Research Technical Completion Report

NHIMS

DESIGN DOCUMENT FOR THE NORTHWEST HYDROLOGIC INFORMATION MANAGEMENT SYSTEM

APPENDIX

by

Mary Jo Bluske
Myron Molnau

Department of Agricultural Engineering

Idaho Water Resources Research Institute
University of Idaho
Moscow, Idaho 83843

August, 1987

The research on which this report is based was financed in part by the United States Department of the Interior as authorized by the Water Research and Development Act of 1978 (P.L. 95-467).

Research Technical Completion Report

14-08-0001-G1222-32

DESIGN DOCUMENT FOR THE NORTHWEST HYDROLOGIC INFORMATION MANAGEMENT SYSTEM (NHIMS)

APPENDIX

by

Mary Jo Bluske
Myron Molnau

Department of Agricultural Engineering

Submitted to:
U.S. Geological Survey
United States Department of the Interior
Washington, D.C. 20242

Idaho Water Resources Research Institute
University of Idaho
Moscow, Idaho 83843

August, 1987

TABLE OF CONTENTS

LIST OF FIGURES

1.0 SCOPE

A data management system called the Hydrologic Information Storage And Retrieval System (HISARS) is used at the University of Idaho to provide various individuals and organizations with easy access to climatic and hydrologic information [10]. However, HISARS is now over ten years old [18], and due to its age and original design, the system is no longer easily updated, and it is difficult to modify the program code. In order to take advantage of more modern computer systems and more sophisticated software, it was decided to replace HISARS with a SAS* based data management system called the Northwest Hydrologic Information Management System (NHIMS). (In previous documents the system was named HISAS, but the name has since been changed.)

The objective of the NHIMS project is to create a data management system that is easy to use, maintain and modify, and at the same time simulates the actions of the existing HISARS system as much as possible. The basis for achieving these objectives is the use of the SAS software system. NHIMS will be written in the SAS language, and will make extensive use of SAS macros. SAS software provides the tools to easily create and manage large data sets, to perform statistical analyses on the data, and to generate output listings in a wide variety of formats. In addition, the SAS system offers good facilities for documenting both the system programs and the data base itself.

The purpose of this document is to describe in detail the design of the NHIMS system. All components of the system will be covered: the general and detailed structure of the system, the program modules, the permanent NHIMS files, global data, and provisions for testing and packaging the system. Future maintenance programmers and SAS programmers who want to access the data independently of the NHIMS system can use this document as a reference.

Providing users easy access to climatic and hydrologic data means that the NHIMS system must perform five main functions:

1) read and decipher the user's commands
2) retrieve the appropriate data from the permanent files
3) list the data in a printed report
4) copy the data to an external file
5) process the data by searching or performing statistical analyses

*SAS is the registered trademark of SAS Institute Inc., Cary, NC, USA.

SCOPE

The NHIMS system will offer some major advantages to both users and maintenance programmers. More options will be available to the user, and in many cases, more flexibility in regards to how the data are accessed. Data updates will be more easily performed, providing more up-to-date information. Maintenance tasks will require much less programming, and programmers will have the ability to add new files (see Appendix 8.1) or new types of processing functions (see Appendix 8.2) to the system. However, the price for such improvements will be an increase in the cost of data retrieval.

NHIMS is designed to run on the IBM 4381 mainframe at the University of Idaho. The software required to run the system are the OS/VS1 operating system and the 5.16 version of the SAS software package. NHIMS should be compatible with future releases of SAS software; also, with minor modifications, the system should be able to run using the VM/CMS operating system as well. At this point in time, the permanent files are stored on a mountable disk pack (USR008), and the program code will be stored there as well. However, as of May 31, 1987, mountable disk packs will no longer be supported by Computer Services; after that time, the main data files will be stored on tape, with the index and pointer files probably to be kept on system disks.

Acknowledgements must be made to Ed Wiser at North Carolina State University, author of the HISARS system [18]. HISARS has served the purpose of providing fast, efficient, and easy data retrieval for many years. The NHIMS system, whose internals are governed by the use of SAS software, nevertheless can attribute much of its design to the original HISARS system.

2.0 APPLICABLE DOCUMENTS

1) Bluske, Mary Jo. September, 1985. "Proposal for the NHIMS* Project."

2) Bluske, Mary Jo. January, 1986. "Software Requirements Document for the NHIMS* Project."

3) Bluske, Mary Jo. July, 1986. "A SAS Based Hydrologic Storage and Retrieval System." Research Technical Completion Report for the Idaho Water Resources Research Institute. With Myron Molnau. Paper no. 14-08-0001-G1014.

4) Bluske, Mary Jo. June, 1986. "SAS Software for Managing Climatic Data." Paper for the 1986 Summer Meeting of the American Society of Agricultural Engineers. With Myron Molnau. Paper no. 86-4018.

5) Calingaert, Peter. 1979. Assemblers, Compilers, and Program Translation. Rockville, MD: Computer Science Press.

6) Dineley, Virginia M., Stephen M. Beatrous, Darylene C. Colbert, and Bruce M. Tindall. 1985. "Writing Efficient SAS DATA Steps." Proceedings SUGI, pp. 684-687.

7) Howard, Neil and Linda Williams Pickle. 1984. "Efficient Data Retrieval: Direct Access Using the Point Option." Proceedings SUGI, pp. 294-298.

8) McGregor, Scott L. and Mary Nelson. 1983. "Some Guidelines For Documenting SAS Source Code." Proceedings SUGI, pp. 119-122.

9) Merlin, Ross Z. 1984. "Design Concepts for SAS Applications." Proceedings SUGI, pp. 283-287.

10) Molnau, Myron. 1983. Climate and Hydrology for Idaho. Idaho Agriculture Experiment Station Misc. Series No. 32(rev).

*The original name of the system, HISAS, was changed to NHIMS.

APPLICABLE DOCUMENTS

11) Muller, K.E., J. Smith, and D.H. Christiansen. 1981. "Rules We Followed and Wish We Had Followed in Managing Datasets, Programs and Printouts." Proceedings SUGI, pp. 401-405.

12) Phillips, Jeff. 1985. "Designing Macro-Based Systems." Proceedings SUGI, pp. 1130-1140.

13) Pressman, Roger S. 1982. Software Engineering: Practitioner's Approach. New York: McGraw-Hill Book Company.

14) Reel, Joy. 1983. "The SAS Macro Language." Proceedings SUGI, pp. 969-978.

15) SAS Institute Inc. SAS Applications Guide, 1980 Edition. Cary, NC: SAS Institute Inc., 1980.

16) SAS Institute Inc. SAS Macro Language Course Notes, 1983 Edition. Cary, NC: SAS Institute Inc., 1983.

17) SAS Institute Inc. SAS User's Guide: Basics, 1982 Edition. Cary, NC: SAS Institute Inc., 1982.

18) Wiser, Edward H. 1975. HISARS, Hydrologic information storage and retrieval system. North Carolina Agriculture Experiment Station Technical Bulletin No. 215.

3.0 SYSTEM DESIGN DESCRIPTION OVERVIEW

The NHIMS system will serve as a data base management system for a collection of hydrologic and climatic data. Included here are a narrative of the general design and a program structure diagram which graphically illustrates the components of the NHIMS system.

3.1 DESIGN OVERVIEW NARRATIVE

The design narrative contains a description of the data base, a general breakdown of the system design, some of the design considerations and guidelines, a description of how the system is to be used, and a detailed outline of the NHIMS command language.

3.1.1 DATA BASE DESCRIPTION

The NHIMS data base consists of several permanent SAS data sets, each containing data on one climatic or hydrologic element. (For efinition of a SAS data set, see [17, p. 361].) Currently there are 10 data sets: air temperature, precipitation, streamflow, snowfall, snow course, evaporation, reservoir storage, peakflow, hourly precipitation, and monthly summary. It is anticipated that more elements will be added in the future; a strategy for accomplishing such additions is given in Appendix 8.1. For most files, each record contains one month's data, 31 daily values, from one weather station. Within a file, the records are sorted by the station number, then the year and month. In general, the files are quite large; for example, the streamflow file already contains 136,000 records, with a new record generated for each station every month.

In addition to the main SAS data sets, the NHIMS data base contains a single pointer file for each element file, and one main index file, all of which are permanent SAS data sets. Because the main files are organized by the station numbers, each pointer file serves as a map, containing the first and last observation numbers for each station in the the main file. Thus, the pointer files will be used in the NHIMS system to allow direct access of information from the data base. The index file, on the other hand, has one entry for each unique station number in all of the NHIMS files, with the corresponding descriptive information for that station. Such information includes the station's name, the county in which it is located, its latitude, longitude, elevation, and other descriptors which give a complete profile of the station.

3.1.2 GENERAL BREAKDOWN OF SYSTEM DESIGN

The design of the NHIMS system is based on the user submitting a group of commands in a batch job, identifying the type and amount of data he wants; NHIMS will retrieve data from the permanent files based on the user's commands. Each job submitted by the user will consist of one or more access requests; one access request calls for the retrieval and output of specific data values. Therefore, the NHIMS system will first separate the user commands into individual access requests, storing each set of commands in a single, temporary data set. Then, for each set of access commands, the system will perform the same five tasks. First, the commands will be parsed to determine the exact data to be retrieved, and what operations are to be performed on the data. Second, the appropriate data are brought from the permanent files into temporary subsets, using the pointer files to directly access the data [7, p. 295], [3, p. 44]. Next, any combination of the three possible actions are taken: the data are listed in a printed report, copied to an external file, or processed in some manner before being printed. Figure 1 contains a system structure diagram which illustrates these general functions.

3.1.3 SYSTEM DESIGN GUIDELINES

Guiding the design of NHIMS is the use of the SAS language, which provides many features for statistical analysis, information storage and retrieval, and the writing of printed reports. NHIMS relies heavily on the SAS macro language, which acts as a preprocessor, generating text for the SAS compiler [14, p. 969], [16, p. 7]. SAS macros allow the conditional generation of program statements [12, p. 1132]. In addition, macro variables store data that can be used subsequently by other DATA and PROC steps. Thus, macro variables provide a method for passing information between DATA and/or PROC steps without having to create a SAS data set in which to store that information. NHIMS makes extensive use of macro variables to store information given by the user, and conditionally generates program statements for the compiler based upon the values of the variables.

Maintainability, ease of system use, and efficiency of data retrieval are the main objectives in the design of NHIMS. Use of the SAS system itself ensures a certain degree of maintainability; standard procedures and special program statements simplify the data management tasks, and minimize the overall time required for maintenance programming. The macro language provides for modular design of the system, easing the future maintenance tasks. Also, PROC SOURCE will be used to permanently store the source code in several members of a partitioned data set; the same proc can be used to retrieve and edit those members. Secondly, in simulating

the existing system, NHIMS will provide users with easy access to the information they are accustomed to getting with HISARS, allowing them to use the same command language that they already know. Moreover, new options will be available in NHIMS that will make the system even more versatile to use. Finally, although SAS software cannot be described as efficient, certain steps were taken to maximize the efficiency of the NHIMS system. The %INCLUDE statement will be used to bring in the source code from a partitioned data set only as needed. The use of the pointer files for direct access, the uniformity of the file structures, and the creation of data subsets during program execution, all represent attempts to make the system operate more efficiently.

3.1.4 SYSTEM USAGE

In order to access the NHIMS system, users must submit a program in IBM job control language (JCL), including an EXEC statement which executes a procedure to be called NHIMS. The NHIMS procedure will be responsible for setting up the JCL statements which access the SAS system, define the external files and a NHIMS library, and identify external devices to which the output is to be directed. The required JCL will be specified in the user's manual, so that a user will not need to learn JCL in order to use the system. The following is a sample of the type of JCL statements that would be required:

```
//Jname  JOB   (project,aaa-bb-cccc),'user'
//       EXEC  NHIMS
//NHIMS  DD    *

       -------  NHIMS COMMANDS  -------

/*
```

For the NHIMS commands, the user will code a series of keywords and operands which will serve as input to the system. The NHIMS command language is almost identical to the one currently used in HISARS, except that more commands and options are available. These commands tell the system what and how much data are requested, and what is to be done with the data. For example, to request daily precipitation data from the Moscow station (number 106152), for the years 1983 and 1984, one must issue the following commands:

```
ACCESS
ELEMENT  PRECIPITATION
STATION  106152
PERIOD   1/1983  TO  12/1984
LIST     DAILY
```

Each job submitted by the user will contain one or more access requests; one access request retrieves, analyzes and prints one specific subset of data. All printed output will be directed to the standard output file with the FILE PRINT statement.

Regarding limitations of system usage, the permanent files will be protected from the user, allowing read-only access to the data. Secondly, the user will not be able to access the entire contents of any of the main files, although listings of the entire index and pointer files will be allowed.

In addition, SAS programmers will be able to use the NHIMS files directly with their own programs, without having to access the NHIMS programs.

3.1.5 COMMAND LANGUAGE SPECIFICATIONS

The NHIMS command language consists of commands (keywords) followed by one or more operands; because most of the commands are identical to the existing HISARS commands, additional descriptions can be obtained from the HISARS user's manual [10]. The available NHIMS commands are:

ACCESS
ELEMENT
STATION
COUNTY
BASIN or HUCODE
REGION or DIVISION
ELEVATION
AREA
PERIOD
POINTERS
NOSORT
METRIC
NOHEADER
AND/OR
LIST
COPY
PROCESS

The format for the commands follows a few simple rules, but is otherwise quite free. The rules are:

a) Each access request must contain one ACCESS command and at least one ELEMENT command.

b) Except for the ACCESS command, which must be first, and the PROCESS command, which must be last, the commands can be listed in any order.

c) Each command must start in column 1, followed by one or more blanks, and then the necessary operand(s). An operand can begin in any column except column 1.

d) The operands for a command can be listed all on one line, separated by blanks, or on multiple lines, as long as the first column is blank if the line contains only operands.

e) Commands may be used multiple times with different operands.

f) The end of an access request is identified by an end of file mark, another ACCESS command, or a PROCESS command.

The following is a detailed description of the NHIMS commands and their operands.

1) ACCESS

The ACCESS command signals the beginning of a group of commands that constitute a single access request; ACCESS must always be the first command in a group of commands. Anything else coded in the same line following the ACCESS command will be ignored, and is therefore a good place for comments.

2) ELEMENT

The ELEMENT command identifies the element or elements for which access is requested. The operand must give the element in one of the standard forms. The following elements will be available in the first start-up of NHIMS:

a) PRECIPITATION or RAINFALL
b) TEMPERATURE
c) STREAMFLOW
d) SNOWFALL
e) EVAPORATION
f) PEAKFLOW
g) SNOWCOURSE or COURSE
h) MONTHLY
i) STORAGE
j) HOURRAIN or HOURPRCP

Any number of elements may be specified in a single access request as long as they are separated by blanks.

3) STATION

The STATION command is used to request access to specific stations. Standard agency codes are used, except that only the numbers, without any punctuation, are allowed.

The 8-digit code of the U.S. Geological Survey is used for the streamflow, peak flow and reservoir files. The 6-digit code of the National Weather Service is used for the precipitation, temperature, evaporation, snowfall, and monthly files. The 5-digit code of the Soil Conservation Service is used for the snow course file.

4) COUNTY

The COUNTY command is used to access stations by county. The operand consists of a string of one or more characters. All stations are retrieved which have the same character string in the name of the county. Multiple county names must be separated by one or more blanks; however, if the county name has 2 words (i.e., Twin Falls), be careful that only one blank is used to separate the two words.

5) BASIN or HUCODE

Either the BASIN or the HUCODE command can be used to access stations by river basin. The operand is the 8-digit hydrologic unit code.

6) REGION or DIVISION

Either the REGION or the DIVISION command can be used to access stations by climatological region. The operand is the 2-digit code devised by the National Weather Service, with values ranging from 1-10. A leading zero in front of a 1-digit number is optional.

7) ELEVATION

The ELEVATION command is used to access stations within a given range of elevations. The operand is given in the form MIN TO MAX. For example, the command ELEVATION 2000 TO 2500 will access all stations between 2000 and 2500 feet elevation, inclusive. Multiple ranges may be given on one or more lines, as long as one complete range is given on the same line. Also, if only a lower limit is desired, only a single limit need be given. Any stations with elevations equal to or above the lower limit will be retrieved.

8) AREA

The AREA command is used to access streamflow stations by drainage area. The format and usage of the operand is identical to that of the ELEVATION command given previously, except that the limits are of drainage areas in square miles. Again, if only a lower limit is required, a single limit is sufficient. Blank values are common for drainage areas; such stations cannot be accessed using the AREA command.

9) PERIOD

The PERIOD command is used to specify the beginning and ending dates of one or more periods of record. Without the PERIOD command, the entire period of record is accessed for every station in an access request. Thus, one can limit the amount of data retrieved to a certain range of dates, saving processing time and money. The format of the operands are BEGIN TO END, where BEGIN and END are dates in the form MONTH/YEAR. For example, the command:

PERIOD 6/1980 TO 8/1980 6/1981 TO 8/1981

would limit access to records for the three summer months in 1980 and 1981. The format requires the slash ('/') immediately preceeded by the month and followed by the calendar year. Any number of periods may be specified, as long as both a beginning and ending date are given. The periods may be given on multiple lines, as long as succeeding lines of operands have a blank in the first column and as long as a complete range is always given on the same line.

10) POINTERS

The POINTERS command is a new command not currently available in HISARS; it is used to explicitly identify the observation numbers of the requested data, eliminating the need for the NHIMS programs to search the pointer files and thus retrieve the data more quickly and efficiently. Obviously, only one element may be requested when using the POINTERS command, for the observation numbers only have meaning when referring to one of the main files. Similarly, do not use the command when requesting a listing of the index or the pointer files. Also, it is necessary to use the STATION command in conjunction with the POINTERS command, in order to give the station number which matches the pointers. The key to using the POINTER command is knowing the information in the appropriate pointer file, probably by first getting a listing of that file.

The format of the operands is BEGIN TO END, where BEGIN is the first observation number for which data is requested, and END is the last observation number. All of the data between BEGIN and END must contain data on one station only. For example, the commands:

```
STATION   106152
POINTERS  45116 TO 46135
```

would tell the system to retrieve data on the Moscow station between the observations 45,116 and 46,135, inclusive.

It is recommended that the POINTERS command be used with only one operand, i.e. one range of pointers. However, it is possible to use multiple ranges, as long as they are given in ascending order on separate lines and the STATION command is used to provide the matching station numbers. For example:

```
STATION   100010  106152  106152
POINTERS  5 TO 500
          45116 TO 45140
          46000 TO 46024
```

The above commands would tell the system to retrieve data on station 100010 from observations 5 to 500 and on station 106152 from observations 45,116 to 45,140 and 46,000 to 46,024, inclusive. Thus, use of the POINTERS command with multiple ranges can be used to retrieve non-contiguous records for one station.

11) SORTED

The SORTED command is a new command not currently available in HISARS; it is used to indicate that the station numbers given by the user are listed in sorted order and the system does not need to sort them before retrieving the data. Thus, the purpose of this command is to make the system run more efficiently. The SORTED command should only be used when explicitly listing station numbers with a STATION command. There are no operands.

The order in which the numbers must be sorted is ascending order. NHIMS treats the station numbers as character data, using the EBCDIC collating sequence. Therefore, the following station numbers are in the proper sorted order for NHIMS:

```
NAYLOR
10UI05
106152
12414500
16D03
16D03000
24MONT
```

12) METRIC

The METRIC command is a new command not currently available in HISARS; it is used to tell the system that all output should be in metric units. There are no operands. The metric units corresponding to standard English units are:

degrees Fahrenheit: degrees Celsius
inches: millimeters (centimeters for snowfall data)
feet: meters
miles: kilometers
cubic feet: cubic meters
acre feet: cubic meters

13) NOHEADER

The NOHEADER command will suppress the heading page that is printed for each access request. If this command is not used, the heading page is automatically printed. The NOHEADER command has no operands.

14) AND/OR

The AND/OR commands can be used to access records that satisfy one or more criteria. For example, the commands:

```
COUNTY    LATAH
OR
DIVISION  2
```

will retrieve records for all stations that are either in Latah county or in Division 2. Note that the word OR is used in the logical sense. However, the above commands used without the OR command will achieve the same results; in other words, the system default is to retrieve records that match any of the criteria given by the user. The main purpose of explicitly coding the OR command is to help the user recognize the logic of the request.

The AND command is used when all of the criteria given by the user must be met. For example, the commands:

```
COUNTY     TWIN FALLS
AND
ELEVATION  2000
```

will retrieve records for all stations that are both in Twin Falls county and are at an elevation that is equal to or greater than 2000 feet. Note that the AND command is also used in the logical sense. There are no operands for the AND/OR commands.

15) LIST

The LIST command directs the system to produce listings of the requested data. The following operands are permitted:

a) INDEX
b) DAILY
c) MONTHLY
d) POINTERS
e) CONTENTS
f) HOURLY

The INDEX operand indicates that a listing of the index data is to be produced. Several options are available. LIST INDEX will produce an index listing for the requested stations that are in the NHIMS system, along with the beginning and ending dates for the period of record. This command produces the same results as it does with HISARS. LIST INDEX ALL will print index listings for all requested stations, whether or not they have data in the NHIMS system. LIST INDEX NHIMS will print index listings for all requested stations that are in the NHIMS system, but no period of record information will be given. LIST INDEX NONHIMS will print index listings for all requested stations that are not in the NHIMS system. For all of the LIST INDEX options, one or more elements must be specified.

The LIST DAILY and LIST MONTHLY commands indicate that data are to be listed for the specified time periods. There is one option available for these commands, and that is to specify the month for which the listing is to begin. For example, LIST MONTHLY 6 will begin the listing with June as the first month. The default is 1 except for snowfall files where the default is 7 and the streamflow and reservoir files where the default is 10.

Listings of monthly values, unlike in HISARS, will no longer contain the ratio of the annual total to the annual average for years with complete records. This is due to the fact that the file must be processed twice in order to get this information, and hence the cost of retrieval is increased. It will still be possible to get these ratios, by using the AVERAGE option, i.e., LIST MONTHLY AVERAGE.

The LIST POINTERS command directs the system to produce listings of the pointer files; either a subset of the files or complete listings can be produced.

The LIST CONTENTS command is used to obtain listings of data in the files which do not have the standard monthly record structure. These files include snow course, peak flow, and monthly summary.

The LIST HOURLY command is used to obtain listings of the hourly precipitation file.

16) COPY

The COPY command is provided to permit users to copy records from the NHIMS data files to other formats, for use with other languages. The available operands are:

a) FORMATTED
b) DIRECT
c) 80

The DIRECT operand causes the SAS data subset itself to be directly copied to an external file. The FORMATTED operand causes the data to be written to a file with a format, and the 80 operand writes formatted data in 80-byte records. Without an operand, the data is written without format. The LRECL and BLKSIZE values for the external files are described in Appendix 8.4.

COPY operands that were available in HISARS, INTEGER and REAL, are allowable with NHIMS, but they have no meaning. Data will be printed as integer or real, depending on the element type. For example, temperature data will be written as integers, and precipitation data will be written as real, irregardless of the operand specified by the user. Thus, no scaling of the data is required, as was sometimes the case with HISARS.

The user will have to supply a JCL DD statement for each requested element; each element will be written to a separate, external file. Within the file, the data will be ordered by station number, and then the date. The DD statements must have special names: OUT1 for the first element, OUT2 for the second element, and so forth for each element. Irregardless of the order in which the user specifies the elements, the files will be written in this order:

PRECIPITATION
TEMPERATURE
STREAMFLOW
SNOWFALL
EVAPORATION
SNOW COURSE
MONTHLY SUMMARY
PEAKFLOW
RESERVOIR STORAGE
HOURLY PRECIPITATION

Thus, if the user asks for precipitation and snowfall elements, OUT1 will contain precipitation data and OUT2 will contain snowfall data.

17) PROCESS

The PROCESS command is provided to perform computations or statistical manipulations on the various files. Each process request must conform to the following format:

```
PROCESS
process name
   options
```

The request must begin with the PROCESS command in column 1, followed by the name of the process on the following line, also beginning in column 1. If the user wishes to request any options, the options are coded on a subsequent line or lines, beginning in any column except column 1.

The options statement can be of two forms. The first form requires the name of the option followed by the specific value(s):

```
PROCESS
HIGH OCCURRENCES
  THRESHOLD  75
```

The second form allows the user to specify different options for each requested station, by giving the station number before the name of the option:

```
STATION  UI100000   13290450
PROCESS
FLOW DURATION TABLE
  UI100000  CLASS  1  10
  13290450  CLASS  40000  60000
```

The following list contains the names of the NHIMS processes, the elements which can be used as input to the processes, and the options with their default values. Complete descriptions of the processes can be found in Section 4 of this document, in [2] or [10]. Output examples can be found in Appendix 8.3 or [10].

a) DAILY STATISTICS
   input : PRECIPITATION, TEMPERATURE, STREAMFLOW, SNOWFALL, EVAPORATION
   option: ONLY
   default: JANUARY TO ANNUAL

b) MONTHLY STATISTICS
input : PRECIPITATION, TEMPERATURE, STREAMFLOW, SNOWFALL, EVAPORATION
option: ONLY
default: JANUARY TO ANNUAL

c) CORRELATION
input : PRECIPITATION, TEMPERATURE, STREAMFLOW, SNOWFALL, EVAPORATION
option: ONLY
default: JANUARY TO ANNUAL

d) HIGHEST or MAXIMUM
input : PRECIPITATION, TEMPERATURE, STREAMFLOW, SNOWFALL, EVAPORATION
option: ONLY
default: JANUARY TO ANNUAL

e) LOWEST or MINIMUM
input : TEMPERATURE, STREAMFLOW, EVAPORATION
option: ONLY
default: JANUARY TO ANNUAL

f) EXTREME
input : TEMPERATURE
option: ONLY
default: JANUARY TO ANNUAL

g) RANK ORDER
input : PRECIPITATION,TEMPERATURE, STREAMFLOW, SNOWFALL
option: ONLY
default: JANUARY TO ANNUAL

h) HIGH OCCURRENCES
input : TEMPERATURE
option: ONLY
default: JANUARY TO ANNUAL
option: THRESHOLD
default: 90

i) LOW OCCURRENCES
input : TEMPERATURE
option: ONLY
default: JANUARY TO ANNUAL
option: THRESHOLD
default: 32

j) DAILY OCCURRENCES
input : PRECIPITATION, TEMPERATURE, SNOWFALL
option: ONLY

default: JANUARY TO ANNUAL
option: THRESHOLD
default: PRCP = 1, TEMP = 80, SNOW = 1

k) MONTHLY OCCURRENCES
input : PRECIPITATION, SNOWFALL
option: ONLY
default: JANUARY TO ANNUAL
option: THRESHOLD
default: PRCP = 5, SNOW = 10

l) FLOW DURATION TABLE
input : STREAMFLOW
option: CLASS
default: 1, 10, 100, 1000, 10000, 100000, 1000000

m) SUMMARY
input : MONTHLY SUMMARY
option: ONLY
default: JANUARY TO ANNUAL

n) CALENDAR
input : TEMPERATURE
option: ONLY
default: none - option dates must be specified

Not all of the processes described in the system requirements [2] will be in the start-up version of NHIMS; however, Appendix 8.2 gives a strategy for adding processes to the system so that new applications can be included in the future.

3.2 SYSTEM PROGRAM STRUCTURE DIAGRAM

The program structure diagram is a complete diagram of all the modules in the NHIMS system and their interrelationships; it is a detailed version of the system structure diagram in Figure 1. Due to the size of the NHIMS system, the program structure diagram has been divided into nine separate figures (Figures 2 - 10). Whenever a module and all of its subordinates cannot fit onto one page, an asterisk next to the module name denotes that its description continues onto a subsequent page of the diagram.

The descriptions of each of the modules in the program structure diagram can be found in Section 4, corresponding to the numbers given in the diagram itself.

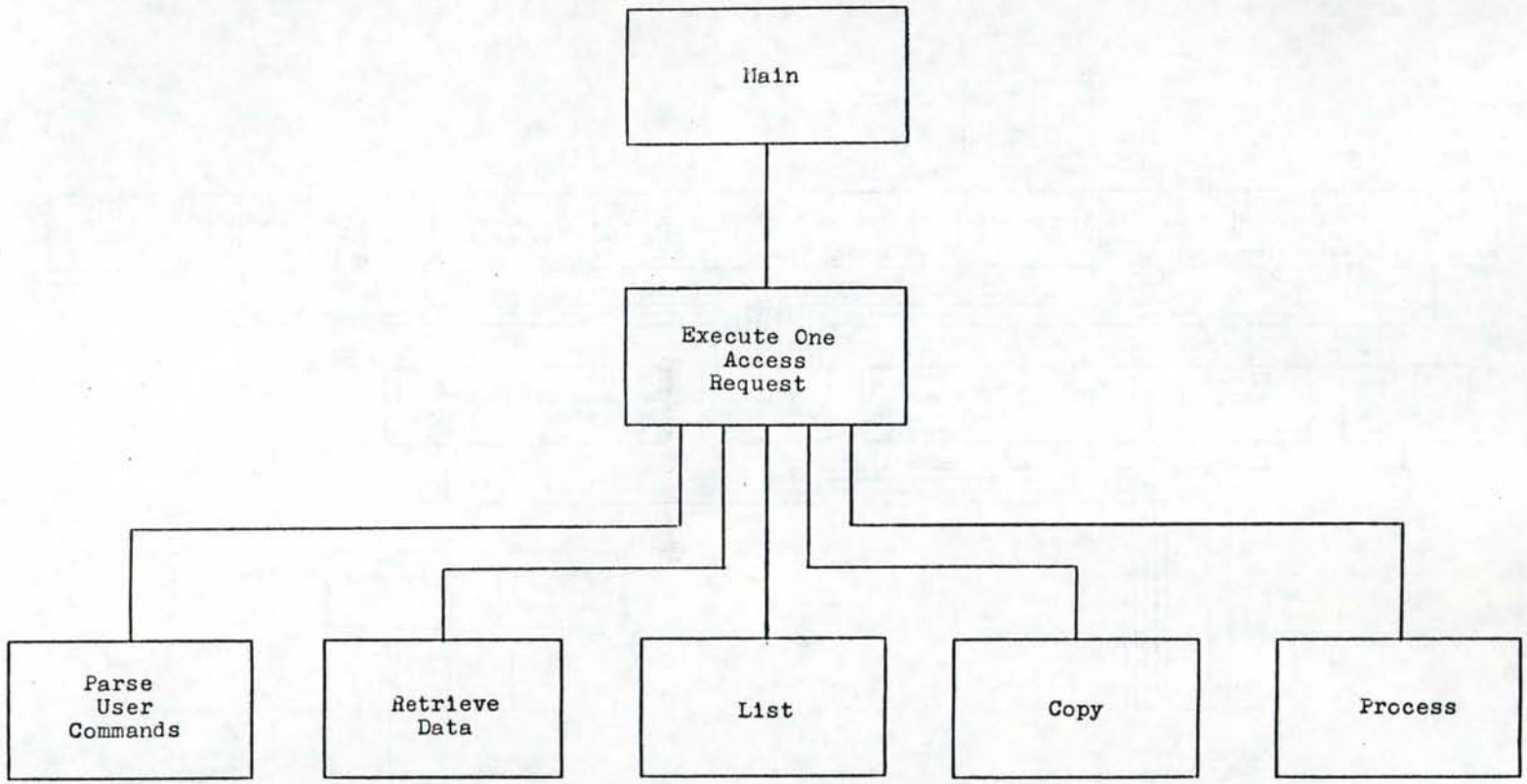

Figure 1. System Structure Diagram.

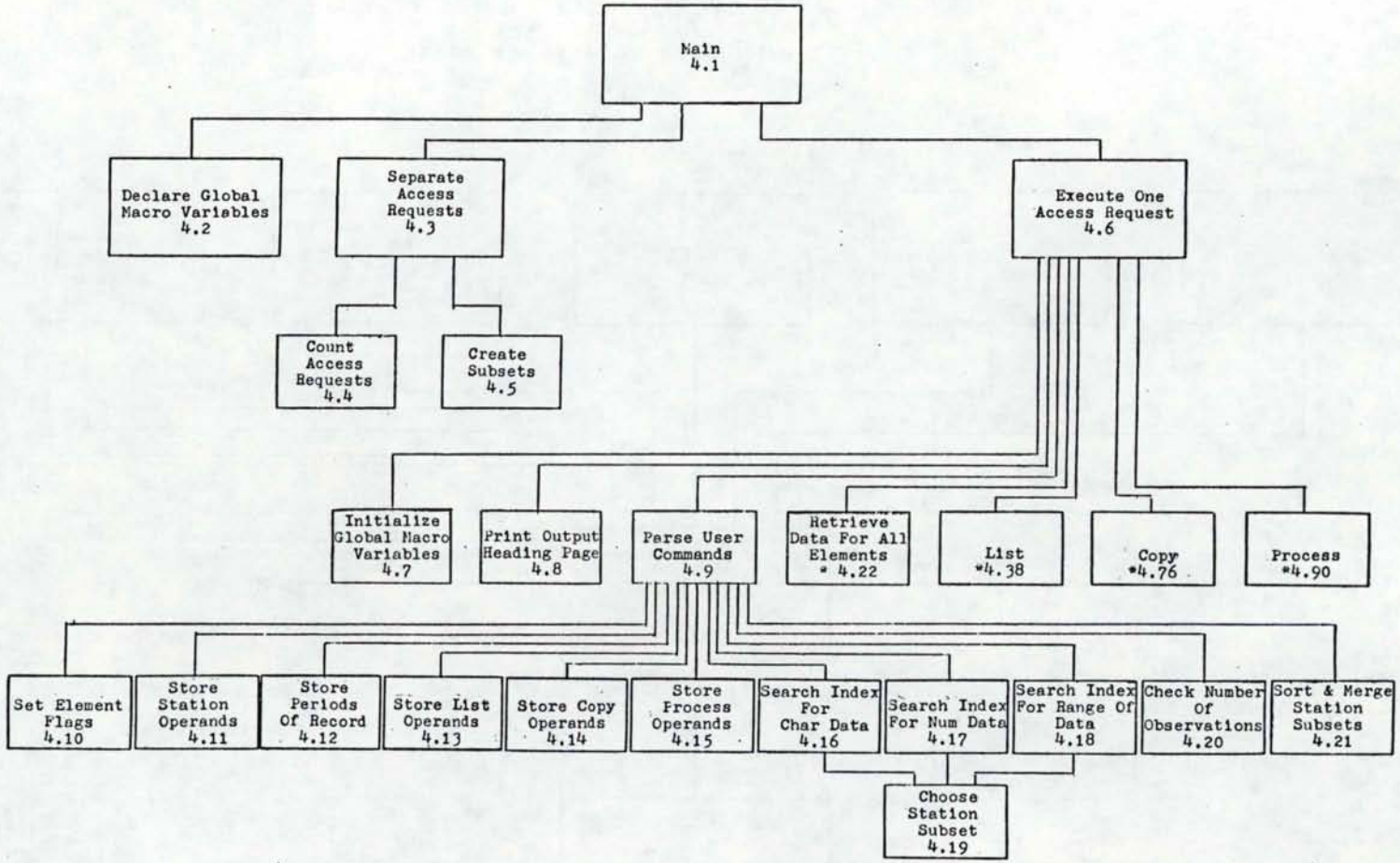

Figure 2. Program Structure Diagram (a).

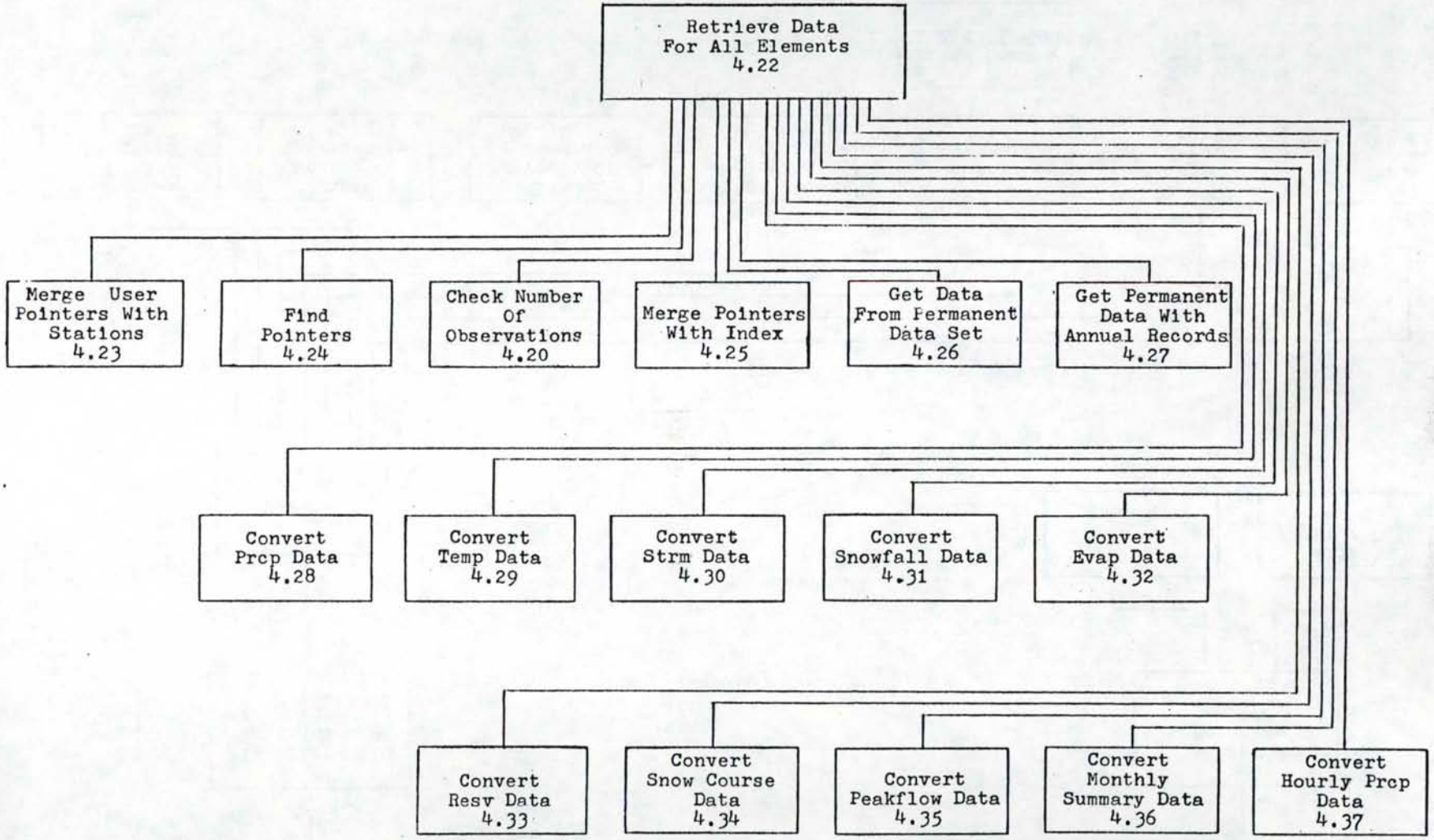

Figure 3. Program Structure Diagram (b).

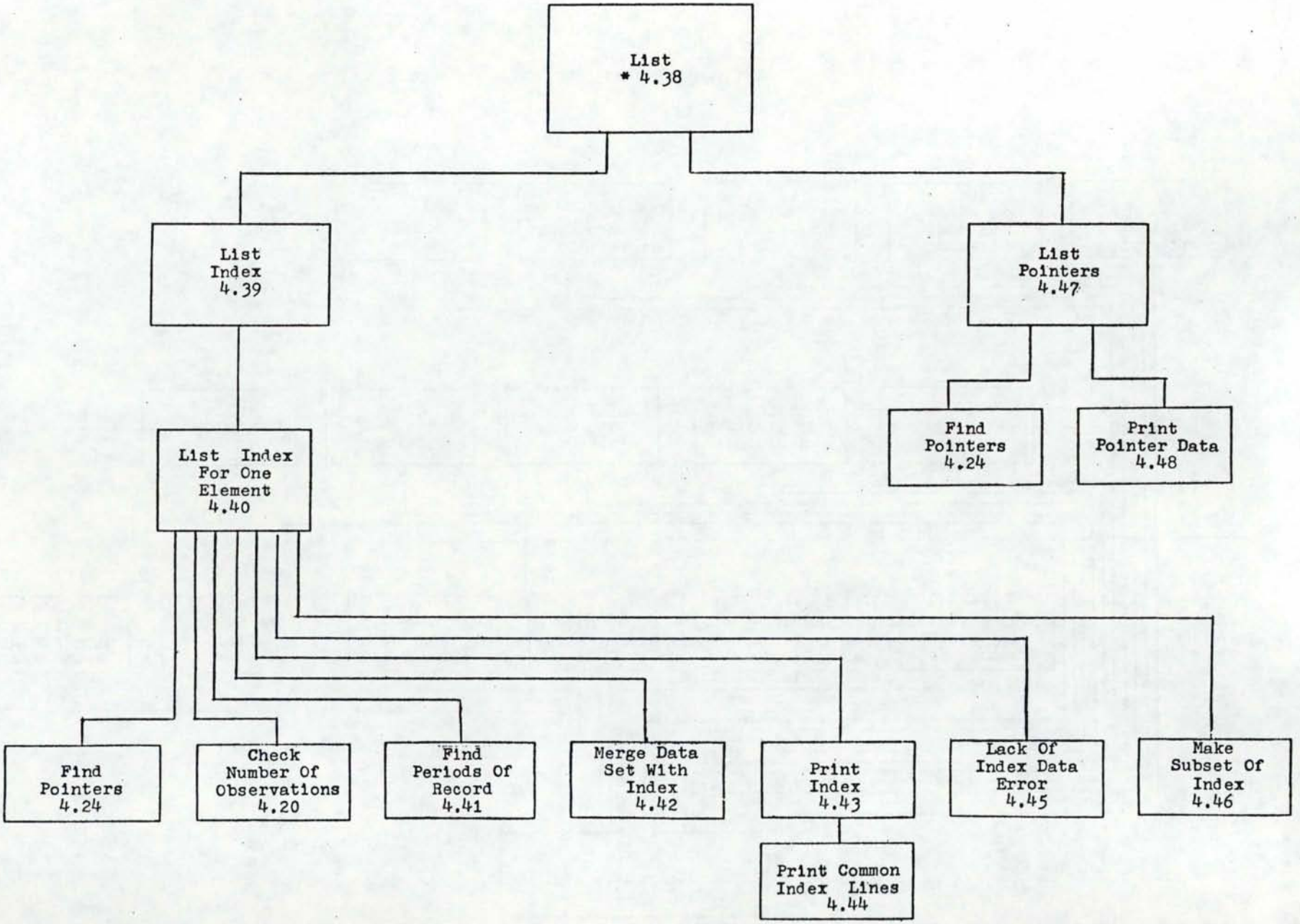

Figure 4. Program Structure Diagram (c).

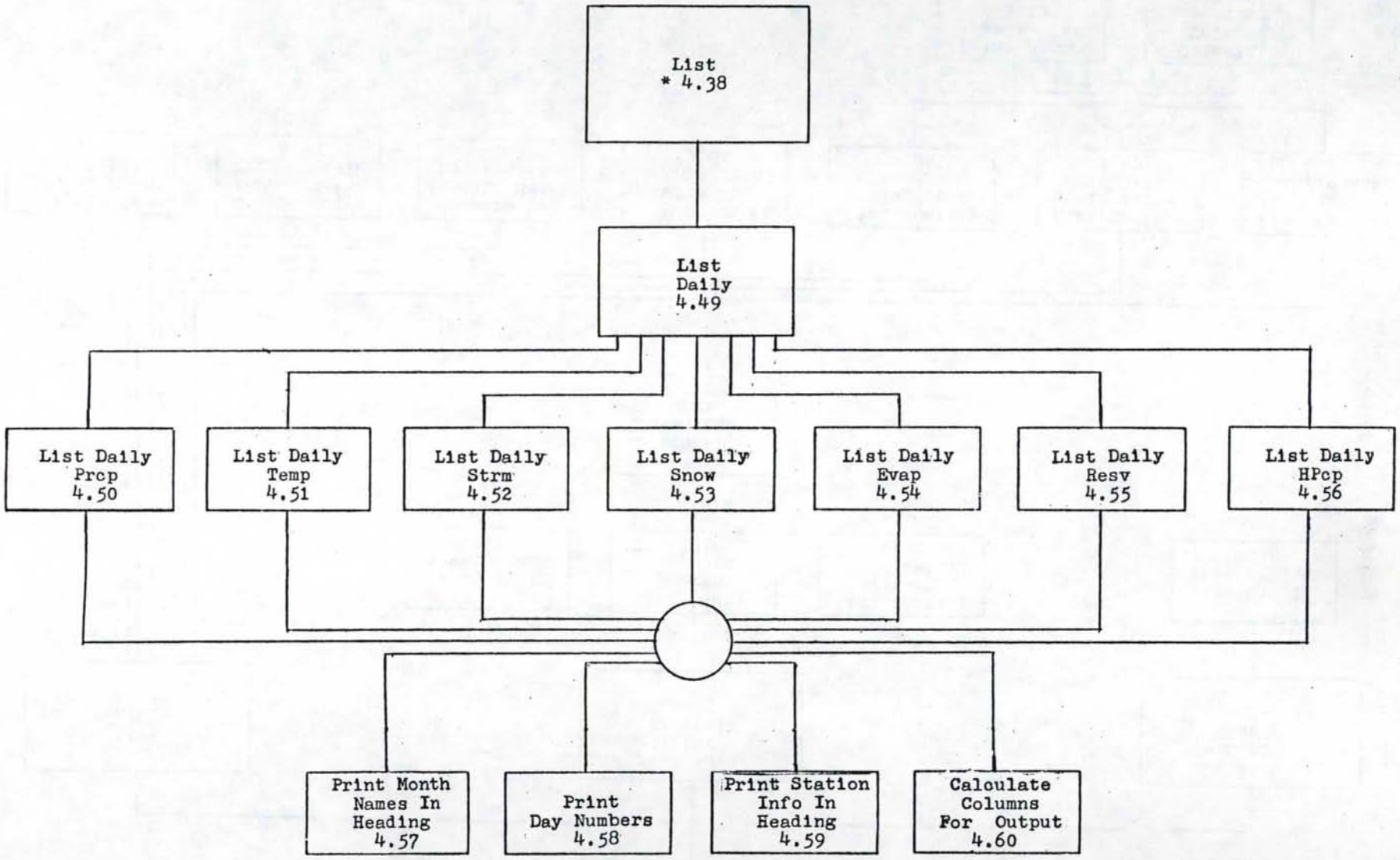

Figure 5. Program Structure Diagram (d).

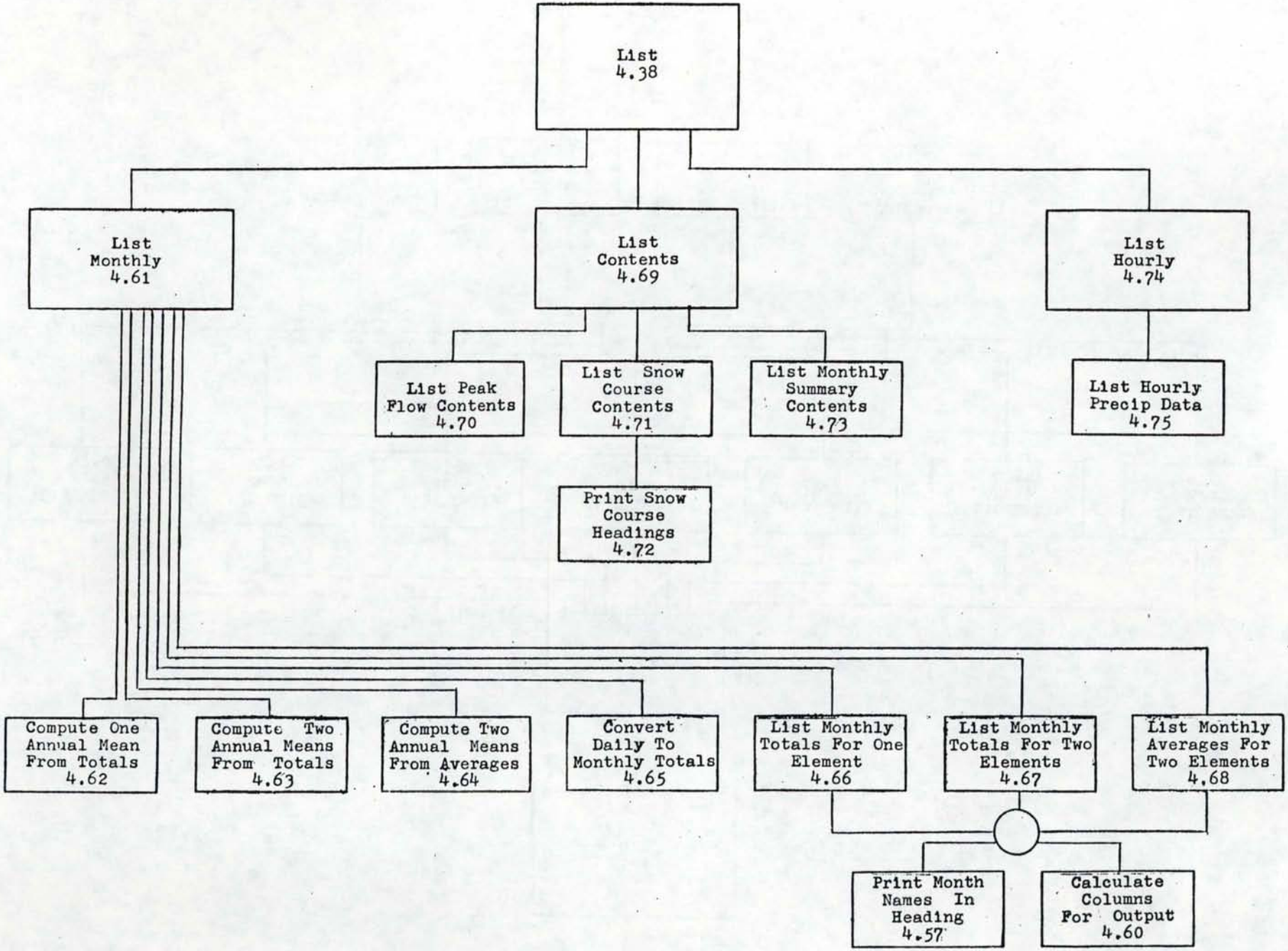

Figure 6. Program Structure Diagram (e).

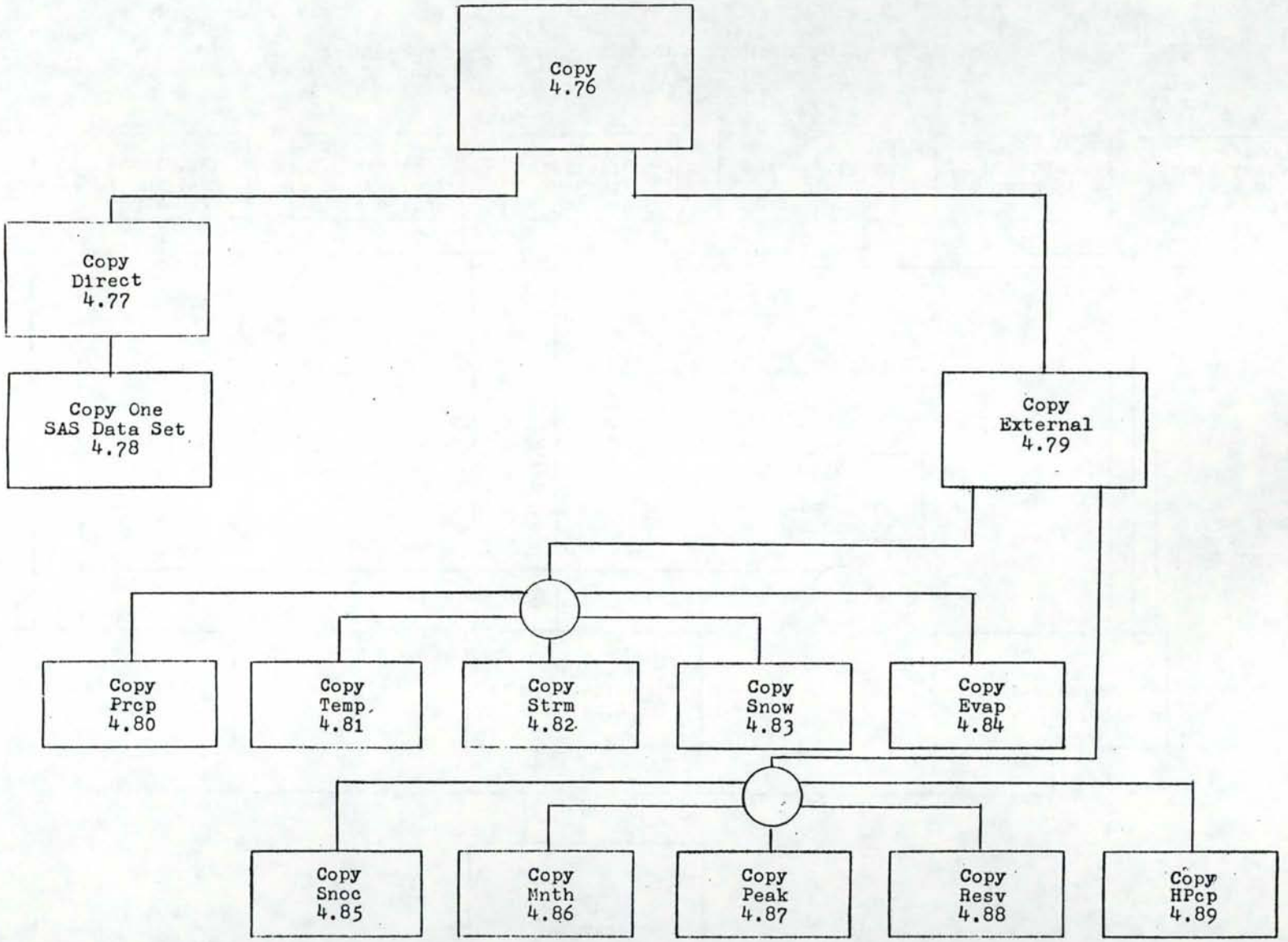

Figure 7. Program Structure Diagram (f).

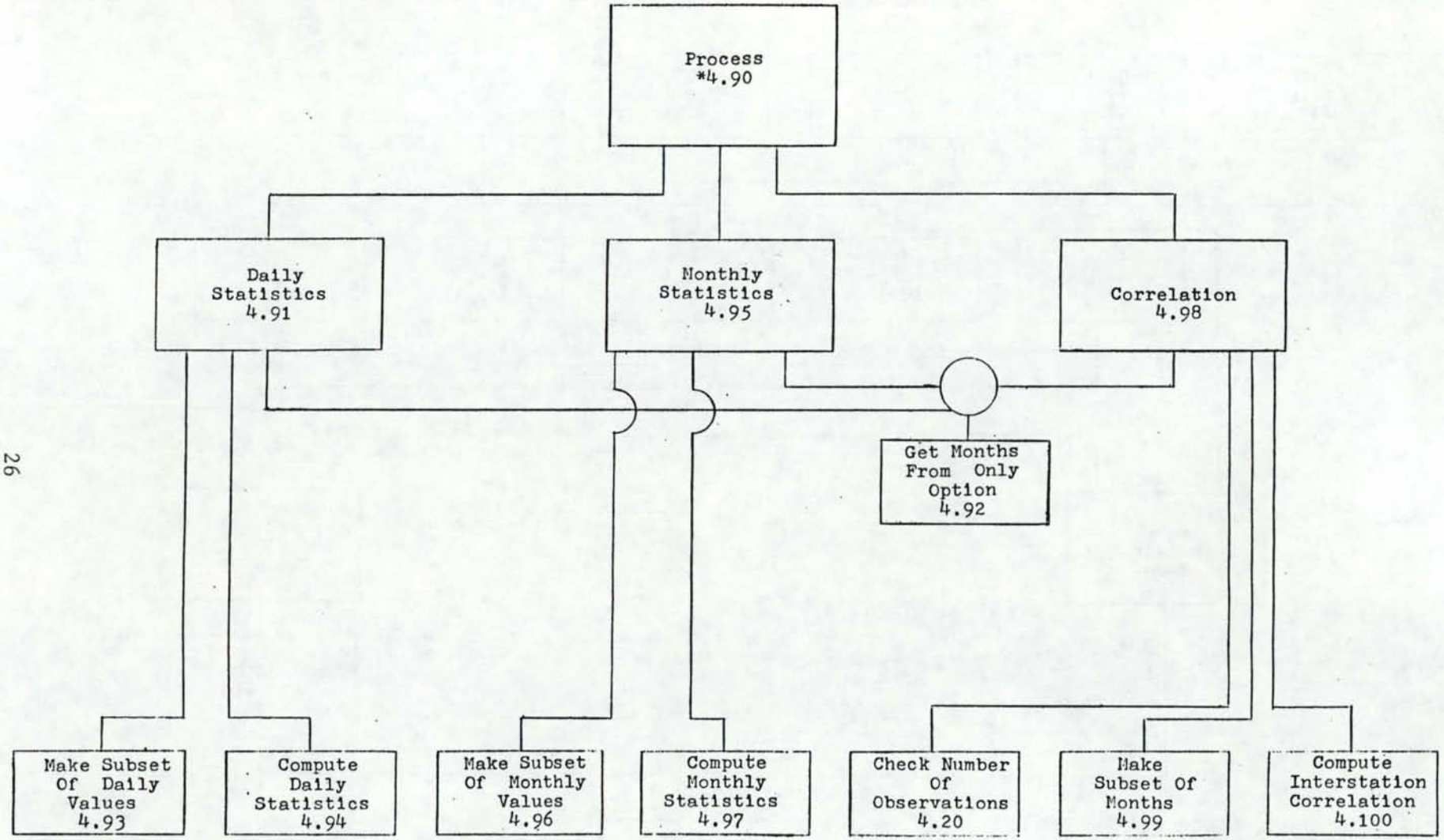

Figure 8. Program Structure Diagram (g).

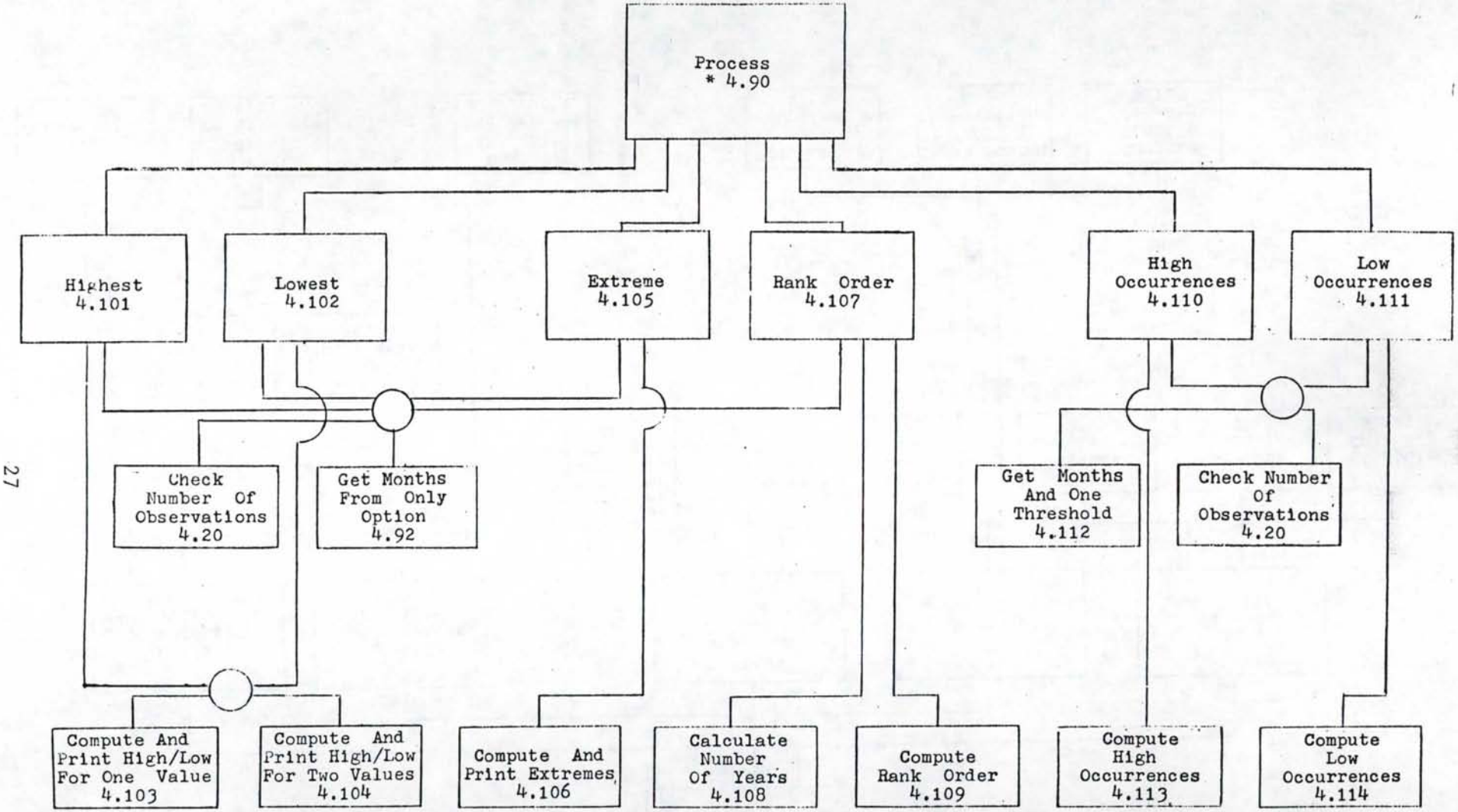

Figure 9. Program Structure Diagram (h).

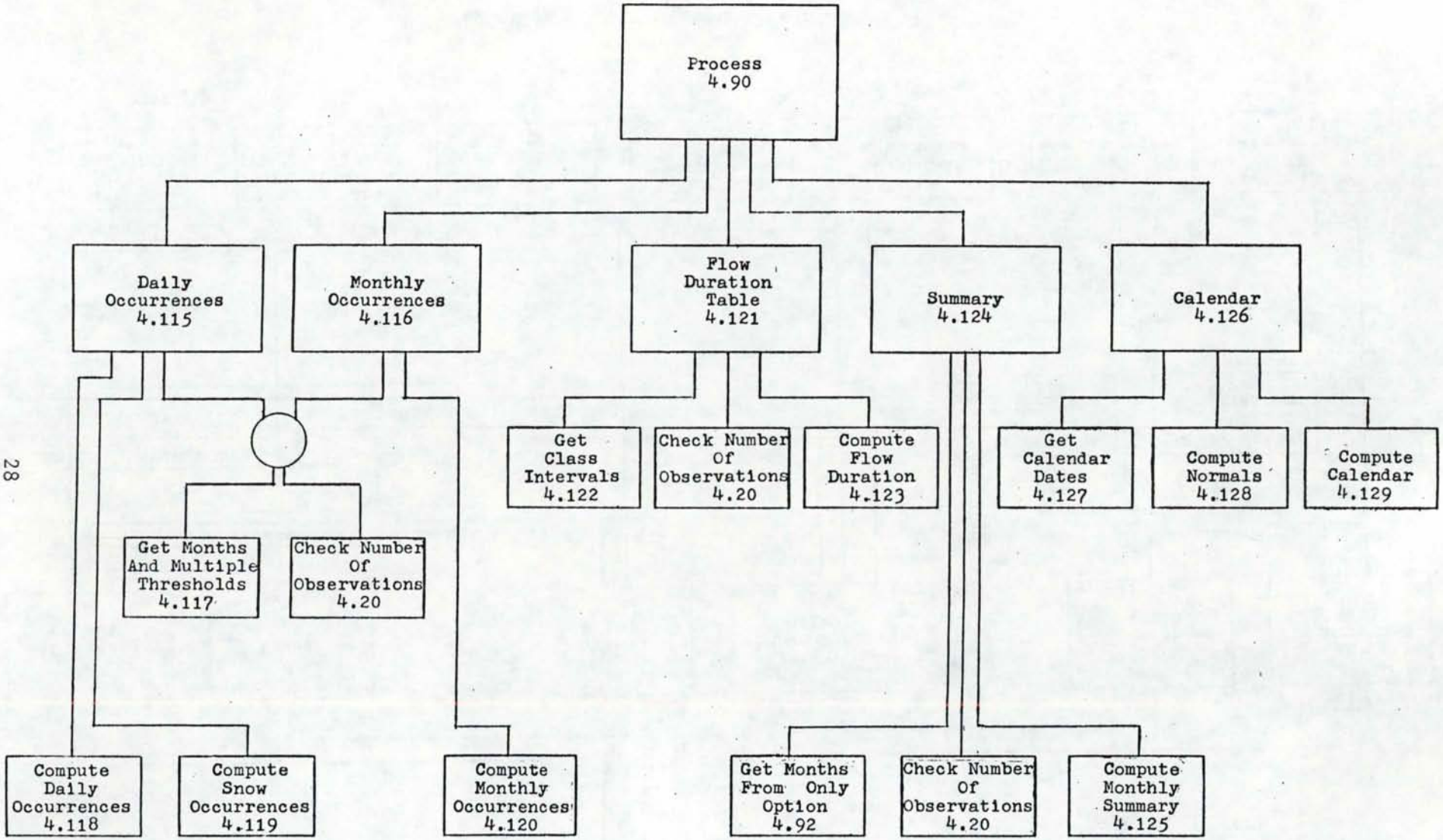

Figure 10. Program Structure Diagram (i).

4.0 MODULE DESCRIPTIONS

The following section describes the individual modules in the NHIMS system. Each module described here is a SAS macro, and the terms 'module' and 'macro' will be used interchangeably. However, before beginning the descriptions, it is important to discuss in more detail the macro language itself, and how it is used.

The macro language within the SAS system acts as a preprocessor, generating text in the form of regular program statements for the compiler [16]. Percent signs, %, signal macro statements, and ampersands, &, identify macro variables to be resolved. Macro statements are processed before other statements, generating SAS code which is then compiled and executed. The macro language allows SAS code to be generated on an iterative or conditional basis, depending on current data values.

A macro is defined by the keywords %MACRO at the beginning and %MEND at the end. For example:

```
%MACRO SAMPLE;
   macro statements
%MEND SAMPLE;
```

A macro call is performed by coding a percent sign and then the name of the macro, which must have been previously defined [17, p.437]. Macros may have either positional or keyword parameters, which act as local macro variables during execution of the macro [16, p. 116]. When calling a macro, the parameter given must be the actual text that will be assigned to the local macro variable; if the name of a variable is given as a parameter in a macro call, that name, and not the value of the variable, is passed to the macro itself.

Macro variables are assigned values either with the %LET statement [17, p. 447]:

```
%LET ELEMENT = PRCP;
```

or with the CALL SYMPUT function [17, p. 458]:

```
CALL SYMPUT('ELEMENT','PRCP');
```

The difference in the above methods is that %LET is performed at compilation time, while CALL SYMPUT is performed at execution time. After execution of the statement:

```
&ELEMENT = 0.5;
```

A variable PRCP would have the value 0.5.

Each of the elements has a 4-character abbreviation which is used extensively throughout the NHIMS system to refer to that element. The abbreviations are:

PRCP : precipitation data	TEMP : temperature data
STRM : streamflow data	SNOW : snowfall data
EVAP : evaporation data	SNOC : snow course data
MNTH : monthly summary data	PEAK : peak flow data
HPCP : hourly precipitation data	RESV : reservoir storage data

Each module description in the following section includes a processing narrative, an interface description, an internal data description, and a design language description.

The processing narrative (section 4.n.1) consists of a brief, textual description of the purpose of the module.

The interface description (section 4.n.2) has five parts:

A) parameter list - the values sent to the macro which provide data for one execution of the macro. These values are macro variables which are local to the macro. There are no output parameters from a SAS macro.

B) global data - includes both global macro variables and temporary SAS data sets which are used or created by the macro and are globally available for the duration of the job. The variable names will be listed here, but are more completely described with all of the global data in Section 5.1. Included with the global data is the NHIMS file which contains the commands submitted by the user.

C) names of other macros called by this macro

D) permanent files used - the permanent SAS data sets accessed by the macro. Permanent data sets can be used only for input, and are described more extensively in Section 5.2.

E) text generated - one or more complete DATA or PROC steps, portions of a DATA or PROC step, or macro statements only. This information is necessary to ensure that the macro generates text in the proper environment. For those macros that generate only a portion of a DATA or PROC step, a list is given of those SAS variables which the macro shares with the DATA or PROC step which calls it.

The internal data description (section 4.n.3) describes all local macro variables, and also regular SAS variables which are used in a DATA step but not stored in a SAS data set for global use.

The design language description (section 4.n.4) contains the SAS-like pseudocode which outlines the module's procedure.

Because the SAS system only allows 8-character variable, data set and macro names, the name given each module is a longer, more descriptive name, with the actual SAS name in parentheses underneath. To maintain compatibility with the version of SAS that runs on CMS, all macro names are limited to 7 characters.

4.1 Description of Module MAIN (MAIN)

4.1.1 Processing Narrative

MAIN is the driver module for the NHIMS system. It is responsible for calling the principal macros which operate the system.

4.1.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:
C_ACCES: counter for the number of access requests
COUNT : control variable for the DO loop

No SAS data sets are created or used.

C) The macros called by this module are:

DECLARE GLOBAL MACRO VARIABLES (4.2 - DGLOBAL)
SEPARATE ACCESS REQUESTS (4.3 - SEPRATE)
EXECUTE ONE ACCESS REQUEST (4.6 - ACCESS)

D) No permanent files are used by this module.

E) The module generates macro statements only.

4.1.3 Internal Data Description

No local data are used in this module.

4.1.4 Design Language Description

```
%DECLARE GLOBAL MACRO VARIABLES
%SEPARATE ACCESS REQUESTS
```

```
%DO COUNT = 1 %TO number of access requests;
    %EXECUTE ONE ACCESS REQUEST
%END;
```

4.2 Description of Module DECLARE GLOBAL MACRO VARIABLES (DGLOBAL)

4.2.1 Processing Narrative

DECLARE GLOBAL MACRO VARIABLES is responsible for using the %GLOBAL statement to declare which macro variables will be globally available for use by the program. The purpose of the declaration is to avoid problems with referencing environments [16, p. 180].

4.2.2 Interface Description

A) There is no parameter list.

B) Global data:
No SAS data sets are created or used.

Macro variables:

```
EL_PRCP: flag for precipitation element
EL_TEMP: flag for temperature element
EL_STRM: flag for streamflow element
EL_SNOW: flag for snowfall element
EL_EVAP: flag for evaporation element
EL_RESV: flag for reservoir storage element
EL_SNOC: flag for snow course element
EL_PEAK: flag for peakflow element
EL_MNTH: flag for monthly element
EL_HPCP: flag for hourly precipitation element
C_ACCES: counter for the number of access requests
COUNT  : control variable for main DO loop
MAIN_DS: identifies need to retreive main data set info
DAILY  : identifies need to retrieve daily data
MONTHLY: identifies need to retrieve monthly data
HOURLY : identifies need to retrieve hourly data
METRIC : identifies request for data in metric units
PERIOD : identifies use of the PERIOD command
SORT   : identifies need to sort station subsets
NOHEAD : identifies need to suppress printing of header page
POINTER: identifies use of user-specified pointers
ELEMENT: identifies use of ELEMENT command
NUM_DS : number of station subsets
NOBS   : number of observations in a data set
LIST   : identifies request for list operation
L_INDEX: identifies request for index listing
```

L_COMP : identifies request for complete file listing
LI_PER : identifies request for period of record with index
LI_ALL : identifies request for list index of all types of stations
LI_NHIM: identifies request for list index of only NHIMS stations
LI_NONM: identifies request for list index of only non-NHIMS stations
L_DAILY: identifies request for daily listing
L_MONTH: identifies request for monthly listing
L_HOUR : identifies request for hourly listing
L_CONT : identifies request for contents listing
L_PTRS : identifies request for listing of pointer file
LM_PART: identifies request for partial monthly listings
L_FIRST: identifies month in which listings begin
COPY : identifies request for copy operation
C_INTGR: identifies request for copy integer
C_REAL : identifies request for copy real
C_DIRCT: identifies request for copy direct
C_80 : identifies request for copy 80-byte records
C_FORM : identifies request for copy formatted
FN : file number of external file for copy output
PROCESS: identifies request for process operation
ANNCORR: identifies request for annual correlations
ANNSTAT: identifies request for annual statistics
ONLYANN: identifies request for only annual processing
DAISTA : identifies request for daily statistics process
STA : alternate name requesting daily statistics process
MONSTA : identifies request for monthly statistics process
COR : identifies request for correlation process
HIG : identifies request for highest process
MAX : alternate name requesting highest process
LOW : identifies request for lowest process
MIN : alternate name requesting lowest process
EXT : identifies request for extreme process
RANORD : identifies request for rank order process
HIGOCC : identifies request for high occurrences process
LOWOCC : identifies request for low occurrences process
DAIOCC : identifies request for daily occurrences process
OCC : alternate name requesting daily occurrences process
MONOCC : identifies request for monthly occurrences process
FLODURTA: identifies request for flow duration table process
SUM : identifies request for summary process
CAL : identifies request for calendar process

C) No macros are called by this module.

D) No permanent files are used by this module.

E) The module generates macro statements only.

4.2.3 Internal Data Description

No local data are used by this module.

4.2.4 Design Language Description

```
%GLOBAL EL_PRCP;
```

Use an identical %GLOBAL statement for each of the macro variables listed in 4.2.2 (B).

4.3 Description of Module SEPARATE ACCESS REQUESTS (SEPRATE)

4.3.1 Processing Narrative

SEPARATE ACCESS REQUESTS initializes the access counter and then calls the modules which read in the user commands and put the individual access requests in separate data sets.

4.3.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:
C_ACCES: counter for the number of access requests

No SAS data sets are created or used.

C) The macros called by this module are:

COUNT ACCESS REQUESTS (4.4 - CNT_REQ)
CREATE SUBSETS (4.5 - SUBSETS)

D) No permanent files are used by this module.

E) The module generates macro statements only.

4.3.3 Internal Data Description

No local data are used in this module.

4.3.4 Design Language Description

```
%LET number of access requests = 0;

%COUNT ACCESS REQUESTS
%CREATE SUBSETS
```

4.4 Description of Module COUNT ACCESS REQUESTS (CNT_REQ)

4.4.1 Processing Narrative

COUNT ACCESS REQUESTS reads the user commands into a SAS data set, counting the number of access requests. All of the user commands are converted to uppercase. If the ACCESS command is not given, an error message is issued and the program is aborted.

4.4.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:
C_ACCES: counter for the number of access requests

SAS data sets:
NHIMS (input) - commands submitted by the user
USRCARDS (output) - with variable: RECORD
PRINT (output) - standard SAS print file

C) No macros are called by this module.

D) No permanent files are used by this module.

E) The module generates two complete DATA steps.

4.4.3 Internal Data Description

No macro variables are used in this module.

SAS variables:
COUNT (num): counter for the number of access requests

4.4.4 Design Language Description

```
DATA USRCARDS (KEEP=RECORD);
  INFILE NHIMS;
  RETAIN COUNT 0;
```

```
    INPUT RECORD $CHAR80.;
    convert RECORD to upper case;
    IF RECORD = 'ACCESS' THEN
         increment COUNT by 1;
         set C_ACCES to the value of COUNT;

DATA _NULL_;
  FILE PRINT;
  IF &C_ACCES = 0 THEN
         issue error message - no ACCESS command given;
         ABORT;
```

4.5 Description of Module CREATE SUBSETS (SUBSETS)

4.5.1 Processing Narrative

CREATE SUBSETS will create a separate data set for each of the access requests, with each data set containing the user commands for one access request. The ACCESS commands are omitted.

4.5.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:
C_ACCES: counter for the number of access requests

SAS data sets:
USRCARDS (input) - with variables:
RECORD
ACCESS1-ACCESSn (output) - with variables:
RECORD

C) No macros are called by this module.

D) No permanent files are used by this module.

E) The module generates one or more complete DATA steps.

4.5.3 Internal Data Description

Macro variables:
C: control variable for the %DO loop

SAS variables:
COUNT (num): counter for the number of access requests

4.5.4 Design Language Description

```
%LOCAL C;
%DO C = 1 %TO &C_ACCES;

    DATA ACCESS&C;

       RETAIN COUNT 0;
       SET USRCARDS;
       IF RECORD = 'ACCESS' THEN
            increment COUNT by 1;
       ELSE
            IF COUNT = proper request number (&C) THEN OUTPUT;
    RUN;
```

4.6 Description of Module EXECUTE ONE ACCESS REQUEST (ACCESS)

4.6.1 Processing Narrative

EXECUTE ONE ACCESS REQUEST calls and controls the macros that execute a single access request.

4.6.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:
MAIN_DS: identifies need to retrieve main data set info
LIST: identifies request for list operation
COPY: identifies request for copy operation
PROCESS: identifies request for process operation

No SAS data sets are created or used.

C) The macros called by this module are:

INITIALIZE GLOBAL MACRO VARIABLES (4.7 - IGLOBAL)
PRINT OUTPUT HEADING PAGE (4.8 - HEADER)
PARSE USER COMMANDS (4.9 - PARSE)
RETRIEVE DATA FOR ALL ELEMENTS (4.22 - RETRIEV)
LIST (4.38 - LISTT)
COPY (4.76 - COPYY)
PROCESS (4.90 - PROCESS)

D) No permanent files are used by this module.

E) The module generates macro statements only.

4.6.3 Internal Data Description

No local data are used in this module.

4.6.4 Design Language Description

```
%INITIALIZE GLOBAL MACRO VARIABLES
%PARSE USER COMMANDS

%IF not need to suppress printing header page %THEN
     %PRINT OUTPUT HEADING PAGE

%IF need to retrieve main data set info (&MAIN_DS) %THEN
    %RETRIEVE DATA FOR ALL ELEMENTS
```

```
%IF request for list operation (&LIST)  %THEN
    %LISTT
%IF request for copy operation (&COPY)  %THEN
    %COPYY
%IF request for process operation (&PROCESS)  %THEN
    %PROCESS
```

4.7 Description of Module INITIALIZE GLOBAL MACRO VARIABLES (IGLOBAL)

4.7.1 Processing Narrative

INITIALIZE GLOBAL MACRO VARIABLES is responsible for assigning initial values to all global macro variables.

4.7.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:

```
EL_PRCP: flag for precipitation element
EL_TEMP: flag for temperature element
EL_STRM: flag for streamflow element
EL_SNOW: flag for snowfall element
EL_EVAP: flag for evaporation element
EL_RESV: flag for reservoir storage element
EL_SNOC: flag for snow course element
EL_PEAK: flag for peakflow element
EL_MNTH: flag for monthly element
EL_HPCP: flag for hourly precipitation element
C_ACCES: counter for the number of access requests
MAIN_DS: identifies need to retreive main data set info
DAILY  : identifies need to retrieve daily data
MONTHLY: identifies need to retrieve monthly data
HOURLY : identifies need to retrieve hourly data
METRIC : identifies request for data in metric units
PERIOD : identifies use of the PERIOD command
SORT   : identifies need to sort station subsets
NOHEAD : identifies need to suppress printing of
         header page
POINTER: identifies use of user-specified pointers
ELEMENT: identifies use of ELEMENT command
NUM_DS : number of station subsets
LIST   : identifies request for list operation
L_INDEX: identifies request for index listing
L_COMP : identifies request for complete file listing
```

LI_PER : identifies request for period of record with index
LI_ALL : identifies request of list index for
all types of stations
LI_NHIM: identifies request of list index for only
NHIMS stations
LI_NONM: identifies request of list index for only
non-NHIMS stations
L_DAILY: identifies request for daily listing
L_MONTH: identifies request for monthly listing
L_HOUR : identifies request for hourly listing
L_CONT : identifies request for contents listing
L_PTRS : identifies request for listing of pointer file
LM_PART: identifies request for partial monthly listings
L_FIRST: identifies month in which listings begin
COPY : identifies request for copy operation
C_INTGR: identifies request for copy integer
C_REAL : identifies request for copy real
C_DIRCT: identifies request for copy direct
C_80 : identifies request for copy 80-byte records
C_FORM : identifies request for copy formatted
FN : file number of external file for copy output
PROCESS: identifies request for process operation
ANNCORR: identifies request for annual correlations
ANNSTAT: identifies request for annual statistics
ONLYANN: identifies request for only annual processing
DAISTA : identifies request for daily statistics process
STA : alternate name requesting daily statistics process
MONSTA : identifies request for monthly statistics process
COR : identifies request for correlation process
HIG : identifies request for highest process
MAX : alternate name requesting highest process
LOW : identifies request for lowest process
MIN : alternate name requesting lowest process
EXT : identifies request for extreme process
RANORD : identifies request for rank order process
HIGOCC : identifies request for high occurrences process
LOWOCC : identifies request for low occurrences process
DAIOCC : identifies request for daily occurrences process
OCC : alternate name requesting daily occurrences process
MONOCC : identifies request for monthly occurrences process
FLODURTA: identifies request for flow duration table process
SUM : identifies request for summary process
CAL : identifies request for calendar process

No SAS data sets are created or used.

C) No macros are called by this module.

D) No permanent files are used by this module.

E) The module generates macro statements only.

4.7.3 Internal Data Description

No local data are used by this module.

4.7.4 Design Language Description

```
%LET EL_PRCP = 0;
```

Each of the global macro variables in 4.7.2 should be initialized to 0 as above, except for the following:

```
%LET SORT    = 1;
%LET L_COMP  = 1;
%LET LI_PER  = 1;
%LET LM_PART = 1;
%LET LI_PER  = 1;
%LET FN      = 1;
```

4.8 Description of Module PRINT OUTPUT HEADING PAGE (HEADER)

4.8.1 Processing Narrative

PRINT OUTPUT HEADING PAGE generates a page of descriptive data on the current access request: the NHIMS logo and version, date and time the job is run, original statements submitted by the user, ownership of NHIMS, and who to contact to get more information.

4.8.2 Interface Description

A) There is no parameter list.

B) Global data:
   Macro variables:
     COUNT: control variable identifying current access request

   SAS data sets:
     ACCESSn (input): user commands for current access request,
                      with variable: RECORD
     PRINT (output): standard SAS print file

C) No macros are called by this module.

D) No permanent files are used by this module.

E) The module generates a complete _NULL_ DATA step.

4.8.3 Internal Data Description

No local macro variables are used by this module.

SAS variables:
TODAYS (date): date when job is run
RIGHTNOW (date): time when job is run

4.8.4 Design Language Description

```
DATA _NULL_;
  PUT 5 matrices (7x7 each) of asterisks, spelling NHIMS;
  PUT 'NORTHWEST HYDROLOGIC INFORMATION MANAGEMENT SYSTEM';
  TODAYS = DATE();
  PUT 'VERSION OF MAY, 1986'  '    RUN ON  '  TODAYS WORDDATE12.;
  RIGHTNOW = TIME();
  PUT '                                AT  '  RIGHTNOW TIME8.;
  PUT 'USER COMMANDS RECEIVED WERE:';
      DO UNTIL end of file of user commands;
         SET ACCESS&COUNT  END = EOF;   /* user commands  */
         PUT  RECORD;
      END;
  PUT at page bottom: 'UNIVERSITY OF IDAHO VERSION OF MAY, 1987';
  PUT 'FOR MORE INFORMATION, CONTACT AG ENGINEERING DEPARTMENT. ';
```

4.9 Description of Module PARSE USER COMMANDS (PARSE)

4.9.1 Processing Narrative

PARSE USER COMMANDS reads the user commands for one access request, determining what data are to be retrieved and what operations are to be performed. Global macro variables are set and several data sets are created containing information for subsequent processing. The STATION1-STATION5 data sets are used to store station numbers that correspond to different AND commands; if no ANDs are used, all station numbers will be stored in STATION1.

4.9.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:

```
POINTER: identifies user-given pointer values
METRIC : identifies request for metric option
COUNT  : control variable identifying current access request
SORT   : identifies need for sorting station numbers
MAIN_DS: identifies need to retrieve main data info
NUM_DS : identifies how many station data sets are created
L_COMP : identifies request for complete file listing
NOBS   : number of observations in a data set
ELEMENT: identifies use of element command by user
NOHEAD : identifies need to suppress printing
         of header page
```

SAS data sets:

```
ACCESSn (input): user commands for current access request,
                 with variables:   RECORD
POINTERS (output): with variables:
   FOBS_PTR  LOBS_PTR
PERIODS  (output): with variables:
   BEGDATE  ENDDATE
PROCESS  (output): with variables:
   PRC_NAME  PARMETER
STATION1-STATION5 (output): each with variable:
   STATION
PRINT (output): standard SAS print file
```

C) The macros called by this module are:

```
SET ELEMENT FLAGS (4.10 - ELEMENT)
STORE STATION OPERANDS (4.11 - STATION)
STORE PERIODS OF RECORD (4.12 - PERIOD)
STORE LIST OPERANDS (4.13 - ST_LIST)
STORE COPY OPERANDS (4.14 - ST_COPY)
STORE PROCESS OPERANDS (4.15 - ST_PROC)
SEARCH INDEX FOR CHARACTER DATA (4.16 - SI_CHAR)
SEARCH INDEX FOR NUMERIC DATA (4.17 - SI_NUM)
SEARCH INDEX FOR PARTIAL DATA (4.17A - SI_PART)
SEARCH INDEX FOR RANGE OF DATA (4.18 - SI_RNGE)
CHECK NUMBER OF OBSERVATIONS (4.20 - NUM_OBS)
SORT AND MERGE STATION SUBSETS (4.21 - SRT_MRG)
```

D) No permanent files are used by this module.

E) The module generates several complete DATA steps.

4.9.3 Internal Data Description

No local macro variables are used by this module.

