Hanford Meteorological Station Computer Codes

**Volume 10 - The ARCHIVE
Computer Code**

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

August 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 program 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 of 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 to DOE and DOE contractors from the
Office of Scientific and Technical Information, P.O. Box 62, Oak Ridge, TN 37831;
prices available from (615) 576-8401. FTS 626-8401.

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

NTIS Price Codes, Microfiche A01

Printed Copy

<u>Pages</u>	<u>Price Codes</u>
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

HANFORD METEOROLOGICAL STATION
COMPUTER CODES

Volume 10 - The ARCHIVE Computer Code

G. L. Andrews
K. W. Burk

August 1989

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

Pacific Northwest Laboratory
Richland, Washington 99352

PREFACE

This report is the tenth 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 and output, a detailed program description and flow chart, and a source listing of the code. A basic knowledge of the VAX VMS operating system and the FORTRAN computer language is assumed throughout these volumes.

The first seven volumes presented the data entry programs used at the HMS for archiving and checking various types of data for errors. All of these programs use a data entry form to facilitate data input, and each code calls the same series of subroutines that control and display the data entry form at the user's terminal. The subroutines were generated from the program FORMEDIT, which was developed by John Wiberg of the Computer Science staff in the Engineering Physics Department at the Pacific Northwest Laboratory. An informal user's guide to the FORMEDIT program that describes how to create, modify, and eliminate data entry forms is available from the HMS system manager. A copy of the FORMEDIT source code and an executable version are kept on the HMS Digital Equipment Corporation (DEC) VAX 11/750 computer system in the subdirectory [FCST.DATA_ENTRY].

The eighth volume describes the REVIEW computer code, which is used to display on a color graphics terminal various meteorological variables from the HMS data base. The ninth volume describes and documents the quality assurance computers codes used at the HMS on a routine basis. An eleventh volume (planned for FY 1990) will describe the MPG computer code, which is used to archive hourly atmospheric mixing heights and hourly 50-ft maximum wind speed and direction.

SUMMARY

The ARCHIVE program is used to

- convert selected hourly meteorological data from binary form to formatted ASCII form
- organize the converted data into monthly files.

There are seven monthly files, which contain the meteorological data from the following sources:

- selected data from the HMS 410-ft tower
- selected data from the 400A 200-ft tower
- selected data from the 300A 200-ft tower
- selected data from the 100N 200-ft tower
- 10-m wind data from the telemetry network
- 2-m temperature data from telemetry network
- precipitation data from the telemetry network.

The monthly files are eventually merged into yearly files that are stored on a removable disk labelled HMS_DATA. The ARCHIVE program executes every night at 12:40 a.m. Pacific Standard Time (PST) (or 00:40 on a 24-hour clock) via the ARCHIVE.COM batch process.

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, for his continued support and to Ora Gifford, the HMS computer systems specialist, for his helpful suggestions and ideas.

CONTENTS

PREFACE	iii
SUMMARY	v
ACKNOWLEDGMENTS	vii
INTRODUCTION	1
DESCRIPTION OF THE HMS METEOROLOGICAL DATA BASES	3
THE ARCHIVE COMPUTER CODE	5
PROGRAM OVERVIEW	5
PROGRAM INITIATION	5
DESCRIPTION OF INPUT FILES	7
DESCRIPTION OF OUTPUT FILES	8
LIMITATIONS	9
TROUBLE-SHOOTING THE ARCHIVE COMPUTER CODE	11
BATCH FILE PROBLEMS/FAILURES	11
DATA PROBLEMS/FAILURES	11
PROGRAM PROBLEMS/FAILURES	12
REFERENCES	13
APPENDIX A - DESCRIPTION AND FLOWCHARTS OF THE ARCHIVE PROGRAM	A.1
APPENDIX B - DESCRIPTION AND FLOWCHART OF THE ARCHIVE COMMAND FILE	B.1
APPENDIX C - DESCRIPTION AND LISTING OF INPUT FILE CONTENTS	C.1
APPENDIX D - DESCRIPTION AND LISTING OF OUTPUT FILE CONTENTS	D.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 (e.g., support to the emergency response facilities) 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 the Pacific Northwest Laboratory (PNL) for the U.S. Department of Energy.

Meteorological data are collected from various sensors and instruments located on and off the Hanford Site (Figure 1). Data are averaged on an hourly and 15-min bases and are sent to the HMS via microwave transmission or dedicated data line. The data are separated at the HMS into an hourly and a 15-min data base. Each data base contains daily files in binary format (see Andrews 1988 for details). These data bases are stored and maintained on the Digital Equipment Corporation (DEC) VAX VMS 11/750 computer system located at the HMS (hereafter referred to as the HMS computer system).

The purpose of the ARCHIVE computer program is twofold:

- convert selected hourly binary data into formatted ASCII data
- organize the converted data into monthly files.

Formatted ASCII files are easier to access on a routine basis. The program is executed once a day and is initiated from a command file that submits itself to the SYS\$BATCH queue on a daily basis. The monthly files are stored on the HMS computer's fixed hard disk and are merged into yearly files (located on removable disk packs) at the end of each year.

This report describes the data bases maintained at the HMS, gives an overview of the ARCHIVE program, describes input and output files accessed by the ARCHIVE program, provides a description of program initiation, and discusses the limitations of the ARCHIVE program. A section on trouble-shooting is included. In addition, the appendixes contain flow charts, detailed descriptions, and source code listings for the ARCHIVE program and related subroutines. A description of the ARCHIVE command file and the data input and output files completes the report.

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

FIGURE 1. Distribution of Meteorological Sensors from the HMS Atmospheric Monitoring Network

DESCRIPTION OF THE HMS METEOROLOGICAL DATA BASES

The HMS computer stores and maintains two meteorological data bases. One data base contains data averaged over each 15-min time period and the other contains data averaged over each hour (Andrews 1988). However, the ARCHIVE computer program only archives hourly meteorological data. A discussion of the hourly data base follows.

These data are collected from the HMS monitoring network (Figure 1). [Glantz and Islam (1988) have published a detailed discussion of the Hanford Meteorology Monitoring Program.] Each data base is composed of daily files and is located on the HMS computer's fixed hard disk (community disk). The hourly data base is located in the [ERDSSYS.DAT.SENSORS] subdirectory. The hourly daily files are named as follows:

- MONDAYR.HLG (e.g., MAR189.HLG or MAR1589.HLG).

MON is the three-character abbreviation of the month (e.g., MAR for March), DA is the numerical representation of the day of the month (e.g., 1 for first day in March), and YR is the numerical representation of the last two digits of the year (e.g., 89 for 1989). Approximately 30 days of the most current hourly data are kept on the HMS computer. Data beyond 30 days are backed up onto magnetic tape and stored at the HMS.

THE ARCHIVE COMPUTER CODE

The ARCHIVE computer program is used at the HMS to automatically archive selected hourly meteorological data into formatted, ASCII monthly files. The program is initiated through a command file that submits itself to the SYS\$BATCH queue on a daily basis. The resultant monthly output files are eventually merged into yearly files.

PROGRAM OVERVIEW

The ARCHIVE computer program is initiated every night at 12:40 a.m. PST (or 00:40 on a 24-hour clock) by the ARCHIVE.COM command file (see section "PROGRAM INITIATION" for further details) via the SYS\$BATCH queue. The program then begins to archive the previous day's data. For example, the ARCHIVE.COM file was executed on January 1, 1989, at 12:40 a.m. PST, which in turn initiated the ARCHIVE program that archived the hourly data collected on December 31, 1988.

The ARCHIVE program essentially consists of two DO loops (see Appendix A for details on the ARCHIVE program). The outer loop moves through seven of the eight input files. These files provide internal identification of each hourly variable to be archived. The eighth input file is accessed independently of the outer loop. The inner loop moves through 24 hours of data for each input file. For each hour, data are obtained from the hourly binary files and converted to formatted, ASCII data and then appended to the appropriate monthly output file. After the ARCHIVE program is finished, control is returned to the ARCHIVE.COM command file where it resubmits itself to the SYS\$BATCH queue for execution at 12:40 a.m. PST the next night (e.g., January 2, 1989).

PROGRAM INITIATION

This section discusses how the ARCHIVE program is initiated. We assume you have some basic knowledge of the VAX VMS batch processing structure and operating system command language.

The ARCHIVE program is initiated from the ARCHIVE.COM batch (command) file. This batch file contains VAX VMS operating system commands, which

perform a variety of functions related to the ARCHIVE program (see Appendix B for details on the ARCHIVE.COM command file). The ARCHIVE.COM file resides in the SYS\$BATCH queue until 12:40 a.m. PST every night. At precisely 12:40 a.m. PST (HMS computer time), the command file is executed, which in turn initiates the ARCHIVE program. When the ARCHIVE program terminates, control is returned to the ARCHIVE.COM command file. At this point the command file determines the next date for execution; the next time of day always remains 12:40 a.m. PST.

Theoretically the ARCHIVE.COM file should resubmit itself even after a computer system crash. Therefore no human interaction is required after the initial submission of the ARCHIVE.COM command file. If the ARCHIVE.COM batch file is missing from the SYS\$BATCH queue, you will have to submit ARCHIVE.COM manually. However, you need to determine if the command process has already executed for the current day, i.e., archived the previous day's data. If it has already executed, follow these procedures to resubmit the ARCHIVE.COM file to the SYS\$BATCH queue:

1. log in to the HMS computer (USERNAME: FCST and PASSWORD:)
2. set default directory to [FCST.ARCHIVE]; type SD [FCST.ARCHIVE] and depress the RETURN key
3. type SUBMIT/RESUBMIT/KEEP/LOG_FILE=[FCST.ARCHIVE]ARCHIVE.LOG/ and depress the RETURN key; continue typing in AFTER=1-JAN-1989:00:40 and depress the RETURN key
4. check the SYS\$BATCH queue for the ARCHIVE process; type BAT and depress the RETURN key.

Please note that the date following the AFTER qualifier is only an example. The current date should be inserted; however, the time always remains 00:40 (12:40 a.m. PST).

If the process has not executed, follow the same procedures except change the third step to

3. type SUBMIT/RESUBMIT/KEEP/LOG_FILE=[FCST.ARCHIVE]ARCHIVE.LOG ARCHIVE and depress the RETURN key.

Notice you leave off the AFTER qualifier.

Once the ARCHIVE.COM command file is manually entered into the SYS\$BATCH queue, there is no need for human interaction unless the process is found to be missing from the SYS\$BATCH queue.

The ARCHIVE.COM command file creates a file called ARCHIVE.LOG, which is stored in the [FCST.ARCHIVE] subdirectory. This log file contains valuable information regarding the execution of the ARCHIVE.COM process. If there are any problems during the execution of the ARCHIVE.COM process, they are logged in the ARCHIVE.LOG file. This file should be reviewed as a first step during the trouble-shooting phase.

DESCRIPTION OF INPUT FILES

The ARCHIVE program accesses eight input files during execution. Seven of these files (the first seven in Table 1) reside in the [FCST.ARCHIVE] subdirectory. The eighth, the HOURLY_MIX_PKGST.DAT file, resides in the [FCST.DAT] subdirectory. The file names and a general description of their contents are given in Table 1. These file names do not change; however, it is possible that the identifications contained within the files may change as more sensors are added to the monitoring network. The first record of the first seven files contains a header describing the file contents. The records following the header contain two items:

- a numeric identification
- a brief description of the numeric identification.

TABLE 1. Description of Input Files Accessed by ARCHIVE

<u>File Name</u>	<u>Contents</u>
TOWER_400_IDS.DAT	Ids for HMS 410-ft tower data
TOWER_200_400A_IDS.DAT	Ids for 400 Area 200-ft tower data
TOWER_200_300A_IDS.DAT	Ids for 300 Area 200-ft tower data
TOWER_200_100N_IDS.DAT	Ids for 100 N 200-ft tower data
TELEMETRY_WINDS_IDS.DAT	Ids for telemetry wind network data
TELEMETRY_TEMPS_IDS.DAT	Ids for telemetry temperature network data
TELEMETRY_PRECIP_IDS.DAT	Ids for telemetry precipitation network data
HOURLY_MIX_PKGST.DAT	Hourly mix depth and peak gust information

The numeric identification follows an I3 format as described in the FORTRAN computer language. The header and the brief description are skipped during the reading process in the ARCHIVE program.

The eighth input file (HOURLY_MIX_PKGST.DAT) contains hourly atmospheric mixing depth and 50-ft maximum hourly wind speed information. This information is entered by the on-duty meteorologist on an hourly basis using the MPG computer code (see the Preface). The ARCHIVE program needs this information because the hourly data base does not include information on the atmospheric mixing depth and 50-ft maximum wind speed data. The HOURLY_MIX_PKGST.DAT file is deleted at the end of the archiving process. A new file is created when the MPG data entry program is executed on the first hour of each day.

Appendix C contains a detailed description of the input files and a listing of each file.

DESCRIPTION OF OUTPUT FILES

Seven monthly output files are accessed by ARCHIVE during execution. These files are stored in the [FCST.DAT] subdirectory until the end of the year. At the end of the year, the files are merged into a yearly file, which is transferred to the HMS_DATA removable hard disk; see Andrews (1988) for details on the directory structure of the HMS_DATA removable hard disk. The output file names and a general description of their contents are listed in Table 2.

TABLE 2. Description of Output Files Accessed by ARCHIVE

<u>File Name</u>	<u>Contents</u>
TOWER_400MMM.DAT	Selected data from the HMS 410-ft tower
TOWER_200_400AMMM.DAT	Selected data from the 400A 200-ft tower
TOWER_200_300AMMM.DAT	Selected data from the 300A 200-ft tower
TOWER_200_100NMMM.DAT	Selected data from the 100N 200-ft tower
TELEMETRY_WINDSMMM.DAT	10-m wind data from telemetry network
TELEMETRY_TEMPMMM.DAT	2-m temperature data from telemetry network
TELEMETRY_PRECIPMMM.DAT	Precipitation data from telemetry network

The MMM in each file name represents the three-character abbreviation of the month (e.g., JAN for January). A new output file is created on the first day of each month (i.e., when data from the first day are archived). See Appendix D for detailed information on the format and file contents of each monthly output file.

LIMITATIONS

The ARCHIVE program is limited to archiving selected meteorological data (i.e., data related to atmospheric dispersion and the general weather conditions on the Hanford Site) from the hourly data base as described in the section "DESCRIPTION OF THE HMS METEOROLOGICAL DATA BASES." Fifteen-minute data are not archived by the ARCHIVE program. Not all meteorological data available from the hourly data base are archived. However, as the data base expands (i.e., as more sensors are added to monitoring network) more data can be archived if desired by modifying the ARCHIVE computer code to accommodate more input files and output files.

Quality assurance is very limited in the ARCHIVE program. The program performs gross error checks for some data but does not perform any quality assurance check for other data. However, if the program detects missing data it fills in the data with a series of 9s. Quality assurance is done more thoroughly at the end of each month with the HMS quality assurance codes (Burk and Andrews 1989).

TROUBLE-SHOOTING THE ARCHIVE COMPUTER CODE

This section provides information for resolving some problems that may occur with the ARCHIVE process and HMS computer system. Basically, there are three types of failure or problem areas: batch file (ARCHIVE.COM), data, and/or program (ARCHIVE.EXE). The following sections discuss a described problem or failure and provide a suggested resolution to the problem or failure.

The ARCHIVE.COM process creates a log file during execution. If errors occur during the process they are logged in a file called ARCHIVE.LOG, which is stored in the [FCST.ARCHIVE] subdirectory. This log file may provide useful information during trouble-shooting.

BATCH FILE PROBLEMS/FAILURES

Problem: The ARCHIVE.COM batch file is missing from the SYS\$BATCH queue.

Recovery: Check the output files in [FCST.DAT] to see if the data have been already archived. Then follow the instructions outlined in the "PROGRAM INITIATION" section of this report.

Problem: The ARCHIVE.COM batch file is in the SYS\$BATCH queue, but the date or time for the next execution is incorrect.

Recovery: First you must delete the incorrect ARCHIVE.COM file from the SYS\$BATCH queue. From the FCST account simply type BAT and depress the RETURN key. A list of batch files appears. Write down the job number for the ARCHIVE process and then type DELETE/ENTRY=job number SYS\$BATCH and depress the RETURN key. Now follow the first set of instructions given in the "PROGRAM INITIATION" section of this report.

DATA PROBLEMS/FAILURES

Problem: Data are not getting into the monthly files or the files are filled with a series of 9s.

Recovery: The hourly HMS meteorological data base is probably not receiving data or the data are corrupt. Check to see if the data-gathering processes are running. This recovery action requires

specific knowledge regarding the data-gathering processes. Seek help from Ora Gifford (373-3216) or the EMS system manager.

Problem: Data contain a series of *s.

Recovery: Stars indicate a problem with formatting in the WRITE_DATA subroutine. The WRITE_DATA subroutine may require modification to handle a problem not addressed in the initial development and testing phases. Determine which variable is having the problem. If the variable is exceeding a reasonable value, some gross error-checking may have to be added to the WRITE_DATA subroutine. This recovery action requires knowledge of the FORTRAN computer language.

PROGRAM PROBLEMS/FAILURES

Problem: Program problems may develop during the execution of the ARCHIVE.EXE process. Although the code has been tested thoroughly, problems can arise if input files are missing from their designated subdirectory. The typical error message associated with this kind of problem is "FILE-NOT-FOUND."

Recovery: If input files are missing from their expected subdirectory they will have to be built. See Appendix C for a list of the contents of each input file.

