5D

Hanford Meteorological Station Computer Codes

Volume 9 - The Quality Assurance Computer Codes

K. W. Burk
G. L. Andrews

February 1989

**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.

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	A10
226-250	A11
251-275	A12
276-300	A13

3 3679 00057 1861

PNL-6279 Vol. 9
UC-605

HANFORD METEOROLOGICAL STATION
COMPUTER CODES

Volume 9 - The Quality Assurance Computer Codes

K. W. Burk
G. L. Andrews

February 1989

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 ninth in a series of volumes that describe computer programs routinely used at the Hanford Meteorological Station (HMS). Each volume consists of a program overview, user's guide, description of input, a detailed program description and flow chart, and a source listing of the code.

The HMS, operated by the Pacific Northwest Laboratory (PNL), provides meteorological and climatological services to the U.S. Department of Energy, Richland Operations Office, and its contractors. The HMS measures, records, and archives meteorological data collected on the Hanford Site hourly, 24 hours each day, throughout the year.

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 are due to Dana Hoitink, the HMS project manager, and to Ora Gifford for his helpful suggestions and ideas.

CONTENTS

PREFACE	iii
ACKNOWLEDGMENTS	v
INTRODUCTION	I
LOGGING ON TO THE HMS COMPUTER SYSTEM	3
THE DATE_TIME_CHECK COMPUTER CODE	5
PROGRAM OVERVIEW	5
USER'S GUIDE	5
DESCRIPTION OF INPUT	6
DESCRIPTION OF OUTPUT	7
LIMITATIONS TO THE PROGRAM, DATA BASE, AND COMPUTER SYSTEM	7
THE TOWERQA COMPUTER CODE	9
PROGRAM OVERVIEW	9
USER'S GUIDE	9
DESCRIPTION OF INPUT	10
DESCRIPTION OF OUTPUT	12
LIMITATIONS TO THE PROGRAM, DATA BASE, AND COMPUTER SYSTEM	15
THE TELEMQA COMPUTER CODE	17
PROGRAM OVERVIEW	17
USER'S GUIDE	17
DESCRIPTION OF INPUT	18
DESCRIPTION OF OUTPUT	20
LIMITATIONS TO THE PROGRAM, DATA BASE, AND COMPUTER SYSTEM	21
THE SFCQA COMPUTER CODE	23
PROGRAM OVERVIEW	23
USER'S GUIDE	23

DESCRIPTION OF INPUT	24
DESCRIPTION OF OUTPUT	25
LIMITATIONS TO THE PROGRAM, DATA BASE, AND COMPUTER SYSTEM .	30
REFERENCES	31
APPENDIX A - TROUBLESHOOTING THE TERMINAL AND COMPUTER SYSTEM PROBLEMS	A.1
APPENDIX B - DTC PROGRAM DESCRIPTION, FLOW CHART, AND CODE LISTINGS	B.1
APPENDIX C - TOWERQA PROGRAM DESCRIPTION, FLOW CHART, AND CODE LISTINGS	C.1
APPENDIX D - TELEMQA PROGRAM DESCRIPTION, FLOW CHART, AND CODE LISTINGS	D.1
APPENDIX E - SFCQA PROGRAM DESCRIPTION, FLOW CHART, AND CODE LISTINGS	E.1

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 mile east of the 200 West Area and is operated by PNL for the U.S. Department of Energy. Meteorological data are collected from various sensors and equipment located on and off the Hanford Site (Figure 1). These data are stored in data bases on the Digital Equipment Corporation (DEC) VAX 11/750 at the HMS (hereafter referred to as the HMS computer). Files from those data bases are routinely transferred to the Emergency Management System (EMS) computer at the Unified Dose Assessment Center (UDAC).

To ensure the quality and integrity of the HMS data, a set of Quality Assurance (QA) computer codes has been written. The codes will be routinely used by the HMS system manager or the data base custodian. The QA codes provide detailed output files that will be used in correcting erroneous data.

The following sections in this volume describe the implementation and operation of the QA computer codes. The appendices contain detailed descriptions, flow charts, and source code listings of each computer code.

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

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

LOGGING ON TO THE HMS COMPUTER SYSTEM

The HMS computer system is a DEC VAX 11/750 and uses VMS as the operating system. Being logged on to the HMS VAX is necessary to be able to run any of the computer codes described in this document.

To log on to the HMS computer system from the terminals at HMS, 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 into 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 any of the computer codes discussed in the following sections of this document.

The source code for all the QA code routines is located in the [FCST.QA_PROGRAMS] sub-directory. The data files that the programs access are located in the [FCST.DAT] sub-directory.

THE DATE TIME CHECK COMPUTER CODE

DATE TIME CHECK (DTC) was developed to check for invalid dates and times for data that are gathered and archived at HMS. DTC will perform a date-time sequence check on surface observation data, tower data (400 ft and 200 ft), and on telemetry data (winds, temperatures, and precipitation).

PROGRAM OVERVIEW

DTC accesses one input file, specified by the user, and produces one output file. The input file contains the type of meteorological data specified by the user. The output file contains a listing of any date-time sequence errors that occurred during processing. The naming convention used for the output file is to add the extension "_DATE.ERR" onto the type of data file processed (e.g., TOWER, TELEM, SFC).

The program begins by clearing the screen and then displaying a message informing the user what the program is and what it does. At the same time, it prompts the user for a selection from the input file menu. The desired data file is then processed, and an output file is created only if there is an error found. The program then asks the user if another file needs to be processed. If so, execution starts again by displaying the menu. If not, execution is complete.

USER'S GUIDE FOR THE DTC COMPUTER CODE

This section describes how to invoke the DTC computer code on the HMS computer system and how to respond to the various prompts associated with the code. Basically, there are five steps to executing the DTC computer code, replying to prompts for information, and terminating the program. These steps are listed below, and each step is discussed in detail in the following paragraphs.

- Log in to the HMS computer (described on page 3).
- Type RUN DTC and depress RETURN key.
- Supply input data information.
- Answer "yes" or "no" to the "would you like to process another file" prompt.
- Terminate program.

The DTC code is invoked by typing RUN DTC when at the '\$' prompt and depressing the RETURN key. When this is done, the screen clears, and a menu appears on the screen. The menu offers seven choices: surface observation data, 400-ft tower data, 200-ft tower data, telemetry wind data, telemetry temperature data, telemetry precipitation data, and exit the program. You now enter your choice and depress the RETURN key. The correct responses from the menu are 1 through 7. An incorrect response will receive an error

message, and you will get another chance to enter a correct response. After a correct response is entered, the program begins processing.

When processing begins, DTC will respond with a message and prompt you for the input file name. Follow the directions in the prompt and type in the desired file name, then depress the RETURN key. The file name you enter will then be checked for validity. If you enter an incorrect file name, an error message will appear and let you know how to correct your mistake. You will then be given another chance to enter the file name.

After an acceptable file name has been entered, you are then required to enter a starting date, or the date at which the input file begins. If the desired file is a month of data, enter the first day of the month, in the requested format. If you are unsure of the starting date, check the input file for the starting date. If the date you entered is incorrect, an error will appear in the output file informing you an error occurred before record number one.

After the file name and date have been successfully entered, the program begins checking the dates and times in the desired file. If an error is found during the checking, a message will be displayed on the screen telling you what the output file will be called. Otherwise, a message will be displayed informing you that no errors were encountered.

When the checking is complete, the program will ask if you wish to have another file checked. Answer by entering a 'y' for yes, or an 'n' for no. If your answer is yes, the execution of the program will continue by displaying the file-type menu on the screen. If your answer is no, the program terminates execution and returns control to the operating system. At this point the '\$' prompt is visible.

DESCRIPTION OF INPUT

DTC is able to access one of three types of files at one time. These three types are the surface observations, tower data, and telemetry data. These data files are located in the [FCST.DAT] subdirectory on the HMS system.

When DTC accesses these files, the only part of the data that is read by the program is the date. In both the surface observations and the tower data, the date is in the format; year, month, day, hour. Each field is 2 characters long (e.g. YYMMDDHH). Therefore, the last two digits of the year are YY (00-99), the month is MM (1-12), the day is DD (1-31), and the hour is HH (1-24).

Telemetry data has a date format that does not follow the other two files. The format is YYDDDHH, where YY is the last two digits in the year (00-99), DDD is the Julian day (1-366), and HH is the hour (1-24). The Julian day is the total number of days that have gone by since January 1 (e.g., the Julian day for February 1 is 32).

DESCRIPTION OF OUTPUT

A formatted output file with error messages describing where possible date-time sequence errors occurred is created by DTC. The name of this file depends on the name of the input file. The output filename will be constructed using the input file name and adding the extension "_DATE.ERR".

There are three types of output files matching the three types of input files. Tower and surface observation data have the same format. The first line identifies the kind of data processed and the name of the input file. Following that are lines of messages, one for each error encountered. The message states that there is an "Error in Date at or Before Record", and the record number follows. Then the date of the record is given.

This is also true with telemetry data. However, the format of the error log for telemetry data is different. The same message is printed with slight variation. The message will be "Error in date at or before JULIAN day" followed by the Julian day in that record and the record number.

Listed below is an example of a date-time sequence error that may occur.

RECORD NUMBER	YEAR	MONTH	DAY	HOURL
123	88	10	04	13
124	88	10	05	14
125	88	10	04	15

ERROR MESSAGES

```
ERROR AT OR BEFORE RECORD 124    DATE: 88 10 05 14
ERROR AT OR BEFORE RECORD 125    DATE: 88 10 04 15
```

(As you can see in this example, one error in record 124 will produce two error messages in the output file because the day value does not match.)

At the end of the error message files, there will be a record that gives the total number of records in the file. For tower and surface observation data, there is one record per hour; for telemetry, there are three records per hour.

LIMITATIONS TO THE PROGRAM, DATA BASE, AND COMPUTER SYSTEM

The DTC computer program does have some limitations. The program is designed to check only the three types of data mentioned earlier: tower data (400-ft and 200-ft), surface observation data, and telemetry data (winds, temperatures, and precipitation). An attempt to use another type of data file will result in the program being terminated by a run-time error.

In addition, the file name specified as input must exist and reside in the correct subdirectory on the HMS computer. To prevent any errors that will terminate the program, always check to make sure that the specified input file exists before trying to execute this program.

It is also very important to follow all the directions used in the prompts. When responding to a prompt, be sure to supply the information in the specified format. Failure to do so will create an error that will end program execution.

THE TOWERQA COMPUTER CODE

PROGRAM OVERVIEW

TOWERQA was developed to provide a quality assurance check for tower data gathered at the HMS. Data from the HMS 400-ft tower and data collected at the three 200-ft towers (at the 100N, 300, and 400 Areas) are included. The code is to be used by the data base administrator and/or custodian. The program is designed to read a user-specified tower data file and check for erroneous data. A complete listing of the input data is echoed to an output file along with any error messages inserted where the error occurred in the data.

USER'S GUIDE

This section describes how to invoke the TOWERQA computer code on the HMS computer system and how to respond to the various prompts associated with the code. Basically, there are five steps to executing the TOWERQA computer code, replying to prompts for information, and terminating the program. These steps are listed below, and each step is discussed in detail in the following paragraphs.

- Log in to the HMS computer (described on page 3).
- Type RUN TOWERQA and depress RETURN key.
- Supply input data information.
- Answer "y" or "n" to the "would you like to process another file" prompt.
- Terminate program.

To invoke the program TOWERQA, the user must be logged on to the HMS computer system and at the '\$' prompt. Then the user must type RUN TOWERQA and depress the RETURN key. When TOWERQA is invoked, a message is displayed to the screen briefly describing what the program does.

The first prompt the user must respond to is which type of tower data to process. A menu is given, and the user must respond by entering the number of the choice desired. There will be three choices. Choice 1 allows the user to process a 400-ft tower data file. Choice 2 will enable the code to process a 200-ft tower data file. Choice 3 is to exit the program. After a valid choice is entered, the program will proceed.

The user is then asked to enter the input file name. The user must type in the file name and depress the RETURN key. If there is an error in the file name, a message is displayed with information on how to correct the problem. Otherwise the program proceeds and displays the name of the output file that will be created as the data is processed. The code will then

display the date as each 24-hour period is processed, to update the user on processing progress.

After the entire file has been processed, the user must respond to a message asking if another file needs to be processed. If so, type a 'y' and depress the RETURN, and the program will continue execution by prompting you for the next input file name. If the user does not wish to process another file, type an 'n' and depress RETURN. This will return control to the main program. A prompt will appear that asks if the user would like to go back to the main menu. If so, type a 'y' and depress RETURN. If not, type an 'n' and execution will terminate, returning you to the system prompt '\$'.

DESCRIPTION OF INPUT FOR 400-FT TOWER DATA

When the user chooses to process 400-ft tower data, TOWERQA will read only the format of data found in 400-ft tower data files. Also, the program will only attempt to open and read a file that has a name beginning with 'TOWER 400'. Each record in the input file contains 21 fields that are divided into 5 groups. A description of the data groups and column numbers is given below.

1. DATE - The date format is YYMMDDHH (columns 1-8): where YY is the last two digits of the year (e.g. 88 for 1988), MM is the numerical equivalent of the month (e.g. 08 for August), DD is the day of the month, and HH is the hour of the day (1 to 24). For example, 88080101 represents August 1, 1988, 1:00 a.m.
2. SOIL TEMPERATURES - There are three temperatures in this order: -0.5 in. (columns 10-13), -15.0 in. (columns 14-17), and -36.0 in. (columns 18-21). They are recorded every six hours. All three are integer fields of length 4. A decimal point is assumed between the third and fourth positions, but is not physically in the data. Therefore this gives tenths of degrees. If data are missing, the field will contain 9999, or 0 if data is unobtainable.
3. TOWER TEMPERATURES - There are 8 temperatures arranged in this order: 3 ft (columns 22-25), 30 ft (columns 26-29), 50 ft (columns 30-33), 100 ft (columns 34-37), 200 ft (columns 38-41), 250 ft (columns 42-45), 300 ft (columns 46-49), and 400 ft (columns 50-53). They are recorded every hour. All the fields are integer and are 4 units long. A decimal is assumed between the third and fourth positions, but is not physically in the data. This will then give assumed tenths of degrees. If data are missing, the field will contain 9999.
4. WIND DATA - There are 6 wind speeds and 3 wind directions, and they are arranged in this order (columns 55-72): 7-ft wind speed, 50-ft wind direction, 50-ft wind speed, 100-ft wind speed, 200-ft wind direction, 200-ft wind speed, 300-ft wind speed, 400-ft wind direction, and 400-ft wind speed. They are recorded every hour.

Each field is integer of length 2. Wind speeds are in mph, and the entry is 99 if wind speed is missing. Wind directions are in tens of degrees (e.g. 18 indicates wind from 180 degrees). Missing directions are coded 99. The code 37, or 370 degrees, indicates a light and variable condition.

5. **PEAK GUST DATA** - This data contains 3 fields of information on the peak gust for the hour. The fields are arranged in this order: the peak gust direction (columns 73-74), peak gust speed (columns 75-76), and the time code indicating the approximate time of the gust (column 77). They are recorded every hour. The direction is in tens of degrees, and the wind speed is in mph. The time code is given the value 0 through 9, dividing the hour into 10 parts (e.g., 0 indicates gust occurred in interval 0 to 5 minutes, 1 indicates minutes 6 to 11 etc.) In order to have a peak gust, the wind must reach a speed of at least 15 mph; otherwise, the three fields will have all zeros.

A more detailed description of the tower data fields is given by Buck and Andrews (1987).

DESCRIPTION OF INPUT FOR 200-FT TOWER DATA

When the user chooses to process 200-ft tower data, TOWERQA will read only the format of data found in 200-ft tower data files. Also, the program will only attempt to open and read a file that has a name beginning with 'TOWER_200_', followed by either 100N, 300A, or 400A. Each record in the input file contains 19 fields that are divided into four groups. A description of the data groups and the column numbers is given below.

1. **DATE** - The date format is YYMMDDHH (columns 1-8): where YY is the last two digits of the year (e.g., 88 for 1988), MM is the numerical equivalent of the month (e.g., 10 for October), DD is the day of the month, and HH is the hour of the day (1 to 24). For example, 88103101 represents October 31, 1988, 1:00 a.m.
2. **TEMPERATURES** - There are six temperatures in this order: 2-m dew point (columns 10-13), 2-m dry bulb (columns 34-37), 10-m dry bulb (columns 38-41), 60-m dry bulb (columns 42-45), maximum temperature (columns 62-65), and minimum temperature (columns 66 to 69). All these are integer fields of length 4. A decimal point is assumed between the third and fourth positions, giving tenths of degrees. Missing data is recoded 9999.
3. **STANDARD DEVIATIONS** - There are five values in this order: delta temperature (columns 14-17), 10-m sigma theta (columns 18-21), 10-m sigma speed (columns 22-25), 60-m sigma theta (columns 26-29), 60-m sigma speed (columns 30-33). All these are integer fields of length 4. A decimal point is assumed between the third and fourth positions of the delta temperature, giving tenths of degrees. The sigma values are in either whole degrees or whole mph.

4. WIND DATA - There are seven values in this order (columns 46-61): 10-m direction, 10-m speed, 25-m direction, 25-m speed, 60-m direction, 60-m speed, 10-m maximum speed. Each field is an integer of length 2. Wind speeds are in miles per hour, and wind directions are in tens of degrees. An entry of 99 indicates missing data.

DESCRIPTION OF OUTPUT FOR 400-FT TOWER DATA

TOWERQA creates a formatted output file to be used for locating and editing any errors in the data file. The name of the output file is created within the main program. The format of the name is 'TOWER_400', followed by a three-character representation of a month. The extension '_QA.ERR' is then used to identify it as a quality assurance error log.

At the top of the file is a line that contains the name of the input file. The following lines will be the actual input data, with any error message inserted between the data records. The error messages contain a brief description of the error, the date of the erroneous record, and the erroneous data itself.

A list of all possible error messages, along with a description of the problem causing the error and the valid range for each, is given below.

Error messages in ".ERR" file:

message: SOIL TEMP. ERROR (-0.5 IN.)

explanation: Soil temperature at -0.5 in. is out of range. (150.0, -30.0) degrees F.

message: SOIL TEMP. ERROR (-15.0 IN.)

explanation: Soil temperature at -15.0 in. is out of range. (93.0, 16.0) degrees F.

message: SOIL TEMP. ERROR (-36.0 IN.)

explanation: Soil temperature at -36.0 in. is out of range. (86.0, 32.0) degrees F.

message: LARGE SOIL TEMP. LEVEL DIFFER.

explanation: The difference in soil temperatures at -15.0 in. and -36.0 in. is greater than 20.0 degrees F. Neither value can be equal to 0 or 9999.

message: LARGE SOIL TEMP HOURLY DIFF

explanation: The difference between current soil temperature and previous soil temperature is larger than 20.0 degrees F for -0.5 in., or larger than 5.0 degrees F for -15.0 in. and -36.0 in.

message: TEMPERATURE OUT OF RANGE

explanation: One of the tower temperatures is not in the range defined by chosen monthly extreme values for HMS. These monthly extreme ranges, in degrees F, are shown below.

JAN. :	(62.0, -13.0)
FEB. :	(62.0, -13.0)
MAR. :	(73.0, 16.0)
APR. :	(85.0, 31.0)
MAY :	(94.0, 38.0)
JUN. :	(100.0, 43.0)
JUL. :	(105.0, 49.0)
AUG. :	(103.0, 50.0)
SEP. :	(96.0, 35.0)
OCT. :	(80.0, 16.0)
NOV. :	(65.0, -3.0)
DEC. :	(59.0, -17.0)

Note: These temperatures are subject to change. These current values are correct as of October 1988.

message: LARGE AIR TEMP CHANGE IN HOUR

explanation: One of the tower temperatures has changed more than 10.0 degrees F in one hour.

message: LARGE TEMP. LEVEL DIFF.

explanation: Two adjacent tower temperatures in the same hour have a difference of greater than 10.0 degrees F.

message: WIND SPEED OUT OF RANGE

explanation: One of the six tower wind speeds is either greater than 50 mph, or less than 0 mph.

message: LARGE HOURLY WIND SPEED DIFF

explanation: One or more of the tower wind speed has changed more than 10 mph since the last hour.

message: LARGE WIND SPEED LEVEL DIFF.

explanation: Two adjacent wind speeds have a difference greater than 10 mph.

message: WIND DIRECTION OUT OF RANGE

explanation: One or more of the three tower wind directions is greater than 370 or less than 0.

message: WIND DIRECTION/SPEED CONFLICT

explanation: One or more of the three wind direction-speed pairs for the tower has one or more of the following problems :

1. a wind direction = 0, and a non-zero wind speed.
2. a wind speed = 0, and a non-zero wind direction.

3. a wind direction = 370, and a wind speed of more than 5 mph.

message: LARGE WIND DIR. LEVEL DIFF.

explanation: Two adjacent wind directions have a difference of more than 40 degrees.

message: PEAK GUST DIRECTION ERROR

explanation: Wind direction for the peak gust is greater than 360, or less than 0.

message: PEAK GUST SPEED ERROR

explanation: Wind speed for the peak gust is greater than 60 mph, or less than 15 mph.

message: PEAK GUST TIME ERROR

explanation: Code for time of the peak gust is greater than 9 or less than.

DESCRIPTION OF OUTPUT FOR 200-FT DATA

TOWERQA creates a formatted output file to be used for locating and editing any errors in the data file. The name of the output file is created within the main program. The format of the name is 'TOWER_200_', followed by 100N, 300A, or 400A, and a three-character representation of a month. The extension '_QA.ERR' is then used to identify it as a quality assurance error log.

At the top of the file is a line that contains the name of the input file. The following lines will be the actual input data, with any error message inserted between the data records. The error messages contain a brief description of the error, the date of the erroneous record, and the erroneous data itself.

A list of all possible error messages, along with a description of the problem causing the error, and the valid range for each, is located below.

Error messages in ".ERR" file:

message: DEW POINT TEMP. OUT OF RANGE

explanation: The dew point temperature is not within the valid range. These monthly extreme ranges, in degrees F, are shown below.

JAN. :	(	62.0,	-13.0	)
FEB. :	(	62.0,	-13.0	)
MAR. :	(	73.0,	16.0	)
APR. :	(	85.0,	31.0	)
MAY :	(	94.0,	38.0	)
JUN. :	(	100.0,	43.0	)

JUL. : (105.0, 49.0)
 AUG. : (103.0, 50.0)
 SEP. : (96.0, 35.0)
 OCT. : (80.0, 16.0)
 NOV. : (65.0, - 3.0)
 DEC. : (59.0, -17.0)

message: TEMP OUT OF RANGE. 2 METER

explanation: The 2-m temperature is not within the valid range shown above.

message: TEMP OUT OF RANGE. 10 METER

explanation: The 10-meter temperature is not within the valid range shown above.

message: TEMP OUT OF RANGE. 60 METER

explanation: The 60-meter temperature is not within the valid range shown above.

message: MAXIMUM TEMP OUT OF RANGE

explanation: The maximum temperature for the hour is not within the range shown above.

message: WIND DIRECTION OUT OF RANGE

explanation: The wind direction given is not within the range 0 to 360.

message: WIND SPEED OUT OF RANGE

explanation: The wind speed given is not within the range 0 to 60.

message: MAXIMUM WIND SPEED OUT OF RANGE

explanation: The maximum wind speed is not within the range 1 to 75.

message: DELTA TEMP. OUT OF RANGE

explanation: The delta temperature is not within the range -10.0 to 15.0.

message: 10 METER SIGMA THETA OUT OF RANGE

explanation: The 10-meter sigma theta is not within the range 60 to 0.

message: 10 METER SIGMA SPEED OUT OF RANGE

explanation: The 10-meter sigma speed is not within the range 10 to 0.

message: 60 METER SIGMA THETA OUT OF RANGE

explanation: The 60-meter sigma theta is not within the range 60 to 0.

message: 60 METER SIGMA SPEED OUT OF RANGE

explanation: The 60-meter sigma speed is not within the range 10 to 0.

LIMITATIONS TO THE PROGRAM, DATABASE, AND COMPUTER SYSTEM

The TOWERQA computer program does have some limitations. The program is designed to check only the tower data (200 ft and 400 ft). An attempt to use another type of data file will result in the program being terminated by a run-time error.

In addition, the file name specified as input must exist and reside in the correct subdirectory on the HMS computer. To prevent any errors that will terminate the program, always check to make sure that the specified input file exists before trying to execute this program.

It is also very important to follow all the directions used in the prompts. When responding to a prompt, be sure to supply the information in the specified format. Failure to do so will create an error that will end program execution.

THE TELEMQA COMPUTER CODE

PROGRAM OVERVIEW

TELEMQA was developed to provide a quality assurance check for the telemetry data (wind, temperature, and precipitation) gathered and archived at HMS. The program will read from a user-specified telemetry data file and check that file for possible errors in the data. A listing of all the input data, along with any messages for errors encountered during processing, is written to an output file.

The program prompts the user for an input file name. Error messages are displayed if the user input is not in the correct format. When processing is complete, the user has the option to continue execution and process another file.

USER'S GUIDE

This section describes how to invoke the TELEMQA computer code on the HMS computer system and what is expected from the user during execution. Basically, there are five steps to executing the TELEMQA computer code, replying to prompts for information, and terminating the program. These steps are listed below, and each step is discussed in detail in the following paragraphs:

- Log in to the HMS computer (described on page 3).
- Type RUN TELEMQA and depress RETURN key.
- Supply input data information.
- Answer "yes" or "no" to "would you like to process another file" prompt.
- Terminate program.

To invoke the program, the system prompt '\$' must be visible. The user must type RUN TELEMQA and depress the RETURN key to start program execution. The screen will clear, and a message will appear describing what the program will do. Along with the message, a menu will appear listing the input file options available.

At this point, the user is asked to enter the number of the choice desired. The choices are 1 for wind data, 2 temperature data, or 3 for precipitation data. When a number is entered, and RETURN is depressed, the program will continue execution.

The user is then prompted to enter the desired input file. Follow the format instruction given in the prompt message. Type in the file name and depress the RETURN key. The validity of the file name is checked, and an error message is displayed if there is an error. If there is no error, the

program will then proceed by displaying the name of the output file that will be created during processing. At the end of each 24-hour period in the data, the date will be displayed on the screen to update you on the progress of the program.

When the processing is complete, a message will appear asking if the user would like to process another file. The user must type in a response, and depress the RETURN key. If the answer was 'y', then the program will continue by prompting for the next file. If the answer was 'n', then execution is terminated and the system prompt, '\$', will be visible.

DESCRIPTION OF INPUT FOR TELEMETRY WIND DATA

When the user chooses wind data, TELEMQA will only read data that follow the format of HMS telemetry wind data. Also, the program will only attempt to open and read from a file that has a name beginning with 'TELEMETRY_WINDS'. Each hour of data contains 3 records. Each record has information for 10 telemetry stations; thus there is space for a total of 30 stations per hour. However, at this time there are not 30 active stations.

There are 37 fields of data for each hour. The first field gives the date. The next two fields, stability and mixing level, on each of the three records will have identical values (i.e. the value for stability will be the same for all three records of the same hour). The following 10 fields on each record are unique, each representing a single station. A description of the data arrangement and the column numbers is given below.

1. DATE - the date format is YYDDDHH (columns 1-7); where YY is the last two digits of the year (e.g. 88 for 1988), DDD is the Julian day (e.g., 32 is February 1), and HH is the hour of the day (1 to 24). For example, 8821301 represents August 1, 1988, 1:00 a.m., non-leap year.
2. RECORD NUMBER - This is a single value (column 8), and ranges from 1 to 3. It signifies which of the three hour records of data is used.
3. STABILITY - This field is an integer value of length 2 (columns 9-10). It is a code that represents the current stability.
4. MIXING LEVEL - This is an integer field of length 4 (columns 11-14). This contains the altitude of the current mixing level in hundreds of feet.
5. BLANK - Seven blank spaces are placed here (columns 15-21).
6. WIND DATA - On each record there are 10 pairs of a wind direction and a wind speed (columns 22-81). There are 2 blank spaces between each pair. The pairs are in the format DDSS; where DD is the wind direction in tens of degrees, and SS is the wind speed in mph. Entries of 9999 and 0 represent no data received from that station

or bad data. An entry of 8888 represents a station that is not active at that time.

DESCRIPTION OF INPUT FOR TELEMETRY TEMPERATURE DATA

When the user chooses temperature data, TELEMQA will only read data that follow the format of HMS telemetry temperature data. Also, the program will only attempt to open and read from a file that has a name beginning with 'TELEMETRY_TEMPS'. Each hour of data contains 3 records. These records have information for 10 telemetry stations. There is space for a total of 30 stations per hour. However, at this time there are not 30 active stations.

There are 31 fields of data for each hour. The first field identifies the date. The following 10 fields on each record are unique, each representing the temperature at a single station. A description of the data arrangement and the column numbers is given below.

1. DATE - the date format is YYDDDHH (columns 1-7); where YY is the last two digits of the year (e.g. 88 for 1988), DDD is the Julian day (e.g., 32 is February 1), and HH is the hour of the day (1 - 24). For example 8821301 represents August 1, 1988, 1:00 a.m., non-leap year.
2. RECORD NUMBER - This is a single value, and ranges from 1 to 3 (column 8). It is used to identify each of the three hourly records. This field is not used for anything in the program.
3. TEMPERATURE DATA - There are ten values on each record for each hour. The fields are integer of size 4 (columns 22-81). There is a decimal assumed to be between the third and fourth digits, giving tenths of degrees.

DESCRIPTION OF INPUT FOR TELEMETRY PRECIPITATION DATA

When the user chooses precipitation data, TELEMQA will only read data that follow the format of HMS telemetry precipitation data. Also, the program will only attempt to open and read from a file that has a name beginning with 'TELEMETRY_PRECIP'. Each hour of data contains one record. This record has information for four telemetry stations.

There are five fields of data for each hour. The first field gives the date. The following four fields on each record are unique, each representing the precipitation amount at a single station. A description of the data arrangement and the column numbers is given below.

1. DATE - the date format is YYDDDHH (columns 1-7); where YY is the last two digits of the year (e.g. 88 for 1988), DDD is the Julian day (e.g., 32 is February 1), and HH is the hour of the day (1 -

24). For example, 8821301 represents August 1, 1988, 1:00 a.m., non-leap year. The date field uses a total of 7 characters.

2. **PRECIPITATION DATA** - There are four values, each integer fields of length 4. Columns 22-25 are Rattlesnake Mountain, columns 28-31 are Yakima Barricade, columns 34-37 are 300 Area, and columns 40-43 are Pasco. There is a decimal point assumed between the second and third digits, giving hundredths of inches. For missing values, 9999 is used. A value of 50 indicates a trace of precipitation.