SAS variables:

```
AND_CNTR (num): counter for number of AND commands
KEYWORD (char): user-specified NHIMS command
WORD     (num): current word in record used by SCAN function
TOGGLE   (num): identifies when to write to PROCESS data set
COLUMN1 (char): value in first column of user command line
OPERAND (char): value of operand in user command
EOF  (boolean): identifies last record from input data set
```

4.9.4 Design Language Description

```
DATA POINTERS(KEEP=FOBS_PTR LOBS_PTR)
     PERIODS (KEEP=BEGDATE ENDDATE)
     PROCESS (KEEP=PRC_NAME PARMETER )
     STATION1(KEEP=STATION)
     STATION2(KEEP=STATION)
     STATION3(KEEP=STATION)
     STATION4(KEEP=STATION)
     STATION5(KEEP=STATION);

   FILE  PRINT;  /* write any error messages to print file */
   RETAIN AND_CNTR 0 KEYWORD $ ;
   LENGTH OPERAND $ 30  KEYWORD $ 30;

   SET ACCESS&COUNT END=EOF;

   WORD = 1;
   COLUMN1 = SUBSTR(RECORD,1,1);
   IF first column is not a blank AND keyword is not 'PROCESS'
      THEN   /* read the next keyword  */
         KEYWORD = SCAN(RECORD,WORD,' ');
         WORD = WORD + 1;
      END;

   SELECT (KEYWORD); /* call macro matching value of keyword */
      WHEN ('ELEMENT')
          DO;
             %SET ELEMENT FLAGS
          END;

      WHEN ('STATION')
          DO;
             %STORE STATION OPERANDS
          END;
```

```
WHEN ('PERIOD')
    DO;
       %STORE PERIODS OF RECORD
    END;
WHEN ('LIST')
    DO;
       %STORE LIST OPERANDS
    END;
WHEN ('COPY')
    DO;
       %STORE COPY OPERANDS
    END;
WHEN ('PROCESS')
    DO;
       %STORE PROCESS OPERANDS
    END;
WHEN ('POINTERS')
    DO;
       set MAIN_DS to 1;
       set L_COMP  to 0;
       set POINTER to 1;
       FOBS_PTR = SCAN(RECORD, WORD, ' ');
       WORD = WORD + 2;
       LOBS_PTR = SCAN(RECORD, WORD, ' ');
       OUTPUT POINTERS;
    END;
WHEN ('COUNTY')
    DO;
       set L_COMP to 0;
       %SEARCH INDEX FOR CHARACTER DATA(COUNTY)
    END;
WHEN ('BASIN')
    DO;
       set L_COMP to 0;
       %SEARCH INDEX FOR PARTIAL DATA(HUCODE)
    END;
WHEN ('HUCODE')
    DO;
       set L_COMP to 0;
       %SEARCH INDEX FOR PARTIAL DATA(HUCODE)
    END;
WHEN ('REGION')
    DO;
       set L_COMP to 0;
       %SEARCH INDEX FOR NUMERIC DATA(DIVISION)
    END;
```

```
      WHEN ('DIVISION')
          DO;
             set L_COMP to 0;
             %SEARCH INDEX FOR NUMERIC DATA(DIVISION)
          END;
      WHEN ('ELEVATION')
          DO;
             set L_COMP to 0;
             %SEARCH INDEX FOR RANGE OF DATA(ELEV)
          END;
      WHEN ('AREA')
          DO;
             set L_COMP to 0;
             %SEARCH INDEX FOR RANGE OF DATA(AREA)
          END;
      WHEN ('METRIC')
          DO;
             set METRIC to 1;
             set L_COMP to 0;
          END;
      WHEN ('NOHEADER')
          DO;
             set NOHEAD to 1;
          END;
      WHEN ('SORTED')
          DO;
             set SORT to 0;
          END;
      WHEN ('AND')
          DO;
             increment and_cntr by 1;
             set L_COMP to 0;
          END;

      OTHERWISE  issue error message that an illegal command
                 KEYWORD was given.

set NUM_DS to the value of AND_CNTR + 1;
RUN;


%IF &ELEMENT = 0 %THEN
    DATA _NULL_;
       FILE PRINT;
       issue error message that no elements were specified.
       ABORT;
    RUN;
%END;
```

```
%CHECK NUMBER OF OBSERVATIONS (STATION1)


DATA _NULL_; /* check for errors in user input */
 FILE PRINT;

 %IF no observations in STATION1 data set (&NOBS = 0)
     AND NOT request for a complete file listing (NOT &L_COMP)
 %THEN
     issue error message that either requested locations cannot
     be found in the index - check location commands  OR  the
     request for processing the entire element file is denied.
     ABORT;

 %IF the number of station subsets equals 1 (&NUM_DS = 1) AND
     there is exactly one observation in that subset (&NOBS= 1)
 %THEN
     set SORT to 0;
 RUN;

%IF NOT a request for a complete file listing %THEN
    %SORT AND MERGE STATION SUBSETS(&NUM_DS)
```

4.10 Description of Module SET ELEMENT FLAGS (ELEMENT)

4.10.1 Processing Narrative

SET ELEMENT FLAGS reads the operands after the keyword ELEMENT, and sets the appropriate flags for those operands. Some simple checks are performed to account for misspelling of the element names by the user. A flag is set to show that the user correctly requested one or more elements.

4.10.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:

- ELEMENT: identifies use of element command by user
- EL_PRCP: identifies request for precipitation data
- EL_TEMP: identifies request for temperature data
- EL_STRM: identifies request for streamflow data
- EL_SNOW: identifies request for snowfall data
- EL_EVAP: identifies request for evaporation data
- EL_PEAK: identifies request for peakflow data
- EL_SNOC: identifies request for snow course data

EL_MNTH: identifies request for data from monthly file
EL_RESV: identifies request for reservoir storage data
EL_HPCP: identifies request for hourly precipitation data

No SAS data sets are created or used.

C) No macros are called by this module.

D) No permanent files are used by this module.

E) The module generates only a portion of a single DATA step. SAS variables shared with the calling module are:

WORD (num): current word used by SCAN function
OPERAND (char): value of operand in user command
RECORD (char): user command from input data set

4.10.3 Internal Data Description

No local macro variables are used by this module.

SAS variables:
FIRST4 (char): value of first 4 characters of operand
NEXTWORD (num): position of next word in record
NEXT_OP (char): next operand in user command

4.10.4 Design Language Description

```
LENGTH FIRST4 $ 4  NEXT_OP $ 10;
OPERAND = SCAN(RECORD, WORD, ' ');

DO WHILE(OPERAND is not equal to a blank);
   add 1 to ELEMENT ;
   increment WORD by 1;

   SELECT (OPERAND);
      WHEN ('PRECIPITATION')
         set EL_PRCP to 1;
      WHEN ('RAINFALL')
         set EL_PRCP to 1;
      WHEN ('TEMPERATURE')
         set EL_TEMP to 1;
      WHEN ('STREAMFLOW')
         set EL_STRM to 1;
      WHEN ('SNOWFALL')
         set EL_SNOW to 1;
      WHEN ('EVAPORATION')
         set EL_EVAP to 1;
      WHEN ('PEAKFLOW')
```

```
   set EL_PEAK to 1;
WHEN ('SNOWCOURSE')
   set EL_SNOC to 1;
WHEN ('COURSE')
   set EL_SNOC to 1;
WHEN ('MONTHLY')
   set EL_MNTH to 1;
WHEN ('STORAGE')
   set EL_RESV to 1;
WHEN ('HOURRAIN')
   set EL_HPCP to 1;
WHEN ('HOURPRCP')
   set EL_HPCP to 1;

OTHERWISE
   DO;  /* check for error in element name */
      FIRST4 = SUBSTR(OPERAND, 1, 4);
      NEXTWORD = WORD + 1;
      NEXT_OP = SCAN(RECORD,NEXTWORD,' ');

      SELECT (FIRST4);

         WHEN ('PREC')
            set EL_PRCP to 1;
         WHEN ('RAIN')
            set EL_PRCP to 1;
         WHEN ('TEMP')
            set EL_TEMP to 1;
         WHEN ('STRE')
            set EL_STRM to 1;
         WHEN ('SNOW')
            DO;
              IF NEXT_OP = 'FALL'  THEN
                 set EL_SNOW to 1;
              ELSE IF NEXT_OP = 'COURSE' THEN
                 set EL_SNOC to 1;
            END;
         WHEN ('EVAP')
            set EL_EVAP to 1;
         WHEN ('PEAK')
            set EL_PEAK to 1;
         WHEN ('COUR')
            set EL_SNOC to 1;
         WHEN ('MONT')
            set EL_MNTH to 1;
         WHEN ('STOR')
            set EL_RESV to 1;
         WHEN ('HOUR')
            set EL_HPCP to 1;
```

```
                OTHERWISE
                   DO;
                     subtract 1 from ELEMENT ;
                     issue error message about invalid element
                         called OPERAND;
                   END;
             END; /* select first4 */

          END;  /* check for error in element name */
       END; /* select operand */

       OPERAND = SCAN(RECORD, WORD, ' ');
    END; /* while operand is not equal to a blank */
```

4.11 Description of Module STORE STATION OPERANDS (STATION)

4.11.1 Processing Narrative

STORE STATION OPERANDS takes all station numbers (operands) listed in a STATION command, and writes them to a data set.

4.11.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:
L_COMP: identifies request for a complete file listing

SAS data sets:
STATION1 (output): with variable STATION

C) No macros are called by this module.

D) No permanent files are used by this module.

E) The module generates only a portion of a single DATA step. SAS variables shared with the calling module are:

```
WORD     (num): current word used by SCAN function
OPERAND (char): value of operand in user command
RECORD  (char): user command from input data set
```

4.11.3 Internal Data Description

No local data are used by this module.

4.11.4 Design Language Description

```
set L_COMP to 0;
OPERAND = SCAN(RECORD, WORD, ' ');

DO WHILE(OPERAND is not equal to a blank);
   increment WORD by 1;
   STATION = OPERAND;
   OUTPUT STATION1;
   OPERAND = SCAN(RECORD, WORD, ' ');
END;
```

4.12 Description of Module STORE PERIODS OF RECORD (PERIOD)

4.12.1 Processing Narrative

STORE PERIODS OF RECORD reads the operands for the PERIOD command, storing the beginning and ending dates of each requested period of record in a data set. A counter is kept of how many different periods are requested, which is stored in a global macro variable.

4.12.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:
- L_COMP: identifies request for a complete file listing
- PERIOD: the number of requested periods of record

SAS data sets:
- PERIODS (output): with variables:
  - BEGDATE ENDDATE

C) No macros are called by this module.

D) No permanent files are used by this module.

E) The module generates only a portion of a single DATA step. SAS variables shared with the calling module are:

- WORD (num): current word used by SCAN function
- OPERAND (char): value of operand in user command
- RECORD (char): user command from input data set

4.12.3 Internal Data Description

No local macro variables are used by this module.

SAS variables:
- PER_CNTR (num): the number of requested periods of record
- BEGMONTH(char): beginning month of period of record
- BEGYEAR (char): beginning year of period of record
- ENDMONTH(char): ending month of period of record
- ENDYEAR (char): ending year of period of record
- POS (num): used to identify position of a '/' in the operand

4.12.4 Design Language Description

```
RETAIN PER_CNTR 0;
set L_COMP to 0;
OPERAND = SCAN(RECORD, WORD, ' ');

DO WHILE(OPERAND is not equal to a blank);
   increment WORD by 2;

   POS = INDEX(OPERAND,'/');
   IF POS NE 0 THEN /* period command has both month and year */

      /* find beginning date of requested period */
      BEGMONTH = SCAN(OPERAND, 1, '/');
      BEGYEAR  = SCAN(OPERAND, 2, '/');
      BEGDATE  = (BEGYEAR * 100) + BEGMONTH;

      /* find ending date of requested period */
      OPERAND  = SCAN(RECORD, WORD, ' ');
      ENDMONTH = SCAN(OPERAND, 1, '/');
      ENDYEAR  = SCAN(OPERAND, 2, '/');
      ENDDATE  = (ENDYEAR * 100) + ENDMONTH;
   END;  /* both month and year */

   ELSE DO;  /* period command has only years specified */
      BEGDATE = OPERAND * 100;
      OPERAND = SCAN(RECORD,WORD,' ');
      ENDDATE = (OPERAND * 100) + 12;
   END;  /* only years specified */

   OUTPUT PERIODS;
   increment PER_CNTR by 1;
   increment WORD by 1;
   OPERAND = SCAN(RECORD, WORD, ' ');
END;  /* while operand is not a blank */

set PERIOD to the value of PER_CNTR;
```

4.13 Description of Module STORE LIST OPERANDS (ST_LIST)

4.13.1 Processing Narrative

STORE LIST OPERANDS reads the operands for the LIST command and sets the corresponding global macro variables.

4.13.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:

```
LIST   : identifies request for list operation
MAIN_DS: identifies need to retrieve main data set info
L_DAILY: identifies request for daily listing
DAILY  : identifies need to retrieve daily data
L_MONTH: identifies request for monthly listing
MONTHLY: identifies need to retrieve monthly data
HOURLY : identifies need to retrieve hourly data
L_INDEX: identifies request for index listing
L_COMP : identifies request for a complete file listing
LI_PER : identifies request for listing of periods of record
LI_ALL : identifies request for index listing of both NHIMS
         and non-NHIMS stations
LI_NHIM: identifies request for index listing of NHIMS
         stations only
LI_NONM: identifies request for index listing of non-NHIMS
         stations only
L_CONT : identifies request for contents listing
L_HOUR : identifies request for hourly listing
L_PTRS : identifies request for listing of pointer file
L_FIRST: month in which to begin listing output
LM_PART: identifies request for partial monthly listing
```

No SAS data sets are created or used.

C) No macros are called by this module.

D) No permanent files are used by this module.

E) The module generates only a portion of a single DATA step. SAS variables shared with the calling module are:

```
WORD     (num): current word used by SCAN function
OPERAND (char): value of operand in user command
RECORD  (char): user command from input data set
```

4.13.3 Internal Data Description

No local data are used by this module.

4.13.4 Design Language Description

```
set LIST to 1;
OPERAND = SCAN(RECORD, WORD, ' ');

DO WHILE(OPERAND is not equal to a blank);
   increment WORD by 1;

   SELECT (OPERAND);

      WHEN ('DAILY')
         DO;
           set DAILY to 1;
           set L_DAILY to 1;
           set MAIN_DS to 1;
           set L_COMP  to 0;
         END;
      WHEN ('MONTHLY')
         DO;
           set MONTHLY to 1;
           set L_MONTH to 1;
           set MAIN_DS to 1;
           set L_COMP  to 0;
         END;
      WHEN ('POINTERS')
         set L_PTRS  to 1;
      WHEN ('INDEX')
         set L_INDEX to 1;
      WHEN ('ALL')
         DO;
           set LI_PER  to 0;
           set LI_ALL  to 1;
      WHEN ('NHIMS')
         DO;
           set LI_PER  to 0;
           set LI_NHIM to 1;
         END;
      WHEN ('NONHIMS')
         DO;
           set LI_PER  to 0;
           set LI_NONM to 1;
         END;
```

```
        WHEN ('CONTENTS')
           DO;
             set L_CONT  to 1;
             set MAIN_DS to 1;
             set L_COMP  to 0;
           END;
        WHEN ('AVERAGE')
           set LM_PART to 0;
        WHEN ('HOURLY')
           DO;
             set HOURLY  to 1;
             set L_HOUR  to 1;
             set MAIN_DS to 1;
             set L_COMP  to 0;
           END;
        OTHERWISE
           DO;
             IF the value of OPERAND is between 1 and 12 THEN
                set L_FIRST to the value of OPERAND;
             ELSE issue error message that user gave the illegal
                  LIST operand OPERAND;
           END;
     END;  /* select operand */

     OPERAND = SCAN(RECORD, WORD, ' ');
  END;  /* while operand is not equal to a blank */
```

4.14 Description of Module STORE COPY OPERANDS (ST_COPY)

4.14.1 Processing Narrative

STORE COPY OPERANDS reads the operands for the COPY command and sets the corresponding global macro variables.

4.14.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:

- COPY : identifies request for copy operation
- DAILY : identifies need to retrieve daily data
- HOURLY : identifies need to retrieve hourly data
- L_COMP : identifies request for a complete file listing
- MAIN_DS: identifies need to retrieve main data set info

```
   C_REAL : identifies request for real output
   C_INTGR: identifies request for integer output
   C_DIRCT: identifies request for SAS data set output
   C_FORM : identifies request for formatted output
   C_80   : identifies request for 80-column formatted output
```

No SAS data sets are created or used.

C) No macros are called by this module.

D) No permanent files are used by this module.

E) The module generates only a portion of a single DATA step.
SAS variables shared with the calling module are:

```
   WORD     (num): current word used by SCAN function
   OPERAND (char): value of operand in user command
   RECORD  (char): user command from input data set
```

4.14.3 Internal Data Description

No local data are used by this module.

4.14.4 Design Language Description

```
set COPY to 1;
set MAIN_DS to 1;
set L_COMP  to 0;
set DAILY to 1;
set HOURLY to 1;

OPERAND = SCAN(RECORD,WORD,' ');
DO WHILE(OPERAND is not equal to a blank);
   increment WORD by 1;

   SELECT (OPERAND);

      WHEN ('REAL')
         set C_REAL to 1;
      WHEN ('INTEGER')
         set C_INTGR to 1;
      WHEN ('DIRECT')
         set C_DIRCT to 1;
      WHEN ('FORMATTED')
         set C_FORM to 1;
      WHEN ('80')
         set C_80 to 1;
      OTHERWISE ;
   END;
END;
```

4.15 Description of Module STORE PROCESS OPERANDS (ST_PROC)

4.15.1 Processing Narrative

STORE PROCESS OPERANDS reads the process commands and operands. A global macro variable is set for each process found; the name of the macro variable is created by taking the first 3 letters of each of the first two words in the name of the process. If the process name has 3 words, the first two letters of the third word are also used. (Example: the process FLOW DURATION TABLE has a corresponding macro variable named FLODURTA, while the process CALENDAR has a corresponding macro variable named CAL.) The names of the processes and the matching parameters are stored in a SAS data set for subsequent use, and various other global macro variables are set.

4.15.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:

```
PROCESS: identifies request for process operation
MAIN_DS: identifies need to retrieve main data set info
DAILY  : identifies need to retrieve daily data
MONTHLY: identifies need to retrieve monthly data
HOURLY : identifies need to retrieve hourly data
L_COMP : identifies request for a complete file listing
DAISTA : identifies request for daily statistics process
STA    : alternate name requesting daily statistics process
MONSTA : identifies request for monthly statistics process
COR    : identifies request for correlation process
HIG    : identifies request for highest process
MAX    : alternate name requesting highest process
LOW    : identifies request for lowest process
MIN    : alternate name requesting lowest process
EXT    : identifies request for extreme process
RANORD : identifies request for rank order process
HIGOCC : identifies request for high occurrences process
LOWOCC : identifies request for low occurrences process
DAIOCC : identifies request for daily occurrences process
OCC    : alternate name requesting daily occurrences process
MONOCC : identifies request for monthly occurrences process
```

```
      FLODURTA: identifies request for flow duration table process
      SUM     : identifies request for summary process
      CAL     : identifies request for calendar process

   SAS data sets:
     PROCESS (output): with variables:
        PRC_NAME  PARMETER
```

C) No macros are called by this module.

D) No permanent files are used by this module.

E) The module generates only a portion of a single DATA step.
SAS variables shared with the calling module are:

```
      RECORD  (char): user command from input data set
      COLUMN1 (char): character in first column of user command
      TOGGLE   (num): identifies when to write to data set
```

4.15.3 Internal Data Description

No local macro variables are used by this module.

SAS variables:

```
   COLUMN7  (char): data in first 7 columns of user command
   MAC_VAR  (char): name of macro variable for a process command
   WORD1    (char): first word of process name
   WORD2    (char): second word of process name
   WORD3    (char): third word of process name
   SUBNAME1 (char): first 3 letters of WORD1
   SUBNAME2 (char): first 3 letters of WORD2
   SUBNAME3 (char): first 2 letters of WORD3
```

4.15.4 Design Language Description

```
   set PROCESS to 1;
   set MAIN_DS to 1;
   set L_COMP  to 0;

   LENGTH MAC_VAR $ 8  PRC_NAME $ 80  PARMETER $ 80;
   RETAIN TOGGLE 0 PRC_NAME $;
       /* TOGGLE identifies the beginning of a new process */
   COLUMN1 = SUBSTR(RECORD,1,1);
   IF there is not a blank in column 1 THEN
      COLUMN7 = SUBSTR(RECORD,1,7);
      IF the word in the first seven columns is not 'PROCESS'
      THEN
      /* next group of statements creates macro variables for  */
      /* each of the processes that have been identified.      */
```

```
/* The macro variable consists of the first 3 characters */
/* of each word in the name of the process.              */

   PRC_NAME = RECORD;
   WORD1 = SCAN(PRC_NAME,1,' ');
   WORD2 = SCAN(PRC_NAME,2,' ');
   WORD3 = SCAN(PRC_NAME,3,' ');
   SUBNAME1 = SUBSTR(WORD1,1,3);
   SUBNAME2 = SUBSTR(WORD2,1,3);
   SUBNAME3 = SUBSTR(WORD3,1,2);
   MAC_VAR  = TRIM(SUBNAME1) !! TRIM(SUBNAME2) !! SUBNAME3 ;
   set the value of MAC_VAR to 1;

/* only output to the PROCESS data set if TOGGLE = 1 */
/* or a new parameter has been read                  */

   TOGGLE = 1;  /* set TOGGLE to 1 since it is not        */
                /* yet known if the process has parameter */
                /* options or not                         */

   IF EOF THEN write values to PROCESS data set;

   SELECT (COLUMN7);

      WHEN ('MONTHLY')
        set MONTHLY to 1;
      WHEN ('HOURLY')
        set HOURLY to 1;
      WHEN ('SUMMARY')     /* request for summary process */
        DO;
          set DAILY to 0;
          set EL_MNTH to 1;
        END;
      WHEN ('CALENDA')     /* request for calendar process */
        DO;
          set DAILY to 1;
          set EL_TEMP to 1;
          set EL_PRCP to 1;
        END;
      OTHERWISE
        set DAILY to 1;
    END;  /* select column7 */

ELSE if first 7 columns are  'PROCESS' THEN
   IF TOGGLE = 1 THEN
      PARMETER = . ;
      write values to PROCESS data set;/* previous process */
                                       /* had no parameters*/
```

```
      TOGGLE = 0;
   PARMETER = .;
   PRC_NAME = 'PROCESS';
   write values to PROCESS data set; /* identify beginning */
                                     /* of new process     */

ELSE IF there is a blank in column 1 THEN
   PARMETER = RECORD;
   write values to PROCESS data set;
   TOGGLE = 0;
```

4.16 Description of Module SEARCH INDEX FOR CHARACTER DATA (SI_CHAR)

4.16.1 Processing Narrative

SEARCH INDEX FOR CHARACTER DATA searches the index, trying to match the value of one variable in each index observation with the operands in a user command. The search only works correctly for character variables. If a match is found, then a macro is called to store the corresponding station numbers in a SAS data set.

4.16.2 Interface Description

A) Parameter: KEYWORD - name of character variable in index file

B) Global data:

No global macro variables are used in this module.

No SAS data sets are directly created or used.

C) Macros called by this module:
%CHOOSE STATION SUBSET (4.19 - %CHOOSE)

D) Permanent files: NHIMS.INDEX

E) The module generates only a portion of a single DATA step. SAS variables shared with the calling module are:

AND_CNTR (num): number of AND commmands given by user
RECORD (char): user command from input data set

4.16.3 Internal Data Description

No local macro variables are used by this module.

SAS variables:
I (num): control variable for DO loop
POS (num): position returned from INDEX function
TOTAL (num): number of observations in index data set

4.16.4 Design Language Description

```
DO I = 1 TO TOTAL;
   SET ddname.INDEX  POINT=I  NOBS=TOTAL;
   POS = INDEX(RECORD,TRIM(&KEYWORD));
   IF POS is not equal to 0 THEN
            /* match was found, so            */
            /* write to proper station subset */
      %CHOOSE STATION SUBSET
END;
```

4.17 Description of Module SEARCH INDEX FOR NUMERIC DATA (SI_NUM)

4.17.1 Processing Narrative

SEARCH INDEX FOR NUMERIC DATA searches the index, trying to match the value of one variable in each index observation with the operands in a user command. The search only works correctly for numeric variables. If a match is found, then a macro is called to store the corresponding station numbers in a SAS data set.

4.17.2 Interface Description

A) Parameter: KEYWORD - name of character variable in index file

B) Global data:
No global macro variables or SAS data sets are used.

C) Macros called by this module:
%CHOOSE STATION SUBSET (4.19 - %CHOOSE)

D) Permanent files: NHIMS.INDEX

E) The module generates only a portion of a single DATA step.
SAS variables shared with the calling module are:

AND_CNTR (num): number of AND commmands given by user
RECORD (char): user command from input data set
OPERAND (char): value of operand in user command
WORD (num): current word used by SCAN function

4.17.3 Internal Data Description

No local macro variables are used by this module.

SAS variables:
```
I     (num): control variable for DO loop
TOTAL (num): number of observations in index data set
```

4.17.4 Design Language Description

```
OPERAND = SCAN(RECORD,WORD,' ');
DO WHILE(OPERAND is not equal to a blank);
   increment WORD by 1;
   DO I = 1 TO TOTAL;
      SET ddname.INDEX  POINT=I  NOBS=TOTAL;
      IF OPERAND = &KEYWORD THEN
            /* match was found, so            */
            /* write to proper station subset */
         %CHOOSE STATION SUBSET
   END;
   OPERAND = SCAN(RECORD,WORD,' ');
END;
```

4.17A Description of Module SEARCH INDEX FOR PARTIAL DATA (SI_PART)

4.17A.1 Processing Narrative

SEARCH INDEX FOR PARTIAL DATA searches the index, trying to match the value of one variable in each index observation with the operands in a user command. The search only works correctly for numeric variables. If a match is found, then a macro is called to store the corresponding station numbers in a SAS data set.
The length of the number in the user operand is checked, so that abbreviated versions of a number can be used in the search (i.e., the first 4 digits of a hucode).

4.17A.2 Interface Description

A) Parameter: KEYWORD - name of character variable in index file

B) Global data:
No global macro variables or SAS data sets are used.

C) Macros called by this module:
%CHOOSE STATION SUBSET (4.19 - %CHOOSE)

D) Permanent files: NHIMS.INDEX

E) The module generates only a portion of a single DATA step. SAS variables shared with the calling module are:

```
AND_CNTR (num): number of AND commmands given by user
RECORD  (char): user command from input data set
OPERAND (char): value of operand in user command
WORD     (num): current word used by SCAN function
STRING  (char): used to convert number to a
                character string
```

4.17A.3 Internal Data Description

No local macro variables are used by this module.

SAS variables:

```
I     (num): control variable for DO loop
TOTAL (num): number of observations in index data set
```

4.17A.4 Design Language Description

```
%* SEARCHES INDEX FOR NUMERIC DATA, OR;
%* PORTIONS OF A NUMERIC FIELD;

  OPERAND = SCAN(RECORD,WORD,' ');
  DO WHILE(OPERAND NE ' ');
     NUMBER  = LENGTH(OPERAND);
     WORD = WORD + 1;
     DO I = 1 TO TOTAL;
        SET INDX.INDEX  POINT=I  NOBS=TOTAL;
        STRING = &KEYWORD;
        STRING = SUBSTR(STRING,1,NUMBER);
        IF STRING = OPERAND THEN
           DO;       /* match was found, so write station */
                     /* number to proper station subset   */
              %CHOOSE
           END;
     END;
     OPERAND = SCAN(RECORD,WORD,' ');
  END;
```

4.18 Description of Module SEARCH INDEX FOR RANGE OF DATA (SI_RNGE)

4.18.1 Processing Narrative

SEARCH INDEX FOR RANGE OF DATA searches the index, trying to match the value of one variable in each index observation with a range of values given in a user command. The search only works correctly for numeric variables. If a match is found, then a macro is called to store the corresponding station numbers in a SAS data set.

4.18.2 Interface Description

A) Parameter: KEYWORD - name of character variable in index file

B) Global data:
No global macro variables are used in this module.

No SAS data sets are directly created or used.

C) Macros called by this module:
%CHOOSE STATION SUBSET (4.n - %CHOOSE)

D) Permanent files: NHIMS.INDEX

E) The module generates only a portion of a single DATA step.
SAS variables shared with the calling module are:

```
AND_CNTR (num): number of AND commmands given by user
RECORD  (char): user command from input data set
OPERAND (char): value of operand in user command
WORD     (num): current word used by SCAN function
```

4.18.3 Internal Data Description

No local macro variables are used by this module.

SAS variables:

```
I      (num): control variable for DO loop
TOTAL  (num): number of observations in index data set
LOWER (char): lower limit of range of requested data
UPPER (char): upper limit of range of requested data
```

4.18.4 Design Language Description

```
OPERAND = SCAN(RECORD,WORD,' ');
DO WHILE(OPERAND is not equal to a blank);
   increment WORD by 2;  /* skip over word TO */

   LOWER = OPERAND;    /* lower range of request */
   OPERAND = SCAN(RECORD,WORD,' ');
   IF OPERAND is not equal to a blank THEN
      UPPER = OPERAND;  /* upper range of request */
      increment WORD by 1;
   ELSE UPPER = missing;
```

```
      DO I = 1 TO TOTAL;
         SET ddname.INDEX  POINT=I  NOBS=TOTAL;
         IF &KEYWORD ¢= LOWER  THEN    /* within lower bound */
            IF (&KEYWORD _= UPPER) OR (UPPER = .)  THEN
               /* write values to output data set */

               %CHOOSE STATION SUBSET
      END;
      OPERAND = SCAN(RECORD,WORD,' ');
   END;
```

4.19 Description of Module CHOOSE STATION SUBSET (CHOOSE)

4.19.1 Processing Narrative

CHOOSE STATION SUBSET checks how many AND commands have been encountered, and outputs the current station number to the proper station subset.

4.19.2 Interface Description

A) There is no parameter list.

B) Global data:
   No global macro variables are used in this module.

   SAS data sets:
   STATION1-STATION5 (output): with variable:
      STATION

C) No macros are called by this module.

D) No permanent files are used.

E) The module generates only a portion of a single DATA step.
   SAS variables shared with the calling module are:

   AND_CNTR (num): number of AND commmands given by user

4.19.3 Internal Data Description

No local macro variables are used by this module.

SAS variables:
```
I      (num): control variable for DO loop
TOTAL  (num): number of observations in index data set
LOWER (char): lower limit of range of requested data
UPPER (char): upper limit of range of requested data
```

4.19.4 Design Language Description

```
IF AND_CNTR = 0 THEN
   write values to STATION1;
ELSE IF AND_CNTR = 1 THEN
   write values to STATION2;
ELSE IF AND_CNTR = 2 THEN
   write values to STATION3;
ELSE IF AND_CNTR = 3 THEN
   write values to STATION4;
ELSE IF AND_CNTR = 4 THEN
   write values to STATION5;
```

4.20 Description of Module CHECK NUMBER OF OBSERVATIONS (NUM_OBS)

4.20.1 Processing Narrative

CHECK NUMBER OF OBSERVATIONS will determine how many observations are in a given data set. A global macro variable is set to the value of the number of observations. The idea for this macro came from [12, p. 1137].

4.20.2 Interface Description

A) Parameter: DSN - name of a SAS data set

B) Global data:

Macro variables:
NOBS: number of observations in a data set

SAS data sets:
Any previously created data set (input)

C) There are no macros called by this module.

D) No permanent files are used by this module.

E) The module generates one complete _NULL_ DATA step.

4.20.3 Internal Data Description

No local macro variables are used in this module.

SAS variables:
NUMBER (num): number of observations in a data set
I (num): value used in the POINT option
_N_ (num): current execution of the DATA step

4.20.4 Design Language Description

```
DATA _NULL_;

   I=1;
   SET &DSN POINT=I NOBS=NUMBER;
   IF _N_ = 1 AND NUMBER = 0 THEN
        set NOBS to 0;
   ELSE
        set NOBS to the value of NUMBER;
```

4.21 Description of Module SORT AND MERGE STATION SUBSETS (SRT_MRG)

4.21.1 Processing Narrative

SORT AND MERGE STATION SUBSETS sorts the station subsets, if necessary, depending on the value of the SORT variable. Then the subsets are merged into one data set of station numbers. This module can sort and merge a variable number of data sets, depending on the value passed as a parameter. The idea for this technique came from [16, p. 252].

4.21.2 Interface Description

A) Parameter: HOWMANY - number of existing station subsets

B) Global data:

Macro variables:
SORT: identifies need to sort station subsets

SAS data sets:
STATION1-STATION5 (input): with variable:
STATION
STATIONS (output): with variable:
STATION

C) There are no macros called by this module.

D) No permanent files are used by this module.

E) The module generates one complete DATA step and perhaps one or more complete PROC steps.

4.21.3 Internal Data Description

Macro variables:
- SETS : the names of the station subsets to be merged
- INS : variable names used in the subsetting IF statement
- C : control variable for DO loop
- S : control variable for DO loop

SAS variables:
- IN1-IN5 (boolean): identifies which data set is contributing to the current observation

4.21.4 Design Language Description

```
%LOCAL SETS INS C S;
%LET SETS = STATION1(IN=IN1);
%LET INS = IN1;

%DO C = 2 %TO &HOWMANY;
    %LET SETS = &SETS STATION&C (IN=IN&C );
    %LET INS  = &INS AND IN&C;
%END;

%IF &SORT is equal to 1  THEN
    %DO S = 1 %TO &HOWMANY;
        PROC SORT DATA=STATION&S;
           BY STATION;
        RUN;
    %END;

DATA STATIONS;
   %IF &HOWMANY = 1 %THEN
       SET STATION1;

   %ELSE
       MERGE &SETS;
          BY STATION;
       IF &INS;
```

4.22 Description of Module RETRIEVE DATA FOR ALL ELEMENTS (RETRIEV)

4.22.1 Processing Narrative

RETRIEVE DATA FOR ALL ELEMENTS checks the flag for each element; if the flag for an element is set, then the macros are called to retrieve data for that element from the permanent files. An error message is given if the stations for a requested element are not found in the pointer files.

4.22.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:

```
EL_PRCP: identifies request for precipitation data
EL_TEMP: identifies request for temperature data
EL_STRM: identifies request for streamflow data
EL_SNOW: identifies request for snowfall data
EL_SNOC: identifies request for snow course data
EL_RESV: identifies request for reservoir data
EL_PEAK: identifies request for peak flow data
EL_EVAP: identifies request for evaporation data
EL_MNTH: identifies request for monthly summary data
EL_HPCP: identifies request for hourly precip data
SUM    : identifies request for monthly summary process
NOBS   : number of observations in a data set
POINTER: identifies use of user-specified pointers
```

PRINT (output) - standard SAS print file

C) The macros called by this module are:

```
MERGE USER-POINTERS WITH STATIONS (4.23 - PTR_STA)
FIND POINTERS (4.24 - FINDPTR)
CHECK NUMBER OF OBSERVATIONS (4.20 - NUM_OBS)
MERGE POINTERS WITH INDEX  (4.25 - PTR_IDX)
GET DATA FROM PERMANENT DATA SET (4.26 - GETDATA)
GET PERMANENT DATA WITH ANNUAL RECORDS (4.27 - GET_ANN)
CONVERT PRECIPITATION DATA (4.28 - CV_PRCP)
CONVERT TEMPERATURE DATA   (4.29 - CV_TEMP)
CONVERT STREAMFLOW  DATA   (4.30 - CV_STRM)
CONVERT SNOWFALL    DATA   (4.31 - CV_SNOW)
CONVERT RESERVOIR   DATA   (4.32 - CV_RESV)
CONVERT EVAPORATION DATA   (4.33 - CV_EVAP)
CONVERT SNOW COURSE DATA   (4.34 - CV_SNOC)
```

```
CONVERT PEAK FLOW    DATA    (4.35 - CV_PEAK)
CONVERT MONTHLY SUMMARY DATA (4.36 - CV_MNTH)
CONVERT HOURLY PRECIPITATION DATA (4.37 - CV_HPCP)
```

D) No permanent files are directly used by this module.

E) The module generates macro statements only, except when an error message is issued; a complete _NULL_ DATA step is then generated.

4.22.3 Internal Data Description

No local data are used by this module.

4.22.4 Design Language Description

```
%IF NOT a request for summary process  %THEN
    %* retrieve data from daily files;

     %IF &EL_PRCP %THEN
        %*  retrieve precip data  ;

           %IF NOT &POINTER %THEN
               %FIND POINTERS (PRCP,PRCP.POINTERS)
           %ELSE
               %MERGE USER-POINTERS WITH STATIONS (PRCP)

           %CHECK NUMBER OF OBSERVATIONS (PRCPPTRS)
           %IF the number of observations is greater than 0 %THEN
                 %*  valid pointer values were found, so retrieve;
                 %MERGE POINTERS WITH INDEX (PRCP)
                 %GET DATA FROM PERMANENT DATA SET (PRCP)
                 %CONVERT PRECIPITATION DATA
           %ELSE
                DATA _NULL_;
                  issue error message 'REQUESTED STATION(S) NOT
                                       FOUND IN PRECIPITATION FILE'
                  reset EL_PRCP to 0;  RUN;


     %IF &EL_TEMP %THEN
        %* retrieve temperature data ;

           %IF NOT &POINTER %THEN
               %FIND POINTERS (TEMP,TEMP.POINTERS)
           %ELSE
               %MERGE USER-POINTERS WITH STATIONS (TEMP)

           %CHECK NUMBER OF OBSERVATIONS (TEMPPTRS)
           %IF the number of observations is greater than 0 %THEN
```

```
         %* valid pointer values were found, so retrieve;
            %MERGE POINTERS WITH INDEX (TEMP) %GET DATA FROM
            PERMANENT DATA SET (TEMP) %CONVERT TEMPERATURE
            DATA
      %ELSE DATA _NULL_;
              issue error message 'REQUESTED STATION(S) NOT
                                   FOUND IN TEMPERATURE FILE'
              reset EL_TEMP to 0;  RUN;

%IF &EL_STRM %THEN
   %* retrieve streamflow data ;

      %IF NOT &POINTER %THEN
          %FIND POINTERS (STRM,STRM.POINTERS)
      %ELSE
          %MERGE USER-POINTERS WITH STATIONS (STRM)

      %CHECK NUMBER OF OBSERVATIONS (STRMPTRS)
      %IF the number of observations is greater than 0 %THEN
         %* valid pointer values were found, so retrieve;
            %MERGE POINTERS WITH INDEX(STRM)
            %GET DATA FROM PERMANENT DATA SET (STRM)
            %CONVERT STREAMFLOW DATA
      %ELSE
            DATA _NULL_;
              issue error message 'REQUESTED STATION(S) NOT
                                   FOUND IN STREAMFLOW FILE'
              reset EL_STRM to 0;
            RUN;

%IF &EL_SNOW %THEN
   %* retrieve snow fall data ;

      %IF NOT &POINTER %THEN
          %FIND POINTERS (SNOW,SNOW.POINTERS)
      %ELSE
          %MERGE USER-POINTERS WITH STATIONS (SNOW)

      %CHECK NUMBER OF OBSERVATIONS (SNOWPTRS)
      %IF the number of observations is greater than 0 %THEN
         %* valid pointer values were found, so retrieve;
            %MERGE POINTERS WITH INDEX (SNOW)
            %GET DATA FROM PERMANENT DATA SET (SNOW)
            %CONVERT SNOW FALL DATA
      %ELSE
            DATA _NULL_;
              issue error message 'REQUESTED STATION(S) NOT
                                   FOUND IN SNOWFALL FILE'
              reset EL_SNOW to 0;
            RUN;
```

```
%IF &EL_EVAP %THEN
   %* retrieve evaporation data

      %IF NOT &POINTER %THEN
          %FIND POINTERS (EVAP,EVAP.POINTERS)
      %ELSE
          %MERGE USER-POINTERS WITH STATIONS (EVAP)

      %CHECK NUMBER OF OBSERVATIONS (EVAPPTRS)

      %IF the number of observations is greater than 0 %THEN
         % valid pointer values were found, so retrieve;
            %MERGE POINTERS WITH INDEX (EVAP)
            %GET DATA FROM PERMANENT DATA SET (EVAP)
            %CONVERT EVAPORATION DATA
      %ELSE
            DATA _NULL_;
              issue error message 'REQUESTED STATION(S) NOT
                                   FOUND IN EVAPORATION FILE'
              reset EL_EVAP to 0;
            RUN;


%IF &EL_RESV %THEN
   %* retrieve reservoir data

      %IF NOT &POINTER %THEN
          %FIND POINTERS (RESV,RESV.POINTERS)
      %ELSE
          %MERGE USER-POINTERS WITH STATIONS (RESV)

      %CHECK NUMBER OF OBSERVATIONS (RESVPTRS)
      %IF the number of observations is greater than 0 %THEN
         %* valid pointer values were found, so retrieve;
            %MERGE POINTERS WITH INDEX (RESV)
            %GET DATA FROM PERMANENT DATA SET (RESV)
            %CONVERT RESERVOIR DATA
      %ELSE
            DATA _NULL_;
              issue error message 'REQUESTED STATION(S) NOT
                                   FOUND IN RESERVOIR FILE'
              reset EL_RESV to 0;
            RUN;
%END;  %*retrieve data from daily files;


%IF a request for summary process OR &EL_MNTH %THEN
    %* retrieve data from monthly summary file;
```

```
            %IF NOT &POINTER %THEN
                %FIND POINTERS (MNTH,MNTH.POINTERS)
            %ELSE
                %MERGE USER-POINTERS WITH STATIONS (MNTH)

            %CHECK NUMBER OF OBSERVATIONS (MNTHPTRS)
            %IF the number of observations is greater than 0 %THEN
               %* valid pointer values were found, so retrieve;
                  %MERGE POINTERS WITH INDEX (MNTH)
                  %GET DATA FROM PERMANENT DATA SET (MNTH)
                  %CONVERT MONTHLY SUMMARY DATA

            %ELSE
                  DATA _NULL_;
                    issue error message 'REQUESTED STATION(S) NOT
                                         FOUND IN MONTHLY FILE'
                    reset EL_MNTH to 0;
                  RUN;


%IF &EL_SNOC %THEN
    %* retrieve data from snow course file;

            %IF NOT &POINTER %THEN
                %FIND POINTERS (SNOC,SNOC.POINTERS)
            %ELSE
                %MERGE USER-POINTERS WITH STATIONS (SNOC)

            %CHECK NUMBER OF OBSERVATIONS (SNOCPTRS)
            %IF the number of observations is greater than 0 %THEN
               %* valid pointer values were found, so retrieve;
                  %MERGE POINTERS WITH INDEX (SNOC)
                  %GET PERMANENT DATA WITH ANNUAL RECORDS (SNOC)
                  %CONVERT SNOW COURSE DATA
            %ELSE
                  DATA _NULL_;
                    issue error message 'REQUESTED STATION(S) NOT
                                         FOUND IN SNOW COURSE FILE'
                    reset EL_SNOC to 0;
                  RUN;

%IF &EL_PEAK %THEN
    %* retrieve peak flow data;

            %IF NOT &POINTER %THEN
                %FIND POINTERS (PEAK,PEAK.POINTERS)
            %ELSE
                %MERGE USER-POINTERS WITH STATIONS (PEAK)
```

```
          %CHECK NUMBER OF OBSERVATIONS (PEAKPTRS)
          %IF the number of observations is greater than 0 %THEN
             %* valid pointer values were found, so retrieve;
                %MERGE POINTERS WITH INDEX (PEAK)
                %GET PERMANENT DATA WITH ANNUAL RECORDS (PEAK)
                %CONVERT PEAK FLOW DATA
          %ELSE
                DATA _NULL_;
                  issue error message 'REQUESTED STATION(S) NOT
                                       FOUND IN PEAK FLOW FILE'
                  reset EL_PEAK to 0;
                RUN;

%IF &EL_HPCP %THEN
    %* retrieve data from hourly precip file;

          %IF NOT &POINTER %THEN
              %FIND POINTERS (HPCP,HPCP.POINTERS)
          %ELSE
              %MERGE USER-POINTERS WITH STATIONS (HPCP)

          %CHECK NUMBER OF OBSERVATIONS (HPCPPTRS)
          %IF the number of observations is greater than 0 %THEN
             %* valid pointer values were found, so retrieve;
                %MERGE POINTERS WITH INDEX (HPCP)
                %GET DATA FROM PERMANENT DATA SET (HPCP)
                %CONVERT HOURLY PRECIPITATION DATA
          %ELSE
                DATA _NULL_;
                 issue error message 'REQUESTED STATION(S) NOT
                                      FOUND IN HOURLY PRECIP FILE'

                 reset EL_HPCP to 0;
                RUN;
```

4.23 Description of Module MERGE USER-POINTERS WITH STATIONS (PTR_STA)

4.23.1 Processing Narrative

MERGE USER-POINTERS WITH STATIONS combines the station numbers and the pointers (FOBS_PTR LOBS_PTR) that are explicitly given by the user. Thus, a data set is created which is identical to that produced by the macro which searches the permanent POINTERS file.

4.23.2 Interface Description

A) Parameter: ELEMENT - 4-character name of element

B) Global data:

There are no global macro variables.

SAS data sets:
- STATIONS (input) - with variables:
  - STATION
- POINTERS (input) - with variables:
  - FOBS_PTR
  - LOBS_PTR
- &ELEMENT.PTRS (output) - with variables:
  - STATION
  - FOBS_PTR
  - LOBS_PTR

C) There are no macros called by this module.

D) No permanent files are accessed by this module.

E) The module generates one complete DATA step.

4.23.3 Internal Data Description

No local data are used by this module.

4.23.4 Design Language Description

```
DATA &ELEMENT.PTRS;

   MERGE POINTERS  STATIONS;
```

4.24 Description of Module FIND POINTERS (FINDPTR)

4.24.1 Processing Narrative

FIND POINTERS merges the POINTERS data set for a particular element with the STATIONS data set, in order to identify the range(s) of observation numbers which contain the requested data. An error message is generated if any of the requested station numbers are not found in the POINTER file. The strategy employed here is a modification of the one found in [7, p. 295].

4.24.2 Interface Description

A) Parameter: ELEMENT - 4-character name of element
DSN - name of the permanent pointers file used as input

B) Global data:

There are no global macro variables.

SAS data sets:
STATIONS (input) - with variables:
STATION
&ELEMENT.PTRS (output) - with variables:
STATION
FOBS_PTR
LOBS_PTR

C) There are no macros called by this module.

D) Permanent files: any of the permanent pointers files.

E) The module generates one complete DATA step.

4.24.3 Internal Data Description

No macro variables are used by this module.

SAS variables:
LEGAL (boolean): identifies data read from &ELEMENT..POINTERS
REQUEST (boolean): identifies data read from STATIONS

4.24.4 Design Language Description

```
DATA &ELEMENT.PTRS;

   MERGE &DSN (IN=LEGAL)  STATIONS(IN=REQUEST);
      BY STATION;
   IF NOT LEGAL THEN issue error message that the station is
                     not a legal request for the element;

   IF LEGAL AND REQUEST;
```

4.25 Description of Module MERGE POINTERS WITH INDEX (PTR_INX)

4.25.1 Processing Narrative

MERGE POINTERS WITH INDEX combines the data from the index file necessary for report writing (NAME and COUNTY) with the matching stations in the subset of the pointers file.

4.25.2 Interface Description

A) Parameter: ELEMENT - 4-character name of element

B) Global data:

No macro variables are used in this module.

SAS data sets:
&ELEMENT.PTRS (input) - with variables:
STATION FOBS_PTR LOBS_PTR
&ELEMENT.PTIX (output) - with variables:
STATION FOBS_PTR LOBS_PTR NAME COUNTY AREA ELEV

C) There are no macros called by this module.

D) Permanent files: ddname.INDEX.

E) The module generates one complete DATA step.

4.25.3 Internal Data Description

There are no macro variables used in this module.

SAS variables:
PTRS (boolean): identifies data read from &ELEMENT.PTRS

4.25.4 Design Language Description

```
DATA &ELEMENT.PTIX;

   MERGE &ELEMENT.PTRS(IN=PTRS) ddname.INDEX;
     BY STATION;
   KEEP STATION FOBS_PTR LOBS_PTR NAME COUNTY AREA ELEV;
   IF PTRS;
```

4.26 Description of Module GET DATA FROM PERMANENT DATA SET (GETDATA)

4.26.1 Processing Narrative

GET DATA FROM PERMANENT DATA SET retrieves the requested observations from the main data file for a particular element. If a specific period of record is requested, only the data in that period are retrieved. The POINT option is used to directly access the data from the main files [7, p. 295].

4.26.2 Interface Description

A) Parameter: ELEMENT - 4-character name of element

B) Global data:

Macro variables:
PERIOD: number of requested periods of record

SAS data sets:
PERIODS (input) - with variables:
BEGDATE
ENDDATE
&ELEMENT.PTIX (input) - with variables:
STATION
FOBS_PTR
LOBS_PTR
NAME COUNTY AREA ELEV
&ELEMENT.DATA (output) - with variables:
STATION NAME COUNTY AREA ELEV, plus
all variables in permanent data set for the element

C) There are no macros called by this module.

D) Permanent files: &ELEMENT..MAINDATA.

E) The module generates one complete DATA step.

4.26.3 Internal Data Description

There are no macro variables used in this module.

SAS variables:
KEYDATE (num): year and month in current observation
I (num): value used in the POINT option
P (num): value used in the POINT option

4.26.4 Design Language Description

```
DATA &ELEMENT.DATA;

   SET &ELEMENT.PTIX;
   DO I = FOBS_PTR TO LOBS_PTR;
      SET &ELEMENT..MAINDATA  POINT=I;

      %IF &PERIOD greater than 0 %THEN
            DROP KEYDATE BEGDATE ENDDATE FOBS_PTR LOBS_PTR;
            KEYDATE = (YEAR * 100) + MONTH;

            DO P = 1 TO number of periods of record (&PERIOD)
               SET PERIODS POINT=P;
               IF BEGDATE _= KEYDATE _= ENDDATE THEN
                  OUTPUT to output data set;
            END;
```

```
    %ELSE
          DROP FOBS_PTR LOBS_PTR;
          OUTPUT to output data set;

    END;
```

4.27 Description of Module GET PERMANENT DATA WITH ANNUAL RECORDS (GET_ANN)

4.27.1 Processing Narrative

GET PERMANENT DATA WITH ANNUAL RECORDS retrieves the requested observations from the main data file for a particular element with annual, instead of monthly, records. If a specific period of record is requested, only the data in that period are retrieved. However, any months specified by the user are ignored and only the years of the period are checked. The POINT option is used to directly access the data from the main files [7, p.295].

4.27.2 Interface Description

A) Parameter: ELEMENT - 4-character name of element

B) Global data:

Macro variables:
PERIOD: number of requested periods of record

SAS data sets:
PERIODS (input) - with variables:
BEGDATE
ENDDATE
&ELEMENT.PTIX (input) - with variables:
STATION
FOBS_PTR
LOBS_PTR
NAME COUNTY AREA ELEV
&ELEMENT.DATA (output) - with variables:
STATION NAME COUNTY AREA ELEV, plus
all variables in permanent data set for the element

C) There are no macros called by this module.

D) Permanent files: &ELEMENT..MAINDATA.

E) The module generates one complete DATA step.

4.27.3 Internal Data Description

There are no macro variables used in this module.

SAS variables:
KEYDATE (num): year and month in current observation
I (num): value used in the POINT option
P (num): value used in the POINT option

4.27.4 Design Language Description

```
DATA &ELEMENT.DATA;

   SET &ELEMENT.PTIX;
   DO I = FOBS_PTR TO LOBS_PTR;
      SET &ELEMENT..MAINDATA  POINT=I;

      %IF &PERIOD greater than 0 %THEN  /* check periods */
            DROP KEYDATE BEGDATE ENDDATE FOBS_PTR LOBS_PTR;
            KEYDATE = YEAR;

            DO P = 1 TO number of periods of record (&PERIOD)
               SET PERIODS POINT=P;
               BEGDATE = INT(BEGDATE / 100); /* drop month */
               ENDDATE = INT(ENDDATE / 100); /* drop month */
               IF BEGDATE _= KEYDATE _= ENDDATE THEN
                  OUTPUT to output data set;
            END;

      %ELSE
            DROP FOBS_PTR LOBS_PTR;
            OUTPUT to output data set;

   END;
```

4.28 Description of Module CONVERT PRECIPITATION DATA (CV_PRCP)

4.28.1 Processing Narrative

CONVERT PRECIPITATION DATA converts the daily values and/or monthly total by the proper factor, according to the metric option. If not needed, daily values are dropped. Output formats are assigned here, according to the metric option. A formatted version of the STATION variable is also calculated.

4.28.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:
METRIC: identifies units for the data DAILY : identifies whether or not to keep daily values

SAS data sets:
PRCPDATA (input) - with variables:
STATION YEAR MONTH PRCP_TOT MONTHSUM NAME
COUNTY AREA ELEV PRECIP1-PRECIP31 P_CODE1-P_CODE31
PRCP (output) - with variables:
same as PRCPDATA, plus: STN_FORM

C) There are no macros called by this module.

D) No permanent files are used by this module.

E) The module generates one complete DATA step.

4.28.3 Internal Data Description

There are no local macro variables used in this module.

SAS variables:
ADJUST: value by which to adjust data
P: control variable for DO loop
STATE: first two digits of station number
STAT_NUM: last 4 digits of station number

4.28.4 Design Language Description

```
DATA PRCP;

   ARRAY PRECIP[31] PRECIP1-PRECIP31;

   %IF &METRIC = 0 %THEN
       RETAIN ADJUST  0.01;   /* units are inches */
       FORMAT PRCP_TOT 6.2 PRECIP1-PRECIP31 6.2;
   %ELSE
       RETAIN ADJUST  0.254;  /* units are millimeters */
       FORMAT PRCP_TOT 6.1 PRECIP1-PRECIP31 6.1;

   SET PRCPDATA;
   multiply PRCP_TOT by the value of ADJUST;
   %IF &DAILY %THEN
       DO P = 1 TO 31;
          multiply daily values by the value of ADJUST;
       DROP STATE STAT_NUM ADJUST P;
```

```
   %ELSE
       DROP P  PRECIP1-PRECIP31
               P_CODE1-P_CODE31 STATE STAT_NUM ADJUST;

   STATE = SUBSTR(STATION,1,2);
   STAT_NUM = SUBSTR(STATION,3,4);
   STN_FORM = TRIM(STATE) !! '-' !! STAT_NUM;
   OUTPUT;
RUN;
```

4.29 Description of Module CONVERT TEMPERATURE DATA (CV_TEMP)

4.29.1 Processing Narrative

CONVERT TEMPERATURE DATA converts the daily values by the proper factor if the metric option is in effect. Output formats are assigned, and a formatted version of the STATION variable is calculated. Also, the mean maximum and minimum temperatures for the month are calculated. If not needed, the daily values are dropped and only the monthly averages are kept.

4.29.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:
- METRIC: identifies units for the data
- DAILY : identifies whether or not to keep daily values

SAS data sets:
- TEMPDATA (input) - with variables:
  STATION YEAR MONTH MONTHSUM NAME COUNTY AREA ELEV
  MAXTMP1-MAXTMP31 MINTMP1-MINTMP31
- TEMP (output) - with variables:
  same as TEMPDATA, plus: STN_FORM AVEMAX AVEMIN
  MXCODE1-MXCODE31 MNCODE1-MNCODE31

C) There are no macros called by this module.

D) No permanent files are used by this module.

E) The module generates one complete DATA step.

4.29.3 Internal Data Description

There are no local macro variables used in this module.

SAS variables:

STATE:	first two digits of station number
STAT_NUM:	last 4 digits of station number
T:	control variable for DO loop

4.29.4 Design Language Description

```
DATA TEMP;

   ARRAY MAXTMP[31] MAXTMP1-MAXTMP31;
   ARRAY MINTMP[31] MINTMP MINTMP31;
   ARRAY MXCODE[31] $ 1 MXCODE1-MXCODE31;
   ARRAY MNCODE[31] $ 1 MNCODE1-MNCODE31;

   FORMAT MAXTMP1-MAXTMP31 4. MINTMP1-MINTMP31 4.
          AVEMAX AVEMIN 7.1;

   SET TEMPDATA;

   DO T = 1 TO 31; /* CHECK FOR ESTIMATES(-80) AND CONVERSIONS */
      IF MAXTMP[T] ¢ -80 THEN
         MXCODE[T] = ' ';
      ELSE IF MAXTMP[T] = . THEN
         MXCODE[T] = 'M';
      ELSE DO;
         MXCODE[T] = 'E';
         MAXTMP[T] = MAXTMP[T] + 200;

      IF MINTMP[T] ¢ -80 THEN
         MNCODE[T] = ' ';
      ELSE IF MINTMP[T] = .  THEN
         MNCODE[T] = 'M';
      ELSE DO;
         MNCODE[T] = 'E';
         MINTMP[T] = MINTMP[T] + 200;

      %IF &METRIC NE 0 %THEN  /* convert daily values to metric */
      MAXTMP[T] = (MAXTMP[T] - 32) * 5/9;
      MINTMP[T] = (MINTMP[T] - 32) * 5/9;
      %END;
   END;  /* CHECK FOR ESTIMATES AND CONVERSIONS */

   AVEMAX = MEAN(OF MAXTMP1-MAXTMP31);
   AVEMIN = MEAN(OF MINTMP1-MINTMP31);

   %IF &DAILY %THEN
       DROP STATE STAT_NUM T;
   %ELSE
       KEEP STATION YEAR MONTH MONTHSUM STN_FORM AVEMAX AVEMIN;
   %END;
```

```
   STATE = SUBSTR(STATION,1,2);
   STAT_NUM = SUBSTR(STATION,3,4);
   STN_FORM = TRIM(STATE) !! '-' !! STAT_NUM;
   OUTPUT;
RUN;
```

4.30 Description of Module CONVERT STREAMFLOW DATA (CV_STRM)

4.30.1 Processing Narrative

CONVERT STREAMFLOW DATA converts the daily values and the monthly total by the proper factor, according to the metric option. Output formats are assigned, and a formatted version of the STATION variable is calculated. Also, the mean, maximum and minimum values for the month are calculated. If not needed, the daily values are dropped and only monthly totals are kept.

4.30.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:

METRIC: identifies units for the data
DAILY : identifies whether or not to keep daily values

SAS data sets:

STRMDATA (input) - with variables:
STATION YEAR MONTH STRM_TOT
NAME COUNTY AREA ELEV FLOW1-FLOW31

STRM (output) - with variables:
same as STRMDATA, plus: STN_FORM MAX_STRM MIN_STRM
AVE_STRM MONTHSUM DAY_DIS

C) There are no macros called by this module.

D) No permanent files are used by this module.

E) The module generates one complete DATA step.

4.30.3 Internal Data Description

There are no local macro variables used in this module.

SAS variables:

ADJUST: value by which to adjust data
STATE: first two digits of station number
STAT_NUM: middle 4 digits of station number
S: control variable for DO loop

```
STAT_END: last 2 digits of station number
DAY_TOT : accumulator for the daily streamflow values
NUM_DAYS: number of daily values stored in accumulator
LAST.STATION: boolean value identifying last station value
```

4.30.4 Design Language Description

```
DATA STRM;

   RETAIN DAY_TOT NUM_DAYS 0;
   ARRAY FLOW[31] FLOW1-FLOW31;
   FORMAT FLOW1-FLOW31 STRM_TOT 7.;

   %IF &METRIC = 0 %THEN
       RETAIN ADJUST 0.01; /* units are cfs */
   %ELSE
       RETAIN ADJUST 0.0002832; /* units are cubic meters/s */
       AREA = AREA * 2.59;   /* units are square kilometers */
   %END;

   SET STRMDATA;
     BY STATION;
   multiply STRM_TOT by the value of ADJUST;
   DO S = 1 TO 31;
      multiply daily values by ADJUST;
      %IF &MONTHLY %THEN
          IF daily value is not missing THEN
             /* calculations for mean daily discharge */
             DAY_TOT = DAY_TOT + STRM;
             NUM_DAYS = NUM_DAYS + 1;
      %END;
   END;

   %IF &DAILY %THEN
       N_MISS = number of missing values in the month;
       IF MONTH has 30 days
          THEN N_MISS = N_MISS - 1;
       ELSE IF MONTH = 2 THEN
               IF MOD(YEAR,4) NE 0 THEN
                    N_MISS = N_MISS - 3;
               ELSE N_MISS = N_MISS - 2;

       IF N_MISS _ 1  THEN
          MAX_STRM = maximum monthly value;
          MIN_STRM = minimum monthly value;
          AVE_STRM = mean monthly value;
```

```
        ELSE
           set MAX_STRM to missing;
           set MIN_STRM to missing;
           IF N_MISS _ 6 THEN
              AVE_STRM = mean monthly value;
           ELSE
              set AVE_STRM to missing;

       DROP STATE STAT_NUM STAT_END S ADJUST N_MISS;

   %ELSE %IF &MONTHLY %THEN
       DROP FLOW1-FLOW31 STATE STAT_NUM STAT_END S ADJUST;
       IF LAST.STATION THEN
          DAY_DIS = DAY_TOT / NUM_DAYS;
          DAY_TOT = 0;
          NUM_DAYS = 0;
   %END;

   MONTHSUM = 0;
   STATE = SUBSTR(STATION,1,2);
   STAT_NUM = SUBSTR(STATION,3,4);
   STAT_END = SUBSTR(STATION,7,2);
   STN_FORM =
       TRIM(STATE) !! '.' !! TRIM(STAT_NUM) !! '.' !! STAT_END;
   OUTPUT;
RUN;
```

4.31 Description of Module CONVERT SNOWFALL DATA (CV_SNOW)

4.31.1 Processing Narrative

CONVERT SNOWFALL DATA converts the daily values and/or monthly total by the proper factor, according to the metric option. If not needed, daily values are dropped. Output formats are assigned here, according to the metric option. A formatted version of the STATION variable is also calculated.

4.31.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:
- METRIC: identifies units for the data
- DAILY : identifies whether or not to keep daily values

```
   SAS data sets:
     SNOWDATA (input) - with variables:
        STATION YEAR MONTH SNOW_TOT MONTHSUM NAME COUNTY AREA ELEV
        SNOW1-SNOW31 DEPTH1-DEPTH31 S_CODE1-S_CODE31D_CODE1-DCODE31
     SNOW (output) - with variables:
        same as SNOWDATA, plus: STN_FORM

C) There are no macros called by this module.

D) No permanent files are used by this module.

E) The module generates one complete DATA step.
```

4.31.3 Internal Data Description

```
There are no local macro variables used in this module.

SAS variables:
     ADJUST: value by which to adjust data
          S: control variable for DO loop
      STATE: first two digits of station number
   STAT_NUM: last 4 digits of station number
```

4.31.4 Design Language Description

```
DATA SNOW;

   ARRAY SNOW[31] SNOW1-SNOW31;
   ARRAY DEPTH[31] DEPTH1-DEPTH31;
   FORMAT SNOW_TOT 7.1 SNOW1-SNOW31 4.1 DEPTH1-DEPTH31 3.;

   SET SNOWDATA;

   %IF &METRIC = 0 %THEN
       RETAIN ADJUST  0.1;    /* units are inches */
   %ELSE
       RETAIN ADJUST  0.254;  /* units are centimeters */
       %IF &DAILY %THEN
       DO S = 1 TO 31 ;
          multiply daily depth values by 2.54;
       END;
   %END;

   multiply SNOW_TOT by the value of ADJUST;
   %IF &DAILY %THEN
       DO S = 1 TO 31;
          multiply daily snowfall values by the value of ADJUST;
       DROP STATE STAT_NUM ADJUST S;
   %ELSE
       DROP S  SNOW1-SNOW31 DEPTH1-DEPTH31  S_CODE1-S_CODE31
               D_CODE1-D_CODE31 STATE STAT_NUM ADJUST;
```

```
   STATE = SUBSTR(STATION,1,2);
   STAT_NUM = SUBSTR(STATION,3,4);
   STN_FORM = TRIM(STATE) !! '-' !! STAT_NUM;
   OUTPUT;
RUN;
```

4.32 Description of Module CONVERT EVAPORATION DATA (CV_EVAP)

4.32.1 Processing Narrative

CONVERT EVAPORATION DATA converts the daily values and/or monthly totals by the proper factor, according to the metric option. If not needed, daily values are dropped. Output formats are assigned here, according to the metric option. A formatted version of the STATION variable is also calculated.

4.32.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:
- METRIC: identifies units for the data
- DAILY : identifies whether or not to keep daily values

SAS data sets:
- EVAPDATA (input) - with variables:
  STATION YEAR MONTH EVAP_TOT WIND_TOT MONTHSUM NAME COUNTY AREA ELEV EVAP1-EVAP31 WIND1-WIND31 E_CODE1-ECODE31 W_CODE1-W_CODE31
- EVAP (output) - with variables:
  same as EVAPDATA, plus: STN_FORM

C) There are no macros called by this module.

D) No permanent files are used by this module.

E) The module generates one complete DATA step.

4.32.3 Internal Data Description

There are no local macro variables used in this module.

SAS variables:

```
     ADJUST: value by which to adjust data
          E: control variable for DO loop
      STATE: first two digits of station number
   STAT_NUM: last 4 digits of station number
```

4.32.4 Design Language Description

```
DATA EVAP;

   ARRAY EVAP[31] EVAP1-EVAP31;
   ARRAY WIND[31] WIND1-WIND31;

   SET EVAPDATA;

   %IF &METRIC = 0 %THEN
       RETAIN ADJUST  0.01;   /* units are inches */
       FORMAT EVAP_TOT WIND_TOT 6.2 EVAP1-EVAP31 4.2
              WIND1-WIND31 3.;
   %ELSE
       RETAIN ADJUST  0.254;  /* units are millimeters */
       FORMAT EVAP_TOT WIND_TOT 6.1  EVAP1-EVAP31 4.1
              WIND1-WIND31 3.;
       WIND_TOT = WIND_TOT * 1.61;
       %IF &DAILY %THEN
       DO E = 1 TO 31 ;
          multiply daily wind values by 1.61;
       END;
   %END;

   multiply EVAP_TOT by the value of ADJUST; %IF &DAILY %THEN
       DO E = 1 TO 31;
          multiply daily evaporation values by the value of ADJUST;
       DROP STATE STAT_NUM ADJUST E;

   %ELSE
       DROP E EVAP1-EVAP31 WIND1-WIND31  E_CODE1-E_CODE31
              W_CODE1-W_CODE31 STATE STAT_NUM ADJUST;

   STATE = SUBSTR(STATION,1,2);
   STAT_NUM = SUBSTR(STATION,3,4);
   STN_FORM = TRIM(STATE) !! '-' !! STAT_NUM;
   OUTPUT;
RUN;
```

4.33 Description of Module CONVERT RESERVOIR DATA (CV_RESV)

4.33.1 Processing Narrative

CONVERT RESERVOIR DATA converts the daily values by the proper factor, according to the metric option. Output formats are assigned, and a formatted version of the STATION variable is calculated. Also, the mean, maximum and minimum values for the month are calculated.

4.33.2 Interface Description

A) There is no parameter list.

B) Global data:

```
Macro variables:
  METRIC: identifies units for the data

SAS data sets:
  RESVDATA (input) - with variables:
     STATION  YEAR  MONTH  UNITCODE  TIMECODE
     NAME  COUNTY  AREA  ELEV  STORGE1-STORGE31
  RESV (output) - with variables:
     same as RESVDATA, plus: STN_FORM MAX_RESV MIN_RESV
     AVE_RESV
```

C) There are no macros called by this module.

D) No permanent files are used by this module.

E) The module generates one complete DATA step.

4.33.3 Internal Data Description

There are no local macro variables used in this module.

```
SAS variables:
     ADJUST: value by which to adjust data
      STATE: first two digits of station number
   STAT_NUM: middle 4 digits of station number
          S: control variable for DO loop
   STAT_END: last 2 digits of station number
```

4.33.4 Design Language Description

```
DATA RESV;

   ARRAY STORGE[31] STORGE1-STORGE31;
   FORMAT STORGE1-STORGE31 8.2;

   SET RESVDATA;

   N_MISS = number of missing values in the month;
   IF MONTH has 30 days
      THEN N_MISS = N_MISS - 1;
   ELSE IF MONTH = 2 THEN
           IF MOD(YEAR,4) NE 0 THEN
                N_MISS = N_MISS - 3;
           ELSE N_MISS = N_MISS - 2;
```

```
%IF &METRIC = 0 %THEN
    IF UNITCODE NE 'A' THEN
       DO S = 1 TO 31;
          multiply daily values by 0.01; /* units are feet */
       END;

%ELSE                    /* need to convert to metric units */
    IF UNITCODE = 'A' THEN
         ADJUST = 1233.5 ;    /* units are cubic meters */
    ELSE ADJUST = 0.00305;    /* units are meters */

    DO S = 1 TO 31;
       multiply daily values by ADJUST;
    END;
%END;

IF N_MISS _ 1  THEN
   MAX_RESV = maximum monthly value;
   MIN_RESV = minimum monthly value;
ELSE
   set MAX_RESV and MIN_RESV to missing;

IF N_MISS _ 6 THEN
   IF UNITCODE NE 'A' THEN
        AVE_RESV = mean monthly value;
   ELSE AVE_RESV = month end contents;
ELSE
   set AVE_RESV to missing;

DROP STATE STAT_NUM STAT_END S ADJUST N_MISS;

STATE = SUBSTR(STATION,1,2);
STAT_NUM = SUBSTR(STATION,3,4);
STAT_END = SUBSTR(STATION,7,2);
STN_FORM = TRIM(STATE) !! '.' !! TRIM(STAT_NUM) !! '.' !! STAT_END;
OUTPUT;
```

4.34 Description of Module CONVERT SNOW COURSE DATA (CV_SNOC)

4.34.1 Processing Narrative

CONVERT SNOW COURSE DATA converts the snow depth and water content values to metric units, if necessary. Output formats are assigned.

4.34.2 Interface Description

A) There is no parameter list.

B) Global data:
   Macro variables:
     METRIC: identifies units for the data

   SAS data sets:
     SNOCDATA (input) - with variables:
        STATE STATION YEAR CARDNO NAME COUNTY AREA ELEV
        MONTH1-MONTH6 DAY1-DAY6 DEPTH1-DEPTH6 WATER1-WATER6
     SNOC (output) - with variables:
        same as SNOCDATA

C) There are no macros called by this module.

D) No permanent files are used by this module.

E) The module generates one complete DATA step.

4.34.3 Internal Data Description

There are no local macro variables used in this module.

SAS variables:
   READING (num): identifies which monthly reading, control variable

4.34.4 Design Language Description

```
DATA SNOC;

   ARRAY DEPTH[6] DEPTH1-DEPTH6;
   ARRAY WATER[6] WATER1-WATER6;
   ARRAY MONTH[6] MONTH1-MONTH6;
   ARRAY DAY[6]   DAY1-DAY6;
   FORMAT DEPTH1-DEPTH6 4. WATER1-WATER6 4.1 ;

   SET SNOCDATA;
   %IF &METRIC NE 0 %THEN
      DO READING = 1 TO 6;
       multiply DEPTH array by 2.54; /* units are centimeters */
       multiply WATTH array by 25.4; /* units are millimeters */
   %END;
   OUTPUT;  RUN;
```

4.35 Description of Module CONVERT PEAK FLOW DATA (CV_PEAK)

4.35.1 Processing Narrative

CONVERT PEAK FLOW DATA converts the gage height values by the proper factor, according to the metric option. Output formats are

assigned, and a formatted version of the STATION variable is calculated. A SAS date value for the month, day and year of the annual occurrence is calculated.

4.35.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:
METRIC: identifies units for the data

SAS data sets:
PEAKDATA (input) - with variables:
STATION WAT_YEAR YEAR MONTH DAY GAGE_HT PEAKFLOW
GH_CODE PK_CODE RD_CODE NAME COUNTY AREA ELEV
PEAK (output) - with variables:
same as PEAKDATA, plus: STN_FORM MONDAYYR

C) There are no macros called by this module.

D) No permanent files are used by this module.

E) The module generates one complete DATA step.

4.35.3 Internal Data Description

There are no local macro variables used in this module.

SAS variables:
STATE: first two digits of station number
STAT_NUM: middle 4 digits of station number
STAT_END: last 2 digits of station number

4.35.4 Design Language Description

```
DATA PEAK;

   FORMAT PEAKFLOW 8.  GAGE_HT 8.2  MONDAYYR DATE9. ;

   SET PEAKDATA;

   %IF &METRIC = 0 %THEN
       multiply GAGE_HT by 0.01; /* units are feet */
   %ELSE
       multiply GAGE_HT  by 0.00305; /* units are meters */
       multiply PEAKFLOW by 0.02832; /* units are cubic meters/s */
```

```
   MONDAYYR = MDY(MONTH,DAY,YEAR);  /* calculate SAS date value */

   DROP STATE STAT_NUM STAT_END  ;

   STATE = SUBSTR(STATION,1,2);
   STAT_NUM = SUBSTR(STATION,3,4);
   STAT_END = SUBSTR(STATION,7,2);
   STN_FORM =
       TRIM(STATE) !! '.' !! TRIM(STAT_NUM) !! '.' !! STAT_END;
   OUTPUT;
 RUN;
```

4.36 Description of Module CONVERT MONTHLY SUMMARY DATA (CV_MNTH)

4.36.1 Processing Narrative

CONVERT MONTHLY SUMMARY DATA converts the monthly summary data values to metric units, if necessary. Output formats are assigned here, according to the metric option. A formatted version of the STATION variable is also calculated.

4.36.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:
- METRIC: identifies units for the data

SAS data sets:
- MNTHDATA (input) - with variables:
  STATION YEAR MONTH NAME COUNTY AREA ELEV
  MAXTEMP MINTEMP TMEAN TDEPART DEGDAYS HIGHEST HIGHDAY
  LOWEST LOWDAY PRECIP PDEPART PPTMAX PMAXDAY SNOFALL
  SNODEPTH SMAXDAY WIND EVAP COOLDAYS PLUS1 PLUS2 PLUS3
  PLUS4 PLUS5
- MNTH (output) - with variables:
  same as MNTHDATA, plus: STN_FORM

C) There are no macros called by this module.

D) No permanent files are used by this module.

E) The module generates one complete DATA step.

4.36.3 Internal Data Description

There are no local macro variables used in this module.

SAS variables:
STATE: first two digits of station number
STAT_NUM: last 4 digits of station number

4.36.4 Design Language Description

```
DATA MNTH;

   SET MNTHDATA;

   FORMAT MAXTEMP MINTEMP TMEAN TDEPART SNOFALL  6.1;
   FORMAT HIGHEST LOWEST DEGDAYS COOLDAYS SNODEPTH WIND 6.;

   %IF &METRIC = 0 %THEN
       FORMAT PRECIP  PPTMAX  PDEPART  EVAP  6.2;
   %ELSE
     %* convert data values to metric units;
       FORMAT PRECIP  PPTMAX  PDEPART  EVAP  6.1;
       MAXTEMP = (MAXTEMP - 32) * 5 / 9;
       MINTEMP = (MINTEMP - 32) * 5 / 9;
       TMEAN = (TMEAN - 32) * 5 / 9;
       TDEPART = TDEPART * 5 / 9;
       DEGDAYS = DEGDAYS * 5 / 9;
       HIGHEST = (HIGHEST - 32) * 5 / 9;
       LOWEST = (LOWEST - 32) * 5 / 9;
       COOLDAYS = COOLDAYS * 5 / 9;
       PRECIP = PRECIP * 25.4;
       PDEPART = PDEPART * 25.4;
       PPTMAX = PPTMAX * 25.4;
       SNOFALL = SNOFALL * 2.54;
       SNODEPTH = SNODEPTH * 2.54;
       EVAP = EVAP * 25.4;
       WIND = WIND * 1.61;
   %END;

   DROP STATE STAT_NUM ;

   STATE = SUBSTR(STATION,1,2);
   STAT_NUM = SUBSTR(STATION,3,4);
   STN_FORM = TRIM(STATE) !! '-' !! STAT_NUM;
   OUTPUT;
RUN;
```

4.37 Description of Module CONVERT HOURLY PRECIPITATION DATA (CV_HPCP)

4.37.1 Processing Narrative

CONVERT HOURLY PRECIPITATION DATA converts the hourly values and/or daily total by the proper factor, according to the metric option. If not needed, hourly values are dropped. Output formats are assigned here, according to the metric option. A formatted version of the STATION variable is also calculated.

4.37.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:
- METRIC : identifies units for the data
- HOURLY : identifies need to retrieve hourly values

SAS data sets:
- HPCPDATA (input) - with variables:
  STATION YEAR MONTH DAY HPCP_TOT DAYSUM NAME
  COUNTY AREA ELEV HPCP1-HPCP24 H_CODE1-H_CODE24
- HPCP (output) - with variables:
  same as HPCPDATA, plus: STN_FORM

C) There are no macros called by this module.

D) No permanent files are used by this module.

E) The module generates one complete DATA step.

4.37.3 Internal Data Description

There are no local macro variables used in this module.

SAS variables:
- ADJUST: value by which to adjust data
- P: control variable for DO loop
- STATE: first two digits of station number
- STAT_NUM: last 4 digits of station number

4.37.4 Design Language Description

```
DATA HPCP;

   ARRAY HPCP[24] HPCP1-HPCP24;
```

```
    %IF &METRIC = 0 %THEN
        RETAIN ADJUST  0.01;    /* units are inches */
        FORMAT HPCP_TOT 6.2 HPCP1-HPCP24 6.2;
    %ELSE
        RETAIN ADJUST  0.254;  /* units are millimeters */
        FORMAT HPCP_TOT 6.1 HPCP1-HPCP24 6.1;

    SET HPCPDATA;
    multiply  HPCP_TOT by the value of ADJUST;
    %IF request for hourly values %THEN
        DO P = 1 TO 24;
           multiply hourly values by the value of ADJUST;
        DROP STATE STAT_NUM ADJUST P;
    %ELSE
        DROP P  HPCP1-HPCP24
                H_CODE1-H_CODE24 STATE STAT_NUM ADJUST;

    STATE = SUBSTR(STATION,1,2);
    STAT_NUM = SUBSTR(STATION,3,4);
    STN_FORM = TRIM(STATE) !! '-' !! STAT_NUM;
    OUTPUT;
RUN;
```

4.38 Description of Module LIST (LISTT)

4.38.1 Processing Narrative

LIST controls the generation of output for the list command.

4.38.2 Interface Description

A) There is no parameter list.

B) Global data:
Macro variables:
L_INDEX: identifies request for list index operation
L_PTRS : identifies request for list pointers operation
L_DAILY: identifies request for list daily operation
L_MONTH: identifies request for list monthly operation
L_CONT : identifies request for list contents operation
L_HOUR : identifies request for list hourly operation

No SAS data sets are created or used.

C) The macros called by this module are:

LIST INDEX (4.39 - LIST_IX)
LIST POINTERS (4.47 - LIST_PT)
LIST DAILY (4.49 - LIST_DY)
LIST MONTHLY (4.61 - LIST_MN)
LIST CONTENTS (4.69 - LIST_CT)
LIST HOURLY (4.74 - LIST_HR)

D) No permanent files are used by this module.

E) The module generates macro statements only.

4.38.3 Internal Data Description

No local data are used in this module.

4.38.4 Design Language Description

```
%IF request for list index operation %THEN
    %LIST INDEX
%IF request for list pointers operation %THEN
    %LIST POINTERS
%IF request for list daily operation %THEN
    %LIST DAILY
```

```
%IF request for list monthly operation %THEN
    %LIST MONTHLY
%IF request for list contents operation %THEN
    %LIST CONTENTS
%IF request for list hourly operation %THEN
    %LIST HOURLY
```

4.39 Description of Module LIST INDEX (LIST_IX)

4.39.1 Processing Narrative

LIST INDEX checks the flags for each element and calls the macro to generate index listings for that element.

4.39.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:

- EL_PRCP: identifies request for precipitation data
- EL_TEMP: identifies request for temperature data
- EL_STRM: identifies request for streamflow data
- EL_SNOW: identifies request for snowfall data
- EL_EVAP: identifies request for evaporation data
- EL_SNOC: identifies request for snow course data
- EL_PEAK: identifies request for peakflow data
- EL_MNTH: identifies request for monthly summary data
- EL_RESV: identifies request for reservoir data
- EL_HPCP: identifies request for hourly precipitation data

No SAS data sets are created or used.

C) The macros called by this module are:

LIST INDEX FOR ONE ELEMENT (4.40 - LI_ELEM)

D) No permanent files are used by this module.

E) The module generates macro statements only.

4.39.3 Internal Data Description

No local data are used in this module.

4.39.4 Design Language Description

```
%IF request for precipitation element %THEN
    %LI_ELEM (PRCP,PRECIPITATION)
%IF request for temperature element %THEN
    %LI_ELEM (TEMP,TEMPERATURE)
%IF request for streamflow element %THEN
    %LI_ELEM (STRM,STREAMFLOW)
%IF request for snowfall element %THEN
    %LI_ELEM (SNOW,SNOWFALL)
%IF request for evaporation element %THEN
    %LI_ELEM (EVAP,EVAPORATION)
%IF request for snow course element %THEN
    %LI_ELEM (SNOC,SNOW COURSE)
%IF request for peakflow element %THEN
    %LI_ELEM (PEAK,PEAKFLOW)
%IF request for monthly element %THEN
    %LI_ELEM (MNTH,MONTHLY SUMMARY)
%IF request for reservoir element %THEN
    %LI_ELEM (RESV,RESERVOIR)
%IF request for hourly precipitation element %THEN
    %LI_ELEM (HPCP,HOURLY PRECIPITATION)
```

4.40 Description of Module LIST INDEX FOR ONE ELEMENT (LI_ELEM)

4.40.1 Processing Narrative

LIST INDEX FOR ONE ELEMENT calls the macros which create subsets of the index file and generate an index listing for one element. In the macro, it is determined whether the user wants a listing of NHIMS stations, non-NHIMS stations, or all types of stations in the index. Also, if the user requests a listing of the NHIMS periods of record, the proper macros are called to find those periods of record and correctly list them. The decision is also made as to whether or not the user wants a listing of the complete index file for an element, or simply the listings for specified stations. The subset created for the macro which prints the output depends on what type of index listing the user requests.

4.40.2 Interface Description

A) Parameters: ELEMENT - 4-character element name
E_TITLE - full name of element for output heading

B) Global data:

Macro variables:
LI_PER : identifies request for NHIMS periods of record
L_COMP : identifies request for complete file listing
MAIN_DS: identifies need to retrieve main data set info
LI_ALL : identifies request for all types of stations
LI_NHIM: identifies request for NHIMS stations only
LI_NONM: identifies request for non-NHIMS stations only

No SAS data sets are created or used.

C) The macros called by this module are:

FIND POINTERS (4.24 - FINDPTR)
CHECK NUMBER OF OBSERVATIONS (4.20 - NUM_OBS)
FIND PERIODS OF RECORD (4.41 - FINDPER)
MERGE DATA SET WITH INDEX (4.42 - IX_MRGE)
PRINT INDEX (4.43 - LI_PRT)
LACK OF INDEX-DATA ERROR (4.45 - LI_ERR)
MAKE SUBSET OF INDEX (4.46 - INX_SUB)

D) No permanent files are used by this module.

E) The module generates macro statements only.

4.40.3 Internal Data Description

No local data are used in this module.

4.40.4 Design Language Description

```
%IF request for list index with NHIMS periods of record %THEN
   %IF NOT request for complete file listing %THEN
      %IF NOT need to retrieve main data set info %THEN
         %DO;
            %FIND POINTERS (&ELEMENT,&ELEMENT..POINTERS)
         %END;
      %CHECK NUMBER OF OBSERVATIONS(&ELEMENT.PTRS)
      %IF number of observations is greater than 0 %THEN
          %FIND PERIODS OF RECORD (&ELEMENT,DSN=&ELEMENT.PTRS)
          %MERGE DATA SET WITH INDEX
                   (DSN=&ELEMENT.PER,SUBIF=AND F_NHIMS='Y')
          %PRINT INDEX (&ELEMENT,&E_TITLE,DSN=REQ_IX)
      %ELSE
          %LACK OF INDEX-DATA ERROR (&E_TITLE)
   %ELSE %IF request for complete file listing %THEN
      %FIND PERIODS OF RECORD (&ELEMENT,DSN=&ELEMENT..POINTERS)
      %MERGE DATA SET WITH INDEX
               (DSN=&ELEMENT.PER,SUBIF=AND F_NHIMS='Y')
      %PRINT INDEX (&ELEMENT,&E_TITLE,DSN=REQ_IX)
```

```
%ELSE %IF not request for NHIMS periods of record %THEN
    %IF request for index listing for all stations %THEN
        %IF NOT a request for complete file listing %THEN
           %MERGE DATA SET WITH INDEX
                 (DSN=STATIONS)
           %CHECK NUMBER OF OBSERVATIONS (REQ_IX)
           %IF number of observations ¢ than 0 %THEN
               %PRINT INDEX (&ELEMENT,&E_TITLE,DSN=REQ_IX)

           %ELSE
               %LACK OF INDEX-DATA ERROR (&E_TITLE)
        %ELSE %IF request for complete file listing %THEN
           %PRINT INDEX (&ELEMENT,&E_TITLE,DSN=ddname.INDEX)

    %ELSE %IF request for index listing of NHIMS stations %THEN
        %IF NOT a request for complete file listing %THEN
           %MERGE DATA SET WITH INDEX
                  (DSN=STATIONS,SUBIF=AND F_NHIMS='Y')
           %CHECK NUMBER OF OBSERVATIONS (REQ_IX)
           %IF number of observations ¢ 0 %THEN
               %PRINT INDEX (&ELEMENT,&E_TITLE,DSN=REQ_IX)
           %ELSE
               %LACK OF INDEX-DATA ERROR (&E_TITLE)

        %ELSE %IF request for complete file listing %THEN
           %MAKE SUBSET OF INDEX (Y)
           %PRINT INDEX (&ELEMENT,&E_TITLE,DSN=SUB_IX)

    %ELSE %IF request for index listing of non-NHIMS stations
    %THEN
        %IF NOT a request for complete file listing %THEN
           %MERGE DATA SET WITH INDEX
                  (DSN=STATIONS,SUBIF=AND F_NHIMS='N')
           %CHECK NUMBER OF OBSERVATIONS (REQ_IX)
           %IF number of observations ¢ 0 %THEN
               %PRINT INDEX (&ELEMENT,&E_TITLE,DSN=REQ_IX)
           %ELSE
               %LACK OF INDEX-DATA ERROR (&E_TITLE)

        %ELSE %IF request for complete file listing %THEN
           %MAKE SUBSET OF INDEX (N)
           %PRINT INDEX (&ELEMENT,&E_TITLE,DSN=SUB_IX)
```

4.41 Description of Module FIND PERIODS OF RECORD (FINDPER)

4.41.1 Processing Narrative

FIND PERIODS OF RECORD will search a permanent data file for the beginning and ending periods of record for one or more stations. Either a temporary or permanent pointer file is used to find the first and last observation numbers.

4.41.2 Interface Description

A) Parameters: ELEMENT - 4-character element name
DSN - data set name of pointer file

B) Global data:

No global macro variables are used by this module.

SAS data sets:
&ELEMENT.PTRS (input) - any temporary pointer file, with variables:
STATION FOBS_PTR LOBS_PTR
&ELEMENT.PER (output) - with variables:
STATION FIRST_YR FIRST_MN LAST_YR LAST_MN MLENGTH

C) No macros are called by this module.

D) Permanent files: any of the MAINDATA files
any of the POINTERS files

E) The module generates one complete DATA step.

4.41.3 Internal Data Description

No local data are used in this module.

4.41.4 Design Language Description

```
DATA &ELEMENT.PER (KEEP = STATION FIRST_YR FIRST_MN MLENGTH
                          LAST_YR LAST_MN);
     SET &DSN;
     SET &ELEMENT..MAINDATA POINT=FOBS_PTR;
     FIRST_YR = YEAR;
     FIRST_MN = MONTH;

     SET &ELEMENT..MAINDATA POINT=LOBS_PTR;
     LAST_YR = YEAR;
     LAST_MN - MONTH;
     MLENGTH = LOBS_PTR - FOBS_PTR + 1;
     OUTPUT;
 RUN;
```

4.42 Description of Module MERGE DATA SET WITH INDEX (IX_MRGE)

4.42.1 Processing Narrative

MERGE DATA SET WITH INDEX performs a merge operation with a SAS data set and the permanent index data set, by the variable STATION. Only observations with stations in both data sets are kept; also, a parameter allows the calling routine to specify an additional qualification to be placed in the subsetting IF statement.

