US Army Corps
of Engineers
Hydrologic Engineering Center

GENERALIZED COMPUTER PROGRAM

HEC-DSS

User's Guide and Utility Manuals

User's Manual

March 1995

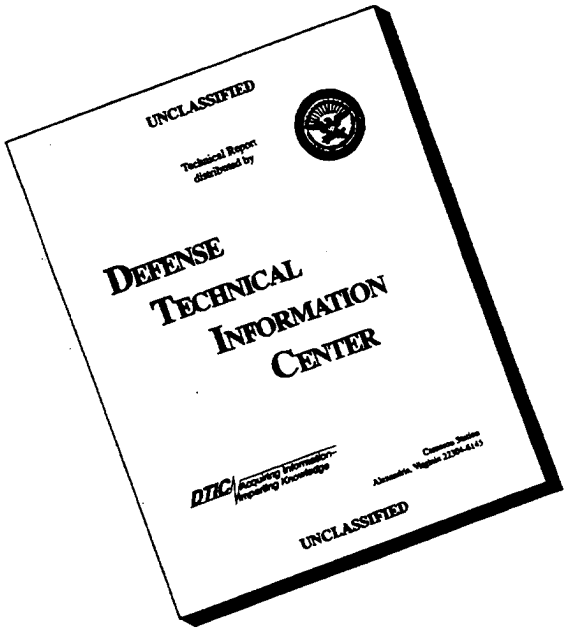
19960807 017

Approved for Public Release. Distribution Unlimited.

CPD-45

DTIC QUALITY INSPECTED 1

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
MARCH 1995

3. REPORT TYPE AND DATES COVERED
Computer Program Document

4. TITLE AND SUBTITLE
HEC-DSS User's Guide and Utility Manuals, User's Manual

5. FUNDING NUMBERS

6. AUTHOR(S)
CEWRC-HEC

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
USA CORPS OF ENGINEERS
WATER RESOURCES SUPPORT CENTER
HYDROLOGIC ENGINEERING CENTER
609 SECOND STREET
DAVIS CA 95616-4687

8. PERFORMING ORGANIZATION REPORT NUMBER
CPD-45

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
Approved for public release.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

HEC's Data Storage System (HEC-DSS or DSS) is a database system that was developed to meet needs for data storage and retrieval for water resource studies. The system enables efficient storage and retrieval of time series data (such as precipitation hyetographs or hydrographs of stage, discharge, etc.) and other data-types for which storage in blocks of contiguous data elements is most appropriate. The DSS consists of a library of subroutines which can be readily used with application programs to enable retrieval and storage of information. At present, approximately 20 application programs have been adapted in this fashion. In addition, DSS utility programs have been developed, data entry programs, a powerful graphics program, a report generator, and a program that performs mathematical transformations. Macros, selection screens, and other user interface features combine with DSS products to provide a set of tools whose application is limited only by the ingenuity of the user.

14. SUBJECT TERMS
HEC-DSS, DSS, Time-Series, Data, Databases

15. NUMBER OF PAGES
447

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT
UNCLASSIFIED

18. SECURITY CLASSIFICATION OF THIS PAGE

19. SECURITY CLASSIFICATION OF ABSTRACT

20. LIMITATION OF ABSTRACT

HEC-DSS

User's Guide and

Utility Program Manuals

Overview

DSSUTL

DSPLAY

DSSMATH

REPGEN

DSSTS

DSSITS

DSSPD

DSSTXT

DWINDO

WATDSS

NWSDSS

PREAD

March 1995

Overview

Hydrologic Engineering Center
Data Storage System

March 1995

Hydrologic Engineering Center
U.S. Army Corps of Engineers
609 Second Street
Davis, California 95616-4687
(916) 756-1104

Acknowledgments

The development of Hydrologic Engineering Center Data Storage System (HEC-DSS) is due to the efforts of a number of individuals. The HEC-DSS was first conceived and designed in 1979 by Arthur F. Pabst, Mark G. Lewis, and Rochelle Barkin. Since that time, it has been upgraded and maintained by William J. Charley under the supervision of Arthur F. Pabst. Rochelle Barkin, Robert D. Carl, William J. Charley, Paul Ely, Dennis J. Huff, and Al E. Montalvo designed and implemented utility programs which support the HEC-DSS. Many others have provided ideas, interfaced application programs, and tested procedures.

Overview

Table of Contents

Chapter Page

1. Overview Summary 1

2. Introduction 3
   2.1 Data Bases in General 3
   2.2 A Storage System for Water Resources Data 3

3. Concepts and Features 5
   3.1 General 5
   3.2 Pathnames 6
   3.3 Data Conventions 6
   3.4 Multiple User Access 7
   3.5 File Protection 7
   3.6 Application Program Interface 8
   3.7 Catalogs 8
   3.8 Utility Programs 11

4. Technical Information 13
   4.1 Data Compression for Regular-Interval Time Series Data 13
   4.2 DSS File Parameters 13
   4.3 DSS on MS-DOS Personal Computers 14
   4.4 DSS on UNIX Workstations 15

Appendices

A. Data Conventions A-1
   Time Series A-1
   Paired Data A-6
   Text Data A-7

B. Glossary of Terms B-1

Chapter 1

Overview Summary

The HEC Data Storage System (HEC-DSS or just DSS) stores data in a fashion convenient for inventory, retrieval, archiving and model use. The DSS was primarily designed for water resource applications. The user may interact with the data base through:

a) Utilities that allow entry, editing, and display of information.

b) Application programs that read from and write to the data base.

c) Library routines that can be incorporated in any program to access data base information.

The DSS provides a means for:

1) storing and maintaining data in a centralized location,

2) providing input to and storing output from application programs,

3) transferring data between application programs, and

4) displaying the data in graphs or tables.

(This page intentionally left blank)

Chapter 2

Introduction

2.1 Data Bases in General

The growth in use of data base systems is in response to the increasing, widespread use of computers in various applications. The ability to rapidly process large amounts of information has motivated the storage of information in a digital form. Essentially, data base systems have replaced written and printed storage of information.

In order to be effective, data base systems, like the paper-based systems they replace, are organized in a predictable format. Information is systematically labelled and stored to facilitate efficient and convenient reference and access.

Many kinds of information change in time. In order to maintain current information, data base systems provide the capability to add new information and revise or delete old information. This is referred to as updating the data base.

To achieve maximum utility, economy of storage, and ease of maintenance, information is kept in a central location for retrieval or updating by multiple users. Data base systems provide measures for minimizing information loss due to human error or computer system malfunctions. Use of a centralized storage location by multiple users also results in a need for measures to manage simultaneous requests for the same piece of information.

2.2 A Storage System for Water Resource Data

There is a substantial need for data base systems in water resources engineering and planning. Systematic, printed records of water resource data have been widely maintained since the beginning of the century, and the number of stations observed has greatly increased with time. Much of this enormous volume of printed data has been translated into digital storage medium, and it is now routine procedure to directly record observations in digital form. A data base system enables the storage of hydrologic data in a central location, and convenient retrieval capability.

Analysis programs are used to rapidly, repeatedly, and systematically evaluate long records of water resource data. In addition, programs often operate sequentially, with the output from one being the input to another. To be used effectively, these programs must be provided with the capability to conveniently store and retrieve data from a central storage location in a common format.

DSS was developed to meet the specific needs of applications in water resources studies. In order to efficiently manage large amounts of related data for practical applications, routines have been developed which organize and transfer data in blocks of continuous data. For example, each block of data for time series applications consists of a record of a time-dependent variable for a day or month or year. By storing data in this manner, an entire year of daily flows (or block of data) can be handled as a single element and accessed by one "search" of the data base. Other routines deal with tables of paired data, such as stage-discharge, stage-damage or discharge-frequency relationships.

Chapter 3

Concepts and Features

3.1 General

A conventional, general purpose data base system could be used for applications in water resources studies, but the large volumes of data typically involved make the generalized systems not the best choice for many applications. The generalized systems are often oriented towards business applications; they assume information is to be organized into collections of different attributes with a common relationship and store all information in character form. For example, the typical data base would efficiently store employee information - name, address, social security number, pay, etc., while the need in water resources may be to store 50 years of daily flows - quite different from the business application. Using an elemental business data base in the water resource application would require an excessive amount of computer time to search and convert character representation of numbers into real numbers.

HEC-DSS uses a block of sequential data as the basic unit of storage. This concept results in a more efficient access of time series or other uniquely related data. Each block contains a series of values of a single variable over a time span appropriate for most applications. The basic concept underlying the DSS is the organization of data into records of continuous, applications-related elements, as opposed to individually addressable data items. This approach is more efficient for water resources applications than that of a conventional data base system because it avoids the processing and storage overhead required to assemble an equivalent record from a conventional system.

DSS consists of a package of FORTRAN subroutines easily interfaced to computer programs, and a set of utility support routines to aid in interpreting and maintaining data in the data base. The subroutines and support programs retrieve (read) and store (write) data in "direct access" type computer files. Direct access files allow efficient retrieval and storage of blocks of data compared to sequential files. As many DSS files as needed may be created - the DSS does not limit the number of files. No "set-up" is required to generate a DSS file; the software does this automatically.

Data is stored in blocks, or records, within a file, and each record is identified by a unique name call a "pathname". Each time data is stored or retrieved from the file, its pathname must be given. Along with the data, information about the data (e.g., units) is stored in a "header array". The DSS automatically stores the name of the program writing the data, the number of times the data has been written to, and the last written date and time.

Through information contained in the pathname and stored in the header, the stored data is completely documented. That is, no additional information is required to identify it. The self documenting nature of the data base allows information to be recognized and understood months or years after it was stored.

3.2 Pathnames

DSS records are referenced by their pathnames. The pathname consists of up to 80 characters and is, by convention, separated into six parts. The parts are referenced by the characters A, B, C, D, E, and F, and are delimited by a slash "/", as follows:

```
/A/B/C/D/E/F/
```

For example, a brief description of the pathname for time series data is as follows:

PATHNAME PART	DESCRIPTION
A	River basin or project name
B	Location or gage Identifier
C	Data variable, e.g., `FLOW`, `PRECIP`
D	Starting date for block of data, such as `01JAN1981` for daily data in 1981
E	Time interval, e.g., `1DAY`, `3HOUR`, `1MON`
F	Additional user-defined description to further define the data, e.g., `PLAN A`

A typical pathname might be:

```
/RED RIVER/BEND MARINA/FLOW/01JAN1975/1DAY/OBS/
```

The DSS software does not sequentially search through the DSS file for data, but uses its pathname to index its position within the file. This technique allows the rapid storage and retrieval of data from the file, regardless of its size.

3.3 Data Conventions

Data of most any type, using a pathname of any structure (up to 80 characters), can be stored by the DSS. To facilitate the ability of application and utility programs to work with and display data, standard record conventions were developed. These conventions define what should be contained in a pathname, how data is stored and what additional information is stored along with the data. For regular-interval time series data (e.g., hourly data), the conventions specify that data are stored in blocks of a standard length, uniform for that time interval, with a pathname that contains the date of the beginning of the block and the time interval. The conventions identify how a pathname for that data should be constructed. Conventions have been defined for regular and irregular interval time series data, paired (curve) data, and text (alphanumeric) data. Conventions for other types of data have been proposed.

Regular-interval time series data is data that occurs at a standard time interval (a list of intervals used in DSS is given in Appendix A). This data is divided into blocks, whose length depends on the time interval (for example, hourly data is stored with a block length of a month,

while daily data is stored with a block length of a year). Only the date and time of the first piece of data for a block is stored; the times of the other data elements are implied by their location within the block. If a data element, or a set of elements, does not exist for a particular time, a missing data flag is placed in that element's location. Data quality flags may optionally be stored along with a regular-interval time series record. These flags can be used by several programs, and edited or displayed with the utility program DSSUTL.

Irregular-interval time series data does not have a constant time interval between values. This type of data is stored with a date/time stamp for each element. The user-selectable block size is based on the amount of data that is to be stored. For example, the user may select a block length of a month or a year. Because a date/time stamp is stored with each data element, approximately twice the amount of space is required compared with regular-interval time series data. Data quality flags may optionally be stored along with an irregular-interval time series record. These flags can be used by several programs, and edited or displayed with DSSUTL.

A convention for paired function data (or paired data) has been defined for data that generally defines a curve. It is used for rating tables, flow-frequency curves, and stage-damage curves, etc.. One paired data record may contain several curves within it, as long as it has a common set of ordinates. For example a stage-damage curve will contain a set of stages, and it may have associated residential damages, commercial damages, agricultural damages, etc.. However, a stage-damage curve and a stage-flow curve cannot be stored in the same record.

A text convention has been defined for lines of standard alpha-numeric characters (where a line begins with a carriage return and ends with a line feed character). This convention does not include characters that are not in a standard line format (such as those that would be used to generate a graphical display).

3.4 Multiple User Access

In order to maximize the effectiveness of a data base, several users should be able to use the same data file at the same time. For example, a flood forecast model may need to retrieve data from the file at the same time data elsewhere in the file is being updated. To do this, the data base must incorporate a method of handling multiple users of the same file at virtually the same time. A DSS multiple user scheme accomplishes this by a first come - first served approach. When a program requests to write to the file, the DSS software will queue that request and hold it until all prior requests are completed. Typical delays take only a fraction of a second and are not detectable by users.

3.5 File Protection

Measures to protect a file from program or system errors are an essential feature of a data base system. The DSS software accounts for this in its method of storage. An abort during the use of a DSS file usually results in no more than the loss of the data being currently written. As is true for all important information, the creation of a backup copy of a file is encouraged.

3.6 Application Program Interface

A systematic interface between application programs and DSS files is provided by a set of FORTRAN subroutines which are linked with application programs. A description of these subroutines and instructions for their use are presented in the HEC-DSS Programmer's Manual. Several of HEC's general application programs have been interfaced with DSS, allowing users to retrieve data for analysis or store results in a DSS file. This gives the user the capability of displaying and analyzing application program results by using the DSS utility programs. Using DSS to store program input and output helps avoid many of the cumbersome tasks associated with analyzing large amounts of data. These tasks include reformatting data, manually entering data, assembling large input files, etc. Programs using the DSS may interact with more than one DSS file simultaneously. Data can be retrieved from one file, analyzed, and then stored in another file.

3.7 Catalogs

Several DSS utility programs will generate a list of the pathnames in a DSS file and store that list in a "catalog" file. The catalog file is a list of the record pathnames in a DSS file, along with their last written date and time and the name of the program that wrote that record. The catalog is usually sorted alphabetically by pathname parts. Each pathname has a record tag and a reference number, either of which may be used in place of the pathname in several of the utility programs. The name given to the catalog file is the DSS file's name with a ".dsc" extension. An example of a catalog file is shown in Figure 1.

A catalog reference number is the sequential number of a pathname in the catalog file. These numbers are provided for quick interactive reference to a record from a utility program. When a number is given, the utility program sequentially searches the catalog file until it finds that number, then returns the associated pathname. Reference numbers are temporary; they may change each time the catalog is updated.

A record tag is a one to eight character semi-permanent record identifier that is not necessarily unique. It is also intended for quick interactive reference to a record from a utility program. Tags (which must begin with a non-numeric character), can be set by the user, or they can be set according to a scheme based on parts of the pathname. For example, a scheme might cause a data record of observed flow with a "B part" of NATP to have a tag of NATP-OF. The default tag is the letter "T" followed by the sequence number of the record. If a tag is not unique, only the pathname from the first matching tag will be used by the utility program. A file's tag scheme may be set, or records may have their tags' changed by the user with RT command in the program DSSUTL.

A special catalog, called a "condensed catalog", is useful primarily for time series data. In this type of catalog, pathname parts are listed in columns, and pathnames for time series data, which differ only by the date (D part), are referenced with just one line. Repeating parts are replaced by dashes for easier reading. The name given to the condensed catalog file is the name of the DSS file with a ".dsd" extension. An example of a condensed catalog is shown in Figure 2.

HEC-DSS Complete Catalog of Record Pathnames in File MASTDB.DSS

Catalog Created on Mar 6, 1990 at 15:24 File Created on Jan 8, 1990
Number of Records: 30 DSS Version 6-DA
Sort Order: ABCFED

Ref. Number	Tag	Program	Last Written Date	Time	Type	Vers	Data	Record Pathname
1	T3	DSSPD	06MAR90	15:22	RTS	1	366	/AMERICAN/AT FAIR OAKS/FLOW/01JAN1988/1DAY/OBS/
2	T4	DSSTS	06MAR90	15:22	RTS	2	365	/AMERICAN/AT FAIR OAKS/FLOW/01JAN1989/1DAY/OBS/
3	T11	DSSTS	06MAR90	15:22	RTS	3	365	/AMERICAN/BLUECANYON/PRECIP-INC/01JAN1987/1DAY/OBS/
4	T9	DSSTS	06MAR90	15:22	RTS	4	366	/AMERICAN/BLUECANYON/PRECIP-INC/01JAN1988/1DAY/OBS/
5	T12	DSSTS	06MAR90	15:22	RTS	1	365	/AMERICAN/BLUECANYON/PRECIP-INC/01JAN1989/1DAY/OBS/
6	T26	DSSTS	20FEB90	09:30	RTS	9	365	/AMERICAN/FOLSOM/ELEV/01JAN1987/1DAY/OBS/
7	T25	DSSTS	20FEB90	09:30	RTS	4	366	/AMERICAN/FOLSOM/ELEV/01JAN1988/1DAY/OBS/
8	T27	DSSTS	20FEB90	09:30	RTS	1	365	/AMERICAN/FOLSOM/ELEV/01JAN1989/1DAY/OBS/
9	T21	DSSTS	20FEB90	09:30	RTS	4	365	/AMERICAN/FOLSOM/EVAP-PAN/01JAN1986/1DAY/OBS/
10	T18	DSSTS	20FEB90	09:30	RTS	1	365	/AMERICAN/FOLSOM/EVAP-PAN/01JAN1987/1DAY/OBS/
11	T16	DSSTS	20FEB90	09:30	RTS	1	366	/AMERICAN/FOLSOM/EVAP-PAN/01JAN1988/1DAY/OBS/
12	T23	DSSTS	20FEB90	09:30	RTS	1	365	/AMERICAN/FOLSOM/FLOW-OUTLET/01JAN1987/1DAY/OBS/
13	T22	DSSTS	20FEB90	09:30	RTS	6	366	/AMERICAN/FOLSOM/FLOW-OUTLET/01JAN1988/1DAY/OBS/
14	T24	DSSTS	20FEB90	09:30	RTS	2	365	/AMERICAN/FOLSOM/FLOW-OUTLET/01JAN1989/1DAY/OBS/
15	T19	DSSTS	20FEB90	09:30	RTS	1	365	/AMERICAN/FOLSOM/FLOW-POWER/01JAN1987/1DAY/OBS/
16	T17	DSSTS	20FEB90	09:30	RTS	2	366	/AMERICAN/FOLSOM/FLOW-POWER/01JAN1988/1DAY/OBS/
17	T20	DSSTS	20FEB90	09:30	RTS	1	365	/AMERICAN/FOLSOM/FLOW-POWER/01JAN1989/1DAY/OBS/
18	T1	DSSPD	06MAR90	15:22	PD	1	52	/CALAVERAS/NEW HOGAN/STAGE-FLOW///USGS/
19	T2	DSSPD	06MAR90	15:22	PD	1	52	/CALAVERAS/NEW HOGAN/STAGE-FLOW/RT#13/30JUN1980/USGS
20	T6	DSSPD	06MAR90	15:22	PD	1	46	/DRY CR/NR GEYSERVILLE/STAGE-FLOW/RT#34/07MAR1986/USGS/
21	T5	DSSPD	06MAR90	15:22	PD	1	66	/DRY CR/NR GEYSERVILLE/STAGE-FLOW/RT #38/30SEP1987/USGS/
22	T8	DSSPD	06MAR90	15:22	PD	1	60	/DRY CR/NR LEMONCOVE/STAGE-FLOW///USGS/
23	T15	DSSPD	06MAR90	15:22	PD	1	14	/DRY CR/WARM SPRINGS/ELEV-AREA//01OCT1983/POLY/
24	T7	DSSPD	06MAR90	15:22	PD	1	62	/DRY CR/WARM SPRINGS/STAGE-FLOW///USGS/
25	T29	DSSTS	06MAR90	15:22	RTS	3	365	/FEATHER/QUINCY/PRECIP-INC/01JAN1982/1DAY/OBS/
26	T28	DSSTS	06MAR90	15:22	RTS	2	365	/FEATHER/QUINCY/PRECIP-INC/01JAN1983/1DAY/OBS/
27	T30	DSSTS	06MAR90	15:22	RTS	2	366	/FEATHER/QUINCY/PRECIP-INC/01JAN1984/1DAY/OBS/
28	T14	DSSTS	06MAR90	15:22	RTS	1	365	/FEATHER/SIERRAVILLE/PRECIP-INC/01JAN1986/1DAY/OBS/
29	T13	DSSTS	06MAR90	15:22	RTS	1	365	/FEATHER/SIERRAVILLE/PRECIP-INC/01JAN1987/1DAY/OBS/
30	T10	DSSTS	06MAR90	15:22	RTS	1	366	/FEATHER/SIERRAVILLE/PRECIP-INC/01JAN1988/1DAY/OBS/

Figure 1 - Catalog File

HEC-DSS Condensed Catalog of Record Pathnames in File MASTDB.DSS

Catalog Created on Mar 6, 1990 at 15:24 File Created on Jan 8, 1990
Number of Records: 30 DSS Version 6-DA
Sort Order: ABCFED

Tag	A Part	B Part	C Part	F Part	E Part	D Part
T3	AMERICAN	AT FAIR OAKS	FLOW	OBS	1DAY	01JAN1988 - 01JAN1989
T11	- - - -	BLUE CANYON	PRECIP-INC	OBS	1DAY	01JAN1987 - 01JAN1989
T26	- - - -	FOLSOM	ELEV	OBS	1DAY	01JAN1987 - 01JAN1989
T21	- - - -	- - - - - - -	EVAP-PAN	OBS	1DAY	01JAN1986 - 01JAN1988
T23	- - - -	- - - - - - -	FLOW-OUTLET	OBS	1DAY	01JAN1987 - 01JAN1989
T19	- - - -	- - - - - - -	FLOW-POWER	OBS	1DAY	01JAN1987 - 01JAN1989
T1	CALAVERAS	NEW HOGAN	STAGE-FLOW	USGS	(null)	(null)
T2	- - - -	- - - - - - -	- - - - -	- - -	30JUN1980	RT #13
T6	DRY CR	NR GEYSERVILLE	STAGE-FLOW	USGS	07MAR1986	RT #34
T5	- - - -	- - - - - - -	- - - - -	- - -	30SEP1987	RT #38
T8	- - - -	NR LEMONCOVE	STAGE-FLOW	USGS	(null)	(null)
T15	- - - -	WARM SPRINGS	ELEV-AREA	POLY	01OCT1983	(null)
T7	- - - -	- - - - - - -	STAGE-FLOW	USGS	(null)	(null)
T29	FEATHER	QUINCY	PRECIP-INC	OBS	1DAY	01JAN1982 - 01JAN1984
T14	- - - -	SIERRAVILLE	PRECIP-INC	OBS	1DAY	01JAN1986 - 01JAN1988

Figure 2 - Condensed Catalog File

3.8 Utility Programs

Utility programs have been developed to manipulate or display data stored in a DSS file. Several of these programs are briefly described below. Users manuals for these products follow this overview, with exception of SHFDSS/DSSSHF and PIP which are found in the Water Control Software Data Acquisition and Flood Damage Analysis Package documents, respectively.

1) DSSUTL - General Utility Program

DSSUTL enables a user to copy, delete, rename, or edit data in a DSS file. DSSUTL may also be used in the transfer of data from one computer site to another. Files can be "squeezed" to reduce storage space by eliminating inactive space caused by record deletion. These functions are implemented by simple commands represented by a two letter code. A sorted and numbered inventory of pathnames in the file can be obtained by using the Catalog (CA) command. The tags or numbers from the list can then be used in lieu of the pathnames when using DSSUTL or other utility programs.

2) DSPLAY - Graphical Display Program

DSPLAY allows the user to easily display data from a DSS file in a tabular or graphical form. Data is retrieved from the DSS file by specifying the pathname (or tag or reference number) of the data the user wishes to display. The time command can be used to define the starting and ending times of the data, if different than the implied block times. Up to seven curves can be plotted on one graph, and a plot can be split into upper and lower graphs for variables with different units. The user can "window" in to enlarge portions of a plot. Graphical data can be edited using cross-hairs and cursor keys. Graph scaling, labels and legends are automatic, unless redefined by the user.

3) REPGEN - Report Format Program

REPGEN automates the production of routine reports in a variety of settings. REPGEN provides for the retrieval and presentation of data from a DSS file or a text file in a pre-specified, user-designed format. The format is the equivalent of a blank form onto which variable information is entered in designated locations. REPGEN could be used, for example, in a water control operations setting to automatically produce reports showing the current stage and flow at selected locations in a river basin.

4) DSSMATH - Mathematical Manipulation of DSS Data

DSSMATH provides a means of mathematically manipulating DSS data in a variety of ways. DSSMATH associates a variable name with a set of data from a DSS file. Data can be retrieved, manipulated by arithmetic operations or functions, and stored back into the same or a

different DSS file. The user may add, subtract, multiply, divide or exponentiate uniform time series data. The user may compute many functions including the absolute value, truncate to whole numbers, round off numbers, remainder, square root, natural logarithm, minimum value, maximum value, mean value, sum of values, running accumulation, successive differences, standard deviation, skewness, correlation coefficients, standard error, interpolation by table lookup, derive different time series, perform Muskingum routing, straddle stagger routing, modified Puls routing, working R&D routing, and fill in missing data.

5) Data Entry Programs

A variety of utility programs are available for entering data into a DSS data base file. Some are designed to enter data from another data base. For example, WATDSS reads data from a file retrieved from the USGS WATSTORE system or a WATSTORE format file from a Compact Disk and enters it into a DSS file. Other programs are designed to enter data "manually" or in a generic form. For example, DSSTS is a prompt driven program for entering regular-interval time series data. A list of data entry programs is as follows:

Program	Purpose
DSSTS	Enter regular-interval time series data manually or from a file.
DSSITS	Enter irregular-interval time series data manually or from a file.
NWSDSS	Load daily and hourly climatological data (e.g., precipitation) from a National Weather Service tape or a NWS format file from a Compact Disk.
WATDSS	Load daily stream flow data from a WATSTORE file.
SHFDSS	Load data from a SHEF format (Standard Hydrometeorological Exchange Format). Companion program DSSSHF writes data from DSS into SHEF.
DWINDO	Edit or enter data in a full screen mode. This requires "forms" to be set up, and is intended for real-time data entry or editing.
DSSPD	Enter paired data manually or from a file.
PIP	Enter flood-damage paired data.
DSSTXT	Enter or display text data.

Chapter 4

Technical Information

4.1 Data Compression for Regular-Interval Time Series Data

Regular-interval time series data may be compressed with one or more of three methods. The method is selected by the user (or storing program) based on the kind of data that is to be stored.

The first method is a repeat counter scheme that flags duplicate values. It uses one bit per value to indicate a repeated value. It is often used for precipitation data, and can compress some data records by 97 percent. This compression method should not be used for data that is updated often (e.g., data stored in real-time), as it would likely cause the record to expand often and require excessive rewrites.

The second method compresses data by storing the differences between each value and the minimum value in the record. This is designed for data where the difference between the maximum value and the minimum value is not too large, and the precision of the values (the number of digits to the right of the decimal place) are known. The software uses the difference between the maximum and the minimum (or base) value, excluding missing data flags, and the precision number to determine the amount of space required for each number. If data is to be updated frequently with this method (e.g., entering real-time data in a master data base file), the base value and a storage size can be specified. This allows the software to update the data without having to recompute a base value and re-compress the record each time. Data compressed by this method are typically precipitation and stage values, and are compressed by 50 to 75 percent of their original size.

The third type of compression stores three significant digits for each value, and is often used to compress flow data. Data records compressed under with method are reduced in size by 50 percent.

The repeat method can be used in combination with the differences method or significant digits method. The differences method may not be used with the significant digits method.

4.2 DSS File Parameters

DSS file size parameters may be set for new files (or squeezed files) where the future size and type of the file is known. The DSS software has two methods to find a record from the pathname. In the first method, the software uses the pathname to look in a "dynamic" hash table for an index address, which points to a block containing the address of the actual data. This

method is intended for where the file size may vary considerably (e.g., a data base file used for computations), or where the future size is not known. In the second method, the hash table is bypassed, and the blocks are pre-allocated when the DSS file is first generated. This typically saves one disk access when retrieving a record, but it can be very inefficient with disk space if the size used is incorrect. This method is intended for somewhat stable data bases that do not change in size frequently (e.g., a master data base file).

The hash table size, or the number of pre-allocated blocks, can also be controlled to optimize storage and retrieval of data according to the expected number of records in the file. It should be noted that any of the sizes will operate with any number of records, but an incorrect size will not be as efficient as the correct size. When a user squeezes a file with the program DSSUTL, the size parameter is automatically adjusted based on the number of records in the file at that time. The eight possible sizes are as follows:

Size Name	Ideal Number of Records	Target Range of Records
TINY	20	1-50
EXTRA-SMALL	50	1-200
SMALL	200	100-1,000
MEDIUM (default)	1,000	200-5,000
LARGE	4,000	1,000-10,000
EXTRA-LARGE	10,000	2,000-20,000
HUGE	20,000	5,000-50,000
EXTRA-HUGE	50,000	>25,000

These items are generally set by the experienced user who desires control over the optimization of data base files. The default parameters are sufficient for most users.

4.3 DSS on MS-DOS Personal Computers

The utility programs mentioned in this document are available for personal computers running the MS-DOS operation system. The use of these programs are similar to their use on other computers. In order to conserve memory, some of the utilities (DSPLAY and DSSUTL) are overlaid and must be placed in a directory that is set in the path (by the AUTOEXEC.BAT file). If the error "Cannot find program" is printed, the program's directory is probably not in the path (this error will occur even if you have typed the full directory when executing the program). It is recommended that the utility programs be installed in the directory "\HECEXE", and this directory name be added to the path in the AUTOEXEC.BAT file.

The graphics DSPLAY program uses device "drivers". Drivers provide a means of plotting on several different types of devices (monitors, pen plotters, printers, etc.), without having information about each device in the program. The specific drivers used by DSPLAY are loaded into memory at execution time. More information about drivers is included in the installation instructions provided with the DSPLAY program diskettes.

DSSUTL and DSPLAY sort catalog files by using either the DOS sort utility, or an external sort program that access extended memory (depending on the size of the catalog file). There must be at least 550 Kbytes of free conventional memory available in order to load the memory extender program and the sort program. If there is insufficient free conventional or extended memory, catalog files will not be sorted. The sort program (GNUSORT.EXE) and the memory extender (DOS4GW.EXE) must reside in the directory \HECEXE.

Several users can access a DSS file on a network server at the same time, if a multiple user access flag is set and the network server provides correct file locking capabilities. Unfortunately, it has been our experience that many network servers do not implement adequate file locking capabilities. Without file locking, the internal addressing tables in a DSS file can be overwritten when two users write new records to the file at approximately the same time. To prevent this damage, DSS file accessed from MS-DOS are opened by default in a single user mode. For network servers where file locking capabilities are implemented, a flag may be set to cause DSS files to be opened in a multi-user access mode. This flag is set by creating the file "\HECEXE\SUP\HECDSS.CFG", and entering the string "`MULTI-USER ACCESS`" on the first line (without quotes). This change will apply to all programs on that computer which access DSS.

4.4 DSS on UNIX Workstations

Most of the DSS utility programs are located in the directory "/usr/hec/bin" (or a link to this directory). Help and other supplemental files are located in the directory "/usr/hec/sup" (or linked directory). The graphics DSPLAY program will display plots on an X terminal device, or write graphics to a postscript file.

Multiple user accesses are accomplished by utilizing the system lock demon. If DSS is to be used in a NFS network mounted drives environment, compatible file-locking software (e.g., the lock demon) must be run both on the client and server machines to provide multiple user access to DSS files. This applies to drives mounted from another UNIX workstation, or from a DOS-PC. (Refer to section 4.3 for NFS access from MS-DOS.)

Appendix A

Data Conventions

Applications of hydrologic software generally involve use and generation of large numbers of data. The adoption of logical, consistent conventions for defined pathnames associated with the data can lead to (1) the capability to readily recognize from a pathname the type and other characteristics of information contained in a record, (2) a reduction in error associated with the storage and retrieval of information, and (3) the capability to readily adapt available macros, screens, etc. for working with standard types of data.

1. Time Series Conventions

This section covers the conventions for both regular-interval and irregular-interval time series data. There are four data "types" recognized by the DSS. They are:

Data Type	Example
PER-AVER	Monthly Flow
PER-CUM	Incremental Precipitation
INST-VAL	Stages
INST-CUM	Precipitation Mass Curve

Both regular and irregular interval time series record pathnames have the same A, B, C, and F parts. Data blocks are labeled with a six part pathname. The parts are referenced by the characters A, B, C, D, E, and F, and are separated by a slash "/", so that a pathname would look as follows:

/A/B/C/D/E/F/

The conventions for the part labels are:

Part A - Group

The A part of the time series pathname describes the general group of the data. It may be a watershed name, a study name, etc., that would cause all associated records to be recognized as a group. The A part is not required.

Part B - Location

This part is the basic location identifier of the data. Generally the site name is used as the B part. A similar identifier, such as a project ID, USGS gage ID, or NWS station ID may be used. The B part is required.

When a hydrograph is routed from one location to another, the recommended B part is LOC1.LOC2, where LOC1 is the identifier <u>to</u> which the flow are routed, and LOC2 is an identifier for the location <u>from</u> which flows are routed. The second location (.LOC2) is optional. For example, it might be left out in situations where there is only one routed hydrograph for a location.

Part C - Parameter

The C part identifies the basic data parameter. A dash is used as a sub separator if further identification is needed. Additional information about the parameter, such as how it was obtained (e.g. "OBSERVED"), should be given in the F part. Examples of valid parameters are:

```
FLOW
ELEV
PH
PRECIP-INC
STAGE
TEMP-AIR
TEMP-WATER
```

Recommended C-parts for flows associated with stream locations are as follows:

C Part	Description
FLOW	Total flow.
FLOW-LOC	Local flow; that is, flow generated only from the subbasin that has an outlet at the specified location.
FLOW-CUM LOC	Cumulative local flow. This is the flow from all subbasins downstream from the nearest upstream reservoirs.
FLOW-COMB	Combined flow. This is the total flow minus the local flow.
FLOW-ROUT	Routed flow.
FLOW-DIVERT	Flow diverted out of the river at this location. The remaining flow would be labeled with a B-part of FLOW.
FLOW-mod	mod is a user-specified modifier for the flow. For example, FLOW-POWER, to designate a hydrograph from a power plant.

Recommended C parts associated with reservoirs are as follows:

C Part	Description
FLOW-IN	Reservoir inflow.
FLOW-OUT	Reservoir outflow.
FLOW-mod	mod is a user-specified modifier to flow associated with a reservoir, e.g. FLOW-IN1 for a component of reservoir inflow.
ELEV-RES	Reservoir elevation.
STORAGE	Reservoir storage.

Part D - Block Start Date

The D part of the pathname identifies the starting date of the data block. It is described for regular-interval and irregular-interval data in the sections that follow.

Part E - Time Interval or Block Length

The E part defines the time interval for regular-interval data, or the block length for irregular-interval data. It is further described in the sections that follow.

Part F - Descriptor

This part of the pathname is used to provide any additional information about the data. Its use may vary from application to application as appropriate, and may contain several additional qualifying pieces of information separated by a dash "-". If several forms of data exist, such as OBSERVED or FORECAST, and PLAN A or TEST 2, etc. they may be reflected in the F part. Generally the order of multi-descriptors of Part F should be from most to least significant.

A. Regular-Interval Time Series Conventions:

Regular-interval time series data is stored in "standard size" blocks whose length depends upon the time interval of the data. For example, daily time interval data are stored in blocks of one year (365 or 366 values), while monthly values are stored in blocks of ten years (120 values). If data does not exist for a portion of the full block, the missing values are set to the missing data flag "-901.0".

The starting and ending times of a block correspond to standard calendar conventions, e.g., for period average monthly data in the 1950's, the D part (date part) of the pathname would be `01JAN1950`, regardless of when the first valid data occurred (e.g., it could start in 1958). The 1960's block starts on `01JAN1960`.

Average period data values are stored at the end of the period over which the data is averaged. For example, daily average values are given a time label of 2400 hours for the appropriate day, average monthly values are labeled at 2400 hours on the last day of the month. If values occur for time other than the end-of-period time, that time offset is stored in the header array. For example, if daily average flow reading's are recorded at 6:00 am (i.e., the average flow from 6:01 am of the previous day to 6:00 am of the current day), then a offset of 360 (minutes) will be stored in the header array.

Part D - Block Start Date

The D part should be a 9-character military style date for the start of the standard data block (not necessarily the start of the first piece of data). Valid dates include `01JAN1982` for daily data, `01MAR1982` for hourly data, and `01JAN1900` for yearly data. Invalid dates include `01JAN82` (7 characters) and `14APR1982` for daily data (`14APR1982` is not the start of a standard daily block).

Part E - Time Interval

This part consists of an integer number together with an alphanumeric time interval specifying the regular data interval. Valid alpha entries are `MIN`, `HOUR`, `WEEK`, `MON`, and `YEAR`. The valid intervals and block lengths are:

Valid Data Intervals				Block Length
1MIN,	2MIN,	3MIN,		One Day
4MIN,	5MIN,	10MIN		
15MIN,	20MIN,	30MIN,		One Month
1HOUR,	2HOUR,	3HOUR,	4HOUR,	
6HOUR,	8HOUR,	12HOUR		
1DAY				One Year
1WEEK,	1MON			One Decade
1YEAR				One Century

Examples of regular-interval pathnames are:

a. Daily USGS observed flow for station 0323150 for calendar year 1954 might be named:

```
/USGS/0323150/FLOW/01JAN1954/1DAY/OBS/
```

b. Six-hourly forecasted flow may have a pathname of:

```
/RED RIVER/DENISION/FLOW/01JUN1954/6HOUR/FORECAST/
```

c. Monthly elevations for Broken Bow Dam produced by application program HEC5 may be named:

```
/ARKANSAS/BROKEN BOW/ELEV/01JAN1950/1MON/PLAN 2R-RUN 5/
```

B. Irregular-Interval Time Series Conventions:

The irregular-interval time series conventions are similar to the regular-interval conventions, except that an explicit date and time is stored with each piece of data, whereas in regular-interval time series the date and time are implied by the location of the data within the block. Data is stored in variable length blocks (regular-interval data is stored in fixed length blocks). The block lengths are days, months, years, and decades.

The number of values that may be stored in one record is indefinite, although it is prudent to choose a size that will be less than 1000. The user selects the appropriate block length. For example, if the data to be stored occurred once every 1-2 hours, a monthly block would be appropriate. If data was recorded once or twice a day, use a yearly block. One would not want to store data that occurred 6 or more times a day in a yearly block (about 2200 values), because that may exceed dimension limits in some DSS programs.

All data is stored in variable length blocks that are incremented a set amount when necessary. Initial space for 100 data values is allocated. Additional increments are for 50 data values, unless otherwise set. (When the 101st data value is added to the record, a new record with a length of 150 values is written.)

Part D - Block Start Date

The D part should be a 9 character military style date for the first day of the standard data block (not necessarily the start of the first piece of data). For example, data stored in a daily block beginning on March 23, 1952 at 3:10 p.m. would have a D part of `23MAR1952`. If the same data were stored in a monthly block, the D part would be `01MAR1952`. The same data in a yearly block would have a D part of `01JAN1952`, and as a decade block, `01JAN1950`.

Part E - Block Length

The E part indicates the length of the time block, i.e. whether it is a day, a month, a year, or a decade. It consists of the letters "IR-" concatenated with the block length:

```
IR-DAY
IR-MONTH
IR-YEAR
IR-DECADE
```

It should be noted that the same data may be stored in blocks of different lengths. The DSS stores these as different records, and treats them as a completely different data set.

An example of a pathname for irregular-interval data is:

```
/SANTA ANA/PRADO/FLOW/01JAN1988/IR-MONTH/OBS/
```

2. Paired Data (Curve Data) Conventions

Paired data is a group of data that represents a two variable relationship. Typical examples are data that make up a curve (e.g., a rating table or a flow-frequency curve). Several curves may be stored in the same record if one of the variables is the same. For example several frequency-damage curves may be stored in the same record, where the curves may be residential, commercial, etc. A scale associated with the variable may be one of three types: linear, logarithmic, or probability. The pathname part identifiers are as follows:

Part A - Group

This part is for general grouping of data. It may be a study name, watershed name, etc., that would cause all records associated in a certain way to sort together, be copied together, delete together, or otherwise recognized as a group.

Part B - Location

This pathname part is the basic location identification of the data. It may be a control point, damage reach ID, station ID, or other identifier.

Part C - Parameters

Because paired data represents a relationship between 2 parameters, this part should contain the 2 parameter names separated by a hyphen (-). Examples of parameters are:

```
ELEV-DAMAGE
ELEV-FLOW
FREQ-FLOW
STATION-ELEV
```

In the above examples, `ELEV`, `FREQ`, and `STATION` are referred to as the first or independent variable, while `DAMAGE`, `FLOW` and `ELEV` are the second or dependent variable.

Part D - Optional Descriptor

This part of the pathname is used to provide any further descriptions of the data. It may vary from application to application as appropriate, and is often null.

Part E - Time Descriptor

This part of the pathname is used only if the paired data is representative of a specific point in time such as a 1995 forecast condition, or rating curve of `21MAR1981`.

Part F - General Descriptor

The F part identifies a unique descriptor of the data such as the situation, condition, or alternative plan name associated with the data. This part is included in labeling of data in some output utilities.

An example of a pathname for paired data is:

```
/ALLEGHENY/NATRONA/ELEV-DAMAGE//1980/FLOOD PROOF PLAN B/
```

3. Text Data Conventions

Text data is defined as generic alpha-numeric lines of text, where each line is preceded by a carriage return character and ends with a line feed character. It does not, at this time, include other types characters, such as those that would be used to create a graphical display. There are no definitive size limitations for a DSS text record, but it is recommended that a record contain no more than about 300 lines of text. There are no conventions set for the structure of the pathname. However, it is recommended that the pathname parts be labeled in a descending order of importance, and that the pathname indicate that the record contains text data and not one of the other types of data.

Text data may be entered manually or from a file with the program DSSTXT. DSSTXT, as well as DSSUTL, can also display (or place in a DSS file) text data.

Appendix B

Glossary of Terms

ASCII - American Standard Code Information Interchange. A standard "format" that defines most of the character symbols on a keyboard. This generally refers to "text" files that can be transferred and understood between different kinds of computers.

Batch (processing) - A means of executing programs or job control instructions in a background environment. Generally the program names or instructions are given in a file, and output is automatically sent to the printer.

Block - A related set or series of data stored together. For example, one year of daily flows would comprise one time series block (or record). The next year would comprise another block.

Blocked File - A type of computer storage file on disk. Blocked files are sequential in nature.

Binary File - A type of file that is usually created specifically for one kind of machine. Binary files are "non-text" files, and include such files as executable programs and object files.

Catalog - A listing of record pathnames and relevant information in a DSS file.

Condensed Catalog - A special catalog file for time series data. The condensed catalog shows pathname parts and the time span for data sets.

Data - The values stored or retrieved in a DSS file (e.g., daily stream gage readings).

Delimiter - A symbol separating a string of characters. A blank, a comma, or a slash may act as a delimiter.

Direct Access File - A type of computer storage file. Data in this type of file can be referenced by its position. For example, to retrieve data in "line" 1000, the computer can go directly to line 1000 to obtain the data, whereas in a sequential file, the computer has to read lines 1-999 first.

DSS (HEC-DSS) - Data Storage System; the general name of the software developed by the Hydrologic Engineering Center for storing and retrieving data.

File - A location on the computer's disk or tape where a user stores a set of information. Each file is given a unique name identifying it.

Header - A set of information stored with each DSS record. A record may have more than one header. Headers may describe the data in that record (e.g., the data's units), data compression information, and information that the user entered.

Job Control Language (JCL) - The means of a user giving instructions directly to the computer.

Interactive - Executing programs or JCL from a terminal or the keyboard.

Multiple User - A means of allowing several users to access and process data from the same DSS data file at the same time. A typical computer file will allow only one person to store and retrieve data at a time.

Option - A way of telling a program how to do something (not what to do it to). Used with commands in several DSS programs. Generally the option is separated from the command by a period.

Parameter - An instruction to a program telling it what to operate on. For example, in DSPLAY or DSSUTL, the pathname or pathname number following a tabulate command is the parameter. The parameter is generally separated from the command by a comma or blank.

Pathname - An unique identifier label for a block of DSS data. The pathname may consist of up to 80 characters and is, by convention, separated into six parts with a slash "/" delimiter. When accessing data in a DSS file, the data's pathname must be given.

Pathname Part - A conventional pathname is separated into six parts, separated by slashes. Each part is identified by one of the letters A, B, C, D, E, or F. The part labels are given in the section "Data Conventions".

Random File - A file allowing reading and writing by multiple users at the same time.

Record - For DSS usage, a record is equivalent to a block of data, i.e., all data stored under one pathname is called a record. For example, a year of daily flows would compromise one time series record.

Reference Number - A listing number for a pathname from the catalog file. The reference number can often be used in place of the complete pathname in DSSUTL and DSPLAY. These numbers may change each time a new catalog is created.

Routine (subroutine) - A set of instructions called by a program. All DSS subroutines are identified by a "Z" as the first character of their name.

Sequential Access File - A type of computer storage file. Data in this type of file must be obtained sequentially. For example, to retrieve data in "line" 1000, lines 1-999 must be read first.

Tag - A one to eight character "semi-permanent" record identifier for interactive use. A record's tag may often be used in place of the complete pathname in DSSUTL and DSPLAY. A tag may be set by the user or by a program.

Version (DSS software) - The software version indicates which group of DSS software created the file. The version appears at the top of the catalog, and is a number followed by two letters (e.g., 6-EA). The number changes on a substantial overhaul of the software. The first letter changes for each major change in the software, and the second letter for every minor change. DSS files with different version numbers are not compatible with each other, while files only with different version letters are compatible.

Word (computer word) - The basic storage element of the computer system. Any number stored (in a non-compacted form) takes at least one word. The length of a word is usually measured in bytes (the storage required for one character). On the PC, one word requires 2 bytes, while on the UNIX computers, 4 bytes comprise one word.

ZCLOSE - A DSS software subroutine that updates and releases a DSS file from a program.

ZOPEN - A DSS software subroutine that attaches a DSS file to a program and obtains basic information from the file, allowing reading and writing to proceed.

ZREAD - A DSS software subroutine that retrieves data from a DSS file based on the pathname used. When data is retrieved, a message is printed giving the pathname of the record read (a program can suppress this message).

ZWRITE - A DSS software subroutine that stores data in a DSS file based on the pathname used. ZWRITE prints out a message indicating which record was written and its version number.

DSSUTL

Hydrologic Engineering Center
Data Storage System Utility Program

User's Manual

Version 6.8
March 1995

Hydrologic Engineering Center
U.S. Army Corps of Engineers
609 Second Street
Davis, California 95616-4687
(916) 756-1104

DSSUTL General Commands

AR Archive
CH Check
DC Data Compression
DP Display Pathnames
ED Edit
FC File Check
FQ File Query
HE Help
IM Import
PA Pathname
RN Rename
SQ Squeeze
TA Tabulate
UD Undelete
WR Write Data

CA Catalog
CO Copy
DE Delete
DU Duplicate
EF Exchange Format
EV Exchange Variable
EX Export
FI Finish
FO Format
OP Open
RE Read Data
RT Record Tag
ST Status
TI Time

DSSUTL

Table of Contents

Chapter **Page**

1. Introduction 1
   - 1.1 Program Execution 1
   - 1.2 Command Structure 1

2. Differences from Previous Version 3
   - 2.1 New Commands 3
   - 2.2 Additional Command Capabilities 3

3. General Commands 5
   - 3.1 Archive 5
   - 3.2 Catalog 7
   - 3.3 Check 10
   - 3.4 Copy 11
   - 3.5 Data Compression 13
   - 3.6 Delete 15
   - 3.7 Display Pathnames 16
   - 3.8 Duplicate 17
   - 3.9 Edit 18
   - 3.10 Exchange Format 20
   - 3.11 Exchange Variable 26
   - 3.12 Export 28
   - 3.13 File Check 29
   - 3.14 Finish 30
   - 3.15 File Query 30
   - 3.16 Format 31
   - 3.17 Help 32
   - 3.18 Import 33
   - 3.19 Open 34
   - 3.20 Pathname 35
   - 3.21 Read Data 36
   - 3.22 Record Tag(s) 37
   - 3.23 Rename 39
   - 3.24 Squeeze 40
   - 3.25 Status 42
   - 3.26 Tabulate 43
   - 3.27 Time 45
   - 3.28 Undelete 46
   - 3.29 Write Data 47

4. Secondary Commands 49
4.1 Close 49
4.2 Debug 49
4.3 Inquire 50
4.4 No Operation 50
4.5 Set 50
4.6 Verify 50

5. Selective Catalog Capabilities 51
5.1 Introduction 51
5.2 Sort Order 51
5.3 Selection of Pathnames Based on Last Written Date or Program 51
5.4 Selection of Pathnames Based on Pathname Parts 52

6. Export - Import 53
6.1 Introduction 53
6.2 Procedure 53
6.3 Examples 54

Chapter 1

Introduction

DSSUTL provides a means of performing utility functions on data stored in the Hydrologic Engineering Center's Data Storage System (HEC-DSS). These functions include tabulating, editing, copying, renaming, and deleting data. The program also offers the capability of formatting and copying data into an ASCII sequential file for transfer to another computer, or for use by a program without DSS capabilities.

1.1 Program Execution

The program is executed by entering its name. The program responds by asking for the name of the DSS file to access. On MS DOS personal computers, the name may be selected by moving the cursor to one of the names in the list that is displayed on the screen (which are files that end in ".dss" in the current directory), then pressing the carriage return. After the DSS file name has been entered, the program gives the prompt "U>", where the user enters commands. To terminate the session, the command FINISH (or just FI) is entered.

If desired, the commands may be specified in a separate input file, and the program run in a "batch mode". To accomplish this, the DSS file name and commands are placed into a file. Comments may also be placed in the input file by using an asterisks (*) at the beginning of the comment line. To execute the program with an input command file, enter:

```
DSSUTL INPUT=file-name
```

The DSS file and output file may also be designated on the execution line by specifying "`DSSFILE=dss-file`", and "`OUTPUT=output-file`".

On MS DOS personal computers DSSUTL must reside in a directory that is set in your path (e.g., \HECEXE), or reside in your current directory, otherwise an error of "Overlay not found" will occur.

1.2 Command Structure

A typical command for DSSUTL has the following structure:

```
Command.options, parameters
```

If a command uses an input or output file, and that file has not been specified in an earlier command, the structure is:

```
Command.options, TO=file-name, parameters
```

Command options (if used) follow the command and are separated by a period (.) and no spaces. If a file name is given, it follows any options separated by a comma and/or blanks. Once a file has been designated, it typically does not need to be given again. A comma and/or blanks separate any parameters from the command (or file name).

The last pathname accessed, and the time window (if set) are stored in a short-term memory. They may be displayed by the STATUS command. The time window will automatically be used by most commands, if it has been set.

Informative output for a single command may be suppressed in DSSUTL by preceding that command with a period (.) For example:

```
.RN C=FLOW
```

will suppress informative output during that command. The verify command will suppress informative output for all commands.

All commands, parameters and pathnames may be given in either upper or lower case. On UNIX computers, the case of filenames are retained.

Chapter 2

Differences from Previous Versions

The following is a brief list of the modifications made since the July 1990 version. For information on a particular change, refer to the documentation for the specific command.

2.1 New Commands

The following commands have been added:

FC -- File Check
Import - Export Commands:
EF -- Exchange Format
EV -- Exchange Variable
EX -- Export Data
IM -- Import data

2.2 Additional Command Capabilities

1. The selective catalog has been enhanced to select pathnames from a (non-abbreviated) catalog based on their records last written date or program name. This, for example, would provide a direct means for archiving records that were last written prior to a given date:

```
AR  TO=archive.dss  LW<20JAN92
```

2. The condensed catalog has been changed so that records that differ only by the date will occupy one line, even if there are missing records within the time span. The D (Date Span) option will override this and show only complete data sets (with regards to time) as a single line. For example, if there is 50 years of hourly records for a data set, and one or more records within that span are missing, that data will show up as two separate lines in the condensed catalog, if the "D" option is specified. This option must be used in combination with the "C" and the "N" options.

(This page intentionally left blank)

Chapter 3

General Commands

3.1 Archive

Use:

```
AR.option, TO=archive-file, parameters
```

The archive command copies records from the current DSS file to the archive file, then deletes those records from the current file. The archive file name, which is specified by "TO=" followed by the file name, only needs to be given on the first archive command (the same archive file will be used on subsequent archive commands). If the archive file is not given, the user will be prompted for it. The archive command operates on a record basis; any time-window set is ignored.

For time series data the "D" (Date) part of the pathname may reference an offset from the current date. For example, to archive hourly data that is three months old, the following command may be given:

```
AR TO=DSSOBS D=M-3M E=1HOUR
```

This command will archive all records with a pathname E part of "1HOUR", and a D part containing the current month minus 3 months. For example, if today is February 22, 1970, the "D" part would be "01NOV1969".

Options:

1. S (Squeeze) - This option squeezes the current file after the archive has been completed. This option should only be used on the last archive command for a file.

2. P (Preserve) - Records with the same pathname in the file being copied to will <u>not</u> be overwritten.

3. C (Compress) - Time series data is compressed according to the file compression method specified in the copy file (a time window does not need to be set). If a method is not set to match that pathname, it will be uncompressed.

Parameters:

1. None - Uses the pathname set in memory.

2. Pathname - If a pathname follows the command (or archive file name), that record is archived.

3. Tag(s) - Uses the record(s) corresponding to the tag(s) specified for archiving.

4. N1, N2, N3-N4 - Obtains pathnames with reference numbers N1, N2, and N3 through N4 from the current catalog file for archiving.

5. Date Reference (D=...) - Uses the current catalog and selects records to archive based upon a "D" part relative to the current date. The "D=" must be followed either by a D (for the current day), a M (for the current month), or a Y (for the current year), a minus (or plus) sign, a number, then another D, M, or Y. For example:

   `D=M-2M` (The current month minus two months)

   `D=D-60D` (The current day minus 60 days)

   The time interval (E part) is usually specified along with this parameter.

6. Selective Catalog (A=..., B=...) - Uses the selective catalog capabilities for obtaining pathnames for records to archive. For more information, refer to the selective catalog section.

Examples:

```
AR TO=DSSARC A=BIG BASIN, D=M-3M, E=1HOUR
AR NATP-F, NATP-P
AR.S A=AFOS, D=Y-2Y
AR TO=ARCHIVE.DSS  LW<01JAN92
AR PROG=DSSTS
```

3.2 Catalog

Use:

```
CA.options, parameters
```

The catalog command displays either the catalog, an abbreviated catalog or the condensed catalog. The catalog is a list of the record pathnames in the DSS file, along with their last written date and time and the name of the program that wrote that record. The catalog is usually sorted alphabetically by pathname parts. Each pathname has a record tag and a reference number, either of which may be used in place of the pathname in most of the commands. The tag(s) are semi-permanent but not necessarily unique. The reference numbers are unique, and are not permanently associated with the pathnames - they may change each time a new catalog is generated.

The name of the catalog file is the name of the DSS file with an extension of ".dsc". If the catalog file does not exist, it will be created. If the DSS file has changed, a new catalog must be created (.N option) to obtain the current list of pathnames in the DSS file.

The abbreviated catalog contains only the list of pathnames, their tag(s) and reference numbers. It can be displayed from a regular catalog (CA.A), or the catalog file itself can be made abbreviated (CA.AN).

The condensed catalog is a special catalog designed for time series data. The parts of the pathname (instead of the full pathname) are listed along with the date span for records where only the "D" (date) part varies. Repeating parts are replaced by dashes for easier reading. The condensed catalog is placed in a separate file named similar to the catalog, except it has an extension of ".dsd". The condensed catalog can only be created (or updated) when a sorted catalog is generated with the default sort order.

Options:

1. None - The catalog file is displayed one screen at a time for an interactive session. If DSSUTL is being run from a batch job, or a macro, the complete catalog is displayed.

2. A (Abbreviated) - This option will cause only the pathnames and their tag(s) and reference numbers to be displayed from the catalog (similar to the Display Pathnames command). This option is recommended for 80 column screens. If a new catalog is created, then the catalog file itself will be abbreviated.

3. C (Condensed) - This option will display the condensed catalog. If the condensed catalog does not exist, it will be created.

4. D (Date Span) - Normally, for a condensed catalog, if a record set has an incomplete data span (missing records in time), that catalog line is marked with an asterisk. This option will instead cause a separate catalog line for each time span. This option must be used with the "C" and "N" options.

5. F (Full) - In an interactive session, this option displays the full catalog without pausing after each screen. The full catalog is automatically displayed in a batch job.

6. M (Map File) - When generating a new catalog, a "map" file will be created using an extension of ".dsm". A map file contains only pathnames, which is useful for creating input files for programs that use pathnames. The selective catalog feature may be used to limit which pathnames are listed.

7. N (New) - This option generates a new catalog. This option should be used after records have been added, deleted or renamed in the DSS file.

8. P (Printer) - This options sends the catalog file (or condensed catalog) to the line printer. On MS DOS personal computers, the printer should be ready before this option is used. The catalog file is also displayed on the screen, unless the ".S" option is also used.

9. U (Unsorted) - This option will cause the catalog not to be sorted when generating a new catalog. An unsorted catalog can be created much faster than a sorted one.

10. S (Suppress) - This option will cause the catalog file not to be displayed on the screen.

Parameters:

1. None - If a new catalog is created, the default sort order is used. This sort is alphabetical by pathname parts in the order A,B,C,F,E,D.

2. Order - The pathname part sort order may be specified when creating a new catalog. This is done by giving an "O=", followed by the part letter order. The part letters must not be separated, and those parts not specified are filled in using the default order. For example:

```
CA.N O=FB
CA.N O=FBACED
```

would generate the same sort order. Do not specify a sort order when generating a condensed catalog.

Remarks:

On MS-DOS computers, the catalog may be sorted by either the DOS sort utility, or an external sort program, named "GNUSORT.EXE", which uses extended memory. GNUSORT.EXE uses a memory extender named DOS4GW.EXE. Both of the extended memory programs must reside in the directory \HECEXE. Approximately 550 Kbytes of free conventional memory are required for the extended memory sorting. (The MS-DOS memmaker program will often help recover this memory.)

Examples:

`CA`	(Displays the catalog file)
`CA.N`	(Creates a new catalog, then displays it)
`CA.CN`	(Creates and displays the condensed catalog file)
`CA.A`	(Displays pathnames, their tag(s) and reference numbers only)
`CA.NPS`	(Generates a new catalog and sends it to the printer - the catalog is not displayed on the screen)
`CA.N O=FB`	(Creates a new catalog, sorting the F parts of the pathnames first, then the B parts of the pathnames)

3.3 Check

Use:

```
CH, parameters
```

The Check command determines if record(s) exists in the DSS file. If the record exists, various information about that record is displayed. If the record has been compressed, the amount of spaced saved is shown.

Parameters:

1. None - Uses the pathname set in memory.

2. Pathname - If a pathname follows the command, that record is checked.

3. Tag(s) - Checks the record(s) corresponding to the tag(s) given.

4. N1, N2, N3-N4 - Obtains pathnames with reference numbers N1, N2 and N3 through N4 from the current catalog file to check.

5. Selective Catalog (A=..., B=...) - Uses the selective catalog capabilities for obtaining pathnames for records to check. For more information, refer to the selective catalog section.

6. ALL - Checks all records in the current catalog file.

Examples:

`CH`	(Checks the current pathname in memory)
`CH NATP-F`	(Checks the record with a tag of "NATP-F")
`CH 2, 4, 6-9`	(Checks records for pathname numbers 2, 4, 6, 7, 8, and 9 from the catalog file)

3.4 Copy

Use:

```
CO.options, TO=copy-file, parameters
```

The copy command copies records from the DSS file opened to the DSS file whose name is designated by "TO=copy-file" following the command, or it will be prompted for if not supplied. If the DSS copy-file does not exist, it will be created. Once the copy-file name has been given, it does not need to be given for other copy commands in the same session.

For time series data, if a time window is set, then only data within that time window will be copied. If a time window is not set, the "D" (Date) part of the pathname may reference an offset from the current date. For example, to copy hourly data that is three months old, the following command may be given:

```
CO TO=DSSOBS D=M-3M E=1HOUR
```

The above command will copy all records with a pathname E part of "1HOUR", and a D part containing the current month minus 3 months. For example, if today is February 22, 1970, the "D" part would be "01NOV1969".

Options:

1. None - Any records with the same pathname in the file being copied to will be overwritten. For regular-interval time series data, unless a time window is set, the record's data compression is not changed.

2. P (Preserve) - Records with the same pathname in the file being copied to will <u>not</u> be overwritten.

3. C (Compress) - Time series data is compressed according to the file compression method specified in the copy file (a time window does not need to be set). (If a method is not set to match that pathname, it will be uncompressed).

Options for Regular-interval Time series Data:

Note: A time window must have been specified to use these options. The options are mutually exclusive and are only useful when copying data to another file with the same pathnames. They are infrequently used.

4. M (Missing) - Replace Missing data flags only. Only -901's (or non-existent records) will be replaced in the copy file.

5. U (Update) - Will not allow missing data flags (-901's) in the originating file to replace valid values in the copy file.

6. R (Regardless) - Copies all data within the time window given, including non-existent records or all missing data. Normally, if the data to be stored for one record is all missing (-901), that record is not written. The R option bypasses this, and will allow a record of all -901's to be copied.

Option for Irregular-Interval Time Series Data:

7. M (Merge) - If data for the same record already exists in the copy file, the M option will merge the two sets together. For example, if data in the copy file exists for Noon, 2 and 4 p.m., and data in the originating file is for 1, 3, and 5 p.m., the M option will cause the final data set to be for Noon, 1, 2, 3, 4, and 5 p.m.. Normally data is replaced.

Parameters:

1. None - Uses the pathname set in memory.

2. Pathname - If a pathname follows the command (or copy file name), that record is copied.

3. Tag(s) - Copies record(s) corresponding to the tag(s) given.

4. N1, N2, N3-N4 - Obtains pathnames with reference numbers N1, N2, and N3 through N4 from the current catalog file for copying.

