory: 391 pages

```

```

0001 C *****
0002 C
0003 C     PROGRAM - GET_EMS_DATA.FOR
0004 C
0005 C     PROGRAMMER - GREGG ANDREWS 378-6104
0006 C
0007 C     PURPOSE - TO OBTAIN EMS DATA FROM EITHER THE FIFTEEN MINUTE OR HOURLY
0008 C               DATA BASE. THE DESIRED DATA IS SPECIFIED BY THE INTERNAL ID.
0009 C
0010 C     ALGORITHM - DECLARE VARIABLES
0011 C               EQUATE HOUR AND MIN TO STIME ARRAY
0012 C               CALL SUBROUTINES TO GET DESIRED EMS DATA FILENAME
0013 C               INITIALIZE RECORD AND BLOCK COUNTER
0014 C               READ FIRST RECORD OF EMS DATA FILE
0015 C               WHILE PASSED HOUR AND MIN NOT EQUAL TO EMS HOUR AND MIN
0016 C                 CONTINUE SEARCH FOR DESIRED HOUR AND MIN
0017 C               END WHILE LOOP
0018 C               READ IN DATA FROM DESIRED HOUR AND MIN
0019 C               SEARCH THROUGH PASSED PARAMETER IDS AND COMPARED WITH
0020 C                 INTERNAL ID
0021 C               IF IDS MATCH EQUATE MEAN VALUE ARRAY TO DESIRED DATA
0022 C                 ARRAY
0023 C               CONTINUE SEARCH THROUGH TWO BLOCKS
0024 C
0025 C PASSED PARAMETERS - DATE is an array of 3 elements defined as
0026 C                     follows - DATE(1) --> YEAR
0027 C                               DATE(2) --> MONTH
0028 C                               DATE(3) --> DAY
0029 C TIME is an array of 2 elements defined as
0030 C                     follows - TIME(1) --> HOUR
0031 C                               TIME(2) --> MINUTE
0032 C DATA_INTERVAL_FLAG 1 = hourly data
0033 C                     0 = fifteen minute data
0034 C PARAM_ID array containing EMS internal ids of
0035 C                     desired data
0036 C NUMBER_PARAMETERS total number of EMS internal
0037 C                     ids desired
0038 C DESIRED_DATA array containing the desired data
0039 C
0040 C THE SUBROUTINE ASSUMES THE PASSED PARAMETERS ARE CORRECT AND ARE VALID
0041 C VALUES. DO ERROR CHECKING IN THE MAIN PROGRAM.
0042 C
0043 C VERSION - 1.0
0044 C
0045 C DATE - NOVEMBER 1987
0046 C
0047 C *****
0048 C
0049 C SUBROUTINE get_ems_data(data, time, data interval flag,
0050 C                        param_id, number parameters,
0051 C                        desired_data,fnctf)
0052 C
0053 C IMPLICIT NONE
0054 C
0055 C
0056 C..... DATA TYPE          VARIABLE          FUNCTION
0057 C
0058 C
0059 C INTEGER*2                block counter      ! keeps track of number of datablocks; max is 2
0060 C INTEGER*2                data(3)            ! data of desired data
0061 C INTEGER*2                time(2)            ! time of desired data
0062 C INTEGER*4                record             ! specifies the record to read from
0063 C INTEGER*4                param_id(300)      ! internal id stored in external file
0064 C INTEGER*2                year               ! used to compare with year_ems
0065 C INTEGER*2                month              ! value of month from user

```

```

0066      INTEGER*2      day          ! value of day from user
0067      INTEGER*2      hour         ! value of hour from user
0068      INTEGER*2      min          ! value of minute from user if quarterly file required
0069      INTEGER*2      year_ems     ! value of year from ems file
0070      INTEGER*2      month_ems    ! value of month from ems file
0071      INTEGER*2      day_ems     ! value of day from ems file
0072      INTEGER*2      hour_ems    ! value of hour from ems file
0073      INTEGER*2      min_ems     ! value of minute from ems file
0074      INTEGER*2      number_information_records ! number of information records in hourly block of ems file
0075      INTEGER*2      number_parameters ! total number of parameters desired
0076      INTEGER*4      internal_id(300) ! internal id for station parameters
0077      INTEGER*2      ier          ! error value following read statement
0078      INTEGER*2      number_data_records ! number of data records in the hourly block from the ems
file
0079      INTEGER*2      i,j,k        ! indexes for do loops
0080      INTEGER*2      data_interval_flag ! indicates whether user wants hourly or quarterly data
! 1 = hourly quarterly = 0
0081      INTEGER*2      missing_data ! counter used to determine amount of missing data
0082
0083      REAL*4      number_data_points(300) ! number of data points used to calculate the mean
0084      REAL*4      mean_value(300) ! mean value of station parameter
0085      REAL*4      sigma(300) ! standard deviation of a parameter; not used
0086      REAL*4      desired_data(300) ! array that contains the desired data
0087
0088      CHARACTER*16   param_name(300) ! parameter id also includes location of parameter
0089      CHARACTER*(30) filename       ! EMS filename built from date and time from user
0090      LOGICAL        fnotf         ! boolean for file not found
0091
0092
0093
0094      C
0095      C..... PUT TIME ELEMENTS INTO HOUR AND MIN VARIABLES
0096      C
0097
0098      hour = time(1)
0099      min = time(2)
0100
0101
0102      C
0103      C..... OPEN EMS DATA FILE AS SPECIFIED BY THE DATE
0104      C
0105
0106
0107      IF(data_interval_flag .eq. 1)
0108      .   CALL get_hTg_filename(filename,date,time) ! HOURLY FILE
0109      IF(data_interval_flag .eq. 0)
0110      .   CALL get_qTg_filename(filename,date,time) ! FIFTEEN MINUTE FILE
0111
0112
0113      OPEN(UNIT=1, FILE= filename, STATUS='OLD', ACCESS='DIRECT',
0114      .   ORGANIZATION='RELATIVE', READONLY, SHARED, IOSTAT=ier)
0115
0116      IF(ier .ne. 0) THEN
0117      20   WRITE(*,20) ier
0118      .   FORMAT(' Error During Open ',
0119      .   'EMS Data File Code - ',I4)
0120      .   fnotf = .TRUE.
0121      .   ENDIF
0122
0123
0124      C
0125      C..... READ FIRST RECORD OF EMS DATA FILE AND INITIALIZE RECORD AND
0126      C..... BLOCK_COUNTER VARIABLES
0127      C
0128

```

```

0129      record = 1
0130      block_counter = 0
0131
0132 400 READ(1, REC=record, IOSTAT=ierr, ERR=1000) year_ems,
0133      month_ems, day_ems,
0134      hour_ems, min_ems,
0135      number_data_records
0136
0137
0138 C
0139 C..... CONTINUE TO READ HOUR AND MIN FROM EMS FILE UNTIL DESIRED HOUR AND
0140 C..... MIN IS FOUND
0141 C
0142
0143
0144      DO WHILE ( (hour_ems .ne. hour) .or.
0145      (min_ems .ne. min) )
0146
0147      number_information_records = 1 ONE INFORMATION RECORD PER 31 DATA RECORD
0148      jint(number_data_records / 31. + .49999)
0149      record = record + 1 + number_information_records + ! FOLLOWING CODE DETERMINES THE NUMBER
! RECORDS TO SKIP
0150
0151      number_data_records
0152      READ(1, REC=record, IOSTAT=ierr, ERR=1000) year_ems,
0153      month_ems, ! READ NEXT DATE AND HOUR
0154      day_ems, hour_ems, ! AND NUMBER OF RECORDS
0155      min_ems, ! TO DETERMINE IF DESIRED
0156      number_data_records ! DATE FROM SCRATCH FILE
0157
0158      ENDDO
0159
0160 C
0161 C..... SHOULD BE AT DESIRED HOUR AND MIN DATA BLOCK NOW MUST DETERMINE
0162 C..... NUMBER OF INFORMATION RECORDS TO SKIP
0163 C
0164      number_information_records = jint(number_data_records / 31.
0165      + .49999)
0166
0167
0168 C
0169 C..... FOLLOWING CODE SHOULD PUT AT BEGINNING OF DATA RECORD BLOCK OF DESIRED
0170 C..... HOUR AND MIN
0171 C
0172
0173      record = record + number_information_records
0174
0175
0176 C
0177 C..... NOW READ ALL DATA FOR THE DESIRED HOUR AND MIN
0178 C
0179
0180      DO i=1, number_data_records
0181      record = record + 1
0182      READ(1, REC=record, IOSTAT=ierr, ERR=1000)
0183      internal_id(i),
0184      param_name(i),
0185      mean_value(i),
0186      sigma(i),
0187      number_data_points(i)
0188
0189 C..... CHECK FOR MISSING DATA AND REPLACE WITH 999.9
0190

```



```

0191      IF((mean_value(i) .EQ. 0.0) .AND.
0192      (sigma(i) .EQ. 0.0) .AND.
0193      (number_data_points(i) .EQ. 0.0)) mean_value(i) = 999.9
0194
0195      ENDDO
0196
0197
0198
0199      C
0200      C..... SEARCH THROUGH ARRAY OF INTERNAL IDS AND CHECK TO SEE IF IT IS A DESIRED
0201      C..... DATA, IF SO STORE IN DESIRED_DATA ARRAY
0202      C
0203
0204      k = 0
0205
0206      88      DO j=1, number_parameters
0207      k = k + 1
0208      DO i=1, number_data_records
0209      IF(para_id(j) .eq. internal_id(i)) THEN
0210      desired_data(k) = mean_value(i)
0211      ENDIF
0212      ENDDO
0213      ENDDO
0214
0215      record = record + 1
0216      block_counter = block_counter + 1
0217
0218      IF(block_counter .EQ. 2) GOTO 1000
0219
0220      GOTO 400      ! loop will terminate when RECORD is greater than the number of records in
                   ! the file
0221                   ! this is necessary since the number of same time stamps can be greater than
                   ! one
0222
0223      1000      CONTINUE
0224      CLOSE(1)
0225
0226      RETURN
0227      END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	870	PIC COM REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	48	PIC COM REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	9820	PIC COM REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	10754	

ENTRY POINTS

Address	Type	Name	References
0-00000000		GEY_EMS_DATA	490

VARIABLES

Address	Type	Name	Attributes	References
**	I*2	BLOCK_COUNTER	50	130= 210(2)= 218
AP-00000000C0	I*2	DATA_INTERVAL_FLAG	40	80 107 109

**	I*2	DAY	66				
2-000025AE	I*2	DAY_EMS	71	132=	151=		
2-000025B0	CHAR	FILENAME	89	187A	189A	113A	
AP-0000001C0	L*4	FNOTF	49	90	120=		
2-000025A8	I*2	HOUR	67	98=	144		
2-000025B8	I*2	HOUR_EMS	72	132=	144	151=	
**	I*2	I	79	186=	182(5)	191(4)	208=
209							
2-000025B4	I*2	IER	210				
151=			77	113=A	116	117	132=
			182=				
**	I*2	J	79	206=	209		
**	I*2	K	79	204=	207(2)=	210	
2-000025A8	I*2	MIN	68	99=	144		
2-000025B2	I*2	MIN_EMS	73	132=	144	151=	
**	I*2	MISSING_DATA	81				
**	I*2	MONTH	85				
2-000025AC	I*2	MONTH_EMS	70	132=	151=		
2-000025B6	I*2	NUMBER_DATA_RECORDS	76	132=	147	149	151=
164							
			180	208			
**	I*2	NUMBER_INFORMATION_RECORDS	74	147=	149	164=	173
AP-000000140	I*2	NUMBER_PARAMETERS	49	75	206		
**	I*4	RECORD	82	129=	132	149(2)=	151
173(2)=							
			181(2)=	182	215(2)=		
**	I*2	YEAR	84				
2-000025AA	I*2	YEAR_EMS	89	132=	151=		

ARRAYS

Address	Type	Name	Attributes	Bytes	Dimensions	References
AP-000000040	I*2	DATE		6	(3)	49 60
107A						
			109A			
AP-000000180	R*4	DESIRED_DATA		1200	(300)	49 88
210=						
2-00000000	I*4	INTERNAL_ID		1200	(300)	76 182=
209						
2-00000000	R*4	MEAN_VALUE		1200	(300)	84 182=
191(2)=						
			210			
2-000004B0	R*4	NUMBER_DATA_POINTS		1200	(300)	83 182=
191						
AP-000000100	I*4	PARAM_ID		1200	(300)	49 63
209						
2-000012C0	CHAR	PARAM_NAME		4800	(300)	88 182=
2-00000E10	R*4	SIGMA		1200	(300)	85 182=
191						
AP-000000000	I*2	TIME		4	(2)	49 61
98						
			99	107A	109A	

LABELS

Address	Label	References				
1-00000000	20'	117	118#			
**	06	206#				
0-000000B6	400	132#	220			
0-00000364	1000	132	161	182	218	223#

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	References
	FOR\$CLOSE	224
	FOR\$OPEN	113
	GET_HLG_FILENAME	107
	GET_QLG_FILENAME	109

KEY TO REFERENCE FLAGS	
=	- Value Modified
#	- Defining Reference
A	- Actual Argument, possibly modified
D	- Data Initialization
(n)	- Number of occurrences on line

COMMAND QUALIFIERS

```

FOR/LIS/EXTEND/CROSS/CHECK=ALL GET_BMS_DATA

/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)
/DEBUG=(NOSYMBOLS,TRACEBACK)
/STANDARD=(NOSYNTAX,NOSOURCE FORM)
/SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP,NODICTIONARY,SINGLE)
/WARNINGS=(GENERAL,NODECLARATIONS,NOULTRIX)
/CONTINUATIONS=10 /CROSS REFERENCE /MOD LINES /EXTEND SOURCE /F77
/NOG_FLOATING /I4 /NOMACHINE_CODE /OPTIMIZE /NOANALYSIS

```

COMPILATION STATISTICS

```

Run Time:      5.22 seconds
Elapsed Time:  8.05 seconds
Page Faults:   603
Dynamic Memory: 403 pages