DESCRIPTION OF OUTPUT FOR TELEMETRY WIND DATA

TELEMQA creates a formatted output file that can be used for locating and editing errors within a data file. The naming convention used for this file is the input file name, followed by the extension '_QA.ERR'.

At the top of the file is a line that contains the name of the input file. The following lines will be the actual input data, with any error messages in between the data records. The error messages contain a brief description of the error, the date of the record, and the suspect data. A list of all possible error messages, along with a description, is located below.

Error Messages in ".ERR" file:

message: **STABILITY OUT OF RANGE**

explanation: Stability code is greater than 7 or less than 1.

message: **MIXING LEVEL OUT OF RANGE**

explanation: The mixing level altitude is out of range. Must be less than 150 (1500 ft), and greater than 0.

message: **WIND DIRECTION ERROR**

explanation: The wind direction for a specific station is out of range. Must be less than 36 (360 degrees) and greater than 0.

message: **WIND SPEED OUT OF RANGE**

explanation: The wind speed for a specific station is not greater than 0 or is greater than the maximum. (75 mph for Rattlesnake Mountain, 50 mph for all other stations).

message: **LARGE WIND SPEED CHANGE**

explanation: The wind speed for a specific station has changed more than 10 mph in 1 hour.

DESCRIPTION OF OUTPUT FOR TELEMETRY TEMPERATURE DATA

The type of output file for the temperature data is the same format as that of the wind data. Only the error messages will be different. A list of all possible error messages, along with a description, is shown below.

Error Messages in ".ERR" file:

message: TEMP OUT OF RANGE

explanation: A temperature for that hour is not within the valid range. These monthly extreme ranges, in degrees F, are shown below.

JAN. :	(	62.0,	-13.0	)
FEB. :	(	62.0,	-13.0	)
MAR. :	(	73.0,	16.0	)
APR. :	(	85.0,	31.0	)
MAY :	(	94.0,	38.0	)
JUN. :	(	100.0,	43.0	)
JUL. :	(	105.0,	49.0	)
AUG. :	(	103.0,	50.0	)
SEP. :	(	96.0,	35.0	)
OCT. :	(	80.0,	16.0	)
NOV. :	(	65.0,	- 3.0	)
DEC. :	(	59.0,	-17.0	)

message: DELTA TEMP FOR AREA TOO LARGE

explanation: The difference between the lowest and highest temperatures in that hour is greater than 25.0 degrees F.

DESCRIPTION OF OUTPUT FOR TELEMETRY PRECIPITATION DATA

The output file format for the precipitation data is the same as that of the wind data. Only the error messages will be different. A list of all possible error messages, along with a description, is shown below.

Error Messages in ".ERR" file:

message: INVALID PRECIP. AMOUNT

explanation: The amount coded was either less than 0, or greater than 25, which is 0.25 in. This is a reasonable value to use for an hour of data.

message: PRECIP. NOT FOUND AT ALL STATIONS

explanation: A precipitation amount was recorded for one or more stations, but was not recorded at all four stations.

LIMITATIONS TO THE PROGRAM, DATABASE, AND COMPUTER SYSTEM

The TELEMQA computer program does have some limitations. The program is designed to check only the HMS telemetry data (wind, temperature, or precipitation). An attempt to use another type of data file will result in the program being terminated by a run-time error.

In addition, the file name specified as input must exist and reside in the correct subdirectory on the HMS computer. To prevent any errors that will terminate the program, always check to make sure that the specified input file exists before trying to execute this program.

It is also very important to follow all the directions used in the prompts. When responding to a prompt, be sure to supply the information in the specified format. Failure to do so will create an error that will end program execution.

There are also limitations to the telemetry data base that must be considered. TELEMQA is written to handle the current format of the telemetry data. The format now in use has only been in use since 1982. Therefore, any data sets created before 1982 will be unreadable to TELEMQA.

THE SFCQA COMPUTER CODE

PROGRAM OVERVIEW

SFCQA was developed to provide a quality assurance check for the surface observation data gathered at the HMS. The code is to be used by the data base administrator and/or custodian. The program is designed to read a user-specified surface observation data file and check for erroneous data. A complete listing of the input data is echoed to an output file along with any error messages inserted where the error occurred in the data.

The program also has the ability to process either a month of data or a year of data. This is specified in the file name (e.g., SFCOBSJAN or SFCOBS88). The extension '.DAT' is assumed for any input file name.

USER'S GUIDE

This section describes how to invoke the SFCQA computer code on the HMS computer system and how to respond to the various prompts associated with the code. Basically, there are five steps to executing the SFCQA computer code, replying to prompts for information, and terminating the program. These steps are listed below, and each step is discussed in detail in the following paragraphs.

- Log in to the HMS computer (described on 3).
- Type RUN SFCQA and depress RETURN key.
- Supply input data information.
- Answer "yes" or "no" to the "would you like to process another file" prompt.
- Terminate program.

To invoke the program SFCQA, the user must be logged on to the HMS computer system and at the '\$' prompt. Then type 'RUN SFCQA', and depress the RETURN key. When SFCQA is invoked, a message is displayed to the screen briefly describing what the program does.

The user is immediately asked to enter the input file name. Follow the format instructions given in the prompt message. The user must type in the file name and depress the RETURN key. If there is an error in the file name, a message is displayed informing the user of the error, and tells how to correct the problem. Otherwise the program proceeds, and displays the name of the output file that will be created as the data is processed. The code will then display the date as each 24-hour period is processed, to update the user on processing progress.

After the entire file has been processed, the user must respond to a message asking if another file needs to be processed. If so, type a 'y' and depress the RETURN; the program will continue execution by prompting for the next input filename. If the user does not wish to process another file, type an 'n' and depress RETURN; execution will terminate, returning you to the system prompt '\$'.

DESCRIPTION OF INPUT

SFCQA is designed to read only the format of data found in surface observation data files. Also, the program will only attempt to open and read a file that has a name beginning with 'SFCOBS'. Each record in the input file contains 21 fields. A description of the data arrangement and the column numbers is given below.

1. CLOUDS (columns 1-5) - the value for this field is the type of cloud at the time of observation. These are recorded every 3 hours. Up to five cloud types can be entered. The lowest cloud type is recorded in the first column, the next layer is recorded in the second column, etc. 00000 is the default and also indicates no clouds were observed.
2. DATE (columns 6-13) - The date is in the format YYMMDDHH: where YY is the last two digits of the current year (e.g., 88 for 1988), MM is the numerical equivalent of the month (e.g, 10 represents October), DD is the day of the month, and HH is the hour of the day (1-24). For example, 88101501 represents October 15, 1988, 1:00 a.m.
3. CEILING HEIGHT (columns 14-16) - The value in this field is the height of the lowest cloud base that is either broken or overcast. The value is in hundreds of feet (e.g, 250 indicates a height of 25,000 ft.)
4. SKY CONDITION (columns 17-20) - The value in this field is a code or codes for each layer of clouds in the sky. The first digit in the entry indicates the sky condition of the lowest cloud layer, the second digit indicates the sky condition for the next highest cloud layer plus the lower layer, and so on. The value 0000 indicates clear skies and is the default value.
5. VISIBILITY (columns 21-23) - The value in this field is the horizontal visibility in miles. For example, 050 indicates a visibility of 5.0 mi. The default value is 150, which is 15 mi.
6. LIQUID PRECIPITATION (columns 25-26) - This field contains a code that indicates the type of liquid precipitation at the time of the observation. A value of 00 indicates no precipitation and is also the default value.

7. FROZEN PRECIPITATION (columns 27-29) - This field contains a code that indicates the type of frozen precipitation received at the time of the observation. A value of 000 is the default value, as well as indicating no frozen precipitation.
8. OBSTRUCTIONS (columns 30-31) - This is a code that indicates the type of obstruction to vision (e.g., smoke, haze, dust). The default value, 00, indicates no obstruction to vision at the time of observation.
9. SEA-LEVEL PRESSURE (columns 32-35) - This is the sea-level pressure in millibars (mb). A sea-level pressure of 1010.2 is recorded as 102. A missing sea-level pressure is coded 9999.
10. DEW POINT TEMPERATURE (columns 36-38) - This is the value for the dew-point temperature in whole °F at the time of observation. A default missing value is recorded 999.
11. WIND DIRECTION (columns 39-40) - The value for this field is a 5-min average of the wind direction in tens of degrees at the time of observation. For example, 18 indicates the wind direction of 180 degrees. 00 is a calm wind, and 37 indicates variable wind directions. The default and missing values are both 999.
12. WIND SPEED (columns 41-42) - This field contains the value for a 5-min average of the wind speed in whole miles per hour. Calm conditions are recorded 00. The value 99 is the default value and indicates a missing value.
13. STATION PRESSURE (columns 43-46) - This is the station pressure in hundredths of inches of mercury (in. Hg). A sea-level pressure of 29.50 is recorded as 2950. A missing sea-level pressure is coded 9999.
14. DRY BULB TEMPERATURE (columns 47-49) - This temperature is in whole °F. A value of 75 indicates a temperature of 75°F. 999 is the default and missing value.
15. WET BULB TEMPERATURE (columns 50-52) - This temperature is in whole °F. A value of 75 indicates a temperature of 75°F. 999 is the default and missing value.
16. RELATIVE HUMIDITY (columns 53-55) - The value represented in this field is the relative humidity in whole percent. The default and missing value is 999.
17. SKY COVER (columns 56-57) - This value represents the amount of sky cover in tenths. For example, 05 indicates that five tenths, or half, of the sky is covered. A value of 0 indicates a clear sky, and is the default value.

18. RAIN (columns 58-59) - This field contains the amount of liquid precipitation in hundredths of an inch recorded during the hour. A value of 04 indicates that 0.04 in. of rain fell during the hour. The default value is 00, which also indicates no liquid precipitation during the hour.
19. SNOW EQUIVALENT (columns 60-61) - The value in this field is the amount of frozen precipitation in water-equivalent form. The units are in hundredths of an inch. An input value of 06 indicates that 0.06 in. (water equivalent) of snow fell during the hour. When 00 is recorded, no frozen precipitation was found.
20. SNOW UNMELTED (columns 62-63) - The value in this field is the amount of unmelted snow in tenths of inches that fell during the hour. Seven-tenths of an inch would be indicated by an 07 value. The default value is 00, which also indicates no snow fell during the hour.
21. SOLAR RADIATION (columns 65-67) - This value represents a 1-hour average of the solar radiation in hundredths of langley per minute. An hourly average of 1.0 is recorded as 100. The default value is 000, which also indicates no solar radiation.

A more detailed description of the surface observation data fields is given by Andrews and Buck (1987).

DESCRIPTION OF OUTPUT

SFCQA creates a formatted output file to be used for locating and editing any errors in the data file. The name of the output file is created within the main program. The format of the name is 'SFCOBS', followed by a three-character representation of a month or a two-number representation for a year. The extension '_QA.ERR' is then used to identify it as a quality assurance error log.

The first line of the output file contains the name of the input file. The following lines will be the actual input data, with any error message inserted between the data records. The error messages contain a brief description of the error, the date of the erroneous record, and the erroneous data itself.

A list of all possible error messages, along with a description of the problem causing the error, is given below.

Error messages in ".ERR" file:

message: ERROR IN CLOUD LAYER SEQUENCE

explanation: When the cloud types for the cloud layers were entered, a layer was skipped (e.g. a layer with a 0 cloud type was followed by a non-zero cloud type).

message: LARGE CHANGE IN CEILING HEIGHT

explanation: The altitude for the ceiling height has changed more than 20,000 ft in the last hour.

message: CEILING W/O ENOUGH SKYCOVER

explanation: There was an altitude recorded for the ceiling, and the sky cover was less than 0.6.

message: CEIL. W/O APPROPRIATE SKY COND.

explanation: There was an altitude recorded for the ceiling, and there was not a cloud layer recorded greater than 5.

message: ERROR IN CONDITION LAYER SEQ.

explanation: There was a layer skipped in the recording of the sky conditions.

message: LARGE HOURLY DIFF IN VISIBILITY

explanation: The visibility has changed 10 or more miles since last hour.

message: TWO TYPES OF LIQUID PRECIP.

explanation: There were two types of liquid precipitation recorded for the same hour.

message: MORE THAN ONE FROZEN PRECIP.

explanation: More than one type of frozen precipitation was recorded.

message: PRECIP AMT. W/O TYPE OF PRECIP

explanation: An amount of precipitation was recorded, but there was no type of precipitation recorded.

message: TYPE OF PRECIP W/O PRECIP AMT.

explanation: A type of precipitation was recorded, but there was no amount of precipitation recorded.

message: TWO TYPES OF OBSTRUCTIONS

explanation: More than one type of vision obstruction recorded.

message: INVALID WIND DIRECTION

explanation: The wind direction was less than zero, or greater than 360.

message: INVALID WIND SPEED

explanation: The wind speed is less than zero, or greater than 80 mph.

message: LARGE HRLY CHANGE IN WIND SPD.

explanation: The change in the wind speed since last hour was greater than 30 mph.

message: ERROR STAT. PRES. OUT OF RANGE

explanation: The station pressure is out of range for that month.
These monthly ranges, in in. Hg, are shown below.

JAN:	(2975, 2818)
FEB:	(2975, 2823)
MAR:	(2975, 2829)
APR:	(2975, 2849)
MAY:	(2975, 2962)
JUN:	(2954, 2860)
JUL:	(2952, 2865)
AUG:	(2955, 2856)
SEP:	(2972, 2865)
OCT:	(2875, 2819)
NOV:	(2975, 2843)
DEC:	(2975, 2810)

message: ERROR SEA PRES. OUT OF RANGE

explanation: The sea-level pressure is out of the range determined for that month. These monthly ranges, in in. Hg, are shown below.

JAN:	(3112, 2896)
FEB:	(3097, 2898)
MAR:	(3079, 2905)
APR:	(3064, 2926)
MAY:	(3053, 2938)
JUN:	(3032, 2936)
JUL:	(3030, 2941)
AUG:	(3032, 2932)
SEP:	(3052, 2941)
OCT:	(3063, 2895)
NOV:	(3090, 2919)
DEC:	(3106, 2886)

message: ERROR INVALID DRY,WET,DEW TEMPS.

explanation: This occurs when the following condition is not satisfied: Dry Bulb Temp. > Wet Bulb Temp. > Dew Point

message: ERROR DRY BULB OUT OF RANGE

explanation: The dry-bulb temperature does not fit into the determined range for that month. These monthly ranges are listed on page 10.

message: LARGE HOURLY CHANGE IN DRY BULB

explanation: The dry-bulb temperature has changed more than 10 degrees since last hour.

message: ERROR WET BULB OUT OF RANGE

explanation: The wet-bulb temperature does not fit into the determined range for that month. These ranges are the same as those used for the dry bulb temperature.

message: **LARGE HOURLY CHANGE IN WET BULB**

explanation: The wet-bulb temperature has changed more than 10 degrees since last hour.

message: **ERROR DEW POINT OUT OF RANGE**

explanation: The dew-point temperature does not fit into the determined range for that month. These ranges are the same as those used for the dry-bulb temperature.

message: **LRG HOURLY CHANGE IN REL HUMID.**

explanation: The relative humidity has changed more than a value of 10% in the last hour.

message: **LARGE AMOUNT OF RAIN PRECIP.**

explanation: This indicates that an unusually high amount of rain fall was recorded. The allowable maximum is 0.4 in.

message: **INVALID PRECIPITATION AMOUNT**

explanation: The value '49' was used for the precipitation code. This is an invalid code.

message: **LARGE AMT. OF SNOW(EQ) PRECIP.**

explanation: This indicates that an unusually high amount of liquid equivalent for frozen precipitation was recorded. To get this message, the value is equal to, or greater than, 0.4 in.

message: **LARGE AMT. OF SNOW(UN) PRECIP.**

explanation: This indicates that an unusually high amount of frozen precipitation was recorded for that hour. The allowable maximum is 3 in.

message: **SOLAR RAD. AT INVALID HOUR**

explanation: This message indicates that solar radiation was recorded for an hour that does not fit into the time period determined for that season. The time ranges used are listed below.

summer (April - September) 4:00 a.m. to 9:00 p.m.

winter (October - March) 6:00 a.m. to 6:00 p.m.

message: **ABNORMAL DROP IN SOLAR RAD.**

explanation: There was a drop of at least 0.20 langley before 12 noon.

message: **ABNORMAL RISE IN SOLAR RAD.**

explanation: There was a rise of at least 0.20 langley after 12 noon.

message: **ABNORMALLY HIGH SOLAR RADIATION**

explanation: The amount recorded was greater than 2.00 langley.

message: VSBY MUST BE < 6 MILES WITH PCPN OR OBSTR

explanation: When there is precipitation or any sort of obstruction recorded (i.e. fog), then visibility should be less than 6 miles.

message: VSBY MUST BE > 6 MILES WITH NO PCPN OR OBSTR

explanation: When there is no precipitation or any obstruction recorded, the visibility should be greater than 6 miles.

message: LARGE CHANGE IN STATION PRESSURE

explanation: The station pressure has changed more than 0.3 inches in 1 hour.

LIMITATIONS TO THE PROGRAM, DATA BASE, AND COMPUTER SYSTEM

The SFCQA computer program does have some limitations. The program is designed to check only the surface observation data collected at HMS. An attempt to use another type of data file will result in the program being terminated by a run-time error.

In addition, the file name specified as input must exist and reside in the correct sub-directory on the HMS computer. To prevent any errors that will terminate the program, always check to make sure that the specified input file exists before trying to execute this program.

It is also very important to follow all the directions used in the prompts. When responding to a prompt, be sure to supply the information in the specified format. Failure to do so will create an error that will end program execution.

REFERENCES

Buck, J.W. and G.L. Andrews. 1987. Hanford Meteorological Station Computer Codes Volume 5 - The TOWER Computer Code. PNL-6279 Vol. 5, Pacific Northwest Laboratory, Richland, Washington.

Andrews, G.L. and J.W. Buck. 1987. Hanford Meteorological Station Computer Codes Volume 6 - The SFC Computer Code. PNL-6279 Vol. 6, Pacific Northwest Laboratory, Richland, Washington.

APPENDIX A

TROUBLESHOOTING THE TERMINAL AND COMPUTER SYSTEM PROBLEMS

APPENDIX A

TROUBLESHOOTING THE TERMINAL AND COMPUTER SYSTEM PROBLEMS

This section provides information that may help resolve problems that occur when operating the QA computer codes. 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 QA computer codes and the suggested recovery procedures. If a series of recovery actions is suggested, you should try each action in sequence until the problem or failure is resolved.

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 Emergency Meteorological System (EMS) computer manager (Eric Stephan, 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 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.

If the problem cannot be resolved, check with Ora Gifford, Ken Burk (376-5913) or Gregg Andrews (376-6104)

APPENDIX B

DESCRIPTION, FLOW CHART, AND CODE LISTINGS FOR THE DTC PROGRAM

APPENDIX B

DESCRIPTION, FLOW CHART, AND CODE LISTINGS FOR THE DTC PROGRAM

DTC.FOR

DTC is the main program that interacts with the user and passes control to other subroutines until the program is terminated by the user. The source code is located in the [FCST.QA_PROGRAMS] subdirectory on the HMS computer system. Figure B.1 shows a flow chart for the DTC program. Figure B.2 shows the hierarchy of the subroutines used by DTC.

The program starts by clearing the screen and displaying a message that tells the user what the code is used for. A menu is also displayed. The user must choose an item on the menu. Once a valid response has been given, DTC passes control to the appropriate subroutine.

When these subroutines receive control, the user is prompted for the input file name. After a valid file name is entered, the file is processed. The user is then asked if another file of the same type needs to be processed. If so, control stays in the subroutine and the user is prompted for the next input file name. If not, control will be passed back to DTC.

When control is passed back to DTC, the user is asked if the program should go back to the main menu. If so, the program resumes execution at the beginning. If not, the program will terminate passing control back to the system.

FIGURE B.1. Flow Chart for DTC Program

FIGURE B.2. Subroutines Used by DTC


```

0066      *          *****',///,
0067      *          3X,'This routine will perform a date and time')
0068      101  FORMAT(3X,'check for the type of data you specify.',/,
0069      *          3X,'Enter the number of ONE of the choices:',/,
0070      *          5X,'(1) Surface Observations',/,
0071      *          5X,'(2) Tower Data (400 ft.)',/,
0072      *          5X,'(3) Tower Data (200 ft.)',/,
0073      *          5X,'(4) Telemetry Wind Data',/,
0074      *          5X,'(5) Telemetry Temperature Data',/,
0075      *          5X,'(6) Telemetry Precipitation Data',/,
0076      *          5X,'(7) Exit Program',/)
0077      WRITE(6,106)
0078      106  FORMAT(3X,'Enter Your Choice > ',8)
0079
0080      READ(5,'(i4)') data_type
0081
0082      IF((data_type .LT. 1) .OR. (data_type .GT. 7)) THEN
0083          WRITE(6,107) data_type
0084      107  FORMAT(///,'*** ERROR - ',i4,' is an INVALID choice',////)
0085          CALL wait(4.0)
0086          GOTO 1
0087      ENDIF
0088
0089      C
0090      C MAKE CALL TO APPROPRIATE SUBROUTINE THAT WILL READ AND CHECK
0091      C THE DATE AND TIME
0092      C
0093
0094      data_error = .FALSE.
0095
0096      infile = '
0097      error_file = '
0098
0099      IF(data_type .EQ. 1) CALL surface(data_error,infile,error_file)
0100
0101      IF(data_type .EQ. 2 .OR.
0102      * data_type .EQ. 3) CALL tower(data_error,infile,error_file)
0103
0104      IF(data_type .EQ. 4 .OR.
0105      * data_type .EQ. 5 .OR.
0106      * data_type .EQ. 6) CALL telem(data_error,infile,error_file)
0107
0108      IF(data_type .EQ. 7) GOTO 113
0109
0110      C
0111      C CHECK TO SEE IF THERE IS AN ERROR IN THE FILE. IF SO, PRINT
0112      C MESSAGE TELLING WHERE TO FIND ERROR LISTING, IF NOT, PRINT
0113      C MESSAGE SAYING SO.
0114      C
0115
0116      IF(data_error) THEN
0117          WRITE(6,108) error_file
0118      108  FORMAT(////,3X,'A Listing of Errors can be found in file : ',/,
0119      *          3X,'>>>>>> ',30a,////)
0120      ELSE
0121          WRITE(6,109) infile
0122      109  FORMAT(////,3X,'There were NO errors found in file : ',a20,////)
0123      ENDIF
0124
0125      C
0126      C ASK USER IF ANOTHER FILE NEEDS TO BE CHECKED
0127      C
0128
0129      WRITE(6,110)
0130      110  FORMAT(/,3X,'Would You like to Check another file? (y or n): ',8)

```

```

0131      READ(5,'(a1)')more
0132      IF((more .EQ. 'y') .OR. (more .EQ. 'Y')) GOTO 1
0133
0134      113  WRITE(6,111)
0135      111  FORMAT(//,3x,' DATE_TIME_CHECK - Execution Complete',/)
0136
0137      STOP ' '
0138      END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	436	PIC COM REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	817	PIC COM REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	124	PIC COM REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	1371	

ENTRY POINTS

Address	Type	Name	References
0-00000000		DTC\$MAIN	

VARIABLES

Address	Type	Name	Attributes	References
2-0000003E 100	I=2	DATA_TYPE		48 88= 02(2) 83 99 101(2) 104(3)
2-00000040	L=4	DATE ERROR		54 94= 99A 101A 104A 110
2-0000001E	CHAR	ERROR FILE		51 97= 99A 101A 104A 117
2-00000000	CHAR	INFILE		50 90= 99A 101A 104A 121
2-0000003C	CHAR	MORE		52 131= 132(2)

LABELS

Address	Label	References
0-0000000C	1	61 80 132
1-00000006	100'	61 63
1-00000007	101'	62 68
1-000001EF	100'	77 70
1-0000020C	107'	83 84
1-0000023C	100'	117 118
1-00000203	109'	121 122
1-00000207	110'	129 130
1-000002F1	111'	134 135
0-0000018E	113	108 134

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	References
	SURFACE	99
	TELEM	104

TOWER
WAIT

101
86

```
-----
      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/CROSS/CHECK=ALL/EXTEND/CONTINUATION=99 DTC
/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)
/DEBUG=(NOSYMBOLS,TRACEBACK)
/STANDARD=(NOSYNTAX,NOSOURCE FORM)
/SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP,NODICTIONARY,SINGLE)
/WARNINGS=(GENERAL,NODECLARATIONS,NOLTRIX)
/CONTINUATIONS=99 /CROSS REFERENCE /MOD LINES /EXTEND_SOURCE /F77
/NOG_FLOATING /I4 /NOMACHINE_CODE /OPTIMIZE
```

COMPILATION STATISTICS

```
Run Time:      3.24 seconds
Elapsed Time:   7.11 seconds
Page Faults:    671
Dynamic Memory: 371 pages
```

```
0001  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002  C
0003  C   SURFACE.FOR
0004  C
0005  C   WRITTEN BY:  K. W. BURK
0006  C
0007  C   PURPOSE :   THIS ROUTINE IS DESIGN TO READ IN AND CHECK THE DATE ON
0008  C                 A SURFACE OBSERVATION DATA FILE SPECIFIED BY THE USER.
0009  C
0010  C   ALGORITHM :  DECLARE VARIABLES
0011  C                 PROMPT USER FOR INPUT FILE
0012  C                 PROMPT USER FOR BEGINNING DATE
0013  C                 OPEN FILE
0014  C                 LOOP UNTIL END OF FILE
0015  C                 READ DATE
0016  C                 CHECK DATE
0017  C                 IF ERROR SET ERROR FLAG
0018  C                 INCRUMENT
0019  C                 END LOOP
0020  C                 RETURN
0021  C
0022  C   VERSION :    1.0
0023  C
0024  C   DATE :       APRIL 1988
0025  C
0026  C   CALLED FROM :  DATE_TIME_CHECK.FOR
0027  C
0028  C   CALLS TO :    ADD_HOUR.FOR
0029  C
0030  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0031
```

```

0032      SUBROUTINE surface(surface_error,surface_file,surface_err_file)
0033
0034      IMPLICIT NONE
0035
0036      C
0037      C      VARIABLE DECLARATIONS
0038      C-----
0039      C      TYPE      | VARIABLE      | COMMENTS
0040      C-----
0041      C
0042
0043      INTEGER*2      check_year      ! Year value used to check current value
0044      INTEGER*2      check_month     ! Month value used to check current value
0045      INTEGER*2      check_day       ! Day value used to check current value
0046      INTEGER*2      check_hour      ! Hour value used to check current value
0047      INTEGER*2      year            ! Year value of current record
0048      INTEGER*2      month           ! Month value of current record
0049      INTEGER*2      day             ! Day value of current record
0050      INTEGER*2      hour            ! Hour value of current record
0051      INTEGER*2      file_name_len   ! Number of characters in the filename
0052      INTEGER*2      i,l             ! Indices
0053      INTEGER*4      record          ! Record number
0054
0055      CHARACTER*20    surface_file    ! Input file name
0056      CHARACTER*20    surface_err_file ! File name that has listing of errors
0057
0058      LOGICAL         surface_error    ! Boolean for error found in surface data
0059
0060      C
0061      C      PROMPT USER FOR NAME OF INPUT SURFACE OBSERVATION FILE,
0062      C      AND BEGINNING DATE OF THAT FILE
0063      C
0064
0065      1      WRITE(6,200)
0066      200      FORMAT(//,5x,'* * * SURFACE OBSERVATIONS * * *')
0067      WRITE(6,201)
0068      201      FORMAT(/,3x,'Enter the name of the Surface Observation input file, ',/,
0069      *          3x,'Assume it is located at [FCST.DAT]: ',/,
0070      *          3x,'(ex. sfcobaway) Enter WITHOUT ".dat" > ',/)
0071
0072      READ(5,'(a)')surface_file
0073
0074      IF((surface_file(1:6) .NE. 'sfcoas') .AND.
0075      *   (surface_file(1:6) .NE. 'SFCOBS')) THEN
0076      TYPE=
0077      TYPE=,'      *** ERROR in filename, must begin with SFCOBS '
0078      GOTO 1
0079      ENDIF
0080
0081
0082      DO 237 i = 1,30
0083      237      IF(surface_file(i:i) .EQ. ' ') GOTO 238 ! FIND THE LENGTH OF THE INPUT
0084      238      l = i - 1 ! FILENAME
0085
0086      surface_err_file = surface_file(1:l)//'_DATE.ERR'
0087
0088      202      WRITE(6,203)
0089      203      FORMAT(//,3x,'Enter the starting DATE and TIME: ',/,
0090      *          8x,'(ex. 88,05,01,01) YY,MM,DD,HH > ',/)
0091
0092      READ(5,'(4(i2,1x))')check_year,check_month,check_day,check_hour
0093
0094      C
0095      C      CHECK THE USER INPUT FOR OBVIOUS ERRORS
0096      C

```

```

0097
0098 IF((check_month .GT. 12) .OR. (check_month .LT. 1)) THEN
0099     TYPE*
0100     TYPE*, '*** ERROR IN ENTERED MONTH ', check_month
0101     GOTO 202
0102 ENDIF
0103
0104 IF((check_day .GT. 31) .OR. (check_day .LT. 1)) THEN
0105     TYPE*
0106     TYPE*, '*** ERROR IN ENTERED DAY ', check_day
0107     GOTO 202
0108 ENDIF
0109
0110 IF((check_hour .GT. 24) .OR. (check_hour .LT. 1)) THEN
0111     TYPE*
0112     TYPE*, '*** ERROR IN ENTERED HOUR ', check_hour
0113     GOTO 202
0114 ENDIF
0115
0116 C
0117 C OPEN INPUT AND THE ERROR OUTPUT FILES
0118 C
0119
0120 OPEN(UNIT=11, FILE='[FCST.DAT] '// surface_file, STATUS='OLD', READONLY)
0121 OPEN(UNIT=21, FILE=surface_err_file, STATUS='NEW')
0122
0123 C
0124 C BEGIN PROCESSING THE DATA
0125 C
0126
0127 TYPE*
0128 TYPE*, ' Beginning to Process Data . . . '
0129
0130 surface_error = .FALSE.
0131 record = 1
0132
0133 WRITE(21, 204) surface_file
0134 204 FORMAT(//, 5x, 'Error Log for the SURFACE datafile : ', a30, /,
0135 *          1x, 80('-',) /)
0136
0137 205 READ(11, '(6x, 4i2)', end=299) year, month, day, hour
0138
0139 IF((check_year .NE. year) .OR. (check_month .NE. month) .OR.
0140 * (check_day .NE. day) .OR. (check_hour .NE. hour)) THEN
0141
0142     surface_error = .TRUE.
0143
0144     WRITE(21, 206) record, year, month, day, hour
0145 206 FORMAT(' Error in date at or before record ', i5, ' date : ', 4(i2, 1x))
0146
0147     check_year = year
0148     check_month = month
0149     check_day = day
0150     check_hour = hour
0151
0152 ENDIF
0153 C
0154 C INCREMENT HOUR BY ONE TO CHECK NEXT RECORD
0155 C
0156
0157 CALL add_hour(check_hour, check_day, check_month, check_year)
0158
0159 record = record + 1
0160 GOTO 205
0161

```