5. Date Reference (D=...) - Uses the current catalog and selects records to copy based upon a "D" part relative to the current date. The "D=" must be followed either by a D (for the current day), a M (for the current month), or a Y (for the current year), a minus (or plus) sign, a number, then another D, M, or Y. For example:
   `D=M-2M` (The current month minus two months)
   `D=D-60D` (The current day minus 60 days)
   The time interval (E part) is typically specified along with this parameter.

6. Selective Catalog (A=..., B=...) - Uses the selective catalog capabilities for obtaining pathnames for records to copy. For more information, refer to the selective catalog section.

7. ALL - Copies all records in the file (does not use the catalog file). This is useful in merging two or more DSS file together. Any time window specified is ignored.

Examples:

```
CO TO=LHDSS D=Y-1Y, E=1YEAR, F=OBS
CO NATP-F, NATP-P
CO C=FLOW
CO LW>20JUN92
CO PROG=HEC1
CO TO=NEWFILE ALL
```

3.5 Data Compression

Use:

```
DC, parameters
```

The data compression command will display or set a file's default data compression methods for regular-interval time series data. This mode of setting data compression parameters is based on matching pathname parts for new data to be stored in the file. Data already stored may be re-compressed by the "C" option in the squeeze command. Compression methods may also be designated by the program that stores the data. Methods for as many pathname part specifiers as needed may be set in a DSS file.

Three methods of data compression are available. They are: 1) The repeat method, which flags repeated values; 2) The delta method, which stores the offset for each value from a base value, and; 3) The significant digits method, which stores only three significant digits for each data value. In addition, the repeat method and delta method can be used together, and the repeat method and the significant digits method can be used together. A complete description of data compression may be found in the DSS overview description.

The delta method requires a precision exponent parameter indicating the accuracy of the data. If the data to be stored is measured to the nearest hundredth (0.01) (e.g., precipitation), the precision exponent would be -2, to the nearest thousandth, the exponent would be -3.

In addition, a "base value" and "data size" parameter may be specified for the delta method. These parameters are typically only used with "real-time data", data that is being updated frequently, and only to increase the efficiency of storing future data. The base value is the expected minimum value that the data will obtain for that record. For example, the base value for incremental precipitation would be 0.0. The data size parameter indicates whether one or two bytes should be pre-allocated for each data value. One byte allocates a difference of 256 units, two bytes allocates a difference of 65,536 units. Typically, hourly precipitation would pre-allocate only one byte (up to 2.56 inches per hour), whereas reservoir elevations would pre-allocate two bytes (up to 65.536 feet difference). If the data changes so that either of the selected values are invalid, the software will automatically select new values (and re-compress the data). If these parameters are not specified, the software will automatically select values based upon the data.

If required parameters are not given, the user will be prompted for them. To remove a compression method, specify "NONE" as the method.

Parameters:

1. ? - The current default file data compression methods are displayed.

2. No parameters - The file's current default file data compression methods are displayed, then pathname part and data compression parameters are prompted for.

3. Pathname parts - One or more pathname parts to compare against the pathnames of new data for this type of compression. The part to match is specified by the part letter, followed by an equal sign then the part. Usually the "C" part (data type) is specified. The "at sign" (@) can be used as a wild character at the end (only) of a portion of a part, so that only the first piece is matched. For example, "C=FLOW@" would match pathnames with a C part of "FLOW", "FLOW-RES OUT", and "FLOW-NATURAL".

   a) Method - The compression method is defined by "METHOD=" (or the abbreviation "ME="), then the method (which can be abbreviated to the first three characters) or the method number. The possible methods and their numbers are:

      0. NONE (removes compression method)
      1. REPEAT
      2. DELTA
      3. REPEAT+DELTA
      4. SIG. DIGITS
      5. REPEAT+SIG.

   b) Precision Exponent (required for the delta method) - The precision exponent for the delta method is identified by "PREC=", then the exponent. For example to set the precision to thousandths, this parameter would be "PREC=-3". If this parameter is not included, it will be prompted for.

   c) Base (optional, only applies to the delta method) - The base value is indicated by "BASE=" followed by the base number.

   d) Size (optional, only applies to the delta method) - The pre-allocation data size is specified by "SIZE=" followed a "1" to allocate one byte per value, or a "2" to allocate two bytes per value.

Examples:

```
DC ?
DC C=PRECIP@, METH=REPEAT+DELTA, PREC=-2
DC C=TEMP-WATER METH=5 BASE=32.0
DC C=FLOW@, F=OBS METH=3
DC C=STAGE, METH=DELTA, BASE=0.0, SIZE=2
DC C=ELEVATION, METHOD=NONE
```

3.6 Delete

Use:

```
DE, parameters
```

The delete command eliminates records from the DSS file. A record is not physically removed from the file until the "squeeze" command is given. Because of this feature, deleted records may be recovered by use of the "undelete" command. Refer to documentation on the undelete command for further information.

Parameters:

1. None - Uses the pathname set in memory.

2. Pathname - If a pathname follows the command, that record is deleted.

3. Tag(s) - Deletes record(s) corresponding to the tag(s) given.

4. N1, N2, N3-N4 - Obtains pathnames with reference numbers N1, N2, and N3 through N4 from the current catalog file to delete.

5. Date Reference (D=...) - Uses the current catalog and selects records to delete based upon a "D" part relative to the current date. The "D=" must be followed either by a D (for the current day), a M (for the current month), or a Y (for the current year), a minus (or plus) sign, a number, then another D, M, or Y. For example:
   `D=M-2M` (The current month minus two months)
   `D=D-60D` (The current day minus 60 days)
   The time interval (E part) is typically specified along with this parameter.

6. Selective Catalog (A=..., B=...) - Uses the selective catalog capabilities for obtaining pathnames for records to delete. For more information, refer to the selective catalog section.

Examples:

```
DE 2-5, 7, 9
DE NATP-F, NATP-P
DE F=COMPUTED
DE PROG=HEC1
DE LW<08FEB94
```

3.7 Display Pathnames

Use:

```
DP.options, parameters
```

The display pathname command lists pathnames, their tag(s) and reference numbers from the catalog file (similar to an abbreviated catalog). If the catalog file does not exist, it will be created.

Options:

1. None - Displays one screen of pathnames at a time. At the end of the screen, the user may enter a new command, or press a carriage return to continue the display. All the pathnames in the catalog are displayed in a batch or macro mode.

2. F (Full) - In an interactive session, this option displays all pathnames in the catalog without pausing after each screen.

3. N (New) - This option generates a new catalog before displaying the pathnames.

4. U (Unsorted) - This option will cause the catalog not to be sorted when generating a new catalog. An unsorted catalog can be created much faster than a sorted one (the default is to sort).

Parameters:

1. None - The pathnames are displayed, beginning with the first one.

2. N (N is a number) - Displays pathnames beginning with reference number N. If N is greater than the number of pathnames, then the last twenty pathnames are displayed.

3. N1-N2 - Displays pathnames with reference numbers N1 through N2.

4. Selective Catalog (A=..., B=...) - Uses the selective catalog capabilities to display pathnames based on their parts. For more information, refer to the selective catalog section.

Examples:

```
DP 9999            (displays the last 20 pathnames)
DP.F C=FLOW
DP LW>20MAR92
```

3.8 Duplicate

Use:

```
DU, parameters
```

The duplicate command duplicates records, giving the new records different pathnames. After giving the duplicate command and the associated parameters, DSSUTL will display a list of the pathnames to be duplicated, then prompt the user for new pathname part(s) or a new pathname for the new record. New parts are specified by entering the part letter followed by an equal sign then the part. If you change your mind, and you decide not to duplicate the data, the word "QUIT" may be entered to return to the main program. A time window may be specified to duplicate time series data.

Option:

1. P (Preserve) - Records with the same pathname in the file will <u>not</u> be overwritten.

Parameters:

1. None - Uses the pathname set in memory.

2. Pathname - If a pathname follows the command, that record is duplicated.

3. Tag(s) - Duplicates record(s) corresponding to the tag(s) given.

4. N1, N2, N3-N4 - Obtains pathnames with reference numbers N1, N2, and N3 through N4 from the current catalog file to duplicate.

5. Date Reference (D=...) - Uses the current catalog and selects records to duplicate based upon a "D" part relative to the current date. The "D=" must be followed either by a D (for the current day), a M (for the current month), or a Y (for the current year), a minus (or plus) sign, a number, then another D, M, or Y. For example:
   `D=M-2M` (The current month minus two months)
   `D=D-60D` (The current day minus 60 days)
   The time interval (E part) is typically specified along with this parameter.

6. Selective Catalog (A=..., B=...) - Uses the selective catalog capabilities for obtaining pathnames for records to duplicate. For more information, refer to the selective catalog section.

Example:

```
DU C=FLOW, F=OBS
F=OBS-RAW
```

(Duplicate all observed flows giving the records a new "F" part.)

3.9 Edit

Use:

```
ED.options, parameters
```

The edit command provides a means of editing data in the DSS file. The data is written in a formatted form to a temporary scratch file, edited by an external editor, then read back in and stored in the DSS file. Several records may be edited at one time.

DSSUTL will use the COED program as the editor, unless the "A" option is specified. Normal COED commands are used to edit the data. When complete, the user enters FILE to save the revisions (or QUIT to disregard the revisions), which transfers control back to DSSUTL. DSSUTL then reads the data from the edit file and stores it in the DSS file. If the file was not modified (i.e., the user entered QUIT), the data will not be read back in. The data in the edit file is read in a free format style.

For regular-interval time series data, the first part of each line is used for the date and time of the first piece of data in that line. This date and time is for informational purposes only, and is **ignored** when the data is read in. The starting date and time printed in the header (just below the pathname) is used to determine the date and time of the first piece of data. The date and time of all other data is implied by its position from the first piece of data (you cannot just add or delete data anywhere). To delete data, replace the data value with the letter "M" or a -901. (including at the end of the data set). Missing data at the beginning and end of the data set are suppressed, unless either a time-window was specified (via the TIME command), or the "C" option is given.

Irregular-interval time series data may be deleted anywhere within the data set, including at the beginning or end. If data is to be added or deleted from a paired data set, the number of ordinates shown in the header must also be modified to reflect any changes.

If no format was specified (using the FORMAT command), the program will select one based on the magnitude of the data.

Options:

1. None - COED is used as the editor.

2. A (Alternative editor) - When this option is specified, the user is transferred into the computer's operating system. At this point, the selected editor should be used to edit the file specified by DSSUTL. At the end of the edit session, control should return to DSSUTL. On MS DOS personal computers, the user must type the word "EXIT" to return to DSSUTL.

3. N (No Quality Flags) - If the record has data quality flags associated with it, this will cause the flags not to be edited (otherwise they will be shown on a separate line in a hex format).

Options for Regular-Interval Time series Data:

4. C (Complete) - All data within the record, including leading and trailing missing data (-901's), are written to the edit file. This option ignores any time window set.

5. D (Data only) - If a time window has been set, then this option causes missing data at the beginning and ending of the window not to be edited. If no time window is given, this option is automatically used.

Parameters:

1. None - Uses the pathname set in memory. For time series data, if a time window has been set, the data within that time window are edited

2. Pathname - If a pathname follows the command, that record is edited.

3. Tag(s) - Edits data for the record(s) corresponding to the tag(s) given.

4. N1, N2, N3-N4 - Obtains pathnames with reference numbers N1, N2, and N3 through N4 from the current catalog file for editing. Any time window set will be adhered to if the data is time series.

5. Date Reference (D=...) - Uses the current catalog and selects records to edit based upon a "D" part relative to the current date. The "D=" must be followed either by a D (for the current day), a M (for the current month), or a Y (for the current year), a minus (or plus) sign, a number, then another D, M, or Y. For example:
   `D=M-2M` (The current month minus two months)
   `D=D-60D` (The current day minus 60 days)
   The time interval (E part) is typically specified along with this parameter.

6. Selective Catalog (A=..., B=...) - Uses the selective catalog capabilities for obtaining pathnames for records to edit. For more information, refer to the selective catalog section.

Examples:

```
ED 2, 3, 5-9
ED NATP-F, NATP-P
ED.C C=FLOW
ED.A B=NORTH SHORE
```

3.10 Exchange Format

Use:

```
EF.options, format
```

The exchange format command defines the format to use for both the export and import commands. The format line indicates which and how exchange variables are to be imported or exported, their dates and times, and pathname parts. The format set will be displayed if a question mark follows the command (i.e., "EF ?").

The exchange format may be up to 256 characters in length. Because long lines are difficult to edit, two forward slashes (//) at the end of the format line indicate that the next line is a continuation of the current format line. DSSUTL will delete the two slashes before appending the next line (which should begin without a command). As many continuation lines may be used as needed, as long as the exchange format does not exceed 256 characters.

Although the exchange format is used for both the import and export commands, the use of the format may be different for these uses. Additional information concerning the exchange format may be found in Chapter 6.

Options:

1. None - The primary format for both the export and import command is defined (unless a question mark follows the command).

2. H - The header (title) format for the export command is defined, or instructions on how to skip a header in the import file is given.

3. M - Indicates how missing data values should be printed in the export file, or what string constitutes missing data in the import file.

<u>Export</u>:

DSSUTL essentially copies the exchange format to the export file, replacing exchange variables (enclosed in square brackets "[]") with data and any reserved variable with the appropriate information. All characters that are not recognized are copied "as is". Thus, placing quotes around a pathname part reserved variable (on the outside of the brackets), and putting commas between each field will produce an ASCII delimited file. Spaces and all other character are copied as is to the export file.

Export Exchange Variables

Exchange variables, which may be located anywhere on the exchange format line, must be enclosed within square brackets "[]" (note that the variable names are not enclosed in brackets when defined with the EV command). An example of such an exchange variable might be "[STG]". If a certain output format is required (e.g., the number of decimal places needed), that format may be specified within the brackets by following the variable name with a colon (:) and the required format. The format for floating point numbers is *columns.precision*, where *columns* is the total number of columns the number will occupy, and *precision* is the number of digits to the right of the decimal to show. Decimal numbers may be printed with a format of I*columns* Examples of valid formats are:

`[STG:.3]`	Stage with a precision of 3 digits to the right of the decimal will be printed.
`[STG:12.3]`	The number will occupy 12 columns, with 3 digits to the right of the decimal printed (blanks will occupy unused columns to the left). This format will cause numbers to line up.
`[STG:I6]`	Print the number as an integer value, occupying 6 columns.

If no format is given, two digits to the right of the decimal are printed.

Export Reserved Variables

The date, time, and pathname parts for data values may be printed in the export file by use of a reserved variable. Reserved variables must be enclosed in square brackets ([]). The reserved variables for exporting are:

```
[APART]
[BPART]
[CPART]
[DPART]
[EPART]
[FPART]
[DATE]
[TIME]
[T:n]        (where n is a column number to tab to)
```

The pathname parts to be printed in the export file are for the data value <u>following</u> that variable. For example, in the export format

```
[APART] --  [BPART], [STG1];  [BPART], [STG2];  [BPART], [STG3]
```

"[APART]" will be replaced with the A part of the pathname for the exchange variable STG1, and the "[BPART]" will be replaced with the B part of the pathname for the variable following that reserved variable.

A pathname part can be explicitly printed by following the part identifier with a colon and the exchange variable name for that part, all within the square brackets. For example:

```
[APART:STG2] -- [BPART:STG1], [STG1];  [BPART], [STG2];  [BPART], [STG3]
```

The string "[APART:STG2]" will be replaced with the A part of the pathname for the exchange variable STG2.

The "[DATE]" and "[TIME]" reserved variables will be replaced with the date and time for the data on that line in the export file. The default date style used is a nine character military style date (e.g.,07JAN1991). The default time style is a 24 hour clock time (e.g., 1630), with midnight reported as "2400" (not "0000").

A date and time style may be defined by following the word DATE or TIME with a colon and a "picture" of the date or time. The picture consists of letters that are replaced with parts of the date or time. For example, the variables

```
[DATE:Www, Mmm D, YY]  [TIME:H:MM am/pm]
```

might be replaced with

```
Mon, Jan 7, 91   2:00 pm
```

The valid strings for the date picture are:

String	Replaced with
D	The day of the month, composed of one or two digits (e.g. "7")
DD	The day of the month, composed of two digits (e.g., "07")
M	The month number, expressed as one or two digits (e.g., "5" for May)
MM	The month number, expressed as two digits (e.g., "05")
MMM	The three character abbreviation of the month (e.g., "JAN"). Lower case and upper case characters are respected (e.g., Mmm will be translated to "Jan")
YY	The year, expressed as two digits (e.g., "91")
YYYY	The year with the century included (e.g., "1991")
W	The day of the week, expressed as a single digit (with 1 as Sunday)
WWW	The three character abbreviation of the day of the week (e.g., "TUE"). Lower case and upper case characters are respected (e.g., Www will be translated to "Tue")
JJJ	The julian day of the year (e.g., " 32" is February 1)
JJJJJ	The julian day since Jan 1, 1900 (e.g., "33269" represents Feb. 1, 1991)

The valid time strings are:

String	Replaced with
H	The hour of the data, composed of one or two digits.
HH	The hour of the data, composed of two digits.
MM	The minutes portion of the time, composed of two digits.
SS	The seconds of the data (set to zero)
AM/PM	If this string is present, the hours are reported in twelve hour clock time, and the appropriate characters are printed. If this string is not present, the hours are printed in 24 hour time. The AM/PM string may be in lower case, and may contain periods. (e.g. "a.m./p.m.")

The date and time pictures may contain other characters as desired. These other characters often include commas, blanks, colons, slashes, etc..

Moving to an explicit column is accomplished by using the tab variable. The tab variable is "[T:n]", where n is the column number to tab to. The tab variable will cause the character following the specifier to be in the column given (regardless if that character is part of a data value, a blank, or any other character). (A tab character will be used if a real tab character is inserted in the format.)

Export Header Format

A single header (or title) line can be written to the export file by providing a header format. The header format follows the same conventions as the regular exchange format, and is specified by using the "H" option with the EF command (e.g., "`EF.H`"). Pathname part, date, time and tab reserved variables may all be given as discussed previously. However, the pathname part reserved variables should explicitly identify the exchange variable (e.g., "`[BPART:STG1]`"). A date or time reserved variable will use the date and time of the first data value. Exchange variables should NOT be given in the header format. An example header format command might be:

```
EF.H [T:5]Lake Report for [BPART:ELEV], on [DATE:Mmm d, YY] //
[T:40][CPART:ELEV] [T:50][CPART:INFLOW] [T:60][CPART:OUTFLOW]
```

Export Missing Data Definition

If the "M" option is used (e.g., "`EF.M`"), the character string following the command will be printed for missing data (instead of the number -901). The string will be centered within the space normally occupied by the number. Examples of missing data definitions are:

```
EF.M -
EF.M Miss
```

Import

Lines read with the import command are interpreted in fields, delimited by commas and (or) blanks. Pathname parts that might contain blanks should be enclosed in quotes (single or double) in the import file. If the parts do not contain blanks, quotes are not necessary.

The items that may be specified on the Exchange format line for importing data are:

```
[exchange variable]
[APART]
[BPART]
[DATE]
[TIME]
[SKIP]
```

Exchange variables do not have format styles associated with them, as used in the export command.

The A part and the B part may be read from the import file. Only one A part and B part may be specified in the format line. These parts will override the parts given in exchange variable pathname. Reading A and B parts from the import file is useful when importing a file with sections of data from different areas (e.g., an import file containing all the lake reports in a basin). If an import file has different B parts on the same line, the B parts must be skipped, and the data identified with different exchange variables (that have pathnames with the correct B part).

If desired, the date and time of the data can be read from the import file, if the file contains both the date and time. If the import file does not contain dates and times (or you do not wish to use those), and the data is regular-interval time series data, the date and time of the first data value must be identified with the DSSUTL TIME command. If the date field contains blanks (e.g., "Jan 7, 1990"), it should be enclosed in quotes. A date and time field must be given on each import line for data to be stored as irregular-interval time series data. (The E part of the pathname identified by the Exchange Variable command identifies whether the data is regular interval or irregular-interval).

All other information in the import file that will not be imported, must be identified by the [SKIP] variable. The [SKIP] variable indicates that the field is unused, and should be ignored by DSSUTL. A field is a series of characters, separated by a comma and/or blank. If quotes surround several strings, all the characters within the quotes should be treated as a single field. Several fields can be skipped with a single [SKIP] by following the "P" with a colon and the number of fields to skip (e.g., [SKIP:n], where n is the number of fields).

Example:

If the following line is typical in a file to be imported:

```
COE "SOUTH LAKE" AM REPORT: 20MAY90, 0800; ELEV:  342.123, PRECIP: 0.15
```

The following exchange format might be given to import that file:

```
EF [SKIP] [BPART] [SKIP:2] [DATE] [TIME] [SKIP] [ELEV] [SKIP] [PREC]
```

Skipping Import Header Lines

If the import file contains lines that are not to be imported, the header exchange format can be used to indicate which lines to ignore. To search the import file for the line to begin importing data on, follow the header exchange format with the word LOCATE, then the string to search for exactly as it appears in the file. For the example above, the following command can be used:

```
EF.H LOCATE AM REPORT
```

(note that there are no quote marks around the string AM REPORT).

If a known number of lines at the top of the import file are to be ignored, follow the exchange format header command with the word SKIP, then the number of lines to skip. For example, to skip 12 lines at the top of the file, use the command:

```
EF.H SKIP 12
```

Also, lines with a specific character string will be ignored if that string follows the word SKIP. For example, to ignore all lines that contain the string "Daily Status" (assuming that this is a title in the import file), use:

```
EF.H SKIP Daily Status
```

(again, note that there are no quotes around Daily Status.)

Defining Missing Import Data

If the import file contains a flag for missing data (such as a string of dashes), the missing data exchange format can be used to define that flag so that the DSS missing data flag (-901) will be used for that value. The exact string that flags missing data should follow the EF.M command. For example:

```
EF.M  ---
EF.M  -9999
```

In both of these example, a -901 value will be used instead of dashes, as in the first example, or instead of the number -9999 as in the second example. Only one missing data flag can be defined for an import file.

3.11 Exchange Variable

Use:

```
EV.option name=pathname, UNITS=units, TYPE=type
```

The exchange variable command sets a user defined short name to be equivalent to a pathname for exporting or importing data. The variable name may then be given in the exchange format (EF command), enclosed in square brackets ([name]). The exchange variable is nothing more than an abbreviation for a pathname.

The variable name can be up to eight characters long, and cannot contain any commas or blanks. An equal sign (=) must follow the variable name, then the pathname (or pathname reference) that is assigned to that name. If data is to be imported, the string "UNITS=units, TYPE=type" should follow the pathname with the appropriate units and type of the data. If data is exported, the units and type are ignored.

Up to 50 exchange variables can be declared at any time, and an exchange variable may be redefined. However, memory is equally divided among all exchange variables, so it may be prudent to clear variables (with the .R "reset" option) if a new set of variables will be defined.

The exchange variables and their associated pathnames defined may be displayed by entering a question mark following the EV command (e.g., "EV, ?"). Additional discussion on exchange variables may be found in Chapter 6.

Option:

1. R - Resets and clears all exchange variables

Parameters:

1. ? - Displays a list of all exchange variables defined.

2. EV *variable-name=pathname-reference* [UNITS=*units,* TYPE=*type*]

   Valid pathname references include:

   a. The pathname itself.

   b. Pathname parts identified by the part letter and an equal sign (e.g., `B=NATP, C=FLOW`). The other pathname parts are obtained from the last pathname referenced (which could be in a previous exchange variable).

   c. The record's tag.

d. The pathname's catalog reference number.

Because the set of commands to import and export data are often defined in an input file or PREAD macro file, the first two procedures are preferred over the last two. The "E part" of the pathname determines whether the data is regular-interval or irregular-interval time series data.

Examples:

For exporting data:

```
EV FLOW=/ALLEGHENY/NATP/FLOW/01JAN1990/1HOUR/OBS/
EV STG=C=STAGE
EV PREC= C=PRECIP-INC, E=IR-MONTH
EV DO=24                    (gets pathname 24 from the catalog file)
```

For importing data:

```
EV STG1=/ALLEGHENY/NATP/FLOW/01JAN1990/IR-MONTH/OBS/ UNITS=FEET TYPE=INST-VAL
EV STG2= B=PITT, E=1HOUR, UNITS=FEET TYPE=INST-VAL
EV STG3= B=FRKP, E=1DAY, UNITS=FEET, TYPE=PER-AVER
```

3.12 Export

Use:

```
EX.option, export-file
```

The export command causes data to be exported to a file whose name follows the export command. If the export file does not exist, it will be created. If the file name is not supplied, DSSUTL will prompt the user for it.

The export command uses the exchange format and exchange variables previously defined. The data is read from the DSS file when the export command is issued (not when the exchange variable command is given). Typically, a time window is designated prior to the export command. This time window applies to all pathnames given. For more information on exporting data, refer to Chapter 6.

Option:

1. None - Exported data is written to the beginning of the file.

2. A - (Append) Exported data is written starting at the end of the file.

Parameters:

1. None - The export file name will be asked for. If data was just exported (in the same session), then this export will append to the end of the file.

2. file-name - If a file name is given, the data will be exported to that file. If the file does not exist, it will be created.

3. * - If an asterisk is used in place of a file name, the data will be printed on the screen (standard out). This is a good way to check the export format before exporting the data to a file.

Examples:

`EX walter.dat`	(exports data to file "walter.dat")
`EX.A lehigh.dat`	(appends data to file "lehigh.dat")
`EX *`	(exports data to the screen)

3.13 File Check

Use:

```
FC
```

The file check command searches the entire DSS file currently opened for any errors in the internal address tables. Any errors or inconsistencies in the internal tables are reported. Because an exhaustive search of the file is made, this procedure will require a significant amount time (about the same amount of time required for a squeeze).

The DSS software has been designed with a high regard for file integrity. Generally a DSS file should not be subject to damage, even from a system crash or power outage. However, some of today's operating systems buffering schemes to improve computer performance can defeat some of the DSS safe guards in the event of a crash. If you experience a crash, check any DSS files in use at that time with this command as soon as possible. Most all data can be recovered from a damaged file by the SQUEEZE command.

If you find any errors in a DSS file that were not the result of a system crash or similar failure (on DOS use CHKDSK to search for lost links), make a backup copy of that file. Then notify HEC of the error and any conditions relevant to use of that file (e.g., what programs wrote to the file, is it on a network drive, etc.). The SQUEEZE command can be used to recover data from the file once a backup copy has been made.

3.14 Finish

Use:

```
FI
```

The Finish command terminates the execution of DSSUTL.

3.15 File Query

Use:

```
FQ
```

The file query command displays attributes of the current DSS files opened, including the file name, DSS version, number of records in the file and the file size.

3.16 Format

Use:

```
FO, (format)
```

The format command sets an output format to use for the write data , edit, or tabulate commands. The format follows the command name and must be a valid FORTRAN format of up to 40 characters. The format can be used to indicate how many data values should be printed on a line, the number of significant digits of the data, or a way of adding other alphanumeric information to a data line. If no format has been set, DSSUTL will chose a format based on the magnitude of the data. To clear a format (and cause DSSUTL to select one), enter just the format command (with no format following).

The format command may be used to produce data lines to be used as input for a program that does not have DSS capabilities. For example, if "IN" records are needed, the following format may be appropriate:

```
FO ('IN',F6.1,9F8.1)
```

The Write Data command is used to write the data to an output file. This file must be edited to remove the pathname, header and END DATA lines.

When regular-interval time series data is tabulated or edited (not the Write Data command), there may be only one format specified (e.g., `(8F10.4)`, not `(2F12.2,4F10.1)`). The number of data values per line may be reduced by the program in order to make them fit within the 80 column limit.

Examples:

`FO (F12.2)`	(One data value per line)
`FO (6F12.4)`	(Increases the number of significant digits)
`FO (T20,6F8.0)`	(Tabulate to column 20, then print the data)
`FO ('FLOWS',2X,6F10.1)`	(insert the word "FLOWS" at the beginning of each line)
`FO`	(Clears previously set format. The format used is selected by DSSUTL based on the magnitude of the data.)

3.17 Help

Use:

```
HE.option, parameters
```

The help command causes information about a command to be displayed. Entering just HELP provides a list of commands.

Options:

1. None - A short description of the command and its options are displayed.

2. A (All) - Displays all help information without paging. Normally, help information is displayed on the screen one page at a time.

3. E (Extended) - A comprehensive description of the command and its options are displayed.

Parameters:

1. None - A list of the available commands is displayed.

2. Command - A description of that command is displayed.

3. USAGE - A description of how to use help and commands is displayed.

4. SC (Selective Catalog) - Documentation on the selective catalog is displayed.

5. EX-IM - Describes the export-import capability.

Examples:

`HE`	(A list of commands is shown)
`HE.E ED`	(A complete description of the edit command is displayed)

3.18 Import

Use:

```
IM, import-file
```

The import command causes data to be read in from the file whose name follows the command, and stored in the DSS file opened. The entire import file is read in and data is stored in the DSS file according to the previously defined exchange format and exchange variables. If the data to be imported is regular-interval time series data, and the data's dates and times are not given in the import file, then the starting time of the data must be specified using the time (TI) command. Because the end of the time window is implied by the beginning time and the number of data, an ending time is not needed . Irregular-interval time series data must have a date and time for each line in the import file.

For more information on importing data, refer to Chapter 6 Export-Import.

3.19 Open

Use:

```
OP, filename, parameters
```

The open command opens the DSS file whose name follows the command. If no file name follows the command, a list of DSS files within the current directory is displayed. By moving the cursor to a file name using the space bar (or on DOS the cursor keys), then pressing the return key, that file will be opened.

File size parameters may be set for new files to be opened. These are intended for use where the size and type of DSS file to be generated is known.

Parameters (New Files Only):

1. Table Type - A "stable" hash (address) table type file will be generated if the parameter "TABLE=STABLE" follows the file name. This type of addressing table is primarily intended for data bases that do not change in size frequently (e.g., a master data base file). A "dynamic" hash table is the default.

2. Size - The hash table size can be controlled to optimize storing and retrieval of data according to the expected number of records to be stored in the file. (This is automatically done during a "squeeze".) The size parameter is specified by "SIZE=", then the expected number of records in the file, or one of following sizes:

Size Name	Target Number	Target Range
TINY	20	1-50
EXTRA-SMALL	50	1-200
SMALL	200	100-1,000
MEDIUM (default)	1,000	200-5,000
LARGE	4,000	1,000-10,000
EXTRA-LARGE	10,000	2,000-20,000
HUGE	20,000	5,000-50,000
EXTRA-HUGE	50,000	>25,000

Examples:

`OP DATAB SIZE=EXTRA-LARGE`	(opens a new dss file with a parameter size of extra large)
`OP DSSDATA SIZE=LARGE TABLE=STABLE`	(opens a new dss file with a stable type table and parameters set to large)

3.20 Pathname

Use:

```
PA, parameters
```

The pathname command changes or sets the (single) pathname in memory, which then may be used by other commands. This command is useful when just a part (or parts) of a pathname needs to be changed, without requiring the user to enter the entire pathname.

Parameters:

1. Pathname - A pathname following the command causes that pathname to be set in memory.

2. Tag - Enters the pathname corresponding to the tag given into memory.

3. N (where N is a number) - Enters the pathname corresponding to the catalog reference number "N" into memory.

4. Pathname Parts - This allows the user to change any of the six pathname parts by using the part letter followed by an equal sign and the new part. Several parts may be entered at the same time. To change a part to null (nothing), the equal sign for that part should be followed by a comma.

Examples:

`PA 14` would enter pathname 14 from the catalog file, which might be:

```
/RED RIVER/DENISON/STAGE/01JAN1955/1DAY/OBS/
```

`PA C=FLOW, F=DIVERSIONS` would change the pathname to:

```
/RED RIVER/DENSION/FLOW/01JAN1955/1DAY/DIVERSIONS/
```

`PA F=,` would change the pathname to:

```
/RED RIVER/DENSION/FLOW/01JAN1955/1DAY//
```

3.21 Read Data

Use:

```
RE.options, filename
```

The Read Data command reads data from a formatted ASCII file (whose name follows the command) that was previously created by the Write Data command in DSSUTL, and stores that data in the DSS file currently opened. This command is intended to be used when transferring DSS data from one computer to another. (See exporting and importing for reading data from another program.)

Options:

1. P (Preserve) - Records with the same pathname in the file will not be overwritten.

The following options apply only to time series data (and are infrequently used).

Regular-Interval Time series Data:

2. C (Compression) - Ignore any data compression settings in the read data file, and use the file compression methods set in the DSS file.

3. M (Missing) - Replace missing data flags only. Only -901's (or non-existent records) will be replaced in the DSS file.

4. U (Update) - Will not allow missing data flags (-901's) in the originating file to replace valid values.

5. R (Regardless) - Normally, if the data to be stored for one record is all missing (-901), that record is not written. The R option bypasses this, and will allow a record of all -901's to be stored.

Irregular-Interval Time series Data:

6. M (Merge) - If data for the same record already exists in the DSS file, the M option will merge the two sets together. For example, if data in the file exists for Noon, 2 and 4 p.m., and data in the read file is for 1, 3, and 5 p.m., the M option will cause the final data set to be for Noon, 1, 2, 3, 4, and 5 p.m.. Normally all data is replaced.

3.22 Record Tags

Use:

```
RT.option, parameters
```

The record tags command controls the tag names given to records. It can set or display the file tag scheme, re-tag all records within the file using the current scheme (or catalog reference number), or set specific record(s) to a given tag.

A record tag is a one to eight character semi-permanent record identifier, that is not necessarily unique. It must begin with a non-numeric character. It can be set by the user, or can be set according to a scheme base on the parts of the pathname. The default record tag is the letter "T" followed by the sequence number (number of records in the file). Tag(s) may be used in place of pathnames in several DSS programs. If more than one pathname has the same tag, only the first one found will be used.

Tag Schemes:

A file tag scheme generates tag(s) based on letters from the pathname. A typical tag might be the location name followed by a portion of the data type. For example, the observed flow at location NATP might have a tag of NATP-FO; the barometric pressure at FLD might be FLD-BP. Each character in the tag scheme is set by specifying the pathname part letter (A, B, C, D, E, or F) followed by the character position number in that part. This is followed by a comma, then another tag character specifier. When a character or symbol without a character position is used, that character is inserted into the tag. The tag "NATP-FO" given above was generated by the following scheme:

```
B1,B2,B3,B4,-,C1,F1
```

This generates a tag using the first through fourth characters of the B part, a dash, the first character of the C part, then the first character of the F part. If no character corresponds to the position given, that character is ignored.

It is also possible to use characters from the second word of a part by preceding the part letter with an underscore "_". The tag "FLD-BP", from a pathname with a C part of "BAROMETRIC PRESSURE", was created using the following scheme:

```
B1,B2,B3,-,C1,_C1
```

Note that the underscore causes the first character position of the second word to start counting at one. Pathname part words are delimited by any of the following characters " -@_+.;:". (If the above C part were "FLOW-NATURAL", the tag would be "FLD-FN".) If there were no second word for that part, that character would be ignored.

Option:

1. S (Set Scheme) - Sets the tag scheme to that designated in the parameters. If the parameter is "NONE", the current scheme is removed, and the default tag(s) are used ("T" followed by the sequence number).

Parameters:

1. ? - The file tag scheme is displayed.

2. Scheme (used with .S option) - Sets the file tag scheme. (This overwrites the scheme already set.)

3. NONE (the word "NONE", used with .S option) - Removes the current file tag scheme.

4. RETAG - Causes all records within the file to be re-tagged.

5. TAG=new-tag, pathname (or other reference) - A new tag for individual record(s) is set by specifying the parameter "TAG=" followed by the new tag, followed by the pathname or other record identifier(s) (such as reference number(s) or selective catalog pathname parts). Because tag(s) may be non-unique, several records may be set to have the same tag with this command.

6. SEQUENCE - A new beginning sequence number may be set by the parameter "SEQUENCE=" (or abbreviated to "SEQ=") followed by a integer number. When a new default tag is created (e.g., T14), this sequence number is used for the numeric portion of the tag.

7. REFERENCE - When the parameter is the word "REFERENCE" (which may be abbreviated to "REF"), all records are re-tagged according to their catalog reference number (a "T" followed by the number). This parameter requires that the catalog must already exist.

Examples:

```
RT  ?                                  (displays the current file tag scheme)
RT.S   B1,B2,B3,-,F1,C1,_C1            (sets the file tag scheme)
RT RETAG                               (Re-tags all records within the file with the
                                       current scheme or new sequence numbers.)
RT  TAG=SAC-FLOW  /SACRAMENTO/I ST/FLOW/01JAN1980/1HOUR/OBS/
RT  TAG=FOL-OUT  B=FOLSOM, C=FLOW-RES OUT
RT  TAG=SRC-TEMP  33-38, 44, 46
RT  TAG=SRC-PRE   T241
RT  REFERENCE                          (Re-tag all records according to their catalog
                                       reference number)
RT SEQUENCE=100                        (Set the current sequence number to 100)
```

3.23 Rename

Use:

```
RN, parameters
```

The rename command renames record pathnames. One or more of the pathname parts may be changed, or the entire pathname may be changed. The rename command does not alter the record tag.

After giving the rename command and the associated parameters, DSSUTL will display a list of the pathnames to be changed, then prompt the user for new pathname part(s) or a new pathname. To change a part, enter the part letter followed by an equal sign and the new part. Alternatively, a entire new pathname may be entered if only one record is being renamed. If the user decides not to rename any pathnames, the word "QUIT" may be entered to return to the main program.

Parameters:

1. None - Uses the pathname set in memory.

2. Pathname - If a pathname follows the command, that record is renamed.

4. Tag(s) - Renames the record(s) corresponding to the tag(s) given.

5. N1, N2, N3-N4 - Obtains pathnames with reference numbers N1, N2, and N3 through N4 from the current catalog file to rename.

6. Selective Catalog (A=..., B=...) - Uses the selective catalog capabilities for obtaining pathnames for records to rename. For more information, refer to the selective catalog section.

7. ALL - All pathnames in the catalog will be renamed.

Example:

To rename all records with a B part of "`STH BEND`" to be "`SOUTH BEND`" the following command is given:

```
RN B=STH BEND
```

The pathnames to be renamed are listed, then the prompt "Enter New Pathname Part(s)" is given. In response, the user types:

```
B=SOUTH BEND
```

More than just one part may be renamed at the same time, if desired. For example:

```
RN B=STH BEND,   C=FLOW
B=SOUTH BEND,   F=COMPUTED
```

3.24 Squeeze

Use:

```
SQ.options, parameters
```

The squeeze command removes inactive space that may have accumulated from actions such as deleting records in the DSS file. (The percentage of inactive space is displayed when a DSS file is closed.) The squeeze command makes the file physically smaller by copying all valid data to a new file, then renaming that file to the old file name. The user should have sufficient disk space for DSSUTL to make a temporary copy of the file.

Unless the "A" option is used, the squeeze command will "tune" the file by adjusting the hash table size. This can be overridden by specifying a size parameter (see below). If the expected number of records of the file is very different from the current number of records, specify the size (or use the "A" option).

Options:

1. A (As is) - This option retains all of the current file parameters (hash table type and size).

2. C (Compression) - Regular-interval time series data is re-compressed according to the compression methods set in the file. (Refer to the DC command.)

3. T (Tags) - The records are re-tagged according to the scheme set by the RT command.

Parameters:

1. Table Type - A different hash (address) table type will be used if the parameter "TABLE=type" is given. The type may be either "DYNAMIC"or "STABLE". A stable addressing table is primarily intended for data bases that do not change in size frequently (e.g., a master data base file). A "dynamic" table is intended where the file size may vary considerably (e.g., a data base file used for computations), or where the intended file size is not known.

2. Size - The hash table size can be controlled to optimize storing and retrieval of data according to the expected number of records in the file. The size parameter is specified by "SIZE=", then the expected number of records, or one of following sizes:

Size Name	Target Number	Target Range
TINY	20	1-50
EXTRA-SMALL	50	1-200
SMALL	200	100-1,000
MEDIUM (default)	1,000	200-5,000
LARGE	4,000	1,000-10,000
EXTRA-LARGE	10,000	2,000-20,000
HUGE	20,000	5,000-50,000
EXTRA-HUGE	50,000	>25,000

Examples:

```
SQ.A                          (no adjustments are made to the file)
SQ.T SIZE=2000
SQ TABLE=STABLE
SQ SIZE=EXTRA-LARGE
```

3.25 Status

Use:

```
ST, parameters
```

If no parameters are given, the status command displays the current memory settings, including the name of the DSS file(s) opened, the pathname in memory and the time window set. If command(s) are entered as parameters, the settings for these commands are displayed. For example, if "ST TIME" is entered, the current time window set will be displayed.

Parameters:

1. None - The status of the current memory settings are displayed.
2. AR - The name of the DSS archive file is displayed.
3. CA - A message is printed indicating whether a catalog exists for the DSS file opened.
4. CO - The name of the DSS copy file is displayed.
5. DC - The file's default data compression methods are displayed.
6. DE - A list of the record pathnames just deleted (and available to undelete) is displayed. (The command UD with no parameters will undelete these records.)
7. EF - Displays the exchange format.
8. EV - Displays a list of the exchange variables, and what pathnames they are set to.
9. FO - The format currently set is displayed.
10. OP - The name of the DSS file opened is displayed.
11. PA - The pathname set in memory is displayed.
12. RT - The file's record tag scheme is displayed.
13. TI - The current time window is displayed.
14. UD - A list of all the record pathnames in the file that are available to undelete is displayed.
15. WR - The name of the file last used for writing data to is displayed.

3.26 Tabulate

Use:

```
TA.options, parameters
```

The tabulate command displays data (one screen at a time) from the specified records. Unless an output format has been previously set using the FORMAT command, DSSUTL will select an appropriate format based on the magnitude of the data.

For regular-interval time series data, the data is displayed in columns of 6 or less, along with the date and time corresponding to the data in the first column. A single column of data, which will have a date and time for each data value, may be produced by setting the FORMAT with one column (e.g.,"`FO (1F12.4)`"). Irregular-interval time series data is always displayed in a single column form.

Options:

1. A (All) - Tabulates all data to the screen without paging. (This option is automatically invoked if the tabulation is written to a file.)

2. F (File) - The data is written to a file instead of to the screen. The name of the file to write to is asked for.

3. P (Printer) - The tabulated data is sent to the printer instead of the screen. The printer should be on-line before the command is given.

Options for Time series Data:

4. N (No Quality Flags) - If the record has data quality flags associated with it, this will cause the flags not to be displayed (otherwise they will be printed on a separate line in a hex format).

5. C (Complete) - All data within the record, including leading and trailing missing data (-901's), are tabulated. This option ignores any time window set.

6. D (Data only) - If a time window has been set, then this option causes missing data at the beginning and ending of the window not to be tabulated. If no time window is set, this option is automatically used.

Parameters:

1. None - Uses the pathname set in memory. For time series data, if a time window has been set, the data within that time window is tabulated.

2. Pathname - If a pathname follows the command, that record is tabulated.

3. Tag(s) - Tabulates the record(s) corresponding to the tag(s) given.

4. N1, N2, N3-N4 - Obtains pathnames with reference numbers N1, N2, and N3 through N4 from the current catalog file for tabulating. Any time window set will be adhered to if the data is time series.

5. Date Reference (D=...) - Uses the current catalog and selects records to tabulate based upon a "D" part relative to the current date. The "D=" must be followed either by a D (for the current day), a M (for the current month), or a Y (for the current year), a minus (or plus) sign, a number, then another D, M, or Y. For example:
   `D=M-2M` (The current month minus two months)
   `D=D-60D` (The current day minus 60 days)
   The time interval (E part) is typically specified along with this parameter.

6. Selective Catalog (A=..., B=...) - Uses the selective catalog capabilities for obtaining pathnames for records to tabulate. For more information, refer to the selective catalog section.

7. ALL - Tabulates data for all pathnames in the catalog file.

Examples:

`TA 2`	(Tabulate the record with pathname reference number 2)
`TA.P C=FLOW`	(Print a tabulation of all records which have a C part of "`FLOW`" in the catalog)
`TA PROG=HEC1`	(Tabulates all data that was written by program HEC1.)

3.27 Time

Use:

```
TI, starting date, time, ending date, time
```

The time command sets the time window used with time series data. If no time window is set, data for the entire record is used. To set a time window, follow the time command by the starting date and time then the ending date and time. The time or date may be in either order, as long as the starting date and time precede the ending date and time.

A time must be a four digit number, given in 24 hour clock time. A date can be one of several styles, but must not contain any spaces within it. (A 7 or 9 character military style date is typically used.)

A time window can be set relative to the system time by the single character "T", optionally followed by a minus (or plus) sign, a number, then a "H" for hours, or a "D" for days, or a "Y" for years. In addition, a fixed hour for the current day may be specified by using "T" as the date portion, then specifying the time in the next field. See the examples below.

A time offset may be specified from what the time window is currently set to by giving just a plus or minus sign followed by a number then a "H" for hours, or a "D" for days, or a "Y" for years. The ending date/time may be changed without affecting the beginning date/time by leaving empty fields (identified by commas) for the beginning of the time window.

To clear the time window, just enter the time command (with no times).

Examples:

`TI 01MAR72, 2400, 14JAN73, 1200`	
`TI 2400, 01MAR72, 1200, 14JAN73`	
`TI T-4H, T`	(current date/time - 4 hours, current date/time)
`TI T, 0200, T, 1600`	(today at 2 a.m., today at 4 p.m.)
`TI T-5Y, T-40D`	(today - 5 years, today - 40 days)
`TI -2D +8H`	(subtract 2 days from the starting date/time, add 8 hours to the ending date/time)
`TI ,,, +12H`	(add 12 hours to the ending date/time)
`TI ,,, 20SEP72`	(Change the ending date to 20 September 1972)
`TI 1200`	(Change the starting time to 1200)
`TI`	(Clear the time window)

3.28 Undelete

Use:

```
UD, parameters
```

The undelete command restores records that have been previously deleted, but not yet physically removed by the squeeze command. If some records were deleted in the previous command, they may be undeleted just by entering the undelete command with no parameters (DSSUTL retains a list of the pathnames from the last delete command).

Any or all records may be undeleted until the file is squeezed. Once the file has been squeezed, all deleted records are physically removed and cannot be recovered. If the catalog has not been updated since the desired records have been deleted, it may be used to specify the records to undelete.

Parameters:

1. None - If records have just been deleted, DSSUTL will use an internal list of pathnames from the last delete command to undelete. If not, the pathname set in memory will be undeleted.

2. ? - A list of the record pathnames that are available to undelete is displayed.

3. Pathname - If a pathname follows the command, that record is undeleted.

4. Tag(s) - Undeletes the record(s) corresponding to the tag(s) given.

5. N1, N2, N3-N4 - Obtains pathnames with reference numbers N1, N2, and N3 through N4 from the current catalog file to undelete (assuming that the catalog has not been updated since those records were deleted).

6. Selective Catalog (A=..., B=...) - Uses the selective catalog capabilities for obtaining pathnames for records to undelete (assuming that the catalog has not been updated since those records were deleted). For more information, refer to the selective catalog section.

7. ALL - Undeletes all records in the DSS file that have been deleted since the last squeeze. This parameter does not use the catalog.

Examples:

`DE F=COMPUTED`	
`UD`	(this will undelete those records that were deleted in the previous command)
`UD D=01JAN1972`	(this will use the catalog file to undelete records with a D part of `01JAN1972`, assuming that it has not been updated since the delete)
`UD ALL`	(undeletes all records in the DSS file)

3.29 Write Data

Use:

```
WR.options, TO=outfile, parameters
```

The Write Data command writes data to an ASCII file in a formatted form. This file can then be transmitted to another computer, where it may be read into a DSS file using the Read Data command. The command can also be used to aid in creating a data input file for a program that does not have DSS capabilities.

The name of the file to write the data to (outfile) may be specified by "TO=" followed by the file name. It only needs to be specified once in a session. If no file name is given, the file name will be asked for.

Unless an output format has been previously set using the FORMAT command, DSSUTL will select an appropriate format based on the magnitude of the data. If the data has many significant digits, it is wise to specify a format, as the data will be truncated to the number of significant digits shown in the file when read back in.

Options for Time series Data:

1. T (Time) - This option will cause a date and time identifier to be written at the beginning of each line, similar to the tabulate command. The date and time correspond to the data value in the first column. These are for informative purposes only as the date and time in the header are the only values that are actually used.

2. C (Complete) - All data within the record, including leading and trailing missing data (-901's), are written. This option ignores any time window set.

3. D (Data only) - If a time window has been set, then this option causes missing data at the beginning and ending of the window not to be written. If no time window was set, this option is automatically used.

4. N (No Quality Flags) - If the record has data quality flags associated with it, this will cause the flags not to be printed (otherwise they will be printed on a separate line in a hex format).

Parameters:

1. None - Uses the pathname set in memory. For time series data, if a time window has been set, the data within that time window is written.

2. Pathname - If a pathname follows the command, that record is written.

3. Tag(s) - Uses the record(s) corresponding to the tag(s) given.

4. N1, N2, N3-N4 - Obtains pathnames with reference numbers N1, N2, and N3 through N4 from the current catalog file to use. Any time window set will be adhered to if the data is time series.

5. Date Reference (D=...) - Uses the current catalog and selects records to write based upon a "D" part relative to the current date. The "D=" must be followed either by a D (for the current day), a M (for the current month), or a Y (for the current year), a minus (or plus) sign, a number, then another D, M, or Y. For example:

   `D=M-2M` (The current month minus two months)

   `D=D-60D` (The current day minus 60 days)

   The time interval (E part) is typically specified along with this parameter.

6. Selective Catalog (A=..., B=...) - Uses the selective catalog capabilities for obtaining pathnames for records to write. For more information, refer to the selective catalog section.

7. ALL - Writes data for all pathnames in the catalog file.

Examples:

`WR TO=DATFIL 3`	(Writes data from the pathname with reference number 3 to the file "DATFIL")
`WR.T C=FLOW`	(Writes data for all records whose pathname in the catalog file has a C part of "FLOW". The data is appended to DATFIL. A date and time is placed at the beginning of each line.)
`WR FLD-FLOW`	(Writes the record with the tag "FLD-FLOW".)

Chapter 4
Secondary Commands

4.1 Close

Use:

```
CL, parameters
```

The close command closes files that have been opened. If no parameters are specified, all files are closed. Specific files may be closed by identifying the files to close.

Parameters:

1. None - All files that have been opened are closed.

2. MAIN - If one of the parameters is "MAIN", the main DSS file is closed.

3. COPY - If one of the parameters is "COPY", the DSS copy file (or secondary file) is closed.

4. TAB - If one of the parameters is "TAB", the tabulate file (used with the "F" option in tabulate) is closed.

5. WR - If one of the parameters is "WR", the write data file is closed.

6. DE - If one of the parameters is "DE", the file containing the list of pathnames of records last deleted is closed.

4.2 Debug

Use:

```
DB, parameters
```

The debug command sets the DSS message level.

Parameters:

1. Number - If a number between 1 and 15 is given, the message level is set to that number. Values between 5 and 9 are used to debug higher level subroutines, while values over 9 debug low level subroutines.

2. RESET - If the parameter "RESET" is given, the message level is set back to its default value.

4.3 Inquire

Use:

```
IN, parameter
```

The inquire command calls the DSS ZINQIR subroutine with the parameter specified, then prints out the character string returned, and the numeric value returned.

4.4 No Operation

Use:

```
NOP  (or "*")
```

The no operation command is ignored by DSSUTL. This command is usually used for comment lines in an input file. An asterisk (*) may be used in place of NOP.

4.5 Set

Use:

```
SET, parameter1, parameter2
```

The set command calls the DSS ZSET subroutine to set various items. Parameter1 must be a character string, and parameter2 must be an integer number. Both parameters must always be set. If the item to be set only requires one parameter, use a dummy value for the other parameter.

4.6 Verify

Use:

```
VE ON/OFF
```

The verify command turns on or off the informative output displayed by DSSUTL. The default is on.

Informative output may be turned off for a single command by preceding that command with a period ".". For example:

```
U>.CO TO=DATFIL2 C=FLOW
```

is the same as

```
U>VE OFF
U>CO TO=DATFIL2 C=FLOW
U>VE ON
```

Chapter 5
Selective Catalog Capabilities

5.1 Introduction

The catalog routines provide a means of obtaining a sorted inventory of the record pathnames in a DSS file and pertinent information about those records (e.g., date last written). The catalog also provides an option of selecting and using pathnames based on their pathname parts. For example, a user can rename all observed flow data by the command:

```
RN C=FLOW, F=OBS
```

5.2 Sort Order

The user may define a sort sequence for the catalog based on the six pathname parts (A, B, C, D, E, or F). If the sort order is not defined, the default sequence of ABCFED is used. In this case all of the A parts are sorted alphabetically first. Then for each set of pathnames with identical A parts, the B parts are sorted, and similarly for the C, F, E and D parts.

The sort order may be defined as a parameter in the catalog command by entering the letter "O" followed by an equal sign and the sort order. Those parts not specified are filled in with the default order. For example:

```
CA O=FB
CA O=FBACED
```

define the same sort sequence.

5.3 Selection of Pathnames Based on Last Written Date or Program

Pathnames may be selected from the catalog based on the date they were last written to, or the name of the program that wrote the record, when using a full (non-abbreviated) catalog. Records may be selected based on their last write date before, after, or on a given date. The two character identifier "LW" (for Last Write), or "PR" (for program) are used to identify the date or program name to select on. This identifier is followed by an equal (or < or > for last written date), then the appropriate date or program name. This identifier may be used wherever, or in addition to, selective pathname parts are given. A summary of the syntax is:

`LW<date`	Selects all pathnames last written before "date"
`LW=date`	Selects all pathnames last written on "date"
`LW>date`	Selects all pathnames last written after "date"
`PR=name`	Selects all pathnames last written by the program "name"

The date should be a 7 character military style date (e.g., 04JUL82), and the program name must be exactly as shown in the catalog file. The "LW=date" cannot be used in combination with the "LW<date" or "LW>date" identifiers. Some examples of this capability are:

```
AR  TO=backup.dss LW<20JAN92
AR  B=SOUTH BEND LW<25JUL92
DE  PR=MYPROG
CO  TO=computed.dss LW=16SEP92
CO  F=PLAN 2  LW>16SEP92
```

5.4 Selection of Pathnames Based on Pathname Parts

The user may select which records to operate on (or catalog) based on pathname parts. The routines compare the requested parts with the actual parts of each pathname to decide which records to operate on. The user may select records based on the comparison of one or several of the pathname parts. This is done by entering the part identifier (A, B, C, D, E, or F) followed by an equal sign and the part specification. The eight types of part specifications that can be used are:

1. `NAME` - NAME must be identical to the part.

2. `NAME@` - The part must begin with NAME.

3. `@NAME` - The part must end with NAME.

4. `@NAME@` - The part must have the character segment NAME within it.

5. `#NAME` - The part must not be NAME.

6. `#NAME@` - The part must not begin with NAME.

7. `#@NAME` - The part must not end with NAME.

8. `#@NAME@` - The part must not have the character segment NAME within it.

Note that the at sign (@) and the pound sign (#) are special characters. The "@" indicates that any characters prior (or subsequent) to it are not compared. The "#" at the beginning of the part negates the specification. For example:

B=SOUTH MARINA, C=@FLOW@, D=@197@, E=1DAY, F=#PLAN 2

This line will obtain all pathnames in catalog with the following characteristics: the data is some type of daily flow for the 1970's at South Marina, except for data under plan 2.

Chapter 6
Export - Import

6.1 Introduction

A capability of DSSUTL is the exportation or importation of time series data for exchange between DSS and spreadsheet and data base programs, such as Lotus 123 and dBase. This is accomplished by defining data sets to be exchanged and an exchange format. When exporting data from DSS, the data is written to an ASCII (text) file with the defined format. The user then exits DSSUTL and executes the PC program and imports that ASCII file. Importing data to DSS essentially follows the reverse procedure. This capability can also serve as a means for a "user-defined" tabulation format.

Exporting or importing data with DSSUTL requires the use of several commands to preform one function. The "exchange format" command is inherently more complex than other commands. Because of this, users typically write these commands in an input file (refer to the introduction at the beginning of this document), or in the PREAD macro file "utlmac". The primary commands used for exporting and importing data are:

Exchange Format: This identifies the format (e.g., the third field contains the time of the data, the fourth field contain the flow, etc.) of the import or export file.

Exchange Variable: This provides a short abbreviation of the DSS pathname of the data to export or import for the exchange format (instead of the full pathname).

Export: This command tells DSSUTL to write data to a given file.

Import: This tells DSSUTL to read data from a given file.

6.2 Procedure

For each data set (DSS pathname record), an "exchange variable" must be provided to identify the DSS data in the exchange format. This essentially provides an abbreviated means of specifying a DSS pathname. The "EV" (exchange variable) command specifies a user defined name that is from one to eight characters long, and a pathname reference. For example, such an exchange variable might be:

```
STG1=/ALLEGHENY/NATP/STAGE/01JAN1991/1HOUR/OBS/
```

Up to 50 exchange variables may be specified for importing or exporting 50 data sets (or values per line).

An "exchange format" must be provided using the "EF" command. The exchange format tells DSSUTL how the data is to be written out or read in. Exchange variables specified in the format are enclosed in square brackets (for example, the above variable would be given as "[STG1]"). The date and time of the data may be designated by the reserved variables `[DATE]`

and [TIME]. When exporting data, the A, B, C, D, E, or F parts of the pathname can also be exported by the reserved variable [APART], [BPART], etc.. (Only one A and B part may be read from an import file.) An example of an exchange format for exporting data might be:

```
COE [DATE], [TIME]:  [BPART] [STG1] [PRECIP]; [BPART] [STG2] [TEMP]
```

In the above format, the date, time and B part correspond to the data value following (e.g., the first `[BPART]` corresponds to `[STG1]`, while the second corresponds to `[STG2]`).

Characters that are not defined variables within square brackets are copied just as they are to the export file. In this example, "COE", commas, spaces, the colon, etc. are copied to the export file.

In addition, the style of the date and time can be specified, as well as the format of the data. Tabs may be defined to position data or other items to specific columns in the export file. A single header line (which can contain items like the A, B, C parts) to be written at the top of the export file can be designated by the EF.H command option.

The format to identify data to be imported uses the same EF format. Imported data is identified by fields (data or characters separated by blanks and/or a comma), and exchange variable. Data exported using the above format may be imported by the following format:

```
[SKIP] [DATE] [TIME] [SKIP] [STG1] [PRECIP] [SKIP] [STG2] [TEMP]
```

The "[SKIP]" identifier tells DSSUTL to skip that field. Since data is read in by fields, the comma, colon and semi-colon are not given. Refer to the exchange format documentation for specifics on this command.

Once the exchange format and exchange variables have been entered, data may be exported with the EXPORT command or imported with the IMPORT command. A time window must be used when exporting data, and may be used when importing (if the import file does not contain the date and time of the data). The file name to export to, or import from, follows the command name.

6.3 Examples

The commands used in exporting and importing data are more complex than other DSSUTL commands. Because of this, it is wise to enter the commands in an input file to DSSUTL, or in the PREAD macro file "UTLMAC". The PREAD macro file can contain many sets of commands that are identified by a macro name. DSSUTL will automatically connect to the macro file in the same directory. To execute those commands, the user enters "!RUN macro-name" at the DSSUTL prompt (where macro-name is the name of the macro to execute). The macro capability is ideal to set up several of the export - import commands. For example, the user might run a macro to set all the exchange variables and the exchange format. Then the user can manually enter a time window and the export command. Thus, the same macro could be used on a daily basis without any modification.

Example 1. Export irregular interval time series data

DSSUTL commands:

```
EV SWS=/SACRAMENTO/SPO/WIND SPEED/07MAY1991/IR-DAY/OBS/
EV SWD=C=WIND DIRECTION
EV SBP=C=BAR PRESSURE
EV SRH=C=REL HUMIDITY
EV ST=C=TEMPERATURE
EF.H  [T:3][DATE:Mmm D, YY]  [T:16][CPART:ST] [T:30][CPART:SRH] //
 [T:45][CPART:SBP] [T:60][CPART:SWS] [T:75][CPART:SWD]
EF  [T:8][TIME:H:MM am/pm]  [T:20][ST:I3] [T:34][SRH:I3] //
 [T:45][SBP:10.2] [T:65][SWS:I3] [T:80][SWD:I3]
TIME 09MAY91 1130 09MAY91 1230
EXP EXP.1
```

Export file (Exp. 1):

May 9, 91	TEMPERATURE	REL HUMIDITY	BAR PRESSURE	WIND SPEED	WIND DIRECTION
11:31 am	-	36	-	-	-
11:33 am	-	-	-	8	2
11:35 am	65	-	-	-	-
11:38 am	-	-	-	7	278
11:39 am	-	33	-	-	-
11:43 am	66	-	-	-	-
11:45 am	-	-	-	6	250
11:46 am	-	33	-	-	-
11:50 am	67	32	-	-	-

Example 2. Export regular interval time series data

DSSUTL commands:

```
EV ELEV=/AMERICAN/FOLSOM/ELEV/01JAN1991/1DAY//
EV IN=C=FLOW-RES IN
EV OUT=C=FLOW-RES OUT
EV STOR=C=STOR-RES EOP
EF [T:3][DATE:Mmm D, YY]    [CPART]: [STOR:I7]    [CPART]: [ELEV:9.3] //
 [CPART]: [IN:I4]     [CPART]: [OUT:I4]
TIME 01APR91 2400 30APR91 2400
EXP EXP.2
```

Export file (Exp. 2):

Apr	1, 91	STOR-RES EOP:	427903	ELEV:	404.840	FLOW-RES IN:	3654	FLOW-RES OUT:	428
Apr	2, 91	STOR-RES EOP:	435059	ELEV:	405.810	FLOW-RES IN:	3954	FLOW-RES OUT:	318
Apr	3, 91	STOR-RES EOP:	440945	ELEV:	406.600	FLOW-RES IN:	3365	FLOW-RES OUT:	361
Apr	4, 91	STOR-RES EOP:	446802	ELEV:	407.380	FLOW-RES IN:	3379	FLOW-RES OUT:	382
Apr	5, 91	STOR-RES EOP:	453619	ELEV:	408.280	FLOW-RES IN:	3907	FLOW-RES OUT:	431
Apr	6, 91	STOR-RES EOP:	461425	ELEV:	409.300	FLOW-RES IN:	4652	FLOW-RES OUT:	682
Apr	7, 91	STOR-RES EOP:	471649	ELEV:	410.620	FLOW-RES IN:	5557	FLOW-RES OUT:	382

Example 3. Import regular interval time series data

DSSUTL commands:

```
EV FLOW=/ALLEGHENY//FLOW//1HOUR/OBS/ UNITS=CFS TYPE=INST-VAL
EV STAGE=C=STAGE UNITS=FEET TYPE=INST-VAL
EV PRECIP=C=PRECIP-INC UNITS=INCHES TYPE=PER-CUM
EF [SKIP] [BPART]  [DATE]  [TIME]  [FLOW]  [STAGE]  [PRECIP]
IMP FILE.IMP
```

Import file (File.imp):

```
COE "NATP"  08MAY91  1700  23177.00  12.67  0.00
COE "NATP"  08MAY91  1800  21371.00  12.49  0.00
COE "NATP"  08MAY91  1900  20678.00  12.42  0.00
COE "NATP"  08MAY91  2000  20876.00  12.44  0.00
COE "NATP"  08MAY91  2100  21074.00  12.46  0.00
```

DSPLAY

Hydrologic Engineering Center
Data Storage System Graphics Utility

User's Manual

Version 2.0
March 1995

Hydrologic Engineering Center
U.S. Army Corps of Engineers
609 Second Street
Davis, California 95616-4687
(916) 756-1104

DSPLAY

Utility Program for Graphical Display of HEC-DSS Data

Table of Contents

Chapter **Page**

1. Introduction 1

2. General Program Structure 3

3. Program Execution 5

4. UNIX and MS-DOS Versions 7

5. HEC-DSS Data Retrieval 9

6. DSPLAY Example Application 11

Appendices

Appendix A - DSPLAY Commands A-1
Appendix B - ACAD Command Document B-1

Chapter 1

Introduction

The **DSPLAY** program is a utility whose main function is to provide a quick and simple means of graphically displaying data stored in an HEC-DSS (HEC Data Storage System) file. A rapid interactive graphical display of data is essential for practical use of a data storage system. This manual only deals with the DSPLAY program, therefore it is advised that for users who are not familiar with HEC-DSS, that they first read the HEC-DSS Overview found at the beginning of this document.