```



```

0066      REAL*4      ytower(8)      ! Y coordinates of wind vectors
0067      REAL*4      towards(8)     ! Wind directions and speeds
0068      REAL*4      xnorth          ! X position of north indicator
0069      REAL*4      ynorth          ! Y position of north indicator
0070      REAL*4      scale           ! Size of north indicator 1 = 1 inch.
0071
0072      CHARACTER*30 headers(14)    ! Headings for plots
0073      CHARACTER*10 alpha(200)     ! Alphanumeric ID's for EMS variables
0074      CHARACTER*2  date(3)        ! YY,MM,DD
0075      CHARACTER*2  time(2)        ! HH,MM
0076
0077
0078
0079      C
0080      C      INITIALIZE PLOT SIZE, SET SIZE
0081      C
0082
0083      CALL plots (0.0, 0.0, 1)
0084      CALL get_plot_par(params)
0085      CALL NEWPAQ
0086      CALL PLTOUT
0087      ISS = LIB$ERASE PAGE(1,1)
0088      CALL FACTOR (.9)
0089
0090
0091      C
0092      C      PLOT GRAPH OUTLINE ON SCREEN
0093      C
0094
0095      CALL NEWPEN(4)
0096      CALL plot (0.0, 0.0, -3)
0097      CALL plot (2.5, 4.5, 3)
0098      CALL plot (2.5, 8.5, 2)
0099      CALL plot (7.0, 8.5, 2)
0100      CALL plot (7.0, 4.5, 2)
0101      CALL plot (2.5, 4.5, 2)
0102
0103      C
0104      C      PLOT THE AXIS LABELS FOR THE TEMPERATURE GRAPH
0105      C
0106
0107      CALL newpen(2)
0108      CALL symbol (1.375,8.1,0.20,'T',0.0,1)
0109      CALL symbol (1.375,7.0,0.20,'0',0.0,1)
0110      IF(area_type .EQ. 13) THEN
0111          CALL symbol (1.375,7.5,0.20,'W 200ft-',0.0,0)
0112      ELSE
0113          CALL symbol (1.375,7.5,0.20,'W 60m -',0.0,0)
0114      ENDIF
0115      CALL symbol (1.375,7.2,0.20,'E',0.0,1)
0116      CALL symbol (1.375,6.0,0.20,'R',0.0,1)
0117      IF(area_type .EQ. 13) THEN
0118          CALL symbol (1.375,6.5,0.20,' 30ft-',0.0,0)
0119      ELSE
0120          CALL symbol (1.375,6.5,0.20,' 10m -',0.0,0)
0121      ENDIF
0122      CALL symbol (1.375,6.4,0.20,'H',0.0,1)
0123      CALL symbol (1.375,6.1,0.20,'E',0.0,1)
0124      CALL symbol (1.375,5.0,0.20,'I',0.0,1)
0125      IF(area_type .EQ. 13) THEN
0126          CALL symbol (1.375,5.5,0.20,'G 3ft-',0.0,0)
0127      ELSE
0128          CALL symbol (1.375,5.5,0.20,'Q 2m -',0.0,0)
0129      ENDIF
0130      CALL symbol (1.375,5.2,0.20,'H',0.0,1)

```

```

0131      CALL symbol (1.375,4.9,0.20,'T',0.0,1)
0132      CALL symbol (3.00,3.0,0.20,'TEMPERATURE DEGREES ( F )',0.0,25)
0133
0134      C
0135      C      PLOT SMALL NORTH INDICATOR NEAR WIND VECTORS
0136      C
0137
0138      CALL newpen(3)
0139      OPEN(UNIT=30,FILE='[andrews.review]NORTH.DAT',STATUS='OLD',READONLY,SHARED)      ! THE COORDINATES
0140      READ(30,1001)xnorth,ynorth,scale      ! AND SCALE FOR
0141      1001      FORMAT(1x,f0.3,1x,f0.3,1x,f4.2)      ! NORTH INDICATOR
0142      CLOSE(30)      ! ARE IN A DATA
0143      CALL plot_north(xnorth,ynorth,scale)      ! FILE
0144
0145      C
0146      C      PLOT LABELS FOR WIND DIRECTION VECTORS
0147      C
0148
0149      CALL symbol (2.3,0.0,0.21,'WIND DIRECTION VECTORS',0.0,22)
0150      IF (area_type .EQ. 13) THEN
0151          CALL symbol (1.7,1.2,0.25,'30ft',0.0,4)
0152          CALL symbol (3.7,1.2,0.25,'100ft',0.0,5)
0153          CALL symbol (5.7,1.2,0.25,'200ft',0.0,5)
0154      ELSE
0155          CALL symbol (1.0,1.2,0.25,'10m',0.0,3)
0156          CALL symbol (3.0,1.2,0.25,'25m',0.0,3)
0157          CALL symbol (5.0,1.2,0.25,'60m',0.0,3)
0158      ENDIF
0159
0160      C
0161      C      PLOT HEADERS AND DATE/TIME
0162      C
0163
0164      INCLUDE 'HEADERS.INC'
0165      CALL symbol (.20, 9.75, 0.35, %REF(headers(area_type)),
0166                  0.0, 30)
0167      CALL symbol (1.5,9.2,0.21,'DATE: - -',0.0,14)
0168      CALL symbol (2.50,9.2,0.21,%REF(date(1)),0.0,2)
0169      CALL symbol (3.19,9.2,0.21,%REF(date(2)),0.0,2)
0170      CALL symbol (3.815,9.2,0.21,%REF(date(3)),0.0,2)
0171      CALL symbol (5.0,9.2,0.21,'TIME: : ',0.0,10)
0172      CALL symbol (6.0,9.2,0.21,%REF(time(1)),0.0,2)
0173      CALL symbol (6.625,9.2,0.21,%REF(time(2)),0.0,2)
0174      CALL newpen(7)
0175      IF (period .EQ. 1) THEN
0176          CALL symbol (9.5,10.4,0.31,'HOURLY DATA',0.0,11)
0177      ELSE
0178          CALL symbol (9.5,10.4,0.31,'15-MINUTE DATA',0.0,14)
0179      ENDIF
0180      CALL symbol (8.9,10.0,0.22,'      Mean',0.0,29)
0181      CALL symbol (8.9,9.7,0.22,'Numeric Alphanumeric Parameter',0.0,32)
0182      CALL symbol (8.9,9.4,0.22,' ID. ID. Value',0.0,30)
0183      CALL symbol (8.9,9.3,0.22,' _____',0.0,32)
0184
0185      ypos = 0.0
0186
0187      C
0188      C      PLOT THE RAW DATA(MEAN VALUES) IN COLUMNS ON RIGHT SIDE OF SCREEN
0189      C
0190
0191      DO i = 1, nparam
0192          fnum = FLOATJ(num_id(i))
0193          CALL number (9.3,ypos,0.20,fnum,0.0,-1)
0194          CALL symbol (10.0,ypos,0.20,%REF(alpha(i)),0.0,10)

```

```

0222 IF ((i .LE. 3) .OR. (i .GE. 10)) THEN
0223   IF (i .EQ. 11) THEN
0224     CALL number (13.0,ypos,0.20,value(i),0.0,2)
0225   ELSE
0226     CALL number (13.0,ypos,0.20,value(i),0.0,1)
0227   ENDIF
0228 ELSE
0229   IF ((i .EQ. 4) .OR. (i .EQ. 6) .OR. (i .EQ. 8)) THEN
0230     IF (value(i) .GT. 999.0) THEN ! DO NOT ROUND
0231       CALL number (13.0,ypos,0.20,value(i),0.0,1) ! MISSING DATA
0232     ELSE
0233       wdir = (IIFIX((value(i) / 10.0) + 0.5)) * 10 ! ROUND WIND
0234       fnum = FLOATI(wdir) ! DIRECTION TO
0235       CALL number (13.0,ypos,0.20,fnum,0.0,-1) ! NEAREST TEN DEGREES
0236     ENDIF
0237   ELSE
0238     IF (value(i) .GT. 999.0) THEN ! DO NOT ROUND
0239       CALL number (13.0,ypos,0.20,value(i),0.0,1) ! MISSING DATA
0240     ELSE
0241       CALL number (13.0,ypos,0.20,value(i),0.0,-1) ! WIND SPEED
0242     ENDIF
0243   ENDIF
0244 ENDIF
0245
0246 ypos = ypos - 0.3
0247
0248 ENDDO
0249
0250 C
0251 C PLOT THE WIND VECTORS FOR THE 10m, 25m, 60m HEIGHTS
0252 C
0253
0254 CALL newpen(3)
0255 DATA xtower / 4.0, 0.0, 8.0, 0.0, 12.0, 0.0 / ! POSITIONS FOR THE
0256 DATA ytower / 5.0, 0.0, 5.0, 0.0, 5.0, 0.0 / ! WIND VECTORS
0257 i = 1
0258 DO j = 4,9
0259   towards(i) = value(j) ! WIND DIRECTIONS AND SPEEDS
0260   i = i + 1
0261 ENDDO
0262 numvect = 0
0263
0264 CALL plot_vector(numvect,towards,xtower,ytower)
0265
0266 C
0267 C PLOT THE SPEEDS UNDER THE WIND VECTORS,
0268 C IF MISSING DATA PLOT 999.9 AND NO MPH LABEL
0269 C
0270
0271 IF (value(5) .GT. 999.0) THEN
0272   CALL number (1.75,1.5,0.20,value(5),0.0,1)
0273 ELSE
0274   CALL symbol (2.1,1.5,0.20,'mph',0.0,3)
0275   CALL number (1.75,1.5,0.20,value(5),0.0,-1)
0276 ENDIF
0277
0278 IF (value(7) .GT. 999.0) THEN
0279   CALL number (3.75,1.5,0.20,value(7),0.0,1)
0280 ELSE
0281   CALL symbol (4.1,1.5,0.20,'mph',0.0,3)
0282   CALL number (3.75,1.5,0.20,value(7),0.0,-1)
0283 ENDIF
0284
0285 IF (value(9) .GT. 999.0) THEN
0286   CALL number (5.75,1.5,0.20,value(9),0.0,1)

```

```

0287      ELSE
0288          CALL symbol (6.1,1.5,0.20,'mph',0.0,3)
0289          CALL number (5.75,1.5,0.20,value(9),0.0,-1)
0290      ENDIF
0291
0292      C
0293      C      PLOT THE VERTICAL LINE GRAPH FOR TEMPERATURES, AND THE
0294      C      X-AXIS CALIBRATIONS AND LABELS.
0295      C
0296
0297      CALL newpen(2)
0298      CALL plot_temps(value(1),value(2),value(3))
0299
0300      C
0301      C      CLOSE PLOTFILE AND DISPLAY
0302      C
0303
0304      CALL Pltend
0305      CALL plots (0.0, 0.0, 1)
0306      CALL NEWPAQ
0307      CALL PLTOUT
0308      ISS = LIBERASE_PAGE(1,1)
0309
0310      RETURN
0311      END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
# \$CODE	1400	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	600	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	2860	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	4934	

ENTRY POINTS

Address	Type	Name	References
0-00000000		PLOT_AREA	363

VARIABLES

Address	Type	Name	Attributes	References
AP-00000040 150	I*2	AREA_TYPE		36 48 110 117 125
2-000002EC 223	R*4 I*2	FNLM I	191	64 210= 220A 234= 235A 53 217= 219 221 222(2)
**	I*4	ISS	224 226 229(3) 230 231 233	49 87= 308=
**	I*2	J	230 239 241 257= 259 280(2)=	53 250= 259
AP-00000140	I*2	NPARAM		36 52 217
2-000002E4	I*2	NUMVECT		50 282= 284A
AP-00000200	I*2	PERIOD		36 55 201
**	R*4	REALX		62
**	R*4	REALY		63

2-000002F8	R*4	SCALE	70	140=	143A		
**	I*2	WDIR	54	233=	234		
2-000002F0	R*4	XNORTH	68	140=	143A		
2-000002F4	R*4	YNORTH	69	140=	143A		
2-000002E8	R*4	YPOS	61	211=	220A	221A	224A
226A							
			231A	235A	239A	241A	246(2)=

ARRAYS

Address	Type	Name	Attributes	Bytes	Dimensions	References		
AP-000000C0	CHAR	ALPHA		3200	(200)	36	73	221A
AP-00000010	CHAR	DATE		6	(3)	36	74	194A 195A
196A								
2-00000080	R*4	DIRECTION		120	(30)	59		
2-000000140	CHAR	HEADERS		420	(14)	72	176D	191A
AP-000000100	I*4	NUM_ID		1200	(300)	36	51	219
2-00000000	R*4	PARAMS		128	(32)	58	84A	
AP-0000001C0	CHAR	TIME		4	(2)	36	75	198A 199A
2-00000120	R*4	TOWERS		24	(6)	67	259=	264A
AP-000000000	R*4	VALUE		1200	(300)	36	60	224A 226A
230								
				231A	233	238	239A	241A
				250	271	272A	275A	278
				279A	282A	285	286A	289A
				298(3)A				
2-000000F8	R*4	XTOWER		24	(6)	65	255D	264A
2-00000110	R*4	YTOWER		24	(6)	66	256D	264A

LABELS

Address	Label	References
1-0000028A	1001'	140 141

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	References
	FACTOR	88
	FORCLOSE	142
	FOROPEN	130
	GET PLOT PAR	84
I*4	LIBERASE_PAGE	50 87 308
	MEMPAQ	85 306
	MEMPEN	95 107 138 200 254
297		
	NUMBER	220 224 226 231 235
239		
		241 272 275 279 282 288
		289
	PLOT	96 97 98 99 100
101		
	PLOTS	83 305
	PLOT_NORTH	149

	PLOT TEMPS			298			
	PLOT VECTOR			264			
	PLTEND			304			
	PLTOUT			86	307		
116	SYMBOL			100	109	111	113 115
		118	120	122	123	124	126
		128	130	131	132	149	151
		152	153	155	156	157	191
		193	194	195	196	197	198
		199	202	204	206	207	208
		209	221	274	281	288	

0001

```

+-----+
| KEY TO REFERENCE FLAGS |
| = - Value Modified    |
| # - Defining Reference |
| A - Actual Argument, possibly modified |
| D - Data Initialization |
| (n) - Number of occurrences on line |
+-----+

```

COMMAND QUALIFIERS

```

FOR/LIS/EXTEND/CROSS/CHECK=ALL PLOT_AREA

/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)
/DEBUG=(NOSYMBOLS,TRACEBACK)
/STANDARD=(NOSYNTAX,NOSOURCE FORM)
/SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP,NOdictionary,SINGLE)
/WARNINGS=(GENERAL,NODECLARATIONS,NOULTRIX)
/CONTINUATIONS=19 /CROSS REFERENCE /NOD LINES /EXTEND SOURCE /F77
/NOG_FLOATING /I4 /NOMACHINE_CODE /OPTIMIZE /NOANALYSIS

```

COMPILATION STATISTICS

```

Run Time:      9.32 seconds
Elapsed Time:  10.31 seconds
Page Faults:   851
Dynamic Memory: 468 pages

```

```

0001  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002  C
0003  C   PLOT_VAR.FOR
0004  C
0005  C   WRITTEN BY:  K.W. BURK
0006  C               G.L. ANDREWS
0007  C
0008  C   PURPOSE:  TO PLOT THE REQUESTED METEOROLOGICAL VARIABLES ON A
0009  C             BACKGROUND MAP OF THE HANFORD RESERVATION.
0010  C             WILL ALSO PLOT THE RAW DATA IN COLUMNS ON THE LEFT
0011  C             SIDE OF THE SCREEN.
0012  C
0013  C   ALGORITHM:  DECLARE VARIABLES
0014  C               PLOT BOUNDARY
0015  C               PLOT MAP
0016  C               PLOT HEADERS, DATE AND TIME
0017  C               GET STABILITY
0018  C               PLOT COLUMNS OF DATA
0019  C               PLOT WIND VECTORS IF NEEDED
0020  C               END ROUTINE
0021  C
0022  C   VERSION :  1.0
0023  C
0024  C   DATE :  FEBRUARY 1988
0025  C
0026  C   CALLED FROM :  MAIN PROGRAM  REVIEW.FOR
0027  C
0028  C   CALLS TO :  PLT GRAPHICS.FOR
0029  C               GET STABILITY.FOR
0030  C               PLOT VECTOR.FOR
0031  C               GET_THETA.FOR
0032  C
0033  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0034  C
0035  C   SUBROUTINE PLOT_VAR(data_read,var_type,value,alpha,num_id,nparam,
0036  C   +             xcord,ycord,date,time,period)
0037  C
0038  C   IMPLICIT NONE
0039  C
0040  C
0041  C   VARIABLE DECLARATIONS
0042  C   -----
0043  C   TYPE      |  VARIABLE      |  COMMENTS
0044  C   -----
0045  C
0046  C
0047  C   INTEGER*2  stab_class(4)  ! Integer array for stability class
0048  C   INTEGER*2  wdir           ! Temporary variable for wind direction
0049  C   INTEGER*2  wspd           ! Temporary variable for wind speed
0050  C   INTEGER*2  period         ! Hourly = 1, 15-minute = 8
0051  C   INTEGER*2  i,j,k         ! Loop indices
0052  C   INTEGER*2  nparam         ! Number of parameters in ID files
0053  C   INTEGER*2  var_type       ! Which variable to plot
0054  C   INTEGER*4  iss            ! Special variable for TRPLOT
0055  C   INTEGER*4  lib$erase page ! Special variable for TRPLOT
0056  C   INTEGER*4  num_id(300)    ! Numeric ID used to access EMS database
0057  C
0058  C   REAL*4     params(32)     ! Used to initialize plotfile
0059  C   REAL*4     xcord(80)      ! X coordinates for stations
0060  C   REAL*4     ycord(80)      ! Y coordinates for stations
0061  C   REAL*4     ypos           ! Position on Y-axis
0062  C   REAL*4     fnum           ! Temporary real variable
0063  C   REAL*4     realx           ! Real x coordinates
0064  C   REAL*4     realy           ! Real y coordinates
0065  C   REAL*4     value(300)     ! Array of raw data from EMS database

```