```

0162 299 record = record - 1
0163
0164 WRITE(21,207) record
0165 207 FORMAT(' Total Number of Records in file ',i6)
0166
0167 CLOSE(11)
0168 CLOSE(21)
0169
0170 RETURN
0171
0172 END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	992	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	838	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	168	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	1798	

ENTRY POINTS

Address	Type	Name	References
0-00000000		SURFACE	32#

VARIABLES

Address	Type	Name	Attributes	References
2-00000004 149=	I=2	CHECK_DAY		45 92= 104(2) 106 139
2-00000006 150=	I=2	CHECK_HOUR	157A	46 92= 110(2) 112 139
2-00000002 148=	I=2	CHECK_MONTH	157A	44 92= 98(2) 100 139
2-00000000 2-0000000C	I=2	CHECK_YEAR DAY	157A	43 92= 130 147= 157A 49 137= 130 144 149
** 2-0000000E	I=2	FILE_NAME_LEN HOUR		51 50 137= 139 144 150
** 2-0000000A	I=2	I L		52 82= 83(2) 84 52 84= 86
** 104	I=2	MONTH		48 137= 139 144 148
AP-00000040 AP-000000C0 AP-00000000 06	I=4	RECORD SURFACE ERROR CHAR SURFACE_ERR FILE CHAR SURFACE_FILE		53 131= 144 159(2)= 162(2)= 32 50 130= 142= 32 50 86= 121A 32 55 73= 75(2) 83
2-00000000	I=2	YEAR	120 133	47 137= 130 144 147

LABELS

Address	Label	References
---------	-------	------------

0-00000028	1	88#	79		
1-000000A3	280'	88	87#		
1-000000CD	281'	88	89#		
0-0000012C	282	88#	181	187	113
1-00000182	283'	88	89#		
1-000001BC	284'	133	134#		
0-000002DC	285	137#	188		
1-000001FD	286'	144	145#		
1-00000239	287'	164	165#		
**	237	82	83#		
0-000000EA	238	83	84#		
0-000003AC	299	137	182#		

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	References
	ADD HOUR	167
	FORCLOSE	187 188
	FOR\$OPEN	128 121

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/CROSS/CHECK=ALL/EXTEND/CONTINUATION=99 SURFACE

/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)
/DEBUG=(NOSYMBOLS,TRACEBACK)
/STANDARD=(NOSYNTAX,NOSOURCE FORM)
/SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP,NODICTIONARY,SINGLE)
/WARNINGS=(GENERAL,NODECLARATIONS,NOULTRIX)
/CONTINUATIONS=99 /CROSS REFERENCE /NOO LINES /EXTEND_SOURCE /F77
/NOO_FLOATING /I4 /NOMACHINE_CODE /OPTIMIZE

```

COMPILATION STATISTICS

```

Run Time:      5.38 seconds
Elapsed Time:   13.85 seconds
Page Faults:    374
Dynamic Memory: 418 pages

```

```

0001  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002  C
0003  C    TOWER.FOR
0004  C

```



```

0070      *      3x,'-----',/,
0071      *      3x,'Ex. for 400 ft. Tower (TOWER_400FOCT)',/,
0072      *      3x,'Ex. for 200 ft. Tower (TOWER_200_100FOCT)',/,
0073      *      3x,'-----',/,
0074      *      3x,'Be sure to follow the above formats and to',/,
0075      *      3x,'enter the filename WITHOUT '.dat'      --> ', $)
0076
0077      READ(5,'(a20)')tower_file
0078
0079      IF((tower_file(1:5) .NE. 'tower') .AND.
0080      * (tower_file(1:5) .NE. 'TOWER')) THEN
0081          TYPE*,
0082          TYPE*, '      *** ERROR in file name, must begin with TOWER'
0083          GOTO 1
0084      ENDIF
0085
0086      DO 237 i = 1,30
0087      237  IF(tower_file(i:i) .EQ. ' ') GOTO 238 ! FIND THE LENGTH OF THE INPUT
0088      238  I = i - 1 ! FILENAME
0089
0090      tower_err_file = tower_file(1:I)//'_DATE.ERR'
0091
0092      302  WRITE(6,303)
0093      303  FORMAT(//,3x,'Enter the starting DATE and TIME:',/,
0094      *      3x,'(ex. 88,05,01,01) YY,MM,DD,HH > ', $)
0095      READ(5,'(4(i2,1x))')check_year,check_month,check_day,check_hour
0096
0097      C
0098      C      CHECK THE USER INPUT FOR OBVIOUS ERRORS
0099      C
0100
0101      IF((check_month .GT. 12) .OR. (check_month .LT. 1)) THEN
0102          TYPE*,
0103          TYPE*, '*** ERROR IN ENTERED MONTH ',check_month
0104          GOTO 302
0105      ENDIF
0106
0107      IF((check_day .GT. 31) .OR. (check_day .LT. 1)) THEN
0108          TYPE*,
0109          TYPE*, '*** ERROR IN ENTERED DAY ',check_day
0110          GOTO 302
0111      ENDIF
0112
0113      IF((check_hour .GT. 24) .OR. (check_hour .LT. 1)) THEN
0114          TYPE*,
0115          TYPE*, '*** ERROR IN ENTERED HOUR ',check_hour
0116          GOTO 302
0117      ENDIF
0118
0119      C
0120      C      OPEN INPUT AND THE ERROR OUTPUT FILES
0121      C
0122
0123      OPEN(UNIT=12,FILE='[FCST.DAT]//tower_file//'.dat',
0124      *      STATUS='OLD',READONLY)
0125      OPEN(UNIT=22,FILE=tower_err_file,STATUS='NEW')
0126
0127      C
0128      C      BEGIN PROCESSING THE DATA
0129      C
0130
0131      TYPE*,
0132      TYPE*, ' Beginning to Process Data . . . '
0133
0134      tower_error = .FALSE.

```

```

0135      record = 1
0136
0137      WRITE(22,304) tower file
0138 304   FORMAT(/,5x,'Error Log for the TOWER datafile : ',a30,/,
0139      *      1x,80(' '),/)
0140
0141 305   READ(12,'(4i2)',end=399) year, month, day, hour
0142
0143      IF((check_year .NE. year) .OR. (check_month .NE. month) .OR.
0144      *   (check_day .NE. day) .OR. (check_hour .NE. hour)) THEN
0145
0146          tower_error = .TRUE.
0147
0148      WRITE(22,306) record, year, month, day, hour
0149 306   FORMAT(' Error in date at or before record ',i5,' date : ',4(i2,1x))
0150
0151          check_year = year
0152          check_month = month
0153          check_day = day
0154          check_hour = hour
0155
0156      ENDIF
0157
0158      C
0159      C   INCRUMENT HOUR BY ONE TO CHECK NEXT RECORD
0160      C
0161
0162      CALL add_hour(check_hour, check_day, check_month, check_year)
0163
0164      record = record + 1
0165      GOTO 305      ! GET NEXT RECORD
0166
0167 399   record = record - 1
0168
0169      WRITE(22,307) record
0170 307   FORMAT(/,3x,' Total Number of Records in file ',i6)
0171
0172      CLOSE(12)
0173      CLOSE(22)
0174
0175      RETURN
0176
0177      END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	1000	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	880	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	168	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	2028	

ENTRY POINTS

Address	Type	Name	References
0-00000000		TOWER	324

VARIABLES

Address	Type	Name	Attributes	References
2-00000004 153=	I*2	CHECK_DAY		45 95= 107(2) 109 143
2-00000006 154=	I*2	CHECK_HOUR	102A	48 95= 113(2) 115 143
2-00000002 152=	I*2	CHECK_MONTH	102A	44 95= 101(2) 103 143
2-00000000 2-0000000C	I*2	CHECK_YEAR DAY	102A	43 95= 143 151= 162A 49 141= 143 148 153
2-0000000E **	I*2	HOUR I		50 141= 143 148 154 51 88= 87(2) 88
**	I*2	L		51 88= 90
2-0000000A **	I*2	MONTH RECORD		48 141= 143 148 152 52 135= 148 164(2)= 187(2)=
109				
AP-000000040 AP-0000000C0 AP-000000080 90	L*4	TOWER ERROR CHAR TOWER_ERR FILE CHAR TOWER_FILE		32 57 134= 148= 32 55 90= 125A 32 54 77= 79(2) 87
2-00000008	I*2	YEAR	123 137	47 141= 143 148 151

LABELS

Address	Label	References
0-00000028 **	1 237	85# 83 88 87#
0-000000EA 1-000000A2 1-000000C2	238 300' 301'	87 88# 85 86# 87 86#
0-0000012C 1-00000243 1-0000020D 0-000002E4 1-000002DA	302 303' 304' 305 306'	92# 164 110 110 92 93# 137 138# 141# 165 148 149#
1-00000316 0-000003B4	307' 309	169 170# 141 167#

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	References
	ADD HOUR	162
	FOR\$CLOSE	172 173
	FOR\$OPEN	123 125

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/CROSS/CHECK=ALL/EXTEND/CONTINUATION=99 TOWER

```

/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)
/DEBUG=(NOSYMBOLS,TRACEBACK)
/STANDARD=(NOSYNTAX,NOSOURCE,FORM)
/SHOW=(NOPREPROCESSOR,NOINCLDDE,MAP,NODICTIONARY,SINGLE)
/WARNINGS=(GENERAL,NODECLARATIONS,NOLTRIX)
/CONTINUATIONS=99 /CROSS REFERENCE /NOD LINES /EXTEND_SOURCE /FTT
/NOD FLOATING /I4 /NOMACHINE CODE /OPTIMIZE

```

COMPILATION STATISTICS

```
Run Time:      5.21 seconds
Elapsed Time:  8.44 seconds
Page Faults:   365
Dynamic Memory: 394 pages
```

```

0001 C
0002 C
0003 C      TELEMETRY.FOR
0004 C
0005 C      WRITTEN BY:  K. W. BURK
0006 C
0007 C      PURPOSE :   THIS ROUTINE IS DESIGN TO READ IN AND CHECK THE DATE ON
0008 C                  A HMS TELEMETRY DATA FILE SPECIFIED BY THE USER.
0009 C
0010 C      ALGORITHM :  DECLARE VARIABLES
0011 C                  PROMPT USER FOR INPUT FILE
0012 C                  PROMPT USER FOR BEGINNING DATE
0013 C                  OPEN FILE
0014 C                  LOOP UNTIL END OF FILE
0015 C                  READ DATE
0016 C                  CONVERT DATE FROM JULIAN TO GREGORIAN
0017 C                  CHECK DATE
0018 C                  IF ERROR SET ERROR FLAG
0019 C                  INCREMENT HOUR
0020 C                  END LOOP
0021 C                  RETURN
0022 C
0023 C      VERSION :    1.0
0024 C
0025 C      DATE :       APRIL 1988
0026 C
0027 C      CALLED FROM :  DATE_TIME_CHECK.FOR
0028 C
0029 C      CALLS TO :    GREGORIAN.FOR
0030 C                  ADD_HOUR.FOR
0031 C
0032 C
0033 C
0034 C      SUBROUTINE telem(telem_error,telem_file,telem_err_file)
0035 C
0036 C      IMPLICIT NONE

```

```

0037
0038 C
0039 C      VARIABLE DECLARATIONS
0040 C-----
0041 C      TYPE      ! VARIABLE      ! COMMENTS
0042 C-----
0043 C
0044
0045 INTEGER*2      check_year      ! Year value used to check current value
0046 INTEGER*2      check_month     ! Month value used to check current value
0047 INTEGER*2      check_day       ! Day value used to check current value
0048 INTEGER*2      check_hour      ! Hour value used to check current value
0049 INTEGER*2      year            ! Year value of current record
0050 INTEGER*2      month           ! Month value of current record
0051 INTEGER*2      day             ! Day value of current record
0052 INTEGER*2      hour            ! Hour value of current record
0053 INTEGER*2      julian_day      ! Julian day read from data file
0054 INTEGER*2      i,i            ! Indices
0055 INTEGER*4      record          ! Record number
0056
0057 CHARACTER*30    telem_file      ! Name of the input file
0058 CHARACTER*30    telem_err_file  ! File that has listing of errors
0059
0060 LOGICAL          telem_error     ! Boolean for error found
0061
0062
0063 C
0064 C      PROMPT USER FOR NAME OF INPUT TELEMETRY DATA FILE,
0065 C      AND BEGINNING DATE OF THAT FILE
0066 C
0067
0068 1  WRITE(6,400)
0069 400 FORMAT(///,5x,'* * * TELEMETRY DATA * * *')
0070 WRITE(6,401)
0071 401 FORMAT(/,3x,'Enter the name of the Telemetry Data input file, ',/,
0072 *      3x,'Assume it is located at [FCST.DAT]: ',/,
0073 *      3x,'Assume file name begins with "TELEMETRY_"',/,
0074 *      3x,'-----',/,
0075 *      3x,'For Telem WIND Data (ex. WINDSOCT)',/,
0076 *      3x,'For Telem TEMPERATURE Data (ex. TEMPSOCT)',/,
0077 *      3x,'For Telem PRECIPITATION Data (ex. PRECIPSOCT)',/,
0078 *      3x,'-----',/,
0079 *      3x,'Be sure to follow the above formats and to ',/,
0080 *      3x,'enter the file specifications WITHOUT ".dat" --> ',/,
0081
0082 READ(5,'(a20)')telem_file
0083
0084 IF((telem_file(1:5) .NE. 'winds' .AND. telem_file(1:5) .NE. 'WINDS') .AND.
0085 * (telem_file(1:5) .NE. 'temps' .AND. telem_file(1:5) .NE. 'TEMPS') .AND.
0086 * (telem_file(1:5) .NE. 'preci' .AND. telem_file(1:5) .NE. 'PRECIP')) THEN
0087   TYPE=
0088   TYPE=, ' *** ERROR in filename specs, no WINDS, TEMPS, or PRECIP'
0089   GOTO 1
0090 ENDIF
0091
0092 DO 237 i = 1,30
0093 237 IF(telem_file(i:i) .EQ. ' ') GOTO 238 ! FIND THE LENGTH OF THE INPUT
0094 238 i = i - 1 ! FILENAME
0095
0096 telem_err_file = 'TELEM_'//telem_file(1:i)//'_DATE.ERR'
0097
0098 402 WRITE(6,403)
0099 403 FORMAT(///,3x,'Enter the starting DATE and TIME:',/,
0100 *      3x,'-----',/,
0101 *      3x,'Even though Telemetry data uses the',/,

```

```

0102      *      3x,'Julian day system, it will be',/,
0103      *      3x,'converted',/,
0104      *      3x,'-----',/,
0105      *      3x,'(ex. 88,06,01,01) YY,MM,DD,HH > ',/,
0106
0107      READ(5,'(4(i2,1x))')check_year,check_month,check_day,check_hour
0108
0109      C
0110      C      CHECK THE USER INPUT FOR OBVIOUS ERRORS
0111      C
0112
0113      IF((check_month .GT. 12) .OR. (check_month .LT. 1)) THEN
0114          TYPE*,
0115          TYPE*, '*** ERROR IN ENTERED MONTH ',check_month
0116          GOTO 402
0117      ENDIF
0118
0119      IF((check_day .GT. 31) .OR. (check_day .LT. 1)) THEN
0120          TYPE*,
0121          TYPE*, '*** ERROR IN ENTERED DAY ',check_day
0122          GOTO 402
0123      ENDIF
0124
0125      IF((check_hour .GT. 24) .OR. (check_hour .LT. 1)) THEN
0126          TYPE*,
0127          TYPE*, '*** ERROR IN ENTERED HOUR ',check_hour
0128          GOTO 402
0129      ENDIF
0130
0131      C
0132      C      OPEN INPUT AND THE ERROR OUTPUT FILES
0133      C
0134
0135      OPEN(UNIT=13,FILE='[FCST.DAT]TELEMETRY_'//telem_file//'.dat',
0136      *      STATUS='OLD',READONLY)
0137      OPEN(UNIT=23,FILE=telem_err_file,STATUS='NEW')
0138
0139      C
0140      C      BEGIN PROCESSING THE DATA
0141      C
0142
0143      TYPE*,
0144      TYPE*, ' Beginning to Process Data . . .'
0145
0146      telem_error = .FALSE.
0147      record = 1
0148
0149      WRITE(23,404)telem_file
0150      404  FORMAT(//,5x,'Error Log for the TELEMETRY datafile : TELEMETRY_',a30,/,
0151      *      1x,80('-'),/)
0152
0153      405  READ(13,'(i2,i3,i2)',end=499)year, julian_day, hour
0154
0155      IF(telem_file(1:6) .NE. 'PRECIP' .AND. telem_file(1:6) .NE. 'precip') THEN
0156          READ(13,*)
0157          READ(13,*)
0158      ENDIF
0159
0160      C
0161      C      CONVERT THE JULIAN DAY TO THE STANDARD MONTH-DAY FORMAT
0162      C
0163
0164      CALL gregorian(year, julian_day, month, day)
0165
0166

```



```

0167      IF((check_year .NE. year) .OR. (check_month .NE. month) .OR.
0168      * (check_day .NE. day) .OR. (check_hour .NE. hour)) THEN
0169
0170          telem_error = .TRUE.
0171
0172          WRITE(23,400)julan day,record
0173 400      FORMAT(' Error in date at or before JULIAN day ',i5,' at record ',i5)
0174
0175          check_year = year
0176          check_month = month
0177          check_day = day
0178          check_hour = hour
0179
0180      ENDIF
0181
0182      C
0183      C      INCRUMENT HOUR BY ONE TO CHECK NEXT RECORD
0184      C
0185
0186      CALL add_hour(check_hour, check_day, check_month, check_year)
0187
0188      IF(telem_file(1:6) .NE. 'PRECIP' .AND. telem_file(1:6) .NE. 'precip') THEN
0189          record = record + 3
0190      ELSE
0191          record = record + 1
0192      ENDIF
0193
0194      GO TO 405          ! GET NEXT RECORD
0195
0196 400      record = record - 1
0197
0198      WRITE(23,407) record
0199 407      FORMAT(/,3x,' Total Number of Records in file ',i0)
0200
0201      CLOSE(13)
0202      CLOSE(23)
0203
0204      RETURN
0205
0206      END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	1148	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	1284	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	192	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	2544	

ENTRY POINTS

Address	Type	Name	References
0-00000000		TELEM	34#

VARIABLES

Address	Type	Name	Attributes	References
2-00000004	I+2	CHECK_DAY		47 167= 119(2) 121 167
177=				

2-00000006 I*2 CHECK_HOUR	186A	48	187=	125(2)	127	187
178=						
2-00000002 I*2 CHECK_MONTH	186A	48	187=	113(2)	115	187
178=						
2-00000005 I*2 CHECK_YEAR	186A	45	187=	187	175=	188A
2-0000000C I*2 DAY		51	184A	187	177	
2-0000000E I*2 HOUR		52	153=	187	178	
** I*2 I		54	92=	93(2)	94	
2-00000015 I*2 JULIAN_DAY		53	153=	184A	172	
** I*2 L		54	94=	98		
2-0000000A I*2 MONTH		58	184A	187	178	
** I*4 RECORD		55	147=	172	189(2)=	191(2)=
198(2)=						
AP-00000040 L*4 TELEM_ERROR	198	34	88	148=	178=	
AP-0000000C0 CHAR TELEM_ERR_FILE		34	58	98=	137A	
AP-000000080 CHAR TELEM_FILE		34	57	82=	84(8)	93
98						
2-00000008 I*2 YEAR	135	149	155(2)	188(2)		
			49	153=	184A	187
						175

LABELS

Address	Label	References
0-00000028	1	88# 89
**	237	92 93#
0-00000122	238	93 94#
1-000000A7	480'	68 89#
1-000000CB	481'	78 71#
0-00000178	482	98# 118 122 128
1-000002B2	483'	98 99#
1-000003B4	484'	149 150#
0-00000328	486	153# 194
1-00000481	486'	172 173#
1-0000043E	487'	198 199#
0-00000448	499	153 198#

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	References
	ADD HOUR	188
	FOR\$CLOSE	281 282
	FOR\$OPEN	135 137
	GREGORIAN	184

0001

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

FOR/LIS/CROSS/CHECK=ALL/EXTEND/CONTINUATION=99 TELEM

COMPILATION STATISTICS

```
Run Time:      8.88 seconds
Elapsed Time:  25.28 seconds
Page Faults:   388
Dynamic Memory: 428 pages
```

```

0001 C
0002 C
0003 C      ADD_HOUR.FOR
0004 C
0005 C      WRITTEN BY :   G. L. ANDREWS
0006 C      MODIFIED   :   K. W. BURK
0007 C
0008 C      PURPOSE :      GIVEN A YEAR, MONTH, DAY, AND HOUR, PASS BACK A DATE AND
0009 C                     TIME THAT IS ONE HOUR GREATER.  WILL HANDLE ANY SITUATION
0010 C                     SUCH AS;  END OF DAY, END OF MONTH, END OF YEAR, AND
0011 C                     LEAP YEAR
0012 C
0013 C      ALGORITHM :    DECLARE VARIABLES
0014 C                     INITIALIZE ARRAYS
0015 C                     CHECK FOR LEAP YEAR
0016 C                     CHECK FOR END OF DAY
0017 C                     CHECK FOR END OF MONTH
0018 C                     CHECK FOR END OF YEAR
0019 C                     RETURN
0020 C
0021 C      VERSION : 1.0  (ADD_HOUR.FOR)
0022 C
0023 C      DATE      :   APRIL 1988
0024 C
0025 C      CALLED FROM :   SURFACE.FOR
0026 C                     TOWER.FOR
0027 C                     TELEM.FOR
0028 C
0029 C
0030 C
0031 C      SUBROUTINE add_hour(chour, cday, cmonth, cyear)
0032 C
0033 C
0034 C      VARIABLE DECLARATIONS
0035 C
0036 C      -----
0037 C      TYPE      !-  VARIABLE              !  COMMENTS
0038 C      -----
0039 C
0040 C      INTEGER*2      cyear              ! Input year, becomes new year

```

```

0041      INTEGER+2      cmonth      ! Input month, becomes new month
0042      INTEGER+2      cday        ! Input day, becomes new day
0043      INTEGER+2      chour       ! Input hour, becomes new hour
0044      INTEGER+2      end_of_month(12) ! Array for last day of each month
0045
0046
0047
0048      DATA end_of_month/31, 28, 31, 30, 31, 30,
0049      *          31, 31, 30, 31, 30, 31/
0050
0051      C
0052      C      CHECK FOR LEAP YEAR
0053      C
0054
0055      IF( MOD(cyear,4) .EQ. 0) THEN
0056          end_of_month(2) = 29
0057      ELSE
0058          end_of_month(2) = 28
0059      ENDIF
0060
0061      C
0062      C      BEGIN CHECK ALL POSSIBILITIES FOR NEW HOUR
0063      C
0064
0065      IF(chour .LT. 24) THEN          ! NOT END OF DAY
0066          chour = chour + 1
0067      ELSE                          ! IT IS END OF DAY
0068          IF(cday .LT. end_of_month(cmonth)) THEN ! END OF MONTH
0069              cday = cday + 1
0070              chour = 1
0071          ELSE
0072              IF(cmonth .LT. 12) THEN          ! NOT END OF YEAR
0073                  cmonth = cmonth + 1
0074                  cday = 1
0075                  chour = 1
0076              ELSE                          ! IS END OF YEAR
0077                  IF(cyear .LT. 99) THEN
0078                      cyear = cyear + 1
0079                      cmonth = 1
0080                      cday = 1
0081                      chour = 1
0082                  ENDIF ! year
0083              ENDIF ! month
0084          ENDIF ! day
0085      ENDIF ! hour
0086
0087      RETURN
0088      END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 SCODE	124	PIC COM REL LCL SHR EXE RD NOWRT LONG
2 BLOCAL	24	PIC COM REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	148	

ENTRY POINTS

Address	Type	Name	References
0-00000000		ADD_HOUR	318

Address	Type	Name	Attributes	References
AP-000000000	I+2	CDAY	31	42 68 69(2)= 74= 88=
AP-000000040	I+2	CHOUR	31	43 66 66(2)= 78= 75= 81=
AP-0000000C0	I+2	CMONTH	31	41 68 72 73(2)= 79=
AP-000000100	I+2	CYEAR	31	40 66 77 78(2)=

Address	Type	Name	Attributes	Bytes	Dimensions	References
2-00000000 00	I+2	END_OF_MONTH		24	(12)	44 48D 58= 58=

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

```

FDR/LIS/CROSS/CHECK=ALL/EXTEND/CONTINUATION=99 ADD_HOUR

/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)
/DEBUG=(NOSYMBOLS,TRACEBACK)
/STANDARD=(NOSYNTAX,NOSOURCE FORM)
/SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP,NODICTIONARY,SINGLE)
/WARNINGS=(GENERAL,NODECLARATIONS,NOLULTRIX)
/CONTINUATIONS=99 /CROSS REFERENCE /MOD LINES /EXTEND_SOURCE /FTT
/MOD FLOATING /I4 /NOMACHINE CODE /OPTIMIZE

```

Run Time:	1.98 seconds
Elapsed Time:	4.74 seconds
Page Faults:	516
Dynamic Memory:	351 pages

B.23


```

0000
0001      ENDIF
0002
0003      500  RETURN
0004
0005      END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	169	PIC COM REL LCL SHR EXE RD NOWRT LONG
2 \$LOCAL	48	PIC COM REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	211	

ENTRY POINTS

Address	Type	Name	References
0-00000000		GREGORIAN	29#

VARIABLES

Address	Type	Name	Attributes	References
AP-000000100	I*2	GDAY		20 36 55= 60= 69= 74=
AP-0000000C0	I*2	GMON		20 34 54= 61= 68= 75=
AP-000000040	I*2	OYEAR		20 32 48
AP-000000000	I*2	JDAY		20 33 55 59 60 69 73
74				
**	L*4	LY_FLAG		38 48= 53
**	I*2	MONTH		36 57= 59 60 61 71= 73
74				
			75	

ARRAYS

Address	Type	Name	Attributes	Bytes	Dimensions	References
2-000000000	I*2	DAYS_MONTH		24	(12)	30 42D 73 74
2-000000018	I*2	DAYS_MONTH_LY		24	(12)	31 43D 59 60

LABELS

Address	Label	References
0-000000A2	500	62 76 83#

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/CROSS/CHECK=ALL/EXTEND/CONTINUATION=99 GREGORIAN

/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)
/DEBUG=(NOSYMBOLS,TRACEBACK)
/STANDARD=(NOSYNTAX,NOSOURCE FORM)
/SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP,MODDICTIONARY,SINGLE)
/WARNINGS=(GENERAL,NODECLARATIONS,NOULTRIX)
/CONTINUATIONS=99 /CROSS REFERENCE /MOD LINES /EXTEND_SOURCE /F77
/NOG_FLOATING /I4 /NOMACHINE_CODE /OPTIMIZE
```

COMPILATION STATISTICS

```
Run Time:      2.25 seconds
Elapsed Time:  4.78 seconds
Page Faults:   588
Dynamic Memory: 355 pages
```

```
0001  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002  C
0003  C      WAIT.FOR
0004  C
0005  C      WRITTEN BY : K. W. BURK
0006  C
0007  C      PURPOSE : TO PROVIDE A WAY TO WAIT, OR PAUSE THE EXECUTION OF A PROGRAM
0008  C                  GIVEN THE NUMBER OF SECONDS THE USER WISHES TO WAIT.
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 : DATE_TIME_CHECK.FOR
0023  C
0024  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0025
0026  SUBROUTINE wait(wait_time)
0027
0028  IMPLICIT NONE
0029
0030  C
0031  C      DECLARE VARIABLES
0032  C-----
0033  C      TYPE      | VARIABLE      | COMMENTS
0034  C-----
0035  C
0036  INTEGER*2      chours      | Current hour
0037  INTEGER*2      cmin       | Current minute
0038  INTEGER*2      csec       | Current second
```



```

0039
0040      REAL*4      wait_time      ! Number of seconds to wait
0041      REAL*4      ctime          ! Current time in total seconds
0042      REAL*4      rhours         ! Real form of chours
0043      REAL*4      rmin           ! Real form of cmin
0044      REAL*4      rsec           ! Real form of csec
0045
0046      CHARACTER*8  systime        ! Current system time
0047
0048
0049      C
0050      C      GET SYSTEM TIME
0051      C
0052
0053      CALL TIME(systime)
0054
0055      C
0056      C      SEPERATE TIME INTO INTEGER PARTS, CONVERT TO REAL, GET TOTAL SECONDS
0057      C
0058
0059      READ(systime(1:2),'(i2)')chours
0060      READ(systime(4:5),'(i2)')cmin
0061      READ(systime(7:8),'(i2)')csec
0062
0063      rsec = FLOATI(csec)
0064      rmin = FLOATI(cmin)
0065      rhours = FLOATI(chours)
0066
0067      ctime = rsec + (rmin * 60.0) + (rhours * 3600.0)
0068
0069      C
0070      C      FIND DELTA TIME AND CHECK
0071      C
0072
0073      1      CONTINUE
0074
0075      IF(SECONDS(ctime) .GT. wait_time) THEN
0076          CONTINUE
0077      ELSE
0078          GOTO 1
0079      ENDIF
0080
0081      RETURN
0082      END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	242	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	9	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	264

VARIABLES

Address	Type	Name	Attributes	References		
2-00000008	I+2	CHOURS		36	59=	65
2-0000000A	I+2	CMIN		37	60=	64
2-0000000C	I+2	CSEC		38	61=	63
2-00000010	R+4	CTIME		41	67=	75
**	R+4	RHOURS		42	65=	67
**	R+4	RMIN		43	64=	67
**	R+4	RSEC		44	63=	67
2-00000050	CHAR	SYSTIME		48	53A	59
AP-000000040	R+4	WAIT_TIME		26	46	75

LABELS

Address	Label	References
0-000000DC	1	73# 78

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	References
R+4	FOR\$SECONDS	76
	FOR\$TIME_T_DS	53

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/CROSS/CHECK=ALL/EXTEND 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 /NOD LINES /EXTEND_SOURCE /F77
/NOQ_FLOATING /I4 /NOMACHINE_CODE /OPTIMIZE
```

COMPILATION STATISTICS

```
Run Time:      2.18 seconds
Elapsed Time:  5.84 seconds
Page Faults:   559
Dynamic Memory: 334 pages
```

APPENDIX C

DESCRIPTION, FLOW CHART, AND CODE LISTINGS FOR THE TOWERQA PROGRAM

APPENDIX C

DESCRIPTION, FLOW CHART, AND CODE LISTINGS FOR THE TOWERQA PROGRAM

TOWERQA.FOR

TOWERQA is the main program that initiates interaction with the user. This code passes control to lower-level subroutines depending on responses to prompts by the user. The source codes are located in the [FCST.QA_PROGRAMS] directory at HMS. Figure C.1 shows a flow chart for the TOWERQA program. Figure C.2 shows the hierarchy of the subroutines used by TOWERQA.