4.42.2 Interface Description

A) Parameters: DSN - name of a SAS data set
SUBIF - value for the subsetting IF statement

B) Global data:

No global macro variables are used by this module.

SAS data sets:
Any SAS data set (input) - with variable:
STATION
REQ_IX (output) - with variables:
all variables from the permanent INDEX data set and the input data set

C) No macros are called by this module.

D) Permanent files: ddname.INDEX

E) The module generates one complete DATA step.

4.42.3 Internal Data Description

No local macro variables are used in this module.

SAS variables:
LEGAL (boolean): identifies data from permanent INDEX file
REQUEST(boolean): identifies data from other input data set

4.42.4 Design Language Description

```
DATA REQ_IX;
   MERGE &DSN (IN=REQUEST)  ddname.INDEX(IN=LEGAL);
      BY STATION;

   IF LEGAL AND REQUEST &SUBIF;
```

4.43 Description of Module PRINT INDEX (LI_PRT)

4.43.1 Processing Narrative

PRINT INDEX prints the requested index listing information, with or without NHIMS periods of record.

4.43.2 Interface Description

A) Parameters: ELEMENT - 4-character element name
E_TITLE - full name of element for output listing
DSN - name of data set used as input

B) Global data:

Macro variables:
LI_PER : identifies request for NHIMS periods of record
LI_ALL : identifies request for all types of stations
LI_NHIM: identifies request for NHIMS stations only
LI_NONM: identifies request for non-NHIMS stations only

SAS data sets:
Any SAS data set (input) - that is a subset of the permanent INDEX file
PRINT (output) - standard SAS print file

C) Macros are called by this module:
PRINT COMMON INDEX LINES (4.44 - LI_LINE)

D) Permanent files: ddname.INDEX

E) The module generates one complete _NULL_ DATA step.

4.43.3 Internal Data Description

No local macro variables are used in this module.

4.43.4 Design Language Description

```
DATA _NULL_;
   FILE PRINT HEADER=NEWPAGE  NOTITLES  LINESLEFT=LEFT;
   RETAIN PAGE_NUM 1  ANYDATA 0;
```

```
SET &DSN  END=EOF;
%IF request for NHIMS periods of record or NHIMS stations
%THEN
    IF F_&ELEMENT = 'Y' THEN DO;

%ELSE %IF request for all types of stations %THEN
    IF F_&ELEMENT = 'Y' OR F_NHIMS = 'N' THEN DO;

%ELSE %IF request for non-NHIMS stations %THEN
    DO;
%END;
      ANYDATA = 1;

      %PRINT COMMON INDEX LINES

      %IF request for NHIMS periods of record %THEN

          /* list NHIMS periods of record */
          PUT 'NHIMS COMPUTER RECORDS AVAILABLE FROM '
               FIRST_MN  2. '-' FIRST_YR  ' TO '
               LAST_MN   2. '-' LAST_YR
               '     LENGTH IN MONTHS  ' MLENGTH;

      %ELSE
          IF F_NHIMS = 'N' THEN
             PUT 'NO NHIMS COMPUTER RECORDS AVAILABLE.';

   /* list periods of record for printed records */
   PUT 'WRITTEN RECORDS AVAILABLE FROM ' RECBEG $7.  ' TO '
        RECEND $7. '.';
   PUT 'SUCH DATA IS AVAILABLE ONLY ON MICROFICHE OR '
       'PRINTED LISTINGS.';
END;

IF EOF AND ANYDATA = 0 THEN
   issue error message that no requested stations were
   found in the &E_TITLE index.

IF LEFT _ 9 THEN
   PUT;
   %IF request for NHIMS periods of record %THEN
      PUT '** WARNING **   DETERMINING THE PERIODS OF RECORD'
          'CAN BE EXPENSIVE. CONSULT USER''S MANUAL FOR LIST'
        'INDEX COMMANDS WHICH WILL EXCLUDE SUCH INFORMATION.'
   %ELSE %IF request for non-NHIMS stations %THEN
      PUT '** WARNING **   STATIONS LISTED HERE MAY OR MAY '
          'NOT HAVE RECORDS FOR THE SPECIFIED ELEMENT.';
```

```
      PUT _PAGE_;
      increment PAGE_NUM by 1;
      RETURN;

   NEWPAGE:
      PUT "&E_TITLE STATIONS" @90 'PAGE ' PAGE_NUM ;
   RETURN;
```

4.44 Description of Module PRINT COMMON INDEX LINES (LI_LINE)

4.44.1 Processing Narrative

PRINT COMMON INDEX LINES writes the lines for the index listings that are common to all listings. A formatted version of the station number is also produced.

4.44.2 Interface Description

A) There is no parameter list.

B) Global data:

No global macro variables are used in this module.

No SAS data sets are created or used.

C) No macros are called by this module.

D) Permanent files: NHIMS.FORMATS

E) The module generates only a portion of a single DATA step. SAS variables shared with the calling module are:

```
STATION (char): station code number
NAME    (char): textual name of station
COUNTY  (char): name of county in which station is located
LAT      (num): latitude of station
LONG     (num): longitude of station
ELEV     (num): elevation of station
DIVISION (num): division in which station is located
HUCODE   (num): hydrologic unit code
```

4.44.3 Internal Data Description

No local macro variables are used in this module.

SAS variables:
```
STATE    (char): first two digits of station code number
STAT_NUM (char): middle four digits of station code number
HOWLONG   (num): length of station number
STAT_END (char): last two digits of 8-character code number
STN_FORM (char): formatted station number
```

4.44.4 Design Language Description

```
/**** NEED TO CREATE FORMATTED STATION NUMBER ****/

LENGTH  STN_FORM $ 10;
STATE = SUBSTR(STATION,1,2);
STAT_NUM = SUBSTR(STATION,3,4);
HOWLONG = LENGTH(STATION);
IF length of the station number equals 6 THEN
   STN_FORM = TRIM(STATE) !! '-' !! STAT_NUM;

ELSE IF length of the station number equals 8 THEN
   STAT_END = SUBSTR(STATION,7,2);
   STN_FORM=TRIM(STATE) !! '.' !! TRIM(STAT_NUM) !!
            '.' !! STAT_END;

ELSE STN_FORM = STATION;

/* LATFORM, LONGFORM, and HUCFORM formats are user-created, */
/* and must be declared previously with PROC FORMAT         */

PUT @2 NAME   @60 COUNTY   @75 'STATION NO. ' @88 STN_FORM;

PUT @10 'LATITUDE'    @20 LAT  LATFORM.
    @35 'LONGITUDE'   @46 LONG LONGFORM.;

PUT @10 'ELEVATION'   @21 ELEV   @30 'FT MSL'
    @41 'DIVISION'    @52 DIVISION Z2.
    @60 'HUCODE'      @72 HUCODE  HUCFORM.;
```

4.45 Description of Module LACK OF INDEX-DATA ERROR (LI_ERR)

4.45.1 Processing Narrative

LACK OF INDEX-DATA ERROR generates an error message whenever the data set to be used in the PRINT INDEX macro is empty.

4.45.2 Interface Description

A) Parameter: E_TITLE - full name of the element

B) Global data:

No global macro variables are used in this module.

SAS data sets:
PRINT (output) - standard SAS print file

C) No macros are called by this module.

D) No permanent files are used.

E) The module generates a single, complete DATA step.

4.45.3 Internal Data Description

No local data are used in this module.

4.45.4 Design Language Description

```
DATA _NULL_;

   FILE PRINT;

   PUT 'THE REQUESTED STATION(S) WERE NOT FOUND IN THE '
       "&E_TITLE INDEX.";
RUN;
```

4.46 Description of Module MAKE SUBSET OF INDEX (INX_SUB)

4.46.1 Processing Narrative

MAKE SUBSET OF INDEX creates a subset of the permanent index file, which contains either all of the NHIMS stations or all of the non-NHIMS stations, depending on the value passed to the macro.

4.46.2 Interface Description

A) Parameter: VALUE - value of the F_NHIMS variable

B) Global data:

No global macro variables are used in this module.

SAS data sets:
SUB_IX (output): subset of permanent index file

C) No macros are called by this module.

D) Permanent files: ddname.INDEX

E) The module generates a single, complete DATA step.

4.46.3 Internal Data Description

No local data are used in this module.

4.46.4 Design Language Description

```
DATA SUB_IX;
   SET ddname.INDEX;
   IF F_NHIMS = "&VALUE";
RUN;
```

4.47 Description of Module LIST POINTERS (LIST_PT)

4.47.1 Processing Narrative

LIST POINTERS checks the element flags; for each requested element, a macro is called to print the listing of the pointers.

4.47.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:

```
EL_PRCP : identifies request for precipitation data
EL_TEMP : identifies request for temperature data
EL_STRM : identifies request for streamflow data
EL_SNOW : identifies request for snowfall data
EL_EVAP : identifies request for evaporation data
EL_PEAK : identifies request for peakflow data
EL_SNOC : identifies request for snow course data
EL_MNTH : identifies request for monthly data
EL_RESV : identifies request for reservoir data
EL_HPCP : identifies request for hourly precipitation data
L_COMP  : identifies request for complete file listing
```

No temporary SAS data sets are created or used.

C) Macros called by this module are:

%FIND POINTERS (4.24 - FINDPTR)
%PRINT POINTER DATA (4.48 - LP_DATA)

D) No permanent files are used.

E) The module generates macro statements only.

4.47.3 Internal Data Description

No local data are used in this module.

4.47.4 Design Language Description

```
%IF request for precipitation element %THEN
    %IF NOT request for complete file listing %THEN
        %FIND POINTERS (PRCP.POINTERS)
        %PRINT POINTER DATA (PRECIPITATION,DSN=PRCPPTRS)
    %ELSE
        %PRINT POINTER DATA (PRECIPITATION,DSN=PRCP.POINTERS)

%IF request for temperature element %THEN
    %IF NOT request for complete file listing %THEN
        %FIND POINTERS (TEMP.POINTERS)
        %PRINT POINTER DATA (TEMPERATURE,DSN=TEMPPTRS)
    %ELSE
        %PRINT POINTER DATA (TEMPERATURE,DSN=TEMP.POINTERS)

%IF request for streamflow element %THEN
    %IF NOT request for complete file listing %THEN
        %FIND POINTERS (STRM.POINTERS)
        %PRINT POINTER DATA (STREAMFLOW,DSN=STRMPTRS)
    %ELSE
        %PRINT POINTER DATA (STREAMFLOW,DSN=STRM.POINTERS)

%IF request for snowfall element %THEN
    %IF NOT request for complete file listing %THEN
        %FIND POINTERS (SNOW.POINTERS)
        %PRINT POINTER DATA (SNOWFALL,DSN=SNOWPTRS)
    %ELSE
        %PRINT POINTER DATA (SNOWFALL,DSN=SNOW.POINTERS)

%IF request for evaporation element %THEN
    %IF NOT request for complete file listing %THEN
        %FIND POINTERS (EVAP.POINTERS)
        %PRINT POINTER DATA (EVAPORATION,DSN=EVAPPTRS)
    %ELSE
        %PRINT POINTER DATA (EVAPORATION,DSN=EVAP.POINTERS)

%IF request for peakflow element %THEN
    %IF NOT request for complete file listing %THEN
        %FIND POINTERS (PEAK.POINTERS)
        %PRINT POINTER DATA (PEAKFLOW,DSN=PEAKPTRS)
```

```
    %ELSE
        %PRINT POINTER DATA (PEAKFLOW,DSN=PEAK.POINTERS)

%IF request for snow course element %THEN
    %IF NOT request for complete file listing %THEN
        %FIND POINTERS (SNOC.POINTERS)
        %PRINT POINTER DATA (SNOW COURSE,DSN=SNOCPTRS)
    %ELSE
        %PRINT POINTER DATA (SNOC COURSE,DSN=SNOC.POINTERS)

%IF request for monthly summary element %THEN
    %IF NOT request for complete file listing %THEN
        %FIND POINTERS (MNTH.POINTERS)
        %PRINT POINTER DATA (MONTHLY SUMMARY,DSN=MNTHPTRS)
    %ELSE
        %PRINT POINTER DATA (MONTHLY SUMMARY,DSN=MNTH.POINTERS)

%IF request for reservoir element %THEN
    %IF NOT request for complete file listing %THEN
        %FIND POINTERS (RESV.POINTERS)
        %PRINT POINTER DATA (RESERVOIR,DSN=RESVPTRS)
    %ELSE
        %PRINT POINTER DATA (RESERVOIR,DSN=RESV.POINTERS)

%IF request for hourly precipitation element %THEN
    %IF NOT request for complete file listing %THEN
        %FIND POINTERS (HPCP.POINTERS)
        %PRINT POINTER DATA (HOURLY PRECIPITATION,DSN=HPCPPTRS)
    %ELSE
        %PRINT POINTER DATA (HOURLY PRECIPITATION,
                             DSN=HPCP.POINTERS)
```

4.48 Description of Module PRINT POINTER DATA (LP_DATA)

4.48.1 Processing Narrative

PRINT POINTER DATA prints a listing of station numbers with the corresponding range of observation numbers from a pointer file.

4.48.2 Interface Description

A) Parameter list: E_TITLE - element name used in heading
DSN (keyword) - name of input data set

B) Global data:

No global macro variables are used in this module.

SAS data sets:
&ELEMENT.PTRS (input) - with variables:
STATION FOBS_PTR LOBS_PTR
PRINT (output) - standard SAS print file

C) No macros are called by this module.

D) Permanent files: &ELEMENT..POINTERS

E) The module generates one complete _NULL_ DATA step.

4.48.3 Internal Data Description

No local macro variables are used in this module.

SAS variables:
LEFT : the number of lines left on the current page
NEWPAGE : label for the statements which print page headings
ROW : current row number for printing
PAGE_NUM: counter for the current page number

4.48.4 Design Language Description

```
DATA _NULL_;

  SET &DSN;
  FILE PRINT  LINESLEFT=LEFT  HEADER = NEWPAGE;
  RETAIN ROW 7  PAGE_NUM 1;
  PUT #ROW @5   STATION
           @15  FOBS_PTR
           @35  LOBS_PTR;

  increment ROW by 1;
  IF LEFT _ 7  THEN   /* go to a new page */
     PUT _PAGE_;
     ROW = 7;
     increment PAGE_NUM by 1;
  RETURN;

  NEWPAGE: PUT #1 @65 'PAGE '  PAGE_NUM;
           PUT #3 @5 "POINTER LISTING FOR &E_TITLE FILE";
           PUT #5 @5  'STATION'
                  @15 'FIRST OBSERVATION'
                  @35 'LAST OBSERVATION';
           RETURN;
 RUN;
```

4.49 Description of Module LIST DAILY (LIST_DY)

4.49.1 Processing Narrative

LIST DAILY calls the macros to print the listings for daily values.

4.49.2 Interface Description

A) There is no parameter list.

B) Global data:
No temporary SAS data sets are created or used.
Macro variables:
EL_PRCP : identifies request for precipitation data
EL_TEMP : identifies request for temperature data
EL_STRM : identifies request for streamflow data
EL_SNOW : identifies request for snowfall data
EL_EVAP : identifies request for evaporation data
EL_RESV : identifies request for reservoir data
L_FIRST : number of the month in which to begin listing

C) Macros are called by this module are:
LIST DAILY PRECIPITATION DATA (4.50 - LD_PRCP)
LIST DAILY TEMPERATURE DATA (4.51 - LD_TEMP)
LIST DAILY STREAMFLOW DATA (4.52 - LD_STRM)
LIST DAILY SNOWFALL DATA (4.53 -LD_SNOW)
LIST DAILY EVAPORATION DATA (4.54 - LD_EVAP)
LIST DAILY RESERVOIR DATA (4.55 - LD_RESV)
LIST DAILY DATA FOR HOURLY PRECIPITATION (4.56 - LD_HPCP)

D) No permanent files are used.
E) The module generates macro statements only.

4.49.3 Internal Data Description

No local data are used in this module.

4.49.4 Design Language Description

```
%IF request for precipitation element %THEN
    %LIST DAILY PRECIPITATION DATA (&L_FIRST)
%IF request for temperature element %THEN
    %LIST DAILY TEMPERATURE DATA (&L_FIRST)
%IF request for streamflow element %THEN
    %LIST DAILY STREAMFLOW DATA (&L_FIRST)
%IF request for snowfall element %THEN
    %LIST DAILY SNOWFALL DATA (&L_FIRST)
%IF request for evaporation element %THEN
    %LIST DAILY EVAPORATION DATA (&L_FIRST)
%IF request for reservoir element %THEN
    %LIST DAILY RESERVOIR DATA (&L_FIRST)
%IF request for hourly precipitation element %THEN
    %LIST DAILY DATA FOR HOURLY PRECIPITATION (&L_FIRST)
```

4.50 Description of Module LIST DAILY PRECIPITATION DATA (LD_PRCP)

4.50.1 Processing Narrative

LIST DAILY PRECIPITATION DATA takes the subset of the precipitation file and generates a listing of the daily data values. Corrections are made for invalid values of the variable MONTH, and daily codes as well as explanatory footnotes are printed.

4.50.2 Interface Description

A) Parameter: FIRSTMON - month in which to begin listing
(keyword - default is 1)

B) Global data:
Macro variables:
METRIC : identifies request for metric units

SAS data sets:
PRCP (input) - with variables:
STATION YEAR MONTH PRECIP1-PRECIP31 P_CODE1-P_CODE31
PRCP_TOT MONTHSUM STN_FORM NAME COUNTY AREA ELEV
PRINT (output) - standard SAS print file

C) Macros are called by this module are:
PRINT MONTH NAMES IN HEADING (4.57 - MONHEAD)
PRINT DAY NUMBERS (4.58 - DAY_PRT)
PRINT STATION INFO IN HEADING (4.59 - ST_HEAD)
CALCULATE COLUMN FOR OUTPUT (4.60 - CAL_COL)

D) No permanent files are used.

E) The module generates one complete _NULL_ DATA step.

4.50.3 Internal Data Description

No local macro variables are used in this module.

SAS variables:

PAGE_NUM	(num):	current page number for heading
MOVE	(num):	constant used to position data values
ADJUSTDN	(num):	adjusts high value of MONTH to correct column
ADJUSTUP	(num):	adjusts low value of MONTH to correct column
MAXDATE	(num):	maximum date (year+month) for one page
DATENOW	(num):	current date (year+month) of observation
CORRECT	(num):	used to correct illegal month values
ROW	(num):	identifies current row for printing
COLUMN	(num):	identifies current column for printing
NEW_COL	(num):	identifies column for printing daily code

```
DAY        (num): control variable for DO loop
EOF    (boolean): identifies last record read from input file
CHK_DATE   (num): SAS date value - month, day, year
POSITION   (num): identifies number of columns printed, 1-12
LAST.STATION (boolean): identifies last occcurence of station
```

4.50.4 Design Language Description

```
DATA _NULL_;
  RETAIN PAGE_NUM 1 MOVE -1;
  ARRAY PRECIP[31] PRECIP1-PRECIP31;
  ARRAY P_CODE[31] P_CODE1-P_CODE31;
  FILE PRINT N=PS NOTITLES;

  %IF &FIRSTMON = 1 %THEN
        ADJUSTDN = 0;
        ADJUSTUP = 0;
  %ELSE
        ADJUSTDN = &FIRSTMON - 1;
        ADJUSTUP = 12 - ADJUSTDN;

  MAXDATE = 999999;
  CORRECT = &FIRSTMON;
  DO UNTIL(POSITION ¢= 12); /* print 12 months per page */
     SET PRCP END=EOF;
       BY STATION;
     DATENOW = (YEAR * 100) + MONTH;
     IF DATENOW ¢ MAXDATE  THEN  /* exceeded date for the page */
                                 /* Begin new page             */
        PUT _PAGE_;
        PAGE_NUM = PAGE_NUM + 1;
        CORRECT = &FIRSTMON;

     IF CORRECT = &FIRSTMON THEN
       DO;          /* beginning new page, so calculate */
                    /* maximum date for page, and print headings */
        IF &FIRSTMON = 1  THEN
           MAXDATE = (YEAR * 100) + 12;
        ELSE IF MONTH ¢= &FIRSTMON  THEN
                MAXDATE = (YEAR + 1) * 100 + ADJUSTDN;
             ELSE
                MAXDATE = (YEAR * 100) + ADJUSTDN;

        PUT #1 @110 'PAGE ' PAGE_NUM;
        PUT #49 @1 'TOTAL';
        PUT #52 @1 'ZERO VALUES ARE PRINTED AS BLANKS.';
        PUT #53 @1 'M:  MISSING DAILY VALUE.';
        PUT #54 @1 'T:  TRACE AMOUNT.';
        PUT #55 @1 'A:  DAILY VALUE WAS ACCUMULATED.';
        PUT #56 @1 'E:  DAILY VALUE WAS ESTIMATED.';
```

```
   PUT #57 @1 'B:  DAILY VALUE WAS BOTH ACCUMULATED AND '
              'ESTIMATED.';
   PUT #59 @1 '*** CAUTION ***    MUCH OF THE DATA BEFORE'
              ' 1970 IS NOT FLAGGED FOR SPECIAL CONDITIONS.'


   %PRINT MONTH NAMES IN HEADING(FIRST=&FIRSTMON)

   %IF &METRIC = 0 %THEN
         PUT #6 @45
               'DAILY PRECIPITATION IN INCHES';
   %ELSE
         PUT #6 @40
         'DAILY PRECIPITATION IN MILLIMETERS';

   %PRINT DAY NUMBERS(FIRSTROW=9)

   %PRINT STATION INFO IN HEADING(FIRST=&FIRSTMON)
END; /* beginning new page, calculate and print headings */

%CALCULATE COLUMN FOR OUTPUT(FIRST=&FIRSTMON)

ROW = 9;
DO DAY = 1 TO 31;   /* print one column (month) of data */
   ROW = ROW + 1;
   IF PRECIP[DAY] ¢ 0 THEN
      DO;      /* PRINT  NON-ZERO VALUES */
        PUT #ROW @COLUMN PRECIP[DAY] P_CODE[DAY] $CHAR1.;
      END;

   ELSE IF MONTHSUM NE 0 THEN
      DO;      /* print code for special condition */
        NEW_COL = COLUMN + 4;
        IF PRECIP[DAY] = . THEN
           DO;   /* check for end-of-month missing values */
             CHK_DATE = MDY(MONTH,DAY,YEAR);
             IF CHK_DATE NE . THEN
                  /* print code for missing value */
                  PUT #ROW @NEW_COL 'M';
           END;
        ELSE
           /* print trace, estimated or accumulated code */
           PUT #ROW @NEW_COL P_CODE[DAY] $CHAR1.;

    IF MOD(DAY,5) = 0 THEN  ROW = ROW + 1;
END;  /* print one column of data */

PUT #49 @COLUMN PRCP_TOT;
```

```
      IF EOF THEN STOP;
      IF LAST.STATION  THEN POSITION = 12;
      IF POSITION ¢= 12 THEN    /* go to a new page */
          PUT _PAGE_;
          increment PAGE_NUM by 1;
      CORRECT = CORRECT + 1;
      IF CORRECT ¢ 12 THEN CORRECT = 1;
  END;  /* print 12 months of data */
```

4.51 Description of Module LIST DAILY TEMPERATURE DATA (LD_TEMP)

4.51.1 Processing Narrative

LIST DAILY TEMPERATURE DATA takes the subset of the temperature file and generates a listing of the daily data values. Corrections are made for invalid values of the variable MONTH, and explanatory footnotes are printed.

4.51.2 Interface Description

A) Parameter: FIRSTMON - month in which to begin listing
(keyword - default is 1)

B) Global data:
Macro variables:
METRIC : identifies request for metric units

SAS data sets:
TEMP (input) - with variables:
STATION YEAR MONTH MAXTMP1-MAXTMP31
MINTMP1-MINTMP31 MXCODE1-MXCODE31 MNCODE1=MNCODE31
MONTHSUM STN_FORM NAME COUNTY AREA ELEV
PRINT (output) - standard SAS print file

C) Macros are called by this module are:
PRINT MONTH NAMES IN HEADING (4.57 - MONHEAD)
PRINT DAY NUMBERS (4.58 - DAY_PRT)
PRINT STATION INFO IN HEADING (4.59 - ST_HEAD)
CALCULATE COLUMN FOR OUTPUT (4.60 - CAL_COL)

D) No permanent files are used.

E) The module generates one complete _NULL_ DATA step.

4.51.3 Internal Data Description

No local macro variables are used in this module.

```
SAS variables:
   PAGE_NUM  (num): current page number for heading
   MOVE      (num): constant used to position data values
   ADJUSTDN  (num): adjusts high value of MONTH to correct column
   ADJUSTUP  (num): adjusts low value of MONTH to correct column
   MAXDATE   (num): maximum date (year+month) for one page
   DATENOW   (num): current date (year+month) of observation
   CORRECT   (num): used to correct illegal month values
   ROW       (num): identifies current row for printing
   COLUMN    (num): identifies current column for printing
   DAY       (num): array subscript for day values 1-31
   NEW_COL   (num): identifies column for printing daily code
   EOF   (boolean): identifies last record read from input file
   POSITION  (num): identifies number of columns printed, 1-12
   LAST.STATION (boolean): identifies last occurrence of current
                           value of STATION
```

4.51.4 Design Language Description

```
DATA _NULL_;
  RETAIN PAGE_NUM 1 MOVE -1;
  ARRAY MAXTMP[31] MAXTMP1-MAXTMP31;
  ARRAY MINTMP[31] MINTMP1-MINTMP31;
  ARRAY MXCODE[31] $ MXCODE1-MXCODE31;
  ARRAY MNCODE[31] $ MNCODE1-MNCODE31;

  FILE PRINT N=PS NOTITLES;

  %IF &FIRSTMON = 1 %THEN
      %DO;
         ADJUSTDN = 0;
         ADJUSTUP = 0;
      %END;
  %ELSE
      %DO;
         ADJUSTDN = &FIRSTMON - 1;
         ADJUSTUP = 12 - ADJUSTDN;
      %END;

  MAXDATE = 999999;
  CORRECT = &FIRSTMON;
  DO UNTIL(POSITION ¢=12);  /* print 12 months of data */
     SET TEMP END=EOF;
       BY STATION ;
     DATENOW = (YEAR * 100) + MONTH;
     IF DATENOW ¢ MAXDATE  THEN /* exceeded date for the page, */
                                /* begin new page              */
        PUT _PAGE_;
        PAGE_NUM = PAGE_NUM + 1;
        CORRECT = &FIRSTMON;
```

```
IF CORRECT = &FIRSTMON THEN
   DO;          /* beginning of new page, so calculate max */
                /* date for the page, and print headings   */
     IF &FIRSTMON = 1  THEN MAXDATE = (YEAR * 100) + 12;
     ELSE IF MONTH ¢= &FIRSTMON  THEN
             MAXDATE = (YEAR + 1) * 100 + ADJUSTDN;
          ELSE
             MAXDATE = (YEAR * 100) + ADJUSTDN;

     PUT #1 @110 'PAGE ' PAGE_NUM;
     PUT #49 @1 'AVEMAX';
     PUT #50 @1 'AVEMIN';
     PUT #54 @1 'MISSING DAILY VALUES ARE PRINTED '
                'AS BLANKS.';
     PUT #55 @1 'E: DAILY VALUE IS ESTIMATED.';


     %PRINT MONTH NAMES IN HEADING(FIRST=&FIRSTMON)

     %IF &METRIC = 0 %THEN
        %DO;
           PUT #6 @35
           'DAILY MAXIMUM AND MINIMUM TEMPERATURES '
           'IN DEGREES F';
        %END;
     %ELSE
        %DO;
           PUT #6 @35
           'DAILY MAXIMUM AND MINIMUM TEMPERATURES '
           'IN DEGREES C';
        %END;

     %PRINT DAY NUMBERS(FIRSTROW=9);

     %PRINT STATION INFO IN HEADING(FIRST=&FIRSTMON)
   END;  /* beginning of a new page */

%CALCULATE COLUMN FOR OUTPUT(FIRST=&FIRSTMON)

ROW = 9;
 DO DAY = 1 TO 31;    /* print one column (month) of data */
    ROW = ROW + 1;
    PUT #ROW @COLUMN MAXTMP[DAY] MXCODE[DAY] $CHAR1.
                     MINTMP[DAY] MNCODE[DAY] $CHAR1.;

   IF MOD(DAY,5) = 0 THEN
       DO;
         ROW = ROW + 1;
       END;
 END;
```

```
        PUT #49 @COLUMN AVEMAX  ;
        PUT #50 @COLUMN AVEMIN  ;

        IF EOF THEN STOP;
        IF LAST.STATION  THEN  POSITION = 12;
        IF POSITION ¢= 12 THEN
          DO;  /* go to a new page */
            PUT _PAGE_;
            PAGE_NUM = PAGE_NUM + 1;
          END;

        CORRECT = CORRECT + 1;
        IF CORRECT ¢ 12 THEN CORRECT = 1;

    END; /* print 12 months of data */
RUN;
```

4.52 Description of Module LIST DAILY STREAMFLOW DATA (LD_STRM)

4.52.1 Processing Narrative

LIST DAILY STREAMFLOW DATA takes the subset of the streamflow file and generates a listing of the daily data values. Corrections are made for invalid values of the variable MONTH, and explanatory footnotes are printed. Formats are assigned depending on the magnitude of the daily values.

4.52.2 Interface Description

A) Parameter: FIRSTMON - month in which to begin listing
(keyword - default is 10)

B) Global data:
Macro variables:
METRIC : identifies request for metric units

SAS data sets:
STRM (input) - with variables:
STATION YEAR MONTH STREAM1-STREAM31 MONTHSUM
MAX_STRM MIN_STRM AVE_STRM STRM_TOT
STN_FORM NAME COUNTY AREA ELEV
PRINT (output) - standard SAS prrint file

C) Macros are called by this module are:
PRINT MONTH NAMES IN HEADING (4.57 - MONHEAD)
PRINT DAY NUMBERS (4.58 - DAY_PRT)
PRINT STATION INFO IN HEADING (4.59 - ST_HEAD)
CALCULATE COLUMN FOR OUTPUT (4.60 - CAL_COL)

D) No permanent files are used.

E) The module generates one complete _NULL_ DATA step.

4.52.3 Internal Data Description

No local macro variables are used in this module.

SAS variables:

```
PAGE_NUM  (num): current page number for heading
MOVE      (num): constant used to position data values
ADJUSTDN  (num): adjusts high value of MONTH to correct column
ADJUSTUP  (num): adjusts low value of MONTH to correct column
CORRECT   (num): used to correct illegal month values
MAXDATE   (num): maximum date (year+month) for one page
DATENOW   (num): current date (year+month) of observation
ROW       (num): identifies current row for printing
COLUMN    (num): identifies current column for printing
DAY       (num): control variable for DO loop
EOF   (boolean): identifies last record read from input file
POSITION  (num): identifies number of columns printed, 1-12
LAST.STATION (boolean): identifies last occurrence of current
                        value of STATION
```

4.52.4 Design Language Description

```
DATA _NULL_;
  RETAIN PAGE_NUM 1 MOVE -1;
  ARRAY FLOW[31] FLOW1-FLOW31;
  FILE PRINT N=PS NOTITLES;

  %IF &FIRSTMON = 1 %THEN
        ADJUSTDN = 0;
        ADJUSTUP = 0;
  %ELSE
        ADJUSTDN = &FIRSTMON - 1;
        ADJUSTUP = 12 - ADJUSTDN;

  MAXDATE = 999999;
  CORRECT = &FIRSTMON;
  DO UNTIL(POSITION ¢= 12); /* print 12 months per page */
     SET STRM END=EOF;
       BY STATION ;
     DATENOW = (YEAR * 100) + MONTH;
     IF DATENOW ¢ MAXDATE  THEN  /* exceeded date for the page, */
                                 /* begin new page             */
        PUT _PAGE_;
        PAGE_NUM = PAGW_NUM + 1;
        CORRECT = &FIRSTMON;
```

```
IF CORRECT = &FIRSTMON THEN
  DO;       /* beginning of a new page, so calculate max */
            /* date for the page, and print headings     */
    IF &FIRSTMON = 1  THEN
       MAXDATE = (YEAR * 100) + 12;
    ELSE IF MONTH ¢= &FIRSTMON  THEN
            MAXDATE = (YEAR + 1) * 100 + ADJUSTDN;
         ELSE
            MAXDATE = (YEAR * 100) + ADJUSTDN;

    PUT #1 @110 'PAGE ' PAGE_NUM;
    PUT #49 @1 'TOTAL';
    PUT #50 @1 'MAX';
    PUT #51 @1 'MIN';
    PUT #52 @1 'MEAN';
    PUT #55 @1 'MISSING DAILY VALUES ARE PRINTED AS BLANKS.';

    PUT #56 @1 'NO MEAN IS COMPUTED IF MORE THAN '
               '5 DAYS ARE MISSING.';
    PUT #57 @1 'THE MAXIMUM AND MINIMUM ARE NOT COMPUTED '
               'IF THERE ARE ANY MISSING DAYS IN THE MONTH.';

    %PRINT MONTH NAMES IN HEADING(FIRST=&FIRSTMON)

    %IF &METRIC = 0 %THEN
          PUT #6 @32
                'MEAN DAILY STREAMFLOW IN CUBIC FEET '
                'PER SECOND';
    %ELSE
          PUT #6 @32
                'MEAN DAILY STREAMFLOW IN CUBIC METERS '
                'PER SECOND';

    %PRINT DAY NUMBERS(FIRSTROW=9)
    %PRINT STATION INFO IN HEADING(FIRST=&FIRSTMON)
  END;  /* beginning of new page  */

%CALCULATE COLUMN FOR OUTPUT(FIRST=&FIRSTMON)

ROW = 9;
DO DAY = 1 TO 31;/* print one column (month) of data */
   ROW = ROW + 1;
   IF FLOW[DAY] ¢ 99 THEN
       PUT #ROW @COLUMN FLOW[DAY] 7. ;
   ELSE IF FLOW[DAY] ¢ 9 THEN
       PUT #ROW @COLUMN FLOW[DAY] 9.1 ;
   ELSE
       PUT #ROW @COLUMN FLOW[DAY] 10.2;

   IF MOD(DAY,5) = 0 THEN  ROW = ROW + 1;
END;      /* print one column (month) of data */
```

```
   IF AVE_STRM ¢ 99 THEN
        PUT #49 @COLUMN STRM_TOT 7. ;
        PUT #50 @COLUMN MAX_STRM 7. ;
        PUT #51 @COLUMN MIN_STRM 7. ;
        PUT #52 @COLUMN AVE_STRM 7. ;
   ELSE IF AVE_STRM ¢ 9 THEN
        PUT #49 @COLUMN STRM_TOT 7. ;
        PUT #50 @COLUMN MAX_STRM 9.1;
        PUT #51 @COLUMN MIN_STRM 9.1;
        PUT #52 @COLUMN AVE_STRM 9.1;
   ELSE
        PUT #49 @COLUMN STRM_TOT 9.1;
        PUT #50 @COLUMN MAX_STRM 10.2;
        PUT #51 @COLUMN MIN_STRM 10.2;
        PUT #52 @COLUMN AVE_STRM 10.2;

   IF EOF THEN STOP;
   IF LAST.STATION  THEN  POSITION = 12;
   IF POSITION ¢= 12 THEN  /* go to a new page */
       PUT _PAGE_;
       PAGE_NUM = PAGE_NUM + 1;
   CORRECT = CORRECT + 1;
   IF CORRECT ¢ 12 THEN CORRECT = 1;
END; /* print 12 months of data */
```

4.53 Description of Module LIST DAILY SNOWFALL DATA (LD_SNOW)

4.53.1 Processing Narrative

LIST DAILY SNOWFALL DATA takes the subset of the snowfall file and generates a listing of the daily data values. Corrections are made for invalid values of the variable MONTH, and explanatory footnotes are printed.

4.53.2 Interface Description

A) Parameter: FIRSTMON - month in which to begin listing
(keyword - default is 7)

B) Global data:

Macro variables:
METRIC : identifies request for metric units

SAS data sets:
SNOW (input) - with variables:
STATION YEAR MONTH MONTHSUM SNOW_TOT
SNOW1-SNOW31 DEPTH1-DEPTH31 S_CODE1-S_CODE31
D_CODE1-D_CODE31 STN_FORM NAME COUNTY AREA ELEV
PRINT (output) - standard SAS print file

C) Macros are called by this module are:
PRINT MONTH NAMES IN HEADING (4.57 - MONHEAD)
PRINT DAY NUMBERS (4.58 - DAY_PRT)
PRINT STATION INFO IN HEADING (4.59 - ST_HEAD)
CALCULATE COLUMN FOR OUTPUT (4.60 - CAL_COL)

D) No permanent files are used.

E) The module generates one complete _NULL_ DATA step.

4.53.3 Internal Data Description

No local macro variables are used in this module.

SAS variables:

PAGE_NUM	(num):	current page number for heading
MOVE	(num):	constant used to position data values
ADJUSTDN	(num):	adjusts high value of MONTH to correct column
ADJUSTUP	(num):	adjusts low value of MONTH to correct column
MAXDATE	(num):	maximum date (year+month) for one page
DATENOW	(num):	current date (year+month) of observation
CORRECT	(num):	used to correct illegal month values
ROW	(num):	identifies current row for printing

```
COLUMN     (num): identifies column for printing snowfall value
DEP_COL    (num): identifies column for printing snow depth
TOT_COL    (num): identifies column for printing monthly total
DAY        (num): control variable for DO loop
CHK_DATE   (num): SAS date value - month, day, year
EOF    (boolean): identifies last record read from input file
POSITION   (num): identifies number of columns printed, 1-12
LAST.STATION (boolean): identifies last occurrence of current
                        value of STATION
```

4.53.4 Design Language Description

```
DATA _NULL_;
  RETAIN PAGE_NUM 1 MOVE -1;
  ARRAY SNOW[31] SNOW1-SNOW31;
  ARRAY DEPTH[31] DEPTH1-DEPTH31;
  ARRAY S_CODE[31] S_CODE1-S_CODE31;
  ARRAY D_CODE[31] D_CODE1-D_CODE31;
  FILE PRINT N=PS NOTITLES;

  %IF &FIRSTMON = 1 %THEN
      %DO;
         ADJUSTDN = 0;
         ADJUSTUP = 0;
      %END;
  %ELSE
      %DO;
         ADJUSTDN = &FIRSTMON - 1;
         ADJUSTUP = 12 - ADJUSTDN;
      %END;

  MAXDATE = 999999;
  CORRECT = &FIRSTMON;
  DO UNTIL(POSITION ¢= 12); /* print 12 months per page */
     SET SNOW END=EOF;
       BY STATION;

     DATENOW = (YEAR * 100) + MONTH;
     IF DATENOW ¢ MAXDATE  THEN  /* exceeded max date for page */
                                 /* Go to a new page           */
        PUT _PAGE_;
        PAGE_NUM = PAGE_NUM + 1;
        CORRECT = &FIRSTMON;

     IF CORRECT = &FIRSTMON THEN
           /* begin printing new page, so calculate maximum */
           /* date for page, and print headings             */
        IF &FIRSTMON = 1  THEN  MAXDATE = (YEAR * 100) + 12;
        ELSE IF MONTH ¢= &FIRSTMON  THEN
              MAXDATE = (YEAR + 1) * 100 + ADJUSTDN;
```

```
      ELSE
         MAXDATE = (YEAR * 100) + ADJUSTDN;

   PUT #1 @110 'PAGE ' PAGE_NUM;
   PUT #49 @1 'TOTAL';
   PUT #52 @1 'ZERO VALUES ARE PRINTED AS BLANKS.';
   PUT #53 @1 'M:  MISSING DAILY VALUE.';
   PUT #54 @1 'T:  TRACE AMOUNT.';
   PUT #55 @1 'A:  DAILY VALUE WAS ACCUMULATED.';
   PUT #56 @1 'E:  DAILY VALUE WAS ESTIMATED.';
   PUT #57 @1
   'B:  DAILY VALUE WAS BOTH ACCUMULATED AND ESTIMATED.';

   %PRINT MONTH NAMES IN HEADING(FIRST=&FIRSTMON)

   %IF &METRIC = 0 %THEN
       PUT #6 @32
       'DAILY SNOWFALL AND SNOW DEPTH ON GROUND IN INCHES';
   %ELSE
       PUT #6 @27
       'DAILY SNOWFALL AND SNOW DEPTH ON GROUND '
       'IN CENTIMETERS';

   %PRINT DAY NUMBERS(FIRSTROW=9)

   %PRINT STATION INFO IN HEADING(FIRST=&FIRSTMON)

END;  /* first time in loop, print headings */

%CALCULATE COLUMN FOR OUTPUT(FIRST=&FIRSTMON)

ROW = 9;
DO DAY = 1 TO 31;   /* print one column (month) of data */
   ROW = ROW + 1;

   IF (MONTHSUM NE 0) OR (SNOW[DAY] NE 0) OR
      (DEPTH[DAY] NE 0) THEN
      DO;  /* write out values for one day */

      /* check for end-of-month missing values */
         CHK_DATE = MDY(MONTH,DAY,YEAR);
         IF CHK_DATE NE . THEN
            DO;   /* legal day - print values */
              IF SNOW[DAY] = . THEN S_CODE[DAY] = 'M';
              IF DEPTH[DAY] = . THEN D_CODE[DAY] = 'M';
              DEP_COL = COLUMN + 7;
              PUT #ROW @COLUMN SNOW[DAY] S_CODE[DAY] $CHAR1.
                   @DEP_COL DEPTH[DAY] D_CODE[DAY] $CHAR1.;
            END;  /* legal day - print values */
       END;  /* write out values for one day*/
```

```
            IF MOD(DAY,5) = 0 THEN
                 ROW = ROW + 1;
        END;  /*  print one column (month) of data */

    TOT_COL = COLUMN + 3;
    PUT #49 @TOT_COL  SNOW_TOT ;

    IF EOF THEN STOP;
    IF LAST.STATION  THEN POSITION = 12;
    IF POSITION ¢= 12 THEN
       DO;                    /* go to a new page */
         PUT _PAGE_;
         PAGE_NUM = PAGE_NUM + 1;
       END;

    CORRECT = CORRECT + 1;
    IF CORRECT ¢ 12 THEN CORRECT = 1;

 END; /* print 12 months of data */
```

4.54 Description of Module LIST DAILY EVAPORATION DATA (LD_EVAP)

4.54.1 Processing Narrative

LIST DAILY EVAPORATION DATA takes the subset of the evaporation file and generates a listing of the daily data values. Corrections are made for invalid values of the variable MONTH, and explanatory footnotes are printed.

4.54.2 Interface Description

A) Parameter: FIRSTMON - month in which to begin listing
(keyword - default is 1)

B) Global data:
Macro variables:
METRIC : identifies request for metric units

SAS data sets:
EVAP (input) - with variables:
STATION YEAR MONTH MONTHSUM EVAP_TOT WIND_TOT
EVAP1-EVAP31 WIND1-WIND31 E_CODE1-E_CODE31
W_CODE1-W_CODE31 STN_FORM NAME COUNTY AREA ELEV
PRINT (output) - standard SAS print file

C) Macros are called by this module are:
PRINT MONTH NAMES IN HEADING (4.57 - MONHEAD)
PRINT DAY NUMBERS (4.58 - DAY_PRT)

PRINT STATION INFO IN HEADING (4.59 - ST_HEAD)
CALCULATE COLUMN FOR OUTPUT (4.60 - CAL_COL)

D) No permanent files are used.

E) The module generates one complete _NULL_ DATA step.

4.54.3 Internal Data Description

No local macro variables are used in this module.

SAS variables:

PAGE_NUM	(num):	current page number for heading
MOVE	(num):	constant used to position data values
ADJUSTDN	(num):	adjusts high value of MONTH to correct column
ADJUSTUP	(num):	adjusts low value of MONTH to correct column
CORRECT	(num):	used to correct illegal month values
MAXDATE	(num):	maximum date (year+month) for one page
DATENOW	(num):	current date (year+month) of observation
ROW	(num):	identifies current row for printing
COLUMN	(num):	identifies current column for printing
WIND_COL	(num):	identifies column for wind movement code
TOT_COL	(num):	identifies column for monthly totals
DAY	(num):	control variable for DO loop
CHK_DATE	(num):	SAS date value - month, day, year
EOF	(boolean):	identifies last record read from input file
POSITION	(num):	identifies number of columns printed, 1-12
LAST.STATION	(boolean):	identifies last occurrence of current value of STATION

4.54.4 Design Language Description

```
DATA _NULL_;
  RETAIN PAGE_NUM 1 MOVE -1;
  ARRAY EVAP[31] EVAP1-EVAP31;
  ARRAY WIND[31] WIND1-WIND31;
  ARRAY E_CODE[31] E_CODE1-E_CODE31;
  ARRAY W_CODE[31] W_CODE1-W_CODE31;
  FILE PRINT N=PS NOTITLES;

  %IF &FIRSTMON = 1 %THEN
      %DO;
         ADJUSTDN = 0;
         ADJUSTUP = 0;
      %END;
  %ELSE
      %DO;
         ADJUSTDN = &FIRSTMON - 1;
         ADJUSTUP = 12 - ADJUSTDN;
      %END;
```

```
MAXDATE = 999999;
CORRECT = &FIRSTMON;
DO UNTIL(POSITION ¢= 12);  /* print 12 months per page */
   SET EVAP END=EOF;
     BY STATION ;
   DATENOW = (YEAR * 100) + MONTH;
   IF DATENOW ¢ MAXDATE  THEN /* exceeded max date for the */
                              /* page, begin new page      */
      PUT _PAGE_;
      PAGE_NUM = PAGE_NUM + 1;
      CORRECT = &FIRSTMON;

   IF CORRECT = &FIRSTMON THEN
      DO;     /* beginning of new page, so calculate max  */
              /* date for  page, and print page headings */
        IF &FIRSTMON = 1  THEN
           MAXDATE = (YEAR * 100) + 12;
        ELSE IF MONTH ¢= &FIRSTMON THEN
                MAXDATE = (YEAR + 1) * 100 + ADJUSTDN;
             ELSE
                MAXDATE = (YEAR * 100) + ADJUSTDN;

        PUT #1 @110 'PAGE ' PAGE_NUM;
        PUT #49 @1 'TOTAL';
        PUT #50 @1 'TOTAL';
        PUT #53 @1 'ZERO VALUES ARE PRINTED AS BLANKS.';
        PUT #54 @1 'M:  MISSING DAILY VALUE.';
        PUT #55 @1 'A:  DAILY VALUE WAS ACCUMULATED.';
        PUT #56 @1 'E:  DAILY VALUE WAS ESTIMATED.';
        PUT #57 @1
            'B:  DAILY VALUE WAS BOTH ACCUMULATED '
            'AND ESTIMATED.';

        %PRINT MONTH NAMES IN HEADING(FIRST=&FIRSTMON)

        %IF &METRIC = 0 %THEN
           PUT #6 @27
           'DAILY EVAPORATION IN INCHES AND '
           'WIND MOVEMENT IN MILES';
        %ELSE
              PUT #6 @20
              'DAILY EVAPORATION IN MILLIMETERS AND '
              'WIND MOVEMENT IN KILOMETERS';

        %PRINT DAY NUMBERS(FIRSTROW=9)

        %PRINT STATION INFO IN HEADING(FIRST=&FIRSTMON)
      END;  /* beginning of a new page */

   %CALCULATE COLUMN FOR OUTPUT(FIRST=&FIRSTMON)
```

```
      ROW = 9;
      DO DAY = 1 TO 31;    /* print one column (month) of data */
         ROW = ROW + 1;
         WIND_COL = COLUMN + 6;

         IF MONTHSUM NE 0 THEN  /* print out special conditions */
            DO;
              CHK_DATE = MDY(MONTH,DAY,YEAR);
              IF CHK_DATE NE . THEN
                 DO;   /* legal day - print values */
                   IF EVAP[DAY] = .  THEN E_CODE[DAY] = 'M';
                   IF WIND[DAY] = .  THEN W_CODE[DAY] = 'M';
                   PUT #ROW @COLUMN EVAP[DAY]  E_CODE[DAY] $CHAR1.
                          @WIND_COL WIND[DAY]  W_CODE[DAY] $CHAR1.;
                 END; /* legal day - print values */
            END;  /* print out special conditions */

         ELSE
            IF EVAP[DAY] ¢ 0 THEN
               DO;   /* print out non-zero values */
                 PUT #ROW @COLUMN EVAP[DAY]  E_CODE[DAY] $CHAR1.
                       @WIND_COL  WIND[DAY]  W_CODE[DAY] $CHAR1.;
               END;

         ELSE IF WIND[DAY] ¢ 0 THEN
            DO;   /* print out non-zero values */
              PUT #ROW @WIND_COL WIND[DAY]  W_CODE[DAY] $CHAR1.;
            END;

            IF MOD(DAY,5) = 0 THEN
                 ROW = ROW + 1;

      END;  /* print one column (month) of data */

      TOT_COL = COLUMN + 3;
      PUT #49 @TOT_COL EVAP_TOT ;
      PUT #50 @TOT_COL WIND_TOT ;

      IF EOF THEN STOP;
      IF LAST.STATION  THEN POSITION = 12;
      IF POSITION ¢= 12 THEN
         DO;                  /* go to a new page */
           PUT _PAGE_;
           PAGE_NUM = PAGE_NUM + 1;
         END;

     CORRECT = CORRECT + 1;
     IF CORRECT ¢ 12 THEN CORRECT = 1;

   END; /* print 12 months of data */
RUN;
```

4.55 Description of Module LIST DAILY RESERVOIR DATA (LD_RESV)

4.55.1 Processing Narrative

LIST DAILY RESERVOIR DATA takes the subset of the reservoir file and generates a listing of the daily data values. Corrections are made for invalid values of MONTH, and explanatory footnotes are printed.

4.55.2 Interface Description

A) Parameter: FIRSTMON - month in which to begin listing
(keyword - default is 10)

B) Global data:
Macro variables:
METRIC : identifies request for metric units

SAS data sets:
RESV (input) - with variables:
STATION YEAR MONTH MONTHSUM UNITCODE TIMECODE
STORGE1-STORGE31 MAX_RESV MIN_RESV AVE_RESV
STN_FORM NAME COUNTY AREA ELEV
PRINT (output) - standard SAS print file

C) Macros are called by this module are:
PRINT MONTH NAMES IN HEADING (4.57 - MONHEAD)
PRINT DAY NUMBERS (4.58 - DAY_PRT)
PRINT STATION INFO IN HEADING (4.59 - ST_HEAD)
CALCULATE COLUMN FOR OUTPUT (4.60 - CAL_COL)

D) No permanent files are used.

E) The module generates one complete _NULL_ DATA step.

4.55.3 Internal Data Description

No local macro variables are used in this module.

SAS variables:

PAGE_NUM	(num):	current page number for heading
MOVE	(num):	constant used to position data values
ADJUSTDN	(num):	adjusts high value of MONTH to correct column
ADJUSTUP	(num):	adjusts low value of MONTH to correct column
CORRECT	(num):	used to correct illegal month values
MAXDATE	(num):	maximum date (year+month) for one page
DATENOW	(num):	current date (year+month) of observation
ROW	(num):	identifies current row for printing
COLUMN	(num):	identifies current column for printing

```
DAY        (num): control variable for DO loop
EOF    (boolean): identifies last record read from input file
POSITION   (num): identifies number of columns printed, 1-12
LAST.STATION (boolean): identifies last occurrence of current
                        value of STATION
```

4.55.4 Design Language Description

```
DATA _NULL_;
  RETAIN PAGE_NUM 1 MOVE -1;
  ARRAY STORGE[31] STORGE1-STORGE31;
  FILE PRINT N=PS NOTITLES;

  %IF &FIRSTMON = 1 %THEN
      %DO;
         ADJUSTDN = 0;
         ADJUSTUP = 0;
      %END;
  %ELSE
      %DO;
         ADJUSTDN = &FIRSTMON - 1;
         ADJUSTUP = 12 - ADJUSTDN;
      %END;

  MAXDATE = 999999;
  CORRECT = &FIRSTMON;
  DO UNTIL(POSITION ¢= 12); /* print 12 months per page */
     SET RESV END=EOF;
       BY STATION;
     DATENOW = (YEAR * 100) + MONTH;
     IF DATENOW ¢ MAXDATE  THEN /* exceeded max date for the */
                                /* page, begin new page      */
        PUT _PAGE_;
        PAGE_NUM = PAGE_NUM + 1;
        CORRECT = &FIRSTMON;

     IF CORRECT = &FIRSTMON THEN
       DO;         /* beginning of a new page, so calculate max */
                   /* date for new page, and print headings     */
         IF &FIRSTMON = 1  THEN
            MAXDATE = (YEAR * 100) + 12;
         ELSE IF MONTH ¢= &FIRSTMON  THEN
                 MAXDATE = (YEAR + 1) * 100 + ADJUSTDN;
              ELSE
                 MAXDATE = (YEAR * 100) + ADJUSTDN;

         PUT #1 @110 'PAGE ' PAGE_NUM;
         PUT #49 @1 'MEAN';
         PUT #50 @1 'MAX';
         PUT #51 @1 'MIN';
```

```
PUT #53 @1 'MISSING DAILY VALUES ARE PRINTED AS BLANKS.';
PUT #55 @1 'FOR LISTINGS OF CONTENTS IN ACRE-FEET, '
           'THE MEAN MONTHLY VALUE IS THE '
           'MONTH END CONTENTS.';
PUT #57 @1 'FOR ELEVATION AND GAGE HEIGHT LISTINGS, '
           'NO MEAN IS COMPUTED IF MORE THAN '
           '5 DAYS ARE MISSING.';
PUT #59 @1 'THE MAXIMUM AND MINIMUM ARE NOT COMPUTED IF '
           'THERE ARE ANY MISSING DAYS IN THE MONTH.';

%PRINT MONTH NAMES IN HEADING(FIRST=&FIRSTMON)

%PRINT DAY NUMBERS(FIRSTROW=9)

%PRINT STATION INFO IN HEADING(FIRST=&FIRSTMON)

%IF &METRIC = 0 %THEN
   %DO;  /*  print headings for english units */
      IF UNITCODE = 'A' THEN
         PUT #6 @25
             'RESERVOIR CONTENTS IN ACRE-FEET,  ' @;
      ELSE IF UNITCODE = 'E' THEN
         PUT #6 @25
        'RESERVOIR ELEVATION IN FEET '
        'ABOVE SEA LEVEL,  ' @;
      ELSE
         PUT #6 @25 'GAGE HEIGHT IN FEET,  '  @;
   %END;
%ELSE
   %DO;   /* print headings for metric units */
      IF UNITCODE = 'A' THEN
         PUT #6 @25
             'RESERVOIR CONTENTS IN CUBIC METERS,  ' @;
      ELSE IF UNITCODE = 'E' THEN
         PUT #6 @25
        'RESERVOIR ELEVATION IN METERS '
        'ABOVE SEA LEVEL,  ' @;
      ELSE
         PUT #6 @25 'GAGE HEIGHT IN METERS,  '  @;
   /* END OF PRINT HEADINGS FOR METRIC UNITS */
   %END;

IF TIMECODE = 3 THEN
     PUT 'MEAN DAILY VALUE'  ;
ELSE IF TIMECODE = 4 THEN
     PUT 'AM READING'  ;
ELSE IF TIMECODE = 5 THEN
     PUT 'PM READING'  ;
ELSE IF TIMECODE = 11 THEN
     PUT 'ONCE DAILY READING AT RANDOM HOURS';
```

```
    ELSE
         PUT 'AT ' TIMECODE 'HOURS' ;

END;  /* beginning of a new page */

%CALCULATE COLUMN FOR OUTPUT(FIRST=&FIRSTMON)

IF UNITCODE = 'A' THEN
    /* format values for acre-feet(meters3), no decimals */
     ROW = 9;
     DO DAY = 1 TO 31;/* print one column (month) of data */
        ROW = ROW + 1;
        PUT #ROW @COLUMN STORGE[DAY] 8. ;

        IF MOD(DAY,5) = 0 THEN
           DO;
             ROW = ROW + 1;
           END;
     END;       /* print one column (month) of data */

     PUT #49 @COLUMN AVE_RESV 8. ;
     PUT #50 @COLUMN MAX_RESV 8. ;
     PUT #51 @COLUMN MIN_RESV 8. ;
   END;  /* format for acre-feet(meters3), no decimals */

ELSE
   DO;  /* format values for feet(meters) with 2 decimals */
     ROW = 9;
     DO DAY = 1 TO 31;/* print one column (month) of data */
        ROW = ROW + 1;
        PUT #ROW @COLUMN STORGE[DAY] 8.2 ;

        IF MOD(DAY,5) = 0 THEN
           DO;
             ROW = ROW + 1;
           END;
     END;       /* print one column (month) of data */

     PUT #49 @COLUMN AVE_RESV 8.2 ;
     PUT #50 @COLUMN MAX_RESV 8.2 ;
     PUT #51 @COLUMN MIN_RESV 8.2 ;
   END;/* format values for feet(meters) with 2 decimals */

IF EOF THEN STOP;
IF LAST.STATION THEN POSITION = 12;
IF POSITION ¢= 12 THEN
   DO;                  /* go to a new page */
     PUT _PAGE_;
     PAGE_NUM = PAGE_NUM + 1;
   END;
```

```
      CORRECT = CORRECT + 1;
      IF CORRECT ¢ 12 THEN CORRECT = 1;

   END; /* print 12 months of data */
RUN;
```

4.56 Description of Module LIST DAILY DATA FOR HOURLY PRECIPITATION (LD_HPCP)

4.56.1 Processing Narrative

LIST DAILY DATA FOR HOURLY PRECIPITATION takes the subset of the hourly precipitation file and generates a listing of the daily totals. Corrections are made for invalid values of the variables MONTH and DAY, and daily codes as well as explanatory footnotes are printed.

4.56.2 Interface Description

A) Parameter: FIRSTMON - month in which to begin listing
(keyword - default is 1)

B) Global data:
Macro variables:
METRIC : identifies request for metric units

SAS data sets:
HPCP (input) - with variables:
STATION YEAR MONTH DAY STN_FORM
HPCP_TOT DAYSUM NAME COUNTY AREA ELEV
PRINT (output) - standard SAS print file

C) Macros are called by this module are:
PRINT MONTH NAMES IN HEADING (4.57 - MONHEAD)
PRINT DAY NUMBERS (4.58 - DAY_PRT)
PRINT STATION INFO IN HEADING (4.59 - ST_HEAD)
CALCULATE COLUMN FOR OUTPUT (4.60 - CAL_COL)

D) No permanent files are used.

E) The module generates one complete _NULL_ DATA step.

4.56.3 Internal Data Description

No local macro variables are used in this module.

SAS variables:
PAGE_NUM (num): current page number for heading
MOVE (num): constant used to position data values

```
ADJUSTDN  (num): adjusts high value of MONTH to correct column
TOTAL     (num): array name for total monthly precipitations
ADJUSTUP  (num): adjusts low value of MONTH to correct column
MAXDATE   (num): maximum date (year+month) for one page
DATENOW   (num): current date (year+month) of observation
CORRECT   (num): used to correct illegal month values
ROW       (num): identifies current row for printing
COLUMN    (num): identifies current column for printing
D         (num): control variable for DO loop
EOF   (boolean): identifies last record read from input file

POSITION  (num): identifies number of columns printed, 1-12
CODE     (char): letter value for daily summary code
LAST.STATION (boolean): identifies last occcurence of station
```

4.56.4 Design Language Description

```
DATA _NULL_; /* prints daily values from hourly precip file */
  RETAIN PAGE_NUM 1 MOVE -1;
  FILE PRINT N=PS NOTITLES;
  ARRAY TOTAL[12]  TOTAL1-TOTAL12;

  %IF &FIRSTMON = 1 %THEN
        ADJUSTDN = 0;
        ADJUSTUP = 0;
  %ELSE
        ADJUSTDN = &FIRSTMON - 1;
        ADJUSTUP = 12 - ADJUSTDN;

  initialize TOTAL array to 0;

  MAXDATE = 999999;
  CORRECT = &FIRSTMON;
  DO UNTIL(POSITION ¢= 12); /* print 12 months per page */
     SET HPCP END=EOF;
       BY STATION ;
     DATENOW = (YEAR * 100) + MONTH;
     IF DATENOW ¢ MAXDATE  THEN  /* exceeded date for the page */
                                 /* Begin new page             */
        PUT _PAGE_;
        PAGE_NUM = PAGE_NUM + 1;
        CORRECT = &FIRSTMON;

     IF CORRECT = &FIRSTMON THEN
       DO;        /* beginning new page, so calculate */
                  /* maximum date for page, and print headings */
        IF &FIRSTMON = 1  THEN
           MAXDATE = (YEAR * 100) + 12;
        ELSE IF MONTH ¢= &FIRSTMON  THEN
                MAXDATE = (YEAR + 1) * 100 + ADJUSTDN;
```

```
      ELSE
         MAXDATE = (YEAR * 100) + ADJUSTDN;

   PUT #1 @110 'PAGE ' PAGE_NUM;
   PUT #49 @1 'TOTAL';
   PUT #52 @1 'ZERO VALUES ARE PRINTED AS BLANKS.';
   PUT #53 @1 'M:  MISSING VALUE(S) DURING DAY.';
   PUT #54 @1 'T:  TRACE AMOUNT(S) DURING DAY.';
   PUT #55 @1 'A:  ACCUMULATED VALUE(S) DURING DAY.';
   PUT #56 @1 'E:  ESTIMATED VALUE(S) DURING DAY.';
   PUT #57 @1 'S:  MELTING SNOW MEASURED DURING DAY.';
   PUT #59 @1 '*** CAUTION ***    MUCH OF THE DATA BEFORE '
              ' 1970 IS NOT FLAGGED FOR SPECIAL CONDITIONS.'

   %PRINT MONTH NAMES IN HEADING(FIRST=&FIRSTMON)

   %IF &METRIC = 0 %THEN
         PUT #6 @45
               'DAILY PRECIPITATION IN INCHES';
   %ELSE PUT #6 @40
         'DAILY PRECIPITATION IN MILLIMETERS';

   %PRINT DAY NUMBERS(FIRSTROW=9)

   %PRINT STATION INFO IN HEADING(FIRST=&FIRSTMON)
END; /* beginning new page, calculate and print headings */

IF 1 _= DAY _= 31  THEN  /* day value is valid */
                         /* ok to print data   */

   %CALCULATE COLUMN FOR OUTPUT(FIRST=&FIRSTMON)
   CODE = ' ';
   IF DAYSUM NE 0 THEN  /* assign letter to daily code */
      IF DAYSUM = '....1...'B THEN
         CODE = 'M';
      ELSE IF DAYSUM = '.......1'B THEN
         CODE = 'T';
      ELSE IF DAYSUM = '.....1..'B THEN
         CODE = 'A';
      ELSE IF DAYSUM = '......1.'B THEN
         CODE = 'E';
      ELSE IF DAYSUM = '...1....'B THEN
         CODE = 'S';

   /* now check proper day value to assign row number */
   /* and leave the right number of blank lines       */

   IF DAY _= 5  THEN
      ROW = DAY + 9;
   ELSE IF DAY _= 10  THEN
```

```
            ROW = DAY + 10;
         ELSE IF DAY _= 15  THEN
            ROW = DAY + 11;
         ELSE IF DAY _= 20  THEN
            ROW = DAY + 12;
         ELSE IF DAY _= 25  THEN
            ROW = DAY + 13;
         ELSE IF DAY _= 30  THEN
            ROW = DAY + 14;
         ELSE
            ROW = DAY + 15;

         IF HPCP_TOT ¢ 0  THEN  /* print the daily total */
            print #ROW @COLUMN HPCP_TOT and CODE;

         TOTAL[POSITION] = TOTAL[POSITION] + HPCP_TOT;

      END;  /* day value is valid */

      IF EOF THEN STOP;
      IF LAST.STATION  THEN POSITION = 12;
      IF POSITION ¢= 12 THEN    /* go to a new page */
          print #49 monthly totals TOTAL1-12 at bottom of page;
          PUT _PAGE_;
          increment PAGE_NUM by 1;
          RETURN;

      CORRECT = CORRECT + 1;
      IF CORRECT ¢ 12 THEN CORRECT = 1;

   END;  /* print 12 months of data */
RUN;
```

4.57 Description of Module PRINT MONTH NAMES IN HEADING (MONHEAD)

4.57.1 Processing Narrative

PRINT MONTH NAMES IN HEADING prints the complete textual names for the 12 months in the heading of a page of output. A user-defined format for converting the month numbers to names is used. Any month can be used to begin the list of names in the heading.

4.57.2 Interface Description

A) Parameter: FIRST - first month to be listed in heading
(keyword - default is 1)

B) Global data:

No global macro variables are used.

No SAS data sets are created or used.

C) No macros are called by this module.

D) Permanent files: NHIMS.FORMATS

E) The module generates only a portion of a single DATA step.
SAS variables shared with the calling module are:

COLUMN (num): identifies current column for printing

4.57.3 Internal Data Description

No local macro variables are used in this module.

SAS variables:
MON_HEAD (num): numeric value of months 1-12
M (num): control variable for DO loop

4.57.4 Design Language Description

```
MON_HEAD = &FIRST;
COLUMN = 9;
DO M = 1 TO 12;  /* print month names in heading  */
   PUT #8 @COLUMN MON_HEAD MNTH. ;
   COLUMN = COLUMN + 10;
   MON_HEAD = MON_HEAD + 1;
   IF MON_HEAD ¢ 12 THEN
      MON_HEAD = 1;
END;
```

4.58 Description of Module PRINT DAY NUMBERS (DAY_PRT)

4.58.1 Processing Narrative

PRINT DAY NUMBERS generates the numbers 1-31 in a vertical column in the left margin. A blank line separates every 5 rows.

4.58.2 Interface Description

A) Parameter: FIRSTROW - row in which to begin printing day numbers
(keyword - default is 9)

B) Global data:

No global macro variables are used.

No SAS data sets are created or used.

C) No macros are called by this module.

D) No permanent files are used.

E) The module generates only a portion of a single DATA step. SAS variables shared with the calling module are:

ROW (num): identifies current row for printing
DAY (num): control variable for the DO loop

4.58.3 Internal Data Description

No local data are used in this module.

4.58.4 Design Language Description

```
ROW = &FIRSTROW;
DO DAY = 1 TO 31; /* print the daily numbers 1-31 */
   ROW = ROW + 1;
   PUT #ROW @3 DAY 2.;
   IF MOD(DAY,5) = 0 THEN
        ROW = ROW + 1;
END;
```

4.59 Description of Module PRINT STATION INFO IN HEADING (ST_HEAD)

4.59.1 Processing Narrative

PRINT STATION INFO IN HEADING prints the information in the heading that is particular to a certain station: the station's name, county and formatted station number. Also, the current year or range of years pertaining to the data on the page is printed.

4.59.2 Interface Description

A) Parameter: FIRST - first month to be listed in heading
(keyword - default is 1)

B) Global data:

No global macro variables are used.

No SAS data sets are created or used.

C) No macros are called by this module.

D) No permanent files are used.

E) The module generates only a portion of a single DATA step. SAS variables shared with the calling module are:

```
NAME     (char): textual name of station
COUNTY   (char): textual name of county
STN_FORM(char): formatted station number
YEAR      (num): calendar year of record
MONTH     (num): month of record
MOVE      (num): constant used to position data values
```

4.59.3 Internal Data Description

No local macro variables are used in this module.

SAS variables:

```
NEXT_YR (num): year succeeding current calendar year
PRE_YR  (num): year preceeding current calendar year
```

4.59.4 Design Language Description

```
PUT #4 @9 NAME  @60 COUNTY   @100 'STATION NO.  '  STN_FORM;
IF &FIRST = 1 THEN
   DO; /* print the calendar year in heading */
     PUT #7 @1 YEAR;
   END;

ELSE IF MONTH ¢= &FIRST THEN
   DO;  /* print the range of years */
     NEXT_YR = YEAR + 1;
     PUT #7 @1 YEAR +MOVE '-' NEXT_YR;
   END;
ELSE
   DO;  /* print range of years  */
     PRE_YR = YEAR - 1;
     PUT #7 @1 PRE_YR +MOVE '-' YEAR;
   END;
```

4.60 Description of Module CALCULATE COLUMNS FOR OUTPUT (CAL_COL)

4.60.1 Processing Narrative

CALCULATE COLUMNS FOR OUTPUT uses the current value of MONTH to determine which block of columns to print the current observation. If the value of MONTH is outside the valid range, it is corrected.

4.60.2 Interface Description

A) Parameter: FIRST - first month to be listed in heading
(keyword - default is 1)

B) Global data:
No global macro variables are used.

No SAS data sets are created or used.

C) No macros are called by this module.

D) No permanent files are used.

E) The module generates only a portion of a single DATA step.
SAS variables shared with the calling module are:

```
MONTH    (num): current month of record
ADJUSTDN (num): adjusts high value of MONTH to correct column
ADJUSTUP (num): adjusts low value of MONTH to correct column
CORRECT  (num): used to correct illegal month values
POSITION (num): identifies number of columns printed, 1-12
```

4.60.3 Internal Data Description

No local data are used in this module.

4.60.4 Design Language Description

```
IF 1 _= MONTH _= 12 THEN
   DO;   /* adjust legal value of month to proper position */
     IF MONTH ¢= &FIRST THEN
          POSITION = MONTH - ADJUSTDN;
     ELSE POSITION = MONTH + ADJUSTUP;
   END;
ELSE DO;  /* correct invalid month to protect formatting */
     POSITION = CORRECT;
     END;
COLUMN = ((POSITION - 1) * 10) + 7;
```

4.61 Description of Module LIST MONTHLY (LIST_MN)

4.61.1 Processing Narrative

LIST MONTHLY checks the proper element flags and calls the appropriate macro to generate the monthly listing for that element. If the user requests more than a partial listing, a macro is called to first calculate the annual mean. Also, the type of units, metric or standard, are determined.

4.61.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:

```
EL_PRCP : identifies request for precipitation data
EL_TEMP : identifies request for temperature data
EL_STRM : identifies request for streamflow data
EL_SNOW : identifies request for snowfall data
EL_EVAP : identifies request for evaporation data
EL_HPCP : identifies request for hourly precipitation data
L_FIRST : identifies month in which to begin listing
LM_PART : identifies request for partial monthly listing
METRIC  : identifies request for metric units
```

No SAS data sets are created or used.

C) The macros called by this module are:

```
COMPUTE ONE ANNUAL MEAN FROM TOTALS (4.62 - MEAN_T1)
COMPUTE TWO ANNUAL MEANS FROM TOTALS (4.63 - MEAN_T2)
COMPUTE TWO ANNUAL MEANS FROM AVERAGES (4.64 - MEAN_A2)
CONVERT DAILY TO MONTHLY TOTALS (4.65 - CON_DAY)
LIST MONTHLY TOTALS FOR ONE ELEMENT (4.66 - LM_TOT1)
LIST MONTHLY TOTALS FOR TWO ELEMENTS (4.67 - LM_TOT2)
LIST MONTHLY AVERAGES FOR TWO ELEMENTS (4.68 - LM_AVE2)
```

D) No permanent files are used by this module.

E) The module generates macro statements only.

4.61.3 Internal Data Description

No local data are used in this module.

4.61.4 Design Language Description

```
%IF request for precipitation data %THEN
    %IF request for partial monthly listing %THEN
        %IF not request for metric units %THEN
            %LIST MONTHLY TOTALS FOR ONE ELEMENT
              (&L_FIRST,PRCP,PRCP_TOT,PRECIPITATION,INCHES,7.2)
        %ELSE %IF request for metric units %THEN
            %LIST MONTHLY TOTALS FOR ONE ELEMENT
                (&L_FIRST,PRCP,PRCP_TOT,PRECIPITATION,
                 MILLIMETERS,7.1)
    %ELSE
       %IF not request for partial monthly listing %THEN
          %COMPUTE ONE ANNUAL MEAN FROM TOTALS
              (&L_FIRST,PRCP,PRCP_TOT)
          %IF not a request for metric units %THEN
            %LIST MONTHLY TOTALS FOR ONE ELEMENT
                (&L_FIRST,PRCPMEAN,PRCP_TOT,PRECIPITATION,
                 INCHES,7.2)
          %ELSE %IF a request for metric units %THEN
            %LIST MONTHLY TOTALS FOR ONE ELEMENT
                (&L_FIRST,PRCPMEAN,PRCP_TOT,PRECIPITATION,
                 MILLIMETERS,7.1)


%IF request for temperature data %THEN
    %IF request for partial monthly listing %THEN
        %IF not a request for metric units %THEN
            %LIST MONTHLY AVERAGES FOR TWO ELEMENTS
                (&L_FIRST,TEMP,AVEMAX,MAXIMUM TEMPERATURES,
                 DEGREES F,7.1,AVEMIN,MINIMUM TEMPERATURES,
                 DEGREES F,7.1)
        %ELSE %IF a request for metric units %THEN
            %LIST MONTHLY AVERAGES FOR TWO ELEMENTS
                (&L_FIRST,TEMP,AVEMAX,MAXIMUM TEMPERATURES,
                 DEGREES C,7.2,AVEMIN,MINIMUM TEMPERATURES,
                 DEGREES C,7.2)
    %ELSE %IF not a request for partial monthly listing %THEN
          %COMPUTE TWO ANNUAL MEANS FROM AVERAGES
              (&L_FIRST,TEMP,AVEMAX,AVEMIN)
          %IF not a request for metric units %THEN
              %LIST MONTHLY AVERAGES FOR TWO ELEMENTS
                (&L_FIRST,TEMPMEAN,AVEMAX,MAXIMUM TEMPERATURES,
                 DEGREES F,7.1,AVEMIN,MINIMUM TEMPERATURES,
                 DEGREES F,7.1)
          %ELSE
          %IF a request for metric units %THEN
              %LIST MONTHLY AVERAGES FOR TWO ELEMENTS
                (&L_FIRST,TEMPMEAN,AVEMAX,MAXIMUM TEMPERATURES,
                 DEGREES C,7.2,AVEMIN,MINIMUM TEMPERATURES,
                 DEGREES C,7.2)
```

```
%IF request for streamflow data %THEN
    %IF request for partial monthly listing %THEN
        %IF not a request for metric units %THEN
            %LIST MONTHLY TOTALS FOR ONE ELEMENT
              (&L_FIRST,STRM,STRM_TOT,STREAMFLOW,CFS-DAYS,7.2)
        %ELSE %IF a request for metric units %THEN
            %LIST MONTHLY TOTALS FOR ONE ELEMENT
              (&L_FIRST,STRM,STRM_TOT,STREAMFLOW,CMS-DAYS,7.2)

    %ELSE %IF not a request for partial monthly listing %THEN
        %COMPUTE ONE ANNUAL MEAN FROM TOTALS
            (&L_FIRST,STRM,STRM_TOT)
        %IF not a request for metric units %THEN
            %LIST MONTHLY TOTALS FOR ONE ELEMENT
                (&L_FIRST,STRMMEAN,STRM_TOT,STREAMFLOW,
                 CFS-DAYS,7.2)
        %ELSE
        %IF a request for metric units %THEN
            %LIST MONTHLY TOTALS FOR ONE ELEMENT
                (&L_FIRST,STRMMEAN,STRM_TOT,STREAMFLOW,
                 CMS-DAYS,7.2)

%IF request for snowfall data %THEN
    %IF request for partial monthly listing %THEN
        %IF not a request for metric units %THEN
            %LIST MONTHLY TOTALS FOR ONE ELEMENT
                (&L_FIRST,SNOW,SNOW_TOT,SNOWFALL,INCHES,7.1)
        %ELSE %IF a request for metric units %THEN
            %LIST MONTHLY TOTALS FOR ONE ELEMENT
              (&L_FIRST,SNOW,SNOW_TOT,SNOWFALL,CENTIMETERS,7.1)

    %ELSE if not a request for partial monthly listing %THEN
        %COMPUTE ONE ANNUAL MEAN FROM TOTALS
            (&L_FIRST,SNOW,SNOW_TOT)
        %IF not a request for metric units %THEN
            %LIST MONTHLY TOTALS FOR ONE ELEMENT
              (&L_FIRST,SNOWMEAN,SNOW_TOT,SNOWFALL,INCHES,7.1)
        %ELSE
        %IF a request for metric units %THEN
            %LIST MONTHLY TOTALS FOR ONE ELEMENT
                (&L_FIRST,SNOWMEAN,SNOW_TOT,SNOWFALL,
                 CENTIMETERS,7.1)

%IF request for evaporation data %THEN
    %IF request for partial monthly listing %THEN
        %IF not a request for metric units %THEN
            %LIST MONTHLY TOTALS FOR TWO ELEMENTS
                (&L_FIRST,EVAP,EVAP_TOT,EVAPORATION,INCHES,7.2,
                 WIND_TOT,WIND MOVEMENT,MILES,7.)
```

```
        %ELSE %IF a request for metric units %THEN
            %LIST MONTHLY TOTALS FOR TWO ELEMENTS
                (&L_FIRST,EVAP,EVAP_TOT,EVAPORATION,MILLIMETERS,
                 7.1,WIND_TOT,WIND MOVEMENT,KILOMETERS,7.)

    %ELSE %IF not a request for partial monthly listing %THEN
        %COMPUTE TWO ANNUAL MEANS FROM TOTALS
            (&L_FIRST,EVAP,EVAP_TOT,WIND_TOT)
        %IF not a request for metric units %THEN
            %LIST MONTHLY TOTALS FOR TWO ELEMENTS
                (&L_FIRST,EVAPMEAN,EVAP_TOT,EVAPORATION,INCHES,
                 7.2,WIND_TOT,WIND MOVEMENT,MILES,7.)
        %ELSE
        %IF a request for metric units %THEN
            %LIST MONTHLY TOTALS FOR TWO ELEMENTS
                (&L_FIRST,EVAPMEAN,EVAP_TOT,EVAPORATION,
                 MILLIMETERS,7.2,WIND_TOT,WIND MOVEMENT,
                 KILOMETERS,7.)


%IF request for hourly precipitation data %THEN
    CONVERT DAILY TO MONTHLY TOTALS(HPCP,HPCP)
    %IF request for partial monthly listing %THEN
        %IF not request for metric units %THEN
            %LIST MONTHLY TOTALS FOR ONE ELEMENT
                (&L_FIRST,HPCPMON,HPCP_MON,HOURLY PRECIPITATION
                 INCHES,7.2)
        %ELSE %IF request for metric units %THEN
            %LIST MONTHLY TOTALS FOR ONE ELEMENT
                (&L_FIRST,HPCPMON,HPCP_MON,HOURLY PRECIPITATION,
                 MILLIMETERS,7.1)
    %ELSE
       %IF not request for partial monthly listing %THEN
          %COMPUTE ONE ANNUAL MEAN FROM TOTALS
              (&L_FIRST,HPCP,HPCP_MON)
          %IF not a request for metric units %THEN
            %LIST MONTHLY TOTALS FOR ONE ELEMENT
                (&L_FIRST,HPCPMEAN,HPCP_MON,
                 HOURLY PRECIPITATION,INCHES,7.2)
          %ELSE %IF a request for metric units %THEN
            %LIST MONTHLY TOTALS FOR ONE ELEMENT
                (&L_FIRST,HPCPMEAN,HPCP_MON,
                 HOURLY PRECIPITATION,MILLIMETERS,7.1)
```

4.62 Description of Module COMPUTE ONE ANNUAL MEAN FROM TOTALS (MEAN_T1)

4.62.1 Processing Narrative

COMPUTE ONE ANNUAL MEAN FROM TOTALS calculates the mean annual value for each station for any element that has one monthly total. This is only performed when the user wants a complete monthly listing with averages.

4.62.2 Interface Description

A) Parameters: ELEMENT - 4-character name of element
FIRSTMON - month in which to begin listing

B) Global data:

No global macro variables are used.

SAS data sets:
&ELEMENT (input) - with variables:
STATION YEAR MONTH STN_FORM NAME COUNTY AREA ELEV
&ELEMENT_TOT MONTHSUM

&ELEMENT.MEAN (input and output) - with variables:
STATION MEAN_ANN

&ELEMENT.COMP (output) - with variables:
STATION YEAR MONTH STN_FORM NAME COUNTY AREA ELEV
&ELEMENT_TOT MONTHSUM MEAN_ANN

C) No macros are called by this module.

D) No permanent files are used by this module.

E) The module generates two complete DATA steps.

4.62.3 Internal Data Description

No local macro variables are used in this module.

SAS variables:
TOTAL (num): array name for accumulated totals for each month
NUMMON (num): array name for number of months counter for each month
AVEMON (num): array name for mean monthly value for each month in the period
POSITION(num): identifies group of columns
ADJUSTDN(num): adjusts month value down to proper position
ADJUSTUP(num): adjusts month value up to proper position
LAST.STATION (boolean): identifies last value of station
INVALID (num): flags an invalid month value
EOF (boolean): identifies last record read from data set

4.62.4 Design Language Description

```
DATA &ELEMENT.MEAN (KEEP=STATION MEAN_ANN);
   ARRAY TOTAL[12]  TOTAL1-TOTAL12;
   ARRAY NUMMON[12] NUMMON1-NUMMON12;
   ARRAY AVEMON[12] AVEMON1-AVEMON12;

   DO POSITION = 1 TO 12;
      initialize TOTAL and NUMMON array elements to 0;
   END;

   %IF &FIRSTMON = 1 %THEN
       ADJUSTDN = 0;
       ADJUSTUP = 0;
   %ELSE
       ADJUSTDN = &FIRSTMON - 1;
       ADJUSTUP = 12 - ADJUSTDN;

   DO UNTIL(LAST.STATION);
      SET &ELEMENT END=EOF;
      INVALID = 0;
      IF 1 _= MONTH _= 12  THEN
         IF MONTH ¢= &FIRSTMON   THEN
            POSITION = MONTH - ADJUSTDN;
         ELSE
            POSITION = MONTH + ADJUSTUP;
      ELSE
         POSITION = 1;
         INVALID = 1;

      IF MONTHSUM does not indicate missing data  THEN
         IF INVALID = 0  THEN
                     /* accumulate the total */
            TOTAL[POSITION] = TOTAL[POSITION] + &ELEMENT._TOT;
                    /* increment month counter */
            NUMMON[POSITION] = NUMMON[POSITION] + 1;
   END; /* until LAST.STATION */

   DO POSITION = 1 TO 12; /* calculate monthly means for period */
      IF NUMMON[POSITION] NE 0 THEN
         AVEMON[POSITION] = TOTAL[POSITION] / NUMMON[POSITION];
      ELSE
         AVEMON[POSITION] = 0;
   END;

   MEAN_ANN = SUM(OF AVEMON1-AVEMON12);
   OUTPUT;
   IF EOF  THEN STOP;   RUN;
```

```
DATA &ELEMENT.COMP;
   MERGE &ELEMENT.MEAN  &ELEMENT;
      BY STATION;
RUN;
```

4.63 Description of Module COMPUTE TWO ANNUAL MEANS FROM TOTALS (MEAN_T2)

4.63.1 Processing Narrative

COMPUTE TWO ANNUAL MEANS FROM TOTALS calculates the annual means for each station for any element that has two monthly totals. This is only performed when the user wants a complete monthly listing with averages.

4.63.2 Interface Description

A) Parameters: ELEMENT - 4-character name of element
ELEMENT2 - 4-character name of second element
FIRSTMON - month in which to begin listing

B) Global data:

No global macro variables are used.

SAS data sets:

&ELEMENT (input) - with variables:
STATION YEAR MONTH STN_FORM NAME COUNTY AREA ELEV
&ELEMENT._TOT &ELEMENT2._TOT MONTHSUM

&ELEMENT.MEAN (input and output) - with variables:
STATION MEAN_ANN MEAN_AN2

&ELEMENT.COMP (output) - with variables:
STATION YEAR MONTH STN_FORM NAME COUNTY AREA ELEV
&ELEMENT._TOT &ELEMENT2._TOT MONTHSUM MEAN_ANN MEAN_AN2

C) No macros are called by this module.

D) No permanent files are used by this module.

E) The module generates two complete DATA steps.

4.63.3 Internal Data Description

No local macro variables are used in this module.

```
SAS variables:
   TOTALA  (num): array name for accumulated totals for first
                  element
   TOTALB  (num): array name for accumulated totals for second
                  element
   NUMA    (num): array name for counter for number of months
                  regarding the first element
   NUMB    (num): array name for counter for number of months
                  regarding the second element
   AVEA    (num): array name for mean monthly value for first
                  element

   AVEB    (num): array for mean monthly value for second element
   POSITION(num): identifies group of columns
   ADJUSTDN(num): adjusts month value down to proper position
   ADJUSTUP(num): adjusts month value up to proper position
   LAST.STATION (boolean): identifies last value of station
   INVALID (num): flags an invalid month value
   EOF     (boolean): identifies last record read from data set
```

4.63.4 Design Language Description

```
  DATA &ELEMENT.MEAN (KEEP=STATION MEAN_ANN MEAN_AN2);
     ARRAY TOTALA[12]  TOTALA1-TOTALA12;
     ARRAY TOTALB[12]  TOTALB1-TOTALB12;
     ARRAY NUMA[12] NUMA1-NUMA12;
     ARRAY NUMB[12] NUMB1-NUMB12;
     ARRAY AVEA[12] AVEA1-AVEA12;
     ARRAY AVEB[12] AVEB1-AVEB12;

     DO POSITION = 1 TO 12;
        initialize TOTALA, TOTALB, NUMA and NUMB arrays to 0;
     END;

     %IF &FIRSTMON = 1 %THEN
         ADJUSTDN = 0;
         ADJUSTUP = 0;
     %ELSE
         ADJUSTDN = &FIRSTMON - 1;
         ADJUSTUP = 12 - ADJUSTDN;

     DO UNTIL(LAST.STATION);
        SET &ELEMENT END=EOF;
        INVALID = 0;
        IF 1 _= MONTH _= 12  THEN
           IF MONTH ¢= &FIRSTMON   THEN
              POSITION = MONTH - ADJUSTDN;
           ELSE
              POSITION = MONTH + ADJUSTUP;
```

```
      ELSE
         POSITION = 1;
         INVALID = 1;

      IF MONTHSUM does not indicate missing data for first
                  element  THEN
         IF INVALID = 0  THEN      /* accumulate and increment */
            TOTALA[POSITION] = TOTALB[POSITION] + &ELEMENT._TOT;
            NUMA[POSITION] = NUMA[POSITION] + 1;

      IF MONTHSUM does not indicate missing data for second
                  element  THEN
         IF INVALID = 0  THEN /* accumulate and increment */
            TOTALB[POSITION] = TOTALB[POSITION] + &ELEMENT2._TOT;
            NUMB[POSITION] = NUMB[POSITION] + 1;
   END; /* until LAST.STATION */

   DO POSITION = 1 TO 12;
      IF NUMA[POSITION] NE 0 THEN
                  /* calculate monthly means for element 1 */
         AVEA[POSITION] = TOTALA[POSITION] / NUMA[POSITION];
      ELSE
         AVEA[POSITION] = 0;
                  /* calculate monthly means for element 2 */
      IF NUMB[POSITION] NE 0 THEN
         AVEB[POSITION] = TOTALB[POSITION] / NUMB[POSITION];
      ELSE
         AVEB[POSITION] = 0;
   END;

   MEAN_ANN = SUM(OF AVEA1-AVEA12);
   MEAN_AN2 = SUM(OF AVEB1-AVEB12);

   OUTPUT;
   IF EOF  THEN STOP;
RUN;


DATA &ELEMENT.COMP;

   MERGE &ELEMENT.MEAN  &ELEMENT;
      BY STATION;
RUN;
```

4.64 Description of Module COMPUTE TWO ANNUAL MEANS FROM AVERAGES (MEAN_A2)

4.64.1 Processing Narrative

COMPUTE TWO ANNUAL MEANS FROM AVERAGES calculates the annual means for each station for any element that has two monthly totals. This is only performed when the user wants a complete monthly listing with averages.