```

0066      REAL*4      potential(80)      ! Array of potential temperatures.
0067      REAL*4      print_size          ! Size of print in columns of data
0068      REAL*4      step_size           ! Size of step(vertical), for each row of data
0069
0070      CHARACTER*30  headers(14)       ! Headings for plots
0071      CHARACTER*16  alpha(200)        ! Alphanumeric ID's for EMS variables
0072      CHARACTER*2   date(3)           ! Desired date; YY,MM,DD
0073      CHARACTER*2   time(2)           ! Desired time; HH,MM
0074      CHARACTER*1   stab_classac(4)   ! Character array for stability class
0075
0076      LOGICAL       data_read          ! True if map data has been read in, saves time
0077
0078
0079      C
0080      C      INITIALIZE PLOTFILE, SET SIZE, DRAW BORDER AND HEADERS
0081      C
0082
0083      CALL PLOTS (0.0, 0.0, 1)
0084      CALL GET_PLOT_PAR(PARAMS)
0085      CALL NEWPAG
0086      CALL PLTOUT
0087      ISS = LIB$ERASE PAGE(1,1)
0088      CALL FACTOR (.9)
0089
0090      C
0091      C      PLOT BOUNDARY
0092      C
0093
0094      CALL NEWPEN(4)
0095      CALL PLOT (0.0, 0.0, -3)
0096      CALL PLOT (0.0, 0.0, 2)
0097      CALL PLOT (0.5, 0.0, 2)
0098      CALL PLOT (0.5, 0.5, 2)
0099      CALL PLOT (0.0, 0.5, 2)
0100      CALL PLOT (0.0, 0.0, 2)
0101
0102      C
0103      C      PLOT ROADS, RIVER, AND HAMFORD BOUNDARY
0104      C
0105
0106      CALL PLT_GRAPHICS(data_read)
0107
0108      C
0109      C      PLOT HEADERS AND DATE/TIME
0110      C
0111
0112      INCLUDE 'HEADERS.INC'           ! FILE WITH HEADINGS FOR PLOTS
0113      CALL NEWPEN(3)
0114      CALL SYMBOL (.20, 9.75, 0.35, %REF(headers(var_type)),
0115                  0.0, 30)
0116      CALL symbol (1.5,9.2,0.21,'DATE: - -',0.0,14)
0117      CALL symbol (2.50,9.2,0.21,%REF(date(1)),0.0,2)
0118      CALL symbol (3.10,9.2,0.21,%REF(date(2)),0.0,2)
0119      CALL symbol (3.815,9.2,0.21,%REF(date(3)),0.0,2)
0120      CALL symbol (5.0,9.2,0.21,'TIME: :',0.0,10)
0121      CALL symbol (6.0,9.2,0.21,%REF(time(1)),0.0,2)
0122      CALL symbol (8.525,9.2,0.21,%REF(time(2)),0.0,2)
0123
0124      CALL newpen(7)
0125      IF ( period .EQ. 1 ) THEN
0126          CALL symbol (9.5,10.4,0.31,'HOURLY DATA',0.0,11)
0127      ELSE
0128          CALL symbol (9.5,10.4,0.31,'15-MINUTE DATA',0.0,14)
0129      ENDIF
0130      CALL symbol (8.9,10.0,0.22,'
Mean',0.0,29)

```

```

0157 CALL symbol (8.9,9.7,0.22, 'Numeric Alphanumeric Parameter',0.0,32)
0158 CALL symbol (8.9,9.4,0.22, ' ID. ID. Value',0.0,30)
0159 CALL symbol (8.9,9.3,0.22, ' ID. ID. Value',0.0,32)
0160
0161 C
0162 C MAKE ANY SPECIAL ADJUSTMENTS TO DATA
0163 C
0164
0165 IF (var_type .EQ. 3) THEN ! CALCULATE THE DELTA TEMP.
0166   value(1) = value(2) - value(1) ! TO CALCULATE THE 022R DELTA
0167   alpha(1) = 'DT-022R(CALC)' ! TEMP, USE THE 200ft TEMP MINUS
0168   num_id(1) = 999 ! THE 30ft TEMP, THIS WILL THEN
0169   nparam = nparam - 1 ! DECREASE THE NUMBER OF PARAMETERS
0170   DO j = 2,4 ! BY ONE. THEN SHIFT THE OTHER VALUES
0171     value(j) = value(j+1) ! UP ONE IN THE ARRAY.
0172     alpha(j) = alpha(j+1)
0173     num_id(j) = num_id(j+1)
0174     xcord(j) = xcord(j+1)
0175     ycord(j) = ycord(j+1)
0176   ENDDO
0177
0178 CALL GET_STABILITY(value,stab_classi,stab_classac)
0179
0180 ENDF
0181
0182 IF (var_type .EQ. 14) THEN ! GO GET THE
0183   ! POTENTIAL
0184   CALL get_theta(nparam,value,data,time,potential) ! TEMPERATURE
0185
0186 ENDF
0187
0188 C
0189 C PLOT THE DESIRED DATA IN COLUMNS AND ON THE MAP
0190 C
0191
0192 ypos = 8.8
0193
0194 IF (var_type .EQ. 14) THEN ! IF POTENTIAL TEMPERATURES IS CHOSEN
0195   step_size = 0.25 ! THEN THE SIZE OF PRINT AND THE
0196   print_size = 0.17 ! SIZE OF ROW-STEP MUST BE
0197 ELSE ! SMALLER BECAUSE THERE IS MORE
0198   step_size = 0.3 ! INFORMATION TO DISPLAY
0199   print_size = 0.2
0200 ENDF
0201
0202 DO i = 1,nparam
0203
0204   realx = xcord(i) * .5
0205   realy = ycord(i) * .5
0206   IF (var_type .EQ. 5) THEN ! THERE IS A SPECIAL WAY
0207     IF (IMOD(i,2) .EQ. 1) THEN ! TO PLOT THE TELEMETRY
0208       fnum = FLOATJ(num_id(i)) ! COLUMNS OF DATA
0209       CALL number (9.0,ypos,0.20,fnum,0.0,-1)
0210       CALL symbol (9.4,ypos,0.20,' ',0.0,1)
0211       fnum = FLOATJ(num_id(i+1))
0212       CALL number (9.5,ypos,0.20,fnum,0.0,-1)
0213       CALL symbol (11.1,ypos,0.20,$REF(alpha(i)),0.0,5)
0214     IF (value(i) .EQ. 999.9) THEN
0215       wdir = IIFIX(value(i))
0216       fnum = FLOATI(IIFIX(value(i+1))) ! TRUNCATE THEN FLOAT
0217       CALL number (13.7,ypos,0.20,fnum,0.0,-1)
0218     ELSE
0219       wdir = (IIFIX ((value(i) / 10.0) + 0.5)) * 10 ! ROUND TO NEAREST TEN DEGREES
0220       CALL number (13.7,ypos,0.20,value(i+1),0.0,-1)
0221     ENDF

```

```

0222      fnum = FLOATI (wdir)
0223      CALL number (12.9,ypos,8.28,fnum,8.8,-1)
0224      CALL symbol (14.1,ypos,8.18,'mph',8.8,3)
0225  ELSE
0226      ypos = ypos + 8.3
0227  ENDIF
0228
0229  ELSE
0230      fnum = FLOATJ(num_id(i))
0231      CALL number (9.8,ypos,print_size,fnum,8.8,-1)          ! PLOT NUMERIC ID
0232      CALL symbol (18.6,ypos,print_size,$REF(alpha(i)),8.8,18) ! PLOT ALPHA-NUMERIC ID
0233  ENDIF
0234  IF ( (var_type .EQ. 1) .OR. (var_type .EQ. 2)
0235  *  .OR. (var_type .EQ. 8) .OR. (var_type .EQ. 9) ) THEN
0236      CALL number (13.8,ypos,8.28,value(i),8.8,1)          ! ONLY ONE DECIMAL
0237      CALL number (realx,realy,8.14,value(i),8.8,1)        ! FOR TEMPERATURES
0238  ENDIF
0239  IF (var_type .EQ. 3) THEN
0240      CALL number (13.8,ypos,8.28,value(i),8.8,2)          ! DELTA TEMP.
0241      CALL symbol (14.8,ypos,8.28,$REF(stab_classac(i)),8.8,1) ! PLOT THE
0242      fnum = FLOATI(stab_classac(i))                        ! STABILITY
0243      CALL number (14.375,ypos,8.28,fnum,8.8,-1)           ! CLASS
0244      CALL number (realx,realy,8.14,value(i),8.8,2)
0245  ENDIF
0246  IF (var_type .EQ. 4) THEN                                ! PRECIPITATION
0247      IF (value(i) .GT. 999.8) THEN
0248          CALL number (13.8,ypos,8.28,value(i),8.8,1)      ! CHECK FOR MISSING
0249          CALL number (realx,realy,8.14,value(i),8.8,1)    ! DATA
0250      ELSE
0251          IF (period .EQ. 1) THEN
0252              fnum = value(i) / 100.8                        ! CALCULATE HUNDRETHS
0253              CALL number (13.8,ypos,8.28,fnum,8.8,2)        ! OF INCHES FOR
0254              CALL number (realx,realy,8.14,fnum,8.8,2)      ! HOURLY
0255          ELSE
0256              fnum = value(i) / 480.8                        ! CALCULATE HUNDRETHS
0257              CALL number (13.8,ypos,8.28,fnum,8.8,2)        ! OF INCHES FOR
0258              CALL number (realx,realy,8.14,fnum,8.8,2)      ! 15-MINUTE
0259          ENDIF
0260      ENDIF
0261  ENDIF
0262  IF ((var_type .EQ. 6) .OR. (var_type .EQ. 7)) THEN        ! WIND SPEED AND DIRECTION
0263      IF (IMOD(i,2) .EQ. 1) THEN                            ! FOR 25m AND 60m
0264          IF (value(i) .GT. 999.8) THEN
0265              CALL number (13.8,ypos,8.28,value(i),8.8,1)  ! MISSING DATA
0266          ELSE
0267              wdir = (IIFIX ((value(i) / 16.8) + 0.5)) * 10 ! NON-MISSING DATA
0268              fnum = FLOATI (wdir)                          ! ROUND TO NEAREST
0269              CALL number (13.8,ypos,8.28,fnum,8.8,-1)      ! TEN DEGREES
0270          ENDIF
0271      ELSE
0272          IF (value(i) .GT. 999.8) THEN
0273              CALL number (13.8,ypos,8.28,value(i),8.8,1)  ! MISSING DATA, NO LABEL
0274          ELSE
0275              CALL number (13.8,ypos,8.28,value(i),8.8,-1) ! WIND SPEEDS
0276              CALL symbol (13.5,ypos,8.18,'mph',8.8,3)      ! LABEL
0277          ENDIF
0278      ENDIF
0279  ENDIF
0280
0281  IF (var_type .EQ. 14) THEN                                ! POTENTIAL
0282      CALL number (13.8,ypos,print_size,potential(i),8.8,1) ! TEMPERATURES
0283      CALL number (realx,realy,8.14,potential(i),8.8,1)
0284  ENDIF
0285
0286  ypos = ypos - step_size

```

```

0287      ENDDO
0288
0289      C
0290      C      PLOT ANY VECTORS FOR ANY WIND VECTORS
0291      C
0292
0293      IF ((var_type .EQ. 5).OR.(var_type .EQ. 6).OR.(var_type .EQ. 7)) THEN
0294          CALL pTot_vector(nparam,value,xcord,ycord)
0295      ENDIF
0296
0297      C
0298      C      CLOSE PLOTFILE AND DISPLAY
0299      C
0300
0301      CALL PLTEND
0302      CALL PLOTS (0.0, 0.0, 1)
0303      CALL NEWPAG
0304      CALL PLTOUT
0305      ISS = LIB$ERASE_PAGE(1,1)
0306
0307      RETURN
0308      END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	1861	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	482	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	2336	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	4599	

ENTRY POINTS

Address	Type	Name	References
0-00000000		PLOT_VAR	35

VARIABLES

Address	Type	Name	Attributes	References
AP-000000040	L+4	DATA READ		35 76 106A
2-00000320	R+4	FNUM		62 208= 209A 211= 212A
218=				
			217A 222= 223A 230= 231A 242=	
			243A 252= 253A 254A 255= 257A	
			258A 258= 260A	
**	I+2	I		51 202= 204 205 207
260				
			211 213 214 215 216 219	
			220 230 232 236 237 240	
			241 242 244 247 248 249	
			252 256 263 264 265 267	
			272 273 275 282 283	
**	I+4	ISS		64 87= 305=
**	I+2	J		51 170= 171(2) 172(2) 173(2)
174(2)				
			175(2)	
**	I+2	K		51

AP-000000100 I+2 NPARAM	35	52	189(2)=	184A	202
294A					
AP-0000002C0 I+2 PERIOD	35	50	161	251	
2-000000334 R+4 PRINT_SIZE	67	190=	199=	231A	232A
202A					
2-00000032C R+4 REALX	63	204=	237A	244A	249A
254A					
	258A	283A			
2-000000330 R+4 REALY	64	205=	237A	244A	249A
254A					
	258A	283A			
** R+4 STEP SIZE	68	195=	198=	200	
AP-000000000 I+2 VAR_TYPE	35	53	140	155	182
194					
	200	234(4)	239	240	202(2)
	203(3)				201
** I+2 WDIR	40	215=	219=	222	267=
208					
** I+2 WSPD	49				
2-000000324 R+4 YPOS	61	192=	209A	210A	212A
213A					
	217A	220A	223A	224A	220(2)=
	232A	235A	240A	241A	231A
	253A	257A	265A	269A	243A
	270A	282A	286(2)=	273A	248A
					275A