REFERENCES

- Andrews, G. L. 1988. The Hanford Meteorological Data Collection System and Data Base. PNL-6509, Pacific Northwest Laboratory, Richland, Washington.
- Glantz, C. S., and M. M. Islam. 1988. The Data Collection Component of the Hanford Meteorology Monitoring Program. PNL-6684, Pacific Northwest Laboratory, Richland Washington.
- Snell, W. 1982. Nuclear Regulatory Commission Staff Computer Programs for Use with Meteorological Data. NUREG-0917, U.S. Nuclear Regulatory Commission, Washington, D.C.

APPENDIX A

DESCRIPTION AND FLOWCHARTS OF THE ARCHIVE PROGRAM

APPENDIX A

DESCRIPTION AND FLOWCHARTS OF THE ARCHIVE PROGRAM

This appendix is a detailed discussion of the ARCHIVE program and presents a flowchart of the ARCHIVE program showing the logical sequence of events. In addition, the subroutines associated with the ARCHIVE program are discussed and associated flowcharts are presented. Figure A.1 shows the subroutine hierarchy structure of the ARCHIVE program. Source code listings of the ARCHIVE code and related subroutines are presented at the back of this appendix. The source code listings contain comments and information regarding the function and purpose of each subroutine.

PROGRAM ARCHIVE

The ARCHIVE program begins by initializing all program variables (Figure A.2). Several DATA blocks are also initialized. Following the initialization stage, the program opens a temporary file (TEMP.OUT). This file is used as an output file where 410-ft tower data are stored. The file is printed at the end of the ARCHIVE.COM process. The information is used by the on-duty meteorologist to summarize 410-ft tower data. After the TEMP.OUT file is opened, the program enters into one of the two main DO loops. The first DO loop is referred to as the outer loop. The outer loop is initialized at one and terminates at seven, corresponding to the seven input files containing the internal identifications. The eighth input file is accessed during the archive of the 410-ft tower data and telemetry wind information. Once inside the outer loop the program opens an input file and reads the first record. The first record of each file contains a header and is not needed for program execution. The program begins reading the internal identifications until it reaches the end-of-file marker. As the program reads each identification, it increments a counter which is used later in the program. The input file containing the identifications is closed once the identifications are read in.

At this point, the program uses a VAX FORTRAN system subroutine (IDATE) to obtain the current date. Next, one day is subtracted from the date using

FIGURE A.1. Subroutine Hierarchy of the ARCHIVE Program

FIGURE A.2. Flowchart of the ARCHIVE Program

the subroutine SUBTRACT_A_DAY. This is done because the ARCHIVE.COM process executes one day ahead of the actual data to be archived. Once the date is adjusted, the corresponding output file is opened. For example, for the input file TOWER_400_IDS.DAT, the output file TOWER_400MMM.DAT (where MMM is the three-character abbreviation of the month) is opened. If an error occurs, the program checks to determine if the error is "FILE-NOT-FOUND" (code 29). If the numeric error is 29, the program assumes a new month has occurred and opens a new monthly file. If the numeric error is not equal to 29, an error message is sent to the log file. After the output file is opened/created, the file containing the atmospheric mixing depth and peak gust is opened for input.

Next, the program enters the second main DO loop, which is referred to as the inner loop. The inner DO loop initializes at 1 and terminates at 24, corresponding to the hours of the day. Next, the variables used in the GET_EMS_DATA subroutine argument list are initialized. If the inner loop's index is equal to 24, adjustments are made to the date and time because the HMS hourly data base is on an 0 to 23 hour format as opposed to a 1 to 24 hour format. Once the date and time are adjusted, the subroutine GET_EMS_DATA is called to obtain the selected hourly meteorological data and return that data to the main program. If the inner loop's index is equal to 24, the date and time are then re-adjusted to the original date and time. At this point the selected hourly data are written to the appropriate output file through the subroutine WRITE_DATA. This process continues until all the input and output files have been processed.

SUBROUTINE IDATE

The IDATE subroutine is a VAX VMS FORTRAN system subroutine that obtains the current date in integer from the computer's clock. A flowchart and source code listing are not presented in this report. For further details see the VAX VMS FORTRAN user's guide.

SUBROUTINE SUBTRACT A DAY

The SUBTRACT_A_DAY subroutine is used to subtract one day from the current date. A flowchart of the SUBTRACT_A_DAY subroutine is presented in Figure A.3. The ARCHIVE PROGRAM passes the current year, month, and day to this subroutine.

FIGURE A.3. Flowchart of the SUBTRACT-A-DAY Program

After the date is adjusted, the modified year, month, and day are passed back to the ARCHIVE program. This modification is done because the ARCHIVE program is initiated one day ahead of data to be archived. For example, the ARCHIVE program was initiated on January 1, 1989 at 12:40 a.m. PST (or 00:40) but one day was subtracted to archive the previous day's (December 31, 1988) data. Figure A.3 is a flowchart showing the logical sequence of activities performed by the SUBTRACT_A_DAY subroutine. The code begins by checking for a beginning of year condition. If so, the date is adjusted to December 31 of the previous year. If it is not the beginning of the year, the code checks for a beginning of the month condition. If so, the code checks for leap year and makes an adjustment. Then the code adjusts the date to the last day of the previous month. If the beginning of the month condition is not true, the code subtracts one day. Once the date has been adjusted, the SUBTRACT_A_DAY subroutine passes control and the modified date back to the ARCHIVE program.

SUBROUTINE GET EMS DATA

The GET_EMS_DATA subroutine's function is to obtain selected meteorological data from the hourly data base. A flowchart of the GET_EMS_DATA subroutine is presented in Figure A.4. The ARCHIVE program passes the desired date, time, type of desired data (hourly) flag, variable identifications, and the number of identifications to the GET_EMS_DATA subroutine. After obtaining the selected meteorological data the GET_EMS_DATA subroutine passes these data back to the ARCHIVE program.

GET_EMS_DATA begins by declaring all subroutine variables and initializing some data arrays used in the subroutine. The type of data desired is then determined and the appropriate subroutine for building the necessary file name is called. If 15-min data are desired the GET_QLG_FILENAME subroutine is initiated. If hourly data are desired the GET_HLG_FILENAME is initiated. For the ARCHIVE program, only the GET_HLG_FILENAME is accessed. Once the file name is generated the GET_EMS_DATA subroutine opens this file. If any errors occur during the opening process, an error message is sent to the ARCHIVE.LOG file. At this point, the first record in the data file is read to obtain an initial date and time to begin the search for the desired date and time [see Andrews (1988) for a detailed description of the 15-min and hourly data base files]. The search takes place in a DO WHILE loop. The loop continues to search through the file until the date and time from the file match the desired date and time. When this condition is met, the subroutine moves out of the DO WHILE loop. Next, all of the data associated with the desired data and time are read into the several data arrays (e.g., INTERNAL_ID, PARAM_NAME, MEAN_VALUE, etc.). A check is performed for data that are missing (i.e., MEAN_VALUE, SIGMA, and NUMBER_DATA_POINTS are 0.0). If data are missing the subroutine assigns a 999.9 value to the MEAN_VALUE. After looping through all of the available data, two DO loops in conjunction with an IF statement perform the task of sorting through all of the data and selecting only the information that is desired. When a selected variable is found the MEAN_VALUE is stored in an array called DESIRED_DATA. This process continues until all the desired data are collected and stored in the DESIRED_DATA array. Next, the record and block counters are incremented. If the block counter is less than 2 the subroutine jumps back to the point of

FIGURE A.4. Flowchart of the GET_EMS_DATA Program

reading the date and time of the next block of data. There are two blocks of data associated with each date and time [see Andrews (1988) for more details on the file structure]. If the block counter is equal to 2 the subroutine jumps to the label 1000 and checks to see if an error occurred during the reading of the desired data file. If an error occurred the elements of the DESIRED_DATA array are given the value 999.9. Finally, the subroutine closes all files and terminates by passing control back to the ARCHIVE program.

SUBROUTINES GET_HLG_FILENAME AND GET_QLG_FILENAME

The GET_HLG_FILENAME and GET_QLG_FILENAME subroutines are identical except where the final file name is assigned to the variable that passes the generated file name back to the ARCHIVE program. The description given in this paragraph will include both subroutines even though the ARCHIVE program accesses only the hourly data base. Where there are differences the discussion will point them out. A flowchart for the GET_HLG_FILENAME and GET_QLG_FILENAME subroutines is presented in Figure A.5.

The subroutine begins by passing the desired date and time of the data, declaring all subroutine variables, and initializing the program DATA blocks. Next, the date and time are assigned to the STIME variables, which are used throughout the subroutine for building the desired file name. There is a special case when the hour equals 0. Data for this hour are kept in the previous day's file. Therefore, one day needs to be subtracted from the date. If the hour is 0 and the date is January 1, the subroutine adjusts the date to December 31. If this condition is not met, the subroutine moves to the ELSE portion of the outer IF-THEN-ELSE statement. Another IF-THEN-ELSE statement is encountered, which begins by checking for the leap-year condition. If the leap-year condition is true, the program enters another IF-THEN-ELSE construct, which begins by checking a beginning-of-month condition and hour 0. If this condition is true, the date is set to the end of the previous month. If not, one day is subtracted from the date. If the year is not a leap year, the same procedure is followed but without the leap-year adjustment. After dropping out of the main IF-THEN-ELSE construct, the program assigns the adjusted year and day to the variables CURRENT_YEAR and CURRENT_DAY. The CURRENT_DAY variable is then evaluated to determine if the first character is blank; if so the SI

FIGURE A.5. Flowchart of the GET_HLG_FILENAME and GET_QLG_FILENAME Programs

variable is set to 2. If not, the SI variable is set to 1. The SI variable represents the number of digits in the day of the month. For example, the SI variable is set to 1 for day 9 of a month and set to 2 for day 10 of a month. Next, the GET_MONTH subroutine is initiated to obtain the three-character abbreviation of the desired month. The file name is assigned to the variable that passes the file name back to the ARCHIVE program. For the GET_HLG_FILENAME subroutine, this variable is HLGFILE and for the GET_QLG_FILENAME subroutine the variable is QLGFILE. The name of the subdirectory that precedes the file name is [ERDSSYS.DAT.SENSORS] for the GET_HLG_FILENAME subroutine and [ERDSSYS.DAT.MESORAD.INPUT] for the GET_QLG_FILENAME subroutine. Finally, the file name is passed back and control is returned to the ARCHIVE program.

SUBROUTINE GET MONTH

The GET_MONTH subroutine's function is to convert an integer month to a three character month. A flowchart of the GET_MONTH subroutine is presented in Figure A.6. The subroutine receives the integer month from the ARCHIVE program and begins by declaring all subroutine variables and initializing a DATA block. Next, the subroutine assigns the three-character abbreviation of

FIGURE A.6. Flowchart of the GET_MONTH Program

the month (e.g., JAN for 1) to the SMONTH variable, which is passed back to the ARCHIVE program. The subroutine then terminates and passes control back to the ARCHIVE program.

SUBROUTINE WRITE DATA

The WRITE_DATA subroutine's function is to output the desired data collected by the GET_EMS_DATA subroutine to the appropriate output file. A flowchart of the WRITE_DATA subroutine is presented in Figure A.7. The WRITE_DATA subroutine consists mainly of a five-tiered IF-THEN-ELSE-IF construct. The subroutine will choose only one level to enter. This construct focuses on the variable FLAG, which is an integer value passed to the WRITE_DATA subroutine from ARCHIVE. The variable FLAG indicates the output files to which the selected data should be written. The possible values for FLAG are:

- 1 corresponds to TOWER_200_100NMMM.DAT
- 2 corresponds to TOWER_200_300AMMM.DAT
- 3 corresponds to TOWER_200_400AMMM.DAT
- 4 corresponds to TOWER_400MMM.DAT
- 5 corresponds to TELEMETRY_WINDSMMM.DAT
- 6 corresponds to TELEMETRY_TEMPSMMM.DAT
- 7 corresponds to TELEMETRY_PRECIPMMM.DAT.

The MMM in the above file names indicates the three-character abbreviation of the current month.

The WRITE_DATA subroutine begins by receiving the FLAG variable, the desired date and time, an array containing the desired data, and the number of variables desired from the ARCHIVE program. Next, all the subroutine variables are initialized. Then the subroutine enters into the main IF-THEN-ELSE-IF construct. At the first level a check for the 200-ft tower from the 100N, 300A, and 400A is made. If this condition is met, the subroutine begins processing the data by making adjustments to any missing data in order to fit the designated format statement for each variable. After adjusting the missing data, the program performs a gross error check on the maximum and minimum temperature data. Following the gross error check, the subroutine sets wind directions to 360.0 if the current value is less than 5.0 degrees and sets wind speeds to 0.0 if the current value is less than 0.5 mph. Finally, the

FIGURE A.7. Flowchart of the WRITE_DATA Program

data are written to the appropriate monthly output file: TOWER_200_100NMMM.DAT, TOWER_200_300AMMM.DAT, or TOWER_200_400AMMM.DAT.

If the first-level condition fails the subroutine checks the second level. If the second-level condition is met, the subroutine begins the process of outputting the 410-ft tower from the HMS data. The process performs exactly the same procedures as in the first level with the addition of writing the data to an additional output file called TEMP.OUT (the primary output file is TOWER_400MMM.DAT). This information is sent to the HMS line printer and used by the on-duty forecaster for summarizing hourly 410-ft tower data.

If the second-level condition fails the subroutine checks the third level. If the third-level condition is met, the subroutine begins the process of outputting the telemetry winds, atmospheric stability, and mixing depth. The process begins by adjusting any missing values to fit the designated format statements for each variable. Next, the wind directions are set to 360.0 if the current value is less than 5.0 degrees and the wind speed is set to 0.0 if the current value is less than 0.5 mph. Following this, the subroutine obtains the atmospheric mixing depth from the HOURLY_MIX_PKGST.DAT file located in the [FCST.DAT] subdirectory. Next, the delta-temperature is computed from the difference of the 200-ft and 300-ft temperature from the 410-ft tower. Once the delta-temperature is computed, the subroutine GET_STABILITY is called to obtain an integer value of the atmospheric stability (e.g., very unstable = 1). Next, the subroutine GET_JULIAN is called to compute the Julian day. Finally, the data are written to the monthly TELEMETRY_WINOSMMM.OAT output file.

If the third-level condition fails, the subroutine checks the fourth level. If the fourth-level condition is met, the subroutine begins the process of outputting the telemetry temperatures. The process begins by calling the GET_JULIAN subroutine, which computes the Julian day given a Gregorian date. Next, the data are written to the monthly TELEMETRY_TEMPSMMM.DAT file.

If the fourth-level condition fails the subroutine checks the fifth level. If the fifth-level condition is met, the subroutine begins the process of outputting the telemetry precipitation information. The process begins by adjusting any missing values to the designated format statement for each variable. Next, the GET_JULIAN subroutine is called to compute the Julian date. Then the data are written to the TELEMETRY_PRECIPMMM.DAT monthly file.

The WRITE_DATA subroutine terminates after performing one of the processes or if none of the conditions were met. The WRITE_DATA subroutine does not pass new information back to the ARCHIVE program.

SUBROUTINE GET_JULIAN

The GET_JULIAN subroutine computes the Julian day given a Gregorian date. A flowchart of the GET_JULIAN subroutine is presented in Figure A.8. The subroutine begins accepting the Gregorian year, month, and day from the ARCHIVE program. Next, all subroutine variables are declared and a DATA block is initialized. The current Gregorian month is then used in conjunction with the DATA block information to compute a Julian day. Next, if the year is a leap year the Julian day is adjusted to reflect the additional day if the Julian day is past February 29. The Julian day is then passed back to the ARCHIVE program and the GET_JULIAN subroutine is terminated.

FIGURE A.8. Flowchart of the GET_JULIAN Program

SUBROUTINE GET STABILITY

The GET_STABILITY subroutine computes an integer and character atmospheric stability based on the delta-temperature from the 200-ft to the 30-ft temperature from the HMS's 410-ft tower. A flowchart of the GET_STABILITY subroutine is presented in Figure A.9. The method used for computing the atmospheric stability is based on a method currently used by the Nuclear Regulatory Commission (Snell 1982). The ARCHIVE program passes a delta-temperature value to the GET_STABILITY subroutine. The GET_STABILITY program then declares all subroutine variables. Next, missing values are assigned to the integer and character values of the atmospheric stability. Following this action, a series of IF-THEN-ELSE constructs are used to determine if the delta-temperature is in a particular temperature range. If a condition is met, an integer and a character value are assigned to the atmospheric stability. At this point, the subroutine passes the integer and character values back to the ARCHIVE program and the GET_STABILITY subroutine is terminated.

This concludes the discussion on the ARCHIVE program and its related subroutines. For further information and detail review the FORTRAN source code listings at the end of this appendix.

REFERENCES

- Andrews, G. L. 1988. The Hanford Meteorological Data Collection System and Data Base. PNL-6509, Pacific Northwest Laboratory, Richland, Washington.
- Snell, W. 1982. Nuclear Regulatory Commission Staff Computer Programs for Use with Meteorological Data. NUREG-0917, U.S. Nuclear Regulatory Commission, Washington, D.C.