The first thing TOWERQA does is clear the screen and display a message describing what the code does. The user is then given a menu of choices on how the program should proceed. The menu consists of two choices: process the 400-ft tower data, or process one of the 200-ft towers' data. Upon selection, TOWERQA passes control to the appropriate subroutine.

When these subroutines receive control, the user is then prompted for an input file name. After a valid file name is entered, the file is processed. A message is displayed telling the user the name of the output file. The user is then asked if another file of the same type needs to be processed. If so, control remains in that subroutine. If not, control is passed back to TOWERQA.

When control is passed back to TOWERQA, the user has the option to go back to the main menu and process another file, or to exit the program.

FIGURE C.1. Flow Chart for TOWERQA Program

FIGURE C.2. Subroutines Used by TOWERQA


```

0066      TYPE*, '***** BAD ENTRY *****'
0067      GOTO 1
0068      ENDIF
0069
0070      IF(foot .EQ. 1) CALL tower400
0071      IF(foot .EQ. 2) CALL tower200
0072      IF(foot .EQ. 3) GOTO 99
0073
0074      TYPE*
0075      WRITE(6,237)
0076 237    FORMAT(3x,'Would you like to go back to main menu? (y or n) : ',8)
0077      READ(5,'(a1)')more
0078      IF(( more .EQ. 'y' ) .OR. ( more .EQ. 'Y' )) GOTO 1
0079
0080 99     type*
0081      type*
0082      STOP ' Program Operation Complete '
0083
0084      END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	292	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	536	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	32	PIC COM REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	860	

ENTRY POINTS

Address	Type	Name	References
0-00000000		TOVERQASMAIN	

VARIABLES

Address	Type	Name	Attributes	References
2-00000004	I*4	FOOT		30 63= 65(2) 76 71 72
2-00000006	CHAR	MORE		41 77= 78(2)

LABELS

Address	Label	References
0-0000000C	1	49# 67 78
1-00000107	11'	66 67#
1-0000003C	22'	49 50#
0-000000F0	99	72 80#
1-000001DC	237'	75 76#

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	References
	TOVER200	71

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/CROSS/CHECK=ALL/EXTEND TOWERQA

/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

COMPILATION STATISTICS

Run Time: 2.37 seconds
 Elapsed Time: 7.19 seconds
 Page Faults: 270
 Dynamic Memory: 334 pages

```

0001  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002  C
0003  C   TOWER400.FOR
0004  C
0005  C   WRITTEN BY : KEN BURK
0006  C
0007  C   PURPOSE : TO PROVIDE QUALITY ASSURANCE CHECKS FOR THE HMS 400 FOOT TOWER
0008  C               AND THE 200 FOOT TOWERS IN THE HAMFORD AREA.
0009  C               ALL PARAMETER WILL BE CHECKED FOR LARGE HOURLY VARIATIONS, AND
0010  C               CHECKED AGAINST EXTREME VALUES.
0011  C               THE PARAMETERS ARE:
0012  C                   SOIL TEMPERATURES
0013  C                   ELEVATED TEMPERATURES (TOWER)
0014  C                   ELEVATED WIND SPEEDS (TOWER)
0015  C                   ELEVATED WIND DIRECTIONS (TOWER)
0016  C                   PEAK GUST DATA
0017  C
0018  C   ALGORITHM : DECLARE VARIABLES
0019  C               PROMPT AND READ INPUT FILE FROM USER
0020  C               OPEN INPUT AND OUTPUT FILE
0021  C               READ FROM INPUT FILE UNTIL END OF FILE
0022  C               PROCESS DATA BY MAKING CALLS TO SUBROUTINES
0023  C               IF NOT END OF FILE LOOP BACK
0024  C               ASK USER IF ANOTHER FILE
0025  C               END
0026  C
0027  C   VERSION : 3.0   October 1988
0028  C
0029  C   VERSION : 2.0   JAN 1988 (MODIFIED BY KEN BURK)
0030  C

```

```

0031 C   VERSION : 1.0   DEC 1984   (GREGG ANDREWS)
0032 C
0033 C   MAKES CALLS TO:  WAIT.FOR
0034 C                   SOIL TEMP.FOR
0035 C                   TOWER TEMP.FOR
0036 C                   WIND DATA.FOR
0037 C                   PEAK_GUST.FOR
0038 C
0039 C   CALLED FROM :    TOWERQA.FOR   (MAIN PROGRAM)
0040 C
0041 C-----
0042 C
0043 SUBROUTINE tower400
0044
0045 IMPLICIT NONE
0046
0047 C
0048 C   VARIABLE DECLARATION
0049 C-----
0050 C   TYPE      ! VARIABLE      ! COMMENTS
0051 C-----
0052 C
0053 INTEGER      oldtemp(11)      ! USED TO COMPARE CURRENT TEMP WITH PREVIOUS
0054 INTEGER      oldstemp(3)      ! USED TO COMPARE CURRENT SOIL TEMP WITH PREVIOUS HOUR
0055 INTEGER      oldwind(9)      ! USED TO COMPARE CURRENT WIND SPD & DIR WITH PREVIOUS
0056 INTEGER      i,l             ! ARRAY INDEX
0057 INTEGER      temp(11)        ! HOLDS THE ELEVEN INTEGER TEMPS
0058 INTEGER      wind(9)         ! HOLDS THE NINE WIND DIRECTIONS AND SPEEDS
0059 INTEGER      peaks           ! PEAK GUST SPEED
0060 INTEGER      peakd           ! PEAK GUST DIRECTION
0061 INTEGER      peakt           ! PEAK GUST TIME
0062
0063 CHARACTER*8   date           ! HOLDS YEAR,MONTH,DAY,HOUR
0064 CHARACTER*20  infile         ! INPUT FILE NAME
0065 CHARACTER*20  outfile        ! OUTPUT FILE NAME
0066 CHARACTER*1   more           ! ANSWER TO QUESTION OF ANOTHER RUN
0067 CHARACTER*2   yesterday      ! CHARACTER REP. OF PREVIOUS DAY
0068 CHARACTER*3   imonth         ! MONTH OF INPUT FILE
0069
0070 LOGICAL       first_rec      ! BOOLEAN TO TELL IF CURRENT REC IS THE FIRST
0071
0072 C
0073 C   PROMPT USER FOR INPUT FILE NAME, THEN OPEN THE INPUT
0074 C   AND OUTPUT FILES
0075 C
0076
0077 1  WRITE(6,22)
0078 22  FORMAT('///,
0079 *      8x,'*****',/,
0080 *      8x,'*      400 FOOT TOWER      ',/,
0081 *      8x,'*****',//)
0082 WRITE(6,11)
0083 11  FORMAT(3X,'Enter the name of the TOWER DATA input file,',/,
0084 *      3x,'Assume it is located at [FCST.DAT]: ',/,
0085 *      3x,'Assume the filename begins with "TOWER 400", and',/,
0086 *      3x,'ends with ".DAT". Enter only the MONTH (ex. OCT) --> ',3)
0087
0088 READ(5,'(A3)')imonth
0089
0090 DO 31 i = 1,3
0091 31  IF(imonth(i:i) .eq. ' ')GOTO 32
0092 32  i = i - 1
0093
0094 outfile = 'TOWER 400'//imonth(1:1)//'.QA.ERR'
0095 infile = 'TOWER 400'//imonth(1:1)//'.DAT'

```

```

0090
0097 OPEN(UNIT=2,FILE='[FCST.DAT] '//infile,STATUS='OLD',READONLY)
0098 OPEN(UNIT=3,FILE=outfile,STATUS='NEW',
0099 * CARRIAGECONTROL = 'LIST')
0100
0101 C
0102 C BEGIN THE QUALITY ASSURANCE PROCESS
0103 C
0104
0105 WRITE(6,66)outfile
0106 66 FORMAT(1x,/,3x,'Output File Name will be . . . ',a20,/,
0107 * /,3x,'Processing has now begun . . . ',/)
0108
0109 CALL wait(3.0)
0110
0111 WRITE(3,249)
0112 249 FORMAT(/,3x,'This file contains the actual data as well as any',/,
0113 * 3x,'Warning and/or Error Messages. The Messages will ',/,
0114 * 3x,'appear ABOVE the record in question.',/)
0115 WRITE(3,250)infile
0116 250 FORMAT(3X,' > > > ERROR LOG < < < FILENAME= ',A20,/)
0117 WRITE(3,251)
0118 251 FORMAT(9X,'Error',25X,'date',7X,'Data',/,3x,65('-'))
0119
0120 first_rec = .TRUE.
0121
0122 yesterday = '00'
0123
0124 2 READ(2,200,END=999)date,(temp(i),I=1,11),(wind(i),I=1,9),peakd,peaks,peakt
0125 200 FORMAT(A8,1X,11(I4),1X,11(I2),I1)
0126
0127 C
0128 C CREATE ARRAYS WITH PREVIOUS HOUR OF DATA, USED TO CHECK FOR
0129 C LARGE HOURLY VARIATION
0130 C
0131
0132 IF(date(5:8) .NE. yesterday) THEN ! PRINT THE DATE TO THE SCREEN
0133 WRITE(6,238)date ! ONLY WHEN THE DAY CHANGES
0134 238 FORMAT(3x,'Date : ',a8)
0135 yesterday = date(5:8)
0136 ENDIF
0137
0138 IF(first_rec) THEN ! IF CURRENT RECORD IS FIRST,
0139 DO i = 1,3 ! CREATE 'OLD' VALUES TO CHECK
0140 oldtemp(i) = temp(i) ! AGAINST
0141 ENDDO
0142 ENDIF ! SOIL TEMPERATURES
0143
0144 IF(first_rec) THEN
0145 DO i = 4,11
0146 oldtemp(i) = temp(i) ! TOWER TEMPERATURES
0147 ENDDO
0148 ENDIF
0149
0150 IF(first_rec) THEN
0151 DO i = 1,9
0152 oldwind(i) = wind(i) ! WIND SPEEDS AND DIRECTIONS
0153 ENDDO
0154 ENDIF
0155
0156
0157 C
0158 C CHECK SOIL TEMPERATURES
0159 C
0160 C

```

```

0161
0162      CALL soil_temp(temp,oldtemp,date)
0163
0164      C
0165      C      CHECK TOWER TEMPERATURES
0166      C
0167
0168      CALL tower_temp(temp,oldtemp,date)
0169
0170      C
0171      C      CHECK WIND SPEEDS
0172      C
0173
0174      CALL wind_data(wind,oldwind,date)
0175
0176
0177      C
0178      C      CHECK PEAK GUST DATA
0179      C
0180
0181      CALL peak_gust(peakd,peaks,peakt,date)
0182
0183
0184      first_rec = .FALSE.
0185
0186      DO i = 1,9                ! CREATE NEW PREVIOUS VALUE ARRAYS
0187          oldwind(i) = wind(i)  ! WIND SPEEDS AND DIRECTIONS
0188      ENDDO
0189
0190      DO i = 1,3
0191          oldtemp(i) = temp(i)
0192      ENDDO
0193
0194      DO i = 4,11
0195          oldtemp(i) = temp(i)  ! AIR TEMPERATURES
0196      ENDDO
0197
0198      C
0199      C      ECHO THE DATA TO THE OUTPUT FILE
0200      C
0201
0202      WRITE(3,200)date,(temp(i),I=1,11),(wind(i),I=1,9),peakd,peaks,peakt
0203      200  FORMAT(1X,A8,1X,11(I4),1X,11(I2),11)
0204
0205
0206      GOTO 2
0207
0208      999  CONTINUE
0209
0210      TYPE=
0211      WRITE(8,237)
0212      237  FORMAT(8X,'Would you like to do another 400 foot TOWER file? (y or n): ',8)
0213      READ(5,'(a1)')more
0214      IF(( more .EQ. 'y' ) .OR. ( more .EQ. 'Y' )) GOTO 1
0215
0216      CLOSE(2)
0217      CLOSE(3)
0218
0219      RETURN
0220      END

```

PROGRAM SECTIONS

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

# \$CODE	1882	PIC COM REL LCL SHR EXE	RD NOWRT LONG
1 \$PDATA	881	PIC COM REL LCL SHR NOEXE	RD NOWRT LONG
2 \$LOCAL	435	PIC COM REL LCL NOSHR NOEXE	RD WRT LONG

Total Space Allocated 2319

ENTRY POINTS

Address	Type	Name	References
0-00000000		TOWER400	435

VARIABLES

Address	Type	Name	Attributes	References
2-000000AC 174A	CHAR	DATE		63 124= 132 133 135 182A 188A
**	I=4	FIRST_REC	181A 282	78 126= 138 144 158 184=
**	I=4	I		58 96= 91(2) 92 124(4)= 139= 148(2)
145=				
			145(2) 151= 152(2) 188= 187(2) 198= 191(2) 194=	
2-000000DF	CHAR	IMONTH		68 88= 91 94 95
2-000000B4	CHAR	INFILE		64 95= 97 115
**	I=4	L		58 92= 94 95
2-000000DC	CHAR	MORE		68 213= 214(2)
2-000000C8	CHAR	OUTFILE		65 94= 98A 185
2-000000E8	I=4	PEAKD		68 124= 181A 282
2-000000E4	I=4	PEAKS		59 124= 181A 282
2-000000EC	I=4	PEAKT		61 124= 181A 282
2-000000DD	CHAR	YESTERDAY		67 122= 132 135=

ARRAYS

Address	Type	Name	Attributes	Bytes	Dimensions	References
2-0000002C	I=4	OLDSTMP		12	(3)	54 146= 182A 191=
2-00000000	I=4	OLDTEMP		44	(11)	53 146= 188A 195=
2-00000038	I=4	OLDWIND		38	(9)	55 152= 174A 187=
2-0000005C 182A	I=4	TEMP		44	(11)	57 124= 148 148
2-00000088 187	I=4	WIND		188A 36	191 (9)	195 282 58 124= 152 174A
				282		

LABELS

Address	Label	References
0-0000000C	1	775 214
0-000001A8	2	1245 286
1-0000007A	11'	82 835
1-00000004	22'	77 785
**	31	98 915

FUNCTIONS AND SUBROUTINES REFERENCED

0681

COMMAND QUALIFIERS

COMPILATION STATISTICS

[illegible]


```

0068
0069 1 WRITE(6,22)
0070 22 FORMAT(////,
0071 * 8x,'*****',/,
0072 * 8x,'* 200 FOOT TOWER *',/,
0073 * 8x,'*****',/))
0074 WRITE(6,11)
0075 11 FORMAT(3x,'Enter the name of the TOWER DATA input file',/,
0076 * 3x,'Assume it is located at [FCST.DAT]',/,
0077 * 3x,'Assume the filename begins with "TOWER 200", and',/,
0078 * 3x,'ends with ".DAT". Enter only which TOWER, and the MONTH',/,
0079 * 3x,'(ex. 100NOCT, or 300A0CT, or 400A0CT) -->',/)
0080
0081 READ(5,'(A7)')which
0082
0083 IF((which(1:4) .NE. '100n' .AND. which(1:4) .NE. '100N') .AND.
0084 * (which(1:4) .NE. '300n' .AND. which(1:4) .NE. '300A') .AND.
0085 * (which(1:4) .NE. '400n' .AND. which(1:4) .NE. '400A')) THEN
0086 type=
0087 type,'***** INVALID CHOICE *****'
0088 type=
0089 GOTO 1
0090 ENDIF
0091
0092 DO 31 i = 1,7
0093 31 IF(which(i:i) .eq. ' ')GOTO 32
0094 32 i = i - 1
0095
0096 outfile = 'TOWER 200'//which(1:1)//' QA.ERR'
0097 infile = 'TOWER_200_'//which(1:1)//'.DAT'
0098
0099 OPEN(UNIT=2,FILE='[FCST.DAT]'//infile,STATUS='OLD',READONLY)
0100 OPEN(UNIT=3,FILE=outfile,STATUS='NEW',
0101 * CARRIAGECONTROL = 'LIST')
0102
0103 C
0104 C BEGIN THE QUALITY ASSURANCE PROCESS
0105 C
0106
0107 WRITE(6,56)outfile
0108 56 FORMAT(1x,/,3x,'Output File Name will be . . .',a25,/,
0109 * /,3x,'Processing has now begun . . .',/)
0110
0111 CALL wait(3.0)
0112
0113 WRITE(3,249)
0114 249 FORMAT(//,3x,'This file contains the actual data as well as any',/,
0115 * 3x,'Warning and/or Error Messages. The Messages will',/,
0116 * 3x,'appear ABOVE the record in question.',/))
0117 WRITE(3,250)infile
0118 250 FORMAT(3x,' > > > ERROR LOG < < < FILENAME= ',a25,/)
0119 WRITE(3,251)
0120 251 FORMAT(3x,65('-'))
0121
0122 first_rec = .TRUE.
0123
0124 yesterday = '60'
0125
0126 2 READ(2,200,END=999)date,temps(1),(sigant(i),i=1,5),(temps(i),i=2,4),
0127 * (winds(i),i=1,7),temps(5),temps(6)
0128 200 FORMAT(A8,1X,9i4,8i2,3i4)
0129
0130 C
0131 C CREATE ARRAYS WITH PREVIOUS HOUR OF DATA, USED TO CHECK FOR
0132 C LARGE HOURLY VARIATION

```



```

0133 C
0134
0135 IF(date(5:8) .NE. yesterday) THEN ! PRINT THE DATE TO THE SCREEN
0136 WRITE(6,238)date ! ONLY WHEN THE DAY CHANGES
0137 238 FORMAT(3x,'Date : ',a8)
0138 yesterday = date(5:8)
0139 ENDIF
0140
0141 C
0142 C CHECK TEMPERATURES
0143 C
0144
0145 CALL temps_288(temps,date)
0146
0147 C
0148 C CHECK SIGMA THETAS AND SIGMA SPEEDS
0149 C
0150
0151 CALL sigma(sigma_t,date)
0152
0153 C
0154 C CHECK WIND DIRECTION AND SPEEDS
0155 C
0156
0157 CALL winds_288(winds,date)
0158
0159
0160 first_rec = .FALSE.
0161
0162 C
0163 C ECHO THE DATA TO THE OUTPUT FILE
0164 C
0165
0166 WRITE(3,299)date,temps(1),(sigma_t(i),i=1,5),(temps(i),i=2,4),
0167 * (winds(i),i=1,7),temps(5),temps(8)
0168 299 FORMAT(1X,A8,1X,8I4,6I2,3I4)
0169
0170
0171 GOTO 2
0172
0173 999 CONTINUE
0174
0175 TYPE=
0176 WRITE(6,237)
0177 237 FORMAT(3x,'Would you like to do another 288 foot TOWER file? (y or n): ',a)
0178 READ(5,'(a1)')more
0179 IF(( more .EQ. 'y' ) .OR. ( more .EQ. 'Y' )) GOTO 1
0180
0181 CLOSE(2)
0182 CLOSE(3)
0183
0184 RETURN
0185 END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	1698	PIC COM REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	878	PIC COM REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	312	PIC COM REL LCL NDSHR NOEXE RD WRT LONG

Total Space Allocated

2286

ENTRY POINTS

Address	Type	Name	References
0-00000000		TOWER200	48#

VARIABLES

Address	Type	Name	Attributes	References
2-00000048 167A	CHAR	DATE		55 128= 135 136 138 145A 151A
**	L+4	FIRST_REC	100	
**	I+4	I		62 122= 168=
2-00000058	CHAR	INFILE		53 92= 93(2) 94 128(8)= 168(8)=
**	I+4	L		56 97= 99 117
**	I+4	L		53 94= 98 97
2-00000082	CHAR	MORE		56 178= 179(2)
2-00000089	CHAR	OUTFILE		57 98= 100A 107
2-00000085	CHAR	WHICH		58 81= 83(8) 93 98 97
2-00000083	CHAR	YESTERDAY		59 124= 135 138=

ARRAYS

Address	Type	Name	Attributes	Bytes	Dimensions	References
2-00000034	I+4	SIGMAT		20	(5)	52 128= 151A 168
2-00000000	I+4	TEMPS		24	(8)	50 128(4)= 145A 168(4)
2-00000018	I+4	WINDS		28	(7)	51 128= 157A 168

LABELS

Address	Label	References
0-0000001C	1	69# 80 179
0-00000026	2	126# 171
1-0000009A	11'	74 76#
1-00000024	22'	89 78#
**	31	92 93#
0-00000012D	32	95 94#
1-00000019A	55'	107 108#
1-0000002BC	200'	126 128#
1-0000002EE	237'	178 177#
1-0000002CD	238'	138 137#
1-0000001E8	249'	113 114#
1-000000282	250'	117 118#
1-0000002B1	251'	119 120#
1-0000002D8	299'	160 168#
0-0000003DC	999	120 173#

FUNCTIONS AND SUBROUTINES REFERENCED

0001

COMMAND QUALIFIERS

COMPILATION STATISTICS

```

0001 C
0002 C
0003 C      SOIL_TEMP.FOR
0004 C
0005 C      WRITTEN BY : KEN BURK
0006 C                GREGG ANDREWS
0007 C
0008 C      PURPOSE :      PROVIDE A QUALITY ASSURANCE CHECK ON THE SOIL
0009 C                    TEMPERATURES THAT ARE FOUND IN THE HWS TOWER DATA.
0010 C
0011 C      ALGORITHM :    DECLARE VARIABLES
0012 C                    CHECK EACH LEVEL FOR EXTREMES
0013 C                    CHECK DIFFERENCE IN LEVEL 2 & 3
0014 C                    CHECK FOR LARGE DIFFERENCE IN HOURS
0015 C                    WHEN DONE PUT CURRENT TEMPERATURES INTO
0016 C                      OLDTEMP
0017 C                    RETURN
0018 C
0019 C      VERSION : 1.0
0020 C

```



```

0080 C
0087 C CHECK HOURLY DIFFERENCE, SHOULD BE LESS THAN 20.0 DEGREES
0088 C
0089
0090 DO i = 1,3
0091     diff = JIABS (oldsoil(i) - soil(i))
0092     IF (diff .GT. 20.0 .AND. i .EQ. 1) THEN
0093         IF ((oldsoil(i) .NE. 9999 .AND. oldsoil(i) .NE. 0) .AND.
0094             * (soil(i) .NE. 9999 .AND. soil(i) .NE. 0)) THEN
0095             WRITE(3,204) yymddhh, soil(i), oldsoil(i)
0096         ENDIF
0097     ENDIF
0098     IF (diff .GT. 50 .AND. i .GE. 2) THEN
0099         IF ((oldsoil(i) .NE. 9999 .AND. oldsoil(i) .NE. 0) .AND.
0100             * (soil(i) .NE. 9999 .AND. soil(i) .NE. 0)) THEN
0101             WRITE(3,204) yymddhh, soil(i), oldsoil(i)
0102             204 FORMAT(/,3x,'LARGE SOIL TEMP HOURLY DIFF',6x,a0,2x,2i5)
0103         ENDIF
0104     ENDIF
0105 ENDDO
0106
0107 RETURN
0108 END

```

! CHECK FOR A
! LARGE TEMPERATURE
! DIFFERENCE BETWEEN
! HOURS.

! LEVELS
! -15.0 IN.
! AND -30.0 IN.

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	097	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	215	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	48	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	960	

ENTRY POINTS

Address	Type	Name	References
0-00000000		SOIL_TEMP	274

VARIABLES

Address	Type	Name	Attributes	References
**	I+4	DIFF		41 91= 92 98
**	I+4	I		45 90= 91(2) 92 93(4) 95(2) 98
99(4)			101(2)	
AP-00000000	CHAR	YYMMDDHH		27 43 54 61 68 81 95
101				

ARRAYS

Address	Type	Name	Attributes	Bytes	Dimensions	References
AP-00000000	I+4	OLDSOIL		12	(3)	27 39 91 93(2)
95				99(2)	101	
AP-00000040	I+4	SOIL		44	(11)	27 38 52(2) 53
54				59(2) 60(2) 61 66(2) 67(2)		
				68 78(2) 79(4) 81(2) 91		

Address	Label	References
1-00000000	200'	54 55#
1-00000028	201'	61 62#
1-00000052	202'	68 69#
1-0000007C	203'	81 82#
1-000000AC	204'	96 101 102#

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/CROSS/CHECK=ALL/EXTEND SOIL TEMP

```

/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

```

COMPILATION STATISTICS

```
Run Time:      4.10 seconds
Elapsed Time:  7.59 seconds
Page Faults:   557
Dynamic Memory: 386 pages
```

```

0001 C
0002 C
0003 C      TOWER_TEMP.FOR
0004 C
0005 C      WRITTEN BY : KEN BURK
0006 C                  GREGG ANDREWS
0007 C
0008 C      PURPOSE :      PROVIDE QUALITY ASSURANCE FOR THE TOWER TEMPERATURES
0009 C                  FOUND IN THE HMS TOWER DATA
0010 C
0011 C      ALGORITHM :  DECLARE VARIABLES
0012 C                  CHECK TEMPERATURES WITH MONTHLY EXTREMES
0013 C                  CHECK PREVIOUS TEMPERATURE DIFFERENCE
0014 C                  CHECK DIFFERENCE BETWEEN LEVELS
0015 C                  RETURN
0016 C
0017 C      VERSION : 1.0
0018 C
0019 C      DATE :      MAY 1988

```



```

0085
0086      DO i = 5,11
0087          IF(JIABS(tower(i) - tower(i-1)) .GT. 150) THEN
0088              IF(( tower(i) .NE. 999) .AND. (tower(i-1) .NE. 999)) THEN
0089                  WRITE(3,303)yyymmddhh,tower(i),tower(i-1)
0090              ENDIF
0091          ENDIF
0092      303      FORMAT(/,3x,'LARGE TEMP. LEVEL DIFF.',10X,A8,2X,I4,2X,I4)
0093      ENDDO
0094
0095
0096      RETURN
0097      END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	508	PIC COM REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	128	PIC COM REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	156	PIC COM REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	786	

ENTRY POINTS

Address	Type	Name	References
0-00000000		TOWER_TEMP	250

VARIABLES

Address	Type	Name	Attributes	References
**	I*4	DIFF		41 73= 74
**	I*4	I		40 58= 59(2) 61 62 72= 73(2)
75(2)				
2-00000000	I*4	MONTH	78 88=	87(2) 88(2) 89(2)
AP-00000000	CHAR	YYMMDDHH		42 50= 59(2)
				25 44 58 62 76 89

ARRAYS

Address	Type	Name	Attributes	Bytes	Dimensions	References
2-00000000	I*4	MAXTEMP		48	(12)	38 480 59
2-00000030	I*4	MINTEMP		48	(12)	39 480 59
AP-00000000	I*4	OLDTOWER		44	(11)	25 37 73 75
AP-00000040	I*4	TOWER		44	(11)	25 38 59(2) 61
62				73 89(2)	75 76 87(2) 88(2)	

LABELS

Address	Label	References
1-00000003	301'	62 830
1-0000002A	302'	76 770
1-00000055	303'	89 920

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/CROSS/CHECK=ALL/EXTEND TOWER_TEMP
/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 /FT7
/NOG FLOATING /I4 /NOMACHINE CODE /OPTIMIZE
```

COMPILATION STATISTICS

```
Run Time:      3.84 seconds
Elapsed Time:  8.44 seconds
Page Faults:   671
Dynamic Memory: 359 pages
```

```

0001 C
0002 C
0003 C      WIND_DATA.FOR
0004 C
0005 C      WRITTEN BY : KEN BURK
0006 C                GREGG ANDREWS
0007 C
0008 C      PURPOSE :   TO PROVIDE A QUALITY ASSURANCE CHECK ON THE
0009 C                  WIND SPEEDS AND DIRECTIONS THAT ARE FOUND IN THE
0010 C                  HMS TOWER DATA
0011 C
0012 C      ALGORITHM :  VARIABLE DECLARATION
0013 C                  CHECK WIND SPEEDS FOR 0-50 INTERVAL
0014 C                  CHECK HOURLY VARIATION
0015 C                  CHECK FOR VALID WIND DIRECTIONS
0016 C                  IF WIND SPEED = 0
0017 C                      CHECK FOR WIND DIRECTION = 0
0018 C                  CHECK DIRECTIONAL DIFFERENCE BETWEEN LEVELS
0019 C                  RETURN
0020 C
0021 C      VERSION : 1.0
0022 C
0023 C      DATE :   MAY 1986
0024 C
0025 C      CALLED FROM :  TOWERQA.FOR (MAIN PROGRAM)
0026 C
0027 C
0028 C
0029 C      SUBROUTINE wind_data (dirspd,olddirspd,yyomddhh)
0030 C
0031 C      IMPLICIT NONE
0032 C