ARRAYS

Address	Type	Name	Attributes	Bytes	Dimensions	References		
AP-000000100	CHAR	ALPHA		3200	(200)	35	71	107=
213A								172(2)=
				232A				
AP-000000240	CHAR	DATE		8	(3)	35	72	143A
145A								144A
				164A				
2-000000170	CHAR	HEADERS		420	(14)	70	124D	140A
AP-000000140	I+4	NUM_ID		1200	(300)	35	60	168=
200								173(2)=
				211	230			
2-000000000	R+4	PARAMS		120	(32)	60	84A	
2-000000000	R+4	POTENTIAL		240	(60)	60	164A	202A
2-00000031C	CHAR	STAB_CLASSC		4	(4)	74	178A	241A
2-000000170	I+2	STAB_CLASSI		8	(4)	47	178A	242
AP-000000200	CHAR	TIME		4	(2)	35	73	147A
184A								148A
AP-0000000C0	R+4	VALUE		1200	(300)	35	65	166(3)=
178A								171(2)=
				184A	214	215	210	219
				220A	230A	237A	240A	244A
				247	248A	249A	252	250
				264	265A	267	272	273A
				275A	294A			
AP-0000001C0	R+4	XCORR		240	(60)	35	69	174(2)=
294A								204
AP-000000200	R+4	YCORR		240	(60)	35	60	175(2)=
294A								205

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	References						
	FACTOR			88				
	GET PLOT PAR			84				
	GET STABILITY			178				
	GET THETA			184				
I+4	LIBERASE_PAGE			55	87	385		
	NEWPAQ			85	383			
	NEWPEN			94	139	158		
	NUMBER			289	212	217	228	223
231		238	237	248	243	244	248	
		249	253	254	257	258	285	
		289	273	275	282	283		
	PLOT			95	98	97	98	99
188	PLOTS			83	382			
	PLOT VECTOR			294				
	PLTEND			381				
	PLTOUT			86	384			
	PLT GRAPHICS			188				
	SYMBOL			148	142	143	144	145
148		147	148	152	154	156	157	
		158	159	218	213	224	232	
		241	278					

KEY TO REFERENCE FLAGS	
.	- Value Modified
#	- Defining Reference
A	- Actual Argument, possibly modified
D	- Data Initialization
(n)	- Number of occurrences on line

COMMAND QUALIFIERS

```

FOR/LIS/EXTEND/CROSS/CHECK=ALL PLOT_VAR

/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)
/DEBUG=(NOSYMBOLS,TRACEBACK)
/STANDARD=(NOSYNTAX,NOSOURCE FORM)
/SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP,MODICTIONARY,SINGLE)
/WARNINGS=(GENERAL,NODECLARATIONS,NOULTRIX)
/CONTINUATIONS=19 /CROSS REFERENCE /NOD LINES /EXTEND SOURCE /F77
/NOQ_FLOATING /I4 /NOMACHINE_CODE /OPTIMIZE /NOANALYSIS

```

COMPILATION STATISTICS

```

Run Time:      18.57 seconds
Elapsed Time:  11.83 seconds
Page Faults:   788
Dynamic Memory: 478 pages

```

```

0001  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002  C
0003  C      WAIT.FOR
0004  C
0005  C      WRITTEN BY : KEN BURK
0006  C
0007  C      PURPOSE : TO PROVIDE A WAY TO 'WAIT', OR STOP PROCESSING FOR THE
0008  C                  NUMBER OF SECONDS SPECIFIED BY THE USER.
0009  C
0010  C      ALGORITHM : DECLARE VARIABLES
0011  C                  GET SYSTEM TIME
0012  C                  SEPERATE SYSTEM TIME INTO INTEGER PARTS
0013  C                  CONVERT INTEGER PARTS INTO REAL
0014  C                  CALCULATE TOTAL SECONDS
0015  C                  LOOP UNTIL DELTA TIME REACHES DESIRED
0016  C                  END ROUTINE
0017  C
0018  C      VERSION : 1.0
0019  C
0020  C      DATE : MARCH 1988
0021  C
0022  C      CALLED FROM : REVIEW.FOR
0023  C
0024  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0025  C
0026  C      SUBROUTINE wait(wait_time)
0027  C
0028  C      IMPLICIT NONE
0029  C
0030  C
0031  C      VARIABLE DECLARATIONS
0032  C
0033  C      TYPE          ! VARIABLE          ! COMMENTS
0034  C      -----
0035  C
0036  C
0037  C      INTEGER*2      chours              ! Current hour that routine was called
0038  C      INTEGER*2      cmin                ! Current minute that routine was called
0039  C      INTEGER*2      csec                ! Current second that routine was called
0040  C
0041  C      REAL*4          ctime              ! Current time in seconds
0042  C      REAL*4          wait_time          ! Number of seconds to 'wait'
0043  C      REAL*4          rhours             ! Real form of chours
0044  C      REAL*4          rmin               ! Real form of cmin
0045  C      REAL*4          rsec              ! Real form of csec
0046  C
0047  C      CHARACTER*8     systime            ! System current time
0048  C
0049  C
0050  C
0051  C      GET SYSTEM CURRENT TIME
0052  C
0053  C
0054  C      CALL TIME(systime)
0055  C
0056  C
0057  C      GET INTEGER PARTS OF CURRENT TIME, CONVERT TO REAL, GET TOTAL SECONDS
0058  C
0059  C
0060  C      READ(systime(1:2),'(i2)')chours    ! HOURS
0061  C      READ(systime(4:5),'(i2)')cmin      ! MINUTES
0062  C      READ(systime(7:8),'(i2)')csec      ! SECONDS
0063  C
0064  C      rsec = FLOATI(csec)
0065  C      rmin = FLOATI(cmin)

```

```

0066      rhours = FLOATI(chours)
0067
0068      ctime = rsec + (rmin * 60.0) + (rhours * 3600.0)
0069
0070      C
0071      C   CHECK DELTA TIME WITH 'WAIT_TIME'
0072      C
0073
0074      1   CONTINUE
0075
0076      IF(SECONDS(ctime) .GT. wait_time) THEN
0077          CONTINUE
0078      ELSE
0079          GOTO 1
0080      ENDIF
0081
0082      RETURN
0083      END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	242	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	8	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	62	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	297	

ENTRY POINTS

Address	Type	Name	References
0-00000000		WAIT	26#

VARIABLES

Address	Type	Name	Attributes	References
2-00000000	I*2	CHOURS		37 60= 66
2-0000000A	I*2	CMIN		38 61= 65
2-0000000C	I*2	CSEC		39 62= 64
2-00000010	R*4	CTIME		41 68= 76
**	R*4	RHOURS		43 68= 68
**	R*4	RMIN		44 65= 68
**	R*4	RSEC		45 64= 68
2-00000000	CHAR	SYSTIME		47 64A 60 61 62
AP-00000040	R*4	WAIT_TIME		28 42 76

LABELS

Address	Label	References
0-000000DC	1	74# 70

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	References
R*4	FOR\$SECONDS	78

0061

KEY TO REFERENCE FLAGS	
=	- Value Modified
#	- Defining Reference
A	- Actual Argument, possibly modified
D	- Data Initialization
(n)	- Number of occurrences on line

COMMAND QUALIFIERS

FOR/LIS/EXTEND/CROSS/CHECK=ALL WAIT

/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)

/DEBUG=(NOSYMBOLS,TRACEBACK)

/STANDARD=(NOSYNTAX,NOSOURCE FORM)

/SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP,NODICTIONARY,SINGLE)

/WARNINGS=(GENERAL,NODECLARATIONS,NOULTRIX)

/CONTINUATIONS=19 /CROSS REFERENCE /NOO LINES /EXTEND SOURCE /FT7

/NOO_FLOATING /I4 /NOMACHINE_CODE /OPTIMIZE /NOANALYSIS

COMPILATION STATISTICS

Run Time: 2.64 seconds

Elapsed Time: 2.78 seconds

Page Faults: 550

Dynamic Memory: 334 pages

```

0001 C *****
0002 C
0003 C PROGRAM - PDT.FOR
0004 C
0005 C PROGRAMMER - GREGG ANDREWS
0006 C
0007 C PURPOSE - TO DETERMINE IF CURRENT DATE AND TIME ARE
0008 C WITHIN THE PACIFIC DAYLIGHT TIME PERIOD;
0009 C FIRST SUNDAY OF APRIL THROUGH THE LAST
0010 C SATURDAY OF OCTOBER. THIS TIME PERIOD
0011 C HOLDS TRUE PROVIDED CONGRESS DOES NOT
0012 C CHANGE THE PERIOD FOR PACIFIC DAYLIGHT
0013 C TIME.
0014 C
0015 C ALGORITHM - DECLARE VARIABLES
0016 C GET SYSTEM MONTH AND DAY
0017 C INITIALIZE MONTH, DAY, AND PDT_FLAG
0018 C CHECK TO SEE IF DATE BETWEEN MAY AND SEPT.
0019 C IF SO SET PDT_FLAG TRUE
0020 C CHECK TO SEE IF DATE FALLS IN LAST THREE WEEKS
0021 C OF APRIL IF SO SET PDT_FLAG TRUE
0022 C CHECK TO SEE IF DATE FALLS BEFORE THIRD WEEK OF
0023 C OCTOBER IF SO SET PDT_FLAG TRUE
0024 C CHECK TO SEE IF FIRST SUNDAY OF APRIL HAS
0025 C OCCURRED IF SO SET PDT_FLAG TRUE
0026 C CHECK TO SEE IF LAST SUNDAY OF OCTOBER HAS
0027 C OCCURRED IF SO SET PDT_FLAG TRUE
0028 C
0029 C
0030 C VERSION - 1.0 MARCH 1988
0031 C
0032 C *****
0033 C
0034 C SUBROUTINE pdt(month, day, pdt_flag)
0035 C
0036 C IMPLICIT NONE
0037 C
0038 C INTEGER*2 month
0039 C INTEGER*2 day
0040 C INTEGER*4 day_num
0041 C INTEGER*4 status
0042 C INTEGER*4 LIB$DAY_OF_WEEK
0043 C INTEGER*4 check
0044 C
0045 C LOGICAL pdt_flag
0046 C
0047 C pdt_flag = .FALSE.
0048 C
0049 C status = LIB$DAY_OF_WEEK(day_num) ! a vax service
0050 C
0051 C IF((month.GE.5) .AND. (month.LE.9)) THEN
0052 C pdt_flag = .TRUE.
0053 C ELSE IF((month.EQ.4) .AND. ((day.GE.8) .AND. (day.LE.30))) THEN
0054 C pdt_flag = .TRUE.
0055 C ELSE IF((month.EQ.10) .AND. (day.LE.24)) THEN
0056 C pdt_flag = .TRUE.
0057 C ELSE IF((month.EQ.4) .AND. ((day.GE.1) .AND. (day.LE.7))) THEN
0058 C IF(day_num .EQ. 7) THEN
0059 C pdt_flag = .TRUE.
0060 C ELSE
0061 C check = day - day_num
0062 C IF(check.GE.1) pdt_flag = .TRUE.
0063 C ENDIF
0064 C ELSE IF((month.EQ.10) .AND. ((day.GE.25) .AND. (day.LE.31))) THEN
0065 C IF(day_num .NE. 7) THEN

```

```

0066             check = (day num + 26) - day
0067             IF(check.GE.0) pdt_flag = .TRUE.
0068             ENDIF
0069         ENDIF
0070
0071
0072         RETURN
0073         END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	160	PIC COM REL LCL SHR EXE RD NOWRT LONG
2 \$LOCAL	18	PIC COM REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	188	

ENTRY POINTS

Address	Type	Name	References
0-00000000	PDT		34#

VARIABLES

Address	Type	Name	Attributes	References
**	I+4	CHECK		43 61= 62 66= 67
AP-00000000	I+2	DAY		34 39 53(2) 55 57(2)
61			64(2) 66	
2-00000000	I+4	DAY_NUM		48 49A 58 61 65
66				
AP-00000040	I+2	MONTH		34 38 61(2) 53 55
57			64	
AP-000000C0	L+4	PDT_FLAG		34 45 47= 52= 54=
56=			59= 62= 67=	
**	I+4	STATUS		41 49=

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	References
I+4	LIB\$DAY_OF_WEEK	42 49

KEY TO REFERENCE FLAGS	
=	- Value Modified
#	- Defining Reference
A	- Actual Argument, possibly modified
D	- Data Initialization
(n)	- Number of occurrences on line

COMMAND QUALIFIERS

FOR/LIS/EXTEND/CROSS/CHECK=ALL PDT

/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)

/DEBUG=(NOSYMBOLS,TRACEBACK)

/STANDARD=(NOSYNTAX,NOSOURCE FORM)

/SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP,NODICTIONARY,SINGLE)

/WARNINGS=(GENERAL,NODECLARATIONS,NOULTRIX)

/CONTINUATIONS=19 /CROSS REFERENCE /NOO LINES /EXTEND SOURCE /F77

/NOG_FLOATING /I4 /NOMACHINE_CODE /OPTIMIZE /NOANALYSIS

COMPILATION STATISTICS

Run Time: 2.21 seconds

Elapsed Time: 2.91 seconds

Page Faults: 498

Dynamic Memory: 338 pages

```

0001 C *****
0002 C
0003 C PROGRAM - PST_ADJUSTMENT.FOR
0004 C
0005 C PROGRAMMER - GREGG ANDREWS
0006 C
0007 C PURPOSE - SUBTRACTS ONE HOUR TO HOUR IF THE DATE FALLS WITHIN PDT.
0008 C           WHICH BEGINS THE FIRST SUNDAY OF APRIL (AT MIDNIGHT
0009 C           ASSUMED IN THIS PROGRAM) AND ENDS ON THE LAST SUNDAY OF
0010 C           OCTOBER (AT MIDNIGHT ASSUMED IN THIS PROGRAM). ADJUSTS FROM
0011 C           PDT TO PST.
0012 C
0013 C ALGORITHM - CHECK FOR END OF MONTH
0014 C           CHECK FOR END OF DAY
0015 C           IF NOT MONTH OR DAY SUBTRACT AN HOUR
0016 C
0017 C PASSED PARAMETERS - gmon - GREGORIAN MONTH
0018 C                   gday - GREGORIAN DAY
0019 C                   ghour - HOUR
0020 C
0021 C VERSION - 1.0   MAR 1988
0022 C             2.0   APR 1988 GJA ELIMINATE UNNECESSARY CODE
0023 C
0024 C *****
0025 C
0026 SUBROUTINE pst_adjustment(gmon, gday, ghour)
0027
0028 IMPLICIT NONE
0029
0030 INTEGER*4 gmon
0031 INTEGER*4 gday
0032 INTEGER*4 ghour
0033 INTEGER*4 end_of_month(12)
0034
0035 DATA end_of_month /31,28,31,30,31,30,31,31,30,31,30,31/
0036
0037
0038 C
0039 C..... MAKE ADJUSTMENT FROM PDT TO PST (SUBTRACT AN HOUR)
0040 C
0041
0042 IF((gday.EQ.1).AND.(ghour.EQ.0)) THEN ! check for beginning of month
0043     gmon = gmon - 1
0044     gday = end_of_month(gmon)
0045     ghour = 23
0046 ELSE IF(ghour.EQ.0) THEN ! check for beginning of day
0047     gday = gday - 1
0048     ghour = 23
0049 ELSE
0050     ghour = ghour - 1
0051 ENDIF
0052
0053 RETURN
0054 END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	64	PIC CON REL LCL SHR EXE RD NOWRT LONG
2 \$LOCAL	48	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	112	

ENTRY POINTS

Address	Type	Name	References
6-00000000		PST_ADJUSTMENT	26#

VARIABLES

Address	Type	Name	Attributes	References
AP-00000000	I+4	QDAY		26 31 42 44= 47(2)=
AP-000000C0	I+4	QHOUR		26 32 42 45= 46
			48=	
			50(2)=	
AP-00000040	I+4	QMON		26 36 43(2)= 44

ARRAYS

Address	Type	Name	Attributes	Bytes	Dimensions	References
2-00000000	I+4	END_OF_MONTH		48	(12)	33 35D 44