FIGURE A.9. Flowchart of the GET_STABILITY Program

```

0001      PROGRAM archive
0002 C *****
0003 C
0004 C PROGRAM      - ARCHIVE.FOR
0005 C
0006 C PROGRAMMER   - GREGG ANDREWS  378-8184
0007 C
0008 C PURPOSE      - TO ARCHIVE SELECTED DATA FORM THE HOURLY HMS
0009 C                METEOROLOGICAL DATA BASE ONTO USER DISK PACKS MARKED
0010 C                AS HMS_DATA.  THESE DISKS ARE STORED IN THE FORECAST
0011 C                ROOM.  THE ARCHIVE PROGRAM IS EXECUTED IN SYS$BATCH
0012 C                AT 00:40 EVERY MORNING BY THE ARCHIVE.COM COMMAND
0013 C                FILE.
0014 C
0015 C DATA ARCHIVED - THE PROGRAM AUTOMATICALLY ARCHIVES THE FOLLOWING
0016 C                HOURLY HMS METEOROLOGICAL DATA:
0017 C
0018 C                400-FT TOWER DATA FROM HMS
0019 C                200-FT TOWER DATA FROM 100N
0020 C                200-FT TOWER DATA FROM 300A
0021 C                200-FT TOWER DATA FROM 400A
0022 C                30-FT TELEMETRY WIND INFORMATION
0023 C                30-FT TELEMETRY TEMPERATURE INFORMATION
0024 C                30-FT TELEMETRY PRECIPITATION INFORMATION.
0025 C
0026 C SUBROUTINES CALLED - IDATE (SYSTEM ROUTINE)
0027 C                      SUBTRACT_A_DAY
0028 C                      GET_HMS_DATA
0029 C                      WRITE_DATA
0030 C
0031 C VERSION      - 1.0  7/14/88 GLA
0032 C              1.1  8/88  GLA  ADDED CODE TO CHANGE HOUR FORMAT FROM
0033 C                      0-23 TO 1-24.
0034 C
0035 C *****
0036 C
0037      IMPLICIT NONE
0038 C
0039      INTEGER*4      number_of_files
0040      INTEGER*4      id(100)
0041      INTEGER*2      year
0042      INTEGER*2      month
0043      INTEGER*2      day
0044      INTEGER*2      yy
0045      INTEGER*2      mm
0046      INTEGER*2      dd
0047      INTEGER*2      ddate(3)
0048      INTEGER*2      ttime(2)
0049      INTEGER*2      data_period
0050      INTEGER*2      i
0051      INTEGER*2      n
0052      INTEGER*2      j

```

```

0053     INTEGER*2      count
0054     INTEGER*2      ier
0055     INTEGER*2      days_in_month_ly(12)
0056     INTEGER*2      days_in_month(12)
0057
0058     REAL*4          archive_data(300)
0059
0060     CHARACTER*3      char_month(12)
0061     CHARACTER*1      dummy
0062     CHARACTER*24      file_name(7)
0063     CHARACTER*18      output_file(7)
0064
0065
0066     DATA days_in_month_ly/31, 29, 31, 30, 31, 30,
0067           .                31, 31, 30, 31, 30, 31/
0068
0069     DATA days_in_month/31, 28, 31, 30, 31, 30,
0070           .                31, 31, 30, 31, 30, 31/
0071
0072     DATA char_month/'JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN',
0073           .           'JUL', 'AUG', 'SEP', 'OCT', 'NOV', 'DEC'/
0074
0075     DATA file_name/'TOWER_200_100N_IDS.DAT',
0076           .         'TOWER_200_300A_IDS.DAT',
0077           .         'TOWER_200_400A_IDS.DAT',
0078           .         'TOWER_400_IDS.DAT',
0079           .         'TELEMETRY_WINDS_IDS.DAT',
0080           .         'TELEMETRY_TEMPS_IDS.DAT',
0081           .         'TELEMETRY_PRECIP_IDS.DAT'/
0082
0083     DATA output_file/'TOWER_200_100N',
0084           .           'TOWER_200_300A',
0085           .           'TOWER_200_400A',
0086           .           'TOWER_400',
0087           .           'TELEMETRY_WINDS',
0088           .           'TELEMETRY_TEMPS',
0089           .           'TELEMETRY_PRECIP'/
0090
0091
0092     PARAMETER(number_of_files = 7)
0093
0094
0095     OPEN(UNIT=9, FILE='[fcst.archive]temp.out', STATUS='new',
0096           .         SHARED, CARRIAGECONTROL='LIST')
0097
0098     C
0099     C..... BEGIN OUTER LOOP WHICH CONTROLS TYPE OF DATA TO BE ARCHIVED
0100     C
0101
0102     DO i=1, number_of_files
0103
0104     C

```

```

0105 C..... GET INTERNAL IDS OF SELECTED DATA
0106 C
0107
0108 OPEN(UNIT=2, FILE=file_name(i), STATUS='OLD', READONLY,
0109      SHARED)
0110 READ(2,'(A)') dummy ! skip information line
0111 count = 1
0112 100 READ(2,'(i3)', END=99, ERR=999) id(count)
0113 count = count + 1
0114 GOTO 100
0115 99 CLOSE (UNIT=2)
0116 count = count - 1
0117 CALL idate(month, day, year)
0118 CALL subtract_a_day(year, month, day)
0119
0120 C
0121 C..... OPEN APPROPRIATE OUTPUT FILE; IF FILE IS NOT FOUND ASSUME A NEW
0122 C..... FILE NEEDS TO BE OPENED
0123 C
0124
0125 OPEN(UNIT=3, FILE='[FCST.DAT]//output_file(i)
0126      //char_month(month)//'.DAT',
0127      STATUS='OLD', IOSTAT=ier, SHARED, ACCESS='APPEND',
0128      CARRIAGECONTROL='LIST')
0129
0130 IF(ier .NE. 0) THEN
0131 IF(ier .EQ. 29) THEN ! write to new monthly file
0132 CLOSE (UNIT=3)
0133 OPEN(UNIT=3,
0134      FILE='[FCST.DAT]//output_file(i)
0135      //char_month(month)//'.DAT',
0136      STATUS='NEW', SHARED,
0137      CARRIAGECONTROL='LIST')
0138 CLOSE (UNIT=3)
0139 OPEN(UNIT=3, FILE='[FCST.DAT]//output_file(i)//
0140      char_month(month)//'.DAT',
0141      STATUS='OLD', IOSTAT=ier, SHARED, ACCESS='APPEND',
0142      CARRIAGECONTROL='LIST')
0143 ELSE
0144      ! some other problem with file
0145 TYPE*, ' '
0146 TYPE*, ' '
0147 TYPE *, ' ERROR ', ier, ' OPENING FILE ', output_file(i)
0148 ENDIF
0149
0150 C
0151 C..... OPEN FILE CONTAINING PEAK QUST AND MIX DEPTH INFORMATION
0152 C
0153
0154 OPEN(UNIT=12, FILE='[FCST.DAT]HOURLY_MIX_PKGST.DAT',
0155      STATUS='OLD', READONLY, SHARED)
0156

```

```

0157 C
0158 C..... BEGIN INNER LOOP WHICH CONTROLS WHICH HOUR OF DATA TO BE ARCHIVED
0159 C
0160
0161 DO j=1, 24
0162 C
0163 C..... GET HOURLY DATA FROM EMS DATA BASE
0164 C
0165
0166 ddate(1) = year
0167 ddate(2) = month
0168 ddate(3) = day
0169 ttime(1) = j ! hour of day
0170 ttime(2) = 0 ! minute; always set to 0 for hourly data
0171 data_period = 1 ! hourly information
0172
0173 C..... FIRST ADJUST J = 0 WHEN J = 24; EMS DATA BASE IS IN 0-23 HOUR FORMAT
0174
0175 IF(j .EQ. 24) THEN
0176
0177 IF(MOD(ddate(1),4) .EQ. 0) THEN
0178 DO n=1, 12
0179 days_in_month(n) = days_in_month_ly(n)
0180 ENDDO
0181 ENDIF
0182
0183 IF((month.EQ.12) .AND. (day.EQ.31)) THEN ! END OF YEAR
0184 ddate(1) = ddate(1) + 1
0185 ddate(2) = 1
0186 ddate(3) = 1
0187 ttime(1) = 0
0188 ELSE IF(days_in_month(ddate(2)) .EQ. ddate(3)) THEN
0189 ddate(2) = ddate(2) + 1
0190 ddate(3) = 1
0191 ttime(1) = 0
0192 ELSE
0193 ddate(3) = ddate(3) + 1
0194 ttime(1) = 0
0195 ENDIF
0196
0197 ENDIF
0198
0199 CALL get_ems_data(ddate, ttime, data_period,
0200 id, count, archive_data)
0201
0202
0203 C
0204 C..... SWITCH BACK TO ORIGINAL DATE AND TIME FOR ARCHIVE FILES
0205 C
0206 IF(j .EQ. 24) THEN
0207 ddate(1) = year
0208 ddate(2) = month

```

```

0209             ddate(3) = day
0210             ttime(1) = j
0211             ENDIF
0212
0213 C
0214 C..... WRITE HOURLY DATA TO MONTHLY OUTPUT FILE
0215 C
0216
0217             CALL write_data(i, ddate, ttime, archive_data,
0218                             count)
0219
0220             ENDDO ! HOUR LOOP
0221             CLOSE (UNIT=3)
0222             CLOSE (UNIT=12)
0223             ENDDO ! FILE LOOP
0224
0225             CALL EXIT
0226
0227 999 TYPE *, ' '
0228 TYPE *, ' '
0229 TYPE *, ' ERROR READING FROM ',file_name(i),' IN PROGRAM ARCHIVE'
0230 CALL EXIT
0231
0232 END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	1029	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	130	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	2352	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	3511	

ENTRY POINTS

Address	Type	Name	References
0-00000000		ARCHIVE	1#

VARIABLES

Address	Type	Name	Attributes	References
2-000007C2	I*2	COUNT		53 111= 112 113(2)= 116(2)= 199A 217A
2-000007BE	I*2	DATA_PERIOD		49 171= 199A

2-000007BC	I*2	DAY	43	117A	118A	188	183	209
**	I*2	DD	46					
2-000007B8	CHAR	DUMMY	61	110=				
2-000007C0	I*2	I	50	102=	108	125	133	139
			146	217A	229			
2-000007C4	I*2	IER	54	125=A	130	131	139=A	148
**	I*2	J	52	161=	169	175	206	210
**	I*2	MM	45					
2-000007BA	I*2	MONTH	42	117A	118A	125	133	139
			167	183	208			
**	I*2	N	51	178=	179(2)			
2-000007B8	I*2	YEAR	41	117A	118A	166	207	
**	I*2	YY	44					

ARRAYS

Address	Type	Name	Attributes	Bytes	Dimensions	References
2-00000190	R*4	ARCHIVE_DATA		1280	(300)	58 199A 217A
2-0000067A	CHAR	CHAR_MONTH		36	(12)	60 72D 125
				133	139	
2-00000662	I*2	DAYS_IN_MONTH		24	(12)	56 89D 179=
				188		
2-0000064A	I*2	DAYS_IN_MONTH_LY		24	(12)	55 88D 179
2-00000640	I*2	DDATE		6	(3)	47 166= 167=
				188=	177	184(2)= 185= 186=
				188(2)	189(2)=	190= 193(2)= 199A
				207=	208=	209= 217A
2-0000069E	CHAR	FILE_NAME		168	(7)	62 75D 108A
				229		
2-00000000	I*4	ID		400	(100)	40 112= 199A
2-00000748	CHAR	OUTPUT_FILE		112	(7)	63 83D 125
				133	139	140
2-00000646	I*2	TTIME		4	(2)	48 169= 170=
				187=	191=	194= 199A 210=
				217A		

PARAMETER CONSTANTS

Type	Name	References
I*4	NUMBER_OF_FILES	39 92# 102

LABELS

Address	Label	References	
0-00000098	99	112	115‡
0-0000005C	100	112‡	114
0-0000037C	999	112	227‡

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	References					
	FOR\$CLOSE	115	132	138	221	222	
	FOR\$EXIT	225	230				
	FOR\$IDATE	117					
	FOR\$OPEN	95	100	125	133	139	154
	GET_EMS_DATA	199					
	SUBTRACT_A_DAY	110					
	WRITE_DATA	217					

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 ARCHIVE

/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:      6.20 seconds
Elapsed Time:   8.35 seconds
Page Faults:    379
Dynamic Memory: 408 pages

```

```

0001 C *****
0002 C
0003 C PROGRAM      - SUBTRACT_A_DAY.FOR
0004 C
0005 C PROGRAMMER   - GREGG ANDREWS
0006 C
0007 C PURPOSE     - TO SUBTRACT ONE DAY FROM THE PASSED DAY.
0008 C
0009 C PASSED PARAMETERS - yyear  - CURRENT YEAR; MAYBE MODIFIED
0010 C                  mmonth  - CURRENT MONTH; MAYBE MODIFIED
0011 C                  dday    - CURRENT DAY; MAYBE MODIFIED
0012 C
0013 C VERSION      - 1.0 7/15/88
0014 C
0015 C *****
0016
0017 SUBROUTINE subtract_a_day(yyear, mmonth, dday)
0018
0019 INTEGER*2 yyear
0020 INTEGER*2 mmonth
0021 INTEGER*2 dday
0022 INTEGER*4 days_in_month_ly(12)
0023 INTEGER*4 days_in_month(12)
0024
0025 DATA days_in_month_ly/31, 29, 31, 30, 31, 30,
0026 .                    31, 31, 30, 31, 30, 31/
0027
0028 DATA days_in_month/31, 28, 31, 30, 31, 30,
0029 .                    31, 31, 30, 31, 30, 31/
0030
0031
0032 C
0033 C..... SUBTRACT ONE DAY FROM PASSED DAY
0034 C
0035
0036 IF((mmonth.EQ.1) .AND. (dday.EQ.1)) THEN ! check for begin of year
0037
0038     yyear = yyear - 1
0039     mmonth = 12
0040     dday = 31
0041
0042 ELSE IF((mmonth.NE.1) .AND. (dday.EQ.1)) THEN ! check for begin of month
0043
0044     mmonth = mmonth - 1
0045     IF(MOD(yyear,4) .EQ. 0) THEN ! check for leap year
0046         dday = days_in_month_ly(mmonth)
0047     ELSE
0048         dday = days_in_month(mmonth)
0049     ENDF
0050
0051 ELSE
0052

```

```

0053      dday = dday - 1
0054
0055      ENDIF
0056
0057      RETURN
0058      END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	116	PIC CON REL LCL SHR EXE RD NOWRT LONG
2 \$LOCAL	96	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	212	

ENTRY POINTS

Address	Type	Name	References
0-00000000		SUBTRACT_A_DAY	17#

VARIABLES

Address	Type	Name	Attributes	References
AP-000000C0	I*2	DDAY		17 21 36 40= 42 46=
AP-000000B0	I*2	MMONTH		48= 53(2)= 17 20 36 39= 42 44(2)=
AP-00000040	I*2	YYEAR		46 48 17 19 36(2)= 45

ARRAYS

Address	Type	Name	Attributes	Bytes	Dimensions	References
2-00000030	I*4	DAYS_IN_MONTH		48	(12)	23 28D 48
2-00000000	I*4	DAYS_IN_MONTH_LY		48	(12)	22 25D 46

```

+-----+
|      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 SUBTRACT_A_DAY

/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: 1.81 seconds

Elapsed Time: 2.85 seconds

Page Faults: 499

Dynamic Memory: 331 pages

```

0001 C *****
0002 C
0003 C     PROGRAM - GET_EMS_DATA.FOR
0004 C
0005 C     PROGRAMMER - GREGG ANDREWS 376-0138
0006 C
0007 C     PURPOSE - TO OBTAIN EMS DATA FROM EITHER THE FIFTEEN MINUTE OR HOURLY
0008 C               DATA BASE. THE DESIRED DATA IS SPECIFIED BY THE INTERNAL ID.
0009 C
0010 C ALGORITHM - DECLARE VARIABLES
0011 C     EQUATE HOUR AND MIN TO STIME ARRAY
0012 C     CALL SUBROUTINES TO GET DESIRED EMS DATA FILENAME
0013 C     INITIALIZE RECORD AND BLOCK COUNTER
0014 C     READ FIRST RECORD OF EMS DATA FILE
0015 C     WHILE PASSED HOUR AND MIN NOT EQUAL TO EMS HOUR AND MIN
0016 C         CONTINUE SEARCH FOR DESIRED HOUR AND MIN
0017 C     END WHILE LOOP
0018 C     READ IN DATA FROM DESIRED HOUR AND MIN
0019 C     SEARCH THROUGH PASSED PARAMETER IDS AND COMPARED WITH
0020 C         INTERNAL ID
0021 C     IF IDS MATCH EQUATE MEAN VALUE ARRAY TO DESIRED DATA
0022 C         ARRAY
0023 C     CONTINUE SEARCH THROUGH TWO BLOCKS
0024 C
0025 C PASSED PARAMETERS - DATE is an array of 3 elements defined as
0026 C                     follows - DATE(1) --> YEAR
0027 C                     DATE(2) --> MONTH
0028 C                     DATE(3) --> DAY
0029 C     TIME is an array of 2 elements defined as
0030 C                     follows - TIME(1) --> HOUR
0031 C                     TIME(2) --> MINUTE
0032 C     DATA_INTERVAL_FLAG 1 = hourly data
0033 C                     0 = fifteen minute data
0034 C     PARAM_ID array containing EMS internal ids of
0035 C                     desired data
0036 C     NUMBER_PARAMETERS total number of EMS internal
0037 C                     ids desired
0038 C     DESIRED_DATA array containing the desired data
0039 C
0040 C     THE SUBROUTINE ASSUMES THE PASSED PARAMETERS ARE CORRECT AND ARE VALID
0041 C     VALUES. DO ERROR CHECKING IN THE MAIN PROGRAM.
0042 C
0043 C VERSION - 1.0
0044 C
0045 C DATE - NOVEMBER 1987
0046 C
0047 C *****
0048 C
0049 C SUBROUTINE get_ems_data(date, time, data_interval_flag,
0050 C                         param_id, number_parameters,
0051 C                         desired_data)
0052 C