The following material describes the current structure and capabilities of the **DSPLAY** program for the UNIX CD-4330 Workstation computer and MS-DOS Microcomputer . The capabilities available on the UNIX version are essentially the same as are available on the MS-DOS Version. The differences or additional capabilities that exist between the two versions are documented in this manual.

(This page intentionally left blank)

Chapter 2

General Program Structure

Two routines make up the DSPLAY program. They are the Data Retrieval Routine and the Data Display Routine.

The Data Retrieval Routine performs two main functions. It provides the interface between the user and the HEC-DSS file and its data. The user defines the data (pathname) that is to be retrieved in this routine for later display. It also provides the user the capability of entering free-format commands interactively that direct the execution of the program. A detailed description of the commands are given in Appendix A.

The Data Display Routine is used to display the data previously defined in the Data Retrieval Routine. Two modes of display are available, a graphical display (**PLOT** command) and a tabular display (**TAB** command).

Up to eight curves and six different y-scales may be displayed at one time. Time-series and paired function data can be displayed by this program, with the limitation that they not be plotted together. The UNIX version allows a total of twenty curves to be used. For time-series data, time is plotted on the abscissa (x-axis) in intervals as small as one minute. For plotted data the time scale is automatically adjusted to provide a time interval that is suitable for the time period presented. Paired function data is displayed with standard x and y labels.

The tabular display function can be used on any alpha-numeric terminal. For large amounts of data, an option has been provided for writing the tabular display to a disk file which can later be sent to a printer.

In addition to the graphical display, the user has the option, once the data has been plotted, to graphically edit the displayed data and save the edited data under a new record or replace the original data record.

A simplified automated computer aided drafting (ACAD) capability is available under the data display module that allows the user to superimpose acad style graphics on the standard DSPLAY plots. For a detailed description of the ACAD capability, refer to Appendix B. At the present time this capability is only available on the UNIX version.

The DSPLAY program operates under the PREAD user interface, permitting the use of macros, selection screens and other execution aids. The PREAD user interface is described in its own document.

(This page intentionally left blank)

Chapter 3

Program Execution

The program is executed on the UNIX computer by entering its name "**dsplay**" in lower case letters. The MS-DOS version is executed in a similar fashion with the addition that device drivers must be loaded first. The loading of the graphics device drivers and execution of the DSPLAY program are done by a batch program "**DSP.BAT**". The DSS filename may be selected by either typing it in or using the arrow cursor keys to move to one of the names in the list that is displayed on the screen, then pressing the carriage return. On the UNIX machine the procedure is the same, except that the space bar key is used to move through the list of names. After the DSS filename has been entered, the program produces the prompt **D>**, where user commands are entered. To exit the program the **FI**NISH command is entered.

The following is a list of the options and parameters that can be entered on the program execution line.

DSPLAY - [option keyword]	MS-DOS
dsplay - [option keyword]	UNIX

Options

A - A few program prompts, such as the program version number and date, are suppressed.

B - All program prompts are suppressed but error messages are output.

C - All program prompts and warning error messages are suppressed. Only critical error messages are output.

DSPLAY [parameters]

Parameters

? - Will produce a listing of parameters that can be entered on the execution line. See Table 1 for a list of the valid parameters.

Examples

```
dsplay
dsplay  dssfile=mas90db.dss
dsplay -c dssfile=mas90db  acad=acfile  term=plotout
dsplay  dss=mas90db  ac=acfile  te=plotout  input=infile  output=out.txt
DSPLAY -C  DSSFILE=C:\DATA\MAS90DB  INPUT=INFILE
```

The first example is an interactive execution in which the program will prompt for the name of the DSS file to use. The next three examples are specific to the UNIX version and the fourth to the DOS version. The second example executes the program and opens the DSS file "mas90db." The third example executes the program, suppresses all program messages (C option), opens the DSS file "mas90db", ACAD file "acfile", and graphics output file "plotout." The fourth example executes the program, opens the DSS file "mas90db", ACAD file "acfile", input file "infile", output file "out.txt" and the graphics output is directed to file "plotout". The fifth example executes the program, suppresses all program messages (C option), opens the DSS file "MAS90DB", and input file "INFILE." The first, second and third examples are interactive executions where the user supplies additional interactive commands to generate plots. The fourth and fifth examples are batch executions in which all commands are read from the input file "INFILE".

Table 1
Keywords for DSPLAY Execution

DSPLAY \?

(UNIX version)

```
DSPLAY: Version Date Dec  1,1992

   UNIT     KEYWORD     *ABREV    **MAX   DEFAULT                          DESCRIPTION
    NOP     INPUT       I          83     /dev/tty                         Standard input
    NOP     OUTPUT      O          83     /dev/tty                         Standard output
    NOP     DSSFILE     D          83                                      DSS file input
    NOP     UDSFILE     U          20     dspdld                           User defined pre/post file
      4     TERM        T          83     /dev/tty                         Graphics output pdn or file
    NOP     SCNFILE     S          83     dspscn                           PREAD Screen file
      7     TABFILE     TA         83     tabfile.out                      Output file for TAB.F command
     30     SCRAT       SCR        83     SCRATCH.008                      Scratch file
    NOP     LOGFILE     L          83     SCRATCH.001                      PREAD log file
     51     SPLOT       SP         83     SCRATCH.010                      Scratch file
    NOP     FUNFILE     F          83     genfun                           PREAD function file
    NOP     MACFILE     M          83     dspmac                           PREAD macro file
    NOP     HELPFILE    H          83     /usr/hec/sup/dsplay.hlp          Help file
     52     ACAD        A          83     acadfile                         ACAD overlay plot file

   * ABREV - SHORTEST ABBREVIATION ALLOWED FOR KEYWORD
   ** MAX - MAXIMUM # OF CHARACTERS FOR FILENAME (OR STRING)
DSPLAY ?
```

(MS-DOS version)

```
DSPLAY: MS-DOS Version 2.0.5 Date Dec  1,1992

   UNIT  KEYWORD    *ABREV  **MAX   DEFAULT                                DESCRIPTION
     5   INPUT      I        30     CON                                    Standard input
     6   OUTPUT     O        30     CON                                    Standard output
   NOP   DSSFILE    D        30                                            DSS file input
   NOP   SCNFILE    S        30     DSPSCN                                 PREAD screen file
     9   TAPE9      T        30     SCRATCH.009                            Scratch file
     7   TABFILE    TAB      30     TABFILE.OUT                            Output file for TAB.F command
    30   SCRAT      SCR      30     SCRATCH.008                            Scratch file
   NOP   LOGFILE    L        30     SCRATCH.001                            PREAD logfile
    39   SCRAT2     SCRAT2   30     SCRATCH.006                            Scratch file
   NOP   FUNFILE    F        30     GENFUN                                 PREAD function file
   NOP   MACFILE    M        30     DSPMAC                                 PREAD macro file
   NOP   HELPFILE   H        30     DSPLAY.HLP                             Help file

   * ABREV - SHORTEST ABBREVIATION ALLOWED FOR KEYWORD
   ** MAX - MAXIMUM # OF CHARACTERS FOR FILENAME (OR STRING)
```

Chapter 4

UNIX and MS-DOS Versions

As mentioned previously, two versions of DSPLAY exist. The main difference between the two versions is the graphics software and the hardware that the programs use. The differences in hardware and software have led to each version having some different capabilities. The following is a list of the main differences between the two versions.

A. Hardware and Software - The UNIX version uses Tektronix, X Window, PostScript and Hewlett Packard software for plotting and is therefore limited to PostScript, HP, X Window or Tektronix hardware or software emulations of this hardware. The following is a list of the hardware that can be used by the UNIX version:

Tektronix 4010 series CRTs	Tektronix 4100 series CRTs
Tektronix 4200 series CRTs	TAB 132/15-G CRT
HP 7475 Pen Plotter	PostScript
X Window workstation	

The MS-DOS Version uses Graphics Software Systems, (GSS) device drivers and software. The number of devices that the MS-DOS Version supports is extensive. A detailed description of the graphics environment settings and a list of the supported hardware devices are provided in separate installation instructions for device drivers.

B. Data Display Limits - A major difference between the UNIX and MS-DOS Versions is the amount of data that can be plotted. The UNIX version can plot twenty curves per plot and the DOS version can plot a maximum of eight curves per plot. The UNIX version utilizes virtual memory and is capable of plotting fourteen times as much data as the MS-DOS Version. The UNIX version can plot up to 75,000 data points for a single curve or any combination in which the total number of data points for all the curves being plotted does not exceed 75,000. The MS-DOS Version can plot up to 3,050 data points per curve as long as the total number of data points for all the curves being plotted does not exceed 5,300. The following is an example of the maximum number of time series data values that can be plotted at one time.

MS-DOS	**UNIX**
1 curve	1 curve
4 months of hourly data	100 months of hourly data
8 years of daily data	200 years of daily data
8 curves	20 curves
1 month of hourly data	5 months of hourly data
1 year of daily data	10 years of daily data

C. DSS Files - The UNIX computer version can have five DSS files open concurrently but the MS-DOS Version uses a virtual scheme that allows it to have a different DSS file assigned to each pathname being plotted.

Chapter 5

HEC-DSS Data Retrieval

There are several methods for retrieving time-series data from an HEC-DSS file. One method is to provide the exact pathname for the block of data that you wish to display. This is done by using the 'PATHNAME' command and entering the pathname. For example, the following command would be used to obtain the entire record for St. Paul.

```
PAT /SCIOTO/STPAUL/FLOW-RES.IN/01JAN1959/1DAY/TEST/
```

As an alternative to entering the full pathname a catalog number or tag may be used. Another method for retrieving data is to enter a time window command and then the pathname command. The program will retrieve data that falls within the time window specified. The time window is entered by the use of the 'TIME' command, which provides the starting and ending military time and dates. Once the 'TIME' command has been entered, it applies to all pathnames that follow. An example of the second method follows:

```
TIME 1200,01MAR1981,1200,01JUL1981
PAT /SCIOTO/STPAUL/FLOW-RES.IN/01JAN1959/1DAY/TEST/
```

With these methods, the user can obtain time-series data from HEC-DSS and display up to eight (DOS) or twenty (UNIX) curves. The only limitation is that no more than six different y-scales can be used at one time.

There are two methods for retrieving paired function data from HEC-DSS. The first is the same as it is for time-series data as explained above. The second method for retrieving paired function data is done by specifying the curve numbers to be plotted from each pathname. This is done by the use of the 'CURVES' command. Paired function data is different from time-series data in that more than one curve may be stored in one pathname. The 'CURVES' command allows the user to specify which curves stored in the pathname are to be plotted. An example of the second method follows:

```
CURVES 4,6,8
PAT /JAMES RIVER/DR1/ELEVATION-DAMAGE//1980/PLAN.B/
```

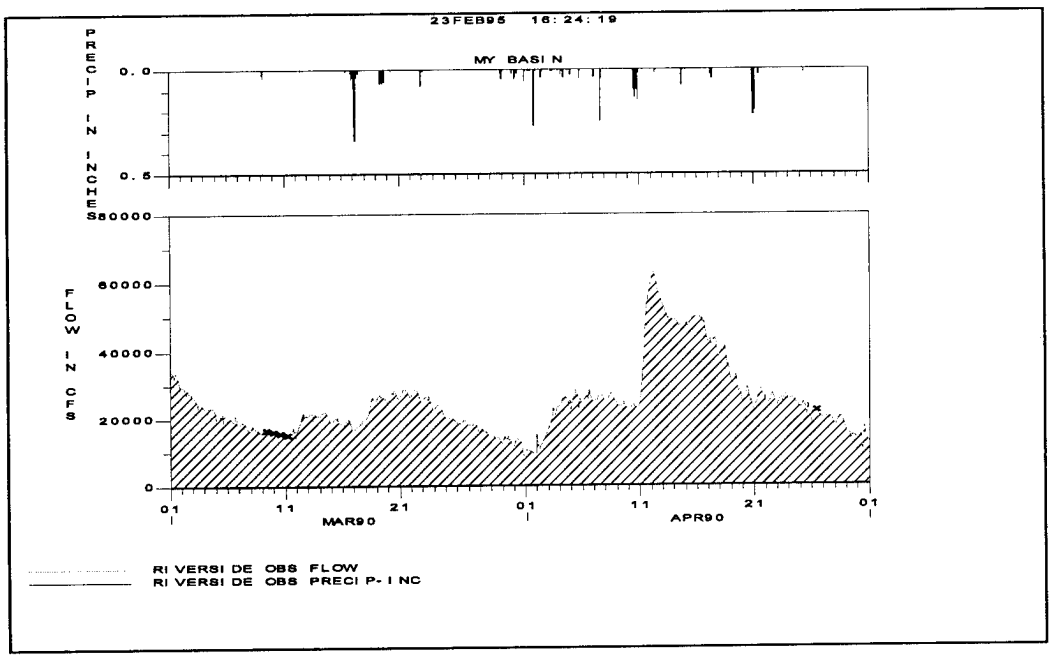
To facilitate the entry of pathnames, alternate input controls are available. The 'CATALOG' command provides a catalog listing of all pathnames by number and tag. Once this list exists, the user can specify the pathname by entering a number or tag instead of the pathname parts (PAT,number).

PREAD functions and macros can be used to enter frequently used pathnames and commands. PREAD is a high level command capability of HEC-DSS. Additional information concerning PREAD can be obtained from the PREAD manual which follows in this document.

Chapter 6

DSPLAY Example Application

The following examples are designed to give a basic understanding of the commands in DSPLAY that are used to produce a plot. The examples show the plot of precipitation and flow plotted in two separate y-axis windows but having a common x-axis. Note how the y-axis for precipitation has been inverted on both of the examples. The commands that produced the plots are listed below along with a short explanation.



Execute and specify SAMPLE.DSS as the DSS file

```
DSPLAY DSS=SAMPLE.DSS
```

Set the line colors to use with each curve being plotted

```
DL CU=1 COLOR=BLUE CU=2 COL=RED CU=3 COL=GREEN CU=4 COL=CYAN
```

Set the line style to use with each curve being plotted

```
DL CU=1 STY=DOTTED CU=2 STY=SOLID CU=3 STY=DASHED CU=4 STY=1234
```

Set the first curve to slashed fill and second curve to be solid fill

```
DL CU=1 SHADE=1 CU=2 SHADE=ON
```

The SW command is used to invert the precipitation which is the second curve (I) and is plotted in the second data type window. Note that the first curve is not inverted (N).

```
SW N,I
```

Set the time window of the data to be plotted

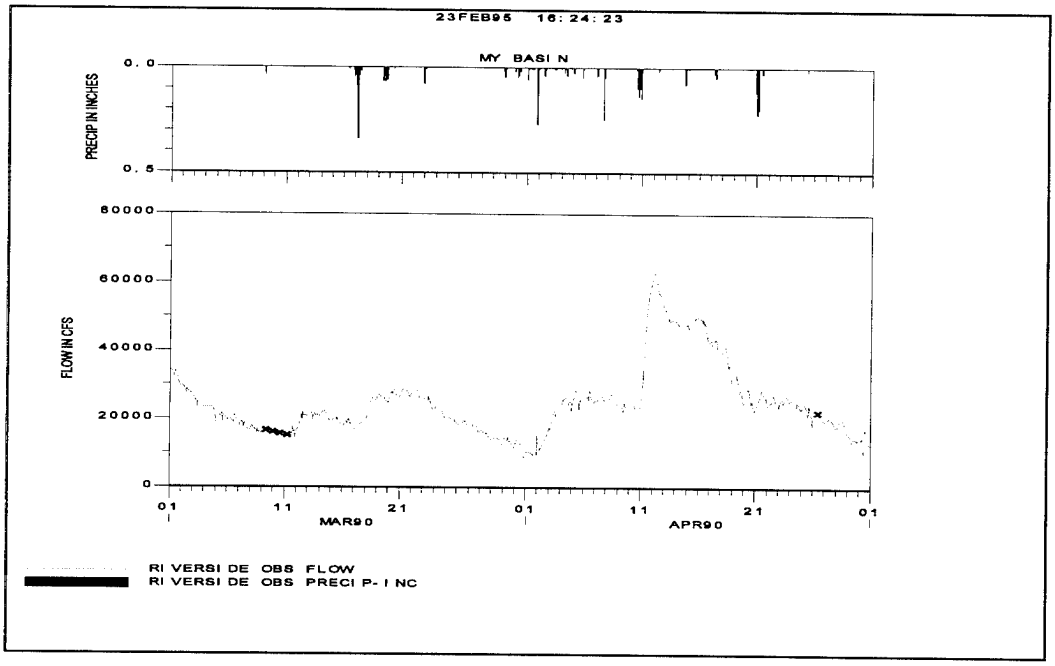
```
TIME 01MAR1990 0001 30APR1990 2400
```

Define the data to be plotted

```
PATH /MY BASIN/RIVERSIDE/FLOW/01MAR1990/1HOUR/OBS/
PATH /MY BASIN/RIVERSIDE/PRECIP-INC/01MAR1990/1HOUR/OBS/
```

Direct the program to plot the data

```
PLOT
```



Set the background to white, character and curve color to black
DL BACK=WHITE CHCOL=BLACK COL=BLACK
Set the grid color to black and the grid box to white
DG COL=BLACK CBOX=WHITE
Turn off the shade option for the first curve
DL CU=1 SHADE=OFF
Turn on the fill option for the legend
LE.P
Rotate the y-axis label 90 degrees
YL.R
Replot the data with the new plotting options specified above
PLOT

Appendix A

DSPLAY Commands

Command Index

Page

ACAD UNIX version 2
ADD 3
AXIS 4
CATALOG 5
CURVES 6
DATE 7
DECIMAL 8
DEVICE UNIX version 9
DEVICE PC version 12
DGRID 13
DLINE 15
DPATHNAM 18
EDIT 19
FACTOR 20
FINISH 21
FRAME 22
GETPATHN 23
HELP 24
INVERT UNIX version 25
LEGEND 26
LIST UNIX version 27
NORMALIZ 28
OPEN 29
PATHNAME 30
PLOT 31
QUIT 32
RESET 33
ROUND 34
ROUTE 35
RPLOT UNIX version 36
SAVE 37
SCALE 38
SCREEN 39
SPLIT 40
SPLOT UNIX version 41
STATUS 42
SWINDOW 43
TABULATE 44
TIME 45
TMARK 46
TSHIFT 47
USERLABE 48
WINDOW 49
XCLOSE UNIX version 51
XLABEL 52
YLABEL 53
YMARKER 54
YRANGE 55
ZOOM 56

Explanation of Command Description Writeup

The following is an explanation of command description layout on subsequent pages.

Use: Shows command use syntax. See below for further definition.

Description: Purpose and effects of the command. Background on its use and additional information, if necessary.

Options: Optional modifiers to command are described here. If no options are part of the command, this section does not appear.

Parameters: Required and optional parameters that may be specified for the command. If 'None' is shown as a parameter, define default behavior. Some commands have sub-commands as parameters and the sub-commands require that an = sign follow it.

Example: Examples of command usage.

Command syntax: **COMMAND[.options] [parameters]**

COMMAND	Name of command, unique 2 or more character abbreviation usually accepted and is case insensitive.
options	Modify behavior of command, if used, period separator between command is required. Options are case insensitive.
parameters	Command arguments names, sizes, etc., as appropriate. Parameters are generally case sensitive.
UPPERCASE	Uppercase designates items that are names or key words to be supplied by user as shown. In many cases the items may be abbreviated and actually entered in either upper or lower case.
lowercase	Lowercase designates items that are always supplied by the user. They may be names, numbers, etc and may actually be entered in upper or lower case.
[ ]	Optional item of command.
....	Preceding item may be repeated.
, or blank	Separator for parameters.
;	Multiple command separator. The concatenate character ";", can be changed to another character by specifying it on the execution line with the "CONCAT=" parameter.

ACAD UNIX version

Use:

```
AC[.option], [parameters]
```

Description:

The ACAD command is used to activate the ACAD capability. It can only be entered after a Plot has been displayed and is at the present time only available on the UNIX computer. For additional information, see Appendix B.

Option:

`E` Edit or create an overlay graphics.

Parameters:

`None`	The program will prompt for an overlay name.
`abc...`	Overlay graphics name to plot or edit.

Example:

`AC,PLOT1`	Will Plot the Graphics Saved under PLOT1 Name.
`AC.E,PLOT1`	ACAD editing or creation.

ADD

Use:

```
ADD [parameters]
```

Description:

The ADD command is a post plot command that allows the user to combine one or more of the curves plotted into a new curve. The new curve that is generated can be plotted and saved with the SAVE command. The curves can be added or subtracted. To subtract a curve the curve number is specified as a negative curve number. The program does not check to see that the data being combined occurs at the same time. Therefore the user must be certain that all data being combined is appropriate. In other words, the data should start at the same time and have the same time interval.

Parameters:

`None`	Nothing happens.
`Number`	+ or - Curve numbers (1,2,3,4,5,6,7).

Example:

`AD 2 1 -3`	Add curves 2 and 1 and subtract curve 3.

AXIS

Use:

```
AX[.options], [parameters]
```

Description:

The AXIS command is used to specify the axis label type for the X axis and Y axis. Up to six Y axis types may be entered. The LOG axis type should only be used with data that is greater than 0.0. In addition, the user can force the x-axis scale to be in probability units by using either the % (percent units 0<>100) or P (decimal units 0<>1) options. Probability data can be plotted as either linear or log by using the N option. Make certain that data plotted as probability will be within a valid range, if it is not, a large amount of errors will be printed out.

Options:

`None`	Both axis set Linear.
`N`	Use normal Linear or Log scales for x-axis scale.
`%`	Use percent probability for x-axis scale.
`P`	Use probability for x-axis scale.

Parameters:

`LIN`	Linear
`LOG`	Log

Example:

`AX,LIN,LOG,LOG`	Linear X axis and log Y axis.

CATALOG

Use:

```
CA[.options], [parameters]
```

Description:

The CATALOG command inventories the DSS file currently open and reports all pathnames and assigns a number to each. If the catalog file does not exist, it will be created. If the DSS file has changed, a new catalog must be created (N option). The catalog is usually sorted alphabetically by pathname parts. Each pathname has a record tag and reference number, either of which may be used in place of the pathname. The tags are semi-permanent but not necessarily unique. The reference numbers are unique, and are not permanently associated with the pathnames. On the MS DOS version, the catalog file is the DSS filename with the extension ".DSC". On the UNIX version, the catalog file is the DSS filename with the extension ".dsc".

The condensed catalog (C option) is designed for time-series data. The pathname parts are listed along with the date span for records where only the date part varies. The condensed catalog can only be created when a sorted catalog is generated with the default sort order.

The CATALOG command displays one page at a time, and the user enters a carriage return to display the following page. If the user enters a "Q" or any other command instead of a carriage return, the program will ignore any additional CATALOG output and execute the command entered.

Options:

`N`	New catalog.
`S`	Suppress list.
`P`	Send catalog to printer.
`U`	Unsorted catalog list.
`A`	Abbreviated list.
`C`	Condensed catalog list.
`X`	X Window environment.

Parameters:

`None`	Catalogs current DSS filename.
`Filename[:]`	Dss filename.

Example:

`CA.P`	Catalog old file and send list to printer.
`CA.N MYDSS`	Create a new catalog listing of DSS file MYDSS.

CURVES

Use:

```
CU[.options], [parameters]
```

Description:

The CURVES command is used to specify which curves are to be plotted from each pathname. This command only applies to paired function data. The S option allows the user to enter a starting curve number which causes the program to set the curve numbers to the starting number through starting number plus seven.

Options:

`S`	Start curve number.
`P`	Applies to previous pathname.

Parameters:

`None`	Uses curves 1, 2, 3, 4, 5, 6, 7 and 8.
`n,m,....`	Numbers that correspond to curves in Pathname.

Example:

`CU,1,5,9`	Curves 1,5, and 9 are to be plotted.
`CU.S 15`	Curves 15 through 22 are to be plotted.

DATE

Use:

```
DA[.options], [parameters]
```

Description:

The DATE command is used to turn ON/OFF the date and time label which appears on the plot. The L option allows the user to center the date any where on the screen by the use of screen coordinates (0-1023 x and 0-780 y) and specify the character size to use (1-4 on Tektronix 4014).

Option:

`L`	Define character size and x-y location of date label.

Parameters:

`None`	Will turn on the date label and place it at its default location.
`ON`	Will turn on the date label and place it at its default location.
`OFF`	Will turn off the date label.
`n,x,y`	Character size, x and y screen coordinate location.

Example:

```
DA.L 1,90,760
DA,OFF
```

DECIMAL

Use:

```
DEC, [parameters]
```

Description:

The DECIMAL command is used to set the number of decimal places to print for each curve that is being tabulated. The tabulation fields are a fixed width of 13 characters, therefore the user should take care that the maximum values to be tabulated will fit within the fixed width.

Parameters:

`None`	Will set all decimal places to 3.
`n,m,...`	Integer values 0 through 10.

Example:

```
DEC 3,3,2,2,1,0,0
```

DEVICE UNIX version

Use:

```
DEV[.options], [parameters]
```

Description:

The DEVICE command is used to specify the type of terminal being used and the environment under which it will function. If no parameters are entered, the default device is assumed to be a nongraphics device. If parameters are entered, it is required that a legitimate device be specified. For the 4510 rasterizer device, the user should first specify the Tektronix device that the rasterizer will emulate.

Options:

X	This option is only valid for the XTERM device. It provides the user the capability of opening additional x-windows without erasing the existing plot on the existing window. The "XC" command can be used to close all existing windows.

Parameters:

None	Device will be set to non-graphics (Alpha).
Device	ALPHA, 4014, 4105-4113, 4207, TABG, 4510 (rasterizer), HDS, BATCH, INTER, XTERM, POST, CPOST and HP

The following parameters are specific to the Tektronix 4510 rasterizer.

SG=	Turn ON or OFF the Tektronix segment option. The segment option is only for Tektronix 4100/4200 series terminals.
ORIENTATION=	Used with the 4510 rasterizer device to specify the orientation of the plot. Options are as follows: LA (landscape), BO (portrait bottom), CE (portrait center), and TO (portrait top). Landscape is the default.
PP=	Turn ON or OFF the pre/post plot option
SF=	Turn ON or OFF the 4100/4200 software font
NO=	Turn ON or OFF the automatic setting of color and line types when a device is changed

The following parameters are specific to the HP plotter.

MODEL=	HP plotter model number
PAPER=	Paper size (A B or C)
VELOCITY=	Pen speed in percent
PORT=	**Not available** - Default set to DEF (deferred plotting).
TERM=	Used to redirect the graphics (HPGL) output to a file name.
GSTOP=	ON/OFF Set it to ON if the program is to allow graphics commands to be specified after the plot command. This allows the use of the ACAD command. Default is OFF.

DEVICE UNIX version (continued)

The following parameters are specific to the XTERM device.

XPOSITION=	Upper left hand corner x-coordinate of graphics window.
YPOSITION=	Upper left hand corner y-coordinate of graphics window.
WIDTH=	Width of graphics window (0-1024).
HEIGHT=	Height of graphics window (0-1024).
INVERT	Used to invert the background and foreground colors.
XWINDOW=	FLOAT/FIX Default is Fixed; which means the graphics window will not be closed automatically after each plot. FLOAT will close graphics window after each plot. The graphics window can be closed selectively by using the XCLOSE command when the window is in FIXED mode.
HOLDPLOT=	ON/OFF When set to ON; will hold the graphics plot in an active state until a key is entered or a mouse button is pressed inside the graphics window. This makes the contents of the graphics window permanent and prevents it from being deleted if another window is placed on top. The default is OFF. Note: commands will not be executed until a key or button is pressed in the graphics window. The following single key commands are available:

KEY	Command
W	Windowing command is initiated
E	Graphical data editing is initiated
T	TAB.X command is initiated
I	Will cause the back/foreground to invert
P	Will replot the data
Q	Quits the plot
X	Clears the plot window
K	Exit and create a new plotting window without erasing the previous plotting window.
0 - 9	Run macro name "MAC#" where # is a number between 0 and 9.
	Activates user input commands from the graphics area.
	Enter "^" followed by a valid DSPLAY or PREAD command.

GINPUT=	ON/OFF When set to ON; input will be accepted from the graphics window when displaying a plot. Note that if a command is given that returns you to the user input, the input prompt will be modified to "GD>". The default is ON.

The following are specific to the PostScript Plotter Option.

POST=	PostScript option is being used. Specify the output postscript file name after the "=". If no filename is specified, the output will be written to a default name of "post".
CPOST=	The same as for "POSTSCRIPT" parameter with the addition that a color printer is to be used.
PSFONT=	This parameter can be used to specify a postscript font name to use.

The default font is Courier. The valid fonts available are as follows and can be specified by number or by name. If the name is used, make certain that it is used exactly as listed below.

No.	Font Name	No.	Font Name
1	Courier	21	Bookman-LightItalic
2	Courier-Bold	22	Helvetica-Narrow
3	Courier-Oblique	23	Helvetica-Narrow-Bold
4	Courier-BoldOblique	24	Helvetica-Narrow-Oblique
5	Times-Roman	25	Helvetica-Narrow-BoldOblique
6	Times-Bold	26	NewCentury-SchlBk-Roman
7	Times-Italic	27	NewCentury-SchlBk-Bold
8	Times-BoldItalic	28	NewCentury-SchlBk-Italic
9	Helvetica	29	NewCentury-SchlBk-BoldItalic
10	Helvetica-Bold	30	Palatino-Roman
11	Helvetica-Oblique	31	Palatino-Bold
12	Helvetica-BoldOblique	32	Palatino-Italic
13	Symbol	33	Palatino-BoldItalic
14	AvantGarde-Book	34	ZapfChancery-MediumItalic
15	AvantGarde-BookOblique	35	ZapfDingbats
16	AvantGarde-Demi	36	Helvetica-Condensed
17	AvantGarde-DemiOblique	37	Helvetica-Condensed-Bold
18	Bookman-Demi	38	Helvetica-Cond-BoldOblique
19	Bookman-DemiItalic	39	Helvetica-Condensed-Oblique
20	Bookman-Light		

`DUPLEXMODE=` Turn duplexmode ON or OFF. Default is OFF.

`TUMBLE=` Turn tumble mode ON or OFF. Default is OFF.

Example:

```
DEV,4107
DEV HP MO=7475 TERM=hpgl.out PA=A
DEV XTERM XPOS=150 YPOS=50 WIDTH=600 HEIGHT=400 INVERT
DEV,4109,4510,TERM=plot.out
DEV POST=postoutput PSFONT=Courier-Bold
```

DEVICE PC version

Use:

```
DEV, [parameters]
```

Description:

The DEVICE command serves two functions on the PC. It selects the device driver to be used and an optional coordinate transform. The coordinate transforms determine how much of the screen or page is used. FULL uses the entire area. ASPECT preserves the aspect ratio, (ie, a circle symbol is drawn as a circle not an ellipse). The DEVICE command is also used to turn the MOUSE on or off. When the MOUSE is turned off, the cursor is controlled by the keyboard arrow keys.

If no parameters are entered, the device is assumed to be a non-graphics device. Enter PC as a parameter in order to set it back to a graphics device.

When the META device driver is specified, DSPLAY will automatically write the plots to separate files that start with the letters "META" followed by a number count that goes from 01 through 99 and has an extension of ".CGM". The program will not overwrite an existing file, therefore it is up to the user to remove all existing META##.CGM files. The user can specify a different name to use by entering META as a command followed by a four character name to use. The advantage of using META files is that they can be easily imported into a large number of third party programs such as Lotus FREE LANCE and be further modified graphically.

Parameters:

`None`	Device will be set to non-graphics (Alpha).
`PC, MOUSE, SCREEN, PRINTER, PLOTTER,META`	Valid devices.
`ON/OFF, FULL, ASPECT, SHORT`	Valid transforms.

Example:

`DEV SCREEN`	Select the screen.
`DEV PRINTER ASPECT`	Use the ASPECT coordinate transform.
`DEV MOUSE ON`	Turn the mouse on.
`DEV META`	Will use the META device driver.
`META PLOT`	Will write ".CGM" plots to file names of the form "PLOT##.CGM"

DGRID

Use:

```
DG, parameters
```

Description:

The define grid (DGRID) command will allow the user to specify the grid line width, color, style, tick type, and line width of the axis box for each of the three possible data windows available in the program and for both the major and minor grids. (Note: Do not use in conjunction with the old GRID command).

Parameters:

DWINDOW=	The user can specify the data window to which the following parameters apply. This is done by specifying **ALL, TOP, MIDDLE, or BOTTOM.** If no DW parameter is used then **ALL** is assumed.
GRID=	The user can specify which of the two grids for which the following parameters apply. This is done by specifying **ALL, MAJOR, or MINOR**.
WIDTH=	The user specifies the width of the line to use with the grid specified. Two widths are available for the Tektronix 4114, **THICK and THIN**. THIN is the standard line width. Only devices that have the capability of producing different line widths will be able to use this option. At the present time only the Tektronix 4114, Tektronix rasterizer, and the Personal Computer are able to provide different line widths. In addition to the two line widths mentioned above, the user can specify a line width in raster units. Depending on the device being used, the program will only allow a valid width to be used. For example, the rasterizer has line widths of **1, 2, 3, and 4 rasters**. The PC has an unlimited line width.
COLOR=	The user specifies the color index number to use with the grid specified. If an HP pen plotter is being used the color index will function as a pen number. Consult your color terminal for information as to what colors have been assigned to each possible color index value. In addition to using color index numbers, the program will accept the following; **BLACK, WHITE, RED, GREEN, BLUE, CYAN, MAGENTA, YELLOW, and ORANGE**.

STYLE=	The user specifies the style of grid line to use by entering **NONE, SOLID, or DOTTED**. Note: The default is NONE, therefore the **STYLE=** must be specified in order to Produce a grid.
TICK=	The user specifies the ticks orientation by entering **IN, OUT, or BOTH**.
CROSS=	The user specifies whether the major grids will have a cross drawn when a dotted style is being used. The user enters **ON or OFF**.
BOX=	The user can set the line width being used for the axis box. The color of the box will be based on the major tick color. See WIDTH command above for additional information of valid parameters to enter.
CBOX=	The user can set the background color being used for the axis box. See COLOR command for valid colors.

Example:

```
DG DW=TOP GR=MAJ COL=GREEN STY=SOLID WI=THICK BOX=THICK CBOX=CYAN
```

DLINE

Use:

```
DL[.options], parameters
```

Description:

The define line (DLINE) command can be used to specify the color, shade pattern, square option, line type, symbol, symbol size, type of data and width of line. In addition it can also be used to set the back-ground color and character color. The graphics options are dependent on the output device being used.

Options:

The following options are only available when the SHADE subcommand is used.

P	Use a pattern instead of solid color for shade index 0.
V	Use the vertical line style of shading.
S	Use the vertical line style of shading that suppresses overlap for curves of increasing magnitude.

Parameters:

CURVE=	The user specifies the curve number that the following parameters are to be used with. Legitimate values are numbers **1 through 7 and ALL**. If no CU parameter is used then the program will assume that the data applies to all curves or if a numeric value between 1 and 7 is found it will be assumed to be the curve number.
STYLE=	The user specifies the line style to use. The following line styles are acceptable; **SOLID, DOTTED, DASHED, NONE, -1 (none), 0 (solid), 1 (long dash), 2 (dash), 3 (dot dash), 4 (dot)**. Software line styles are specified by concatenating integers describing line segment length and visibility. Odd integers are visible and even integers are invisible. A maximum of six integers can be specified for the software line styles.
COLOR=	The user specifies the color index number to use with the curve specified. If an HP pen plotter is being used the color index will function as a pen number. Consult your color terminal for information as to what colors have been assigned to each possible color index value. In addition to using color index numbers, the program will accept the following; **BLACK, WHITE, RED, GREEN, BLUE, CYAN, MAGENTA, YELLOW, and ORANGE**.

`WIDTH=`

The user specifies the width of the line to use with the curve specified. Two widths are available for the Tektronix 4114; **THICK and THIN**. THIN is the standard line width. Only devices that have the capability of producing different line widths will be able to use this option. At the present time only the Tektronix 4114, Tektronix rasterizer, and the Personal Computer are able to provide different line widths. In addition to the two line widths mentioned above, the user can specify a line width in raster units. Depending on the device being used, the program will only allow a valid width to be used. For example, the rasterizer has four line widths of **1, 2, 3, and 4**. The PC has an unlimited line width.

`SYMBOL=`

The user specifies the symbol to use with the curve specified. The user enters an index number to specify the symbol. In addition to index numbers, the user can use the following symbol names; **CIRCLE, CROSS, SQUARE, TRIANGLE, STAR, DIAMOND, and WEDGE**. The following index numbers are also valid entries.

0	no symbol	7	bar circle
1	circle	8	cross
2	x	9	up arrow
3	triangle	10	down arrow
4	square	11	wedge
5	star	33-127	ASCII characters
6	diamond		

`SIZESYMB=`

The user specifies a size factor that can be used to make the symbol smaller by using a number less than 1 or larger by using a number greater than 1.

`SQUARE=`

The user can control the use of the square form of plotting the curve by entering **DEF** (use default pathname header setting), **ON** (to force it on based on the ending time period), **START** (to force on based on starting time period) and **OFF** (to force it off).

DLINE (continued)

`SHADE=` The user can turn **ON/OFF** the shade option. The shading option is defaulted to shading all curves when using a Tektronix 41/4200 series terminal or a PC and shading only the first curve when a Tektronix 4014 terminal or monochrome terminal is used. The user has the option of shading specific curves by specifying the number of raster units between solid vertical shade lines for the 4014 or turning off the shading altogether by specifying a **0 for the 4014 or a -1 for the color terminals**. The user also has the option of specifying a pattern to be used on the color series terminal by specifying **numbers greater than 0 or by using the .P option if the shade number is set to 0**. In addition to using a pattern index number, the user can specify the following types; **HATCH, VERTICAL, HORIZONTAL, GRID, BRICK, DOTTED, and SLANTED**. The Tektronix color terminals can be forced to use the vertical line shading by using the **.V option**. The patterns available on the Tektronix and PC can be determined by reading the terminal manual or by using DSPLAY to plot the different patterns.

`TYPE=` The user can temporarily change the type of the data, as defined by the header information in the pathname, by entering it for the curves being used. The program assigns **types 1 2 3 4 5 and 6** as the temporary types, therefore the user should set the temporary type for each curve being used. In other words, if this parameter is used with one curve, it must be used with all curves being plotted. To turn the types back to their default setting, just set one curve to **OFF**. To restore the temporary data types, just set one curve to **ON**.

`CHCOL=` The user specifies the color index number to use for the character color. If an HP pen plotter is being used the color index will function as a pen number. Consult your color terminal for information as to what colors have been assigned to each possible color index value. In addition to using color index numbers, the program will accept the following; **BLACK, WHITE, RED, GREEN, BLUE, CYAN, MAGENTA, YELLOW, and ORANGE**.

`BACKG=` The user specifies the color index number to use for the background color. Consult your color terminal for information as to what colors have been assigned to each possible color index value. In addition to using color index numbers, the program will accept the following; **BLACK, WHITE, RED, GREEN, BLUE, CYAN, MAGENTA, YELLOW, and ORANGE**.

Example:

```
DL CU=1 COL=GREEN STY=SOLID WI=THICK CU=2 COL=BLUE STY=DOT
```

DPATHNAM

Use:

```
DP[.options], [parameters]
```

Description:

The DPATHNAM command displays pathnames and their reference numbers from the catalog file (see catalog command). The DP command has the selective display capability which provides the option of selecting and listing only certain pathnames based on matching pathname parts.

Option:

`F`	In an interactive session, this option displays all pathnames in the catalog without pausing after each screen. All pathnames are automatically displayed in a batch job.
`X`	Activate the x-window style of output. This option is only valid with an XTERM device.
`C`	Display information about record. This information is the same as that printed in the CHECK command in DSSUTL.

Parameters:

`None`	Displays all pathnames.
`A=..B=..etc`	Display catalog based on pathname parts.
`n`	Integer number of starting reference pathname number to display.
`n-m`	Displays pathnames n through m.

Example:

`DP,C=FLOW`	Displays all pathnames that have a C part equal to FLOW.

EDIT

Use:

```
ED[.options], [parameter]
```

Description:

The EDIT command is used to interactively edit the last curve plotted on the screen by the use of Cross Hair Cursor. The user can edit any of the curves plotted by entering the curve number as a parameter. The user must use specific keys to edit the plotted data, all other keys will be ignored. The Cross Hair Cursor is positioned by the use of the arrow keys or by a mouse if present and active. The left mouse button is equivalent to entering a "C" on the keyboard and the right mouse button is equivalent to a "D".

Options:

P	Single Point Editing.
D	Do not Merge Old Data With Digitized Data.
S	Single data point editing by use of the numeric keys; Valid numeric keys are: **4 (left), 6 (right), 8 (up), 2 (down), 7 (first value), 9 (last value), 1 (left missing value) and 3 (right missing value)**. The < moves to the left half way between the present and previous left position and > does the same but to the right. The + or . keys increase and the - or 0 keys decrease the rate of cross hair movement.

Parameters:

None	Edit the last curve plotted.
n	Curve number (1-7) to edit.

Valid Keys:

C	Digitize a point on the screen.
I	Input y-axis data value through the keyboard. The x-axis data value is based on the cross hairs.
M	A missing value (-901) is digitized. Do not mix the use of M with C and I digitized points when using line editing mode.
B	Cancel previous digitized point.
D	One entry specifies that present digitized line segment is finished. Two entries in a row will terminate editing and plot the edited curve.

Example:

ED	Cross hair digitizing

FACTOR

Use:

```
FA, parameters
```

Description:

The FACTOR command is used to change the size of the plot. There are three methods available for changing the size of the plot. The first method is to specify a factor that is less than 1.0. The second method is to specify an integer number between 1 and 14 which corresponds to a predefined screen location (minimum and maximum x y screen coordinates). The third method specifies the screen location based on a row and column location. The third method requires eight parameters be set initially, but once the parameters have been set, the user only has to specify those parameters that change. (NOTE: x-axis 1023 rasters; y-axis 780 rasters.)

Parameters:

none	Set Screen coordinates back to default settings.
x	Real Number Greater than 0.0 and less than .99
1	Screen Coordinates: **100 420 160 720** (LEFT HALF).
2	Screen Coordinates: **600 920 160 720** (RIGHT HALF).
3	Screen Coordinates: **160 850 150 350** (LOWER HALF).
4	Screen Coordinates: **160 850 520 720** (UPPER HALF).
5	Screen Coordinates: **100 420 150 370** (LOWER LEFT QUARTER).
6	Screen Coordinates: **600 920 150 370** (LOWER RIGHT QUARTER).
7	Screen Coordinates: **100 420 500 720** (UPPER LEFT QUARTER).
8	Screen Coordinates: **600 920 500 720** (UPPER RIGHT QUARTER).
9	Screen Coordinates: **100 420 150 290** (LOWER LEFT SIXTH).
10	Screen Coordinates: **100 420 370 510** (MIDDLE LEFT SIXTH).
11	Screen Coordinates: **100 420 590 750** (UPPER LEFT SIXTH).
12	Screen Coordinates: **600 920 150 290** (LOWER RIGHT SIXTH).
13	Screen Coordinates: **600 920 370 510** (MIDDLE RIGHT SIXTH).
14	Screen Coordinates: **600 920 590 750** (UPPER RIGHT SIXTH).
n,m,k,n,m,k,n,m	Row, Column, Total Rows, Total Columns, Left margin, Right margin, Bottom margin, Top margin in rasters.

Example:

```
FA,.5
FA 1,1,2,2,100,100,150,25
```

FINISH

Use:

```
FI
```

Description:

The FI command is used to terminate the execution of the program.

Example:

`FI`	Terminates execution.

FRAME

Use:

```
FR[.options], [parameters]
```

Description:

The FRAME command is used to set the plot's frame settings which consist of interval in raster units and the number of iterations which determine the thickness of the frame. In addition to the standard frame, four user defined frames can be defined by using the U option. The user defined frame can be sized and positioned any place on the screen.

Option:

`U`	User defined option.

Parameters:

`None`	Will turn on the frame.
`OFF`	Will turn off the frame.
`n,m,k,x,x,y,y`	Frame number, Interval, Iterations, x-min, x-max, y-min, y-max

Example:

```
FR,1,5
FR.U 1,1,4,18,500,10,90
```

GETPATHN

Use:

```
GE, [parameters]
```

Description:

The GETPATHN command is used to select pathnames based on a catalog sort by part or parts of a pathname. Only the first seven pathnames that meet the selected pathname parts are plotted.

Parameters:

`None`	The first seven pathnames in the catalog will be used.
`A=.., B=..,etc`	Select pathnames based on parts specified.

Example:

`GE,A=SCIOTO,C=FLOW`	The first seven pathnames that have the A part of SCIOTO and B part of FLOW will be plotted.

HELP

Use:

```
HE [.option] [parameters]
```

Description:

The HELP command is used to request a listing of the commands available and to provide a detailed description of the commands.

The HELP command displays one page at a time and the user enters a carriage return to display the following page. If the user enters a "Q" or any other command instead of a carriage return, the program will ignore any additional HELP output and execute the command entered.

Option:

`X`	Activate the x-window style of output. This option is only valid with an XTERM device.

Parameters:

`None`	Provide list of available commands.
`ALL`	Define all commands.
`Command`	Define specified command.

Example:

`HELP,OPEN`	Define OPEN command.

INVERT UNIX version

Use:

```
IN
```

Description:

The INVERT command is only functional when the XTERM device is active. It allows the user to invert the background and foreground colors.

Example:

```
INVERT
```

LEGEND

Use:

```
LE[.options], [parameters]
```

Description:

The LEGEND command is used to specify user defined labels for each of the curves or it can be used to set the default legend labels for both paired function and time series data. The default legend labels are composed of the A - F pathname parts for both paired and time series data, but paired data also uses the curve label stored in the header. The curve label is defined in this command as the G part.

Options:

A	Will use the B part of the pathname and add the user defined label to it.
XT	Redefine the time series default legend components.
XP	Redefine the paired function default legend components.
L	Legend rows, columns, character size, and location.
P	Turn ON/OFF polygon fill option. Default is OFF.

Parameters:

None	Will set the legend based on pathname parts BFC (time series) and BDGF (paired data).
n,abc...	Curve number, Alpha-numeric label of no more than 40 characters.
ABCDEFG	Pathname parts (ABCDEFG).
n,k,k,x,x,y,y	Rows, columns, character size, x-min, x-max, y-min, y-max.
OFF	Will turn off the legend labels when used with the L option.

Example:

```
LE,1, COMPUTED STAGE AT MARYSVILLE
LE.A,2, - OBSERVED STAGE
LE.XT ABFC
LE.XP ABEGF
LE.L 7,1,4,600,1020,100,450
LE.L OFF
```

LISTX UNIX version

Use:

```
LI, [parameters]
```

Description:

The LISTX command is used to list a text file in an X Window. This option is only active when an XTERM device has been specified. The file listed can be sent to the default printer by clicking on the hardcopy button.

Parameters:

`None`	Nothing will happen.
`Filename`	The full path and name of the file being listed.

Example:

```
LIST /usr2/reports/flooddam.rpt
```

NORMALIZ

Use:

```
NO [parameters]
```

Description:

The NORMALIZ command is used to normalize the y-axis data values by subtracting from each the first data value.

Parameters:

`None`	Turn off normalizing of the data.
`ON`	Turn on normalizing of the data.
`OFF`	Turn off normalizing of the data.

Example:

```
NO, ON
```

OPEN

Use:

```
OP, [parameters]
```

Description:

The OPEN command is used to open a new DSS file. A maximum of five DSS files may be open at one time. The last DSS file opened or referenced by a PA, PL, or TA command is considered the default DSS file. The user may refer to any opened DSS file by entering the name followed by a ':' or by a number (1-5) followed by a ':'. On the PC version, if no filename follows the command, a list of DSS files within that directory is displayed. By moving the cursor to a filename then pressing the return key, that file will be opened.

Parameters:

`None`	Program will prompt for a DSS filename.
`Filename[:]`	Dss filename.

Example:

```
OP,YOURDSS
```

PATHNAME

Use:

```
PA [.options], [parameters]
```

Description:

The PATHNAME command is used to define the pathnames to be used for plotting or tabulating. In addition to entering the pathname parts, the user can specify the DSS file that the pathname belongs to by entering as the first parameter the DSS filename followed by a colon. If no DSS filename is entered, the last referenced DSS file will be used.

Options:

S	Switch the X and Y axis for paired data.
A	Modify all pathnames entered by the pathname parts entered.
C	Display information about record. This information is the same as that printed in the CHECK command in DSSUTL.

Parameters:

[dssfile:]	May precede any of the pathname references that follow and will continue to be used until changed.
/a/b/c/d/e/f/	All pathname parts defined.
A=..,B=...	Pathname parts A through F specified.
n,m,...	Pathname numbers based on catalog listing.
tag1,tag2...	Pathname tags based on catalog listing.

Examples:

```
PA,/SCI/STJOE/FLOOD-RES IN/01JAN59/6HOUR/TEST/
PA,A=SCI,B=STJOE,C=FLOOD-RES IN,D=01JAN59,E=6HOUR,F=TEST
PA,25
PA,DSSDAT:/SCI/STJOE/FLOOD-RES IN/01JAN59/6HOUR/TEST/
PA,MASDB:T12, T1812
PA.A B=NEWSTJOE
```

PLOT

Use:

```
PL[.options], [parameters]
```

Description:

The PLOT command is used to execute the plotting of the DSS data. The user may enter the pathname parts, pathname numbers or tags as parameters. In addition to entering the pathname parts, the user can specify the DSS file that the pathname belongs to by entering as the first parameter, the DSS filename followed by a colon. If no DSS filename is entered, the last referenced DSS file will be used.

Options:

O	Will cause all curves to be shifted in time to start at the same time as the first curve.
M	Will cause all missing value markers not to be displayed.
X	Plotting curve order is based on maximum values.
A	Plotting curve order is based on maximum mean values.
E	Do not erase the screen and do not exit graphics mode. (Note: This option can be used to place multiple plots on a single screen).
0	Plot only the curves.
1	Plot curves and legend only.
C	Display information about record. This information is the same as that printed in the CHECK command in DSSUTL.

Parameters:

None	Will plot previously defined pathnames.
[dssfile:]	May precede any of the pathname references that follow and will continue to be used until changed.
/a/b/c/d/e/f/	All pathname parts defined.
A=..,B=...	Pathname parts A through F specified.
n,m,...	Pathname numbers based on catalog listing.
tag1,tag2...	Pathname tags based on catalog listing.

Examples:

```
PL,/SCI/STJOE/FLOOD-RES IN/01JAN59/6HOUR/TEST/
PL,A=SCI,B=STJOE,C=FLOOD-RES IN,D=01JAN59,E=6HOUR,F=TEST
PL DSSFILE: 25,30,41
PL MASDB: T12, T1812
```

QUIT

Use:

QU

Description:

The QUIT command is used to exit the DATA DISPLAY MODULE. A carriage return and no command will have the same effect as the QUIT command. The QUIT command is used primarily when PREAD macros are being used, in a batch mode and when several individual plots are being plotted together by the use of the "E" option of the PLOT command. The QUIT command will signal a hardcopy when the plotting is being directed to a printer.

RESET

Use:

```
RE[.options]
```

Description:

The RESET command is used to clear out the pathnames back to initial conditions and turn off time shift command if entered. To do a global reset back to initial conditions on all graphics parameters use the A option.

Option:

`A`	Will cause all graphics settings to be reset to default settings.

Example:

`RE`	Resets pathnames defined back to zero.
`RE.A`	Resets all graphics and pathnames to initial default settings.

ROUND

Use:

```
RO, [parameters]
```

Description:

The ROUND command is used to specify the value to round the edited data points. It is impossible to edit with the screen cross hairs to an exact value, therefore it is usually practical to set the data being digitized to be rounded to some appropriate value. For example, if flow values are being edited, the values may be rounded to the nearest 10 to 100 cfs.

Parameters:

`None`	Turn off the rounding of the data.
`x`	Positive Real Numbers.

Example:

`RO,100`	Round Digitized Data to Nearest 100.

ROUTE

Use:

```
ROUT, parameters
```

Description:

The ROUTE command is a post plot command that allows the user to do a Muskingum routing of the curves plotted. The new curve that is generated can be plotted and saved with the SAVE command. The program requires that the user specify the curve number to be routed, the number of routing steps, the Muskingum K (travel time) and X (attenuation).

Parameters:

NSTPS	Number of routing steps = K/(t2-t1) Constraint 2KX < (t2-t1) <= K
K	Muskingum travel time coefficient in hours
X	Muskingum attenuation coefficient (0.0 <= X <= .5) X of 0.0 produces maximum attenuation.

Example:

`ROUTE 1 3 9.5 .4`	Route curve 1 using 3 routing steps, 9.5 hours travel time, and attenuation coefficient of 0.4.

RPLOT UNIX version

Use:

```
RP[.option], parameters
```

Description:

The RPLOT command is used to replot plots that have been saved by the SPLOT command. This option turns off the SPLOT option once entered. This Command is not available on the PC version.

Options:

`S`	Prevent screen clear from occurring.
`Z`	Plot from a DSS file, plot name is a pathname.

Parameters:

`abc...`	Plot Name not to Exceed 8 Characters.
`CATALOG`	Will List Plot Names Available.

Example:

`RP,PLOT1`	Will Plot the Graphics Saved under PLOT1 Name.

SAVE

Use:

```
SA, [parameters]
```

Description:

The SAVE command is used to save an edited curve to a DSS file. This command can only be used after the EDIT command has been used.

Parameters:

`None`	The program will prompt for a pathname to save the edited data.
`A=..,B=...`	Pathname parts to use for saving the edited data.
`REPLACE`	Original pathname will be used to save the edited data.

Example:

`SAVE F=EDITED`	Save edited data in original pathname but use an F part of EDITED.

SCALE

Use:

```
SCA, [parameters]
```

Description:

The SCALE command is used to set an absolute x or y axis scale. The scale can be set by specifying the minimum, maximum, and the number of divisions to use for both the major and minor ticks. For the y axis, the data type index must also be specified. (Note: To permanently clear both X and Y scales, enter SCALE with no parameters.)

Parameters:

`None`	Scale option is turned off.
`X,a,b,m,n`	X, minimum x, maximum x, number of major and minor divisions (X, ON/OFF).
`Y,m,a,b,n`	Y, data type index, minimum y, maximum y, number of major and minor divisions (Y, ON/OFF).
`X,a,b,c,d`	X, minimum date, time, maximum date, time (Y, ON/OFF).

Example:

```
SCA X 0.0,100.0,10,5
SCA X OFF
SCA Y 1 0.0 10000 5
SCA Y OFF
SCA X 01JAN1989 0100 17JAN1989 2400
SCA OFF
```

SCREEN

Use:

```
SCR, [parameters]
```

Description:

The SCREEN command is used to define the physical screen plotting dimensions, based on raster units that range between 0 and 1023 on the X-axis and 0 and 780 on the Y-axis.

Parameters:

`None`	Sets screen coordinate window to 160, 850, 155 and 691.
`n,m,i,j`	Integer numbers of xmin, xmax, ymin and ymax.

Example:

`SCR,150,400,200,400`	Xmin, Xmax, Ymin, Ymax.

SPLIT

Use:

```
SPLI, [parameters]
```

Description:

The SPLIT command is used to specify the percentage that the plotting windows will occupy when multiple windows have been specified by the SWINDOW command. The first parameter value sets the percentage of the bottom window for a two window screen and the second parameter value sets the middle window for a three window screen. The number of windows per screen is based on the number of data types being plotted and the settings specified by the SWINDOW command.

Parameters:

`None`	Will set the parameters to the following: 2 windows 75% 25%, 3 windows 38% 37% 25%.
`n`	Percent of bottom window (For two or three window screen).
`n,m`	Percent of bottom window, Percent of middle window (For three window screen).

Example:

`SPLI,75`	Will split the screen with 75% for bottom window and 25% for the top window.
`SPLI,35,35`	Will split the screen 35% for bottom window, 35% for the middle window, and 30% for the top window.

SPLOT UNIX version

Use:

```
SPLO[.options], parameters
```

Description:

The SPLOT command is used to save a plot in a nonstandard format. Each plot is saved under a user defined name entered along with the command. Only one plot is saved for each command entered and saved under the name specified. The name used to save a plot must be different for each plot saved. If the Z option is used, the plot name has to be a DSS pathname. This command is not available on the PC version.

Options:

`S`	Prevent screen clear from occurring.
`Z`	Plot from a DSS file, plot name is a pathname.

Parameters:

`abc...`	An Alphanumeric Plot Name not to Exceed 8 Characters or a DSS pathname if the Z option is used.

Example:

`SPLO,PLOT1`	Will Save the Following Plot under PLOT1 Name.
`SPLO.z,/COTT/SP12////`	Will Save the Following Plot in an HEC-DSS file under pathname "`/COTT/SP12////`"

STATUS

Use:

```
ST, [parameters]
```

Description:

The STATUS command is used to get a list of the present settings and pathnames entered. The user can specify which settings to display by entering the command names that were used to set them as parameters.

The STATUS command displays one page at a time and the user enters a carriage return to display the following page. If the user enters a "Q" or any other command instead of a carriage return, the program will ignore any additional STATUS output and execute the command entered.

Parameters:

`None`	List all status settings.
`ALL`	List all status settings.
`Command`	List status of specified commands.

Example:

`ST,DEV,DLINE`	Will list current settings for DEVICE and DLINE.

SWINDOW

Use:

```
SW, [parameters]
```

Description:

The SWINDOW command is used to specify user defined instructions as to how the program will display different types of data that are being plotted together. The user has the capability of plotting up to six different types of data together on three separate windows. The user can specify the window (B-bottom, M-middle, or T-top), Y-axis label location (L-left or R-right), and direction of Y-axis (N-normal or I-inverted) for each of the possible six different data types that can be plotted together.

Parameters:

`None`	Sets window parameters for each curve to the following: LBN LMN LTN RBN RMN RTN.
`B M T`	Window letters (B-bottom, M-middle, or T-top).
`L R`	Y-axis label location (L-left or R-right).
`N I`	Y-axis direction (N-normal or I-inverted).

Example:

`SW,LBN,RBN`	First and second data types will plot on one window, with the first y-axis label appearing on the left and the second data type on the right, and both will be plotted in a normal fashion.
`SW,LBN,LTI`	First data type will plot on the bottom and the second on the top of a two window screen, with the y-axis labels on the left. The second data type is plotted inverted.

TABULATE

Use:

```
TA[.options], [parameters]
```

Description:

The TABULATE command is used to display the data retrieved in a tabular format. The user may enter the pathname parts, pathname numbers or tags as parameters. In addition to entering the pathname parts, the user can specify the DSS file that the pathname belongs to by entering, as the first parameter, the DSS filename followed by a colon. If no DSS filename is entered, the last referenced DSS file will be used.

The number of decimal places the data will be tabulated can be set by using the DECIMAL command.

The TABULATE command displays one page at a time and the user enters a carriage return to display the following page. If the user enters a "Q" or any other command instead of a carriage return, the program will ignore any additional TABULATE output and execute the command entered.

Options:

F	Will write the tabulation to an output file "tabfile.out".
S	Statistics of the data will be tabulated.
R	Rewind tabfile output.
X	Activate the X-Window style of output. This option is only valid with an XTERM device.

Parameters:

[dssfile:]	May precede any of the pathname references that follow and will continue to be used until changed.
/a/b/c/d/e/f	All pathname parts defined.
A=..,B=...	Selective pathname parts A through F specified.
n,m,...	Pathname numbers based on catalog listing.
tag1,tag2...	Pathname tags based on catalog listing.

Example:

```
TA,/SCI/STJOE/FLOOD-RES IN/01JAN59/6HOUR/TEST/
TA,A=SCI,B=STJOE,C=FLOOD-RES IN,D=01JAN59,E=6HOUR,F=TEST
TA.F,25,30,41
TA, T12, T1812
```

TIME

Use:

```
TI[.options], [parameters]
```

Description:

The TIME command is used to specify the starting and ending Military Time and Dates which will be used to define the time window for retrieving data from DSS for all the pathnames that follow. If no time window is set, data for the entire record is used.

A time must be given in 24 hour clock time. The date can be one of several styles, but must not contain any spaces within it. A 7 or 9 character military style date may be used.

A time window can be set relative to the system time by the single character "T", optionally followed by a +- sign, a number, then an "H" (hours), or a "D" (days), or a "Y" (years). In addition, a fixed hour for the current day may be specified by using "T" as the date portion, then specifying the time in the next field.