```

+-----+
|      KEY TO REFERENCE FLAGS      |
|  =  - Value Modified              |
|  #  - Defining Reference          |
|  A  - Actual Argument, possibly  |
|       modified                   |
|  D  - Data Initialization         |
|  (n) - Number of occurrences on  |
|       line                       |
+-----+

```

COMMAND QUALIFIERS

```

FOR/LIS/EXTEND/CROSS/CHECK=ALL PST_ADJUSTMENT

/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)
/DEBUG=(NOSYMBOLS,TRACEBACK)
/STANDARD=(NOSYNTAX,NOSOURCE FORM)
/SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP,NODICTIONARY,SINGLE)
/WARNINGS=(GENERAL,NODECLARATIONS,NOULTRIX)
/CONTINUATIONS=19 /CROSS REFERENCE /NOO LINES /EXTEND SOURCE /FTT
/NOG_FLOATING /I4 /NOMACHINE_CODE /OPTIMIZE /NOANALYSIS

```

COMPILATION STATISTICS

```

Run Time:          1.47 seconds
Elapsed Time:      2.58 seconds
Page Faults:       528
Dynamic Memory:    331 pages

```

```

0001
0002 C *****
0003 C
0004 C      SUBROUTINE - GET_HLG_FILENAME
0005 C
0006 C PROGRAMMER - GREGG ANDREWS
0007 C
0008 C      PURPOSE - TO RETURN THE FILE NAME OF THE CURRENT ERDS HOURLY LOG FILE.
0009 C                USES THE SYSTEM TIME TO BUILD THE NAME. PROGRAM WAS ORIGINALLY
0010 C                WRITTEN BY KAREN S. MODIFIED TO FIT NEEDS OF THE GET_EMS_DATA
0011 C                SUBROUTINE
0012 C
0013 C
0014 C DEFINITION OF STIME -   STIME(1) --> YEAR
0015 C                        STIME(2) --> MONTH
0016 C                        STIME(3) --> DAY
0017 C                        STIME(4) --> HOUR
0018 C
0019 C ALGORITHM -   EQUATE DATE AND TIME TO STIME ARRAY
0020 C               CHECK FOR SPECIAL CASE OF HOUR # AND JAN. 1
0021 C               CHECK FOR LEAP YEAR
0022 C               CHECK FOR BEGIN OF MONTH
0023 C               ELSE
0024 C               CHECK FOR BEGIN OF MONTH
0025 C               ENDIF
0026 C               WRITE MODIFIED DAY AND UNMODIFIED YEAR TO CURRENT VARIABLES
0027 C               BUILD FILENAME
0028 C
0029 C IT IS NECESSARY TO MODIFY THE DAY WHEN HOUR EQUAL # BECAUSE DATA FOR
0030 C THE # HOUR IS KEPT IN THE PREVIOUS'S DAY FILE; DON'T ASK !!!!.
0031 C
0032 C   PASSED PARAMETERS - EMS_FILENAME this filename built in this subroutine
0033 C                       DATE date passed from GET_EMS_DATA (E.G., 871231)
0034 C                       TIME time passed from GET_EMS_DATA (E.G., 2315)
0035 C
0036 C VERSION - 1.0
0037 C
0038 C DATE - NOVEMBER 1987
0039 C
0040 C *****
0041
0042 SUBROUTINE get_hlg_filename(hlgfile,ddate,ttime)
0043
0044 IMPLICIT NONE
0045
0046 INTEGER*2 i
0047 INTEGER*2 ai, stime(4), ddate(3), ttime(2)
0048 INTEGER      days_in_month_ly(12), days_in_month(12)
0049 CHARACTER current_year*10, current_month*3, current_day*2
0050 CHARACTER*(30) hlgfile
0051
0052 DATA days_in_month/31, 28, 31, 30, 31, 30, 31, 31, 30, 31,
0053        30, 31/
0054
0055 DATA days_in_month_ly/31, 29, 31, 30, 31, 30, 31, 31, 30,
0056        31, 30, 31/
0057
0058
0059 C
0060 C..... PUT INFORMATION FROM DOATE AND TTIME INTO STIME ARRAY
0061 C
0062
0063 stime(1) = ddate(1)
0064 stime(2) = ddate(2)
0065 stime(3) = ddate(3)

```

```

0066      stime(4) = ttime(1)
0067
0068      C
0069      C..... CHECK FOR SPECIAL CASE OF JAN. 1 HOUR #; HOUR # IS CONTAINED IN THE
0070      C..... THE PREVIOUS DAY'S FILE
0071      C
0072
0073
0074      IF( (stime(2).EQ.1) .AND. (stime(3).EQ.1) .AND.
0075      .    (stime(4).EQ.0) ) THEN
0076
0077          stime(1) = stime(1) - 1
0078          stime(2) = 12
0079          stime(3) = 31
0080
0081      ELSE
0082
0083          IF(MOD(stime(1),4) .EQ. 0) THEN ! CHECK FOR LEAP YEAR
0084              IF( (stime(3).EQ.1) .AND. (stime(4).EQ.0) ) THEN
0085                  stime(2) = stime(2) - 1
0086                  stime(3) = days_in_month_ly(stime(2))
0087              ELSE
0088                  IF(stime(4).EQ.0) stime(3) = stime(3) - 1
0089              ENDIF
0090          ELSE
0091              IF( (stime(3).EQ.1) .AND. (stime(4).EQ.0) ) THEN
0092                  stime(2) = stime(2) - 1
0093                  stime(3) = days_in_month(stime(2))
0094              ELSE
0095                  IF(stime(4).EQ.0) stime(3) = stime(3) - 1
0096              ENDIF
0097          ENDIF
0098      ENDIF
0099
0100      WRITE(current_year,'(i16)') stime(1)
0101      WRITE(current_day,'(i2)') stime(3)
0102
0103
0104      C
0105      C..... ELIMINATE ANY SPACES IN DAY (I.E. IF DAY IS <= 0)
0106      C
0107
0108
0109      IF(current_day(1:1) .eq. ' ') THEN
0110          si = 2
0111      ELSE
0112          si = 1
0113      ENDIF
0114
0115      CALL get_month(stime(2),current_month)
0116
0117      hlgfile = '[EROSYS.DAT.SENSORS.622R] '//
0118      .          current_month//current_day(si:2)//
0119      .          current_year(15:18)//'.HLG'
0120
0121      RETURN
0122      END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
------	-------	------------

0	SCODE	420	PIC	CON	REL	LCL	SHR	EXE	RD	NOWRT	LONG
1	SPDATA	32	PIC	CON	REL	LCL	SHR	NOEXE	RD	NOWRT	LONG
2	SLOCAL	212	PIC	CON	REL	LCL	NOSHR	NOEXE	RD	WRT	LONG

Total Space Allocated 884

ENTRY POINTS

Address	Type	Name	References
0-00000000		GET_HLG_FILENAME	42#

VARIABLES

Address	Type	Name	Attributes	References
2-00000070	CHAR	CURRENT DAY	40	101 109 117
2-00000078	CHAR	CURRENT MONTH	40	115A 117
2-00000088	CHAR	CURRENT YEAR	40	100 117
AP-00000040	CHAR	HLGFILE	42	50 117=
**	I+2	I	48	
**	I+2	SI	47	110= 112= 117

ARRAYS

Address	Type	Name	Attributes	Bytes	Dimensions	References
2-00000030	I+4	DAYS_IN_MONTH		48	(12)	48 520
93						
2-00000000	I+4	DAYS_IN_MONTH_LY		48	(12)	48 550
88						
AP-00000000	I+2	DOATE		8	(3)	42 47
63			64	85		
2-00000000	I+2	STIME		8	(4)	47 63=
64=			65=	68=	74(3) 77(2)= 78=	
			70=	83	84(2) 85(2)= 86(2)=	
			88(3)=	91(2)	92(2)= 93(2)= 95(3)=	
			100	101	115A	
AP-000000C0	I+2	TTIME		4	(2)	42 47
65						

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	References
	GET_MONTH	115

KEY TO REFERENCE FLAGS	
=	- Value Modified
#	- Defining Reference
A	- Actual Argument, possibly modified
D	- Data Initialization
(n)	- Number of occurrences on line

COMMAND QUALIFIERS

FOR/LIS/EXTEND/CROSS/CHECK=ALL GET_HLG_FILENAME

/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)

/DEBUG=(NOSYMBOLS,TRACEBACK)

/STANDARD=(NOSYNTAX,NOSOURCE FORM)

/SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP,NODICTIONARY,SINGLE)

/WARNINGS=(GENERAL,NODECLARATIONS,NOULTRIX)

/CONTINUATIONS=19 /CROSS REFERENCE /NOD LINES /EXTEND SOURCE /F77

/NOG_FLOATING /I4 /NOMACHINE_CODE /OPTIMIZE /NOANALYSIS

COMPILATION STATISTICS

Run Time: 3.54 seconds

Elapsed Time: 4.24 seconds

Page Faults: 838

Dynamic Memory: 376 pages

```

0001
0002 C *****
0003 C
0004 C      SUBROUTINE - GET_QLO_FILENAME
0005 C
0006 C PROGRAMMER - GREGG ANDREWS
0007 C
0008 C      PURPOSE - TO RETURN THE FILE NAME OF THE CURRENT ERDS HOURLY LOG FILE.
0009 C                USES THE SYSTEM TIME TO BUILD THE NAME. PROGRAM WAS ORIGINALLY
0010 C                WRITTEN BY KAREN S. MODIFIED TO FIT NEEDS OF THE GET_EMS_DATA
0011 C                SUBROUTINE
0012 C
0013 C
0014 C DEFINITION OF STIME -   STIME(1) --> YEAR
0015 C                        STIME(2) --> MONTH
0016 C                        STIME(3) --> DAY
0017 C                        STIME(4) --> HOUR
0018 C
0019 C ALGORITHM -   EQUATE DATE AND TIME TO STIME ARRAY
0020 C               CHECK FOR SPECIAL CASE OF HOUR 0 AND JAN. 1
0021 C               CHECK FOR LEAP YEAR
0022 C               CHECK FOR BEGIN OF MONTH
0023 C               ELSE
0024 C               CHECK FOR BEGIN OF MONTH
0025 C               ENDIF
0026 C               WRITE MODIFIED DAY AND UNMODIFIED YEAR TO CURRENT VARIABLES
0027 C               BUILD FILENAME
0028 C
0029 C IT IS NECESSARY TO MODIFY THE DAY WHEN HOUR EQUAL 0 BECAUSE DATA FOR
0030 C THE 0 HOUR IS KEPT IN THE PREVIOUS'S DAY FILE; DON'T ASK !!!!.
0031 C
0032 C
0033 C      PASSED PARAMETERS - EMS FILENAME, DATE, AND TIME
0034 C
0035 C VERSION - 1.0
0036 C
0037 C DATE - NOVEMBER 1987
0038 C
0039 C *****
0040 C
0041 C      SUBROUTINE get_qlg_filename(qlgfile,ddate,ttime)
0042 C
0043 C      IMPLICIT NONE
0044 C
0045 C      INTEGER*2 i
0046 C      INTEGER*2 si, stime(4), ddate(3), ttime(2)
0047 C      INTEGER      days_in_month_ly(12), days_in_month(12)
0048 C      CHARACTER current_year*10, current_month*3, current_day*2
0049 C      CHARACTER*(30) qlgfile
0050 C
0051 C      DATA days_in_month/31, 28, 31, 30, 31, 30, 31, 31, 30, 31,
0052 C              30, 31/
0053 C
0054 C      DATA days_in_month_ly/31, 29, 31, 30, 31, 30, 31, 31, 30,
0055 C              31, 30, 31/
0056 C
0057 C
0058 C
0059 C ..... PUT INFORMATION FROM DDATE AND TTIME INTO STIME ARRAY
0060 C
0061 C
0062 C      stime(1) = ddate(1)
0063 C      stime(2) = ddate(2)
0064 C      stime(3) = ddate(3)
0065 C      stime(4) = ttime(1)

```

```

0066
0067 C
0068 C..... CHECK FOR SPECIAL CASE OF JAN. 1 HOUR 0; HOUR 0 IS CONTAINED IN THE
0069 C..... THE PREVIOUS DAY'S FILE
0070 C
0071
0072
0073 IF( (stime(2).EQ.1) .AND. (stime(3).EQ.1) .AND.
0074 (stime(4).EQ.0) )THEN
0075
0076 stime(1) = stime(1) - 1
0077 stime(2) = 12
0078 stime(3) = 31
0079
0080 ELSE
0081
0082 IF(MOD(stime(1),4) .EQ. 0) THEN ! CHECK FOR LEAP YEAR
0083 IF( (stime(3).EQ.1) .AND. (stime(4).EQ.0) ) THEN
0084 stime(2) = stime(2) - 1
0085 stime(3) = days_in_month_ly(stime(2))
0086 ELSE
0087 IF(stime(4).EQ.0) stime(3) = stime(3) - 1
0088
0089 ENDIF
0090 ELSE
0091 IF( (stime(3).EQ.1) .AND. (stime(4).EQ.0) ) THEN
0092 stime(2) = stime(2) - 1
0093 stime(3) = days_in_month(stime(2))
0094 ELSE
0095 IF(stime(4).EQ.0) stime(3) = stime(3) - 1
0096
0097 ENDIF
0098
0099 ENDIF
0100
0101 WRITE(current_year,'(i16)') stime(1)
0102 WRITE(current_day,'(i2)') stime(3)
0103
0104 C
0105 C..... ELIMINATE ANY SPACES IN DAY (I.E. IF DAY IS <= 0)
0106 C
0107
0108 IF(current_day(1:1) .eq. ' ') THEN
0109 si = 2
0110 ELSE
0111 si = 1
0112
0113 ENDIF
0114
0115 CALL get_month(stime(2),current_month)
0116
0117 qlgfile = '[ERDSSYS.DAT.MESORAD.INPUT]//
0118 current_month//current_day(ai:2)//
0119 current_year(15:16)///'.QLG'
0120
0121 RETURN
0122 END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
------	-------	------------

0 %CODE	420	PIC	CON	REL	LCL	SHR	EXE	RD	NOWRT	LONG
1 %PDATA	83	PIC	CON	REL	LCL	SHR	NOEXE	RD	NOWRT	LONG
2 %LOCAL	212	PIC	CON	REL	LCL	NOSHR	NOEXE	RD	WRT	LONG

Total Space Allocated 685

ENTRY POINTS

Address	Type	Name	References
0-00000000		GET_QLO_FILENAME	41‡

VARIABLES

Address	Type	Name	Attributes	References
2-00000070	CHAR	CURRENT_DAY		48 100 108 116
2-00000078	CHAR	CURRENT_MONTH		48 114A 116
2-00000086	CHAR	CURRENT_YEAR		48 99 116
**	I*2	I		45
AP-00000040	CHAR	QLGFILE		41 49 116=
**	I*2	SI		48 100= 111= 116

ARRAYS

Address	Type	Name	Attributes	Bytes	Dimensions	References
2-00000030	I*4	DAYS_IN_MONTH		48	(12)	47 51D
92						
2-00000060	I*4	DAYS_IN_MONTH_LY		48	(12)	47 54D
85						
AP-00000080	I*2	DDATE		6	(3)	41 48
62						
			63	84		
2-00000060	I*2	STIME		8	(4)	48 62=
63=						
			64=	65=	73(3)	76(2)= 77=
			78=	82	83(2)	84(2)= 85(2)=
			87(3)=	98(2)	91(2)=	92(2)= 94(3)=
			99	100	114A	
AP-000000C0	I*2	TTIME		4	(2)	41 48
65						

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	References
	GET_MONTH	114

KEY TO REFERENCE FLAGS	
=	- Value Modified
‡	- Defining Reference
A	- Actual Argument, possibly modified
D	- Data Initialization
(n)	- Number of occurrences on line

COMMAND QUALIFIERS

FOR/LIS/EXTEND/CROSS/CHECK=ALL GET_QLG_FILENAME

/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)

/DEBUG=(NOSYMBOLS,TRACEBACK)

/STANDARD=(NOSYNTAX,NOSOURCE FORM)

/SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP,NODICTIONARY,SINGLE)

/WARNINGS=(GENERAL,NODECLARATIONS,NOLTRIX)

/CONTINUATIONS=19 /CROSS REFERENCE /MOD LINES /EXTEND SOURCE /F77

/NOG_FLOATING /I4 /NOMACHINE_CODE /OPTIMIZE /NOANALYSIS

COMPILATION STATISTICS

Run Time: 3.54 seconds

Elapsed Time: 4.29 seconds

Page Faults: 624

Dynamic Memory: 375 pages