```

```

0033 C
0034 C   VARIABLE DECLARATIONS
0035 C -----
0036 C   TYPE      ! VARIABLE      ! COMMENTS
0037 C -----
0038 C
0039 C
0040 INTEGER      dirspd(9)      ! Array for wind directions and speeds
0041 INTEGER      olddirspd(9)   ! Previous hour of speed and directions
0042 INTEGER      i              ! Loop index
0043
0044 CHARACTER*8   yymaddhh      ! Year, month, day, hour
0045
0046
0047 C
0048 C   CHECK WIND SPEED FOR UPPER AND LOWER EXTREMES
0049 C
0050
0051 IF((dirspd(1) .GT. 50 .OR. dirspd(1) .LT. 0) .AND.
0052 * (dirspd(1) .NE. 99)) WRITE(3,401) yymaddhh, dirspd(1)
0053
0054 IF((dirspd(3) .GT. 50 .OR. dirspd(3) .LT. 0) .AND.
0055 * (dirspd(3) .NE. 99)) WRITE(3,401) yymaddhh, dirspd(3)
0056
0057 IF((dirspd(4) .GT. 50 .OR. dirspd(4) .LT. 0) .AND.
0058 * (dirspd(4) .NE. 99)) WRITE(3,401) yymaddhh, dirspd(4)
0059
0060 IF((dirspd(6) .GT. 50 .OR. dirspd(6) .LT. 0) .AND.
0061 * (dirspd(6) .NE. 99)) WRITE(3,401) yymaddhh, dirspd(6)
0062
0063 IF((dirspd(7) .GT. 50 .OR. dirspd(7) .LT. 0) .AND.
0064 * (dirspd(7) .NE. 99)) WRITE(3,401) yymaddhh, dirspd(7)
0065
0066 IF((dirspd(9) .GT. 50 .OR. dirspd(9) .LT. 0) .AND.
0067 * (dirspd(9) .NE. 99)) WRITE(3,401) yymaddhh, dirspd(9)
0068
0069 401   FORMAT(/,3X,'WIND SPEED OUT OF RANGE ',9X,A8,2X,I3)
0070
0071 C
0072 C   CHECK FOR LARGE CHANGE IN WIND SPEED SINCE LAST HOUR
0073 C
0074
0075 DO i = 1,9
0076   IF(i .EQ. 1 .OR. i .EQ. 3 .OR. i .EQ. 4 .OR.
0077 *   i .EQ. 6 .OR. i .EQ. 7 .OR. i .EQ. 9) THEN
0078
0079     IF((JABS(dirspd(i) - olddirspd(i)) .GT. 20) THEN
0080       IF((dirspd(i) .NE. 99) .AND. (olddirspd(i) .NE. 99)) THEN
0081         WRITE(3,402) yymaddhh, dirspd(i), olddirspd(i)
0082       ENDIF
0083     ENDIF
0084
0085   ENDIF
0086 END DO
0087
0088 402   FORMAT(/,3X,'LARGE HOURLY WIND SPEED DIFF ',2X,A8,2X,6I4)
0089
0090 C
0091 C   CHECK DIFFERENCE BETWEEN LEVELS FOR WIND SPEED > 10
0092 C
0093
0094 IF(JABS(dirspd(1) - dirspd(3)) .GT. 10) THEN
0095   IF((dirspd(1) .NE. 999) .AND. (dirspd(3) .NE. 999)) THEN
0096     WRITE(3,400) yymaddhh, dirspd(1), dirspd(3)
0097   ENDIF

```

```

0098      ENDIF
0099
0100      IF(JIABS(dirspd(3) - dirspd(4)) .GT. 10) THEN
0101          IF((dirspd(3) .NE. 999) .AND. (dirspd(4) .NE. 999)) THEN
0102              WRITE(3,486)yyymmddhh,dirspd(3),dirspd(4)
0103          ENDIF
0104      ENDIF
0105
0106      IF(JIABS(dirspd(4) - dirspd(5)) .GT. 10) THEN
0107          IF((dirspd(4) .NE. 999) .AND. (dirspd(5) .NE. 999)) THEN
0108              WRITE(3,486)yyymmddhh,dirspd(4),dirspd(5)
0109          ENDIF
0110      ENDIF
0111
0112      IF(JIABS(dirspd(5) - dirspd(6)) .GT. 10) THEN
0113          IF((dirspd(5) .NE. 999) .AND. (dirspd(6) .NE. 999)) THEN
0114              WRITE(3,486)yyymmddhh,dirspd(5),dirspd(6)
0115          ENDIF
0116      ENDIF
0117
0118      IF(JIABS(dirspd(6) - dirspd(7)) .GT. 10) THEN
0119          IF((dirspd(6) .NE. 999) .AND. (dirspd(7) .NE. 999)) THEN
0120              WRITE(3,486)yyymmddhh,dirspd(6),dirspd(7)
0121          ENDIF
0122      ENDIF
0123
0124      486  FORMAT(/,3X,'LARGE WIND SPEED LEVEL DIFF.',5X,A8,2X,2I4)
0125
0126      C
0127      C      CHECK WIND DIRECTIONS FOR VALIDITY
0128      C
0129
0130      IF((dirspd(2) .GT. 37 .OR. dirspd(2) .LT. 0)) THEN
0131          IF(dirspd(2) .NE. 99) THEN
0132              WRITE(3,483)yyymmddhh,dirspd(2)
0133          ENDIF
0134      ENDIF
0135
0136      IF((dirspd(5) .GT. 37 .OR. dirspd(5) .LT. 0)) THEN
0137          IF(dirspd(5) .NE. 99) THEN
0138              WRITE(3,483)yyymmddhh,dirspd(5)
0139          ENDIF
0140      ENDIF
0141
0142      IF((dirspd(8) .GT. 37 .OR. dirspd(8) .LT. 0)) THEN
0143          IF(dirspd(8) .NE. 99) THEN
0144              WRITE(3,483)yyymmddhh,dirspd(8)
0145          ENDIF
0146      ENDIF
0147
0148      483  FORMAT(/,3X,'WIND DIRECTION OUT OF RANGE',6X,A8,2X,3I3)
0149
0150      C
0151      C      CHECK IF WIND SPEED = 0 THEN DIRECTION = 0
0152      C
0153
0154      IF((dirspd(2) .EQ. 0 .AND. dirspd(3) .NE. 0) .OR.
0155      * (dirspd(5) .EQ. 0 .AND. dirspd(6) .NE. 0) .OR.
0156      * (dirspd(8) .EQ. 0 .AND. dirspd(9) .NE. 0))
0157      * WRITE(3,484)yyymmddhh,dirspd(2),dirspd(3),dirspd(5),dirspd(6),dirspd(8),dirspd(9)
0158
0159      C
0160      C      CHECK IF WIND DIRECTION = 0 THEN WIND SPEED SHOULD BE 0
0161      C
0162
0163      IF((dirspd(3) .EQ. 0 .AND. dirspd(2) .NE. 0) .OR.
0164      * (dirspd(6) .EQ. 0 .AND. dirspd(5) .NE. 0) .OR.

```

```

0163      * (dirspd(9) .EQ. 0 .AND. dirspd(8) .NE. 0))
0164      * WRITE(3,484)yyymmddhh,dirspd(2),dirspd(3),dirspd(5),dirspd(6),dirspd(8),dirspd(9)
0165
0166      C
0167      C      CHECK IF WIND DIRECTION = 27 THEN WIND SPEED SHOULD BE <= 5
0168      C
0169
0170      IF((dirspd(2) .EQ. 37 .AND. dirspd(3) .GT. 5) .OR.
0171      * (dirspd(5) .EQ. 37 .AND. dirspd(6) .GT. 5) .OR.
0172      * (dirspd(8) .EQ. 37 .AND. dirspd(9) .GT. 5))
0173      * WRITE(3,484)yyymmddhh,dirspd(2),dirspd(3),dirspd(5),dirspd(6),dirspd(8),dirspd(9)
0174
0175      484      FORMAT(/,3X,'WIND DIRECTION/SPEED CONFLICT ',3X,A8,2X,3(' ',I2,' ',I2,' ');',1X))
0176
0177      C
0178      C      CHECK DIFFERENCE BETWEEN LEVELS FOR WIND DIRECTION > 40 DEGREES
0179      C
0180
0181      IF((JABS(dirspd(2) - dirspd(5)) .GT. 40) THEN
0182      IF((dirspd(2) .NE. 99) .AND. (dirspd(5) .NE. 99)) THEN
0183      WRITE(3,485)yyymmddhh,dirspd(2),dirspd(5)
0184      ENDIF
0185      ENDIF
0186
0187      IF((JABS(dirspd(5) - dirspd(8)) .GT. 40) THEN
0188      IF((dirspd(5) .NE. 99) .AND. (dirspd(8) .NE. 99)) THEN
0189      WRITE(3,485)yyymmddhh,dirspd(5),dirspd(8)
0190      ENDIF
0191      ENDIF
0192
0193      485      FORMAT(/,3X,'LARGE WIND DIR. LEVEL DIFF.',6X,A8,2X,2I4)
0194
0195
0196      RETURN
0197      END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	1799	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	281	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	48	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	2128	

ENTRY POINTS

Address	Type	Name	References
0-00000000		WIND_DATA	204

VARIABLES

Address	Type	Name	Attributes	References
**	I=4 I			42 75= 76(8) 79(2) 80(2) 81(2)
AP-00000000	CHAR	YYMMDDHH		20 44 51 54 57 60 63
00			81 96 102 108 114 120 132 137	

142 152 161 170 183 189

ARRAYS

Address	Type	Name	Attributes	Bytes	Dimensions	References			
AP-000000040	I*4	DIRSPD		36	(9)	20	40	51(4)	54(4)
57(4)				60(4)	63(4)	66(4)	79	80	
				81	94(2)	95(2)	98(2)	100(2)	
				101(2)	102(2)	106(2)	107(2)	108(2)	
				112(2)	113(2)	114(2)	118(2)	119(2)	
				120(2)	130(2)	131	132	135(2)	
				136	137	140(2)	141	142	
				152(12)	161(12)	170(12)	181(2)	182(2)	
				183(2)	187(2)	188(2)	189(2)		
AP-000000080	I*4	OLDDIRSPD		36	(9)	20	41	79	80
81									

LABELS

Address	Label	References						
1-000000000	401'	51	54	57	60	63	66	69#
1-000000020	402'	81	88#					
1-000000081	403'	132	137	142	146#			
1-0000000AC	404'	152	161	170	175#			
1-0000000EE	405'	183	189	193#				
1-000000055	406'	95	102	100	114	120	124#	

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/CROSS/CHECK=ALL/EXTEND WIND_DATA
/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)
/DEBUG=(NOSYMBOLS,TRACEBACK)
/STANDARD=(NOSYNTAX,NOSOURCE FORM)
/SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP,NODICTIONARY,SINGLE)
/WARNINGS=(GENERAL,NODECLARATIONS,NOLTRIX)
/CONTINUATIONS=10 /CROSS REFERENCE /NOD LINES /EXTEND_SOURCE /F77
/NOG_FLOATING /I4 /NOMACHINE_CODE /OPTIMIZE
```

COMPILATION STATISTICS

```
Run Time: 9.64 seconds
Elapsed Time: 17.52 seconds
Page Faults: 692
Dynamic Memory: 474 pages
```

```

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 C
0003 C      PEAK_GUST.FOR
0004 C
0005 C      WRITTEN BY : KEN BURK
0006 C                  GREGG ANDREWS
0007 C
0008 C      PURPOSE :   TO PROVIDE A QUALITY ASSURANCE CHECK ON THE
0009 C                  PEAK GUST DATA FOUND IN THE HMS TOWER DATA
0010 C
0011 C      ALGORITHM :  VARIABLE DECLARATION
0012 C                  CHECK THE DIRECTION FOR VALIDITY
0013 C                  CHECK SPEED FOR 00-15 RANGE
0014 C                  CHECK TIME OF GUST 0-9
0015 C                  RETURN
0016 C
0017 C      VERSION : 1.0
0018 C
0019 C      DATE :   MAY 1988
0020 C
0021 C      CALLED FROM :  TOWERQA.FOR (MAIN PROGRAM)
0022 C
0023 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0024 C
0025 C      SUBROUTINE peak_gust (gustd,gusta,gustt,yyymmddhh)
0026 C
0027 C      IMPLICIT NONE
0028 C
0029 C
0030 C      VARIABLE DECLARATIONS
0031 C-----
0032 C      TYPE      !  VARIABLE      !  COMMENTS
0033 C-----
0034 C
0035 C
0036 C      INTEGER      gustd      ! Direction of peak gust
0037 C      INTEGER      gusta      ! Speed of peak gust
0038 C      INTEGER      gustt      ! Time of peak gust
0039 C
0040 C      CHARACTER*8   yyymmddhh ! Year, month, day, hour
0041 C
0042 C
0043 C
0044 C      CHECK THE DIRECTION
0045 C
0046 C
0047 C      IF(gustd .GT. 360 .OR. gustd .LT. 0) THEN
0048 C          IF(gustd .NE. 999) THEN
0049 C              WRITE(3,501)yyymmddhh,gustd
0050 C          ENDIF
0051 501  FORMAT(/,3X,'PEAK GUST DIRECTION ERROR',8X,A8,2X,I3)
0052 C          ENDIF
0053 C
0054 C
0055 C      CHECK THE SPEED
0056 C
0057 C
0058 C      IF((gusta .GT. 00) .OR. (gusta .LT. 15 )) THEN
0059 C          IF((gusta .NE. 99) .AND. (gusta .NE. 0)) THEN
0060 C              WRITE(3,502)yyymmddhh,gusta
0061 502  FORMAT(/,3X,'PEAK GUST SPEED ERROR',12X,A8,2X,I3)
0062 C          ENDIF
0063 C          ENDIF
0064 C
0065 C

```

```

0066 C CHECK THE TIME
0067 C
0068
0069 IF(guatt .LT. 0 .OR. guatt .GT. 0) THEN
0070 WRITE(8,503)ymmdhh,guatt
0071 503 FORMAT(/,3X,'PEAK GUST TIME ERROR',13X,A8,2X,I3)
0072 ENDIF
0073
0074
0075 RETURN
0076 END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	254	PIC COM REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	108	PIC COM REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	8	PIC COM REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated		320

ENTRY POINTS

Address	Type	Name	References
0-00000000		PEAK_GUST	25#

VARIABLES

Address	Type	Name	Attributes	References
AP-00000040	I=4	GUSTD		25 38 47(2) 48 49
AP-00000080	I=4	GUSTS		25 37 58(2) 59(2) 60
AP-000000C0	I=4	GUSTT		25 38 59(2) 70
AP-00000100	CHAR	YMMDDHH		25 48 49 66 76

LABELS

Address	Label	References
1-00000000	501'	49 51#
1-00000027	502'	60 61#
1-0000004A	503'	76 71#

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/CROSS/CHECK=ALL/EXTEND PEAK_GLST

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

COMPILATION STATISTICS

Run Time: 1.88 seconds
 Elapsed Time: 3.76 seconds
 Page Faults: 486
 Dynamic Memory: 326 pages

```

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 C
0003 C      TEMPS_200.FOR
0004 C
0005 C      WRITTEN BY: KEN BURK
0006 C
0007 C      PURPOSE : TO PROVIDE QUALITY ASSURANCE CHECK FOR A PORTION OF THE
0008 C                  HMS 200 FOOT TOWER DATA. THESE PARAMETERS WILL BE CHECKED
0009 C                  TO MAKE SURE THEY FIT WITHIN A DETERMINED RANGE.
0010 C                  THE PARAMETERS ARE:
0011 C                      DEM POINT
0012 C                      2M TEMP
0013 C                      10M TEMP
0014 C                      60M TEMP
0015 C                      MAX TEMP
0016 C                      MIN TEMP
0017 C
0018 C      ALGORITHM : DECLARE VARIABLES
0019 C                  CHECK TOWER TEMPS
0020 C                  RETURN
0021 C                  END
0022 C
0023 C      VERSION : 1.0  OCTOBER 1988
0024 C
0025 C      CALLED FROM :  TOWER200.FOR
0026 C
0027 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0028 C
0029 C      SUBROUTINE temps_200(temps,date)
0030 C
0031 C      IMPLICIT NONE
0032 C
0033 C
0034 C      VARIABLE DECLARATIONS
0035 C-----
0036 C      TYPE      !  VARIABLE      !  COMMENTS
0037 C-----
0038 C
0039 C
0040 C      INTEGER    temps(6)          ! TOWER TEMPERATURES
0041 C      INTEGER    i,j,k             ! ARRAY AND LOOP INDICES
0042 C      INTEGER    maxtemps(12)      ! MAX TEMPS FOR EACH MONTH
0043 C      INTEGER    mintemps(12)      ! MIN TEMPS FOR EACH MONTH
0044 C      INTEGER    month             ! INTEGER MONTH
0045 C

```



```

0046 CHARACTER*8      date          ! DATE YY MM DD HH
0047 CHARACTER*28     msg(6)        ! ERROR MESSAGES
0048
0049 LOGICAL           error         ! BOOLEAN FOR ERROR OR NOERROR
0050
0051
0052 DATA maxtemps / 620, 620, 730, 850, 940, 1000, 1050, 1030, 960, 800, 650, 590 /
0053 DATA mintemps / -130, -130, 100, 310, 380, 430, 490, 500, 350, 100, -30, -170 /
0054
0055 DATA msg / 'DEW POINT TEMP. OUT OF RANGE',
0056 *          'TEMP OUT OF RANGE. 2 METER ',
0057 *          'TEMP OUT OF RANGE. 10 METER ',
0058 *          'TEMP OUT OF RANGE. 80 METER ',
0059 *          'MAXIMUM TEMP OUT OF RANGE ',
0060 *          'MINIMUM TEMP OUT OF RANGE ' /
0061
0062 error = .FALSE.
0063
0064 READ(date(3:4), '(i2)') month
0065
0066 C
0067 C BEGIN CHECK
0068 C
0069
0070 DO i = 1,6
0071
0072 IF((temps(i) .GT. maxtemps(month) .OR.
0073 *   temps(i) .LT. mintemps(month)) .AND. (temps(i) .NE. 9999)) THEN
0074
0075 IF(error) THEN
0076 WRITE(3,507)msg(i),date,temps(i)
0077 507 FORMAT(2x,a20,3x,'DATE ',a8,5x,i4)
0078 ELSE
0079 error = .TRUE.
0080 WRITE(3,508)msg(i),date,temps(i)
0081 508 FORMAT(/,2x,a20,3x,'DATE ',a8,5x,i4)
0082 ENDIF
0083
0084 ENDIF
0085
0086 ENDDO
0087
0088 RETURN
0089 END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	351	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	44	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	384	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	899	

ENTRY POINTS

Address	Type	Name	References
0-00000000		TEMPS_200	200

VARIABLES

ARRAYS

LABELS

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

FOR/LIS/CROSS/CHECK=ALL/EXTEND TEMPS 200

COMPILATION STATISTICS

[illegible]

C.32

```

0069      IF((winds(i) .LT. 0 .OR. winds(i) .GT. 60) .AND.
0070      * (winds(i) .NE. 99)) THEN
0071      WRITE(3,502)date,winds(i)
0072 502      FORMAT(/,2x,'WIND SPEED OUT OF RANGE   DATE: ',A8,5X,I3)
0073      GOTO 2
0074      ENDIF
0075      ENDDO
0076 2.      CONTINUE
0077
0078      C
0079      C      CHECK MAXIMUM WIND SPEED
0080      C
0081      IF((winds(7) .LT. 1 .OR. winds(7) .GT. 75) .AND.
0082      * (winds(7) .NE. 9999)) THEN
0083      WRITE(3,503)date,winds(7)
0084 503      FORMAT(/,2x,'MAXIMUM WIND SPEED OUT OF RANGE   DATE: ',A8,5X,I4)
0085      ENDIF
0086
0087      RETURN
0088      END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	314	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 \$PODATA	146	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	28	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	487	

ENTRY POINTS

Address	Type	Name	References
0-00000000		WINDS_200	36#

VARIABLES

Address	Type	Name	Attributes	References
AP-00000000	CHAR	DATE		36 50 58 71 83
**	I=4	I		48 55= 56(3) 58 68= 69(3) 71

ARRAYS

Address	Type	Name	Attributes	Bytes	Dimensions	References
AP-00000040	I=4	WINDS		28	(7)	36 47 58(3) 68
69(3)				71	81(3)	83

LABELS

Address	Label	References
0-0000008A	1	60 63#
0-000000F6	2	73 76#
1-00000000	501'	50 50#
1-00000030	502'	71 72#

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/CROSS/CHECK=ALL/EXTEND WINDS_200

/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 /NOB LINES /EXTEND_SOURCE /F77

/NOG_FLOATING /I4 /NOMACHINE_CODE /OPTIMIZE

COMPILATION STATISTICS

Run Time: 2.32 seconds

Elapsed Time: 3.00 seconds

Page Faults: 505

Dynamic Memory: 330 pages

```

0001  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002  C
0003  C      SIGMA.FOR
0004  C
0005  C      WRITTEN BY :  KEN BURK
0006  C
0007  C      PURPOSE :  TO PROVIDE QUALITY ASSURANCE CHECKS ON A FEW OF THE
0008  C                  PARAMETERS FOUND IN THE 200 FOOT TOWER DATA COLLECTED AT
0009  C                  HMS.  THESE PARAMETERS ARE:
0010  C                  DELTA TEMPERATURE
0011  C                  10 METER SIGMA THETA
0012  C                  10 METER SIGMA SPEED
0013  C                  60 METER SIGMA THETA
0014  C                  60 METER SIGMA SPEED
0015  C
0016  C      ALGORITHM :  DECLARE VARIABLES
0017  C                  CHECK DELTA TEMP
0018  C                  CHECK 10 METER SIGMA THETA
0019  C                  CHECK 10 METER SIGMA SPEED
0020  C                  CHECK 60 METER SIGMA THETA
0021  C                  CHECK 60 METER SIGMA SPEED
0022  C                  RETURN
0023  C                  END
0024  C
0025  C      VERSION :  1.0  OCTOBER 1988
0026  C
0027  C      CALLED FROM :  TOWER200.FOR
0028  C
0029  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0030

```

```

0031      SUBROUTINE sigma(sigmat,date)
0032
0033      IMPLICIT NONE
0034
0035      C
0036      C      VARIABLE DECLARATIONS
0037      C-----
0038      C      TYPE      ! VARIABLE      ! COMMENTS
0039      C-----
0040      C
0041
0042      INTEGER      sigmat(5)      ! VALUES PASSED INTO THIS ROUTINE
0043
0044      CHARACTER*8   date          ! DATE YY,MM,DD,HH
0045
0046      C
0047      C      CHECK DELTA TEMPERATURE
0048      C
0049      IF((sigmat(1) .GT. 150 .OR. sigmat(1) .LT. -100) .AND.
0050      * (sigmat(1) .NE. 9999)) THEN
0051      WRITE(3,401)date,sigmat(1)
0052      401  FORMAT(/,2x,'DELTA TEMP. OUT OF RANGE   DATE: ',a8,5x,i4)
0053      ENDIF
0054
0055      C
0056      C      CHECK 10 METER SIGMA THETA
0057      C
0058      IF((sigmat(2) .GT. 80 .OR. sigmat(2) .LT. 0) .AND.
0059      * (sigmat(2) .NE. 9999)) THEN
0060      WRITE(3,402)date,sigmat(2)
0061      402  FORMAT(/,2x,'10 METER SIGMA THETA OUT OF RANGE   DATE: ',a8,5x,i4)
0062      ENDIF
0063
0064      C
0065      C      CHECK 60 METER SIGMA THETA
0066      C
0067      IF((sigmat(4) .GT. 80 .OR. sigmat(4) .LT. 0) .AND.
0068      * (sigmat(4) .NE. 9999)) THEN
0069      WRITE(3,403)date,sigmat(4)
0070      403  FORMAT(/,2x,'60 METER SIGMA THETA OUT OF RANGE   DATE: ',a8,5x,i4)
0071      ENDIF
0072
0073      C
0074      C      CHECK 10 METER SIGMA SPEED
0075      C
0076      IF((sigmat(3) .GT. 10 .OR. sigmat(3) .LT. 0) .AND.
0077      * (sigmat(3) .NE. 9999)) THEN
0078      WRITE(3,404)date,sigmat(3)
0079      404  FORMAT(/,2x,'10 METER SIGMA SPEED OUT OF RANGE   DATE: ',a8,5x,i4)
0080      ENDIF
0081
0082      C
0083      C      CHECK 60 METER SIGMA SPEED
0084      C
0085      IF((sigmat(5) .GT. 10 .OR. sigmat(5) .LT. 0) .AND.
0086      * (sigmat(5) .NE. 9999)) THEN
0087      WRITE(3,405)date,sigmat(5)
0088      405  FORMAT(/,2x,'60 METER SIGMA SPEED OUT OF RANGE   DATE: ',a8,5x,i4)
0089      ENDIF
0090
0091      RETURN
0092      END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	357	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	263	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	28	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated		648

ENTRY POINTS

Address	Type	Name	References
0-00000000		SIGMA	31#

VARIABLES

Address	Type	Name	Attributes	References
AP-00000000	CHAR	DATE		31 44 51 60 69 78 87

ARRAYS

Address	Type	Name	Attributes	Bytes	Dimensions	References
AP-00000040	I+4	SIGMAT		20	(5)	31 42 49(3) 51
58(3)				60 85(3)	87(3) 87	69 76(3) 78

LABELS

Address	Label	References
1-00000000	401'	51 62#
1-00000020	402'	60 61#
1-00000030	403'	69 76#
1-00000090	404'	78 79#
1-000000D0	405'	87 88#

```

-----
      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/CROSS/CHECK=ALL/EXTEND SIGMA

/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
/NOO_FLOATING /I4 /NOMACHINE_CODE /OPTIMIZE

COMPILATION STATISTICS

Run Time: 2.69 seconds
Elapsed Time: 5.32 seconds
Page Faults: 586
Dynamic Memory: 342 pages

```

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 C
0003 C      WAIT.FOR
0004 C
0005 C      WRITTEN BY : K. W. BURK
0006 C
0007 C      PURPOSE : TO PROVIDE A WAY TO WAIT, OR PAUSE THE EXECUTION OF A PROGRAM
0008 C                GIVEN THE NUMBER OF SECONDS THE USER WISHES TO WAIT.
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 : DATE_TIME_CHECK.FOR
0023 C
0024 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0025
0026      SUBROUTINE wait(wait_time)
0027
0028      IMPLICIT NONE
0029
0030 C
0031 C      DECLARE VARIABLES
0032 C-----
0033 C      TYPE      ! VARIABLE      ! COMMENTS
0034 C-----
0035 C
0036      INTEGER*2   chours          ! Current hour
0037      INTEGER*2   cmin            ! Current minute
0038      INTEGER*2   csec            ! Current second
0039
0040      REAL*4       wait_time      ! Number of seconds to wait
0041      REAL*4       ctime          ! Current time in total seconds
0042      REAL*4       rhours         ! Real form of chours
0043      REAL*4       rmin           ! Real form of cmin
0044      REAL*4       rsec           ! Real form of csec
0045
0046      CHARACTER*8  systime        ! Current system time
0047
0048
0049 C
0050 C      GET SYSTEM TIME
0051 C
0052
0053      CALL TIME(systime)
0054

```



```

0055 C
0056 C      SEPERATE TIME INTO INTEGER PARTS, CONVERT TO REAL, GET TOTAL SECONDS
0057 C
0058
0059      READ(sysime(1:2),'(i2)')chours
0060      READ(sysime(4:5),'(i2)')cmin
0061      READ(sysime(7:8),'(i2)')csec
0062
0063      rsec = FLOATI(csec)
0064      rmin = FLOATI(cmin)
0065      rhours = FLOATI(chours)
0066
0067      ctime = rsec + (rmin * 60.0) + (rhours * 3600.0)
0068
0069 C
0070 C      FIND DELTA TIME AND CHECK
0071 C
0072
0073 1      CONTINUE
0074
0075      IF(SECNDS(ctime) .GT. wait_time) THEN
0076          CONTINUE
0077      ELSE
0078          GOTO 1
0079      ENDIF
0080
0081      RETURN
0082      END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	242	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	3	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		26f

VARIABLES

Address	Type	Name	Attributes	References
2-00000000	I*2	CHOURS	36	59= 65
2-0000000A	I*2	CMIN	37	60= 64
2-0000000C	I*2	CSEC	38	61= 63
2-00000010	R*4	CTIME	41	67= 76
..	R*4	RHOURS	42	66= 67
..	R*4	RMIN	43	64= 67
..	R*4	RSEC	44	63= 67
2-00000000	CHAR	SYSTIME	46	53A 59 60 61
AP-00000040	R*4	WAIT TIME	26	48 75

LABELS

Address	Label	References
0-000000DC	1	73# 78

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	References
R=4	FORSECNDS	75
	FORTIME_T_DS	58

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/CROSS/CHECK=ALL/EXTEND WAIT

/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)

/DEBUG=(NOSYMBOLS,TRACEBACK)

/STANDARD=(NOSYNTAX,NOSOURCE FORM)

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

/WARNINGS=(GENERAL,MODECLARATIONS,NOULTRIX)

/CONTINUATIONS=19 /CROSS REFERENCE /NOD LINES /EXTEND_SOURCE /F77

/NOG_FLOATING /I4 /NOMACHINE_CODE /OPTIMIZE

COMPILATION STATISTICS

Run Time:	2.19 seconds
Elapsed Time:	5.64 seconds
Page Faults:	559
Dynamic Memory:	334 pages

APPENDIX D

DESCRIPTION, FLOW CHART, AND CODE LISTINGS FOR THE TELEMQA PROGRAM

APPENDIX D

DESCRIPTION, FLOW CHART, AND CODE LISTINGS FOR THE TELEMQA PROGRAM

TELEMQA.FOR

TELEMQA is the main program that is executed, and it passes control to subroutines until the program is terminated by the user. The source code is located in the [FCST.QA PROGRAMS] directory on the HMS computer. Figure D.1 shows a flow chart for the TELEMQA program. Figure D.2 shows the hierarchy of all the subroutines associated with TELEMQA.

When TELEMQA begins execution, the screen is cleared and the main menu is displayed. The user is then asked to enter the type of telemetry data desired. The user has the choice of wind data, temperature data, or precipitation data. After a valid response, the user is asked to enter the input file name. If a valid file name is entered, the program continues execution by passing control to the processing subroutine.

When the subroutines receive control, they process the input data file specified by the user. This processing creates an output file. When processing is complete, control is passed back to the main program, TELEMQA.

When TELEMQA is back in control, the user is asked if another file needs to be processed. If so, then execution begins again. If not, execution is terminated, and control is passed to the operating system.