Options:

`P`	Applies to previous pathname.
`F`	Retrieve leading and trailing missing values if present.
`W`	Use the time window set for x-axis scale.

Parameters:

`None`	Turns off the time window.
`st,sd,et,ed`	Starting time, Starting date, Ending time, Ending date.

Example:

```
TI,01JAN1959,2400,31DEC1959,2400
TI,2400,01JAN1959,2400,31DEC1959
TI T-4H, T
TI T, 0200, T, 1600
TI
```

TMARK

Use:

```
TM, [parameters]
```

Descriptions:

The TMARK command is used to set the forecast date marker. The User can set a date or have the program read the date and time from the 'm' and 's' function key definition in file GENFUN which have been set by the MODCON program or through the PREAD TEACH command. The user can turn off the forecast marker by entering OFF as a parameter. The user can turn the forecast marker back on by entering ON as a parameter.

Parameters:

`None`	Will turn on the time mark.
`ON`	Will turn on the time mark.
`OFF`	Will turn off the time mark.
`n,date`	Time 24 hour clock, date (DayMonthYear)

Example:

```
TM,2400,01JAN1959
TM OFF
```

TSHIFT

Use:

```
TS[.options], parameters
```

Description:

The TSHIFT command is used to modify the data stored in a curve by shifting it by a specified number of days, raising or lowering its y-axis values by a specified amount, and multiplying all the y-axis values by a factor. If only a curve number is entered, then the curve will be shifted only in time and set to start at the same time as the first curve. The number of days to shift the curve can be specified by a pair of dates. The number of days to shift the curve is determined by the difference between the two dates entered. Only the time axis can be modified if dates are used.

Option:

`C`	Cross hairs will be used to shift the data; if a Y is entered after each location specified by the cross hairs, the y-axis will be shifted and the x-axis will be shifted; if any other key is used, only the x-axis will be shifted.

Parameters:

`n,m,x,y`	Curve number, Time shift in days, y-axis shift, y-axis factor (Standard Method).
`n,d1,d2`	Curve number, Date1, Date2 (Optional Method).

Example:

`TS,2`	Will shift curve number 2 to start at same time as curve 1.
`TS,1,10.5,0,1.5`	Shift curve 1 by 10.5 days and multiply by a factor of 1.5.

```
TS 2 01JAN77 1200 16FEB88 1000
```

USERLABE

Use:

```
US, [parameters]
```

Description:

The USERLABE command is used to specify a user defined label that will appear on the plot. The L option allows the user to center the label any where on the screen by the use of screen coordinates (0-1023 x and 0-780 y) and specify the character size to use (1-4 on Tektronix 4014). If no label is desired, enter 'NULL' as a parameter.

Option:

`L`	Define character size and x-y location of user label.

Parameters:

`None`	Will use the A pathname part as a label.
`abc...`	Alpha-numeric label of no more than 80 characters.
`'NULL'`	Do not plot a user label.
`n,x,y`	Character size and center x-y location of label. Only valid with the L option.

Example:

```
US,SCIOTO BASIN STUDY
US.L 1,90,760
```

WINDOW

Use:

```
WI, [parameters]
```

Description:

The WINDOW command is used to set a data window. The data will not be altered by this command, only the plot will be windowed. If no parameters are entered, cross hairs will appear. The user then manipulates the cross hairs and keys to either enter the window limits or enter a Z which activates the zoom window. The zoom window can be moved by moving the cross hairs to the location of the desired window and entering another Z. If the window is too small, repeated Z will enlarge the window. It is recommended that the zoom option only be used with 4100 series Tektronix terminals in the UNIX version. See ZOOM command for additional zooming capability.

Parameters:

`None`	The cross hairs will be used to establish the window boundary.
`x,x`	Axis Window Minimum and Maximum Values.
`y,y`	Axis Window Minimum and Maximum Values.

Valid Keys:

`W` This command is used to window out a part of the plot. There are three methods of windowing on the plotted data. Each method requires that two windowing locations be entered by using the cross hair cursor. After the first location is entered, the cross hair cursor appears for a second time, and the user specifies the second location of the window. When the 'W' is entered for the second cross hair cursor, the data within the window specified is plotted. The actual plot window is adjusted to provide a neat even tick interval. The three methods for windowing are described below. A window based on the time scale is achieved by moving the cross hair cursor horizontally and defining the maximum and minimum time scale values. A window based on the y-scale is achieved by moving the cross hair cursor vertically and defining the maximum and minimum y-scale values. A window based on both the time and y-scales is achieved by moving the cross hair cursor vertically and horizontally to define the maximum and minimum time and y-scale values.

`V` This command is used to draw a rectangle that shows the user the window and to set the plot range based on the absolute values set by the cross hairs. When this option is used, the cross hair cursor can be used by the user to redefine the window by entering a 'W'. If a 'W' is entered, the windowing procedure is repeated so that the window can be adjusted. If any other character is entered, the program proceeds to plot the data within the window.

Z This command is used to turn on the ZOOM command. See ZOOM and WINDOW commands for additional information. If the user enters any other character besides the ones mentioned above, the program will window based on the cross hair locations.

XCLOSE UNIX version

Use:

```
XC
```

Description:

The XCLOSE command is only functional when the XTERM device is active. It allows the user to close the graphics windows when the graphics window is in FIXED mode. Fixed graphics window mode keeps the graphics window open at all times until the program is terminated or the DEVICE command is used to set the XWINDOW subparameter equal to FLOATING mode.

Example:

```
XCLOSE
```

XLABEL

Use:

```
XL, [parameters]
```

Description:

The XLABEL command is used to specify a user label that will appear on the X-Axis of a plot. The XL Command should only be used with DSS paired data. The US.L command can be used to label the x-axis of DSS time series data.

Parameters:

`None`	Will turn off the user x-axis label and use the default x-axis label which is based on the pathname parts.
`abc...`	Alpha-numeric label of no more than 40 characters.
`'NULL'`	Do not plot a user label.

Example:

```
XL, DAMAGE IN 1000 $
```

YLABEL

Use:

```
YL[.option], [parameters]
```

Description:

The YLABEL command is used to specify a user label that will appear on the Y-Axis of a plot for each of the six possible data types that can be plotted. The location and character size to use with the y-axis label can be specified by using the L option. The label will be centered based on the location specified. Note: Option R is only available on the PC version.

Options:

`L`	Define character size and x-y location of y-axis label.
`R`	Toggle to turn on/off 90 degree rotation of y-axis labels. (only available on PC version).

Parameters:

`None`	Will turn off the user defined y-axis label and use the default y-axis label based on the pathname parts.
`n,m,x,y`	Data type number, character size, x, y. Only valid when used with the L option.
`n,abc...`	Data type number, Alpha-numeric label of no more than 40 characters.
`n,'NULL'`	Data type number, 'NULL' - Do not plot a label.

Example:

```
YL.L 1 3 85 450
YL,1,DAMAGE IN 1000 $
YL,2,FLOW IN CFS
```

YMARKER

Use:

```
YM, [parameters]
```

Description:

The YMARKER command is used to provide Y-Axis value markers for the curve numbers specified by the user. The markers will use the same color as the curve number specified. Up to ten markers can be specified by the user. The user specifies curve number, marker value, character size, X-axis location in screen coordinates, position of label and label. The label associated with the marker is limited to a maximum of 20 characters. Note: x-location: -1 (Left), -2 (Centered), and -3 (Right) A-above marker, B-below marker, C-centered on marker, CLEAR - Clear all markers, OFF - Turn off the markers but do not clear them, ON - Turn on the markers.

Parameters:

`None`	Turn off the markers.
`n,x,m,x,p,abc..`	Curve, value, character size, x-location, position, label.
`ON`	Turn on the markers.
`OFF`	Turn off the markers.
`CLEAR`	Clear all markers and turn off the function.

Example:

```
YM 1 8.23 1 500 A TOP OF BANK
YM,OFF
YM CLEAR
```

YRANGE

Use:

```
YR[.options], [parameters]
```

Description:

The YRANGE command is used to specify the acceptable range that the Y-axis data should be. The program brackets or sets a window based on this range. The User has the option of having a relative or absolute window by placing a ".A" for absolute or ".R" for relative. Up to six y-axis range pairs can be entered.

Options:

`A`	Range values are used to set an absolute window.
`R`	Range values are used to set a relative window.
`S`	Use range values for scale only.
`X`	Specify a range for paired data in the x-axis.

Parameters:

`None`	Turns off the option.
`r,s,r,s...`	Minimum and Maximum range value pairs for each data type.

Example:

`YR,0,10000,0,5`	Sets the Range or Window from 0 through 10000 for the first data type and 0 through 5 for the second data type.

ZOOM

Use:

```
ZO, [parameter]
```

Description:

The ZOOM command is used to specify a window of the data. The data will not be altered by this command, only the plot will be windowed. A rectangular box will appear centered about the maximum y-axis value of the data curves plotted on the first window operation. On following windows, the box will be centered on the plot. The user then uses the numeric keyboard to either move or modify the size of the window. Repeated Z's will enlarge or decrease the window. Enter a W to plot the window defined by the box. It is recommended that the zoom option only be used with 4100 series Tektronic terminals or on the PC version.

Parameters:

None	Will window on first window data type.
n	Specifies which window data type (1-6) to ZOOM in.

The following defines the use of the NUMERIC KEYS to move and/or modify the size of the zoom window.

Valid Keys:

0	Toggle for specifying move or modify (default) size of window.
-	Toggle for enlarging (default) or reducing size of window.
5	Refresh plot; only valid for 4107-9 terminals.
1 - 9	Provide relative movement for moving or modifying size of window.
W	Plot the window.

Appendix B

ACAD Command Document

(for UNIX Version of DSPLAY)

ACAD Command (UNIX version)

USER'S MANUAL

Invoking the ACAD command in the DSPLAY program initiates a simplified Automated Computer Aided Drafting (ACAD) capability for Tektronix 41/4200 series terminals and X Window workstations. This capability is not available on the PC version. The user can overlap ACAD style graphics on top of the standard DSPLAY plots. The ACAD capability has been implemented in the data display side of DSPLAY.

The ACAD capability is based on the Tektronix screen coordinate system of 1023 units in the x axis and 779 units in the y axis. The maximum number of objects that can be used per ACAD overlay is 100 and the maximum number of coordinates used for poly lines or polygons is 400. The program will warn the user if the limits of an overlay have been exceeded. When this happens, the user should close the overlay and start a new overlay.

There is no limit on the number of overlays that can be created since they are stored permanently in an ASCII file. The user specifies the ACAD filename on the execution line with the command [**DSPLAY ACAD=filename**]. If the user does not specify an ACAD file, the program will store the data under a permanent file named "acadfile". The following describes in detail the syntax, drafting options available, and structure of the **ACAD** command.

Command Syntax:

The syntax of the **ACAD** command is similar to DSPLAY commands in that it has options and parameters. The options are part of the command and are delimited by a period. The parameters follow the command or options and are delimited by a comma or a space.

Options: E - Edit an existing ACAD overlay or create a new ACAD overlay.

I - Edit an existing ACAD icon overlay or create a new icon overlay. The icon is a special type of ACAD overlay that can be incorporated within a standard ACAD overlay. The value of icons is that they can be created separately and then scaled and placed at any location on a standard ACAD overlay. The creation of an icon is the same as for a standard ACAD overlay, with the exception that the user specifies a rectangular scaling window for the icon.

Parameters ACAD name list. If the **E** or **I** options are being used, only one name is allowed. The program will only allow you to edit one overlay at a time. Enter a "?" to get a list of the available ACAD overlay names.

Edit Command Syntax:

The syntax used in the edit mode consists of one or two letter combinations. The commands can be grouped into two parts; object commands and edit commands. The object command will instruct the program to create that object. In addition to the above commands, there are two full length commands that allow the user to quit the editing without saving any of the changes (**QUIT**) and the capability to refresh the graphics screen (**REFRESH**). The REFRESH command is useful when the screen gets cluttered with editing garbage left over from scale, copy, and delete commands.

Screen editing is accomplished by the use of the cross hairs to specify screen locations and by program prompted questions. The program has default values for all prompted questions and the default value can be changed by typing a new value or accepting the default value by entering a carriage return. The cross hair normally does not use the character entered as a command, with the exception of the poly line (P) command and the copy command (CO,CW).

Object Commands

B Text block command. This command allows the user to define a rectangular screen area that is used to display text from a file. The limitations are 80 lines by 80 columns. The user provides the name of the text file, starting and ending line numbers of the text, character color, font style, and character size. If the user specifies soft font, the text characters will be sized to fit the text block window.

C Circle command. This command allows the user to create a circle. The cross hairs are used to specify the center and radius of the circle. The program will prompt for color, line thickness[1], and polygon fill option.

R Rectangle command. This command allows the user to create a rectangle. The cross hairs are used to specify opposite corners of the rectangle. The program will prompt for color, line thickness[1], and polygon fill option.

T Triangle command. This command is used to generate an isosceles triangle. The cross hairs are used to specify one of the equal sides of the triangle. The program will prompt for color, line thickness[1], and polygon fill option.

P Poly line command. This command is used to generate a poly line or a polygon. The cross hairs are used to specify the coordinate locations of the poly line. The program will prompt for color, line thickness[1], line style, and fill option when the user uses the **C** cross hair command to terminate coordinate input. The following cross hair key character commands are used to terminate the poly line or polygon.

- **Q** - Use to specify the end of the poly line.
- **D** - Delete the last coordinate entered.
- **C** - Use to close a poly line or polygon. The polygon option is only available if the line style is a solid line.
- **A** - Used to specify the end of a poly line with an arrow head terminator.

L Label command. This command is used to input a label. The label can be input with either soft or hardware fonts. If soft fonts are used, the character size can be scaled and rotated. If hard fonts are used, the character size is limited to two different sizes (4-smallest and 1-largest). The cross hairs are used to specify the location of the label. For hardware fonts, the user can center, right justify, or left justify the label. For soft fonts, the labels are always left justified. The program will prompt for the character color, type of font, size of font, angle (only for soft fonts), and label position (only for hardware fonts).

---

[1] Only applies when using a Tektronix 4510 rasterizer. Sizes are 0 (thinnest) to 3 (thickest).

G Grid command. This command is used to draw a solid or dotted grid. The cross hairs are used to specify the opposite corners of a rectangular region where the grid will be plotted. The program will prompt for the number of x and y grid divisions, color, line thickness, and line style (solid or dotted).

I Icon command. This command allows the user to insert an icon in the standard ACAD overlay. The user is prompted for the scaling option. If scaling is chosen, the cross hair is used to specify a rectangular window in which the icon is forced to fit. If no scaling is done, the cross hair is used to specify the lower left hand corner of the icon rectangular window.

Edit Commands

The edit commands are used to edit existing objects. The edit commands can consist of a second character that can be used to make specific the object that they are to act on. The cross hair is used to pick the object. The cross hair should be placed as close to the center of the object to be picked as possible. In addition, if the second character is a **W**, a rectangular window can be specified by the use of cross hairs to define the objects to be acted on by the edit command. The following is a description of the editing commands.

S Scale command. This command allows a user to pick one object or a group of objects and specify an x and y scale factor to use for scaling. For scaling labels and circles, the x and y scaling factors must be the same.

M Move command. This command allows a user to pick one object or a group of objects and move them to another location. The program will prompt the user to use the cross hairs to pick the object or objects, specify an offset reference point and an offset vector.

CO Copy command. This command allows a user to pick one object or a group of objects and copy them to another location. The program will prompt the user to use the cross hairs to pick the object or objects, specify an offset reference point and an offset vector. To copy the object or objects to multiple locations, the user can enter the C character key for the offset vector and the program will continue to copy the objects to the cross hair locations. To terminate the copying process, use any character key that is not the C character.

D Delete command. This command allows a user to pick one object or a group of objects and delete them.

ACAD File Structure

The following is a description of the ACAD file structure. The ACAD file is in ASCII format and can be edited by the user with an editing program like COED. In addition the user can write other programs that utilize the file structure described below to create graphics that can then be used by the DSPLAY'S ACAD capability.

The ACAD file structure or format is in fixed format. There are two types of overlays; standard and icon. Each overlay is delimited by a tag line that has the word **#TAG [overlay name]**. The icon overlay has [<_>] as its first object definition. The format used and the variables used in both icons and standard overlays are described below.

BLOCK - Format

1st variable - character - A4 - '<B>:' - Object name
2nd variable - integer - 1X,I2 - Color index
3rd variable - integer - 1X,I3 - Character size[2]
4th variable - integer - 1X,I4 - Screen window, x axis minimum
5th variable - integer - 1X,I4 - Screen window, x axis maximum
6th variable - integer - 1X,I4 - Screen window, y axis minimum
7th variable - integer - 1X,I4 - Screen window, y axis maximum
8th variable - integer - 1X,I5 - Starting line number
9th variable - integer - 1X,I5 - Ending line number
10th variable - character - 1X,A1 - 'T' soft font or 'F' hard font
11th variable - character - 1X,A20 - Filename

CIRCLE - Format

1st variable - character - A4 - '<C>:' - Object name
2nd variable - integer - 1X,I2 - Color index
3rd variable - integer - 1X,I1 - Line thickness (0,1,2,3)
4th variable - integer - 1X,I4 - X coordinate of center of circle
5th variable - integer - 1X,I4 - Y coordinate of center of circle
6th variable - integer - 1X,I4 - Radius of circle
7th variable - character - 1X,A1 - 'T' solid fill or 'F' no fill

---

[2] The character size for soft fonts is based on a default size that the user changes by a multiplication factor. The factor is multiplied by 10 and stored as an integer value. For hard fonts the character sizes are 1 (large) and 4 (small).

RECTANGLE - Format

1st variable - character - A4 - '<R>:' - Object name
2nd variable - integer - 1X,I2 - Color index
3rd variable - integer - 1X,I1 - Line thickness (0,1,2,3)
4th variable - integer - 1X,I4 - X minimum coordinate of rectangle
5th variable - integer - 1X,I4 - X maximum coordinate of rectangle
6th variable - integer - 1X,I4 - Y minimum coordinate of rectangle
7th variable - integer - 1X,I4 - Y maximum coordinate of rectangle
8th variable - character - 1X,A1 - 'T' solid fill or 'F' no fill

TRIANGLE - Format

1st variable - character - A4 - '<T>:' - Object name
2nd variable - integer - 1X,I2 - Color index
3rd variable - integer - 1X,I1 - Line thickness (0,1,2,3)
4th variable - integer - 1X,I4 - X minimum coordinate of triangle
5th variable - integer - 1X,I4 - X maximum coordinate of triangle
6th variable - integer - 1X,I4 - Y minimum coordinate of triangle
7th variable - integer - 1X,I4 - Y maximum coordinate of triangle
8th variable - character - 1X,A1 - 'T' solid fill or 'F' no fill

POLYLINE - Format

1st variable - character - A4 - '<P>:' - Object name
2nd variable - integer - 1X,I2 - Color index
3rd variable - integer - 1X,I1 - Line thickness
4th variable - integer - 1X,I5 - Line style (1 solid, 2 dotted ...)
5th variable - integer - 1X,I2 - Number of coordinate pairs on this line
6th variable - character - A1 - 'T' solid fill, 'F' no fill, or '+' poly line continuation
7th - 18th variables - 12(1X,I4) - X and Y coordinate pairs that describe the poly line

LABEL - Format

1st variable - character - A4 - '<L>:' - Object name
2nd variable - integer - 1X,I2 - Color index
3rd variable - integer - 1X,I3 - Character size[2]
4th variable - integer - 1X,I3 - Angle of label (0 - 360)
5th variable - integer - 1X,I4 - X coordinate location of label
6th variable - integer - 1X,I4 - Y coordinate location of label
7th variable - character - A1 - 'T' soft font or 'F' hard font
8th variable - character - A52 - Label

GRID - Format

1st variable - character - A4 - '<G>:' - Object name
2nd variable - integer - 1X,I2 - Color index
3rd variable - integer - 1X,I1 - Line thickness (0,1,2,3)
4th variable - integer - 1X,I4 - X minimum coordinate of rectangle
5th variable - integer - 1X,I4 - X maximum coordinate of rectangle
6th variable - integer - 1X,I4 - Y minimum coordinate of rectangle
7th variable - integer - 1X,I4 - Y maximum coordinate of rectangle
8th variable - integer - 1X,I2 - X number of divisions
9th variable - integer - 1X,I2 - Y number of divisions
10th variable - character - 1X,A1 - 'T' solid or 'F' dotted

ICON - Format for standard overlay

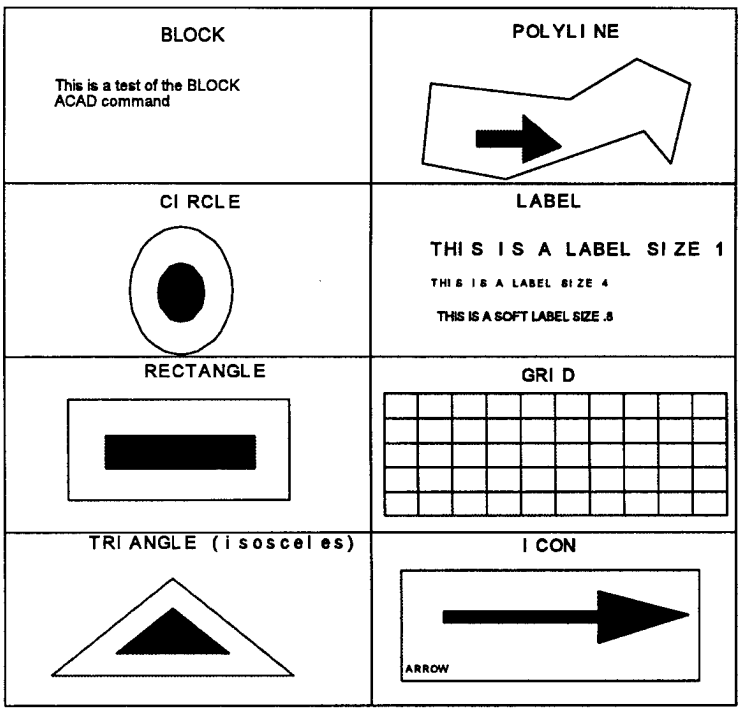
1st variable - character - A4 - '<I>:' - Object name
2th variable - integer - 1X,I4 - Icon scale window, x axis minimum
3th variable - integer - 1X,I4 - Icon scale window, x axis maximum
4th variable - integer - 1X,I4 - Icon scale window, y axis minimum
5th variable - integer - 1X,I4 - Icon scale window, y axis maximum
6th variable - character - A20 - Icon name

ICON - Format for icon overlay
1st variable - character - A4 - '<_>:' - Object name
2th variable - integer - 1X,I4 - Icon window, x axis minimum
3th variable - integer - 1X,I4 - Icon window, x axis maximum
4th variable - integer - 1X,I4 - Icon window, y axis minimum
5th variable - integer - 1X,I4 - Icon window, y axis maximum

EXAMPLE ACAD FILE

The following is an example of an ACAD file.

```
#TAG ARROW
<_>:   38  806   17  572
<P>:  7 0     1  6 T  166  286  537  286  537  196  755  322  537  429  537  339
<P>:  7 0     1  3 +  537  339  166  339  166  286
#TAG EX1
<G>:  1 0   51  858  125  747  2  4 T
<L>:  1   1   0  231  718 F BLOCK
<L>:  1   1   0  222  565 F CIRCLE
<L>:  1   1   0  205  414 F RECTANGLE
<L>:  1   1   0  143  262 F TRIANGLE (isosceles)
<L>:  1   1   0  610  717 F POLYLINE
<L>:  1   1   0  615  566 F LABEL
<L>:  1   1   0  621  410 F GRID
<L>:  1   1   0  621  259 F ICON
<B>:  1   4  106  485  597  712     1    20 F BLOCK
<C>:  2 0  245  495   57 F
<C>:  2 0  245  493   26 T
<R>:  5 0  121  364  309  398 F
<R>:  5 0  162  326  337  366 T
<T>:  8 0  102  235  152  239 F
<T>:  8 0  173  235  172  212 T
<P>:  4 0     1  6 F  511  609  603  595  756  638  786  609  807  680  748  702
<P>:  4 0     1  4 +  748  702  674  666  520  680  511  609
<P>:  5 0     1  6 T  571  624  622  624  622  610  664  624  622  652  622  638
<P>:  5 0     1  3 +  622  638  571  638  571  624
<P>:  1 0     1  6 F  531  709  603  695  622  652  622  666  622  652  611  661
<L>:  1   4   0  521  497 F THIS IS A LABEL SIZE 4
<L>:  1   1   0  520  523 F THIS IS A LABEL SIZE 1
<L>:  1   6   0  520  466 T THIS IS A SOFT LABEL SIZE .8
<G>:  1 0  469  848  295  403 10  5 T
<I>:  490  817  151  245 ARROW
<L>:  1   7   0  492  153 T ARROW
<R>:  1 0  487  817  147  246 F
```



DSSMATH

Hydrologic Engineering Center
Utility Program for Mathematical Manipulation of HEC-DSS Data

User's Manual

Version 2.0
March 1995

Hydrologic Engineering Center
U.S. Army Corps of Engineers
609 Second Street
Davis, California 95616-4687
(916) 756-1104

DSSMATH
Utility Program for
Mathematical Manipulation of HEC-DSS Data
User's Manual

Table of Contents

Chapter		Page
	Preface	
1.	Purpose	1
2.	Description	1
3.	Use	
	3.1 Running the Program	2
	3.2 Use of Commands	2
	3.2.1 Data Management	4
	3.2.1.1 Program Memory	4
	3.2.1.2 Data Retrieval and Storage	5
	3.2.1.3 Memory Management Functions	5
	3.2.2 Computations	6
	3.2.2.1 Arithmetic Computations	6
	3.2.2.2 Functions	7
	3.2.3 Screening and Replacement	9
	3.2.4 Tabulation and Show Commands	9
	3.2.5 Miscellaneous Commands	9
	3.3 Contingency Processing	9

List of Figures

1. Example Input 4

List of Tables

1. Commands 3
2. Compute Functions 7

Appendices

A. DSSMATH Commands A-1
B. Compute Functions B-1
C. Programmer's Guide C-1
D. Compute Functions Examples D-1

PREFACE

This manual provides documentation for Program **DSSMATH.** It provides a detailed description of DSSMATH's capabilities and application.

Program DSSMATH provides capabilities for arithmetic computations and transformations of data stored in the Hydrologic Engineering Center's Data Storage System (HEC-DSS).

This document was prepared by Alfredo E. Montalvo, Technical Assistance Division, HEC. Art Pabst was Chief, Technical Assistance Division and Darryl W. Davis, Director, HEC, during preparation of this report.

DSSMATH

1. Purpose

DSSMATH enables mathematical manipulation of data stored in the Hydrologic Engineering Center's Data Storage System (HEC-DSS). The program provides capabilities for: arithmetic computations, transformations, such as stage to flow; screening; and estimation of missing or erroneous values. The program may be used in an automated batch environment for processing a real-time data stream, or it can be used interactively to perform ad hoc operations.

2. Description

DSSMATH supports regular, irregular-interval HEC-DSS time-series data and HEC-DSS paired function data. The standard memory configuration (UNIX and DOS Lahey Extended) can accommodate 10 time series records of up to 30000 values each, 5 paired-data records each with 30 dependent variables with a maximum of 500 values per record, and 15 scalars. Data may be accessed in several HEC-DSS files at the same time. Besides the DOS Lahey Extended implementation, a DOS version that runs under 640K of conventional memory is available. Its configuration is 5 time-series records consisting of 3180 values each and 15 paired functions consisting of 15 sets of dependent variables, with 300 values per paired function.

Time-series records and look-up tables are referenced in the various processing functions by user-assigned labels. The labels are one to six characters long and are assigned when the data is computed (CO command) or retrieved (GET command) from a HEC-DSS file. The HEC-DSS file name and data pathname are specified as parameters to the GET command (See Figure 1).

In either an interactive or in a batch setting, DSSMATH proceeds according to a sequence of instructions provided by the user. A typical sequence of instructions consists of a retrieval of data from the HEC-DSS file, a series of computations involving the data, and storage of results in a HEC-DSS file.

Computations in DSSMATH consist of arithmetic manipulations of time series data, such as multiplying a time series by a constant or another time series, and special functions that provide more complex and specific capabilities, such as rating table transformations.

Elementary screening functions flag data values internally to facilitate subsequent corrections. Replacements for flagged and missing values are provided by simple, linear estimation functions. Internal flags are used to assure that missing data do not erroneously affect computations. However, internal flags can not be read or written to HEC-DSS at present, and are not compatible with HEC-DSS Version 6 flag scheme.

The features of PREAD are provided in DSSMATH. PREAD facilitates efficient repetition of routine procedures. For example, a complex series of computations can be stored in a PREAD macro and started, for example, as a selection from a screen menu. (Ref: PREAD, Functions, Macros, Menus and Screens User Information)

3. Use

3.1 Running the Program

DSSMATH is initiated with: **DSSMATH [parameter] ...**

"parameter" has the form "key word=xxx" and may be one or more of the following specifications:

Key word	Default	Description
INPUT	terminal	Terminal or batch job input file
OUTPUT	terminal	Terminal or batch job output file
FUNFILE	mathfun	PREAD function file
MACFILE	mathmac	PREAD macro file
SCNFILE	genscn	PREAD screen definition file
TABFILE	tabfile.out	PREAD screen definition file

In the interactive environment, the default output is the user's terminal display. In the batch environment, the default output is the job output file.

3.2 Use of Commands

The processing of time-series data with DSSMATH, either interactively or through a batch input file, is directed with commands. Commands are specified with two-character mnemonics in columns 1-2 of the input stream. The generalized form of the commands is: **XX[.options] [parameter 1] [parameter 2] ....**

Commands typically require parameters. Parameters supply specific information about the objects or entities to be processed or specifications to control the process.

Options are single characters used to specify alternative ways of processing.

Input lines beginning with two asterisks (**) are used for notes or comments and are simply echoed in the output.

A list of commands is presented in Table 1. The commands are described in detail in Appendix **A**.

Table 1 Commands

Command	Description
**	Comment.
CATALOG	Display a catalog of a DSS file.
CLEAR	Remove a variable data label from memory.
COMPUTE	Perform a computation.
DIAG	Toggles DSS diagnostic trace output on and off.
DPATH	Display a selective catalog of a DSS file.
FINISH	Terminate and exit the program.
GET	Retrieve data from a DSS file.
HELP	List commands.
OPEN	Open a DSS file.
PUT	Store data in a DSS file.
SD	Set data descriptions.
SHOW	Displays selected internal information about the data variable denoted by specified variable label.
SMISSING	Set missing value indicators.
SP	Set data pathname.
STATUS	Display key program variables states or values.
TABULATE	Tabulate values of time-series or paired data.
TIME	Set a time-window for time series data.
$CO	Resume processing subsequent to error.
$AB	Abort (stop) processing subsequent to error.

```
** Set a time window
TI T-2D T
**
** Get the "raw" data
GE STG=RAWDB:/SCIOTO/HIGH3/STAGE/01JUL1986/1HOUR/OBS/
**
** Get a look-up table for the station
GE TB=TABLEDB:/SCIOTO/HIGH3/STAGE-FLOW///USGS TABLE 6/
**
** Compute flows using the lookup table
COM FLW=RTABLE(STG,TB)
**
** Save the result
PUT FLW=MASTDB:/SCIOTO/HIGH3/FLOW/01JUL1986/1HOUR/COMP/
**
** Contingency checkpoint
$CONTINUE
**
** Clear program memories
CL  ALL
.
.
.
** Terminate
FI
```

Figure 1 Example Input

A typical command sequence retrieves a time series record, computes a dependent time series, and stores the result. An example input sequence that computes flow values from stage values is shown in Figure 1.

3.2.1 Data Management

3.2.1.1 Program Memory. The standard memory configuration can accommodate 10 time series records of up to 30000 values each, 5 paired-data records each with 30 dependent variables with a maximum of 500 values per record, and 15 scalars. The DOS version configuration is limited to 5 time-series records of 3180 values each, 15 paired-data records each with 15 dependent variables with 300 values per record, and 15 scalars.

Time series records may be regular or irregular and all data values carry a time and quality flag. The quality flags indicate four levels: 1) N for no flag; 2) E for estimated; 3) Q for questioned; and 4) M for undefined or missing. Time series records may be retrieved from files or may be computed in DSSMATH. However, currently, the data quality flags used in DSSMATH are not the standard HEC-DSS quality flags and may not be read or stored to DSS.

Paired data records include rating tables (ie., stage-flow) and polynomial coefficients. Paired-data records are usually retrieved from files, although at the present time only two compute functions," MATE and MRGP" manipulate a paired-data record.

Scalars are variables that are results of assignments or computations. Storage for scalars is provided in order to use the scalar result of a computation in a subsequent computation.

Each record or variable in the program memory is assigned an alphanumeric label as it is retrieved or computed. The label is then used to refer to that time-series in further processing. The label may be one to six characters long. It must begin with an alphabetic character but may contain numeric characters. A label should not be the same as any of the commands or function names.

The most recently used time window, pathname, and DSS file name are kept in the program's memory. A maximum of five DSS files can be kept open on a continuous basis. Once a sixth DSS file is requested, the first DSS file opened is automatically closed and the new DSS file requested is opened. The number of DSS files that can be open on a continuous basis is limited to three for the 640K DOS version.

3.2.1.2 Data Retrieval and Storage. The commands "GE" (get) and "PU" (put) are used, respectively, to retrieve and store time series and paired function data. The location of the data in DSS files is determined by both a file name and a pathname (See Figure 1). Data may be retrieved and stored in different files as desired. A warning message is printed if the retrieval exceeds any storage limits. Options are provided with the command "PU" to control overwriting of previously stored data.

The time window command "TI" controls the span of the data retrieved with "GE" and, therefore, defines the period of time for processing. The time window must be defined before any processing can be accomplished. The time window should be defined carefully, since arithmetic computations and several of the compute functions check for concurrence of the data.

When regular-interval time series data are retrieved, the time window is temporarily adjusted, if necessary, to include data belonging to exact interval boundaries. Options with the retrieval command "GE" provide retrievals outside the range of the time window to facilitate use of irregular-interval time series data.

3.2.1.3 Memory Management Functions. The status command ("ST") may be used to show memory contents, including all variable label names and types (time series, paired function, or scalar), the current time window, and the most recently used DSS pathname and DSS filename. "ST" options will also show the pathnames, units and types of time series and paired-function variables and values assigned to scalars. The command "CL" releases memory either globally or selectively, by label.

3.2.2 Computations. The COMPUTE command generates values of a dependent time series, paired function, or scalar from a simple arithmetic operation or a more complex function of one or more time series, paired functions, or scalars. If the dependent time series or paired function variable is not in memory it is created with undefined pathname, type, and units. However, the type and units may be assigned by the function being used.

The computation of a dependent time series may be subjected to a test based on a logical IF condition that compares values of two variables or a variable and a constant. The variables may be concurrent time series or scalars. If the IF condition is not satisfied for a particular time, then the corresponding value of the dependent variable is unchanged. If the value was not previously defined, it will remain undefined.

It is left to the user to provide appropriate parameters in computations: for example, the program will allow temperature values to be added to flows. However, arithmetic computations do require concurrent time series variables, and DSS data types or units are checked in certain compute functions. Usually, data types and units of a result must be specified explicitly as a separate processing step with the "SD" command.

3.2.2.1 Arithmetic Computations. An arithmetic computation of a time-series record consists of addition (+), subtraction (-), multiplication (*), division (/), or exponentiation (**) by a constant, previously defined scalar or time-series. The following are examples:

COM Y=X+2 add 2 to time series "X"
COM Y=X/A "A" may be a scalar or time-series

One of the independent variables in an arithmetic computation may be redefined (ie., the dependent variable may be the same as one of the independent variables). However, if a scalar and a time series variable are the independent variables, the dependent variable is a time series.

When arithmetic computations involve two independent time-series, the dependent variable will consist of the arithmetic combination of the concurrent successive values of the two series. "Concurrent successive values" means equal numbers of values with the same times. The "TS1" or "TS4" compute functions are useful for aligning two parallel time series vectors in time. The "TS1" function generates a regular-interval time series by interpolation at regular intervals. The "TS4" function generates a time series interpolated to the times of a pattern time series.

Two time series involved in an arithmetic computation may have different units and types. Units and types of results must be explicitly defined by the user with the "SD" command.

3.2.2.2 Functions. Computations may involve functions that operate upon one or more time-series records and result in a scalar or new time-series record. The functions are summarized below.

Table 2 Compute Functions

Name	Description
ABS	Absolute function.
ACC	Running accumulation.
AMREG	Apply multiple linear regression equation.
CMA	Centered moving average smoothing.
CONIC	Conic interpolation based on elevation area table
CORR	Compute correlation coefficients.
COS	Cosine trigonometric function.
COUNT	Count the number of valid and missing data.
DDT	Differences per unit time.
DECPAR	Decaying basin wetness parameter.
DIFF	Successive differences.
ESTLIN	Estimate values for missing data.
ESTPPT	Estimate values for missing precipitation.
FMA	Forward moving average.
GENTSR	Generate a regular interval time series.
INT	Truncate to whole numbers.
LAST	Last valid value in a time series.
LOG	Natural log base "e".
LOG10	Log base 10.
MATE	Generate data pairs from two time series variables.
MAX	Maximum value in a time series.
MEAN	Mean value in a time series.
MIN	Minimum value in a time series.
MREG	Multiple linear regression coefficient function.
MRG	Merge two time series.
MRGP	Merge two paired data series.
MUSK	Muskingum routing function.
NINT	Round to nearest whole number.
OLY	Olympic smoothing.
PERCON	Period constants.

Table 2 (continued)

Name	Description
POLY	Polynomial transformation.
POLY2	Polynomial transformation with integral.
PULS	Modified Puls or Working R&D routing function.
QAC	Flow accumulator gage processor.
RND	Round off.
RTABLE	Rating table interpolation.
RTABLR	Reverse rating table interpolation.
RTABL2	Two variable rating table interpolation.
SCRN1	Screen for possible erroneous values based on max and min range.
SCRN2	Screen for erroneous data values based on a forward moving average of max value.
SDEV	Compute the standard deviation of one independent variable.
SELECT	Extract time series data at unique time specification.
SHIFT	Shift adjustment.
SIN	Sine trigonometric function.
SKEW	Compute the skew coefficient of one independent variable.
SQRT	Square root function.
SS	Straddle Stagger routing function.
SSW	Willmington District Straddle Stagger routing function.
TAN	Tangent trigonometric function.
TS1	Interpolate data at regular time intervals.
TS2	Period averages at regular intervals.
TS3	Period minimums and maximums at regular intervals.
TS4	Interpolate data at irregular intervals.
TSCYCL	Time Series Cyclic Analysis.
TSHIFT	Shift time series in time.
TSNAP	Snap Irregular times to nearest Regular period
TTSR	Transform time series to regular.
TTSI	Transform time series to irregular.
1/X	Inverse function.

A detailed description and examples can be found in appendices B and D respectively.

In addition to the functions summarized above, DSSMATH allows the user who is proficient in FORTRAN, to create additional compute functions. See Appendix **C** for instructions on creating additional functions.

A detailed set of examples for each of the functions is available in Appendix D. Most of the examples are based on real applications developed at Corps of Engineers District offices.

Some computations result in a scalar value. Scalar values may also be assigned labels and used in subsequent processing, but cannot be retrieved or stored in DSS files.

3.2.3 Screening and Replacement

The screening and replacement capabilities of DSSMATH are implemented in the compute functions "SCRN1", "SCRN2", "ESTLIN", and "ESTPPT". Data values that exceed the specified limits are flagged internally to facilitate subsequent corrections with estimation functions or graphical editing. **Note**: Program DATCHK is also capable in performing this type of screening and is substantially more powerful than DSSMATH.

3.2.4 Tabulation and Show commands

Tabulation capabilities of DSSMATH are nearly identical with those of the program DSPLAY. Time series and paired function data may be tabulated with the "TA" command. The "SHOW" command can also be used to tabulate the variables and has the additional capability of displaying scalar variables.

3.2.5 Miscellaneous Commands

The DSS file catalog command "CA" is used for listing the pathnames of records in a DSS file. Data variable pathnames and descriptions (eg., units) can be specified using the "SP" and "SD" commands, respectively. On-line help is available through the "HE" command. The finish command, "FI", terminates processing.

3.3 Contingency Processing

Certain error conditions can render subsequent processing inappropriate. For example, if data cannot be retrieved, it cannot be processed. These errors, which simply result in warnings in the interactive environment, may be processed with recovery procedures when input are taken from an input file or a PREAD macro by specifying recovery checkpoints in the input.

An input line beginning with a dollar sign ($) is an error recovery checkpoint. "$CONTINUE" indicates where to resume processing when an error is encountered. "$ABORT" indicates the program is to stop if an error has been encountered and no "$CONTINUE" was found. If no recovery checkpoints are found, processing stops.

Appendix A

DSSMATH Commands

Appendix A - DSSMATH Commands

Command Syntax

Brackets ([ ]) and parentheses (...) symbols are notation used for describing commands and are not part of the actual command syntax.

UPPERCASE	Items in uppercase are required key words as shown.
lowercase	Items in lowercase are to be supplied by the user.
[ ]	Items within brackets are optional.
...	Items immediately preceding parentheses may be repeated.

Commas and blanks are used to separate items and are interchangeable except as otherwise noted.

Periods are required when options are specified. No blanks should appear between command and period or between period and options. The command and options are case insensitive, but the parameters are generally case sensitive.

Index of Commands

		Page
**	Comment	2
CATALOG	Display a catalog of a DSS file	2
CLEAR	Remove a variable data label from memory	2
COMPUTE	Perform a computation	3
DECIMAL	Set the number of decimal places to use in tabular output	3
DIAG	Diagnostic DSS trace	4
DPATH	Display a selective catalog of a DSS file	4
FINISH	Terminate and exit the program	4
GET	Retrieve data from a DSS file	4
HELP	Get help on commands and functions	5
OPEN	Open DSS file	5
PUT	Store data in a DSS file	6
SD	Set data descriptions	6
SHOW	Display internal data variable information	7
SMISSING	Set missing value indicators	7
SP	Set data pathname	8
STATUS	Display key program variables states or values	8
TABULATE	Tabulate values of time-series or paired data	8
TIME	Set a time-window for time-series data	9
$CO	Resume processing subsequent to error	9
$AB	Abort (stop) processing subsequent to error	9

Name: Comment
Use: **[parameters] ...

Description: May be used to annotate input command streams.

Parameters:
text - Any text up to 130 characters per line.

Example: ** Set the time window

Name: CATALOG **Display a catalog of a DSS file**
Use: CA [.options] [parameters]

Description: Catalog (list) the records (pathnames) in a DSS file. The catalog is listed at the terminal one screen-full at a time. The numbers associated with each record may be used in other commands to refer to the record in lieu of a pathname.

Options:

None	(list old catalog in full mode)
N	(create new catalog)
A	(create abbreviated catalog)

Parameters:

None	Catalog last opened or referenced DSS filename.
filename:	Is the name of a DSS file, followed by a colon.

Example: `CA.NA MASDSS:`

Name: CLEAR **Remove a variable data label from memory**
Use: CL [parameters]

Description: Clear releases memory slots. If the parameter 'ALL' is present, all slots are released. If any labels are specified, then only the memory slots represented by the labels are released.

Parameters:

None	No action is taken.
ALL	All variable labels are cleared and all data is initialized.
label...	Will clear specified variable labels.

Example: `CL FLOW1 FLOW2 STOR1`

Name: COMPUTE **Perform a computation**
Use: CO [parameters]

Description: DSS data header information are checked in some computations. If any of the independent variable values are flagged as missing then the computed value is missing. Data types and units of the dependent variable are generally undefined, but may be set by some functions.

Parameters: [IF(x1 operator x2)] result=expression

"result" is a label for the resultant of the computation. The result may be a currently defined variable and may be used as an independent variable in subsequent computations.

"operator" is one of the following relationships:

LT less than	LE less than or equal
EQ* equal	NE not equal
GT greater than	GE greater than or equal

* Note: No precision tolerance is used for the "EQ" operator.

"x1" and "x2" are time series, scalars or constants. Time series variables must be concurrent with the operand.

"expression" is either a simple arithmetic operation or a function. It may reference one or more labels for variables which have been previously computed or retrieved.

Example:
```
CO IF(FLOW LT 0.0) FLOW=0.0
CO IF(FLOW GT 0.0) FLOW=FLOW**.5
```

Name: DECIMAL **Set the number of decimal places to use in tabular output**
Use: DE,[parameters]

Description: Used to set the number of decimal places to print for each variable lable that is being tabulated. The tabulation fields are a fixed width of 13 characters. Therefore, the user should take care that the maximum values to be tabulated will fit within the fixed width. This command is only valid for time-series data.

Parameters:

None	Will set all decimal places to 3.
n,m,...	Integer values 0 through 10.

Example:
```
DE 1 1 4
```

Name: DIAG **Diagnostic DSS trace**
Use: DI [parameters]

Description: Toggles diagnostic trace output on and off. Caution, this command can generate extensive output.

Parameters:

None	Diagnostic trace is off.
ON	Turn on diagnostic trace.
OFF	Turn off diagnostic trace.

Name: DPATH **Display a selective catalog of a DSS file**
Use: DP [.options] [parameters]

Description: Displays pathnames, tags and reference numbers from the catalog file (see CATALOG command). The DP command has the selective display capability which provides the option of selecting and listing only certain pathnames based on matching pathname parts.

Options:

None	(uses last referenced DSS file catalog)
N	(create new catalog)
A	(create abbreviated catalog)

Parameters:

None	Displays all pathnames.
filename:	Is the name of a DSS file, followed by a colon.
A=..B=..etc	Selective catalog based on pathname parts.

Example: `DP MASDSS: B=RED CREEK C=FLOW`

Name: FINISH **Terminate and exit the program**
Use: FI

Name: GET **Retrieve data from a DSS file**
Use: GE [.options] [parameters]

Description: Retrieves data from a DSS data file. The data may be time-series, or paired-function data (e.g. rating tables or polynomial coefficients).

Options:

P	Includes the value just prior to the time window.
N	Includes the next value after the end of the time window.

Parameters: label=[filename:]pathname

"label" is an alphanumeric identifier for the data.

"filename:" is the DSS file name to use and it will be automatically opened if necessary. If omitted, the previously opened DSS file is used.

"pathname" may be an explicit pathname, tag or pathname catalog number, or the previously specified pathname may be modified by specifying replacement pathname parts. If, for example, the previously defined pathname were: `/SCIOTO/CISG3/FLOW/01FEB1986/1HOUR/OBS/` and the following were specified: B=HIGH3 then the new specification would become: `/SCIOTO/HIGH3/FLOW/01FEB1986/1HOUR/OBS/`

Example: `GET FLOW=mastdb.dss:/SCIOTO/CHIANO/FLOW/01JAN1984/1HOUR/OBS/`

Name: HELP **Get help on commands and functions**

Use: HE [parameters]

Description: Displays on-line documentation for a command or function. If no parameter is given, a list of commands is displayed. In order to get a list of the functions, enter "HE FUNCTION".

Parameters:

"name" Command or function name for which a detailed description is needed.

FUNCTION Will display a listing of the available DSSMATH functions.

Example:
```
HELP  COMPUTE
HELP  TTSR
```

Name: OPEN **Open a DSS file**

Use: OP [parameters]

Description: The OPEN command is used to open a new DSS file. A maximum of five DSS files can be kept open on a continuous basis. The GET and PUT commands can also be used to open DSS files.

Parameters:

"name" The complete file name and its path. A maximum of 64 characters is allowed on the DOS version and 80 characters on the UNIX/"DOS Lahey" versions.

Example: `OPEN d:\data\mastdb.dss`

Name: PUT **Store data in a DSS file**
Use: PU [.options] [parameters]

Description: The PUT command is used to take the data in a variable label and store it in a DSS file. If no parameters are specified, the data will be stored in the variables pathname, if one has been given, or in the last pathname defined.

Options:

A All data in the time-series identified by label is written out to the DSS file "filename" and DSS record pathname, replacing any existing data.

R Replaces existing data but it will not write out new DSS records consisting entirely of missing values.

M Same as option **R**, except that existing data will not be replaced by a missing value.

Default for regular time series is to replace only missing values; existing non-missing values will not be replaced. Default for irregular time series is to replace data occurring at the same time and insert data at new times.

Parameters: label=[filename:]pathname

"label" identifies the time series to be written

"filename:" is the name of a DSS file to receive the data. If it is not specified, then the last previously referenced "filename" will be used. The DSS file will be automatically opened, if necessary.

"pathname" is a definition of the pathname to use. The full pathname may be explicitly defined, or pathname parts may be used to modify the last previously defined pathname or the pathname associated with the label, if defined.

Example: `PUT.A FLOW=/usr2/data/mastdb.dss:F=COMPUTED`

Name: SD **Set data descriptions**
Use: SD [parameters]

Description: Data descriptions are data items contained in DSS time series and paired function headers.

Parameters: label,parameter

"label" is the DSSMATH label for data.

"parameter" indicate which items to change and what the new values are. "parameter" has the form "item"="value", where "item" is TYPE or UNITS for time series data and TYPE, UNITS or LABELS for paired function data. The designation for "item" may be abbreviated, such as U= for UNITS= . "value" is an appropriate entry for the particular "item". For paired function data, enter one "value" for each curve in a series separated by commas. For example: U=FEET,CFS.

Example: `SD  FLOW UNITS=CFS  TYPE=PER-AVER`

Name: SHOW Display internal data variable information
Use: SH [parameters]

Description: Displays selected internal information about the data variable denoted as a parameter to the SHOW command. Its main use is for debugging. However, this command is the only way to see the value of a scalar variable. For time-series it shows the data's array position, time (internal representation), time (external representation), value, and data quality flag. For paired data, it shows the DSS header information.

Parameters: label,parameter

None	No action is taken.
label	Label is the name of the data variable to display.
SCALARS	Display all scalar variables and their values.

Example: `SHOW  FLOW`

Name: SMISSING Set missing value indicators
Use: SM [parameters]

Description: Missing value indicators are used to define the numeric values with which missing data are indicated. Up to 10 numbers can be specified. If no parameters are specified, default values -901.0 and -902.0 are used.

Caution: DSS will only accept the default values of -901.0 and -902.0 as valid, therefore be careful about writing data back to your DSS file when the missing value indicators have been changed from the default settings.

Parameters: "parameter" All valid numeric values.

Example: `SM -99999. 999999. -901. -902`

Name: SP **Set data pathname**
Use: SP [parameters]

Description: The SP command is used to define or set the default pathname for a variable label name.

Parameters: label,parameter

"label" is the DSSMATH label for data.

"parameter" consists of either a pathname or a pathname part. If "parameter" is a part, it has the form "part"="value", where "part" is A, B, C, D, E, or F, and "value" is appropriate. If parts are specified, the resulting pathname for the data consists of the current default pathname modified by replacement of the specified parts.

Example: `SP FLOW A=SCIOTO B=COSCO C=FLOW D=01JAN1984 E=1HOUR F=CALCULATED`

Name: STATUS **Display key program variables status or values**
Use: ST [parameters]

Description: The status command may be used to check on the status of all the variables defined and their data descriptions. It can also be used to show the status of specific variables by specifying them as parameters. .

Parameters:

"label"... Show status information for each label specified.

ALL Show status information on all labels that exist.

Example: `STAT FLOW,STOR`

Name: TABULATE **Tabulate values of time-series or paired data**
Use: TA [.options] [parameters]

Description: Tabulates the data represented by the variable label. Up to seven labels may be tabulated. Both time series or paired function data may be tabulated.

Options:

F Send the output to the file specified on the execution line by the use of the parameter **TAB=**filename.

Parameters:

"label" Name or label used for the data variable.

Example: `TAB FLOW STOR ELEV`

Name: TIME **Set a time-window for time-series data**
Use: TI [parameters]

Description: Starting and ending times and dates may be expressed in a variety of ways, including implicit and relative times and dates:

04MAR1983 0700	complete, explicit expression for starting and ending time.
D 0700	current date, time explicit
D	current date, time implicit (2400)
T	current date and time
T-2H	two hours ago
T-3D	three days ago
T-15M	15 minutes ago
D-30D	30 days ago, 2400 hrs
-D +D	expands the existing time window one day at each end

Example: `TIME T-30H T`

Name: $CO **Resume processing subsequent to error**
Use: $CO

Description: Recognized only in a batch job or when a PREAD macro is being used. Used to designate a point in the input command stream to resume processing when a preceding error has caused processing to terminate.

Name: $AB **Abort (stop) processing subsequent to error**
Use: $AB

Description: Recognized only in a batch job or when a PREAD macro is being used. Used to specify termination of the job if an error is encountered while processing according to a batch input stream.

Appendix B

Compute Functions

Appendix B - Compute Functions

Index of Functions

		Page
ABS	Absolute function	3
ACC	Running accumulation	3
AMREG	Apply multiple linear regression equation.	3
CONIC	Conic interpolation from elevation/area table	4
CORR	Compute correlation coefficients	4
COS	Cosine trigonometric function	5
COUNT	Count the number of valid and missing data values	5
CMA	Centered moving average smoothing	5
DDT	Differences per unit time	6
DECPAR	Decaying basin wetness parameter	6
DIFF	Successive differences	7
ESTLIN	Estimate values for missing data	7
ESTPPT	Estimate values for missing precipitation data	7
FMA	Forward moving average	8
GENTSR	Generate a regular interval time series	8
INT	Truncate to whole numbers	8
LAST	Last valid value in a time series	8
LOG	Natural log base "e"	9
LOG10	Log base 10	9
MATE	Generate data pairs from two time-series	9
MAX	Maximum value in a time series	9
MEAN	Mean value in a time series	10
MIN	Minimum value in a time series	10
MREG	Multiple linear regression function	10
MRG	Merge two time series	11
MRGP	Merge two paired data series	11
MUSK	Muskingum hydrologic routing	11
NINT	Round to nearest whole number	11
OLY	Olympic smoothing	12
PERCON	Period constants	13
POLY	Polynomial transformation	13
POLY2	Polynomial transformation with integral	13
PULS	Modified Puls or Working R&D routing function	14
QAC	Flow accumulator gage processor	14
RND	Round off	15
RTABLE	Rating table interpolation	15
RTABLR	Reverse rating table interpolation	15
RTABL2	Two-variable rating table interpolation	16

SCRN1 Screen for possible erroneous values based on maximum/minimum range 16
SCRN2 Screen for possible erroneous values based on forward moving average maximum . . 17
SDEV Compute the standard deviation of one independent variable. 17
SELECT Extract time series data at unique time specification. 17
SHIFT Shift adjustment 19
SIN Sine trigonometric function 19
SKEW Compute the skew coefficient of one independent variable 19
SQRT Square root function 19
SS Straddle Stagger routing function 20
SSW Wilmington District Straddle Stagger routing function 20
TAN Tangent trigonometric function 20
TS1 Interpolate data at regular intervals 21
TS2 Period averages at regular intervals 21
TS3 Period mins or maxs at regular intervals 21
TS4 Interpolate data at irregular intervals 22
TSCYCL Time series cyclic analysis 22
TSHIFT Shift time series in time 23
TSNAP Snap Irregular times to nearest Regular period 23
TTSR Transform time series to regular 23
TTSI Transform time series to irregular 24
1/X Inverse function 24

B - 2 DSSMATH Functions

Name: **ABS** **Absolute function**

Use: CO TY=ABS(TX)

Description: Compute the absolute value of TX and store in TY. TX and TY can be the same variable. TX can be scalar, time series or paired data. If a value of TX is undefined resulting values of TY are undefined. If a concurrent IF condition is not satisfied, TY is unchanged. If TY is a time series, its units and type are set equal to the units and type of TX.

Name: **ACC** **Running accumulation**

Use: CO TY=ACC(TX)

Description: Compute a running accumulation of the values in TX and store the result in TY. TX and TY can be the same variable. If a TX value is undefined or a concurrent IF condition is not satisfied, the value of TX is not added to the accumulation and the corresponding TY remains the same as the previous TY. Units of TY are the same as those of TX. If TX is typed as PER-AVER or PER-CUM, TY will be typed INST-VAL or INST-CUM, respectively.

Name: **AMREG** **Apply multiple linear regression equation.**

Use: CO TY=AMREG(PF,TX1,TX2,...,TXn,min,max)

Description: This function is used to calculate a regular time series (TY) based on a linear regression equation of the general form (Y = B0 + B1*X1 + B2*X2 + Bn*Xn). Where Y is the dependent variable and the X1, X2, and Xn are independent variables. The AMREG function requires that the function parameters TY, TX1, TX2, and TXn be regular time series variables and that they all have the same time interval. It is further required that the time variables be retrieved in the same sequential order as specified in the function parameter list. In other words TX1 is retrieved from DSS first followed by TX2 and so on. The number of independent time series variables that can be specified is limited to one less than the maximum number of time series variables that the program allows. On the DOS PC this value will be 4 independent variables. The linear regression coefficients (B0 - Bn) are stored in a paired function variable (PF) which is normally retrieved from DSS. Two optional function parameters "min max" are available to specify a range of values that control which values in the dependent variable are to be accepted as valid computed values. These two parameters must be specified as the last two parameters and if specified, both must be specified as real numbers. Units of TY are the same as those of TX1. An IF condition has no effect.

Name: **CONIC** **Conic interpolation from elevation/area table**

Use: CO TY=CONIC(TX,TB,input,output,[scale])

Description: Interpolate values for TX using conic interpolation table TB and store the result in TY. TX and TY variables must be regular or irregular time series data. TB is a paired data variable that can be created using program DSSPD. The first paired data values contain the initial conic depth in feet or meters "x(1)" and storage in acre-feet or cubic meters below the first elevation "y(1)". The rest of the paired values "x(2--),y(2--)" contain the elevation in ft-msl or m-msl and area in acres or sq. meters. If the initial conic depth is undefined, the function will calculate one. The third and fourth parameters are used to specify the type of input (TX) and output (TY) data desired. The input type has to be either STORAGE or ELEVATION. The output type has to be either STORAGE, ELEVATION, or AREA. An optional fifth parameter can be used to specify a scale value to use with the input and output storage values. The default scale value is 1.0. An IF condition has no effect. The units and type must be set by the use of the "SD" command.

Name: **CORR** **Compute correlation coefficients**

Use: CO SY=CORR(TX1,TX2,INDEX)

Description: Compute the number of valid pairs for correlation, regression constant, regression coefficient, determination coefficients, standard errors of regression, determination coefficient adjusted for degrees of freedom, and standard error adjusted for degrees of freedom between the two variables TX1 and TX2. TX1 and TX2 must be of the same type (either uniform time series or irregular time series) and contain the same number or rows or ordinates. An IF condition has no effect. Variable INDEX is used to set the scalar variable SY as follows:

Index	Description
1	Number of valid pairs for correlation.
2	Regression constant.
3	Regression coefficient.
4	Determination coefficient.
5	Standard error of regression.
6	Determination coefficient adjusted for degrees of freedom.
7	Standard error adjusted for degrees of freedom.

Name: **COS** **Cosine trigonometric function**

Use: CO TY=COS(TX)

Description: Compute the cosine of TX in radian and store in TY. TX and TY can be the same variable. TX can be scalar, time series or paired data. If a value of TX is undefined resulting values of TY are undefined. If a concurrent IF condition is not satisfied, TY is unchanged. The units and type of TY are set equal to the units and type of TX.

Name: **COUNT** **Count the number of valid and missing data values**

Use: CO SY=COUNT(TX,TYPE)

Description: This function is used to count the valid and missing values in time series variable TX and stores it in scalar variable SY. Possible values for TYPE are: "VALID" and "MISSING". If a concurrent IF condition is not satisfied, then the number of valid values is not incremented and the value is assumed to be missing and the missing counter incremented.

Name: **CMA** **Centered moving average smoothing**

Use: CO TY=CMA(TX,NP[,CLEVEL])

Description: Compute a centered, moving average of NP values in TX and store in TY. NP must be odd and greater than 2. The default CLEVEL is "LEVEL3" and NP/2 values at the beginning and end of TY will be undefined. If a value in TX is undefined or a concurrent IF condition is not satisfied, then the resulting TY values are the averages of one less value. TX must be a regular-interval time-series and TY and TX cannot be the same variable. Useful for filtering instantaneous data containing high-frequency variations. Units and type are set equal to TX values. The following levels are available:

LEVEL1 Only valid values will be averaged. Valid values that do not have all valid values within the averaging period number will be missing. Values at the start and end of the data that do not have enough valid values to average will be assigned missing values.

LEVEL2 Only valid values will be averaged. Valid values that do not have all valid values within the averaging period number will be missing. Valid values at the start and end of the data that do not have enough valid values to average will be averaged over a reduced number of values.

LEVEL3 (Default setting) All values will be averaged based on valid values within the

averaging period number. If all values are missing for the period number, the value will be missing. Values at the start and end of the data that do not have enough valid values to average will be assigned missing values.

LEVEL4 All values will be averaged based on valid values within the averaging period number. If all values are missing for the period number, the value will be missing. Values at the start and end of the data that do not have valid values to average will be averaged based on a reduced number of values.

Note: Questionable and estimated flagged values are used in the computations.

Name: **DDT** **Differences per unit time**

Use: CO TY=DDT(TX)

Description: Computes the successive differences in TX per time period:

TY(t)=(TX(t)-TX(t-1))/DT

where DT is the time difference in days between t and t-1. If a value of TX is undefined, TY is undefined. If a concurrent IF condition is not satisfied, TY is unchanged. TY and TX cannot be the same variable. TY is assigned the type 'PER-AVER.' The units of TY are undefined and should be set by the user with the SD command. An example of the use of DDT is the computation of reservoir inflow from outflow and the change in storage, where the change in storage is transformed to average flow volume per day by the function. The conversion factor for storage to flow is 0.50416 when the storage change is in ac-ft day and 1.1574×10^{-5} when it is in m^3 day.

Name: **DECPAR** **Decaying basin wetness parameter**

Use: CO TY=DECPAR(TY,TZ,R)

Description: Compute a time-series of parameters TY as a function of TZ and R:

TY(t)=R*TY(t-1)+TZ(t)

where R is the decay rate and TZ is precipitation. R is less than 1. The function extends TY: the first value in the series TY is used as the starting value, and any other TY are computed in the sequence. If the first value in TY is undefined, it is assumed to be zero. The first TZ value is ignored. TY and TZ must be regular-interval time series with identical times. R should be appropriate for the interval. The type and units of TY are unchanged. An IF condition has no effect.