4.64.2 Interface Description

A) Parameters: ELEMENT - 4-character name of element
E_VAR1 - variable name of first element
E_VAR2 - variable name of second element

B) Global data:

No global macro variables are used.

SAS data sets:
&ELEMENT (input) - with variables:
STATION YEAR MONTH STN_FORM NAME COUNTY AREA ELEV
&E_VAR1 &E_VAR2 MONTHSUM

&ELEMENT.MEAN (input and output) - with variables:
STATION MEAN_ANN MEAN_AN2

&ELEMENT.COMP (output) - with variables:
STATION YEAR MONTH STN_FORM NAME COUNTY AREA ELEV
&E_VAR1 &E_VAR2 MONTHSUM MEAN_ANN MEAN_AN2

C) No macros are called by this module.

D) No permanent files are used by this module.

E) The module generates two complete DATA steps.

4.64.3 Internal Data Description

No local macro variables are used in this module.

SAS variables:
EL_TOT1 (num): accumulator for monthly averages - first element
EL_TOT2 (num): accumulator for monthly averages - second element
EL_CNTR1(num): counter for number of months - first element
EL_CNTR2(num): counter for number of months - second element
POSITION(num): identifies group of columns
LAST.STATION (boolean): identifies last value of station
INVALID (num): flags an invalid month value
EOF (boolean): identifies last record read from data set

4.64.4 Design Language Description

```
DATA &ELEMENT.MEAN (KEEP=STATION MEAN_ANN MEAN_AN2);

   initialize EL_TOT1, EL_TOT2, EL_CNTR1 and EL_CNTR2 to 0;

   DO UNTIL(LAST.STATION);
      SET &ELEMENT END=EOF;
      IF 1 _= MONTH _= 12  THEN
         INVALID = 0;
      ELSE
         INVALID = 1;

      IF MONTHSUM does not indicate missing data for first
                 element  THEN
         IF INVALID = 0  THEN
            EL_TOT1 = EL_TOT1 + &E_VAR1; /* accumulate average */
            EL_CNTR1= EL_CNTR1 + 1; /* increment month counter */

      IF MONTHSUM does not indicate missing data for second
                 element  THEN
         IF INVALID = 0  THEN
            EL_TOT2 = EL_TOT2 + &E_VAR2; /* accumulate average */
            EL_CNTR2 = EL_CNTR2 + 1; /* increment month counter */

   END; /* until LAST.STATION */

   MEAN_ANN = EL_TOT1 / EL_CNTR1;
   MEAN_AN2 = EL_TOT2 / EL_CNTR2;

   OUTPUT;
   IF EOF  THEN STOP;
RUN;


DATA &ELEMENT.COMP;

   MERGE &ELEMENT.MEAN  &ELEMENT;
      BY STATION;
RUN;
```

4.65 Description of Module CONVERT DAILY TO MONTHLY TOTALS (CON_DAY)

4.65.1 Processing Narrative

CONVERT DAILY TO MONTHLY TOTALS will convert a file with daily records to monthly records by accumulating the daily totals for each month. The daily summary code is also converted into a corresponding monthly summary code.

4.65.2 Interface Description

A) Parameters: DSN - name of the input data set
ELEMENT - 4-character name of the element

B) Global data:

No global macro variables are used.

SAS data sets:
Can have any name(input) - must have variables:
STATION YEAR MONTH DAY STN_FORM NAME COUNTY
DAYSUM &ELEMENT._TOT

&ELEMENT.MON (output) - with variables:
STATION YEAR MONTH STN_FORM NAME COUNTY
MONTHSUM &ELEMENT._MON

C) No macros are called by this module.

D) No permanent files are used by this module.

E) The module generates one complete DATA step.

4.65.3 Internal Data Description

No local macro variables are used in this module.

SAS variables:
MISS (num): identifies missing code in DAYSUM
ACCUM (num): identifies accumulated code in DAYSUM
ESTIMATE (num): identifies estimated code in DAYSUM
TRACE (num): identifies trace code in DAYSUM
EOF (boolean): identifies when last record is read from input data set
LAST.MONTH (boolean): identifies last occurrence of current value of MONTH

4.65.4 Design Language Description

```
DATA &ELEMENT.MON (KEEP= STATION YEAR MONTH STN_FORM NAME COUNTY
                         &ELEMENT._MON  MONTHSUM);

   MISS = 0;
   ACCUM = 0;
   ESTIMATE = 0;
   TRACE = 0;
   &ELEMENT._MON = 0;
```

```
   DO UNTIL(LAST.MONTH); /* accumulate daily totals for month */
      SET &DSN  END=EOF;
         BY STATION YEAR MONTH;

      IF &ELEMENT._TOT is not equal to missing THEN
         &ELEMENT._MON = &ELEMENT._MON + &ELEMENT._TOT;

      IF DAYSUM = '....1...'B  THEN /* missing data in day */
         MISS = 8;
      IF DAYSUM = '.....1..'B  THEN /* accumulated data in day */
         ACCUM = 4;
      IF DAYSUM = '......1.'B  THEN /* estimated data in day */
         ESTIMATE = 2;
      IF DAYSUM = '.......1'B  THEN /* trace data in day */
         TRACE = 1;
   END;  /* accumulate daily totals for the month */

   MONTHSUM = MISS + ACCUM + ESTIMATE + TRACE;
   OUTPUT;
   IF EOF  THEN STOP;
RUN;
```

4.66 Description of Module LIST MONTHLY TOTALS FOR ONE ELEMENT (LM_TOT1)

4.66.1 Processing Narrative

LIST MONTHLY TOTALS FOR ONE ELEMENT produces the monthly listing for elements that have monthly totals, with only one total per month. Annual totals are calculated, with an option to also list the percentage of each annual total to the mean annual value for the retrieved period of record. In addition, for each of the months, i.e., for all January months, February months, etc., means are calculated, with the corresponding number of months and the percentage of the mean to the mean annual value.

4.66.2 Interface Description

A) Parameters:

FIRSTMON	- month in which to begin listing
DSN	- name of the input data set
E_VAR	- name of the variable to be listed
E_TITLE	- name of the element for the heading
UNITS	- units of the monthly data
FORMAT	- format in which to print data

B) Global data:

Global macro variables:

LM_PART : identifies request for partial monthly listing

SAS data sets:
Can have any name(input) - must have variables:
STATION YEAR MONTH STN_FORM NAME COUNTY AREA ELEV
&E_VAR MONTHSUM MEAN_ANN(optional)
(if streamflow data, must have: DAY_DIS VOL_AREA)

PRINT (output) - standard SAS print file

C) Macros called by this module:

PRINT MONTH NAMES IN HEADING (4.57 - MONHEAD)
CALCULATE COLUMN FOR OUTPUT (4.60 - CAL_COL)

D) No permanent files are used by this module.

E) The module generates one _NULL_ DATA step.

4.66.3 Internal Data Description

No local macro variables are used in this module.

SAS variables:

PAGE_NUM	(num):	current page number for heading
MOVE	(num):	constant used to position data values
TOTAL	(num):	array name for accumulated monthly totals
NUMMON	(num):	array name for counter for number of months
AVEMON	(num):	array name for monthly means
PERCEN	(num):	array name for percentages of monthly means to annual
POSITION	(num):	identifies number of columns printed, 1-12
ADJUSTDN	(num):	adjusts high value of MONTH to correct position
ADJUSTUP	(num):	adjusts low value of MONTH to correct position
ROWCOUNT	(num):	identifies the number of rows printed
ROW	(num):	identifies current row for output
LAST.STATION	(boolean):	identifies last occurrence of current value of STATION
MAXDATE	(num):	maximum date allowed (YEAR+MONTH) in current row
ANNUAL	(num):	accumulator of monthly totals for one year
MISS_YR	(num):	flag for years with missing data
M_COUNT	(num):	flag for years with missing months
CORRECT	(num):	used to correct illegal month values
EOF	(boolean):	identifies last record read from input file
DATENOW	(num):	date (YEAR+MONTH) of current observation
YR	(num):	last 2 digits of YEAR, used in heading
NEXT_YR	(num):	value of YR - 1, used in heading
PRE_YR	(num):	value of YR + 1, used in heading
COLUMN	(num):	identifies current column for output
CODE	(char):	charcter value for MONTHSUM (M, A, E, T)
PER_AVG	(num):	percentage of annual total to mean annual value
FOOTROW	(num):	identifies rows for printing footnotes

4.66.4 Design Language Description

```
DATA _NULL_;

  RETAIN PAGE_NUM 1  MOVE -1;
  ARRAY TOTAL[12]  TOTAL1-TOTAL12;
  ARRAY NUMMON[12]  NUMMON1-NUMMON12;
  ARRAY AVEMON[12] AVEMON1-AVEMON12;
  ARRAY PERCEN[12] PERCEN1-PERCEN12;

  FILE PRINT N=PS;

  DO POSITION = 1 TO 12;
     initialize TOTAL and NUMMON arrays;
  END;

  %IF &FIRSTMON = 1 %THEN
       ADJUSTDN = 0;
       ADJUSTUP = 0;
  %ELSE
       ADJUSTDN = &FIRSTMON - 1;
       ADJUSTUP = 12 - ADJUSTDN;

  ROWCOUNT = 0;
  ROW = 9;

  DO UNTIL(LAST.STATION); /* print monthly data on one station */
     increment ROW by 1;
     increment ROWCOUNT by 1;
     MAXDATE = 999999;
     ANNUAL = 0;
     MISS_YR = 0;
     M_COUNT = 0;
     CORRECT = &FIRSTMON;

     PUT #1 @110 'PAGE ' PAGE_NUM;

     %PRINT MONTH NAMES IN HEADING (FIRST=&FIRSTMON)

     PUT #8 @120 'ANNUAL'  @127  '% AVE ';

     PUT #6 @45 "TOTAL MONTHLY &E_TITLE IN &UNITS" ;

     DO UNTIL(POSITION ¢=12);  /* print one row(year) of data */

        SET &DSN END=EOF;
           BY STATION;
        M_COUNT = M_COUNT + 1;
```

```
DATENOW = (YEAR * 100) + MONTH;
IF DATENOW ¢ MAXDATE  THEN  /* begin a new row */
   PUT #ROW @122  ANNUAL &FORMAT;
   IF MOD(ROWCOUNT,5) = 0 THEN
      ROW = ROW + 1;
   ROW = ROW + 1;
   ROWCOUNT = ROWCOUNT + 1;
   ANNUAL = 0;
   MISS_YR = 0;
   M_COUNT = 1;
   CORRECT = &FIRSTMON;
END;  /* begin a new row */

IF CORRECT = &FIRSTMON  THEN  /* begin printing a row */
   PUT #4 @9 NAME  @60 COUNTY  @100 STN_FORM;
   IF &FIRSTMON = 1  THEN
      DO;  /* print calendar year in left margin */
        PUT #ROW @1 YEAR;
        MAXDATE = (YEAR * 100) + 12;
   ELSE
      DO;   /* print range of years in left margin*/
        IF YEAR _ 1900 THEN YR = YEAR - 1800;
        ELSE
           IF YEAR _ 2000 THEN YR = YEAR - 1900;
           ELSE YR = YEAR - 2000;

        IF MONTH ¢= &FIRSTMON  THEN
           DO;  /* range is year to next year */
             NEXT_YR = YR + 1;
             PUT #ROW @1 YR +MOVE '-' NEXT_YR;
             MAXDATE = (YEAR + 1) * 100 + ADJUSTDN;
        ELSE DO; /* range is previous year to year */
             PRE_YR = YR - 1;
             PUT #ROW @1 PRE_YR +MOVE '-' YR;
             MAXDATE = (YEAR * 100) + ADJUSTDN;
      END;  /* print range of years in left margin */

END;   /* begin printing a row */

%CALCULATE COLUMN FOR OUTPUT

/* now convert MONTHSUM to letter code */

IF MONTHSUM = 0  THEN
   CODE = ' ';
ELSE IF MONTHSUM = '....1...'B  THEN
   CODE = 'M';
ELSE IF MONTHSUM = '.......1'B  THEN
   CODE = 'T';
```

```
      ELSE IF MONTHSUM = '.....1..'B  THEN
         CODE = 'A';
      ELSE IF MONTHSUM = '......1.'B  THEN
         CODE = 'E';

      PUT #ROW @COLUMN &E_VAR &FORMAT CODE $CHAR1.;

      ANNUAL = ANNUAL + &E_VAR;
      IF CODE NE 'M'  THEN
         TOTAL[POSITION] = TOTAL[POSITION] + &E_VAR;
         NUMMON[POSITION] = NUMMON[POSITION] + 1;
      ELSE MISS_YR = 1;   /* flag - do not figure % averages */
                          /* for years with missing data     */

      IF EOF OR LAST.STATION  THEN
         POSITION = 12;
      IF M_COUNT NE POSITION  THEN
         MISS_YR = 1;

      IF POSITION ¢= 12 THEN
         PUT #ROW @122 ANNUAL &FORMAT;
         %IF not a request for partial monthly listing  %THEN
            IF MISS_YR EQ 0  THEN /* no missing data in year */
               PER_AVG = ANNUAL / MEAN_ANN * 100;
               PUT #ROW @129 PER_AVG 3.;
         %END;
      CORRECT = CORRECT + 1;
      IF CORRECT ¢ 12  THEN CORRECT = 1;

   END;  /* print one year(row) of data */

   IF ROW ¢ 48  THEN   /* print footnotes and */
                       /* go to a new page    */
      ROW = ROW + 2;
      PUT #ROW @1 'M:  MISSING VALUES DURING MONTH.';
      PUT      @1 'A:  ACCUMULATED VALUES DURING MONTH.';
      PUT      @1 'E:  ESTIMATED VALUES DURING MONTH.';
      PUT      @1 'T:  TRACE VALUES FOR THE MONTH.';
      PUT     @1 '*** CAUTION ***  MUCH OF THE DATA BEFORE 1970 '
                 'IS NOT FLAGGED FOR SPECIAL CONDITIONS.';

      PUT _PAGE_;
      increment PAGE_NUM by 1;
      ROW = 9;
      ROWCOUNT = 0;
   END;  /*  go to a new page  */

   ELSE IF MOD(ROWCOUNT,5) = 0  THEN
      increment ROW by 1;

END; /* print data for one station */
```

```
DO POSITION = 1 TO 12; /* calculate monthly(vertical) means */
   IF NUMMON[POSITION] NE 0 THEN
      AVEMON[POSITION] = TOTAL[POSITION] / NUMMON[POSITION] ;
   ELSE
      AVEMON[POSITION] = 0;
END;

%IF request for partial monthly listing   %THEN
    MEAN_ANN = SUM(OF AVEMON1-AVEMON12);
%END;

FOOTROW = ROW + 2;
PUT #FOOTROW @1  'MEAN';
PUT @1 'NO. OF ';
PUT @1 ' MONTHS';
PUT @1 'PERCENT';
PUT @1 ' ANNUAL';

DO POSITION = 1 TO 12;
   COLUMN = ((POSITION - 1) * 10) + 7;
   PERCEN[POSITION] = AVEMON[POSITION] / MEAN_ANN * 100;
   FOOTROW = ROW + 2;
   PUT #FOOTROW @COLUMN AVEMON[POSITION] &FORMAT;
   FOOTROW = FOOTROW + 2;
   PUT #FOOTROW @COLUMN NUMMON[POSITION] 3.;
   FOOTROW = FOOTROW + 2;
   PUT #FOOTROW @COLUMN PERCEN[POSITION] 5.1;
END;

FOOTROW = ROW + 9;

%IF element title to be printed = 'STREAMFLOW'  %THEN
    VOL_AREA = DAY_DIS / AREA;
    PUT #FOOTROW @40 "MEAN ANNUAL DISCHARGE"
                 @65 MEAN_ANN 9.2   "&UNITS" ;
    FOOTROW = FOOTROW + 1;
    PUT #FOOTROW @40 "MEAN DAILY DISCHARGE" ;

    %IF not request for metric units  %THEN
        PUT #FOOTROW @65 DAY_DIS 9.2 'CFS ';
        PUT @65 VOL_AREA 9.2  'CFS/SQ MI    (' AREA  ' SQ MI)';

    %ELSE
        PUT #FOOTROW @65 DAY_DIS 9.2 'CMS ';
        PUT @65 VOL_AREA 9.2  'CMS/SQ KM    (' AREA  ' SQ KM)';

 %ELSE
        PUT #FOOTROW @40 "MEAN ANNUAL &E_TITLE "
                         MEAN_ANN  " &UNITS" ;
```

```
IF FOOTROW ¢ 50  THEN /* print footnotes on next page */
   PUT _PAGE_;
   increment PAGE_NUM by 1;
   PUT #1 @110  'PAGE ' PAGE_NUM;
   ROW = 4;
ELSE
   ROW = ROW + 13;

PUT #ROW @1 'M:  MISSING VALUES DURING MONTH.';
PUT      @1 'A:  ACCUMULATED VALUES DURING MONTH.';
PUT      @1 'E:  ESTIMATED VALUES DURING MONTH.';
PUT      @1 'T:  TRACE VALUES FOR THE MONTH.';
PUT      @1 '*** CAUTION ***  MUCH OF THE DATA BEFORE 1970 '
            'IS NOT FLAGGED FOR SPECIAL CONDITIONS.';

IF EOF  THEN STOP;
ELSE
   PUT _PAGE_;
   increment PAGE_NUM by 1
   RETURN;
RUN;
```

4.67 Description of Module LIST MONTHLY TOTALS FOR TWO ELEMENTS (LM_TOT2)

4.67.1 Processing Narrative

LIST MONTHLY TOTALS FOR TWO ELEMENTS produces the monthly listing for elements that have monthly totals, with two totals per month. Annual totals are calculated, with an option to also list the percentage of each annual total to the mean annual value for the retrieved period of record. In addition, for each of the months, i.e., for all January months, February months, etc., means are calculated, with the corresponding number of months and the percentage of the mean to the mean annual value.

4.67.2 Interface Description

A) Parameters:

FIRSTMON	- month in which to begin listing
DSN	- name of the input data set
E_VAR	- name of first variable to be listed
E_TITLE	- name of first element for the heading
UNITS	- units of monthly data for first element
FORMAT	- format to print data for first element
E_VAR2	- name of second variable to be listed
E_TITLE2	- name of second element for the heading
UNITS2	- units of monthly data for second element
FORMAT2	- format to print data for second element

B) Global data:

Global macro variables:
LM_PART : identifies request for partial monthly listing

SAS data sets:
Can have any name(input) - must have variables:
STATION YEAR MONTH STN_FORM NAME COUNTY AREA ELEV
&E_VAR &E_VAR2 MONTHSUM
MEAN_ANN(optional) MEAN_AN2(optional)

PRINT (output) - standard SAS print file

C) Macros called by this module:

PRINT MONTH NAMES IN HEADING (4.57 - MONHEAD)
CALCULATE COLUMN FOR OUTPUT (4.60 - CAL_COL)

D) No permanent files are used by this module.

E) The module generates one _NULL_ DATA step.

4.67.3 Internal Data Description

No local macro variables are used in this module.

SAS variables:

PAGE_NUM	(num):	current page number for heading
MOVE	(num):	constant used to position data values
TOTALA	(num):	array name for accumulated monthly totals for first element
TOTALB	(num):	array name for accumulated monthly totals for second element
NUMA	(num):	array name for counter for number of months for first element
NUMB	(num):	array name for counter for number of months for second element
AVEA	(num):	array name for monthly means for first element
AVEB	(num):	array name for monthly means for second element
PERA	(num):	array name for percentages of monthly means to mean annual for first element
PERB	(num):	array name for percentages of monthly means to mean annual for second element
POSITION	(num):	identifies number of columns printed, 1-12
ADJUSTDN	(num):	adjusts high value of MONTH to correct position
ADJUSTUP	(num):	adjusts low value of MONTH to correct position
ROWCOUNT	(num):	identifies the number of rows printed
ROW	(num):	identifies current row for printing first element
ROW2	(num):	identifies current row for printing extra element

```
LAST.STATION (boolean): identifies last occurrence of current
                        value of STATION
MAXDATE   (num): maximum date allowed (YEAR+MONTH) in current row
ANNUAL    (num): accumulator of monthly totals for one year for
                 first element
ANNUAL2   (num): accumulator of monthly totals for one year for
                 extra element
MISS_YR   (num): flag for years with missing data - first element
MISS_YR2  (num): flag for years with missing data - extra element
M_COUNT   (num): flag of years with missing months - first element
M_COUNT2  (num): flag of years with missing months - extra element
CORRECT   (num): used to correct illegal month values
EOF  (boolean): identifies last record read from input file
DATENOW   (num): date (YEAR+MONTH) of current observation
YR        (num): last 2 digits of YEAR, used in heading
NEXT_YR   (num): value of YR - 1, used in heading
PRE_YR    (num): value of YR + 1, used in heading
COLUMN    (num): identifies current column for output
CODE     (char): charcter value for MONTHSUM for first element
CODE2    (char): charcter value for MONTHSUM for extra element
PER_AVG   (num): percentage of annual total to mean annual value
                 for first element
PER_AVG2  (num): percentage of annual total to mean annual value
                 for extra element
FOOTROW   (num): identifies rows for printing footnotes for first
                 element
FOOTROW2  (num): identifies rows for printing footnotes for extra
                 element
NUMBER    (num): minimum matching value of NUMMON and NUMMON2
```

4.67.4 Design Language Description

```
DATA _NULL_;

  RETAIN PAGE_NUM 1  MOVE -1;
  ARRAY TOTALA[12]  TOTALA1-TOTALA12;
  ARRAY TOTALB[12]  TOTALB1-TOTALB12; /* 2ND ELEM */
  ARRAY NUMA[12]    NUMA1-NUMA12;
  ARRAY NUMB[12]    NUMB1-NUMB12;   /* 2ND  ELEM */
  ARRAY AVEA[12]    AVEA1-AVEA12;
  ARRAY AVEB[12]    AVEB1-AVEB12; /* 2ND ELEM */
  ARRAY PERA[12]    PERA1-PERA12;
  ARRAY PERB[12]    PERB1-PERB12;   /* 2ND ELEM */

  FILE PRINT N=PS;

  DO POSITION = 1 TO 12;
     initialize TOTALA, TOTALB, NUMA and NUMB arrays;
  END;
```

```
%IF &FIRSTMON = 1 %THEN
     ADJUSTDN = 0;
     ADJUSTUP = 0;
%ELSE
     ADJUSTDN = &FIRSTMON - 1;
     ADJUSTUP = 12 - ADJUSTDN;

ROWCOUNT = 0;
ROW = 9;
ROW2 = 10;

DO UNTIL(LAST.STATION); /* print monthly data on one station */
   increment ROW by 1;
   increment ROW2 by 1;
   increment ROWCOUNT by 1;
   MAXDATE = 999999;
   ANNUAL   = 0;
   ANNUAL2  = 0;
   MISS_YR  = 0;
   MISS_YR2 = 0
   M_COUNT  = 0;
   M_COUNT2 = 0;
   CORRECT = &FIRSTMON;

   PUT #1 @110 'PAGE ' PAGE_NUM;

   %PRINT MONTH NAMES IN HEADING (FIRST=&FIRSTMON)

   PUT #8 @120 'ANNUAL'  @127  '% AVE ';

   PUT #6 @35 "TOTAL MONTHLY &E_TITLE IN &UNITS"
              "AND &E_TITLE2 IN &UNITS2";

   DO UNTIL(POSITION ¢=12);  /* print one row(year) of data */

      SET &DSN END=EOF;
         BY STATION;
      DATENOW = (YEAR * 100) + MONTH;
      IF DATENOW ¢ MAXDATE  THEN  /* begin a new row */
         PUT #ROW @122  ANNUAL &FORMAT;
         PUT #ROW2 @122 ANNUAL2 &FORMAT2;
         IF MOD(ROWCOUNT,5) = 0 THEN
            ROW = ROW + 1;
            ROW2 = ROW2 + 1;
         ROW = ROW + 2;
         ROW2 = ROW2 + 2;
         ROWCOUNT = ROWCOUNT + 1;
         ANNUAL = 0;
         ANNUAL2 = 0;
         MISS_YR = 0;
```

```
   MISS_YR2 = 0;
   M_COUNT = 0;
   M_COUNT2 = 0;
   CORRECT = &FIRSTMON;
END;  /* begin a new row */

M_COUNT = M_COUNT + 1;
M_COUNT2 = M_COUNT2 + 1;
IF CORRECT = &FIRSTMON  THEN  /* begin printing a row */
   PUT #4 @9 NAME  @60 COUNTY  @100 STN_FORM;
   IF &FIRSTMON = 1  THEN
      DO;  /* print calendar year in left margin */
        PUT #ROW @1 YEAR;
        MAXDATE = (YEAR * 100) + 12;
   ELSE
      DO;  /* print range of years in left margin */
        IF YEAR _ 1900 THEN  YR = YEAR - 1800;
        ELSE
           IF YEAR _ 2000 THEN YR = YEAR - 1900;
           ELSE  YR = YEAR - 2000;

        IF MONTH ¢= &FIRSTMON  THEN
           DO;  /* range is year to next year  */
           NEXT_YR = YR + 1;
           PUT #ROW @1 YR +MOVE '-' NEXT_YR;
           MAXDATE = (YEAR + 1) * 100 + ADJUSTDN;
        ELSE DO; /* range is previous year to year */
           PRE_YR = YR - 1;
           PUT #ROW @1 PRE_YR +MOVE '-' YR;
           MAXDATE = (YEAR * 100) + ADJUSTDN;
      END;  /* print range of years in left margin */

END;   /* begin printing a row */

%CALCULATE COLUMN FOR OUTPUT

/* now convert MONTHSUM to letter code */
/* first check low order bits for one element */

IF MONTHSUM = 0  THEN
   CODE = ' ';
ELSE IF MONTHSUM = '....1...'B  THEN
   CODE = 'M';
ELSE IF MONTHSUM = '.......1'B  THEN
   CODE = 'T';
ELSE IF MONTHSUM = '.....1..'B  THEN
   CODE = 'A';
ELSE IF MONTHSUM = '......1.'B  THEN
   CODE = 'E';
```

```
/* now check high order bits for 2nd element */

IF MONTHSUM = 0  THEN
   CODE2 = ' ';
ELSE IF MONTHSUM = '1.......'B  THEN
   CODE2 = 'M';
ELSE IF MONTHSUM = '...1....'B  THEN
   CODE2 = 'T';
ELSE IF MONTHSUM = '.1......'B  THEN
   CODE2 = 'A';
ELSE IF MONTHSUM = '..1.....'B  THEN
   CODE2 = 'E';

PUT #ROW @COLUMN &E_VAR &FORMAT CODE $CHAR1.;
PUT #ROW2 @COLUMN &E_VAR2 &FORMAT2 CODE2 $CHAR1.;

ANNUAL = ANNUAL + &E_VAR;
ANNUAL2 = ANNUAL2 + &E_VAR2;
IF CODE NE 'M'  THEN
   TOTALA[POSITION] = TOTALA[POSITION] + &E_VAR;
   NUMA[POSITION] = NUMA[POSITION] + 1;
ELSE MISS_YR = 1;   /* flag - do not figure % averages */
                    /* for years with missing data     */

IF CODE2 NE 'M'  THEN
   TOTALB[POSITION] = TOTALB[POSITION] + &E_VAR2;
   NUMB[POSITION] = NUMB[POSITION] + 1;
ELSE MISS_YR2 = 1;   /* flag - do not figure % averages */
                     /* for years with missing data     */

IF EOF OR LAST.STATION  THEN
   POSITION = 12;
IF M_COUNT NE POSITION  THEN
   MISS_YR = 1;
IF M_COUNT2 NE POSITION THEN
   MISS_YR2 = 1;

IF POSITION ¢= 12 THEN
   PUT #ROW @122 ANNUAL &FORMAT;
   PUT #ROW2 @122 ANNUAL2 &FORMAT2;
   %IF not a request for partial monthly listing  %THEN
      IF MISS_YR EQ 0  THEN /* no missing data in year */
                            /* for element 1           */
         PER_AVG = ANNUAL / MEAN_ANN * 100;
         PUT #ROW @129 PER_AVG 3.;

      IF MISS_YR2 EQ 0  THEN /* no missing data in year */
                             /* for element 2           */
         PER_AVG2 = ANNUAL2 / MEAN_AN2 * 100;
         PUT #ROW2 @129 PER_AVG2 3.;
   %END;
```

```
      CORRECT = CORRECT + 1;
      IF CORRECT ¢ 12  THEN CORRECT = 1;

   END;  /* print one year(row) of data */

   IF ROW ¢ 48  THEN    /* print footnotes and */
                        /* go to a new page    */
      ROW = ROW + 2;
      PUT #ROW @1 'M:  MISSING VALUES DURING MONTH.';
      PUT       @1 'A:  ACCUMULATED VALUES DURING MONTH.';
      PUT       @1 'E:  ESTIMATED VALUES DURING MONTH.';
      PUT       @1 'T:  TRACE VALUES FOR THE MONTH.';
      PUT      @1 '*** CAUTION ***  MUCH OF THE DATA BEFORE 1970 '
                  'IS NOT FLAGGED FOR SPECIAL CONDITIONS.';

      PUT _PAGE_;
      increment PAGE_NUM by 1;
      ROW = 9;
      ROW2 = 10;
      ROWCOUNT = 0;
   END;  /*  go to a new page  */

   ELSE IF MOD(ROWCOUNT,5) = 0  THEN
      increment ROW by 1;
      increment ROW2 by 1;

END; /* print data for one station */

DO POSITION = 1 TO 12; /* calculate monthly(vertical) means */
   IF NUMA[POSITION] NE 0 THEN
      AVEA[POSITION] = TOTALA[POSITION] / NUMA[POSITION] ;
   ELSE
      AVEA[POSITION] = 0;

   IF NUMB[POSITION] NE 0 THEN
      AVEB[POSITION] = TOTALB[POSITION] / NUMB[POSITION];
   ELSE
      AVEB[POSITION] = 0;
END;

%IF request for partial monthly listing   %THEN
    MEAN_ANN = SUM(OF AVEA1-AVEA12);
    MEAN_AN2 = SUM(OF AVEB1-AVEB12);
%END;

FOOTROW = ROW + 3;
PUT #FOOTROW @1  'MEANS';
PUT;
```

```
      PUT @1 'NO. OF ';
      PUT @1 ' MONTHS';
      PUT @1 'PERCENT';
      PUT @1 ' ANNUAL';

      DO POSITION = 1 TO 12;
         COLUMN = ((POSITION - 1) * 10) + 7;
         PERA[POSITION] = AVEA[POSITION] / MEAN_ANN * 100;
         PERB[POSITION] = AVEB[POSITION] / MEAN_AN2 * 100;
         FOOTROW = ROW + 3;
         FOOTROW2 = ROW + 4;
         PUT #FOOTROW @COLUMN AVEA[POSITION] &FORMAT;
         PUT #FOOTROW2 @COLUMN AVEB[POSITION] &FORMAT2;
         FOOTROW = FOOTROW + 3;
         NUMBER = MIN(NUMA[POSITION],NUMB[POSITION]);
         PUT #FOOTROW @COLUMN NUMBER 3.;
         FOOTROW = FOOTROW + 2;
         FOOTROW2 = FOOTROW2 + 5;
         PUT #FOOTROW @COLUMN PERA[POSITION] 5.1;
         PUT #FOOTROW2 @COLUMN PERB[POSITION] 5.1;
      END;

      FOOTROW = ROW + 11;
      FOOTROW2 = ROW + 12;
      PUT #FOOTROW @40 "MEAN ANNUAL &E_TITLE "
                   @67 MEAN_ANN &FORMAT
                   @75 " &UNITS" ;
      PUT #FOOTROW2 @40 "MEAN ANNUAL &E_TITLE2 "
                    @67 MEAN_AN2 &FORMAT2
                    @75 " &UNITS2" ;

      IF FOOTROW ¢ 50  THEN /* print footnotes on next page */
         PUT _PAGE_;
         increment PAGE_NUM by 1;
         PUT #1 @110  'PAGE ' PAGE_NUM;
         ROW = 4;
      ELSE
         ROW = ROW + 15;

      print same footnotes as above;
      IF EOF  THEN STOP;
              ELSE
                 PUT _PAGE_;
                 increment PAGE_NUM by 1;
RUN;
```

4.68 Description of Module LIST MONTHLY AVERAGES FOR TWO ELEMENTS (LM_AVE2)

4.68.1 Processing Narrative

LIST MONTHLY AVERAGES FOR TWO ELEMENTS produces the monthly listing for elements that have monthly averages, with two averages per month. Annual averages are calculated, with an option to also list the percentage of each annual average to the mean annual value for the retrieved period of record. In addition, for each of the months, i.e., for all January months, February months, etc., means are calculated, with the corresponding number of months.

4.68.2 Interface Description

A) Parameters:

FIRSTMON	- month in which to begin listing
DSN	- name of the input data set
E_VAR	- name of first variable to be listed
E_TITLE	- name of first element for the heading
UNITS	- units of monthly data for first element
FORMAT	- format to print data for first element
E_VAR2	- name of second variable to be listed
E_TITLE2	- name of second element for the heading
UNITS2	- units of monthly data for second element
FORMAT2	- format to print data for second element

B) Global data:

Global macro variables:

LM_PART : identifies request for partial monthly listing

SAS data sets:

Can have any name(input) - must have variables:

STATION YEAR MONTH STN_FORM NAME COUNTY AREA ELEV
&E_VAR &E_VAR2 MONTHSUM
MEAN_ANN(optional) MEAN_AN2(optional)

PRINT (output) - standard SAS print file

C) Macros called by this module:

PRINT MONTH NAMES IN HEADING (4.57 - MONHEAD)
CALCULATE COLUMN FOR OUTPUT (4.60 - CAL_COL)

D) No permanent files are used by this module.

E) The module generates one _NULL_ DATA step.

4.68.3 Internal Data Description

No local macro variables are used in this module.

SAS variables:

```
   PAGE_NUM (num): current page number for heading
   MOVE     (num): constant used to position data values
   TOTALA   (num): array name for accumulated monthly averages
                   for first element
   TOTALB   (num): array name for accumulated monthly averages
                   for second element
   NUMA     (num): array name for counter for number of months
                   for first element
   NUMB     (num): array name for counter for number of months
                   for second element
   AVEA     (num): array name for monthly means for first element
   AVEB     (num): array name for monthly means for second element
   POSITION (num): identifies number of columns printed, 1-12
   ADJUSTDN (num): adjusts high value of MONTH to correct position
   ADJUSTUP (num): adjusts low value of MONTH to correct position
   ROWCOUNT (num): identifies the number of rows printed
   ROW      (num): identifies current row for printing first element
   ROW2     (num): identifies current row for printing extra element
   EL_TOT   (num): accumulator for all averages for first element
   EL_TOT2  (num): accumulator for all averages for extra element
   EL_CNTR  (num): counter for averages accumulated - first element
   EL_CNTR2 (num): counter for averages accumulated - extra element
   LAST.STATION (boolean): identifies last occurrence of current
                           value of STATION
   MAXDATE  (num): maximum date allowed (YEAR+MONTH) in current row
   ANNUAL   (num): accumulator of monthly totals for one year for
                   first element
   ANNUAL2  (num): accumulator of monthly totals for one year for
                   extra element
   MISS_YR  (num): flag for years with missing data - first element
   MISS_YR2 (num): flag for years with missing data - extra element
   M_COUNT  (num): flag of years with missing months - first element
   M_COUNT2 (num): flag of years with missing months - extra element
   CORRECT  (num): used to correct illegal month values
   EOF  (boolean): identifies last record read from input file
   DATENOW  (num): date (YEAR+MONTH) of current observation
   YR       (num): last 2 digits of YEAR, used in heading
   NEXT_YR  (num): value of YR - 1, used in heading
   PRE_YR   (num): value of YR + 1, used in heading
   COLUMN   (num): identifies current column for output
   CODE    (char): charcter value for MONTHSUM for first element
   CODE2   (char): charcter value for MONTHSUM for extra element
   PER_AVG  (num): percentage of annual total to mean annual value
                   for first element
   PER_AVG2 (num): percentage of annual total to mean annual value
                   for extra element
```

```
FOOTROW  (num): identifies rows for printing footnotes for first
                element
FOOTROW2 (num): identifies rows for printing footnotes for extra
                element
NUMBER   (num): minimum of matching values of NUMMON and NUMMON2
```

4.68.4 Design Language Description

```
DATA _NULL_;

 RETAIN PAGE_NUM 1  MOVE -1  EL_TOT EL_CNTR EL_TOT2 EL_CNTR2;
 ARRAY TOTALA[12]  TOTALA1-TOTALA12;
 ARRAY TOTALB[12]  TOTALB1-TOTALB12; /* 2ND ELEM */
 ARRAY NUMA[12]    NUMA1-NUMA12;
 ARRAY NUMB[12]    NUMB1-NUMB12;  /* 2ND  ELEM */
 ARRAY AVEA[12]    AVEA1-AVEA12;
 ARRAY AVEB[12]    AVEB1-AVEB12; /* 2ND  ELEM */

 FILE PRINT N=PS;

 %IF &FIRSTMON = 1 %THEN
      ADJUSTDN = 0;
      ADJUSTUP = 0;
 %ELSE
      ADJUSTDN = &FIRSTMON - 1;
      ADJUSTUP = 12 - ADJUSTDN;

 DO POSITION = 1 TO 12;
    initialize TOTALA, TOTALB, NUMA and NUMB arrays;
 END;

 ROWCOUNT = 0;
 ROW = 9;
 ROW2 = 10;
 initialize EL_TOT, EL_CNTR, EL_TOT2 and EL_CNTR2 to 0;

 DO UNTIL(LAST.STATION); /* print monthly data on one station */
    increment ROW by 1;
    increment ROW2 by 1;
    increment ROWCOUNT by 1;
    MAXDATE = 999999;
    ANNUAL   = 0;
    ANNUAL2  = 0;
    MISS_YR  = 0;
    MISS_YR2 = 0
    M_COUNT  = 0;
    M_COUNT2 = 0;
    CORRECT = &FIRSTMON;
```

```
PUT #1 @110 'PAGE ' PAGE_NUM;

%PRINT MONTH NAMES IN HEADING (FIRST=&FIRSTMON)

PUT #8 @120 'ANNUAL'  @127  '% AVE ';

PUT #6 @30 "AVERAGE MONTHLY &E_TITLE IN &UNITS"
           "AND &E_TITLE2 IN &UNITS2";

DO UNTIL(POSITION ¢=12);  /* print one row(year) of data */

   SET &DSN END=EOF;
      BY STATION;
   DATENOW = (YEAR * 100) + MONTH;
   IF DATENOW ¢ MAXDATE  THEN  /* begin a new row */
      ANNUAL = ANNUAL / M_COUNT;
      ANNUAL2 = ANNUAL2 / M_COUNT2;
      PUT #ROW @122  ANNUAL &FORMAT;
      PUT #ROW2 @122 ANNUAL2 &FORMAT2;
      IF MOD(ROWCOUNT,5) = 0 THEN
         ROW = ROW + 1;
         ROW2 = ROW2 + 1;
      ROW = ROW + 2;
      ROW2 = ROW2 + 2;
      ROWCOUNT = ROWCOUNT + 1;
      ANNUAL = 0;
      ANNUAL2 = 0;
      MISS_YR = 0;
      MISS_YR2 = 0;
      M_COUNT = 0;
      M_COUNT2 = 0;
      CORRECT = &FIRSTMON;
   END;  /* begin a new row */

   M_COUNT = M_COUNT + 1;
   M_COUNT2 = M_COUNT2 + 1;
   IF CORRECT = &FIRSTMON  THEN  /* begin printing a row */
      PUT #4 @9 NAME  @60 COUNTY  @100 STN_FORM;
      IF &FIRSTMON = 1  THEN
         DO;  /* print calendar year in left margin */
           PUT #ROW @1 YEAR;
           MAXDATE = (YEAR * 100) + 12;
      ELSE
         DO;   /* print range of years in left margin */
           IF YEAR _ 1900  THEN  YR = YEAR - 1800;
           ELSE
              IF YEAR _ 2000 THEN YR = YEAR - 1900;
              ELSE  YR = YEAR - 2000;

           IF MONTH ¢= &FIRSTMON  THEN
              DO;  /* range is year to next year */
```

```
            NEXT_YR = YR + 1;
            PUT #ROW @1 YR +MOVE '-' NEXT_YR;
            MAXDATE = (YEAR + 1) * 100 + ADJUSTDN;
        ELSE DO; /* range is previous year to year */
            PRE_YR = YR - 1;
            PUT #ROW @1 PRE_YR +MOVE '-' YR;
            MAXDATE = (YEAR * 100) + ADJUSTDN;
    END;  /* print range of years in left margin */

END;   /* begin printing a row */

%CALCULATE COLUMN FOR OUTPUT

/* now convert MONTHSUM to letter code */
/* first check low order bits for one element */

IF MONTHSUM = 0  THEN
   CODE = ' ';
ELSE IF MONTHSUM = '....1...'B  THEN
   CODE = 'M';
ELSE IF MONTHSUM = '.......1'B  THEN
   CODE = 'T';
ELSE IF MONTHSUM = '.....1..'B  THEN
   CODE = 'A';
ELSE IF MONTHSUM = '......1.'B  THEN
   CODE = 'E';

/* now check high order bits for 2nd element */

IF MONTHSUM = 0  THEN
   CODE2 = ' ';
ELSE IF MONTHSUM = '1.......'B  THEN
   CODE2 = 'M';
ELSE IF MONTHSUM = '...1....'B  THEN
   CODE2 = 'T';
ELSE IF MONTHSUM = '.1......'B  THEN
   CODE2 = 'A';
ELSE IF MONTHSUM = '..1.....'B  THEN
   CODE2 = 'E';

PUT #ROW @COLUMN &E_VAR &FORMAT CODE $CHAR1.;
PUT #ROW2 @COLUMN &E_VAR2 &FORMAT2 CODE2 $CHAR1.;

ANNUAL = ANNUAL + &E_VAR;
ANNUAL2 = ANNUAL2 + &E_VAR2;
IF CODE NE 'M'  THEN
   EL_TOT = EL_TOT + &E_VAR;
   EL_CNTR = EL_CNTR + 1;
   TOTALA[POSITION] = TOTALA[POSITION] + &E_VAR;
   NUMA[POSITION] = NUMA[POSITION] + 1;
ELSE MISS_YR = 1;   /* flag - do not figure % averages */
                    /* for years with missing data     */
```

```
   IF CODE2 NE 'M'  THEN
      EL_TOT2 = EL_TOT2 + &E_VAR2;
      EL_CNTR2 = EL_CNTR2 + 1;
      TOTALB[POSITION] = TOTALB[POSITION] + &E_VAR2;
      NUMB[POSITION] = NUMB[POSITION] + 1;
   ELSE MISS_YR2 = 1;   /* flag - do not figure % averages */
                        /* for years with missing data     */

  IF EOF OR LAST.STATION  THEN
      POSITION = 12;
   IF M_COUNT NE POSITION  THEN
      MISS_YR = 1;
   IF M_COUNT2 NE POSITION THEN
      MISS_YR2 = 1;

   IF POSITION ¢= 12 THEN
      ANNUAL = ANNUAL / M_COUNT;
      ANNAUL2 = ANNUAL2 / M_COUNT2;
      PUT #ROW @122 ANNUAL &FORMAT;
      PUT #ROW2 @122 ANNUAL2 &FORMAT2;
      %IF not a request for partial monthly listing  %THEN
         IF MISS_YR EQ 0  THEN /* no missing data in year */
                               /* for element 1           */
            PER_AVG = ANNUAL / MEAN_ANN ;
            PUT #ROW @129 PER_AVG 3.;

         IF MISS_YR2 EQ 0  THEN /* no missing data in year */
                                /* for element 2           */
            PER_AVG2 = ANNUAL2 / MEAN_AN2 ;
            PUT #ROW2 @129 PER_AVG2 3.;
      %END;

   CORRECT = CORRECT + 1;
   IF CORRECT ¢ 12  THEN CORRECT = 1;

END;  /* print one year(row) of data */

IF ROW ¢ 48  THEN   /* print footnotes and */
                    /* go to a new page    */
   PUT #52 @1 'M:  MISSING VALUES DURING MONTH.';
   PUT #53 @1 'E:  ESTIMATED VALUES DURING MONTH.';
   PUT #55 @1 '*** CAUTION ***  MUCH OF THE DATA BEFORE 1970 '
              'IS NOT FLAGGED FOR SPECIAL CONDITIONS.';

   PUT _PAGE_;
   increment PAGE_NUM by 1;
   ROW = 9;
```

```
      ROW2 = 10;
      ROWCOUNT = 0;
   END;  /*  go to a new page  */

   ELSE IF MOD(ROWCOUNT,5) = 0  THEN
      increment ROW by 1;
      increment ROW2 by 1;
END; /* print data for one station */

DO POSITION = 1 TO 12; /* calculate monthly(vertical) means */
   IF NUMA[POSITION] NE 0 THEN
      AVEA[POSITION] = TOTALA[POSITION] / NUMA[POSITION] ;
   ELSE
      AVEA[POSITION] = 0;

   IF NUMB[POSITION] NE 0 THEN
      AVEB[POSITION] = TOTALB[POSITION] / NUMB[POSITION];
   ELSE
      AVEB[POSITION] = 0;
END;

%IF request for partial monthly listing   %THEN
    MEAN_ANN = EL_TOT / EL_CNTR;
    MEAN_AN2 = EL_TOT2 / EL_CNTR2;
%END;

FOOTROW = ROW + 3;
PUT #FOOTROW @1  'MEANS';
PUT;
PUT @1 'NO. OF ';
PUT @1 ' MONTHS';

DO POSITION = 1 TO 12;
   COLUMN = ((POSITION - 1) * 10) + 7;
   FOOTROW = ROW + 3;
   FOOTROW2 = ROW + 4;
   PUT #FOOTROW @COLUMN AVEA[POSITION] &FORMAT;
   PUT #FOOTROW2 @COLUMN AVEB[POSITION] &FORMAT2;
   FOOTROW = FOOTROW + 3;
   NUMBER = MIN(NUMA[POSITION],NUMB[POSITION]);
   PUT #FOOTROW @COLUMN NUMBER 3.;
END;

FOOTROW = ROW + 8;
FOOTROW2 = ROW + 9;
PUT #FOOTROW @40 "MEAN ANNUAL &E_TITLE "
             @67 MEAN_ANN &FORMAT
             @75 " &UNITS" ;
PUT #FOOTROW2 @40 "MEAN ANNUAL &E_TITLE2 "
              @67 MEAN_AN2 &FORMAT2
              @75 " &UNITS2" ;
```

```
IF FOOTROW ¢ 50  THEN /* print footnotes on next page */
   PUT _PAGE_;
   increment PAGE_NUM by 1;
   PUT #1 @110  'PAGE ' PAGE_NUM;
   ROW = 4;
ELSE
   ROW = ROW + 15;

print footnotes as above;
IF EOF  THEN STOP;
        ELSE
           PUT _PAGE_;
           increment PAGE_NUM by 1;
           RETURN;
RUN;
```

4.69 Description of Module LIST CONTENTS (LIST_CT)

4.69.1 Processing Narrative

LIST CONTENTS checks the proper element flags and calls the appropriate macro to generate the contents listing for that element.

4.69.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:
- EL_MNTH : identifies request for monthly summary data
- EL_PEAK : identifies request for peak flow data
- EL_SNOC : identifies request for snow course data

No SAS data sets are created or used.

C) The macros called by this module are:

LIST PEAK FLOW CONTENTS (4.70 - LC_PEAK)
LIST SNOW COURSE CONTENTS (4.71 - LC_SNOC)
LIST MONTHLY SUMMARY CONTENTS (4.73 - LC_MNTH)

D) No permanent files are used by this module.

E) The module generates macro statements only.

4.69.3 Internal Data Description

No local data are used in this module.

4.69.4 Design Language Description

```
%IF request for monthly summary data %THEN
    %LIST MONTHLY SUMMARY CONTENTS

%IF request for peakflow data %THEN
    %LIST PEAKFLOW CONTENTS

%IF request for snow course data %THEN
    %LIST SNOW COURSE CONTENTS
```

4.70 Description of Module LIST PEAK FLOW CONTENTS (LC_PEAK)

4.70.1 Processing Narrative

LIST PEAK FLOW CONTENTS prints out the annual records from the peak flow file. Footnotes are printed at the bottom of each page to explain the meanings of any codes. If the maximum gage height for a water year occurred on another date than the peak flow, its value is printed on the same line.

4.70.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:
METRIC : identifies request for metric units

SAS data sets:
PEAK (input) - with variables:
STATION WAT_YEAR YEAR MONTH DAY MONDAYYR STN_FORM
NAME COUNTY AREA ELEV
GAGE_HT PEAKFLOW GH_CODE PK_CODE RD_CODE

PRINT (output) - standard SAS print file

C) No macros are called by this module.

D) Permanent files: NHIMS.FORMATS

E) The module generates one _NULL_ DATA step.

4.70.3 Internal Data Description

No local macro variables are used in this module.

SAS variables:

```
PREVIOUS (num): identifies if previous record contained peak
                flow for the same water year
ROW      (num): identifies current row for printing values
PAGE_NUM (num): identifies current page number for output
AR_UNITS(char): units for drainage area values
GH_UNITS(char): units for gage height values
PK_UNITS(char): units for peak flow values
LAST.STATION (boolean): identifies last occurrence of current
                        value of STATION
```

4.70.4 Design Language Description

```
DATA _NULL_; /* lists peakflow contents */

   FILE PRINT N=PS;
   RETAIN PREVIOUS 'N'  ROW 10  PAGE_NUM 1;

   SET PEAK;
     BY STATION;

   IF ROW = 10  THEN  /* begin new page, so print headings */

      %IF not a request for metric units %THEN
          AR_UNITS = 'SQ MI';
          GH_UNITS = 'FEET  ';
          PK_UNITS = 'CFS';
      %ELSE %IF a request for metric units %THEN
          AR_UNITS = 'SQ KM';
          GH_UNITS = 'METERS';
          PK_UNITS = 'CMS';
          AREA = AREA * 2.59;
          ELEV = ELEV * 0.305;
      %END;

      PUT #1 @100 'PAGE ' PAGE_NUM;
      PUT #2 @1 NAME  @65 COUNTY  @90 'STATION ' STN_FORM;
      PUT #4 @40 'DRAINAGE AREA = ' AREA 10.1 ' ' AR_UNITS ;
      PUT #5 @40 'GAGE DATUM = ' ELEV 8.  ' '  GH_UNITS;

      PUT #7  @4 'WATER'        @11 'ANNUAL PEAK'  @25 'DATE'
             @33 'PEAK FLOW'    @46 'GAGE HEIGHT OF'
             @62 'GAGE HEIGHT'  @75 'ANNUAL MAX GAGE'
             @93 'DATE'         @100 'MAX GAGE HEIGHT';
```

```
      PUT #8  @5 'YEAR'          @10 'DISCH, ' PK_UNITS
              @35 'CODE'         @44 'ANN PEAK, ' GH_UNITS
              @65 'CODE'         @76 'HEIGHT, ' GH_UNITS
              @105 'CODE';

   END;  /* begin new page, so print headings */

   IF PREVIOUS = 'N' THEN /* print annual peakflow data */
      PUT #ROW @4  WAT_YEAR 5.      @10 PEAKFLOW 10.
               @23 MONDAYYR DATE9. @34 PK_CODE PKFORM.
               @37 RD_CODE RDFORM. @46 GAGE_HT 10.2
               @66 GH_CODE GHFORM.;

   ELSE IF PREVIOUS = 'Y' THEN
         /* print max gage height on same row as peak flow */
         /* from previous record  */
      PUT #ROW @78 GAGE_HT 10.2  @91 MONDAYYR DATE9.
               @106 GH_CODE GHFORM. ;


   IF GH_CODE NE '2' THEN /* next record is not for max gage ht */
      ROW = ROW + 1;
      IF MOD(ROW,5) = 0  THEN  ROW = ROW + 1;
      PREVIOUS = 'N';
   ELSE    /* next record has max gage ht for same year  */
      PREVIOUS = 'Y';

   IF (LAST.STATION)  OR  (ROW ¢ 51)
      THEN  /* print footnotes, begin new page */

      PUT #54 @1  'BW - GAGE HEIGHT WAS DUE TO BACKWATER '
              @49 'NM - NOT MAXIMUM GAGE HEIGHT FOR WATER YEAR';
      PUT #55 @1  'MD - DISCHARGE GIVEN IS A MAXIMUM DAILY'
              @49 'ES - DISCHARGE ESTIMATED FROM ANOTHER SITE';
      PUT #56 @1  'DF - DISCHARGE GIVEN DUE TO DAM FAILURE'
              @49 'LT - ACTUAL DISCHARGE IS LESS THAN INDICATED '
                  'VALUE';
      PUT #57 @1  'UR - UNKNOWN EFFECT OF REGULATION OR DIVERSION'
              @49 'KR - KNOWN SIGNIFICANT EFFECT OF REGULATION '
                  'OR DIVERSION';
      PUT #59 @1  'NOTE:  GAGE DATUM MAY NOT BE EFFECTIVE FOR '
                  'ENTIRE PERIOD OF RECORD.  GAGE HEIGHTS SHOWN';
      PUT #60 @8  'ONLY FOR CURRENT DATUM.  CONSULT USGS WATER '
                  'SUPPLY PAPERS FOR CHANGES IN DATUM.';

      PUT _PAGE_;
      PAGE_NUM = PAGE_NUM + 1;
      ROW = 10;
      RETURN;
   END;  /* print footnotes, begin new page */
RUN;
```

4.71 Description of Module LIST SNOW COURSE CONTENTS (LC_SNOC)

4.71.1 Processing Narrative

LIST SNOW COURSE CONTENTS prints out the annual records from the snow course file. Before listing, the records are first sorted. The output listings are organized by when the measurements were taken: beginning of month, mid-month month or special measurements.

4.71.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:
METRIC : identifies request for metric units

SAS data sets:
SNOC (input) - with variables:
STATION STATE YEAR CARDNO NAME COUNTY AREA ELEV
MONTH1-MONTH6 DAY1-DAY6 DEPTH1-DEPTH6 WATER1-WATER6

PRINT (output) - standard SAS print file

C) Macros called by this module:
PRINT SNOW COURSE HEADINGS (4.72 - SC_HEAD)

D) No permanent files are used by this module.

E) The module generates one PROC step and one _NULL_ DATA step.

4.71.3 Internal Data Description

No local macro variables are used in this module.

SAS variables:
READING (num): identifies measurement, control variable
ROW (num): identifies current row for printing values
COLUMN (num): identifies current column for printing values
PAGE_NUM (num): identifies current page number for output
LAST.STATION (boolean): identifies last occurrence of current value of STATION
LAST.CARDNO (boolean): identifies last occurrence of current value of CARDNO

4.71.4 Design Language Description

```
PROC SORT DATA=SNOC;
  BY STATION CARDNO;
RUN;


DATA _NULL_; /* lists snow course contents */

   FILE PRINT N=PS;
   RETAIN ROW 8  PAGE_NUM 1;
   ARRAY DEPTH[6] DEPTH1-DEPTH6;
   ARRAY WATER[6] WATER1-WATER6;
   ARRAY MONTH[6] MONTH1-MONTH6;
   ARRAY DAY[6]   DAY1-DAY6;

   SET SNOC;
     BY STATION CARDNO;

   IF (FIRST.CARDNO) OR (ROW = 8)   THEN
                 /* print new set of headings */
      PUT #1 @100 'PAGE ' PAGE_NUM;
      PUT #2 @1 NAME   @65 COUNTY  @86 'STATION NO.  ' STATION;
      PUT #4 @50  'SNOW COURSE RECORDS';
      %IF not a request for metric units %THEN
          PUT #6 @35  'SNOW DEPTHS AND WATER CONTENT IN INCHES';
      %ELSE
          PUT #6 @30  'SNOW DEPTHS IN CENTIMETERS AND '
                      'WATER CONTENT IN MILLIMETERS';
      IF ROW ¢ 8  THEN ROW = ROW + 4; /* data for new cardno */

      IF CARDNO = 1  THEN  /* beginning month measurement */
         %PRINT SNOW COURSE HEADINGS (1)

      ELSE IF CARDNO = 2 THEN  /* mid-month measurement */
         %PRINT SNOW COURSE HEADINGS (15)

      ELSE                    /* special measurement  */
         %PRINT SNOW COURSE HEADINGS (32)

      ROW = ROW + 2;
   END;  /* print new set of headings */

   PUT #ROW @1 YEAR;
   DO READING = 1 TO 6;
      COLUMN = ((READING - 1) * 20) + 9;
      PUT #ROW @COLUMN  MONTH[READING] 2. '/'  DAY[READING] 2.
                        '  '  DEPTH[READING] 3.
                        '  '  WATER[READING] 4.1;
   END;
```

```
      ROW = ROW + 1;
      IF MOD(ROW,5) = 0    THEN ROW = ROW + 1;

      IF LAST.STATION THEN
         PUT _PAGE;
         increment PAGE_NUM by 1;
         ROW = 8;
         RETURN;
      ELSE IF LAST.CARDNO THEN
         IF ROW ¢ 47  THEN
            PUT _PAGE_;
            increment PAGE_NUM by 1;
            ROW = 8;
            RETURN;
      ELSE IF ROW ¢ 56 THEN
            PUT _PAGE_;
            increment PAGE_NUM by 1;
            ROW = 8;
            RETURN;
RUN;
```

4.72 Description of Module PRINT SNOW COURSE HEADINGS (SC_HEAD)

4.72.1 Processing Narrative

PRINT SNOW COURSE HEADINGS prints out the three types of headings for the snow course contents listing.

4.72.2 Interface Description

A) Parameter: DATE - day signifying when measurement was taken

B) Global data:

No global macro variables are used.

No SAS data sets are created or used.

C) No macros are called by this module.

D) Permanent files: NHIMS.FORMATS

E) The module generates only a portion of a single DATA step. SAS variables shared with the calling module are:

READING (num): identifies measurement, control variable
ROW (num): identifies current row for printing values
COLUMN (num): identifies current column for printing values

4.72.3 Internal Data Description

No local data are used in this module.

4.72.4 Design Language Description

```
PUT #ROW @1 'WATER';
DO READING = 1 TO 6;
   COLUMN = ((READING - 1) * 20) + 11;
   %IF &DATE _ 31 %THEN
                  /* a beginning or midmonth measurement */
      PUT #ROW @COLUMN  READING  MNTH.  ' '  &DATE  3.;
   %ELSE          /*  a special measurement */
      PUT #ROW @COLUMN  'SUPPLEMENTARY';
   %END;
 END;

 ROW = ROW + 1;
 PUT #ROW @1 'YEAR';
 DO READING = 1 TO 6;
    COLUMN = ((READING - 1) * 20) + 8;
    PUT #ROW @COLUMN  'DATE  DEPTH    WC' ;
 END;
```

4.73 Description of Module LIST MONTHLY SUMMARY CONTENTS (LC_MNTH)

4.73.1 Processing Narrative

LIST MONTHLY SUMMARY CONTENTS prints out the values from the monthly summary file. Annual totals are calculated and printed.

4.73.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:
   METRIC : identifies request for metric units

SAS data sets:
   MNTH (input) - with variables:
      STATION YEAR MONTH STN_FORM NAME COUNTY AREA ELEV
      MAXTEMP MINTEMP TMEAN TDEPART DEGDAYS HIGHEST HIGHDAY
      LOWEST LOWDAY PRECIP PDEPART PPTMAX PMAXDAY SNOFALL
      SNODEPTH SMAXDAY WIND EVAP COOLDAYS PLUS1-PLUS6

PRINT (output) - standard SAS print file

C) No macros are called by this module.

D) No permanent files are used by this module.

E) The module generates one _NULL_ DATA step.

4.73.3 Internal Data Description

No local macro variables are used in this module.

SAS variables:

```
MAX_ANN  (num): accumulator for mean annual maximum temp
MAX_CNTR (num): counter for number of months with max temps
MAX_MISS(char): identifies a month with missing max temps
MIN_ANN  (num): accumulator for mean annual minimum temp
MIN_CNTR (num): counter for number of months with min temps
MIN_MISS(char): identifies a month with missing min temps
TMN_ANN  (num): accumulator for mean annual temperature
TMN_CNTR (num): counter for number of months with mean temps
HIGH_ANN (num): highest daily temperature during year
LOW_ANN  (num): lowest daily temperature during year
HEAT_ANN (num): total annual heating degree days
COOL_ANN (num): total annual cooling degree days
PRCP_ANN (num): total annual precipitation
PMAX_ANN (num): maximum daily precipitation during year
SNO_ANN  (num): total annual snowfall
DEP_ANN  (num): maximum depth of snow on ground during year
EVAP_ANN (num): total annual evaporation
WIND_ANN (num): total annual wind movement
ROW      (num): identifies current row for printing values
PAGE_NUM (num): identifies current page number for output
LAST.STATION (boolean): identifies last occurrence of current
                        value of STATION
LAST.YEAR (boolean): identifies last occurrence of current
                        value of YEAR
```

4.73.4 Design Language Description

```
DATA _NULL_; /* lists monthly summary contents */

  FILE PRINT N=PS;

   RETAIN ROW 13  PAGE_NUM 1  MAX_ANN 0  MAX_CNTR 0  MIN_ANN 0
          MIN_CNTR 0  TMN_ANN 0  TMN_CNTR 0  HIGH_ANN -60
          LOW_ANN 130  HEAT_ANN 0  COOL_ANN 0  PRCP_ANN 0
          PMAX_ANN 0  SNO_ANN 0  DEP_ANN 0  EVAP_ANN 0
          WIND_ANN 0  ;
```

```
SET MNTH;
  BY STATION YEAR;

IF ROW = 13   THEN   /* print page headings */
   PUT #1 @100 'PAGE ' PAGE_NUM;
   PUT #2 @1 NAME   @65 COUNTY  @86 'STATION NO.  ' STN_FORM;
   PUT #4 @50  'MONTHLY SUMMARIZED STATION DATA';
   %IF not a request for metric units %THEN
       PUT #6 @5  'DATE'  @30  'TEMPERATURES (F)
              @80 'PRECIPITATION (IN)'
              @112 'EVAP/WIND (IN/MI)';
   %ELSE
       PUT #6 @5  'DATE'  @30  'TEMPERATURES (C)'
              @75 'PRECIPITATION (MM/SNOW(CM))'
              @112 'EVAP/WIND (MM/KM)';
   %END;

   /* now print column headings */
   PUT #8 row of dashes;
   PUT #9 @1  '¦'  @13 '¦'  @15 '¦'  @53 'HEAT'
          @60 'COOL'  @66 '¦'  @68 '¦'  @97 'GREATEST'
          @110 '¦'  @112 '¦'  @121 'MONTHLY'  @130 '¦';
   PUT #10 @1  '¦'  @13 '¦'  @15 '¦'
          @16 'MEAN'  @22 'MEAN'  @52 'DEGREE'  @59 'DEGREE'
          @66 '¦'  @68 '¦'  @75 'GREATEST'  @90 'TOTAL'
          @99 'SNOW'  @110 '¦'  @112 '¦'  @115 'TOTAL'
          @122 'WIND'  @130 '¦';

   PUT #11 @1  '¦'  @2 'MONTH'  @8 'YEAR'  @13 '¦'  @15 '¦'
          @16 'MAX'  @22 'MIN'  @29 'MEAN'  @35 'HI'
          @38 'DATE'  @43 'LOW'  @47 'DATE'  @53 'DAYS'
          @60 'DAYS'  @66 '¦'  @68 '¦'  @69 'TOTAL'
          @77 'DAY'  @84 'DATE'  @89 'SNOWFALL'
          @99 'DEPTH'  @106 'DATE'  @110 '¦'  @112 '¦'
          @115 'EVAP'  @122 'RUN'  @130 '¦';
   PUT #12 row of dashes;
END;  /* print page headings */

/* now print one month's data values */
PUT #ROW @3 MONTH 2. @8 YEAR 4.
         @15 MAXTEMP 5.1  PLUS1 $1. @22  MINTEMP 5.1  PLUS2 $1.
         @29 TMEAN 5.1  @35 HIGHEST 3.  @38 HIGHDAY 2.
         @43 LOWEST 3.  @47 LOWDAY 2.   @52 DEGDAYS 5.
         @59 COOLDAYS 5.  @69 PRECIP   PLUS5 $1.
         @76 PPTMAX   PLUS3 $1.  @85 PMAXDAY 2.
         @89 SNOFALL  5.1  PLUS6 $1.  @97 SNODEPTH 4. PLUS4 $1.
         @107 SMAXDAY 2.  @114 EVAP   @121  WIND;

/* now accumulate values for annual means, totals, max/mins */
```

```
IF (PLUS1 NE 'M') AND (PLUS1 NE 'I') THEN /* no missing data */
   MAX_ANN = MAX_ANN + MAXTEMP;
   MAX_CNTR = MAX_CNTR + 1;
   MAX_MISS = 'N';
 IF (PLUS2 NE 'M') AND (PLUS2 NE 'I') THEN /* no missing data */
   MIN_ANN = MIN_ANN + MINTEMP;
   MIN_CNTR = MIN_CNTR + 1;
   MIN_MISS = 'N';
 IF (MAX_MISS = 'N') AND (MIN_MISS = 'N') THEN
   TMN_ANN = TMN_ANN + TMEAN;
   TMN_CNTR = TMN_CNTR + 1;
 HIGH_ANN = MAX(HIGH_ANN,HIGHEST);
 LOW_ANN  = MIN(LOW_ANN,LOWEST);
 HEAT_ANN = HEAT_ANN + DEGDAYS;
 COOL_ANN = COOL_ANN + COOLDAYS;
 PRCP_ANN = PRCP_ANN + PRECIP;
 PMAX_ANN = MAX(PMAX_ANN,PPTMAX);
 SNO_ANN  = SNO_ANN + SNOFALL;
 DEP_ANN  = MAX(DEP_ANN,SNODEPTH);
 EVAP_ANN = EVAP_ANN + EVAP;
 WIND_ANN = WIND_ANN + WIND;


 IF LAST.YEAR THEN  /* print annual values */
   ROW = ROW + 1;
   MAX_ANN = MAX_ANN / MAX_CNTR;
   MIN_ANN = MIN_ANN / MIN_CNTR;
   TMN_ANN = TMN_ANN / TMN_CNTR;


   PUT #ROW @1 'ANNUAL' @8 YEAR 4.
         @15 MAX_ANN 5.1   @22 MIN_ANN  5.1
         @29 TMN_ANN 5.1   @35 HIGH_ANN 3.
         @43 LOW_ANN 3.    @52 HEAT_ANN 5.
         @59 COOL_ANN 5.   @69 PRCP_ANN 6.2
         @76 PMAX_ANN 6.2
         @89 SNO_ANN  5.1  @97 DEP_ANN 4.
         @114 EVAP_ANN 6.2  @121  WIND_ANN 6.1;
   MAX_ANN = 0;
   MAX_CNTR = 0;
   MIN_ANN = 0;
   MIN_CNTR = 0;
   TMN_ANN = 0;
   TMN_CNTR = 0;
   HIGH_ANN = -60;
   LOW_ANN  = 130;
   HEAT_ANN = 0;
   COOL_ANN = 0;
   PRCP_ANN = 0;
   PMAX_ANN = 0;
```

```
      SNO_ANN  = 0;
      DEP_ANN  = 0;
      EVAP_ANN = 0;
      WIND_ANN = 0;
      ROW = ROW + 1;
   END;  /* print annual values */

   ROW = ROW + 1;
   IF (ROW ¢ 54) OR (LAST.STATION) THEN
      /* print footnotes, go to a new page, 3 years/page  */
      PUT #58 @1  '+ - VALUE OCCURRED ON MORE THAN ONE DAY'
              @60 'T - TRACE MONTHLY VALUE' ;
      PUT #59 @1  'M - MISSING RECORDS IN CALCULATION '
              @60 'I - MONTHLY VALUE BASED ON INCOMPLETE PERIOD' ;
      PUT #60 @1  'A - ACCUMULATED AMOUNT'
              @60 'E - ESTIMATED AMOUNT' ;
      PUT _PAGE_;
      increment PAGE_NUM by 1;
      ROW = 13 ;
      RETURN;
   END;  /* go to a new page */
RUN;
```

4.74 Description of Module LIST HOURLY (LIST_HR)

4.74.1 Processing Narrative

LIST HOURLY checks the proper element flags and calls the appropriate macro to generate listings for the hourly data.

4.74.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:
EL_HPCP : identifies request for hourly precipitation data

No SAS data sets are created or used.

C) The macros called by this module are:

LIST HOURLY PRECIPITATION DATA (4.75 - LH_HPCP)

D) No permanent files are used by this module.

E) The module generates macro statements only.

4.74.3 Internal Data Description

No local data are used in this module.

4.74.4 Design Language Description

```
%IF request for hourly precipitation data %THEN
    %LIST HOURLY PRECIPITATION DATA
```

4.75 Description of Module LIST HOURLY PRECIPITATION DATA (LH_HPCP)

4.75.1 Processing Narrative

LIST HOURLY PRECIPITATION DATA takes the subset of the hourly precipitation file and generates a listing of the hourly data values. Invalid values of the variable DAY are not printed. Hourly codes as well as explanatory footnotes are printed.

4.75.2 Interface Description

A) There are no parameters.

B) Global data:

Macro variables:
METRIC : identifies request for metric units

SAS data sets:
HPCP (input) - with variables:
STATION YEAR MONTH DAY DAYSUM STN_FORM HPCP_TOT
HPCP1-HPCP24 H_CODE1-H_CODE24 NAME COUNTY AREA ELEV
PRINT (output) - standard SAS print file

C) No macros are called by this module.

D) Permanent files: NHIMS.FORMATS

E) The module generates one complete _NULL_ DATA step.

4.75.3 Internal Data Description

No local macro variables are used in this module.

SAS variables:
PAGE_NUM (num): current page number for heading
MOVE (num): constant used to position data values
ROW (num): identifies current row for printing

```
COLUMN    (num): identifies current column for printing
NEW_COL   (num): identifies column for printing daily code
H         (num): control variable for hourly DO loop
D         (num): control variable for daily DO loop
LAST.MONTH (boolean): identifies last occurrence of current
                      value of MONTH
EOF   (boolean): identifies last record read from input file
```

4.75.4 Design Language Description

```
DATA _NULL_;  /* step to print hourly precipitation data */
  RETAIN PAGE_NUM 1 MOVE -1;
  ARRAY HPCP[24] HPCP1-HPCP24;
  ARRAY H_CODE[24] H_CODE1-H_CODE24;

  FILE PRINT N=PS NOTITLES;
  SET HPCP END=EOF;
    BY STATION YEAR MONTH;

  IF FIRST.MONTH  THEN
     DO;       /* print headings and left-margin day numbers */

       PUT #1 @110 'PAGE ' PAGE_NUM;
       PUT #4 @9 NAME  @60 COUNTY   @100 'STATION NO.  '
                 STN_FORM;
       %IF &METRIC = 0 %THEN
             PUT #6 @45
                   'HOURLY PRECIPITATION IN INCHES';
       %ELSE
             PUT #6 @40
             'HOURLY PRECIPITATION IN MILLIMETERS';

       PUT #7 @1  MONTH MNTH.  +MOVE ',  '  YEAR ;
       COLUMN = 6;
       DO H = 1 TO 24;  /* print hour numbers in heading  */
          PUT #8 @COLUMN H Z2. ;
          COLUMN = COLUMN + 5;
       END;
       PUT #8  @127 'TOTAL';

       ROW = 9;
       DO D = 1 TO 31; /* print 1-31 in left margin */
          ROW = ROW + 1;
          PUT #ROW @2 D 2.;
          IF MOD(D,5) = 0 THEN
               ROW = ROW + 1;
       END;

     END; /* beginning new page, calculate and print headings */
```

```
IF 1 _= DAY _= 31 THEN  /* valid day, print values */

   /* calculate row for output of current record */

   IF DAY _= 5 THEN
      ROW = DAY + 9;
   ELSE IF DAY _= 10 THEN
      ROW = DAY + 10;
   ELSE IF DAY _= 15 THEN
      ROW = DAY + 11;
   ELSE IF DAY _= 20 THEN
      ROW = DAY + 12;
   ELSE IF DAY _= 25 THEN
      ROW = DAY + 13;
   ELSE IF DAY _= 30 THEN
      ROW = DAY + 14;
   ELSE
      ROW = DAY + 15;

   DO H = 1 TO 24;   /* print hourly values */
      COLUMN = ((H - 1) * 5) + 6;

      IF HPCP[H] ¢ 0  THEN  /* print non-zero value */
         PUT #ROW @COLUMN HPCP[H]  H_CODE[H] $CHAR1. ;
      ELSE IF DAYSUM NE 0  THEN
         NEW_COL = COLUMN + 2;
         PUT #ROW @NEW_COL  H_CODE[H] $CHAR1. ;
   END;  /* print hourly values */

   PUT #ROW @127 HPCP_TOT ;

END;  /* valid day, print values */

IF LAST.MONTH  THEN /* print footnotes, start new page */

   PUT #52 @1 'ZERO VALUES ARE PRINTED AS BLANKS.';
   PUT #53 @1 'M:  MISSING HOURLY VALUE.';
   PUT #54 @1 'T:  TRACE AMOUNT.';
   PUT #55 @1 'A:  HOURLY VALUE WAS ACCUMULATED.';
   PUT #56 @1 'E:  HOURLY VALUE WAS ESTIMATED.';
   PUT #57 @1 'B:  HOURLY VALUE WAS BOTH ACCUMULATED AND '
              'ESTIMATED.';
   PUT #58 @1 'S:  MELTING SNOW IN HOURLY MEASUREMENT.';
   PUT #59 @1 '*** CAUTION ***    MUCH OF THE DATA '
              'BEFORE 1970 IS NOT FLAGGED FOR '
              'SPECIAL CONDITIONS.'
   PUT _PAGE_;

IF EOF THEN STOP;

RUN;
```

4.76 Description of Module COPY (COPYY)

4.76.1 Processing Narrative

COPY checks the user requests and calls the proper macros to copy the selected data to an external file.

4.76.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:
C_DIRCT: identifies request for copy direct operation

No SAS data sets are created or used.

C) The macros called by this module are:

COPY DIRECT (4.77 - C_DIR)
COPY TO AN EXTERNAL FILE (4.79 - C_EXT)

D) No permanent files are used by this module.

E) The module generates macro statements only.

4.76.3 Internal Data Description

No local data are used in this module.

4.76.4 Design Language Description

```
%IF request for direct copy  %THEN
    %COPY DIRECT
%ELSE
    %COPY TO AN EXTERNAL FILE
```

4.77 Description of Module COPY DIRECTLY TO A SAS DATA SET (C_DIR)

4.77.1 Processing Narrative

COPY DIRECT checks the requested elements and calls the proper macro actually perform the copy operation. The specified data is always copied to an external SAS data set. After each data set is copied, the file number is incremented by one for subsequent copies.

4.77.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:

FN	: file number of external file for copy output
EL_PRCP	: identifies request for precipitation data
EL_TEMP	: identifies request for temperature data
EL_STRM	: identifies request for streamflow data
EL_SNOW	: identifies request for snowfall data
EL_EVAP	: identifies request for evaporation data
EL_MNTH	: identifies request for monthly summary datra
EL_SNOC	: identifies request for snow course data
EL_PEAK	: identifies request for peak flow data
EL_RESV	: identifies request for reservoir data
EL_HPRP	: identifies request for hourly precipitation data

No SAS data sets are created or used.

C) The macros called by this module are:

COPY ONE SAS DATA SET (4.78 - CD_SAS)

D) No permanent files are used by this module.

E) The module generates macro statements only.

4.77.3 Internal Data Description

No local data are used in this module.

4.77.4 Design Language Description

```
%IF request for precipitation data %THEN
   %COPY ONE SAS DATA SET (PRCP,PRECIPITATION)
   %LET FN = %EVAL(&FN + 1);

%IF request for temperature data %THEN
   %COPY ONE SAS DATA SET (TEMP,TEMPERATURE)
   %LET FN = %EVAL(&FN + 1);

%IF request for streamflow data %THEN
   %COPY ONE SAS DATA SET (STRM,STREAMFLOW)
   %LET FN = %EVAL(&FN + 1);

%IF request for snowfall data %THEN
   %COPY ONE SAS DATA SET (SNOW,SNOWFALL)
   %LET FN = %EVAL(&FN + 1);
```

```
%IF request for evaporation data %THEN
   %COPY ONE SAS DATA SET (EVAP,EVAPORATION)
   %LET FN = %EVAL(&FN + 1);

%IF request for monthly summary data %THEN
   %COPY ONE SAS DATA SET (MNTH,MONTHLY SUMMARY)
   %LET FN = %EVAL(&FN + 1);

%IF request for snow course data %THEN
   %COPY ONE SAS DATA SET (SNOC,SNOW COURSE)
   %LET FN = %EVAL(&FN + 1);

%IF request for peak flow data %THEN
   %COPY ONE SAS DATA SET (PEAK,PEAKFLOW)
   %LET FN = %EVAL(&FN + 1);

%IF request for reservoir data %THEN
   %COPY ONE SAS DATA SET (RESV,RESERVOIR)
   %LET FN = %EVAL(&FN + 1);

%IF request for hourly precipitation data %THEN
   %COPY ONE SAS DATA SET (HPRP,HOURLY PRECIPITATION)
   %LET FN = %EVAL(&FN + 1);
```

4.78 Description of Module COPY ONE SAS DATA SET (CD_SAS)

4.78.1 Processing Narrative

COPY ONE SAS DATA SET sends a SAS data set to an external file. The DCB information for the external file is the SAS default. A message identifying which stations were printed to which file is sent to the print file.

4.78.2 Interface Description

A) Parameter: ELEMENT - 4-character element name, which is the name of the input file
E_TITLE - full name of element for heading

B) Global data:
Macro variables:
FN : file number of external file for copy output

SAS data sets:
&ELEMENT (input) - any variables are allowed

OUT&FN..&ELEMENT (output) - a copy of the input file sent to external ddname of OUT&FN and SAS data set name of &ELEMENT

PRINT (output) - standard SAS print file, used for messages.

C) No macros are called by this module.

D) No permanent files are used by this module.

E) The module generates one complete DATA step.

4.78.3 Internal Data Description

No local macro variables are used in this module.

SAS variables:
FIRST.STATION and LAST.STATION

4.78.4 Design Language Description

```
DATA OUT&FN..&ELEMENT;
   FILE PRINT;
   SET &ELEMENT;
      BY STATION;
   IF FIRST.STATION  THEN
      put statement to the message file identifying which
      stations were written to which external file, and the
      beginning year and month;
   IF LAST.STATION  THEN
      put statement to the message file identifying the ending
      year and month.
```

4.79 Description of Module COPY TO AN EXTERNAL FILE (C_EXT)

4.79.1 Processing Narrative

COPY TO AN EXTERNAL FILE checks the elements request by the user calls the appropriate macro to copy the data to an external file. After each element has been copied, the global macro variable which identifies the external file is incremented.

4.79.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:

FN	: file number of external file for copy output
EL_PRCP	: identifies request for precipitation data
EL_TEMP	: identifies request for temperature data
EL_STRM	: identifies request for streamflow data
EL_SNOW	: identifies request for snowfall data
EL_EVAP	: identifies request for evaporation data
EL_MNTH	: identifies request for monthly summary datra
EL_SNOC	: identifies request for snow course data
EL_PEAK	: identifies request for peak flow data
EL_RESV	: identifies request for reservoir data
EL_HPRP	: identifies request for hourly precipitation data

No SAS data sets are created or used.

C) The macros called by this module are:

```
%COPY PRECIPITATION DATA (4.80 - CP_PRCP)
%COPY TEMPERATURE DATA   (4.81 - CP_TEMP)
%COPY STREAMFLOW DATA    (4.82 - CP_STRM)
%COPY SNOWFALL DATA      (4.83 - CP_SNOW)
%COPY EVAPORATION DATA   (4.84 - CP_EVAP)
%COPY SNOW COURSE DATA   (4.85 - CP_SNOC)
%COPY MONTHLY SUMMARY DATA (4.86 - CP_MNTH)
%COPY PEAKFLOW DATA      (4.87 - CP_PEAK)
%COPY RESERVOIR DATA     (4.88 - CP_RESV)
%COPY HOURLY PRECIPITATION DATA (4.89 - CP_HPCP)
```

D) No permanent files are used by this module.

E) The module generates macro statements only.

4.79.3 Internal Data Description

No local data are used in this module.

4.79.4 Design Language Description

```
%IF request for precipitation data %THEN
    %COPY PRECIPITATION DATA
    %LET FN = %EVAL(&FN + 1);

%IF request for temperature data %THEN
    %COPY TEMPERATURE DATA
    %LET FN = %EVAL(&FN + 1);

%IF request for streamflow data %THEN
    %COPY STREAMFLOW DATA
    %LET FN = %EVAL(&FN + 1);
```

```
%IF request for snowfall data %THEN
    %COPY SNOWFALL DATA
    %LET FN = %EVAL(&FN + 1);

%IF request for evaporation data %THEN
    %COPY EVAPORATION DATA
    %LET FN = %EVAL(&FN + 1);

%IF request for monthly summary data  %THEN
    %COPY MONTHLY SUMMARY DATA
    %LET FN = %EVAL(&FN + 1);

%IF request for snow course data %THEN
    %COPY SNOW COURSE DATA
    %LET FN = %EVAL(&FN + 1);

%IF request for reservoir data %THEN
    %COPY RESERVOIR DATA
    %LET FN = %EVAL(&FN + 1);

%IF request for peak flow data %THEN
    %COPY PEAK FLOW DATA
    %LET FN = %EVAL(&FN + 1);

%IF request for hourly precipitation data %THEN
    %COPY HOURLY PRECIPITATION DATA
    %LET FN = %EVAL(&FN + 1);
```

4.80 Description of Module COPY PRECIPITATION DATA (CP_PRCP)

4.80.1 Processing Narrative

COPY PRECIPITATION DATA sends the precipitation subset to an external file. The data is written in either formatted records of 80 or 380 bytes, or unformatted records. A message is written to identify which stations were copied, along with the beginning and ending dates. If a station number has alphabetic characters, a warning message is written; in the case of unformatted records, a zero value is written in place of the alphanumeric station number.

4.80.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:
FN : file number of external file for copy output

SAS data sets:
PRCP (input) - with variables:
STATION YEAR MONTH STN_FORM NAME COUNTY AREA ELEV
MONTHSUM PRCP_TOT PRECIP1-PRECIP31 P_CODE1-P_CODE31

OUT&FN (output) - external file, may be temporary or permanent

PRINT (output) - standard SAS print file

C) No macros are called by this module.

D) No permanent SAS data sets are used.

E) The module generates one _NULL_ DATA step.

4.80.3 Internal Data Description

No local macro variables are used in this module.

SAS variables:

```
STAT_NO (num): numeric version of the station number
DAY     (num): zero value for the day
NUMDAYS (num): number of days in the current month
BLANKS (char): string of blank characters
ZERO    (num): zero value for non-applicable data
LEGAL   (num): identifies position of alphabetic characters in
               STATION
FIRST.STATION (boolean): identifies first occurrence of current
                         value of STATION
LAST.STATION  (boolean): identifies las occurrence of current
                         value of STATION
```

4.80.4 Design Language Description

```
DATA _NULL_;  /* copies precipitation data to an external file */

   LENGTH STAT_NO 6 ;
   ARRAY  PRECIP[31]  PRECIP1-PRECIP31;
   SET PRCP ;
      BY STATION;

   DAY = 0;
   NUMDAYS = 31;
   IF MONTH = 4 OR MONTH = 6 OR MONTH = 9 OR MONTH = 11
      THEN NUMDAYS = 30;
   ELSE IF MONTH = 2 THEN
                    IF MOD(YEAR,4) = 0 THEN NUMDAYS = 29;
                                       ELSE NUMDAYS = 28;

   BLANKS = '     ';
   ZERO = 0;
   LEGAL = INDEXC(STATION,'all alphabetic characters');
```

```
FILE PRINT;
IF FIRST.STATION THEN
   put NAME  COUNTY  STN_FORM  YEAR MONTH out to message file;
   IF LEGAL NE 0  THEN
      %IF request for formatted data  %THEN
         put message that the station number contains
         character values; be sure to read them in with a
         character format;

      %ELSE %IF request for unformatted data %THEN
         put message that the station number contains
         character values and cannot be written as an
         unformatted data item; instead, a zero value will
         be written for the station number;

IF LAST.STATION
   put YEAR MONTH out to message file;


%IF request for formatted data %THEN
    %IF request for 80-byte records %THEN
        FILE  OUT&FN  LRECL=80  BLKSIZE = 9440;
        PUT BLANKS $4.  STATION  $6.
            (YEAR MONTH DAY MONTHSUM NUMDAYS) (5*10.)
             PRCP_TOT 10.2  ZERO 10.;
        DO I = 1 TO 31;
           PUT (PRECIP[I])  (8*10.2)  @;
        END;
        PUT;

    %ELSE
        FILE  OUT&FN  LRECL=380  BLKSIZE = 4560;
        PUT BLANKS $4.  STATION  $6.
            (YEAR MONTH DAY MONTHSUM NUMDAYS) (5*10.)
             PRCP_TOT 10.2  ZERO 10. @;
        DO I = 1 TO 31;
           PUT PRECIP[I]  10.2  @;
        END;
        PUT;

%ELSE  /* request for unformatted data */
     FILE  OUT&FN  LRECL=284  BLKSIZE = 9372;
     IF LEGAL = 0  THEN
        STAT_NO = STATION;
     ELSE
        STAT_NO = 0;
```

```
          PUT (STAT_NO YEAR MONTH DAY MONTHSUM NUMDAYS PRCP_TOT
               ZERO)   (8*RB4.) @;
          DO I = 1 TO 31;
             PUT PRECIP[I]  RB4.  @;
          END;
          PUT;
RUN;
```

4.81 Description of Module COPY TEMPERATURE DATA (CP_TEMP)

4.81.1 Processing Narrative

COPY TEMPERATURE DATA sends the temperature subset to an external file. The data is written in either formatted records of 80 or 380 bytes, or unformatted records. A message is written to identify which stations were copied, along with the beginning and ending dates. If a station number has alphabetic characters, a warning message is written; in the case of unformatted records, a zero value is written in place of the alphanumeric station number.

4.81.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:
  FN : file number of external file for copy output

SAS data sets:
  TEMP (input) - with variables:
       STATION YEAR MONTH STN_FORM NAME COUNTY AREA ELEV
       MONTHSUM AVEMAX AVEMIN MAXTMP1-MAXTMP31
       MINTMP1-MINTMP31

  OUT&FN (output) - external file, may be temporary or permanent

  PRINT (output) - standard SAS print file

C) No macros are called by this module.