FIGURE D.1. Flow Chart for the TELEMQA Program

FIGURE D.2. Subroutines Used in TELEMQA


```

0066      *      3X,'*****',///)
0067 1      WRITE(6,101)
0068 101     FORMAT(//,3X,'There are three types of Telemetry Data.',/,
0069      *      3X,'Enter the NUMBER of the data you want to use.',/,
0070      *      /,3X,' 1) Telemetry WIND Data',/,
0071      *      3X,' 2) Telemetry TEMPERATURE Data',/,
0072      *      3X,' 3) Telemetry PRECIPITATION Data',/,
0073      *      3X,'Enter the NUMBER --> ',8)
0074      READ(5,'(i3)')choice
0075
0076      C
0077      C      PROMPT FOR INPUT FILENAME
0078      C
0079
0080 2      WRITE(6,201)
0081 201     FORMAT(///,3X,'Now enter the filename. You can assume the',/,
0082      *      3X,'file is located at [FCST.DAT] ',/,
0083      *      3X,'You DO NOT need to type in "TELEMETRY" as',/,
0084      *      3X,'of the filename, and DO NOT type in ".DAT" .')
0085      IF(choice .EQ. 1) example = 'WINDSOCT'
0086      IF(choice .EQ. 2) example = 'TEMPSOCT'
0087      IF(choice .EQ. 3) example = 'PRECIPDOCT'
0088      WRITE(6,202)example
0089 202     FORMAT(3X,'( ex. ',a9,' )',/,
0090      *      3X,'Enter the FILENAME --> ',8)
0091      READ(5,'(a30)')infile
0092
0093      C
0094      C      CHECK INPUT FILENAME
0095      C
0096
0097      IF((infile(1:5) .NE. 'winda' .AND. infile(1:5) .NE. 'WINDS') .AND.
0098      *      (infile(1:5) .NE. 'tempe' .AND. infile(1:5) .NE. 'TEMPS') .AND.
0099      *      (infile(1:5) .NE. 'preci' .AND. infile(1:5) .NE. 'PRECI')) THEN
0100      TYPE=
0101      TYPE=, ' *** ERROR in filename specs, no WINDS,TEMPS,or PRECIP'
0102      GOTO 2
0103      ENDIF
0104
0105      DO 237 i = 1,30
0106 237     IF(infile(i:i) .EQ. ' ') GOTO 238 ! FIND THE LENGTH OF THE INPUT
0107 238     i = i - 1 ! FILENAME
0108
0109      outfile = 'TELEM'//infile(1:i)//' QA.ERR'
0110      infile = 'TELEMETRY_'//infile(1:i)//'.DAT'
0111
0112      OPEN(UNIT=2,FILE='[FCST.DAT]'//infile,STATUS='OLD',READONLY)
0113      OPEN(UNIT=3,FILE=outfile,STATUS='NEW',CARRIAGECONTROL = 'LIST')
0114
0115      C
0116      C      BEGIN THE QUALITY ASSURANCE PROCESSING
0117      C
0118
0119      WRITE(6,102)outfile
0120 102     FORMAT(1X,/,3X,'OUTPUT FILE NAME WILL BE . . . ',A30,/,
0121      *      /,3X,'Processing has now begun . . . ',/)
0122
0123      CALL wait(5.0)
0124
0125      WRITE(3,240)
0126 240     FORMAT(///,3X,'This file contains the actual data as well as any',/,
0127      *      3X,'Warning and/or Error Messages. The Messages will ',/,
0128      *      3X,'appear ABOVE the record in question.',/)
0129      WRITE(3,103)infile
0130 103     FORMAT(3X,'>>> ERROR LOG <<< FILENAME = ',A30)

```

```

0131      WRITE(3,104)
0132 104   FORMAT(1X,75('-'))
0133
0134      IF(choice .EQ. 1) CALL telem_wind
0135      IF(choice .EQ. 2) CALL telem_tesp
0136      IF(choice .EQ. 3) CALL telem_prdp
0137
0138      CLOSE(2)
0139      CLOSE(3)
0140
0141      WRITE(6,111)
0142 111   FORMAT(//,3X,'Processing complete . . .',//,
0143 *      3X,'Would you like to process another file ? (y or n) : ',3)
0144      READ(5,'(a1)')more
0145      IF((more .EQ. 'Y') .OR. (more .EQ. 'y')) GOTO 1
0146
0147      STOP ' Program Operation Complete '
0148
0149      END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	832	PIC COM REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	1228	PIC COM REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	212	PIC COM REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	2272	

ENTRY POINTS

Address	Type	Name	References
0-00000000		TELEMQ\$MAIN	

VARIABLES

Address	Type	Name	Attributes	References
2-00000040 138	I=4	CHOICE		40 74= 86 88 87 134 135
2-00000061 **	CHAR I=4 I	EXAMPLE		61 85= 86= 87= 88 47 105= 106(2) 107
2-0000008A 129 **	CHAR I=4 L	INFILE		62 91= 97(6) 106 109 110(2)= 112 47 107= 109 110
2-00000000	CHAR	MORE		60 144= 145(2)
2-00000028	CHAR	OUTFILE		63 109= 113A 119

LABELS

Address	Label	References
0-00000030	1	67# 145
0-00000068	2	88# 102
1-0000005A	100'	68 61#
1-00000124	101'	67 68#
1-000002FD	102'	119 128#
1-000003E5	103'	129 136#

1-0000040B	104'	131	132#
1-00000410	111'	141	142#
1-00000213	201'	80	81#
1-000002CA	202'	80	89#
**	237	105	106#
0-00000195	238	106	107#
1-0000034B	249'	125	126#

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	References
	FOR\$CLOSE	138 139
	FOR\$OPEN	112 113
	TELEM PRCP	130
	TELEM TEMP	135
	TELEM WIND	134
	WAIT	123

```

      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/CROSS/CHECK=ALL/EXTEND/CONTINUATION=99 TELEMQA

/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)
/DEBUG=(NOSYMBOLS,TRACEBACK)
/STANDARD=(NOSYNTAX,NOSOURCE FORM)
/SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP,NODICTIONARY,SINGLE)
/WARNINGS=(GENERAL,NODECLARATIONS,NOLTRIX)
/CONTINUATIONS=99 /CROSS REFERENCE /NOD LINES /EXTEND_SOURCE /F77
/NOQ_FLOATING /I4 /NOMACHINE_CODE /OPTIMIZE

```

COMPILATION STATISTICS

```

Run Time:      4.06 seconds
Elapsed Time:  8.72 seconds
Page Faults:   259
Dynamic Memory: 891 pages

```

```

0001  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002  C
0003  C   TELEM_WIND.FOR
0004  C
0005  C   WRITTEN BY : KEN BURK
0006  C
0007  C   PURPOSE : TO PROVIDE SOME QUALITY ASSURANCE SPECIFICALLY TO THE
0008  C   TELEMETRY WIND DATA COLLECTED AT HMS. AN OUTPUT FILE
0009  C   WILL BE CREATED THAT GIVES MESSAGES FOR ANY PROBLEMS.
0010  C
0011  C   ALGORITHM : DECLARE VARIABLES

```



```

0077      IF(date(3:5) .NE. yesterday) THEN      ! DISPLAY THE DATE TO THE SCREEN
0078      WRITE(6,100)date                        ! WHEN A DAY HAS BEEN PROCESSED
0079 100      FORMAT(3x,'DATE: ',a8)
0080      yesterday = date(3:5)
0081      ENDIF
0082
0083      C
0084      C      CHECK THE STABILITY CODE
0085      C
0086
0087      IF((stabil .LT. 1) .OR. (stabil .GT. 7)) THEN
0088      IF(stabil .NE. 99) THEN
0089      WRITE(3,107)date,stabil
0090 107      FORMAT(/,2X,'STABILITY OUT OF RANGE',8X,A8,16X,I2)
0091      ENDIF
0092      ENDIF
0093
0094      C
0095      C      CHECK THE MIXING LEVEL FOR A VALID VALUE
0096      C
0097
0098      IF((mix_level .LT. 0) .OR. (mix_level .GT. 150)) THEN
0099      IF(mix_level .NE. 9999) THEN
0100      WRITE(3,108)date,mix_level
0101 108      FORMAT(/,2X,'MIXING LEVEL OUT OF RANGE',3X,A8,16X,I4)
0102      ENDIF
0103      ENDIF
0104
0105      C
0106      C      CHECK THE WIND DIRECTIONS FOR VALIDITY
0107      C
0108
0109      DO i = 1,30
0110      IF((wind_dir(i) .LT. 0) .OR. (wind_dir(i) .GT. 360)) THEN
0111      IF((wind_dir(i) .NE. 80) .AND. (wind_dir(i) .NE. 90)) THEN
0112      WRITE(3,109)date,i,wind_dir(i)
0113 109      FORMAT(/,2X,'WIND DIRECTION ERROR',8X,A8,2X,'STATION ',I2,7X,I2)
0114      ENDIF
0115      ENDIF
0116      ENDDO
0117
0118      C
0119      C      CHECK THE WIND SPEEDS FOR MINIMUM AND MAXIMUM RANGE
0120      C
0121
0122      DO i = 1,30
0123
0124      IF(i .EQ. 20) THEN                        ! THIS STATION IS RATTLESNAKE MTN
0125      max_wind = 75                            ! WHICH IS KNOWN TO HAVE MUCH
0126      ELSE                                      ! HIGHER WINDS
0127      max_wind = 50
0128      ENDIF
0129
0130      IF((wind_spd(i) .LT. 0) .OR. (wind_spd(i) .GT. max_wind)) THEN
0131      IF((wind_spd(i) .NE. 80) .AND. (wind_spd(i) .NE. 99)) THEN
0132      WRITE(3,110)date,i,wind_spd(i)
0133 110      FORMAT(/,2X,'WIND SPEED OUT OF RANGE',5X,A8,2X,'STATION ',I2,7X,I2)
0134      ENDIF
0135      ENDIF
0136      ENDDO
0137
0138      C
0139      C      CHECK WIND SPEEDS FOR LARGE HOURLY CHANGE, > 10 MPH.
0140      C
0141

```

```

0142      DO i = 1,30
0143
0144          IF(JIABS(old_winds(i) - wind_spd(i)) .GT. 10) THEN
0145              IF((old_winds(i) .NE. 99 .AND. old_winds(i) .NE. 88) .AND.
0146      *      (wind_spd(i) .NE. 99 .AND. wind_spd(i) .NE. 88)) THEN
0147                  WRITE(3,112)date,i,old_winds(i),wind_spd(i)
0148      112      FORMAT(/,2x,'LARGE WIND SPEED CHANGE',5x,A8,2x'STATION ',12,6x,I3,1x,I3)
0149              ENDIF
0150          ENDIF
0151
0152      old_winds(i) = wind_spd(i)
0153
0154      ENDDO
0155
0156      C
0157      C      ECHO THE DATA TO THE OUTPUT FILE
0158      C
0159
0160      WRITE(3,185)date,rec,stabil,mix_level,(wind_dir(i),wind_spd(i),i=1,30)
0161      185      FORMAT(1X,A7,I1,I2,I4,7X,10(I2,I2,2X),/,22X,10(I2,I2,2X),/,22X,10(I2,I2,2X))
0162
0163
0164      GOTO 2
0165
0166      99      CONTINUE
0167
0168      RETURN
0169      END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	1021	PIC COM REL LCL SHR EXE RO NOWRT LONG
1 \$PDATA	347	PIC COM REL LCL SHR NOEXE RO NOWRT LONG
2 \$LOCAL	512	PIC COM REL LCL NOSHR NOEXE RO WRT LONG
Total Space Allocated	1880	

ENTRY POINTS

Address	Type	Name	References
0-00000000		TELE_WIND	27#

VARIABLES

Address	Type	Name	Attributes	References
2-000001E9 112	CHAR	DATE		48 66= 77 78 88 89 100
** ** 72(2)	I*4 I*4	FIRST_REC I	132 147 160	51 53= 59 74= 58 59 66(3)= 70= 71(2)
** 2-000001F4	I*4 I*4	MAX_WIND MIX_LEVEL	109= 132(2) 110(2) 142= 144(2) 111(2) 145(4) 112(2) 146(4) 122= 147(3) 124 152(2) 130(2) 131(2) 160(3)=	48 126= 127= 130 98(2) 99 100 160
2-000001EC 2-000001F0 2-000001E7	I*4 I*4 CHAR	REC STABIL YESTERDAY	38 39 40	66= 100 68= 87(2) 88 89 160 54= 77 80=

ARRAYS

Address	Type	Name	Attributes	Bytes	Dimensions	References			
2-00000168	I+4	OLD_WINDO		128	(38)	44	59=	72=	
2-000000F0	I+4	OLD_WINDS		128	(38)	43	58=	71=	144
145(2)				147	152=				
2-00000078	I+4	WIND_DIR		128	(38)	42	66=	72	110(2)
111(2)				112	168				
2-00000000	I+4	WIND_SPD		128	(38)	41	68=	71	130(2)
131(2)				132	144	145(2)	147	152	
				168					

LABELS

Address	Label	References		
0-0000002C	2	66#	104	
0-000000FC	99	66	166#	
1-00000000	105'	66	67#	
1-00000032	106'	78	79#	
1-0000003F	107'	89	90#	
1-00000063	108'	100	101#	
1-0000008A	109'	112	113#	
1-000000BA	110'	132	133#	
1-000000ED	112'	147	148#	
1-00000124	185'	168	181#	

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/CROSS/CHECK=ALL/EXTEND/CONTINUATION=99 TELE_WIND

/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)
/DEBUG=(NOSYMBOLS,TRACEBACK)
/STANDARD=(NOSYNTAX,NOSOURCE FORM)
/SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP,NODICTIONARY,SINGLE)
/WARNINGS=(GENERAL,NODECLARATIONS,NOULTRIX)
/CONTINUATIONS=99 /CROSS REFERENCE /MOD LINES /EXTEND_SOURCE /F77
/NOQ_FLOATING /I4 /NOMACHINE_CODE /OPTIMIZE

```

COMPILATION STATISTICS

```

Run Time:      5.50 seconds
Elapsed Time:  9.68 seconds
Page Faults:   683
Dynamic Memory: 411 pages

```



```

0064      CALL gregorian(year,jday,month,day)      ! GET THE MONTH FROM THE JULIAN DAY
0065
0066      IF(jday .NE. yesterday) THEN              ! DISPLAY THE DATE TO THE SCREEN
0067          WRITE(6,106)year,month,day            ! WHEN A DAY HAS BEEN PROCESSED
0068      106      FORMAT(3x,'DATE : ',I2,I2,I2)
0069          yesterday = jday
0070      ENDIF
0071
0072      C
0073      C      CHECK THE TEMPERATURES WITH MAX/MIN RANGE
0074      C
0075
0076      temp_err = .FALSE.
0077      DO i = 1,36
0078
0079          IF(temps(i) .LT. min_temp(month) .OR.
0080             *      temps(i) .GT. max_temp(month)) THEN
0081              IF(temps(i) .NE. 9999 .AND. temps(i) .NE. #) THEN
0082                  IF(temp_err) THEN                ! IF THERE WAS AN ERR ON THE LAST
0083                      WRITE(3,4015)jday,hour,i,temps(i) ! STATION THEN NO NEED TO PRINT
0084                  ELSE                               ! ANOTHER BLANK LINE
0085                      WRITE(3,401) jday,hour,i,temps(i)
0086                  ENDIF
0087                  temp_err = .TRUE.
0088      401      FORMAT(/,2x,'TEMP OUT OF RANGE',3x,'JDAY ',I3,' HOUR ',I2,
0089             *      3x,'STATION ',I2,3x,'DATA',I5)
0090      4015     FORMAT( 2x,'TEMP OUT OF RANGE',3x,'JDAY ',I3,' HOUR ',I2,
0091             *      3x,'STATION ',I2,3x,'DATA',I5)
0092      ELSE
0093          temp_err = .FALSE.
0094      ENDIF
0095      ENDIF
0096      ENDDO
0097
0098      C
0099      C      CHECK THE DELTA TEMP FOR THE AREA
0100      C
0101
0102      min = 5555
0103      max = -555
0104
0105      DO i = 1,36
0106          IF(i .EQ. 26 .OR. i .EQ. 16) GOTO 571
0107          IF(temps(i) .GT. max .AND. temps(i) .NE. 9999) max = temps(i)
0108          IF(temps(i) .LT. min .AND. temps(i) .NE. # ) min = temps(i)
0109      571      CONTINUE
0110      ENDDO
0111
0112      IF(ABS(max - min) .GT. 250) THEN
0113          WRITE(3,402)jday,hour,max,min
0114      402      FORMAT(/,2x,'DELTA TEMP FOR AREA TOO LARGE',3x,'JDAY ',I3,' HOUR ',I2,
0115             *      3x,'DATA ',3x,I4,3x,I4)
0116      ENDIF
0117
0118      C
0119      C      ECHO THE DATA TO THE OUTPUT FILE
0120      C
0121
0122      WRITE(3,106)year,jday,hour,rec,(temps(i),i=1,10)
0123      WRITE(3,106)year,jday,hour,(temps(i),i=11,20)
0124      WRITE(3,107)year,jday,hour,(temps(i),i=21,30)
0125      106      FORMAT(i2,i3,i2,i1,11X,10(1X,I5))
0126      107      FORMAT(i2,i3,i2,'2',11X,10(1X,I5))
0127      107      FORMAT(i2,i3,i2,'3',11X,10(1X,I5))
0128

```

```

0129
0130      GOTO 2
0131
0132      99      CONTINUE
0133
0134      RETURN
0135      END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	917	PIC COM REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	334	PIC COM REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	264	PIC COM REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	1515	

ENTRY POINTS

Address	Type	Name	References
0-00000000		TELETEMP	27#

VARIABLES

Address	Type	Name	Attributes	References
2-000000E8	I*4	DAY		45 64A 67
2-000000EC	I*4	HOURL		48 61= 83 85 113 122 123
124				
2-000000DC	I*4	I		48 61(2)= 77= 79(2) 81(2) 83(2) 85(2)
105=				
2-000000F0	I*4	JDAY	106(2) 107(3) 108(3)	122(2)= 123(2)= 124(2)=
113			47	61= 64A 66 69 83 85
**	I*4	MAX	122 123	41 103= 107(2)= 112 113
**	I*4	MIN		42 102= 108(2)= 112 113
2-000000E4	I*4	MONTH		45 64A 67 79(2)
2-000000D8	I*4	REC		38 61= 122
**	I*4	TEMP_ERR		68 76= 82 87= 93=
2-000000E0	I*4	YEAR		45 61= 64A 67 122 123 124
**	I*4	YESTERDAY		48 59= 88 89=

ARRAYS

Address	Type	Name	Attributes	Bytes	Dimensions	References
2-00000078	I*4	MAX TEMP		48	(12)	43 520 79
2-000000A8	I*4	MIN TEMP		48	(12)	44 550 79
2-00000080	I*4	TEMP'S		128	(30)	39 61= 79(2) 81(2)
83						
				85	107(3) 108(3)	122 123
				124		

Address	Label	References
0-00000018	2	61, 130
0-00000038C	99	61, 132
1-00000000	105'	61, 62
1-0000002C	106'	67, 68
1-00000110	105'	122, 125
1-000000124	106'	123, 126
1-000000139	107'	124, 127
1-00000003E	401'	85, 88
1-0000000C7	402'	113, 114
0-000000238	571	106, 109
1-000000083	4015'	83, 90

Type	Name	References
	GREGORIAN	64

```

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

```

```
FOR/LIS/CROSS/CHECK=ALL/EXTEND/CONTINUATION=99 TELM_TEMP
/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)
/DEBUG=(NOSYMBOLS,TRACEBACK)
/STANDARD=(NOSYNTAX,NOSOURCE FORM)
/SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP,NODICTIONARY,SINGLE)
/WARNINGS=(GENERAL,NODECLARATIONS,NOULTRIX)
/CONTINUATIONS=99 /CROSS REFERENCE /MOD LINES /EXTEND_SOURCE /F77
/NOG FLOATING /I4 /NOMACHINE CODE /OPTIMIZE
```

```
Run Time:      4.87 seconds
Elapsed Time:  8.87 seconds
Page Faults:   631
Dynamic Memory: 428 pages
```

D.15


```

0074
0075      precip = .FALSE.
0076      DO 345 i = 1,4
0077 345      IF(rain(i) .GT. 0 .AND. rain(i) .LE. 25) precip = .TRUE.
0078
0079      IF(precip .AND. (rain(1) .EQ. 0 .OR. rain(2) .EQ. 0 .OR.
0080      *      rain(3) .EQ. 0 .OR. rain(4) .EQ. 0)) THEN
0081      WRITE(3,302)date
0082 302      FORMAT(/,3x,'PRECIP. NOT FOUND AT ALL STATIONS',3X,A7)
0083      ENDIF
0084
0085      C
0086      C      ECHO THE DATA TO THE OUTPUT FILE
0087      C
0088
0089      WRITE(3,105)date,(rain(i),i=1,4)
0090 105      FORMAT(a7,12X,4(2x,i4))
0091
0092      GO TO 2
0093
0094 99      CONTINUE
0095
0096      RETURN
0097      END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	464	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	128	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	66	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	648	

ENTRY POINTS

Address	Type	Name	References
0-00000000		TELEM_PRCP	27#

VARIABLES

Address	Type	Name	Attributes	References
2-00000013	CHAR	DATE		42 49= 62 63(2) 65 81
89				
2-0000001C	I=4	I		38 49(2)= 62= 63(2) 64 65 76=
77(2)			89(2)=	
**	L=4	PRECIP		44 75= 77= 79
2-00000010	CHAR	YESTERDAY		41 47= 62 65=

ARRAYS

Address	Type	Name	Attributes	Bytes	Dimensions	References
2-00000000	I=4	RAIN		16	(4)	39 49= 63(2) 84
65				77(2)	79(4) 89	

LABELS

Address	Label	References	
0-00000018	2	49#	92
0-000001CC	99	49	94#
1-00000000	105'	49	50#
1-0000000E	100'	53	54#
1-0000000F	105'	89	98#
1-00000020	301'	85	88#
1-00000044	302'	81	82#
**	345	76	77#

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/CROSS/CHECK=ALL/EXTEND/CONTINUATION=99 TELEM_PRCP
/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)
/DEBUG=(NOSYMBOLS,TRACEBACK)
/STANDARD=(NOSYNTAX,NOSOURCE FORM)
/SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP,NODICTIONARY,SINGLE)
/WARNINGS=(GENERAL,NODECLARATIONS,NOULTRIX)
/CONTINUATIONS=99 /CROSS REFERENCE /NOD LINES /EXTEND_SOURCE /F77
/NOG_FLOATING /I4 /NOMACHINE_CODE /OPTIMIZE
```

COMPILATION STATISTICS

```
Run Time:      3.10 seconds
Elapsed Time:  6.43 seconds
Page Faults:   574
Dynamic Memory: 374 pages
```

```
0001  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002  C
0003  C      WAIT.FOR
0004  C
0005  C      WRITTEN BY : K. W. BURK
0006  C
0007  C      PURPOSE : TO PROVIDE A WAY TO WAIT, OR PAUSE THE EXECUTION OF A PROGRAM
0008  C                  GIVEN THE NUMBER OF SECONDS THE USER WISHES TO WAIT.
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 :  DATE_TIME_CHECK.FOR
0023 C
0024 C-----
0025
0026 SUBROUTINE wait(wait_time)
0027
0028 IMPLICIT NONE
0029
0030 C
0031 C    DECLARE VARIABLES
0032 C-----
0033 C          TYPE      ! VARIABLE      ! COMMENTS
0034 C-----
0035 C
0036 C          INTEGER*2   chours         ! Current hour
0037 C          INTEGER*2   cmin           ! Current minute
0038 C          INTEGER*2   caec           ! Current second
0039
0040 C          REAL*4      wait_time      ! Number of seconds to wait
0041 C          REAL*4      ctime          ! Current time in total seconds
0042 C          REAL*4      rhours         ! Real form of chours
0043 C          REAL*4      rmin           ! Real form of cmin
0044 C          REAL*4      raec           ! Real form of caec
0045
0046 C          CHARACTER*8  systime       ! Current system time
0047
0048
0049 C
0050 C    GET SYSTEM TIME
0051 C
0052
0053 CALL TIME(systime)
0054
0055 C
0056 C    SEPERATE TIME INTO INTEGER PARTS, CONVERT TO REAL, GET TOTAL SECONDS
0057 C
0058
0059 READ(systime(1:2),'(i2)')chours
0060 READ(systime(4:5),'(i2)')cmin
0061 READ(systime(7:8),'(i2)')caec
0062
0063 raec = FLOATI(caec)
0064 rmin = FLOATI(cmin)
0065 rhours = FLOATI(chours)
0066
0067 ctime = raec + (rmin * 60.0) + (rhours * 3600.0)
0068
0069 C
0070 C    FIND DELTA TIME AND CHECK
0071 C
0072
0073 1 CONTINUE
0074
0075 IF(SECNDS(ctime) .GT. wait_time) THEN
0076   CONTINUE
0077 ELSE
0078   GOTO 1
0079 ENDIF
0080
0081 RETURN
0082 END

```

PROGRAM SECTIONS

Name	Bytes	Attributes							
# \$CODE	242	PIC	CON	REL	LCL	SHR	EXE	RD	NOWRT LONG
1 \$PDATA	3	PIC	CON	REL	LCL	SHR	NOEXE	RD	NOWRT LONG
2 \$LOCAL	52	PIC	CON	REL	LCL	NOSHR	NOEXE	RD	WRT LONG
Total Space Allocated		297							

ENTRY POINTS

Address	Type	Name	References
0-00000000	WAIT		204

VARIABLES

Address	Type	Name	Attributes	References				
2-00000000	I+2	CHOURS		38	59=	65		
2-0000000A	I+2	CMIN		37	60=	64		
2-0000000C	I+2	CSEC		38	61=	63		
2-00000010	R+4	CTIME		41	67=	75		
**	R+4	RHOURS		42	65=	67		
**	R+4	RMIN		43	64=	67		
**	R+4	RSEC		44	63=	67		
2-00000000	CHAR	SYSTIME		46	63A	69	68	61
AP-00000040	R+4	WAIT_TIME		26	48	75		

LABELS

Address	Label	References
0-000000DC	1	73# 78

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	References
R+4	FOR\$ECNDS	75
	FOR\$TIME_T_OS	63

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/CROSS/CHECK=ALL/EXTEND WAIT

/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

/NOO_FLOATING /I4 /NOMACHINE_CODE /OPTIMIZE

COMPILATION STATISTICS

Run Time:	2.13 seconds
Elapsed Time:	5.64 seconds
Page Faults:	559
Dynamic Memory:	334 pages

APPENDIX E

DESCRIPTION, FLOW CHART, AND CODE LISTINGS FOR THE SFCQA PROGRAM

APPENDIX E

DESCRIPTION, FLOW CHART, AND CODE LISTINGS FOR THE SFCQA PROGRAM

SFCQA is the main program that interacts with the user and uses other subroutines until execution is terminated by the user. The source code for SFCQA is located on the HMS computer system in the [FCST.QA_PROGRAMS] directory. Figure E.1 shows a flow chart for the SFCQA program. Figure E.2 shows the hierarchy of all the subroutines used by SFCQA.

The program begins by clearing the screen and displaying a message briefly describing what the code is for. The user is then asked to enter the input file name. When a valid file name is entered, the code passed control to a series of subroutines. Each subroutine processes a portion of each record in the input file.

When the subroutines are done processing the data in the input file, control is then passed back to SFCQA. At this point, the program asks the user if another file needs to be processed. If so, the program will go back to the beginning of the code. If not, the program is terminated.