```

0001
0002 C *****
0003 C
0004 C SUBROUTINE - GET_MONTH
0005 C
0006 C PROGRAMMER - GREGG ANDREWS
0007 C
0008 C     PURPOSE - TO RETURN THE THREE CHARACTER STRING REPRESENTATION
0009 C               OF AN INTEGER MONTH (E.G., 3 = MAR)
0010 C
0011 C ALGORITHM - PASS INTEGER VALUE OF MONTH PASS BACK THREE CHAR
0012 C
0013 C     PASSED PARAMETERS - TIME(2) AND MONTH WHERE TIME(2) IS THE INTEGER
0014 C                       VALUE AND MONTH IS THE CHARACTER VALUE.
0015 C
0016 C VERSION - 1.0
0017 C
0018 C DATE - NOVEMBER 1987
0019 C
0020 C *****
0021
0022     SUBROUTINE get_month(imonth,smmonth)
0023
0024
0025     IMPLICIT NONE
0026
0027     INTEGER*2 imonth
0028     CHARACTER*3 smmonth, months(12)
0029
0030     DATA months / 'JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN',
0031                   'JUL', 'AUG', 'SEP', 'OCT', 'NOV', 'DEC' /
0032
0033     smmonth = months(imonth)
0034
0035     RETURN
0036     END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	30	PIC CON REL LCL SHR EXE RD NOWRT LONG
2 \$LOCAL	44	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	83	

ENTRY POINTS

Address	Type	Name	References
0-00000000		GET_MONTH	22

VARIABLES

Address	Type	Name	Attributes	References
AP-00000040	I*2	IMONTH		22 27 33
AP-00000080	CHAR	SMONTH		22 28 33

ARRAYS

Address	Type	Name	Attributes	Bytes	Dimensions	References
2-00000000	CHAR	MONTHS		36	(12)	28 300 33

```

+-----+
|               KEY TO REFERENCE FLAGS               |
|  =  - Value Modified                               |
|  #  - Defining Reference                           |
|  A  - Actual Argument, possibly modified            |
|  D  - Data Initialization                           |
|  (n) - Number of occurrences on line                |
+-----+

```

COMMAND QUALIFIERS

FOR/LIS/EXTEND/CROSS/CHECK=ALL GET_MONTH

/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)

/DEBUG=(NOSYMBOLS,TRACEBACK)

/STANDARD=(NOSYNTAX,NOSOURCE FORM)

/SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP,NODICTIONARY,SINGLE)

/WARNINGS=(GENERAL,NODECLARATIONS,NOULTRIX)

/CONTINUATIONS=19 /CROSS REFERENCE /MOD LINES /EXTEND SOURCE /F77

/NOG_FLOATING /I4 /NOMACHINE_CODE /OPTIMIZE /NOANALYSIS

COMPILATION STATISTICS

Run Time: 1.85 seconds

Elapsed Time: 1.76 seconds

Page Faults: 506

Dynamic Memory: 315 pages

```

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 C
0003 C      GET_STABILITY.FOR
0004 C
0005 C      WRITTEN BY :  G. L. ANDREWS
0006 C      MODIFIED BY:  K. W. BURK
0007 C
0008 C      PURPOSE :  DETERMINES THE ATMOSPHERIC STABILITY USING THE
0009 C                  DELTA-TEMPERATURE SCHEME DEVELOPED BY THE NRC; NUREG-0917
0010 C                  CONTAINS SOME INFORMATION ON THE SCHEME.  THE
0011 C                  DELTA-TEMPERATURE IS SUPPLIED BY THE EMS
0012 C                  DATA BASE.
0013 C
0014 C      ALGORITHM :  DECLARE VARIABLES
0015 C                  FIND STABILITY CLASS
0016 C                  END ROUTINE
0017 C
0018 C
0019 C      VERSION:  2.0
0020 C
0021 C      COATE:  FEBRUARY 1988
0022 C
0023 C      CALLED FROM :  PLOT_VAR.FOR
0024 C
0025 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0026 C
0027 C      SUBROUTINE get_stability(delta_temp, classi, classc)
0028 C
0029 C      IMPLICIT NONE
0030 C
0031 C
0032 C      VARIABLE DECLARATIONS
0033 C-----
0034 C      TYPE      |  VARIABLE      |  COMMENTS
0035 C-----
0036 C
0037 C
0038 C      INTEGER*2  classi(4)      ! Integer array for stability class
0039 C      INTEGER    i              ! Loop index
0040 C
0041 C      REAL*4     delta_temp(4)  ! Array of delta temperatures
0042 C
0043 C      CHARACTER*1 classc(4)     ! Character array for stability class
0044 C
0045 C
0046 C
0047 C      DETERMINE STABILITY CLASS
0048 C
0049 C
0050 C      DO i = 1,4
0051 C
0052 C          classi(i) = 99
0053 C          classc(i) = '99'
0054 C
0055 C          IF((delta_temp(i).GT.4.0) .AND. (delta_temp(i).LT.30)) THEN      ! VERY STABLE
0056 C              classi(i) = 7
0057 C              classc(i) = 'Q'
0058 C          ENDIF
0059 C
0060 C          IF(delta_temp(i) .LE. 4.0) THEN      ! STABLE
0061 C              classi(i) = 6
0062 C              classc(i) = 'F'
0063 C          ENDIF
0064 C
0065 C          IF(delta_temp(i) .LE. 1.5) THEN      ! SLIGHTLY STABLE

```

```

0068         classi(i) = 5
0069         classc(i) = 'E'
0070     ENDIF
0071     IF(delta temp(i) .LE. -0.5) THEN      ! NEUTRAL
0072         classi(i) = 4
0073         classc(i) = 'D'
0074     ENDIF
0075     IF(delta temp(i) .LE. -1.5) THEN      ! SLIGHTLY UNSTABLE
0076         classi(i) = 3
0077         classc(i) = 'C'
0078     ENDIF
0079     IF(delta temp(i) .LE. -1.7) THEN      ! UNSTABLE
0080         classi(i) = 2
0081         classc(i) = 'B'
0082     ENDIF
0083     IF(delta temp(i) .LE. -1.9) THEN      ! VERY UNSTABLE
0084         classi(i) = 1
0085         classc(i) = 'A'
0086     ENDIF
0087
0088     ENDDO
0089
0090     RETURN
0091
0092     END
0093

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	376	PIC COM REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	2	PIC COM REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	72	PIC COM REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	449	

ENTRY POINTS

Address	Type	Name	References
0-00000000		GET_STABILITY	270

VARIABLES

Address	Type	Name	Attributes	References
**	I=4	I		39 50= 62 63 65(2) 66 57
60			61 62 65 66 67 70 71 72	
			75 76 77 80 81 82 85 86	
			87	

ARRAYS

Address	Type	Name	Attributes	Bytes	Dimensions	References
AP-00000000	CHAR	CLASSC		4	(4)	27 43 53= 57=
82=						

AP-000000000 I+2 CLASSI
81=

AP-000000040 R+4 DELTA_TEMP
65

67=	72=	77=	82=	87=		
8 (4)			27	38	52=	58=
66=	71=	76=	81=	86=		
18 (4)			27	41	55(2)	60
70	75	80	85			

KEY TO REFERENCE FLAGS	
=	- Value Modified
#	- Defining Reference
A	- Actual Argument, possibly modified
D	- Data Initialization
(n)	- Number of occurrences on line

COMMAND QUALIFIERS

FOR/LIS/EXTEND/CROSS/CHECK=ALL GET_STABILITY

/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)

/DEBUG=(NOSYMBOLS,TRACEBACK)

/STANDARD=(NOSYNTAX,NOSOURCE FORM)

/SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP,NOdictionary,SINGLE)

/WARNINGS=(GENERAL,NODECLARATIONS,NOULTRIX)

/CONTINUATIONS=10 /CROSS REFERENCE /MOD LINES /EXTEND SOURCE /F77

/NOG_FLOATING /I4 /NOMACHINE_CODE /OPTIMIZE /NOANALYSIS

COMPILATION STATISTICS

Run Time: 2.85 seconds
Elapsed Time: 3.31 seconds
Page Faults: 640
Dynamic Memory: 830 pages


```

0066      REAL*4      p          ! Intermediate calculation
0067
0068      CHARACTER*2   ddate(3)   ! Desired date
0069      CHARACTER*2   dtime(2)   ! Desired time
0070      CHARACTER*3   months(12) ! Three character months
0071
0072      DATA elev /224., 232., 283., 136., 144., 194., 188., 168., 218.,
0073 +      121., 129., 179., 1889., 388., 212., 244., 124., 197., 288.,
0074 +      178., 333., 139., 381., 174., 271., 191., 284., 148., 128.,
0075 +      161., 124., 135., 128./
0076
0077      DATA months /'JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL',
0078 *      'AUG', 'SEP', 'OCT', 'NOV', 'DEC' /
0079
0080      r = 2.8784E8
0081      g = 9.805E2
0082
0083      elev1 = elev(1) * 100.
0084
0085      C
0086      C      CONVERT THE DESIRED DATE FROM CHARACTER TO INTEGER
0087      C
0088
0089      READ(ddate(1),'(i2)')iddate(1)
0090      READ(ddate(2),'(i2)')iddate(2)
0091      READ(ddate(3),'(i2)')iddate(3)
0092      READ(dtime(1),'(i2)')itime
0093
0094      C
0095      C      CHECK TO SEE IF THE DESIRED DATE IS THE SAME AS THE
0096      C      CURRENT SYSTEM DATE. IF IT IS, THEN OPEN CURRENT DAY'S
0097      C      SURFACE OBSERVATIONS FILE, IF NOT THEN OPEN MONTHLY ARCHIVE FILE.
0098      C
0099
0100      CALL IDATE(syear,syear,syear)          ! GET SYSTEM DATE
0101
0102      IF((iddate(1) .EQ. syear) .AND. (iddate(2) .EQ. syear) .AND.
0103 *      (iddate(3) .EQ. syear)) THEN
0104
0105          OPEN(UNIT=55,FILE='fcst.dat)dailysfcobs.dat',
0106 *          STATUS='OLD',READONLY,SHARED)    ! CURRENT DAY FILE
0107
0108      ELSE
0109
0110          OPEN(UNIT=55,FILE='fcst.dat)sfcobs'//months(iddate(2))//'.dat',
0111 *          STATUS='OLD',READONLY,SHARED)    ! MONTHLY ARCHIVE FILE
0112
0113      ENDIF
0114
0115      C
0116      C      READ FROM THE SFCOBS FILE UNTIL THE DESIRED DATE IS FOUND
0117      C
0118
0119      55 READ (55,120,END=999)pdate(1),pdate(2),pdate(3),phour,press
0120      120 FORMAT(5x,4i2,29x,i4)
0121
0122      IF((pdate(1) .NE. iddate(1)) .OR. (pdate(2) .NE. iddate(2)) .OR. ! DATE NOT
0123 *      (pdate(3) .NE. iddate(3)) .OR. (phour .NE. itime)) GOTO 55 ! EQUAL
0124
0125      CLOSE(55)
0126
0127      C
0128      C      BEGIN TO PROCESS THE TEMPERATURES USING THE PRESSURE JUST READ IN
0129      C
0130

```



```

0131      DO n = 1,nloop
0132
0133          p1 = (pres/100.)*33.8639          ! CONVERT PRESSURE TO MILLIBARS
0134
0135          tk(n) = (5.+(tempa(n) - 32)/9.) *273.
0136
0137          IF(n.EQ.1)THEN
0138              theto = tk(n) * (1000. / p1)**.2857
0139          ELSE
0140              tbar = (tk(1) + tk(n)) / 2.
0141              lp = LOG (p1 * 1000.) + (elev1 - (elev(n) * 100.)) * g / (r * tbar)
0142              p = EXP(lp) / 1000.
0143              theto = tk(n) * (1000./p)**.2857
0144          ENDIF
0145
0146          IF(tempa(n) .GT. 999.9) THEN      ! IF THE TEMPERATURE IS MISSING
0147              potential(n) = 999.9          ! MAKE POTENTIAL MISSING ALSO
0148          ELSE
0149              potential(n) = theto
0150          ENDIF
0151
0152
0153      ENDDO
0154
0155      RETURN
0156  999  TYPE*, '*****'
0157      TYPE*, ' ERROR WHILE READING STATION PRESSURE,'
0158      TYPE*, '   END OF FILE OCCURED.'
0159      TYPE*, '*****'
0160      CALL wait(4)
0161
0162      RETURN
0163      END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	918	PIC.COM REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	182	PIC.COM REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	680	PIC.COM REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	1680	

ENTRY POINTS

Address	Type	Name	References
0-00000000		GET_THETA	339

VARIABLES

Address	Type	Name	Attributes	References
2-00000164	R*4	ELEV1	60	89= 141
**	R*4	Q	59	81= 141
2-0000016E	I*2	ITIME	52	92= 122
**	R*4	LP	65	141= 142
**	I*2	M	53	131= 135(2) 137 138 140 141
143			146 147 149	

AP-000000040	I*2	ML00P	33	44	131	
**	R*4	P	66	142=	143	
**	R*4	P1	61	133=	138	141
2-000000154	I*2	PHOUR	46	119=	122	
2-000000156	I*2	PRES	47	119=	133	
**	R*4	R	58	86=	141	
2-00000015A	I*2	SYSD	49	100A	102	
2-00000015B	I*2	SYSM	48	100A	102	
2-00000015C	I*2	SYSY	50	100A	102	
**	R*4	TBAR	64	146=	141	
**	R*4	THETO	63	138=	143=	149

ARRAYS

Address	Type	Name	Attributes	Bytes	Dimensions	References
AP-0000000C0	CHAR	DOATE		6	(3)	33 68 89 90
91						
AP-000000100	CHAR	DTIME		4	(2)	33 69 92
2-000000000	R*4	ELEV		132	(33)	57 72D 83 141
2-00000012A	I*2	IDDATE		6	(3)	51 89= 98= 91=
102(3)						
2-000000130	CHAR	MONTHS		110	122(3)	70 77D 110
				36	(12)	
2-000000124	I*2	PDATE		6	(3)	45 119(3)= 122(3)
AP-000000140	R*4	POTENTIAL		240	(60)	33 56 147= 149=
AP-000000000	R*4	TEMPS		1200	(300)	33 55 135 146
2-000000004	R*4	TK		160	(40)	62 135= 138 146(2)
143						