```

```

0053      IMPLICIT NONE
0054
0055      C
0056      C..... DATA TYPE          VARIABLE          FUNCTION
0057      C
0058
0059      INTEGER*2    block_counter    ! keeps track of number of datablocks; max is 2
0060      INTEGER*2    date(3)          ! date of desired data
0061      INTEGER*2    time(2)          ! time of desired data
0062      INTEGER*4    record            ! specifies the record to read from
0063      INTEGER*4    param_id(300)    ! internal id stored in external file
0064      INTEGER*2    year              ! used to compare with year_ems
0065      INTEGER*2    month             ! value of month from user
0066      INTEGER*2    day               ! value of day from user
0067      INTEGER*2    hour              ! value of hour from user
0068      INTEGER*2    min               ! value of minute from user if quarterly file required
0069      INTEGER*2    year_ems          ! value of year from ems file
0070      INTEGER*2    month_ems         ! value of month from ems file
0071      INTEGER*2    day_ems           ! value of day from ems file
0072      INTEGER*2    hour_ems          ! value of hour from ems file
0073      INTEGER*2    min_ems           ! value of minute from ems file
0074      INTEGER*2    number_information_records ! number of information records in hourly block of ems file
0075      INTEGER*2    number_parameters ! total number of parameters desired
0076      INTEGER*4    internal_id(300)  ! internal id for station parametertere
0077      INTEGER*2    ier               ! error value following read sttment
0078      INTEGER*2    iier              ! error value following read sttment
0079      INTEGER*2    number_data_records ! number of data records in the hourly block from the ems file
0080      INTEGER*2    i,j,k             ! indexes for do loops
0081      INTEGER*2    data_interval_flag ! indicates whether user wants hourly or quarterly data
                                1 = hourly  quarterly = 0
0082      INTEGER*2    missing_data      ! counter used to determine amount of missing data
0083
0084      REAL*4       number_data_points(300) ! number of data points used to calculate the mean
0085      REAL*4       mean_value(300)        ! mean value of station parameter
0086      REAL*4       sigma(300)             ! standard deviation of a parameter; not used
0087      REAL*4       desired_data(300)      ! array that contains the desired data
0088
0089      CHARACTER*10  param_name(300)       ! parameter id also includes location of parameter
0090      CHARACTER*(30) filename             ! EMS filename built from date and time from user
0091
0092
0093
0094      C
0095      C..... PUT TIME ELEMENTS INTO HOUR AND MIN VARIABLES
0096      C
0097
0098      hour = time(1)
0099      min  = time(2)
0100
0101      C
0102      C..... INITIALIZE DATA ARRAYS
0103      C

```

```

0104
0105      DO i=1, 300
0106          mean_value(i)      = 0.0
0107          sigma(i)           = 0.0
0108          number_data_points(i) = 0.0
0109          desired_data(i)     = 0.0
0110      ENDDO
0111
0112      C
0113      C..... OPEN EMS DATA FILE AS SPECIFIED BY THE DATE
0114      C
0115
0116
0117      IF(data_interval_flag .eq. 1)
0118          CALL get_hlg_filename(filename,date,time)    ! HOURLY FILE
0119      IF(data_interval_flag .eq. 0)
0120          CALL get_qtg_filename(filename,date,time)    ! FIFTEEN MINUTE FILE
0121
0122
0123      OPEN(UNIT=1, FILE= filename, STATUS='OLD', ACCESS='DIRECT',
0124          ORGANIZATION='RELATIVE', READONLY, SHARED, IOSTAT=ier)
0125
0126      IF(ier .ne. 0) THEN
0127          WRITE(*,20) ier
0128      20      FORMAT(' Error During Open ',
0129          'EMS Data File Code - ',I4)
0130      ENDIF
0131
0132
0133      C
0134      C..... READ FIRST RECORD OF EMS DATA FILE AND INITIALIZE RECORD AND
0135      C..... BLOCK_COUNTER VARIABLES
0136      C
0137
0138      record = 1
0139      block_counter = 0
0140
0141      400  READ(1, REC=record, IOSTAT=ier,ERR=1000) year_ems,
0142          .                               month_ems,day_ems,
0143          .                               hour_ems, min_ems,
0144          .                               number_data_records
0145
0146
0147      C
0148      C..... CONTINUE TO READ HOUR AND MIN FROM EMS FILE UNTIL DESIRED HOUR AND
0149      C..... MIN IS FOUND
0150      C
0151
0152
0153      iier = 0
0154      DO WHILE ( (hour_ems .ne. hour) .or.
0155          (min_ems .ne. min) )

```



```

0156
0157      number_information_records =                ! ONE INFORMATION RECORD PER 31 DATA RECORD
0158      .      jnint(number_data_records / 31. + .49999)
0159      record = record + 1 + number_information_records + ! FOLLOWING CODE DETERMINES THE NUMBER RECORDS TO SKIP
0160      .      number_data_records
0161      READ(1,REC=record,IOSTAT=iier,ERR=1000) year_ems,
0162      .      month_ems,                ! READ NEXT DATE AND HOUR AND NUMBER
                                           OF RECORDS
0163      .      day_ems, hour_ems, ! TO DETERMINE IF DESIRED DATE FROM
                                           SCRATCH FILE
0164      .      min_ems,
0165      .      number_data_records
0166      ENDDO
0167
0168
0169      C
0170      C..... SHOULD BE AT DESIRED HOUR AND MIN DATA BLOCK NOW MUST DETERMINE
0171      C..... NUMBER OF INFORMATION RECORDS TO SKIP
0172      C
0173
0174      number_information_records = jnint(number_data_records / 31.
0175      .      + .49999)
0176
0177
0178      C
0179      C..... FOLLOWING CODE SHOULD PUT AT BEGINNING OF DATA RECORD BLOCK OF DESIRED
0180      C..... HOUR AND MIN
0181      C
0182
0183      record = record + number_information_records
0184
0185
0186      C
0187      C..... NOW READ ALL DATA FOR THE DESIRED HOUR AND MIN
0188      C
0189
0190      DO i=1, number_data_records
0191      record = record + 1
0192      READ(1,REC=record,IOSTAT=iier,ERR=1000)
0193      .      internal_id(i),
0194      .      param_name(i),
0195      .      mean_value(i),
0196      .      sigma(i),
0197      .      number_data_points(i)
0198
0199      C
0200      C..... CHECK FOR MISSING DATA AND REPLACE WITH 999.9
0201      C
0202
0203      IF((mean_value(i) .EQ. 0.0) .AND.
0204      .      (sigma(i) .EQ. 0.0) .AND.
0205      .      (number_data_points(i) .EQ. 0.0)) mean_value(i) = 999.9

```

```

0206      IF(number_data_points(i) .LE. 0.0) mean_value(i) = 999.9
0207      ENDDO
0208
0209
0210
0211      C
0212      C..... SEARCH THROUGH ARRAY OF INTERNAL IDS AND CHECK TO SEE IF IT IS A DESIRED
0213      C..... DATA, IF SO STORE IN DESIRED_DATA ARRAY
0214      C
0215
0216      k = 0
0217
0218      DO j=1, number_parameters
0219          k = k + 1
0220      DO i=1, number_data_records
0221          IF(param_id(j) .eq. internal_id(i)) THEN
0222              desired_data(k) = mean_value(i)
0223          ENDIF
0224      ENDDO
0225      ENDDO
0226
0227      record = record + 1
0228      block_counter = block_counter + 1
0229
0230      IF(block_counter .EQ. 2) GOTO 1000
0231
0232      GOTO 400      ! loop will terminate when RECORD is greater than the number of records in the file
0233                  ! this is necessary since the number of same time stamps can be greater than one
0234
0235      1000 IF(iier .NE. 0) THEN
0236          DO i=1, number_parameters
0237              desired_data(i) = 999.9
0238          ENDDO
0239      ENDIF
0240      CLOSE (UNIT=1)
0241
0242      RETURN
0243      END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	1046	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	48	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	9832	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	10926	

ENTRY POINTS

Address	Type	Name	References
0-00000000		GET_EMS_DATA	49#

VARIABLES

Address	Type	Name	Attributes	References
**	I*2	BLOCK_COUNTER	59	139= 228(2)= 230
AP-000000C0	I*2	DATA_INTERVAL_FLAG	49	81 117 119
**	I*2	DAY	68	
2-000025AC	I*2	DAY_EMS	71	141= 161=
2-00002580	CHAR	FILENAME	90	117A 119A 123A
**	I*2	HOURL	87	98= 154
2-000025AE	I*2	HOURL_EMS	72	141= 154 161=
**	I*2	I	80	105= 106 107 108 109
			190=	192(5) 203(4) 206(2) 220= 221
			222	236= 237
2-000025B2	I*2	IER	77	123=A 126 127 141= 192=
2-000025B4	I*2	IIER	78	153= 161= 235
**	I*2	J	80	216= 221
**	I*2	K	80	216= 219(2)= 222
2-000025A6	I*2	MIN	68	99= 154
2-000025B0	I*2	MIN_EMS	73	141= 154 161=
**	I*2	MISSING_DATA	82	
**	I*2	MONTH	65	
2-000025AA	I*2	MONTH_EMS	70	141= 161=
2-000025B6	I*2	NUMBER_DATA_RECORDS	79	141= 157 159 181= 174
			190	220
**	I*2	NUMBER_INFORMATION_RECORDS	74	157= 159 174= 183
AP-000000140	I*2	NUMBER_PARAMETERS	49	75 218 236
**	I*4	RECORD	82	138= 141 159(2)= 161 183(2)=
			191(2)=	192 227(2)=
**	I*2	YEAR	64	
2-000025A8	I*2	YEAR_EMS	89	141= 181=

ARRAYS

Address	Type	Name	Attributes	Bytes	Dimensions	References
AP-000000040	I*2	DATE		8	(3)	49 60 117A
				119A		
AP-000000180	R*4	DESIRED_DATA		1200	(300)	49 87 109=
				222=	237=	
2-00000000	I*4	INTERNAL_ID		1200	(300)	78 192= 221
2-00000960	R*4	MEAN_VALUE		1200	(300)	85 106= 192=
				203(2)=	206=	222

2-000004B0	R*4	NUMBER_DATA_POINTS	1200 (300)	84	108=	192=
			203	208		
AP-000000100	I*4	PARAM_ID	1200 (300)	49	63	221
2-000012C0	CHAR	PARAM_NAME	4800 (300)	89	192=	
2-00000E10	R*4	SIGMA	1200 (300)	86	107=	192=
			203			
AP-000000000	I*2	TIME	4 (2)	49	61	98
			99	117A	119A	

LABELS

Address	Label	References
1-00000000	20'	127 128#
0-00000100	400	141# 232
0-000003D0	1000	141 161 192 230 235#

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	References
	FOR\$CLOSE	240
	FOR\$OPEN	123
	GET_HLG_FILENAME	117
	GET_QLG_FILENAME	119

```

+-----+
|          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 GET_SMS_DATA

/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:	5.98 seconds
Elapsed Time:	7.72 seconds
Page Faults:	645
Dynamic Memory:	418 pages

```

0001 C *****
0002 C
0003 C      SUBROUTINE - GET_HLG_FILENAME
0004 C
0005 C PROGRAMMER - GREGG ANDREWS
0006 C
0007 C      PURPOSE - TO RETURN THE FILE NAME OF THE CURRENT ERDS HOURLY LOG FILE.
0008 C      USES THE SYSTEM TIME TO BUILD THE NAME. PROGRAM WAS ORIGINALLY
0009 C      WRITTEN BY KAREN S. MODIFIED TO FIT NEEDS OF THE GET_EMS_DATA
0010 C      SUBROUTINE
0011 C
0012 C
0013 C DEFINITION OF STIME - STIME(1) --> YEAR
0014 C      STIME(2) --> MONTH
0015 C      STIME(3) --> DAY
0016 C      STIME(4) --> HOUR
0017 C
0018 C ALGORITHM - EQUATE DATE AND TIME TO STIME ARRAY
0019 C      CHECK FOR SPECIAL CASE OF HOUR 0 AND JAN. 1
0020 C      CHECK FOR LEAP YEAR
0021 C      CHECK FOR BEGIN OF MONTH
0022 C      ELSE
0023 C      CHECK FOR BEGIN OF MONTH
0024 C      ENDIF
0025 C      WRITE MODIFIED DAY AND UNMODIFIED YEAR TO CURRENT VARIABLES
0026 C      BUILD FILENAME
0027 C
0028 C IT IS NECESSARY TO MODIFY THE DAY WHEN HOUR EQUAL 0 BECAUSE DATA FOR
0029 C THE 0 HOUR IS KEPT IN THE PREVIOUS'S DAY FILE; DON'T ASK !!!!.
0030 C
0031 C      PASSED PARAMETERS - EMS FILENAME this filename built in this subroutine
0032 C      DATE date passed from GET_EMS_DATA (E.G., 871231)
0033 C      TIME time passed from GET_EMS_DATA (E.G., 2315)
0034 C
0035 C VERSION - 1.0
0036 C
0037 C DATE - NOVEMBER 1987
0038 C
0039 C *****
0040 C
0041 C      SUBROUTINE get_hlg_filename(hlgfile,ddate,tttime)
0042 C
0043 C      IMPLICIT NONE
0044 C
0045 C      INTEGER*2 si, stime(4), ddate(3), tttime(2), i
0046 C      INTEGER      days_in_month_ly(12), days_in_month(12)
0047 C      CHARACTER current_year*16, current_month*3, current_day*2
0048 C      CHARACTER*(38) hlgfile
0049 C
0050 C      DATA days_in_month/31, 28, 31, 30, 31, 30, 31, 31, 30, 31,
0051 C      30, 31/
0052 C

```

```

0053 DATA days_in_month_ly/31, 29, 31, 30, 31, 30, 31, 31, 30,
0054      31, 30,31/
0055
0056
0057 C
0058 C..... PUT INFORMATION FROM DDATE AND TTIME INTO STIME ARRAY
0059 C
0060
0061 stime(1) = ddate(1)
0062 stime(2) = ddate(2)
0063 stime(3) = ddate(3)
0064 stime(4) = ttime(1)
0065
0066 C
0067 C..... CHECK FOR SPECIAL CASE OF JAN. 1 HOUR 0; HOUR 0 IS CONTAINED IN THE
0068 C..... THE PREVIOUS DAY'S FILE
0069 C
0070
0071 *
0072 IF( (stime(2).EQ.1) .AND. (stime(3).EQ.1) .AND.
0073      (stime(4).EQ.0) )THEN
0074
0075     stime(1) = stime(1) - 1
0076     stime(2) = 12
0077     stime(3) = 31
0078
0079 ELSE
0080
0081     IF(MOD(stime(1),4) .EQ. 0) THEN ! CHECK FOR LEAP YEAR
0082         IF( (stime(3).EQ.1) .AND. (stime(4).EQ.0) ) THEN
0083             stime(2) = stime(2) - 1
0084             stime(3) = days_in_month_ly(stime(2))
0085         ELSE
0086             IF(stime(4).EQ.0) stime(3) = stime(3) - 1
0087         ENDIF
0088     ELSE
0089         IF( (stime(3).EQ.1) .AND. (stime(4).EQ.0) ) THEN
0090             stime(2) = stime(2) - 1
0091             stime(3) = days_in_month(stime(2))
0092         ELSE
0093             IF(stime(4).EQ.0) stime(3) = stime(3) - 1
0094         ENDIF
0095     ENDIF
0096 ENDIF
0097
0098 WRITE(current_year,'(i16)') stime(1)
0099 WRITE(current_day, '(i2)' ) stime(3)
0100
0101
0102 C
0103 C..... ELIMINATE ANY SPACES IN DAY (I.E. IF DAY IS <= 9)
0104 C

```

```

0105
0106
0107     IF(current_day(1:1) .eq. ' ' ) THEN
0108         si = 2
0109     ELSE
0110         si = 1
0111     ENDIF
0112
0113     CALL get_month(stime(2),current_month)
0114
0115     hlgfile = '[ERDSSYS.DAT.SENSORS]'//
0116             current_month//current_day(si:2)//
0117             current_year(15:18)//'.HLG'
0118
0119     RETURN
0120     END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	420	PIC COM REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	27	PIC CDN REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	212	PIC COM REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	659	

ENTRY POINTS

Address	Type	Name	References
0-00000000		GET_HLG_FILENAME	41#

VARIABLES

Address	Type	Name	Attributes	References
2-0000007B	CHAR	CURRENT_DAY	47	99 107 115
2-00000078	CHAR	CURRENT_MONTH	47	113A 115
2-00000088	CHAR	CURRENT_YEAR	47	98 115
AP-00000040	CHAR	HLGFILE	41	48 115=
**	I*2	I	45	
**	I*2	SI	45	100= 110= 115

ARRAYS

Address	Type	Name	Attributes	Bytes	Dimensions	References
2-00000030	I*4	DAYS_IN_MONTH		48	(12)	48 50D 91
2-00000000	I*4	DAYS_IN_MONTH_LY		48	(12)	46 53D 84
AP-00000000	I*2	DDATE		8	(3)	41 45 81
				82	63	
2-00000000	I*2	STIME		8	(4)	45 81= 62=
				83=	84=	72(3) 75(2)= 78=
				77=	81	82(2) 83(2)= 84(2)=
				86(3)=	89(2)	90(2)= 91(2)= 93(3)=
				98	99	113A
AP-000000C0	I*2	TTIME		4	(2)	41 45 84

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	References
	GET_MONTH	113

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 GET_HLG_FILENAME

/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)
/DEBUG=(NOSYMBOLS,TRACEBACK)
/STANDARD=(NOSYNTAX,NOSOURCE_FORM)
/SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP,NODICTIONARY,SINGLE)
/WARNINGS=(GENERAL,NODECLARATIONS,NOULTRIX)
/CONTINUATIONS=19 /CROSS_REFERENCE /NOO_LINES /EXTEND_SOURCE /F77
/NOG_FLOATING /I4 /NOMACHINE_CODE /OPTIMIZE