D) No permanent SAS data sets are used.

E) The module generates one _NULL_ DATA step.

4.81.3 Internal Data Description

No local macro variables are used in this module.

```
SAS variables:
   STAT_NO (num): numeric version of the station number
   DAY     (num): zero value for the day
   NUMDAYS (num): number of days in the current month
   BLANKS (char): string of blank characters
   LEGAL   (num): identifies position of alphabetic characters in
                  STATION
   FIRST.STATION (boolean): identifies first occurrence of current
                            value of STATION
   LAST.STATION  (boolean): identifies las occurrence of current
                            value of STATION
```

4.81.4 Design Language Description

```
 DATA _NULL_;  /* copies temperature data to an external file */

    LENGTH STAT_NO 6 ;
    ARRAY  MAXTMP[31]  MAXTMP1-MAXTMP31;
    ARRAY  MINTMP[31]  MINTMP1-MINTMP31;
    SET TEMP ;
       BY STATION;

    DAY = 0;
    NUMDAYS = 31;
    IF MONTH = 4 OR MONTH = 6 OR MONTH = 9 OR MONTH = 11
       THEN NUMDAYS = 30;
    ELSE IF MONTH = 2 THEN
                     IF MOD(YEAR,4) = 0 THEN NUMDAYS = 29;
                                        ELSE NUMDAYS = 28;

    BLANKS = '     ';
    LEGAL = INDEXC(STATION,'all alphabetic characters');

    FILE PRINT;
    IF FIRST.STATION THEN
       put NAME  COUNTY  STN_FORM  YEAR MONTH out to message file;
       IF LEGAL NE 0  THEN
          %IF request for formatted data  %THEN
             put message that the station number contains
             character values; be sure to read them in with a
             character format;

          %ELSE %IF request for unformatted data %THEN
             put message that the station number contains
             character values and cannot be written as an
             unformatted data item; instead, a zero value will
             be written for the station number;

    IF LAST.STATION
       put YEAR MONTH out to message file;
```

```
%IF request for formatted data %THEN
    %IF request for 80-byte records %THEN
        FILE  OUT&FN  LRECL=80  BLKSIZE = 9440;
        PUT BLANKS $4.  STATION  $6.
            (YEAR MONTH DAY MONTHSUM NUMDAYS) (5*10.)
             AVEMAX 10.1  AVEMIN 10.1;
        DO I = 1 TO 31;
           PUT (MAXTMP[I] MINTMP[I]) (16*5.)   @;
        END;
        PUT;

    %ELSE
        FILE  OUT&FN  LRECL=380  BLKSIZE = 4560;
        PUT BLANKS $4.  STATION  $6.
            (YEAR MONTH DAY MONTHSUM NUMDAYS) (5*10.)
             AVEMAX 10.1  AVEMIN 10.1  @;
        DO I = 1 TO 31;
           PUT MAXTMP[I] 5.  MINTMP[I] 5.  @;
        END;
        PUT;

%ELSE  /* request for unformatted data */
     FILE  OUT&FN  LRECL=284  BLKSIZE = 9372;
     IF LEGAL = 0  THEN
        STAT_NO = STATION;
     ELSE
        STAT_NO = 0;

     PUT (STAT_NO YEAR MONTH DAY MONTHSUM NUMDAYS AVEMAX
          AVEMIN)   (8*RB4.) @;
     DO I = 1 TO 31;
        PUT MAXTMP[I]  RB4.  MINTMP[I] RB4.  @;
     END;
     PUT;
RUN;
```

4.82 Description of Module COPY STREAMFLOW DATA (CP_STRM)

4.82.1 Processing Narrative

COPY STREAMFLOW DATA sends the streamflow subset to an external file. The data is written in either formatted records of 80 or 380 bytes, or unformatted records. A message is written to identify which stations were copied, along with the beginning and ending dates. If a station number has alphabetic characters, a warning message is written; in the case of unformatted records, a zero value is written in place of the alphanumeric station number.

4.82.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:
FN : file number of external file for copy output

SAS data sets:
STRM (input) - with variables:
STATION YEAR MONTH STN_FORM NAME COUNTY AREA ELEV
MONTHSUM STRM_TOT FLOW1-FLOW31

OUT&FN (output) - external file, may be temporary or permanent

PRINT (output) - standard SAS print file

C) No macros are called by this module.

D) No permanent SAS data sets are used.

E) The module generates one _NULL_ DATA step.

4.82.3 Internal Data Description

No local macro variables are used in this module.

SAS variables:
STAT_NO (num): numeric version of the station number
DAY (num): zero value for the day
NUMDAYS (num): number of days in the current month
BLANKS (char): string of blank characters
ZERO (num): zero value for non-applicable data
LEGAL (num): identifies position of alphabetic characters in STATION
FIRST.STATION (boolean): identifies first occurrence of current value of STATION
LAST.STATION (boolean): identifies las occurrence of current value of STATION

4.82.4 Design Language Description

```
DATA _NULL_;  /* copies streamflow data to an external file */

   LENGTH STAT_NO 6 ;
   ARRAY  FLOW[31]  FLOW1-FLOW31;
   SET STRM ;
      BY STATION;
```

```
DAY = 0;
NUMDAYS = 31;
IF MONTH = 4 OR MONTH = 6 OR MONTH = 9 OR MONTH = 11
   THEN NUMDAYS = 30;
ELSE IF MONTH = 2 THEN
                  IF MOD(YEAR,4) = 0 THEN NUMDAYS = 29;
                                     ELSE NUMDAYS = 28;

BLANKS = '  ';
ZERO = 0;
LEGAL = INDEXC(STATION,'all alphabetic characters');

FILE PRINT;
IF FIRST.STATION THEN
   put NAME  COUNTY  STN_FORM  YEAR MONTH out to message file;
   IF LEGAL NE 0  THEN
      %IF request for formatted data  %THEN
         put message that the station number contains
         character values; be sure to read them in with a
         character format;

      %ELSE %IF request for unformatted data %THEN
         put message that the station number contains
         character values and cannot be written as an
         unformatted data item; instead, a zero value will
         be written for the station number;

IF LAST.STATION
   put YEAR MONTH out to message file;


%IF request for formatted data %THEN
    %IF request for 80-byte records %THEN
        FILE  OUT&FN  LRECL=80  BLKSIZE = 9440;
        PUT BLANKS $2.  STATION  $8.
            (YEAR MONTH DAY MONTHSUM NUMDAYS) (5*10.)
             STRM_TOT 10.2  ZERO 10.;
        DO I = 1 TO 31;
           PUT (FLOW[I])  (8*10.2)  @;
        END;
        PUT;

    %ELSE
        FILE  OUT&FN  LRECL=380  BLKSIZE = 4560;
        PUT BLANKS $2.  STATION  $8.
            (YEAR MONTH DAY MONTHSUM NUMDAYS) (5*10.)
             STRM_TOT 10.2  ZERO 10. @;
```

```
            DO I = 1 TO 31;
               PUT FLOW[I]  10.2  @;
            END;
            PUT;

      %ELSE  /* request for unformatted data */
           FILE  OUT&FN  LRECL=284  BLKSIZE = 9372;
           IF LEGAL = 0  THEN
              STAT_NO = STATION;
           ELSE
              STAT_NO = 0;

           PUT (STAT_NO YEAR MONTH DAY MONTHSUM NUMDAYS STRM_TOT
                ZERO)   (8*RB4.) @;
           DO I = 1 TO 31;
              PUT FLOW[I]  RB4.  @;
           END;
           PUT;
RUN;
```

4.83 Description of Module COPY SNOWFALL DATA (CP_SNOW)

4.83.1 Processing Narrative

COPY SNOWFALL DATA sends the snowfall subset to an external file. The data is written in either formatted records of 80 or 380 bytes, or unformatted records. A message is written to identify which stations were copied, along with the beginning and ending dates. If a station number has alphabetic characters, a warning message is written; in the case of unformatted records, a zero value is written in place of the alphanumeric station number.

4.83.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:
FN : file number of external file for copy output

SAS data sets:
SNOW (input) - with variables:
STATION YEAR MONTH STN_FORM NAME COUNTY AREA ELEV
MONTHSUM SNOW_TOT MAXDEPTH SNOW1-SNOW31
DEPTH1-DEPTH31 S_CODE1-S_CODE31 D_CODE1-D_CODE31

OUT&FN (output) - external file, may be temporary or permanent

PRINT (output) - standard SAS print file

C) No macros are called by this module.

D) No permanent SAS data sets are used.

E) The module generates one _NULL_ DATA step.

4.83.3 Internal Data Description

No local macro variables are used in this module.

SAS variables:

```
STAT_NO (num): numeric version of the station number
DAY     (num): zero value for the day
NUMDAYS (num): number of days in the current month
BLANKS (char): string of blank characters
LEGAL   (num): identifies position of alphabetic characters in
               STATION
FIRST.STATION (boolean): identifies first occurrence of current
                         value of STATION
LAST.STATION  (boolean): identifies las occurrence of current
                         value of STATION
```

4.83.4 Design Language Description

```
DATA _NULL_;  /* copies snowfall data to an external file */

   LENGTH STAT_NO 6 ;
   ARRAY  SNOW[31]  SNOW1-SNOW31;
   ARRAY  DEPTH[31]  DEPTH1-DEPTH31;
   SET SNOW ;
      BY STATION;

   DAY = 0;
   NUMDAYS = 31;
   IF MONTH = 4 OR MONTH = 6 OR MONTH = 9 OR MONTH = 11
      THEN NUMDAYS = 30;
   ELSE IF MONTH = 2 THEN
                    IF MOD(YEAR,4) = 0 THEN NUMDAYS = 29;
                                       ELSE NUMDAYS = 28;

   BLANKS = '     ';
   LEGAL = INDEXC(STATION,'all alphabetic characters');

   FILE PRINT;
   IF FIRST.STATION THEN
      put NAME  COUNTY  STN_FORM  YEAR MONTH out to message file;
      IF LEGAL NE 0  THEN
```

```
          %IF request for formatted data  %THEN
             put message that the station number contains
             character values; be sure to read them in with a
             character format;

          %ELSE %IF request for unformatted data %THEN
             put message that the station number contains
             character values and cannot be written as an
             unformatted data item; instead, a zero value will
             be written for the station number;

       IF LAST.STATION
          put YEAR MONTH out to message file;


       %IF request for formatted data %THEN
           %IF request for 80-byte records %THEN
              FILE  OUT&FN  LRECL=80  BLKSIZE = 9440;
              PUT BLANKS $4.  STATION  $6.
                  (YEAR MONTH DAY MONTHSUM NUMDAYS) (5*10.)
                   SNOW_TOT 10.1  MAXDEPTH 10.;
              DO I = 1 TO 31;
                 PUT SNOW[I] 5.1  DEPTH[I] 5.   @;
              END;
              PUT;

           %ELSE
              FILE  OUT&FN  LRECL=380  BLKSIZE = 4560;
              PUT BLANKS $4.  STATION  $6.
                  (YEAR MONTH DAY MONTHSUM NUMDAYS) (5*10.)
                   SNOW_TOT 10.1  MAXDEPTH 10.  @;
              DO I = 1 TO 31;
                 PUT SNOW[I] 5.1  DEPTH[I] 5.  @;
              END;
              PUT;

       %ELSE  /* request for unformatted data */
            FILE  OUT&FN  LRECL=284  BLKSIZE = 9372;
            IF LEGAL = 0  THEN
               STAT_NO = STATION;
            ELSE
               STAT_NO = 0;

            PUT (STAT_NO YEAR MONTH DAY MONTHSUM NUMDAYS SNOW_TOT
                 MAXDEPTH)   (8*RB4.) @;
            DO I = 1 TO 31;
               PUT SNOW[I]  RB4.  DEPTH[I] RB4.  @;
            END;
            PUT;
RUN;
```

4.84 Description of Module COPY EVAPORATION DATA (CP_EVAP)

4.84.1 Processing Narrative

COPY EVAPORATION DATA sends the evaporation subset to an external file. The data is written in either formatted records of 80 or 380 bytes, or unformatted records. A message is written to identify which stations were copied, along with the beginning and ending dates. If a station number has alphabetic characters, a warning message is written; in the case of unformatted records, a zero value is written in place of the alphanumeric station number.

4.84.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:
FN : file number of external file for copy output

SAS data sets:
EVAP (input) - with variables:
STATION YEAR MONTH STN_FORM NAME COUNTY AREA ELEV
MONTHSUM EVAP_TOT WIND_TOT EVAP1-EVAP31
WIND1-WIND31 E_CODE1-E_CODE31 W_CODE1-W_CODE31

OUT&FN (output) - external file, may be temporary or permanent

PRINT (output) - standard SAS print file

C) No macros are called by this module.

D) No permanent SAS data sets are used.

E) The module generates one _NULL_ DATA step.

4.84.3 Internal Data Description

No local macro variables are used in this module.

SAS variables:
STAT_NO (num): numeric version of the station number
DAY (num): zero value for the day
NUMDAYS (num): number of days in the current month
BLANKS (char): string of blank characters
LEGAL (num): identifies position of alphabetic characters in STATION

```
FIRST.STATION (boolean): identifies first occurrence of current
                         value of STATION
LAST.STATION  (boolean): identifies las occurrence of current
                         value of STATION
```

4.84.4 Design Language Description

```
DATA _NULL_;  /* copies evaporation data to an external file */

   LENGTH STAT_NO 6 ;
   ARRAY  EVAP[31]  EVAP1-EVAP31;
   ARRAY  WIND[31]  WIND1-WIND31;
   SET EVAP ;
      BY STATION;

   DAY = 0;
   NUMDAYS = 31;
   IF MONTH = 4 OR MONTH = 6 OR MONTH = 9 OR MONTH = 11
      THEN NUMDAYS = 30;
   ELSE IF MONTH = 2 THEN
                    IF MOD(YEAR,4) = 0 THEN NUMDAYS = 29;
                                       ELSE NUMDAYS = 28;

   BLANKS = '    ';
   LEGAL = INDEXC(STATION,'all alphabetic characters');

   FILE PRINT;
   IF FIRST.STATION THEN
      put NAME  COUNTY  STN_FORM  YEAR MONTH out to message file;
      IF LEGAL NE 0  THEN
         %IF request for formatted data  %THEN
            put message that the station number contains
            character values; be sure to read them in with a
            character format;

         %ELSE %IF request for unformatted data %THEN
            put message that the station number contains
            character values and cannot be written as an
            unformatted data item; instead, a zero value will
            be written for the station number;

   IF LAST.STATION
      put YEAR MONTH out to message file;


   %IF request for formatted data %THEN
       %IF request for 80-byte records %THEN
           FILE  OUT&FN  LRECL=80  BLKSIZE = 9440;
           PUT BLANKS $4.  STATION  $6.
               (YEAR MONTH DAY MONTHSUM NUMDAYS) (5*10.)
                EVAP_TOT 10.2  WIND_TOT 10.;
```

```
            DO I = 1 TO 31;
               PUT EVAP[I] 5.2  WIND[I] 5.   @;
            END;
            PUT;

         %ELSE
            FILE  OUT&FN  LRECL=380  BLKSIZE = 4560;
            PUT BLANKS $4.  STATION  $6.
                (YEAR MONTH DAY MONTHSUM NUMDAYS) (5*10.)
                 EVAP_TOT 10.2  WIND_TOT 10.  @;
            DO I = 1 TO 31;
               PUT EVAP[I] 5.2  WIND[I] 5.  @;
            END;
            PUT;

      %ELSE  /* request for unformatted data */
           FILE  OUT&FN  LRECL=284  BLKSIZE = 9372;
           IF LEGAL = 0  THEN
              STAT_NO = STATION;
           ELSE
              STAT_NO = 0;

           PUT (STAT_NO YEAR MONTH DAY MONTHSUM NUMDAYS EVAP_TOT
                WIND_TOT)   (8*RB4.) @;
           DO I = 1 TO 31;
              PUT EVAP[I]  RB4.  WIND[I] RB4.  @;
           END;
           PUT;
RUN;
```

4.85 Description of Module COPY SNOW COURSE DATA (CP_SNOC)

4.85.1 Processing Narrative

COPY SNOW COURSE DATA sends the snow course subset to an external file. The data is written in either formatted records of 80 or 380 bytes, or unformatted records. A message is written to identify which stations were copied, along with the beginning and ending dates. If the user request unformatted records, a warning message is written, and a zero value is written in place of the alphanumeric station number.

4.85.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:
FN : file number of external file for copy output

SAS data sets:
SNOC (input) - with variables:
STATION YEAR CARDNO STN_FORM NAME COUNTY AREA ELEV
MONTH1-MONTH6 DAY1-DAY6 DEPTH1-DEPTH6 WATER1-WATER6

OUT&FN (output) - external file, may be temporary or permanent

PRINT (output) - standard SAS print file

C) No macros are called by this module.

D) No permanent SAS data sets are used.

E) The module generates one _NULL_ DATA step.

4.85.3 Internal Data Description

No local macro variables are used in this module.

SAS variables:
STAT_NO (num): numeric version of the station number
NUM_MEAS(num): number of measurement in the year
BLANKS (char): string of blank characters
ZERO (num): zero value for non-applicable data
FIRST.STATION (boolean): identifies first occurrence of current value of STATION
LAST.STATION (boolean): identifies las occurrence of current value of STATION

4.85.4 Design Language Description

```
DATA _NULL_;  /* copies snow course data to an external file */

   LENGTH STAT_NO 6 ;
   ARRAY  MONTH[6]  MONTH1-MONTH6;
   ARRAY  DAY[6]    DAY1-DAY6;
   ARRAY  DEPTH[6]  DEPTH1-DEPTH6;
   ARRAY  WATER[6]  WATER1-WATER6;
   SET SNOC ;
      BY STATION;

   NUM_MEAS = 6;
   ZERO = 0;
   BLANKS = '      ';
```

```
FILE PRINT;
IF FIRST.STATION THEN
   put NAME  COUNTY  STN_FORM  YEAR  out to message file;
   %IF request for formatted data  %THEN
      put message that the station number contains
      character values; be sure to read them in with a
      character format;

   %ELSE %IF request for unformatted data %THEN
      put message that the station number contains
      character values and cannot be written as an
      unformatted data item; instead, a zero value will
      be written for the station number;

IF LAST.STATION
   put YEAR  out to message file;

%IF request for formatted data %THEN
    %IF request for 80-byte records %THEN
        FILE  OUT&FN  LRECL=80  BLKSIZE = 9440;
        PUT BLANKS $5.  STATION  $5.
            (YEAR ZERO ZERO CARDNO NUM_MEAS ZERO ZERO)
            (7*10.);
        DO I = 1 TO 6;
           PUT MONTH[I] 10.  DAY[I] 10.
              DEPTH[I] 10.  WATER[I] 10.1  @;
        END;
        PUT;

    %ELSE
        FILE  OUT&FN  LRECL=380  BLKSIZE = 4560;
        PUT BLANKS $5.  STATION  $5.
            (YEAR ZERO ZERO CARDNO NUM_MEAS ZERO ZERO)
            (7*10.)  @;
        DO I = 1 TO 6;
           PUT MONTH[I] 10.  DAY[I] 10.
              DEPTH[I] 10.  WATER[I] 10.1  @;
        END;
        PUT;

%ELSE  /* request for unformatted data */
     FILE  OUT&FN  LRECL=284  BLKSIZE = 9372;
     STAT_NO = 0;
     PUT (STAT_NO YEAR ZERO ZERO CARDNO NUM_MEAS ZERO ZERO)
          8*RB4.) @;
     DO I = 1 TO 6;
          PUT (MONTH[I]  DAY[I]  DEPTH[I]  WATER[I])
              (24*RB4.)  @;
     END;
     PUT;
RUN;
```

4.86 Description of Module COPY MONTHLY SUMMARY DATA (CP_MNTH)

4.86.1 Processing Narrative

COPY MONTHLY SUMMARY DATA sends the monthly summary subset to an external file. The data is written in either formatted records of 80 or 380 bytes, or unformatted records. A message is written to identify which stations were copied, along with the beginning and ending dates. If a station number has alphabetic characters, a warning message is written; in the case of unformatted records, a zero value is written in place of the alphanumeric station number.

4.86.2 Interface Description

A) There is no parameter list.

B) Global data:
   Macro variables:
     FN : file number of external file for copy output

   SAS data sets:
     MNTH (input) - with variables:
          STATION YEAR MONTH STN_FORM NAME COUNTY AREA ELEV
          MAXTEMP MINTEMP TMEAN TDEPART HIGHEST HIGHDAY
          LOWEST LOWDAY DEGDAYS COOLDAYS PRECIP PDEPART
          PPTMAX PMAXDAY SNOFALL SNODEPTH SMAXDAY EVAP WIND

     OUT&FN (output) - external file, may be temporary or permanent

     PRINT (output) - standard SAS print file

C) No macros are called by this module.

D) No permanent SAS data sets are used.

E) The module generates one _NULL_ DATA step.

4.86.3 Internal Data Description

No local macro variables are used in this module.

SAS variables:
   STAT_NO (num): numeric version of the station number
   DAY (num): zero value for the day
   NUM_VAL (num): number of values in one month
   BLANKS (char): string of blank characters
   ZERO (num): zero value for non-applicable conditions

```
LEGAL   (num): identifies alphabetic characters in STATION
FIRST.STATION (boolean): identifies first occurrence of current
                         value of STATION
LAST.STATION  (boolean): identifies las occurrence of current
                         value of STATION
```

4.86.4 Design Language Description

```
DATA _NULL_;  /* copies monthly summ data to an external file */

   LENGTH STAT_NO 6 ;
   SET MNTH ;
      BY STATION;

   NUM_VAL = 19;
   DAY = 0;
   BLANKS = '    ';
   ZERO = 0;
   LEGAL = INDEXC(STATION,'all alphabetic characters');

   FILE PRINT;
   IF FIRST.STATION THEN
      put NAME  COUNTY  STN_FORM  YEAR MONTH out to message file;
      IF LEGAL NE 0  THEN
         %IF request for formatted data  %THEN
            put message that the station number contains
            character values; be sure to read them in with a
            character format;

         %ELSE %IF request for unformatted data %THEN
            put message that the station number contains
            character values and cannot be written as an
            unformatted data item; instead, a zero value will
            be written for the station number;

   IF LAST.STATION
      put YEAR MONTH out to message file;

   %IF request for formatted data %THEN
       %IF request for 80-byte records %THEN
           FILE  OUT&FN  LRECL=80  BLKSIZE = 9440;
           PUT BLANKS $4.  STATION  $6.
               (YEAR MONTH DAY  ZERO NUM_VAL ZERO ZERO) (7*10.);
           PUT (MAXTEMP MINTEMP TMEAN TDEPART) (4*10.1)
               (HIGHEST HIGHDAY LOWEST LOWDAY) (4*10.);
           PUT (DEGDAYS COOLDAYS) (2*10.)
               (PRECIP PDEPART PPTMAX) (3*10.2)
               PMAXDAY 10.  SNOFALL  10.1  SNODEPTH  10.;
           PUT SMAXDAY 10.  EVAP 10.2  WIND 10. ;
```

```
        %ELSE
            FILE  OUT&FN  LRECL=380  BLKSIZE = 4560;
            PUT BLANKS $4.  STATION  $6.
                (YEAR MONTH DAY  ZERO NUM_VAL ZERO ZERO) (7*10.) @;
            PUT (MAXTEMP MINTEMP TMEAN TDEPART) (4*10.1)
                (HIGHEST HIGHDAY LOWEST LOWDAY) (4*10.) @;
            PUT (DEGDAYS COOLDAYS) (2*10.)
                (PRECIP PDEPART PPTMAX) (3*10.2)
                PMAXDAY 10.  SNOFALL  10.1  SNODEPTH  10. @;
            PUT SMAXDAY 10.  EVAP 10.2  WIND 10. ;

     %ELSE  /* request for unformatted data */
          FILE  OUT&FN  LRECL=284  BLKSIZE = 9372;
          IF LEGAL = 0  THEN
             STAT_NO = STATION;
          ELSE
             STAT_NO = 0;

          PUT (STAT_NO YEAR MONTH DAY  ZERO NUM_VAL ZERO ZERO)
              (8*RB4.) @;
          PUT (MAXTEMP MINTEMP TMEAN TDEPART
               HIGHEST HIGHDAY LOWEST LOWDAY) (8*RB4.) @;
          PUT (DEGDAYS COOLDAYS PRECIP PDEPART PPTMAX
               PMAXDAY SNOFALL SNODEPTH)  (8*RB4.) @;
          PUT (SMAXDAY EVAP WIND)  (3*RB4.) ;
RUN;
```

4.87 Description of Module COPY PEAKFLOW DATA (CP_PEAK)

4.87.1 Processing Narrative

COPY PEAKFLOW DATA sends the peakflow subset to an external file. The data is written in either formatted records of 80 or 380 bytes, or unformatted records. A message is written to identify which stations were copied, along with the beginning and ending dates. If a station number has alphabetic characters, a warning message is written; in the case of unformatted records, a zero value is written in place of the alphanumeric station number.

4.87.2 Interface Description

A) There is no parameter list.

B) Global data:
   Macro variables:
     FN : file number of external file for copy output

SAS data sets:
RESV (input) - with variables:
STATION WAT_YEAR STN_FORM NAME COUNTY AREA ELEV YEAR
MONTH DAY PEAKFLOW GAGE_HT GH_CODE PK_CODE RD_CODE

OUT&FN (output) - external file, may be temporary or permanent

PRINT (output) - standard SAS print file

C) No macros are called by this module.

D) No permanent SAS data sets are used.

E) The module generates one _NULL_ DATA step.

4.87.3 Internal Data Description

No local macro variables are used in this module.

SAS variables:

```
STAT_NO (num): numeric version of the station number
GH      (num): numeric version of gage height code
PK      (num): numeric version of peak flow code
RD      (num): numeric version of regulation and diversion code
BLANKS (char): string of blank characters
ZERO    (num): zero value for non-applicable data
LEGAL   (num): identifies position of alphabetic characters in
               STATION
FIRST.STATION (boolean): identifies first occurrence of current
                         value of STATION
LAST.STATION  (boolean): identifies las occurrence of current
                         value of STATION
```

4.87.4 Design Language Description

```
DATA _NULL_;  /* copies peakflow data to an external file */

  LENGTH STAT_NO 6  GH 2  PK 2  RD 2;
  SET PEAK ;
     BY STATION;
  BLANKS = '  ';
  ZERO = 0;
  LEGAL = INDEXC(STATION,'all alphabetic characters');
  GH = GH_CODE;
  PK = PK_CODE;
  RD = RD_CODE;

  FILE PRINT;
  IF FIRST.STATION THEN
     put NAME  COUNTY  STN_FORM  WAT_YEAR out to message file;
     IF LEGAL NE 0  THEN
```

```
            %IF request for formatted data  %THEN
               put message that the station number contains
               character values; be sure to read them in with a
               character format;

            %ELSE %IF request for unformatted data %THEN
               put message that the station number contains
               character values and cannot be written as an
               unformatted data item; instead, a zero value will
               be written for the station number;

      IF LAST.STATION
         put WAT_YEAR out to message file;

      %IF request for formatted data %THEN
          %IF request for 80-byte records %THEN
              FILE  OUT&FN  LRECL=80  BLKSIZE = 9440;
              PUT BLANKS $2.  STATION  $8.
                  (YEAR MONTH DAY GH  PK  RD  ZERO) (7*10.);
              PUT PEAKFLOW 10.  GAGE_HT 10.2  ;
          %ELSE
              FILE  OUT&FN  LRECL=380  BLKSIZE = 4560;
              PUT BLANKS $2.  STATION  $8.
                  (YEAR MONTH DAY GH  PK  RD  ZERO) (7*10.) @;
              PUT PEAKFLOW 10.  GAGE_HT 10.2  ;

      %ELSE  /* request for unformatted data */
           FILE  OUT&FN  LRECL=284  BLKSIZE = 9372;
           IF LEGAL = 0  THEN
              STAT_NO = STATION;
           ELSE
              STAT_NO = 0;
           PUT (STAT_NO YEAR MONTH DAY GH PK RD ZERO) (8*RB4.) @;
           PUT PEAKFLOW RB4.  GAGE_HT RB4. ;
RUN;
```

4.88 Description of Module COPY RESERVOIR DATA (CP_RESV)

4.88.1 Processing Narrative

COPY RESERVOIR DATA sends the reservoir subset to an external file. The data is written in either formatted records of 80 or 380 bytes, or unformatted records. A message is written to identify which stations were copied, along with the beginning and ending dates. If a station number has alphabetic characters, a warning message is written; in the case of unformatted records, a zero value is written in place of the alphanumeric station number.

4.88.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:
FN : file number of external file for copy output

SAS data sets:
RESV (input) - with variables:
STATION YEAR MONTH STN_FORM NAME COUNTY AREA ELEV
STORGE1-STORGE31 UNITCODE TIMECODE

OUT&FN (output) - external file, may be temporary or permanent

PRINT (output) - standard SAS print file

C) No macros are called by this module.

D) No permanent SAS data sets are used.

E) The module generates one _NULL_ DATA step.

4.88.3 Internal Data Description

No local macro variables are used in this module.

SAS variables:

```
STAT_NO (num): numeric version of the station number
DAY     (num): zero value for the day
NUMDAYS (num): number of days in the current month
BLANKS (char): string of blank characters
ZERO    (num): zero value for non-applicable data
LEGAL   (num): identifies position of alphabetic characters in
               STATION
FIRST.STATION (boolean): identifies first occurrence of current
                         value of STATION
LAST.STATION  (boolean): identifies las occurrence of current
                         value of STATION
```

4.88.4 Design Language Description

```
DATA _NULL_;  /* copies reservoir data to an external file */

   LENGTH STAT_NO 6 ;
   ARRAY  STORGE[31]  STORGE1-STORGE31;
   SET RESV ;
      BY STATION;
```

```
DAY = 0;
NUMDAYS = 31;
IF MONTH = 4 OR MONTH = 6 OR MONTH = 9 OR MONTH = 11
   THEN NUMDAYS = 30;
ELSE IF MONTH = 2 THEN
                  IF MOD(YEAR,4) = 0 THEN NUMDAYS = 29;
                                     ELSE NUMDAYS = 28;

BLANKS = '  ';
ZERO = 0;
LEGAL = INDEXC(STATION,'all alphabetic characters');

FILE PRINT;
IF FIRST.STATION THEN
   put NAME  COUNTY  STN_FORM  YEAR MONTH out to message file;
   IF LEGAL NE 0  THEN
      %IF request for formatted data  %THEN
         put message that the station number contains
         character values; be sure to read them in with a
         character format;

      %ELSE %IF request for unformatted data %THEN
         put message that the station number contains
         character values and cannot be written as an
         unformatted data item; instead, a zero value will
         be written for the station number;

IF LAST.STATION
   put YEAR MONTH out to message file;


%IF request for formatted data %THEN
    %IF request for 80-byte records %THEN
        FILE  OUT&FN  LRECL=80  BLKSIZE = 9440;
        PUT BLANKS $2.  STATION  $8.
            (YEAR MONTH DAY ZERO NUMDAYS) (5*10.)
             ZERO 10.  ZERO 10.;
        DO I = 1 TO 31;
           PUT (STORGE[I])  (8*10.2)  @;
        END;
        PUT;

    %ELSE
        FILE  OUT&FN  LRECL=380  BLKSIZE = 4560;
        PUT BLANKS $2.  STATION  $8.
            (YEAR MONTH DAY ZERO NUMDAYS) (5*10.)
             ZERO 10.  ZERO 10. @;
        DO I = 1 TO 31;
           PUT STORGE[I]  10.2  @;
        END;
        PUT;
```

```
     %ELSE  /* request for unformatted data */
          FILE  OUT&FN  LRECL=284  BLKSIZE = 9372;
          IF LEGAL = 0  THEN
             STAT_NO = STATION;
          ELSE
             STAT_NO = 0;

          PUT (STAT_NO YEAR MONTH DAY ZERO NUMDAYS ZERO
               ZERO)   (8*RB4.) @;
          DO I = 1 TO 31;
             PUT STORGE[I]  RB4.  @;
          END;
          PUT;
RUN;
```

4.89 Description of Module COPY HOURLY PRECIPITATION DATA (CP_HPCP)

4.89.1 Processing Narrative

COPY HOURLY PRECIPITATION DATA sends the hourly precipitation subset to an external file. The data is written in either formatted records of 80 or 380 bytes, or unformatted records. A message is written to identify which stations were copied, along with the beginning and ending dates. If a station number has alphabetic characters, a warning message is written; in the case of unformatted records, a zero value is written in place of the alphanumeric station number.

4.89.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:
FN : file number of external file for copy output

SAS data sets:
HPCP (input) - with variables:
STATION YEAR MONTH DAY STN_FORM NAME COUNTY AREA ELEV
DAYSUM HPCP_TOT HPCP1-HPCP31 H_CODE1-H_CODE31

OUT&FN (output) - external file, may be temporary or permanent

PRINT (output) - standard SAS print file

C) No macros are called by this module.

D) No permanent SAS data sets are used.

E) The module generates one _NULL_ DATA step.

4.89.3 Internal Data Description

No local macro variables are used in this module.

SAS variables:

```
STAT_NO (num): numeric version of the station number
NUMHRS  (num): number of hours in the day
BLANKS (char): string of blank characters
ZERO    (num): zero value for non-applicable data
LEGAL   (num): identifies position of alphabetic characters in
               STATION
FIRST.STATION (boolean): identifies first occurrence of current
                         value of STATION
LAST.STATION  (boolean): identifies las occurrence of current
                         value of STATION
```

4.89.4 Design Language Description

```
DATA _NULL_;  /* copies hourly precip data to an external file */

   LENGTH STAT_NO 6 ;
   ARRAY  HPCP[24]  HPCP1-HPCP31;
   SET HPCP ;
      BY STATION;

   NUMHRS = 24;
   BLANKS = '    ';
   ZERO = 0;
   LEGAL = INDEXC(STATION,'all alphabetic characters');

   FILE PRINT;
   IF FIRST.STATION THEN
      put NAME  COUNTY  STN_FORM  YEAR MONTH DAY
          out to message file;
      IF LEGAL NE 0  THEN
         %IF request for formatted data  %THEN
            put message that the station number contains
            character values; be sure to read them in with a
            character format;

         %ELSE %IF request for unformatted data %THEN
            put message that the station number contains
            character values and cannot be written as an
            unformatted data item; instead, a zero value will
            be written for the station number;
```

```
      IF LAST.STATION
         put YEAR MONTH DAY out to message file;


      %IF request for formatted data %THEN
          %IF request for 80-byte records %THEN
              FILE  OUT&FN  LRECL=80  BLKSIZE = 9440;
              PUT BLANKS $4.  STATION  $6.
                  (YEAR MONTH DAY DAYSUM NUMHRS) (5*10.)
                   HPCP_TOT 10.2  ZERO 10.;
              DO I = 1 TO 24;
                 PUT (HPCP[I])  (8*10.2)  @;
              END;
              PUT;

          %ELSE
              FILE  OUT&FN  LRECL=380  BLKSIZE = 4560;
              PUT BLANKS $4.  STATION  $6.
                  (YEAR MONTH DAY DAYSUM NUMHRS) (5*10.)
                   HPCP_TOT 10.2  ZERO 10. @;
              DO I = 1 TO 31;
                 PUT HPCP[I]  10.2  @;
              END;
              PUT;

      %ELSE  /* request for unformatted data */
           FILE  OUT&FN  LRECL=284  BLKSIZE = 9372;
           IF LEGAL = 0  THEN
              STAT_NO = STATION;
           ELSE
              STAT_NO = 0;

           PUT (STAT_NO YEAR MONTH DAY DAYSUM NUMHRS HPCP_TOT
                ZERO)   (8*RB4.) @;
           DO I = 1 TO 31;
              PUT HPCP[I]  RB4.  @;
           END;
           PUT;
RUN;
```

4.90 Description of Module PROCESS (PROCESS)

4.90.1 Processing Narrative

PROCESS checks all of the possible process requests available to the user, and calls the proper macro to satisfy the request. If no match is made with any of the available processes, then an error message is generated, and the job is aborted.

4.90.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:

DAISTA	: identifies request for daily statistics process
STA	: identifies request for daily statistics process
MONSTA	: identifies request for monthly statistics process
COR	: identifies request for correlation process
HIG	: identifies request for highest (maximum) process
MAX	: identifies request for highest (maximum) process
LOW	: identifies request for lowest (minimum) process
MIN	: identifies request for lowest (minimum) process
EXT	: identifies request for extreme process
RANORD	: identifies request for rank order process
HIGOCC	: identifies request for high occurrences process
LOWOCC	: identifies request for low occurrences process
DAIOCC	: identifies request for daily occurrences process
OCC	: identifies request for daily occurrences process
MONOCC	: identifies request for monthly occurrences process
FLODURTA	: identifies request for flow duration table process
SUM	: identifies request for summary process
CAL	: identifies request for calendar process

No SAS data sets are created or used, unless the proper process cannot be found; then, an error message is sent to PRINT, the standard SAS print file.

C) The macros called by this module are:

DAILY STATISTICS (4.91 - D_STATS)
MONTHLY STATISTICS (4.95 - M_STATS)
CORRELATION (4.98 - CORR)
HIGHEST (4.101 - HIGHEST)
LOWEST (4.102 - LOWEST)
EXTREME (4.105 - EXTREME)
RANK ORDER (4.107 - RANKORD)
HIGH OCCURRENCES (4.110 - HI_OCC)

LOW OCCURRENCES (4.111 - LO_OCC)
DAILY OCCURRENCES (4.115 - DAY_OCC)
MONTHLY OCCURRENCES (4.116 - MON_OCC)
FLOW DURATION TABLE (4.121 - FLO_DUR)
SUMMARY (4.124 - SUMMARY)
CALENDAR (4.126 - CALENDR)

D) No permanent files are used by this module.

E) The module generates macro statements only.

4.90.3 Internal Data Description

Macro variables:
CHECK : identifies whether or not the requested process has been found

No local SAS variables are used in this module.

4.90.4 Design Language Description

```
%LOCAL CHECK;
%LET CHECK = 0;
%IF request for DAILY STATISTICS process %THEN
    %LET CHECK = 1;
    %DAILY STATISTICS
%IF request for MONTHLY STATISTICS process %THEN
    %LET CHECK = 1;
    %MONTHLY STATISTICS
%IF request for CORRELATION process %THEN
    %LET CHECK = 1;
    %CORRELATION
%IF request for HIGHEST process %THEN
    %LET CHECK = 1;
    %HIGHEST
%IF request for LOWEST process %THEN
    %LET CHECK = 1;
    %LOWEST
%IF request for EXTREME process %THEN
    %LET CHECK = 1;
    %EXTREME
%IF request for RANK ORDER process %THEN
    %LET CHECK = 1;
    %RANK ORDER
%IF request for HIGH OCCURRENCES process %THEN
    %LET CHECK = 1;
    %HIGH OCCURRENCES
%IF request for LOW OCCURRENCES process %THEN
    %LET CHECK = 1;
    %LOW OCCURRENCES
```

```
%IF request for DAILY OCCURRENCES process %THEN
    %LET CHECK = 1;
    %DAILY OCCURRENCES

%IF request for MONTHLY OCCURRENCES process %THEN
    %LET CHECK = 1;
    %MONTHLY OCCURRENCES
%IF request for FLOW DURATION TABLE process %THEN
    %LET CHECK = 1;
    %FLOW DURATION TABLE
%IF request for SUMMARY process %THEN
    %LET CHECK = 1;
    %SUMMARY
%IF request for CALENDAR process %THEN
    %LET CHECK = 1;
    %CALENDAR

%IF NOT &CHECK  %THEN

    DATA _NULL_;
      FILE PRINT;
      Issue error message that although the user requested a
      process operation, the type of process could not be
      identified.
      ABORT;
    RUN;
```

4.91 Description of Module DAILY STATISTICS (D_STATS)

4.91.1 Processing Narrative

DAILY STATISTICS first determines which months are requested in the ONLY option. Then all pertaining elements are checked, subsets are made of the required data, and the statistics are computed. Only one execution of the daily statistics process is allowed per access request.

4.91.2 Interface Description

A) There is no parameter list.

B) Global data:
   Macro variables:
     EL_PRCP : identifies request for precipitation data
     EL_TEMP : identifies request for temperature data
     EL_STRM : identifies request for streamflow data
     EL_SNOW : identifies request for snowfall data
     EL_EVAP : identifies request for evaporation data

   No SAS data sets are created or used.

C) The macros called by this module are:

GET MONTHS FROM ONLY OPTION (4.92 - GET_MON)
MAKE SUBSET OF DAILY VALUES (4.93 - DAY_SUB)
COMPUTE DAILY STATISTICS (4.94 - C_DSTAT)

D) No permanent files are used by this module.

E) The module generates macro statements only.

4.91.3 Internal Data Description

Macro variables:
F_MONTH : first month for which data was requested
F_YEAR : first year for which data was requested
L_MONTH : last month for which data was requested
L_YEAR : last year for which data was requested

No local SAS variables are used in this module.

4.91.4 Design Language Description

```
%LOCAL F_MONTH;
%LOCAL F_YEAR;
%LOCAL L_MONTH;
%LOCAL L_YEAR;

%GET MONTHS FROM ONLY OPTION(DSTATS,DAILY STATISTICS,STATISTICS)

%IF request for precipitation data %THEN
    %MAKE SUBSET OF DAILY VALUES(PRCP,PRECIP)
    %COMPUTE DAILY STATISTICS(PRCP,PRECIPITATION)

%IF request for temperature data %THEN
    %MAKE SUBSET OF DAILY VALUES(TEMP,MAXTMP)
    %COMPUTE DAILY STATISTICS(TEMP,MAXIMUM TEMPERATURES)

    %MAKE SUBSET OF DAILY VALUES(TEMP,MINTMP)
    %COMPUTE DAILY STATISTICS(TEMP,MINIMUM TEMPERATURES)

%IF request for streamflow data %THEN
    %MAKE SUBSET OF DAILY VALUES(STRM,FLOW)
    %COMPUTE DAILY STATISTICS(STRM,STREAMFLOW)

%IF request for snowfall data %THEN
    %MAKE SUBSET OF DAILY VALUES(SNOW,SNOW)
    %COMPUTE DAILY STATISTICS(SNOW,SNOWFALL)
```

```
%IF request for evaporation data %THEN
    %MAKE SUBSET OF DAILY VALUES(EVAP,EVAP)
    %COMPUTE DAILY STATISTICS(EVAP,EVAPORATION)
```

4.92 Description of Module GET MONTHS FROM ONLY OPTION (GET_MON)

4.92.1 Processing Narrative

GET MONTHS FROM ONLY OPTION parses the range of months specified in the ONLY option and creates a data set with the beginning and ending months for that range. If the ONLY option was not given by the user, default values for the range of months are assigned.

4.92.2 Interface Description

A) Parameters: DSN - name of input data set
NAME1 - name of process given by user
NAME2 - alternate name of process given by user

B) Global data:

No global macro variables are used in this module.

SAS data sets:

PROCESS (input) - with variables:
PRC_NAME PARMETER

&DSN (output) - with variables:
BEGMONTH ENDMONTH RANGE

PRINT (output) - standard SAS print file

C) No macros are called by this module.

D) No permanent files are used by this module.

E) The module generates one complete DATA step.

4.92.3 Internal Data Description

No local macro variables are used in this module.

SAS variables:

MON (char): array name for month names and ANNUAL
WORD (char): array name for words scanned in parameter

4.92.4 Design Language Description

```
DATA &DSN (KEEP=BEGMONTH  ENDMONTH  RANGE);
   /* data step to identify the months specified */
   /* in the ONLY option                         */
   LENGTH MON1-MON13 $ 10  WORD1-WORD4  $ 10  RANGE $ 18;

   RETAIN  MON1  'JANUARY'  MON2  'FEBRUARY'  MON3  'MARCH'
           MON4  'APRIL'    MON5  'MAY'       MON6  'JUNE'
           MON7  'JULY'     MON8  'AUGUST'    MON9  'SEPTEMBER'
           MON10 'OCTOBER'  MON11 'NOVEMBER'  MON12 'DECEMBER'
           MON13 'ANNUAL';

   FILE PRINT;
   ARRAY MON[13] MON1-MON13;
   SET PROCESS;

   IF PRC_NAME = "&NAME1"  OR PRC_NAME = "&NAME2"  THEN
      DO;
        BEGMONTH = 1;
        ENDMONTH = 13;
        RANGE = 'JANUARY-ANNUAL';
        IF PARMETER is equal to missing THEN
             OUTPUT;
        ELSE IF PARMETER is not equal to missing THEN DO;
               WORD1 = SCAN(PARMETER,1,' ');
               IF WORD1 = 'ONLY'  THEN
                  DO;
                    WORD2 = SCAN(PARMETER,2,' ');
                    WORD3 = SCAN(PARMETER,3,' ');
                    IF WORD3 = 'TO' THEN
                       WORD4 = SCAN(PARMETER,4,' ');
                       RANGE = TRIM(WORD2) !! '-' !! WORD4;
                    ELSE
                       WORD4 = ' ';
                       RANGE = WORD2;

                    DO I = 1 TO 13;
                       IF WORD2 = MON[I]  THEN
                          BEGMONTH = I;
                    END;
                    IF WORD4 NE ' '  THEN
                       DO I = 1 TO 13;
                          IF WORD4 = MON[I]  THEN
                             ENDMONTH = I;
                       END;
                    ELSE ENDMONTH = BEGMONTH;

                    OUTPUT;
                  END;
 RUN;
```

4.93 Description of Module MAKE SUBSET OF DAILY VALUES (DAY_SUB)

4.93.1 Processing Narrative

MAKE SUBSET OF DAILY VALUES first creates a subset of the requested element data according to the months specified in the ONLY option. Then, the data is reshaped for the daily statistics process; one observation contains the value for one day, and a lag variable is created to retain the previous day's value for correlation.

4.93.2 Interface Description

A) Parameters: ELEMENT - 4-character element name
ARRRAY - name of the daily array in the input file

B) Global data:
Macro variables:
ONLYANN : identifies request for ONLY ANNUAL option
ANNSTAT : identifies request for annual statistics

SAS data sets:
&ELEMENT (input) - with variables:
STATION YEAR MONTH STN_FORM NAME COUNTY
&ARRRAY.1-&ARRRAY.31
(any array name, as long as array has 31 values)

DSTATS (input) - with variables:
BEGMONTH ENDMONTH RANGE

&ELEMENT.MON (input and output) - with variables:
same as &ELEMENT data set

&ELEMENT.STAT (output) - with variables:
STATION MONTH STN_FORM NAME COUNTY &ELEMENT.DAY
LAGDAY

C) No macros are called by this module.

D) No permanent files are used by this module.

E) The module generates two complete DATA steps.

4.93.3 Internal Data Description

```
Macro variables:
   F_MONTH : first month for which data was requested
   F_YEAR  : first year for which data was requested
   L_MONTH : last month for which data was requested
   L_YEAR  : last year for which data was requested

SAS variables:
   I       (num): control variable for DO loop
   CARRY   (num): hold lag value from previous observation
   NUMDAYS(num): number of days in a particular month
   DAY     (num): control variable for DO loop

   FIRST.STATION (boolean): first occurrence of current value of
                            the STATION variable
   LAST.STATION  (boolean): last occurrence of current value of
                            the STATION variable
```

4.93.4 Design Language Description

```
   DATA &ELEMENT.MON(DROP= BEGMONTH ENDMONTH RANGE);

         SET &ELEMENT;

         DO I = 1 TO 1;
            SET DSTATS POINT=I;
            /* check range of dates from only option - if annual */
            /* is requested, ENDMONTH = 13, keep entire record */

            IF RANGE NE 'ANNUAL'  THEN
                 set ONLYANN to 0;/* request more than annuals */
            ELSE set ONLYANN to 1;/* request is ONLY ANNUAL */
            IF ENDMONTH = 13  THEN
               set ANNSTAT to 1;
               OUTPUT;
            ELSE IF BEGMONTH _= MONTH _= ENDMONTH  THEN
                   set ANNSTAT to 0;
                   OUTPUT;
            ELSE IF BEGMONTH ¢ ENDMONTH THEN
                    IF (MONTH ¢= BEGMONTH) OR (MONTH _= ENDMONTH)
                       THEN DO;
                         set ANNSTAT to 0;
                         OUTPUT;
         END;   RUN;


   DATA &ELEMENT.STAT(KEEP= STATION MONTH STN_FORM NAME COUNTY
                            &ELEMENT.DAY LAGDAY);
      RETAIN  CARRY;
      ARRAY  &ARRRAY.[31]  &ARRRAY.1-&ARRRAY.31;
```

```
      SET &ELEMENT.MON;
         BY STATION;
      IF FIRST.STATION  THEN
         CARRY = .;
         set F_MONTH to the value of MONTH;
         set F_YEAR to the value of YEAR;
      IF LAST.STATION  THEN
         set L_MONTH to the value of MONTH;
         set L_YEAR to the value of YEAR;

      NUMDAYS = 31;
      IF MONTH = 4 OR MONTH = 6 OR MONTH = 9 OR MONTH = 11  THEN
         NUMDAYS = 30;
      ELSE IF MONTH = 2  THEN
         IF MOD(YEAR,4) NE 0  THEN  NUMDAYS = 28;
                              ELSE  NUMDAYS = 29;

      DO DAY = 1 TO NUMDAYS;
         &ELEMENT.DAY = &ARRRAY.[DAY];
         LAGDAY = CARRY;
         OUTPUT;
         CARRY = &ARRRAY.[DAY];
      END;
RUN;
```

4.94 Description of Module COMPUTE DAILY STATISTICS (C_DSTAT)

4.94.1 Processing Narrative

COMPUTE DAILY STATISTICS calculates basic statistical information for a subset of data, by each of the requested stations. The analyses are performed by month using daily values; however, annual analyses can also be obtained. PROC MEANS is used to compute means, standard deviations, number of observations, number of missing values, maximums, minimums, skewness and kurtosis. PROC CORR is used to calculate serial correlations between daily values.

4.94.2 Interface Description

A) Parameters: ELEMENT : 4-character element name
E_TITLE : full name of the element for heading

B) Global data:

Macro variables:
ONLYANN : identifies request for ONLY ANNUAL option
ANNSTAT : identifies request for annual statistics

SAS data sets:

&ELEMENT.STAT (input) - with variables:
STATION MONTH STN_FORM NAME COUNTY &ELEMENT.DAY
LAGDAY

&ELEMENT.CSUB (input and output) - with variables:
STATION MONTH &ELEMENT.DAY LAGDAY _TYPE_ _NAME_

&ELEMENT.CNEW (input and output) - with variables:
STATION MONTH LAGDAY _NAME_

&ELEMENT.MSUB (input and output) - with variables:
STATION MONTH STN_FORM NAME COUNTY &ELEMENT.DAY MEAN
STD N NMISS MAXIMUM MINIMUM SKEWNESS KURTOSIS

&ELEMENT.BOTH (input and output) - with variables:
STATION MONTH STN_FORM NAME COUNTY &ELEMENT.DAY MEAN
STD N NMISS MAXIMUM MINIMUM SKEWNESS KURTOSIS
LAGDAY _NAME_

&ELEMENT.CANN (input and output) - with variables:
STATION &ELEMENT.DAY LAGDAY _TYPE_ _NAME_

&ELEMENT.CAN2 (input and output) - with variables:
STATION MONTH LAGDAY _NAME_

&ELEMENT.MANN (input and output) - with variables:
STATION STN_FORM NAME COUNTY &ELEMENT.DAY MEAN
STD N NMISS MAXIMUM MINIMUM SKEWNESS KURTOSIS

&ELEMENT.ANN (input and output) - with variables:
STATION MONTH STN_FORM NAME COUNTY &ELEMENT.DAY MEAN
STD N NMISS MAXIMUM MINIMUM SKEWNESS KURTOSIS
LAGDAY _NAME_

&ELEMENT.ALL (input and output) - with variables:
STATION MONTH STN_FORM NAME COUNTY &ELEMENT.DAY MEAN
STD N NMISS MAXIMUM MINIMUM SKEWNESS KURTOSIS
LAGDAY _NAME_

PRINT (output) - standard SAS print file

C) No macros are called by this module.

D) Permanent files: NHIMS.FORMATS

E) The module generates several complete DATA and PROC steps.

4.94.3 Internal Data Description

Macro variables:

```
   F_MONTH : first month for which data was requested
   F_YEAR  : first year for which data was requested
   L_MONTH : last month for which data was requested
   L_YEAR  : last year for which data was requested
```

SAS variables:

```
   ROW    (num): identifies row in which to print values
   COLUMN (num): identifies column in which to print values
   FIRST.STATION (boolean): identifies first occurrence of current
                            value of the STATION variable
   LAST.STATION  (boolean): identifies last occurrence of current
                            value of the STATION variable
```

4.94.4 Design Language Description

```
 %IF not a request for ONLY ANNUAL statistics %THEN
 %DO;

   PROC SORT DATA=&ELEMENT.STAT;
      BY STATION MONTH;
   RUN;


   PROC CORR DATA=&ELEMENT.STAT  OUT=&ELEMENT.CSUB  NOPRINT
                 NOSIMPLE;
      BY STATION MONTH;
      VAR &ELEMENT.DAY  LAGDAY;
   RUN;

   DATA  &ELEMENT.CNEW(DROP=&ELEMENT.DAY  _TYPE_);
   /* only keep part of PROC CORR output.          */
   /* Then can easily merge with PROC MEANS output */

      SET &ELEMENT.CSUB;
      IF  _NAME_  =  "&ELEMENT.DAY";
   RUN;

   PROC MEANS DATA=&ELEMENT.STAT  NOPRINT  N  NMISS MEAN STD
                  MAXIMUM MINIMUM  SKEWNESS KURTOSIS;
      BY STATION MONTH;
      VAR &ELEMENT.DAY;
      ID STN_FORM  NAME  COUNTY;
      OUTPUT OUT=&ELEMENT.MSUB  MEAN=MEAN  STD=STD N=N
             NMISS=NMISS  MAX=MAXIMUM  MIN=MINIMUM
             SKEWNESS=SKEWNESS  KURTOSIS=KURTOSIS;
   RUN;
```

```
   DATA &ELEMENT.BOTH; /* combine output from PROCS MEANS, CORR */
      MERGE &ELEMENT.MSUB  &ELEMENT.CNEW;
        BY STATION MONTH;
   RUN;

%END;  /* not a request for ONLY ANNUAL statistics */


%IF request for annual statistics  %THEN

   PROC CORR DATA=&ELEMENT.STAT  OUT=&ELEMENT.CANN  NOPRINT
                NOSIMPLE;
      BY STATION ;
      VAR &ELEMENT.DAY  LAGDAY;
   RUN;


   DATA  &ELEMENT.CAN2(DROP=&ELEMENT.DAY  _TYPE_);
   /* only keep part of PROC CORR output.          */
   /* Then can easily merge with PROC MEANS output */

      SET &ELEMENT.CANN;
      MONTH = 13;  /* assign month value the annual value */
      IF  _NAME_  =  "&ELEMENT.DAY";
   RUN;


   PROC MEANS DATA=&ELEMENT.STAT  NOPRINT  N  NMISS MEAN STD
                  MAXIMUM MINIMUM  SKEWNESS KURTOSIS;
      BY STATION ;
      VAR &ELEMENT.DAY;
      ID STN_FORM  NAME  COUNTY;
      OUTPUT OUT=&ELEMENT.MANN  MEAN=MEAN  STD=STD N=N
             NMISS=NMISS  MAX=MAXIMUM  MIN=MINIMUM
             SKEWNESS=SKEWNESS  KURTOSIS=KURTOSIS;
   RUN;


   DATA &ELEMENT.ANN; /* combine output from PROCS MEANS, CORR */
      MERGE &ELEMENT.MANN  &ELEMENT.CAN2;
        BY STATION ;
   RUN;


   %IF not a request for ONLY ANNUAL statistics  %THEN
   %* combine the annual and daily statistics;

      DATA &ELEMENT.ALL; /* combine annual and daily statistics */
         MERGE &ELEMENT.ANN  &ELEMENT.BOTH;
           BY STATION MONTH;
```

```
      RUN;
   %END;
END;   %* request for annual statistics */


   DATA _NULL_; /* step to print daily statistics */

     RETAIN ROW  12;
     FILE PRINT N=PS;

     %IF a request for ONLY ANNUAL statistics  %THEN
         SET &ELEMENT.ANN;
            BY STATION;
     %ELSE %IF a request for annual statistics %THEN
         SET &ELEMENT.ALL;
            BY STATION;
     %ELSE
         SET &ELEMENT.BOTH;
            BY STATION;
     %END;

         IF FIRST.STATION  THEN
            print STN_FORM, NAME, and COUNTY in heading;
            print 'STATISTICAL ANALYSIS OF DAILY &E_TITLE";
            print "&F_MONTH / &F_YEAR  TO  &L_MONTH / &L_YEAR";
            print column headings:  'MONTH  NUMBER OF OBS
                  NUMBER OF MISSING VALUES  MEAN  STANDARD DEV
                  MAXIMUM  MINIMUM  SKEWNESS  KURTOSIS
                  SERIAL CORRELATION;

         print one observation under the above headings,
         using MNTH. format for the MONTH values;

         IF LAST.STATION  THEN
            PUT_PAGE_;
            initialize ROW;

     RUN;
```

4.95 Description of Module MONTHLY STATISTICS (M_STATS)

4.95.1 Processing Narrative

MONTHLY STATISTICS first determines which months are requested in the ONLY option. Then all pertaining elements are checked, subsets are made of the required data, and the statistics are computed. Only one execution of the monthly statistics process is allowed per access request.

4.95.2 Interface Description

A) There is no parameter list.

B) Global data:
Macro variables:
EL_PRCP : identifies request for precipitation data
EL_TEMP : identifies request for temperature data
EL_STRM : identifies request for streamflow data
EL_SNOW : identifies request for snowfall data
EL_EVAP : identifies request for evaporation data

No SAS data sets are created or used.

C) The macros called by this module are:

GET MONTHS FROM ONLY OPTION (4.92 - GET_MON)
MAKE SUBSET OF MONTHLY VALUES (4.96 - MON_SUB)
COMPUTE MONTHLY STATISTICS (4.97 - C_MSTAT)

D) No permanent files are used by this module.

E) The module generates macro statements only.

4.95.3 Internal Data Description

Macro variables:
F_MONTH : first month for which data was requested
F_YEAR : first year for which data was requested
L_MONTH : last month for which data was requested
L_YEAR : last year for which data was requested

No local SAS variables are used in this module.

4.95.4 Design Language Description

```
%LOCAL F_MONTH;
%LOCAL F_YEAR;
%LOCAL L_MONTH;
%LOCAL L_YEAR;

%GET MONTHS FROM ONLY OPTION(MSTATS,MONTHLY STATISTICS,
                             MONTHLY STATS)

%IF request for precipitation data %THEN
    %MAKE SUBSET OF MONTHLY VALUES(PRCP,PRCP_TOT)
    %COMPUTE MONTHLY STATISTICS(PRCP,PRCP_TOT,PRECIPITATION)
```

```
%IF request for temperature data %THEN
    %MAKE SUBSET OF MONTHLY VALUES(TEMP,AVEMAX)
    %COMPUTE MONTHLY STATISTICS(TEMP,AVEMAX,MAXIMUM TEMPERATURES)

    %MAKE SUBSET OF MONTHLY VALUES(TEMP,AVEMIN)
    %COMPUTE MONTHLY STATISTICS(TEMP,AVEMIN,MINIMUM TEMPERATURES)

%IF request for streamflow data %THEN
    %MAKE SUBSET OF MONTHLY VALUES(STRM,STRM_TOT)
    %COMPUTE MONTHLY STATISTICS(STRM,STRM_TOT,STREAMFLOW)

%IF request for snowfall data %THEN
    %MAKE SUBSET OF MONTHLY VALUES(SNOW,SNOW_TOT)
    %COMPUTE MONTHLY STATISTICS(SNOW,SNOW_TOT,SNOWFALL)

%IF request for evaporation data %THEN
    %MAKE SUBSET OF MONTHLY VALUES(EVAP,EVAP_TOT)
    %COMPUTE MONTHLY STATISTICS(EVAP,EVAP_TOT,EVAPORATION)
```

4.96 Description of Module MAKE SUBSET OF MONTHLY VALUES (MON_SUB)

4.96.1 Processing Narrative

MAKE SUBSET OF MONTHLY VALUES first creates a subset of the requested element data according to the months specified in the ONLY option. Then, the data is reshaped for the monthly statistics process; a lag variable is created to retain the previous month's value for correlation.

4.96.2 Interface Description

A) Parameters: ELEMENT - 4-character element name
MON_VAL - variable name of the monthly value in the input file

B) Global data:
Macro variables:
ONLYANN : identifies request for ONLY ANNUAL option
ANNSTAT : identifies request for annual statistics

SAS data sets:
&ELEMENT (input) - with variables:
STATION YEAR MONTH STN_FORM NAME COUNTY
&MON_VAL

MSTATS (input) - with variables:
BEGMONTH ENDMONTH RANGE

&ELEMENT.MON (input and output) - with variables:
same as &ELEMENT data set

&ELEMENT.STAT (output) - with variables:
STATION MONTH STN_FORM NAME COUNTY &MON_VAL LAGMON

C) No macros are called by this module.

D) No permanent files are used by this module.

E) The module generates two complete DATA steps.

4.96.3 Internal Data Description

Macro variables:
F_MONTH : first month for which data was requested
F_YEAR : first year for which data was requested
L_MONTH : last month for which data was requested
L_YEAR : last year for which data was requested

SAS variables:
I (num): control variable for DO loop
NUMDAYS(num): number of days in a particular month
FIRST.STATION (boolean): first occurrence of current value of the STATION variable
LAST.STATION (boolean): last occurrence of current value of the STATION variable

4.96.4 Design Language Description

```
DATA &ELEMENT.MON(DROP= BEGMONTH ENDMONTH RANGE);

      SET &ELEMENT;
      DO I = 1 TO 1;
         SET MSTATS POINT=I;
         /* check range of dates from only option - if annual */
         /* is requested, ENDMONTH = 13, keep entire record */

         IF RANGE NE 'ANNUAL'  THEN
            set ONLYANN to 0; /* request more than annual data */
         ELSE
            set ONLYANN to 1;/* request is ONLY ANNUAL */
         IF ENDMONTH = 13  THEN
            set ANNSTAT to 1;
            OUTPUT;
         ELSE IF BEGMONTH _= MONTH _= ENDMONTH  THEN
                 DO;
                   set ANNSTAT to 0;
                   OUTPUT;
         ELSE IF BEGMONTH ¢ ENDMONTH THEN
                 IF (MONTH ¢= BEGMONTH) OR (MONTH _= ENDMONTH)
```

```
                        THEN DO;
                          set ANNSTAT to 0;
                          OUTPUT;
          END;
      RUN;


  DATA &ELEMENT.STAT(KEEP= STATION MONTH STN_FORM NAME
                           COUNTY  &MON_VAL LAGMON);
      SET &ELEMENT.MON;
         BY STATION;
      LAGMON = LAG(&MON_VAL);

      IF FIRST.STATION  THEN
         LAGMON = . ;
         set F_MONTH to the value of MONTH;
         set F_YEAR to the value of YEAR;
      IF LAST.STATION  THEN
         set L_MONTH to the value of MONTH;
         set L_YEAR to the value of YEAR;
RUN;
```

4.97 Description of Module COMPUTE MONTHLY STATISTICS (C_MSTAT)

4.97.1 Processing Narrative

COMPUTE MONTHLY STATISTICS calculates basic statistical information for a subset of data, by each of the requested stations. The analyses are performed by month using monthly totals or averages; however, annual analyses can also be obtained. PROC MEANS is used to compute means, standard deviations, number of observations, number of missing values, maximums, minimums, skewness and kurtosis. PROC CORR calculates serial correlations between monthly values.

4.97.2 Interface Description

A) Parameters: ELEMENT : 4-character element name
MON_VAL : variable name of monthly total or average
E_TITLE : full name of the element for heading

B) Global data:

Macro variables:
ONLYANN : identifies request for ONLY ANNUAL option
ANNSTAT : identifies request for annual statistics

SAS data sets:
&ELEMENT.STAT (input) - with variables:
STATION MONTH STN_FORM NAME COUNTY &MON_VAL LAGMON

&ELEMENT.CSUB (input and output) - with variables:
STATION MONTH &MON_VAL LAGMON _TYPE_ _NAME_

&ELEMENT.CNEW (input and output) - with variables:
STATION MONTH LAGMON _NAME_

&ELEMENT.MSUB (input and output) - with variables:
STATION MONTH STN_FORM NAME COUNTY &MON_VAL MEAN
STD N NMISS MAXIMUM MINIMUM SKEWNESS KURTOSIS

&ELEMENT.BOTH (input and output) - with variables:
STATION MONTH STN_FORM NAME COUNTY &MON_VAL MEAN
STD N NMISS MAXIMUM MINIMUM SKEWNESS KURTOSIS
LAGMON _NAME_

&ELEMENT.CANN (input and output) - with variables:
STATION &MON_VAL LAGMON _TYPE_ _NAME_

&ELEMENT.CAN2 (input and output) - with variables:
STATION MONTH LAGMON _NAME_

&ELEMENT.MANN (input and output) - with variables:
STATION STN_FORM NAME COUNTY &MON_VAL MEAN
STD N NMISS MAXIMUM MINIMUM SKEWNESS KURTOSIS

&ELEMENT.ANN (input and output) - with variables:
STATION MONTH STN_FORM NAME COUNTY &MON_VAL MEAN
STD N NMISS MAXIMUM MINIMUM SKEWNESS KURTOSIS
LAGMON _NAME_

&ELEMENT.ALL (input and output) - with variables:
STATION MONTH STN_FORM NAME COUNTY &MON_VAL MEAN
STD N NMISS MAXIMUM MINIMUM SKEWNESS KURTOSIS
LAGMON _NAME_

PRINT (output) - standard SAS print file

C) No macros are called by this module.

D) Permanent files: NHIMS.FORMATS

E) The module generates several complete DATA and PROC steps.

4.97.3 Internal Data Description

Macro variables:
F_MONTH : first month for which data was requested
F_YEAR : first year for which data was requested
L_MONTH : last month for which data was requested
L_YEAR : last year for which data was requested

```
SAS variables:
   ROW    (num): identifies row in which to print values
   COLUMN (num): identifies column in which to print values
   FIRST.STATION (boolean): identifies first occurrence of current
                            value of the STATION variable
   LAST.STATION  (boolean): identifies last occurrence of current
                            value of the STATION variable
```

4.97.4 Design Language Description

```
 %IF not a request for ONLY ANNUAL statistics %THEN
 %DO;

   PROC SORT DATA=&ELEMENT.STAT;
      BY STATION MONTH;   RUN;

   PROC CORR DATA=&ELEMENT.STAT  OUT=&ELEMENT.CSUB  NOPRINT
                  NOSIMPLE; /* calculate monthly correlations */
      BY STATION MONTH;
      VAR &MON_VAL  LAGMON;   RUN;

   DATA  &ELEMENT.CNEW(DROP=&MON_VAL  _TYPE_);
   /* only keep part of PROC CORR output.          */
   /* Then can easily merge with PROC MEANS output */

      SET &ELEMENT.CSUB;
      IF  _NAME_  =  "&MON_VAL";   RUN;

   /* now calculate monthly statistics with PROC MEANS */

   PROC MEANS DATA=&ELEMENT.STAT  NOPRINT  N  NMISS MEAN STD
                  MAXIMUM MINIMUM  SKEWNESS KURTOSIS;
      BY STATION MONTH;
      VAR &MON_VAL;
      ID STN_FORM  NAME  COUNTY;
      OUTPUT OUT=&ELEMENT.MSUB  MEAN=MEAN  STD=STD N=N
             NMISS=NMISS  MAX=MAXIMUM  MIN=MINIMUM
             SKEWNESS=SKEWNESS  KURTOSIS=KURTOSIS;
   RUN;

   DATA &ELEMENT.BOTH; /* combine output from PROCS MEANS, CORR */
      MERGE &ELEMENT.MSUB  &ELEMENT.CNEW;
        BY STATION MONTH;
   RUN;

%END;  /* not a request for ONLY ANNUAL statistics */
```

```
%IF request for annual statistics  %THEN

   PROC CORR DATA=&ELEMENT.STAT  OUT=&ELEMENT.CANN  NOPRINT
                 NOSIMPLE; /* correlate values for entire year */
      BY STATION ;
      VAR &ELEMENT.DAY  LAGDAY;
   RUN;


   DATA  &ELEMENT.CAN2(DROP=&ELEMENT.DAY  _TYPE_);
   /* only keep part of PROC CORR output.          */
   /* Then can easily merge with PROC MEANS output */

      SET &ELEMENT.CANN;
      MONTH = 13;  /* assign month  the annual value - 13 */
      IF  _NAME_  =  "&ELEMENT.DAY";
   RUN;

  /* now calculate statistics for annual period */

   PROC MEANS DATA=&ELEMENT.STAT  NOPRINT  N  NMISS MEAN STD
                  MAXIMUM MINIMUM  SKEWNESS KURTOSIS;
      BY STATION ;
      VAR &ELEMENT.DAY;
      ID STN_FORM  NAME  COUNTY;
      OUTPUT OUT=&ELEMENT.MANN  MEAN=MEAN  STD=STD N=N
             NMISS=NMISS  MAX=MAXIMUM  MIN=MINIMUM
             SKEWNESS=SKEWNESS  KURTOSIS=KURTOSIS;
   RUN;

   DATA &ELEMENT.ANN; /* combine output from PROCS MEANS, CORR */
      MERGE &ELEMENT.MANN  &ELEMENT.CAN2;
        BY STATION ;   RUN;

   %IF not a request for ONLY ANNUAL statistics  %THEN
   %* combine the annual and monthly statistics;

      DATA &ELEMENT.ALL; /* combine annual, monthly statistics */
         MERGE &ELEMENT.ANN  &ELEMENT.BOTH;
           BY STATION MONTH;
      RUN;
   %END;
END;   %* request for annual statistics */


   DATA _NULL_; /* step to print monthly statistics */

     FILE PRINT N=PS;
     RETAIN ROW  12;
```

```
%IF a request for ONLY ANNUAL statistics  %THEN
    SET &ELEMENT.ANN;
       BY STATION;
%ELSE %IF a request for annual statistics %THEN
    SET &ELEMENT.ALL;
       BY STATION;
%ELSE
    SET &ELEMENT.BOTH;
       BY STATION;
%END;

    IF FIRST.STATION  THEN
       print STN_FORM, NAME, and COUNTY in heading;
       print 'STATISTICAL ANALYSIS OF MONTHLY &E_TITLE";
       print "&F_MONTH / &F_YEAR  TO  &L_MONTH / &L_YEAR";
       print column headings:  'MONTH  NUMBER OF OBS
             NUMBER OF MISSING VALUES  MEAN  STANDARD DEV
             MAXIMUM  MINIMUM  SKEWNESS  KURTOSIS
             SERIAL CORRELATION;

    print one observation under the above headings,
    using MNTH. format for the MONTH values, and
    putting serial correlation one row above other data;

    IF LAST.STATION  THEN
       PUT_PAGE_;
       initialize ROW;

RUN;
```

4.98 Description of Module CORRELATION (CORR)

4.98.1 Processing Narrative

CORRELATION first determines which months are requested in the ONLY option. Then all pertaining elements are checked, and the proper macros are called to make subsets of the required data, correlate the daily values, and print the results. Only one execution of the correlation process is allowed per access request.

4.98.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:
  EL_PRCP : identifies request for precipitation data
  EL_TEMP : identifies request for temperature data
  EL_STRM : identifies request for streamflow data
  EL_SNOW : identifies request for snowfall data
  EL_EVAP : identifies request for evaporation data

No SAS data sets are created or used.

C) The macros called by this module are:

GET MONTHS FROM ONLY OPTION (4.92 - GET_MON)
MAKE SUBSET OF MONTHS (4.99 - MAKESUB)
CHECK NUMBER OF OBSERVATIONS (4.20 - NUM_OBS)
COMPUTE INTERSTATION CORRELATION (4.100 - C_CORR)

D) No permanent files are used by this module.

E) The module generates macro statements only.

4.98.3 Internal Data Description

No local data are used in this module.

4.98.4 Design Language Description

```
%GET MONTHS FROM ONLY OPTION(CORR,CORRELATION,CORELATION)

%IF request for precipitation data %THEN
    %MAKE SUBSET OF MONTHS(PRCP)
    %CHECK NUMBER OF OBSERVATIONS(STATIONS)
    %COMPUTE INTERSTATION CORRELATION(PRCP,PRECIP,PRECIPITATION)

%IF request for temperature data %THEN
    %MAKE SUBSET OF MONTHS(TEMP)
    %CHECK NUMBER OF OBSERVATIONS(STATIONS)
    %COMPUTE INTERSTATION CORRELATION(TEMP,MAXTMP,
                                      MAXIMUM TEMPERATURES)
    %COMPUTE INTERSTATION CORRELATION(TEMP,MINTMP,
                                      MINIMUM TEMPERATURES)

%IF request for streamflow data %THEN
    %MAKE SUBSET OF MONTHS(STRM)
    %CHECK NUMBER OF OBSERVATIONS(STATIONS)
    %COMPUTE INTERSTATION CORRELATION(STRM,FLOW,STREAMFLOW)

%IF request for snowfall data %THEN
    %MAKE SUBSET OF MONTHS(SNOW)
    %CHECK NUMBER OF OBSERVATIONS(STATIONS)
    %COMPUTE INTERSTATION CORRELATION(SNOW,SNOW,SNOWFALL)
```

```
%IF request for evaporation data %THEN
    %MAKE SUBSET OF MONTHS(EVAP)
    %CHECK NUMBER OF OBSERVATIONS(STATIONS)
    %COMPUTE INTERSTATION CORRELATION(EVAP,EVAP,EVAPORATION)
```

4.99 Description of Module MAKE SUBSET OF MONTHS (MAKESUB)

4.99.1 Processing Narrative

MAKE SUBSET OF MONTHS checks the months specified in the ONLY option, and makes a subset of those months. Global macro variables are set according to the annual data requested.

4.99.2 Interface Description

A) Parameters: ELEMENT : 4-character element name

B) Global data:

Macro variables:
- ONLYANN : identifies request for ONLY ANNUAL option
- ANNCORR : identifies request for annual correlations

SAS data sets:

&ELEMENT (input) - with variables:
STATION YEAR MONTH STN_FORM NAME COUNTY
an array of 31 daily values

CORR (input) - with variables:
BEGMONTH ENDMONTH RANGE

&ELEMENT.CORR (output) - with variables:
same as &ELEMENT

C) No macros are called by this module.

D) No permanent files are used by this module.

E) The module generates one complete DATA step.

4.99.3 Internal Data Description

No local macro variables are used in this module.

SAS variables:
I (num): control variable for DO loop

4.99.4 Design Language Description

```
DATA &ELEMENT.CORR(DROP= BEGMONTH ENDMONTH RANGE);

      SET &ELEMENT;
      DO I = 1 TO 1;
         SET CORR POINT=I;
         /* check range of dates from only option - if annual */
         /* is requested, ENDMONTH = 13, keep entire record */

         IF RANGE NE 'ANNUAL'  THEN
            set ONLYANN to 0; /* request more than annual data */
         ELSE
            set ONLYANN to 1;/* request is ONLY ANNUAL */
         IF ENDMONTH = 13  THEN
            set ANNCORR to 1;
            OUTPUT;
         ELSE IF BEGMONTH _= MONTH _= ENDMONTH  THEN
                 DO;
                   set ANNCORR to 0;
                   OUTPUT;
         ELSE IF BEGMONTH ¢ ENDMONTH THEN
                 IF (MONTH ¢= BEGMONTH) OR (MONTH _= ENDMONTH)
                    THEN DO;
                      set ANNCORR to 0;
                      OUTPUT;
      END;
   RUN;
```

4.100 Description of Module COMPUTE INTERSTATION CORRELATION (C_CORR)

4.100.1 Processing Narrative

COMPUTE INTERSTATION CORRELATION reshapes the input data so that one observation contains a daily value for each station to be correlated. PROC CORR computes the correlations, and writes the output to an external file so that the number of pairs can be retrieved in addition to the correlations. The listing step reads the PROC CORR output as character data from that external file. Correlations are performed by month, but annual correlations can be obtained in addition to or instead of the monthly ones.

4.100.2 Interface Description

A) Parameters: ELEMENT : 4-character element name
ARRRAY : array name of 31 daily values
E_TITLE : full name of element for heading

B) Global data:

Macro variables:

```
NOBS     : the number of stations in the STATIONS data set
ONLYANN  : identifies request for ONLY ANNUAL option
ANNCORR  : identifies request for annual correlations
```

SAS data sets:

&ELEMENT.CORR (input) - with variables:
STATION YEAR MONTH STN_FORM NAME COUNTY
an array of 31 daily values

SUBSET1-SUBSET&NOBS (input and output) - with variables:
YEAR MONTH DAY STN_&S (S = same subscript as data set name)

&ELEMENT.COMB (input and output) - with variables:
YEAR MONTH DAY STN&1-STN_&NOBS

FT22F001 (input and output) - external file for PROC CORR output

&ELEMENT.PTIX (input) - with variables:
STATION NAME COUNTY FOBS_PTR LOBS_PTR

PRINT (output) - standard SAS print file

C) No macros are called by this module.

D) Permanent files: NHIMS.FORMATS

E) The module generates several complete DATA and PROC steps.

4.100.3 Internal Data Description

Macro variables:

```
      S: control variable for %DO loop
MERGES: names of the subset data sets for MERGE statement
   VARS: names of the variables for the VAR statement
 F_YEAR: first year of requested period of record
F_MONTH: first month of requested period of record
 L_YEAR: last year of requested period of record
L_MONTH: last month of requested period of record
```

SAS variables:

```
COUNT    (num): counter for the number of requested stations
EOF  (boolean): identifies last record read from input file
MOVE     (num): constant used to line up output values
ST       (num): control variable for DO loop
```

```
CHECK     (num): identifies position of month number in PROC
                 CORR output
ONELINE (char): one record from the external file
POS1      (num): identifies position of string 'NUMBER'
POS2      (num): identifies position of string 'N ='
PAIRS    (char): value of the number of pairs if in heading
FIRST.STATION (boolean): first occurrence of current value
                         of STATION variable
LAST.STATION  (boolean): last occurrence of current value
                         of STATION variable
```

4.100.4 Design Language Description

```
%LOCAL S;
%LOCAL MERGES;
%LOCAL VARS;
%LOCAL F_YEAR;
%LOCAL F_MONTH;
%LOCAL L_YEAR;
%LOCAL L_MONTH;

%DO S = 1 %TO &NOBS;
%* create a subset data set for each station requested ;
%* NOBS equals the number of stations requested;

    DATA SUBSET&S(KEEP=YEAR MONTH DAY STN_&S);

       RETAIN COUNT 0 ;
       ARRAY &ARRRAY.[31]  &ARRRAY.1-&ARRRAY.31;
       SET &ELEMENT.CORR END=EOF;
         BY STATION;

       IF FIRST.STATION THEN
          increment COUNT by 1;
          set F_MONTH to the value of MONTH;
          set F_YEAR to the value of YEAR;

       IF LAST.STATION  THEN
          set L_MONTH to the value of MONTH;
          set L_YEAR to the value of YEAR;
       IF COUNT = &S  THEN
          DO DAY = 1 TO 31;
             IF &ARRRAY[DAY] is not equal to missing THEN
                STN_&S = &ARRRAY[DAY];
                OUTPUT;
          END;
    RUN;
%END;

%* now merge the data sets by date ;
```

```
%LET MERGES = SUBSET1;
%LET VARS = STN_1;

%DO S = 2 %TO &NOBS;
    %LET MERGES = &MERGES SUBSET&S;
    %LET VARS = &VARS STN_&S;
%END;


    DATA &ELEMENT.COMB;

       MERGE &MERGES;
          BY YEAR MONTH DAY;
    RUN;

%IF  not a request for ONLY ANNUAL correlations %THEN
%DO;
%*  calculate correlations by month ;

    PROC SORT DATA=&ELEMENT.COMB;
       BY MONTH;
    RUN;


    PROC PRINTTO UNIT=22  NEW; /* direct  proc output to a */
                               /* temporary disk file      */
    RUN;


    PROC CORR DATA=&ELEMENT.COMB NOSIMPLE  ;
       TITLE1 'NEW MONTH';
       BY MONTH;
       VAR &VARS;
    RUN;


    PROC PRINTTO;   /* direct output back to printer  */
    RUN;

    DATA _NULL_; /* step to print output from PROC CORR */

       RETAIN  MOVE  -1;
       INFILE FT22F001 END=EOF;
       print "CORRELATION ANALYSIS OF DAILY &ELEMENT ";
       print "&F_MONTH / &F_YEAR  TO  &L_MONTH / &L_YEAR";

       DO ST = 1 TO &NOBS; /* print index at top of page */
          SET &ELEMENT.PTIX ;
          print 'STATION' +MOVE ST, STATION, NAME, and COUNTY ;
       END;
```

```
DO UNTIL(CHECK NE 0);

   INPUT ONELINE $CHAR132.;
   CHECK = INDEX(ONELINE,'MONTH=');
   IF CHECK NE 0 THEN
      CHECK = CHECK + 6;
      MONTH = SUBSTR(ONELINE,CHECK,2);
   IF EOF THEN
      issue error message that correlation information
      in the external file was not found, and STOP;
END;

DO UNTIL(POS1 NE 0 OR POS2 NE 0);

   INPUT ONELINE $CHAR132.;
   POS1 = INDEX(ONELINE,'NUMBER');
   IF POS1 NE 0  THEN
      PAIRS = '    '; /* number of pairs explicit */
   POS2 = INDEX(ONELINE,'N =');
   IF POS2 NE 0  THEN
      POS2 = POS2 + 3; /* number of pairs in heading */
      PAIRS = SUBSTR(ONELINE,POS2,5);

   IF EOF THEN
      issue error message that correlation information
      in the external file was not found, and STOP;
      POS1 = 1;
END;

PUT;
PUT 'MONTH = ' MONTH MNTH. @;
IF PAIRS  is not equal to a blank THEN
   PUT 'NUMBER OF PAIRS = '  PAIRS;
ELSE
   PUT;
PUT;

INPUT ONELINE $CHAR132. ;

DO UNTIL(NEW_MON NE 0);

   PUT ONELINE $CHAR132.;
   IF EOF  THEN STOP;
   INPUT ONELINE $CHAR132. ;
   NEW_MON = INDEX(ONELINE,'NEW MONTH');
END;
PUT _PAGE_;
RETURN;
```

```
 RUN;

%END;  %* calculate correlations by month ;


 %IF request for annual correlations %THEN

   PROC PRINTTO UNIT=22  NEW; /* direct  proc output to a */
                              /* temporary disk file      */
   RUN;


    PROC CORR DATA=&ELEMENT.COMB  NOSIMPLE ;
       VAR &VARS;
    RUN;


   PROC PRINTTO ; /* redirect output to printer */
   RUN;


   DATA _NULL_; /* step to print annual correlations */

      FILE PRINT;
      RETAIN  MOVE  -1;
      INFILE FT22F001 END=EOF;
      print "CORRELATION ANALYSIS OF DAILY &ELEMENT ";
      print "&F_MONTH / &F_YEAR  TO  &L_MONTH / &L_YEAR";

      DO ST = 1 TO &NOBS; /* print index at top of page */
         SET &ELEMENT.PTIX ;
         print 'STATION' +MOVE ST, STATION, NAME, and COUNTY ;
      END;

      DO UNTIL(POS1 NE 0 OR POS2 NE 0);

         INPUT ONELINE $CHAR132.;
         POS1 = INDEX(ONELINE,'NUMBER');
         IF POS1 NE 0  THEN
            PAIRS = '    '; /* number of pairs explicit */

         POS2 = INDEX(ONELINE,'N =');
         IF POS2 NE 0  THEN
            POS2 = POS2 + 3; /* number of pairs in heading */
            PAIRS = SUBSTR(ONELINE,POS2,5);

         IF EOF THEN
            issue error message that correlation information
            in the external file was not found, and STOP;
            POS1 = 1;
      END;
```

```
        PUT;
        IF PAIRS  is not equal to a blank THEN
           PUT 'NUMBER OF PAIRS = '  PAIRS;
        ELSE
           PUT;
        PUT;

        INPUT ONELINE $CHAR132. ;

        DO UNTIL(EOF);

           PUT ONELINE $CHAR132.;
           IF EOF  THEN STOP;
           INPUT ONELINE $CHAR132. ;
        END;

        IF EOF THEN STOP;
        PUT _PAGE_;
        RETURN;

   RUN;

%END;
```

4.101 Description of Module HIGHEST (HIGHEST)

4.101.1 Processing Narrative

HIGHEST calls macros to determine which months are requested in the ONLY option, and to compute the highest monthly value(s) for each of the pertinent elements. Different parameter values are sent depending on the value of the metric option. The highest process may be executed multiple times in one access request.

4.101.2 Interface Description

A) There is no parameter list.

B) Global data:
   Macro variables:
      EL_PRCP : identifies request for precpitation data
      EL_TEMP : identifies request for temperature data
      EL_STRM : identifies request for streamflow data
      EL_SNOW : identifies request for snowfall data
      EL_EVAP : identifies request for evaporation data
      METRIC : identifies request for metric units

   No SAS data sets are created or used.

C) The macros called by this module are:

   GET MONTHS FROM ONLY OPTION (4.92 - GET_MON)
   CHECK NUMBER OF OBSERVATIONS (4.20 - NUM_OBS)
   COMPUTE AND PRINT HIGHS/LOWS FOR ONE VALUE (4.103 - HI_LO_1)
   COMPUTE AND PRINT HIGHS/LOWS FOR TWO VALUES (4.104 - HI_LO_2)

D) No permanent files are used by this module.

E) The module generates macro statements only.

4.101.3 Internal Data Description

No local data are used in this module.