LABELS

Address	Label	References
0-00000010C	55	119# 122
1-000000087	126'	119 120#
0-00000031C	999	119 156#

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	References
	FOR\$CLOSE	125
	FOR\$IDATE	100
	FOR\$OPEN	105 110
R*4	MTH\$ALDO	141
R*4	MTH\$EXP	142
	WAIT	100

KEY TO REFERENCE FLAGS	
=	- Value Modified
#	- Defining Reference
A	- Actual Argument, possibly modified
D	- Data Initialization
(n)	- Number of occurrences on line

COMMAND QUALIFIERS

FOR/LIS/EXTEND/CROSS/CHECK=ALL GET_THETA

/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)

/DEBUG=(NOSYMBOLS,TRACEBACK)

/STANDARD=(NOSYNTAX,NOSOURCE FORM)

/SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP,NODICTIONARY,SINGLE)

/WARNINGS=(GENERAL,NODECLARATIONS,NOULTRIX)

/CONTINUATIONS=19 /CROSS REFERENCE /NOD LINES /EXTEND SOURCE /F77

/NOG_FLOATING /I4 /NOMACHINE_CODE /OPTIMIZE /NOANALYSIS

COMPILATION STATISTICS

Run Time: 4.88 seconds

Elapsed Time: 5.88 seconds

Page Faults: 871

Dynamic Memory: 388 pages

```

0001  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002  C
0003  C   PLOT_NORTH.FOR
0004  C
0005  C   WRITTEN BY: K.W. BURK
0006  C
0007  C   PURPOSE: PLOTS NORTH SYMBOL GIVEN THE NEW
0008  C             X AND Y COORDINATES, AND THE SCALE.
0009  C             THIS WILL MAKE THE SYMBOL MUCH EASIER
0010  C             TO MOVE IF NEEDED.
0011  C
0012  C   ALGORITHM: DECLARE VARIABLES
0013  C             PLOT 'N' SYMBOL
0014  C             PLOT ARROW THROUGH 'N'
0015  C             END ROUTINE
0016  C
0017  C   VERSION : 1.0
0018  C
0019  C   DATE : FEBRUARY 1988
0020  C
0021  C   CALLED FROM: PLOT_AREA.FOR
0022  C
0023  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0024
0025  SUBROUTINE PLOT_NORTH(northx,northy,s)
0026
0027  IMPLICIT NONE
0028
0029  C   VARIABLE DECLARATIONS
0030  C-----
0031  C   TYPE      | VARIABLE | COMMENTS
0032  C-----
0033  C
0034
0035  REAL*4      a      ! New x coordinate
0036  REAL*4      b      ! New y coordinate
0037  REAL*4      northx ! X coordinate input
0038  REAL*4      northy ! Y coordinate input
0039  REAL*4      s      ! Scale of size, 1 = 1 inch.
0040
0041  a = northx / s
0042  b = northy / s
0043
0044  CALL plot (a,s,b,s, 3)          ! BEGIN PLOTTING 'N'
0045  CALL plot (a,s,(b+1.0)*s, 2)
0046  CALL plot ((a+.25)*s,(b+1.0)*s, 2)
0047  CALL plot ((a+.75)*s,(b+.333)*s, 2)
0048  CALL plot ((a+.75)*s,(b+1.0)*s, 2)
0049  CALL plot ((a+1.0)*s,(b+1.0)*s, 2)
0050  CALL plot ((a+1.0)*s,b,s, 2)
0051  CALL plot ((a+.75)*s,b,s, 2)
0052  CALL plot ((a+.25)*s,(b+.666)*s, 2)
0053  CALL plot ((a+.25)*s,b,s, 2)
0054  CALL plot (a,s,b,s, 2)
0055
0056  CALL plot ((a+.5)*s,(b+.6)*s, 3) ! BEGIN PLOTTING ARROW
0057  CALL plot ((a+.5)*s,(b+.23)*s, 2) ! THROUGH 'N'
0058  CALL plot ((a+.5)*s,(b+.77)*s, 3)
0059  CALL plot ((a+.5)*s,(b+1.5)*s, 2)
0060  CALL plot ((a+.25)*s,(b+1.25)*s, 2)
0061  CALL plot ((a+.5)*s,(b+1.25)*s, 2)
0062
0063  RETURN
0064  END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
# \$CODE	579	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	8	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	32	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	619	

ENTRY POINTS

Address	Type	Name	References
0-00000000		PLOT_NORTH	25#

VARIABLES

Address	Type	Name	Attributes	References
** 49	R*4	A		35 41= 44 45 46 47 48
			50 51 52 53 54 55 56 57 58	
** 49	R*4	B		36 42= 44 45 46 47 48
			50 51 52 53 54 55 56 57 58	
AP-000000040	R*4	NORTHX		25 37 41
AP-000000080	R*4	NORTHY		25 38 42
AP-0000000C0	R*4	S		25 39 41 42 44(2) 45(2) 46(2)
47(2)			48(2) 49(2) 50(2) 51(2) 52(2) 53(2) 54(2) 55(2)	
			57(2) 58(2) 59(2) 60(2) 61(2)	

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	References
51	PLOT	44 45 46 47 48 49 50
		52 53 54 55 56 57 58 59 60
		61

0001

KEY TO REFERENCE FLAGS	
=	- Value Modified
#	- Defining Reference
A	- Actual Argument, possibly modified
D	- Data Initialization
(n)	- Number of occurrences on line

COMMAND QUALIFIERS

FOR/LIS/EXTEND/CROSS/CHECK=ALL PLOT_NORTH

/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)

/DEBUG=(NOSYMBOLS,TRACEBACK)

/STANDARD=(NOSYNTAX,NOSOURCE FORM)

/SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP,NODICTIONARY,SINGLE)

/WARNINGS=(GENERAL,NODECLARATIONS,NOULTRIX)

/CONTINUATIONS=19 /CROSS REFERENCE /NOO LINES /EXTEND SOURCE /FTT

/NOO FLOATING /I4 /NOMACHINE_CODE /OPTIMIZE /NOANALYSIS

COMPILATION STATISTICS

Run Time: 2.94 seconds

Elapsed Time: 3.89 seconds

Page Faults: 548

Dynamic Memory: 342 pages

```

0001  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002  C
0003  C      PLOT_TEMPS.FOR
0004  C
0005  C      WRITTEN BY:  K.W. BURK
0006  C                  Q.L. ANDREWS
0007  C
0008  C      PURPOSE:  PLOT THE TOWER TEMPERATURES FOR AREA PLOTS.
0009  C                  IT WILL SCALE THE XAXIS AND PLOT A VERTICAL
0010  C                  LINE GRAPH.
0011  C
0012  C      ALGORITHM:  DECLARE VARIABLES
0013  C                  FIND MAX AND MIN TEMPS
0014  C                  PLOT DASHES ON X-AXIS
0015  C                  FIND TOWER TEMP COORDINATES
0016  C                  PLOT SYMBOLS ON GRAPH
0017  C                  CONNECT SYMBOLS WITH LINES
0018  C                  END ROUTINE
0019  C
0020  C      VERSION :  1.0
0021  C
0022  C      DATE :  FEBRUARY 1980
0023  C
0024  C      CALLED FROM:  PLOT_AREA.FOR
0025  C
0026  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0027
0028  SUBROUTINE PLOT_TEMPS(temp1,temp2,temp3)
0029
0030  IMPLICIT NONE
0031
0032  C
0033  C      VARIABLE DECLARATIONS
0034  C-----
0035  C      TYPE      |      VARIABLE      |      COMMENTS
0036  C-----
0037  C
0038
0039  INTEGER*2      min      ! Minimum tower temperature
0040  INTEGER*2      max      ! Maximum tower temperature
0041  INTEGER*2      mndiff   ! Difference between min and max
0042  INTEGER*2      ttemp(3) ! Three temps rounded and converted to integer
0043  INTEGER*2      i,j      ! Array and loop indices
0044  INTEGER*2      xnum     ! Number plotted along x-axis
0045
0046  REAL*4      temp1      ! First temp passed as input
0047  REAL*4      temp2      ! Second temp passed as input
0048  REAL*4      temp3      ! Third temp passed as input
0049  REAL*4      xscale     ! Scale along the x-axis
0050  REAL*4      xtemp      ! X coordinate of temp on graph
0051  REAL*4      ytemp      ! Y coordinate of temp on graph
0052  REAL*4      fnum       ! A real temporary variable
0053  REAL*4      x1,x2,x3   ! X coordinates of temperatures on graph
0054
0055
0056  max = -999
0057  min = 999
0058
0059  IF(temp1 .GT. 999.0) THEN
0060      ttemp(1) = 999      ! DO NOT ROUND UP ANY MISSING
0061      ttemp(2) = 999      ! DATA, LEAVE AS 999
0062      ttemp(3) = 999
0063  ELSE
0064      ttemp(1) = IIFIX(temp1 * 0.5) ! ROUND TEMPERATURES TO NEAREST
0065      ttemp(2) = IIFIX(temp2 * 0.5) ! WHOLE DEGREE, IF NOT MISSING

```

```

0066      ttemp(3) = IIFIX(temp3 + 0.5)
0067  ENDIF
0068
0069  DO i = 1,3
0070      IF (ttemp(i) .LT. min) min = ttemp(i)
0071      IF (ttemp(i) .GT. max) max = ttemp(i)
0072  ENDDO
0073
0074      min = min - 8          ! ADD DEGREES TO EACH END OF THE
0075      max = max + 8          ! GRAPH TO GIVE BETTER APPEARANCE.
0076
0077      mdiff = max - min      ! DIFFERENCE BETWEEN MAX AND MIN
0078
0079      xscale = 4.5 / (FLOATI(mdiff)) ! THIS WILL BE THE SPACE BETWEEN VALUES
0080                                     ! ON THE X-AXIS.
0081
0082      xtemp = 2.5            ! STARTING POINT FOR THE VALUES ON THE
0083                               ! X-AXIS
0084      xnum = min
0085
0086  DO i = 1,mdiff+1
0087
0088      IF( mdiff .GE. 10) THEN          ! IF THE DIFFERENCE
0089          IF( IMOD(i,2) .EQ. 1) THEN    ! IS TOO BIG, THEN
0090              CALL symbol(xtemp-.05,4.42,0.20,'|',0.0,1) ! ONLY PLOT EVERY-
0091              fnum = FLOATI(xnum)        ! OTHER VALUE ON
0092              CALL number(xtemp-.10,4.20,0.18,fnum,0.0,-1) ! THE X-AXIS.
0093          ENDIF
0094      ELSE
0095          CALL symbol(xtemp-.05,4.42,0.20,'|',0.0,1)
0096          fnum = FLOATI(xnum)
0097          CALL number(xtemp-.10,4.20,0.18,fnum,0.0,-1)
0098      ENDIF
0099
0100      DO j = 1,3
0101          IF( xnum .EQ. ttemp(j)) THEN ! FIND WHERE THE THREE TOWER
0102              IF( j .EQ. 1) THEN        ! TEMPERATURES BELONG ON THE
0103                  ytemp = 5.5            ! GRAPH BY COMPARING THEM
0104                  x1 = xtemp              ! TO THE TEMPS ALONG THE X-AXIS.
0105              ENDIF
0106              IF( j .EQ. 2) THEN
0107                  ytemp = 6.5
0108                  x2 = xtemp
0109              ENDIF
0110              IF( j .EQ. 3) THEN
0111                  ytemp = 7.5
0112                  x3 = xtemp
0113              ENDIF
0114          ENDIF
0115      ENDDO
0116      xtemp = xtemp + xscale            ! INCRUMENT X-AXIS
0117      xnum = xnum + 1
0118  ENDDO
0119
0120  CALL symbol (0.0,0.0,0.05, 11 ,0.0,-1)
0121  CALL symbol (x1,5.5,0.14, 5 ,0.0,-1)
0122  CALL symbol (0.0,0.0,0.05, 11 ,0.0,-1)
0123  CALL symbol (x2,6.5,0.14, 5 ,0.0,-1)
0124  CALL symbol (0.0,0.0,0.05, 11 ,0.0,-1)
0125  CALL symbol (x3,7.5,0.14, 5 ,0.0,-1)
0126
0127  CALL plot (x1,5.5, 3)                ! PLOT LINES.
0128  CALL plot (x2,6.5, 2)
0129  CALL plot (x3,7.5, 2)
0130

```



```

0131      RETURN
0132      END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	481	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	68	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	276	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	823	

ENTRY POINTS

Address	Type	Name	References
0-00000000		PLOT_TEMPS	280

VARIABLES

Address	Type	Name	Attributes	References
2-00000000	R+4	FNUM		52 91= 92A 96= 97A
**	I+2	I		43 69= 70(2) 71(2) 86= 89
**	I+2	J		43 100= 101 102 106 110
**	I+2	MAX		40 55= 71(2)= 75(2)= 77
**	I+2	MIN		39 57= 70(2)= 74(2)= 77 84
**	I+2	MMDIFF		41 77= 79 86 88
AP-000000040	R+4	TEMP1		28 46 59 64
AP-000000080	R+4	TEMP2		28 47 65
AP-0000000C0	R+4	TEMP3		28 46 60
2-0000000C	R+4	X1		53 104= 121A 127A
2-00000010	R+4	X2		53 100= 123A 128A
2-00000014	R+4	X3		53 112= 125A 129A
**	I+2	XNUM		44 84= 91 96 101 117(2)=
**	R+4	XSCALE		40 79= 110
**	R+4	XTEMP		50 82= 90 92 95 97 104
108				
			112	116(2)=
**	R+4	YTEMP		51 103= 107= 111=

ARRAYS

Address	Type	Name	Attributes	Bytes	Dimensions	References
2-00000000	I+2	TTEMP		6	(3)	42 66= 61= 62=
64=				65=	66=	70(2) 71(2) 101

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	References
NUMBER		92 97

	PLOT	127	128	129				
	SYMBOL	98	95	128	121	122	123	124
125								
0001								

```

+-----+
|               KEY TO REFERENCE FLAGS               |
|  =  - Value Modified                               |
|  #  - Defining Reference                           |
|  A  - Actual Argument, possibly modified           |
|  D  - Data Initialization                           |
|  (n) - Number of occurrences on line                |
+-----+

```

COMMAND QUALIFIERS

```

FOR/LIS/EXTEND/CROSS/CHECK=ALL PLOT_TEMPS

/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)
/DEBUG=(NOSYMBOLS,TRACEBACK)
/STANDARD=(NOSYNTAX,NOSOURCE FORM)
/SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP,NODICTIONARY,SINGLE)
/WARNINGS=(GENERAL,NODECLARATIONS,NOULTRIX)
/CONTINUATIONS=19 /CROSS REFERENCE /MOD LINES /EXTEND SOURCE /F77
/MOD FLOATING /I4 /NOMACHINE_CODE /OPTIMIZE /NOANALYSIS

```

COMPILATION STATISTICS

```

Run Time:      4.19 seconds
Elapsed Time:  6.63 seconds
Page Faults:   688
Dynamic Memory: 379 pages