```

COMPILATION STATISTICS

```

Run Time:          3.63 seconds
Elapsed Time:      5.03 seconds
Page Faults:       599
Dynamic Memory:    375 pages

```

```

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

```

```

0053 DATA days_in_month_ly/31, 29, 31, 30, 31, 30, 31, 31, 30,
0054      31, 30,31/
0055
0056
0057 C
0058 C..... PUT INFORMATION FROM DDATE AND TTIME INTO STIME ARRAY
0059 C
0060
0061 stime(1) = ddate(1)
0062 stime(2) = ddate(2)
0063 stime(3) = ddate(3)
0064 stime(4) = ttime(1)
0065
0066 C
0067 C..... CHECK FOR SPECIAL CASE OF JAN. 1 HOUR 0; HOUR 0 IS CONTAINED IN THE
0068 C..... THE PREVIOUS DAY'S FILE
0069 C
0070
0071
0072 IF( (stime(2).EQ.1) .AND. (stime(3).EQ.1) .AND.
0073      (stime(4).EQ.0) )THEN
0074
0075     stime(1) = stime(1) - 1
0076     stime(2) = 12
0077     stime(3) = 31
0078
0079 ELSE
0080
0081     IF(MOD(stime(1),4) .EQ. 0) THEN ! CHECK FOR LEAP YEAR
0082         IF( (stime(3).EQ.1) .AND. (stime(4).EQ.0) ) THEN
0083             stime(2) = stime(2) - 1
0084             stime(3) = days_in_month_ly(stime(2))
0085             ELSE
0086                 IF(stime(4).EQ.0) stime(3) = stime(3) - 1
0087         ENDIF
0088     ELSE
0089         IF( (stime(3).EQ.1) .AND. (stime(4).EQ.0) ) THEN
0090             stime(2) = stime(2) - 1
0091             stime(3) = days_in_month(stime(2))
0092             ELSE
0093                 IF(stime(4).EQ.0) stime(3) = stime(3) - 1
0094         ENDIF
0095     ENDIF
0096 ENDIF
0097
0098 WRITE(current_year,'(i10)') stime(1)
0099 WRITE(current_day, '(i2)' ) stime(3)
0100
0101
0102 C
0103 C..... ELIMINATE ANY SPACES IN DAY (I.E. IF DAY IS <= 9)
0104 C

```

```

0105
0106
0107     IF(current_day(1:1) .eq. ' ' ) THEN
0108         si = 2
0109     ELSE
0110         si = 1
0111     ENDIF
0112
0113     CALL get_month(stime(2),current_month)
0114
0115     qlgfile = '[ERDSSYS.DAT.WESORAD.INPUT] '//
0116             current_month//current_day(si:2)//
0117             current_year(15:16)//'.QLG'
0118
0119
0120     RETURN
0121     END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	420	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	33	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	212	PIC CON REL LCL NUSHR NOEXE RD WRT LONG
Total Space Allocated	665	

ENTRY POINTS

Address	Type	Name	References
0-00000000		GET_QLG_FILENAME	41#

VARIABLES

Address	Type	Name	Attributes	References
2-00000070	CHAR	CURRENT_DAY		47 99 107 115
2-00000078	CHAR	CURRENT_MONTH		47 113A 115
2-00000080	CHAR	CURRENT_YEAR		47 98 115
**	I*2	I		45
AP-00000040	CHAR	QLGFILE		41 48 115=
**	I*2	SI		45 108= 110= 115

ARRAYS

Address	Type	Name	Attributes	Bytes	Dimensions	References
2-00000030	I*4	DAYS_IN_MONTH		48	(12)	46 50D 91
2-00000000	I*4	DAYS_IN_MONTH_LY		48	(12)	46 53D 84
AP-00000080	I*2	DDATE		8	(3)	41 45 61
				82	63	
2-00000000	I*2	STIME		8	(4)	45 61= 82=
				63=	84=	72(3) 75(2)= 76=
				77=	81	82(2) 83(2)= 84(2)=
				86(3)=	89(2)	90(2)= 91(2)= 93(3)=
				98	99	113A
AP-000000C0	I*2	TTIME		4	(2)	41 45 84

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	References
	GET_MONTH	113

	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 GET_QLG_FILENAME

/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)
/DEBUG=(NOSYMBOLS,TRACEBACK)
/STANDARD=(NOSYNTAX,NOSOURCE_FORM)
/SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP,NODICTIONARY,SINGLE)
/WARNINGS=(GENERAL,NODECLARATIONS,NOULTRIX)
/CONTINUATIONS=19 /CROSS_REFERENCE /NOD_LINES /EXTEND_SOURCE /F77
/NOG_FLOATING /I4 /NOMACHINE_CODE /OPTIMIZE

```

COMPILATION STATISTICS

```

Run Time:          3.84 seconds
Elapsed Time:      4.75 seconds
Page Faults:       585
Dynamic Memory:    375 pages

```

```

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

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	39	PIC CON REL LCL SHR EXE RD NOWRT LONG
2 \$LOCAL	44	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	83	

ENTRY POINTS

Address	Type	Name	References
0-00000000		GET_MONTH	22#

VARIABLES

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

ARRAYS

Address	Type	Name	Attributes	Bytes	Dimensions	References
2-00000000	CHAR	MONTHS		36	(12)	26 30D 33

```

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

```

COMMAND QUALIFIERS

```

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

/CHECK=(BOUNDS,OVERFLOW,UNDERFLOW)
/DEBUG=(NOSYMBOLS,TRACEBACK)
/STANDARD=(NOSYNTAX,NOSOURCE_FORM)
/SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP,NODICTIONARY,SINGLE)
/WARNINGS=(GENERAL,NODECLARATIONS,NOULTRIX)
/CONTINUATIONS=10 /CROSS_REFERENCE /NO_LINES /EXTEND_SOURCE /F77
/NOG_FLOATING /I4 /NOMACHINE_CODE /OPTIMIZE

```

COMPILATION STATISTICS

```

Run Time:      1.20 seconds
Elapsed Time:  2.26 seconds
Page Faults:   466
Dynamic Memory: 315 pages

```

```

0001 C *****
0002 C
0003 C PROGRAM - WRITE_DATA
0004 C
0005 C PROGRAMMER - GREGG ANDREWS
0006 C
0007 C PURPOSE - TO WRITE THE ARCHIVE DATA TO THE APPROPRIATE
0008 C             MONTHLY FILE. DATA IS CONVERTED FROM INTEGER
0009 C             TO FLOATING POINT BEFORE IT IS WRITTEN. FLAGS
0010 C             1 THROUGH 3 REPRESENT THE 200-FT TOWER DATA FROM THE
0011 C             100N, 300A, AND 400A AREAS. FLAG 4 REPRESENTS DATA
0012 C             FOR THE HMS 400-FT TOWER. FLAG 5 REPRESENTS DATA FOR
0013 C             WIND TELEMETRY NETWORK SYSTEM. FLAG 6 REPRESENTS
0014 C             TEMPERATURE DATA FROM THE WIND TELEMETRY NETWORK. FLAG
0015 C             7 REPRESENTS PRECIPITATION DATA FROM THE WIND TELEMETRY
0016 C             NETWORK. HOURLY MIXING LEVEL INFORMATION IS ENTERED BY THE
0017 C             FORECASTER ALONG WITH THE HOURLY PEAK GUST INFORMATION.
0018 C
0019 C PASSED PARAMETERS - FLAG      - indicates which to archive
0020 C                        DATE      - archive date
0021 C                        TIME      - archive hour
0022 C                        DATA     - archive data
0023 C                        NUM_OF_VAR - number of variables to archive
0024 C
0025 C VERSION - 1.0      8/1/88      GREGG ANDREWS 376-6104
0026 C      1.1      1/24/89      GLA modified the telemetry wind part of the
0027 C                        code to insert zeros where blanks
0028 C                        were.
0029 C *****
0030
0031 SUBROUTINE write_data(flag, date, time, data, num_of_var)
0032
0033 IMPLICIT NONE
0034
0035 INTEGER*2 i
0036 INTEGER*2 n
0037 INTEGER*2 flag
0038 INTEGER*2 data(3)
0039 INTEGER*2 time(2)
0040 INTEGER*2 num_of_var
0041 INTEGER*2 pkgst_spd
0042 INTEGER*2 pkgst_dir
0043 INTEGER*2 pkgst_time
0044 INTEGER*2 jday
0045 INTEGER*2 stabi
0046 INTEGER*2 mixing_level
0047 INTEGER*2 record
0048
0049 CHARACTER*1 stabc
0050 CHARACTER*2 wnd(50)
0051
0052 REAL*4      data(300)

```



```

0053     REAL*4    tdelta
0054
0055
0056     IF ((flag.EQ.1) .OR. (flag.EQ.2) .OR. (flag.EQ.3)) THEN ! ARCHIVE 200-FT TOWER DATA
0057
0058     C..... SET VARIABLE TO MISSING DATA CODE IF VALUE EQUAL TO 999.9
0059
0060     DO m=1, num_of_var
0061         IF(data(m) .EQ. 999.9) THEN
0062             IF((m.EQ.3) .OR. (m.EQ.4) .OR. (m.EQ.5) .OR.
0063                (m.EQ.8) .OR. (m.EQ.18)) data(m) = 9999.9
0064             IF((m.EQ.11) .OR. (m.EQ.13) .OR. (m.EQ.15))
0065                data(m) = 99.9
0066             IF((m.EQ.10) .OR. (m.EQ.12) .OR. (m.EQ.14))
0067                data(m) = 998.9
0068         ENDIF
0069     ENDDO
0070
0071     C..... PERFORM A GROSS ERROR CHECK ON HOURLY MAX AND MIN DRY BULB TEMP
0072
0073     IF((data(17).GT.150.0) .OR. (data(17).LT.-40.0)) data(17) = 999.9
0074     IF((data(18).GT.150.0) .OR. (data(18).LT.-40.0)) data(18) = 999.9
0075
0076     C..... SET WIND DIRECTION TO 360 IF <= 0; ALSO SET WIND DIRECTION
0077     C..... TO 0.0 IF WIND SPEED EQUAL TO 0.0
0078
0079     IF(data(10) .LT. 5.0) data(10) = 360.0
0080     IF(data(12) .LT. 5.0) data(12) = 360.0
0081     IF(data(14) .LT. 5.0) data(14) = 360.0
0082     IF(data(11) .LT. 0.5) data(10) = 0.0
0083     IF(data(13) .LT. 0.5) data(12) = 0.0
0084     IF(data(15) .LT. 0.5) data(14) = 0.0
0085
0086     C..... WRITE DATA TO OUTPUT FILE
0087
0088     WRITE(3,10) (data(i),i=1,3), time(1), IIFIX((data(1)+.05)*10.0),
0089                IIFIX((data(2)+.05)*10.0), (ININT(data(i)),i=3,6),
0090                (IIFIX((data(i)+.05)*10.0),i=7,9), IIFIX((data(10)+5.0)/10.0),
0091                ININT(data(11)), IIFIX((data(12)+5.0)/10.0),
0092                ININT(data(13)), IIFIX((data(14)+5.0)/10.0),
0093                ININT(data(15)), ININT(data(18)),
0094                IIFIX((data(17)+.05)*10.0),IIFIX((data(18)+.05)*10.0)
0095 10     FORMAT(4I2, 1X, 9I4, 8I2, 3I4)
0096
0097     ELSE IF (flag .EQ. 4) THEN ! ARCHIVE HMS 400-FT TOWER DATA
0098
0099     C..... CHECK FOR MISSING DATA
0100
0101     DO i=1, num_of_var
0102         IF((data(i) .EQ. 999.9) .AND. (i .GT. 11)) THEN
0103             IF((i.EQ.13) .OR. (i.EQ.16) .OR. (i.EQ.19)) THEN
0104                 data(i) = 998.9

```

```

0105             ELSE
0106                 data(i) = 99.0
0107             ENDIF
0108         ENDIF
0109     ENDDO
0110
0111
0112 C..... SET WIND DIRECTION TO 360.0 IF EQUAL 0.0; ALSO SET WIND DIRECTION
0113 C..... TO 0.0 IF WIND SPEED EQUAL TO 0.0
0114
0115     IF(data(13) .LT. 5.0) data(13) = 360.0
0116     IF(data(16) .LT. 5.0) data(16) = 360.0
0117     IF(data(19) .LT. 5.0) data(19) = 360.0
0118     IF(data(14) .LT. 0.5) data(13) = 0.0
0119     IF(data(17) .LT. 0.5) data(16) = 0.0
0120     IF(data(20) .LT. 0.5) data(19) = 0.0
0121
0122 C..... GET PEAK INFORMATION AND WRITE DATA TO OUTPUT FILE
0123
0124     READ(12,'(19X,I2,1X,I2,1X,I1)') pkgst_dir, pkgst_spd, pkgst_time
0125     WRITE(3,20) (data(i),i=1,3), time(1), (IIFIX((data(i)+.05)*10.0),i=1,11),
0126     .           ININT(data(12)),           IIFIX((data(13)+5.0)/10.0),
0127     .           ININT(data(14)),           ININT(data(15)),
0128     .           IIFIX((data(16)+5.0)/10.0),ININT(data(17)),
0129     .           ININT(data(18)),           IIFIX((data(19)+5.0)/10.0),
0130     .           ININT(data(20)),
0131     .           pkgst_dir, pkgst_spd, pkgst_time
0132     WRITE(9,20) (data(i),i=1,3), time(1), (IIFIX((data(i)+.05)*10.0),i=1,11),
0133     .           ININT(data(12)),           IIFIX((data(13)+5.0)/10.0),
0134     .           ININT(data(14)),           ININT(data(15)),
0135     .           IIFIX((data(16)+5.0)/10.0),ININT(data(17)),
0136     .           ININT(data(18)),           IIFIX((data(19)+5.0)/10.0),
0137     .           ININT(data(20)),
0138     .           pkgst_dir, pkgst_spd, pkgst_time
0139 20      FORMAT(4I2, 1X, 11I4, 1X, 11I2, I1)
0140
0141     ELSE IF (flag .EQ. 5) THEN      ! ARCHIVE WIND TELEMETRY DATA
0142
0143 C..... CHECK FOR MISSING DATA
0144
0145     DO i=1, 50
0146         IF(data(i) .EQ. 999.9) THEN
0147             IF(MOD(i,2) .EQ. 0) THEN
0148                 data(i) = 99.0
0149             ELSE
0150                 data(i) = 990.0
0151             ENDIF
0152         ENDIF
0153     ENDDO
0154
0155 C..... IF WIND DIRECTION < 5.0 THEN SET TO 360.0; IF WIND SPEED <
0156 C..... .5 THEN SET WIND DIRECTION TO 0.0

```

```

0157
0158      DO i=2, 50, 2
0159          IF(data(i-1) .LT. 5.0) data(i-1) = 300.0
0160          IF(data(i) .LT. 0.5) data(i-1) = 0.0
0161      ENDDO
0162
0163      C..... GET MIXING DEPTH AND COMPUTE STABILITY
0164
0165      READ(12,'(14X,I3)') mixing_level
0166      IF((data(51).GT.900.0) .OR. (data(52).GT.900.0)) THEN
0167          stabi = 9
0168          GOTO 100
0169      ENDIF
0170      tdelta = data(51) - data(52) ! 200-FT temp - 30-ft temp
0171      tdelta = ((tdelta * .550) / 51.813) * 100.0 ! CONVERT TO C AND NORMALIZE TO 100 METERS
0172      CALL get_stability(tdelta, stabi, stabc)
0173      100 CALL get_julian(date(1), date(2), date(3), jday)
0174
0175
0176      C..... INSERT ZEROS WHERE BLANKS WERE
0177
0178      DO i=1, 50, 2
0179          WRITE(wnd(i), '(I2)') IIFIX((data(i)+5.0)/10.0)
0180          WRITE(wnd(i+1), '(I2)') ININT(data(i+1))
0181
0182          IF(wnd(i)(1:1) .EQ. ' ') wnd(i)(1:1) = '0'
0183          IF(wnd(i+1)(1:1) .EQ. ' ') wnd(i+1)(1:1) = '0'
0184      ENDDO
0185
0186      C..... WRITE DATA TO OUTPUT FILE
0187
0188      record = 1
0189      WRITE(3,30) date(1), jday, time(1), record, stabi, mixing_level,
0190          (wnd(i), wnd(i+1), i=1,19,2)
0191      record = 2
0192      WRITE(3,30) date(1), jday, time(1), record, stabi, mixing_level,
0193          (wnd(i), wnd(i+1), i=21,39,2)
0194      record = 3
0195      WRITE(3,40) date(1), jday, time(1), record, stabi, mixing_level,
0196          (wnd(i), wnd(i+1), i=41,49,2)
0197      30 FORMAT(I2, I3, I2, I1, I2, 1X, I3, 5X, 10(2X,2A2))
0198      40 FORMAT(I2, I3, I2, I1, I2, 1X, I3, 5X, 5(2X,2A2), 5(2X,'8888'))
0199
0200      ELSE IF (flag .EQ. 0) THEN ! ARCHIVE TEMPERATURE DATA FROM TELEMETRY NETWORK
0201
0202      CALL get_julian(date(1), date(2), date(3), jday)
0203
0204      C..... WRITE DATA TO OUTPUT DATA FILE
0205
0206      record = 1
0207      WRITE(3,50) date(1), jday, time(1), record,
0208          (IIFIX((data(i)+.05)*10.0), i=1,10)

```