4.101.4 Design Language Description

```
%GET MONTHS FROM ONLY OPTION (HIGH,HIGHEST,MAXIMUM)
%CHECK NUMBER OF OBSERVATIONS (HIGH)

%IF request for precipitation data %THEN
    %IF not request for metric units %THEN
        %COMPUTE AND PRINT HIGHS/LOWS FOR ONE VALUE
            (PRCP,HIGH,MAX,PRECIP,PRECIPITATION,INCHES,8.2)
```

```
    %ELSE %IF a request for metric units %THEN
        %COMPUTE AND PRINT HIGHS/LOWS FOR ONE VALUE
            (PRCP,HIGH,MAX,PRECIP,PRECIPITATION,MILLIMETERS,8.1)

%IF request for temperature data %THEN
    %IF not request for metric units %THEN
        %COMPUTE AND PRINT HIGHS/LOWS FOR TWO VALUES
            (TEMP,HIGH,MAX,MAXTMP,MINTMP,MAXIMUM TEMPERATURES,
             MINIMUM TEMPERATURES,DEGREES F,DEGREES F,8.,8.)
    %ELSE %IF  a request for metric units %THEN
        %COMPUTE AND PRINT HIGHS/LOWS FOR TWO VALUES
            (TEMP,HIGH,MAX,MAXTMP,MINTMP,MAXIMUM TEMPERATURES,
             MINIMUM TEMPERATURES,DEGREES C,DEGREES C,8.1,8.1)

%IF request for streamflow data %THEN
    %IF not request for metric units %THEN
        %COMPUTE AND PRINT HIGHS/LOWS FOR ONE VALUE
            (STRM,HIGH,MAX,FLOW,STREAMFLOW,CFS,8.2)
    %ELSE %IF a request for metric units %THEN
        %COMPUTE AND PRINT HIGHS/LOWS FOR ONE VALUE
            (STRM,HIGH,MAX,FLOW,STREAMFLOW,CMS,8.2)

%IF request for snowfall data %THEN
    %IF not request for metric units %THEN
        %COMPUTE AND PRINT HIGHS/LOWS FOR TWO VALUES
            (SNOW,HIGH,MAX,SNOW,DEPTH,SNOWFALL,
             SNOW DEPTH,INCHES,INCHES,8.1,8.)
    %ELSE %IF a request for metric units %THEN
        %COMPUTE AND PRINT HIGHS/LOWS FOR TWO VALUES
            (SNOW,HIGH,MAX,SNOW,DEPTH,SNOWFALL,
             SNOW DEPTH,CENTIMETERS,CENTIMETERS,8.1,8.)

%IF request for evaporation data %THEN
    %IF not request for metric units %THEN
        %COMPUTE AND PRINT HIGHS/LOWS FOR TWO VALUES
            (EVAP,HIGH,MAX,EVAP,WIND,EVAPORATION,
             WIND MOVEMENT,INCHES,MILES,8.2,8.)
    %ELSE %IF a request for metric units %THEN
        %COMPUTE AND PRINT HIGHS/LOWS FOR TWO VALUES
            (EVAP,HIGH,MAX,EVAP,WIND,EVAPORATION,
             WIND MOVEMENT,MILLIMETERS,KILOMETERS,8.1,8.)
```

4.102 Description of Module LOWEST (LOWEST)

4.102.1 Processing Narrative

LOWEST calls macros to determine which months are requested in the ONLY option, and to compute the lowest monthly value(s) for each of

the pertinent elements. Different parameter values are sent depending on the value of the metric option. The lowest process may be executed multiple times in one access request.

4.102.2 Interface Description

A) There is no parameter list.

B) Global data:
Macro variables:
EL_TEMP : identifies request for temperature data
EL_STRM : identifies request for streamflow data
EL_EVAP : identifies request for evaporation data
METRIC : identifies request for metric units

No SAS data sets are created or used.

C) The macros called by this module are:

GET MONTHS FROM ONLY OPTION (4.92 - GET_MON)
CHECK NUMBER OF OBSERVATIONS (4.20 - NUM_OBS)
COMPUTE AND PRINT HIGHS/LOWS FOR ONE VALUE (4.103 - HI_LO_1)
COMPUTE AND PRINT HIGHS/LOWS FOR TWO VALUES (4.104 - HI_LO_2)

D) No permanent files are used by this module.

E) The module generates macro statements only.

4.102.3 Internal Data Description

No local data are used in this module.

4.102.4 Design Language Description

```
%GET MONTHS FROM ONLY OPTION (LOW,LOWEST,MINIMUM)
%CHECK NUMBER OF OBSERVATIONS (LOW)

%IF request for temperature data %THEN
    %IF not request for metric units %THEN
        %COMPUTE AND PRINT HIGHS/LOWS FOR TWO VALUES
            (TEMP,LOW,MIN,MAXTMP,MINTMP,MAXIMUM TEMPERATURES,
             MINIMUM TEMPERATURES,DEGREES F,DEGREES F,8.,8.)
    %ELSE %IF a request for metric units %THEN
        %COMPUTE AND PRINT HIGHS/LOWS FOR TWO VALUES
            (TEMP,LOW,MIN,MAXTMP,MINTMP,MAXIMUM TEMPERATURES,
             MINIMUM TEMPERATURES,DEGREES C,DEGREES C,8.1,8.1)

%IF request for streamflow data %THEN
    %IF not request for metric units %THEN
        %COMPUTE AND PRINT HIGHS/LOWS FOR ONE VALUE
```

```
            (STRM,LOW,MIN,FLOW,STREAMFLOW,CFS,8.2)
    %ELSE %IF a request for metric units %THEN
        %COMPUTE AND PRINT HIGHS/LOWS FOR ONE VALUE
            (STRM,LOW,MIN,FLOW,STREAMFLOW,CMS,8.2)

%IF request for evaporation data %THEN
    %IF not request for metric units %THEN
        %COMPUTE AND PRINT HIGHS/LOWS FOR TWO VALUES
            (EVAP,LOW,MIN,EVAP,WIND,EVAPORATION,
             WIND MOVEMENT,INCHES,MILES,8.2,8.)
    %ELSE %IF a request for metric units %THEN
        %COMPUTE AND PRINT HIGHS/LOWS FOR TWO VALUES
            (EVAP,LOW,MIN,EVAP,WIND,EVAPORATION,
             WIND MOVEMENT,MILLIMETERS,KILOMETERS,8.1,8.)
```

4.103 Description of Module COMPUTE AND PRINT HIGHS/LOWS FOR ONE VALUE (HI_LO_1)

4.103.1 Processing Narrative

COMPUTE AND PRINT HIGHS/LOWS FOR ONE VALUE first makes a subset of the data for one element according to the months requested in the ONLY option. This module works for elements with one data value recorded per day. One maximum or minimum for the month is calculated for each month in the subset. The second data step lists the monthly maximums or minimums for each requested year. The computations and listings are repeated each time the function is specified in one access request.

4.103.2 Interface Description

A) Parameters:

DSN	- name of data set with the element's data
PROC	- name of data set with the requested months for the process
FUNC	- name of SAS function
ARRRAY	- name of array to be processed
E_TITLE	- name of element for title
UNITS	- units of data items
FORMAT	- format for data items

B) Global data:

Macro variables:

NOBS : the number of observations in a data set (in this case, the number of times the same process is given in one access request

SAS data sets:

&DSN (input) - with variables:

STATION STN_FORM NAME COUNTY YEAR MONTH MONTHSUM
array of 31 daily values

&FUNC1-&FUNC&N (input and output) - with variables:
STATION STN_FORM NAME COUNTY YEAR MONTH MONTHSUM
&FUNC.VALUE BEGMONTH ENDMONTH RANGE

&PROC (input) - with variables:
BEGMONTH ENDMONTH RANGE

PRINT (output) - the standard SAS print file

C) No macros are called by this module.

D) Permanent files: NHIMS.FORMATS

E) The module generates two or more complete DATA steps.

4.103.3 Internal Data Description

Macro variables:
N : control variable for the %DO loop - number of times to execute the macro

SAS variables:

I	(num):	control variable for DO loop
PERIOD	(num):	array name for highs or lows for each month in period (all Januarys, all Februarys, etc.)
ANNUAL	(num):	annual high or low
CORRECT	(num):	used to correct invalid value of month
ROW	(num):	row in which to print values
COLUMN	(num):	column in which to print values
EOF	(boolean):	identifies last record read from input file
M	(num):	control variable for DO loop
CODE	(char):	character value for monthly summary variable
EXTREME	(num):	high or low for the entire period of record
FIRST.STATION	(boolean):	first occurrence of current value of the STATION variable
LAST.STATION	(boolean):	last occurrence of current value of the STATION variable
LAST.YEAR	(boolean):	last occurrence of current value of the YEAR variable

4.103.4 Design Language Description

```
%DO N = 1 %TO &NOBS;
  %* NOBS ¢ 1 only if user wants more than one execution  ;
  %* of the same process in the same access request;
```

```
DATA &FUNC&N(KEEP=STATION YEAR MONTH &FUNC.VALUE MONTHSUM NAME
                  COUNTY STN_FORM BEGMONTH ENDMONTH RANGE );
   SET &DSN;
   DO I = &N TO &N;
      SET &PROC POINT=I;
      /* check range of dates from only option - if annual */
      /* is requested, ENDMONTH = 13, keep entire record */

      IF ENDMONTH = 13  THEN
         &FUNC.VALUE = &FUNC(OF &ARRRAY.1-&ARRRAY.31);
         OUTPUT;
      ELSE IF BEGMONTH _= MONTH _= ENDMONTH  THEN
               DO;
                 &FUNC.VALUE = &FUNC(OF &ARRRAY.1-&ARRRAY.31);
                 OUTPUT;
      ELSE IF BEGMONTH ¢ ENDMONTH THEN
               IF (MONTH ¢= BEGMONTH) OR (MONTH _= ENDMONTH)
                  THEN DO;
                    &FUNC.VALUE =
                         &FUNC(OF &ARRRAY.1-&ARRRAY.31);
                    OUTPUT;
   END;    RUN;

DATA _NULL_;  /* step to print out highs or lows */
   FILE PRINT N=PS;
   ARRAY PERIOD[12] PERIOD1-PERIOD12;
   RETAIN ANNUAL   PERIOD1-PERIOD12  CORRECT;

   ROW = 8;
   CORRECT = 1;
   initialize PERIOD array to missing;

   DO UNTIL(LAST.STATION);  /* print data for one station */
       SET &FUNC&N  END=EOF;
          BY STATION YEAR;
       IF FIRST.STATION OR ROW = 8  THEN /* print headings */
          PUT #2  @1  NAME
                  @65 COUNTY
                  @86 'STATION NO. ' STN_FORM;
          PUT #4  @20  "&PROC.EST DAILY &E_TITLE  IN &UNITS";

          COLUMN = 9;
          DO M = 1 TO 12; /* print month names in heading */
             PUT #6 @COLUMN M MNTH. ;
             COLUMN = COLUMN + 8;
       END;

       IF 1 _= MONTH _= 12 THEN
          COLUMN = ((MONTH - 1) * 8) + 7;
          PERIOD[MONTH] = &FUNC(PERIOD[MONTH],&FUNC.VALUE)
```

```
            ELSE
               COLUMN = ((CORRECT - 1) * 8) + 7;
               PERIOD[CORRECT] = &FUNC(PERIOD[CORRECT],&FUNC.VALUE)

            CODE = ' ';
            IF MONTHSUM NE 0  THEN
               DO;
                  IF MONTHSUM = '....1...'B THEN
                     CODE = 'M';
                  ELSE IF MONTHSUM = '.....1..'B  THEN
                     CODE = 'A';
                  ELSE IF MONTHSUM = '......1.'B  THEN
                     CODE = 'E';
                  ELSE IF MONTHSUM = '.......1'B  THEN
                     CODE = 'T';
               END;

            IF RANGE NE 'ANNUAL'  THEN  /* not ONLY ANNUAL, so */
                                        /* print monthly values */
               PUT #ROW @COLUMN &FUNC.VALUE &FORMAT  CODE $CHAR1.;

            IF ENDMONTH = 13  THEN  /* request for annual data */
               ANNUAL = &FUNC(ANNUAL,&FUNC.VALUE);
               IF LAST.YEAR  THEN
                  PUT #ROW @125 ANNUAL &FORMAT;
                  ANNUAL = .;

            IF LAST.YEAR  THEN
               ROW = ROW + 1;
               CORRECT = 1;
               IF ROW ¢ 55 THEN
                  print footnotes;
                  ROW = 8;
                  PUT _PAGE_;

            ELSE CORRECT = MONTH + 1;

         END;  /* print data for one station */


         EXTREME = &FUNC(OF PERIOD1-PERIOD12);
         print PERIOD array at bottom of columns;
         print EXTREME for period at bottom of page;
         print footnotes;

         IF EOF  THEN STOP;
         ELSE PUT _PAGE_;
   RUN;

%END;
```

4.104 Description of Module COMPUTE AND PRINT HIGHS/LOWS FOR TWO VALUES (HI_LO_2)

4.104.1 Processing Narrative

COMPUTE AND PRINT HIGHS/LOWS FOR TWO VALUES first makes a subset of the data for one element according to the months requested in the ONLY option. This module works for elements with two data values recorded per day. Two maximums or minimums for the month are calculated for each month in the subset. The second data step lists the monthly maximums or minimums for each requested year. The computations and listings are repeated for each time the function is specified in one access request.

4.104.2 Interface Description

A) Parameters:

DSN	- name of data set with the element's data
PROC	- name of data set with the requested months for the process
FUNC	- name of SAS function
ARRRAY1	- name of first array to be processed
ARRRAY2	- name of second array to be processed
E_TITL1	- name of first value for title
E_TITL2	- name of second value for title
UNITS1	- units of data items for first value
UNITS2	- units of data items for second value
FORMAT1	- format for data items for first value
FORMAT2	- format for data items for second value

B) Global data:

Macro variables:

NOBS : the number of observations in a data set (in this case, the number of times the same process is given in one access request

SAS data sets:

&DSN (input) - with variables:
STATION STN_FORM NAME COUNTY YEAR MONTH MONTHSUM
two arrays of 31 daily values each

&FUNC1-&FUNC&N (input and output) - with variables:
STATION STN_FORM NAME COUNTY YEAR MONTH MONTHSUM
&FUNC.VAL1 &FUNC.VAL2 BEGMONTH ENDMONTH RANGE

&PROC (input) - with variables:
BEGMONTH ENDMONTH RANGE

PRINT (output) - the standard SAS print file

C) No macros are called by this module.

D) Permanent files: NHIMS.FORMATS

E) The module generates two or more complete DATA steps.

4.104.3 Internal Data Description

Macro variables:

N : control variable for the %DO loop - number of times to execute the macro

SAS variables:

I	(num):	control variable for DO loop
PERA	(num):	array name for highs or lows for each month in period (all Januarys, all Februarys, etc.) for first monthly value
PERB	(num):	array name for highs or lows for each month in period (all Januarys, all Februarys, etc.) for second monthly value
ANNUAL1	(num):	annual high or low for first monthly value
ANNUAL2	(num):	annual high or low for second monthly value
CORRECT	(num):	used to correct invalid value of month
ROW	(num):	row in which to print values for first value
ROW2	(num):	row in which to print values for second value
COLUMN	(num):	column in which to print values
EOF	(boolean):	identifies last record read from input file
M	(num):	control variable for DO loop
CODE1	(char):	value of monthly summary for first value
CODE2	(char):	value of monthly summary for second value
EXTREME1	(num):	high or low for entire period for first value
EXTREME2	(num):	high or low for entire period for second value
FIRST.STATION	(boolean):	first occurrence of current value of the STATION variable
LAST.STATION	(boolean):	last occurrence of current value of the STATION variable
LAST.YEAR	(boolean):	last occurrence of current value of the YEAR variable

4.104.4 Design Language Description

```
%DO N = 1 %TO &NOBS;
  %* NOBS ¢ 1 only if user wants more than one execution  ;
  %* of the same process in the same access request;

  DATA &FUNC&N(KEEP=STATION YEAR MONTH &FUNC.VAL1 &FUNC.VAL2
                    MONTHSUM  RANGE
                    NAME COUNTY STN_FORM BEGMONTH ENDMONTH);
```

```
   SET &DSN;
   DO I = &N TO &N;
      SET &PROC POINT=I;
      /* check range of dates from only option - if annual */
      /* is requested, ENDMONTH = 13, keep entire record */

      IF ENDMONTH = 13  THEN
         &FUNC.VAL1 = &FUNC(OF &ARRRAY1.1-&ARRRAY1.31);
         &FUNC.VAL2 = &FUNC(OF &ARRRAY2.1-&ARRRAY2.31);
         OUTPUT;
      ELSE IF BEGMONTH _= MONTH _= ENDMONTH  THEN
              DO;
                &FUNC.VAL1 = &FUNC(OF &ARRRAY1.1-&ARRRAY1.31);
                &FUNC.VAL2 = &FUNC(OF &ARRRAY2.1-&ARRRAY2.31);
                OUTPUT;
      ELSE IF BEGMONTH ¢ ENDMONTH THEN
              IF (MONTH ¢= BEGMONTH) OR (MONTH _= ENDMONTH)
                 THEN DO;
                     &FUNC.VAL1 =
                          &FUNC(OF &ARRRAY1.1-&ARRRAY1.31);
                     &FUNC.VAL2 =
                          &FUNC(OF &ARRRAY2.1-&ARRRAY2.31);
                     OUTPUT;
   END;
RUN;


DATA _NULL_;  /* step to print out highs or lows */
   FILE PRINT N=PS;
   ARRAY PERA[12] PERA1-PERA12;
   ARRAY PERB[12] PERB1-PERB12;
   RETAIN ANNUAL1 ANNUAL2  PERA1-PERA12
          PERB1-PERB12  CORRECT;

   ROW = 8;
   ROW2 = 9;
   CORRECT = 1;

   initialize PERA and PERB arrays to missing;

   DO UNTIL(LAST.STATION);  /* print data for one station */
       SET &FUNC&N  END=EOF;
          BY STATION YEAR;
       IF FIRST.STATION OR ROW = 8  THEN /* print headings */
          PUT #2  @1  NAME
                  @65 COUNTY
                  @86 'STATION NO. ' STN_FORM;
          PUT #4  @10  "&PROC.EST DAILY &E_TITL1  IN &UNITS1"
                       " AND &E_TITL2 IN &UNITS2"
```

```
   COLUMN = 9;
   DO M = 1 TO 12; /* print month names in heading */
      PUT #6 @COLUMN M MNTH. ;
      COLUMN = COLUMN + 8;
END;

IF 1 _= MONTH _= 12 THEN
   COLUMN = ((MONTH - 1) * 8) + 7;
   PERA[MONTH] = &FUNC(PERA[MONTH],&FUNC.VAL1)
   PERB[MONTH] = &FUNC(PERB[MONTH],&FUNC.VAL2)
ELSE
   COLUMN = ((CORRECT - 1) * 8) + 7;
   PERA[CORRECT] = &FUNC(PERA[CORRECT],&FUNC.VAL1)
   PERB[CORRECT] = &FUNC(PERB[CORRECT],&FUNC.VAL2)

CODE1 = ' ';
CODE2 = ' ';
IF MONTHSUM NE 0  THEN
   DO;
     IF MONTHSUM = '....1...'B THEN
        CODE1 = 'M';
     ELSE IF MONTHSUM = '.....1..'B  THEN
        CODE1 = 'A';
     ELSE IF MONTHSUM = '......1.'B  THEN
        CODE1 = 'E';
     ELSE IF MONTHSUM = '.......1'B  THEN
        CODE1 = 'T';

     IF MONTHSUM = '1.......'B THEN
        CODE2 = 'M';
     ELSE IF MONTHSUM = '.1......'B  THEN
        CODE2 = 'A';
     ELSE IF MONTHSUM = '..1.....'B  THEN
        CODE2 = 'E';
     ELSE IF MONTHSUM = '...1....'B  THEN
        CODE2 = 'T';
   END;

IF RANGE NE 'ANNUAL'  THEN  /* not ONLY ANNUAL, so */
                            /* print monthly values */
   PUT #ROW  @COLUMN &FUNC.VAL1 &FORMAT1
                     CODE1 $CHAR1.;
   PUT #ROW2 @COLUMN &FUNC.VAL2 &FORMAT2
                     CODE2 $CHAR1.;


IF ENDMONTH = 13  THEN  /* request for annual data */
   ANNUAL1 = &FUNC(ANNUAL1,&FUNC.VAL1);
   ANNUAL2 = &FUNC(ANNUAL2,&FUNC.VAL2);
   IF LAST.YEAR  THEN
      PUT #ROW  @125 ANNUAL1 &FORMAT1;
```

```
                PUT #ROW2 @125 ANNUAL2 &FORMAT2;
                ANNUAL1 = .;
                ANNUAL2 = .;

            IF LAST.YEAR  THEN
               ROW = ROW + 2;
               ROW2 = ROW + 1;
               CORRECT = 1;
               IF ROW ¢ 55 THEN
                  print footnotes;
                  ROW = 8;
                  ROW2 = 9;
                  PUT _PAGE_;

               ELSE CORRECT = MONTH + 1;

         END;  /* print data for one station */

         EXTREME1 = &FUNC(OF PERA1-PERA12);
         EXTREME2 = &FUNC(OF PERB1-PERB12);

         print PERA and PERB arrays at bottom of columns;
         print EXTREME1 and EXTREME2 at bottom of page;
         print footnotes;

         IF EOF  THEN STOP;
         ELSE PUT _PAGE_;
   RUN;

%END;
```

4.105 Description of Module EXTREME (EXTREME)

4.105.1 Processing Narrative

EXTREME calls macros to determine which months are requested in the ONLY option, and to compute the extreme (highest high and lowest low) monthly values for each of the pertinent elements. Different parameter values are sent depending on the value of the metric option. The highest process may be executed multiple times in one access request.

©[E4.105.2 Interface Description©[F

A) There is no parameter list.

B) Global data:

Macro variables:
EL_TEMP : identifies request for temperature data
METRIC : identifies request for metric units

No SAS data sets are created or used.

C) The macros called by this module are:

GET MONTHS FROM ONLY OPTION (4.92 - GET_MON)
CHECK NUMBER OF OBSERVATIONS (4.20 - NUM_OBS)
COMPUTE AND PRINT EXTREMES (4.106 - PRT_EX)

D) No permanent files are used by this module.

E) The module generates macro statements only.

4.105.3 Internal Data Description

No local data are used in this module.

4.105.4 Design Language Description

```
%GET MONTHS FROM ONLY OPTION (EXTREM,EXTREME,EXTREME)
%CHECK NUMBER OF OBSERVATIONS (EXTREM)

%IF request for temperature data %THEN
    %IF not request for metric units %THEN
        %COMPUTE AND PRINT EXTREMES FOR TWO VALUES
            (TEMP,MAXTMP,MINTMP,MAXIMUM TEMPERATURES,
             MINIMUM TEMPERATURES,DEGREES F,DEGREES F,8.,8.)
    %ELSE %IF a request for metric units %THEN
        %COMPUTE AND PRINT EXTREMES FOR TWO VALUES
            (TEMP,MAXTMP,MINTMP,MAXIMUM TEMPERATURES,
             MINIMUM TEMPERATURES,DEGREES C,DEGREES C,8.1,8.1)
```

4.106 Description of Module COMPUTE AND PRINT EXTREMES (PRT_EX)

4.106.1 Processing Narrative

COMPUTE AND PRINT EXTREMES first makes a subset of the data for one element according to the months requested in the ONLY option. This module works for elements with one maximum and one minimum value recorded per day. Two extremes are calculated for each month in the subset. The second data step lists the monthly extremes for each requested year. The computations and listings are repeated each time the function is specified in one access request.

4.106.2 Interface Description

A) Parameters:
- DSN - name of data set with the element's data
- ARRRAY1 - name of first array to be processed
- ARRRAY2 - name of second array to be processed
- E_TITL1 - name of first value for title
- E_TITL2 - name of second value for title
- UNITS1 - units of data items for first value
- UNITS2 - units of data items for second value
- FORMAT1 - format for data items for first value
- FORMAT2 - format for data items for second value

B) Global data:

Macro variables:

NOBS : the number of observations in a data set (in this case, the number of times the same process is given in one access request

SAS data sets:

&DSN (input) - with variables:
STATION STN_FORM NAME COUNTY YEAR MONTH MONTHSUM
two arrays of 31 daily values each

EXTREM1-EXTREM&N (input and output) - with variables:
STATION STN_FORM NAME COUNTY YEAR MONTH MONTHSUM
HIGH LOW BEGMONTH ENDMONTH RANGE

EXTREM (input) - with variables:
BEGMONTH ENDMONTH RANGE

PRINT (output) - the standard SAS print file

C) No macros are called by this module.

D) Permanent files: NHIMS.FORMATS

E) The module generates two or more complete DATA steps.

4.106.3 Internal Data Description

Macro variables:

N : control variable for the %DO loop - number of times to execute the macro

SAS variables:

I (num): control variable for DO loop
PERHI (num): array name for highs for each month in period (all Januarys, all Februarys, etc.) for maximum monthly value

```
PERLO    (num): array name for lows for each month
                in period (all Januarys, all Februarys, etc.)
                for minimum monthly value
ANNUAL1 (num): annual high or low for first monthly value
ANNUAL2 (num): annual high or low for second monthly value
CORRECT (num): used to correct invalid value of month
ROW     (num): row in which to print values for first value
ROW2    (num): row in which to print values for second value
COLUMN  (num): column in which to print values
EOF (boolean): identifies last record read from input file
M       (num): control variable for DO loop
CODE1  (char): value of monthly summary for first value
CODE2  (char): value of monthly summary for second value
EXTREME1(num): high or low for entire period for first value
EXTREME2(num): high or low for entire period for second value
FIRST.STATION (boolean): first occurrence of current value of
                         the STATION variable
LAST.STATION  (boolean): last occurrence of current value of
                         the STATION variable
LAST.YEAR     (boolean): last occurrence of current value of
                         the YEAR variable
```

4.106.4 Design Language Description

```
%DO N = 1 %TO &NOBS;
  %* NOBS ¢ 1 only if user wants more than one execution  ;
  %* of the same process in the same access request;

  DATA EXTREM&N(KEEP=STATION YEAR MONTH HIGH LOW MONTHSUM NAME
                     COUNTY STN_FORM BEGMONTH ENDMONTH RANGE );
     SET &DSN;
     DO I = &N TO &N;
        SET EXTREM POINT=I;
        /* check range of dates from only option - if annual */
        /* is requested, ENDMONTH = 13, keep entire record */

        IF ENDMONTH = 13  THEN DO;
           HIGH = MAX(OF &ARRRAY1.1-&ARRRAY1.31);
           LOW  = MIN(OF &ARRRAY2.1-&ARRRAY2.31);
           OUTPUT; END;
          ELSE IF BEGMONTH _= MONTH _= ENDMONTH  THEN
                DO;
                  HIGH = MAX(OF &ARRRAY1.1-&ARRRAY1.31);
                  LOW  = MIN(OF &ARRRAY2.1-&ARRRAY2.31);
                  OUTPUT; END;
        ELSE IF BEGMONTH ¢ ENDMONTH THEN
                IF (MONTH ¢= BEGMONTH) OR (MONTH _= ENDMONTH)
                   THEN DO;
                       HIGH = MAX(OF &ARRRAY1.1-&ARRRAY1.31);
                       LOW  = MIN(OF &ARRRAY2.1-&ARRRAY2.31);
                       OUTPUT;
     END;     RUN;
```

```
DATA _NULL_;  /* step to print out highs and lows */
    FILE PRINT N=PS;
    ARRAY PERHI[12] PERHI1-PERHI12;
    ARRAY PERLO[12] PERLO1-PERLO12;
    RETAIN ANNUAL1 ANNUAL2  PERHI1-PERHI12
           PERLO1-PERLO12  CORRECT;

    initialize ROW to 8 and ROW2 to 9;
    CORRECT = 1;
    initialize PERHI and PERLO arrays to missing;

    DO UNTIL(LAST.STATION);  /* print data for one station */
        SET EXTREM&N  END=EOF;
           BY STATION YEAR;
        IF FIRST.STATION OR ROW = 8  THEN /* print headings */
           PUT #2  @1  NAME
                   @65 COUNTY
                   @86 'STATION NO. ' STN_FORM;
           PUT #4  @10 "HIGHEST DAILY &E_TITL1  IN &UNITS1"
                       " AND LOWEST DAILY &E_TITL2 IN &UNITS2"
           COLUMN = 9;
           DO M = 1 TO 12; /* print month names in heading */
              PUT #6 @COLUMN M MNTH. ;
              COLUMN = COLUMN + 8;
        END; /* print headings */

        IF 1 <= MONTH <= 12 THEN
           COLUMN = ((MONTH - 1) * 8) + 7;
           PERHI[MONTH] = MAX(PERHI[MONTH],HIGH)
           PERLO[MONTH] = MIN(PERLO[MONTH],LOW)
        ELSE
           COLUMN = ((CORRECT - 1) * 8) + 7;
           PERHI[CORRECT] = MAX(PERHI[CORRECT],HIGH)
           PERLO[CORRECT] = MIN(PERLO[CORRECT],LOW)

        CODE1 = ' ';
        CODE2 = ' ';
        IF MONTHSUM NE 0  THEN /* assign monthly codes */
             IF MONTHSUM = '....1...'B THEN
                CODE1 = 'M';

             ELSE IF MONTHSUM = '.....1..'B  THEN
                CODE1 = 'A';
             ELSE IF MONTHSUM = '......1.'B  THEN
                CODE1 = 'E';
             ELSE IF MONTHSUM = '.......1'B  THEN
                CODE1 = 'T';
```

```
            IF MONTHSUM = '1.......'B THEN
               CODE2 = 'M';
            ELSE IF MONTHSUM = '.1......'B  THEN
               CODE2 = 'A';
            ELSE IF MONTHSUM = '..1.....'B  THEN
               CODE2 = 'E';
            ELSE IF MONTHSUM = '...1....'B  THEN
               CODE2 = 'T';
         END;  /* assign monthly codes */

          IF RANGE NE 'ANNUAL'  THEN  /* not ONLY ANNUAL, so */
                                      /* print monthly values */
             PUT #ROW  @COLUMN HIGH &FORMAT1  CODE1 $CHAR1.;
             PUT #ROW2 @COLUMN LOW  &FORMAT2  CODE2 $CHAR1.;

          IF ENDMONTH = 13  THEN  /* request for annual data */
             ANNUAL1 = MAX(ANNUAL1,HIGH);
             ANNUAL2 = MIN(ANNUAL2,LOW);
             IF LAST.YEAR  THEN
                PUT #ROW  @125 ANNUAL1 &FORMAT1;
                PUT #ROW2 @125 ANNUAL2 &FORMAT2;
                ANNUAL1 = .;
                ANNUAL2 = .;

          IF LAST.YEAR  THEN
             ROW = ROW + 2;
             ROW2 = ROW + 1;
             CORRECT = 1;
             IF ROW ¢ 55 THEN
                print footnotes;
                initialize ROW and ROW2;
                PUT _PAGE_;
          ELSE CORRECT = MONTH + 1;

      END;  /* print data for one station */

      EXTREME1 = MAX(OF PERHI1-PERHI12);
      EXTREME2 = MIN(OF PERLO1-PERLO12);

      print PERHI and PERLO arrays at bottom of columns;
      print EXTREME1 and EXTREME2 at bottom of page;
      print footnotes;
      IF EOF  THEN STOP;
      ELSE PUT _PAGE_;
   RUN;
%END;
```

4.107 Description of Module RANK ORDER (RANKORD)

4.107.1 Processing Narrative

RANK ORDER calls macros to determine which months are requested in the ONLY option, to determine how many years are in the requested period of record, and to sort and list the monthly values. Different parameter values are sent depending on the value of the metric option. The highest process may be executed multiple times in one access request.

4.107.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:

```
EL_STRM : identifies request for streamflow data
EL_PRCP : identifies request for precipitation data
EL_TEMP : identifies request for temperature data
EL_SNOW : identifies request for snowfall data
METRIC  : identifies request for metric units
```

No SAS data sets are created or used.

C) The macros called by this module are:

```
GET MONTHS FROM ONLY OPTION  (4.92 - GET_MON)
CHECK NUMBER OF OBSERVATIONS (4.20 - NUM_OBS)
CALCULATE NUMBER OF YEARS    (4.108 - CAL_YRS)
COMPUTE RANK ORDER  (4.109 - C_RANK)
```

D) No permanent files are used by this module.

E) The module generates macro statements only.

4.107.3 Internal Data Description

No local data are used in this module.

4.107.4 Design Language Description

```
%GET MONTHS FROM ONLY OPTION (RANKORD,RANK ORDER,RANK)
%CHECK NUMBER OF OBSERVATIONS (RANKORD)

%IF request for streamflow data %THEN
    %CALCULATE NUMBER OF YEARS (STRM)
    %IF not request for metric units %THEN
        %COMPUTE RANK ORDER (STRM,STREAMFLOW,CFS,FLOW,10.2)
    %ELSE %IF a request for metric units %THEN
        %COMPUTE RANK ORDER (STRM,STREAMFLOW,CMS,FLOW,10.2)
```

```
%IF request for precipitation data %THEN
    %CALCULATE NUMBER OF YEARS (PRCP)
    %IF not request for metric units %THEN
        %COMPUTE RANK ORDER (PRCP,PRECIPITATION,INCHES,PRECIP,10.2)
    %ELSE %IF a request for metric units %THEN
        %COMPUTE RANK ORDER (PRCP,PRECIPITATION,MILLIMETERS,
                             PRECIP,10.1)

%IF request for temperature data %THEN
    %CALCULATE NUMBER OF YEARS (TEMP)
    %IF not request for metric units %THEN
        %COMPUTE RANK ORDER (TEMP,MAXIMUM TEMPERATURES,DEGREES F,
                             MAXTMP,10.)
    %ELSE %IF a request for metric units %THEN
        %COMPUTE RANK ORDER (TEMP,MAXIMUM TEMPERATURES,DEGREES C,
                             MAXTMP,10.1)

%IF request for snow fall data %THEN
    %CALCULATE NUMBER OF YEARS (SNOW)
    %IF not request for metric units %THEN
        %COMPUTE RANK ORDER (SNOW,SNOWFALL,INCHES,SNOW,10.1)
    %ELSE %IF a request for metric units %THEN
        %COMPUTE RANK ORDER (SNOW,SNOWFALL,CENTIMETERS,SNOW,10.1)
```

4.108 Description of Module CALCULATE NUMBER OF YEARS (CAL_YRS)

4.108.1 Processing Narrative

CALCULATE NUMBER OF YEARS reads the subset of data to be processed, and retains in a data set the number of years in the period of record for each station.

4.108.2 Interface Description

A) Parameters: DSN - the name of the input data set

B) Global data:
No global macro variables are used in this module.

SAS data sets:
&DSN (input) - with variables:
STATION YEAR and any other variables

YEARS (output) - with variables:
STATION NUM_YRS

C) No macros are called by this module.

D) No permanent files are used by this module.

E) The module generates one complete DATA step.

4.108.3 Internal Data Description

No local macro variables are used in this module.

SAS variables:

```
FIRST_YR  (num): first year of period of record for a station
FIRST.STATION (boolean): first occurrence of current value of
                         the STATION variable
LAST.STATION  (boolean): last occurrence of current value of
                         the STATION variable
```

4.108.4 Design Language Description

```
DATA YEARS(KEEP=STATION NUM_YRS);
   RETAIN FIRST_YR;
   SET &DSN;
     BY STATION ;

   IF FIRST.STATION  THEN
      FIRST_YR = YEAR;
   IF LAST.STATION  THEN
      NUM_YRS = YEAR - FIRST_YR + 1;
      OUTPUT;
RUN;
```

4.109 Description of Module COMPUTE RANK ORDER (C_RANK)

4.109.1 Processing Narrative

COMPUTE RANK ORDER first makes a subset of the data for one element according to the months requested in the ONLY option. At the same time, the data is reshaped so that each observation contains one daily value. The data is sorted in descending order by the daily values for each station, and five times the number of requested years are kept for output. Each of the daily values is printed next to the date of occurrence. The computations and listings are repeated each time the function is specified in one access request.

4.109.2 Interface Description

A) Parameters: DSN - the name of the input data set
E_TITLE - the full name of the element for heading
UNITS - units for the daily value
ARRRAY - array name for the 31 daily values
FORMAT - output format for the daily values

B) Global data:
Macro variables:
NOBS : number of observations in a data set

SAS data sets:
&DSN (input) - with variables:
STATION YEAR and any other variables

RANKORD (input) - with variables:
BEGMONTH ENDMONTH RANGE

RANK1-RANK&N (input and output) - with variables:
STATION YEAR MONTH DAY DAYVALUE NAME COUNTY STN_FORM
BEGMONTH ENDMONTH RANGE

YEARS (input) - with variables:
STATION NUM_YRS

ORDER1-ORDER&N (input and output) - with variables:
same as RANK1-RANK&N

PRINT (output) - standard SAS print file

C) No macros are called by this module.

D) No permanent files are used by this module.

E) The module generates several complete DATA and PROC steps.

4.109.3 Internal Data Description

Macro variables:
N : control variable for the %DO loop - number of times to execute the macro

SAS variables:
LEGAL (num): identifies legal months from ONLY option
HOWMANY (num): the number of sorted daily values to keep
COUNT (num): the number of records kept in the data set
DATE (num): month, day, and year as SAS date value
ROW (num): row for printing data values
COLUMN (num): column for printing data values

4.109.4 Design Language Description

```
%DO N = 1 %TO &NOBS;
  %* NOBS ¢ 1 only if user wants more than one execution  ;
  %* of the same process in the same access request;

  DATA RANK&N  (KEEP=STATION YEAR MONTH DAY DAYVALUE NAME
                     COUNTY STN_FORM BEGMONTH ENDMONTH RANGE);
     SET &DSN;
     DO I = &N TO &N;
        SET RANKORD POINT=I;
        /* check range of dates from only option - if annual */
        /* is requested, ENDMONTH = 13, keep entire record */

        IF ENDMONTH = 13  THEN
           LEGAL = 1;
        ELSE IF BEGMONTH _= MONTH _= ENDMONTH  THEN
                DO;
                  LEGAL = 1;
        ELSE IF BEGMONTH ¢ ENDMONTH THEN
                IF (MONTH ¢= BEGMONTH) OR (MONTH _= ENDMONTH)
                   THEN
                       LEGAL = 1;

        IF LEGAL = 1 THEN DO;
           NUMDAYS = 31;
           IF MONTH = 4 OR MONTH = 6 OR MONTH = 9 OR
              MONTH = 11  THEN
                 NUMDAYS = 30;
           ELSE IF MONTH = 2  THEN
                   IF MOD(YEAR,4) NE 0  THEN NUMDAYS = 28;
                                        ELSE NUMDAYS = 29;
           DO DAY = 1 TO NUMDAYS;
                 DAYVALUE = &ARRAY[DAY];
                 OUTPUT;
           END;
        END;  /* if legal month then output */
     END; /* check range of dates from ONLY option */
  RUN;

      PROC SORT DATA=RANK&N;
     BY STATION  DESCENDING DAYVALUE;
  RUN;

    DATA ORDER&N(DROP=COUNT HOWMANY);
     /* keep only 5 * number of years */

     SET YEARS;
     IF NUM_YRS ¢ 45  THEN
        NUM_YRS = 45 ;  /* keep 45 years max */
     HOWMANY = NUM_YRS * 5;
     COUNT = 1;
```

```
      DO UNTIL(LAST.STATION);
         SET RANK&N;
            BY STATION;
         IF COUNT _= HOWMANY THEN OUTPUT;
         COUNT = COUNT + 1;
      END;
   RUN;


   DATA _NULL_; /* step to print rank ordering */

      DO TIMES = 1 TO 5;  /* print one column  */
         COLUMN = ((TIMES - 1) * 25) + 5;
         ROW = 11;
         COUNT = 1;

         DO UNTIL(COUNT ¢ NUM_YRS);

            SET ORDER&N  END=EOF;
                BY STATION;
            IF FIRST.STATION  THEN
               print headings;
               print YEAR and MONTH in heading;
            DATE = MDY(MONTH,DAY,YEAR);
            PUT #ROW @COLUMN DAYVALUE &FORMAT  DATE DATE9. ;
            ROW = ROW + 1;
            COUNT = COUNT + 1;
            IF LAST.STATION  THEN
               print YEAR and MONTH in heading;
               COUNT = 100;  /* out of range value */
               PUT _PAGE_;
            IF EOF  THEN STOP;

         END;

      END;
   RUN;

%END;
```

4.110 Description of Module HIGH OCCURRENCES (HI_OCC)

4.110.1 Processing Narrative

HIGH OCCURRENCES calls the macros to create a subset of data for the months requested in the ONLY option, and to locate a threshold given by the user. Default thresholds are passed as parameters, and the defaults differ depending on the value of the metric option. Macros are then called to compute and list the high occurrences. High occurrences may be executed multiple times in one access request.

4.110.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:
METRIC : identifies request for metric units

No SAS data sets are created or used.

C) The macros called by this module are:

GET MONTHS AND ONE THRESHOLD (4.112 - GET_M1T)
CHECK NUMBER OF OBSERVATIONS (4.20 - NUM_OBS)
COMPUTE HIGH OCCURRENCES (4.113 - C_HIOCC)

D) No permanent files are used by this module.

E) The module generates macro statements only.

4.110.3 Internal Data Description

No local data are used in this module.

4.110.4 Design Language Description

```
%GET MONTHS AND ONE THRESHOLD
            (HI_OCC,HIGH OCCURRENCES,HIGH OCCURENCES)
%CHECK NUMBER OF OBSERVATIONS (HI_OCC)

%IF not a request for metric units %THEN
   %COMPUTE HIGH OCCURRENCES(90)
%ELSE %IF a request for metric units %THEN
   %COMPUTE HIGH OCCURRENCES(32)
```

4.111 Description of Module LOW OCCURRENCES (LO_OCC)

4.111.1 Processing Narrative

LOW OCCURRENCES calls the macros to create a subset of data for the months requested in the ONLY option, and to locate a threshold given by the user. Default thresholds are passed as parameters, and the defaults differ depending on the value of the metric option. Macros are then called to compute and list the low occurrences. Low occurrences may be executed multiple times in one access request.

4.111.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:
METRIC : identifies request for metric units

No SAS data sets are created or used.

C) The macros called by this module are:

GET MONTHS AND ONE THRESHOLD (4.112 - GET_M1T)
CHECK NUMBER OF OBSERVATIONS (4.20 - NUM_OBS)
COMPUTE LOW OCCURRENCES (4.114 - C_LOOCC)

D) No permanent files are used by this module.

E) The module generates macro statements only.

4.111.3 Internal Data Description

No local data are used in this module.

©[E4.111.4 Design Language Description©[F

```
%GET MONTHS AND ONE THRESHOLD
            (LO_OCC,LOW OCCURRENCES,LOW OCCURENCES)
%CHECK NUMBER OF OBSERVATIONS (LO_OCC)

%IF not a request for metric units %THEN
   %COMPUTE LOW OCCURRENCES(32)
%ELSE %IF a request for metric units %THEN
   %COMPUTE LOW OCCURRENCES(0)
```

4.112 Description of Module GET MONTHS AND ONE THRESHOLD (GET_M1T)

4.112.1 Processing Narrative

GET MONTHS AND ONE THRESHOLD parses the options given by the user with the HIGH or LOW OCCURRENCES processes, and finds the requested months and threshold. The user may give a different threshold for each station requested, simply by explicitly coding the station number before the keyword THRESHOLD. If any of the options are not recognized, the default options will be used.

4.112.2 Interface Description

A) Parameters: DSN - data set in which to store the options
NAME1 - name of the process
NAME2 - a second name by which to recognize the process

B) Global data:

No global macro variables are used in this module.

SAS data sets:
PROCESS (input) - with variables:
PRC_NAME PARMETER

&DSN (output) - with variables:
BEGMONTH ENDMONTH RANGE THRESHLD STN_REQ

C) No macros are called by this module.

D) No permanent files are used by this module.

E) The module generates one complete DATA step.

4.112.3 Internal Data Description

No local macro variables are used in this module.

SAS variables:

MON	(char):	array containing month names
WORD	(char):	array with words scanned from process parameter
PREVIOUS	(num):	identifies when to write an observation
EOF	(boolean):	identifies the last record read from input file
POS	(num):	position of keyword THRESHOLD in parameter

4.112.4 Design Language Description

```
DATA &DSN (KEEP=BEGMONTH  ENDMONTH  RANGE THRESHLD STN_REQ);
   /* data step to identify the months specified */
   /* in the ONLY option and the threshold       */
   LENGTH MON1-MON13 $ 10  WORD1-WORD4  $ 10  RANGE $ 18
          THRESHLD 2   STN_REQ $ 8;
```

```
RETAIN  MON1  'JANUARY'   MON2  'FEBRUARY'   MON3  'MARCH'
        MON4  'APRIL'     MON5  'MAY'        MON6  'JUNE'
        MON7  'JULY'      MON8  'AUGUST'     MON9  'SEPTEMBER'
        MON10 'OCTOBER'   MON11 'NOVEMBER'   MON12 'DECEMBER'
        MON13 'ANNUAL'   PREVIOUS 0;

ARRAY MON[13] MON1-MON13;

/* set default values */
BEGMONTH = 1;
ENDMONTH = 13;
RANGE = 'JANUARY-ANNUAL';
THRESHLD = .;
STN_REQ = . ;
PREVIOUS = 0;

DO UNTIL(PRC_NAME = 'PROCESS');
   SET PROCESS END=EOF;

   IF PRC_NAME = "&NAME1"  OR PRC_NAME = "&NAME2"  THEN
      DO; /* get information on this process */

     IF PARMETER is equal to missing
        THEN OUTPUT;   /* output default values */

     ELSE IF PARMETER is not equal to missing THEN DO;
            WORD1 = SCAN(PARMETER,1,' ');
            IF WORD1 = 'ONLY'  THEN
               DO;
                 PREVIOUS = 1;
                 WORD2 = SCAN(PARMETER,2,' ');
                 WORD3 = SCAN(PARMETER,3,' ');
                 IF WORD3 = 'TO' THEN
                    WORD4 = SCAN(PARMETER,4,' ');
                    RANGE = TRIM(WORD2) !! '-' !! WORD4;
                 ELSE
                    WORD4 = ' ';
                    RANGE = WORD2;

                 DO I = 1 TO 13;
                    IF WORD2 = MON[I]  THEN
                       BEGMONTH = I;
                 END;
                 IF WORD4 NE ' '  THEN
                    DO I = 1 TO 13;
                       IF WORD4 = MON[I]  THEN
                          ENDMONTH = I;
                    END;
```

```
            ELSE ENDMONTH = BEGMONTH;
          END;

        ELSE IF WORD1 = 'THRESHOLD'  THEN
          DO;
            PREVIOUS = 1;
            WORD2 = SCAN(PARMETER,2,' ');
            THRESHLD = WORD2;
          END;
        ELSE
          DO;
            POS = INDEX(PARMETER,'THRESHOLD')
            IF POS NE 0  THEN
              DO; /* user requested different    */
                  /* thresholds for each station */
                STN_REQ = SCAN(PARMETER,1,' ');
                WORD2 = SCAN(PARMETER,3,' ');
                THRESHLD = WORD2;
                OUTPUT; /* save each station as */
                        /* separate observation */
              END;
            ELSE /* illegal option - set to default */
              PREVIOUS = 1;
              PARMETER = .;

IF EOF THEN
  IF PREVIOUS NE 0 THEN OUTPUT;
  STOP;
END;  /* until process name = process */
IF PREVIOUS NE 0 THEN  OUTPUT;

 RUN;
```

4.113 Description of Module COMPUTE HIGH OCCURRENCES (C_HIOCC)

4.113.1 Processing Narrative

COMPUTE HIGH OCCURRENCES makes a subset of the element data containing the requested months; if the user did not specify a threshold, the default threshold is set. The high occurrences, or number of days above the threshold, are calculated for years with complete records, and the values are then listed. The computations and listings are repeated each time the process is specified in one access request.

4.113.2 Interface Description

A) Parameters: DEFAULT - default threshold

B) Global data:

Macro variables:
NOBS - the number of observations in a data set
METRIC - identifies request for metric units

SAS data sets:
TEMP (input) - with variables:
STATION STN_FORM NAME COUNTY YEAR MONTH MONTHSUM
MAXTMP1-MAXTMP31 MINTMP1-MINTMP31 AVEMAX AVEMIN

HI_OCC (input) - with variables:
BEGMONTH ENDMONTH RANGE THRESHLD STN_REQ

HI_OCC1-HI_OCC&N (input and output) - with variables:
STATION YEAR MONTH MONTHSUM STN_FORM NAME COUNTY
MAXTMP1-MAXTMP31 THRESHLD STN_REQ

HO_SUB1-HO_SUB&N (input and output) - with variables:
STATION STN_FORM NAME COUNTY YEAR ABOVE
FIRST_DY LAST_DY FIRST_MN LAST_MN THRESHLD

PRINT (output) - standard SAS print file

C) No macros are called by this module.

D) No permanent files are used by this module.

E) The module generates several complete DATA steps.

4.113.3 Internal Data Description

Macro variables:
N : control variable for the %DO loop - number of times to execute the macro

SAS variables:

I	(num):	control variable for DO loop
MISS_YR	(num):	identifies missing data in one year
FIRST	(num):	flags when first day and month have been found
MATCH	(num):	flags match of stations when user specifies a different threshold for each station
EOF	(boolean):	identifies last record read from input file
DAY	(num):	control variable for DO loop
LAST.YEAR	(boolean):	last occurrence of current value of YEAR
YR_CNTR	(num):	counter for the number of years
DAY_TOT	(num):	accumulator for number of days above threshold
F_DATE	(num):	SAS date value for first day, month and year
L_DATE	(num):	SAS date value for last day, month, and year

```
MN_DAYS   (num): mean number of days above the threshold
ROW       (num): row in which to print current data value
COLUMN    (num): column in which to print current data value
LAST.STATION (boolean): last occurrence of current value of
                        STATION
```

4.113.4 Design Language Description

```
%DO N = 1 %TO &NOBS;
  %* NOBS ¢ 1 only if user wants more than one execution  ;
  %* of the same process in the same access request, or if;
  %* different thresholds are requested for each station. ;

  DATA HI_OCC&N(KEEP=STATION YEAR MONTH MONTHSUM NAME COUNTY
                    MAXTMP1-MAXTMP31 STN_FORM THRESHLD STN_REQ);

     SET TEMP;
     DO I = &N TO &N;
        SET HI_OCC POINT=I;
        /* check range of dates from only option - if annual */
        /* is requested, ENDMONTH = 13, keep entire record */

        IF THRESHLD is equal to missing THEN
           THRESHLD = &DEFAULT;
        IF ENDMONTH = 13  THEN
           OUTPUT;
        ELSE IF BEGMONTH _= MONTH _= ENDMONTH  THEN
                DO;
                  OUTPUT;
        ELSE IF BEGMONTH ¢ ENDMONTH THEN
                IF (MONTH ¢= BEGMONTH) OR (MONTH _= ENDMONTH)
                   THEN DO;
                     OUTPUT;
     END;

  RUN;

  DATA HO_SUB&N(KEEP=STATION YEAR NAME COUNTY STN_FORM
                     ABOVE FIRST_DY LAST_DY
                     FIRST_MN LAST_MN THRESHLD);
     ARRAY MAXTMP[31]  MAXTMP1-MAXTMP31;
     MISS_YR = 0;
     ABOVE = 0;
     FIRST = 0;
     MATCH = 0;

     DO UNTIL (LAST.YEAR);

        SET HI_OCC&N END=EOF;
          BY STATION YEAR;
```

```
      IF STN_REQ = missing  OR  STN_REQ = STATION  THEN
         DO;  /* search info in this record */

           MATCH = 1;
           IF MONTHSUM EQ '1.......'B THEN
              MISS_YR = 1;

           IF MISS_YR EQ 0 THEN
              DO;     /* no missing data in month or year */
                DO DAY = 1 TO 31;
                   /* search month for days over threshold */

                   IF MAXTMP[DAY] ¢ THRESHLD  THEN
                      ABOVE = ABOVE + 1;
                      LAST_DY  = DAY;
                      LAST_MN  = MONTH;
                      IF FIRST = 0  THEN
                         FIRST = 1;
                         FIRST_DY = DAY;
                         FIRST_MN = MONTH;
                END;
   END;  /* do until last.year */

   IF MISS_YR = 0  AND  MATCH = 1  THEN
      OUTPUT;  /* keep record, year had no missing data  */
               /* and record matches stations/thresholds */
               /* requested by user                      */

   IF EOF THEN STOP;
RUN;


DATA _NULL_; /* step to print out high occurrences */

  RETAIN YR_CNTR 0  DAY_TOT 0;

  SET HO_SUB&N END=EOF;
    BY STATION;
  print station info in heading;

  PUT #5 @1 'YEAR'  @12 'NUM OF DAYS TEMPERATURE'
         @34 'FIRST DAY'  @55 'LAST DAY';

  %IF not a request for metric units %THEN
     PUT #6 @15 'ABOVE  ' THRESHLD  'DEGREES F'
            @30 'ABOVE  ' THRESHLD  'DEGREES F'
            @51 'ABOVE  ' THRESHLD  'DEGREES F';
```

```
        %ELSE %IF a request for metric units %THEN
           PUT #6 @15 'ABOVE  ' THRESHLD  'DEGREES C'
                  @30 'ABOVE  ' THRESHLD  'DEGREES C'
                  @51 'ABOVE  ' THRESHLD  'DEGREES C';
        F_DATE = MDY(FIRST_MN,FIRST_DY,YEAR);
        L_DATE = MDY(LAST_MN,LAST_DY,YEAR);

        print YEAR  ABOVE  F_DATE L_DATE  under headings;

        YR_CNTR = YR_CNTR + 1;
        DAY_TOT = DAY_TOT + ABOVE;

        IF LAST.STATION  THEN
           MN_DAYS = DAY_TOT / YR_CNTR;
           print MN_DAYS  YR_CNTR; /* print mean and # of years */
           IF EOF THEN STOP;
                  ELSE
                      YR_CNTR =0;
                      DAY_TOT = 0;
                      PUT _PAGE_;
    RUN;
%END;
```

4.114 Description of Module COMPUTE LOW OCCURRENCES (C_LOOCC)

4.114.1 Processing Narrative

COMPUTE LOW OCCURRENCES makes a subset of the element data containing the requested months; if the user did not specify a threshold, the default threshold is set. The low occurrences, or number of days below the threshold, are calculated for years with complete records, and the values are then listed. The computations and listings are repeated each time the process is specified in one access request.

4.114.2 Interface Description

A) Parameters: DEFAULT - default threshold

B) Global data:

Macro variables:
- NOBS - the number of observations in a data set
- METRIC - identifies request for metric units

SAS data sets:
- TEMP (input) - with variables:
  STATION STN_FORM NAME COUNTY YEAR MONTH MONTHSUM
  MAXTMP1-MAXTMP31 MINTMP1-MINTMP31 AVEMAX AVEMIN

LO_OCC (input) - with variables:
BEGMONTH ENDMONTH RANGE THRESHLD STN_REQ

LO_OCC1-LO_OCC&N (input and output) - with variables:
STATION YEAR MONTH MONTHSUM STN_FORM NAME COUNTY
MINTMP1-MINTMP31 THRESHLD STN_REQ

LO_SUB1-LO_SUB&N (input and output) - with variables:
STATION STN_FORM NAME COUNTY YEAR BELOW
FIRST_DY LAST_DY FIRST_MN LAST_MN THRESHLD

PRINT (output) - standard SAS print file

C) No macros are called by this module.

D) No permanent files are used by this module.

E) The module generates several complete DATA steps.

4.114.3 Internal Data Description

Macro variables:
N : control variable for the %DO loop - number of times to execute the macro

SAS variables:

I	(num):	control variable for DO loop
MISS_YR	(num):	identifies missing data in one year
FIRST	(num):	flags when first day and month have been found
MATCH	(num):	flags match of stations when user specifies a different threshold for each station
EOF	(boolean):	identifies last record read from input file
DAY	(num):	control variable for DO loop
LAST.YEAR	(boolean):	last occurrence of current value of YEAR
YR_CNTR	(num):	counter for the number of years
DAY_TOT	(num):	accumulator for number of days below threshold
PER_TOT	(num):	accumulator for number of days ¢= threshold
F_DATE	(num):	SAS date value for first day, month and year
L_DATE	(num):	SAS date value for last day, month, and year
DIFFER	(num):	number of consecutive days ¢= threshold
MN_DAYS	(num):	mean number of days above the threshold
MN_PER	(num):	mean value for DIFFER for all years in period
ROW	(num):	row in which to print current data value
COLUMN	(num):	column in which to print current data value
LAST.STATION	(boolean):	last occurrence of current value of STATION

4.114.4 Design Language Description

```
%DO N = 1 %TO &NOBS;
  %* NOBS ¢ 1 only if user wants more than one execution  ;
  %* of the same process in the same access request;

  DATA LO_OCC&N(KEEP=STATION YEAR MONTH MONTHSUM NAME COUNTY
                MINTMP1-MINTMP31 STN_FORM THRESHLD STN_REQ);

     SET TEMP;
     DO I = &N TO &N;
        SET LO_OCC POINT=I;
        /* check range of dates from only option - if annual */
        /* is requested, ENDMONTH = 13, keep entire record */

        IF THRESHLD is equal to missing THEN
           THRESHLD = &DEFAULT;

        IF ENDMONTH = 13  THEN
           OUTPUT;
        ELSE IF BEGMONTH _= MONTH _= ENDMONTH  THEN
                DO;
                  OUTPUT;
        ELSE IF BEGMONTH ¢ ENDMONTH THEN
                IF (MONTH ¢= BEGMONTH) OR (MONTH _= ENDMONTH)
                   THEN DO;
                     OUTPUT;
     END;

  RUN;


  DATA LO_SUB&N(KEEP=STATION YEAR NAME COUNTY STN_FORM
                     BELOW FIRST_DY LAST_DY
                     FIRST_MN LAST_MN THRESHLD);
     ARRAY MINTMP[31]  MINTMP1-MINTMP31;
     MISS_YR = 0;
     BELOW = 0;
     FIRST = 0;
     MATCH = 0;

     DO UNTIL (LAST.YEAR);

        SET LO_OCC&N END=EOF;
          BY STATION YEAR;

        IF STN_REQ = missing  OR  STN_REQ = STATION  THEN
           DO;  /* search info in this record */

             MATCH = 1;
```

```
            IF MONTHSUM EQ '1.......'B THEN
               MISS_YR = 1;

            IF MISS_YR EQ 0 THEN
               DO;     /* no missing data in month or year */
                 DO DAY = 1 TO 31;
                    /* search month for days over threshold */

                    IF (MINTMP[DAY] _ THRESHLD) AND
                       (MINTMP[DAY] NE .)
                       THEN
                          BELOW = BELOW + 1;
                          IF MONTH _= 7  THEN
                             DO;
                                LAST_DY  = DAY;
                                LAST_MN  = MONTH;
                             END;
                          ELSE
                             DO;
                                IF FIRST = 0  THEN
                                   FIRST = 1;
                                   FIRST_DY = DAY;
                                   FIRST_MN = MONTH;
                             END;
               END;
   END;  /* do until last.year */

   IF MISS_YR = 0  AND  MATCH = 1  THEN
      OUTPUT;  /* keep record, year had no missing data  */
               /* and record matches stations/thresholds */
               /* requested by user                      */

   IF EOF THEN STOP;
RUN;

DATA _NULL_; /* step to print out low occurrences */

  RETAIN YR_CNTR 0  DAY_TOT 0  PER_TOT 0;

  SET LO_SUB&N END=EOF;
    BY STATION;
  print station info in heading;

  PUT #5 @1 'YEAR'  @12 'NUM OF DAYS TEMPERATURE'
         @34 'LAST DAY'  @55 'FIRST DAY'
         @75 'LENGTH OF PERIOD';

  %IF not a request for metric units %THEN
     PUT #6 @15 'BELOW  ' THRESHLD  'DEGREES F'
            @30 'BELOW  ' THRESHLD  'DEGREES F'
            @51 'BELOW  ' THRESHLD  'DEGREES F'
            @75 '¢=  32 DEGREES F';
```

```
        %ELSE %IF a request for metric units %THEN
           PUT #6 @15 'BELOW  ' THRESHLD  'DEGREES C'
                  @30 'BELOW  ' THRESHLD  'DEGREES C'
                  @51 'BELOW  ' THRESHLD  'DEGREES C'
                  @75 '¢=  32 DEGREES C';


        F_DATE = MDY(FIRST_MN,FIRST_DY,YEAR);
        L_DATE = MDY(LAST_MN,LAST_DY,YEAR);

        DIFFER = F_DATE - L_DATE; /* length of period equal to */
                                  /* or above threshold   */

        print YEAR  BELOW  F_DATE L_DATE  DIFFER under headings;

        YR_CNTR = YR_CNTR + 1;
        DAY_TOT = DAY_TOT + ABOVE;
        PER_TOT = PER_TOT + DIFFER;

        IF LAST.STATION  THEN
           MN_DAYS = DAY_TOT / YR_CNTR;
           MN_PER  = PER_TOT / YR_CNTR;
           print MN_DAYS  YR_CNTR  MN_PER;
                 /* print two means and # of years */
           IF EOF THEN STOP;
                  ELSE
                      YR_CNTR =0;
                      DAY_TOT = 0;
                      PER_TOT = 0;
                      PUT _PAGE_;
    RUN;
%END;
```

4.115 Description of Module DAILY OCCURRENCES (DAY_OCC)

4.115.1 Processing Narrative

DAILY OCCURRENCES calls the macros to create a subset of data for the month requested in the ONLY option, and to locate the thresholds given by the user. Default thresholds are passed as parameters, and the defaults differ depending on the value of the metric option. Macros are then called to compute and list the daily occurrences. Daily occurrences may be executed multiple times in one access request.

4.115.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:
- EL_PRCP : identifies request for precipitation data
- EL_TEMP : identifies request for temperature data
- EL_SNOW : identifies request for snowfall data
- METRIC : identifies request for metric units

No SAS data sets are created or used.

C) The macros called by this module are:

GET MONTHS AND MULTIPLE THRESHOLDS (4.117 - GET_MMT)
CHECK NUMBER OF OBSERVATIONS (4.20 - NUM_OBS)
COMPUTE DAILY OCCURRENCES (4.118 - C_DOCC)
COMPUTE SNOW OCCURRENCES (4.119 - C_SOCC)

D) No permanent files are used by this module.

E) The module generates macro statements only.

4.115.3 Internal Data Description

No local data are used in this module.

4.115.4 Design Language Description

```
%GET MONTHS AND MULTIPLE THRESHOLDS
            (DY_OCC,DAILY OCCURRENCES,OCCURRENCES)

%CHECK NUMBER OF OBSERVATIONS (DY_OCC)

%IF request for precipitation data %THEN
   %IF not a request for metric units %THEN
       %COMPUTE DAILY OCCURRENCES(PRCP,PRECIP,PRECIPITATION,
                                  INCHES,1)
   %ELSE %IF a request for metric units %THEN
       %COMPUTE DAILY OCCURRENCES(PRCP,PRECIP,PRECIPITATION,
                                  MM,25.4)

%IF request for temperature data %THEN
   %IF not a request for metric units %THEN
       %COMPUTE DAILY OCCURRENCES(TEMP,MAXTMP,MAXIMUM TEMPERATURE,
                                  DEG F,80)
   %ELSE %IF a request for metric units %THEN
       %COMPUTE DAILY OCCURRENCES(TEMP,MAXTMP,MAXIMUM TEMPERATURE,
                                  DEG C,26)
```

```
%IF request for snowfall data %THEN
   %IF not a request for metric units %THEN
       %COMPUTE SNOW OCCURRENCES(INCHES,1)
   %ELSE %IF a request for metric units %THEN
      %COMPUTE SNOW OCCURRENCES(CM,2.54)
```

4.116 Description of Module MONTHLY OCCURRENCES (MON_OCC)

4.116.1 Processing Narrative

MONTHLY OCCURRENCES calls the macros to create a subset of data for the months requested in the ONLY option, and to locate the thresholds given by the user. Default thresholds are passed as parameters, and the defaults differ depending on the value of the metric option. Macros are then called to compute and list the monthly occurrences. Monthly occurrences may be executed multiple times in one access request.

4.116.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:

- EL_PRCP : identifies request for precipitation data
- EL_SNOW : identifies request for snowfall data
- METRIC : identifies request for metric units

No SAS data sets are created or used.

C) The macros called by this module are:

GET MONTHS AND MULTIPLE THRESHOLDS (4.117 - GET_MMT)
CHECK NUMBER OF OBSERVATIONS (4.20 - NUM_OBS)
COMPUTE MONTHLY OCCURRENCES (4.120 - C_MOCC)

D) No permanent files are used by this module.

E) The module generates macro statements only.

4.116.3 Internal Data Description

No local data are used in this module.

4.116.4 Design Language Description

```
%GET MONTHS AND MULTIPLE THRESHOLDS
            (MN_OCC,MONTHLY OCCURRENCES,MONTHLY OCCURENCES)

%CHECK NUMBER OF OBSERVATIONS (MN_OCC)

%IF request for precipitation data %THEN
   %IF not a request for metric units %THEN
       %COMPUTE MONTHLY OCCURRENCES(PRCP,PRCP_TOT,PRECIPITATION,
                                    INCHES,5)
   %ELSE %IF a request for metric units %THEN
       %COMPUTE MONTHLY OCCURRENCES(PRCP,PRCP_TOT,PRECIPITATION,
                                    MM,127)

%IF request for snowfall data %THEN
   %IF not a request for metric units %THEN
       %COMPUTE MONTHLY OCCURRENCES(SNOW,SNOW_TOT,SNOWFALL,
                                    INCHES,10)
   %ELSE %IF a request for metric units %THEN
       %COMPUTE MONTHLY OCCURRENCES(SNOW,SNOW_TOT,SNOWFALL,
                                    CM,25.4)
```

4.117 Description of Module GET MONTHS AND MULTIPLE THRESHOLDS (GET_MMT)

4.117.1 Processing Narrative

GET MONTHS AND MULTIPLE THRESHOLDS parses the options given by the user with the DAILY or MONTHLY OCCURRENCES processes, and finds the requested months and thresholds. The user may give a different set of thresholds for each station requested, simply by explicitly coding the station number before the keyword THRESHOLD. If any of the options are not recognized, the default options will be used.

4.117.2 Interface Description

A) Parameters: DSN - data set in which to store the options
NAME1 - name of the process
NAME2 - a second name by which to recognize the process

B) Global data:

No global macro variables are used in this module.

SAS data sets:

PROCESS (input) - with variables:
PRC_NAME PARMETER

&DSN (output) - with variables:
BEGMONTH ENDMONTH RANGE COUNT THRES1-THRES10 STN_REQ

C) No macros are called by this module.

D) No permanent files are used by this module.

E) The module generates one complete DATA step.

4.117.3 Internal Data Description

No local macro variables are used in this module.

SAS variables:

MON	(char):	array containing month names
WORD	(char):	array with words scanned from process parameter
PREVIOUS	(num):	identifies when to write an observation
EOF	(boolean):	identifies the last record read from input file
POS	(num):	position of keyword THRESHOLD in parameter
HOWMANY	(num):	counts how many thresholds have been found
NEXT	(num):	number of next word to be scanned in parameter

4.117.4 Design Language Description

```
DATA &DSN (KEEP=BEGMONTH ENDMONTH RANGE COUNT THRES1-THRES10
           STN_REQ);
   /* data step to identify the months specified  */
   /* in the ONLY option and up to ten thresholds */
   LENGTH MON1-MON13 $ 10  WORD1-WORD4  $ 10  RANGE $ 18
          STN_REQ $ 8;
   RETAIN  MON1  'JANUARY'  MON2  'FEBRUARY'  MON3  'MARCH'
           MON4  'APRIL'    MON5  'MAY'       MON6  'JUNE'
           MON7  'JULY'     MON8  'AUGUST'    MON9  'SEPTEMBER'
           MON10 'OCTOBER'  MON11 'NOVEMBER'  MON12 'DECEMBER'
           MON13 'ANNUAL'   PREVIOUS 0;

   ARRAY MON[13] MON1-MON13;
   ARRAY THRES[10] THRES1-THRES10;

   /* set default values */
   initialize THRES array to missing;
   BEGMONTH = 1;
   ENDMONTH = 13;
   RANGE = 'JANUARY-ANNUAL';
   STN_REQ = . ;
   PREVIOUS = 0;
   COUNT = 1;

   DO UNTIL(PRC_NAME = 'PROCESS');
      SET PROCESS END=EOF;
```

```
IF PRC_NAME = "&NAME1"  OR PRC_NAME = "&NAME2"  THEN
   DO; /* get information on this process */

  IF PARMETER is equal to missing
     THEN OUTPUT;    /* output default values */

  ELSE IF PARMETER is not equal to missing THEN DO;
        WORD1 = SCAN(PARMETER,1,' ');
        IF WORD1 = 'ONLY'  THEN
           DO;
             PREVIOUS = 1;
             WORD2 = SCAN(PARMETER,2,' ');
             WORD3 = SCAN(PARMETER,3,' ');
             IF WORD3 = 'TO' THEN
                WORD4 = SCAN(PARMETER,4,' ');
                RANGE = TRIM(WORD2) !! '-' !! WORD4;
             ELSE
                WORD4 = ' ';
                RANGE = WORD2;

             DO I = 1 TO 13;
                IF WORD2 = MON[I]  THEN
                   BEGMONTH = I;
             END;

             IF WORD4 NE ' '  THEN
                DO I = 1 TO 13;
                   IF WORD4 = MON[I]  THEN
                      ENDMONTH = I;
                END;
             ELSE ENDMONTH = BEGMONTH;
           END;  /* if word = 'ONLY'  */

        ELSE IF WORD1 = 'THRESHOLD'  THEN
           DO;
             PREVIOUS = 1;
             WORD2 = SCAN(PARMETER,2,' ');
             NEXT = 3;
             HOWMANY = 1;
             DO WHILE(WORD2 is not blank);
                IF HOWMANY >= 10  THEN
                   COUNT = HOWMANY;
                   THRES[HOWMANY] = WORD2;
                WORD2 = SCAN(PARMETER,NEXT,' ');
                NEXT = NEXT + 1;
                HOWMANY = HOWMANY + 1;
             END;
           END;  /* if word1 = 'THRESHOLD' */
        ELSE
           DO;
```

```
                    POS = INDEX(PARMETER,'THRESHOLD')
                    IF POS NE 0  THEN
                       DO; /* user requested different    */
                           /* thresholds for each station */
                         STN_REQ = SCAN(PARMETER,1,' ');
                         WORD2 = SCAN(PARMETER,3,' ');
                         NEXT = 4;
                         HOWMANY = 1;
                         DO WHILE(WORD2 is not blank);
                            IF HOWMANY _= 10  THEN
                               COUNT = HOWMANY;
                               THRES[HOWMANY] = WORD2;
                            WORD2 = SCAN(PARMETER,NEXT,' ');
                            NEXT = NEXT + 1;
                            HOWMANY = HOWMANY + 1;
                         END;
                         OUTPUT; /* save each station as */
                       END;      /* separate observation */
                    ELSE /* illegal option - set to default */
                       PREVIOUS = 1;
                       PARMETER = .;

     IF EOF THEN
        IF PREVIOUS NE 0 THEN OUTPUT;
        STOP;
     END;  /* until process name = process */
     IF PREVIOUS NE 0 THEN  OUTPUT;
RUN;
```

4.118 Description of Module COMPUTE DAILY OCCURRENCES (C_DOCC)

4.118.1 Processing Narrative

COMPUTE DAILY OCCURRENCES makes a subset of the element data containing the requested months; if necessary, default thresholds are set. The daily occurrences, or number of days above each threshold, are calculated for years with complete records, and the values are then listed. The computations and listings are repeated each time the process is specified in one access request.

4.118.2 Interface Description

A) Parameters:
- DSN - data set name containing element data
- ARRRAY - array name of 31 daily values
- E_TITLE - full name of element for heading
- UNITS - units of the daily values
- DEFAULT - single default threshold

B) Global data:

Macro variables:
NOBS - the number of observations in a data set

SAS data sets:
DSN (input) - with variables:
STATION STN_FORM NAME COUNTY YEAR MONTH MONTHSUM
&ARRRAY.1-&ARRRAY.31

DY_OCC (input) - with variables:
BEGMONTH ENDMONTH RANGE COUNT THRES1-THRES10 STN_REQ

DY_OCC1-DY_OCC&N (input and output) - with variables:
STATION YEAR MONTH MONTHSUM STN_FORM NAME COUNTY
&ARRRAY.1-&ARRRAY.31 THRES1-THRES10 COUNT STN_REQ

DY_SUB1-DY_SUB&N (input and output) - with variables:
STATION STN_FORM NAME COUNTY YEAR MONTH ABOVE1-ABOVE10
THRES1-THRES10 COUNT

PRINT (output) - standard SAS print file

C) No macros are called by this module.

D) No permanent files are used by this module.

E) The module generates several complete DATA steps.

4.118.3 Internal Data Description

Macro variables:
N : control variable for the %DO loop - number of times to execute the macro

SAS variables:

I	(num):	control variable for DO loop
MISS_YR	(num):	identifies missing data in one year
MATCH	(num):	flags match of stations when user specifies different thresholds for each station
EOF	(boolean):	identifies last record read from input file
T	(num):	control variable for DO loop (thresholds)
DAY	(num):	control variable for DO loop (days)
YR_CNTR	(num):	counter for the number of years
TOTAL	(num):	array to accumulate number of days ¢ threshold
AVE	(num):	array of mean number of days ¢ threshold
START	(num):	column in which to begin listing one year's data
ROW	(num):	row in which to print current data value
COLUMN	(num):	column in which to print current data value
LAST.YEAR	(boolean):	last occurrence of current value of YEAR
FIRST.STATION	(boolean):	first occurrence of current value of STATION
LAST.STATION	(boolean):	last occurrence of current value of STATION

4.118.4 Design Language Description

```
%DO N = 1 %TO &NOBS;
  %* NOBS ¢ 1 only if user wants more than one execution  ;
  %* of the same process in the same access request, or if;
  %* different thresholds are requested for each station. ;

  DATA DY_OCC&N(KEEP=STATION YEAR MONTH MONTHSUM NAME COUNTY
                    &ARRRAY.1-&ARRRAY.31 STN_FORM STN_REQ
                    COUNT THRES1-THRES10);

     ARRAY THRES[10] THRES1-THRES10;
     SET &DSN;
     DO I = &N TO &N;
        SET DY_OCC POINT=I;

        IF THRES[1]  is equal to missing THEN
           THRES[1] = &DEFAULT;
        /* check range of dates from only option - if annual */
        /* is requested, ENDMONTH = 13, keep entire record */
        IF ENDMONTH = 13  THEN
           OUTPUT;
        ELSE IF BEGMONTH _= MONTH _= ENDMONTH  THEN
                DO;
                  OUTPUT;
        ELSE IF BEGMONTH ¢ ENDMONTH THEN
                IF (MONTH ¢= BEGMONTH) OR (MONTH _= ENDMONTH)
                   THEN DO;
                     OUTPUT;
     END;

  RUN;

  DATA DY_SUB&N(KEEP=STATION YEAR MONTH NAME COUNTY STN_FORM
                    ABOVE1-ABOVE10 THRES1-THRES10  COUNT);
     ARRAY &ARRRAY.[31]  &ARRRAY.1-&ARRRAY.31;
     ARRAY THRES[10]  THRES1-THRES10;
     ARRAY ABOVE[10]  ABOVE1-ABOVE10;

     initialize ABOVE array to 0;
     MISS_YR = 0;
     MATCH = 0;

     DO UNTIL (LAST.YEAR);

        SET DY_OCC&N END=EOF;
          BY STATION YEAR;
```

```
      IF STN_REQ = missing  OR  STN_REQ = STATION  THEN
         DO;  /* search info in this record */

           MATCH = 1;
           IF MONTHSUM EQ '1.......'B THEN
              MISS_YR = 1;

           IF MISS_YR EQ 0 THEN
              DO;     /* no missing data in month or year */
                DO T = 1 TO COUNT;
                /* search month for days over thresholds */

                   DO DAY = 1 TO 31;
                      IF &ARRRAY.[DAY] ¢ &THRES[T]  THEN
                         ABOVE[T] = ABOVE[T] + 1;
                   END;
                END; /* from 1 to number of thresholds */
   END;  /* do until last.year */

   IF MISS_YR = 0  AND  MATCH = 1  THEN
      OUTPUT;  /* keep record, year had no missing data  */
               /* and record matches stations/thresholds */
               /* requested by user                      */

   IF EOF THEN STOP;
RUN;


DATA _NULL_; /* step to print out daily occurrences */

  RETAIN YR_CNTR 0  TOTAL1-TOTAL10;

  ARRAY ABOVE[10] ABOVE1-ABOVE10;
  ARRAY THRES[10] THRES1-THRES10;
  ARRAY AVE[10] AVE1-AVE10;

  SET DY_SUB&N END=EOF;
    BY STATION;

  IF FIRST.STATION THEN

     print station info in heading;
     initialize TOTAL array to 0;
     ROW = 8;

     IF COUNT _ 3  THEN
        START = 50;
     ELSE IF COUNT _ 5 THEN
        START = 40;
```

```
            ELSE IF COUNT _ 7 THEN
               START = 30;
            ELSE IF COUNT _ 9 THEN
               START = 20;
            ELSE
               START = 10;

            PUT #5 @35 "NUMBER OF DAYS &E_TITLE ABOVE " ;
            PUT #6 @1 'YEAR' ;
            COLUMN = START;

            DO T = 1 TO COUNT;
               PUT #6 @COLUMN  THRES[T] " &UNITS";
               COLUMN = COLUMN + 10;
            END;

         END;  /* first.station */

         COLUMN = START;
         print #ROW @1 YEAR;
         DO T = 1 TO COUNT;
            PUT #ROW @COLUMN ABOVE[T] 10.;
            COLUMN = COLUMN + 10;
            TOTAL[T] = TOTAL[T] + ABOVE[T];
         END;
         YR_CNTR = YR_CNTR + 1;

         IF LAST.STATION  THEN
            COLUMN = START;
            ROW = ROW + 3;
            PUT #ROW @1 'MEAN' ;
            DO T = 1 TO COUNT;
               AVE[T] = TOTAL[T] / YR_CNTR;
               PUT #ROW @COLUMN AVE[T] 10.2;
               COLUMN = COLUMN + 10;
            END;
            print YR_CNTR; ; /* print mean and # of years */
            IF EOF THEN STOP;
                  ELSE
                      YR_CNTR =0;
                      PUT _PAGE_;
    RUN;
%END;
```

4.119 Description of Module COMPUTE SNOW OCCURRENCES (C_SOCC)

4.119.1 Processing Narrative

COMPUTE SNOW OCCURRENCES makes a subset of the snowfall data containing the requested months; if necessary, a default threshold is set. The snow occurrences, or number of days above a threshold, are calculated for the snowfall season, September 1 to August 31, and the values are then listed. The computations and listings are repeated each time the process is specified in one access request.

4.119.2 Interface Description

A) Parameters: UNITS - units for the daily values
DEFAULT - default threshold

B) Global data:
Macro variables:
NOBS - the number of observations in a data set

SAS data sets:
SNOW (input) - with variables:
STATION STN_FORM NAME COUNTY YEAR MONTH MONTHSUM
SNOW1-SNOW31 DEPTH1-DEPTH31 SNOW_TOT

DY_OCC (input) - with variables:
BEGMONTH ENDMONTH RANGE COUNT THRES1-THRES10 STN_REQ

SN_OCC1-SN_OCC&N (input and output) - with variables:
STATION YEAR MONTH MONTHSUM STN_FORM NAME COUNTY
SNOW1-SNOW31 DEPTH1-DEPTH31 D_CODE1-D_CODE31 THRESHLD
STN_REQ SNOW_YR

SN_SUB1-SN_SUB&N (input and output) - with variables:
STATION STN_FORM NAME COUNTY YEAR MONTH ABOVE THRESHLD
FIRST_DY LAST_DY FIRST_MN LAST_MN DAYONE1-DAYONE4 SNOW_YR

PRINT (output) - standard SAS print file

C) No macros are called by this module.

D) No permanent files are used by this module.

E) The module generates several complete DATA steps.

4.119.3 Internal Data Description

No local data are used in this module.

Macro variables:
N : control variable for the %DO loop - number of times to execute the macro

```
SAS variables:
   I          (num): control variable for DO loop
   LEGAL      (num): identifies legal month requested
   MISS_YR    (num): identifies missing data in one year
   FIRST      (num): flags when first day and month have been found
   MATCH      (num): flags match of stations when user specifies a
                     different threshold for each station
   EOF   (boolean): identifies last record read from input file
   DAY        (num): control variable for DO loop
   LAST.SNOW_YR(boolean): last occurrence of current value of
                          SNOW_YR
   YR_CNTR    (num): counter for the number of years
   DAY_TOT    (num): accumulator for number of days above threshold
   TOTAL      (num): array for accumulating snow depths on first day
                     of months Dec-Mar
   T_CNTR     (num): array used to count the number of months with
                     valid depth data on first day of months 12-3
   AVE        (num): array with mean snow depths on first day of
                     months Dec-Mar
   SEASON     (num): beginning year of snow season
   F_DATE     (num): SAS date value for first day, month and year
   L_DATE     (num): SAS date value for last day, month, and year
   MN_DAYS    (num): mean number of days above the threshold
   ROW        (num): row in which to print current data value
   COLUMN     (num): column in which to print current data value
   LAST.STATION (boolean): last occurrence of current value of
                           STATION
```

4.119.4 Design Language Description

```
%DO N = 1 %TO &NOBS;
  %* NOBS ¢ 1 only if user wants more than one execution  ;
  %* of the same process in the same access request, or if;
  %* different thresholds are requested for each station. ;

  DATA SN_OCC&N(KEEP=STATION YEAR MONTH MONTHSUM NAME COUNTY
                     SNOW1-SNOW31 DEPTH1-DEPTH31 THRESHLD
                     D_CODE1-D_CODE31 SNOW_YR STN_FORM STN_REQ);

     ARRAY THRES[10] THRES1-THRES10;
     SET SNOW;
     DO I = &N TO &N;
        SET DY_OCC POINT=I;
        /* check range of dates from only option - if annual */
        /* is requested, ENDMONTH = 13, keep entire record */
        LEGAL = 0;
        IF ENDMONTH = 13  THEN
           LEGAL = 1;
        ELSE IF BEGMONTH _= MONTH _= ENDMONTH  THEN
                  LEGAL = 1;
```

```
      ELSE IF BEGMONTH ¢ ENDMONTH THEN
              IF (MONTH ¢= BEGMONTH) OR (MONTH _= ENDMONTH)
                 THEN   LEGAL = 1;

      IF LEGAL = 1 THEN
         DO;   /* keep this record */
           IF THRES[1] is equal to missing THEN
              THRESHLD = &DEFAULT;
           ELSE
              THRESHLD = THRES[1];
           IF MONTH ¢= 9 THEN
              SNOW_YR = YEAR + 1;
           ELSE
              SNOW_YR = YEAR;
           OUTPUT;
           END;
   END;
RUN;

DATA SN_SUB&N(KEEP=STATION YEAR MONTH NAME COUNTY STN_FORM
                   ABOVE FIRST_DY LAST_DY FIRST_MN LAST_MN
                   DAYONE1-DAYONE4 SNOW_YR THRESHLD);

   ARRAY SNOW[31]  SNOW1-SNOW31;
   ARRAY DEPTH[31] DEPTH1-DEPTH31;
   ARRAY DAYONE[4] DAYONE1-DAYONE4;
   ABOVE = 0;
   FIRST = 0;
   MATCH = 0;

   DO UNTIL (LAST.SNOW_YR);

      SET SN_OCC&N END=EOF;
        BY STATION SNOW_YR;

      IF STN_REQ = missing  OR  STN_REQ = STATION  THEN
         DO;  /* search info in this record */

           MATCH = 1;
           IF MONTH = 12 THEN
              DAYONE1 = DEPTH[1];
           ELSE IF MONTH = 1 THEN
              DAYONE2 = DEPTH[1];
           ELSE IF MONTH = 2 THEN
              DAYONE3 = DEPTH[1];
           ELSE IF MONTH = 3 THEN
              DAYONE4 = DEPTH[1];
```

```
            DO DAY = 1 TO 31;
               /* search month for days over threshold */
               IF SNOW[DAY] ¢ THRESHLD  THEN
                  ABOVE = ABOVE + 1;
                  LAST_DY  = DAY;
                  LAST_MN  = MONTH;

                  IF FIRST = 0  THEN
                     FIRST = 1;
                     FIRST_DY = DAY;
                     FIRST_MN = MONTH;
             END;
   END;  /* do until last.snow_yr */

   IF MATCH = 1  THEN
      OUTPUT;  /* keep record, matches stations/thresholds */
               /* requested by user                        */
   IF EOF THEN STOP;
RUN;

DATA _NULL_; /* step to print out snow occurrences */

  RETAIN YR_CNTR 0 DAY_TOT 0 TOTAL1-TOTAL4
         T_CNTR1-T_CNTR4 ROW;
  declare TOTAL, T_CNTR, and AVE arrays;

  SET SN_SUB&N END=EOF;
    BY STATION;
  IF FIRST.STATION THEN
     DO; ROW = 8;
       initialize TOTAL and T_CNTR arrays to 0;
       print station info in heading;

       PUT #5 @1 'SNOWFALL'
              @12 'NUM OF DAYS SNOWFALL'
              @34 'FIRST SNOWFALL'  @55 'LAST SNOWFALL'
              @75 'DEPTH OF SNOW ON FIRST DAY OF:';

       PUT #6 @2  'SEASON'
              @16 '¢= '  THRESHLD 4.1 " &UNITS"
              @35 '¢= '  THRESHLD 4.1 " &UNITS"
              @55 '¢= '  THRESHLD 4.1 " &UNITS"
              @76 'DEC'  @86 'JAN'  @96 'FEB' @106 'MAR';
     END;

  F_DATE = MDY(FIRST_MN,FIRST_DY,YEAR);
  L_DATE = MDY(LAST_MN,LAST_DY,YEAR);
  SEASON = SNOW_YR - 1;

  ROW = ROW + 1;
  print SEASON/SNOW_YR  ABOVE  F_DATE L_DATE;
  DO D = 1 TO 4;
```

```
            IF DAYONE[D] NE missing THEN
               PUT DAYONE[D];
               TOTAL[D] = TOTAL[D] + DAYONE[D];
               T_CNTR[D] = T_CNTR[D] + 1;
            ELSE
               PUT 'M';
         END;

         YR_CNTR = YR_CNTR + 1;
         DAY_TOT = DAY_TOT + ABOVE;

         IF LAST.STATION  THEN
            MN_DAYS = YR_CNTR / DAY_TOT;
            PUT YR_CNTR, MN_DAYS;
            DO D = 1 TO 4;
               IF T_CNTR[D] NE 0  THEN
                  AVE[D] = TOTAL[D] / T_CNTR[D];
               ELSE
                  AVE[D] = 0;
               PUT AVE[D];
            END;

            IF EOF THEN STOP;
                  ELSE
                      YR_CNTR =0;
                      DAY_TOT = 0;
                      PUT _PAGE_;
    RUN;
%END;
```

4.120 Description of Module COMPUTE MONTHLY OCCURRENCES (C_MOCC)

4.120.1 Processing Narrative

COMPUTE MONTHLY OCCURRENCES makes a subset of the element data containing the requested months; if the user did not specify any thresholds, default thresholds are set. The monthly occurrences, or number of days when the monthly total was above each threshold, are calculated for years with complete records, and the values are then listed. The computations and listings are repeated each time the process is specified in one access request.

4.120.2 Interface Description

A) Parameters: DSN - data set name containing element data
MON_VAL - variable name of monthly total
E_TITLE - full name of element for heading
UNITS - units of the daily values
DEFAULT - single default threshold

B) Global data:
Macro variables:
NOBS - the number of observations in a data set

SAS data sets:
DSN (input) - with variables:
STATION STN_FORM NAME COUNTY YEAR MONTH MONTHSUM &MON_VAL

MN_OCC (input) - with variables:
BEGMONTH ENDMONTH RANGE COUNT THRES1-THRES10 STN_REQ

MN_OCC1-MN_OCC&N (input and output) - with variables:
STATION YEAR MONTH MONTHSUM STN_FORM NAME COUNTY
&MON_VAL THRES1-THRES10 COUNT STN_REQ

MN_SUB1-MN_SUB&N (input and output) - with variables:
STATION STN_FORM NAME COUNTY YEAR MONTH ABOVE1-ABOVE10
THRES1-THRES10 COUNT

PRINT (output) - standard SAS print file

C) No macros are called by this module.