Name: **DIFF** **Successive differences**

Use: CO TY=DIFF(TX)

Description: Compute the successive differences of TX and store in TY. TX and TY cannot be the same variable. TX must be of type INST-VAL or INST-CUM. If a value of TX is undefined, resulting values of TY are undefined. If a concurrent IF condition is not satisfied, TY is unchanged. The type and units of TY are undefined and should be set using the SD command.

Name: **ESTLIN** **Estimate values for missing data**

Use: CO TY=ESTLIN(TX,NMAX,QFLAG)

Description: Linearly interpolate estimates for values in TX with quality flags equal to or lesser in quality to QFLAG and place the results in TY. Do not estimate more that NMAX continuous missing values. Possible flags, in order of decreasing quality, are: Q = questionable and M = missing. An IF condition has no effect. TX must be a time-series, and TX and TY may be the same variable. Type and units of TY are the same as TX.

Name: **ESTPPT** **Estimate values for missing precipitation data**

Use: CO TY=ESTPPT(TX,NMAX,QFLAG)

Description: Linearly interpolate estimates for cumulative precipitation values in TX with quality flags equal to or lesser in quality to QFLAG and place the results in TY. Possible flags, in order of decreasing quality, are: Q = questionable and M = missing. If the values bracketing the missing period are increasing with time, do not estimate more that NMAX continuous missing values. If the values bracketing the missing period are equal, then estimate any number of missing values. If the values bracketing the missing period are decreasing with time, do not estimate any missing values. TX must be a data type of INST-CUM. An IF condition has no effect. The units and type of TY are set equal to the units and type of TX.

Name: **FMA** **Forward moving average**

Use: CO TY=FMA(TX,NP)

Description: Compute a moving average of the last NP values in TX and store in TY. NP must be greater than 2. NP values at the beginning of TY will be missing. If a value in TX is missing, the value is not used for computing TY, and the average is over one less value. At least 2 values of TX must be defined, else TY is missing. Useful for computing flow durations. Also, may be used to determine parameters for use in the SCRN2 screening function. The units and type of TY are set equal to the units and type of TX. An IF condition has no effect.

Name: **GENTSR** **Generate a regular interval time series**

Use: CO TY=GENTSR(DT,TOFF,Y0,QFLAG)

Description: Generate a new, regular-interval time series TY with a time interval of DT, a time interval offset of TOFF, a constant value Y0, and all values internally flagged with QFLAG. TOFF is time from the beginning of the standard interval to the actual time of the data. DT and TOFF are expressed as units of time as nnT, where T is M for minutes, H for hours, and D for days. Possible flags are: N = none, M = missing, E = estimated, and Q = questionable. If Y0 is -901., then the flag is automatically M. Units and type must be set independently using the SD command. An IF condition has no effect.

Name: **INT** **Truncate to whole numbers**

Use: CO TY=INT(TX)

Description: Truncate to a whole number TX and store in TY. TX and TY can be the same variable. TX can be scalar, time series or paired data. If a value of TX is undefined, resulting values of TY are undefined. If a concurrent IF condition is not satisfied, TY is unchanged. The units and type of TY are set equal to the units and type of TX.

Name: **LAST** **Last valid value in a time series**

Use: CO SY=LAST(TX)

Description: Find the last valid value in time-series TX and place it in scalar SY. Ignores missing values or values concurrent with an unsatisfied IF condition.

Name: **LOG** **Natural log base "e"**

Use: CO TY=LOG(TX)

Description: Compute the natural log base e of TX and store in TY. TX and TY can be the same variable. TX can be scalar, time series or paired data. If a value of TX is undefined, resulting values of TY are undefined. If a concurrent IF condition is not satisfied, TY is unchanged. If a value in TX is 0 or negative, the value in TY will be set to missing. The units and type of TY are set equal to the units and type of TX.

Name: **LOG10** **Log base 10**

Use: CO TY=LOG10(TX)

Description: Compute the log base 10 of TX and store in TY. TX and TY can be the same variable. TX can be scalar, time series or paired data. If a value of TX is undefined, resulting values of TY are undefined. If a concurrent IF condition is not satisfied, TY is unchanged. If a value in TX is 0 or negative, the value in TY will be set to missing. The units and type of TY are set equal to the units and type of TX.

Name: **MATE** **Generate data pairs from two time-series**

Use: CO PF=MATE(TX,TY,SORT/NOSORT)

Description: Derive a paired-function PF by pairing values in the time series TX and TY. TX and TY values must have identical times. (Functions TS1 and TS4 may be useful in conjunction with MATE). The paired data are sorted into ascending order of TX if SORT is specified. The units of PF are respectively the same as those of TX and TY. The types and label of PF are undefined and must be set with the SD command. An IF condition has no effect.

Name: **MAX** **Maximum value in a time series**

Use: CO SY=MAX(TX)

Description: Find the maximum value in time-series TX and place it in scalar SY. Ignores missing values or values concurrent with an unsatisfied IF condition.

Name: **MEAN** **Mean value in a time series**

Use: CO SY=MEAN(TX)

Description: Compute the mean value in time-series TX and place the result in scalar SY. Ignores missing values or values concurrent with an unsatisfied IF condition.

Name: **MIN** **Minimum value in a time series**

Use: CO SY=MIN(TX)

Description: Find the minimum value in time-series TX and place the result in scalar SY. Ignores missing values or values concurrent with an unsatisfied IF condition.

Name: **MREG** **Multiple linear regression function**

Use: CO PF=MREG(TY,TX1,TX2,...,TXn,min,max)

Description: This function is used to determine the coefficients (Bn) of a linear regression equation of the general form (Y = B0 + B1*X1 + B2*X2 + Bn*Xn). Where Y is the dependent variable and the X1, X2, and Xn are independent variables. The MREG function requires that the function parameters TY, TX1, TX2, and TXn be regular time series variables and that they all have the same time interval. It is further required that the time variables be retrieved in the same sequential order as specified in the function parameter list. In other words TY is retrieved from DSS first followed by TX1 and so on. The number of independent time series variables that can be specified is limited to one less than the maximum number of time series variables that the program allows. On the DOS PC this value will be 4 independent variables. The calculated linear regression coefficients will be stored in a paired function variable specified by the user (PF). The user should store this variable to DSS with the PUT command if it is to be used later with the AMREG function. Two optional function parameters "min, max" are available to specify a range of values that control which values in the dependent variable are to be used for calculating the regression coefficients. These two parameters must be specified as the last two parameters and if specified, both must be specified as real numbers. The statistics output is written to file "MREG.REP" in addition to being printed out in the regular output.

Name: **MRG** **Merge two time series**

Use: CO TY=MRG(TX,TZ,QFLAG)

Description: Merge TX with TZ. TY includes all values in TX and TZ, except where TX and TZ occur at the same time. TX and TY cannot be the same variable. If TX and TZ occur at the same time, the value for TX is used unless it is flagged with a quality equal to or less than QFLAG and TZ is flagged with a quality greater than QFLAG. Possible flags, in order of decreasing quality, are: N = no flag, E = estimated, Q = questionable and M = missing. An IF condition has no effect. The type and units of TY are undefined and must be set by the SD command.

Name: **MRGP** **Merge two paired data series**

Use: CO PY=MRGP(PX,PZ,NCURVE)

Description: Merge PZ with PX. PY includes all values in PX and the curve specified by the index curve number NCURVE in PZ. It is required that PX and PZ have the same number of data values and that the X axis values for both PX and PZ are identical. NCURVE is used to specify the curve number data in PZ that is to be merged with PX. A value of 0 for NCURVE signifies that all curves in PZ are to be merged with PX.

Name: **MUSK** **Muskingum hydrologic routing**

Use: CO TY=MUSK(TX,NR,K,X)

Description: Route the uniform time series variable TX by the Muskingum hydrologic routing method and store it in variable TY. TX and TY can be the same variable. The "IF" compute option has no effect on this function. The "K" parameter is the Muskingum "k" in hours, "X" parameter is the Muskingum "x" (range between 0 and .5), and "NR" is the number of routing subreaches. The units and type of TY are set equal to the units and type of TX.

Name: **NINT** **Round to nearest whole number**

Use: CO TY=NINT(TX)

Description: Round to nearest whole number TX and store in TY. TX and TY can be the same variable. TX can be scalar, time series or paired data. If a value of TX is undefined resulting values of TY are undefined. If a concurrent IF condition is not satisfied, TY is unchanged. The units and type of TY are set equal to the units and type of TX.

Name: **OLY** **Olympic smoothing**

Use: CO TY=OLY(TX,NP[,CLEVEL])

Description: Compute a smoothed time-series from TX using the Olympic smoothing scheme: same as centered, moving average except the minimum and maximum values in the span NP are ignored. Place the result in TY. The units and type of TY are set equal to the units and type of TX. Variables TY and TX cannot be the same variable. The units and type of TY are set equal to the units and type of TX. The following levels are available:

LEVEL1 Only valid values will be averaged. Valid values that do not have all valid values within the averaging period number will be missing. Values at the start and end of the data that do not have enough valid values to average will be assigned missing values.

LEVEL2 Only valid values will be averaged. Valid values that do not have all valid values within the averaging period number will be missing. Valid values at the start and end of the data that do not have enough valid values to average will be averaged over a reduced number of values.

LEVEL3 (Default setting) All values will be averaged based on valid values within the averaging period number. If all values are missing for the period number, the value will be missing. Values at the start and end of the data that do not have enough valid values to average will be assigned missing values.

LEVEL4 All values will be averaged based on valid values within the averaging period number. If all values are missing for the period number, the value will be missing. Values at the start and end of the data that do not have valid values to average will be averaged based on a reduced number of values.

Note: Questionable and estimated flagged values are used in the computations.

Name: **PERCON** **Period constants**

Use: CO TY=PERCON(TS,TX)

Description: Generates a series TY at times concurrent with TX and with values equal to the previous chronological TS value. Variables TX and TY cannot be the same variable. If no previous TS is present, then TY values are undefined. TS and TX may be irregular- or regular-interval time series. An IF condition has no effect. The units and type of TY are undefined and should be set with the SD command.

Name: **POLY** **Polynomial transformation**

Use: CO TY=POLY(TX,TP)

Description: Compute a polynomial transformation of TX using the polynomial coefficients TP. Store the result in TY. Variables TX and TY cannot be the same variable. If a TX value is missing TY is undefined. If a concurrent IF condition is not satisfied, TY is unchanged. Units and type of TY are defined by TP. TP can be created with the utility DSSPD.

Name: **POLY2** **Polynomial transformation with integral**

Use: CO TY=POLY2(TX,TP)

Description: Compute a polynomial transformation of TX using the integral of the polynomial defined by coefficients TP. Variables TX and TY cannot be the same variable. Store the result in TY. If a TX value is missing TY is undefined. If a concurrent IF condition is not satisfied, TY is unchanged. Units and type are defined by TP. TP can be created with the utility DSSPD.

Name: **PULS** **Modified Puls or Working R&D routing function**

Use: CO TY=PULS(TX,PF,X,NR,STOR1,Q01)

Description: Route the uniform time series variable TX by the Modified Puls or Working R&D hydrologic routing method and store it in variable TY. Variables TY and TX may be the same variable. The variable PF must have been previously defined in a GET command. It references a storage-discharge paired function relationship for the reach. Storage must be the first variable, discharge the second variable, and each variable must repeat only once. Variable X is the wedge coefficient (Muskingum X) for use in working R&D. Use 0.0 value to route by Modified Puls method. Variable NR is the number of routing reaches. STOR1 and Q01 are initial storage and flow respectively. Use a value of -1 for both STOR1 and Q01 in order to use the first flow value in TX and interpolate a corresponding storage value from PF. The IF compute command has no effect on this function. The units and type of TY are set equal to the units and type of TX.

Name: **QAC** **Flow accumulator gage processor**

Use: CO TY=QAC(TX,TC)

Description: Compute period-average flows from a flow accumulator type gage:

TY(t) = (TX(t)-TX(t-1))/(TC(t)-TC(t-1))

TX and TC are respectively, time series of the accumulated flow and the count. TX and TC values must occur at the same times. Variables TY and TX cannot be the same variable. If corresponding values for TX and TC decrease from the previous period, then the accumulation is assumed to have reset to zero at the beginning of the interval. An IF condition has no effect. TY is assigned the type 'PER-AVER' and the units are set equal to the units of TX.

Name: **RND** **Round off**

Use: CO TY=RND(TX,NDIG,IPLAC)

Description: Round off values to NDIG or IPLAC, whichever controls. NDIG is the number of digits to round to and can range from 1 to 8. IPLAC is a magnitude of 10 to which to round to: for example, -1 specifies rounding to one-tenth (0.1). The number of digits shown is never less than 1, however. The following example illustrates the effects on rounding if NDIG=3 and IPLACE=0 (ie., round to ones place):

	Rounds	to
1445.1	"	1450.
144.51	"	145.
14.451	"	14.
1.4451	"	1.0

If TX is undefined, TY is undefined. If a concurrent IF condition is unsatisfied, TY is unchanged. Variable TY and TX can be the same variable.

Name: **RTABLE** **Rating table interpolation**

Use: CO TY=RTABLE(TX,TB)

Description: Interpolate values for TX using table TB and store the result in TY. Variables TY and TX cannot be the same variable. TB must be created using the program DSSPD (use -R option) with specific information in the header: type of interpolation, offset, shift, and datum. Program DSSUTL can be used to specify or modify the paired data header information. If the type of interpolation is LOGLOG, table x values are adjusted by subtracting the offset. The shift is added to and the datum subtracted from all incoming TX values. The header information in TB is used to define the units of TY. The type of TY should be set using the SD command. An IF statement has no effect on computation.

Name: **RTABLR** **Reverse rating table interpolation**

Use: CO TY=RTABLR(TX,TB)

Description: Interpolate values for TX using the reverse of table TB and store the result in TY. Variables TY and TX cannot be the same variable. TB must be created using the program DSSPD (option /R) with specific information in the header: type of interpolation, offset, shift, and datum. Program DSSUTL can be used to specify or modify the paired data header information. If the type of interpolation is LOGLOG, table x values are adjusted by subtracting

the offset. The shift is subtracted from and the datum added to all resulting TX values. The header information in TB also defines the units of TY. The type of TY should be set using the SD command. An IF statement has no effect on computation.

Name: **RTABL2** **Two-variable rating table interpolation**

Use: CO TY=RTABL2(TX,TZ,TB)

Description: TY is a function of two independent variables TX and TZ. Variable TY and TX cannot be the same variable. The functional relationship is specified by the table TB, which consists of sets of TX/TY pairs, each set corresponding to a TZ value. TB is created with the program DSSPD (option /R) with values for TZ specified as labels for the sets of TX/TY pairs. RTABL2 interpolates linearly in table TB. No extrapolation is done: if the TX or TY value is outside the range bounded by TB, TY is set to missing. TX and TZ must be concurrent time series. The units and type of TY must be set using the SD command. An IF statement has no effect on computation.

Name: **SCRN1** **Screen for erroneous values based on maximum/minimum range**

Use: CO TY=SCRN1(TX,XMIN,XMAX,MAXDEL,QFLAG)

Description: Flag any value in TX that falls outside the range XMIN - XMAX or exceeds the maximum change MAXDEL from the previous value. Variables TY and TX cannot be the same variable. The maximum change comparison is done only when the consecutive values are not flagged. Possible values for QFLAG are: M = missing data or Q = questionable. The result is placed in TY. The units and type of TY are set equal to the units and type of TX. An IF statement has no effect on computation.

Name: **SCRN2** **Screen for erroneous values based on forward moving average maximum**

Use: CO TY=SCRN2(TX,NPTS,MAXDEL,QFLAG)

Description: Flag any value in TX that exceeds the maximum change MAXDEL from the forward moving average of NPTS ending at the previous value. Missing values in TX are not counted in the moving average and the divisor of the average is less one for every missing value. Values which fail the screen are not counted either. At least 2 values must be defined else the moving average is undefined and the screen passes the relevant TX value. Possible flags are: M = missing data or Q = questionable. Variables TY and TX cannot be the same variable. The result is placed in TY. SCRN2 is useful for detecting and removing spikes. The FMA function may be useful for determining appropriate NPTS and MAXDEL parameters. The units and type of TY are set equal to the units and type of TX. An IF statement has no effect on computation.

Name: **SDEV** **Compute the standard deviation of one independent variable.**

Use: CO SY=SDEV(TX)

Description: This function is used to compute the standard deviation of one independent variable. TX can be either regular or irregular time series. SY is a scalar and is set to the computed standard deviation. This function requires a minimum of three valid values to compute.

Name: **SELECT** **Extract time series data at unique time specification.**

Use: CO TS= SELECT(TX,LEVEL,RANGE,FLAG,TWINDO,TFLAG)

Description: This function is used to extract data from any regular or irregular time series, based on any number of unique time specifications. The extraction process can take place on any of five different selection LEVELs which can be mutually inclusive or exclusive as defined by the FLAG and RANGE parameters. Each LEVEL has a RANGE parameter that is either a specific value or a range of values. The extracted data is defaulted to be irregular time series, but the user has the option to redefine it as regular by the TFLAG parameter. CAUTION: Defining the extracted data to be regular will cause it to be stored in a consecutive manner at regular time interval specified on the E pathname part without regard to its actual irregular date and time. Variables TS and TX cannot be the same variable. If a concurrent IF condition is not satisfied, TX is not selected. The units and type of TS are set equal to the units and type of TX.

Parameter	Description
TX	Input variable name of regular or irregular time series.
LEVEL	Level of time specific selection.
RANGE	Beginning and ending values of LEVEL. If only one value is specified, then the beginning and ending values are assumed to be the same. The following table shows all the valid LEVEL and RANGE values.

LEVEL	RANGE	Example
YEAR	Numeric: Four digit year value or a range of values. #### or ####-####	1938 or 1938-1945
MONTH	Alpha: First three characters of a month or a range of months. aaa or aaa-aaa	JAN or OCT-FEB
DAYMON	Numeric/Alpha: One or two digit day value or a range of values or the key word "LASTDAY" which specifies the last day of the month.	15 or 1-15 or LASTDAY or 15-LASTDAY
DAYWEE	Alpha: First three characters of a week day or a range of week days. aaa or aaa-aaa Note: Sunday is day 1 and Saturday is 7.	MON or SUN-SAT
TIME	Numeric: Four digit military style 24 hour clock consisting of a single time or a range of time. #### or ####-####	2300 or 0300-0600

Parameter	Description
FLAG	The valid entries for the processing FLAG are "INCLUDE" (default) or "EXCLUDE". This entry controls the inclusion or exclusion of all data specified by LEVEL and RANGE.
TWINDO	This parameter is only used with the TIME component of the LEVEL parameter and is expressed in minutes before and after the time of day within which the data will be extracted. The window is assumed to be symmetric about the time specified in RANGE. At the time boundaries of 0000 and 2400, the time window will not cross into the previous or next day.
TFLAG	The valid entries for the TFLAG are "IRREGULAR" (default) or "REGULAR". This parameter allows the user to control how DSSMATH treats the data extracted as it relates to its internal definition of the type of time series data it is. This allows the user to override the protection in the program to write irregular data as regular.

Name: **SHIFT** **Shift adjustment**

Use: CO TY=SHIFT(TS,TX)

Description: Generate a time series of shift adjustments TY with times at TX and actual shifts TS, possibly at other times. TX and TY cannot be the same variable. TS values are interpolated at TX times. The interpolation is linear between TS values, except when TX time is greater than last TS time. Then, the last value of TS is held constant for the remaining TX times. If no previous TS value brackets TY, then TY is set to zero. An IF condition has no effect. The units and type of TY are set equal to the units and type of TS.

Name: **SIN** **Sine trigonometric function**

Use: CO TY=SIN(TX)

Description: Compute the sine of TX in radian and store in TY. TX and TY can be the same variable. TX can be scalar, time series or paired data. If a value of TX is undefined resulting values of TY are undefined. If a concurrent IF condition is not satisfied, TY is unchanged. The units and type of TY are set equal to the units and type of TX.

Name: **SKEW** **Compute the skew coefficient of one independent variable**

Use: CO SY=SKEW(TX)

Description: This function is used to compute the skew coefficient of one independent variable. TX can be either regular or irregular time series. SY is a scalar and is set to the computed skew coefficient. This function requires a minimum of three valid values to compute.

Name: **SQRT** **Square root function**

Use: CO TY=SQRT(TX)

Description: Compute the square root of TX and store in TY. TX and TY can be the same variable. TX can be scalar, time series or paired data. If a value of TX is undefined, resulting values of TY are undefined. If a concurrent IF condition is not satisfied, TY is unchanged. If a value in TX is negative, the value in TY will be set to missing. The units and type of TY are set equal to the units and type of TX.

Name: **SS** **Straddle Stagger routing function**

Use: CO TY=SS(TX,NAVG,LAG,NR)

Description: Route the uniform time series variable TX by the Straddle Stagger hydrologic routing method and store it in variable TY. Note: Variables TY and TX cannot be the same variable and variable TX is permanently modified by being lagged by the SS function. Variable NAVG is the number of ordinates to average. Variable LAG is the number of ordinates to lag hydrograph. Variable NR is the number of subreaches to use. The "IF" compute option has no effect on this function. The units and type of TY are set equal to the units and type of TX.

Name: **SSW** **Wilmington District Straddle Stagger routing function**

Use: CO TY=SSW(TX,NAVG,LAG,NR)

Description: Route the uniform time series variable TX by the Wilmington District Straddle Stagger hydrologic routing method and store it in variable TY. Note: Variables TY and TX cannot be the same variable. This function is similar to function SS except that LAG is usually zero and the result of averaging NAVG values is stored in the NAVGed value. Variable NAVG is the number of ordinates to average. Variable LAG is the number of ordinates to lag hydrograph. Variable NR is the number of subreaches to use. For example, if NAVG is 7, the results for ordinates 13 through 19 are stored in ordinate 19. For the same example, if LAG is set to 2, the result is stored in ordinate 21. The "IF" compute option has no effect on this function. The units and type of TY are set equal to the units and type of TX.

Name: **TAN** **Tangent trigonometric function**

Use: CO TY=TAN(TX)

Description: Compute the tangent of TX in radian and store in TY. TX and TY can be the same variable. TX can be scalar, time series or paired data. If a value of TX is undefined resulting values of TY are undefined. If a concurrent IF condition is not satisfied, TY is unchanged. The units and type of TY are set equal to the units and type of TX.

Name: **TS1** **Interpolate data at regular intervals**

Use: CO TY=TS1(TX,DT,TOFF)

Description: Derive a new, regular-interval time series TY from TX. TY will have a time span defined by the current time window, a time interval of DT, and a time interval offset of TOFF. TOFF is time from the beginning of the standard interval to the actual time of the data. DT and TOFF are expressed as units of time as nnT, where T is M for minutes, H for hours, and D for days. TX may be a regular or irregular time series. Units and type of TY are set equal to the units and type of TX. TY and TX cannot be the same variable. An IF condition has no effect.

Name: **TS2** **Period averages at regular intervals**

Use: CO TY=TS2(TX,DT,TOFF)

Description: Derive a new, regular-interval time-series TY from the instantaneous values in TX with a time span defined by the current time window, a time interval of DT, and an interval offset of TOFF. DT and TOFF are expressed as units of time as nnT, where T is M for minutes, H for hours, and D for days. TOFF is time from the beginning of the standard interval to the actual time of the data. TX may be an irregular or regular time series, but must be of type IN-ST-VAL. TY will be typed PER-AVER. Example of use: deriving daily average flow from hourly observed values. TY and TX cannot be the same variable. An IF condition has no effect.

Name: **TS3** **Period mins or maxs at regular intervals**

Use: CO TY=TS3(TX,DT,TOFF,EXT)

Description: Finds either the minima or maxima in TX at DT intervals, with interval offsets at TOFF, and puts the result in TY. EXT is used to indicate the extreme of interest: 'MIN' or 'MAX'. DT and TOFF are expressed as units of time as nnT, where T is M for minutes, H for hours, and D for days. TOFF is time from the beginning of the standard interval to the actual time of the data. TY is typed PER-EXTR. Example of use: finding daily minima and maxima from hourly instantaneous observations. TY and TX cannot be the same variable. An IF condition has no effect.

Name: **TS4** **Interpolate data at irregular intervals**

Use: CO TY=TS4(TX,TZ)

Description: Derive a new, irregular-interval time series TY from TX at the times for TZ (TZ provides time pattern). Units and type are preserved for TY based on TX. TY and TX cannot be the same variable. An IF condition has no effect.

Name: **TSCYCL** **Time series cyclic analysis**

Use: CO XX=TSCYCL(TX,TREAT,STATFILE,DSSFILE)

Description: Derive a set of statistics from cyclic regular time series TX. TX must be regular data at a 1HOUR, 1DAY, or 1MONTH interval. The time series TX may be, for example, 30 years of 1DAY data. For each interval, (e.g., 1st day of the year, 2nd day of the year, ... , 365th day of the year), statistics are determined. The 14 statistics determined for each interval are: maximum, minimum, average, standard deviation, 5%, 10%, 25%, 50% (median), 75%, 90%, 95% percentiles, date of maximum, date of minimum, and number of values processed. These results are written to a specified DSS file, and two text output files. The variable XX is a dummy variable and is ignored. An IF condition has no effect.

TREAT controls the treatment of newly generated statistic values when missing data occur in the interval. It must be expressed in either the form nn# or nn%. If nn# is used nn gives the number of missing data values in the interval that will be accepted in the computation of the new value. If nn% is used nn gives the percent of missing data values in the interval that will be accepted in the computation of the new value. If the criteria is met a data value will be generated, if not a missing value will occur.

STATFILE is the base name, not more than 6 characters, of two files that will be written containing the same data written to the DSS file. The first file will contain all of the statistics generated. It will be in the form of a wide table. Its name will be 'statfile.sts' or 'statfile.t' where 'statfile' is the portion of the name the user specifies. This file is wide, so it is hard to view on some systems. The second file is a subset of the first named 'statfile.sum', or 'statfile.s'. The second file is limited to 80 columns wide for easy viewing. The second file contains: maximums with their dates, minimums with their dates, average, and the 50% percentile(median) values.

DSSFILE is the DSS file into which the cyclic results will be written. The results will be, for example, a record of all the maximums for each day of the year over the 30 year period. Fourteen records will be added to the DSS file specified, each containing one of the derived statistics. The records will use the pseudo year 3000 for storing the statistics.

Name: **TSHIFT** **Shift time series in time**

Use: CO TY=TSHIFT(TX,DT)

Description: Derive a time series TY by shifting times in TX by DT. DT is specified in the form nnT, where nn is the number of time units T and T may be M for minutes, H for hours, or D for days. Units and type of TY are set equal to the units and type of TX. An IF condition has no effect.

Name: **TSNAP** **Snap Irregular times to nearest Regular period**

Use: CO TY=TSNAP(TX,DT,TOFF,TBACK,TFORWARD)

Description: Derive a regular time series from existing time series TX. TX may be regular or irregular. The new time series will be at a time interval of DT, interval offset of TOFF, look back of TBACK, and look forward of TFORWARD. Parameters: DT, TOFF,TBACK, and TFORWARD are expressed as units of time "nnT", where "nn" is a number and "T" is MIN (minutes), H (hours), D (days), W (weeks), MON(months), or Y (years). Units and type of TY are set equal to the units and type of TX. An IF condition has no effect.

Name: **TTSR** **Transform time series to regular**

Use: CO TY=TTSR(TX,DT,TOFF,FUNCT,TREAT,TYPE)

Description: Derive a new regular time series from existing time series TX. Variables TY and TX cannot be the same variable. An IF condition has no effect. TX may be regular or irregular. The new time series will be at a time interval of DT, and an interval offset of TOFF. DT and TOFF must each be expressed in the form nnT, where T may be MIN, HOUR, DAY, WEEK, SEM, TRI, MON or YEAR (e.g. 1DAY). The units and type of TY must be defined by the use of the SD command. FUNCT is one of the following:

INT - Interpolation at end of interval
MAX - Maximum over interval
MIN - Minimum over interval
AVE - Average over interval
ACC - Accumulation over interval
ITG - Integration over interval
NUM - Number of valid data over interval

TREAT controls the treatment of the new generated data value when missing data occur in the interval. It must be expressed in either the form

nn# or nn%. If nn# is used nn gives the number of missing data values in the interval that will be accepted in the computation of the new value. If nn% is used nn gives the percent of missing data values in the interval that will be accepted in the computation of the new value. If the criteria is met a data value will be generated, if not, a missing value will occur.

TYPE provides the user control to override how the interpolation and processing of data occurs. TYPE must be one of the following: DEFAULT, INST-VAL, PER-AVER, or PER-CUM. If DEFAULT is used, processing depends on the data type stored in the DSS data record. Otherwise processing will be performed as if the data type were as given by TYPE. Data type INST-VAL considers the data to change linearly from the previous data value to the current data value over the interval. Data type PER-AVER considers the data to be constant at the current data value over the interval. Data type PER-CUM considers the data to increase from zero (0.0) up to the current value over the interval. Note: It is required that valid values exist at the start of the data being used.

Name: **TTSI** **Transform time series to irregular**

Use: CO TY=TTSI(TX,TZ,FUNCT,TREAT,TYPE)

Description: Derive a new irregular time series from existing time series TX. TX may be regular or irregular. TZ is an existing time series at the desired new spacing of TY. All other parameters and restrictions are as shown for the function TTSR.

Name: **1/X** **Inverse function**

Use: CO TY=1/X(TX)

Description: Divide 1 by variable TX and store in TY. TX and TY can be the same variable. TX can be scalar, time series or paired data. If a value of TX is undefined, resulting values of TY are undefined. If a concurrent IF condition is not satisfied, TY is unchanged. If a value in TX is zero, the value in TY will be set to missing. The units and type of TY are set equal to the units and type of TX.

Appendix C

Programmer's Guide

Appendix C - Programmer's Guide

This appendix provides instructions on how a user, who is proficient in the use of the Fortran computer program language, can modify existing DSSMATH compute functions and create new user defined functions. The DSSMATH program has been structured to accommodate the relatively easy addition of new functions. The user modifies one standard DSSMATH subroutine "USRFUN", in which a user defined function is identified and then takes one of the existing function subroutines and modifies it to incorporate the new function. The most difficult task is the understanding of the large number of variables that DSSMATH uses to store all the information dealing with the function parameters and data values. All the variables are documented in the common blocks in which they reside.

The following instructions are given as an example in which a simple user defined function is created. The name of the new function will be **ABS(TX).** The function will take the absolute value of all values in the **TX** variable.

Step 1. - Modify DSSMATH subroutine USRFUN.FOR

The shaded sections of code reflect the additions or modifications to the existing code to add the new user function **ABS.** Existing code is commented out; if it needs to be modified, it is duplicated and changes are made to the duplicated code.

```
      SUBROUTINE USRFUN (ISTAT)
C
C     PROVIDES A LINK WITH FUNCTIONS DEVELOPED LOCALLY
C     ****************************************************************
C
C     SUBROUTINE AUTHOR:  DENNIS HUFF
C     HYDROLOGIC ENGINEERING CENTER
C     DEVELOPMENT DATE:   5 FEB 88
C     REVISION LOG:
C     SUBROUTINE PARAMETERS:
C
C     ISTAT    -  (ISTAT = -1) ===>  SOMETHING WENT WRONG WHILE
C                      PROCESSING THE COMMAND LINE FOR THE FILE NAME.
C                (ISTAT = 0 ) ===>  EVERYTHING IS OK.
$ADD C.IOCOM
$ADD C.CFUNCT
C
C     DETERMINE WHICH FUNCTION IS BEING USED
C
C     ****************************************************************
C       IF(CFUNEX(1:4).EQ.'USER') THEN
C         CALL USRSUB(ISTAT)
        IF(CFUNEX(1:3).EQ.'ABS') THEN
          CALL ABSFUN(ISTAT)
C
C     ****************************************************************
      ELSE
         ISTAT = -1
         WRITE(IFOUT,*)' ERROR - FUNCTION  ',CFUNEX(1:6),'  NOT VALID'
      ENDIF
C
      RETURN
      END
```

Step 2. - Modify DSSMATH subroutine F.ACC

The second step is to pick one of the existing DSSMATH functions and modify it to create the new function **ABS.** In the example below, a copy of the existing file **ACC.FOR** is modified to produce the new **ABSFUN.FOR** subroutine. **ACC** was chosen because it is a function that is similar to function **ABS**. The following is a listing of the DSSMATH function **ACC**. The common blocks have been turned on so the variable definitions are visible.

```
~~~
```

```
C$ADD C.DSSDA1                                                          $
C                                                                       $
C     ******************************************************************$
C     THIS COMMON BLOCK CONTAINS VARIABLES DEALING WITH THE DSS DATA AND $
C     VARIABLE NAMES BEING USED TO HOLD THIS DATA                       $
C                                                                       $
      PARAMETER (NTS=10,NPF=5,NSCA=15,NPFC=30,NTSV=5000,NPFV=5000,      $
     & NPAT=NTS+NPF,NVAR=NTS+NPF+NSCA)                                  $
C                                                                       $
      COMMON /DSSDA1/ NHEADW(NPAT),HEADW(100,NPAT),NTYPE(NPAT),         $
     & NPNP(6,NPAT),NPATH(NPAT),ISTIME,ISDATE,IETIME,IEDATE,IHORIZ(NPAT)$
     & ,TSDATE(NTSV,NTS),TSY(NTSV,NTS),PFX(NPFV,NPF),PFY(NPFV,NPF),     $
     & NCURVE(NPF),NDATA(NPAT),IPATH,SCALAR(NSCA),IQFLAG(NTSV,NTS)      $
C                                                                       $
      INTEGER HEADW                                                     $
C                                                                       $
      COMMON /RDSSDA/ RBUFF                                             $
      REAL RBUFF(NTSV*2+100)                                            $
C                                                                       $
      COMMON /CDSSD1/ VARLBL,CPNP,CPATP,CSTIME,CSDATE,CETIME,CEDATE,    $
     & CUNITS,CTYPE,PATHNM,CARYLB                                       $
      CHARACTER VARLBL(NVAR)*6,CPNP(6,NPAT)*32,CPATP(6)*32,             $
     & CSTIME*4,CSDATE*7,CETIME*4,CEDATE*7,                             $
     & CUNITS(2,NPAT)*8,CTYPE(2,NPAT)*8,PATHNM(NPAT)*80,                $
     & CARYLB(NPFC,NPF)*8                                               $
C                                                                       $
      COMMON /LDSSDA/ LTWSET                                            $
      LOGICAL LTWSET                                                    $
C                                                                       $
C     DESCRIPTION OF VARIABLES                                          $
C                                                                       $
C     NPAT      -  MAXIMUM NUMBER OF PATHNAMES                          $
C     NTS       -  MAXIMUM NUMBER OF TIME SERIES VECTORS                $
C     NPF       -  MAXIMUM NUMBER OF PAIRED FUNCTIONS                   $
C     NSCA      -  MAXIMUM NUMBER OF SCALARS                            $
C     NPFC      -  MAXIMUM NUMBER OF DEPENDENT VECTORS PER PAIRED       $
C                  FUNCTION                                             $
C     NTSV      -  MAXIMUM SIZE OF TIME SERIES VECTOR                   $
C     NPFV      -  MAXIMUM SIZE OF PAIRED FUNCTION VECTOR               $
C     NVAR      -  MAXIMUM NUMBER OF USER VARIABLES                     $
C     NHEADW()   -  LENGTH OF HEADER ARRAY                              $
C     HEADW()    -  DSS DATA HEADER ARRAY                               $
C     VARLBL()  -  USER VARIABLE LABELS                                 $
C                  FIRST NTS SLOTS POINT TO TS DATA                     $
C                  NEXT NPF SLOTS POINT TO PF DATA                      $
C                  LAST NSCA SLOTS POINT TO SCALARS                     $
C     NTYPE()   -  TYPE OF DATA THAT IS HELD BY PATHNAME                $
C                  1 - TIME SERIES REGULAR TIME INTERVAL                $
C                  2 - TIME SERIES IRREGULAR TIME INTERVAL              $
C                  3 - PAIRED FUNCTION DATA                             $
C                  4 - SCALAR VARIABLE                                  $
C                  5 - CHARACTER LABEL                                  $
C     IHORIZ    -  VARIABLE NUMBER TO APPEAR ON THE HORIZONTAL AXIS FOR $
C                  PLOTTING (1 OR 2) - PAIRED DATA ONLY                 $
C     IQFLAG()  -  INTEGER ARRAY USED TO INDICATE THE FOLLOWING FOR     $
C                  DATA VALUES STORED IN TSY() ARRAY.                   $
C                  0 - NO QUALITY STATUS IMPLIED -                      $
C                  1 - ESTIMATED VALUE                                  $
C                  2 - QUESTIONED VALUE                                 $
C                  3 - MISSING (UNDEFINED) VALUE FLAG                   $
C     ISTIME    -  TIME WINDOW FOR STARTING TIME  (MINUTES)             $
C     IETIME    -  TIME WINDOW FOR ENDING TIME    (MINUTES)             $
C     ISDATE    -  TIME WINDOW FOR STARTING DATE  (DAY COUNT)           $
C     IEDATE    -  TIME WINDOW FOR ENDING DATE    (DAY COUNT)           $
C     CSTIME    -  CHARACTER STARTING TIME (24 HOUR TIME)               $
C     CETIME    -  CHARACTER ENDING TIME   (24 HOUR TIME)               $
C     CSDATE    -  CHARACTER STARTING DATE (MILITARY STYLE)             $
C     CEDATE    -  CHARACTER ENDING DATE   (MILITARY STYLE)             $
C     CPATP()   -  LAST PATHNAME ENTERED                                $
C     IPATH     -  POINTER FROM CPATP TO CPNP                           $
```

```
C     CPNP()     -  PATHNAME PARTS A-F                                          $
C     NPNP()     -  LENGTH OF PATHNAME PARTS A-F                                $
C     CUNITS()   -  UNITS OF DSS DATA                                           $
C     CTYPE()    -  TYPE OF DSS DATA                                            $
C     PATHNM()   -  FULL PATHNAMES                                              $
C     NPATH()    -  LENGTH OF PATHNAMES                                         $
C     CARYLB()   -  LABELS FOR PAIRED FUNCTION DEPENDENT VECTORS                $
C     TSDATE()   -  TIME SERIES DATES (JULIAN DAYS WITH FRACTIONS)              $
C     TSY()      -  TIME SERIES DEPENDENT VARIABLE                              $
C     PFX()      -  PAIRED FUNCTION INDEPENDENT VARIABLE                        $
C     PFY()      -  PAIRED FUNCTION DEPENDENT VARIABLES                         $
C     NCURVE()   -  NUMBER OF DEPENDENT VARIABLES PER PAIRED FUNCTION           $
C     NDATA()    -  NUMBER OF ORDINATES PER CURVE (PF OR TS)                    $
C     LTWSET     -  LOGICAL FLAG TO INDICATE IF THE TIME WINDOW IS SET          $
C     RBUFF      -  DSS WORK SPACE                                              $
C     SCALAR()   -  SCALAR VARIABLES                                            $
C                                                                               $
C     ******************************************************************        $
C                                                                               $
      COMMON /TBUFFR/ XTBUFF(NTSV),ITBUFX(NTSV)                                 $
C                                                                               $
C     XTBUFF() -  TEMPORARY BUFFER SPACE USED TO STORE A REAL VARIABLE          $
C     ITBUFX() -  TEMPORARY BUFFER SPACE USED TO STORE AN INTEGER VARIABLE      $
C
```

```
C$ADD C.IOCOM
C                                                                               $
C     ******************************************************************        $
C     THIS COMMON BLOCK HOLD VARIABLES DEALING WITH INPUT AND OUTPUT            $
C     UNIT NUMBERS AND OPERATING ENVIRONMENT                                    $
C                                                                               $
      COMMON /IOCOM/ IFOUT,INPUT,IENVIR                                         $
C                                                                               $
C     VARIABLE DESCRIPTION                                                      $
C                                                                               $
C   INPUT   -  INPUT UNIT NUMBER                                                $
C   IFOUT   -  OUTPUT UNIT NUMBER                                               $
C   IENVIR  - =0 - INTERACTIVE EXECUTION                                        $
C             >0 - REAL TIME                                                    $
C             <0 - BATCH                                                        $
C     ******************************************************************        $
C                                                                               $
```

```
C$ADD C.CDEPND
C                                                                               $
C     ******************************************************************        $
C     THIS COMMON BLOCK CONTAINS VARIABLES DEALING WITH THE DEPENDENT           $
C     VARIABLE USED IN THE  COMPUTE COMMAND                                     $
C                                                                               $
      COMMON /CDEPND/ ICDPVA,ICDPTY,IDPVIF(5000)                                $
      COMMON /CCDEPN/ CDPVAR                                                    $
      CHARACTER CDPVAR*6                                                        $
C                                                                               $
C     DESCRIPTION OF VARIABLES                                                  $
C     CDPVAR   -  CHARACTER VARIABLE USED TO STORE DEPENDENT VARIABLE           $
C                 NAME                                                          $
C     ICDPVA   -  INTEGER VARIABLE USED TO SPECIFY TWO THINGS:                  $
C                 0 - DEPENDENT VARIABLE CDPVAR HAS NOT BEEN DEFINED            $
C               0 < - INDEX OF CDPVAR LOCATION OF ITS DATA                      $
C     ICDPTY   -  INTEGER VARIABLE  USED TO SPECIFY THE TYPE OF DATA            $
C                 STORED  IN DEPENDENT VARIABLE CDPVAR                          $
C     IDPVIF()  -  INTEGER ARRAY USED TO INDICATE IF THE DEPENDENT              $
C                  VARIABLE SHOULD BE PROCESSED BASED ON THE IF FUNCTION        $
C                  RESULTS                                                      $
C                  0 - PROCESS THE DATA VALUE                                   $
C                  1 - DO NOT PROCESS THE DATA VALUE                            $
C                  2 - ONE OF THE IF FUNCTION PARAMETERS WAS MISSING            $
C                      AND THEREFORE DO NOT PROCESS THE DATA VALUE              $
```

```
C                                                                       $
C     ******************************************************************$
C                                                                       $
```

```
C$ADD C.CFUNCT
C                                                                       $
C     ******************************************************************$
C     THIS COMMON BLOCK CONTAINS VARIABLES DEALING WITH THE FUNCTIONS   $
C     USED IN THE COMPUTE COMMAND                                       $
C                                                                       $
      COMMON /CFUNCT/ ICFUNP(10),ICFPTY(10),ICFPSG(10),XCFUNP(10),      $
     . NFUNP,LFUNOP                                                     $
      LOGICAL LFUNOP                                                    $
      COMMON /CCFUNC/ CFUNEX,CFUNP                                      $
      CHARACTER*10 CFUNEX,CFUNP(10)                                     $
C     DESCRIPTION OF VARIABLES                                          $
C                                                                       $
C     CFUNEX   -  CHARACTER VARIABLE USED TO STORE THE FUNCTION NAME    $
C     CFUNP    -  CHARACTER VARIABLE ARRAY THAT HOLDS THE NAMES OF      $
C                 FUNCTION PARAMETERS                                   $
C     ICFUNP   -  INTEGER ARRAY USED FOR TWO PURPOSES                   $
C                 0 - INDICATES THAT FUNCTION PARAMETER IS A CONTANT    $
C               0 < - INDEX OF FUNCTION PARAMETER LOCATION              $
C     ICFPTY   -  INTEGER ARRAY USED TO SPECIFY THE TYPE OF DATA STORED $
C                  SEE DESCRIPTION OF NTYPE() VARIABLE IN DSSDA1 COMMON $
C     ICFPSG   -  INTEGER ARRAY USED TO HOLD SIGN OF PARAMETER VARIABLE $
C                 1 FOR POSITIVE AND -1 FOR NEGATIVE PARAMETER          $
C     LFUNOP   -  LOGICAL VARIABLE USED TO INDICATE IF THE  FUNCTION    $
C                 OPERATION IS BEING USED                               $
C     NFUNP    -  NUMBER OF FUNCTION PARAMETERS                         $
C     XCFUNP   -  ARRAY USED FOR REAL NUMBER CONSTANT VALUE OF FUNCTION $
C     ******************************************************************$
C                                                                       $
```

```
C
C     SET SUBROUTINE PERCHK PARAMETER CHECKS FOR THIS FUNCTION
      DIMENSION ILGPTY(1),ITSA1(1),ITSA2(1)
      DATA NFPR,NITSA1,NITSA2 /1,0,0/
      DATA ILGPTY /12/
C
      CALL PERCHK( NFPR,ILGPTY,NITSA1,ITSA1,NITSA2,ITSA2,
     . ISTAT)
      IF(ISTAT.LT.0) GO TO 9000
C
      CALL STHDPV(1,ISTAT)
      IF(ISTAT.NE.0) GO TO 9000
```

```
C       IF(CTYPE(1,ICFUNP(1)).EQ.'PER-CUM') THEN
C       .   CTYPE(1,ICDPVA) = 'INST-CUM'
C       ELSE
C       .   CTYPE(1,ICDPVA) = CTYPE(1,ICFUNP(1))
C       ENDIF
      CTYPE(1,ICDPVA) = CTYPE(1,ICFUNP(1))
```

```
      CUNITS(1,ICDPVA)= CUNITS(1,ICFUNP(1))
C
C     PROCESS THE DATA
```

```
C       SUM = 0.
      DO 1000 I=1,NDATA(ICFUNP(1))
C        CHECK THE IF FUNCTION FLAG
       .  IF(IDPVIF(I).NE.0) THEN
```

```
C       .   .   TSY(I,ICDPVA)=SUM
                TSY(I,ICDPVA) = TSY(I,ICFUNP(1))
```

```
       .   .   IQFLAG(I,ICDPVA) = 0
```

```
C         CHECK TO SEE IF THE DATA IS MISSING
        . ELSE IF(IQFLAG(I,ICFUNP(1)).GT.2) THEN
        . .  IQFLAG(I,ICDPVA)=3
        . .  TSY(I,ICDPVA)=-901
C         COMPUTE THE ACCUMULATION
        . ELSE
```

~~C . . SUM = SUM + TSY(I,ICFUNP(1))~~
~~C . . TSY(I,ICDPVA)=SUM~~

```
                TSY(I,ICDPVA) = ABS(TSY(I,ICFUNP(1)))

        . .  IQFLAG(I,ICDPVA)=0
        . ENDIF
  1000  . CONTINUE
C
      NDATA(ICDPVA)=NDATA(ICFUNP(1))
C
      NTYPE(ICDPVA) = NTYPE(ICFUNP(1))
      NHEADW(ICDPVA) = NHEADW(ICFUNP(1))
C
  9000 RETURN
      END
C
```

Appendix D

Compute Function Examples

Appendix D - Compute Function Examples

Index of Functions

		Page
ABS	Absolute function	3
ACC	Running accumulation	5
AMREG	Apply multiple linear regression equation	6
CONIC	Conic interpolation from elevation/area table	8
CORR	Compute correlation coefficients	10
COS	Cosine trigonometric function	12
COUNT	Count the number of valid and missing data values	13
CMA	Centered moving average smoothing	14
DDT	Differences per unit time	16
DECPAR	Decaying basin wetness parameter	18
DIFF	Successive differences	20
ESTLIN	Estimate values for missing data	21
ESTPPT	Estimate values for missing precipitation data	22
FMA	Forward moving average	23
GENTSR	Generate a regular interval time series	24
INT	Truncate to whole numbers	25
LAST	Last valid value in a time series	25
LOG	Natural log base "e"	25
LOG10	Log base 10	25
MATE	Generate data pairs from two time-series	26
MAX	Maximum value in a time series	28
MEAN	Mean value in a time series	28
MIN	Minimum value in a time series	28
MREG	Multiple linear regression function	29
MRG	Merge two time series	31
MRGP	Merge two paired data series	32
MUSK	Muskingum hydrologic routing	33
NINT	Round to nearest whole number	35
OLY	Olympic smoothing	36
PERCON	Period constants	39
POLY	Polynomial transformation	39
POLY2	Polynomial transformation with integral	42
PULS	Modified Puls or Working R&D routing function	45
QAC	Flow accumulator gage processor	47
RND	Round off	47
RTABLE	Rating table interpolation	49
RTABLR	Reverse rating table interpolation	49
RTABL2	Two-variable rating table interpolation	51
SCRN1	Screen for erroneous values based on maximum/minimum range	56

SCRN2	Screen for erroneous values based on forward moving average maximum	57
SDEV	Compute the standard deviation of one independent variable.	58
SELECT	Extract time series data at unique time specification.	59
SHIFT	Shift adjustment	64
SIN	Sine trigonometric function	12
SKEW	Compute the skew coefficient of one independent variable	57
SQRT	Square root function	66
SS	Straddle Stagger routing function	66
SSW	Wilmington District Straddle Stagger routing function	66
TAN	Tangent trigonometric function	12
TS1	Interpolate data at regular intervals	68
TS2	Period averages at regular intervals	68
TS3	Period mins or maxs at regular intervals	68
TS4	Interpolate data at irregular intervals	70
TSCYCL	Time series cyclic analysis	71
TSHIFT	Shift time series in time	75
TSNAP	Snap Irregular times to nearest Regular period	76
TTSR	Transform time series to regular	77
TTSI	Transform time series to irregular	82

Introduction

Appendix D is designed to help the user understand the DSSMATH functions by the use of examples. The examples are primarily designed to demonstrate the use of the function and may or may not be good examples in real life applications, but where possible, valid real life examples were used. Each of the functions has, at a minimum, one example of its use. Where appropriate, graphics and tabular outputs have been used to make it more clear as to the functions use. The examples are shown as a set of commands associated with a PREAD macro. The example macros consist of more than one function in many of the examples. Some of the functions have multiple options and require more than one example to demonstrate their use.

Name: **ABS Absolute function**
Use: CO TY=ABS(TX)

Compute the absolute value of TX and store in TY. TX and TY can be the same variable. TX can be scalar, time series or paired data. If a value of TX is undefined, resulting values of TY are undefined. If a concurrent IF condition is not satisfied, TY is unchanged.

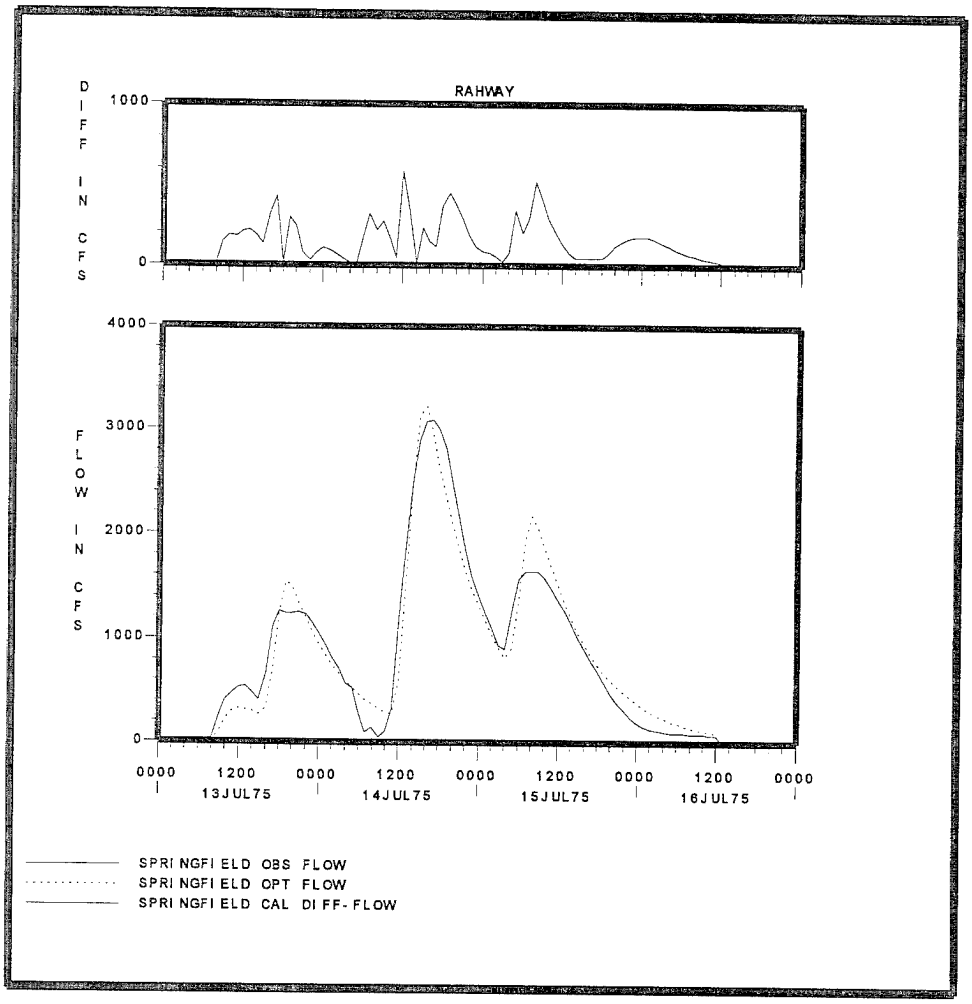
Example: The following example uses the ABS function to convert a set of difference values between observed flow and computed to positive values. This is done to allow the computation of the absolute maximum difference and the average difference.

Macro:

```
MACRO TABS
TIME 13JUL1975 0800 16JUL1975 1200
GET OBS=WS6N.DSS:/RAHWAY/SPRINGFIELD/FLOW/01JUL1975/1HOUR/OBS/
GET CAL=WS6N.DSS:/RAHWAY/SPRINGFIELD/FLOW/01JUL1975/1HOUR/OPT/
COMP DIFF=OBS-CAL
COMP DABS=ABS(DIFF)
PUT.A DABS=/RAHWAY/SPRINGFIELD/DIFF-FLOW/01JUL1975/1HOUR/CAL/
COMP MAXVAL=MAX(DABS)
COMP AVGDIF=MEAN(DABS)
SHOW MAXVAL AVGDIF
ENDMACRO
```

Execution:

```
I>!R TABS
    -----DSS---ZOPEN:  Existing File Opened,  File: WS6N.DSS
                       Unit:  71;  DSS Version: 6-GX
 -----DSS--- ZREAD Unit  71; Vers.    1:  /RAHWAY/SPRINGFIELD/FLOW/01JUL1975/1HOUR/OBS/
 -----DSS--- ZREAD Unit  71; Vers.    1:  /RAHWAY/SPRINGFIELD/FLOW/01JUL1975/1HOUR/OPT/
 -----DSS---ZWRITE Unit  71; Vers.    3:  /RAHWAY/SPRINGFIELD/DIFF-FLOW/01JUL1975/1HOUR/CAL/

 MAX=     591.306  MIN=     11.733  MEAN=     166.694  LAST VALUE=     24.544


 VARIABLE=MAXVAL
 MAXVAL  =  591.306200

 VARIABLE=AVGDIF
 AVGDIF  =  166.694000
```



Name: **ACC** **Running accumulation**
Use: CO TY=ACC(TX)

Compute a running accumulation of the values in TX and store the result in TY. If a TX value is undefined or a concurrent IF condition is not satisfied, the value of TX is not added to the accumulation and the corresponding TY remains the same as the previous TY. Units of TY are the same as those of TX. If TX is typed as PER-AVER or PER-CUM, TY will be typed INST-VAL or INST-CUM, respectively.

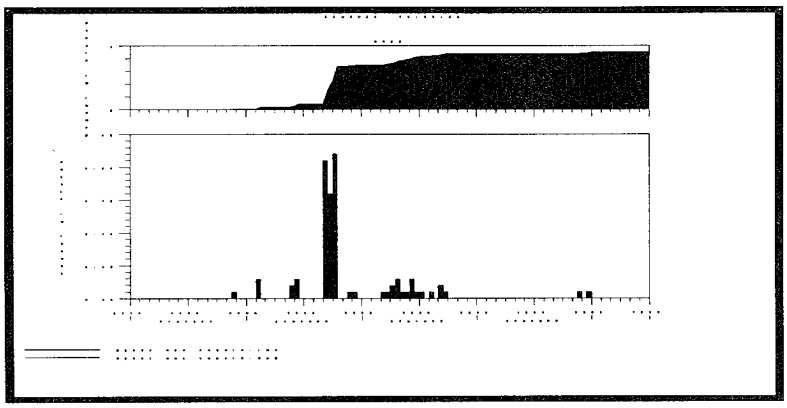
Example: The following example uses the accumulation function to take incremental precipitation and convert into a cumulative value.

Macro:

```
MACRO TACC
** COMPUTE A RUNNING ACCUMULATION OF THE VALUES IN PPT
** AND STORE IN PACC
CL ALL
TIME 01APR83 0100 05APR83 2400
GET PPT=testdb.dss:/GOES/NBKD3/PRECIP-INC/01ARP1983/1HOUR/OBS/
COMP PACC=ACC(PPT)
PUT.A PACC=C=PRECIP-CUM F=CAL
$CONTINUE INCASE OF ERROR
ENDMACRO
```

Execution:

```
I>!R TACC
  ALL VARIABLES HAVE BEEN CLEARED
    -----DSS---ZOPEN:  Existing File Opened,  File: TESTDB.DSS
                       Unit:  71;  DSS Version: 6-IA
 -----DSS--- ZREAD Unit  71; Vers.  145:  /GOES/NBKD3/PRECIP-INC/01APR1983/1HOUR/OBS/
  FINAL ACCUMULATED VALUE =   8.999997E-01
 -----DSS---ZWRITE Unit  71; Vers.    2:  /GOES/NBKD3/PRECIP-CUM/01APR1983/1HOUR/CAL/
```



Name: AMREG **Apply multiple linear regression equation.**
Use: CO TY=AMREG(PF,TX1,TX2,...,TXn,min,max)

This function is used to calculate a regular time series (TY) based on a linear regression equation of the general form (Y = B0 + B1*X1 + B2*X2 + Bn*Xn). Where Y is the dependent variable and the X1, X2, and Xn are independent variables. The AMREG function requires that the function parameters TY, TX1, TX2, and TXn be regular time series variables and that they all have the same time interval. It is further required that the time variables be retrieved in the same sequential order as specified in the function parameter list. In other words TX1 is retrieved from DSS first followed by TX2 and so on. The number of independent time series variables that can be specified is limited to one less than the maximum number of time series variables that the program allows. On the DOS PC this value will be 4 independent variables. The linear regression coefficients (B0 - Bn) are stored in a paired function variable (PF) which is normally retrieved from DSS. Two optional function parameters "min max" are available to specify a range of values that control which values in the dependent variable are to be accepted as valid computed values. These two parameters must be specified as the last two parameters and if specified, both must be specified as real numbers. Units of TY are the same as those of TX1. An IF condition has no effect.

Example: The following example uses the AMREG function to take previously determined linear regression coefficients determined by MREG function and apply them to flow values at two upstream gages and determine flow values at a downstream gage.

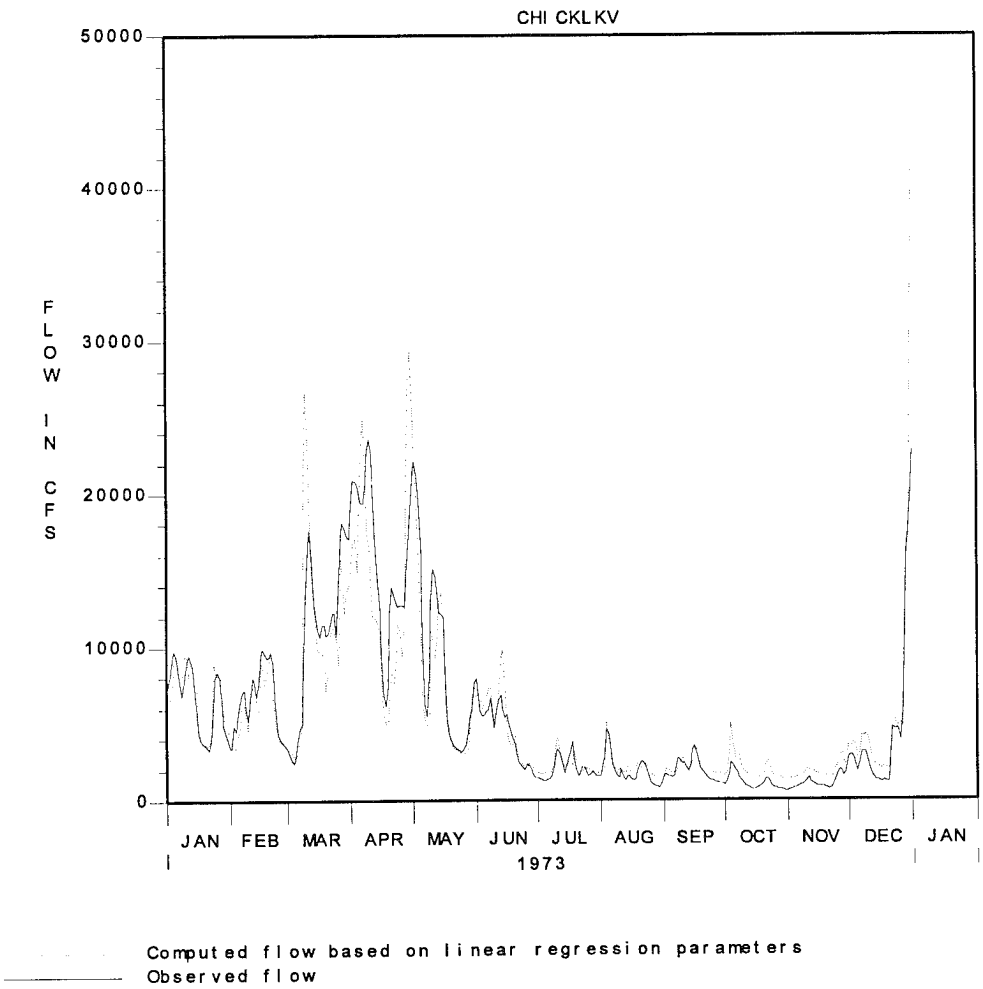
Macro:

```
MACRO AMREG2
CLEAR ALL
TIME 2400 01OCT1972 2400 30SEP1977
GET PF=MREG.DSS:/LINEAR REGRESSION/COEFFICIENTS/5YEARS////
GET TX2=MREG.DSS:/CHICKENT/TX2/FLOW/01JAN1972/1DAY/OBS/
GET TX3=MREG.DSS:/BUCKNER/TX3/FLOW/01JAN1972/1DAY/OBS/
COMP TX2=TSHIFT(TX2,4D)
COMP TX3=TSHIFT(TX3,1D)
COMP TX4=AMREG(PF,TX2,TX3)
PUT.A TX4=A=CHICKLKV F=COMPUTED5
ENDMACRO
```

Execution:

```
I>!R AMREG2
ALL VARIABLES HAVE BEEN CLEARED
    -----DSS---ZOPEN:  Existing File Opened,  File: MREG.DSS
                       Unit:  71;  DSS Version: 6-HD
 -----DSS--- ZREAD Unit  71; Vers.   11:  /LINEAR REGRESSION/COEFFICIENTS/5YEARS////
 -----DSS--- ZREAD Unit  71; Vers.    3:  /CHICKENT/TX2/FLOW/01JAN1972/1DAY/OBS/
 -----DSS--- ZREAD Unit  71; Vers.    1:  /CHICKENT/TX2/FLOW/01JAN1973/1DAY/OBS/
 -----DSS--- ZREAD Unit  71; Vers.    1:  /CHICKENT/TX2/FLOW/01JAN1974/1DAY/OBS/
 -----DSS--- ZREAD Unit  71; Vers.    1:  /CHICKENT/TX2/FLOW/01JAN1975/1DAY/OBS/
 -----DSS--- ZREAD Unit  71; Vers.    1:  /CHICKENT/TX2/FLOW/01JAN1976/1DAY/OBS/
 -----DSS--- ZREAD Unit  71; Vers.    1:  /CHICKENT/TX2/FLOW/01JAN1977/1DAY/OBS/
 -----DSS--- ZREAD Unit  71; Vers.    5:  /BUCKNER/TX3/FLOW/01JAN1972/1DAY/OBS/
```

```
-----DSS--- ZREAD Unit  71; Vers.    3:  /BUCKNER/TX3/FLOW/01JAN1973/1DAY/OBS/
-----DSS--- ZREAD Unit  71; Vers.    3:  /BUCKNER/TX3/FLOW/01JAN1974/1DAY/OBS/
-----DSS--- ZREAD Unit  71; Vers.    3:  /BUCKNER/TX3/FLOW/01JAN1975/1DAY/OBS/
-----DSS--- ZREAD Unit  71; Vers.    3:  /BUCKNER/TX3/FLOW/01JAN1976/1DAY/OBS/
-----DSS--- ZREAD Unit  71; Vers.    3:  /BUCKNER/TX3/FLOW/01JAN1977/1DAY/OBS/
WARNING -- EXISTING VARIABLE RE-DEFINED
WARNING -- EXISTING VARIABLE RE-DEFINED
-----DSS---ZWRITE Unit  71; Vers.    4:  /CHICKLKV/TX3/FLOW/01JAN1972/1DAY/COMPUTED5/
-----DSS---ZWRITE Unit  71; Vers.    4:  /CHICKLKV/TX3/FLOW/01JAN1973/1DAY/COMPUTED5/
-----DSS---ZWRITE Unit  71; Vers.    4:  /CHICKLKV/TX3/FLOW/01JAN1974/1DAY/COMPUTED5/
-----DSS---ZWRITE Unit  71; Vers.    4:  /CHICKLKV/TX3/FLOW/01JAN1975/1DAY/COMPUTED5/
-----DSS---ZWRITE Unit  71; Vers.    4:  /CHICKLKV/TX3/FLOW/01JAN1976/1DAY/COMPUTED5/
-----DSS---ZWRITE Unit  71; Vers.    4:  /CHICKLKV/TX3/FLOW/01JAN1977/1DAY/COMPUTED5/
   -----DSS---ZCLOSE Unit:  71,   File: MREG.DSS
```



Name: **CONIC** **Conic interpolation from elevation/area table**

Use: CO TY=CONIC(TX,TB,input,output,[scale])

Interpolate values for TX using conic interpolation table TB and store the result in TY. TX and TY variables must be regular or irregular time series data. TB is a paired data variable that can be created using program DSSPD. The first paired data values contain the initial conic depth in feet or meters "x(1)" and storage in acre-feet or cubic meters below the first elevation "y(1)". The rest of the paired values "x(2--),y(2--)" contain the elevation in ft-msl or m-msl and area in acres or sq. meters. If the initial conic depth is undefined, the function will calculate one. The third and fourth parameters are used to specify the type of input (TX) and output (TY) data desired. The input type has to be either STORAGE or ELEVATION. The output type has to be either STORAGE, ELEVATION, or AREA. An optional fifth parameter can be used to specify a scale value to use with the input and output storage values. The default scale value is 1.0.

Example: The following example uses the CONIC function to convert elevation values to storage and surface area based on the conic interpolation table stored in variable PF.

Macro:

```
MACRO TCONIC
CLEAR ALL
TIME 01JAN1993 2400 10JAN1993 2400
GET TX=TESTDB.DSS:/CONIC
FUNCTION/EAHGH/STAGE/01JAN1993/1DAY/TEST/
GET PF=TESTDB.DSS:/CONIC DATA/EAHGH/ELEV-AREA////
COMP TY=CONIC(TX,PF,ELEV,STOR)
COMP TA=CONIC(TX,PF,ELEV,AREA)
SD TY UNITS=AC-FT TYPE=INST
SD TA UNITS=ACRES TYPE=INST
PUT.A TY=TESTDB.DSS:/CONIC
FUNCTION/EAHGH/STORAGE/01JAN1993/1DAY/CAL/
PUT.A TY=TESTDB.DSS:/CONIC
FUNCTION/EAHGH/AREA/01JAN1993/1DAY/CAL/
TAB TX TY TA
TAB PF
ENDMACRO
```

Execution:

```
I>!R TCONIC
  ALL VARIABLES HAVE BEEN CLEARED
    -----DSS---ZOPEN:  Existing File Opened,  File: TESTDB.DSS
```

```
                    Unit:  71;  DSS Version: 6-IA
-----DSS--- ZREAD Unit  71; Vers.    1:  /CONIC FUNCTION/EAHGH/STAGE/01JAN1993/1DAY/TEST/
-----DSS--- ZREAD Unit  71; Vers.    1:  /CONIC DATA/EAHGH/ELEV-AREA////
-----DSS---ZWRITE Unit  71; Vers.    4:  /CONIC FUNCTION/EAHGH/STORAGE/01JAN1993/1DAY/CAL/
-----DSS---ZWRITE Unit  71; Vers.    1:  /CONIC FUNCTION/EAHGH/AREA/01JAN1993/1DAY/CAL/

Variables ----        TX           TY           TA
Units     ----       FEET        AC-FT        ACRES
Time   Date
2400 01JAN1993     3336.000   486002.400     5777.025
2400 02JAN1993     3338.000   497635.600     5856.243
2400 03JAN1993     3340.000   509427.800     5936.000
2400 04JAN1993     3360.000   639317.800     7050.000
2400 05JAN1993     3380.000   793573.900     8310.000
2400 06JAN1993     3400.000   973569.400     9593.000
2400 07JAN1993     3450.000  1538441.000    13160.000
2400 08JAN1993     3500.000  2330557.000    18613.000
2400 09JAN1993     3525.000  2832292.000    21421.140
2400 10JAN1993     3565.000  3774320.000    25736.000

VARIABLE=PF
 UNITS = FEET       ACRES
 TYPES = UNT        UNT
 LABEL = SURFACE
 SHIFT =            OFFSET =            DATUM =
   X-AXIS          SURFACE
     292.7000  486000.0000          Initial conic depth in feet "x(1)" and storage in
                                    acre-feet below the first elevation "y(1)"

    3336.0000    5777.0000
    3340.0000    5936.0000
    3350.0000    6500.0000
    3360.0000    7050.0000
    3370.0000    7750.0000
    3380.0000    8310.0000
    3390.0000    9052.0000
    3400.0000    9593.0000
    3410.0000   10220.0000
    3420.0000   10920.0000
    3430.0000   11600.0000
    3440.0000   12380.0000
    3450.0000   13160.0000
    3460.0000   14110.0000
    3470.0000   15570.0000
    3480.0000   16480.0000
    3490.0000   17182.0000
    3500.0000   18613.0000
    3510.0000   19880.0000
    3520.0000   20850.0000
    3530.0000   22000.0000
    3540.0000   22970.0000
    3550.0000   24080.0000
    3560.0000   25160.0000
    3565.0000   25736.0000
```

Name: **CORR** **Compute correlation coefficients**
Use: CO SY=CORR(TX1,TX2,INDEX)

Compute the correlation coefficients, determination coefficients, and standard errors of regression between the two variables TX1 and TX2. TX1 and TX2 must be of the same type (either uniform time series or irregular time series) and contain the same number of rows or ordinates. Variable INDEX is used to set the scalar variable SY as follows:

Index	Description
1	Number of valid pairs for correlation.
2	Regression constant.
3	Regression coefficient.
4	Determination coefficient.
5	Standard error of regression.
6	Determination coefficient adjusted for degrees of freedom.
7	Standard error adjusted for degrees of freedom.

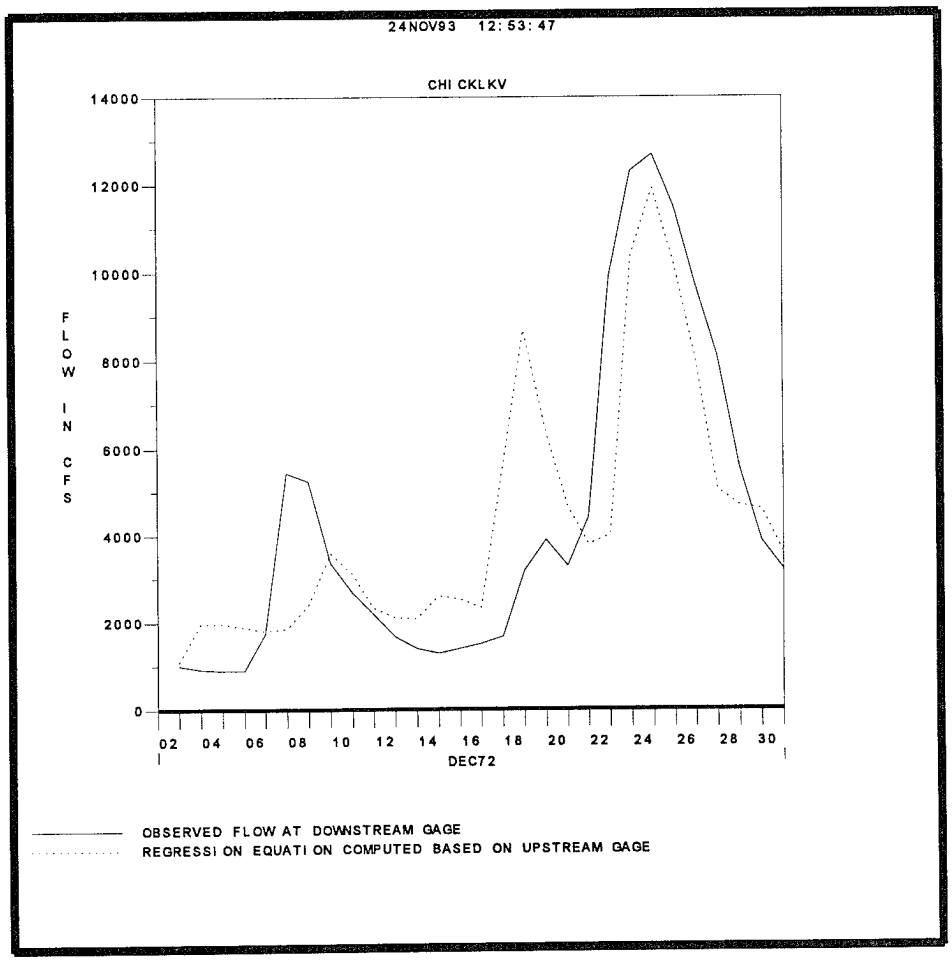
Example: The following example uses the statistical function CORR to determine a linear regression equation for a downstream flow gage dependent on the observed flow values of one upstream flow gage. Note the use of the time shift function and the TIME.N command to account for the routing time of flows from the upstream to the downstream gage.

Macro:

```
MACRO TCORR
CL ALL
TIME 0100 03DEC1972 2400 30DEC1972
GET TY=MREG.DSS:/CHICKLKV/TX1/FLOW/01JAN1972/1DAY/OBS/
TIME.N 0100 01DEC1972 2400 28DEC1972
GET TX=MREG.DSS:/CHICKENT/TX2/FLOW/01JAN1972/1DAY/OBS/
COMP TX=TSHIFT(TX,+2D)
COMP RCON=CORR(TX,TY,2)
COMP RCOE=CORR(TX,TY,3)
COMP TYCOM=RCOE*TX
COMP TYCOM=TYCOM+RCON
PUT.A TYCOM=MREG.DSS:/CHICKLKV/TX1/FLOW/01JAN1972/1DAY/REG-EQ/
ENDMACRO
```

Execution:

```
I>!R TCORR

  ALL VARIABLES HAVE BEEN CLEARED
    -----DSS---ZOPEN:  Existing File Opened,  File: MREG.DSS
                       Unit:  71;  DSS Version: 6-HD
 -----DSS--- ZREAD Unit  71; Vers.    4:  /CHICKLKV/TX1/FLOW/01JAN1972/1DAY/OBS/
 -----DSS--- ZREAD Unit  71; Vers.    3:  /CHICKENT/TX2/FLOW/01JAN1972/1DAY/OBS/
 WARNING -- EXISTING VARIABLE RE-DEFINED
  NUMBER OF VALID PAIRS FOR CORRELATION      28.000000
  REGRESSION CONSTANT    1465.112000
  REGRESSION COEFFICIENT       2.002728
  DETERMINATION COEFFICIENT   6.243303E-01
  STANDARD ERROR OF REGRESSION    2257.026000
  DETERMINATION COEFFICIENT ADJ DEGREES OF FREEDOM   6.098815E-01
  STANDARD ERROR ADJ FOR DEGREES OF FREEDOM    2300.020000
  WARNING -- EXISTING VARIABLE RE-DEFINED
 -----DSS---ZWRITE Unit  71; Vers.    5:  /CHICKLKV/TX1/FLOW/01JAN1972/1DAY/REG-EQ/
```



Name: **COS, SIN, TAN** **Trigonometric functions**
Use: CO TY=COS(TX), TY=SIN(TX), TY=TAN(TX)

Compute the trigonometric function of TX and store in TY. TX and TY can be the same variable. TX can be scalar, time series or paired data. If a value of TX is undefined resulting values of TY are undefined. If a concurrent IF condition is not satisfied, TY is unchanged.

Example: The following examples are only meant to show the proper syntax of its application. Note that the parameter in the trigonometric function must be expressed in radian and not in degrees.

Macro:

```
MACRO TCOS
!* Change 45 degrees to radians
COMP RAD=45/180
COMP RAD=RAD*3.141593
COMP SCOS=COS(RAD)
COMP SHIN=SIN(RAD)
COMP STAN=TAN(RAD)
SHOW SCOS,SSIN,STAN
$CONTINUE
ENDMACRO
```

Execution:

```
I>!R TCOS

VARIABLE=SCOS
 SCOS     =    7.071067E-01

VARIABLE=SHIN
 SHIN     =    7.071067E-01

VARIABLE=STAN
 STAN     =    1.000
```

Name: **COUNT** **Count valid or missing data values**
Use: CO SY=COUNT(TX,TYPE)

This function is used to count the valid and missing values in time series variable TX and stores it in scalar variable SY. Possible values for TYPE are: "VALID" and "MISSING".

Example: The following example of the COUNT function is only meant to show the proper syntax of its application.

Macro:

```
MACRO TCOUNT
CL ALL
TIME 0100 08APR83 1200 10APR83
GET TX=testdb.dss:/SCIOTO/MISSING   RIVER/FLOW/01APR1983/1HOUR/OBS/
COMP NVALID=COUNT(TX,VALID)
COMP NMISS=COUNT(TX,MISSING)
SHOW NVALID,NMISS
ENDMACRO
```

Execution:

```
I>!R TCOUNT
  ALL VARIABLES HAVE BEEN CLEARED
    -----DSS---ZOPEN:  Existing File Opened,  File: TESTDB.DSS
                       Unit:  71;  DSS Version: 6-IA
 -----DSS--- ZREAD Unit  71; Vers.    1:  /SCIOTO/MISSING RIVER/FLOW/01APR1983/1HOUR/OBS/

  NUMBER VALID=      55  NUMBER MISSING=       5
  NUMBER VALID=      55  NUMBER MISSING=       5
 VARIABLE=NVALID
 NVALID  =      55.000000
 VARIABLE=NMISS
 NMISS   =       5.000000

Variables ----     TX
Units     ----     CFS
Time   Date
0030 08APR1983    11995.000                                0730 09APR1983     4888.000
0130 08APR1983     5628.000                                0830 09APR1983        M
0230 08APR1983     4832.000                                0930 09APR1983        M
0330 08APR1983     3241.000                                1030 09APR1983        M
0430 08APR1983     4837.000                                1130 09APR1983        M
0530 08APR1983     3243.000                                1230 09APR1983     9761.000
0630 08APR1983     4042.000                                1330 09APR1983    11380.000
0730 08APR1983     5641.000                                1430 09APR1983     8169.000
0830 08APR1983     6445.000                                1530 09APR1983    11398.000
0930 08APR1983     4058.000
1030 08APR1983     4065.000
1130 08APR1983     5670.000
1230 08APR1983     4878.000
1330 08APR1983     3283.000
1430 08APR1983     4084.000
1530 08APR1983     4084.000
1630 08APR1983        M
1730 08APR1983     3286.000
1830 08APR1983     3284.000
1930 08APR1983     3283.000
2030 08APR1983     3281.000
2130 08APR1983     3279.000
2230 08APR1983     2476.000
2330 08APR1983     3275.000
0030 09APR1983     2473.000
0130 09APR1983     2448.000
0230 09APR1983     2448.000
0330 09APR1983     3275.000
0430 09APR1983     5682.000
0530 09APR1983     5682.000
0630 09APR1983     6486.000
```

Name: CMA **Centered moving average smoothing**
Use: CO TY=CMA(TX,NP[,CLEVEL])

Compute a centered, moving average of NP values in TX and store in TY. NP must be odd and greater than 2. The default CLEVEL is "LEVEL3" and NP/2 values at the beginning and end of TY will be undefined. If a value in TX is undefined or a concurrent IF condition is not satisfied, then the resulting TY values are the averages of one less value. TX must be a regular-interval time-series and TY and TX cannot be the same. Useful for filtering instantaneous data containing high-frequency variations. Units and type are unchanged. The following levels are available:

LEVEL1 Only valid values will be averaged. Valid values that do not have all valid values within the averaging period number will be missing. Values at the start and end of the data that do not have enough valid values to average will be assigned missing values.

LEVEL2 Only valid values will be averaged. Valid values that do not have all valid values within the averaging period number will be missing. Valid values at the start and end of the data that do not have enough valid values to average will be averaged over a reduced number of values.

LEVEL3 (Default setting) All values will be averaged based on valid values within the averaging period number. If all values are missing for the period number, the value will be missing. Values at the start and end of the data that do not have enough valid values to average will be assigned missing values.

LEVEL4 All values will be averaged based on valid values within the averaging period number. If all values are missing for the period number, the value will be missing. Values at the start and end of the data that do not have valid values to average will be averaged based on a reduced number of values.

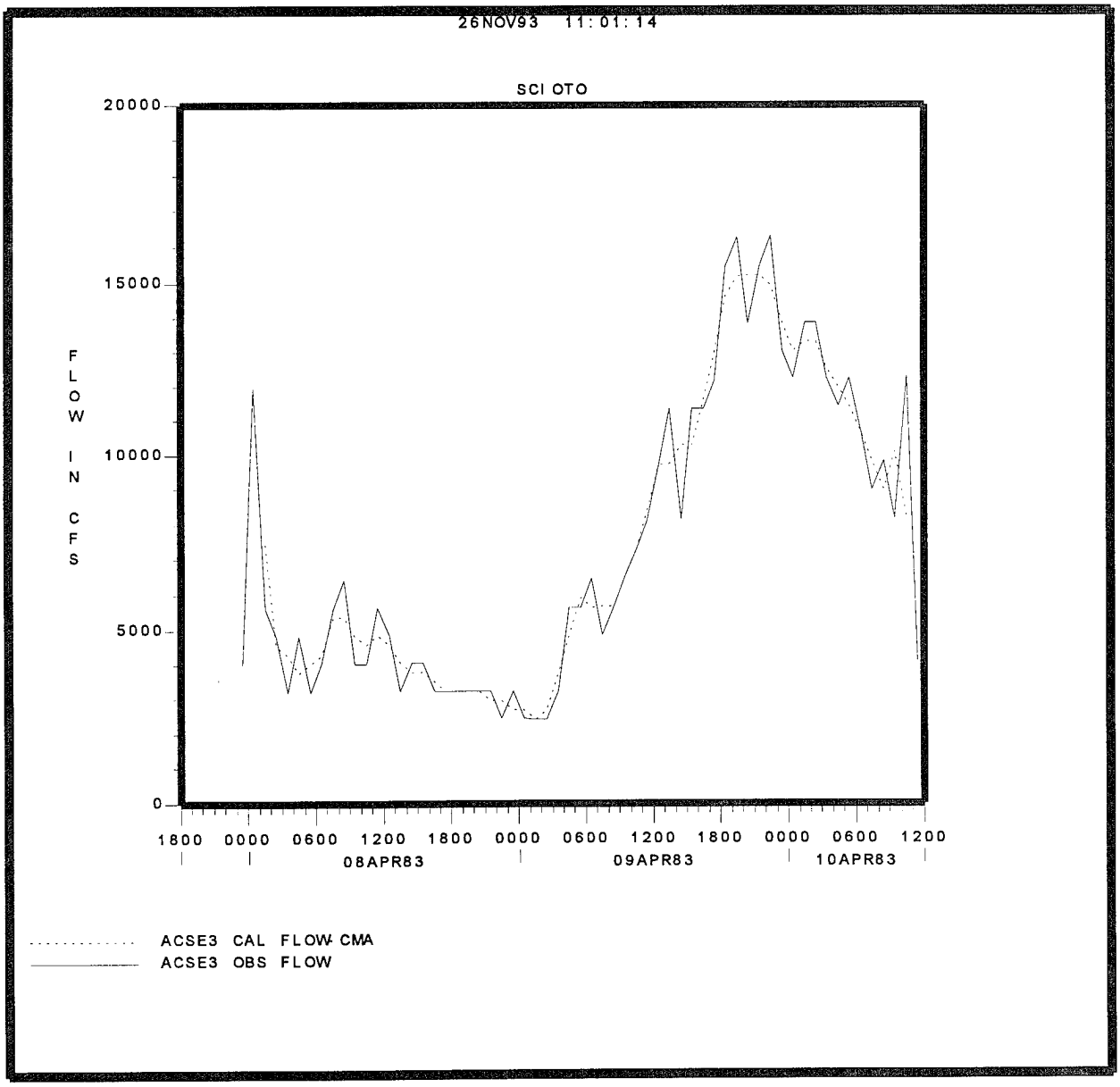
Example: The following example uses the CMA function to take flow values and smooth them using a moving average of 3.

Macro:

```
MACRO TCMA
** COMPUTE A CENTERED ,MOVING AVERAGE OF NUMPOINT VALUES IN FLOW AND PUT.A
CL ALL
TIME 08APR1983 0100 10APR1983 1200
GET FLOW=testdb.dss:/SCIOTO/ACSE3/FLOW/01APR1983/1HOUR/OBS/
COMP AVE=CMA(FLOW,3,LEVEL3)
PUT.A AVE=C=FLOW-CMA F=CAL
$CONTINUE INCASE OF ERROR
ENDMACRO
```

Execution:

```
I>!R TCMA
 ALL VARIABLES HAVE BEEN CLEARED
   -----DSS---ZOPEN:  Existing File Opened,  File: TESTDB.DSS
                      Unit:  71;  DSS Version: 6-IA
-----DSS--- ZREAD Unit  71; Vers.    3:  /SCIOTO/ACSE3/FLOW/01APR1983/1HOUR/OBS/
-----DSS---ZWRITE Unit  71; Vers.    1:  /SCIOTO/ACSE3/FLOW-CMA/01APR1983/1HOUR/CAL/
```



Name: **DDT** **Differences per unit time**
Use: CO TY=DDT(TX)

Computes the successive differences in TX per time period:

$$TY(t)=(TX(t)-TX(t-1))/DT$$

where DT is the time difference in days between t and t-1. If a value of TX is undefined, TY is undefined. If a concurrent IF condition is not satisfied, TY is unchanged. TY and TX cannot be the same variable. TY is assigned the type 'PER-AVER.' The units of TY are undefined and should be set by the user with the SD command. An example of the use of DDT is the computation of reservoir inflow from outflow and the change in storage, where the change in storage is transformed to average flow volume per day by the function. The conversion factor for storage to flow is 0.50416 when the storage change is in ac-ft day.

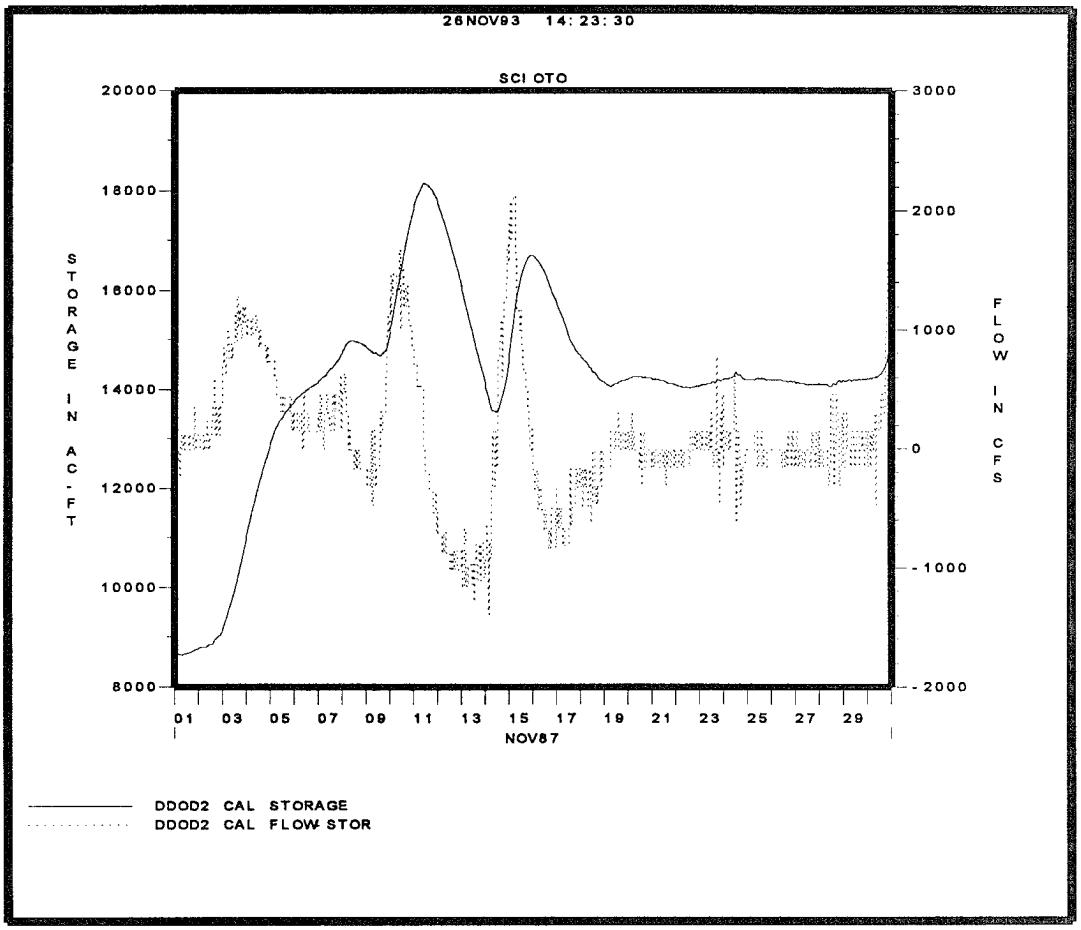
Example: The following example uses the DDT function to convert storage into change in storage per day. The macro in total converts pool elevation to storage, tailwater stage to outflow, and computes inflow to a reservoir.

Macro:

```
MACRO TDDT
CL ALL
TI 01NOV87 0100 30NOV87 2400
GET PELEV=MASTDB:/SCIOTO/DDOD2/ELEV/01APR1983/1HOUR/OBS/
GET PELST=/SCIOTO/DDOD2/ELEV-STOR////
COM STOR=RTABLE(PELEV,PELST)
SD STOR UNITS=AC-FT TYPE=INST-VAL
PUT.A STOR=/SCIOTO/DDOD2/STORAGE/01APR1983/1HOUR/CAL/
COM DSTIN=DDT(STOR)
COM DSTOR=DSTIN*.50416
SD DSTOR UNITS=CFS TYPE=PER-AVE
PUT.A DSTOR=/SCIOTO/DDOD2/FLOW-STOR/01APR1983/1HOUR/CAL/
CLEAR PELEV PELST
GET TELEV=/SCIOTO/DDOOF/ELEV/01APR1983/1HOUR/OBS/
GET OSTFL=/SCIOTO/DDOOF/ELEV-FLOW///USGS TABLE NO.10/
COM QOINST=RTABLE(TELEV,OSTFL)
CLEAR TELEV DSTIN
COM QOUT=TS2(QOINST,1H,0H)
COM QIN=DSTOR+QOUT
SD QIN U=CFS TY=PER-AVER
PUT.A QIN=MASTDB:/SCIOTO/DDOD2/FLOW-RES IN/01APR1983/1HOUR/CAL/
$CONTINUE INCASE OF ERROR
ENDMACRO
```

Execution:

```
I>!R TDDT
  ALL VARIABLES HAVE BEEN CLEARED
    -----DSS---ZOPEN:  Existing File Opened,  File: MASTDB.DSS
                       Unit:  71;  DSS Version: 6-GU
 -----DSS--- ZREAD Unit  71; Vers.  140:  /SCIOTO/DDOD2/ELEV/01NOV1987/1HOUR/OBS/
 -----DSS--- ZREAD Unit  71; Vers.    1:  /SCIOTO/DDOD2/ELEV-STOR////
 LOGLOG TRANSFORMATION
 OFFSET =     890.00     SHIFT =        .00     DATUM =        .00
 -----DSS---ZWRITE Unit  71; Vers.    1:  /SCIOTO/DDOD2/STORAGE/01NOV1987/1HOUR/CAL/
 -----DSS---ZWRITE Unit  71; Vers.    1:  /SCIOTO/DDOD2/FLOW-STOR/01NOV1987/1HOUR/CAL/
  PELEV CLEARED
  PELST CLEARED
 -----DSS--- ZREAD Unit  71; Vers.  140:  /SCIOTO/DDOOF/ELEV/01NOV1987/1HOUR/OBS/
 -----DSS--- ZREAD Unit  71; Vers.    1:  /SCIOTO/DDOOF/ELEV-FLOW///USGS TABLE NO.10/
 LOGLOG TRANSFORMATION
 OFFSET =      79.00     SHIFT =        .00     DATUM =     800.00
  TELEV CLEARED
  DSTIN CLEARED
 -----DSS---ZWRITE Unit  71; Vers.    1:  /SCIOTO/DDOD2/FLOW-RES IN/01NOV1987/1HOUR/CAL/
```



Name: **DECPAR** **Decaying basin wetness parameter**
Use: CO TY=DECPAR(TY,TZ,R)

Compute a time-series of parameters TY as a function of TZ and R:

TY(t)=R*TY(t-1)+TZ(t)

where R is the decay rate and TZ is precipitation. R is less than 1. The function extends TY: the first value in the series TY is used as the starting value, and any other TY are computed in the sequence. If the first value in TY is undefined, it is assumed to be zero. The first TZ value is ignored. TY and TZ must be regular-interval time series with identical times. R should be appropriate for the interval. The type and units of TY are unchanged. An IF condition has no effect.

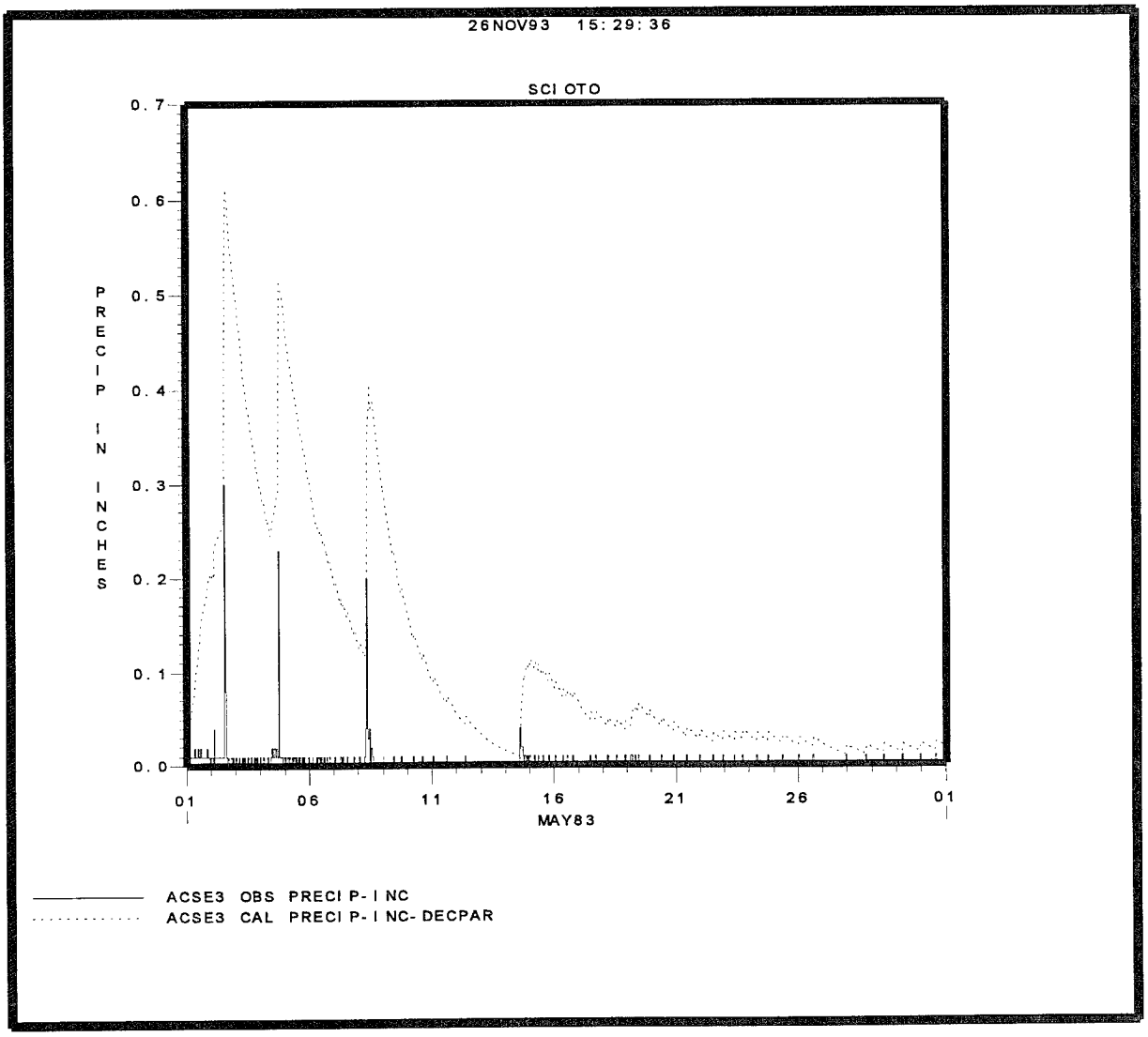
Example: The following example uses the DECPAR function to create a set of decaying basin wetness parameters.

Macro:

```
MACRO TDECPAR
** PAR(t)=RATE*PAR(t-1) + PPT(t)   DECAYING WETNESS PARAMETER
CL ALL
TI 01MAY83 0100 31MAY83 2400
GET PPT=testdb.dss:/SCIOTO/ACSE3/PRECIP-INC/01MAY1983/1HOUR/OBS/
GET PAR=testdb.dss:/SCIOTO/ACSE3/PRECIP-INC/01MAY1983/1HOUR/OBS/
COM PAR=DECPAR(PAR,PPT,.97)
PUT.A PAR=C=PRECIP-INC-DECPAR,F=CAL
$CONTINUE INCASE OF ERROR
ENDMACRO
```

Execution:

```
I>!R TDECPAR
  ALL VARIABLES HAVE BEEN CLEARED
    -----DSS---ZOPEN:  Existing File Opened,  File: TESTDB.DSS
                       Unit:  71;  DSS Version: 6-IA
 -----DSS--- ZREAD Unit  71; Vers.   22:  /SCIOTO/ACSE3/PRECIP-INC/01MAY1983/1HOUR/OBS/
 -----DSS--- ZREAD Unit  71; Vers.   22:  /SCIOTO/ACSE3/PRECIP-INC/01MAY1983/1HOUR/OBS/
 TYPE FOR PARAM NO. 3 IS =           4
 WARNING -- EXISTING VARIABLE RE-DEFINED
 R =    9.700000E-01
 -----DSS---ZWRITE Unit  71; Vers.    1:
/SCIOTO/ACSE3/PRECIP-INC-DECPAR/01MAY1983/1HOUR/CAL/
```



Name: **DIFF** **Successive differences**
Use: CO TY=DIFF(TX)

Compute the successive differences of TX and store in TY. TX and TY cannot be the same variable. TX must be of type INST-VAL or INST-CUM. If a value of TX is undefined, resulting values of TY are undefined. If a concurrent IF condition is not satisfied, TY is unchanged. The type and units of TY are undefined and should be set using the SD command.

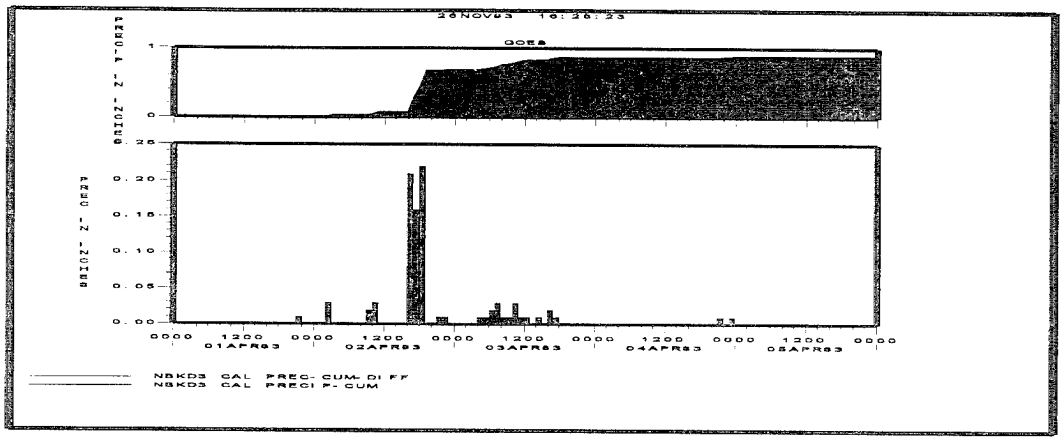
Example: The following example uses the DIFF function to take cumulative precipitation and convert it into incremental precipitation.

Macro:

```
MACRO TDIFF
** SUCCESSIVE DIFFERENCE
CL ALL
TIME 01APR83 0100 05APR83 2400
GET PPT=testdb.dss:/GOES/NBKD3/PRECIP-CUM/01APR1983/1HOUR/CAL/
COM PDIFF=DIFF(PPT)
SD PDIFF UNITS=INCHES TYPE=PER-CUM
PUT.A PDIFF=C=PREC-CUM-DIFF,F=CAL
$CONTINUE INCASE OF ERROR
ENDMACRO
```

Execution:

```
I>!R TDIFF
  ALL VARIABLES HAVE BEEN CLEARED
    -----DSS---ZOPEN:  Existing File Opened,  File: TESTDB.DSS
                       Unit:  71;  DSS Version: 6-IA
 -----DSS--- ZREAD Unit  71; Vers.    2:  /GOES/NBKD3/PRECIP-CUM/01APR1983/1HOUR/CAL/
 -----DSS---ZWRITE Unit  71; Vers.    1:  /GOES/NBKD3/PREC-CUM-DIFF/01APR1983/1HOUR/CAL/
```



Name: **ESTLIN** **Estimate values for missing data**
Use: CO TY=ESTLIN(TX,NMAX,QFLAG)

Linearly interpolate estimates for values in TX with quality flags equal to or lesser in quality to QFLAG and place the results in TY. Do not estimate more that NMAX continuous missing values. Possible flags, in order of decreasing quality, are: Q = questionable and M = missing. An IF condition has no effect. TX must be a time-series, and TX and TY may be the same variable. Type and units of TY are the same as TX.

Example: The following example uses the ESTLIN function to linearly interpolate values for missing values. The maximum number of missing consecutive values it will interpolate values for is specified as four.

Macro:

```
MACRO TESTLIN
** Estimate values for missing flow data. Do not attempt to estimate
** more than 4 values and only consider missing, not questionable values.

CL ALL
TI 24MAY83 1600 25MAY83 1600
GET FLOW=testdb.dss:/SCIOTO/BMRI2/FLOW/01MAY1983/1HOUR/OBS/
COM EFLOW=ESTLIN(FLOW,4,M)
PUT.A EFLOW=B=TESTLIN,F=CAL
TAB.F FLOW EFLOW
$CONTINUE INCASE OF ERROR
ENDMACRO
```

Execution:

```
I>!R TESTLIN
 ALL VARIABLES HAVE BEEN CLEARED
   -----DSS---ZOPEN:  Existing File Opened,  File: TESTDB.DSS
                      Unit:  71;  DSS Version: 6-IA
-----DSS--- ZREAD Unit  71; Vers.   24:  /SCIOTO/BMRI2/FLOW/01MAY1983/1HOUR/OBS/

 ** WARNING: FLAGGED DATA AT BEGINNING OF RECORD
 ** WARNING: FLAGGED DATA AT END OF RECORD
 ** WARNING: TOO MANY MISSING POINTS BETWEEN NON MISSING VALUES

-----DSS---ZWRITE Unit  71; Vers.    6:  /SCIOTO/TESTLIN/FLOW/01MAY1983/1HOUR/CAL/

Variables ----      FLOW         EFLOW
Units     ----       CFS           CFS
Time   Date
1600 24MAY1983        M            M
1700 24MAY1983       93.934       93.934
1800 24MAY1983       92.643       92.643
1900 24MAY1983       91.676       91.676
2000 24MAY1983       90.710       90.710
2100 24MAY1983       90.388       90.388
2200 24MAY1983       88.781       88.781
2300 24MAY1983       87.817       87.817
2400 24MAY1983       87.817       87.817
0100 25MAY1983        M            M
0200 25MAY1983        M            M
0300 25MAY1983        M            M
0400 25MAY1983        M            M
0500 25MAY1983        M            M
0600 25MAY1983       82.696       82.696
0700 25MAY1983       82.376       82.376
0800 25MAY1983       81.419       81.419
0900 25MAY1983       81.101       81.101
1000 25MAY1983        M           80.464
1100 25MAY1983       79.827       79.827
1200 25MAY1983       79.508       79.508
1300 25MAY1983       79.190       79.190
1400 25MAY1983       78.554       78.554
1500 25MAY1983       77.602       77.602
1600 25MAY1983        M            M
```

Name: **ESTPPT** **Estimate values for missing precipitation data**
Use: CO TY=ESTPPT(TX,NMAX,QFLAG)

Linearly interpolate estimates for cumulative precipitation values in TX with quality flags equal to or lesser in quality to QFLAG and place the results in TY. Possible flags, in order of decreasing quality, are: Q = questionable and M = missing. If the values bracketing the missing period are increasing with time, do not estimate more that NMAX continuous missing values. If the values bracketing the missing period are equal, then estimate any number of missing values. If the values bracketing the missing period are decreasing with time, do not estimate any missing values. TX must be a data type of INST-CUM. An IF condition has no effect.

Example: The following example uses the ESTPPT function to take cumulative precipitation and linearly interpolate values for missing values. The maximum number of missing consecutive values it will interpolate values for is specified as five.

Macro:

```
MACRO TESTPPT
CL ALL
TI 01APR83 0100 03APR83 1000
GET PPT=testdb.dss:/SCIOTO/ACSE3/PRECIP/01APR1983/1HOUR/OBS/
COM EPPT=ESTPPT(PPT,5,M)
PUT.A EPPT=B=TESTPPT,F=CAL
TAB.F PPT EPPT
$CONTINUE INCASE OF ERROR
ENDMACRO
```

Execution:

```
I>!R TESTPPT
 ALL VARIABLES HAVE BEEN CLEARED
   -----DSS---ZOPEN:  Existing File Opened,  File: TESTDB.DSS
                      Unit:  71;  DSS Version: 6-IA
-----DSS--- ZREAD Unit  71; Vers.  142:  /SCIOTO/ACSE3/PRECIP/01APR1983/1HOUR/OBS/
 ** WARNING: SOME POINTS WERE NOT ESTIMATED
-----DSS---ZWRITE Unit  71; Vers.    3:  /SCIOTO/TESTPPT/PRECIP/01APR1983/1HOUR/CAL/

Variables ----      PPT          EPPT
Units     ----    INCHES        INCHES
Time   Date
0100 01APR1983  13.630        13.630          0100 02APR1983  13.630        13.630          0100 03APR1983  14.720        14.720
0200 01APR1983  13.630        13.630          0200 02APR1983  13.630        13.630          0200 03APR1983  14.750        14.750
0300 01APR1983  13.630        13.630          0300 02APR1983  13.630        13.630          0300 03APR1983  M             14.780
0400 01APR1983  M             13.630          0400 02APR1983  13.630        13.630          0400 03APR1983  M             14.810
0500 01APR1983  M             13.630          0500 02APR1983  13.630        13.630          0500 03APR1983  M             14.840
0600 01APR1983  M             13.630          0600 02APR1983  13.630        13.630          0600 03APR1983  14.870        14.870
0700 01APR1983  M             13.630          0700 02APR1983  13.630        13.630          0700 03APR1983  14.900        14.900
0800 01APR1983  M             13.630          0800 02APR1983  13.630        13.630          0800 03APR1983  14.920        14.920
0900 01APR1983  M             13.630          0900 02APR1983  13.630        13.630          0900 03APR1983  14.950        14.950
1000 01APR1983  13.630        13.630          1000 02APR1983  13.650        13.650          1000 03APR1983  14.970        14.970
1100 01APR1983  13.630        13.630          1100 02APR1983  M             M
1200 01APR1983  13.630        13.630          1200 02APR1983  M             M
1300 01APR1983  13.630        13.630          1300 02APR1983  M             M
1400 01APR1983  13.630        13.630          1400 02APR1983  M             M
1500 01APR1983  13.630        13.630          1500 02APR1983  M             M
1600 01APR1983  13.630        13.630          1600 02APR1983  M             M
1700 01APR1983  13.630        13.630          1700 02APR1983  14.410        14.410
1800 01APR1983  13.630        13.630          1800 02APR1983  14.450        14.450
1900 01APR1983  13.630        13.630          1900 02APR1983  14.490        14.490
2000 01APR1983  13.630        13.630          2000 02APR1983  14.540        14.540
2100 01APR1983  13.630        13.630          2100 02APR1983  14.580        14.580
2200 01APR1983  13.630        13.630          2200 02APR1983  14.620        14.620
2300 01APR1983  13.630        13.630          2300 02APR1983  14.650        14.650
2400 01APR1983  13.630        13.630          2400 02APR1983  14.690        14.690
```

Name: **FMA** **Forward moving average**
Use: CO TY=FMA(TX,NP)

Compute a moving average of the last NP values in TX and store in TY. NP must greater than 2. NP values at the beginning of TY will be missing. If a value in TX is missing, the value is not used for computing TY, and the average is over one less value. At least 2 values of TX must be defined, else TY is missing. Useful for computing flow durations. Also, may be used to determine parameters for use in the SCRN2 screening function. Units and type are unchanged.

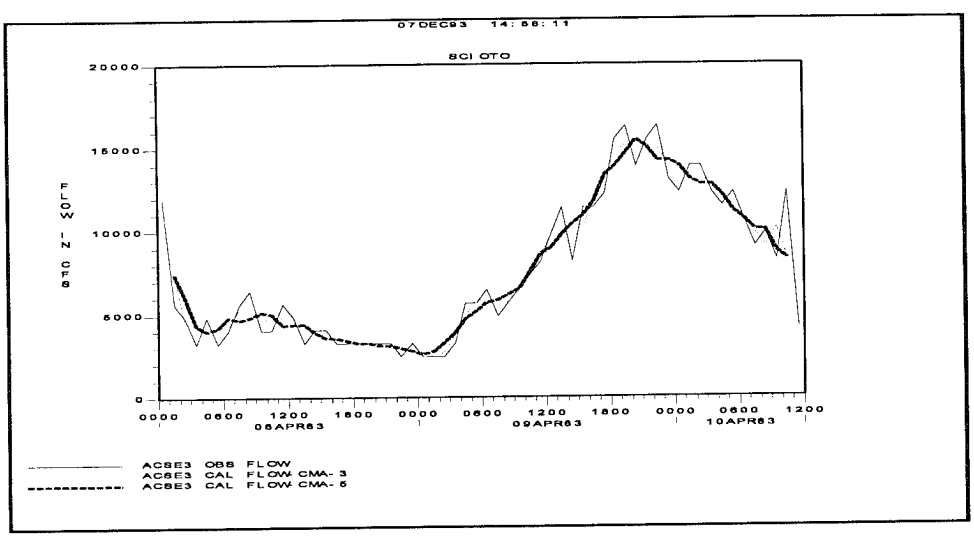
Example: The following example uses the forward moving average function to take flow values and smooth them using a forward moving average of 3 and 5.

Macro:

```
MACRO TFMA
CL ALL
TIME 08APR1983 0100 10APR1983 1200
GET FLOW=testdb.dss:/SCIOTO/ACSE3/FLOW/01APR1983/1HOUR/OBS/
COMP AVE=FMA(FLOW,3)
PUT.A AVE=C=FLOW-FMA-3 F=CAL
COMP AVE=FMA(FLOW,5)
PUT.A AVE=C=FLOW-FMA-5 F=CAL
$CONTINUE INCASE OF ERROR
ENDMACRO
```

Execution:

```
I>!R TFMA
 ALL VARIABLES HAVE BEEN CLEARED
   -----DSS---ZOPEN:  Existing File Opened,  File: TESTDB.DSS
                      Unit:  71;  DSS Version: 6-IA
-----DSS--- ZREAD Unit  71; Vers.    3:  /SCIOTO/ACSE3/FLOW/01APR1983/1HOUR/OBS/
-----DSS---ZWRITE Unit  71; Vers.    1:  /SCIOTO/ACSE3/FLOW-FMA-3/01APR1983/1HOUR/CAL/
WARNING -- EXISTING VARIABLE RE-DEFINED
-----DSS---ZWRITE Unit  71; Vers.    1:  /SCIOTO/ACSE3/FLOW-FMA-5/01APR1983/1HOUR/CAL/
```



Name: GENTSR **Generate a regular interval time series**
Use: CO TY=GENTSR(DT,TOFF,Y0,QFLAG)

Generate a new, regular-interval time series TY with a time interval of DT, a time interval offset of TOFF, a constant value Y0, and all values internally flagged with QFLAG. TOFF is time from the beginning of the standard interval to the actual time of the data. DT and TOFF are expressed as units of time as nnT, where T is M for minutes, H for hours, and D for days. Possible flags are: N = none, M = missing, E = estimated, and Q = questionable. If Y0 is -901., then the flag is automatically M. Units and type must be set independently using the SD command. An IF condition has no effect.

Example: The following example uses the GENTSR function to generate a regular time series that has all its values set to missing. The generated time series is then merged with an irregular time series that reports at hourly intervals, but not always every hour. The following macro inserts missing values into the irregular time series and then uses the TS1 function to convert the irregular time series to regular.

Macro:

```
MACRO TGENTSR
CL ALL
TIME 01MAY1983 0100 02MAY1983 0100
COM FLOWG=GENTSR(1H,0H,-901.,M)
GET FLOW=testdb.dss:/SCIOTO/BMRI2/FLOW/01MAY1983/IR-MONTH/OBS-I/
COM FLOWM=MRG(FLOW,FLOWG,N)
COM FLOWR=TS1(FLOWM,1H,0H)
TAB.F FLOW FLOWG FLOWR
PUT.A FLOWR=testdb.dss:/SCIOTO/BMRI2/FLOW/01MAY1983/1HOUR/CAL-I/
$CONTINUE INCASE OF ERROR
ENDMACRO
```

Execution:

```
I>!R TGENTSR
 ALL VARIABLES HAVE BEEN CLEARED
*** WARNING : QUALITY FLAG RESET TO NONE
   -----DSS---ZOPEN:  Existing File Opened,  File: TESTDB.DSS
                      Unit:  71;  DSS Version: 6-IA

Variables ----        FLOW          FLOWG         FLOWR
Units     ----          CFS                                CFS
Time   Date
0100 01MAY1983   100.735     M       100.735
0200 01MAY1983   100.085     M       100.085
0300 01MAY1983    99.436     M        99.436
0400 01MAY1983     -         M         M
0500 01MAY1983    98.140     M        98.140
0600 01MAY1983     -         M         M
0700 01MAY1983    96.844     M        96.844
0800 01MAY1983    96.196     M        96.196
0900 01MAY1983    95.873     M        95.873
1000 01MAY1983     -         M         M
1100 01MAY1983    97.491     M        97.491
1200 01MAY1983   102.035     M       102.035
1300 01MAY1983   106.597     M       106.597
1400 01MAY1983     -         M         M
1500 01MAY1983   109.211     M       109.211
1600 01MAY1983   111.504     M       111.504
1700 01MAY1983   112.815     M       112.815
1800 01MAY1983     -         M         M
1900 01MAY1983   115.443     M       115.443
2000 01MAY1983   115.443     M       115.443
2100 01MAY1983   114.457     M       114.457
2200 01MAY1983     -         M         M
2300 01MAY1983   112.815     M       112.815
2400 01MAY1983   112.487     M       112.487
0100 02MAY1983   111.504     M       111.504
```

Name: **INT** **Truncate to whole numbers**

Use: CO TY=INT(TX)

Truncate to a whole number TX and store in TY. TX and TY can be the same variable. TX can be scalar, time series or paired data. If a value of TX is undefined resulting values of TY are undefined. If a concurrent IF condition is not satisfied, TY is unchanged.

Name: **LAST** **Last valid value in a time series**

Use: CO SY=LAST(TX)

Find the last valid value in time-series TX and place it in scalar SY. Ignores missing values or values concurrent with an unsatisfied IF condition.

Name: **LOG** **Natural log base "e"**

Use: CO TY=LOG(TX)

Compute the natural log base e of TX and store in TY. TX and TY can be the same variable. TX can be scalar, time series or paired data. If a value of TX is undefined, resulting values of TY are undefined. If a concurrent IF condition is not satisfied, TY is unchanged. If a value in TX is 0 or negative, the value in TY will be set to missing.

Name: **LOG10** **Log base 10**

Use: CO TY=LOG10(TX)

Compute the log base 10 of TX and store in TY. TX and TY can be the same variable. TX can be scalar, time series or paired data. If a value of TX is undefined resulting values of TY are undefined. If a concurrent IF condition is not satisfied, TY is unchanged. If a value in TX is 0 or negative, the value in TY will be set to missing.

Example: The following demonstrate the use of the LAST, INT, LOG, and LOG10 functions.

Macro:

```
MACRO TLAST
CL ALL
TIME 01APR83 0100 05APR83 2400
GET PPT=testdb.dss:/GOES/NBKD3/PRECIP-INC/01ARP1983/1HOUR/OBS/
COMP PACC=ACC(PPT)
COMP SUM=LAST(PACC)
COMP SUMI=INT(SUM)
COMP SLOG=LOG(SUM)
COMP SLOG10=LOG10(SUM)
SHOW SUM SUMI SLOG SLOG10
$CONTINUE INCASE OF ERROR
ENDMACRO
```

Execution:

```
I>!R TLAST
 ALL VARIABLES HAVE BEEN CLEARED
   -----DSS---ZOPEN:  Existing File Opened,  File: TESTDB.DSS
                      Unit:  71;  DSS Version: 6-IA
-----DSS--- ZREAD Unit  71; Vers.  145:  /GOES/NBKD3/PRECIP-INC/01APR1983/1HOUR/OBS/

 FINAL ACCUMULATED VALUE =   8.999997E-01
MAX=        .900  MIN=        .000  MEAN=        .566  LAST VALUE=        .900
VARIABLE=SUM
SUM     =   8.999997E-01
VARIABLE=SUMI
SUMI    =   0.000000E+00
VARIABLE=SLOG
SLOG    =  -1.053608E-01
VARIABLE=SLOG10
SLOG10  =  -4.575762E-02
```

Name: **MATE** **Generate data pairs from two time-series**
Use: CO PF=MATE(TX,TY,SORT/NOSORT)

Derive a paired-function PF by pairing values in the time series TX and TY. TX and TY values must have identical times. (Functions TS1 and TS4 may be useful in conjunction with MATE). The paired data are sorted into ascending order of TX if SORT is specified. The units of PF are respectively the same as those of TX and TY. The types of PF are undefined and must be set with the SD command. An IF condition has no effect.

Example: The following example uses the MATE function to take observed stage and flow values that occur at the same time and combine them into a paired data variable. Two computations are performed on the same data in this example to demonstrate the option of generating a sorted and an unsorted paired data variable.

Macro:

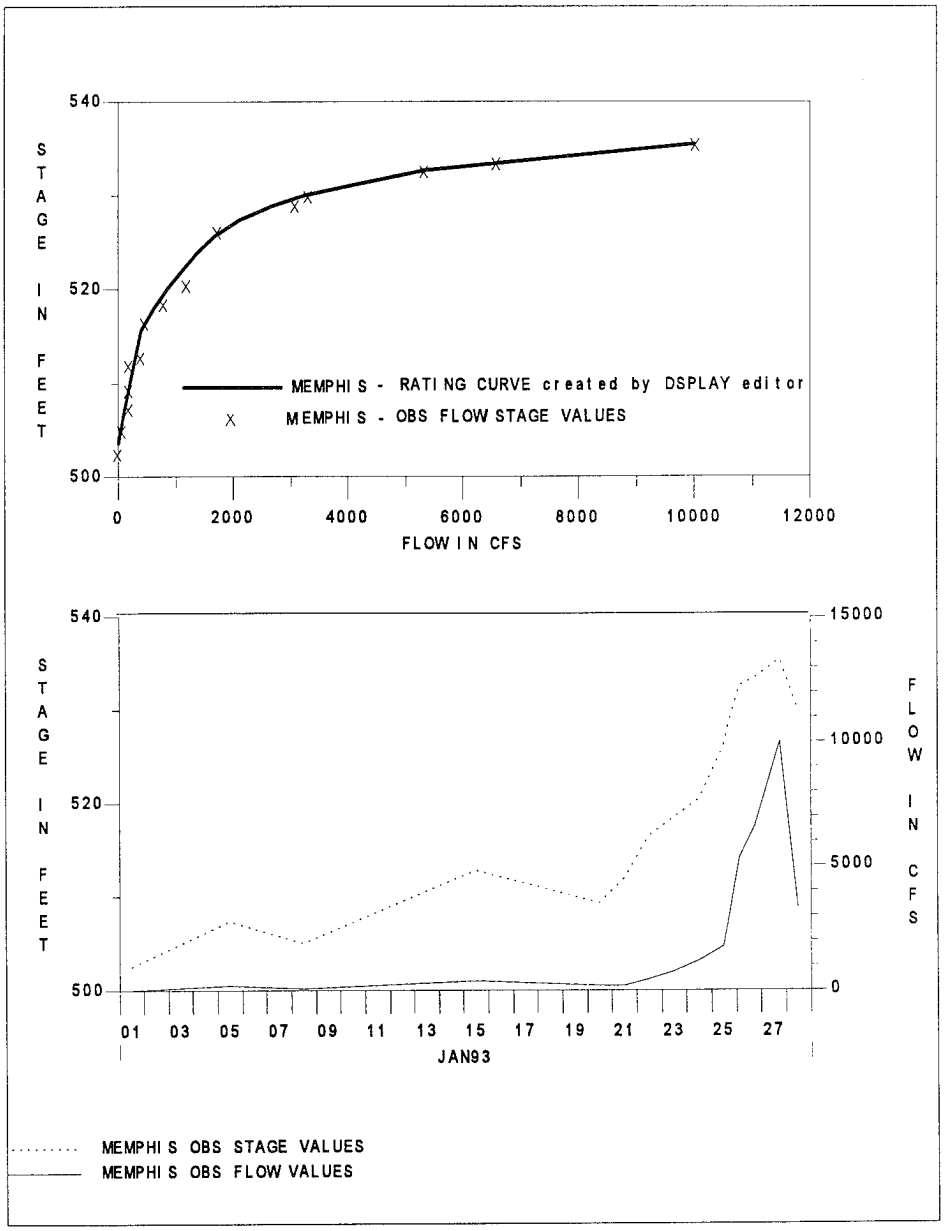
```
MACRO TMATE
** COMPUTE DATA PAIRS FROM TWO TIME-SERIES
CL ALL
TIME 01JAN93 0100 30JAN93 2400
GET STAGE=testdb.dss:/DAVIS/MEMPHIS/STAGE/01JAN1993/IR-MONTH/OBS/
GET FLOW=testdb.dss:/DAVIS/MEMPHIS/FLOW/01JAN1993/IR-MONTH/OBS/
COMP PAIRS=MATE(FLOW,STAGE,SORT)
COMP PAIRN=MATE(FLOW,STAGE,NOSORT)
SD PAIRS LABEL=RATING-S
PUT.A PAIRS=testdb.dss:/DAVIS/MEMPHIS/FLOW-STAGE/01JAN1993/SORTED/CAL/
SD PAIRN LABEL=RATING-N
PUT.A PAIRN=testdb.dss:/DAVIS/MEMPHIS/FLOW-STAGE/01JAN1993/NOSORTED/CAL/
TAB.F FLOW STAGE
TAB.F PAIRS PAIRN
$CONTINUE INCASE OF ERROR
ENDMACRO
```

Execution:

```
I>!R TMATE
 ALL VARIABLES HAVE BEEN CLEARED
   -----DSS---ZOPEN:  Existing File Opened,  File: TESTDB.DSS
                      Unit:  71;  DSS Version: 6-IA
-----DSS---ZWRITE Unit  71; Vers.    1:  /DAVIS/MEMPHIS/FLOW-STAGE/01JAN1993/SORTED/CAL/
-----DSS---ZWRITE Unit  71; Vers.    1:  /DAVIS/MEMPHIS/FLOW-STAGE/01JAN1993/NOSORTED/CAL/
```

Variables ----		FLOW	STAGE
Units	----	CFS	FEET
Time	Date		
1200	01JAN1993	10.000	502.500
1200	05JAN1993	200.000	507.400
1200	08JAN1993	85.000	505.100
1200	15JAN1993	400.000	512.900
1200	20JAN1993	205.000	509.400
1200	21JAN1993	200.000	512.000
1200	22JAN1993	475.000	516.500
1200	23JAN1993	800.000	518.500
1200	24JAN1993	1200.000	520.500
1200	25JAN1993	1750.000	526.300
1800	25JAN1993	3100.000	529.000
0300	26JAN1993	5350.000	532.800
1800	26JAN1993	6600.000	533.600
1800	27JAN1993	10035.000	535.500
1200	28JAN1993	3330.000	530.000