```

```

0001  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002  C
0003  C   PLOT_VECTOR.FOR
0004  C
0005  C   WRITTEN BY : G. L. ANDREWS
0006  C               K. B. BURK
0007  C
0008  C   PURPOSE: THIS ROUTINE WILL PLOT ANY WIND VECTORS GIVEN
0009  C             THE NUMBER OF VECTORS, THE DIRECTION, THE SPEED, AND
0010  C             THE COORDINATES OF THE STATION.
0011  C
0012  C   ALGORITHM: DECLARE VARIABLES
0013  C                WIND AND SPEED INTO SEPERATE ARRAYS
0014  C                PLOT VECTORS
0015  C                END ROUTINE
0016  C
0017  C   VERSION : 1.0
0018  C
0019  C   DATE : FEBRUARY 1988
0020  C
0021  C   CALLED FROM : PLOT VAR.FOR
0022  C                 PLOT_AREA.FOR
0023  C
0024  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0025
0026      SUBROUTINE Plot_vector(nvars,dirspd,xdot,ydot)
0027
0028      IMPLICIT NONE
0029
0030  C
0031  C   VARIABLE DECLARATIONS
0032  C-----
0033  C   TYPE      |   VARIABLE   |   COMMENTS
0034  C-----
0035  C
0036      INTEGER*2  nvars      ! Number of variables
0037      INTEGER*2  i,j,k      ! Array and loop indices
0038      INTEGER*2  halfvars   ! Half the number of variables
0039
0040      REAL*4     dirspd(300) ! Directions and speeds
0041      REAL*4     xdot(60)    ! X coordinates input
0042      REAL*4     ydot(60)    ! Y coordinates input
0043      REAL*4     xtelea(60)  ! X coordinates of stations
0044      REAL*4     ytelea(60)  ! Y coordinates of stations
0045      REAL*4     px1,py1     ! Beginning of vector
0046      REAL*4     px2,py2     ! End of vector
0047      REAL*4     arrlen      ! Arrow length
0048      REAL*4     arrmin      ! Minimum length of arrow
0049      REAL*4     arrinc      ! Scale length of arrow
0050      REAL*4     spd(30)     ! Array of speeds
0051      REAL*4     direction(30) ! Array of converted directions
0052      REAL*4     dir(30)     ! Array of directions
0053      REAL*4     point       ! Direction in radians
0054      REAL*4     dtr         ! Conversion for degrees to radians
0055      REAL*4     adh1,adh2    ! Used to plot arrow heads
0056      REAL*4     bpx1,bpy1    ! Used to plot arrow heads
0057      REAL*4     bpx2,bpy2    ! Used to plot arrow heads
0058
0059      arrmin = 0.3
0060      arrinc = 0.1
0061      dtr = 0.0174532925
0062
0063
0064  C
0065  C   SEPERATE THE WIND DIRECTIONS FROM THE WIND SPEEDS.

```

```

0066 C      THE WIND DIRECTIONS ARE AT THE POSITIONS IN THE ARRAY WITH ODD INDECIES,
0067 C      THE WIND SPEEDS ARE AT THE POSITIONS IN THE ARRAY WITH EVEN INDECIES.
0068 C
0069
0070      i = 1
0071      DO j = 1,nvars-1,2                                ! ODD INDECIES.
0072          IF(dirapd(j) .GE. 361.0) THEN                    ! DO NOT ROUND
0073              dir(i) = 999                                ! A MISSING VALUE.
0074          ELSE
0075              dir(i) = FLOATI((IIFIX((dirapd(j)/10.0)+0.5))*10) ! ROUND DIRECTION TO
0076          ENDIF                                           ! NEAREST TEN DEGREES
0077          xtelea(i) = xdot(j)
0078          ytelea(i) = ydot(j)
0079          i = i + 1
0080      ENDDO
0081      i = 1
0082      DO j = 2,nvars,2                                    ! EVEN INDECIES.
0083          IF(dirapd(j) .EQ. 266.0) THEN
0084              spd(i) = 999
0085          ELSE
0086              spd(i) = dirapd(j)
0087          ENDIF
0088          i = i + 1
0089      ENDDO
0090
0091 C
0092 C      DRAW AN ARROW ORIGINATING AT EACH STATION LOCATION TO REPRESENT
0093 C      THE MAGNITUDE AND FLOW DIRECTION OF THE WINDS.
0094 C
0095
0096      halfvars = nvars / 2
0097
0098      DO 700 J = 1, halfvars
0099
0100          px1 = 0.0
0101          py1 = 0.0
0102          px2 = 0.0
0103          py2 = 0.0
0104          arrlen = 0.0
0105          point = 0.0
0106          adh1 = 0.0
0107          adh2 = 0.0
0108          bpx1 = 0.0
0109          bpy1 = 0.0
0110          bpx2 = 0.0
0111          bpy2 = 0.0
0112
0113          px1 = (xtelea(J) * 0.5)
0114          py1 = (ytelea(J) * 0.5)
0115
0116          CALL symbol (0.0,0.0,0.05, 11 , 0.0,-1)
0117
0118          CALL symbol (px1, py1, 0.14, 5, 0.0, -1)
0119
0120          CALL symbol (0.0,0.0,0.05, 11 , 0.0,-1)
0121
0122          IF ( j .EQ. 25) THEN                                ! CHECK FOR
0123              IF (spd(J) .LT. 0.51 .OR. spd(J) .GT. 99.0) GOTO 700 ! RMTN
0124          ELSE
0125              IF (spd(J) .LT. 0.51 .OR. spd(J) .GT. 50.0) GOTO 700
0126          ENDIF
0127
0128          arrlen = arrmin + (arrinc * (spd(J) / 5.0))
0129          direction(J) = (360.0 - (dir(J) - 90.0)) + 180.0
0130          IF (direction(J) .GT. 360.0) direction(J) = direction(J) - 360.0

```

```

0131      point = direction(J) * dtr
0132      px2 = COS(point) * arrlen + px1
0133      py2 = SIN(point) * arrlen + py1
0134
0135      CALL PLOT (px1, py1, 3)
0136      CALL PLOT (px2, py2, 2)
0137
0138      arrlen = arrlen / 4.0
0139      adh1 = direction(J) + 135.0
0140      IF (adh1 .GT. 360.0) adh1 = adh1 - 360.0
0141      adh2 = direction(J) - 135.0
0142      IF (adh2 .LT. 1.0) adh2 = adh2 + 360.0
0143
0144      bpx1 = COS(adh1 * dtr) * arrlen + px2
0145      bpy1 = SIN(adh1 * dtr) * arrlen + py2
0146      bpx2 = COS(adh2 * dtr) * arrlen + px2
0147      bpy2 = SIN(adh2 * dtr) * arrlen + py2
0148
0149      CALL PLOT (bpx1, bpy1, 2)
0150      CALL PLOT (bpx2, bpy2, 2)
0151      CALL PLOT (px2, py2, 2)
0152  700  CONTINUE
0153
0154      RETURN
0155      END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	819	PIC COM REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	32	PIC COM REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	1050	PIC COM REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	1907	

ENTRY POINTS

Address	Type	Name	References
0-00000000		PLOT_VECTOR	264

VARIABLES

Address	Type	Name	Attributes	References
**	R+4	ADH1		55 100= 139= 140(3)= 144 145
**	R+4	ADH2		55 107= 141= 142(3)= 146 147
**	R+4	ARRINC		49 50= 128
**	R+4	ARRLEN		47 164= 128= 132 133 138(2)= 144
145			146 147	
**	R+4	ARRMIN		48 59= 120
2-0000035C	R+4	BPX1		56 108= 144= 149A
2-00000364	R+4	BPX2		57 110= 146= 150A
2-00000360	R+4	BPY1		58 109= 145= 149A
2-00000300	R+4	BPY2		57 111= 147= 150A
**	R+4	DTR		54 61= 131 144 145 146 147

```

      **      I+2 HALFVARS      38      98=      98
      **      I+2 I            37      78=      73      75      77      78
79(2)=      81=
      **      I+2 J            84      86      88(2)=      71=      72      75      77      78      82=
      83
      88      98=      113      114      122      123(2)      125(2)      128
      129(2)      138(3)      131      139      141
      **      I+2 K
AP-000000040 I+2 NVARS      37
      26      36      71      82      96
      **      R+4 POINT      63      105=      131=      132      133
2-00000034C R+4 PX1      46      106=      113=      118A      132      135A
2-000000354 R+4 PX2      46      102=      132=      136A      144      146      151A
      2-000000356 R+4 PY1      46      101=      114=      118A      133      135A
      2-000000358 R+4 PY2      46      103=      133=      136A      145      147      151A

```

ARRAYS

Address	Type	Name	Attributes	Bytes	Dimensions	References
2-000000200	R+4	DIR		128	(30)	52 73= 75= 129
2-000000258 139	R+4	DIRECTION		128	(30)	51 129= 138(3)= 131
AP-000000000 83	R+4	DIRSPD		141 1280	(300)	26 40 72 75
2-0000001E0 125(2)	R+4	SPD		88 128	(30)	50 84= 86= 123(2)
AP-0000000C0	R+4	XDOT		128 246	(60)	26 41 77
2-000000000	R+4	XTELEM		246	(60)	43 77= 113
AP-000000100	R+4	YDOT		246	(60)	26 42 78
2-000000F0	R+4	YTELEM		246	(60)	44 78= 114

LABELS

Address	Label	References
0-00000032C	700	98 123 125 152#

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	References
R+4	MTH\$COS	132 144 146
R+4	MTH\$SIN	133 145 147
	PLOT	136 136 149 150 151
	SYMBOL	116 118 120

KEY TO REFERENCE FLAGS	
=	- Value Modified
#	- Defining Reference
A	- Actual Argument, possibly modified
D	- Data Initialization
(n)	- Number of occurrences on line

COMMAND QUALIFIERS

FOR/LIS/EXTEND/CROSS/CHECK=ALL PLOT_VECTOR

/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)

/DEBUG=(NOSYMBOLS,TRACEBACK)

/STANDARD=(NOSYNTAX,NOSOURCE FORM)

/SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP,NODICTIONARY,SINGLE)

/WARNINGS=(GENERAL,NODECLARATIONS,NOLTRIX)

/CONTINUATIONS=19 /CROSS REFERENCE /NOD LINES /EXTEND SOURCE /FTT

/NOG_FLOATING /I4 /NOMACHINE_CODE /OPTIMIZE /NOANALYSIS

COMPILATION STATISTICS

Run Time: 5.19 seconds

Elapsed Time: 6.83 seconds

Page Faults: 627

Dynamic Memory: 487 pages

```

0001 C*****
0002 C
0003 C   SUB-PROGRAM : GRAPHICS.FOR
0004 C
0005 C   PROGRAMMER : GREGG ANDREWS
0006 C
0007 C   PURPOSE : TO READ IN DIGITIZED INFORMATION FOR THE HANFORD
0008 C             MAP BACKGROUND. THIS INFORMATION INCLUDES THE
0009 C             ROADS, RIVERS, AND HANFORD BOUNDARY. THE INFORM-
0010 C             ATION IS THEN PASSED TO THE PLOTTING SUBROUTINES
0011 C             PLOTZ, WNDPLT, AND WNDFLD IN A COMMON BLOCK.
0012 C
0013 C   DATE : 3/13/87
0014 C
0015 C   VERSION : 1.0
0016 C
0017 C*****
0018 C
0019 C   SUBROUTINE PLT_GRAPHICS(READ_map)
0020 C
0021 C
0022 C   INTEGER GRAPHICS RECORDS
0023 C   INTEGER COLOR ! Color of pen
0024 C   INTEGER ACTION ! Action of pen; up or down
0025 C   LOGICAL READ_map ! Flag for reading file with graphics data
0026 C
0027 C   PARAMETER (GRAPHICS_RECORDS = 1321) ! Number of digitized points
0028 C
0029 C   DIMENSION X_CORD(GRAPHICS_RECORDS), Y_CORD(GRAPHICS_RECORDS),
0030 C             COLOR(GRAPHICS_RECORDS), ACTION(GRAPHICS_RECORDS)
0031 C
0032 C   IF(.NOT. READ_map) THEN
0033 C
0034 C       OPEN(UNIT=18, FILE='[andrews.REVIEW]MESOI GRAPHICS.DAT',
0035 C            STATUS='OLD', READONLY, SHARED)
0036 C
0037 C       DO I=1, GRAPHICS_RECORDS
0038 C           READ(18,*) X_CORD(I), Y_CORD(I),
0039 C                     ACTION(I), COLOR(I)
0040 C           CALL NEWPEN(COLOR(I))
0041 C           CALL PLOT(X_CORD(I), Y_CORD(I), ACTION(I))
0042 C       ENDDO
0043 C       CLOSE UNIT=18
0044 C       READ_map = .TRUE.
0045 C   ENDIF
0046 C
0047 C   DO I=1, GRAPHICS_RECORDS
0048 C       CALL NEWPEN(COLOR(I))
0049 C       CALL PLOT(X_CORD(I), Y_CORD(I), ACTION(I))
0050 C   ENDDO
0051 C
0052 C   RETURN
0053 C   END
0054 C

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	341	PIC COM REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	85	PIC COM REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	21288	PIC COM REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	21578	

ENTRY POINTS

Address	Type	Name	References
0-00000000		PLT_GRAPHICS	19‡

VARIABLES

Address	Type	Name	Attributes	References
**	R+4	CLOSEUNIT		43=
**	I+4	I		37= 38(4) 40 41(3) 47=
40			49(3)	
AP-000000040	L+4	READ_MAP		10 26 32 44=

ARRAYS

Address	Type	Name	Attributes	Bytes	Dimensions	References
2-000014A4	I+4	ACTION		5284	(1321)	24 29 38= 41A
49A						
2-00000000	I+4	COLOR		5284	(1321)	23 29 38= 40A
40A						
2-00002940	R+4	X_CORD		5284	(1321)	29 38= 41A 49A
2-00003DEC	R+4	Y_CORD		5284	(1321)	29 38= 41A 49A

PARAMETER CONSTANTS

Type	Name	References
I+4	GRAPHICS_RECORDS	22 27‡ 29(4) 37 47

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	References
	FOR\$OPEN	34
	NEWPEN	40 48
	PLOT	41 49

KEY TO REFERENCE FLAGS	
=	- Value Modified
‡	- Defining Reference
A	- Actual Argument, possibly modified
D	- Data Initialization
(n)	- Number of occurrences on line

COMMAND QUALIFIERS

FOR/LIS/EXTEND/CROSS/CHECK=ALL PLT_GRAPHICS

/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)

/DEBUG=(NOSYMBOLS,TRACEBACK)

/STANDARD=(NOSYNTAX,NOSOURCE FORM)
/SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP,MODICTIONARY,SINGLE)
/WARNINGS=(GENERAL,MODECLARATIONS,MOLTRIX)
/CONTINUATIONS=19 /CROSS REFERENCE /NOD LINES /EXTEND SOURCE /F77
/NOG_FLOATING /I4 /NOMACHINE_CODE /OPTIMIZE /NOANALYSIS

COMPILATION STATISTICS

Run Time: 2.84 seconds
Elapsed Time: 2.84 seconds
Page Faults: 667
Dynamic Memory: 334 pages

DISTRIBUTION

No. of
Copies

OFFSITE

- 2 DOE/Office of Scientific and
Technical Information

ONSITE

DOE Richland Operations Office

J. J. Sutey

28 Pacific Northwest Laboratory

G. L. Andrews (5)
K. A. Borgeson
J. W. Buck
J. W. Falco
O. P. Gifford
M. J. Graham
J. M. Hales
P. C. Hays
D. J. Hoitink
D. J. Newland
W. T. Pennell
P. J. Perrault
E. R. Preston
J. V. Ramsdell
G. N. Reinecke
M. N. Schwartz
J. A. Stottlemire
L. L. Wendell
R. E. Wildung
Publishing Coordination
Technical Report Files (5)