D) No permanent files are used by this module.

E) The module generates several complete DATA steps.

4.120.3 Internal Data Description

```
Macro variables:
   N : control variable for the %DO loop - number of times to
       execute the macro

SAS variables:
   I          (num): control variable for DO loop
   MISS_YR    (num): identifies missing data in one year
   MATCH      (num): flags match of stations when user specifies
                     different thresholds for each station
   EOF   (boolean): identifies last record read from input file
   T          (num): control variable for DO loop (thresholds)
   YR_CNTR    (num): counter for the number of years
   TOTAL      (num): array to accumulate number of days ¢ threshold
   AVE        (num): array of mean number of days ¢ threshold
   START      (num): column in which to begin listing one year's data
   ROW        (num): row in which to print current data value
   COLUMN     (num): column in which to print current data value
   LAST.YEAR (boolean): last occurrence of current value of YEAR
   FIRST.STATION (boolean): first occurrence of current value of
                            STATION
   LAST.STATION  (boolean): last occurrence of current value of
                            STATION
```

4.120.4 Design Language Description

```
%DO N = 1 %TO &NOBS;
   %* NOBS ¢ 1 only if user wants more than one execution  ;
   %* of the same process in the same access request, or if;
   %* different thresholds are requested for each station. ;

   DATA MN_OCC&N(KEEP=STATION YEAR MONTH MONTHSUM NAME COUNTY
                      &MON_VAL STN_FORM STN_REQ
                      COUNT THRES1-THRES10);

      ARRAY THRES[10] THRES1-THRES10;
      SET &DSN;
      DO I = &N TO &N;
         SET MN_OCC POINT=I;

         IF THRES[1]  is equal to missing THEN
            THRES[1] = &DEFAULT;
         /* check range of dates from only option - if annual */
         /* is requested, ENDMONTH = 13, keep entire record */
         IF ENDMONTH = 13  THEN
            OUTPUT;
         ELSE IF BEGMONTH _= MONTH _= ENDMONTH  THEN
                 DO;
                   OUTPUT;
```

```
      ELSE IF BEGMONTH ¢ ENDMONTH THEN
              IF (MONTH ¢= BEGMONTH) OR (MONTH _= ENDMONTH)
                 THEN DO;
                   OUTPUT;
   END;
RUN;

DATA MN_SUB&N(KEEP=STATION YEAR MONTH NAME COUNTY STN_FORM
                   ABOVE1-ABOVE10 THRES1-THRES10  COUNT);
   ARRAY THRES[10]  THRES1-THRES10;
   ARRAY ABOVE[10]  ABOVE1-ABOVE10;

   initialize ABOVE array to 0;
   MISS_YR = 0;
   MATCH = 0;

   DO UNTIL (LAST.YEAR);

      SET MN_OCC&N END=EOF;
        BY STATION YEAR;

      IF STN_REQ = missing  OR  STN_REQ = STATION  THEN
         DO;  /* search info in this record */
             MATCH = 1;
           IF MONTHSUM EQ '1.......'B THEN
              MISS_YR = 1;

           IF MISS_YR EQ 0 THEN
              DO;     /* no missing data in month or year */
                DO T = 1 TO COUNT;
                /* search month for days over thresholds */

                IF &MON_VAL ¢ &THRES[T]  THEN
                   ABOVE[T] = ABOVE[T] + 1;
                END; /* from 1 to number of thresholds */
   END;  /* do until last.year */

   IF MISS_YR = 0  AND  MATCH = 1  THEN
      OUTPUT;  /* keep record, year had no missing data  */
               /* and record matches stations/thresholds */
               /* requested by user                      */

   IF EOF THEN STOP;
RUN;

DATA _NULL_; /* step to print out monthly occurrences */

  RETAIN YR_CNTR 0  TOTAL1-TOTAL10;

  ARRAY ABOVE[10] ABOVE1-ABOVE10;
  ARRAY THRES[10] THRES1-THRES10;
  ARRAY AVE[10] AVE1-AVE10;
```

```
SET MN_SUB&N END=EOF;
  BY STATION;

IF FIRST.STATION THEN

   print station info in heading;
   initialize TOTAL array to 0;
   ROW = 8;

   IF COUNT _ 3  THEN
      START = 50;
   ELSE IF COUNT _ 5 THEN
      START = 40;
   ELSE IF COUNT _ 7 THEN
      START = 30;
   ELSE IF COUNT _ 9 THEN
      START = 20;
   ELSE
      START = 10;

   PUT #5 @35 "NUMBER OF MONTHS &E_TITLE ABOVE " ;
   PUT #6 @1 'YEAR' ;
   COLUMN = START;

   DO T = 1 TO COUNT;
      PUT #6 @COLUMN  THRES[T] " &UNITS";
      COLUMN = COLUMN + 10;
   END;
END;  /* first.station */

COLUMN = START;
print #ROW @1 YEAR;
DO T = 1 TO COUNT;
   PUT #ROW @COLUMN ABOVE[T] 10.;
   COLUMN = COLUMN + 10;
   TOTAL[T] = TOTAL[T] + ABOVE[T];
END;
YR_CNTR = YR_CNTR + 1;

IF LAST.STATION  THEN
   COLUMN = START;
   ROW = ROW + 3;
   PUT #ROW @1 'MEAN' ;
   DO T = 1 TO COUNT;
      AVE[T] = TOTAL[T] / YR_CNTR;
      PUT #ROW @COLUMN AVE[T] 10.2;
      COLUMN = COLUMN + 10;
   END;
   print YR_CNTR; ; /* print mean and # of years */
   IF EOF THEN STOP;
         ELSE
             YR_CNTR =0;
             PUT _PAGE_;
RUN;
```

```
%END;
```

4.121 Description of Module FLOW DURATION TABLE (FLO_DUR)

4.121.1 Processing Narrative

FLOW DURATION TABLE calls the macros to determine the classes specified by the user, and compute the flow durations, i.e., the number of days in which the daily value was within certain class intervals. Multiple executions of the flow duration table process are allowed in one access request.

4.121.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:
METRIC : identifies request for metric units

No SAS data sets are created or used.

C) The macros called by this module are:

GET CLASS INTERVALS (4.122 - GET_CLS)
CHECK NUMBER OF OBSERVATIONS (4.20 - NUM_OBS)
COMPUTE FLOW DURATION TABLE (4.123 - C_FLOW)

D) No permanent files are used by this module.

E) The module generates macro statements only.

4.121.3 Internal Data Description

No local data are used in this module.

4.121.4 Design Language Description

```
%GET CLASS INTERVALS(CLASS,FLOW DURATION TABLE,FLOW DURATION)
%CHECK NUMBER OF OBSERVATIONS (CLASS)

%IF not a request for metric units %THEN
   %COMPUTE FLOW DURATION TABLE (ACRE-FEET,CFS,SQ MI,1)

%ELSE
   %COMPUTE FLOW DURATION TABLE (CMS-DAYS,CMS,SQ KM,0.02832)
```

4.122 Description of Module GET CLASS INTERVALS (GET_CLS)

4.122.1 Processing Narrative

GET CLASS INTERVALS parses the options given by the user with the FLOW DURATION TABLE process, and finds the requested class intervals. The user may give a different set of class intervals for each station requested, simply by explicitly coding the station number before the keyword CLASS. If any of the options are not recognized, the default options will be used.

4.122.2 Interface Description

A) Parameters: DSN - data set in which to store the options
NAME1 - name of the process
NAME2 - a second name by which to recognize the process

B) Global data:

No global macro variables are used in this module.

SAS data sets:
PROCESS (input) - with variables:
PRC_NAME PARMETER

&DSN (output) - with variables:
COUNT CLASS1-CLASS30 STN_REQ

C) No macros are called by this module.

D) No permanent files are used by this module.

E) The module generates one complete DATA step.

4.122.3 Internal Data Description

No local macro variables are used in this module.

SAS variables:

WORD	(char):	array with words scanned from process parameter
PREVIOUS	(num):	identifies when to write an observation
EOF	(boolean):	identifies the last record read from input file
POS	(num):	position of keyword CLASS in parameter
HOWMANY	(num):	counts how many thresholds have been found
NEXT	(num):	number of next word to be scanned in parameter
NEW_STAT	(num):	flags a new user-specified station number

4.122.4 Design Language Description

```
DATA &DSN (KEEP= COUNT CLASS1-CLASS30  STN_REQ);
   /* data step to identify the classes specified  */

   LENGTH WORD1-WORD4 $ 30 STN_REQ $ 8;
   RETAIN  PREVIOUS 0;

   ARRAY CLASS[30] CLASS1-CLASS30;
   /* set default values */

   initialize CLASS array to missing;
   STN_REQ = . ;
   PREVIOUS = 0;
   COUNT = 1;
   NEW_STAT = 0;

   DO UNTIL(PRC_NAME = 'PROCESS');
      SET PROCESS END=EOF;

      IF PRC_NAME = "&NAME1"  OR PRC_NAME = "&NAME2"  THEN
         DO; /* get information on this process */

        IF PARMETER is equal to missing
           THEN    /* output default values */
               COUNT = 7;
               OUTPUT;

        ELSE IF PARMETER is not equal to missing THEN DO;

               WORD1 = SCAN(PARMETER,1,' ');
               IF WORD1 = 'CLASS'  THEN
                  DO;
                    PREVIOUS = 1;
                    WORD2 = SCAN(PARMETER,2,' ');
                    NEXT = 3;
                    HOWMANY = COUNT;

                    DO WHILE(WORD2 is not blank);
                       IF HOWMANY _= 30  THEN
                          COUNT = HOWMANY;
                          CLASS[HOWMANY] = WORD2;
                       WORD2 = SCAN(PARMETER,NEXT,' ');
                       NEXT = NEXT + 1;
                       HOWMANY = HOWMANY + 1;
                    END;

                  END;
```

```
                ELSE
                   DO;  /* check for CLASS option at all */
                     POS = INDEX(PARMETER,'class')
                     IF POS NE 0  THEN
                        DO; /* user requested different    */
                            /* classes for each station    */
                          IF NEW_STAT = 0 THEN
                             NEW_STAT = 1;
                          ELSE
                             OUTPUT; /* save previous station   */
                                     /* as separate observation */
                             initialize CLASS array to missing
                               and COUNT to 1;

                          PREVIOUS = 1;
                          STN_REQ = SCAN(PARMETER,1,' ');
                          WORD2 = SCAN(PARMETER,3,' ');
                          NEXT = 4;
                          HOWMANY = COUNT;
                          DO WHILE(WORD2 is not blank);
                             IF HOWMANY _= 30  THEN
                                COUNT = HOWMANY;
                                CLASS[HOWMANY] = WORD2;
                             WORD2 = SCAN(PARMETER,NEXT,' ');
                             NEXT = NEXT + 1;
                             HOWMANY = HOWMANY + 1;
                          END;
                        END;
                     ELSE /* illegal option - set to default */
                        PREVIOUS = 1;
                        PARMETER = .;

          IF EOF THEN
             IF PREVIOUS NE 0 THEN OUTPUT;
             STOP;
          END;  /* until process name = process */
          IF PREVIOUS NE 0 THEN  OUTPUT;

           RUN;
```

4.123 Description of Module COMPUTE FLOW DURATION TABLE (C_FLOW)

4.123.1 Processing Narrative

COMPUTE FLOW DURATION TABLE first combines the requested class intervals with the element data; if necessary, default class intervals are assigned. The flow durations, or number of days within each class interval, are calculated for each complete water year, and the values are then listed. Totals and means for each

class are given in the footnotes. The computations and listings are repeated each time the process is specified in one access request.

4.123.2 Interface Description

A) Parameters: V_UNITS - volume units: standard or metric
F_UNITS - flow units of the daily value
A_UNITS - area units for the drainage area
SCALE - converts class values to metric units

B) Global data:
Macro variables:
NOBS - the number of observations in a data set

SAS data sets:
STRM (input) - with variables:
STATION STN_FORM NAME COUNTY YEAR MONTH MONTHSUM
FLOW1-FLOW31 STRM_TOT AREA

CLASS (input) - with variables:
COUNT CLASS1-CLASS30 STN_REQ

FLOW1-FLOW&N (input and output) - with variables:
STATION YEAR MONTH STN_FORM NAME COUNTY AREA
FLOW1-FLOW31 CLASS1-CLASS30 WAT_YR COUNT STN_REQ

DUR1-DUR&N (input and output) - with variables:
STATION STN_FORM NAME COUNTY YEAR MONTH ABOVE1-ABOVE10
CLASS1-CLASS30 COUNT AREA FLOW_YR WAT_YR

PRINT (output) - standard SAS print file

C) No macros are called by this module.

D) No permanent files are used by this module.

E) The module generates several complete DATA steps.

4.123.3 Internal Data Description

Macro variables:
N : control variable for the %DO loop - number of times to execute the macro

SAS variables:
DEFAULT (num): array name for default class values
ADJUST (num): converts default class values to metric units
D (num): control variable for DO loop
RANGE (num): array of upper bounds for class intervals
COMPLETE (num): flags a complete water year
CURR_MON (num): flags the current month as begin legal

```
LESS      (num): one less than the number of class intervals
C         (num): control variable for DO loop
NEXT      (num): forward counter in a DO loop
DAY       (num): control variable for DO loop
DAY_TOT   (num): total number of days in period of record
DIS_TOT   (num): total discharge in period of record
ACCUM     (num): array which contains total number of days in
                 class intervals ¢= current interval
PERCNT    (num): array with percentage of total days that are
                 in class intervals ¢= current interval
VOLUME    (num): array with flow per unit of area for each class
AVE       (num): array with ratio of class per mean daily flow
MEAN_DAY  (num): mean daily flow for the period of record
DAYSUSED  (num): total number of days - number of days in
                 lower class intervals
EOF   (boolean): identifies last record read from input file
START     (num): column in which to begin listing one year's data
ROW       (num): row in which to print current data value
COLUMN    (num): column in which to print current data value
FIRST.STATION (boolean): first occurrence of current value of
                         STATION
LAST.STATION  (boolean): last occurrence of current value of
                         STATION
```

4.123.4 Design Language Description

```
%DO N = 1 %TO &NOBS;
  %* NOBS ¢ 1 only if user requests multiple executions ;
  %* of the process in one access request, or requests  ;
  %* different classes for each station.                ;

  DATA FLOW&N(KEEP=STATION YEAR MONTH NAME COUNTY STN_FORM
                   FLOW1-FLOW31 AREA  CLASS1-CLASS30  WAT_YR
                   COUNT);
     RETAIN ADJUST &SCALE  DEFAULT1 1  DEFAULT2 10 DEFAULT3 100
            DEFAULT4 1000  DEFAULT5 10000  DEFAULT6 100000
            DEFAULT7 1000000;
     ARRAY CLASS[30]  CLASS1-CLASS30;
     ARRAY DEFAULT[7] DEFAULT1-DEFAULT7;
     DO C = &N TO &N;
        SET CLASS POINT=C; /* data set with class intervals */

           SET STRM END=EOF;

           IF STN_REQ = missing  OR  STN_REQ = STATION  THEN
              DO;  /* keep info in this record */
```

```
            IF CLASS[1] = .  THEN
                 DO;  /* assign default class values  */
                   DO D = 1 TO 7; /* adjust for metric */
                      CLASS[D] = DEFAULT[D] * ADJUST;
                   END;
                 END;
            convert year to wat_yr; /* oct 1 - sept 30 */
            OUTPUT;
          END;
      IF EOF THEN STOP;
   END;
RUN;

DATA DUR&N(KEEP=STATION YEAR MONTH NAME COUNTY STN_FORM AREA
               WAT_YR COUNT  CLASS1-CLASS30 ABOVE1-ABOVE30
               FLOW_YR);
   ARRAY CLASS[30] CLASS1-CLASS30;
   ARRAY ABOVE[30] ABOVE1-ABOVE30;
   ARRAY RANGE[30] RANGE1-RANGE30;

   COMPLETE = 0;
   initialize ABOVE array to 0;
   CURR_MON = 10;
   FLOW_YR = 0;

   DO UNTIL(LAST.WAT_YR);  /* new record is one wat_yr */

      SET FLOW&N END=EOF;
        BY STATION WAT_YR;

      IF FIRST.STATION THEN
         /* calculate upper ranges for class intervals */
         LESS = COUNT - 1;
         DO C = 1 TO LESS;
            NEXT = C + 1;
            RANGE[C] = CLASS[NEXT];
         END;
         RANGE[COUNT] = 2000000; /* out of range value */
      END;

      IF CURR_MON = MONTH  THEN
         DO;  /* only keep complete water years */
           COMPLETE = COMPLETE + 1;
           DO C = 1 TO COUNT;
              DO DAY = 1 TO 31;
                 IF FLOW[DAY] is not equal to missing THEN
                    FLOW_YR = FLOW_YR + FLOW[DAY];
                 IF CLASS[C] _= FLOW[DAY] _= RANGE[C]
                    THEN
                       ABOVE[C] = ABOVE[C] + 1;
              END;
```

```
            END;
        END;

        IF CURR_MON = 12
           THEN CURR_MON = 1;
           ELSE CURR_MON = CURR_MON + 1;
    END;  /* last.wat_yr */

    IF COMPLETE = 12  THEN  OUTPUT;  /* complete water year */
    IF EOF  THEN STOP;
RUN;

DATA _NULL_; /* step to print flow duration table */

  FILE PRINT N=PS;
  RETAIN DAY_TOT 0  DIS_TOT 0 START 1;

  ARRAY ABOVE[30] ABOVE1-ABOVE30;
  ARRAY CLASS[30] CLASS1-CLASS30;
  ARRAY ACCUM[30] ACCUM1-ACCUM30;
  ARRAY PERCNT[30] PERCNT1-PERCNT30;
  ARRAY VOLUME[30] VOLUME1-VOLUME30;
  ARRAY AVE[30] AVE1-AVE30;

  SET DUR&N END=EOF;
    BY STATION;

  IF FIRST.STATION THEN
     print station info in heading;
     initialize DAY_TOT and DIS_TOT to 0;
     ROW = 9;

     IF COUNT _ 6  THEN
        START = 50;
     ELSE IF COUNT _ 11 THEN
        START = 40;
     ELSE IF COUNT _ 16 THEN
        START = 30;
     ELSE IF COUNT _ 21 THEN
        START = 20;
     ELSE IF COUNT _ 26 THEN
        START = 10;
     ELSE
        START = 1;
```

```
   PUT #4 @40 'FLOW DURATION TABLE'   ;
   COLUMN = START;
   PUT #6 @COLUMN 'CLASS'  ;
   COLUMN = COLUMN + 6 ;
   DO C = 1 TO COUNT;
      PUT #6 @COLUMN  C  2. ;
      COLUMN = COLUMN + 4;
   END;

   PUT #7 @START 'WATER';
   PUT #8 @START 'YEAR'
          @40 'NUMBER OF DAYS IN CLASS';

END;  /* first.station */

COLUMN = START;
print #ROW @COLUMN WAT_YR 4.;
COLUMN = COLUMN + 6;
DO C = 1 TO COUNT;
   PUT #ROW @COLUMN ABOVE[C] 4.;
   COLUMN = COLUMN + 4;
   DAY_TOT = DAY_TOT + ABOVE[C];
END;

DIS_TOT = DIS_TOT + FLOW_YR ;
ROW = ROW + 1;

IF ROW ¢ 55  THEN
  go to a new page;
  print station and table headings;

IF LAST.STATION  THEN
   IF AREA = 0  THEN AREA = . /* avoid divide by zero */
   IF ROW ¢ 39
     go to a new page;
     initialize row number;
   COLUMN = 1;
   ROW = ROW + 2;
   MEAN_DAY = DIS_TOT / DAY_TOT;
   %IF not a request for metric units %THEN
       /* convert CFS-DAYS to ACRE-FEET */

       DIS_TOT = DIS_TOT * 1.98;
   %END;
   print DIS_TOT, MEAN_DAY, and AREA in
         &V_UNITS, &F_UNITS, and &A_UNITS;
```

```
            DAYSUSED = 0;
            DO C = 1 TO COUNT;
               ACCUM[C] = DAY_TOT - DAYSUSED; /* number of days */
                                              /* ¢= this class  */
               DAYSUSED = DAYSUSED + ABOVE[C];
               PERCNT[C] = ACCUM[C] / DAY_TOT * 100;
               VOLUME[C] = CLASS[C] / AREA;
               AVE[C] = CLASS[C] / MEAN_DAY;
               print the above array values in columnar form at
               bottom of page, as in current system ;
            END;
            IF EOF THEN STOP;
                  ELSE
                      PUT _PAGE_;
    RUN;
%END;
```

4.124 Description of Module SUMMARY (SUMMARY)

4.124.1 Processing Narrative

SUMMARY calls the macros to identify the months requested by the user, and list the data from the monthly summary file. This process is almost the same as the LIST MONTHLY SUMMARY CONTENTS module (4.73); the main difference is that data listings are only produced by SUMMARY for the elements that are explicitly given by the user.

4.124.2 Interface Description

A) There is no parameter list.

B) Global data:

No global macro variables are used in this module.

No SAS data sets are created or used.

C) The macros called by this module are:

GET MONTHS FROM ONLY OPTION (4.92 - GET_MON)
CHECK NUMBER OF OBSERVATIONS (4.20 - NUM_OBS)
COMPUTE MONTHLY SUMMARY (4.125 - C_SUMM)

D) No permanent files are used by this module.

E) The module generates macro statements only.

4.124.3 Internal Data Description

No local data are used in this module.

4.124.4 Design Language Description

```
%GET MONTHS FROM ONLY OPTION
            (SUMMARY,SUMMARY,SUMARY)
%CHECK NUMBER OF OBSERVATIONS (SUMMARY)

%COMPUTE MONTHLY SUMMARY
```

4.125 Description of Module COMPUTE MONTHLY SUMMARY (C_SUMM) (Not Operational)

4.125.1 Processing Narrative

COMPUTE MONTHLY SUMMARY prints out the values from the monthly summary file. Annual totals are calculated and printed, if the user requests annual with the ONLY option.

4.125.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:

- NOBS : the number of observations in a data set
- EL_PRCP: identifies request for precipitation data
- EL_TEMP: identifies request for temperature data
- EL_SNOW: identifies request for snowfall data
- EL_EVAP: identifies request for evaporation data
- METRIC : identifies request for metric units

SAS data sets:

SUMMARY (input) - with variables:
BEGMONTH ENDMONTH RANGE

MNTH (input) - with variables:
STATION YEAR MONTH STN_FORM NAME COUNTY AREA ELEV
MAXTEMP MINTEMP TMEAN TDEPART DEGDAYS HIGHEST HIGHDAY
LOWEST LOWDAY PRECIP PDEPART PPTMAX PMAXDAY SNOFALL
SNODEPTH SMAXDAY WIND EVAP COOLDAYS PLUS1-PLUS6

SUMM1-SUMM&N (input and output) - with variables:
same as MNTH, with BEGMONTH ENDMONTH RANGE

PRINT (output) - standard SAS print file

C) No macros are called by this module.

D) No permanent files are used by this module.

E) The module generates two or more complete DATA steps.

4.125.3 Internal Data Description

Macro variables:
N : control variable for the %DO loop - number of times to execute the macro

SAS variables:
MAX_ANN (num): accumulator for mean annual maximum temp
MAX_CNTR (num): counter for number of months with max temps
MAX_MISS(char): identifies a month with missing max temps
MIN_ANN (num): accumulator for mean annual minimum temp
MIN_CNTR (num): counter for number of months with min temps
MIN_MISS(char): identifies a month with missing min temps
TMN_ANN (num): accumulator for mean annual temperature
TMN_CNTR (num): counter for number of months with mean temps
HIGH_ANN (num): highest daily temperature during year
LOW_ANN (num): lowest daily temperature during year
HEAT_ANN (num): total annual heating degree days
COOL_ANN (num): total annual cooling degree days
PRCP_ANN (num): total annual precipitation
PMAX_ANN (num): maximum daily precipitation during year
SNO_ANN (num): total annual snowfall
DEP_ANN (num): maximum depth of snow on ground during year
EVAP_ANN (num): total annual evaporation
WIND_ANN (num): total annual wind movement
ROW (num): identifies current row for printing values
PAGE_NUM (num): identifies current page number for output
LAST.STATION (boolean): identifies last occurrence of current value of STATION
LAST.YEAR (boolean): identifies last occurrence of current value of YEAR

4.125.4 Design Language Description

```
%DO N = 1 %TO &NOBS;
  %* NOBS ¢ 1 only if user wants more than one execution  ;
  %* of the same process in the same access request;

  DATA SUMM&N; /* keep only the requested months */

     SET MNTH; /* element data subset */
     DO I = &N TO &N;
        SET SUMMARY POINT=I; /* contains range of dates */
        /* check range of dates from only option - if annual */
        /* is requested, ENDMONTH = 13, keep entire record */
```

```
        IF ENDMONTH = 13  THEN
           OUTPUT;
        ELSE IF BEGMONTH _= MONTH _= ENDMONTH  THEN
                DO;
                  OUTPUT;

        ELSE IF BEGMONTH ¢ ENDMONTH THEN
                IF (MONTH ¢= BEGMONTH) OR (MONTH _= ENDMONTH)
                   THEN DO;
                     OUTPUT;
     END;
  RUN;

DATA _NULL_; /* lists monthly summary contents */

   FILE PRINT N=PS;

   RETAIN ROW 13  PAGE_NUM 1  MAX_ANN 0  MAX_CNTR 0  MIN_ANN 0
          MIN_CNTR 0  TMN_ANN 0  TMN_CNTR 0  HIGH_ANN -60
          LOW_ANN 130  HEAT_ANN 0  COOL_ANN 0  PRCP_ANN 0
          PMAX_ANN 0  SNO_ANN 0  DEP_ANN 0  EVAP_ANN 0
          WIND_ANN 0  ;

   SET SUMM&N;
     BY STATION YEAR;

   IF ROW = 13   THEN   /* print page headings */
      PUT #1 @100 'PAGE ' PAGE_NUM;
      PUT #2 @1 NAME   @65 COUNTY  @86 'STATION NO.  ' STN_FORM;
      PUT #4 @50  'MONTHLY SUMMARIZED STATION DATA';
      %IF not a request for metric units %THEN
          PUT #6 @5  'DATE'  @30  'TEMPERATURES (F)
                 @80 'PRECIPITATION (IN)'
                 @112 'EVAP/WIND (IN/MI)';
      %ELSE
          PUT #6 @5  'DATE'  @30  'TEMPERATURES (C)'
                 @75 'PRECIPITATION (MM/SNOW(CM))'
                 @112 'EVAP/WIND (MM/KM)';
      %END;

      /* now print column headings */
      PUT #8 row of dashes;
      PUT #9 @1  '¦'  @13 '¦'  @15 '¦'  @53 'HEAT'
             @60 'COOL'  @66 '¦'  @68 '¦'  @97 'GREATEST'
             @110 '¦'  @112 '¦'  @121 'MONTHLY'  @130 '¦';
      PUT #10 @1 '¦'  @13 '¦'  @15 '¦'
              @16 'MEAN'  @22 'MEAN'  @52 'DEGREE'  @59 'DEGREE'
              @66 '¦'  @68 '¦'  @75 'GREATEST'  @90 'TOTAL'
              @99 'SNOW'  @110 '¦'  @112 '¦'  @115 'TOTAL'
              @122 'WIND'  @130 '¦';
```

```
    PUT #11 @1  '¦'  @2 'MONTH'  @8 'YEAR'  @13 '¦'  @15 '¦'
            @16 'MAX'  @22 'MIN'  @29 'MEAN'  @35 'HI'
            @38 'DATE'  @43 'LOW'  @47 'DATE'  @53 'DAYS'
            @60 'DAYS'  @66 '¦'  @68 '¦'  @69 'TOTAL'
            @77 'DAY'  @84 'DATE'  @89 'SNOWFALL'
            @99 'DEPTH'  @106 'DATE'  @110 '¦'  @112 '¦'
            @115 'EVAP'  @122 'RUN'  @130 '¦';
    PUT #12 row of dashes;
END;  /* print page headings */

IF RANGE NE 'ANNUAL'  THEN  /* print all available data */
   /* now print one month's data values */

   PUT #ROW @3 MONTH 2. @8 YEAR 4.  @;

   %IF request for temperature data %THEN
       PUT  @15 MAXTEMP 5.1  PLUS1 $1.
            @22  MINTEMP 5.1  PLUS2 $1.
            @29 TMEAN 5.1  @35 HIGHEST 3.  @38 HIGHDAY 2.
            @43 LOWEST 3.  @47 LOWDAY 2.   @52 DEGDAYS 5.
            @59 COOLDAYS 5.  @;

   %IF request for precipitation data %THEN
       PUT  @69 PRECIP   PLUS5 $1.
            @76 PPTMAX   PLUS3 $1.  @85 PMAXDAY 2. @;

   %IF request for snowfall data %THEN
       PUT  @89 SNOFALL  5.1  PLUS6 $1.
            @97 SNODEPTH 4. PLUS4 $1.
            @107 SMAXDAY 2.  @;

   %IF request for evaporation data %THEN
       PUT  @114 EVAP   @121  WIND  @;

   PUT;  /* to release the output line */
END;  /* if range ne 'ANNUAL' */

IF ENDMONTH = 13  THEN
   /* accumulate values for annual means, totals, max/mins */

  IF (PLUS1 NE 'M') AND (PLUS1 NE 'I') THEN
                                /* no missing data */
     MAX_ANN = MAX_ANN + MAXTEMP;
     MAX_CNTR = MAX_CNTR + 1;
     MAX_MISS = 'N';
   IF (PLUS2 NE 'M') AND (PLUS2 NE 'I') THEN
                                /* no missing data */
     MIN_ANN = MIN_ANN + MINTEMP;
     MIN_CNTR = MIN_CNTR + 1;
     MIN_MISS = 'N';
```

```
    IF (MAX_MISS = 'N') AND (MIN_MISS = 'N') THEN

    TMN_ANN  = TMN_ANN + TMEAN;
    TMN_CNTR = TMN_CNTR + 1;
    HIGH_ANN = MAX(HIGH_ANN,HIGHEST);
    LOW_ANN  = MIN(LOW_ANN,LOWEST);
    HEAT_ANN = HEAT_ANN + DEGDAYS;
    COOL_ANN = COOL_ANN + COOLDAYS;
    PRCP_ANN = PRCP_ANN + PRECIP;
    PMAX_ANN = MAX(PMAX_ANN,PPTMAX);
    SNO_ANN  = SNO_ANN + SNOFALL;
    DEP_ANN  = MAX(DEP_ANN,SNODEPTH);
    EVAP_ANN = EVAP_ANN + EVAP;
    WIND_ANN = WIND_ANN + WIND;

    IF LAST.YEAR THEN  /* print annual values */
       ROW = ROW + 1;
       MAX_ANN = MAX_ANN / MAX_CNTR;
       MIN_ANN = MIN_ANN / MIN_CNTR;
       TMN_ANN = TMN_ANN / TMN_CNTR;

       PUT #ROW @1  'ANNUAL' @8 YEAR 4.
             @15 MAX_ANN 5.1   @22 MIN_ANN  5.1
             @29 TMN_ANN 5.1   @35 HIGH_ANN 3.
             @43 LOW_ANN 3.    @52 HEAT_ANN 5.
             @59 COOL_ANN 5.   @69 PRCP_ANN 6.2
             @76 PMAX_ANN 6.2
             @89 SNO_ANN  5.1  @97 DEP_ANN 4.
             @114 EVAP_ANN 6.2  @121  WIND_ANN 6.1;
       MAX_ANN = 0;
       MAX_CNTR = 0;
       MIN_ANN = 0;
       MIN_CNTR = 0;
       TMN_ANN = 0;
       TMN_CNTR = 0;
       HIGH_ANN = -60;
       LOW_ANN  = 130;
       HEAT_ANN = 0;
       COOL_ANN = 0;
       PRCP_ANN = 0;
       PMAX_ANN = 0;
       SNO_ANN  = 0;
       DEP_ANN  = 0;
       EVAP_ANN = 0;
       WIND_ANN = 0;
       ROW = ROW + 1;
    END;  /* print annual values */
END;  /* if endmonth = 13 */
```

```
   ROW = ROW + 1;
   IF (ROW ¢ 54) OR (LAST.STATION) THEN
      /* print footnotes, go to a new page,    */
      PUT #57 @1  '+ - VALUE OCCURRED ON MORE THAN ONE DAY'
              @60 'T - TRACE MONTHLY VALUE' ;
      PUT #58 @1  'M - MISSING RECORDS IN CALCULATION '
              @60 'I - MONTHLY VALUE BASED ON INCOMPLETE '
                  'PERIOD' ;
      PUT #59 @1  'A - ACCUMULATED AMOUNT'
              @60 'E - ESTIMATED AMOUNT' ;
      PUT _PAGE_;
      increment PAGE_NUM by 1;
      ROW = 13 ;
      RETURN;
   END;  /* go to a new page */

RUN;
```

4.126 Description of Module CALENDAR (CALENDR)

4.126.1 Processing Narrative

CALENDAR calls the macros to determine the months and years for which monthly calendars should be generated, to compute the long-term normals and list the results in calendar form. Only one execution of the calendar process is allowed per access request.

4.126.2 Interface Description

A) There is no parameter list.

B) Global data:

No global macro variables are used in this module.

No SAS data sets are created or used.

C) The macros called by this module are:

GET DATES FOR CALENDAR (4.127 - GET_DAT)
COMPUTE NORMALS (4.128 - C_NORMS)
COMPUTE CALENDAR (4.129 - C_CAL)

D) No permanent files are used by this module.

E) The module generates macro statements only.

4.126.3 Internal Data Description

No local data are used in this module.

4.126.4 Design Language Description

```
%GET DATES FOR CALENDAR(CALDATES,CALENDAR,CALENDER)

%COMPUTE NORMALS

%COMPUTE CALENDAR
```

4.127 Description of Module GET DATES FOR CALENDAR (GET_DAT)

4.127.1 Processing Narrative

GET DATES FOR CALENDAR parses data given in the ONLY option and creates a data set with specific months and years for which to generate a calendar page. If the ONLY option was not given by the user, an error message is issued and the program aborts.

4.127.2 Interface Description

A) Parameters: DSN - name of input data set
NAME1 - name of process given by user
NAME2 - alternate name of process given by user

B) Global data:

No global macro variables are used in this module.

SAS data sets:
PROCESS (input) - with variables:
PRC_NAME PARMETER

&DSN (output) - with variables:
BEGMONTH BEGYEAR ENDMONTH ENDYEAR

PRINT (output) - standard SAS print file

C) No macros are called by this module.

D) No permanent files are used by this module.

E) The module generates one complete DATA step.

4.127.3 Internal Data Description

No local macro variables are used in this module.

SAS variables:
MON (char): array for month names
WORD (char): array name for words scanned in parameter
I (num): control variable for DO loop

4.127.4 Design Language Description

```
DATA &DSN (KEEP=BEGMONTH BEGYEAR ENDMONTH ENDYEAR );
   /* data step to identify the months specified */
   /* in the ONLY option                         */
   LENGTH MON1-MON12 $ 10 WORD1-WORD6 $ 10
          BEGMONTH 6 ENDMONTH 6 BEGYEAR 6 ENDYEAR 6;

   RETAIN  MON1  'JANUARY'  MON2  'FEBRUARY'  MON3  'MARCH'
           MON4  'APRIL'    MON5  'MAY'       MON6  'JUNE'
           MON7  'JULY'     MON8  'AUGUST'    MON9  'SEPTEMBER'
           MON10 'OCTOBER'  MON11 'NOVEMBER'  MON12 'DECEMBER';
```

```
FILE PRINT;
ARRAY MON[12] MON1-MON12;
SET PROCESS;

IF PRC_NAME = "&NAME1"  OR PRC_NAME = "&NAME2"  THEN
   DO;
     IF PARMETER is not equal to missing THEN DO;
            WORD1 = SCAN(PARMETER,1,' ');
            IF WORD1 = 'ONLY'  THEN
               DO;
                 WORD2 = SCAN(PARMETER,2,' ');
                 WORD3 = SCAN(PARMETER,3,' ');
                 WORD4 = SCAN(PARMETER,4,' ');
                 IF WORD4 = 'TO' THEN
                    WORD5 = SCAN(PARMETER,5,' ');
                    WORD6 = SCAN(PARMETER,6,' ');

                 DO I = 1 TO 12;
                    IF WORD2 = MON[I]  THEN
                       BEGMONTH = I;
                 END;
                 BEGYEAR = WORD3;

                 IF WORD4 NE ' '  THEN
                    DO I = 1 TO 12;
                       IF WORD5 = MON[I]  THEN
                          ENDMONTH = I;
                    END;
                    ENDYEAR = WORD6;
                 ELSE
                    ENDMONTH = BEGMONTH;
                    ENDYEAR = BEGYEAR;
                 IF ENDMONTH = missing  THEN
                    ENDMONTH = BEGMONTH;
                 IF ENDYEAR = missing THEN
                    ENDYEAR - BEGYEAR;

                 IF BEGYEAR NE .  THEN
                    IF BEGMONTH NE .  THEN
                       OUTPUT;
                    ELSE issue error message that requested
                         month is not legal and ABORT;
                 ELSE issue error message that user did not
                      request a specific year or that the
                      requested year is not legal and ABORT;
               END;

            ELSE IF WORD1 NE 'ONLY'  THEN
               DO;
                 issue error message that illegal option
                  was used on the CALENDAR process and
                  ABORT;
```

```
        ELSE IF PARMETER is equal to missing THEN
             issue error message that no specific calendar
             dates were requested and ABORT  ;

    RUN;
```

4.128 Description of Module COMPUTE NORMALS (C_NORMS)

4.128.1 Processing Narrative

COMPUTE NORMALS creates subsets for the requested months of both the temperature and precipitation files. The two subsets are each sorted by station and month, and the daily normals for the requested period of record are calculated.

4.128.2 Interface Description

A) There is no parameter list.

B) Global data:

No global macro variables are used in this module.

SAS data sets:

CALDATES (input) - with variables:
BEGMONTH BEGYEAR ENDMONTH ENDYEAR

TEMP (input) - with variables:
STATION STN_FORM NAME COUNTY YEAR MONTH MAXTMP1-MAXTMP31 MINTMP1-MINTMP31 MONTHSUM AVEMAX AVEMIN

PRCP (input) - with variables:
STATION STN_FORM NAME COUNTY YEAR MONTH PRECIP1-PRECIP31 MONTHSUM PRCP_TOT

T_SUB (input and output) - with variables:
STATION STN_FORM NAME COUNTY YEAR MONTH MAXTMP1-MAXTMP31 MINTMP1-MINTMP31 MONTHSUM

P_SUB (input and output) - with variables:
STATION STN_FORM NAME COUNTY YEAR MONTH PRECIP1-PRECIP31 MONTHSUM

T_NORM (output) - with variables:
STATION STN_FORM NAME COUNTY MONTH DAY MAXNORM MAXIMUM MINNORM MINIMUM

P_NORM (output) - with variables:
STATION STN_FORM NAME COUNTY MONTH DAY PRCPNORM MAXPRCP

C) No macros are called by this module.

D) No permanent files are used by this module.

E) The module generates several complete DATA and PROC steps.

4.128.3 Internal Data Description

No local macro variables are used in this module.

SAS variables:

```
I        (num): control variable for DO loop
MAXACC   (num): array with accumulated maximum daily values
MAXCNT   (num): array with counts of number of days with maximums
MAXYR    (num): array with years in which maximum daily value
                occurred
HIGH     (num): array with maximum daily values for period
MINACC   (num): array with accumulated minimum daily values
MINCNT   (num): array with counts of number of days with minimums
MINYR    (num): array with years in which minimum daily value
                occurred
LOW      (num): array with minimum daily values for period
ACCUM    (num): array with accumulated precipitation daily values
COUNT    (num): array with counts of number of days with precip
PRCPYR   (num): array with years in which maximum daily
                precipitation occurred
LAST.MONTH (boolean): last occurrence of current value of MONTH
```

4.128.4 Design Language Description

```
DATA T_SUB(KEEP=STATION YEAR MONTH MONTHSUM NAME COUNTY
                MAXTMP1-MAXTMP31 MINTMP1-MINTMP31 STN_FORM);
   /* get subset of temperature data for requested months */
   SET TEMP;
   DO I = 1  TO 1;
      SET CALDATES POINT=I;
      /* check range of dates from only option -  */

      IF BEGMONTH _= MONTH _= ENDMONTH  THEN
         OUTPUT;
      ELSE IF BEGMONTH ¢ ENDMONTH THEN
            IF (MONTH ¢= BEGMONTH) OR (MONTH _= ENDMONTH)
               THEN  OUTPUT;
   END;   RUN;
```

```
DATA P_SUB(KEEP=STATION YEAR MONTH NAME COUNTY
               PRECIP1-PRECIP31 STN_FORM);
   /* get subset of precipitation data for requested months */
   SET PRCP;
   DO I = 1  TO 1;
      SET CALDATES POINT=I;
      /* check range of dates from only option -  */

      IF BEGMONTH _= MONTH _= ENDMONTH  THEN
         OUTPUT;
      ELSE IF BEGMONTH ¢ ENDMONTH THEN
              IF (MONTH ¢= BEGMONTH) OR (MONTH _= ENDMONTH)
                 THEN  OUTPUT;
   END;  RUN;


PROC SORT DATA=T_SUB;
   BY STATION MONTH;
RUN;

PROC SORT DATA=P_SUB;
   BY STATION MONTH;
RUN;

DATA T_NORM(KEEP=STATION STN_FORM MONTH DAY NAME COUNTY
                 MAXNORM MINNORM MAXIMUM MINIMUM );
   /* calculate max and min temperature normals  */

   ARRAY declarations;
   initialize MAXACC, MAXCNT, MINACC, MINCNT arrays to 0;
   initialize HIGH array to -60;
   initialize LOW  array to 130;

     DO UNTIL(LAST.MONTH);

        SET T_SUB;
            BY STATION MONTH;
        DO DAY = 1 TO 31; /* accumulate values, find max, min */

           IF MAXTMP[DAY] NE missing THEN
              MAXACC[DAY] = MAXACC[DAY] + MAXTMP[DAY];
              MAXCNT[DAY] = MAXCNT[DAY] + 1;
              IF MAXTMP[DAY] ¢= HIGH[DAY]  THEN
                 HIGH[DAY] = MAXTMP[DAY];
                 MAXYR[DAY] = YEAR;

           IF MINTMP[DAY] NE missing THEN
              MINACC[DAY] = MINACC[DAY] + MINTMP[DAY];
              MINCNT[DAY] = MINCNT[DAY] + 1;
```

```
            IF MINTMP[DAY] _= LOW[DAY]  THEN
               LOW[DAY] = MINTMP[DAY];
               MINYR[DAY] = YEAR;

         END;  /* accumulate values, find maxes and mins */
      END; /* last.month */

      DO DAY = 1 TO 31;  /* calculate norms and output one */
                         /* day per observation            */

         MAXNORM = MAXACC[DAY] / MAXCNT[DAY];
         MINNORM = MINACC[DAY] / MINCNT[DAY];
         MAXIMUM = HIGH[DAY] * 100 + MINYR[DAY] - 1900;
         IF LOW[DAY] ¢= 0 THEN
           MINIMUM = LOW[DAY] * 100 + MINYR[DAY] - 1900;
         ELSE
           MINIMUM = LOW[DAY] * 100 - MINYR[DAY] + 1900;
         OUTPUT;
      END;
RUN;


DATA P_NORM(KEEP=STATION STN_FORM MONTH DAY NAME COUNTY
                 PRCPNORM MAXPRCP );
   /* calculate precipitation normals  */

   ARRAY declarations;
   initialize ACCUM, COUNT, and HIGH arrays to 0;

     DO UNTIL(LAST.MONTH);

        SET P_SUB;
            BY STATION MONTH;

        DO DAY = 1 TO 31; /* accumulate values, find maxes */

           IF PRECIP[DAY] NE missing THEN
              ACCUM[DAY] = ACCUM[DAY] + PRECIP[DAY];
              COUNT[DAY] = COUNT[DAY] + 1;
              IF PRECIP[DAY] ¢= HIGH[DAY]  THEN
                 HIGH[DAY] = PRECIP[DAY];
                 PRCPYR[DAY] = YEAR;

        END;  /* accumulate values, find maxes  */
     END; /* last.month */

     DO DAY = 1 TO 31;  /* calculate norms and output one */
                        /* day per observation            */
```

```
        PRCPNORM = ACCUM[DAY] / COUNT[DAY];
        MAXPRCP = (HIGH[DAY] * 10000 + PRCPYR[DAY] -1900)
                  * 0.0001;
        OUTPUT;
      END;

  RUN;
```

4.129 Description of Module COMPUTE CALENDAR (C_CAL)

4.129.1 Processing Narrative

COMPUTE CALENDAR first creates and outline of dates for PROC CALENDAR, with each observation containing the date for one day. Then the temperature and precipitation normals are merged, the normals are combined with the date outline, and the data is sorted and given to PROC CALENDAR. The output from PROC CALENDAR is the output given to the user.

4.129.2 Interface Description

A) There is no parameter list.

B) Global data:

Macro variables:
METRIC : identifies request for metric units

SAS data sets:
STATIONS (input) - with variables:
STATION

CALDATES (input) - with variables:
BEGMONTH BEGYEAR ENDMONTH ENDYEAR

OUTLINE (input and output) - with variables:
STATION MONTH DAY MONDAYYR

T_NORM (input) - with variables:
STATION STN_FORM NAME COUNTY MONTH DAY MAXNORM MAXIMUM MINNORM MINIMUM

P_NORM (input) - with variables:
STATION STN_FORM NAME COUNTY MONTH DAY PRCPNORM MAXPRCP

WEATHER (input and output) - with variables:
STATION STN_FORM NAME COUNTY MONTH DAY MAXNORM MAXIMUM MINNORM MINIMUM PRCPNORM MAXPRCP NORM

CAL_DATA (input and output) - with variables:
STATION STN_FORM NAME COUNTY MONTH DAY MAXNORM MAXIMUM
MINNORM MINIMUM PRCPNORM MAXPRCP NORM MONDAYYR

PRINT (output) - standard SAS print file

C) No macros are called by this module.

D) Permanent files: NHIMS.FORMATS

E) The module generates several complete DATA and PROC steps.

4.129.3 Internal Data Description

No local macro variables are used in this module.

SAS variables:

I	(num):	control variable for DO loop
CURR_MON	(num):	current value for month
CURR_YR	(num):	current value for year
FINAL	(num):	final month for which to generate calendar
NUMDAYS	(num):	number of days in the month

4.129.4 Design Language Description

```
DATA OUTLINE(KEEP=STATION MONTH DAY MONDAYYR);
   /* data set with outline of dates for  PROC CALENDAR */

   SET STATIONS;

   DO I = 1 TO 1; /* set dates for one station */

      SET CALDATES POINT=I;
      CURR_MON = BEGMONTH;
      CURR_YR  = BEGYEAR;
      FINAL = ENDMONTH + 1;

      DO UNTIL(CURR_MON = FINAL);

         MONTH = CURR_MON;
         YEAR  = CURR_YR;
         NUMDAYS = 31;
         IF MONTH = 4 OR MONTH = 6 OR MONTH = 9 OR
            MONTH = 11  THEN
               NUMDAYS = 30;
         ELSE IF MONTH = 2  THEN
              IF MOD(YEAR,4) = 0  THEN NUMDAYS = 29;
                                  ELSE NUMDAYS = 28;
```

```
            DO DAY = 1 TO NUMDAYS;
               MONDAYYR = MDY(MONTH,DAY,YEAR);
               OUTPUT;
            END;

            IF CURR_MON = 12  THEN
               CURR_MON = 1;
               CURR_YR = CURR_YR + 1;
            ELSE CURR_MON = CURR_MON + 1;

         END;  /* current month ¢= endmonth */

   END;  /* set dates for one station */
RUN;

PROC SORT DATA=OUTLINE; /* sort in case wrapping around of years, */
   BY STATION MONTH DAY;      /* i.e., NOVEMBER 1985 TO MAY 1986  */
RUN;


DATA WEATHER;   /* combine temp and precip data by date */
   LENGTH NORM $ 5;
   MERGE T_NORM  P_NORM ;
     BY STATION MONTH DAY;
     NORM = .;
RUN;

DATA CAL_DATA; /* combine requested dates with weather data */
   MERGE OUTLINE  WEATHER;
     BY STATION MONTH DAY;
RUN;


PROC SORT DATA=CAL_DATA; /* put dates in order for PROC CALENDAR */
   BY MONDAYYR;
RUN;


PROC CALENDAR DATA=CAL_DATA; /* uses user-defined formats */
   TITLE 'CLIMATE CALENDAR ';

   %IF not a request for metric units %THEN
       TITLE3 'TEMPERATURES IN DEGREES F AND PRECIPITATION IN '
              'INCHES';
   %ELSE %IF a request for metric units %THEN
       TITLE3 'TEMPERATURES IN DEGREES C AND PRECIPITATION IN '
              'MILLIMETERS';
   %END;
   BY STATION  NAME  COUNTY;
   ID MONDAYYR;
   VAR NORM MAXNORM MINNORM PRCPNORM MAXIMUM MINIMUM MAXPRCP;
```

```
    FORMAT  NORM      FNOR. ;
    FORMAT  MAXNORM   FMAX. ;
    FORMAT  MAXIMUM   FHMX. ;
    FORMAT  MINNORM   FMIN. ;
    FORMAT  MINIMUM   FLMN. ;
    FORMAT  PRCPNORM  FPPT. ;
    FORMAT  MAXPRCP   FHPT. ;

RUN;
```

5.0 GLOBAL DATA AND FILE DESCRIPTIONS

In the NHIMS system, data is globally available from permanent SAS data sets, temporary SAS data sets, and global macro variables. In Section 5.1, the global macro variables and temporary data sets will be described, along with cross reference matrices identifying the modules that utilize them. In Section 5.2, the permanent data sets will be described, and a similar cross reference matrix will be provided. A SAS procedure, PROC FREQ, was used to generate all of the cross reference matrices.

5.1 GLOBAL DATA DESCRIPTIONS

In this section, the global macro variables are described; a group of cross reference matrices then identifies which modules use those variables. In order to make the matrices manageable, the variables were divided into logical groups. The first matrix refers to the variables which flag the different elements; the second matrix concerns the variables which identify the types of data to be retrieved: hourly, daily, monthly. The next three matrices refer to the variables which flag the list, copy, and process operations. The variables NOBS and METRIC have matrices all to themselves because so many macros use them. And finally, the last matrix refers to all miscellaneous variables not previously mentioned.

Next, the temporary SAS data sets are described, by briefly describing the type of data each one contains, and listing the names of the variables. Subsequently, a cross reference matrix identifies the modules that use these temporary data sets.

5.1.1 Description of Global Macro Variables

Within NHIMS, global macro variables are used as flags in the conditional macro statements, identifying when certain SAS statements should or should not be sent to the compiler. Usually, the values of these variables reflect the various data requests made by the user. The global macro variables available in NHIMS are:

```
EL_PRCP: flag for precipitation element
EL_TEMP: flag for temperature element
EL_STRM: flag for streamflow element
EL_SNOW: flag for snowfall element
EL_EVAP: flag for evaporation element
EL_RESV: flag for reservoir storage element
EL_SNOC: flag for snow course element
EL_PEAK: flag for peakflow element
EL_MNTH: flag for monthly element
EL_HPCP: flag for hourly precipitation element
C_ACCES: counter for the number of access requests
COUNT  : control variable for main DO loop
```

```
MAIN_DS: identifies need to retreive main data set info
DAILY  : identifies need to retrieve daily data
MONTHLY: identifies need to retrieve monthly data
HOURLY : identifies need to retrieve hourly data
METRIC : identifies request for data in metric units
PERIOD : identifies use of the PERIOD command
SORT   : identifies need to sort station subsets
POINTER: identifies use of user-specified pointers
ELEMENT: identifies use of ELEMENT command
NUM_DS : number of station subsets
NOBS   : number of observations in a data set
LIST   : identifies request for list operation
L_INDEX: identifies request for index listing
L_COMP : identifies request for complete file listing
LI_PER : identifies request for period of record with index
L_DAILY: identifies request for daily listing
L_MONTH: identifies request for monthly listing
L_HOUR : identifies request for hourly listing
L_CONT : identifies request for contents listing
L_PTRS : identifies request for listing of pointer file
LM_PART: identifies request for partial monthly listings
L_FIRST: identifies month in which listings begin
COPY   : identifies request for copy operation
C_INTGR: identifies request for copy integer
C_REAL : identifies request for copy real
C_DIRCT: identifies request for copy direct
C_80   : identifies request for copy 80-byte records
C_FORM : identifies request for copy formatted
FN     : file number of external file for copy output
PROCESS: identifies request for process operation
ANNCORR: identifies request for annual correlations
ANNSTAT: identifies request for annual statistics
ONLYANN: identifies request for only annual processing
DAISTA : identifies request for daily statistics process
STA    : alternate name requesting daily statistics process
MONSTA : identifies request for monthly statistics process
COR    : identifies request for correlation process
HIG    : identifies request for highest process
MAX    : alternate name requesting highest process
LOW    : identifies request for lowest process
MIN    : alternate name requesting lowest process
EXT    : identifies request for extreme process
RANORD : identifies request for rank order process
HIGOCC : identifies request for high occurrences process
LOWOCC : identifies request for low occurrences process
DAIOCC : identifies request for daily occurrences process
OCC    : alternate name requesting daily occurrences process
MONOCC : identifies request for monthly occurrences process
FLODURTA: identifies request for flow duration table process
SUM    : identifies request for summary process
CAL    : identifies request for calendar process
```

5.1.2 Cross Reference Matrices for Global Macro Variables

CROSS REFERENCE MATRIX OF MODULES VS. GLOBAL MACRO VARIABLES
THAT REFER TO THE INDIVIDUAL ELEMENTS

TABLE OF MODULE BY VARIABLE

MODULE VARIABLE

Frequency	EL_EVAP	EL_HPCP	EL_MNTH	EL_PEAK	EL_PRCP
CORR (4.98)	1	0	0	0	1
C_DIR (4.77)	1	1	1	1	1
C_EXT (4.79)	1	1	1	1	1
C_SUMM (4.125)	1	0	0	0	1
DAY_OCC (4.115)	0	0	0	0	1
DGLOBAL (4.2)	1	1	1	1	1
D_STATS (4.91)	1	0	0	0	1
ELEMENT (4.10)	1	1	1	1	1
EXTREME (4.105)	0	0	0	0	0
HIGHEST (4.101)	1	0	0	0	1
IGLOBAL (4.7)	1	1	1	1	1
LIST_CT (4.69)	0	0	1	1	0
LIST_DY (4.49)	1	0	0	0	1
LIST_HR (4.74)	0	1	0	0	0
LIST_IX (4.39)	1	1	1	1	1
LIST_MN (4.61)	1	1	0	0	1
LIST_PT (4.47)	1	1	1	1	1
LOWEST (4.102)	1	0	0	0	0
MON_OCC (4.116)	0	0	0	0	1
M_STATS (4.95)	1	0	0	0	1
RANKORD (4.107)	0	0	0	0	1
RETRIEV (4.22)	1	1	1	1	1
Total	16	10	9	9	18

(Continued)

CROSS REFERENCE MATRIX OF MODULES VS. GLOBAL MACRO VARIABLES
THAT REFER TO THE INDIVIDUAL ELEMENTS

TABLE OF MODULE BY VARIABLE

MODULE VARIABLE

Frequency	EL_RESV	EL_SNOC	EL_SNOW	EL_STRM	EL_TEMP
CORR (4.98)	0	0	1	1	1
C_DIR (4.77)	1	1	1	1	1
C_EXT (4.79)	1	1	1	1	1
C_SUMM (4.125)	0	0	1	0	1
DAY_OCC (4.115)	0	0	1	0	1
DGLOBAL (4.2)	1	1	1	1	1
D_STATS (4.91)	0	0	1	1	1
ELEMENT (4.10)	1	1	1	1	1
EXTREME (4.105)	0	0	0	0	1
HIGHEST (4.101)	0	0	1	1	1
IGLOBAL (4.7)	1	1	1	1	1
LIST_CT (4.69)	0	1	0	0	0
LIST_DY (4.49)	1	0	1	1	1
LIST_HR (4.74)	0	0	0	0	0
LIST_IX (4.39)	1	1	1	1	1
LIST_MN (4.61)	0	0	1	1	1
LIST_PT (4.47)	1	1	1	1	1
LOWEST (4.102)	0	0	0	1	1
MON_OCC (4.116)	0	0	1	0	0
M_STATS (4.95)	0	0	1	1	1
RANKORD (4.107)	0	0	1	1	1
RETRIEV (4.22)	1	1	1	1	1
Total	9	9	18	16	19

CROSS REFERENCE MATRIX OF MODULES VS. GLOBAL MACRO VARIABLES
THAT REFER TO THE CATEGORY OF DATA RETRIEVAL

TABLE OF MODULE BY VARIABLE

MODULE / VARIABLE	DAILY	HOURLY	MONTHLY
Frequency			
CV_EVAP (4.32)	1	0	0
CV_HPCP (4.37)	0	1	0
CV_PRCP (4.28)	1	0	0
CV_SNOW (4.31)	1	0	0
CV_STRM (4.30)	1	0	0
CV_TEMP (4.29)	1	0	0
DGLOBAL (4.2)	1	1	1
IGLOBAL (4.7)	1	1	1
ST_COPY (4.14)	1	1	0
ST_LIST (4.13)	1	1	1
ST_PROC (4.15)	1	1	1
Total	10	6	4

CROSS REFERENCE MATRIX OF MODULES VS. GLOBAL MACRO VARIABLES
THAT REFER TO THE LIST OPERATIONS

TABLE OF MODULE BY VARIABLE

MODULE VARIABLE

Frequency	LI_ALL	LI_NHIM	LI_NONM	LI_PER	LM_PART
DGLOBAL (4.2)	0	0	0	1	1
IGLOBAL (4.7)	0	0	0	1	1
LIST (4.38)	0	0	0	0	0
LIST_DY (4.49)	0	0	0	0	0
LIST_MN (4.61)	0	0	0	0	1
LIST_PT (4.47)	0	0	0	0	0
LI_ELEM (4.40)	1	1	1	1	0
LI_PRT (4.43)	1	1	1	1	0
LM_AVE2 (4.68)	0	0	0	0	1
LM_TOT1 (4.66)	0	0	0	0	1
LM_TOT2 (4.67)	0	0	0	0	1
PARSE (4.9)	0	0	0	0	0
PERIOD (4.12)	0	0	0	0	0
STATION (4.11)	0	0	0	0	0
ST_COPY (4.14)	0	0	0	0	0
ST_LIST (4.13)	1	1	1	1	1
ST_PROC (4.15)	0	0	0	0	0
Total	3	3	3	5	7

(Continued)

CROSS REFERENCE MATRIX OF MODULES VS. GLOBAL MACRO VARIABLES
THAT REFER TO THE LIST OPERATIONS

TABLE OF MODULE BY VARIABLE

MODULE VARIABLE

Frequency	L_COMP	L_CONT	L_DAILY	L_FIRST	L_HOUR
DGLOBAL (4.2)	1	1	1	1	1
IGLOBAL (4.7)	1	1	1	1	1
LIST (4.38)	0	1	1	0	1
LIST_DY (4.49)	0	0	0	1	0
LIST_MN (4.61)	0	0	0	1	0
LIST_PT (4.47)	1	0	0	0	0
LI_ELEM (4.40)	1	0	0	0	0
LI_PRT (4.43)	0	0	0	0	0
LM_AVE2 (4.68)	0	0	0	0	0
LM_TOT1 (4.66)	0	0	0	0	0
LM_TOT2 (4.67)	0	0	0	0	0
PARSE (4.9)	1	0	0	0	0
PERIOD (4.12)	1	0	0	0	0
STATION (4.11)	1	0	0	0	0
ST_COPY (4.14)	1	0	0	0	0
ST_LIST (4.13)	1	1	1	1	1
ST_PROC (4.15)	1	0	0	0	0
Total	10	4	4	5	4

(Continued)

CROSS REFERENCE MATRIX OF MODULES VS. GLOBAL MACRO VARIABLES
THAT REFER TO THE LIST OPERATIONS

TABLE OF MODULE BY VARIABLE

MODULE VARIABLE

Frequency	L_INDEX	L_MONTH	L_PTRS
DGLOBAL (4.2)	1	1	1
IGLOBAL (4.7)	1	1	1
LIST (4.38)	1	1	1
LIST_DY (4.49)	0	0	0
LIST_MN (4.61)	0	0	0
LIST_PT (4.47)	0	0	0
LI_ELEM (4.40)	0	0	0
LI_PRT (4.43)	0	0	0
LM_AVE2 (4.68)	0	0	0
LM_TOT1 (4.66)	0	0	0
LM_TOT2 (4.67)	0	0	0
PARSE (4.9)	0	0	0
PERIOD (4.12)	0	0	0
STATION (4.11)	0	0	0
ST_COPY (4.14)	0	0	0
ST_LIST (4.13)	1	1	1
ST_PROC (4.15)	0	0	0
Total	4	4	4

CROSS REFERENCE MATRIX OF MODULES VS. GLOBAL MACRO VARIABLES
THAT REFER TO THE COPY OPERATIONS

TABLE OF MODULE BY VARIABLE

MODULE / VARIABLE	C_80	C_DIRCT	C_FORM	C_INTGR	C_REAL	FN
Frequency						
CD_SAS (4.78)	0	0	0	0	0	1
COPY (4.76)	0	1	0	0	0	0
CP_EVAP (4.84)	0	0	0	0	0	1
CP_HPCP (4.89)	0	0	0	0	0	1
CP_MNTH (4.86)	0	0	0	0	0	1
CP_PEAK (4.87)	0	0	0	0	0	1
CP_PRCP (4.80)	0	0	0	0	0	1
CP_RESV (4.88)	0	0	0	0	0	1
CP_SNOC (4.85)	0	0	0	0	0	1
CP_SNOW (4.83)	0	0	0	0	0	1
CP_STRM (4.82)	0	0	0	0	0	1
CP_TEMP (4.81)	0	0	0	0	0	1
C_DIR (4.77)	0	0	0	0	0	1
C_EXT (4.79)	0	0	0	0	0	1
DGLOBAL (4.2)	1	1	1	1	1	1
IGLOBAL (4.7)	1	1	1	1	1	1
ST_COPY (4.14)	1	1	1	1	1	0
Total	3	4	3	3	3	15

CROSS REFERENCE MATRIX OF MODULES VS. GLOBAL MACRO VARIABLES
THAT REFER TO THE INDIVIDUAL PROCESSES

TABLE OF MODULE BY VARIABLE

MODULE VARIABLE

Frequency	ANNCORR	ANNSTAT	CAL	COR	DAIOCC	DAISTA
C_CORR (4.100)	1	0	0	0	0	0
C_DSTAT (4.94)	0	1	0	0	0	0
C_MSTAT (4.97)	0	1	0	0	0	0
DAY_SUB (4.93)	0	1	0	0	0	0
DGLOBAL (4.2)	1	1	1	1	1	1
IGLOBAL (4.7)	1	1	1	1	1	1
MAKESUB (4.99)	1	0	0	0	0	0
MON_SUB (4.96)	0	1	0	0	0	0
PROCESS (4.90)	0	0	1	1	1	1
RETRIEV (4.22)	0	0	0	0	0	0
ST_PROC (4.15)	0	0	1	1	1	1
Total	4	6	4	4	4	4

(Continued)

CROSS REFERENCE MATRIX OF MODULES VS. GLOBAL MACRO VARIABLES
THAT REFER TO THE INDIVIDUAL PROCESSES

TABLE OF MODULE BY VARIABLE

MODULE VARIABLE

Frequency	EXT	FLODURTA	HIG	HIGOCC	LOW	LOWOCC
C_CORR (4.100)	0	0	0	0	0	0
C_DSTAT (4.94)	0	0	0	0	0	0
C_MSTAT (4.97)	0	0	0	0	0	0
DAY_SUB (4.93)	0	0	0	0	0	0
DGLOBAL (4.2)	1	1	1	1	1	1
IGLOBAL (4.7)	1	1	1	1	1	1
MAKESUB (4.99)	0	0	0	0	0	0
MON_SUB (4.96)	0	0	0	0	0	0
PROCESS (4.90)	1	1	1	1	1	1
RETRIEV (4.22)	0	0	0	0	0	0
ST_PROC (4.15)	1	1	1	1	1	1
Total	4	4	4	4	4	4

(Continued)

CROSS REFERENCE MATRIX OF MODULES VS. GLOBAL MACRO VARIABLES
THAT REFER TO THE INDIVIDUAL PROCESSES

TABLE OF MODULE BY VARIABLE

MODULE / VARIABLE	MAX	MIN	MONOCC	MONSTA	OCC	ONLYANN
Frequency						
C_CORR (4.100)	0	0	0	0	0	1
C_DSTAT (4.94)	0	0	0	0	0	1
C_MSTAT (4.97)	0	0	0	0	0	1
DAY_SUB (4.93)	0	0	0	0	0	1
DGLOBAL (4.2)	1	1	1	1	1	1
IGLOBAL (4.7)	1	1	1	1	1	1
MAKESUB (4.99)	0	0	0	0	0	1
MON_SUB (4.96)	0	0	0	0	0	1
PROCESS (4.90)	1	1	1	1	1	0
RETRIEV (4.22)	0	0	0	0	0	0
ST_PROC (4.15)	1	1	1	1	1	0
Total	4	4	4	4	4	8

(Continued)

CROSS REFERENCE MATRIX OF MODULES VS. GLOBAL MACRO VARIABLES
THAT REFER TO THE INDIVIDUAL PROCESSES

TABLE OF MODULE BY VARIABLE

MODULE VARIABLE

Frequency	RANORD	STA	SUM
C_CORR (4.100)	0	0	0
C_DSTAT (4.94)	0	0	0
C_MSTAT (4.97)	0	0	0
DAY_SUB (4.93)	0	0	0
DGLOBAL (4.2)	1	1	1
IGLOBAL (4.7)	1	1	1
MAKESUB (4.99)	0	0	0
MON_SUB (4.96)	0	0	0
PROCESS (4.90)	1	1	1
RETRIEV (4.22)	0	0	1
ST_PROC (4.15)	1	1	1
Total	4	4	5

CROSS REFERENCE MATRIX
MODULES VS. GLOBAL MACRO VARIABLE METRIC

TABLE OF MODULE BY VARIABLE

MODULE VARIABLE

Frequency	METRIC
CV_EVAP (4.32)	1
CV_HPCP (4.37)	1
CV_MNTH (4.36)	1
CV_PEAK (4.35)	1
CV_PRCP (4.28)	1
CV_RESV (4.33)	1
CV_SNOC (4.34)	1
CV_SNOW (4.31)	1
CV_STRM (4.30)	1
CV_TEMP (4.29)	1
C_CAL (4.126)	1
C_SUMM (4.125)	1
DAY_OCC (4.115)	1
DGLOBAL (4.2)	1
EXTREME (4.105)	1
FLO_DUR (4.121)	1
HIGHEST (4.101)	1
HI_OCC (4.110)	1
Total	36

(Continued)

CROSS REFERENCE MATRIX
MODULES VS. GLOBAL MACRO VARIABLE METRIC

TABLE OF MODULE BY VARIABLE

MODULE	VARIABLE
Frequency	METRIC
IGLOBAL (4.7)	1
LC_MNTH (4.73)	1
LC_PEAK (4.70)	1
LC_SNOC (4.71)	1
LD_EVAP (4.54)	1
LD_HPCP (4.56)	1
LD_PRCP (4.50)	1
LD_RESV (4.55)	1
LD_SNOW (4.53)	1
LD_STRM (4.52)	1
LD_TEMP (4.51)	1
LH_HPCP (4.75)	1
LIST_MN (4.61)	1
LOWEST (4.102)	1
LO_OCC (4.111)	1
MON_OCC (4.116)	1
PARSE (4.9)	1
RANKORD (4.107)	1
Total	36

CROSS REFERENCE MATRIX
MODULES VS. GLOBAL MACRO VARIABLE NOBS

TABLE OF MODULE BY VARIABLE

MODULE VARIABLE

Frequency	NOBS
C_CORR (4.100)	1
C_DOCC (4.118)	1
C_FLOW (4.123)	1
C_HIOCC (4.113)	1
C_LOOCC (4.114)	1
C_MOCC (4.120)	1
C_RANK (4.109)	1
C_SOCC (4.119)	1
C_SUMM (4.125)	1
DGLOBAL (4.2)	1
HI_LO_1 (4.103)	1
HI_LO_2 (4.104)	1
NUM_OBS (4.20)	1
PARSE (4.9)	1
PRT_EX (4.106)	1
RETRIEV (4.22)	1
Total	16

CROSS REFERENCE MATRIX OF MODULES VS. GLOBAL MACRO VARIABLES
THAT HAVE MISCELLANEOUS USES

TABLE OF MODULE BY VARIABLE

MODULE VARIABLE

Frequency	COPY	COUNT	C_ACCES	ELEMENT	LIST	MAIN_DS
ACCESS (4.6)	1	0	0	0	1	1
CNT_REQ (4.4)	0	0	1	0	0	0
DGLOBAL (4.2)	1	1	1	1	1	1
ELEMENT (4.10)	0	0	0	1	0	0
GETDATA (4.26)	0	0	0	0	0	0
GET_ANN (4.27)	0	0	0	0	0	0
HEADER (4.8)	0	1	0	0	0	0
IGLOBAL (4.7)	1	0	1	1	1	1
LI_ELEM (4.40)	0	0	0	0	0	1
MAIN (4.1)	0	1	1	0	0	0
PARSE (4.9)	0	1	0	1	0	1
PERIOD (4.12)	0	0	0	0	0	0
RETRIEV (4.22)	0	0	0	0	0	0
SEPRATE (4.3)	0	0	1	0	0	0
SRT_MRG (4.21)	0	0	0	0	0	0
ST_COPY (4.14)	1	0	0	0	0	1
ST_LIST (4.13)	0	0	0	0	1	1
ST_PROC (4.15)	0	0	0	0	0	1
SUBSETS (4.5)	0	0	1	0	0	0
Total	4	4	6	4	4	8

(Continued)

CROSS REFERENCE MATRIX OF MODULES VS. GLOBAL MACRO VARIABLES
THAT HAVE MISCELLANEOUS USES

TABLE OF MODULE BY VARIABLE

MODULE VARIABLE

Frequency	NUM_DS	PERIOD	POINTER	PROCESS	SORT
ACCESS (4.6)	0	0	0	1	0
CNT_REQ (4.4)	0	0	0	0	0
DGLOBAL (4.2)	1	1	1	1	1
ELEMENT (4.10)	0	0	0	0	0
GETDATA (4.26)	0	1	0	0	0
GET_ANN (4.27)	0	1	0	0	0
HEADER (4.8)	0	0	0	0	0
IGLOBAL (4.7)	1	1	1	1	1
LI_ELEM (4.40)	0	0	0	0	0
MAIN (4.1)	0	0	0	0	0
PARSE (4.9)	1	0	1	0	1
PERIOD (4.12)	0	1	0	0	0
RETRIEV (4.22)	0	0	1	0	0
SEPRATE (4.3)	0	0	0	0	0
SRT_MRG (4.21)	0	0	0	0	1
ST_COPY (4.14)	0	0	0	0	0
ST_LIST (4.13)	0	0	0	0	0
ST_PROC (4.15)	0	0	0	1	0
SUBSETS (4.5)	0	0	0	0	0
Total	3	5	4	4	4

5.1.3 Description of Temporary SAS Data Sets

A key strategy in the design of the NHIMS system is to make subsets of the main data files, and then rearrange, merge, and analyze the subsets in order to provide the user with the data he requests. The following list briefly describes all of the temporary data sets used in NHIMS, in alphabetical order. Some of the data sets begin with &EL., which stands for the 4-character element name described in the introduction to the module descriptions, Section 4.0. The names of the data sets can begin with any of the 4-character names.

ACCESSn : one or more data sets which contain the user commands; each data set has commands for one access request

RECORD (char): one 80-byte record from user command file

CALDATES : contains months and years for calendar process

BEGMONTH (num): beginning month of calendar process option
BEGYEAR (num): beginning year of calendar process option
ENDMONTH (num): ending month of calendar process option
ENDYEAR (num): ending year of calendar process option

CAL_DATA : contains station information, dates, long-term normals, and extremes to be printed with PROC CALENDAR

STATION (char): station code number
STN_FORM (char): formatted version of station code number
NAME (char): station name
COUNTY (char): county in which station is located
MONTH (num): month in which data were recorded
DAY (num): day during which data were recorded
MAXNORM (num): normal maximum temperature for month/day
MAXIMUM (num): extreme max temperature and year when recorded
MINNORM (num): normal minimum temperature for month/day
MINIMUM (num): extreme min temperature and year when recorded
PRCPNORM (num): normal precipitation for month/day
MAXPRCP (num): maximum precipitation and year when recorded
NORM (num): dummy variable used for daily calendar heading
MONDAYYR (num): month, day, and year as SAS date variable

CLASS : contains class intervals for flow duration table process

COUNT (num): number of class intervals specified (default=7)
CLASS1-30 (num): array with lower limits of class intervals
STN_REQ (char): station number, if user requests different class intervals for each station

CORR : contains range of months requested in correlation process

```
BEGMONTH (num): beginning month for process with ONLY option
ENDMONTH (num): ending month for process with ONLY option
RANGE   (char): range of months for process with ONLY option
```

DSTATS : contains range of months requested in daily statistics

```
BEGMONTH (num): beginning month for process with ONLY option
ENDMONTH (num): ending month for process with ONLY option
RANGE   (char): range of months for process with ONLY option
```

DURn : data for flow duration process

```
STATION  (char): station code number
STN_FORM (char): formatted version of station code number
NAME     (char): station name
COUNTY   (char): county in which station is located
AREA      (num): drainage area
WAT_YR    (num): water year (OCT. 1 - SEP. 30)
YEAR      (num): calendar year
MONTH     (num): month in which data were recorded
COUNT     (num): number of class intervals specified (default=7)
CLASS1-30 (num): array with lower limits of class intervals
ABOVE1-30 (num): array with the number of days in each interval
FLOW_YR   (num): total discharge for the water year
```

DY_OCC : contains data from the ONLY and THRESHOLD options with the daily occurrences process

```
BEGMONTH (num): beginning month for process with ONLY option
ENDMONTH (num): ending month for process with ONLY option
RANGE   (char): range of months for process with ONLY option
COUNT     (num): number of thresholds specified (default=1)
THRES1-10 (num): array with threshold levels
STN_REQ  (char): station number, if user requests different
                 thresholds for each station
```

DY_OCCn : one or more data sets (one is created each time the process is requested) with monthly records for daily occurrences process

```
   STATION  (char): station code number
   STN_FORM (char): formatted version of station code number
   NAME     (char): station name
   COUNTY   (char): county in which station is located
   YEAR      (num): calendar year
   MONTH     (num): month in which data were recorded
   MONTHSUM  (num): monthly code summary

   &ARRRAY1-31 (num): array of daily values, name depends on
                      the element specified
   COUNT     (num): number of thresholds specified (default=1)
   THRES1-10 (num): array with threshold levels
   STN_REQ  (char): station number, if user requests different
                    thresholds for each station
```

DY_SUBn : one or more data sets (one is created each time the process is requested) with calculations for the daily occurrences process

```
   STATION  (char): station code number
   STN_FORM (char): formatted version of station code number
   NAME     (char): station name
   COUNTY   (char): county in which station is located
   YEAR      (num): calendar year
   MONTH     (num): month in which data were recorded
   COUNT     (num): number of thresholds specified (default=1)
   THRES1-10 (num): array with threshold levels
   ABOVE1-10 (num): number of days above each threshold
```

&EL.ALL : combination of annual statistics with either the daily or monthly statistics

```
   STATION  (char): station code number
   STN_FORM (char): formatted version of station code number
   NAME     (char): station name
   COUNTY   (char): county in which station is located
   MONTH     (num): month in which data were recorded
   &EL.DAY or &MON_VAL (num): daily or monthly value
   MEAN      (num): mean of daily or monthly values
   STD       (num): standard deviation of daily or monthly values
   N         (num): number of observations used in calculations
   NMISS     (num): number of missing values found
   MAXIMUM   (num): maximum value
   MINIMUM   (num): minimum value
   SKEWNESS  (num): the measure of skewness
   KURTOSIS  (num): the measure of kurtosis
   LAGDAY or LAGMON (num): the lag value (from previous record)
   _NAME_   (char): the name of the daily or monthly variable
```

&EL.ANN : combination of annual statistics generated by PROC MEANS with those from PROC CORR

same variable names as &EL.ALL

&EL.BOTH : combination of daily or monthly statistics generated by PROC MEANS with those from PROC CORR

same variable names as &EL.ALL

&EL.CAN2 : annual statistics from PROC CORR, after dropping unnecessary values and adding annual month of 13

STATION (char): station code number
MONTH (num): month in which data were recorded
LAGDAY or LAGMON (num): the lag value (from previous record)
NAME (char): the name of the daily or monthly variable

&EL.CANN : annual statistics from PROC CORR

STATION (char): station code number
&EL.DAY or &MON_VAL (num): daily or monthly value
LAGDAY or LAGMON (num): the lag value (from previous record)
NAME (char): the name of the daily or monthly variable
TYPE (char): the type of observation

&EL.CNEW : daily or monthly statistics from PROC CORR, after dropping the unnecessary values

STATION (char): station code number
MONTH (num): month in which data were recorded
LAGDAY or LAGMON (num): the lag value (from previous record)
NAME (char): the name of the daily or monthly variable

&EL.COMB : data used in the correlation process, with the daily values of each station to be correlated in one record

YEAR (num): calendar year
MONTH (num): month in which data were recorded
DAY (num): day during which data were recorded
STN1-n (num): array of daily data values for each station

&EL.COMP : data set used as input for list monthly macro, when user requests complete listing of monthly averages

```
STATION  (char): station code number
STN_FORM (char): formatted version of station code number
NAME     (char): station name
COUNTY   (char): county in which station is located
AREA      (num): drainage area of station
ELEV      (num): elevation of station
YEAR     (num): calendar year
MONTH     (num): month in which data were recorded
MONTHSUM  (num): monthly code summary
&EL._TOT or &E_VAR1 (num): monthly total or average
&EL2._TOT or &E_VAR2 (num): monthly total or average for second
                            value (only for elements with two
                            daily values)
MEAN_ANN  (num): mean annual total or average
MEAN_AN2  (num): mean annual total or average for second value
                 (only for elements with two daily values)
```

&EL.CORR : a subset of data for the correlation process, with just the months specified in the ONLY option

```
STATION  (char): station code number
STN_FORM (char): formatted version of station code number
NAME     (char): station name
COUNTY   (char): county in which station is located
YEAR     (num): calendar year
MONTH     (num): month in which data were recorded
&ARRRAY1-31 (num): array of daily values, name depends on
                   the element specified
```

&EL.CSUB : daily or monthly statistics from PROC CORR

```
STATION  (char): station code number
MONTH     (num): month in which data were recorded
&EL.DAY or &MON_VAL (num): daily or monthly value
LAGDAY or LAGMON (num): the lag value (from previous record)
_NAME_   (char): the name of the daily or monthly variable
_TYPE_   (char): the type of observation
```

&EL.DATA : a subset of the permanent data set for one element

```
STATION  (char): station code number
NAME     (char): station name
COUNTY   (char): county in which station is located
AREA      (num): drainage area of station
ELEV      (num): elevation of station
YEAR     (num): calendar year
all other variables from permanent data set for the element
```

&EL.MANN : annual statistics calculated from PROC MEANS

```
STATION  (char): station code number
STN_FORM (char): formatted version of station code number
NAME     (char): station name
COUNTY   (char): county in which station is located
&EL.DAY or &MON_VAL (num): daily or monthly value
MEAN      (num): mean of daily or monthly values
STD       (num): standard deviation of daily or monthly values
N         (num): number of observations used in calculations
NMISS     (num): number of missing values found
MAXIMUM   (num): maximum value
MINIMUM   (num): minimum value
SKEWNESS  (num): the measure of skewness
KURTOSIS  (num): the measure of kurtosis
```

&EL.MEAN : contains mean annual values for list monthly, when user requests complete listing of monthly averages

```
STATION  (char): station code number
MEAN_ANN  (num): mean annual total or average
MEAN_AN2  (num): mean annual total or average for second value
                 (only for elements with two daily values)
```

&EL.MON : subset of data set for one element, used for both daily and monthly statistics processes, with only the months specified in the ONLY option

```
STATION  (char): station code number
STN_FORM (char): formatted version of station code number
NAME     (char): station name
COUNTY   (char): county in which station is located
YEAR      (num): calendar year
MONTH     (num): month in which data were recorded
&ARRRAY.1-31 or &MON_VAL (num): either 31 daily values or one
                                monthly total
```

&EL.MSUB : daily or monthly statistics from PROC MEANS

```
STATION  (char): station code number
STN_FORM (char): formatted version of station code number
NAME     (char): station name
COUNTY   (char): county in which station is located
MONTH     (num): month in which data were recorded
```

```
   &EL.DAY or &MON_VAL (num): daily or monthly value
   MEAN       (num): mean of daily or monthly values
   STD        (num): standard deviation of daily or monthly values
   N          (num): number of observations used in calculations
   NMISS      (num): number of missing values found
   MAXIMUM    (num): maximum value
   MINIMUM    (num): minimum value
   SKEWNESS   (num): the measure of skewness
   KURTOSIS   (num): the measure of kurtosis
```

&EL.PER : contains the beginning and ending periods of NHIMS records for stations from one element file

```
   STATION   (char): station code number
   FIRST_YR   (num): first year for which records are available
   FIRST_MN   (num): first month for which records are available
   LAST_YR    (num): last year for which records are available
   LAST_MN    (num): last month for which records are available
   MLENGTH    (num): number of observations for the station
```

&EL.PTIX : contains data from pointer file and index file for the requested stations

```
   STATION   (char): station code number
   NAME      (char): station name
   COUNTY    (char): county in which station is located
   AREA       (num): drainage area of station
   ELEV       (num): elevation of station
   FOBS_PTR   (num): first observation number for the station
   LOBS_PTR   (num): last observation number for the station
```

&EL.PTRS : subset of one of the permanent pointer files, with the pointers for the requested stations

```
   STATION   (char): station code number
   FOBS_PTR   (num): first observation number for the station
   LOBS_PTR   (num): last observation number for the station
```

&EL.STAT : data subset used in daily and monthly statistics processes, with one daily or monthly value per observation; contains the lag value for PROC CORR and is also used by PROC MEANS

```
   STATION   (char): station code number
   STN_FORM  (char): formatted version of station code number
   NAME      (char): station name
```

```
   COUNTY    (char): county in which station is located
   MONTH      (num): month in which data were recorded
   &EL.DAY or &MON_VAL (num): daily or monthly value
   LAGDAY or LAGMON (num): the lag value (from previous record)
```

EVAP : subset of permanent evaporation file, after conversions

```
   STATION  (char): station code number
   STN_FORM (char): formatted version of station code number
   NAME     (char): station name
   COUNTY   (char): county in which station is located
   AREA      (num): drainage area of station
   ELEV      (num): elevation of station
   YEAR      (num): calendar year
   MONTH     (num): month in which data were recorded
   EVAP1-31  (num): array with daily evaporation values
   WIND1-31  (num): array with daily wind movement values
   E_CODE1-31(char): array with daily evaporation codes
   W_CODE1-31(char): array with daily wind movement codes
   EVAP_TOT  (num): total monthly evaporation
   WIND_TOT  (num): total monthly wind movement
   MONTHSUM  (num): monthly summary of codes
```

EXTREM : contains range of months requested in extreme process

```
   BEGMONTH (num): beginning month for process with ONLY option
   ENDMONTH (num): ending month for process with ONLY option
   RANGE   (char): range of months for process with ONLY option
```

EXTREMn : one or more data subsets for the extreme process
(one is created each time the process is requested)

```
   STATION  (char): station code number
   STN_FORM (char): formatted version of station code number
   NAME     (char): station name
   COUNTY   (char): county in which station is located
   YEAR      (num): calendar year
   MONTH     (num): month in which data were recorded
   MONTHSUM  (num): monthly summary of codes
   HIGH      (num): maximum daily value in the month
   LOW       (num): minimum daily value in the month
   BEGMONTH  (num): beginning month for process with ONLY option
   ENDMONTH  (num): ending month for process with ONLY option
   RANGE    (char): range of months for process with ONLY option
```

FLOWn : one or more data subsets for the flow duration process, with just the months requested in the ONLY option (one is created each time the process is requested)

```
STATION  (char): station code number
STN_FORM (char): formatted version of station code number
NAME     (char): station name
COUNTY   (char): county in which station is located
AREA      (num): drainage area of station
WAT_YR    (num): water year (Oct. 1 - Sep. 30)
YEAR      (num): calendar year
MONTH     (num): month in which data were recorded
FLOW1-31  (num): array with daily streamflow values
COUNT     (num): number of class intervals specified (default=7)
CLASS1-30 (num): array with lower limits of class intervals
STN_REQ  (char): station number, if user requests different
                 class intervals for each station
```

HIGH : contains range of months requested in highest process

```
BEGMONTH (num): beginning month for process with ONLY option
ENDMONTH (num): ending month for process with ONLY option
RANGE   (char): range of months for process with ONLY option
```

HI_OCC : contains range of months and threshold requested in high occurrences process

```
BEGMONTH (num): beginning month for process with ONLY option
ENDMONTH (num): ending month for process with ONLY option
RANGE   (char): range of months for process with ONLY option
THRESHLD  (num): threshold level
STN_REQ  (char): station number, if user requests different
                 class intervals for each station
```

HI_OCCn : one or more data subsets for the high occurrences process, with the months requested in the ONLY option (one is created each time the process is requested)

```
STATION  (char): station code number
STN_FORM (char): formatted version of station code number
NAME     (char): station name
COUNTY   (char): county in which station is located
YEAR      (num): calendar year
MONTH     (num): month in which data were recorded
MONTHSUM  (num): monthly summary of codes
MAXTMP1-31(num): array with daily maximum temperatures
THRESHLD  (num): threshold level
STN_REQ  (char): station number, if user requests different
                 class intervals for each station
```

HO_SUBn : one or more data subsets for the high occurrences process (one is created each time the process is requested), with the number of days above the threshold, and the first and last dates of occurrence

STATION	(char):	station code number
STN_FORM	(char):	formatted version of station code number
NAME	(char):	station name
COUNTY	(char):	county in which station is located
YEAR	(num):	calendar year
THRESHLD	(num):	threshold level
ABOVE	(num):	number of days above threshold
FIRST_DY	(num):	first day of year above threshold
FIRST_MN	(num):	first month of year above threshold
LAST_DY	(num):	last day of year above threshold
LAST_MN	(num):	last month of year above threshold