```

0209      record = 2
0210      WRITE(3,50) date(1), jday, time(1), record,
0211              (IIFIX((data(i)+.05)*10.0),i=11,20)
0212      record = 3
0213      WRITE(3,60) date(1), jday, time(1), record,
0214              (IIFIX((data(i)+.05)*10.0),i=21,25)
0215      50      FORMAT(I2, I3, I2, I1, 11X, 10(1X,I5))
0216      60      FORMAT(I2, I3, I2, I1, 11X, 5(1X,I5))
0217
0218      ELSE IF (flag .EQ. 7) THEN    ! ARCHIVE PRECIPITATION DATA FROM TELEMETRY NETWORK
0219
0220      C..... CHECK FOR MISSING DATA
0221
0222      DO i=1, num_of_var
0223          IF((data(i).EQ.999.9) .OR. (data(i).GT.100.0)) THEN
0224              data(i) = 99.99
0225          ENDIF
0226      ENDDO
0227
0228      CALL get_julian(date(1), date(2), date(3), jday)
0229
0230      C..... WRITE DATA TO OUTPUT FILE
0231
0232      WRITE(3,70) date(1), jday, time(1), (IIFIX(data(i)),i=1,num_of_var)
0233      70      FORMAT(I2,I3,I2,12X,4(2X,I4))
0234
0235      ENDIF
0236
0237      RETURN
0238      END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	3402	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	188	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	292	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	3880	

ENTRY POINTS

Address	Type	Name	References
0-00000000		WRITE_DATA	31#

VARIABLES

Address	Type	Name	Attributes	References
AP-000000040	I*2	FLAG	31	37 58(3) 97 141 200 218
**	I*2	I	35	88(8)= 101= 102(2) 103(3) 104 106 125(4)=
			132(4)=	145= 146 147 148 150 158= 159(2)
			160(2)	178= 179(2) 180(2) 182(2) 183(2) 189(3)= 192(3)=
			195(3)=	207(2)= 210(2)= 213(2)= 222= 223(2) 224 232(2)=
2-00000008C	I*2	JDAY	44	173A 189 192 195 202A 207 210
			213	228A 232
**	I*2	M	38	60= 81 82(8) 84(4) 86(4)
2-000000070	I*2	MIXING_LEVEL	46	165= 189 192 195
AP-0000000140	I*2	MUM_OF_VAR	31	40 80 101 222 232
2-000000068	I*2	PKGST_DIR	42	124= 125 132
2-000000066	I*2	PKGST_SPD	41	124= 125 132
2-00000006A	I*2	PKGST_TIME	43	124= 125 132
**	I*2	RECORD	47	188= 189 191= 192 194= 195 206=
			207	209= 210 212= 213
2-000000004	CHAR	STABC	49	172A
2-00000000E	I*2	STABI	45	167= 172A 189 192 195
2-000000078	R*4	TDelta	63	170= 171(2)= 172A

ARRAYS

Address	Type	Name	Attributes	Bytes	Dimensions	References
AP-0000000100	R*4	DATA		1200	(300)	31 52 61 62= 64=
				86=	73(3)=	74(3)= 79(2)= 80(2)=
				81(2)=	82(2)=	83(2)= 84(2)= 86(13)
				102	104=	106= 115(2)= 116(2)=
				117(2)=	118(2)=	119(2)= 120(2)= 125(10)
				132(10)	146	148= 150= 159(2)=
				160(2)=	166(2)	170(2) 179 180
				207	210	213 223(2) 224=
				232		
AP-0000000000	I*2	DATE		8	(3)	31 38 88 125 132
				173(3)A	189	192 195 202(3)A
				207	210	213 228(3)A 232
AP-00000000C0	I*2	TIME		4	(2)	31 39 88 125 132
				189	192	195 207 210
				213	232	
2-0000000000	CHAR	WWD		100	(50)	50 179 180 182(2)= 183(2)=
				189(2)	192(2)	195(2)

LABELS

Address	Label	References			
1-00000000	10'	88	95#		
1-00000020	20'	125	132	139#	
1-00000030	30'	189	192	197#	
1-00000057	40'	195	198#		
1-00000060	50'	207	210	215#	
1-00000094	60'	213	218#		
1-000000AB	70'	232	233#		
0-000007AE	100	168	173#		

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	References			
	GET_JULIAN	173	202	228	
	GET_STABILITY	172			
I*2	WTH\$ININT	88(5)	125(8)	132(8)	180

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 WRITE_DATA

/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:      15.06 seconds
Elapsed Time:  17.57 seconds
Page Faults:   577
Dynamic Memory: 584 pages
```

```

0001 C *****
0002 C
0003 C      PROGRAM - GET_JULIAN
0004 C
0005 C
0006 C      PROGRAMMER - GREGG ANDREWS 376-0130
0007 C
0008 C
0009 C      PURPOSE - DETERMINES THE JULIAN DAY FOR A GIVEN MONTH AND DAY.
0010 C
0011 C
0012 C ALGORITHM - PASS MONTH AND DAY PASS BACK INTEGER JULIAN DAY.
0013 C
0014 C
0015 C PASSED PARAMETERS -      YEAR - year of julian day to be computed
0016 C                      MONTH - month of julian day to be computed
0017 C                      DAY   - gregorian day of julian day to be computed
0018 C                      JULIAN_DAY - julian day
0019 C
0020 C THIS SUBROUTINE ASSUMES THAT THE DATA PASSED IS VALID.
0021 C
0022 C
0023 C VERSION - 1.0
0024 C
0025 C
0026 C DATE - NOVEMBER 1987
0027 C
0028 C *****
0029
0030 SUBROUTINE get_julian(year, month, day, julian_day)
0031
0032 IMPLICIT NONE
0033
0034 INTEGER*2 year, month, day, julian_day, days_in_month(12)
0035
0036 DATA days_in_month/ 0, 31, 59, 90, 120, 151,
0037 .                  181, 212, 243, 273, 304, 334/
0038
0039
0040 C
0041 C..... COMPUTE JULIAN DAY
0042 C
0043
0044 julian_day = days_in_month(month) + day
0045
0046 IF((MOD(year,4).EQ.0) .AND. (month.ge.3))
0047 .   julian_day = julian_day + 1
0048
0049
0050 RETURN
0051 END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	57	PIC CON REL LCL SHR EXE RD MCWRT LONG
2 \$LOCAL	24	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	81	

ENTRY POINTS

Address	Type	Name	References
0-00000000		GET_JULIAN	30#

VARIABLES

Address	Type	Name	Attributes	References
AP-00000000	I*2	DAY		30 34 44
AP-00000010	I*2	JULIAN_DAY		30 34 44= 46(2)=
AP-00000020	I*2	MONTH		30 34 44 46
AP-00000040	I*2	YEAR		30 34 46

ARRAYS

Address	Type	Name	Attributes	Bytes	Dimensions	References
2-00000000	I*2	DAYS_IN_MONTH		24	(12)	34 36D 44

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

COMMAND QUALIFIERS

```

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

/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:          1.35 seconds
Elapsed Time:      2.39 seconds
Page Faults:       478
Dynamic Memory:    315 pages
```

```

0001 C *****
0002 C
0003 C      PROGRAM - GET_STABILITY
0004 C
0005 C
0006 C      PROGRAMMER - GREGG ANDREWS  378-0138
0007 C
0008 C
0009 C      PURPOSE - DETERMINES THE ATMOSPHERIC STABILITY USING THE
0010 C      DELTA-TEMPERATURE SCHEME DEVELOPED BY THE NRC; NUREG-0917
0011 C      CONTAINS SOME INFORMATION ON THE SCHEME.  THE
0012 C      DELTA-TEMPERATURE IS SUPPLIED BY THE EMS FIFTEEN MINUTE
0013 C      DATA BASE.
0014 C
0015 C
0016 C ALGORITHM - PASS DELTA-TEMPERATURE VALUE PASS BACK INTEGER
0017 C      STABILITY CLASS
0018 C
0019 C
0020 C PASSED PARAMETERS -      DELTA_TEMP value is supplied by the ems
0021 C      data base.
0022 C      STAB_CLASS1 integer value passed back to the
0023 C      the main calling program
0024 C      STAB_CLASSC character value passed back to the
0025 C      the main calling program
0026 C
0027 C THE IS SUBROUTINE ASSUMES THAT THE PASSED DATA IS VALID.
0028 C
0029 C VERSION - 1.0
0030 C
0031 C
0032 C DATE - NOVEMBER 1987
0033 C
0034 C *****
0035
0036 SUBROUTINE get_stability(delta_temp, stab_class1, stab_classc)
0037
0038 INTEGER*2 stab_class1
0039 REAL*4 delta_temp
0040 CHARACTER*1 stab_classc
0041
0042 C
0043 C..... INITIALIZE STAB_CLASS
0044 C
0045
0046 stab_class1 = 9
0047 stab_classc = 'M'
0048
0049 C
0050 C..... DETERMINE STABILITY CLASS
0051 C
0052

```

```

0053      IF((delta_temp.GT.4.0) .AND. (delta_temp.LT.30)) THEN    ! VERY STABLE
0054          stab_classi = 7
0055          stab_classc = 'Q'
0056      ENDIF
0057
0058      IF(delta_temp .LE. 4.0) THEN    ! STABLE
0059          stab_classi = 6
0060          stab_classc = 'F'
0061      ENDIF
0062
0063      IF(delta_temp .LE. 1.5) THEN    ! SLIGHTLY STABLE
0064          stab_classi = 5
0065          stab_classc = 'E'
0066      ENDIF
0067
0068      IF(delta_temp .LE. -0.5) THEN    ! NEUTRAL
0069          stab_classi = 4
0070          stab_classc = 'D'
0071      ENDIF
0072
0073      IF(delta_temp .LE. -1.5) THEN    ! SLIGHTLY UNSTABLE
0074          stab_classi = 3
0075          stab_classc = 'C'
0076      ENDIF
0077
0078      IF(delta_temp .LE. -1.7) THEN    ! UNSTABLE
0079          stab_classi = 2
0080          stab_classc = 'B'
0081      ENDIF
0082
0083      IF(delta_temp .LE. -1.9) THEN    ! VERY UNSTABLE
0084          stab_classi = 1
0085          stab_classc = 'A'
0086      ENDIF
0087
0088      RETURN
0089      END

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	154	PIC CON REL LCL SHR EXE RD NOWRT LONG
2 \$LOCAL	8	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	162	

ENTRY POINTS

Address	Type	Name	References
0-00000000		GET_STABILITY	38‡

VARIABLES

Address	Type	Name	Attributes	References
AP-000000040	R+4	DELTA_TEMP		38 39 53(2) 58 83 88 73 78
AP-0000000C0	CHAR	STAB_CLASSC		83 38 48 47= 55= 88= 85= 78= 75=
AP-000000080	I+2	STAB_CLASSI		88= 85= 38 38 48= 54= 59= 84= 89= 74=
				79= 84=

```

-----
|          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 GET_STABILITY

```

/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:      1.98 seconds
Elapsed Time:   3.83 seconds
Page Faults:    481
Dynamic Memory: 328 pages

```


APPENDIX B

DESCRIPTION AND FLOWCHART OF THE ARCHIVE COMMAND FILE

APPENDIX B

DESCRIPTION AND FLOWCHART OF THE ARCHIVE COMMAND FILE

This appendix describes in detail the ARCHIVE.COM command file and presents a flowchart indicating the logical functions of the ARCHIVE.COM process. A listing of the ARCHIVE.COM file is presented at the end of this appendix.

The ARCHIVE.COM command file contains VAX VMS operating system commands that control various functions related to the ARCHIVE process. Figure B.1 shows a flowchart containing the logical sequence of commands. The process begins by turning on the VERIFY utility, which records all actions in the log file. This step is followed by setting the default directory to the [FCST.ARCHIVE] subdirectory. This directory contains the executable version of the ARCHIVE program and some of the input files necessary for the ARCHIVE process. Next, a warning utility is activated to direct the process to the FATAL_ERROR label. The commands following this label send an error message to the ARCHIVE.LOG file. At this point the ARCHIVE.EXE file is initiated. If an error occurs during this process, control is returned to ARCHIVE.COM, which then directs itself to send an error message to the ARCHIVE.LOG file. Next, the TEMP.OUT file is sent to the HMS line printer; ARCHIVE.COM waits 15 seconds before deleting the TEMP.OUT file. This step is followed by purging previous versions of the ARCHIVE.LOG file and deleting the HOURLY_MIX_PKGST.DAT, which is located in the [FCST.DAT] subdirectory. If no errors have occurred, the process moves to the portion of the command file that calculates the next date of submission to the SYS\$BATCH queue. The variable TIME_STRING contains this information. The SUBMIT command contains qualifiers that restart the ARCHIVE.COM command file after the system failure (hardware or software). In addition, a log file is maintained throughout the entire process. This file will contain valuable information regarding any errors during the process. The time of the next submission is always 12:40 a.m. PST (or 00:40) the next night. This process takes approximately 15 to 20 min to complete.

FIGURE B.1. Flowchart of the ARCHIVE COMMAND File

```

$ !
$ !
$ !
$ ! THIS COMMAND FILE SUBMITS THE PROGRAM ARCHIVE
$ ! TO SYS$BATCH EVERY DAY AT 0040 TO FACILITATE THE
$ ! ARCHIVING OF SELECTED EMS DATA.
$ !
$ ! PROGRAMMER : GREGG ANDREWS 376-6104 8/88
$ !
$ !
$ ! THE FOLLOWING DCL COMMANDS RUN THE ARCHIVE PROGRAM,
$ !
$ set verify
$ set default [fcst.archive]
$ ON warning THEN GOTO fatal_error
$ r archive
$ print/noburst/noflag_page temp.out
$ wait 00:00:15
$ del temp.out;*
$ pur *.log
$ del [fcst.dat]hourly_mix_pkgst.dat;*
$ GOTO no_error
$ !
$ !
$ ! THE FOLLOWING DCL COMMANDS AND CODE WILL SUBMIT THE NEXT BATCH JOB
$ ! IN ABSOLUTE TIME FOR THE NEXT HOUR. THIS CODE IS NECESSARY IN ORDER
$ ! TO PREVENT TIME CREEPING. AN ALGORITHM FOLLOWS:
$ !
$ ! GET SYSTEM CURRENT TIME
$ ! EXTRACT YEAR,MONTH,DAY AND HOUR USING F$EXTRACT
$ ! CONVERT YEAR,DAY AND HOUR TO INTEGER USING F$INTEGER
$ ! INITIALIZE DAYS IN MONTH
$ ! IF LEAP YEAR THEN
$ ! SET FEB TO 29
$ ! CHECK FOR END OF MONTH
$ ! IF NO GOTO DAY CHECK
$ ! CHECK FOR END OF DAY
$ ! IF NO GOTO HOUR CHECK
$ ! INCREMENT HOUR
$ ! CONVERT YEAR,DAY, AND HOUR TO CHARACTER STRINGS USING F$STRING
$ ! SUBMIT BATCH JOB FOR 20 MINUTES PASSED THE NEXT HOUR
$ !
$ !
$ fatal_error:
$ print_error_message.doc
$ print *.log
$ no_error:
$ current_time = f$time()
$ length = f$locate("-",current_time) ! DETERMIN LENGTH TO 1ST
$ _"
$ dd = f$extract(0,length,current_time) ! EXTRACT DAY
$ mm = f$extract(length+1,3,current_time) ! EXTRACT MONTH

```

```

$ yy          = f$extract(length+5,4,current_time) ! EXTRACT YEAR
$ !
$ !
$ dd          = f$integer(dd)                      ! CONVERT DAY TO INTEGER
$ yy          = f$integer(yy)                      ! CONVERT YEAR TO INTEGER
$ !
$ !
$ Jan = 31                                         ! INITIALIZE DAYS IN MONTH
$ Feb = 28
$ Mar = 31
$ Apr = 30
$ May = 31
$ Jun = 30
$ Jul = 31
$ Aug = 31
$ Sep = 30
$ Oct = 31
$ Nov = 30
$ Dec = 31
$ !
$ !
$ check = yy / 4                                  ! CHECK FOR LEAP YEAR
$ if (check .ne. ((yy+1) / 4)) then goto END_OF_MONTH_CHECK
$   If (check .ne. ((yy+2) / 4)) then goto END_OF_MONTH_CHECK
$     If (check .ne. ((yy+3) / 4)) then goto END_OF_MONTH_CHECK
$       Feb = 29
$ !
$ !
$ END_OF_MONTH_CHECK:
$ if (mm .nes. "JAN") then goto L1
$   if(dd.ne.Jan) then goto ADD_A_DAY
$     mm = "FEB"
$     dd = 1
$     goto FINISHED
$ L1:
$ if (mm .nes. "FEB") then goto L2
$   if(dd.ne.Feb) then goto ADD_A_DAY
$     mm = "MAR"
$     dd = 1
$     goto FINISHED
$ L2:
$ if (mm .nes. "MAR") then goto L3
$   if(dd.ne.Mar) then goto ADD_A_DAY
$     mm = "APR"
$     dd = 1
$     goto FINISHED
$ L3:
$ if (mm .nes. "APR") then goto L4
$   if(dd.ne.Apr) then goto ADD_A_DAY
$     mm = "MAY"
$     dd = 1
$     goto FINISHED

```

```

$ L4:
$ if (mm .nes. "MAY") then goto L5
$   if(dd.ne.May) then goto ADD_A_DAY
$     mm = "JUN"
$     dd = 1
$     goto FINISHED
$ L5:
$ if (mm .nes. "JUN") then goto L6
$   if(dd.ne.Jun) then goto ADD_A_DAY
$     mm = "JUL"
$     dd = 1
$     goto FINISHED
$ L6:
$ if (mm .nes. "JUL") then goto L7
$   if(dd.ne.Jul) then goto ADD_A_DAY
$     mm = "AUG"
$     dd = 1
$     goto FINISHED
$ L7:
$ if (mm .nes. "AUG") then goto L8
$   if(dd.ne.Aug) then goto ADD_A_DAY
$     mm = "SEP"
$     dd = 1
$     goto FINISHED
$ L8:
$ if (mm .nes. "SEP") then goto L9
$   if(dd.ne.Sep) then goto ADD_A_DAY
$     mm = "OCT"
$     dd = 1
$     goto FINISHED
$ L9:
$ if (mm .nes. "OCT") then goto L10
$   if(dd.ne.Oct) then goto ADD_A_DAY
$     mm = "NOV"
$     dd = 1
$     goto FINISHED
$ L10:
$ if (mm .nes. "NOV") then goto L11
$   if(dd.ne.Nov) then goto ADD_A_DAY
$     mm = "DEC"
$     dd = 1
$     goto FINISHED
$ L11:
$ if (mm .nes. "DEC") then goto ADD_A_DAY
$   if(dd.ne.Dec) then goto ADD_A_DAY
$     yy = yy + 1
$     mm = "JAN"
$     dd = 1
$     goto FINISHED
$ !
$ ADD_A_DAY:
$   dd = dd + 1

```

```

$ !
$ !
$ FINISHED:
$ !
$ !
$ yy          = f$string(yy)          ! CONVERT YEAR TO CHAR
$ dd          = f$string(dd)          ! CONVERT DAY TO CAHR
$ !
$ !
$ time_string = f$fao("!6(AS)",dd,"-",mm,"-",yy,":00:40:00")
$ submit/restart/keep/log_file=[fcst.archive]archive.log/-
after='time_string' archive
$ !
$ ! END OF COMMAND FILE ARCHIVE.COM

```

APPENDIX C

DESCRIPTION AND LISTING OF INPUT FILE CONTENTS

APPENDIX C

DESCRIPTION AND LISTING OF INPUT FILE CONTENTS

This appendix describes in detail the contents of each input file used in the ARCHIVE process. Each file is addressed individually and a listing of each file's contents accompanies each description.