FIGURE E.1. Flow Chart for the SFCQA Program

FIGURE E.2. Subroutines Used by SFCQA

```

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 C
0003 C   SFCQA.FOR
0004 C
0005 C   WRITTEN BY : KEN BURK
0006 C               GREGG ANDREWS
0007 C
0008 C   PURPOSE : TO PROVIDE QUALITY ASSURANCE CHECKS FOR THE SURFACE OBSERVATION
0009 C              DATA THAT IS COLLECTED AND ARCHIVED AT HMS.
0010 C              ALL PARAMETER WILL BE CHECKED FOR LARGE HOURLY VARIATIONS, AND
0011 C              CHECKED AGAINST EXTREME VALUES.
0012 C              THE PARAMETERS ARE:
0013 C                  CLOUD DATA
0014 C                  CEILING HEIGHT
0015 C                  SKY CONDITIONS
0016 C                  VISIBILITY
0017 C                  WEATHER/VISIBILITY INFORMATION
0018 C                  SEA-LEVEL PRESSURE
0019 C                  DEW POINT TEMP
0020 C                  WIND DIRECTION AND SPEED
0021 C                  STATION PRESSURE
0022 C                  DRY & WET BULB TEMP
0023 C                  RELATIVE HUMIDITY
0024 C                  SKY COVER
0025 C                  PRECIPITATION AMOUNTS
0026 C                  SOLAR RADIATION
0027 C
0028 C   ALGORITHM : DECLARE VARIABLES
0029 C                PROMPT AND READ INPUT FILE FROM USER
0030 C                OPEN INPUT AND OUTPUT FILE
0031 C                READ FROM INPUT FILE UNTIL END OF FILE
0032 C                PROCESS DATA BY MAKING CALLS TO SUBROUTINES
0033 C                IF NOT END OF FILE LOOP BACK
0034 C                ASK USER IF ANOTHER FILE
0035 C                END
0036 C
0037 C   VERSION : 1.0   JUNE 1988
0038 C
0039 C   MAKES CALLS TO: WAIT.FOR
0040 C                   ATMOS.FOR
0041 C                   SKY.FOR
0042 C                   OBSTR.FOR
0043 C                   PRESSURE.FOR
0044 C                   WIND.FOR
0045 C                   TEMPS.FOR
0046 C                   PRECIP.FOR
0047 C                   SOLAR.FOR
0048 C
0049 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0050 C
0051 C   IMPLICIT NONE
0052 C
0053 C   VARIABLE DECLARATION
0054 C
0055 C-----
0056 C   TYPE      ! VARIABLE      ! COMMENTS
0057 C-----
0058 C
0059 C   INTEGER*2  clouds(5)      ! CLOUD CODES
0060 C   INTEGER*2  date(4)        ! DATE YY,MM,DD,HH
0061 C   INTEGER*2  ceiling        ! ALTITUDE OF CEILING
0062 C   INTEGER*2  old ceiling    ! ALTITUDE OF CEILING FOR PREVIOUS HOUR
0063 C   INTEGER*2  cond(4)        ! SKY CONDITIONS
0064 C   INTEGER*2  visible        ! VISIBILITY MILES

```

```

0066 INTEGER*2 old_visible ! VISIBILITY AT PREVIOUS HOUR
0067 INTEGER*2 liq_precip(2) ! TYPE OF LIQUID PRECIPITATION
0068 INTEGER*2 fr_precip(3) ! TYPE OF FROZEN PRECIPITATION
0069 INTEGER*2 obstruct(2) ! TYPE OF OBSTRUCTIONS
0070 INTEGER*2 sea_pres ! SEALEVEL PRESSURE
0071 INTEGER*2 dew_pt ! DEW POINT TEMPERATURE
0072 INTEGER*2 wind_dir ! WIND DIRECTION
0073 INTEGER*2 old_wind_dir ! WIND DIRECTION FOR PREVIOUS HOUR
0074 INTEGER*2 wind_spd ! WIND SPEED
0075 INTEGER*2 old_wind_spd ! WIND SPEED FOR PREVIOUS HOUR
0076 INTEGER*2 stat_pres ! STATION PRESSURE
0077 INTEGER*2 old_stat_pres ! STATION PRESSURE FOR PREVIOUS HOUR
0078 INTEGER*2 dry_bulb ! DRY BULB TEMPERATURE
0079 INTEGER*2 old_dry_bulb ! DRY BULB TEMP FOR PREVIOUS HOUR
0080 INTEGER*2 wet_bulb ! WET BULB TEMPERATURE
0081 INTEGER*2 old_wet_bulb ! WET BULB TEMP FOR PREVIOUS HOUR
0082 INTEGER*2 rel_hum ! RELATIVE HUMIDITY
0083 INTEGER*2 old_rel_hum ! RELATIVE HUMIDITY FOR PREVIOUS HOUR
0084 INTEGER*2 sky_cover ! AMOUNT OF SKY COVER, PERCENTAGE
0085 INTEGER*2 rain ! AMOUNT OF RAIN
0086 INTEGER*2 snoweq ! AMOUNT OF SNOW (MELTED)
0087 INTEGER*2 snowun ! AMOUNT OF SNOW (UNMELTED)
0088 INTEGER*2 solarad ! SOLAR RADIATION
0089 INTEGER*2 old_solarad ! SOLAR RADIATION FOR PREVIOUS HOUR
0090 INTEGER*2 weather ! A CODED WEATHER INDICATOR
0091 INTEGER*2 i ! ARRAY AND LOOP INDEX
0092 INTEGER*4 today ! NUMERICAL FORM OF DAY
0093 INTEGER*4 yesterday ! DATE OF PREVIOUS DAY
0094
0095 CHARACTER*1 more ! BOOLEAN FOR ANOTHER FILE
0096 CHARACTER*20 infile ! NAME OF THE INPUT FILE
0097 CHARACTER*20 outfile ! CONSTRUCTED OUTPUT FILE NAME
0098
0099 LDGICAL first_rec ! BOOLEAN FOR FIRST RECORD
0100 LDGICAL out_of_range(3) ! BOOLEAN ARRAY FOR TEMPS ERRORS
0101
0102
0103 C
0104 C PROMPT USER FOR INPUT FILE NAME, THEN OPEN THE INPUT
0105 C AND OUTPUT FILES
0106 C
0107
0108 WRITE(6,10)
0109 10 FORMAT('//////////',
0110 * 8X,'*****',/,
0111 * 8X,'* QUALITY ASSURANCE PROGRAM *',/,
0112 * 8X,'* ',/,
0113 * 8X,'* HMS SURFACE OBSERVATION DATA *',/,
0114 * 8X,'*****',/))
0115 1 WRITE(6,11)
0116 11 FORMAT(/,3X,'Enter the name of the SFCOBS DATA input file,',/,
0117 * 3X,'Assume it is located at [FCST.DAT]: ',/,
0118 * 3X,'(ex. sfcobasay or sfcoba86) Enter WITHOUT ".dat" > ',8)
0119
0120 READ(5,'(A15)')infile
0121
0122 IF((infile(1:6) .NE. 'sfcoba') .AND.
0123 * (infile(1:6) .NE. 'SFCOBS')) THEN
0124 TYPE*,', '
0125 TYPE*,', '
0126 TYPE*, '*** ERROR in file name; Must begin with "sfcoba"'
0127 GOTO 1
0128 ENDIF
0129
0130 outfile = infile

```



```

0131
0132 IF(infile(9:) .EQ. ' ') THEN ! FILENAME IS FOR A YEAR DATA FILE
0133 outfile(9:16) = 'QA.ERR'
0134 infile(9:12) = 'dat'
0135 ELSE ! FILENAME IS FOR A MONTH DATA FILE
0136 outfile(10:16) = 'QA.ERR'
0137 infile(10:13) = 'dat'
0138 ENDIF
0139
0140 OPEN(UNIT=2,FILE='[FCST.DAT] '//infile,STATUS='OLD',READONLY)
0141 OPEN(UNIT=3,FILE=outfile,STATUS='NEW',
0142 * CARRIAGECONTROL = 'LIST')
0143
0144 C
0145 C BEGIN THE QUALITY ASSURANCE PROCESS
0146 C
0147
0148 WRITE(6,55)outfile
0149 55 FORMAT(1x,/,3x,'Output File Name will be . . . ',a20,/,
0150 * /,3x,'Processing has now begun . . . ',/)
0151
0152 CALL wait(4.0)
0153
0154 WRITE(3,249)
0155 249 FORMAT(/,3x,'This file contains the actual data as well as any',/,
0156 * 3x,'Warning and/or Error Messages. The Messages will ',/,
0157 * 3x,'appear ABOVE the record in question.',/)
0158 WRITE(3,250)infile
0159 250 FORMAT(3x,' > > > ERROR LOG < < < FILENAME= ',a16,/)
0160 WRITE(3,251)
0161 251 FORMAT(9x,'Error',25x,'Date',7x,'DATA',/,3x,65('-'))
0162
0163 first_rec = .TRUE.
0164
0165 DO 68 i = 1,3
0166 68 out_of_range(i) = .FALSE.
0167
0168 yesterday = 0
0169
0170 2 READ(2,200,END=999)(clouds(i),i=1,5),date,ceiling,(cond(i),i=1,4),visible,
0171 * (liq_precip(i),i=1,2),(fr_precip(i),i=1,3),(obstruct(i),i=1,2),
0172 * sea_pres,dew_pt,wind_dir,wind_spd,
0173 * stat_pres,dry_bulb,wet_bulb,rel_hum,sky_cover,rain,snoweq,snowun,
0174 * weather,solarad
0175
0176 200 FORMAT(6i1,4i2,i3,4i1,i3,1x,2i1,3i1,2i1,i4,i3,i2,i2,i4,i3,i3,i3,i2,
0177 * 3i2,i1,i3)
0178
0179 today = date(3)
0180 IF(today .NE. yesterday) THEN ! DISPLAY DATE TO THE SCREEN
0181 WRITE(6,100)date
0182 100 FORMAT(3x,'DATE: ',4(i2,1x))
0183 yesterday = today
0184 ENDIF
0185
0186 IF(first_rec) THEN ! IF FIRST RECORD, CREATE PREVIOUS VALUES
0187 ! BY USING CURRENT VALUES. FIRST REC ONLY.
0188 old_ceiling = ceiling
0189 old_visible = visible
0190 old_wind_dir = wind_dir
0191 old_wind_spd = wind_spd
0192 old_dry_bulb = dry_bulb
0193 old_wet_bulb = wet_bulb
0194 old_rel_hum = rel_hum
0195 old_solarad = solarad

```

```

0196         old_stat_pres = stat_pres
0197         first_rec = .FALSE.
0198
0199     ENDIF
0200
0201 C
0202 C     CHECK CLOUD CODES AND CEILING ALTITUDE
0203 C
0204
0205         CALL atmoa(clouds,ceiling,old_ceiling,date)
0206
0207 C
0208 C     CHECK SKY CONDITION AND VISIBILITY DISTANCE
0209 C
0210
0211         CALL sky(cond,visible,old_visible,sky_cover,ceiling,date)
0212
0213 C
0214 C     CHECK WEATHER AND/OR OBSTRUCTION TO VISION CODES
0215 C
0216
0217         CALL obatr(liq_precip,fr_precip,obstruct,date,visible,weather)
0218
0219 C
0220 C     CHECK THE WIND DIRECITON AND WIND SPEED
0221 C
0222
0223         CALL wind(wind_dir,old_wind_dir,wind_spd,old_wind_spd,date)
0224
0225 C
0226 C     CHECK THE STATION AND SEALEVEL PRESSURES
0227 C
0228
0229         CALL pressur(stat_pres,sea_pres,date,old_stat_pres)
0230
0231 C
0232 C     CHECK THE TEMPERATURES, DEW POINT, AND RELATIVE HUMIDITY
0233 C
0234
0235         CALL tempe(dry_bulb,old_dry_bulb,wet_bulb,old_wet_bulb,
0236 *             dew_pt,rel_hum,old_rel_hum,date,out_of_range)
0237
0238 C
0239 C     CHECK THE AMOUNTS OF PRECIPITATION FOR RAIN, SNOW EQ., SNOW UNMELT.
0240 C
0241
0242         CALL precip(rain,snoweq,snowun,liq_precip,fr_precip,date)
0243
0244 C
0245 C     CHECK THE DIRECT SOLAR RADIATION CODE
0246 C
0247
0248         CALL solar(solarad,old_solarad,date)
0249
0250
0251 C
0252 C     REPLACE PREVIOUS VALUES WITH CURRENT, GO UP AND READ NEXT RECORD
0253 C
0254
0255         old_ceil
0256         old_visible = visible
0257         old_wind_dir = wind_dir
0258         old_wind_spd = wind_spd
0259         old_dry_bulb = dry_bulb
0260         old_wet_bulb = wet_bulb

```

```

0261      old_rel_hum = rel_hum
0262      old_solarad = solarad
0263      old_stat_pres = stat_pres
0264
0265      C
0266      C      ECHO THE DATA TO THE OUTPUT FILE
0267      C
0268
0269      WRITE(3,299)(clouds(i),i=1,5),date,ceiling,(cond(i),i=1,4),visible,
0270      * (liq_precip(i),i=1,2),(fr_precip(i),i=1,3),(obstruct(i),i=1,2),
0271      * sea_pres,dew_pt,wind_dir,wind_spd,
0272      * stat_pres,dry_bulb,wet_bulb,rel_hum,sky_cover,rain,snoweq,snowun,
0273      * weather,solarad
0274
0275      299  FORMAT(1X,5i1,4i2,i3,4i1,i3,1x,2i1,3i1,2i1,i4,i3,i2,i2,i4,i3,i3,i3,i2,
0276      *          3i2,i1,i3)
0277
0278
0279      GOTO 2
0280
0281      999  CONTINUE
0282
0283      TYPE*
0284      WRITE(6,237)
0285      237  FORMAT(3x,'Would you like to do another file? (y or n) : ',%)
0286      READ(5,'(a1)')more
0287      IF(( more .EQ. 'y' ) .OR. ( more .EQ. 'Y' )) THEN
0288
0289          CLOSE(2)
0290          CLOSE(3)
0291          GOTO 1
0292
0293      ENDIF
0294
0295
0296      STOP ' Program Operation Complete '
0297
0298      END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	1811	PIC CON REL LCL SHR EXE RD NDWRT LONG
1 \$PDATA	981	PIC CON REL LCL SHR NOEXE RD NDWRT LONG
2 \$LDCAL	568	PIC CON REL LCL NOSH NOEXE RD WRT LONG
Total Space Allocated	3892	

ENTRY POINTS

Address	Type	Name	References
0-00000000		SFCQASMAIN	

VARIABLES

Address	Type	Name	Attributes	References
2-0000005E	I*2	CEILING		82 170= 188 205A 211A 255 289
2-00000088	I*2	DEW_PT		71 170= 235A 289
2-00000076	I*2	DRY_BULB		78 170= 192 235A 259 289
**	L*4	FIRST_REC		99 183= 188 197=

**	I*2	I	91	165=	168	178(10)=	269(10)=
2-00000035	CHAR	INFILE	96	120=	122(2)	130	132
140							134=
							137=
2-00000034	CHAR	MORE	95	286=	287(2)		
2-00000068	I*2	OLD_CEILING	83	188=	205A	255=	
2-00000070	I*2	OLD_DRY_BULB	79	192=	235A	250=	
2-00000080	I*2	OLD_REL_HUM	83	194=	235A	261=	
2-0000008C	I*2	OLD_SOLARAD	89	195=	248A	262=	
2-00000074	I*2	OLD_STAT_PRES	77	196=	229A	263=	
2-00000064	I*2	OLD_VISIBLE	86	189=	211A	258=	
2-0000007C	I*2	OLD_WET_BULB	81	193=	235A	260=	
2-0000006C	I*2	OLD_WIND_DIR	73	190=	223A	257=	
2-00000078	I*2	OLD_WIND_SPD	75	191=	223A	258=	
2-00000049	CHAR	OUTFILE	97	130=	133=	130=	141A
2-00000084	I*2	RAIN	85	170=	242A	269	148
2-0000007E	I*2	REL_HUM	82	170=	194	235A	261
2-00000066	I*2	SEA_PRES	78	170=	229A	269	269
2-00000082	I*2	SKY_COVER	84	170=	211A	269	
2-00000086	I*2	SNOWEQ	88	170=	242A	269	
2-00000088	I*2	SNOWUN	87	170=	242A	269	
2-0000008A	I*2	SOLARAD	88	170=	195	248A	262
2-00000072	I*2	STAT_PRES	76	170=	196	229A	263
**	I*4	TODAY	92	170=	180	183	
2-00000062	I*2	VISIBLE	86	170=	189	211A	217A
							258
							269
2-0000008E	I*2	WEATHER	90	170=	217A	269	
2-0000007A	I*2	WET_BULB	80	170=	193	235A	260
2-0000006A	I*2	WIND_DIR	72	170=	196	223A	257
							269
2-0000008E	I*2	WIND_SPD	74	170=	191	223A	258
**	I*4	YESTERDAY	93	160=	180	183=	269

ARRAYS

Address	Type	Name	Attributes	Bytes	Dimensions	References
2-0000000C	I*2	CLOUDS		10	(5)	80 170= 205A 269
2-0000001E	I*2	COND		8	(4)	64 170= 211A 269
2-00000016	I*2	DATE		8	(4)	61 170= 179 181
205A				211A 242A	217A 248A	223A 269
2-0000002A	I*2	FR_PRECIP		8	(3)	68 170= 217A 242A
269						
2-00000026	I*2	LIQ_PRECIP		4	(2)	67 170= 217A 242A
269						
2-00000030	I*2	OBSTRUCT		4	(2)	69 170= 217A 269
2-00000000	L*4	OUT_OF_RANGE		12	(3)	160 160= 235A

LABELS

Address	Label	Referen	6-00000044	1	115#	127
291						

0-0000022C	2	170#	279
1-00000053	10'	100	109#
1-0000011E	11'	115	116#
1-00000188	55'	148	149#
** 68			
1-0000032D	100'	185	186#
1-000002F4	200'	181	182#
1-0000037D	237'	170	176#
1-00000206	240'	284	285#
1-000002A0	250'	154	155#
1-000002CF	251'	160	169#
1-00000342	299'	160	161#
0-000005D4	999	269	275#
		170	281#

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	References
	ATMOS	205
	FOR\$CLOSE	289 290
	FOR\$OPEN	140 141
	OBSTR	217
	PRECIP	242
	PRESSUR	229
	SKY	211
	SOLAR	248
	TEMPS	235
	WAIT	152
	WIND	223

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/CROSS/LIS/CHECK=ALL/EXTEND SFCQA

/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

COMPILATION STATISTICS

Run Time: 8.85 seconds
 Elapsed Time: 17.87 seconds
 Page Faults: 378
 Dynamic Memory: 441 pages

```

0001  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002  C
0003  C      ATMOS.FOR
0004  C
0005  C      WRITTEN BY : KEN BURK
0006  C
0007  C      PURPOSE : PERFORMS CHECK ON CLOUD CODES AND THE CEILING HEIGHT
0008  C                  FOR SURFACE OBSERVATION DATA
0009  C
0010  C      ALGORITHM : DECLARE VARIABLES
0011  C                  CHECK CLOUD CODES FOR STRAY VALUES
0012  C                  CHECK CHANGE IN CEILING HEIGHT C          RETURN
0013  C
0014  C
0015  C      VERSION : 1.0
0016  C
0017  C      DATE : JUNE 1988
0018  C
0019  C      CALLED FROM : SFCQA.FOR (MAIN PROGRAM)
0020  C
0021  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0022  C
0023  C      SUBROUTINE atmos(puffs,top,old_top,time)
0024  C
0025  C      IMPLICIT NONE
0026  C
0027  C
0028  C      VARIABLE DECLARATIONS
0029  C-----
0030  C      TYPE      ! VARIABLE      ! COMMENTS
0031  C-----
0032  C
0033  C
0034  C      INTEGER*2      puffs(5)      ! CLOUD CODES
0035  C      INTEGER*2      top            ! CEILING ALTITUDE
0036  C      INTEGER*2      old_top        ! CEILING ALTITUDE PREVIOUS HOUR
0037  C      INTEGER*2      time(4)        ! DATE YY,MM,DD,HH
0038  C      INTEGER*2      i,j,k          ! ARRAY AND LOOP INDICES
0039  C
0040  C
0041  C
0042  C      CHECK EACH CLOUD CODES FOR STRAY VALUES
0043  C
0044  C
0045  C      IF((time(4) .EQ. 1) .OR. (time(4) .EQ. 4) .OR.      ! CLOUD DATA IS ONLY
0046  C      * (time(4) .EQ. 7) .OR. (time(4) .EQ. 10) .OR.      ! RECORDED EVERY 3
0047  C      * (time(4) .EQ. 13) .OR. (time(4) .EQ. 16) .OR.      ! HOURS.
0048  C      * (time(4) .EQ. 22)) THEN
0049  C
0050  C      DO i = 1,4
0051  C
0052  C      DO j = i+1,5
0053  C
0054  C      IF((puffs(i) .EQ. 0) .AND.      ! GOING LEFT TO RIGHT, A ZERO SHOULD
0055  C      * (puffs(j) .NE. 0)) THEN      ! NOT BE FOLLOWED BY A NON-ZERO
0056  C      WRITE(3,200)(time(k),k=1,4),puffs(i),puffs(j)
0057  C      200 FORMAT(/,3x,'ERROR IN CLOUD LAYER SEQUENCE',5x,4i2,5x,i1,3x,i1)
0058  C      ENDF
0059  C
0060  C      ENDDO

```

```

0061
0062      ENDDO
0063
0064      ENDIF
0065
0066      C
0067      C      CHECK FOR LARGE CHANGE IN CEILING ALTITUDE
0068      C
0069
0070      IF((IABS(old_top - top) .GT. 100) THEN
0071
0072          IF((old_top .NE. 0) .AND. (top .NE. 0)) THEN
0073
0074              WRITE(3,201)(time(k),k=1,4),old_top,top
0075      201      FORMAT(/,3x,'LARGE CHANGE IN CEILING HEIGHT',4x,4i2,5x,i3,3x,i3)
0076
0077          ENDIF
0078
0079      ENDIF
0080
0081      RETURN
0082      END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	340	PIC COM REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	99	PIC COM REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	40	PIC COM REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	479	

ENTRY POINTS

Address	Type	Name	References
0-00000000	ATMOS		23#

VARIABLES

Address	Type	Name	Attributes	References
**	I*2	I		38 50= 52 54 56
**	I*2	J		38 52= 54 56
**	I*2	K		38 56(2)= 74(2)=
AP-00000000	I*2	OLD TOP		23 36 70 72 74
AP-00000000	I*2	TOP		23 36 70 72 74

ARRAYS

Address	Type	Name	Attributes	Bytes	Dimensions	References
AP-00000040	I*2	PUFFS		10	(5)	23 34 54(2) 56(2)
AP-000000100 74	I*2	TIME		8	(4)	23 37 45(7) 56

LABELS

44

```

0090      max_sky = skies(1)                ! FIND THE MAXIMUM VALUE
0091      IF(skies(2) .GT. max_sky) max_sky = skies(2) ! RECORDED FOR THE SKY
0092      IF(skies(3) .GT. max_sky) max_sky = skies(3) ! CONDITIONS
0093      IF(skies(4) .GT. max_sky) max_sky = skies(4)
0094
0095      IF(max_sky .LT. 5) THEN                ! NO APPROPRIATE CLOUD LAYERS
0096          WRITE(3,303) (time(k),k=1,4),(skies(k),k=1,4)
0097      303      FORMAT(/,3x,'CEIL. W/O APPROPRIATE SKY COND.',3X,4I2,5X,4I1)
0098      ENDIF
0099
0100      ENDIF
0101
0102      RETURN
0103      END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	496	PIC CDM REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	196	PIC CDM REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	48	PIC CDM REL LCL NUSHR NOEXE RD WRT LONG
Total Space Allocated	726	

ENTRY POINTS

Address	Type	Name	References
0-00000000	SKY		29#

VARIABLES

Address	Type	Name	Attributes	References
AP-000000140	I+2	CEILING		29 45 83
**	I+2	I		44 54= 58 59 59
**	I+2	J		44 58= 58 59
**	I+2	K		44 59(2)= 73(2)= 86(2)= 96(4)=
**	I+2	MAX_SKY		47 96= 91(2)= 92(2)= 93(2)= 95
AP-0000000C0	I+2	OLD VISUAL		29 42 71 73
AP-000000180	I+2	SKY COVER		29 46 85 86
AP-000000580	I+2	VISUAL		29 41 71 73

ARRAYS

Address	Type	Name	Attributes	Bytes	Dimensions	References
AP-000000040	I+2	SKIES		8	(4)	29 40 58(2) 59(2)
98				91(2)	92(2)	93(2) 96
AP-000000180	I+2	TIME		8	(4)	29 43 59 73
86				96		

LABELS

Address	Label	References
1-00000000	300'	59 88#

1-00000031	301'	73	74#
1-00000004	302'	86	87#
1-00000093	303'	96	97#

```

-----
      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/CROSS/LIS/CHECK=ALL/EXTEND SKY

```

/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
/NOG_FLOATING /I4 /NOMACHINE_CODE /OPTIMIZE

```

COMPILATION STATISTICS

```

Run Time:      3.49 seconds
Elapsed Time:  0.22 seconds
Page Faults:   582
Dynamic Memory: 378 pages

```

```

0001  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002  C
0003  C      OBSTR.FOR
0004  C
0005  C      WRITTEN BY : KEN BURK
0006  C
0007  C      PURPOSE :   PROVIDE QUALITY ASSURANCE CHECK FOR THE WEATHER AND/OR
0008  C                  OBSTRUCTION TO VISION PARAMETERS FOUND IN THE SURFACE
0009  C                  OBSERVATION DATA FILES.  SPECIFICALLY THESE ARE:
0010  C                      TYPE OF LIQUID PRECIPITATION
0011  C                      TYPE OF FROZEN PRECIPITATION
0012  C                      TYPE OF OBSTRUCTION
0013  C
0014  C      ALGORITHM :  DECLARE VARIABLES
0015  C                      CHECK FOR VALID LIQUID PRECIP CODES
0016  C                      CHECK FOR VALID FROZEN PRECIP CODES
0017  C                      CHECK FOR VALID OBSTRUCTION CODES
0018  C                      RETURN
0019  C
0020  C      VERSION : 1.0
0021  C
0022  C      DATE :   JUNE 1988
0023  C
0024  C      CALLED FROM :  SFCQA.FOR  (MAIN PROGRAM)
0025  C
0026  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0027
0028      SUBROUTINE  obatr(lprecp,fprecp,block,time,visible,wx)
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      lprec(2)      ! TYPE OF LIQUID PRECIPITATION
0040      INTEGER*2      fprec(3)      ! TYPE OF FROZEN
0041      INTEGER*2      block(2)      ! TYPE OF OBSTRUCTION
0042      INTEGER*2      time(4)      ! DATE, YY,MM,DD,HH
0043      INTEGER*2      i,j,k      ! ARRAY AND LOOP INDECIES
0044      INTEGER*2      visible      ! VISIBILITY DISTANCE
0045      INTEGER*2      wx      ! WEATHER INDICATOR
0046
0047      LOGICAL      precip      ! IF THERE ARE ANY TYPE OF PRECIP
0048      LOGICAL      obstruction      ! IF THERE IS ANY TYPE OF OBSTR.
0049
0050
0051      C
0052      C      CHECK FOR VALID LIQUID PRECIP CODES
0053      C
0054
0055      IF((lprec(1) .NE. 0) .AND. (lprec(2) .NE. 0)) THEN
0056
0057          WRITE(3,400)(time(i),i=1,4),lprec(1),lprec(2)
0058      400      FORMAT(/,3x,'TWO TYPES OF LIQUID PRECIP.',7x,4i2,5x,i1,3x,i1)
0059
0060      ENDIF
0061
0062      C
0063      C      CHECK FOR VALID FROZEN PRECIP CODES
0064      C
0065
0066      IF((fprec(1) .NE. 0 .AND. fprec(2) .NE. 0) .OR.
0067      * (fprec(1) .NE. 0 .AND. fprec(3) .NE. 0) .OR.
0068      * (fprec(2) .NE. 0 .AND. fprec(3) .NE. 0)) THEN
0069
0070          WRITE(3,401)(time(i),i=1,4),fprec(1),fprec(2),fprec(3)
0071      401      FORMAT(/,3x,'MORE THAN ONE FROZEN PRECIP.',6x,4i2,5x,i1,1x,i1,1x,i1)
0072
0073      ENDIF
0074
0075      C
0076      C      CHECK FOR VALID OBSTRUCTION CODES
0077      C
0078
0079      IF((block(1) .NE. 0) .AND. (block(2) .NE. 0)) THEN
0080
0081          WRITE(3,402)(time(i),i=1,4),block(1),block(2)
0082      402      FORMAT(/,3x,'TWO TYPES OF OBSTRUCTIONS',9x,4i2,5x,i1,3x,i1)
0083
0084      ENDIF
0085
0086      C
0087      C      CHECK THE VISIBILITY AND PRECIP/OBSTR CODES
0088      C
0089      precip = .FALSE.
0090      obstruction = .FALSE.
0091      DO 40 i = 1,3
0092      40      IF(fprec(i) .NE. 0) precip = .TRUE.
0093      DO 41 i = 1,2
0094      41      IF(lprec(i) .NE. 0) precip = .TRUE.

```

```

0096      DO 42 i = 1,2
0097      42      IF(block(i) .NE. 0 ) obstruction = .TRUE.
0098      IF((precip .OR. obstruction) .AND. (visible .GE. 60) .AND.
0099      * (wx .NE. 9)) THEN
0100      WRITE(3,483)visible
0101      483      FORMAT(/,3x,'VSBY MUST BE < 0 MILES WITH PCPN OR OBSTR.',3X,I4)
0102      ENDOF
0103
0104      IF((.NOT. precip) .AND. (.NOT. obstruction) .AND. (visible .LE. 60)) THEN
0105      WRITE(3,484)visible
0106      484      FORMAT(/,3x,'VSBY MUST BE > 0 MILES WITH NO PCPN OR OBSTR.',3X,I4)
0107      ENDOF
0108
0109      RETURN
0110      END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 SCODE	533	PIC COM REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	251	PIC COM REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	60	PIC COM REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	884	

ENTRY POINTS

Address	Type	Name	References
0-00000000		OBSTR	284

VARIABLES

Address	Type	Name	Attributes	References
** 94	I+2	I		43 57(2)= 70(2)= 81(2)= 91= 92 93=
**	I+2	J	95= 96	43
**	I+2	K		43
**	L+4	OBSTRUCTION		48 96= 96= 98 104
**	L+4	PRECIP		47 89= 92= 94= 98 104
AP-000000140	I+2	VISIBL		28 44 98 100 104 105
AP-000000180	I+2	WX		28 45 98

ARRAYS

Address	Type	Name	Attributes	Bytes	Dimensions	References
AP-0000000C0 96	I+2	BLOCK		4	(2)	28 41 79(2) 81(2)
AP-000000000 92	I+2	FPRECP		8	(3)	28 40 86(8) 70(3)
AP-000000040 94	I+2	LPRECP		4	(2)	28 39 55(2) 57(2)
AP-000000100 81	I+2	TIME		8	(4)	28 42 57 70

LABELS

Address	Label	References
**	40	91 92#
**	41	93 94#
**	42	95 96#
1-00000000	400'	57 58#
1-0000002F	401'	70 71#
1-00000003	402'	81 82#
1-00000009	403'	100 101#
1-0000000C	404'	105 106#

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/CROSS/CHECK=ALL/EXTEND/CONTINUATION=99 OBSTR
/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)
/DEBUG=(NOSYMBOLS,TRACEBACK)
/STANDARD=(NOSYNTAX,NOSOURCE FORM)
/SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP,MODICTIONARY,SINGLE)
/WARNINGS=(GENERAL,MODEDECLARATIONS,NOULTRIX)
/CONTINUATIONS=99 /CROSS REFERENCE /NOD LINES /EXTEND_SOURCE /FTT
/NOQ FLOATING /I4 /NOMACHINE CODE /OPTIMIZE
```

COMPILATION STATISTICS

```
Run Time:      3.93 seconds
Elapsed Time:  6.23 seconds
Page Faults:   332
Dynamic Memory: 388 pages
```

```

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 C
0003 C      WIND.FOR
0004 C
0005 C      WRITTEN BY : KEN BURK
0006 C
0007 C      PURPOSE :      PROVIDE QUALITY ASSURANCE CHECK FOR THE WIND DIRECTION
0008 C                     AND WIND SPEED FOUND IN THE SURFACE OBSERVATION
0009 C                     DATA FILES.
0010 C                     WIND DIRECTION IS CODED USING A 36 POINT SYSTEM,
0011 C                     1=10 DEGREES, 33=330 DEGREES.  # = NO WIND.
0012 C
0013 C      ALGORITHM :  DECLARE VARIABLES
0014 C                     CHECK FOR LARGE HOURLY CHANGE IN DIRECTION
0015 C                     CHECK FOR LARGE HOURLY CHANGE IN SPEED
0016 C                     RETURN