HPCP : subset of permanent hourly precip file, after conversions

STATION	(char):	station code number
STN_FORM	(char):	formatted version of station code number
NAME	(char):	station name
COUNTY	(char):	county in which station is located
AREA	(num):	drainage area of station
ELEV	(num):	elevation of station
YEAR	(num):	calendar year
MONTH	(num):	month in which data were recorded
DAY	(num):	day in which data were recorded
HPCP1-24	(num):	array with hourly precipitation values
H_CODE1-24	(char):	array with hourly precipitation codes
HPCP_TOT	(num):	total daily precipitation
DAYSUM	(num):	daily summary of codes

HPCPMON : subset of hourly precip file, with daily records converted to monthly records

STATION	(char):	station code number
STN_FORM	(char):	formatted version of station code number
NAME	(char):	station name
COUNTY	(char):	county in which station is located
YEAR	(num):	calendar year
MONTH	(num):	month in which data were recorded
HPCPMON	(num):	total monthly precipitation
MONTHSUM	(num):	monthly summary of codes

LOW : contains range of months requested in lowest process

```
BEGMONTH (num): beginning month for process with ONLY option
ENDMONTH (num): ending month for process with ONLY option
RANGE   (char): range of months for process with ONLY option
```

LO_OCC : contains range of months and threshold requested in low occurrences process

```
BEGMONTH (num): beginning month for process with ONLY option
ENDMONTH (num): ending month for process with ONLY option
RANGE   (char): range of months for process with ONLY option
THRESHLD  (num): threshold level
STN_REQ  (char): station number, if user requests different
                 class intervals for each station
```

LO_OCCn : one or more data subsets for the low occurrences process, with the months requested in the ONLY option (one is created each time the process is requested)

```
STATION  (char): station code number
STN_FORM (char): formatted version of station code number
NAME     (char): station name
COUNTY   (char): county in which station is located
YEAR      (num): calendar year
MONTH     (num): month in which data were recorded
MONTHSUM  (num): monthly summary of codes
MINTMP1-31(num): array with daily minimum temperatures
THRESHLD  (num): threshold level
STN_REQ  (char): station number, if user requests different
                 class intervals for each station
```

LO_SUBn : one or more data subsets for the low occurrences process (one is created each time the process is requested), with the number of days below the threshold, and the first and last dates of occurrence

```
STATION  (char): station code number
STN_FORM (char): formatted version of station code number
NAME     (char): station name
COUNTY   (char): county in which station is located
YEAR      (num): calendar year
THRESHLD  (num): threshold level
BELOW     (num): number of days below threshold
FIRST_DY  (num): first day after July 31 below threshold
FIRST_MN  (num): first month after July 31 below threshold
LAST_DY   (num): last day before July 31 below threshold
LAST_MN   (num): last month before July 31 below threshold
```

MAXn : contains maximum monthly values for highest process

```
STATION  (char): station code number
STN_FORM (char): formatted version of station code number
NAME     (char): station name
COUNTY   (char): county in which station is located
YEAR      (num): calendar year
MONTH     (num): month in which data were recorded
MONTHSUM  (num): monthly summary of codes
MAXVALUE or MAXVAL1 (num): maximum monthly data item
MAXVAL2  (num): maximum monthly data item of second value for
                elements with two values per day
BEGMONTH (num): beginning month for process with ONLY option
ENDMONTH (num): ending month for process with ONLY option
RANGE   (char): range of months for process with ONLY option
```

MINn : contains minimum monthly values for lowest process

```
STATION  (char): station code number
STN_FORM (char): formatted version of station code number
NAME     (char): station name
COUNTY   (char): county in which station is located
YEAR      (num): calendar year
MONTH     (num): month in which data were recorded
MONTHSUM  (num): monthly summary of codes
MINVALUE or MINVAL1 (num): minimum monthly data item
MINVAL2  (num): minimum monthly data item of second value for
                elements with two values per day
BEGMONTH (num): beginning month for process with ONLY option
ENDMONTH (num): ending month for process with ONLY option
RANGE   (char): range of months for process with ONLY option
```

MNTH : subset of permanent monthly summary file after conversions

```
STATION  (char): station code number
STN_FORM (char): formatted version of station code number
NAME     (char): station name
COUNTY   (char): county in which station is located
AREA      (num): drainage area of station
ELEV      (num): elevation of station
YEAR      (num): calendar year
MONTH     (num): month in which data were recorded
MAXTEMP   (num): mean maximum monthly temperature
MINTEMP   (num): mean minimum monthly temperature
```

```
   TMEAN     (num): mean monthly temperature
   TDEPART   (num): departure from normal temperature
   DEGDAYS   (num): heating degree days
   COOLDAYS  (num): cooling degree days
   HIGHEST   (num): highest temperature during month
   HIGHDAY   (num): day of occurrence of highest temperature
   LOWEST    (num): lowest temperature during month
   LOWDAY    (num): day of occurrence of lowest temperature
   PRECIP    (num): total monthly precipitation
   PDEPART   (num): departure from normal precipitation
   PPTMAX    (num): maximum daily precipitation during month
   PMAXDAY   (num): day of occurrence of maximum precipitation
   SNOFALL   (num): total snowfall during month
   SNODEPTH  (num): maximum depth of snow on ground during month
   SMAXDAY   (num): day of occurrence of maximum snow depth
   WIND      (num): total wind movement during month
   EVAP      (num): total evaporation during month
   PLUS1-5  (char): array of codes for monthly values

MN_OCC : contains data from the ONLY and THRESHOLD options with
         the monthly occurrences process

   BEGMONTH (num): beginning month for process with ONLY option
   ENDMONTH (num): ending month for process with ONLY option
   RANGE   (char): range of months for process with ONLY option
   COUNT     (num): number of thresholds specified (default=1)
   THRES1-10 (num): array with threshold levels
   STN_REQ  (char): station number, if user requests different
                    thresholds for each station


MN_OCCn : one or more data sets (one is created each time the
          process is requested) with monthly records for
          monthly occurrences process

   STATION  (char): station code number
   STN_FORM (char): formatted version of station code number
   NAME     (char): station name
   COUNTY   (char): county in which station is located
   YEAR      (num): calendar year
   MONTH     (num): month in which data were recorded
   MONTHSUM  (num): monthly code summary
   &MON_VAL  (num): total monthly value
   COUNT     (num): number of thresholds specified (default=1)
   THRES1-10 (num): array with threshold levels
   STN_REQ  (char): station number, if user requests different
                    thresholds for each station
```

MN_SUBn : one or more data sets (one is created each time the process is requested) with calculations for the monthly occurrences process

STATION	(char):	station code number
STN_FORM	(char):	formatted version of station code number
NAME	(char):	station name
COUNTY	(char):	county in which station is located
YEAR	(num):	calendar year
MONTH	(num):	month in which data were recorded
COUNT	(num):	number of thresholds specified (default=1)
THRES1-10	(num):	array with threshold levels
ABOVE1-10	(num):	number of months above each threshold

MSTATS : contains range of months requested in monthly statistics

BEGMONTH	(num):	beginning month for process with ONLY option
ENDMONTH	(num):	ending month for process with ONLY option
RANGE	(char):	range of months for process with ONLY option

ORDERn : contains data for the rank ordering process, with only the top ranked data values for output. (5 * # of years)

STATION	(char):	station code number
STN_FORM	(char):	formatted version of station code number
NAME	(char):	station name
COUNTY	(char):	county in which station is located
YEAR	(num):	calendar year
MONTH	(num):	month in which data were recorded
DAY	(num):	day in which data were recorded
DAYVALUE	(num):	daily value
BEGMONTH	(num):	beginning month for process with ONLY option
ENDMONTH	(num):	ending month for process with ONLY option
RANGE	(char):	range of months for process with ONLY option

OUTLINE : outline of requested dates for calendar process

STATION	(char):	station code number
MONTH	(num):	month in which data were recorded
DAY	(num):	day in which data were recorded
MONDAYYR	(num):	month, day, and year as SAS date value

PEAK : subset of permanent peak flow file, after conversions

STATION	(char):	station code number
STN_FORM	(char):	formatted version of station code number
NAME	(char):	station name
COUNTY	(char):	county in which station is located

```
   AREA       (num): drainage area of station
   ELEV       (num): elevation of station
   WAT_YEAR   (num): water year (Oct. 1 - Sep. 30)
   YEAR       (num): calendar year
   MONTH      (num): month in which data were recorded
   DAY        (num): day in which data were recorded
   GAGE_HT    (num): gage height
   PEAKFLOW   (num): annual peak flow
   GH_CODE   (char): code for gage height
   PK_CODE   (char): code for peak flow
   RD_CODE   (char): code for regulation and diversion
```

PERIODS : requested periods of record from PERIOD command

```
   BEGDATE (num): beginning year * 100 + month
   ENDDATE (num): ending year * 100 + month
```

P_NORM : contains precipitation normals for calendar process

```
   STATION   (char): station code number
   STN_FORM  (char): formatted version of station code number
   NAME      (char): station name
   COUNTY    (char): county in which station is located
   MONTH      (num): month in which data were recorded
   DAY        (num): day in which data were recorded
   PRCPNORM   (num): long-term normal precipitation for month/day
   MAXPRCP    (num): maximum precipitation recorded for month/day
```

P_SUB : contains subset of precipitation data with the months specified in the ONLY option of the calendar process

```
   STATION   (char): station code number
   STN_FORM  (char): formatted version of station code number
   NAME      (char): station name
   COUNTY    (char): county in which station is located
   YEAR       (num): calendar year
   MONTH      (num): month in which data were recorded
   MONTHSUM   (num): monthly summary of codes
   PRECIP1-31(num): array of daily precipitation values
```

POINTERS : contains pointers given by user in POINTERS command

```
   FOBS_PTR (num): first observation number for a station
   LOBS_PTR (num): last observation number for a station
```

PRCP : subset of permanent precipitation file, after conversions

```
STATION  (char): station code number
STN_FORM (char): formatted version of station code number
NAME     (char): station name
COUNTY   (char): county in which station is located
AREA      (num): drainage area of station
ELEV      (num): elevation of station
YEAR      (num): calendar year
MONTH     (num): month in which data were recorded
PRECIP1-31(num): array with daily precipitation values
P_CODE1-31(char): array with daily precipitation codes
PRCP_TOT  (num): total monthly precipitation
MONTHSUM  (num): monthly summary of codes
```

PROCESS : contains the process commands given by the user

```
PRC_NAME (char): the name of the requested process
PARMETER (char): the options given for the process
```

RANKORD : contains range of months for rank order process

```
BEGMONTH (num): beginning month for process with ONLY option
ENDMONTH (num): ending month for process with ONLY option
RANGE   (char): range of months for process with ONLY option
```

RANKn : contains observations with one daily value per obs,
before sorting for rank ordering process

same variables as ORDERn

REQ_IX : contains a subset of the permanent index data set

same variables as NHIMS.INDEX (see section 5.2)

RESV : subset of permanent reservoir file, after conversions

```
STATION  (char): station code number
STN_FORM (char): formatted version of station code number
NAME     (char): station name
COUNTY   (char): county in which station is located
AREA      (num): drainage area of station
ELEV      (num): elevation of station
YEAR      (num): calendar year
```

```
   MONTH      (num): month in which data were recorded
   STORGE1-31(num): array with daily reservoir storage values
   UNITCODE (char): code for units for daily values
   TIMECODE  (num): code for time of day when data were recorded
```

SNOC : subset of permanent snow course file, after conversions

```
   STATION  (char): station code number
   STN_FORM (char): formatted version of station code number
   NAME     (char): station name
   COUNTY   (char): county in which station is located
   AREA      (num): drainage area of station
   ELEV      (num): elevation of station
   CARDNO    (num): code identifying when measurements were taken
   YEAR      (num): calendar year
   MONTH1-6  (num): array of months in which readings were taken
   DAY1-6    (num): array of days in which readings were taken
   DEPTH1-6  (num): array with snow depth values
   WATER1-6  (num): array with water equivalent values
```

SNOW : subset of permanent snowfall file, after conversions

```
   STATION  (char): station code number
   STN_FORM (char): formatted version of station code number
   NAME     (char): station name
   COUNTY   (char): county in which station is located
   AREA      (num): drainage area of station
   ELEV      (num): elevation of station
   YEAR      (num): calendar year
   MONTH     (num): month in which data were recorded
   SNOW1-31  (num): array with daily snowfall values
   S_CODE1-31(char): array with daily snowfall codes
   SNOW1-31  (num): array with daily snow depth values
   D_CODE1-31(char): array with daily snow depth codes
   SNOW_TOT  (num): total monthly snow fall
   MONTHSUM  (num): monthly summary of codes
```

SN_OCCn : one or more data sets (one is created each time the process is requested) with monthly records for snow occurrences process

```
   STATION  (char): station code number
   STN_FORM (char): formatted version of station code number
   NAME     (char): station name
   COUNTY   (char): county in which station is located
   SNOW_YR   (num): snow season year (Sep. 1 - Aug. 31)
   YEAR      (num): calendar year
   MONTH     (num): month in which data were recorded
```

```
   MONTHSUM  (num): monthly code summary
   SNOW1-31  (num): array of daily snowfall values
   DEPTH1-31 (num): array of daily snow depth values
   D_CODE1-31(char): array of daily snow depth codes
   THRESHLD  (num): threshold level
   STN_REQ  (char): station number, if user requests different
                    thresholds for each station
```

SN_SUBn : one or more data sets (one is created each time the process is requested) with calculations for the snow occurrences process

```
   STATION  (char): station code number
   STN_FORM (char): formatted version of station code number
   NAME     (char): station name
   COUNTY   (char): county in which station is located
   SNOW_YR   (num): snow season year (Sep. 1 - Aug. 31)
   YEAR      (num): calendar year
   MONTH     (num): month in which data were recorded
   THRESHLD  (num): threshold level
   ABOVE     (num): number of days above threshold
   FIRST_DY  (num): first day of year above threshold
   FIRST_MN  (num): first month of year above threshold
   LAST_DY   (num): last day of year above threshold
   LAST_MN   (num): last month of year above threshold
   DAYONE1-4 (num): array of snow depths on first day of 4 months
```

STATIONS : the merged station subsets, with the requested station numbers

```
   STATION  (char): station code number
```

STATIONn : one or more data sets (one more is created for each AND command) with requested station numbers

```
   STATION  (char): station code number
```

STRM : subset of permanent streamflow file, after conversions

```
   STATION  (char): station code number
   STN_FORM (char): formatted version of station code number
   NAME     (char): station name
   COUNTY   (char): county in which station is located
   AREA      (num): drainage area of station
   ELEV      (num): elevation of station
   YEAR      (num): calendar year
```

```
   MONTH     (num): month in which data were recorded
   STRM1-31  (num): array with daily streamflow values
   STRM_TOT  (num): total monthly streamflow
   MONTHSUM  (num): monthly summary of codes
```

SUB_IX : subset of the permanent index file

same variables as NHIMS.INDEX (see section 5.2)

SUBSETn : data used in the correlation process, with the daily values of each station in a separate data set

```
   YEAR     (num): calendar year
   MONTH    (num): month in which data were recorded
   DAY      (num): day during which data were recorded
   STNn     (num): daily data value for one station
```

SUMMARY : contains range of months requested for summary process

```
   BEGMONTH (num): beginning month for process with ONLY option
   ENDMONTH (num): ending month for process with ONLY option
   RANGE   (char): range of months for process with ONLY option
```

SUMMn : subset of the monthly summary file, with the requested months from the ONLY option

```
   BEGMONTH (num): beginning month for process with ONLY option
   ENDMONTH (num): ending month for process with ONLY option
   RANGE   (char): range of months for process with ONLY option
   all the variables from MNTH (see above)
```

TEMP : subset of permanent temperature file, after conversions

```
   STATION  (char): station code number
   STN_FORM (char): formatted version of station code number
   NAME     (char): station name
   COUNTY   (char): county in which station is located
   AREA      (num): drainage area of station
   ELEV      (num): elevation of station
   YEAR      (num): calendar year
   MONTH     (num): month in which data were recorded
   MAXTMP1-31(num): array with daily maximum temperatures
   MINTMP1-31(num): array with daily minimum temperatures
   MXCODE1-31(char): array with daily codes for maximum temperature
```

```
   MNCODE1-31(char): array with daily codes for minimum temperature
   AVEMAX    (num): mean monthly maximum temperature
   AVEMIN    (num): mean monthly minimum temperature
   MONTHSUM  (num): monthly summary of codes
```

T_NORM : contains temperature normals for calendar process

```
   STATION  (char): station code number
   STN_FORM (char): formatted version of station code number
   NAME     (char): station name
   COUNTY   (char): county in which station is located
   MONTH     (num): month in which data were recorded
   DAY       (num): day in which data were recorded
   MAXNORM   (num): long-term normal max temperature for month/day
   MINNORM   (num): long-term normal min temperature for month/day
   MAXIMUM   (num): extreme max temperature recorded for month/day
   MINIMUM   (num): extreme min temperature recorded for month/day
```

T_SUB : contains subset of temperature data with the months
specified in the ONLY option of the calendar process

```
   STATION  (char): station code number
   STN_FORM (char): formatted version of station code number
   NAME     (char): station name
   COUNTY   (char): county in which station is located
   YEAR      (num): calendar year
   MONTH     (num): month in which data were recorded
   MONTHSUM  (num): monthly summary of codes
   MAXTMP1-31(num): array of daily maximum temperatures
   MINTMP1-31(num): array of daily minimum temperatures
```

USRCARDS : the commands submitted by the user

```
   RECORD (char): one 80-byte record from user command file
```

WEATHER : combination of the temperature and precipitation
normals and extremes for calendar process

```
   STATION  (char): station code number
   STN_FORM (char): formatted version of station code number
   NAME     (char): station name
   COUNTY   (char): county in which station is located
   MONTH     (num): month in which data were recorded
   DAY       (num): day in which data were recorded
   MAXNORM   (num): long-term normal max temperature for month/day
   MINNORM   (num): long-term normal min temperature for month/day
   MAXIMUM   (num): extreme max temperature recorded for month/day
```

MINIMUM (num): extreme min temperature recorded for month/day
PRCPNORM (num): long-term normal precipitation for month/day
MAXPRCP (num): maximum precipitation recorded for month/day
NORM (num): dummy variable used for daily calendar heading

YEARS : contains number of years in the period of record to be used in rank ordering process

STATION (char): station code number
NUM_YRS (num): the number of years in requested period of record

5.1.4

CROSS REFERENCE MATRIX
MODULES VS. TEMPORARY SAS DATA SETS

TABLE OF MODULE BY DATASET

MODULE DATASET

Frequency	ACCESSn	CALDATES	CAL_DATA	CLASS	CORR
C_CAL (4.129)	0	1	1	0	0
C_DOCC (4.118)	0	0	0	0	0
C_FLOW (4.123)	0	0	0	1	0
C_NORMS (4.128)	0	1	0	0	0
C_SOCC (4.119)	0	0	0	0	0
DAY_SUB (4.93)	0	0	0	0	0
GET_CLS (4.122)	0	0	0	1	0
GET_DAT (4.127)	0	1	0	0	0
GET_MMT (4.117)	0	0	0	0	0
GET_MON (4.92)	0	0	0	0	1
HEADER (4.8)	1	0	0	0	0
MAKESUB (4.99)	0	0	0	0	1
PARSE (4.9)	1	0	0	0	0
SUBSETS (4.5)	1	0	0	0	0
Total	3	3	1	2	2

(Continued)

CROSS REFERENCE MATRIX
MODULES VS. TEMPORARY SAS DATA SETS

TABLE OF MODULE BY DATASET

MODULE DATASET

Frequency	DSTATS	DURn	DY_OCC	DY_OCCn	DY_SUBn
C_CAL (4.129)	0	0	0	0	0
C_DOCC (4.118)	0	0	1	1	1
C_FLOW (4.123)	0	1	0	0	0
C_NORMS (4.128)	0	0	0	0	0
C_SOCC (4.119)	0	0	1	0	0
DAY_SUB (4.93)	1	0	0	0	0
GET_CLS (4.122)	0	0	0	0	0
GET_DAT (4.127)	0	0	0	0	0
GET_MMT (4.117)	0	0	1	0	0
GET_MON (4.92)	1	0	0	0	0
HEADER (4.8)	0	0	0	0	0
MAKESUB (4.99)	0	0	0	0	0
PARSE (4.9)	0	0	0	0	0
SUBSETS (4.5)	0	0	0	0	0
Total	2	1	3	1	1

(Continued)

CROSS REFERENCE MATRIX
MODULES VS. TEMPORARY SAS DATA SETS

TABLE OF MODULE BY DATASET

MODULE DATASET

Frequency	&EL.ALL	&EL.ANN	&EL.BOTH	&EL.CAN2	&EL.CANN
CV_EVAP (4.32)	0	0	0	0	0
CV_HPCP (4.37)	0	0	0	0	0
CV_MNTH (4.36)	0	0	0	0	0
CV_PEAK (4.35)	0	0	0	0	0
CV_PRCP (4.28)	0	0	0	0	0
CV_RESV (4.33)	0	0	0	0	0
CV_SNOC (4.34)	0	0	0	0	0
CV_SNOW (4.31)	0	0	0	0	0
CV_STRM (4.30)	0	0	0	0	0
CV_TEMP (4.29)	0	0	0	0	0
C_CORR (4.100)	0	0	0	0	0
C_DSTAT (4.94)	1	1	1	1	1
C_MSTAT (4.97)	1	1	1	1	1
DAY_SUB (4.93)	0	0	0	0	0
GETDATA (4.26)	0	0	0	0	0
GET_ANN (4.27)	0	0	0	0	0
MAKESUB (4.99)	0	0	0	0	0
MEAN_A2 (4.64)	0	0	0	0	0
MEAN_T1 (4.62)	0	0	0	0	0
MEAN_T2 (4.63)	0	0	0	0	0
MON_SUB (4.96)	0	0	0	0	0
Total	2	2	2	2	2

(Continued)

CROSS REFERENCE MATRIX
MODULES VS. TEMPORARY SAS DATA SETS

TABLE OF MODULE BY DATASET

MODULE	DATASET				
Frequency	&EL.CNEW	&EL.COMB	&EL.COMP	&EL.CORR	&EL.CSUB
CV_EVAP (4.32)	0	0	0	0	0
CV_HPCP (4.37)	0	0	0	0	0
CV_MNTH (4.36)	0	0	0	0	0
CV_PEAK (4.35)	0	0	0	0	0
CV_PRCP (4.28)	0	0	0	0	0
CV_RESV (4.33)	0	0	0	0	0
CV_SNOC (4.34)	0	0	0	0	0
CV_SNOW (4.31)	0	0	0	0	0
CV_STRM (4.30)	0	0	0	0	0
CV_TEMP (4.29)	0	0	0	0	0
C_CORR (4.100)	0	1	0	1	0
C_DSTAT (4.94)	1	0	0	0	1
C_MSTAT (4.97)	1	0	0	0	1
DAY_SUB (4.93)	0	0	0	0	0
GETDATA (4.26)	0	0	0	0	0
GET_ANN (4.27)	0	0	0	0	0
MAKESUB (4.99)	0	0	0	1	0
MEAN_A2 (4.64)	0	0	1	0	0
MEAN_T1 (4.62)	0	0	1	0	0
MEAN_T2 (4.63)	0	0	1	0	0
MON_SUB (4.96)	0	0	0	0	0
Total	2	1	3	2	2

(Continued)

CROSS REFERENCE MATRIX
MODULES VS. TEMPORARY SAS DATA SETS

TABLE OF MODULE BY DATASET

MODULE DATASET

Frequency	&EL.DATA	&EL.MANN	&EL.MEAN	&EL.MON
CV_EVAP (4.32)	1	0	0	0
CV_HPCP (4.37)	1	0	0	0
CV_MNTH (4.36)	1	0	0	0
CV_PEAK (4.35)	1	0	0	0
CV_PRCP (4.28)	1	0	0	0
CV_RESV (4.33)	1	0	0	0
CV_SNOC (4.34)	1	0	0	0
CV_SNOW (4.31)	1	0	0	0
CV_STRM (4.30)	1	0	0	0
CV_TEMP (4.29)	1	0	0	0
C_CORR (4.100)	0	0	0	0
C_DSTAT (4.94)	0	1	0	0
C_MSTAT (4.97)	0	1	0	0
DAY_SUB (4.93)	0	0	0	1
GETDATA (4.26)	1	0	0	0
GET_ANN (4.27)	1	0	0	0
MAKESUB (4.99)	0	0	0	0
MEAN_A2 (4.64)	0	0	1	0
MEAN_T1 (4.62)	0	0	1	0
MEAN_T2 (4.63)	0	0	1	0
MON_SUB (4.96)	0	0	0	1
Total	12	2	3	2

(Continued)

CROSS REFERENCE MATRIX
MODULES VS. TEMPORARY SAS DATA SETS

TABLE OF MODULE BY DATASET

MODULE DATASET

Frequency	&EL.MSUB	&EL.PER	&EL.PTIX	&EL.PTRS	&EL.STAT
C_CORR (4.100)	0	0	1	0	0
C_DSTAT (4.94)	1	0	0	0	1
C_MSTAT (4.97)	1	0	0	0	1
DAY_SUB (4.93)	0	0	0	0	1
FINDPER (4.41)	0	1	0	1	0
FINDPTR (4.24)	0	0	0	1	0
GETDATA (4.26)	0	0	1	0	0
GET_ANN (4.27)	0	0	1	0	0
IX_MRGE (4.42)	0	1	0	0	0
LP_DATA (4.48)	0	0	0	1	0
MON_SUB (4.96)	0	0	0	0	1
PTR_INX (4.25)	0	0	1	1	0
PTR_STA (4.23)	0	0	0	1	0
Total	2	2	4	5	4

(Continued)

CROSS REFERENCE MATRIX
MODULES VS. TEMPORARY SAS DATA SETS

TABLE OF MODULE BY DATASET

MODULE	DATASET				
Frequency	EVAP	EXTREM	EXTREMn	FLOWn	HIGH
CD_SAS (4.78)	1	0	0	0	0
CON_DAY (4.65)	0	0	0	0	0
CP_EVAP (4.84)	1	0	0	0	0
CP_HPCP (4.89)	0	0	0	0	0
CV_EVAP (4.32)	1	0	0	0	0
CV_HPCP (4.37)	0	0	0	0	0
C_FLOW (4.123)	0	0	0	1	0
C_HIOCC (4.113)	0	0	0	0	0
DAY_SUB (4.93)	1	0	0	0	0
GET_M1T (4.112)	0	0	0	0	0
GET_MON (4.92)	0	1	0	0	1
HI_LO_1 (4.103)	0	0	0	0	1
HI_LO_2 (4.104)	1	0	0	0	1
LD_EVAP (4.54)	1	0	0	0	0
LD_HPCP (4.56)	0	0	0	0	0
LH_HPCP (4.75)	0	0	0	0	0
LM_TOT1 (4.66)	0	0	0	0	0
LM_TOT2 (4.67)	1	0	0	0	0
MAKESUB (4.99)	1	0	0	0	0
MEAN_T1 (4.62)	0	0	0	0	0
MEAN_T2 (4.63)	1	0	0	0	0
MON_SUB (4.96)	1	0	0	0	0
PRT_EX (4.106)	0	1	1	0	0
Total	10	2	1	1	3

(Continued)

CROSS REFERENCE MATRIX
MODULES VS. TEMPORARY SAS DATA SETS

TABLE OF MODULE BY DATASET

MODULE / Frequency	HI_OCC	HI_OCCn	HO_SUBn	HPCP	HPCPMON
CD_SAS (4.78)	0	0	0	1	0
CON_DAY (4.65)	0	0	0	1	1
CP_EVAP (4.84)	0	0	0	0	0
CP_HPCP (4.89)	0	0	0	1	0
CV_EVAP (4.32)	0	0	0	0	0
CV_HPCP (4.37)	0	0	0	1	0
C_FLOW (4.123)	0	0	0	0	0
C_HIOCC (4.113)	1	1	1	0	0
DAY_SUB (4.93)	0	0	0	0	0
GET_M1T (4.112)	1	0	0	0	0
GET_MON (4.92)	0	0	0	0	0
HI_LO_1 (4.103)	0	0	0	0	0
HI_LO_2 (4.104)	0	0	0	0	0
LD_EVAP (4.54)	0	0	0	0	0
LD_HPCP (4.56)	0	0	0	1	0
LH_HPCP (4.75)	0	0	0	1	0
LM_TOT1 (4.66)	0	0	0	0	1
LM_TOT2 (4.67)	0	0	0	0	0
MAKESUB (4.99)	0	0	0	0	0
MEAN_T1 (4.62)	0	0	0	1	0
MEAN_T2 (4.63)	0	0	0	0	0
MON_SUB (4.96)	0	0	0	0	0
PRT_EX (4.106)	0	0	0	0	0
Total	2	1	1	7	2

(Continued)

CROSS REFERENCE MATRIX
MODULES VS. TEMPORARY SAS DATA SETS

TABLE OF MODULE BY DATASET

MODULE DATASET

Frequency	LOW	LO_OCC	LO_OCCn	LO_SUBn	MAXn	MINn
CD_SAS (4.78)	0	0	0	0	0	0
CP_MNTH (4.86)	0	0	0	0	0	0
CV_MNTH (4.36)	0	0	0	0	0	0
C_LOOCC (4.114)	0	1	1	1	0	0
C_MOCC (4.120)	0	0	0	0	0	0
C_SUMM (4.125)	0	0	0	0	0	0
GET_M1T (4.112)	0	1	0	0	0	0
GET_MMT (4.117)	0	0	0	0	0	0
GET_MON (4.92)	1	0	0	0	0	0
HI_LO_1 (4.103)	1	0	0	0	1	1
HI_LO_2 (4.104)	1	0	0	0	1	1
LC_MNTH (4.73)	0	0	0	0	0	0
MON_SUB (4.96)	0	0	0	0	0	0
Total	3	2	1	1	2	2

(Continued)

CROSS REFERENCE MATRIX
MODULES VS. TEMPORARY SAS DATA SETS

TABLE OF MODULE BY DATASET

MODULE DATASET

Frequency	MNTH	MN_OCC	MN_OCCn	MN_SUBn	MSTATS
CD_SAS (4.78)	1	0	0	0	0
CP_MNTH (4.86)	1	0	0	0	0
CV_MNTH (4.36)	1	0	0	0	0
C_LOOCC (4.114)	0	0	0	0	0
C_MOCC (4.120)	0	1	1	1	0
C_SUMM (4.125)	1	0	0	0	0
GET_M1T (4.112)	0	0	0	0	0
GET_MMT (4.117)	0	1	0	0	0
GET_MON (4.92)	0	0	0	0	1
HI_LO_1 (4.103)	0	0	0	0	0
HI_LO_2 (4.104)	0	0	0	0	0
LC_MNTH (4.73)	1	0	0	0	0
MON_SUB (4.96)	0	0	0	0	1
Total	5	2	1	1	2

(Continued)

CROSS REFERENCE MATRIX
MODULES VS. TEMPORARY SAS DATA SETS

TABLE OF MODULE BY DATASET

MODULE DATASET

Frequency	ORDERn	OUTLINE	PEAK	PERIODS	P_NORM	P_SUB
CD_SAS (4.78)	0	0	1	0	0	0
CP_PEAK (4.87)	0	0	1	0	0	0
CV_PEAK (4.35)	0	0	1	0	0	0
C_CAL (4.129)	0	1	0	0	1	0
C_NORMS (4.128)	0	0	0	0	1	1
C_RANK (4.109)	1	0	0	0	0	0
GETDATA (4.26)	0	0	0	1	0	0
GET_ANN (4.27)	0	0	0	1	0	0
LC_PEAK (4.70)	0	0	1	0	0	0
PARSE (4.9)	0	0	0	1	0	0
PERIOD (4.12)	0	0	0	1	0	0
Total	1	1	4	4	2	1

(Continued)

CROSS REFERENCE MATRIX
MODULES VS. TEMPORARY SAS DATA SETS

TABLE OF MODULE BY DATASET

MODULE DATASET

Frequency	POINTERS	PRCP	PROCESS
CAL_YRS (4.108)	0	1	0
CD_SAS (4.78)	0	1	0
CP_PRCP (4.80)	0	1	0
CV_PRCP (4.28)	0	1	0
C_DOCC (4.118)	0	1	0
C_MOCC (4.120)	0	1	0
C_NORMS (4.128)	0	1	0
C_RANK (4.109)	0	1	0
DAY_SUB (4.93)	0	1	0
GET_CLS (4.122)	0	0	1
GET_DAT (4.127)	0	0	1
GET_M1T (4.112)	0	0	1
GET_MMT (4.117)	0	0	1
GET_MON (4.92)	0	0	1
HI_LO_1 (4.103)	0	1	0
LD_PRCP (4.50)	0	1	0
LM_TOT1 (4.66)	0	1	0
MAKESUB (4.99)	0	1	0
MEAN_T1 (4.62)	0	1	0
MON_SUB (4.96)	0	1	0
PARSE (4.9)	1	0	1
PTR_STA (4.23)	1	0	0
ST_PROC (4.15)	0	0	1
Total	2	15	7

(Continued)

CROSS REFERENCE MATRIX
MODULES VS. TEMPORARY SAS DATA SETS

TABLE OF MODULE BY DATASET

MODULE DATASET

Frequency	RANKORD	RANKn	REQ_IX	RESV
CD_SAS (4.78)	0	0	0	1
CP_RESV (4.88)	0	0	0	1
CV_RESV (4.33)	0	0	0	1
C_RANK (4.109)	0	1	0	0
GET_MON (4.92)	1	0	0	0
IX_MRGE (4.42)	0	0	1	0
LD_RESV (4.55)	0	0	0	1
LI_PRT (4.43)	0	0	1	0
Total	1	1	2	4

(Continued)

CROSS REFERENCE MATRIX
MODULES VS. TEMPORARY SAS DATA SETS

TABLE OF MODULE BY DATASET

MODULE DATASET

Frequency	SNOC	SNOW	SN_OCCn	SN_SUBn
CAL_YRS (4.108)	0	1	0	0
CD_SAS (4.78)	1	1	0	0
CP_SNOC (4.85)	1	0	0	0
CP_SNOW (4.83)	0	1	0	0
CV_SNOC (4.34)	1	0	0	0
CV_SNOW (4.31)	0	1	0	0
C_MOCC (4.120)	0	1	0	0
C_RANK (4.109)	0	1	0	0
C_SOCC (4.119)	0	1	1	1
DAY_SUB (4.93)	0	1	0	0
HI_LO_2 (4.104)	0	1	0	0
LC_SNOC (4.71)	1	0	0	0
LD_SNOW (4.53)	0	1	0	0
LM_TOT1 (4.66)	0	1	0	0
MAKESUB (4.99)	0	1	0	0
MEAN_T1 (4.62)	0	1	0	0
MON_SUB (4.96)	0	1	0	0
Total	4	14	1	1

(Continued)

CROSS REFERENCE MATRIX
MODULES VS. TEMPORARY SAS DATA SETS

TABLE OF MODULE BY DATASET

MODULE DATASET

Frequency	STATIONS	STATIONn	STRM	SUB_IX
CAL_YRS (4.108)	0	0	1	0
CD_SAS (4.78)	0	0	1	0
CHOOSE (4.19)	0	1	0	0
CP_STRM (4.82)	0	0	1	0
CV_STRM (4.30)	0	0	1	0
C_CAL (4.129)	1	0	0	0
C_FLOW (4.123)	0	0	1	0
C_RANK (4.109)	0	0	1	0
DAY_SUB (4.93)	0	0	1	0
FINDPTR (4.24)	1	0	0	0
HI_LO_1 (4.103)	0	0	1	0
INX_SUB (4.46)	0	0	0	1
IX_MRGE (4.42)	1	0	0	0
LD_STRM (4.52)	0	0	1	0
LI_PRT (4.43)	0	0	0	1
LM_TOT1 (4.66)	0	0	1	0
MAKESUB (4.99)	0	0	1	0
MEAN_T1 (4.62)	0	0	1	0
MON_SUB (4.96)	0	0	1	0
PARSE (4.9)	0	1	0	0
PTR_STA (4.23)	1	0	0	0
SRT_MRG (4.21)	1	1	0	0
STATION (4.11)	0	1	0	0
Total	5	4	13	2

(Continued)

CROSS REFERENCE MATRIX
MODULES VS. TEMPORARY SAS DATA SETS

TABLE OF MODULE BY DATASET

MODULE / DATASET	SUBSETn	SUMMARY	SUMMn	TEMP	T_NORM	T_SUB
Frequency						
CAL_YRS (4.108)	0	0	0	1	0	0
CD_SAS (4.78)	0	0	0	1	0	0
CP_TEMP (4.81)	0	0	0	1	0	0
CV_TEMP (4.29)	0	0	0	1	0	0
C_CAL (4.129)	0	0	0	0	1	0
C_CORR (4.100)	1	0	0	0	0	0
C_DOCC (4.118)	0	0	0	1	0	0
C_HIOCC (4.113)	0	0	0	1	0	0
C_LOOCC (4.114)	0	0	0	1	0	0
C_NORMS (4.128)	0	0	0	1	1	1
C_RANK (4.109)	0	0	0	1	0	0
C_SUMM (4.125)	0	1	1	0	0	0
DAY_SUB (4.93)	0	0	0	1	0	0
GET_MON (4.92)	0	1	0	0	0	0
HI_LO_2 (4.104)	0	0	0	1	0	0
LD_TEMP (4.51)	0	0	0	1	0	0
LM_AVE2 (4.68)	0	0	0	1	0	0
MAKESUB (4.99)	0	0	0	1	0	0
MEAN_A2 (4.64)	0	0	0	1	0	0
MON_SUB (4.96)	0	0	0	1	0	0
PRT_EX (4.106)	0	0	0	1	0	0
Total	1	2	1	17	2	1

(Continued)

CROSS REFERENCE MATRIX
MODULES VS. TEMPORARY SAS DATA SETS

TABLE OF MODULE BY DATASET

MODULE DATASET

Frequency	USRCARDS	WEATHER	YEARS
CAL_YRS (4.108)	0	0	1
CNT_REQ (4.4)	1	0	0
C_CAL (4.129)	0	1	0
C_RANK (4.109)	0	0	1
SUBSETS (4.5)	1	0	0
Total	2	1	2

5.2 FILE DESCRIPTIONS

The external files to be managed by the Northwest Hydrologic Information Management System (NHIMS) are permanent SAS data sets. Each type of climatic or hydrologic element is stored in a separate file, and one record of a file usually contains one month's data for a particular weather station. All records are of fixed length, and 4 bytes of each record are reserved for internal use by the SAS system itself.

There are currently NHIMS files for nine different elements: precipitation, air temperature, stream flow, snowfall, evaporation, reservoir storage, snow course, peak flow, and hourly precipitation. In addition, three other files are included in the system. One is the monthly summary file, which contains a summary of the temperature, precipitation, snowfall, and evaporation data for each station-month. Another is the index file which contains additional information on all of the NHIMS stations, such as the station's name, county, elevation, latitude and longitude, etc. Also, a user-defined format library is available for use; the library contains additional SAS formats in load module form for use with NHIMS.

In the following file descriptions, each subsection begins with the name of a SAS data library, which contains all data sets relating to a particular element. The library name is used as the operand for the DSN parameter in the JCL DD statement. For example, to access precipitation data, first identify the proper library to the operating system:

```
//ddname  DD  DSN=NHIMS.PRECIP.DAILY,DISP=SHR
```

After each library name, all SAS data sets in the library are named and described. The SAS data set name is used in conjunction with the ddname to refer to a particular file within the SAS program. For example, to access the precipitation data, use the data set name MAINDATA in a SET statement:

```
SET ddname.MAINDATA;
```

Once a data set has been named in a SET statement, all values are accessed simply by using the individual variable names.

Most of the libraries contain two data sets: one with the main climatic or hydrologic data (MAINDATA), and one with ranges of observation or record numbers (POINTERS) which can be used with the POINT option of the SET statement in order to directly access values from the main data file [7, p. 295], [17, p. 133]. The POINTERS data set for each main file identifies the beginning and ending record numbers where each station's data can be found; the number of records in the POINTERS file equals the number of unique stations in the main file.

Each data set description has additional notes which name the units of all numerical values, explain any codes, and identify which numbers must be adjusted on input, because some numerical values are scaled up in order to be stored as integers. All illegal end-of-month values, such as February 31, are stored as missing. The PROC CONTENTS procedure can also be used to obtain a description of any of the data sets.

In Section 5.2.13, a cross reference matrix is provided to show which NHIMS macros access the permanent files. The permanent files are given in alphbetical order, and the names have been abbreviated in order to fit into the constraints of PROC FREQ, which was used to generate the matrix.

5.2.1 NHIMS.INDEX

A) SAS Data Set Name - INDEX.

Number of Records - 2,047.

Number of Bytes/Record - 142.

Total Length of File in Bytes - 290,674.

RECORD DESCRIPTION

FIELD	VARIABLE NAME	TYPE
Station code number	STATION	CHAR
Station name	NAME	CHAR
Station's county	COUNTY	CHAR
Division or region	DIVISION	NUM
Hydrologic unit code	HUCODE	NUM
Drainage area	AREA	NUM
Latitude of station	LAT	NUM
Longitude of station	LONG	NUM
Elevation of station	ELEV	NUM
Beginning date of printed records for station	REC_BEG	CHAR
Ending date of printed records for station	REC_END	CHAR
Flag for NHIMS station	F_NHIMS	CHAR
Flag for temperature data	F_TEMP	CHAR
Flag for precipitation data	F_PRCP	CHAR
Flag for snowfall data	F_SNOW	CHAR
Flag for streamflow data	F_STRM	CHAR

Flag for month summary data	F_MNTH	CHAR
Flag for divisional data	F_DIVN	CHAR
Flag for reservoir data	F_RESV	CHAR
Flag for peak flow data	F_PEAK	CHAR
Flag for snow course data	F_SNOC	CHAR
Flag for evaporation data	F_EVAP	CHAR
Flag for hourly precip data	F_HPCP	CHAR
Extra flag for new element	F_ELEM1	CHAR
Extra flag for new element	F_ELEM2	CHAR

NOTES

1. The units are square miles for drainage area and feet above sea level for elevation.

2. The beginning and ending dates for printed records refer to the data available at the State Climate Office, whether on paper, microfiche, or computer format of some type; these records are not necessarily available from the NHIMS data base. The values are in the form MM-YYYY, where MM is the month and YYYY is the year. If the month is not known, zeroes are stored. If the entire date is unknown, the word UNKNOWN is stored in place of the date. The word ACTIVE is stored in place of the ending date for active stations.

3. The flags (all variables that begin with F_) identify which element files contain data for the current station. The F_NHIMS flag tells whether the station has data in any of the NHIMS files, while the other flags pertain to one particular file. There are some stations in the index which have no data in NHIMS. The values of the flag variables will be either 'N' for no data or 'Y' for yes, data are available for that element.

5.2.2 NHIMS.PRECIP.DAILY

A) SAS Data Set Name - MAINDATA.

Number of Records - 81,498.

Number of Bytes/Record - 144.

Total Length of File in Bytes - 11,735,712.

RECORD DESCRIPTION

FIELD	VARIABLE NAME	TYPE
Station code number	STATION	CHAR
Calendar year	YEAR	NUM
Month	MONTH	NUM
Monthly summary of codes	MONTHSUM	NUM
Monthly total precipitation X 100	PRCP_TOT	NUM
Remarks code for daily precipitation amount	P_CODE1-P_CODE31	CHAR
Daily precipitation amount X 100	PRECIP1-PRECIP31	NUM

NOTES

1. Missing numeric data are stored as a single period (.) .

2. The units for daily precipitation amounts and the monthly total precipitation are inches.

3. The monthly total and the 31 daily precipitation amounts are adjusted in order to be stored as integers; to get the true data values, each number must be multiplied by 0.01.

4. The daily codes for precipitation amounts are blank if there are no special conditions. Otherwise, the possible values for the codes are:

   - An 'T' means the corresponding daily value was a trace value

   - An 'A' means the corresponding daily value was accumulated

   - An 'E' means the corresponding daily value was estimated

   - A 'B' means the corresponding daily value was both estimated and accumulated

5. The monthly summary variable summarizes the information in the daily code variables. The summary is stored as an integer number, which is used to flag special conditions during the month. If no special conditions exist, then the value is zero; otherwise, the values of the flags mean:

   - 8: Missing values during month

   - 4: Accumulated values during month

- 2: Estimated values during month

- 1: Trace amounts during month

If more than one condition is flagged, then value of MONTHSUM will be the sum of all the applicable conditions.

Examples of how to decipher and use the monthly summary codes can be found in Appendix 8.5.

B) SAS Data Set Name - POINTERS.

Number of Records(Stations) - 294.

Number of Bytes/Record - 18.

Total Length of File in Bytes - 5,292.

RECORD DESCRIPTION

FIELD	VARIABLE NAME	TYPE
Station code number	STATION	CHAR
First observation(record)	FOBS_PTR	NUM
Last observation(record)	LOBS_PTR	NUM

5.2.3 NHIMS.TEMP.AIR

A) SAS Data Set Name - MAINDATA.

Number of Records - 73,376.

Number of Bytes/Record - 141.

Total Length of File in Bytes - 10,346,016.

RECORD DESCRIPTION

FIELD	VARIABLE NAME	TYPE
Station code number	STATION	CHAR
Calendar year	YEAR	NUM
Month	MONTH	NUM
Flag for missing/estimated data during month	MONTHSUM	NUM

31 maximum daily temperatures	MAXTMP1-MAXTMP31	NUM
31 minimum daily temperatures	MINTMP1-MINTMP31	NUM

NOTES

1. Missing numeric data are stored as a single period (.) .

2. The units for temperature are degrees Fahrenheit.

3. Daily values that are estimates are stored as out-of-range negative numbers; each one is scaled down by a factor of 200. If the monthly flag shows there are estimates for one month, each daily value should be checked to see if it is less than -80; if so, it is an estimate and should be scaled up by 200. The number -80 is used because it is out of range for a low temperature; yet if you add the scaling factor 200, it is also out of range for a high temperature.

4. The monthly summary variable summarizes the information in the daily code variables. The summary is stored as an integer number, which is used to flag special conditions during the month. If no special conditions exist, then the value is zero; otherwise, the values of the flags mean:

   - 128: Missing maximum temperatures during month

   - 32: Estimated maximum temperatures during month

   - 8: Missing minimum temperatures during month

   - 2: Estimated minimum temperatures during month

   If more than one condition is flagged, then value of MONTHSUM will be the sum of all the applicable conditions.

B) SAS Data Set Name - POINTERS.

Number of Records(Stations) - 251.

Number of Bytes/Record - 18.

Total Length of File in Bytes - 4,518.

RECORD DESCRIPTION

FIELD	VARIABLE NAME	TYPE
Station code number	STATION	CHAR
First observation(record)	FOBS_PTR	NUM
Last observation(record)	LOBS_PTR	NUM

5.2.4 NHIMS.STREAM

A) SAS Data Set Name - MAINDATA.

Number of Records - 136,828.

Number of Bytes/Record - 146.

Total Length of File in Bytes - 19,976,888.

RECORD DESCRIPTION

FIELD	VARIABLE NAME	TYPE
Station code number	STATION	CHAR
Calendar year	YEAR	NUM
Month	MONTH	NUM
Monthly total discharge X 100	STRM_TOT	NUM
31 daily values: mean daily discharge X 100	FLOW1-FLOW31	NUM

NOTES

1. The 31 daily values and the monthly total are adjusted in order to be stored as integers; to get the true data values, each number must be multiplied by 0.01.

2. All missing data are stored as a single period (.) .

3. The units for discharge are cubic feet per second (cfs).

B) SAS Data Set Name - POINTERS.

Number of Records(Stations) - 569.

Number of Bytes/Record - 20.

Total Length of File in Bytes - 11,380.

RECORD DESCRIPTION

FIELD	VARIABLE NAME	TYPE
Station code number	STATION	CHAR
First observation(record)	FOBS_PTR	NUM
Last observation(record)	LOBS_PTR	NUM

5.2.5 NHIMS.SNOW.FALL

A) SAS Data Set Name - MAINDATA.

Number of Records - 32,324.

Number of Bytes/Record - 237.

Total Length of File in Bytes - 7,660,788.

RECORD DESCRIPTION

FIELD	VARIABLE NAME	TYPE
Station code number	STATION	CHAR
Calendar year	YEAR	NUM
Month	MONTH	NUM
Monthly summary of codes	MONTHSUM	NUM
Monthly total snowfall X 10	SNOW_TOT	NUM
Remarks code for daily snowfall amount	S_CODE1-S_CODE31	CHAR
Remarks code for daily snow depth	D_CODE1-D_CODE31	CHAR
Daily snowfall amount X 10	SNOW1-SNOW31	NUM
Daily snow depth	DEPTH1-DEPTH31	NUM

NOTES

1. Missing numeric data are stored as a single period (.) .
2. The units for snowfall amounts, snow depths, and the monthly total are inches.
3. The monthly total and the 31 daily snowfall amounts are adjusted in order to be stored as integers; to get the true data values, each number must be multiplied by 0.1.

4. The daily codes for snowfall amounts and snow depths are blank if there are no special conditions. Otherwise, the possible values for the codes are:

   - An 'T' means the corresponding daily value is a trace
   - An 'A' means the corresponding daily value is accumulated
   - An 'E' means the corresponding daily value is estimated
   - A 'B' means the corresponding daily value is both estimated and accumulated

5. The monthly summary variable summarizes the information in the daily code variables. The summary is stored as an integer number, which is used to flag special conditions during the month. If no special conditions exist, then the value is zero; otherwise, the values of the flags mean:

   - 128: Snowfall amounts missing during month
   - 64: Snowfall amounts accumulated during month
   - 32: Snowfall amounts estimated during month
   - 16: Trace snowfall amounts during month
   - 8: Snow depths missing during month
   - 4: Snow depths accumulated during month
   - 2: Snow depths estimated during month
   - 1: Trace snow depths during month

   If more than one condition is flagged, the value of MONTHSUM will be the sum of all the applicable conditions.

   Examples of how to decipher and use the monthly summary codes can be found in Appendix 8.5.

B) SAS Data Set Name - POINTERS.

Number of Records(Stations) - 208.

Number of Bytes/Record - 18.

Total Length of File in Bytes - 3,744.

RECORD DESCRIPTION

FIELD	VARIABLE NAME	TYPE
Station code number	STATION	CHAR
First observation(record)	FOBS_PTR	NUM
Last observation(record)	LOBS_PTR	NUM

5.2.6 NHIMS.EVAP

A) SAS Data Set Name - MAINDATA.

Number of Records - 2,296.

Number of Bytes/Record - 271.

Total Length of File in Bytes - 622,216.

RECORD DESCRIPTION

FIELD	VARIABLE NAME	TYPE
Station code number	STATION	CHAR
Calendar year	YEAR	NUM
Month	MONTH	NUM
Total monthly wind movement	WIND_TOT	NUM
Total monthly evaporation X 100	EVAP_TOT	NUM
Monthly summary of codes	MONTHSUM	NUM
31 daily flags for estimated or accumulated wind values	W_CODE1-W_CODE31	CHAR
31 daily flags for estimated or accumulated evaporation values	E_CODE1-E_CODE31	CHAR
31 daily wind movement values	WIND1-WIND31	NUM
31 daily evaporation values X 100	EVAP1-EVAP31	NUM

NOTES

1. The 31 daily evaporation values and the total monthly evaporation are adjusted to be stored as integers; to get the true values, each number must be multiplied by 0.01.

2. Missing numeric data are stored as a single period (.) .

3. The units for evaporation are inches; the units for wind movement are miles.

4. If non-blank, the daily codes for wind and evaporation identify the following special conditions:

   - An 'A' means the corresponding daily value is accumulated

   - An 'E' means the corresponding daily value is estimated

   - A 'B' means the corresponding daily value is both estimated and accumulated

5. The monthly summary variable summarizes the information in the daily code variables. The summary is stored as an integer number, which is used to flag special conditions during the month. If no special conditions exist, then the value is zero; otherwise, the values of the flags mean:

   - 128: Wind movements missing during month

   - 64: Wind movements accumulated during month

   - 32: Wind movements estimated during month

   - 8: Evaporation values missing during month

   - 4: Evaporation values accumulated during month

   - 2: Evaporation values estimated during month

   If more than one condition is flagged, the value of MONTHSUM will be the sum of all the applicable conditions.

   Examples of how to decipher and use the monthly summary codes can be found in Appendix 8.5.

B) SAS Data Set Name - POINTERS.

Number of Records(Stations) - 16.

Number of Bytes/Record - 18.

Total Length of File in Bytes - 288.

RECORD DESCRIPTION

FIELD	VARIABLE NAME	TYPE
Station code number	STATION	CHAR
First observation(record)	FOBS_PTR	NUM
Last observation(record)	LOBS_PTR	NUM

5.2.7 NHIMS.RESVOIR

A) SAS Data Set Name - MAINDATA.

Number of Records - 7,618.

Number of Bytes/Record - 145.

Total Length of File in Bytes - 1,104,610.

RECORD DESCRIPTION

FIELD	VARIABLE NAME	TYPE
Station code number	STATION	CHAR
Calendar Year	YEAR	NUM
Month	MONTH	NUM
Code for time of day	TIMECODE	NUM
Code for data units	UNITCODE	CHAR
31 values: daily reservoir storage	STORGE1-STORGE31	NUM

NOTES

1. The 31 daily values may or may not need to be adjusted, depending on the value of the units code:

   - If UNITCODE = 'A', then the units of storage are acre-feet, and no adjustment of the daily values is necessary.

   - If UNITCODE = 'E', then the units of storage are feet of elevation above sea level, and the daily values must be multiplied by 0.01 on input.

   - If UNITCODE = 'S', then the units of storage are feet above datum, and the daily values must be multiplied by 0.01 on input.

2. All missing data are stored as a single period (.) .

3. The units for the time code variable are hours in 24-hour time.

B) SAS Data Set Name - POINTERS.

Number of Records(Stations) - 56.

Number of Bytes/Record - 20.

Total Length of File in Bytes - 1,120.

RECORD DESCRIPTION

FIELD	VARIABLE NAME	TYPE
Station code number	STATION	CHAR
First observation(record)	FOBS_PTR	NUM
Last observation(record)	LOBS_PTR	NUM

5.2.8 NHIMS.PEAKS

A) SAS Data Set Name - MAINDATA.

Number of Records - 17,249.

Number of Bytes/Record - 31.

Total Length of File in Bytes - 534,719.

RECORD DESCRIPTION

FIELD	VARIABLE NAME	TYPE
Station code number	STATION	CHAR
Water Year of Peak Flow	WAT_YEAR	NUM
Month of Peak Flow	MONTH	NUM
Day of Peak Flow	DAY	NUM
Calendar Year of Peak Flow	YEAR	NUM
Gage Height X 100	GAGE_HT	NUM
Annual Peak Flow	PEAKFLOW	NUM
Remarks Code for Gage Height	GH_CODE	CHAR
Remarks Code for Peak Flow	PK_CODE	CHAR
Remarks Code for Regulation or Diversion	RD_CODE	CHAR

NOTES

1. The values for gage height are adjusted in order to be stored as integers; to get the true data values, each number must be multiplied by 0.01.

2. Missing numeric data are stored as a single period (.); missing or non-applicable character data are stored as blanks.

3. The units for peak flow are cubic feet per second; the units for gage height are feet.

4. For the gage height remarks code:

   - a '1' indicates that the gage height was due to backwater; NHIMS will print a 'BW' message.

   - a '2' indicates that the gage height was not the maximum for the water year; NHIMS will print an 'NM' message. Also, the record immediately following contains information for the same water year. This information may be additional peakflow records or the maximum gage height value. If there are multiple records for one water year, the last one will always contain the maximum gage height.

5. For the peak flow remarks code:

   - a '1' indicates that the value given is a maximum daily; NHIMS will print an 'MD' message.

   - a '2' indicates that the discharge is estimated from information at another site; NHIMS will print an 'ES' message.

   - a '3' indicates that the maximum was due to dam failure; NHIMS will print a 'DF' message.

   - a '4' indicates that the actual discharge is less than the indicated value; NHIMS will print an 'LT' message.

6. For the regulation and diversion remarks code:

   - a '1' indicates an unknown effect of regulation or diversion; NHIMS will print an 'UR' message.

   - a '2' indicates a known significant effect of regulation or diversion; NHIMS will print a 'KR' message.

B) SAS Data Set Name - POINTERS.

Number of Records(Stations) - 730.

Number of Bytes/Record - 20.

Total Length of File in Bytes - 14,600.

RECORD DESCRIPTION

FIELD	VARIABLE NAME	TYPE
Station code number	STATION	CHAR
First observation(record)	FOBS_PTR	NUM
Last observation(record)	LOBS_PTR	NUM

5.2.9 NHIMS.SNOW.COURSE

A) SAS Data Set Name - SNOWWY.

Number of Records - 2,354.

Number of Bytes/Record - 79.

Total Length of File in Bytes - 185,966.

RECORD DESCRIPTION

FIELD	VARIABLE NAME	TYPE
FIPS code for state	STATE	NUM
Code showing when monthly measurement occurred	CARDNO	NUM
Monthly snow depths	DEPTH1-DEPTH6	NUM
Month of measurement	MONTH1-MONTH6	NUM
Day of measurement	DAY1-DAY6	NUM
Water Year	YEAR	NUM
Snow water equivalent	WATER1-WATER6	NUM
SCS station code number	STATION	CHAR
Station type	TYPE	CHAR

NOTES

1. This file contains snow course data for Wyoming. Eventually, it will be combined with the Idaho snow course data to make one MAINDATA file.

2. The units for snow depth and snow water equivalent are inches.

3. The possible values for the CARDNO variable are:

   - 1: first-of-month measurement

   - 2: mid-month measurement

   - 3-6: special measurements

4. The station type variable (TYPE) has the following possible values:

   - A: aerial station

   - M: soil moisture station

   - other possible values, the meaning presently unknown

B) SAS Data Set Name - SNOWID.

Number of Records - 8,698.

Number of Bytes/Record - 79.

Total Length of File in Bytes - 687,142.

RECORD DESCRIPTION

FIELD	VARIABLE NAME	TYPE
FIPS code for state	STATE	NUM
Code showing when monthly measurement occurred	CARDNO	NUM
Monthly snow depths	DEPTH1-DEPTH6	NUM
Month of measurement	MONTH1-MONTH6	NUM
Day of measurement	DAY1-DAY6	NUM

Water Year	YEAR	NUM
Snow water equivalent	WATER1-WATER6	NUM
SCS station code number	STATION	CHAR
Station type	TYPE	CHAR

NOTES

1. This file contains snow course data for Idaho. Eventually, it will be combined with the Wyoming snow course data to make one MAINDATA file.

2. The units for snow depth and snow water equivalent are inches.

3. The possible values for the CARDNO variable are:

   - 1: first-of-month measurement

   - 2: mid-month measurement

   - 3-6: special measurements

4. The station type variable (TYPE) has the following possible values:

   - A: aerial station

   - M: soil moisture station

   - other possible values, the meaning presently unknown

C) SAS Data Set Name - PTR_SWYO.

Number of Records(Stations) - 55.

Number of Bytes/Record - 23.

Total Length of File in Bytes - 1,265.

RECORD DESCRIPTION

FIELD	VARIABLE NAME	TYPE
SCS station code	STATION	CHAR
First observation(record)	FOBS_PTR	NUM
Last observation(record)	LOBS_PTR	NUM

First year of recorded data for the station	F_YEAR	NUM
Last year of recorded data for the station	L_YEAR	NUM

NOTES

1. This is the pointer file for the Wyoming snow course data. Please take note that the contents of this file are still provisional.

D) SAS Data Set Name - ID_PTRS.

Number of Records(Stations) - 395.

Number of Bytes/Record - 17.

Total Length of File in Bytes - 6,715.

RECORD DESCRIPTION

FIELD	VARIABLE NAME	TYPE
SCS station code	STATION	CHAR
First observation(record)	FOBS_PTR	NUM
Last observation(record)	LOBS_PTR	NUM

NOTES

1. This is the pointer file for the Idaho snow course data. Please take note that the contents of this file are still provisional.

E) SAS Data Set Name - SNOWNAME.

Number of Records - 365.

Number of Bytes/Record - 112.

Total Length of File in Bytes - 40,880.

RECORD DESCRIPTION

FIELD	VARIABLE NAME	TYPE
FIPS code for state	STATE	NUM
Section number	SECTION	NUM
Elevation	ELEV	NUM

Latitude	LAT	NUM
Longitude	LONG	NUM
SCS station code	STATION	CHAR
Station type	TYPE	CHAR
Township	TOWNSHIP	CHAR
Range	RANGE	CHAR
Station name	NAME	CHAR
Location narrative	LOCATION	CHAR

NOTES

1. This file contains much of the same data as the index file.

5.2.10 NHIMS.MONTHLY

A) SAS Data Set Name - MAINDATA.

Number of Records - 95,821.

Number of Bytes/Record - 84.

Total Length of File in Bytes - 8,048,964.

RECORD DESCRIPTION

FIELD	VARIABLE NAME	TYPE
Station code number	STATION	CHAR
Calendar year	YEAR	NUM
Month	MONTH	NUM
Mean maximum monthly temperature	MAXTEMP	NUM
Mean minimum monthly temperature	MINTEMP	NUM
Monthly mean temperature	TMEAN	NUM
Departure from normal temperature	TDEPART	NUM
Heating degree days	DEGDAYS	NUM
Highest maximum temperature during month	HIGHEST	NUM
Day of occurrence of highest temperature	HIGHDAY	NUM
Lowest minimum temperature during month	LOWEST	NUM
Day of occurrence of lowest temperature	LOWDAY	NUM
Total precipitation during month	PRECIP	NUM

Precipitation departure from normal	PDEPART	NUM
Maximum daily precipitation during month	PPTMAX	NUM
Date of maximum daily precipitation	PMAXDAY	NUM
Total monthly snowfall	SNOFALL	NUM
Maximum depth of snow on ground during month	SNODEPTH	NUM
Day of occurrence of maximum snow depth	SMAXDAY	NUM
Total wind run for month	WIND	NUM
Total monthly evaporation	EVAP	NUM
Cooling degree days	COOLDAYS	NUM
Flag for maximum monthly temperature	PLUS1	CHAR
Flag for minimum monthly temperature	PLUS2	CHAR
Flag for maximum monthly precipitation	PLUS3	CHAR
Flag for maximum monthly snow depth	PLUS4	CHAR
Flag for total monthly precipitation	PLUS5	CHAR
Flag for total monthly snowfall	PLUS6	CHAR

NOTES

1. Values of the PLUS flags will have the following meanings:

   - +: Value occurred on more than one day
   - A: Accumulated amount
   - B: Accumulated amount includes estimated values
   - E: Estimated amount
   - I: Monthly value based on an incomplete period
   - M: Data element for the flag is missing
   - T: Trace value

2. The units are inches for snowfall, evaporation and precipitation, degrees Fahrenheit for temperature, miles for wind movement, and degree Fahrenheit-days for heating and cooling degree days.

B) SAS Data Set Name - POINTERS.

Number of Records(Stations) - 258.

Number of Bytes/Record - 18.

Total Length of File in Bytes - 4,644.

RECORD DESCRIPTION

FIELD	VARIABLE NAME	TYPE
Station code number	STATION	CHAR
First observation(record)	FOBS_PTR	NUM
Last observation(record)	LOBS_PTR	NUM

5.2.11 NHIMS.PRECIP.HOURLY

A) SAS Data Set Name - MAINDATA.

Number of Records - 158,400.

Number of Bytes/Record - 94.

Total Length of File in Bytes - 14,889,600.

Stored on Tape B00782, Label 1.

RECORD DESCRIPTION

FIELD	VARIABLE NAME	TYPE
Station code number	STATION	CHAR
Calendar year	YEAR	NUM
Month	MONTH	NUM
Day	DAY	NUM
Daily summary of codes	DAYSUM	NUM
Daily total rainfall X 100	HPCP_TOT	NUM
Remarks code for hourly rainfall amount	H_CODE1-H_CODE24	CHAR
Hourly rainfall amount X 100	HPCP1-HPCP24	NUM

NOTES

1. Missing numeric data are stored as a single period (.) .
2. The units for hourly precipitation amounts and the daily total precipitation are inches.
3. The daily total and the 24 hourly precipitation amounts are adjusted in order to be stored as integers; to get the true data values, each number must be multiplied by 0.01.

4. The hourly codes for precipitation amounts are blank if there are no special conditions. Otherwise, the possible values for the codes are:

   - An 'M' means the corresponding hourly value was missing
   - An 'T' means the corresponding hourly value was a trace value
   - An 'A' means the corresponding hourly value was accumulated
   - An 'E' means the corresponding hourly value was estimated
   - A 'B' means the corresponding hourly value was both estimated and accumulated
   - An 'S' means the corresponding hourly value contained melting snow

5. The daily summary variable summarizes the information in the hourly code variables. The summary is stored as an integer number, which is used to flag special conditions during the month. If no special conditions exist, then the value is zero; otherwise, the values of the flags mean:

   - 16: Melting snow in measurement(s) during month
   - 8: Missing values during month
   - 4: Accumulated values during month
   - 2: Estimated values during month
   - 1: Trace amounts during month

   If more than one condition is flagged, then value of DAYSUM will be the sum of all the applicable conditions.

   Examples of how to decipher and use the monthly summary codes can be found in Appendix 8.5.

B) SAS Data Set Name - POINTERS.

Number of Records(Stations) - 115.

Number of Bytes/Record - 18.

Total Length of File in Bytes - 2,070.

Stored on Tape B00782, Label 1.

RECORD DESCRIPTION

FIELD	VARIABLE NAME	TYPE
Station code number	STATION	CHAR
First observation(record)	FOBS_PTR	NUM
Last observation(record)	LOBS_PTR	NUM

5.2.12 NHIMS.FORMATS

The format library contains permanent, user-defined SAS formats for use by the NHIMS system. The formats have been created using PROC FORMAT, are stored in load module form, and can be used by any SAS program which properly identifies the library. In order to use the formats, override the LIBRARY DD statement in the SAS cataloged procedure:

```
//LIBRARY  DD  DISP=(OLD,PASS),DSN=NHIMS.FORMATS
```

The remainder of this section describes the formats.

A) Format name - MNTH.

This is a value format which converts the numeric values 1-13 to the full name of the corresponding month. Month 13 has the value 'ANNUAL'. The statement:

```
PUT  MONTH  MNTH. ;
```

would print 'JANUARY' if the value of MONTH is 1, or would print 'AUGUST' if the value of MONTH is 8.

B) Format name - LATFORM.

This is a picture format intended for use with six-digit latitude values. It formats a numeric value into the picture '00-00-00'. The statement:

```
PUT  LAT  LATFORM. ;
```

would print the value 464400 as 46-44-00.

C) Format name - LONGFORM.

This is a picture format intended for use with seven-digit longitude values. It formats a numeric value into the picture '000-00-00'. The statement:

```
PUT  LONG  LONGFORM. ;
```

would print the value 1165800 as 116-58-00.

D) Format name - HUCFORM.

This is a picture format intended for use with eight-digit hydrologic unit codes. It formats a numeric value into the picture '00-00-00-00'. The statement:

```
PUT  HUCODE  HUCFORM. ;
```

would print the value 17060096 as 17-06-00-96.

E) Format name - $GHFORM.

This is a value format which converts the character values 1 or 2 to the corresponding character code for gage height in the peak flow file. The statement:

```
PUT  GH_CODE  $GHFORM. ;
```

would print 'BW' if the value of GH_CODE is '1', or would print 'NM' if the value of GH_CODE is '2'. Any other values would be printed as blanks.

F) Format name - $PKFORM.

This is a value format which converts the character values 1-4 to the corresponding character code for peak flow in the peak flow file. The statement:

```
PUT  PK_CODE  $PKFORM. ;
```

would print 'MD' if PK_CODE = '1';
'ES' if PK_CODE = '2';
'DF' if PK_CODE = '3';
'LT' if PK_CODE = '4';
Any other values would be printed as blanks.

G) Format name - $RDFORM.

This is a value format which converts the character values 1-2 to the corresponding character code for regulation and diversion in the peak flow file. The statement:

```
PUT RD_CODE $RDFORM. ;
```

would print 'UR' if RD_CODE = '1';
'KR' if RD_CODE = '2';
Any other values would be printed as blanks.

H) Format name - FMAX.

This is a picture format used with the calendar process which places a normal maximum temperature in the following picture string:

009.9 MAX _____

I) Format name - FMIN.

This is a picture format used with the calendar process which places a normal minimum temperature in the following picture string:

009.9 MIN _____

J) Format name - FPPT.

This is a picture format used with the calendar process which places a normal precipitation value in the following picture string:

09.99 PPT _____

K) Format name - FNOR.

This is a picture format used with the calendar process which writes a heading in the daily boxes in the following form:

NORMAL ACTUAL

L) Format name - FHMX.

This is a picture format used with the calendar process which places the highest maximum temperature and its 2-digit year of occurrence in the following picture string:

009/99 HIGH MAX

M) Format name - FLMN.

This is a picture format used with the calendar process which places the lowest minimum temperature and its 2-digit year of occurrence in the following picture string:

009/99 LOW MIN

N) Format name - FHPT.

This is a picture format used with the calendar process which places the highest precipitation value and its 2-digit year of occurrence in the following picture string:

9.99/99 HIGH PPT

5.2.13 CROSS REFERENCE MATRIX FOR PERMANENT FILES

TABLE OF MODULE BY FILENAME

MODULE / FILENAME Frequency	EVAPMAIN	FORMATS	HPCPMAIN	INDEX	MNTHMAIN	Total
C_CAL (4.129)	0	1	0	0	0	1
C_CORR (4.100)	0	1	0	0	0	1
C_DSTAT (4.94)	0	1	0	0	0	1
C_MSTAT (4.97)	0	1	0	0	0	1
FINDPER (4.41)	1	0	1	0	1	11
FINDPTR (4.24)	0	0	0	0	0	1
GETDATA (4.26)	1	0	1	0	1	8
GET_ANN (4.27)	0	0	0	0	0	2
HI_LO_1 (4.103)	0	1	0	0	0	1
HI_LO_2 (4.104)	0	1	0	0	0	1
IX_MRGE (4.42)	0	0	0	1	0	1
LC_PEAK (4.70)	0	1	0	0	0	1
LH_HPCP (4.75)	0	1	0	0	0	1
LI_LINE (4.44)	0	1	0	0	0	1
LI_PRT (4.43)	0	0	0	1	0	1
LP_DATA (4.48)	0	0	0	0	0	1
MONHEAD (4.57)	0	1	0	0	0	1
PRT_EX (4.106)	0	1	0	0	0	1
PTR_INX (4.25)	0	0	0	1	0	1
SC_HEAD (4.72)	0	1	0	0	0	1
SI_CHAR (4.16)	0	0	0	1	0	1
SI_NUM (4.17)	0	0	0	1	0	1
SI_RNGE (4.18)	0	0	0	1	0	1
Total	2	12	2	6	2	41

(Continued)

CROSS REFERENCE MATRIX FOR PERMANENT FILES

TABLE OF MODULE BY FILENAME

MODULE / Frequency	PEAKMAIN	POINTERS	PRCPMAIN	RESVMAIN	SNOCMAIN	Total
C_CAL (4.129)	0	0	0	0	0	1
C_CORR (4.100)	0	0	0	0	0	1
C_DSTAT (4.94)	0	0	0	0	0	1
C_MSTAT (4.97)	0	0	0	0	0	1
FINDPER (4.41)	1	1	1	1	1	11
FINDPTR (4.24)	0	1	0	0	0	1
GETDATA (4.26)	0	0	1	1	0	8
GET_ANN (4.27)	1	0	0	0	1	2
HI_LO_1 (4.103)	0	0	0	0	0	1
HI_LO_2 (4.104)	0	0	0	0	0	1
IX_MRGE (4.42)	0	0	0	0	0	1
LC_PEAK (4.70)	0	0	0	0	0	1
LH_HPCP (4.75)	0	0	0	0	0	1
LI_LINE (4.44)	0	0	0	0	0	1
LI_PRT (4.43)	0	0	0	0	0	1
LP_DATA (4.48)	0	1	0	0	0	1
MONHEAD (4.57)	0	0	0	0	0	1
PRT_EX (4.106)	0	0	0	0	0	1
PTR_INX (4.25)	0	0	0	0	0	1
SC_HEAD (4.72)	0	0	0	0	0	1
SI_CHAR (4.16)	0	0	0	0	0	1
SI_NUM (4.17)	0	0	0	0	0	1
SI_RNGE (4.18)	0	0	0	0	0	1
Total	2	3	2	2	2	41

(Continued)

CROSS REFERENCE MATRIX FOR PERMANENT FILES

TABLE OF MODULE BY FILENAME

MODULE / FILENAME Frequency	SNOWMAIN	STRMMAIN	TEMPMAIN	Total
C_CAL (4.129)	0	0	0	1
C_CORR (4.100)	0	0	0	1
C_DSTAT (4.94)	0	0	0	1
C_MSTAT (4.97)	0	0	0	1
FINDPER (4.41)	1	1	1	11
FINDPTR (4.24)	0	0	0	1
GETDATA (4.26)	1	1	1	8
GET_ANN (4.27)	0	0	0	2
HI_LO_1 (4.103)	0	0	0	1
HI_LO_2 (4.104)	0	0	0	1
IX_MRGE (4.42)	0	0	0	1
LC_PEAK (4.70)	0	0	0	1
LH_HPCP (4.75)	0	0	0	1
LI_LINE (4.44)	0	0	0	1
LI_PRT (4.43)	0	0	0	1
LP_DATA (4.48)	0	0	0	1
MONHEAD (4.57)	0	0	0	1
PRT_EX (4.106)	0	0	0	1
PTR_INX (4.25)	0	0	0	1
SC_HEAD (4.72)	0	0	0	1
SI_CHAR (4.16)	0	0	0	1
SI_NUM (4.17)	0	0	0	1
SI_RNGE (4.18)	0	0	0	1
Total	2	2	2	41

6.0 TEST PROVISIONS

The following two sections provide the guidelines to be followed in order to test the NHIMS system. First, unit testing will be performed on each module in the system, verifying that each module works correctly as an independent unit. Subsequently, integration testing will be performed in order to ensure that the individual modules work together as a system.

Use of the SAS system itself provides a measure of verification that the system works; for instance, PROC SORT and PROC MEANS will correctly sort data and compute standard statistics. One of the main testing stategies in NHIMS is to make sure that the data used as input to the SAS PROCs is valid, so that the subsequent output can also be considered valid.

All modules will be uploaded from the diskettes on which they are currently stored to an account on CMS. An in-stream JCL procedure called NHIMS will be used in the testing phase to set up external file definitions; this procedure is a modification of the SAS cataloged procedure, and at the end of the project will become itself a permanent, cataloged procedure for general use with NHIMS. The testing of individual modules will entail submitting jobs directly from CMS to OS/VS1. After unit testing, the modules will be stored in a partitioned data set on the USR008 disk pack; logical groupings of modules will be individual members of the pds. During integration testing, the members of the pds will be retrieved as needed by NHIMS, using the SAS statement %INCLUDE [12, p. 1132]. PROC SOURCE will be used to store the source code in the partitioned data set, and can be subsequently used to retrieve modules that need to be edited.

6.1 UNIT TEST GUIDELINES

Each module in the NHIMS system will be individually tested to ensure that it correctly performs its own particular function. Due to the large number of modules, an individual description of what is needed to test each module will not be given here; such information can be found in the detailed module descriptions of Section 4. Instead, this section describes general stategies for performing the unit testing.

The test data for the NHIMS system are of three types. First, the original commands submitted by the user are what initially drive the system, identifying the data to be retrieved and the operations to be performed. Next, the global macro variables trigger the conditional generation of SAS program statements. Third, the actual climatic or hydrologic data from the permanent files provide the raw material for most of the SAS program statements.

The type of test data needed by a module depends on the logical function of that module. The logical functions fit into five groups:

- parse the user commands
- retrieve data from the permanent files
- list the data
- copy the data to an external file
- process the data

In general, the parsing modules will test the possible combinations of the user commands; such commands will form the input to the modules, and the assignment of values to global macro variables will be the output from the modules. The data retrieval routines will get the data from the permanent files, creating subsets and performing conversions, depending on the values of the macro variables already assigned. The data subsets will then become the input to the remaining routines, and the operations to be performed on the subsets will also be triggered by the values of the macro variables.

6.2 INTEGRATION TEST GUIDELINES

The NHIMS in-stream procedure will be used to perform all integration testing. The integration strategy will be to organize the modules into the five logical groups mentioned above, and test each of these groups separately; when that step is successfully completed, the groups will be combined, in a top-down, depth-first manner, until the system is complete.

The only test data required for integration testing will be the user commands submitted as a batch job. %PUT statements will be placed in key modules within a logical group in order to trace the system's execution. Also, comparisons can be made with the existing HISARS system to ensure that the same commands produce the same results.

It will be possible to test a wide variety of user commands without attempting to produce all possible combinations of those commands. Instead, divide the commands into logical groups. For example, first test the possible combinations of commands that identify the different types and ranges of data to be retrieved. When satisfied that those commands have been successfully performed, keep the retrieval commands simple while trying the possible combinations of the listing commands. Follow the same strategy for the process and copy commands.

TEST PROVISIONS

It will be necessary to measure and keep track of the CPU times used and the costs incurred during integration testing. Also, performance measurements should be made regarding the difference between compiling all the macros in the system versus simply including and compiling logical groups of macros as needed.

Because the main NHIMS files may be stored on tapes in the future, some of the integration tests should be performed on the backup tapes to make sure that the same system will work irregardless of the storage medium. All that should need to be changed are the DD statements in the NHIMS procedure. Again, performance measurements should be taken, in order to compare the CPU times between accessing the two different storage mediums.

7.0 PACKAGING

At the completion of the NHIMS project, the following items will be delivered to Dr. Myron Molnau, Agricultural Engineering Department, University of Idaho:

- The Design Document for the NHIMS system, which can be used as a programmer's maintenance manual

- A User's manual for the NHIMS system

- the Design Document and User's manual on diskette

- the source code for NHIMS stored in a partitioned data set on the USR008 disk pack

- written documentation of the structure of the partitioned data set

- a notebook of programs, file creation histories, sample data files, documentation of erroneous data in permanent files, and miscellaneous information gathered during the course of working on the NHIMS project

The Idaho Water Resources Research Institute, which has been responsible for funding this project, will also receive copies of the first three items in the above list.

8.0 APPENDIX

The appendix contains instructions on how to add new element files and new processes to the NHIMS system. In addition, examples are given of output listings (although a more complete set of listings are given in [2], [10]), the reasons for choosing the DCB information in the copy modules, and the strategy for a programmer using the information in the monthly code summaries.

8.1 Strategy For Adding New Element Files to the NHIMS System

A) In order to add another permanent file to NHIMS, the file must conform to the following characteristics:

- be a permanent SAS data set
- have a data set name of MAINDATA
- be organized by weather station and date
- contain data for one day, month, or year per record
- have a STATION and YEAR variable
- be sorted by STATION, then YEAR (also MONTH and DAY, if applicable)

B) A pointer file must be created to serve as a map of the main file, with the same structure as all other pointer files.

C) One must create a 4-character abbreviation for the element name, and use that name in creating a global macro variable for the element; for example, to add a file of soil temperatures, use the abbreviation STMP, and a macro variable EL_STMP.

D) One must add a variable to the permanent index file to flag the fact that NHIMS contains data for that element. Also, match the element's pointer file with the stations in the index file, and set the flag variable to 'Y' for each station in NHIMS. If there are stations in the pointer file not in the index, records for those stations must be added to the index.

E) Modules that must be modified are:

- DECLARE GLOBAL MACRO VARIABLES (4.2)
- INITIALIZE GLOBAL MACRO VARIABLES (4.7)
- SET ELEMENT FLAGS (4.10)
- RETRIEVE DATA FOR ALL ELEMENTS (4.22)

- LIST INDEX (4.39)
- LIST POINTERS (4.47)
- LIST DAILY (4.49), MONTHLY (61), HOURLY (4.74) or CONTENTS (4.69), whichever apply
- COPY DIRECTLY TO A SAS DATA SET (4.77)
- COPY EXTERNAL (4.79)

New modules will have to be written to perform daily or contents listings, and to do an external copy. New modules may have to be written to perform the other listings, depending on the structure of the new file.

To process the new element, see Appendix 8.2.

F) Add documentation for the new element, including all changes made to the system, to this document.

8.2 Strategy For Adding New Processes to the NHIMS System

A) One must first create a global macro variable to represent the name of the process. To conform with the other processes, this is done by taking the first three letters of each of the first two words in the name of the process, and the first two letters of the third word in the name. For example, the macro variables for the following processes are:

CALENDAR - CAL
HIGH OCCURRENCES - HIGOCC
FLOW DURATION TABLE - FLODURTA

B) The modules that must be modified are:

- DECLARE GLOBAL MACRO VARIABLES (4.2)
- INITIALIZE GLOBAL MACRO VARIABLES (4.7)
- PROCESS (4.90)

New modules will have to be added to make a subset of the data for the process and to perform its intended functions. However, existing modules may or may not be used in order to determine the options for the process. For example, if the process simply has one option, ONLY, the module GET MONTHS FROM THE ONLY OPTION (4.92) may be used to determine which months are requested by the user. However, if the options are much more complicated, new modules will have to be written. Examples to follow can be seen in modules 4.92, 4.112, 4.117, 4.122 and 4.127.

C) Add documentation for the new process, including all of the changes made to the system, to this document.

8.3 Example Outputs from NHIMS

This section contains 2 examples of output listings from NHIMS; illustrated are outputs from list index and list monthly. More complete listings of outputs can be found in [2] and [10].

```
   TEMPERATURE STATIONS                                                    PAGE 1

NUTTERVILLE HILL                                          LATAH          STATION NO.  UI-0001
        LATITUDE  46-48-00       LONGITUDE  116-53-00
        ELEVATION           FT MSL     DIVISION  02      HUCODE  17-06-00-96

        NHIMS RECORDS ARE AVAILABLE FROM 10-1969 TO 03-1982.    NUMBER OF RECORDS:  150

        WRITTEN OR MICROFICHE RECORDS ARE AVAILABLE FROM  07-1955 TO 12-1981.

MOSCOW UNIV OF IDAHO                                      LATAH          STATION NO.  10-6152
        LATITUDE  46-44-00       LONGITUDE  116-58-00
        ELEVATION  2660     FT MSL     DIVISION  02      HUCODE  17-06-00-96

        NHIMS RECORDS ARE AVAILABLE FROM 01-1900 TO 12-1985.    NUMBER OF RECORDS:  1031

        WRITTEN OR MICROFICHE RECORDS ARE AVAILABLE FROM  02-1892 TO ACTIVE .

POTLATCH 3  NNE                                           LATAH          STATION NO.  10-7301
        LATITUDE  46-58-00       LONGITUDE  116-53-00
        ELEVATION  2600     FT MSL     DIVISION  02      HUCODE  17-06-00-96

        NHIMS RECORDS ARE AVAILABLE FROM 03-1915 TO 12-1985.    NUMBER OF RECORDS:  784

        WRITTEN OR MICROFICHE RECORDS ARE AVAILABLE FROM  03-1915 TO ACTIVE .

        ** NOTE **
        DETERMINING THE PERIODS OF RECORD CAN BE EXPENSIVE. CONSULT THE USER'S MANUAL
        FOR LIST INDEX COMMANDS WHICH WILL EXCLUDE SUCH INFORMATION.
```

Figure 11. Sample Output From List Index.

MOSCOW UNIV OF IDAHO LATAH STATION NO. 10-6152

MONTHLY SUMMARIZED STATION DATA

DATE		TEMPERATURES (F)									PRECIPITATION (IN)						EVAP/WIND (IN/MI)	
MONTH	YEAR	MEAN MAX	MEAN MIN	MEAN	HI	DATE	LOW	DATE	HEAT DEGREE DAYS	COOL DEGREE DAYS	TOTAL	GREATEST DAY	DATE	TOTAL SNOWFALL	GREATEST SNOW DEPTH	DATE	TOTAL EVAP	WIND RUN
1	1980	30.8	16.1+	23.5	45	12	-5	29	1281	0	3.65	1.02	5	23.5	18	10		
2	1980	41.7	30.6+	36.2	56	27	17	16	829	0	1.71	0.42+	18	1.5	4	1		
3	1980	44.8	30.5	37.7	54	1	22	6	842	0	2.67	0.48	14	4.5	0			
4	1980	61.1	38.7	49.9	82	28	27	1	448	2	1.51	0.48	29	0.0T	0			
5	1980	64.4	42.1	53.3	81	5	33	16	358	0	4.80	1.42	26	0.0	0		5.02	1578
6	1980	67.7	44.2+	56.0	80	8	34	5	266	0	1.99	0.49	12	0.0	0		5.78	1775
7	1980	80.6	48.0	64.3	97	22	37	1	70	59	1.12	0.59	10	0.0	0		8.37	1273
8	1980	78.5	42.9	60.7	90	11	30	25	145	20	1.00	0.59	31	0.0	0		8.55	1698
9	1980	73.6	43.5	58.6	87	5	33	21	192	6	1.08	0.54	13	0.0	0		5.78	1729
10	1980	63.4	36.8	50.1	86	7	22	22	465	10	0.75	0.56	14	0.0	0		3.94	1754
11	1980	46.0	33.4	39.7	66	4	20	13	754	0	3.90	1.06	7	0.0	0			
12	1980	42.2	30.3	36.3	59	26	-4	7	887	0	3.88	0.83	25	6.5	4+	6		
ANNUAL	1980	57.9	36.4	47.2	97		-5		6537	97	28.06	1.42		36.0	18			

+ - VALUE OCCURRED ON MORE THAN ONE DAY
M - MISSING RECORDS IN MONTH
T - TRACE MONTHLY VALUE
I - BASED ON INCOMPLETE PERIOD
A - ACCUMULATED AMOUNT
E - ESTIMATED AMOUNT

Figure 12. Sample Output from List Monthly.

8.4 DCB Information for the Copy Modules

The logical record lengths for the types of external files created by the COPY command are 80, 284, and 380. An LRECL of 80 bytes is produced when the user specifically requests that record length; otherwise, formatted files use 380-byte records, since that will contain the longest record from any of the element files. For unformatted records, all data is written with a real binary format in 4 bytes, so the maximum record length needed for any of the elements is 284 bytes.

Using the DISKEFF command on CMS, the best block sizes were chosen to match the above record lengths on 3350 devices. For 80-byte records, a BLKSIZE = 9440 was chosen because it gave 99% efficiency; considerably larger block sizes did very little to improve efficiency or the number of records per track. For the same reasons, block sizes of 9372 for 284-byte records and 4560 for 380-byte records were chosen.

8.5 Examples Of Using Monthly Summary Codes

For precipitation, if MONTHSUM = 11, there will be one or more days in that month with missing, estimated, and trace values. Begin by subtracting the largest possible powers of 2.

```
 11
 -8 missing values
---
  3
 -2 estimated values
---
  1
 -1 trace values
---
  0
```

For evaporation, if MONTHSUM = 136, there will be one or more days in that month with missing wind and missing evaporation values.

```
 136
-128 missing wind
----
   8
  -8 missing evaporation
---
  0
```

Such a method as above can be used to determine which conditions are flagged by the value of the monthly summary integer. Also, one can check the range of the number to determine the flagged conditions.

For example, with precipitation, if MONTHSUM ¢= 8, then there will be at least one missing day during the month.

An alternative method is to check the individual bits of the number in order to get the same information. For more information about bit checking, refer to the section on SAS Expressions in the SAS User Guide: Basics.