The ARCHIVE program accesses eight input files during execution. Seven of these files contain internal identifications of selected meteorological variables and the first record contains information about the file. The files are located in the [FCST.ARCHIVE] subdirectory. These internal identifications uniquely identify each meteorological parameter in the hourly and 15-min data bases (Andrews 1988). The eighth file contains hourly atmospheric mixing depth and peak gust information. These data are hand-entered by the on-duty meteorologist.

The first seven files follow an (I3) FORTRAN format while the eighth file follows a (1X,I6,1X,I2,4X,I3,2X,I2,1X,I2,1X,I1) FORTRAN format.

The TOWER_400_IDS.DAT contains internal identifications for meteorological variables measured on the 410-ft tower located at the HMS. There are, in total, 20 identifications contained in the file. Figure C.1 displays the file contents exactly as they appear in the file.

```
THIS FILE CONTAINS INTERNAL IDS FOR THE 410-FT TOWER AT HMS
339      1/2 IN SOIL TEMP
340      15 IN SOIL TEMP
341      36 IN SOIL TEMP
233      3 FT DRY BULB TEMP
9        30 FT DRY BULB TEMP
10       50 FT DRY BULB TEMP
11      100 FT DRY BULB TEMP
16      200 FT DRY BULB TEMP
14      250 FT DRY BULB TEMP
12      300 FT DRY BULB TEMP
15      400 FT DRY BULB TEMP
1        7 FT WIND SPEED
19      50 FT WIND DIRECTION
```

FIGURE C.1. Contents of the TOWER_400_IDS.DAT File

3	50 FT WIND SPEED
5	100 FT WIND SPEED
21	200 FT WIND DIRECTION
6	200 FT WIND SPEED
7	300 FT WIND SPEED
24	400 FT WIND DIRECTION
8	400 FT WIND SPEED

FIGURE C.1. (contd)

The TOWER_200_400A_IDS.DAT contains internal identifications for meteorological variables measured on the 200-ft tower located at the 400 Area (FFTF). There are a total of 18 identifications contained in the file. Figure C.2 displays the file contents exactly as they appear in the file.

THIS FILE CONTAINS INTERNAL IDS FOR THE 200-FT TOWER AT THE 400A

113	2 M DEW POINT TEMP
112	60-10 M DELTA TEMP
111	10 M SIGMA THETA
108	10 M SIGMA SPEED
109	60 M SIGMA THETA
106	60 M SIGMA SPEED
105	2 M DRY BULB TEMP
104	10 M DRY BULB TEMP
100	60 M DRY BULB TEMP
103	10 M WIND DIRECTION
99	10 M WIND SPEED
102	25 M WIND DIRECTION
98	25 M WIND SPEED
101	60 M WIND DIRECTION
97	60 M WIND SPEED
236	10 M HOURLY MAX WIND SPEED
256	2 M HOURLY MAX DRY BULB TEMP
261	2 M HOURLY MIN DRY BULB TEMP

FIGURE C.2. Contents of the TOWER_200_400A_IDS.DAT File

The TOWER_200_300A_IDS.DAT contains internal identifications for meteorological variables measured on the 200-ft tower located at the 300 Area. There are a total of 18 identifications contained in the file. Figure C.3 displays the file contents exactly as they appear in the file.

THIS FILE CONTAINS INTERNAL IDS FOR THE 200-FT TOWER AT THE 300A

37	2 M DEW POINT TEMP
159	60-10 M DELTA TEMP
155	10 M SIGMA THETA
149	10 M SIGMA SPEED
157	60 M SIGMA THETA
151	60 M SIGMA SPEED
143	2 M DRY BULB TEMP
144	10 M DRY BULB TEMP
145	60 M DRY BULB TEMP
117	10 M WIND DIRECTION
114	10 M WIND SPEED
115	25 M WIND DIRECTION
135	25 M WIND SPEED
116	60 M WIND DIRECTION
136	60 M WIND SPEED
244	10 M HOURLY MAX WIND SPEED
257	2 M HOURLY MAX DRY BULB TEMP
260	2 M HOURLY MIN DRY BULB TEMP

FIGURE C.3. Contents of the TOWER_200_300A_IDS.DAT File

The TOWER_200_100N_IDS.DAT contains internal identifications for meteorological variables measured on the 200-ft tower located at the 100N Area. There are a total of 18 identifications contained in the file. Figure C.4 displays the file contents exactly as they appear in the file.

THIS FILE CONTAINS INTERNAL IDS FOR THE 200-FT TOWER AT THE 100N

160	2 M DEW POINT TEMP
158	60-10 M DELTA TEMP
152	10 M SIGMA THETA
146	10 M SIGMA SPEED
154	60 M SIGMA THETA
148	60 M SIGMA SPEED
140	2 M DRY BULB TEMP
141	10 M DRY BULB TEMP
142	60 M DRY BULB TEMP
137	10 M WIND DIRECTION
132	10 M WIND SPEED
138	25 M WIND DIRECTION
133	25 M WIND SPEED
139	60 M WIND DIRECTION
134	60 M WIND SPEED
49	10 M HOURLY MAX WIND SPEED
258	2 M HOURLY MAX DRY BULB TEMP
259	2 M HOURLY MIN DRY BULB TEMP

FIGURE C.4. Contents of the TOWER_200_100N_IDS.DAT File

The TELEMETRY_WINDS_IDS.DAT file contains internal identifications for meteorological variables measured on the 10-m telemetry meteorological network (Figure 1). There are a total of 52 identifications contained in the file. All but the last two identifications correspond to wind direction and wind speed. The last two are identifications for temperatures from the 410-ft tower which are used to calculate atmospheric stability. Figure C.5 displays the file contents exactly as they appear in the file.

THIS FILE CONTAINS INTERNAL IDS FOR 10-M WIND TELEMETRY STATIONS

215	PROSSER WIND DIRECTION
29	PROSSER WIND SPEED
216	EOC WIND DIRECTION
30	EOC WIND SPEED
217	ARMY WIND DIRECTION
31	ARMY WIND SPEED
218	RATTLESNAKE SPRINGS WIND DIRECTION
32	RATTLESNAKE SPRINGS WIND SPEED
219	EDNA WIND DIRECTION
33	EDNA WIND SPEED
221	200E WIND DIRECTION
34	200E WIND SPEED
220	200W WIND DIRECTION
35	200W WIND SPEED
222	WAHLUKEE SLOPE WIND DIRECTION
36	WAHLUKEE SLOPE WIND SPEED
103	FFTF (400A) WIND DIRECTION
99	FFTF (400A) WIND SPEED
223	YAKIMA BARRICADE WIND DIRECTION
38	YAKIMA BARRICADE WIND SPEED
117	300A WIND DIRECTION
114	300A WIND SPEED
224	WYE BARRICADE WIND DIRECTION
40	WYE BARRICADE WIND SPEED
137	100N WIND DIRECTION
132	100N WIND SPEED
225	WPPSS WIND DIRECTION
42	WPPSS WIND SPEED
226	FRANKLIN COUNTY WIND DIRECTION
43	FRANKLIN COUNTY WIND SPEED
13	GABLE MOUNTAIN WIND DIRECTION
44	GABLE MOUNTAIN WIND SPEED
227	RINGOLD WIND DIRECTION

FIGURE C.5. Contents of the TELEMETRY_WINDS_IDS.DAT File

45	RINGOLD WIND SPEED
228	RICHLAND WIND DIRECTION
46	RICHLAND WIND SPEED
229	SAGE HILL WIND DIRECTION
47	SAGE HILL WIND SPEED
230	RATTLESNAKE MOUNTAIN WIND DIRECTION
48	RATTLESNAKE MOUNTIAN WIND SPEED
18	HMS WIND DIRECTION
2	HMS WIND SPEED
235	PASCO WIND DIRECTION
50	PASCO WIND SPEED
95	GABLE WEST WIND DIRECTION
94	GABLE WEST WIND SPEED
96	100F WIND DIRECTION
93	100F WIND SPEED
386	VERNITA BRIDGE WIND DIRECTION
387	VERNITA BRIDGE WIND SPEED
16	200-FT TEMPERATURE FROM THE HMS 400-FT TOWER
9	30-FT TEMPERATURE FROM THE HMS 400-FT TOWER

FIGURE C.5. (contd)

The TELEMETRY_TEMPS_IDS.DAT file contains internal identifications for 2-m atmospheric dry bulb temperature as measured by the 10-m telemetry meteorological network (Figure 1). There are a total of 25 identifications contained in the file. Figure C.6 displays the file contents exactly as they appear in the file.

THIS FILE CONTAINS INTERNAL IDS FOR 2-M TEMP TELEMETRY STATIONS		
346	AT-PROS-2M	AT = ATMOSPHERIC TEMPERATURE
241	AT-EOC-2M	
297	AT-ARMY-2M	
246	AT-RSPG-2M	
281	AT-EDNA-2M	
342	AT-200E-2M	
282	AT-200W-2M	
283	AT-WAHL-2M	
105	AT-FFTF-2M	
278	AT-YAKB-2M	
143	AT-300A-2M	
284	AT-WYEB-2M	
140	AT-100A-2M	
286	AT-WPPS-2M	
309	AT-FRNK-2M	

FIGURE C.6. Contents of the TELEMETRY_TEMPS_IDS.DAT File

285	AT-GABL-2M
310	AT-RING-2M
347	AT-RICH-2M
289	AT-SAGE-2M
239	AT-RMTN-2M
233	AT-6T-3F
380	AT-PASC-2M
348	AT-GBLW-2M
349	AT-100F-2M
391	AT-VERN-2M

FIGURE C.6. (contd)

The TELEMETRY_PRECIP_IDS.DAT file contains internal identifications for precipitation data as measured by the 10-m telemetry meteorological network (Figure 1). There are a total of 4 identifications contained in the file. Figure C.7 displays the file contents exactly as they appear in the file.

THIS FILE CONTAINS INTERNAL IDS FOR PRECIP DATA AT TELEMETRY SITES	
238	RF-RSPG
274	RF-YAKB
663	RF-300A
276	RF-PASC

FIGURE C.7. Contents of the TELEMETRY_PRECIP_IDS.DAT File

The eighth input file contains hourly atmospheric mixing depth and maximum 50-ft wind speed (peak gust) information. The file name is HOURLY_MIX_PKGST.DAT and resides in the [FCST.DAT] subdirectory. This file is maintained by the MPG program (see the Preface). The first record contains data for the first hour of the day and a new record is added each hour. At midnight of the current day the file contains 24 records corresponding to the hours of the current day. After the ARCHIVE.COM is complete HOURLY_MIX_PKGST.DAT is deleted. A new file is created on the next execution of the MPG program. The file HOURLY_MIX_PKGST.DAT consists of:

- Gregorian date (e.g., 890302)
- hour (e.g., 1)
- atmospheric mixing depth in meters (e.g., 50)
- direction of peak gust in tens-of-degrees from north (e.g., 32)
- speed in mph (e.g., 17)
- time code (e.g., 9).

The examples given above correspond to the first record in Figure C.8. The figure displays only six records of the HOURLY_MIX_PKGST.DAT file.

890302	1	50	32	17	9
890302	2	1	0	0	0
890302	3	1	0	0	0
890302	4	1	0	0	0
890302	5	1	0	0	0
890302	6	1	0	0	0

FIGURE C.8. Contents of the HOURLY_MIX_PKGST.DAT File

REFERENCE

Andrews, G. L. 1988. The Hanford Meteorological Data Collection System and Data Base. PNL-6509, Pacific Northwest Laboratory, Richland, Washington.

APPENDIX D

DESCRIPTION AND LISTING OF OUTPUT FILE CONTENTS

APPENDIX D

DESCRIPTION AND LISTING OF OUTPUT FILE CONTENTS

This appendix describes in detail the contents and format of the seven monthly output files accessed by the ARCHIVE computer program. Each file is described and listed. In addition, each description gives information about the record format and unit of measure of each variable. Each file name contains MMM, which indicates the three-character abbreviation of the current month.

Seven monthly output files are accessed by the ARCHIVE program during execution (see Table 2). These files reside in the [FCST.DAT] subdirectory until the end of the year, when they are merged into a yearly file. The yearly files are stored on removable disk packs labelled HMS_DATA and are located in the HMS computer room.

The TOWER_400MMM.DAT file contains hourly meteorological data from the HMS 410-ft tower. Each record in the file corresponds to one hour of data. An example output of the TOWER_400MMM.OAT file is given in Figure D.1. Table D.1 indicates the position of each variable in each record, format of each variable, and the unit of measure. The complete FORTRAN FORMAT statement for this file is (4I2, 1X, 11I4, 1X, 11I2, 1I).

```
89 3 1 1 353 399 413 330 331 325 323 314 318 307 309 536 8 9 11010 110 0 00
89 3 1 2 343 399 414 313 313 307 304 296 299 289 291 535 7 935 9103510 0 00
89 3 1 3 336 398 414 306 306 299 296 288 292 282 284 534 7 935 9103510 0 00
89 3 1 4 330 398 415 294 294 287 284 275 280 269 270 534 8103511113511 0 00
89 3 1 5 324 397 415 279 278 271 268 260 263 253 255 533 8 93410103411 0 00
89 3 1 6 321 395 414 268 267 261 258 249 253 243 244 532 8 93310103410 0 00
89 3 1 7 319 394 414 263 262 255 252 244 248 238 240 531 8 93210113310 0 00
89 3 1 8 316 392 414 257 256 249 246 237 241 231 233 531 910321112331132151
89 3 1 9 316 391 414 253 250 243 238 230 234 225 226 631 9103210103310 0 00
89 3 110 318 389 414 255 251 244 241 230 234 224 226 531 7 832 8 833 8 0 00
89 3 111 321 387 415 257 252 244 241 231 234 224 226 531 7 832 9 832 8 0 00
89 3 112 324 386 415 262 257 250 245 237 241 231 231 432 6 733 7 734 7 0 00
```

FIGURE D.1. Sample Output from the TOWER_400MMM.DAT Output File

TABLE D.1. Information About the TOWER_400MMM.DAT Output File

Column(s)	Variable	FORTTRAN Format	Unit of Measure
1-2	Year	I2	Whole Year
3-4	Month	I2	Whole Month
5-6	Day	I2	Whole Day
7-8	Hour	I2	Whole Hour
9	Blank	1X	N/A
10-13	-1/2-in. Temperature	I4	Tenths of °F
14-17	- 15-in. Temperature	I4	Tenths of °F
18-21	- 36-in. Temperature	I4	Tenths of °F
22-25	3-ft Temperature	I4	Tenths of °F
26-29	30-ft Temperature	I4	Tenths of °F
30-33	50-ft Temperature	I4	Tenths of °F
34-37	100-ft Temperature	I4	Tenths of °F
38-41	200-ft Temperature	I4	Tenths of °F
42-45	250-ft Temperature	I4	Tenths of °F
46-49	300-ft Temperature	I4	Tenths of °F
50-53	400-ft Temperature	I4	Tenths of °F
54	Blank	1X	N/A
55-56	7-ft Wind Speed	I2	Whole mph
57-58	50-ft Wind Direction	I2	Tens of Degree
59-60	50-ft Wind Speed	I2	Whole mph
61-62	100-ft Wind Speed	I2	Whole mph
63-64	200-ft Wind Direction	I2	Tens of Degree
65-66	200-ft Wind Speed	I2	Whole mph
67-68	300-ft Wind Speed	I2	Whole mph
69-70	400-ft Wind Direction	I2	Tens of Degree
71-72	400-ft Wind Speed	I2	Whole mph
73-74	Wind Direction of Maximum	I2	Tens of Degree
	50-ft Wind Speed (if >= 15 mph)		
75-76	Maximum 50-ft Wind Speed (if >= 15 mph)	I2	Whole mph
77	Time code for Maximum 50-ft Wind Speed	I1	Code