```

```

0017 C
0018 C   VERSION : 1.0
0019 C
0020 C   DATE :   JUNE 1988
0021 C
0022 C   CALLED FROM :   SFCQA.FOR   (MAIN PROGRAM)
0023 C
0024 C-----
0025 C
0026 C   SUBROUTINE wind(dir,old_dir,spd,old_spd,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   dir           ! CURRENT WIND DIRECITON
0038 C   INTEGER*2   old_dir      ! PREVIOUS WIND DIRECTION
0039 C   INTEGER*2   twind1       ! TEMPORARY WIND DIRECTION
0040 C   INTEGER*2   twind2       ! TEMPORARY WIND DIRECTION
0041 C   INTEGER*2   spd          ! CURRENT WIND SPEED
0042 C   INTEGER*2   old_spd      ! PREVIOUS WIND SPEED
0043 C   INTEGER*2   time(4)      ! DATE YY,MM,DD,HH
0044 C   INTEGER*2   i,j,k        ! ARRAY AND LOOP INDECIES
0045 C
0046 C
0047 C
0048 C   CHECK FOR LARGE HOURLY CHANGE IN WIND DIRECTION
0049 C
0050 C
0051 C   twind1 = dir           ! USE TEMPORARY VARIABLE SO THAT ORIGINAL DATE
0052 C   twind2 = old_dir      ! REMAINS UNCHANGED
0053 C
0054 C
0055 C   CHECK FOR A WIND DIRECTION CHANGE THAT IS LESS THAN 90 DEGREES
0056 C   BUT NUMERICALLY IS MORE THAN 90 DEGREES (E.G. 36 TO 2 IS REALLY
0057 C   A CHANGE OF 38 DEGREES, 360 TO 20, BUT NUMERICALLY IS A CHANGE
0058 C   OF 38, OR 338 DEGREES)
0059 C
0060 C   MODIFICATION; AFTER FURTHER REVIEW, THIS WAS DETERMINED TO BE A
0061 C   USELESS CHECK BECAUSE WIND DIRECTION CHANGES TOO MUCH
0062 C   16-7-88   KW BURK
0063 C
0064 C
0065 C   IF((twind1 .GT. 26 .AND. twind1 .LT. 37) .AND.
0066 C   *   (twind2 .LT. 9)) THEN           ! POTENTIAL PROBLEM
0067 C   twind2 = twind2 + 36                 ! IN WIND DIR CHANGE
0068 C   ENDIF
0069 C
0070 C   IF((twind2 .GT. 26 .AND. twind2 .LT. 37) .AND.
0071 C   *   (twind1 .LT. 9)) THEN           ! POTENTIAL PROBLEM
0072 C   twind1 = twind1 + 36                 ! IN WIND DIR CHANGE
0073 C   ENDIF
0074 C
0075 C   IF(IIABS(twind1 - twind2) .GT. 9) THEN           ! CHECK DIFFERENCE
0076 C
0077 C   WRITE(3,500)(time(i),i=1,4),dir,old_dir
0078 C500   FORMAT(3x,'LARGE HRLY CHANGE IN WIND DIR.',4x,4i2,5x,i2,3x,i2)
0079 C
0080 C   ENDIF
0081 C

```

```

0002 C
0003 C CHECK FOR LARGE HOURLY CHANGE IN WIND SPEED
0004 C
0005
0006 IF(IIABS(spd - old_spd) .GT. 30) THEN
0007
0008 WRITE(3,501)(time(i),i=1,4),spd,old_spd
0009 501 FORMAT(/,3x,'LARGE HRLY CHANGE IN WIND SPD.',4x,4i2,5x,i2,3x,i2)
0010
0011 ENDIF
0012
0013 RETURN
0014 END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	120	PIC COM REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	50	PIC COM REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	20	PIC COM REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	190	

ENTRY POINTS

Address	Type	Name	References
0-00000000		WIND	26#

VARIABLES

Address	Type	Name	Attributes	References
AP-000000040	I*2	DIR		26 37 51
**	I*2	I		44 88(2)=
**	I*2	J		44
**	I*2	K		44
AP-000000080	I*2	OLD_DIR		26 38 52
AP-000000100	I*2	OLD_SPD		26 42 88 88
AP-0000000C0	I*2	SPD		26 41 88 88
**	I*2	TWIND1		39 51=
**	I*2	TWIND2		45 52=

ARRAYS

Address	Type	Name	Attributes	Bytes	Dimensions	References
AP-000000140	I*2	TIME		8	(4)	26 43 88

LABELS

Address	Label	References
1-00000000	501'	88 89#

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/CROSS/LIS/CHECK=ALL/EXTEND WIND

/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)

/DEBUG=(NOSYMBOLS,TRACEBACK)

/STANDARD=(NOSYNTAX,NOSOURCE FORM)

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

/WARNINGS=(GENERAL,NODECLARATIONS,NOLTRIX)

/CONTINUATIONS=19 /CROSS REFERENCE /MOD LINES /EXTEND_SOURCE /F77

/NOG_FLOATING /I4 /NOMACHINE_CODE /OPTIMIZE

COMPILATION STATISTICS

Run Time: 1.73 seconds

Elapsed Time: 4.18 seconds

Page Faults: 624

Dynamic Memory: 326 pages

```

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 C
0003 C PRESSUR.FOR
0004 C
0005 C WRITTEN BY : KEN BURK
0006 C
0007 C PURPOSE : PROVIDE QUALITY ASSURANCE CHECK FOR THE STATION AND
0008 C SEA-LEVEL PRESSURE FOUND IN THE SURFACE OBSERVATION
0009 C DATA FILES.
0010 C
0011 C ALGORITHM : DECLARE VARIABLES
0012 C CHECK STATION PRES. FOR MONTHLY EXTREME RANGES
0013 C CONVERT SEA LEVEL PRES. TO IN. HG
0014 C CHECK SEA LEVEL PRES. FOR MONTHLY EXTREME RANGES
0015 C RETURN
0016 C
0017 C VERSION : 1.0
0018 C
0019 C DATE : JUNE 1988
0020 C
0021 C CALLED FROM : SFCQA.FOR (MAIN PROGRAM)
0022 C
0023 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0024 C
0025 C SUBROUTINE pressur(hgstat,mbsa,time,old_hgstat)
0026 C
0027 C IMPLICIT NONE
0028 C
0029 C
0030 C VARIABLE DECLARATIONS
0031 C
0032 C -----
0033 C TYPE | VARIABLE | COMMENTS
0034 C -----

```

```

0035
0036      INTEGER*2      hgstat      ! STATION PRESSURE IN. MERCURY
0037      INTEGER*2      old_hgstat   ! PREVIOUS STATION PRESSURE
0038      INTEGER*2      mbsaa        ! SEA LEVEL PRESSURE MILLIBARS
0039      INTEGER*2      time(4)      ! DATE YY,MM,HH,DD
0040      INTEGER*2      max_stat(12) ! MONTHLY MAX STATION PRESSURES
0041      INTEGER*2      min_stat(12) ! MONTHLY MIN STATION PRESSURES
0042      INTEGER*2      max_sea(12)  ! MONTHLY MAX SEA LEVEL PRESSURES
0043      INTEGER*2      min_sea(12)  ! MONTHLY MIN SEA LEVEL PRESSURES
0044      INTEGER*2      temp1_pres    ! TEMPORARY PRESSURE VALUE
0045      INTEGER*2      month        ! MONTH IN THE DATE
0046      INTEGER*2      i            ! ARRAY INDEX
0047
0048      REAL*4          temp2_pres   ! TEMPORARY PRESSURE VALUE
0049
0050
0051
0052      DATA max_stat/ 2975, 2975, 2975, 2975, 2975, 2954, 2952, 2955, 2972,
0053 *              2975, 2975, 2975 /
0054
0055      DATA min_stat/ 2818, 2823, 2829, 2849, 2862, 2866, 2865, 2856, 2865,
0056 *              2819, 2843, 2818 /
0057
0058      DATA max_sea / 3112, 3097, 3079, 3064, 3053, 3032, 3030, 3032, 3052,
0059 *              3063, 3090, 3106 /
0060
0061      DATA min_sea / 2898, 2898, 2905, 2926, 2938, 2936, 2941, 2932, 2941,
0062 *              2895, 2919, 2888 /
0063
0064
0065      month = time(2)
0066
0067      C
0068      C CHECK AND MAKE SURE THE STATION PRESSURE FITS INTO THE
0069      C MONTHLY EXTREME RANGE
0070      C
0071
0072      IF(( hgstat .GT. max_stat(month)) .OR.
0073 *      ( hgstat .LT. min_stat(month))) THEN
0074
0075          WRITE(3,800)(time(i),i=1,4),hgstat
0076      800  FORMAT(/,3x,'ERROR STAT. PRES. OUT OF RANGE',4x,4i2,5x,i4)
0077
0078      ENDIF
0079
0080      IF( IABS(old hgstat - hgstat) .GE. 30 ) THEN
0081          WRITE(3,802)old hgstat,hgstat
0082      802  FORMAT(/,3x,'LARGE CHANGE IN STAT. PRESSURE',4X,T4,3X,I4)
0083      ENDIF
0084
0085      C
0086      C CONVERT SEA LEVEL PRESSURE FROM MILLIBARS TO INCHES Hg
0087      C
0088
0089      IF(mbsaa .LT. 1000) THEN          ! PRESSURE IS REALLY > 1000
0090
0091          temp2_pres = FLOATI(mbsaa) / 10.0
0092          temp2_pres = temp2_pres + 1000
0093
0094      ELSE                                ! PRESSURE IS REALLY < 1000
0095
0096          temp2_pres = FLOATI(mbsaa) / 10.0
0097
0098      ENDIF
0099

```

```

0100      temp1_pres = IIFIX(temp2_pres * 0.02953 * 100) ! INTEGER VALUE, IN INCHES Hg
0101
0102      C
0103      C   CHECK AND MAKE SURE SEA LEVEL PRESSURE FITS INTO THE
0104      C   MONTHLY EXTREME RANGE
0105      C
0106
0107      IF(( temp1_pres .GT. max_sea(month)) .OR.
0108      *   ( temp1_pres .LT. min_sea(month))) THEN
0109
0110      WRITE(3,801)(time(i),i=1,4),mbsea
0111      801      FORMAT(/,3x,'ERROR SEA PRES. OUT OF RANGE',6x,4i2,5x,.4)
0112
0113      ENDIF
0114
0115      RETURN
0116      END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	337	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	134	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	116	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	587	

ENTRY POINTS

Address	Type	Name	References
0-00000000		PRESSUR	25

VARIABLES

Address	Type	Name	Attributes	References
AP-000000040	I+2	HQSTAT		25 36 72(2) 75 80 81
**	I+2	I		40 75(2)= 110(2)=
AP-000000080	I+2	MBSEA		25 30 80 91 90 110
**	I+2	MONTH		45 65= 72(2) 107(2)
AP-000000100	I+2	OLD_HQSTAT		25 37 80 81
**	I+2	TEMP1 PRES		44 100= 107(2)
**	R+4	TEMP2 PRES		40 91= 92(2)= 90= 100

ARRAYS

Address	Type	Name	Attributes	Bytes	Dimensions	References
2-000000030	I+2	MAX_SEA		24	(12)	42 50D 107
2-000000060	I+2	MAX_STAT		24	(12)	40 52D 72
2-000000090	I+2	MIN_SEA		24	(12)	43 61D 107
2-000000120	I+2	MIN_STAT		24	(12)	41 55D 72
AP-0000000C0	I+2	TIME		8	(4)	25 30 65 75

LABELS

Address	Label	References
1-00000000	000'	75 70#
1-0000005A	001'	110 111#
1-0000002E	002'	81 02#

KEY TO REFERENCE FLAGS

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

COMMAND QUALIFIERS

FDR/CROSS/LIS/CHECK=ALL/EXTEND PRESSUR

```
/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

COMPILATION STATISTICS

Run Time: 2.90 seconds

Elapsed Time: 4.49 seconds

Page Faults: 669

Dynamic Memory: 331 pages

```

0001 C
0002 C
0003 C      TEMPS.FOR
0004 C
0005 C      WRITTEN BY : KEN BURK
0006 C
0007 C      PURPOSE :   PROVIDE QUALITY ASSURANCE CHECK FOR THE TEMPERATURES
0008 C                  AND RELATIVE HUMIDITY PARAMETERS FOUND IN THE SURFACE
0009 C                  OBSERVATION DATA FILES.
0010 C
0011 C      ALGORITHM :  DECLARE VARIABLES
0012 C                  CHECK THAT DRY > WET > DEW
0013 C                  CHECK IF DRY BULB IN MONTHLY EXTREME RANGE
0014 C                  CHECK FOR LARGE HOURLY CHANGE
0015 C                  CHECK IF WET BULB IN MONTHLY EXTREME RANGE
0016 C                  CHECK FOR LARGE HOURLY CHANGE
0017 C                  CHECK IF DEW POINT IN MONTHLY EXTREME RANGE
0018 C                  CHECK FOR LARGE HOURLY CHANGE
0019 C                  CHECK RELATIVE HUMIDITY FOR LARGE HOURLY CHANGE
0020 C                  RETURN
0021 C
0022 C      VERSION : 1.0
0023 C      MODIFICATIONS :  10-6-88 KW BURK.  DROPPED UPPER BOUNDS, AND RAISED

```

```

0024 C          LOWER BOUNDS ON TEMPERATURE RANGES. WHEN A GROUP OF
0025 C          ADJACENT TEMPS ARE OUT OF RANGE, WRITE MESSAGE FOR FIRST
0026 C          OCCURANCE, THEN LET OTHERS PASS THROUGH.
0027 C
0028 C          DATE :    JUNE 1988
0029 C
0030 C          CALLED FROM :  SFCQA.FOR  (MAIN PROGRAM)
0031 C
0032 C          CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0033 C
0034 C          SUBROUTINE temps(temp_dry,old temp dry,temp_wet,old temp_wet,
0035 C          *              temp_dew,humid,old_humid,time,out_of_range)
0036 C
0037 C          IMPLICIT NONE
0038 C
0039 C
0040 C          VARIABLE DECLARATIONS
0041 C-----
0042 C          TYPE      !   VARIABLE      ! COMMENTS
0043 C-----
0044 C
0045 C
0046 C          INTEGER*2    temp_dry      ! DRY BULB TEMPERATURE
0047 C          INTEGER*2    old temp dry  ! PREVIOUS DRY BULB TEMP
0048 C          INTEGER*2    temp_wet      ! WET BULB TEMPERATURE
0049 C          INTEGER*2    old temp wet  ! PREVIOUS WET BULB TEMP
0050 C          INTEGER*2    temp_dew      ! DEW POINT TEMPERATURE
0051 C          INTEGER*2    humid         ! RELATIVE HUMIDITY
0052 C          INTEGER*2    old humid     ! PREVIOUS RELATIVE HUMIDITY
0053 C          INTEGER*2    time(4)       ! DATE YY,MM,DD,HH
0054 C          INTEGER*2    max_dry_bulb(12) ! MONTHLY MAXIMUMS
0055 C          INTEGER*2    min_dry_bulb(12) ! MONTHLY MINIMUMS
0056 C          INTEGER*2    i             ! ARRAY AND LOOP INDEX
0057 C          INTEGER*2    month         ! MONTH OF DATA
0058 C
0059 C          LOGICAL      out_of_range(3) ! BOOLEAN; IF LAST TEMP WAS OUT
0060 C          ! OF RANGE
0061 C
0062 C          DATA max_dry_bulb / 82, 82, 73, 85, 94, 106, 105, 103, 96, 88, 85, 59 /
0063 C          DATA min_dry_bulb / -13, -13, 16, 31, 38, 43, 49, 50, 35, 16, -3, -17 /
0064 C
0065 C
0066 C          month = time(2)
0067 C
0068 C
0069 C          MAKE SURE DRY BULB > WET BULB > DEW POINT
0070 C
0071 C
0072 C          IF(( temp_dry .LT. temp_wet ) .OR.
0073 C          *   ( temp_wet .LT. temp_dew )) THEN
0074 C
0075 C          IF((temp_dry .NE. 999) .AND. (temp_wet .NE. 999) .AND.
0076 C          *   (temp_dew .NE. 999 )) THEN
0077 C
0078 C          WRITE(3,000)(time(i),i=1,4),temp_dry,temp_wet,temp_dew
0079 C          800 FORMAT(/,3x,'ERROR INVALID DRY,WET,DEW TEMPS.',2x,i2,5x,i3,2x,i3,2x,i3)
0080 C
0081 C          ENDIF
0082 C
0083 C          ENDIF
0084 C
0085 C
0086 C          CHECK DRY BULB TEMPERATURE TO SEE IF IT FITS INTO A
0087 C          MONTHLY EXTREME RANGE
0088 C

```

```

0089      IF((temp_dry .GT. max_dry_bulb(month)) .OR.
0090      * (temp_dry .LT. min_dry_bulb(month))) THEN
0091
0092      IF(temp_dry .NE. 999 .AND. .NOT.(out_of_range(1))) THEN ! MISSING VALUE
0093      ! OR PREVIOUS
0094      out_of_range(1) = .TRUE. ! TEMP WAS OUT OF
0095      WRITE(3,800)(time(i),i=1,4),temp_dry ! RANGE
0096      800      FORMAT(/,3x,'ERROR DRY BULB OUT OF RANGE',7x,4i2,5x,i3)
0097
0098      ENDIF
0099
0100      ELSE
0101
0102      out_of_range(1) = .FALSE. ! VALUE WAS WITHIN RANGE
0103
0104      ENDIF
0105
0106      IF(IIABS(temp_dry - old_temp_dry) .GT. 10) THEN
0107
0108      IF((temp_dry .NE. 999) .AND. (old_temp_dry .NE. 999)) THEN
0109
0110      WRITE(3,801)(time(i),i=1,4),temp_dry,old_temp_dry
0111      801      FORMAT(/,3x,'LARGE HOURLY CHANGE IN DRY BULB',3x,4i2,5x,i3,3x,i3)
0112
0113      ENDIF
0114
0115      ENDIF
0116
0117      C
0118      C CHECK WET BULB TEMPERATURE TO SEE IF IT FITS
0119      C INTO THE MONTHLY EXTREME RANGE
0120      C
0121
0122      IF((temp_wet .GT. max_dry_bulb(month)) .OR.
0123      * (temp_wet .LT. min_dry_bulb(month))) THEN
0124
0125      IF(temp_wet .NE. 999 .AND. .NOT.(out_of_range(2))) THEN ! MISSING VALUE
0126      ! OR THE PREVIOUS
0127      out_of_range(2) = .TRUE. ! TEMP WAS OUT
0128      WRITE(3,802)(time(i),i=1,4),temp_wet ! OF RANGE
0129      802      FORMAT(/,3x,'ERROR WET BULB OUT OF RANGE',7x,4i2,5x,i3)
0130
0131      ENDIF
0132
0133      ELSE
0134
0135      out_of_range(2) = .FALSE. ! VALUE WAS WITHIN RANGE
0136
0137      ENDIF
0138
0139      IF(IIABS(temp_wet - old_temp_wet) .GT. 10) THEN
0140
0141      IF((temp_wet .NE. 999) .AND. (old_temp_wet .NE. 999)) THEN
0142
0143      WRITE(3,803)(time(i),i=1,4),temp_wet,old_temp_wet
0144      803      FORMAT(/,3x,'LARGE HOURLY CHANGE IN WET BULB',3x,4i2,5x,i3,3x,i3)
0145
0146      ENDIF
0147
0148      ENDIF
0149
0150      C
0151      C CHECK DEW POINT TEMPERATURE TO SEE IF IT FITS
0152      C INTO THE MONTHLY EXTREME RANGE
0153      C

```

```

0154 C
0155
0156 IF((temp_dew .GT. max_dry_bulb(month)) .OR.
0157 * (temp_dew .LT. min_dry_bulb(month))) THEN
0158
0159 IF(temp_dew .NE. 999 .AND. .NOT.(out_of_range(3))) THEN ! MISSING VALUE
0160 ! OR PREVIOUS
0161 out_of_range(3) = .TRUE. ! TEMP WAS OUT
0162 WRITE(3,884)(time(i),i=1,4),temp_dew ! OF RANGE
0163 884 FORMAT(/,3x,'ERROR DEW POINT OUT OF RANGE',6x,4i2,5x,i3)
0164
0165 ENDIF
0166
0167 ELSE
0168
0169 out_of_range(3) = .FALSE. ! VALUE WAS WITHIN RANGE
0170
0171 ENDIF
0172
0173 C
0174 C CHECK THE HOURLY CHANGE IN THE RELATIVE HUMIDITY
0175 C
0176
0177 IF(IIABS(humid - old_humid) .GT. 10) THEN
0178
0179 IF((humid .NE. 999) .AND. (old_humid .NE. 999)) THEN
0180
0181 WRITE(3,886)(time(i),i=1,4),humid,old_humid
0182 886 FORMAT(/,3x,'LRG HOURLY CHANGE IN REL HUMID.',3x,4i2,5x,i3,3x,i3)
0183
0184 ENDIF
0185
0186 ENDIF
0187
0188 RETURN
0189
0190
0191 END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	844	PIC COM REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	339	PIC COM REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	88	PIC COM REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	1271	

ENTRY POINTS

Address	Type	Name	References
0-00000000		TEMP5	34j

VARIABLES

Address	Type	Name	Attributes	References
AP-000000100	I+2	HUMID		34 51 177 179 181
**	I+2	I		56 78(2)= 96(2)= 111(2)= 129(2)= 144(2)=
162(2)=		181(2)=		

**	I+2	MONTH	57	68=	98(2)	123(2)	156(2)		
AP-0000001C0	I+2	OLD HUMID	34	52	177	179	181		
AP-000000000	I+2	OLD_TEMP_DRY	34	47	107	109	111		
AP-000000100	I+2	OLD_TEMP_WET	34	49	140	142	144		
AP-000000140	I+2	TEMP_DEW	34	50	72	75	78	156(2)	159
102									
AP-000000040	I+2	TEMP_DRY	34	46	72	75	78	90(2)	93
90									
			107	109	111				
AP-0000000C0	I+2	TEMP_WET	34	48	72(2)	75	78	123(2)	126
129									
			140	142	144				

ARRAYS

Address	Type	Name	Attributes	Bytes	Dimensions	References
2-000000000	I+2	MAX_DRY_BULB		24	(12)	54 62D 90 123
150						
2-000000100	I+2	MIN_DRY_BULB		24	(12)	55 63D 90 123
150						
AP-000000240	L+4	OUT_OF_RANGE		12	(3)	34 59 93 95=
103=						
				120	128=	130= 159 161=
				169=		
AP-000000200	I+2	TIME		8	(4)	34 53 66 78
90						
				111	120	144 162 181

LABELS

Address	Label	References
1-000000030	000'	90 97#
1-000000060	001'	111 112#
1-000000090	002'	120 130#
1-0000000C1	003'	144 145#
1-0000000F4	004'	162 163#
1-000000120	005'	181 182#
1-000000000	000'	78 79#

```

-----
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/CROSS/LIS/CHECK=ALL/EXTEND TEMPS

/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)
/DEBUG=(NOSYMBOLS,TRACEBACK)
/STANDARD=(NOSYNTAX,NOSOURCE FORM)
/SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP,MODDICTIONARY,SINGLE)

```



```

0053 C
0054
0055 IF(drops .EQ. 49) THEN
0056
0057 WRITE(3,700)(time(i),i=1,4),drops
0058 700 FORMAT(/,3x,'INVALID PRECIPITATION AMOUNT',5x,4i2,5x,i2)
0059
0060 ENDIF
0061
0062 IF((drops .GT. 4) .AND. (drops .NE. 50)) THEN ! 4 TENTHS OF AN INCH
0063
0064 WRITE(3,701)(time(i),i=1,4),drops
0065 701 FORMAT(/,3x,'LARGE AMOUNT OF RAIN PRECIP.',5x,4i2,5x,i2)
0066
0067 ENDIF
0068
0069 C
0070 C CHECK AMOUNT OF SNOW EQUIVALENT (MELTED) FOR VALIDITY, AND FOR
0071 C A REASONABLE HOURLY AMOUNT
0072 C
0073
0074 IF(flakeq .EQ. 49) THEN
0075
0076 WRITE(3,700)(time(i),i=1,4),flakeq
0077
0078 ENDIF
0079
0080 IF((flakeq .GT. 4) .AND. (flakeq .NE. 50)) THEN ! 4 TENTHS OF AN INCH
0081
0082 WRITE(3,702)(time(i),i=1,4),flakeq
0083 702 FORMAT(/,3x,'LARGE AMT. OF SNOW(EQ) PRECIP.',4x,4i2,5x,i2)
0084
0085 ENDIF
0086
0087 C
0088 C CHECK AMOUNT OF SNOW (UNMELTED) FOR VALIDITY, AND FOR
0089 C A REASONABLE HOURLY AMOUNT
0090 C
0091
0092 IF(flakun .EQ. 49) THEN
0093
0094 WRITE(3,700)(time(i),i=1,4),flakun
0095
0096 ENDIF
0097
0098 IF((flakun .GT. 30) .AND. (flakun .NE. 50)) THEN ! 3 INCHES
0099
0100 WRITE(3,703)(time(i),i=1,4),flakun
0101 703 FORMAT(/,3x,'LARGE AMT. OF SNOW(UN) PRECIP.',4x,4i2,5x,i2)
0102
0103 ENDIF
0104
0105 C
0106 C CHECK TO MAKE SURE THAT IF AN AMOUNT OF PRECIP. IS RECORDED,
0107 C THEN A TYPE OF PRECIP. IS ALSO RECORDED
0108 C
0109
0110 IF(drops .NE. 0 .AND. (liquid(1) .EQ. 0 .OR. liquid(2) .EQ. 0) ) THEN
0111 WRITE(3,704)(time(i),i=1,4),drops,liquid(1),liquid(2)
0112 704 FORMAT(/,3x,'PRECIP AMT. W/O TYPE OF PRECIP',4x,4i2,5x,i2,3x,2i1)
0113 ENDIF
0114 IF(flakeq .NE. 0 .OR. flakun .NE. 0) .AND.
0115 * ((frozen(1) .EQ. 0 .OR. frozen(2) .EQ. 0 .OR. frozen(3) .EQ. 0)) THEN
0116 WRITE(3,705)(time(i),i=1,4),flakeq,flakun,(frozen(i),i=1,3)
0117 705 FORMAT(/,3x,'PRECIP AMT. W/O TYPE OF PRECIP',4x,4i2,5x,i2,3x,3i1)

```

```

0118      ENDIF
0119
0120      C
0121      C      CHECK TO MAKE SURE THAT IF A TYPE OF PRECIP. IS RECORDED,
0122      C      THEN AN AMOUNT IS ALSO RECORDED
0123      C
0124      IF((liquid(1) .NE. 0 .OR. liquid(2) .NE. 0) .AND. drops .EQ. 0) THEN
0125          WRITE(3,706)(time(i),i=1,4),liquid(1),liquid(2),drops
0126      706      FORMAT(/,3X,'TYPE OF PRECIP W/O PRECIP AMT.',4X,4I2,5X,2I1,3X,I2)
0127      ENDIF
0128      IF((frozen(1) .NE. 0 .OR. frozen(2) .NE. 0 .OR. frozen(3) .NE. 0) .AND.
0129      *      (flakeq .EQ. 0 .OR. flakun .EQ. 0)) THEN
0130          WRITE(3,707)(time(i),i=1,4),(frozen(i),i=1,4),flakeq,flakun
0131      707      FORMAT(/,3X,'TYPE OF PRECIP W/O PRECIP AMT.',4X,4I2,5X,3I1,3X,2I2)
0132      ENDIF
0133
0134      RETURN
0135
0136      END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	948	PIC COM REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	392	PIC COM REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	68	PIC COM REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	1408	

ENTRY POINTS

Address	Type	Name	References
0-00000000		PRECIP	304

VARIABLES

Address	Type	Name	Attributes	References
AP-00000040 111	I+2	DROPS		30 41 55 57 82(2) 84 110
AP-00000080 110	I+2	FLAKEQ	124 125	30 42 74 76 80(2) 82 114
AP-000000C0 110	I+2	FLAKUN	128 130	30 43 92 94 98(2) 100 114
** I+2 I 100(2)= 111(2)=			128 130	47 57(2)= 64(2)= 76(2)= 82(2)= 94(2)= 110(4)= 125(2)= 130(4)=

ARRAYS

Address	Type	Name	Attributes	Bytes	Dimensions	References
AP-00000140 128(3)	I+2	FROZEN		6	(3)	30 45 114(3) 116
				130		

12

1

1

E.34

```

0076
0077      ENDIF
0078
0079      C
0080      C      CHECK FOR ABNORMALLY HIGH VALUE
0081      C
0082
0083      IF(radiation .GT. 200 .AND. radiation .NE. 500) THEN
0084          WRITE(3,601)(time(i),i=1,4),radiation
0085      601      FORMAT(/,3x,'ABNORMALLY HIGH SOLAR RADIATION',3X,4I2,5X,I6)
0086      ENDIF
0087
0088      C
0089      C      CHECK FOR LARGE RISES AFTER NOON, AND LARGE DROPS AT TIMES BEFORE
0090      C      NOON
0091      C
0092
0093      IF(radiation .NE. 500 .AND. old_rad .NE. 500)
0094      *      delta_rad = radiation - old_rad
0095
0096      IF(hour .LE. 12 .AND. delta_rad .LT. -20) THEN
0097          WRITE(3,602)(time(i),i=1,4),radiation,old_rad
0098      602      FORMAT(/,3x,'ABNORMAL DROP IN SOLAR RAD.',7X,4I2,5X,I3,3X,I3)
0099      ENDIF
0100      IF(hour .GT. 12 .AND. delta_rad .GT. 20) THEN
0101          WRITE(3,603)(time(i),i=1,4),radiation,old_rad
0102      603      FORMAT(/,3x,'ABNORMAL RISE IN SOLAR RAD.',7X,4I2,5X,I3,3X,I3)
0103      ENDIF
0104
0105      RETURN
0106      END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	486	PIC COM REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	183	PIC COM REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	24	PIC COM REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	687	

ENTRY POINTS

Address	Type	Name	References
0-00000000		SOLAR	307

VARIABLES

Address	Type	Name	Attributes	References
2-00000000	I=2	DELTA_RAD		43 93= 96 100
**	I=2	HOUR		48 56= 60(2) 70(2) 96 100
**	I=2	I		47 63(2)= 73(2)= 84(2)= 97(2)= 101(2)=
**	I=2	MONTH		45 55= 58(2)
AP-00000000	I=2	OLD_RAD		30 42 93(2) 97 101
AP-00000040	I=2	RADIATION		30 41 60 63 70 73 83(2)
84			93(2) 97 101	

1


```

0001      RETURN
0002      END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	242	PIC COM REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	3	PIC COM REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	52	PIC COM 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		36 59= 66
2-0000000A	I*2	CMIN		37 60= 64
2-0000000C	I*2	CSEC		38 61= 63
2-00000010	R*4	CTIME		41 67= 76
**	R*4	RHOURS		42 65= 67
**	R*4	RMIN		43 64= 67
**	R*4	RSEC		44 63= 67
2-00000000	CHAR	SYSTIME		46 63A 59 68 61
AP-00000040	R*4	WAIT_TIME		26 48 76

LABELS

Address	Label	References
0-000000DC	1	73# 78

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	References
R*4	FOR\$SECONDS	76
	FOR\$TIME_T_DS	63

```

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/CROSS/CHECK=ALL/EXTEND 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 /MOD LINES /EXTEND_SOURCE /F77

/NOG_FLOATING /I4 /NOMACHINE_CODE /OPTIMIZE

COMPILATION STATISTICS

Run Time: 2.13 seconds

Elapsed Time: 5.84 seconds

Page Faults: 659

Dynamic Memory: 334 pages

DISTRIBUTION

No. of
Copies

OFFSITE

2 DOE/Office of Scientific and
Technical Information

ONSITE

DOE Richland Operation Office

J. J. Sutey

32 Pacific Northwest Laboratory

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