The TOWER_200_AREAMMM.DAT monthly output file contains hourly meteorological data from the 200-ft towers located at either the 100N Area, 300 Area, or 400 Area (FFTF). Data from each area are stored in separate files. The character string AREA in the above file name represents one of the following: 100N, 300A, or 400A. The MMM represents the three-character identification of the current month. Each record represents one hour of data. An example output of TOWER_200_AREAMMM.DAT file is shown in Figure D.2. Table D.2 indicates the position of each variable in each record, format of each variable,

```

89 3 1 1 259 -6 43 2 41 2 351 348 345 113 114 116 20 354 347
89 3 1 2 247 -6 29 2 25 2 344 341 338 114 115 116 20 348 340
89 3 1 3 234 -6 45 2 36 2 335 332 329 114 115 116 18 340 330
89 3 1 4 227 -6 44 2 35 2 324 321 318 11436153616 19 330 318
89 3 1 5 220 -6 40 1 30 1 315 312 310351335143515 17 318 314
89 3 1 6 222 -7 39 2 27 2 307 303 300351435153517 19 312 299
89 3 1 7 231 -7 16 2 8 2 296 293 290341434153417 20 300 294
89 3 1 8 221 -7 27 2 12 1 294 289 286341334143515 17 300 292
89 3 1 9 214 -8 25 2 7 1 292 287 282341434153516 20 294 290
89 3 110 206 -9 15 2 4 1 291 285 280341534163417 20 293 289
89 3 111 199 -9 30 2 19 1 298 291 285351335143514 19 309 290
89 3 112 186 -9 40 2 39 2 303 296 291 1143615 116 21 308 298

```

FIGURE D.2. Sample Output from the TOWER_200_AREAMMM.DAT Output File

TABLE D.2. Information About the TOWER_200_AREAMMM.DAT Output File

Column(s)		Variable	FORTTRAN Format	Unit of Measure
1-2		Year	I2	Whole Year
3-4		Month	I2	Whole Month
5-6		Day	I2	Whole Day
7-8		Hour	I2	Whole Hour
9		Blank	1X	N/A
10-13	2-m	Dew Point Temperature	I4	Tenths of °F
14-17	60m-10m	Delta Temperature	I4	Tenths of °F
18-21	10-m	Sigma Theta	I4	Whole Degrees
22-25	10-m	Sigma Wind Speed	I4	Whole mph
26-29	60-m	Sigma Theta	I4	Whole Degrees
30-33	60-m	Sigma Wind Speed	I4	Whole mph
34-37	2-m	Temperature	I4	Tenths of °F
38-41	10-m	Temperature	I4	Tenths of °F
42-45	60-m	Temperature	I4	Tenths of °F
46-47	10-m	Wind Direction	I2	Tens of Degrees
48-49	10-m	Wind Speed	I2	Whole mph
50-51	25-m	Wind Direction	I2	Tens of Degrees
52-53	25-m	Wind Speed	I2	Whole mph
54-55	60-m	Wind Direction	I2	Tens of Degrees
56-57	60-m	Wind Speed	I2	Whole mph
58-61	10-m	Hourly Maximum Wind Speed	I4	Whole mph
62-65	2-m	Hourly Maximum Temperature	I4	Tenths of °F
66-69	2-m	Hourly Minimum Temperature	I4	Tenths of °F

and the unit of measure. The information from Table D.2 is applicable to all three files. The complete FORTRAN FORMAT statement for this file is (4I2, 1X, 9I4, 6I2, 3I4).

The TELEMETRY_WINDSMMM.DAT monthly output file contains hourly meteorological information from HMS's wind telemetry network. Currently, there are a total of 25 wind stations reporting wind direction and wind speed to the HMS computer. Each hour of information occupies three records in this file. Figure D.3 shows an example hour from a TELEMETRY_WINDSMMM.DAT file. Table D.3 indicates the position of each variable in each record, format of each variable, and the unit of measure. Please note the table is segmented into three parts, which corresponds to the three records. The complete FORTRAN FORMAT statement for this file is (I2, I3, I2, I1, I2, 1X, I3, 5X, (10(2X,2I2))).

```

89 60 11 2 14      3613 3513 0111 3509 3410 0110 3509 0113 0215 0109
89 60 12 2 14      0113 3612 0217 3611 3610 0209 0210 3612 0212 3418
89 60 13 2 14      0107 3611 3607 0109 3615 8888 8888 8888 8888 8888

```

FIGURE D.3. Sample Output from the TELEMETRY_WINDSMMM.DAT Output File

TABLE D.3. Information About the TELEMETRY_WINDSMMM.DAT Output File

Column(s)	Variable	FORTRAN Format	Unit of Measure
<u>Hourly Record 1</u>			
1-2	Year	I2	Whole Year
3-5	Julian Day	I3	Whole Julian Day
6-7	Hour	I2	Whole Hour
8	Hourly Record Number	I1	Integer
9-10	Atmospheric Stability	I2	NRC Code
11	Blank	1X	N/A
12-14	Mixing Depth	I3	Decameters
15-21	Blank	N/A	N/A
22-23	10-m Prosser Wind Direction	I2	Tens of Degrees
24-25	10-m Prosser Wind Speed	I2	Whole mph
26-27	Blank	2X	N/A
28-29	10-m EOC Wind Direction	I2	Tens of Degrees
30-31	10-m EOC Wind Speed	I2	Whole mph
32-33	Blank	2X	N/A
34-35	10-m Army Loop Wind Direction	I2	Tens of Degrees
36-37	10-m Army Loop Wind Speed	I2	Whole mph
38-39	Blank	2X	N/A
40-41	10-m Rattlesnake Springs Wind Direction	I2	Tens of Degrees
42-43	10-m Rattlesnake Springs Wind Speed	I2	Whole mph
44-45	Blank	2X	N/A
46-47	10-m Edna Wind Direction	I2	Tens of Degrees
48-49	10-m Edna Wind Speed	I2	Whole mph

TABLE D.3. (contd)

Column(s)	Variable		FORTTRAN Format	Unit of Measure
52-53	10-m	200E Wind Direction	I2	Tens of Degrees
54-55	10-m	200E Wind Speed	I2	Whole mph
56-57		Blank	2X	N/A
58-59	10-m	200W Wind Direction	I2	Tens of Degrees
60-61	10-m	200W Wind Speed	I2	Whole mph
62-63		Blank	2X	N/A
64-65	10-m	Wahlukee Slope Wind Direction	I2	Tens of Degrees
66-67	10-m	Wahlukee Slope Wind Speed	I2	Whole mph
68-69		Blank	2X	N/A
70-71	10-m	FFTF Wind Direction	I2	Tens of Degrees
72-73	10-m	FFTF Wind Speed	I2	Whole mph
74-75		Blank	2X	N/A
76-77	10-m	Yakima Barricade Wind Direction	I2	Tens of Degrees
78-79	10-m	Yakima Barricade Wind Speed	I2	Whole mph

Hourly Record 2

1-2		Year	I2	Whole Year
3-5		Julian Day	I3	Whole Julian Day
6-7		Hour	I2	Whole Hour
8		Hourly Record Number	I1	Integer
9-10		Atmospheric Stability	I2	NRC Code
11		Blank	1X	N/A
12-14		Mixing Depth	I3	Decameters
15-21		Blank	N/A	N/A
22-23	10-m	300A Wind Direction	I2	Tens of Degrees
24-25	10-m	300A Wind Speed	I2	Whole mph
26-27		Blank	2X	N/A
28-29	10-m	WYE Barricade Wind Direction	I2	Tens of Degrees
30-31	10-m	WYE Barricade Wind Speed	I2	Whole mph
32-33		Blank	2X	N/A
34-35	10-m	100N Wind Direction	I2	Tens of Degrees
36-37	10-m	100N Wind Speed	I2	Whole mph
38-39		Blank	2X	N/A
40-41	10-m	WPPSS Wind Direction	I2	Tens of Degrees
42-43	10-m	WPSS Wind Speed	I2	Whole mph
44-45		Blank	2X	N/A
46-47	10-m	Franklin County Wind Direction	I2	Tens of Degrees
48-49	10-m	Franklin County Wind Speed	I2	Whole mph
50-51		Blank	2X	N/A
52-53	10-m	Gable Mountain Wind Direction	I2	Tens of Degrees
54-55	10-m	Gable Mountain Wind Speed	I2	Whole mph
56-57		Blank	2X	N/A
58-59	10-m	Ringold Wind Direction	I2	Tens of Degrees
60-61	10-m	Ringold Wind Speed	I2	Whole mph
62-63		Blank	2X	N/A
64-65	10-m	Richland Wind Direction	I2	Tens of Degrees

TABLE D.3. (contd)

Column(s)		Variable	FORTTRAN Format	Unit of Measure
66-67	10-m	Richland Wind Speed	I2	Whole mph
68-69		Blank	2X	N/A
70-71	10-m	Sage Hill Wind Direction	I2	Tens of Degrees
72-73	10-m	Sage Hill Wind Speed	I2	Whole mph
74-75		Blank	2X	N/A
76-77	10-m	Rattlesnake Mountain Wind Direction	I2	Tens of Degrees
78-79	10-m	Rattlesnake Mountain Wind Speed	I2	Whole mph

Hourly Record 3

1-2		Year	I2	Whole Year
3-5		Julian Day	I3	Whole Julian Day
6-7		Hour	I2	Whole Hour
8		Hourly Record Number	I1	Integer
9-10		Atmospheric Stability	I2	NRC Code
11		Blank	1X	N/A
12-14		Mixing Depth	I3	Decameters
15-21		Blank	N/A	N/A
22-23	10-m	HMS Wind Direction	I2	Tens of Degrees
24-25	10-m	HMS Wind Speed	I2	Whole mph
26-27		Blank	2X	N/A
28-29	10-m	Pasco Wind Direction	I2	Tens of Degrees
30-31	10-m	Pasco Wind Speed	I2	Whole mph
32-33		Blank	2X	N/A
34-35	10-m	Gable Mountain Wind Direction	I2	Tens of Degrees
36-37	10-m	Gable Mountain Wind Speed	I2	Whole mph
38-39		Blank	2X	N/A
40-41	10-m	100F Wind Direction	I2	Tens of Degrees
42-43	10-m	100F Wind Speed	I2	Whole mph
44-45		Blank	2X	N/A
46-47	10-m	Vernita Bridge Wind Direction	I2	Tens of Degrees
48-49	10-m	Vernita Bridge Wind Speed	I2	Whole mph

The TELEMETRY_TEMPSMMM.DAT monthly output file contains hourly 2-m temperature information from HMS's telemetry network. Currently, there are 25 telemetry stations reporting to the HMS computer. Each hour of information occupies three records in this file. Figure D.4 shows an example hour from a TELEMETRY_TEMPSMMM.DAT file. Table D.4 indicates the position of each variable in each record, format of each variable, and the unit of measure. Please note the table is segmented into three parts which corresponds to the three records. The complete FORTRAN FORMAT statement for this file is (I2, I3, I2, I1, 11X, 10(1X,I5)).

89 60 11	343	313	341	334	346	336	337	302	345	311
89 60 12	351	329	333	344	322	296	334	351	316	221
89 60 13	330	348	326	347	322					

FIGURE D.4. Sample Output from the TELEMETRY_TEMPMMM.DAT Output File

TABLE D.4. Information About the TELEMETRY_TEMPMMM.DAT Output File

Column(s)	Variable		FORTTRAN Format	Unit of Measure
<u>Hourly Record 1</u>				
1-2		Year	I2	Whole Year
3-5		Julian Day	I3	Whole Julian Day
6-7		Hour	I2	Whole Hour
8		Hourly Record Number	I1	Integer
9-20		Blank	12X	N/A
21-25	2-m	Prosser Temperature	I5	Tenths of °F
26		Blank	1X	N/A
27-31	2-m	EOC Temperature	I5	Tenths of °F
32		Blank	1X	N/A
33-37	2-m	Army Loop Temperature	I5	Tenths of °F
38		Blank	1X	N/A
39-43	2-m	Rattlesnake Springs Temperature	I5	Tenths of °F
44		Blank	1X	N/A
45-49	2-m	Edna Temperature	I5	Tenths of °F
50		Blank	1X	N/A
51-55	2-m	200E Temperature	I5	Tenths of °F
56		Blank	1X	N/A
57-61	2-m	200W Temperature	I5	Tenths of °F
62		Blank	1X	N/A
63-67	2-m	Wahlukee Slope Temperature	I5	Tenths of °F
68		Blank	1X	N/A
69-73	2-m	FFTF Temperature	I5	Tenths of °F
74		Blank	1X	N/A
75-79	2-m	Yakima Temperature	I5	Tenths of °F
<u>Hourly Record 2</u>				
1-2		Year	I2	Whole Year
3-5		Julian Day	I3	Whole Julian Day
6-7		Hour	I2	Whole Hour
8		Hourly Record Number	I1	Integer
9-20		Blank	12X	N/A
21-25	2-m	300A Temperature	I5	Tenths of °F
26		Blank	1X	N/A
27-31	2-m	WYE Barricade Temperature	I5	Tenths of °F
32		Blank	1X	N/A
33-37	2-m	100N Temperature	I5	Tenths of °F
38		Blank	1X	N/A

TABLE D.4. (contd)

Column(s)		Variable	FORTTRAN Format	Unit of Measure
39-43	2-m	WPPSS Temperature	I5	Tenths of °F
44		Blank	1X	N/A
45-49	2-m	Franklin County Temperature	I5	Tenths of °F
50		Blank	1X	N/A
51-55	2-m	Gable Mountain Temperature	I5	Tenths of °F
56		Blank	1X	N/A
57-61	2-m	Ringold Temperature	I5	Tenths of °F
62		Blank	1X	N/A
63-67	2-m	Richland Temperature	I5	Tenths of °F
68		Blank	1X	N/A
69-73	2-m	Sage Hill Temperature	I5	Tenths of °F
74		Blank	1X	N/A
75-79	2-m	Rattlesnake Mountain Temperature	I5	Tenths of °F

Hourly Record 3

1-2		Year	I2	Whole Year
3-5		Julian Day	I3	Whole Julian Day
6-7		Hour	I2	Whole Hour
8		Hourly Record Number	I1	Integer
9-20		Blank	12X	N/A
21-25	2-m	HMS Temperature	I5	Tenths of °F
26		Blank	1X	N/A
27-31	2-m	Pasco Temperature	I5	Tenths of °F
32		Blank	1X	N/A
33-37	2-m	Gable West Temperature	I5	Tenths of °F
38		Blank	1X	N/A
39-43	2-m	100F Temperature	I5	Tenths of °F
44		Blank	1X	N/A
45-49	2-m	Vernita Bridge Temperature	I5	Tenths of °F

The TELEMETRY_PRECIPMMM.DAT monthly output file contains hourly precipitation information from HMS's telemetry network. Currently, there are four telemetry stations reporting precipitation data to the HMS computer. Each hour of information occupies one record in this file. Figure D.5 shows an example of 12 hours from a TELEMETRY_PRECIPMMM.DAT file. Table D.5 indicates the position of each variable in each record, format of each variable, and the unit of measure. The complete FORTRAN FORMAT statement for this file is (I2, I3, I2, 12X, 4(2X,I4)).

89 60 1	2	3	6	2
89 60 2	10	14	12	10
89 60 3	0	0	0	0
89 60 4	0	0	0	0
89 60 5	0	0	0	0
89 60 6	0	0	0	0
89 60 7	0	0	0	0
89 60 8	0	0	0	0
89 60 9	0	0	0	0
89 6010	0	0	0	0
89 6011	0	0	0	0
89 6012	0	0	0	0

FIGURE D.5. Sample Output from the TELEMETRY_PRECIPMMM.DAT Output File

TABLE D.5. Information About the TELEMETRY_PRECIPMMM.DAT Output File

Column(s)	Variable	FORTTRAN Format	Unit of Measure
1-2	Year	I2	Whole Year
3-5	Julian Day	I3	Whole Julian Day
6-7	Hour	I2	Whole Hour
8-21	Blank	14X	N/A
22-25	Precipitation at Rattlesnake Springs	I4	Hundredths of Inches
26-27	Blank	2X	N/A
28-31	Precipitation at Yakima Barricade	I4	Hundredths of Inches
32-33	Blank	2X	N/A
34-37	Precipitation at 300 Area	I4	Hundredths of Inches
38-39	Blank	2X	N/A
40-43	Precipitation at Pasco Airport	I4	Hundredths of Inches

DISTRIBUTION

No. of
Copies

OFFSITE

2 DOE/Office of Scientific and
Technical Information

ONSITE

DOE Richland Operation Office

E. C. Norman

32 Pacific Northwest Laboratory

G. L. Andrews (5)

K. W. Burk (5)

J. W. Falco

O. P. Gifford

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

R. L. Skaggs

J. A. Stottlemire

R. E. Wildung

Publishing Coordination

Technical Report Files (5)