```
VARIABLE=PAIRS
 UNITS = CFS       FEET

TYPES = UNT        UNT
 LABEL = RATING-S
 SHIFT = OFFSET = DATUM =

   X-AXIS          RATING-S
      10.0000      502.5000
      85.0000      505.1000
     200.0000      507.4000
     200.0000      512.0000
     205.0000      509.4000
     400.0000      512.9000
     475.0000      516.5000
     800.0000      518.5000
    1200.0000      520.5000
    1750.0000      526.3000
    3100.0000      529.0000
    3330.0000      530.0000
    5350.0000      532.8000
    6600.0000      533.6000
   10035.0000      535.5000

VARIABLE=PAIRN
 UNITS = CFS       FEET
 TYPES = UNT       UNT
 LABEL = RATING-N
 SHIFT = OFFSET = DATUM =

   X-AXIS          RATING-N
      10.0000      502.5000
     200.0000      507.4000
      85.0000      505.1000
     400.0000      512.9000
     205.0000      509.4000
     200.0000      512.0000
     475.0000      516.5000
     800.0000      518.5000
    1200.0000      520.5000
    1750.0000      526.3000
    3100.0000      529.0000
    5350.0000      532.8000
    6600.0000      533.6000
   10035.0000      535.5000
    3330.0000      530.0000
```



Name: **MAX** **Maximum value in a time series**
Use: CO SY=MAX(TX)

Find the maximum value in time-series TX and place it in scalar SY. Ignore missing values or values concurrent with an unsatisfied IF condition.

Name: **MEAN** **Mean value in a time series**
Use: CO SY=MEAN(TX)

Compute the mean value in time-series TX and place the result in scalar SY. Ignore missing values or values concurrent with an unsatisfied IF condition.

Name: **MIN** **Minimum value in a time series**
Use: CO SY=MIN(TX)

Find the minimum value in time-series TX and place the result in scalar SY. Ignore missing values or values concurrent with an unsatisfied IF condition.

Example: The following demonstrate the use of the MAX, MIN, and MEAN functions.

Macro:

```
MACRO TMEAN
CL ALL
TIME 09APR1983 1200 10APR1983 1200
GET FLOW=testdb.dss:/SCIOTO/ACSE3/FLOW/01APR1983/1HOUR/OBS/
COM FMIN=MIN(FLOW)
COM FMAX=MAX(FLOW)
COM FMEAN=MEAN(FLOW)
SHOW FMIN FMAX FMEAN
TAB.F FLOW
$CONTINUE INCASE OF ERROR
ENDMACRO
```

Execution:

```
I>!R TMEAN
 ALL VARIABLES HAVE BEEN CLEARED
   -----DSS---ZOPEN:  Existing File Opened,  File: TESTDB.DSS
                      Unit:  71;  DSS Version: 6-IA
-----DSS--- ZREAD Unit  71; Vers.    3:  /SCIOTO/ACSE3/FLOW/01APR1983/1HOUR/OBS/

MAX=   16321.000  MIN=    4156.000  MEAN=   11737.840  LAST VALUE=   4156.000
MAX=   16321.000  MIN=    4156.000  MEAN=   11737.840  LAST VALUE=   4156.000
MAX=   16321.000  MIN=    4156.000  MEAN=   11737.840  LAST VALUE=   4156.000

VARIABLE=FMIN
FMIN  =  4156.000000
VARIABLE=FMAX
FMAX  = 16321.000000
VARIABLE=FMEAN
FMEAN = 11737.840000

Variables ----     FLOW
Units     ----      CFS
Time   Date
1130 09APR1983     8146.000
1230 09APR1983     9761.000
1330 09APR1983    11380.000
1430 09APR1983     8169.000
1530 09APR1983    11398.000
1630 09APR1983    11403.000
1730 09APR1983    12217.000
1830 09APR1983    15466.000
1930 09APR1983    16282.000
2030 09APR1983    13872.000
2130 09APR1983    15501.000
2230 09APR1983    16321.000
2330 09APR1983    13093.000
0030 10APR1983    12278.000
0130 10APR1983    13902.000
0230 10APR1983    13908.000
0330 10APR1983    12287.000
0430 10APR1983    11479.000
0530 10APR1983    12294.000
0630 10APR1983    10669.000
0730 10APR1983     9043.000
0830 10APR1983     9863.000
0930 10APR1983     8234.000
1030 10APR1983    12324.000
1130 10APR1983     4156.000
```

Name: **MREG** **Multiple linear regression function**
Use: CO PF=MREG(TY,TX1,TX2,...,TXn,min,max)

This function is used to determine the coefficients (Bn) of a linear regression equation of the general form (Y = B0 + B1*X1 + B2*X2 + Bn*Xn). Where Y is the dependent variable and the X1, X2, and Xn are independent variables. The MREG function requires that the function parameters TY, TX1, TX2, and TXn be regular time series variables and that they all have the same time interval. It is further required that the time variables be retrieved in the same sequential order as specified in the function parameter list. In other words TY is retrieved from DSS first followed by TX1 and so on. The number of independent time series variables that can be specified is limited to one less than the maximum number of time series variables that the program allows. On the DOS PC this value will be 4 independent variables. The calculated linear regression coefficients will be stored in a paired function variable (PF) specified by the user. The user should store this variable to DSS with the PUT command if it is to be used later with the AMREG function. Two optional function parameters "min, max" are available to specify a range of values that control which values in the dependent variable are to be used for calculating the regression coefficients. These two parameters must be specified as the last two parameters and if specified, both must be specified as real numbers. The statistics output is written to file "MREG.REP" in addition to being printed out in the regular output.

Example: The following example uses the MREG function to determine the linear regression coefficients for a downstream flow gage in relation to two upstream flow gages. The coefficients are saved as paired data to a DSS file and may later be used by the AMREG function to predict flows in the future from the observed flow values at the two upstream gages.

Macro:

```
MACRO MREG2
TIME 0100 01OCT1972 2400 30NOV1977
GET TX1=MREG.DSS:/CHICKLKV/TX1/FLOW/01JAN1972/1DAY/OBS/
GET TX2=MREG.DSS:/CHICKENT/TX2/FLOW/01JAN1972/1DAY/OBS/
GET TX3=MREG.DSS:/BUCKNER/TX3/FLOW/01JAN1972/1DAY/OBS/
COMP TX2=TSHIFT(TX2,4D)
COMP TX3=TSHIFT(TX3,1D)
COMP PF=MREG(TX1,TX2,TX3)
PUT.A PF=MREG.DSS:/LINEAR REGRESSION/COEFFICIENTS/5YEARS////
ENDMACRO
```

Execution:

```
I>!R MREG2
    -----DSS---ZOPEN:  Existing File Opened,  File: MREG.DSS
                       Unit:  71;  DSS Version: 6-HD
 -----DSS--- ZREAD Unit  71; Vers.    4:  /CHICKLKV/TX1/FLOW/01JAN1972/1DAY/OBS/
 -----DSS--- ZREAD Unit  71; Vers.    3:  /CHICKLKV/TX1/FLOW/01JAN1973/1DAY/OBS/
 -----DSS--- ZREAD Unit  71; Vers.    3:  /CHICKLKV/TX1/FLOW/01JAN1974/1DAY/OBS/
 -----DSS--- ZREAD Unit  71; Vers.    3:  /CHICKLKV/TX1/FLOW/01JAN1975/1DAY/OBS/
 -----DSS--- ZREAD Unit  71; Vers.    3:  /CHICKLKV/TX1/FLOW/01JAN1976/1DAY/OBS/
 -----DSS--- ZREAD Unit  71; Vers.    3:  /CHICKLKV/TX1/FLOW/01JAN1977/1DAY/OBS/
 -----DSS--- ZREAD Unit  71; Vers.    3:  /CHICKENT/TX2/FLOW/01JAN1972/1DAY/OBS/
 -----DSS--- ZREAD Unit  71; Vers.    1:  /CHICKENT/TX2/FLOW/01JAN1973/1DAY/OBS/
 -----DSS--- ZREAD Unit  71; Vers.    1:  /CHICKENT/TX2/FLOW/01JAN1974/1DAY/OBS/
```

```
-----DSS--- ZREAD Unit  71; Vers.    1:  /CHICKENT/TX2/FLOW/01JAN1975/1DAY/OBS/
-----DSS--- ZREAD Unit  71; Vers.    1:  /CHICKENT/TX2/FLOW/01JAN1976/1DAY/OBS/
-----DSS--- ZREAD Unit  71; Vers.    1:  /CHICKENT/TX2/FLOW/01JAN1977/1DAY/OBS/
-----DSS--- ZREAD Unit  71; Vers.    5:  /BUCKNER/TX3/FLOW/01JAN1972/1DAY/OBS/
-----DSS--- ZREAD Unit  71; Vers.    3:  /BUCKNER/TX3/FLOW/01JAN1973/1DAY/OBS/
-----DSS--- ZREAD Unit  71; Vers.    3:  /BUCKNER/TX3/FLOW/01JAN1974/1DAY/OBS/
-----DSS--- ZREAD Unit  71; Vers.    3:  /BUCKNER/TX3/FLOW/01JAN1975/1DAY/OBS/
-----DSS--- ZREAD Unit  71; Vers.    3:  /BUCKNER/TX3/FLOW/01JAN1976/1DAY/OBS/
-----DSS--- ZREAD Unit  71; Vers.    3:  /BUCKNER/TX3/FLOW/01JAN1977/1DAY/OBS/
WARNING -- EXISTING VARIABLE RE-DEFINED
WARNING -- EXISTING VARIABLE RE-DEFINED
 Multiple Regression Written to: "mreg.rep"
ERROR SUM OF SQUARES   1.032530E+10
REGRESSION SUM OF SQUARES   5.177439E+10
TOTAL SUM OF SQUARES   6.209969E+10
COEFFICIENT OF MULTIPLE DETERMINATION   8.337302E-01

REGRESSION COEFFICIENTS
  NUMBER     VALUE
     0    1108.491000
     1        .525309
     2       3.366069
 INDEX     1882.000000
 ---------------------------------------------------
Mean absolute error   =   .40 % (1882. d)
Mean negative error   =  -.17 % ( 576. d)
Mean positive error   =   .50 % (1306. d)
Total volume error    =  -.30 % (1882. d)

Percentage of errors <= 5 %    =      12.805530
Percentage of errors <= 10 %   =      23.910730
Percentage of errors <= 15 %   =      34.909670
Percentage of errors <= 20 %   =      43.942620
Percentage of errors <= 25 %   =      51.328370
Percentage of errors  > 25 %   =      48.671630
 ---------------------------------------------------
-----DSS---ZWRITE Unit  71; Vers.   11:  /LINEAR REGRESSION/COEFFICIENTS/5YEARS////
-----DSS---ZCLOSE Unit:  71,   File: MREG.DSS
```

Name: **MRG** **Merge two time series**
Use: CO TY=MRG(TX,TZ,QFLAG)

Merge TX with TZ. TY includes all values in TX and TZ, except where TX and TZ occur at the same time. In that case, the value for TX is used unless it is flagged with a quality equal to or less than QFLAG and TZ is flagged with a quality greater than QFLAG. Possible flags, in order of decreasing quality, are: N = no flag, E = estimated, Q = questionable and M = missing. An IF condition has no effect. The type and units of TY are undefined and must be set by the SD command.

Execution: The following example uses the GENTSR function to generate a regular time series that has all its values set to missing. The generated time series is then merged, using MRG, with an irregular time series that reports at hourly intervals, but not always every hour. The following macro inserts missing values into the irregular time series and then uses the TS1 function to convert the irregular time series to regular.

Macro:

```
MACRO TGENTSR
CL ALL
TIME 01MAY1983 0100 02MAY1983 0100
COM FLOWG=GENTSR(1H,0H,-901.,M)
GET FLOW=testdb.dss:/SCIOTO/BMRI2/FLOW/01MAY1983/IR-MONTH/OBS-I/
COM FLOWM=MRG(FLOW,FLOWG,N)
COM FLOWR=TS1(FLOWM,1H,0H)
TAB.F FLOW FLOWG FLOWR
PUT.A FLOWR=testdb.dss:/SCIOTO/BMRI2/FLOW/01MAY1983/1HOUR/CAL-I/
$CONTINUE INCASE OF ERROR
ENDMACRO
```

Execution:

```
I>!R TGENTSR
 ALL VARIABLES HAVE BEEN CLEARED
*** WARNING : QUALITY FLAG RESET TO NONE
   -----DSS---ZOPEN:  Existing File Opened,  File: TESTDB.DSS
                      Unit:  71;  DSS Version: 6-IA

Variables ----   FLOW     FLOWG        FLOWR
Units     ----   CFS                   CFS
Time   Date
0100 01MAY1983  100.735   M          100.735
0200 01MAY1983  100.085   M          100.085
0300 01MAY1983   99.436   M           99.436
0400 01MAY1983    -       M            M
0500 01MAY1983   98.140   M           98.140
0600 01MAY1983    -       M            M
0700 01MAY1983   96.844   M           96.844
0800 01MAY1983   96.196   M           96.196
0900 01MAY1983   95.873   M           95.873
1000 01MAY1983    -       M            M
1100 01MAY1983   97.491   M           97.491
1200 01MAY1983  102.035   M          102.035
1300 01MAY1983  106.597   M          106.597
1400 01MAY1983    -       M            M
1500 01MAY1983  109.211   M          109.211
1600 01MAY1983  111.504   M          111.504
1700 01MAY1983  112.815   M          112.815
1800 01MAY1983    -       M            M
1900 01MAY1983  115.443   M          115.443
2000 01MAY1983  115.443   M          115.443
2100 01MAY1983  114.457   M          114.457
2200 01MAY1983    -       M            M
2300 01MAY1983  112.815   M          112.815
2400 01MAY1983  112.487   M          112.487
0100 02MAY1983  111.504   M          111.504
```

Name: MRGP **Merge two paired data series**
Use: CO PY=MRGP(PX,PZ,NCURVE)

Merge PZ with PX. PY includes all values in PX and the curve specified by the index curve number NCURVE in PZ. It is required that PX and PZ have the same number of data values and that the X axis values for both PX and PZ are identical. NCURVE is used to specify the curve number data in PZ that is to be merged with PX. A value of 0 for NCURVE signifies that all curves in PZ are to be merged with PX.

Example: The following example uses the MRGP function to merge two paired data curves into a single paired data that contains both curves.

Macro:

```
MACRO TMRGP
TIME T-3D T
GET XX=testdb.dss:/PULS//FLOW-STORAGE////
GET YY=F=TEST
COMP ZZ=MRGP(XX,YY,1)
PUT ZZ=B=TMRGP
TAB.F XX YY ZZ
ENDMACRO
```

Execution:

```
I>!R TMRGP
 -----DSS---ZOPEN:  Existing File Opened,  File: TESTDB.DSS
                       Unit:  71;  DSS Version: 6-IA
 -----DSS--- ZREAD Unit  71; Vers.    2:  /PULS//FLOW-STORAGE////
 -----DSS--- ZREAD Unit  71; Vers.    1:  /PULS//FLOW-STORAGE///TEST/
 -----DSS---ZWRITE Unit  71; Vers.    6:  /PULS/TMRGP/FLOW-STORAGE///TEST/
 -----DSS---ZCLOSE Unit:  71,   File: TESTDB.DSS
```

```
VARIABLE=XX
  UNITS = AC-FT      CFS
  TYPES = UNT        UNT
  LABEL = FLOW
  SHIFT =  OFFSET =  DATUM =

    X-AXIS          FLOW
        .0000        .0000
      10.0000      50.0000
     100.0000     500.0000
     120.0000    1000.0000
     320.0000    5000.0000
     550.0000   10000.0000
     800.0000   20000.0000
    1500.0000   40000.0000
    3000.0000   80000.0000
```

```
VARIABLE=YY
  UNITS = AC-FT      CFS
  TYPES = UNT        UNT
  LABEL = FLOW
  SHIFT =  OFFSET =  DATUM =

    X-AXIS          FLOW
        .0000        .0000
      10.0000      50.0000
     100.0000    3921.5690
     120.0000    4405.7790
     320.0000    9247.8850
     550.0000   15310.4600
     800.0000   24705.8800
    1500.0000   54453.4600
    3000.0000   95374.0000
```

```
VARIABLE=ZZ
  UNITS = AC-FT      AC-FT
  TYPES = UNT        UNT
  LABEL = FLOW
  LABEL = FLOW
  SHIFT =  OFFSET =  DATUM =

    X-AXIS          FLOW          FLOW
        .0000        .0000        .0000
      10.0000      50.0000      50.0000
     100.0000     500.0000    3921.5690
     120.0000    1000.0000    4405.7790
     320.0000    5000.0000    9247.8850
     550.0000   10000.0000   15310.4600
     800.0000   20000.0000   24705.8800
    1500.0000   40000.0000   54453.4600
    3000.0000   80000.0000   95374.0000
```

Name: **MUSK** **Muskingum hydrologic routing**
Use: CO TY=MUSK(TX,NR,K,X)

Route the uniform time series variable TX by the Muskingum hydrologic routing method and store it in variable TY. The "IF" compute option has no effect on this function. The "K" parameter is the Muskingum "k" in hours, "X" parameter is the Muskingum "x" (range between 0 and .5), and "NR" is the number of routing subreaches.

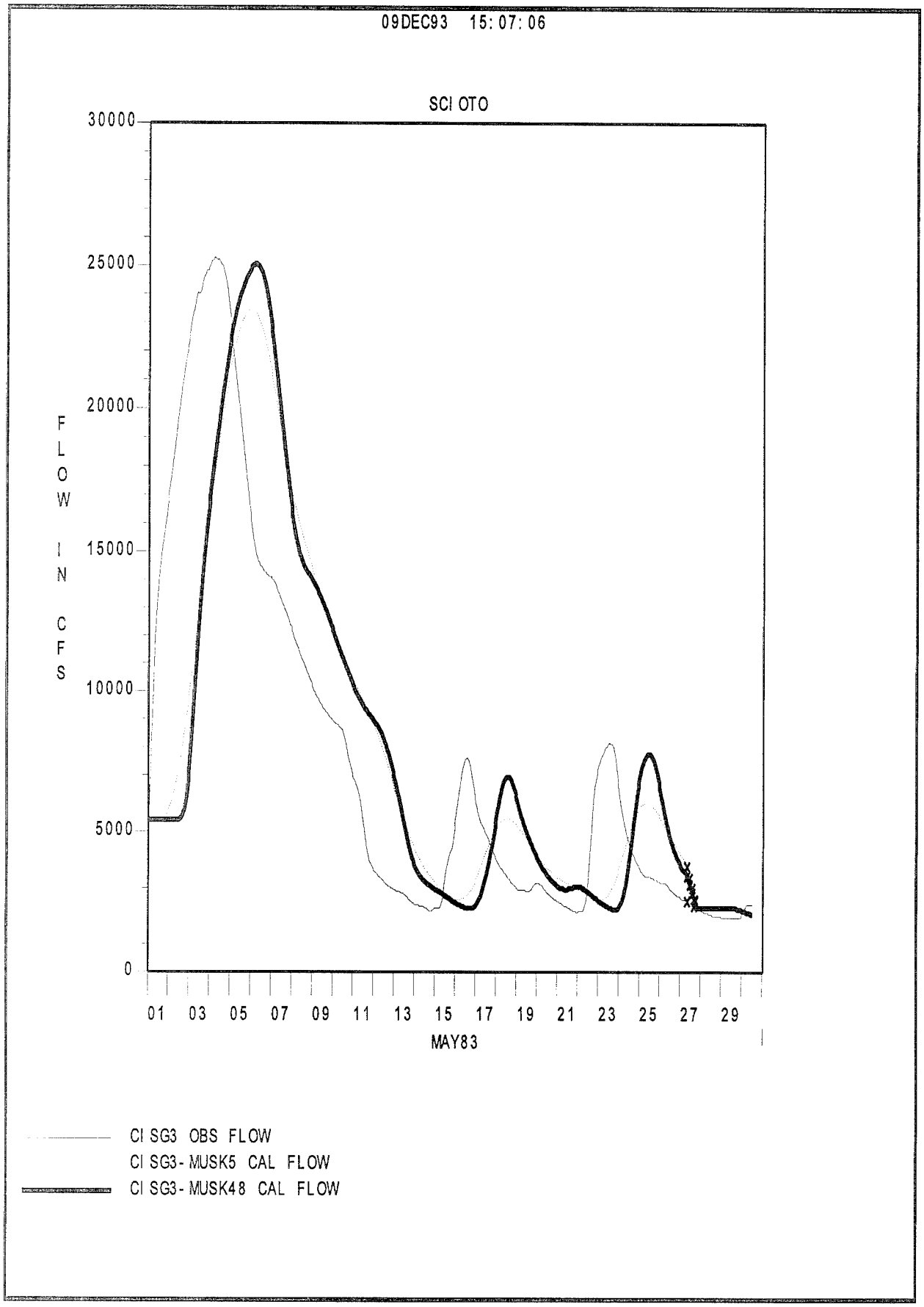
Example: The following example uses the MUSK function to perform a Muskingum routing using 5 and 48 routing steps.

Macro:

```
MACRO TMUSK
** COMPUTE A MUSKINGUM ROUTING
CL ALL
TIME 01APR1983 0100 30MAY1983 1200
GET FLOW=testdb.dss:/SCIOTO/CISG3/FLOW/01MAY1983/1HOUR/OBS/
COMP RFLOW=MUSK(FLOW,5,48,.01)
PUT.A RFLOW=B=CISG3-MUSK5,F=CAL
COMP RFLOW=MUSK(FLOW,48,48,.01)
PUT.A RFLOW=B=CISG3-MUSK48,F=CAL
$CONTINUE INCASE OF ERROR
ENDMACRO
```

Execution:

```
I>!R TMUSK
 ALL VARIABLES HAVE BEEN CLEARED
   -----DSS---ZOPEN:  Existing File Opened,  File: TESTDB.DSS
                      Unit:  71;  DSS Version: 6-IA
-----DSS--- ZREAD Unit  71; Vers.    1:  /SCIOTO/CISG3/FLOW/01APR1983/1HOUR/OBS/
-----DSS--- ZREAD Unit  71; Vers.   22:  /SCIOTO/CISG3/FLOW/01MAY1983/1HOUR/OBS/
 WARNING - NUMBER OF ROUTING STEPS SPECIFIED   5  NOT EQUAL TO COMPUTED   48
-----DSS---ZWRITE Unit  71; Vers.    1:  /SCIOTO/CISG3-MUSK5/FLOW/01APR1983/1HOUR/CAL/
-----DSS---ZWRITE Unit  71; Vers.    1:  /SCIOTO/CISG3-MUSK5/FLOW/01MAY1983/1HOUR/CAL/
WARNING -- EXISTING VARIABLE RE-DEFINED
-----DSS---ZWRITE Unit  71; Vers.    1:  /SCIOTO/CISG3-MUSK48/FLOW/01APR1983/1HOUR/CAL/
-----DSS---ZWRITE Unit  71; Vers.    1:  /SCIOTO/CISG3-MUSK48/FLOW/01MAY1983/1HOUR/CAL/
```


09DEC93 15:07:06
SCI OTO
30000
25000
20000
15000
10000
5000
0
FLOW IN CFS
01 03 05 07 09 11 13 15 17 19 21 23 25 27 29
MAY83
CI SG3 OBS FLOW
CI SG3-MUSK5 CAL FLOW
CI SG3-MUSK48 CAL FLOW

Name: **NINT** **Round to nearest whole number**
Use: CO TY=NINT(TX)

Round to nearest whole number TX and store in TY. TX and TY can be the same variable. TX can be scalar, time series or paired data. If a value of TX is undefined resulting values of TY are undefined. If a concurrent IF condition is not satisfied, TY is unchanged.

Example: The following example demonstrates how the NINT function will round up when a value is greater than .5 and round down when it is less than .5.

Macro:

```
MACRO TNINT
COMP VRUP=NINT(10.50)
COMP VRDN=NINT(10.49999)
SHOW VRUP VRDN
ENDMACRO
```

Execution:

```
I>!R TNINT
VARIABLE=VRUP
VRUP     =       11.000000

VARIABLE=VRDN
VRDN     =       10.000000
```

Name: OLY **Olympic smoothing**
Use: CO TY=OLY(TX,NP[,CLEVEL])

Compute a smoothed time-series from TX using the Olympic smoothing scheme: same as centered, moving average except the minimum and maximum values in the span NP are ignored. Place the result in TY. Units and type are unchanged. The following levels are available:

LEVEL1 Only valid values will be averaged. Valid values that do not have all valid values within the averaging period number will be missing. Values at the start and end of the data that do not have enough valid values to average will be assigned missing values.

LEVEL2 Only valid values will be averaged. Valid values that do not have all valid values within the averaging period number will be missing. Valid values at the start and end of the data that do not have enough valid values to average will be averaged over a reduced number of values.

LEVEL3 (Default setting) All values will be averaged based on valid values within the averaging period number. If all values are missing for the period number, the value will be missing. Values at the start and end of the data that do not have enough valid values to average will be assigned missing values.

LEVEL4 All values will be averaged based on valid values within the averaging period number. If all values are missing for the period number, the value will be missing. Values at the start and end of the data that do not have valid values to average will be averaged based on a reduced number of values.

Note: Questionable and estimated flagged values are used in the computations.

Example: The following example uses the OLY function to take flow values and smooth them using a moving average of 3 and 5.

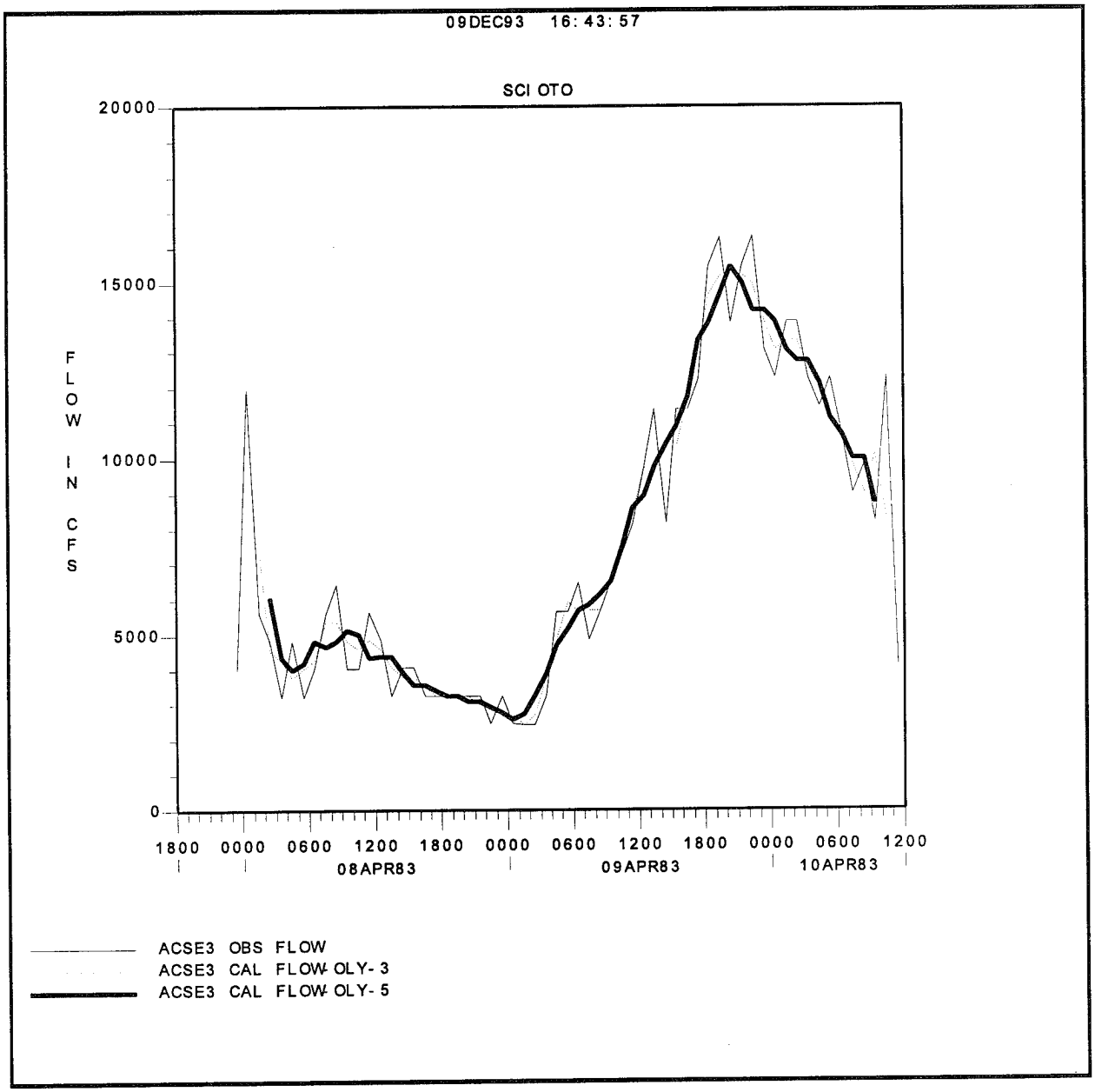
Macro:

```
MACRO TOLY
CL ALL
TIME 08APR1983 0100 10APR1983 1200
GET FLOW=testdb.dss:/SCIOTO/ACSE3/FLOW/01APR1983/1HOUR/OBS/
COMP AVE3=OLY(FLOW,3,LEVEL3)
PUT.A AVE3=C=FLOW-OLY-3 F=CAL
COMP AVE5=OLY(FLOW,5,LEVEL3)
PUT.A AVE5=C=FLOW-OLY-5 F=CAL
TAB.F FLOW AVE3 AVE5
$CONTINUE INCASE OF ERROR
ENDMACRO
```

Execution:

```
I>!R TOLY
 ALL VARIABLES HAVE BEEN CLEARED
   -----DSS---ZOPEN:  Existing File Opened,  File: TESTDB.DSS
                      Unit:  71;  DSS Version: 6-IA
-----DSS--- ZREAD Unit  71; Vers.    3:  /SCIOTO/ACSE3/FLOW/01APR1983/1HOUR/OBS/
```

```
-----DSS---ZWRITE Unit  71; Vers.    1:  /SCIOTO/ACSE3/FLOW-OLY-3/01APR1983/1HOUR/CAL/
WARNING -- EXISTING VARIABLE RE-DEFINED
-----DSS---ZWRITE Unit  71; Vers.    1:  /SCIOTO/ACSE3/FLOW-OLY-5/01APR1983/1HOUR/CAL/
```



Variables ----		FLOW	AVE3	AVE5
Units ----		CFS	CFS	CFS
Time	Date			
0030	08APR1983	11995.000	M	M
0130	08APR1983	5628.000	7485.000	M
0230	08APR1983	4832.000	4567.000	6106.600
0330	08APR1983	3241.000	4303.333	4356.200
0430	08APR1983	4837.000	3773.667	4039.000
0530	08APR1983	3243.000	4040.667	4200.800
0630	08APR1983	4042.000	4308.667	4841.600
0730	08APR1983	5641.000	5376.000	4685.800
0830	08APR1983	6445.000	5381.333	4850.200
0930	08APR1983	4058.000	4856.000	5175.800
1030	08APR1983	4065.000	4597.667	5023.200
1130	08APR1983	5670.000	4871.000	4390.800
1230	08APR1983	4878.000	4610.333	4396.000

1330	08APR1983	3283.000	4081.667	4399.800
1430	08APR1983	4084.000	3817.000	3923.200
1530	08APR1983	4084.000	3818.333	3604.800
1630	08APR1983	3287.000	3552.333	3605.000
1730	08APR1983	3286.000	3285.667	3444.800
1830	08APR1983	3284.000	3284.333	3284.200
1930	08APR1983	3283.000	3282.667	3282.600
2030	08APR1983	3281.000	3281.000	3120.600
2130	08APR1983	3279.000	3012.000	3118.800
2230	08APR1983	2476.000	3010.000	2956.800
2330	08APR1983	3275.000	2741.333	2790.200
0030	09APR1983	2473.000	2732.000	2624.000
0130	09APR1983	2448.000	2456.333	2783.800
0230	09APR1983	2448.000	2723.667	3265.200
0330	09APR1983	3275.000	3801.667	3907.000
0430	09APR1983	5682.000	4879.667	4714.600
0530	09APR1983	5682.000	5950.000	5202.600
0630	09APR1983	6486.000	5685.333	5687.400
0730	09APR1983	4888.000	5691.000	5854.600
0830	09APR1983	5699.000	5701.667	6184.600
0930	09APR1983	6518.000	6516.333	6516.600
1030	09APR1983	7332.000	7332.000	7491.200
1130	09APR1983	8146.000	8413.000	8627.400
1230	09APR1983	9761.000	9762.333	8957.600
1330	09APR1983	11380.000	9770.000	9770.800
1430	09APR1983	8169.000	10315.670	10422.200
1530	09APR1983	11398.000	10323.330	10913.400
1630	09APR1983	11403.000	11672.670	11730.600
1730	09APR1983	12217.000	13028.670	13353.200
1830	09APR1983	15466.000	14655.000	13848.000
1930	09APR1983	16282.000	15206.670	14667.600
2030	09APR1983	13872.000	15218.330	15488.400
2130	09APR1983	15501.000	15231.330	15013.800
2230	09APR1983	16321.000	14971.670	14213.000
2330	09APR1983	13093.000	13897.330	14219.000
0030	10APR1983	12278.000	13091.000	13900.400
0130	10APR1983	13902.000	13362.670	13093.600
0230	10APR1983	13908.000	13365.670	12770.800
0330	10APR1983	12287.000	12558.000	12774.000
0430	10APR1983	11479.000	12020.000	12127.400
0530	10APR1983	12294.000	11480.670	11154.400
0630	10APR1983	10669.000	10668.670	10669.600
0730	10APR1983	9043.000	9858.333	10020.600
0830	10APR1983	9863.000	9046.667	10026.600
0930	10APR1983	8234.000	10140.330	8724.000
1030	10APR1983	12324.000	8238.000	M
1130	10APR1983	4156.000	M	M

Name: **PERCON** **Period constants**
Use: CO TY=PERCON(SY,TX)

Generates a series TY at times concurrent with TX and with values equal to the previous chronological SY value. If no previous SY is present, then TY values are undefined. SY and TX may be irregular- or regular-interval time series. An IF condition has no effect.

Name: **POLY** **Polynomial transformation**
Use: CO TY=POLY(TX,TP)

Compute a polynomial transformation of TX using the polynomial coefficients TP. Store the result in TY. If a TX value is missing TY is undefined. If a concurrent IF condition is not satisfied, TY is unchanged. Units and type of TY are defined by TP. TP can be created with the utility DSSPD.

Example: The following example uses the PERCON and POLY functions to determine the surface area (AC) and evaporation (ACFT) for Black Butte Reservoir. Note the extensive use of PREAD function keys and variable parameters used with macros.

Macro:

```
MACRO TPERCON
!TEACH i TESTDB.DSS
!TEACH j TESTDB.DSS
!TEACH k TESTDB.DSS
!TEACH m .M
TIME 0100 01JAN93 1200 01DEC93
!R DEV ^i ^j ^k "STONY CR" "BLACK BUTTE"
ENDMACRO
```

```
MACRO DEV  %HPDB %EPDB %EVDB %APART %BPART
****|**********|****************************|**********************
**  Compute    Daily EVAP-RES AF
**                 (Evap for Reservoir in ACFT)
**  INPUT:
**    %HPDB   Reservoir Elevation Database
**    %EPDB   Pan Evap Database
**    %EVDB   Lake Evap Database
**    %APART  'A' part of pathname
**    %BPART  'B' part of pathname
**  OUTPUT:
**    EV$     EVAP-RES AF (Reservoir Evaportion, acft)
**  VARIABLES:
**    EP$     Daily EVAP-PAN (Pan Evaporation at reservoir, inches)
**    HP$     Daily ELEV (Reservoir Elevation from Mean Sea Level). Be sure
**            to retrieve the previous value outside the time window.
**    POLY$   ELEV-AREA polynomial table for reservoir
**    COEF$   Evaporation coefficient time-series (period average values)
*********************************************************************
CL ALL
GE EP$=%EPDB:/%APART/%BPART/EVAP-PAN//1DAY/OBS/
TI.N -30D
GE.P MCOEF$=TESTDB.DSS:/ALL CORPS RES//EVAP COEFS//1MON/AVG MONTHLY/
TI.N +30D
CO COEF$=PERCON(MCOEF$,EP$)
TAB.F MCOEF$ EP$ COEF$
CO EV$=EP$*COEF$
CL MCOEF$
!RUN DLA  ^k  ^k '%APART' '%BPART'
GE LA$=^k:/%APART/%BPART/AREA-LAKE//1DAY//
CO EV$=LA$*EV$
CO EV$=RND(EV$,8,0)
SD EV$ UNITS=ACFT TYPE=PER-CUM
PU^m EV$=%EVDB:/%APART/%BPART/EVAP-LAKE AF//1DAY//
$CONTINUE
ENDMACRO
```

```
MACRO DLA %DHPDB %DLADB %APART %BPART
****|**********|****************************|************
**  Compute    Daily Midnight Lake Area
**  ELEV-AREA POLYNOMIALS
**  INPUT:
**    %DHPDB  Daily Reservoir Elevation Database
**    %DLADB  Daily Lake Area Database
**    %APART  'A' part of pathname
**    %BPART  'B' part of pathname
**  VARIABLES:
**    DHP$    ELEV (Reservoir Elevation from Mean Sea Level).
**    POLY$   ELEV-AREA polynomial table for reservoir
*******************************************************
GE DHP$=%DHPDB:/%APART/%BPART/ELEV//1DAY//
GE POLY$=TESTDB.DSS:/%APART/%BPART/ELEV-AREA///POLY/
STAT POLY$
TAB.F POLY$
CO DLA$=POLY(DHP$,POLY$)
CL POLY$
CO DLA$=RND(DLA$,8,0)
SD DLA$ UNITS=ACRES TYPE=INST-VAL
PU^m DLA$=%DLADB:/%APART/%BPART/AREA-LAKE//1DAY//
CL DLA$
$CONTINUE
ENDMACRO
```

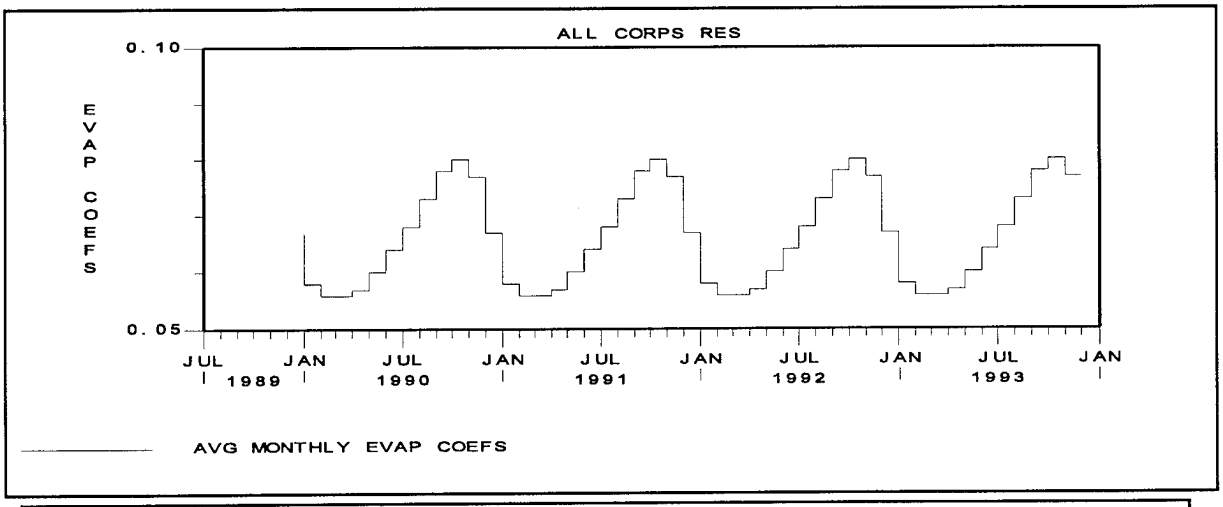
Execution:

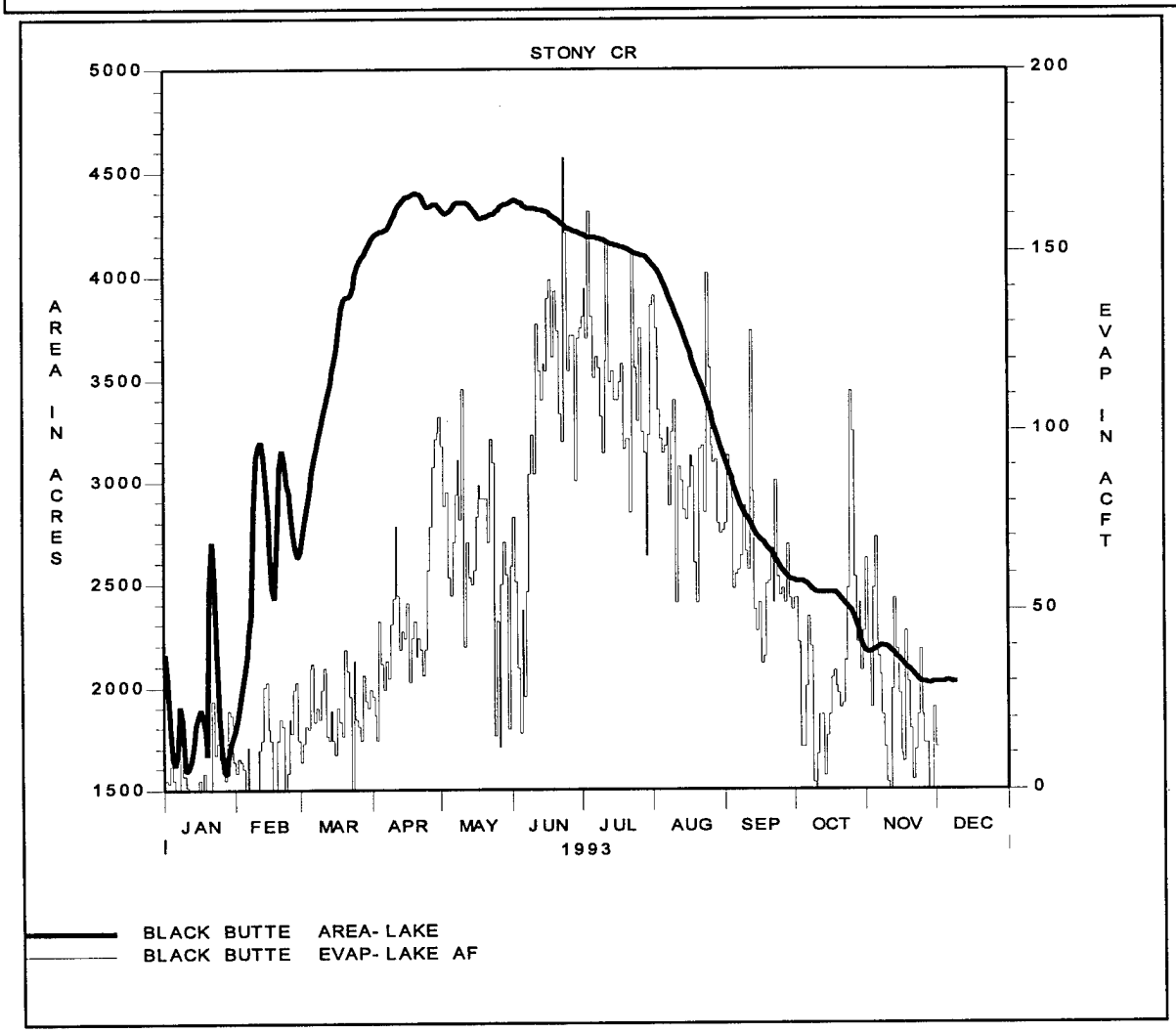
```
I>!R TPERCON
ALL VARIABLES HAVE BEEN CLEARED
   -----DSS---ZOPEN:  Existing File Opened,  File: TESTDB.DSS
                      Unit:  71;  DSS Version: 6-IA
-----DSS--- ZREAD Unit  71; Vers.    1:  /STONY CR/BLACK BUTTE/EVAP-PAN/01JAN1993/1DAY/OBS/
-----DSS--- ZREAD Unit  71; Vers.    3:  /ALL CORPS RES//EVAP COEFS/01JAN1990/1MON/AVG MONTHLY/
```

Variables ----	MCOEF$	EP$	COEF$
Units ----		INCHES	UNK
Time Date			
0002 01DEC1992	.077	-	-
0002 01JAN1993	.067	-	-
2400 01JAN1993	-	.000	.067
2400 02JAN1993	-	.020	.067
2400 03JAN1993	-	.020	.067
2400 04JAN1993	-	.060	.067
2400 05JAN1993	-	.030	.067
2400 06JAN1993	-	.000	.067
2400 07JAN1993	-	.000	.067
2400 08JAN1993	-	.080	.067
2400 09JAN1993	-	.040	.067
2400 10JAN1993	-	.040	.067
2400 11JAN1993	-	.010	.067
2400 12JAN1993	-	.000	.067
2400 13JAN1993	-	.000	.067
2400 14JAN1993	-	.000	.067
2400 15JAN1993	-	.000	.067
2400 16JAN1993	-	.020	.067
2400 17JAN1993	-	.000	.067
2400 18JAN1993	-	.040	.067
2400 19JAN1993	-	.000	.067
2400 20JAN1993	-	.000	.067
2400 21JAN1993	-	.000	.067
2400 22JAN1993	-	.150	.067

2400 23JAN1993	-	.070	.067
2400 24JAN1993	-	.100	.067
2400 25JAN1993	-	.080	.067
2400 26JAN1993	-	.070	.067
2400 27JAN1993	-	.030	.067
2400 28JAN1993	-	.120	.067
2400 29JAN1993	-	.190	.067
2400 30JAN1993	-	.180	.067
2400 31JAN1993	-	.070	.067
0002 01FEB1993	.058	-	-
2400 01FEB1993	-	.050	.058
2400 02FEB1993	-	.080	.058
2400 03FEB1993	-	.070	.058
2400 04FEB1993	-	.050	.058
2400 05FEB1993	-	.000	.058
2400 06FEB1993	-	.090	.058
2400 07FEB1993	-	.000	.058
2400 08FEB1993	-	.000	.058
2400 09FEB1993	-	.000	.058
2400 10FEB1993	-	.000	.058
2400 11FEB1993	-	.060	.058
2400 12FEB1993	-	.080	.058
2400 13FEB1993	-	.170	.058
2400 14FEB1993	-	.180	.058
2400 15FEB1993	-	.110	.058
2400 16FEB1993	-	.100	.058
2400 17FEB1993	-	.000	.058
2400 18FEB1993	-	.000	.058
	-	-	-
	.	.	.

```
 MCOEF$ CLEARED
-----DSS--- ZREAD Unit  71; Vers.  400:  /STONY CR/BLACK BUTTE/ELEV/01JAN1993/1DAY//
-----DSS--- ZREAD Unit  71; Vers.    4:  /STONY CR/BLACK BUTTE/ELEV-AREA///POLY/
VARIABLE=POLY$
 UNITS = FEET-MSL   ACRES
 TYPES = UNT        UNT
 LABEL =
 SHIFT =      0.0   OFFSET =      0.0   DATUM =  375.0
   X-AXIS
      -1.09760          1.0000
        .49540          2.0000
       -.02000          3.0000
        .00080          4.0000
      -.11936E-04       5.0000
       .77074E-07       6.0000
      -.18529E-09       7.0000
 POLY$ CLEARED
WARNING -- EXISTING VARIABLE RE-DEFINED
-----DSS---ZWRITE Unit  71; Vers.  417:  /STONY CR/BLACK BUTTE/AREA-LAKE/01JAN1993/1DAY//
 DLA$ CLEARED
-----DSS--- ZREAD Unit  71; Vers.  417:  /STONY CR/BLACK BUTTE/AREA-LAKE/01JAN1993/1DAY//
WARNING -- EXISTING VARIABLE RE-DEFINED
-----DSS---ZWRITE Unit  71; Vers.   10:  /STONY CR/BLACK BUTTE/EVAP-LAKE AF/01JAN1993/1DAY//
```


ALL CORPS RES
0.10
0.05
EVAP COEFS
JUL
JAN
JUL
JAN
JUL
JAN
JUL
JAN
JUL
JAN
1989
1990
1991
1992
1993
AVG MONTHLY EVAP COEFS


STONY CR
5000
4500
4000
3500
3000
2500
2000
1500
AREA IN ACRES
200
150
100
50
0
EVAP IN ACFT
JAN
FEB
MAR
APR
MAY
JUN
JUL
AUG
SEP
OCT
NOV
DEC
1993
BLACK BUTTE AREA- LAKE
BLACK BUTTE EVAP- LAKE AF

Name: **POLY2** **Polynomial transformation with integral**
Use: CO TY=POLY2(TX,TP)

Compute a polynomial transformation of TX using the integral of the polynomial defined by coefficients TP. Store the result in TY. If a TX value is missing TY is undefined. If a concurrent IF condition is not satisfied, TY is unchanged. Units and type are defined by TP. TP can be created with the utility DSSPD.

Example: The following example uses the PERCON and POLY2 functions to determine the storage volume for Black Butte Reservoir. Note the extensive use of PREAD function keys and variable parameters used with macros.

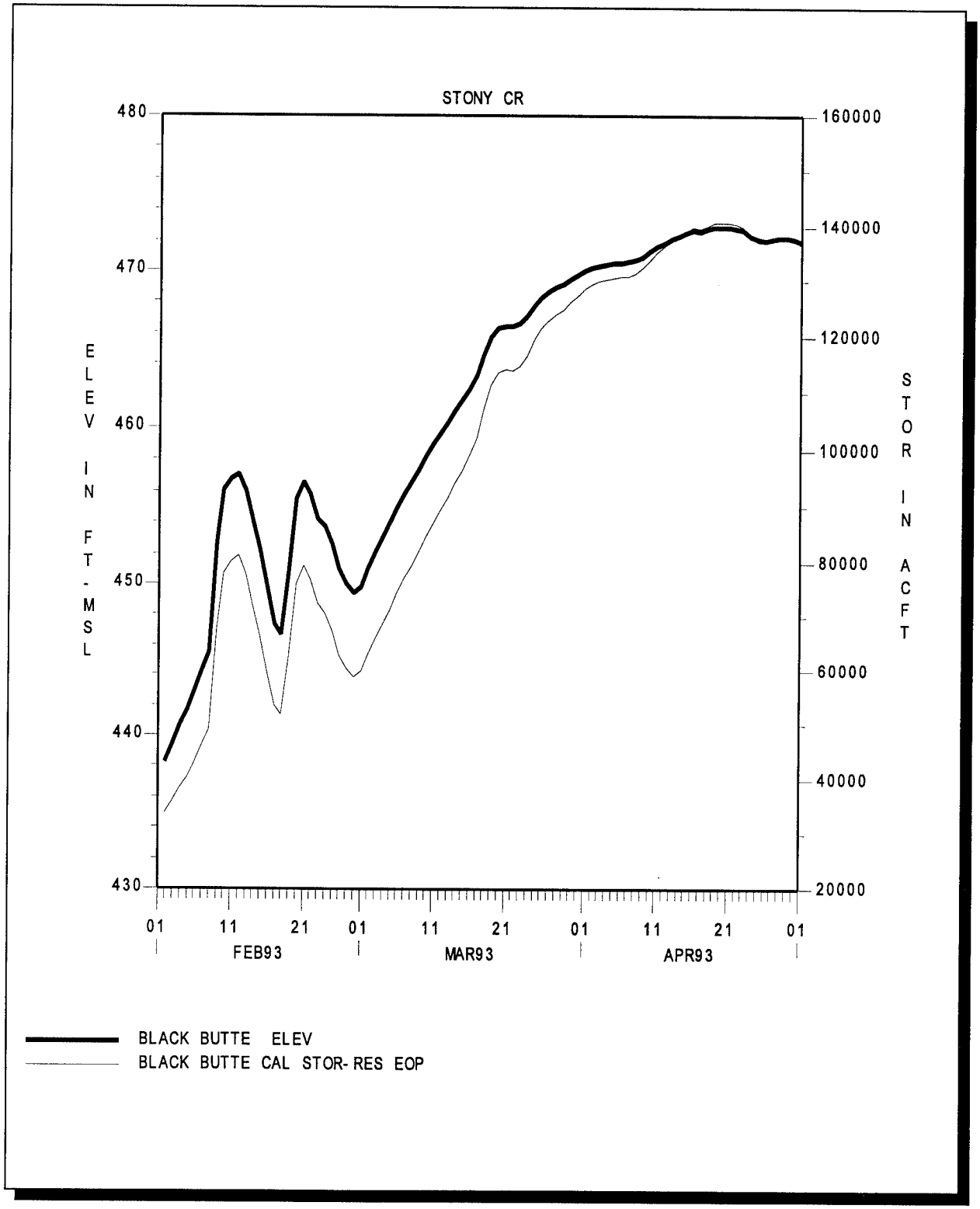
Macro:

```
MACRO TPOLY2
!TEACH j TESTDB.DSS
!TEACH k TESTDB.DSS
!TEACH m .M
TIME 0100 01FEB93 2400 30APR93
!R DLS  ^j ^k "STONY CR" "BLACK BUTTE"
ENDMACRO

MACRO DLS %HPDB %LSDB %APART %BPART
****|**********|******************************|**********************
**  Compute    Daily STOR-RES EOP
**  INPUT:
**    %HPDB   Elevation Database
**    %LSDB   Storage Database
**    %APART  "A" Part of Pathname
**    %BPART  "B" Part of Pathname
*********************************************************************
GE HP$=%HPDB:/%APART/%BPART/ELEV//1HOUR//
GE POLY=TESTDB.DSS:/%APART/%BPART/ELEV-AREA///POLY/
CO X$=GENTSR(1D,0D,-901,M)
CO HP$$=PERCON(HP$,X$)
CO LS$=POLY2(HP$$,POLY)
** Round storage to units
CO LSR$=RND(LS$,8,0)
SD LSR$ UNITS=ACFT  TYPE=INST-VAL
PU^m LSR$=%LSDB:/%APART/%BPART/STOR-RES EOP//1DAY/CAL/
$CONTINUE
ENDMACRO
```

Execution:

```
I>!R  TPOLY2
   -----DSS---ZOPEN:  Existing File Opened,  File: TESTDB.DSS
                      Unit:  71;  DSS Version: 6-IA
-----DSS--- ZREAD Unit  71; Vers.  693:  /STONY CR/BLACK BUTTE/ELEV/01FEB1993/1HOUR//
-----DSS--- ZREAD Unit  71; Vers.  783:  /STONY CR/BLACK BUTTE/ELEV/01MAR1993/1HOUR//
-----DSS--- ZREAD Unit  71; Vers.  756:  /STONY CR/BLACK BUTTE/ELEV/01APR1993/1HOUR//
-----DSS--- ZREAD Unit  71; Vers.    6:  /STONY CR/BLACK BUTTE/ELEV-AREA///POLY/

*** WARNING : QUALITY FLAG RESET TO NONE
DATUM=     375.000000
-----DSS---ZWRITE Unit  71; Vers.    5:  /STONY CR/BLACK BUTTE/STOR-RES EOP/01JAN1993/1DAY/CAL/
```

Variables ----		X$	HP$	HP$$	LS$	LSR$
Units ----			FT-MSL	UNK	ACRES	ACFT
Time	Date					
0100	01FEB1993	-	437.590	-	-	-
0200	01FEB1993	-	437.610	-	-	-
0300	01FEB1993	-	437.620	-	-	-
0400	01FEB1993	-	437.640	-	-	-
0500	01FEB1993	-	437.660	-	-	-
0600	01FEB1993	-	437.680	-	-	-
0700	01FEB1993	-	437.690	-	-	-
0800	01FEB1993	-	437.730	-	-	-
0900	01FEB1993	-	437.760	-	-	-
1000	01FEB1993	-	437.790	-	-	-
1100	01FEB1993	-	437.830	-	-	-
1200	01FEB1993	-	437.860	-	-	-
1300	01FEB1993	-	437.890	-	-	-
1400	01FEB1993	-	437.930	-	-	-
1500	01FEB1993	-	437.970	-	-	-
1600	01FEB1993	-	438.010	-	-	-
1700	01FEB1993	-	438.050	-	-	-
1800	01FEB1993	-	438.090	-	-	-
1900	01FEB1993	-	438.130	-	-	-
2000	01FEB1993	-	438.160	-	-	-
2100	01FEB1993	-	438.200	-	-	-
2200	01FEB1993	-	438.250	-	-	-
2300	01FEB1993	-	438.280	-	-	-
2400	01FEB1993	M	438.310	438.310	33958.550	33959.000
0100	02FEB1993	-	438.360	-	-	-
0200	02FEB1993	-	438.390	-	-	-
0300	02FEB1993	-	438.420	-	-	-
0400	02FEB1993	-	438.460	-	-	-
0500	02FEB1993	-	438.500	-	-	-
0600	02FEB1993	-	438.540	-	-	-
0700	02FEB1993	-	438.570	-	-	-
0800	02FEB1993	-	438.620	-	-	-
0900	02FEB1993	-	438.660	-	-	-
1000	02FEB1993	-	438.710	-	-	-
1100	02FEB1993	-	438.770	-	-	-
1200	02FEB1993	-	438.820	-	-	-
1300	02FEB1993	-	438.880	-	-	-
1400	02FEB1993	-	438.930	-	-	-
1500	02FEB1993	-	438.970	-	-	-
1600	02FEB1993	-	439.050	-	-	-
1700	02FEB1993	-	439.090	-	-	-
1800	02FEB1993	-	439.150	-	-	-
1900	02FEB1993	-	439.210	-	-	-
2000	02FEB1993	-	439.270	-	-	-
2100	02FEB1993	-	439.330	-	-	-
2200	02FEB1993	-	439.390	-	-	-
2300	02FEB1993	-	439.430	-	-	-
2400	02FEB1993	M	439.470	439.470	36143.720	36144.000
0100	03FEB1993	-	439.530	-	-	-
0200	03FEB1993	-	439.580	-	-	-
0300	03FEB1993	-	439.640	-	-	-
0400	03FEB1993	-	439.690	-	-	-
0500	03FEB1993	-	439.740	-	-	-
0600	03FEB1993	-	439.790	-	-	-
0700	03FEB1993	-	439.840	-	-	-
0800	03FEB1993	-	439.890	-	-	-
0900	03FEB1993	-	439.950	-	-	-
1000	03FEB1993	-	439.990	-	-	-
1100	03FEB1993	-	440.050	-	-	-


STONY CR
480
470
460
450
440
430
ELEV IN FT-MSL
160000
140000
120000
100000
80000
60000
40000
20000
STOR IN ACFT
01
11
21
01
11
21
01
11
21
01
FEB93
MAR93
APR93
BLACK BUTTE ELEV
BLACK BUTTE CAL STOR-RES EOP

Name: **PULS** **Modified Puls or Working R&D routing function**
Use: CO TY=PULS(TX,PF,X,NR,STOR1,Q01)

Route the uniform time series variable TX by the Modified Puls or Working R&D hydrologic routing method and store it in variable TY. The variable PF must have been previously defined in a GET command. It references a storage-discharge paired function relationship for the reach. Storage must be the first variable, discharge the second variable, and each variable must repeat only once. Variable X is the wedge coefficient (Muskingum X) for use in working R&D. Use 0.0 value to route by Modified Puls method. Variable NR is the number of routing reaches. STOR1 and Q01 are initial storage and flow respectively. Use a value of -1 for both STOR1 and Q01 in order to use the first flow value in TX and interpolate a corresponding storage value from PF. The IF compute command has no effect on this function.

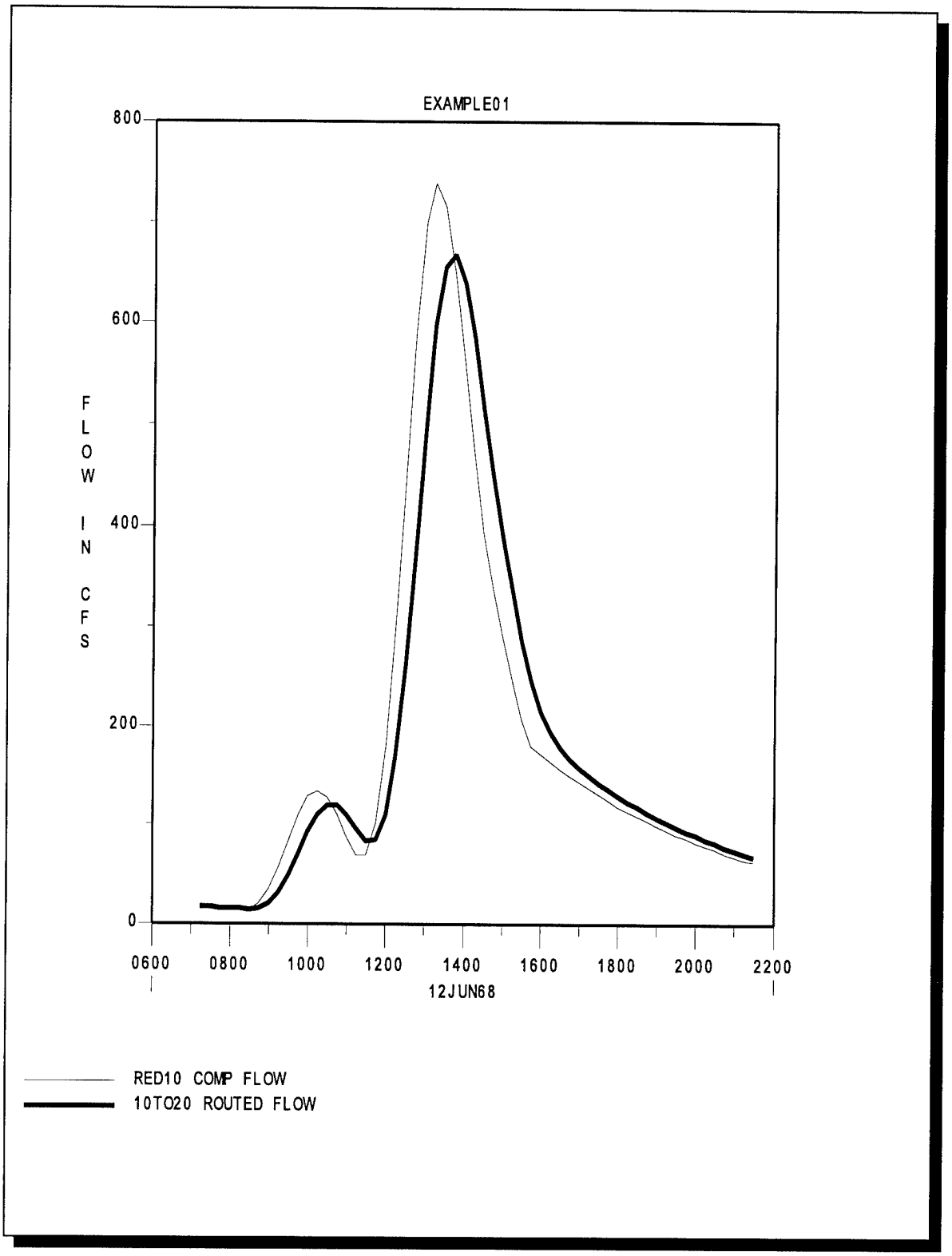
Example: The following example uses the PULS function to perform a PULS routing on a computed hydrograph.

Macro:

```
MACRO TPULS
** COMPUTE A PULS ROUTING
CL ALL
TIME 12JUN1968 0600 12JUN1968 2200
GET PD=HEC101:/EXAMPLE01/10TO20/STOR-FLOW////
GET FLOW=/EXAMPLE01/RED10/FLOW/12JUN1968/15MIN/COMP/
COMP RFLOW=PULS(FLOW,PD,0.0,1,-1,-1)
PUT.A RFLOW=B=TPULS,F=CAL
$CONTINUE INCASE OF ERROR
ENDMACRO
```

Execution:

```
I>!R TPULS
 ALL VARIABLES HAVE BEEN CLEARED
   -----DSS---ZOPEN:  Existing File Opened,  File: HEC101.DSS
                      Unit:  71;  DSS Version: 6-IA
-----DSS--- ZREAD Unit  71; Vers.    1:  /EXAMPLE01/10TO20/STOR-FLOW////
-----DSS--- ZREAD Unit  71; Vers.    1:  /EXAMPLE01/RED10/FLOW/01JUN1968/15MIN/COMP/
-----DSS---ZWRITE Unit  71; Vers.    1:  /EXAMPLE01/TPULS/FLOW/01JUN1968/15MIN/CAL/
```


EXAMPLE01
800
600
400
200
0
FLOW IN CFS
0600
0800
1000
1200
1400
1600
1800
2000
2200
12JUN68
RED10 COMP FLOW
10TO20 ROUTED FLOW

Name: **QAC** **Flow accumulator gage processor**
Use: CO TY=QAC(TX,TC)

Compute period-average flows from a flow accumulator type gage:

$$TY(t) = (TX(t)-TX(t-1))/(TC(t)-TC(t-1))$$

TX and TC are respectively, time series of the accumulated flow and the count. TX and TC values must occur at the same times. If corresponding values for TX and TC decrease from the previous period, then the accumulation is assumed to have reset to zero at the beginning of the interval. An IF condition has no effect. TY is assigned the type 'PER-AVER.'

Name: **RND** **Round off**
Use: CO TY=RND(TX,NDIG,IPLAC)

Round off values to NDIG or IPLAC, whichever controls. NDIG is the number of digits to round to and can range from 1 to 8. IPLAC is a magnitude of 10 to which to round to: for example, -1 specifies rounding to one-tenth (0.1). The number of digits shown is never less than 1, however. The following example illustrates the effects on rounding if NDIG=3 and IPLACE=0 (ie., round to ones place):

Rounds	to
1445.1	1450.
144.51	145.
14.451	14.
1.4451	1.

If TX is undefined, TY is undefined. If a concurrent IF condition is unsatisfied, TY is unchanged.

Example: The following example uses the QAC function to take cumulative flow readings (15 minute) and their count and convert them into period average flow values. The RND function is used to round the flow values.

Macro:

```
MACRO TQAC1
TIME 01JAN93 0100 30JAN93 2400
!R HQR2 TESTDB.DSS TESTDB.DSS 'STONY CR' 'BLACK BUTTE' FLOW
ENDMACRO
MACRO HQR2 %INDB %QRDB %APART %BPART %CPART
****|**********|****************************|****************************
**  Compute     Hourly FLOW or FLOW-RES OUT     for NO USGS RTable
**              Used when no USGS rating table is available
**  INPUT:
**    QV$       FLOW-CUM (Accumulated flow, cfs, HADA-PO).  Be sure to
retrieve
**              the previous value outside the time window.
**    NC$       N COUNT (N count, #, HADA-PO)
**  OUTPUT:
**    QR$       FLOW (OUTFLOW, STREAM FLOW, cfs) rounded to tenth if < 10 cfs
**              interpolated to 1 hour values
*********************************************************************
*
GE.P QV$=%INDB:/%APART/%BPART/FLOW-CUM//1HOUR//
GE.P NC$=%INDB:/%APART/%BPART/N COUNT//1HOUR//
CO QR$=QAC(QV$,NC$)
CO QRR$=QR$
!RUN RNDQR  QRR$ -1
SD QR$ UNITS=CFS TYPE=PER-AVER
PU.M QR$=%QRDB:/%APART/%BPART/%CPART//1HOUR//
$CONTINUE
ENDMACRO
```

```
MACRO RNDQR %QR %IPLAC
****|**********|****************************|*********************
**  Round       FLOW - HADA Gages              IF FLOW < 0 THEN set to 0
**  Round Flows -- HADA gaged stations
**
**  INPUT/OUTPUT:
**          %QR      Time series of flows to be rounded
**          %IPLAC   Magnitude of 10 to round to ( -1 rounds to one-tenth)
*********************************************************************
***
** Round flow to nearest %IPLAC cfs for all flows
CO %QR=RND(%QR,8,%IPLAC)
** Round flow to nearest one cfs for flows > 10
CO IF(%QR GE 10) %QR=RND(%QR,8,0)
** account for negative flows caused by RTABLE doing negative interpolation
CO IF(%QR LT 0) %QR=0
ENDMACRO
```

Execution:

```
I>!R TQAC1
   -----DSS---ZOPEN:  Existing File Opened,  File: TESTDB.DSS
                      Unit:  71;  DSS Version: 6-IA
-----DSS--- ZREAD Unit  71; Vers.  749:  /STONY CR/BLACK BUTTE/FLOW-CUM/01JAN1993/1HOUR//
-----DSS--- ZREAD Unit  71; Vers.  749:  /STONY CR/BLACK BUTTE/N COUNT/01JAN1993/1HOUR//
WARNING -- EXISTING VARIABLE RE-DEFINED
WARNING -- EXISTING VARIABLE RE-DEFINED
-----DSS---ZWRITE Unit  71; Vers.    2:  /STONY CR/BLACK BUTTE/FLOW/01JAN1993/1HOUR//
Variables ----      QV$          NC$          QR$          QRR$
Units     ----      CFS                       CFS           CFS
Time   Date
0100 01JAN1993     2116.000        4.000        M            M
0200 01JAN1993     4232.000        8.000      529.000      529.000
0300 01JAN1993     6371.000       12.000      534.750      535.000
0400 01JAN1993     8510.000       16.000      534.750      535.000
0500 01JAN1993    10650.000       20.000      535.000      535.000
0600 01JAN1993    12789.000       24.000      534.750      535.000
0700 01JAN1993    14928.000       28.000      534.750      535.000
0800 01JAN1993    17067.000       32.000      534.750      535.000
0900 01JAN1993    19206.000       36.000      534.750      535.000
1000 01JAN1993    21346.000       40.000      535.000      535.000
1100 01JAN1993    23485.000       44.000      534.750      535.000
1200 01JAN1993    27120.000       48.000      908.750      909.000
1300 01JAN1993    32592.000       52.000     1368.000     1368.000
1400 01JAN1993    40215.000       56.000     1905.750     1906.000
1500 01JAN1993    49915.000       60.000     2425.000     2425.000
1600 01JAN1993    59930.000       64.000     2503.750     2504.000
1700 01JAN1993    69945.000       68.000     2503.750     2504.000
1800 01JAN1993    79960.000       72.000     2503.750     2504.000
1900 01JAN1993    89976.000       76.000     2504.000     2504.000
2000 01JAN1993    99991.000       80.000     2503.750     2504.000
2100 01JAN1993   110056.000       84.000     2516.250     2516.000
2200 01JAN1993   120104.000       88.000     2512.000     2512.000
2300 01JAN1993   130119.000       92.000     2503.750     2504.000
2400 01JAN1993   140135.000       96.000     2504.000     2504.000
0100 02JAN1993    10015.000        4.000     2503.750     2504.000
0200 02JAN1993    20030.000        8.000     2503.750     2504.000
0300 02JAN1993    30046.000       12.000     2504.000     2504.000
0400 02JAN1993    40061.000       16.000     2503.750     2504.000
0500 02JAN1993    50076.000       20.000     2503.750     2504.000
0600 02JAN1993    60091.000       24.000     2503.750     2504.000
0700 02JAN1993    70106.000       28.000     2503.750     2504.000
0800 02JAN1993    80122.000       32.000     2504.000     2504.000
0900 02JAN1993    89589.000       36.000     2366.750     2367.000
1000 02JAN1993    98741.000       40.000     2288.000     2288.000
1100 02JAN1993   110323.000       44.000     2895.500     2896.000
1200 02JAN1993   124563.000       48.000     3560.000     3560.000
1300 02JAN1993   140703.000       52.000     4035.000     4035.000
1400 02JAN1993   158913.000       56.000     4552.500     4553.000
1500 02JAN1993   181581.000       60.000     5667.000     5667.000
1600 02JAN1993   202367.000       64.000     5196.500     5197.000
1700 02JAN1993   222702.000       68.000     5083.750     5084.000
1800 02JAN1993   242813.000       72.000     5027.750     5028.000
1900 02JAN1993   262924.000       76.000     5027.750     5028.000
2000 02JAN1993   283035.000       80.000     5027.750     5028.000
2100 02JAN1993   303147.000       84.000     5028.000     5028.000
2200 02JAN1993   323258.000       88.000     5027.750     5028.000
2300 02JAN1993   343258.000       92.000     5000.000     5000.000
2400 02JAN1993   363258.000       96.000     5000.000     5000.000
0100 03JAN1993    20000.000        4.000     5000.000     5000.000
0200 03JAN1993    39884.000        8.000     4971.000     4971.000
0300 03JAN1993    59769.000       12.000     4971.250     4971.000
0400 03JAN1993    79653.000       16.000     4971.000     4971.000
0500 03JAN1993    99538.000       20.000     4971.250     4971.000
0600 03JAN1993   119422.000       24.000     4971.000     4971.000
```

Name: **RTABLE** **Rating table interpolation**
Use: CO TY=RTABLE(TX,TB)

Interpolate values for TX using table TB and store the result in TY. TB must be created using the program DSSPD (use /R option) with specific information in the header: type of interpolation, offset, shift, and datum. If the type of interpolation is LOGLOG, table x values are adjusted by subtracting the offset. The shift is added to and the datum subtracted from all incoming TX values. The header information in TB also defines the units of TY.

Name: **RTABLR** **Reverse rating table interpolation**
Use: CO TY=RTABLR(TX,TB)

Interpolate values for TX using the reverse of table TB and store the result in TY. TB must be created using the program DSSPD (option /R) with specific information in the header: type of interpolation, offset, shift, and datum. If the type of interpolation is LOGLOG, table x values are adjusted by subtracting the offset. The shift is subtracted from and the datum added to all resulting TX values. The header information in TB also defines the units of TY.

Example: The following example uses the RTABLE and RTABLR functions to compute flow values at Black Butte reservoir.

Macro:

```
MACRO TRTABL1
TIME 01JAN93 0100 30JAN93 2400
!R HQR1 TESTDB.DSS TESTDB.DSS 'STONY CR' 'BLACK BUTTE' FLOW
ENDMACRO

MACRO RNDQR %QR %IPLAC
****|**********|*****************************|************************
**  Round      FLOW - HADA Gages              IF FLOW < 0 THEN set to 0
**  Round Flows -- HADA gaged stations
**
**  INPUT/OUTPUT:
**        %QR       Time series of flows to be rounded
**        %IPLAC    Magnitude of 10 to round to ( -1 rounds to one-tenth)
*********************************************************************
** Round flow to nearest %IPLAC cfs for all flows
CO %QR=RND(%QR,8,%IPLAC)
** Round flow to nearest one cfs for flows > 10
CO IF(%QR GE 10) %QR=RND(%QR,8,0)
** account for negative flows caused by RTABLE doing negative interpolation
CO IF(%QR LT 0) %QR=0
ENDMACRO
```

```
MACRO HQR1 %INDB %QRDB %APART %BPART %CPART
****|**********|*****************************|****************************
**  Compute    Hourly FLOW or FLOW-RES OUT    w/ HADA & USGS RTable
**  Compute FLOW (from accumulated flow and N count ,ESI) Type 1 Short Period
**                 used when a HADA PO and USGS rating table exists
**  INPUT:
**    %INDB    Database for ACCUMULATED FLOW and N-COUNT
**    %QRDB    Flow Database
**    %APART   'A' part of pathname
**    %BPART   'B' part of pathname
**    %CPART   Flow type being computed -- 'C' part -- (FLOW-RES OUT or FLOW)
**  OUTPUT:
**    QR$      FLOW (OUTFLOW, STREAM FLOW, cfs) rounded to tenth if < 10 cfs
**
*********************************************************************
*
GE.P QV$=%INDB:/%APART/%BPART/FLOW-CUM//1HOUR//
GE.P NC$=%INDB:/%APART/%BPART/N COUNT//1HOUR//
GE HADA$=TESTDB.DSS:/%APART/%BPART/STAGE-FLOW///HADA PO/
GE USGS$=TESTDB.DSS:/%APART/%BPART/STAGE-FLOW///USGS/
CO TQR$=QAC(QV$,NC$)
CO AHR$=RTABLR(TQR$,HADA$)
CO QR$=RTABLR(AHR$,USGS$)
!RUN RNDQR  QR$ -1
SD QR$ UNITS=CFS TYPE=PER-AVER
PU.M QR$=%QRDB:/%APART/%BPART/%CPART//1HOUR//
$CONTINUE
ENDMACRO
```

Execution:

```
I>!R TRTABL1
   -----DSS---ZOPEN:  Existing File Opened,  File: TESTDB.DSS
                      Unit:  71;  DSS Version: 6-IA
-----DSS--- ZREAD Unit  71; Vers.  749:  /STONY CR/BLACK BUTTE/FLOW-CUM/01JAN1993/1HOUR//
-----DSS--- ZREAD Unit  71; Vers.  749:  /STONY CR/BLACK BUTTE/N COUNT/01JAN1993/1HOUR//
-----DSS--- ZREAD Unit  71; Vers.    8:  /STONY CR/BLACK BUTTE/STAGE-FLOW///HADA PO/
-----DSS--- ZREAD Unit  71; Vers.   10:  /STONY CR/BLACK BUTTE/STAGE-FLOW///USGS/
LINLIN TRANSFORMATION
OFFSET =        .00     SHIFT =        .00     DATUM =        .00
LOGLOG TRANSFORMATION
OFFSET =        .90     SHIFT =        .00     DATUM =        .00
WARNING -- EXISTING VARIABLE RE-DEFINED
WARNING -- EXISTING VARIABLE RE-DEFINED
-----DSS---ZWRITE Unit  71; Vers.    7:  /STONY CR/BLACK BUTTE/FLOW/01JAN1993/1HOUR//
```

Variables ----		AHR$	TQR$	QR$
Units ----		FEET	CFS	CFS
Time	Date			
0100	01JAN1993	M	M	M
0200	01JAN1993	4.550	529.000	525.000
0300	01JAN1993	4.560	534.750	530.000
0400	01JAN1993	4.560	534.750	530.000
0500	01JAN1993	4.560	535.000	530.000
0600	01JAN1993	4.560	534.750	530.000
0700	01JAN1993	4.560	534.750	530.000
0800	01JAN1993	4.560	534.750	530.000
0900	01JAN1993	4.560	534.750	530.000
1000	01JAN1993	4.560	535.000	530.000
1100	01JAN1993	4.560	534.750	530.000
1200	01JAN1993	5.145	908.750	896.000
1300	01JAN1993	5.642	1368.000	1351.000
1400	01JAN1993	6.070	1905.750	1892.000
1500	01JAN1993	6.383	2425.000	2403.000
1600	01JAN1993	6.430	2503.750	2489.000
1700	01JAN1993	6.430	2503.750	2489.000
1800	01JAN1993	6.430	2503.750	2489.000
1900	01JAN1993	6.430	2504.000	2489.000
2000	01JAN1993	6.430	2503.750	2489.000
2100	01JAN1993	6.438	2516.250	2503.000
2200	01JAN1993	6.435	2512.000	2498.000
2300	01JAN1993	6.430	2503.750	2489.000
2400	01JAN1993	6.430	2504.000	2489.000
0100	02JAN1993	6.430	2503.750	2489.000
0200	02JAN1993	6.430	2503.750	2489.000
0300	02JAN1993	6.430	2504.000	2489.000
0400	02JAN1993	6.430	2503.750	2489.000
0500	02JAN1993	6.430	2503.750	2489.000
0600	02JAN1993	6.430	2503.750	2489.000
0700	02JAN1993	6.430	2503.750	2489.000
0800	02JAN1993	6.430	2504.000	2489.000
0900	02JAN1993	6.347	2366.750	2341.000
1000	02JAN1993	6.300	2288.000	2259.000
1100	02JAN1993	6.628	2895.500	2869.000
1200	02JAN1993	6.935	3560.000	3544.000
1300	02JAN1993	7.129	4035.000	4008.000
1400	02JAN1993	7.328	4552.500	4520.000
1500	02JAN1993	7.691	5667.000	5548.000
1600	02JAN1993	7.556	5196.500	5157.000
1700	02JAN1993	7.524	5083.750	5067.000
1800	02JAN1993	7.508	5027.750	5022.000
1900	02JAN1993	7.508	5027.750	5022.000
2000	02JAN1993	7.508	5027.750	5022.000
2100	02JAN1993	7.508	5028.000	5022.000
2200	02JAN1993	7.508	5027.750	5022.000
2300	02JAN1993	7.500	5000.000	5000.000
2400	02JAN1993	7.500	5000.000	5000.000
0100	03JAN1993	7.500	5000.000	5000.000
0200	03JAN1993	7.489	4971.000	4968.000
0300	03JAN1993	7.489	4971.250	4968.000
0400	03JAN1993	7.489	4971.000	4968.000
0500	03JAN1993	7.489	4971.250	4968.000
0600	03JAN1993	7.489	4971.000	4968.000
0700	03JAN1993	7.478	4942.250	4936.000
0800	03JAN1993	7.478	4942.250	4936.000
0900	03JAN1993	7.287	4445.250	4410.000
1000	03JAN1993	6.653	2951.250	2922.000
1100	03JAN1993	6.653	2951.500	2922.000

Paired Data Variables

```
VARIABLE=HADA$
 UNITS = FEET        CFS
 TYPES = UNT         UNT
 LABEL = FLOW
 SHIFT =       0.0   OFFSET =       0.0   DATUM =       0.0

   X-AXIS          FLOW
       .0000        .0000
       .9000        .0000
      1.0000        .1000
      1.1000        .3000
      1.2000        .7000
      1.3000       1.3000
      1.4000       2.3000
      1.5000       3.5000
      1.6000       4.9000
      1.7000       7.0000
      1.8000       9.8000
      2.0000      17.7000
      2.2000      28.3000
      2.4000      41.8000
      2.6000      57.3000
      2.7000      65.8000
      2.8000      76.0000
      3.0000     101.0000
      3.2500     139.9000
      3.5000     184.0000
      4.0000     314.0000
      4.5000     500.0000
      5.0000     790.0000
      5.5000    1200.0000
      6.0000    1790.0000
      6.5000    2620.0000
      7.0000    3700.0000
      7.5000    5000.0000
      9.0000   10250.0000
     11.5000   24000.0000
```

```
VARIABLE=USGS$
 UNITS = FEET        CFS
 TYPES = UNT         UNT
 LABEL = FLOW
 SHIFT =       0.0   OFFSET =       0.9   DATUM =       0.0

   X-AXIS          FLOW
       .0000        .0000
       .9000        .0000
      1.0000        .1000
      1.1000        .3000
      1.2000        .7000
      1.3000       1.3000
      1.4000       2.3000
      1.5000       3.5000
      1.6000       4.9000
      1.7000       7.0000
      1.8000       9.8000
      2.0000      17.7000
      2.2000      28.3000
      2.4000      41.8000
      2.6000      57.3000
      2.7000      65.8000
      2.8000      76.0000
      3.0000     101.0000
      3.2500     139.9000
      3.5000     184.0000
      4.0000     314.0000
      4.5000     500.0000
      5.0000     790.0000
      5.5000    1200.0000
      6.0000    1790.0000
      6.5000    2620.0000
      7.0000    3700.0000
      7.5000    5000.0000
      8.0000    6530.0000
      9.0000   10250.0000
      9.5000   12500.0000
     11.5000   24000.0000
```

Name: **RTABL2** **Two-variable rating table interpolation**
Use: CO TY=RTABL2(TX,TZ,TB)

TY is a function of two independent variables TX and TZ. The functional relationship is specified by the table TB, which consists of sets of TX/TY pairs, each set corresponding to a TZ value. TB is created with the program DSSPD (option /R) with values for TZ specified as labels for the sets of TX/TY pairs. RTABL2 interpolates linearly in table TB. No extrapolation is done: if the TX or TY value is outside the range bounded by TB, TY is set to missing. TX and TZ must be concurrent time series. The header information in TB also defines the units of TY.

Example: The following example uses the RTABL2 function to calculate the outflows from Lynx Lake from stage readings at a gage (LYNX) downstream from the outlet works. The gage rating is effected by backwater on the Cheat River. A family of rating curves has been established for LYNX gage that relates each rating curve to a stage reading at the PAW gage.

Macro:

```
MACRO TRTABL2
CL ALL
TIME 28NOV1993 0100 02DEC1993 2400
GET LABL1=TESTDB.DSS:/MONONGAHELA/PMRP1/STAGE-L/01JUN1989/1DAY/OBS/
CO LABL2=ESTLIN(LABL1,24,M)
CO LABLE=TS1(LABL2,1H,0H)
TAB.F LABL1 LABL2 LABLE
PUT LABLE=TESTDB.DSS:/MONONGAHELA/PMRP/STAGE/01JUN1989/1HOUR/CMPTD/
TIME 01DEC1993 0100 02DEC1993 2400
GET TABLE=TESTDB.DSS:/MONONGAHELA/PMAW/STAGE-FLOW////
GET LABLE=TESTDB.DSS:/MONONGAHELA/PMRP/STAGE/01JUN1989/1HOUR/CMPTD/
GET STAGE=TESTDB.DSS:/MONONGAHELA/PMAW/STAGE/01JUN1989/1HOUR/OBS/
COMP FLOW=RTABL2(STAGE,LABLE,TABLE)
SD FLOW U=CFS TY=INST-VAL
PUT FLOW=B=LNNW,C=FLOW,F=CAL
ENDMACRO
```

Execution:

```
I>!R TRTABL2
 ALL VARIABLES HAVE BEEN CLEARED
    -----DSS---ZOPEN:  Existing File Opened,  File: TESTDB.DSS
                       Unit:  71;  DSS Version: 6-IA
 -----DSS--- ZREAD Unit  71; Vers.    1:  /MONONGAHELA/PMRP1/STAGE-L/01JAN1993/1DAY/OBS/
 -----DSS---ZWRITE Unit  71; Vers.    2:  /MONONGAHELA/PMRP/STAGE/01NOV1993/1HOUR/CMPTD/
 -----DSS---ZWRITE Unit  71; Vers.    3:  /MONONGAHELA/PMRP/STAGE/01DEC1993/1HOUR/CMPTD/
 WARNING - TIME SERIES VARIABLES WERE CLEARED
 BECAUSE START DAY OF TIME WINDOW HAS CHANGED.
 NOTE:  CLEAR CAN BE SUPPRESSED VIA .N OPTION.
 -----DSS--- ZREAD Unit  71; Vers.    4:  /MONONGAHELA/PMAW/STAGE-FLOW////
 -----DSS--- ZREAD Unit  71; Vers.    3:  /MONONGAHELA/PMRP/STAGE/01DEC1993/1HOUR/CMPTD/
 -----DSS--- ZREAD Unit  71; Vers.  200:  /MONONGAHELA/PMAW/STAGE/01DEC1993/1HOUR/OBS/
 -----DSS---ZWRITE Unit  71; Vers.    2:  /MONONGAHELA/LNNW/FLOW/01DEC1993/1HOUR/CAL/
```

Variables ----	LABL1	LABL2	LABLE
Units ----	FEET	FEET	FEET
Time Date			
0100 28NOV1993	-	-	M
0200 28NOV1993	-	-	M
0300 28NOV1993	-	-	M
0400 28NOV1993	-	-	M
0500 28NOV1993	-	-	M
0600 28NOV1993	-	-	M
0700 28NOV1993	13.700	13.700	13.700
0800 28NOV1993	-	-	13.675
0900 28NOV1993	-	-	13.650
1000 28NOV1993	-	-	13.625
1100 28NOV1993	-	-	13.600
1200 28NOV1993	-	-	13.575
1300 28NOV1993	-	-	13.550
1400 28NOV1993	-	-	13.525
1500 28NOV1993	-	-	13.500
1600 28NOV1993	-	-	13.475
1700 28NOV1993	-	-	13.450
1800 28NOV1993	-	-	13.425
1900 28NOV1993	-	-	13.400
2000 28NOV1993	-	-	13.375
2100 28NOV1993	-	-	13.350
2200 28NOV1993	-	-	13.325
2300 28NOV1993	-	-	13.300
2400 28NOV1993	-	-	13.275
0100 29NOV1993	-	-	13.250
0200 29NOV1993	-	-	13.225
0300 29NOV1993	-	-	13.200
0400 29NOV1993	-	-	13.175
0500 29NOV1993	-	-	13.150
0600 29NOV1993	-	-	13.125
0700 29NOV1993	13.100	13.100	13.100
0800 29NOV1993	-	-	13.104
0900 29NOV1993	-	-	13.108
1000 29NOV1993	-	-	13.113
1100 29NOV1993	-	-	13.117
1200 29NOV1993	-	-	13.121
1300 29NOV1993	-	-	13.125
1400 29NOV1993	-	-	13.129
1500 29NOV1993	-	-	13.133
1600 29NOV1993	-	-	13.137
1700 29NOV1993	-	-	13.142
1800 29NOV1993	-	-	13.146
1900 29NOV1993	-	-	13.150
2000 29NOV1993	-	-	13.154
2100 29NOV1993	-	-	13.158
2200 29NOV1993	-	-	13.163
2300 29NOV1993	-	-	13.167
2400 29NOV1993	-	-	13.171
0100 30NOV1993	-	-	13.175
0200 30NOV1993	-	-	13.179
0300 30NOV1993	-	-	13.183
0400 30NOV1993	-	-	13.188
0500 30NOV1993	-	-	13.192
0600 30NOV1993	-	-	13.196
0700 30NOV1993	13.200	13.200	13.200
0800 30NOV1993	-	-	13.167
0900 30NOV1993	-	-	13.133
1000 30NOV1993	-	-	13.100
1100 30NOV1993	-	-	13.067
1200 30NOV1993	-	-	13.033
1300 30NOV1993	-	-	13.000
1400 30NOV1993	-	-	12.967
1500 30NOV1993	-	-	12.933
1600 30NOV1993	-	-	12.900
1700 30NOV1993	-	-	12.867
1800 30NOV1993	-	-	12.833
1900 30NOV1993	-	-	12.800
2000 30NOV1993	-	-	12.767
2100 30NOV1993	-	-	12.733
2200 30NOV1993	-	-	12.700
2300 30NOV1993	-	-	12.667
2400 30NOV1993	-	-	12.633
0100 01DEC1993	-	-	12.600
0200 01DEC1993	-	-	12.567
0300 01DEC1993	-	-	12.533
0400 01DEC1993	-	-	12.500
0500 01DEC1993	-	-	12.467
0600 01DEC1993	-	-	12.433
0700 01DEC1993	12.400	12.400	12.400
0800 01DEC1993	-	-	12.383
0900 01DEC1993	-	-	12.367
1000 01DEC1993	-	-	12.350
1100 01DEC1993	-	-	12.333
1200 01DEC1993	-	-	12.317
1300 01DEC1993	-	-	12.300
1400 01DEC1993	-	-	12.283
1500 01DEC1993	-	-	12.267
1600 01DEC1993	-	-	12.250
1700 01DEC1993	-	-	12.233
1800 01DEC1993	-	-	12.217
1900 01DEC1993	-	-	12.200
2000 01DEC1993	-	-	12.183
2100 01DEC1993	-	-	12.167
2200 01DEC1993	-	-	12.150
2300 01DEC1993	-	-	12.133
2400 01DEC1993	-	-	12.117
0100 02DEC1993	-	-	12.100
0200 02DEC1993	-	-	12.083
0300 02DEC1993	-	-	12.067
0400 02DEC1993	-	-	12.050
0500 02DEC1993	-	-	12.033
0600 02DEC1993	-	-	12.017
0700 02DEC1993	12.000	12.000	12.000
0800 02DEC1993	-	-	M
0900 02DEC1993	-	-	M
1000 02DEC1993	-	-	M
1100 02DEC1993	-	-	M
1200 02DEC1993	-	-	M
1300 02DEC1993	-	-	M
1400 02DEC1993	-	-	M
1500 02DEC1993	-	-	M
1600 02DEC1993	-	-	M
1700 02DEC1993	-	-	M
1800 02DEC1993	-	-	M
1900 02DEC1993	-	-	M
2000 02DEC1993	-	-	M
2100 02DEC1993	-	-	M
2200 02DEC1993	-	-	M
2300 02DEC1993	-	-	M
2400 02DEC1993	-	-	M
0100 01DEC1993	12.600	10.700	6804.998
0200 01DEC1993	12.567	10.730	6896.498
0300 01DEC1993	12.533	10.730	6896.498
0400 01DEC1993	12.500	10.730	6896.498
0500 01DEC1993	12.467	10.700	6804.998
0600 01DEC1993	12.433	10.460	6173.333
0700 01DEC1993	12.400	10.520	6313.333
0800 01DEC1993	12.383	10.490	6243.332
0900 01DEC1993	12.367	10.460	6173.333
1000 01DEC1993	12.350	10.460	6173.333
1100 01DEC1993	12.333	10.460	6173.333
1200 01DEC1993	12.317	10.460	6173.333
1300 01DEC1993	12.300	10.460	6173.333
1400 01DEC1993	12.283	10.460	6173.333
1500 01DEC1993	12.267	10.490	6243.332
1600 01DEC1993	12.250	10.460	6173.333
1700 01DEC1993	12.233	10.490	6243.332
1800 01DEC1993	12.217	10.460	6173.333
1900 01DEC1993	12.200	10.460	6173.333
2000 01DEC1993	12.183	10.460	6173.333
2100 01DEC1993	12.167	10.460	6173.333

Paired Data Variable (*TABLE*)

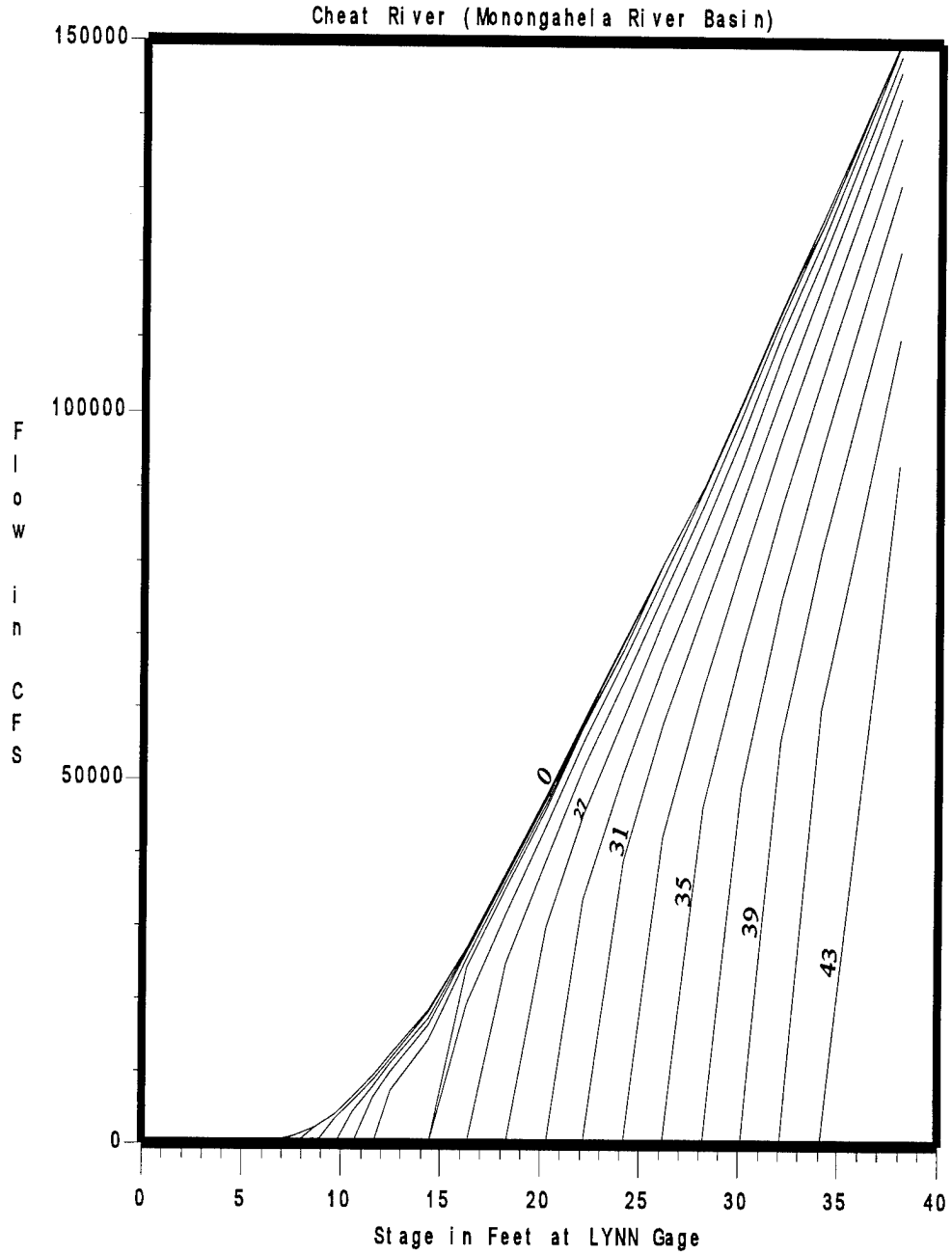
```
VARIABLE=TABLE
UNITS = FEET        CFS
TYPES = UNT         UNT
LABEL = 0
LABEL = 14
LABEL = 15
LABEL = 16
LABEL = 17
LABEL = 18
LABEL = 19
LABEL = 20
LABEL = 21
LABEL = 23
LABEL = 25
LABEL = 27
LABEL = 29
LABEL = 31
LABEL = 33
LABEL = 35
LABEL = 37
LABEL = 39
LABEL = 41
LABEL = 43
SHIFT= 0.0 OFFSET=0.0 DATUM= 0.00
```

X-AXIS	0	14	15	16	17
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
5.0000	0.0000	0.0000	0.0000	0.0000	0.0000
6.7000	600.0000	600.0000	0.0000	0.0000	0.0000
7.7000	1400.0000	1400.0000	1350.0000	0.0000	0.0000
8.7000	2600.0000	2600.0000	2550.0000	2400.0000	0.0000
9.7000	4400.0000	4400.0000	4400.0000	4400.0000	3550.0000
10.6000	6500.0000	6500.0000	6500.0000	6500.0000	6050.0000
11.6000	9550.0000	9550.0000	9550.0000	9550.0000	9100.0000
12.5000	12400.0000	12400.0000	12400.0000	12400.0000	12000.0000
14.4000	18800.0000	18800.0000	18800.0000	18800.0000	18500.0000
16.3000	27200.0000	27200.0000	27200.0000	27200.0000	27200.0000
18.3000	37400.0000	37400.0000	37400.0000	37400.0000	37400.0000
20.3000	48000.0000	48000.0000	48000.0000	48000.0000	48000.0000
22.2000	58500.0000	58500.0000	58500.0000	58500.0000	58500.0000
24.2000	69500.0000	69500.0000	69500.0000	69500.0000	69500.0000
26.2000	80000.0000	80000.0000	80000.0000	80000.0000	80000.0000
28.2000	90500.0000	90500.0000	90500.0000	90500.0000	90500.0000
30.1000	102000.0000	102000.0000	102000.0000	102000.0000	102000.0000
32.1000	114400.0000	114400.0000	114400.0000	114400.0000	114400.0000
34.1000	126300.0000	126300.0000	126300.0000	125300.0000	126300.0000
38.0000	150400.0000	150400.0000	150400.0000	150400.0000	150400.0000

X-AXIS	18	19	20	21	23
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
5.0000	0.0000	0.0000	0.0000	0.0000	0.0000
6.7000	0.0000	0.0000	0.0000	0.0000	0.0000
7.7000	0.0000	0.0000	0.0000	0.0000	0.0000
8.7000	0.0000	0.0000	0.0000	0.0000	0.0000
9.7000	0.0000	0.0000	0.0000	0.0000	0.0000
10.6000	4700.0000	0.0000	0.0000	0.0000	0.0000
11.6000	8200.0000	6500.0000	0.0000	0.0000	0.0000
12.5000	11400.0000	10000.0000	7600.0000	0.0000	0.0000
14.4000	17500.0000	16500.0000	14500.0000	0.0000	0.0000
16.3000	27000.0000	26500.0000	25700.0000	24500.0000	19500.0000
18.3000	37200.0000	37000.0000	36400.0000	35300.0000	32000.0000
20.3000	47900.0000	47500.0000	47000.0000	46400.0000	44000.0000
22.2000	58500.0000	58000.0000	57900.0000	57500.0000	55500.0000
24.2000	69500.0000	69400.0000	68500.0000	68500.0000	67000.0000
26.2000	80000.0000	80000.0000	80000.0000	80000.0000	78500.0000
28.2000	90500.0000	90500.0000	90500.0000	90500.0000	90000.0000
30.1000	102000.0000	102000.0000	102000.0000	102000.0000	101500.0000
32.1000	114400.0000	114400.0000	114400.0000	114400.0000	114000.0000
34.1000	126300.0000	126300.0000	126300.0000	126300.0000	126300.0000
38.0000	150400.0000	150400.0000	150400.0000	150400.0000	150400.0000

X-AXIS	25	27	29	31	33
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
5.0000	0.0000	0.0000	0.0000	0.0000	0.0000
6.7000	0.0000	0.0000	0.0000	0.0000	0.0000
7.7000	0.0000	0.0000	0.0000	0.0000	0.0000
8.7000	0.0000	0.0000	0.0000	0.0000	0.0000
9.7000	0.0000	0.0000	0.0000	0.0000	0.0000
10.6000	0.0000	0.0000	0.0000	0.0000	0.0000
11.6000	0.0000	0.0000	0.0000	0.0000	0.0000
12.5000	0.0000	0.0000	0.0000	0.0000	0.0000
14.4000	0.0000	0.0000	0.0000	0.0000	0.0000
16.3000	0.0000	0.0000	0.0000	0.0000	0.0000
18.3000	25100.0000	0.0000	0.0000	0.0000	0.0000
20.3000	39400.0000	30000.0000	0.0000	0.0000	0.0000
22.2000	52000.0000	45500.0000	34000.0000	0.0000	0.0000
24.2000	64000.0000	59000.0000	51300.0000	39200.0000	0.0000
26.2000	76000.0000	72500.0000	66200.0000	58000.0000	42500.0000
28.2000	88000.0000	85000.0000	80000.0000	73900.0000	63000.0000
30.1000	100000.0000	97500.0000	93500.0000	88400.0000	80300.0000
32.1000	113000.0000	111000.0000	108000.0000	103000.0000	97000.0000
34.1000	125500.0000	123800.0000	121300.0000	117700.0000	112400.0000
38.0000	150400.0000	150000.0000	148400.0000	146300.0000	142800.0000

X-AXIS	35	37	39	41	43
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
5.0000	0.0000	0.0000	0.0000	0.0000	0.0000
6.7000	0.0000	0.0000	0.0000	0.0000	0.0000
7.7000	0.0000	0.0000	0.0000	0.0000	0.0000
8.7000	0.0000	0.0000	0.0000	0.0000	0.0000
9.7000	0.0000	0.0000	0.0000	0.0000	0.0000
10.6000	0.0000	0.0000	0.0000	0.0000	0.0000
11.6000	0.0000	0.0000	0.0000	0.0000	0.0000
12.5000	0.0000	0.0000	0.0000	0.0000	0.0000
14.4000	0.0000	0.0000	0.0000	0.0000	0.0000
16.3000	0.0000	0.0000	0.0000	0.0000	0.0000
18.3000	0.0000	0.0000	0.0000	0.0000	0.0000
20.3000	0.0000	0.0000	0.0000	0.0000	0.0000
22.2000	0.0000	0.0000	0.0000	0.0000	0.0000
24.2000	0.0000	0.0000	0.0000	0.0000	0.0000
26.2000	0.0000	0.0000	0.0000	0.0000	0.0000
28.2000	46500.0000	0.0000	0.0000	0.0000	0.0000
30.1000	68400.0000	49300.0000	0.0000	0.0000	0.0000
32.1000	87800.0000	75000.0000	55800.0000	0.0000	0.0000
34.1000	105200.0000	95300.0000	82000.0000	60000.0000	0.0000
38.0000	137500.0000	131000.0000	122000.0000	110000.0000	93400.0000


Cheat River (Monongahela River Basin)
150000
100000
50000
0
Flow in CFS
0
5
10
15
20
25
30
35
40
Stage in Feet at LYNN Gage
0
32
31
35
39
43

Variables ----		LABLE	STAGE	FLOW
Units	----	FEET	FEET	CFS
Time	Date			
0100	01DEC1993	12.600	10.700	6804.998
0200	01DEC1993	12.567	10.730	6896.498
0300	01DEC1993	12.533	10.730	6896.498
0400	01DEC1993	12.500	10.730	6896.498
0500	01DEC1993	12.467	10.700	6804.998
0600	01DEC1993	12.433	10.460	6173.333
0700	01DEC1993	12.400	10.520	6313.333
0800	01DEC1993	12.383	10.490	6243.332
0900	01DEC1993	12.367	10.460	6173.333
1000	01DEC1993	12.350	10.460	6173.333
1100	01DEC1993	12.333	10.460	6173.333
1200	01DEC1993	12.317	10.460	6173.333
1300	01DEC1993	12.300	10.460	6173.333
1400	01DEC1993	12.283	10.460	6173.333
1500	01DEC1993	12.267	10.490	6243.332
1600	01DEC1993	12.250	10.460	6173.333
1700	01DEC1993	12.233	10.490	6243.332
1800	01DEC1993	12.217	10.460	6173.333
1900	01DEC1993	12.200	10.460	6173.333
2000	01DEC1993	12.183	10.460	6173.333
2100	01DEC1993	12.167	10.460	6173.333
2200	01DEC1993	12.150	10.460	6173.333
2300	01DEC1993	12.133	10.460	6173.333
2400	01DEC1993	12.117	10.460	6173.333
0100	02DEC1993	12.100	9.320	3716.000
0200	02DEC1993	12.083	9.290	3662.000
0300	02DEC1993	12.067	9.290	3662.000
0400	02DEC1993	12.050	9.260	3608.001
0500	02DEC1993	12.033	9.260	3608.001
0600	02DEC1993	12.017	9.260	3608.001
0700	02DEC1993	12.000	9.260	3608.001
0800	02DEC1993	11.967	10.400	6033.332
0900	02DEC1993	11.933	10.400	6033.332
1000	02DEC1993	11.900	11.090	7994.500
1100	02DEC1993	11.867	10.280	5753.333
1200	02DEC1993	11.833	10.340	5893.333
1300	02DEC1993	11.800	10.190	5543.332
1400	02DEC1993	11.767	10.610	6530.498
1500	02DEC1993	11.733	10.430	6103.333
1600	02DEC1993	11.700	10.730	6896.498
1700	02DEC1993	11.667	10.400	6033.332
1800	02DEC1993	11.633	10.700	6804.998
1900	02DEC1993	11.600	10.460	6173.333
2000	02DEC1993	11.567	11.000	7719.999
2100	02DEC1993	11.533	9.920	4913.333
2200	02DEC1993	11.500	9.650	4310.000
2300	02DEC1993	11.467	6.800	680.000
2400	02DEC1993	11.433	6.320	465.882

Name: **SCRN1** **Screen for possible erroneous values based on maximum/minimum range**

Use: CO TY=SCRN1(TX,XMIN,XMAX,MAXDEL,QFLAG)

Flag any value in TX that falls outside the range XMIN - XMAX or exceeds the maximum change MAXDEL from the previous value. The maximum change comparison is done only when the consecutive values are not flagged. Possible values for QFLAG are: M = missing data or Q = questionable. The result is placed in TY.

Name: **SCRN2** **Screen for possible erroneous values based on forward moving average maximum**

Use: CO TY=SCRN2(TX,NPTS,MAXDEL,QFLAG)

Flag any value in TX that exceeds the maximum change MAXDEL from the forward moving average of NPTS ending at the previous value. Missing values in TX are not counted in the moving average and the divisor of the average is less one for every missing value. Values which fail the screen are not counted either. At least 2 values must be defined else the moving average is undefined and the screen passes the relevant TX value. Possible flags are: M = missing data or Q = questionable. The result is placed in TY. SCRN2 is useful for detecting and removing spikes. The FMA function may be useful for determining appropriate NPTS and MAXDEL parameters.

Example: The following example uses the SCRN1 and SCRN2 functions to demonstrate their use in flagging possible bad data.

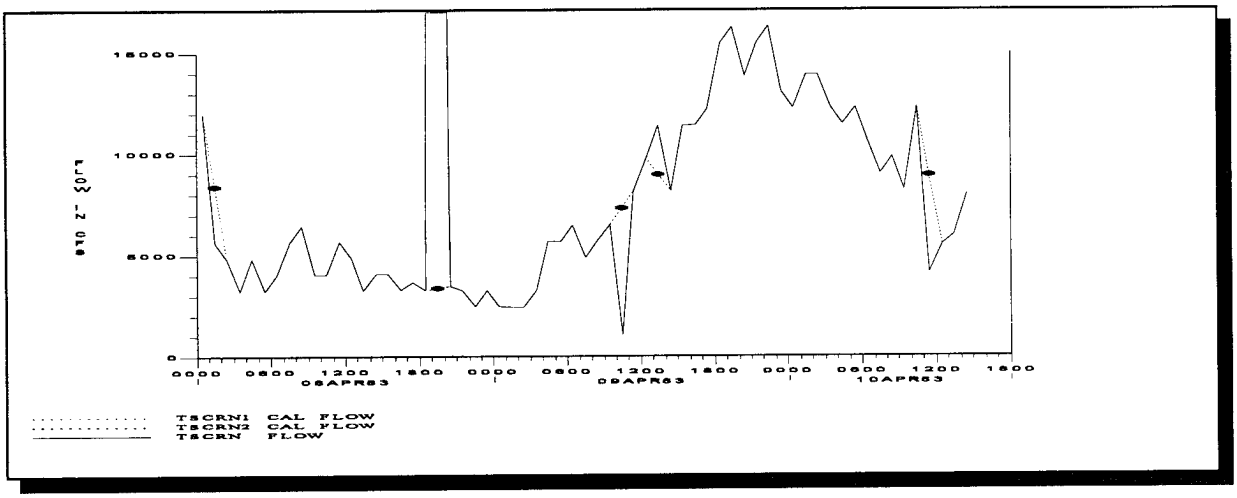
Macro:

```
MACRO TSCRN3
CL ALL
TIME 08APR1983 0100 10APR1983 1500
GET FLOW=testdb.dss://TSCRN/FLOW/01APR1983/1HOUR//
COMP SFLOW1=SCRN1(FLOW,0,50000,5000,M)
COMP SFLOW2=SCRN2(FLOW,4,5000,M)
PUT.A SFLOW1=B=TSCRN1,F=CAL
PUT.A SFLOW2=B=TSCRN2,F=CAL
TAB.F FLOW SFLOW1 SFLOW2
$CONTINUE INCASE OF ERROR
ENDMACRO
```

Execution:

```
I>!R TSCRN3
  ALL VARIABLES HAVE BEEN CLEARED
    -----DSS---ZOPEN:  Existing File Opened,  File: TESTDB.DSS
                       Unit:  71;  DSS Version: 6-IA
 -----DSS--- ZREAD Unit  71; Vers.    6:  //TSCRN/FLOW/01APR1983/1HOUR//
  DMIN,DMAX,DEL =    0.000000E+00    50000.000000     5000.000000
 -----DSS---ZWRITE Unit  71; Vers.    6:  //TSCRN1/FLOW/01APR1983/1HOUR/CAL/
 -----DSS---ZWRITE Unit  71; Vers.    6:  //TSCRN2/FLOW/01APR1983/1HOUR/CAL/
```

Variables ----	FLOW	SFLOW1	SFLOW2
Units ----	CFS	CFS	CFS
Time Date			
0030 08APR1983	12005.000	12005.000	12005.000
0130 08APR1983	5638.000	M	5638.000
0230 08APR1983	4842.000	4842.000	4842.000
0330 08APR1983	3251.000	3251.000	3251.000
0430 08APR1983	4847.000	4847.000	4847.000
0530 08APR1983	3253.000	3253.000	3253.000
0630 08APR1983	4052.000	4052.000	4052.000
0730 08APR1983	5651.000	5651.000	5651.000
0830 08APR1983	6455.000	6455.000	6455.000
0930 08APR1983	4068.000	4068.000	4068.000
1030 08APR1983	4075.000	4075.000	4075.000
1130 08APR1983	5680.000	5680.000	5680.000
1230 08APR1983	4888.000	4888.000	4888.000
1330 08APR1983	3293.000	3293.000	3293.000
1430 08APR1983	4094.000	4094.000	4094.000
1530 08APR1983	4094.000	4094.000	4094.000
1630 08APR1983	3297.000	3297.000	3297.000
1730 08APR1983	3686.274	3686.274	3686.274
1830 08APR1983	3294.000	3294.000	3294.000
1930 08APR1983	99999.000	M	M
2030 08APR1983	3490.196	3490.196	3490.196
2130 08APR1983	3289.000	3289.000	3289.000
2230 08APR1983	2486.000	2486.000	2486.000
2330 08APR1983	3285.000	3285.000	3285.000
0030 09APR1983	2483.000	2483.000	2483.000
0130 09APR1983	2448.000	2448.000	2448.000
0230 09APR1983	2458.000	2458.000	2458.000
0330 09APR1983	3285.000	3285.000	3285.000
0430 09APR1983	5692.000	5692.000	5692.000
0530 09APR1983	5692.000	5692.000	5692.000
0630 09APR1983	6496.000	6496.000	6496.000
0730 09APR1983	4898.000	4898.000	4898.000
0830 09APR1983	5709.000	5709.000	5709.000
0930 09APR1983	6528.000	6528.000	6528.000
1030 09APR1983	1058.823	M	1058.823
1130 09APR1983	8156.000	8156.000	8156.000
1230 09APR1983	9771.000	9771.000	9771.000
1330 09APR1983	11390.000	11390.000	M
1430 09APR1983	8179.000	8179.000	8179.000
1530 09APR1983	11408.000	11408.000	11408.000
1630 09APR1983	11413.000	11413.000	11413.000
1730 09APR1983	12227.000	12227.000	12227.000
1830 09APR1983	15476.000	15476.000	15476.000
1930 09APR1983	16292.000	16292.000	16292.000
2030 09APR1983	13882.000	13882.000	13882.000
2130 09APR1983	15511.000	15511.000	15511.000
2230 09APR1983	16331.000	16331.000	16331.000
2330 09APR1983	13103.000	13103.000	13103.000
0030 10APR1983	12288.000	12288.000	12288.000
0130 10APR1983	13912.000	13912.000	13912.000
0230 10APR1983	13918.000	13918.000	13918.000
0330 10APR1983	12297.000	12297.000	12297.000
0430 10APR1983	11489.000	11489.000	11489.000
0530 10APR1983	12304.000	12304.000	12304.000
0630 10APR1983	10679.000	10679.000	10679.000
0730 10APR1983	9053.000	9053.000	9053.000
0830 10APR1983	9873.000	9873.000	9873.000
0930 10APR1983	8244.000	8244.000	8244.000
1030 10APR1983	12334.000	12334.000	12334.000
1130 10APR1983	4166.000	M	M
1230 10APR1983	5550.000	5550.000	5550.000
1330 10APR1983	6000.000	6000.000	6000.000
1430 10APR1983	8010.000	8010.000	8010.000



Name: **SDEV** **Compute the standard deviation of one independent variable.**

Use: CO SY=SDEV(TX)

This function is used to compute the standard deviation of one independent variable. TX can be either regular or irregular time series. SY is a scalar and is set to the computed standard deviation. This function requires a minimum of three valid values to compute.

Name: **SKEW** **Compute the skew coefficient of one independent variable**

Use: CO SY=SKEW(TX)

This function is used to compute the skew coefficient of one independent variable. TX can be either regular or irregular time series. SY is a scalar and is set to the computed skew coefficient. This function requires a minimum of three valid values to compute.

Example: The following example uses the SDEV and SKEW functions to determine the standard deviation and skew of a set of time series data. Note that the SDEV and SKEW function are the same, with the only difference being that each sets either the scalar variable to standard deviation or skew.

Macro:

```
MACRO SDSKEW
TIME 01MAY1983 0100 06MAY1983 2300
CL ALL
GE TX=TESTDB.DSS:/SCIOTO/BMRI2/FLOW/01MAY1983/1HOUR/OBS/
COMP SDEV=SDEV(TX)
COMP TSKEW=SKEW(TX)
SHOW SDEV TSKEW
ENDMACRO
```

Execution:

```
I>!R SDSKEW
  ALL VARIABLES HAVE BEEN CLEARED
    -----DSS---ZOPEN:  Existing File Opened,  File: TESTDB.DSS
                       Unit:  71;  DSS Version: 6-IA
 -----DSS--- ZREAD Unit  71; Vers.   24:
/SCIOTO/BMRI2/FLOW/01MAY1983/1HOUR/OBS/
  NUMBER OF VALID PAIRS        143
  NUMBER OF MISSING VALUES          0
  MEAN     143.799200
  STANDARD DEVIATION      48.882410
  SKEW COEFFICIENT   9.202288E-01
  NUMBER OF VALID PAIRS        143
  NUMBER OF MISSING VALUES          0
  MEAN     143.799200
  STANDARD DEVIATION      48.882410
  SKEW COEFFICIENT   9.202288E-01

 VARIABLE=SDEV
 SDEV    =      48.882410
 VARIABLE=TSKEW
 TSKEW   =   9.202288E-01
```

Name: **SELECT** **Extract time series data at unique time specification.**
Use: CO TS= SELECT(TX,LEVEL,RANGE,FLAG,TWINDOW,TFLAG)

This function is used to extract data from any regular or irregular time series, based on any number of unique time specifications. The extraction process can take place on any of five different selection LEVELs which can be mutually inclusive or exclusive as defined by the FLAG and RANGE parameters. Each LEVEL has a RANGE parameter that is either a specific value or a range of values. The extracted data is defaulted to be irregular time series, but the user has the option to redefine it as regular by the TFLAG parameter. CAUTION: Defining the extracted data to be regular will cause it to be stored in a consecutive manner at regular time interval specified on the E pathname part without regard to its actual irregular date and time.

Parameter	Description
TX	Input variable name of regular or irregular time series.
LEVEL	Level of time specific selection.
RANGE	Beginning and ending values of LEVEL. If only one value is specified, then the beginning and ending values are assumed to be the same. The following table shows all the valid LEVEL and RANGE values.

LEVEL	RANGE	Example
YEAR	Numeric: Four digit year value or a range of values. #### or ####-####	1938 or 1938-1945
MONTH	Alpha: First three characters of a month or a range of months. aaa or aaa-aaa	JAN or OCT-FEB
DAYMON	Numeric/Alpha: One or two digit day value or a range of values or the key word "LASTDAY" which specifies the last day of the month.	15 or 1-15 or LASTDAY or 15-LASTDAY
DAYWEE	Alpha: First three characters of a week day or a range of week days. aaa or aaa-aaa Note: Sunday is day 1 and Saturday is 7.	MON or SUN-SAT
TIME	Numeric: Four digit military style 24 hour clock consisting of a single time or a range of time. #### or ####-####	2300 or 0300-0600

Parameter	Description
FLAG	The valid entries for the processing FLAG are "INCLUDE" (default) or "EXCLUDE". This entry controls the inclusion or exclusion of all data specified by LEVEL and RANGE.
TWINDOW	This parameter is only used with the TIME component of the LEVEL parameter and is expressed in minutes before and after the time of day within which the data will be extracted. The window is assumed to be symmetric about the time specified in RANGE. At the time boundaries of 0000 and 2400, the time window will not cross into the previous or next day.

TFLAG The valid entries for the TFLAG are "IRREGULAR" (default) or "REGULAR". This parameter allows the user to control how DSSMATH treats the data extracted as it relates to its internal definition of the type of time series data it is. This allows the user to override the protection in the program to write irregular data as regular.

Examples : This first example is designed to demonstrate how the SELECT function works and is not a real working example. The second example given is a real working example acquired from NPD in which the SELECT function is used to select the drawdown elevations for the last day in April for the upper rule curve and for a reservoir simulation study.

Example 1:

Macro:

```
MACRO SELECT
CL ALL
TI 01JAN19 0100 31DEC24 2400
GET TY=RMS.DSS:/ALLEGHENY/P109.159/PRECIP-INC/01APR1919/IR-MONTH/HARRIS/
COMP TS=SELECT(TY,YEAR,1922-1922,INCLUDE,0,IRREGULAR)
PUT.A TS=F=SELECT YEAR
CLEAR TY
COMP TX=SELECT(TS,MONT,SEP-OCT,INCLUDE,0,IRREGULAR)
PUT.A TX=F=SELECT MONTH
COMP TS1=SELECT(TX,DAYM,20-LASTDAY,INCLUDE,0,IRREGULAR)
PUT.A TS1=F=SELECT DAYMONTH
COMP TX1=SELECT(TS1,DAYW,MON-SAT,INCLUDE,0,IRREGULAR)
PUT.A TX1=F=SELECT DAYWEEK
COMP TS2=SELECT(TX1,TIME,2400,INCLUDE,10,IRREGULAR)
PUT.A TS2=F=SELECT TIME
$CONTINUE INCASE OF ERROR
ENDMACRO
```

Execution:

```
I>!R SELECT
  ALL VARIABLES HAVE BEEN CLEARED
    -----DSS---ZOPEN:  Existing File Opened,  File: RMS.DSS
                       Unit:  71;  DSS Version: 6-GM
 -----DSS---ZWRITE Unit  71; Vers.   11:
/ALLEGHENY/P109.159/PRECIP-INC/01MAY1922/IR-MONTH/SELECT YEAR/
   TY CLEARED
 -----DSS---ZWRITE Unit  71; Vers.   11:
/ALLEGHENY/P109.159/PRECIP-INC/01SEP1922/IR-MONTH/SELECT MONTH/
```

Variables ----	TS	TX	TS1	TX1	TS2
Units ----	INCHES	INCHES	INCHES	INCHES	INCHES
Time Date					
0400 30JUL1922	.529	-	-	-	-
1600 30JUL1922	.529	-	-	-	-
2000 30JUL1922	.529	-	-	-	-
2400 30JUL1922	.529	-	-	-	-
0400 01AUG1922	.529	-	-	-	-
1200 01AUG1922	.529	-	-	-	-
0400 22AUG1922	.529	-	-	-	-
2000 25AUG1922	.529	-	-	-	-
2400 26AUG1922	.529	-	-	-	-
0400 28AUG1922	.529	-	-	-	-
1600 28AUG1922	.529	-	-	-	-
2000 28AUG1922	.529	-	-	-	-
2400 28AUG1922	.529	-	-	-	-
0400 29AUG1922	.529	-	-	-	-
1600 29AUG1922	.529	-	-	-	-
2000 29AUG1922	.529	-	-	-	-
2400 29AUG1922	.529	-	-	-	-
0400 31AUG1922	.529	-	-	-	-
1200 31AUG1922	.529	-	-	-	-
0400 21SEP1922	.483	.483	.483	.483	-
2000 24SEP1922	.455	.455	.455	-	-
2400 25SEP1922	.455	.455	.455	.455	.455
0400 27SEP1922	.455	.455	.455	.455	-
1600 27SEP1922	.455	.455	.455	.455	-
2000 27SEP1922	.455	.455	.455	.455	-
2400 27SEP1922	.455	.455	.455	.455	.455
0400 28SEP1922	.455	.455	.455	.455	-
1600 28SEP1922	.455	.455	.455	.455	-
2000 28SEP1922	.455	.455	.455	.455	-
2400 28SEP1922	.455	.455	.455	.455	.455
0400 30SEP1922	.455	.455	.455	.455	-
1200 30SEP1922	.455	.455	.455	.455	-
0400 26OCT1922	.356	.356	.356	.356	-
2000 29OCT1922	.356	.356	.356	-	-
2400 30OCT1922	.356	.356	.356	.356	.356
0400 01NOV1922	.356	-	-	-	-
1600 01NOV1922	.356	-	-	-	-
2000 01NOV1922	.356	-	-	-	-
2400 01NOV1922	.356	-	-	-	-
0400 02NOV1922	.356	-	-	-	-
1600 02NOV1922	.356	-	-	-	-
2000 02NOV1922	.356	-	-	-	-
2400 02NOV1922	.356	-	-	-	-
0400 04NOV1922	.356	-	-	-	-
1200 04NOV1922	.356	-	-	-	-
0400 05DEC1922	.275	-	-	-	-
2000 08DEC1922	.000	-	-	-	-
2400 09DEC1922	.152	-	-	-	-
0400 11DEC1922	.152	-	-	-	-

Example 2:

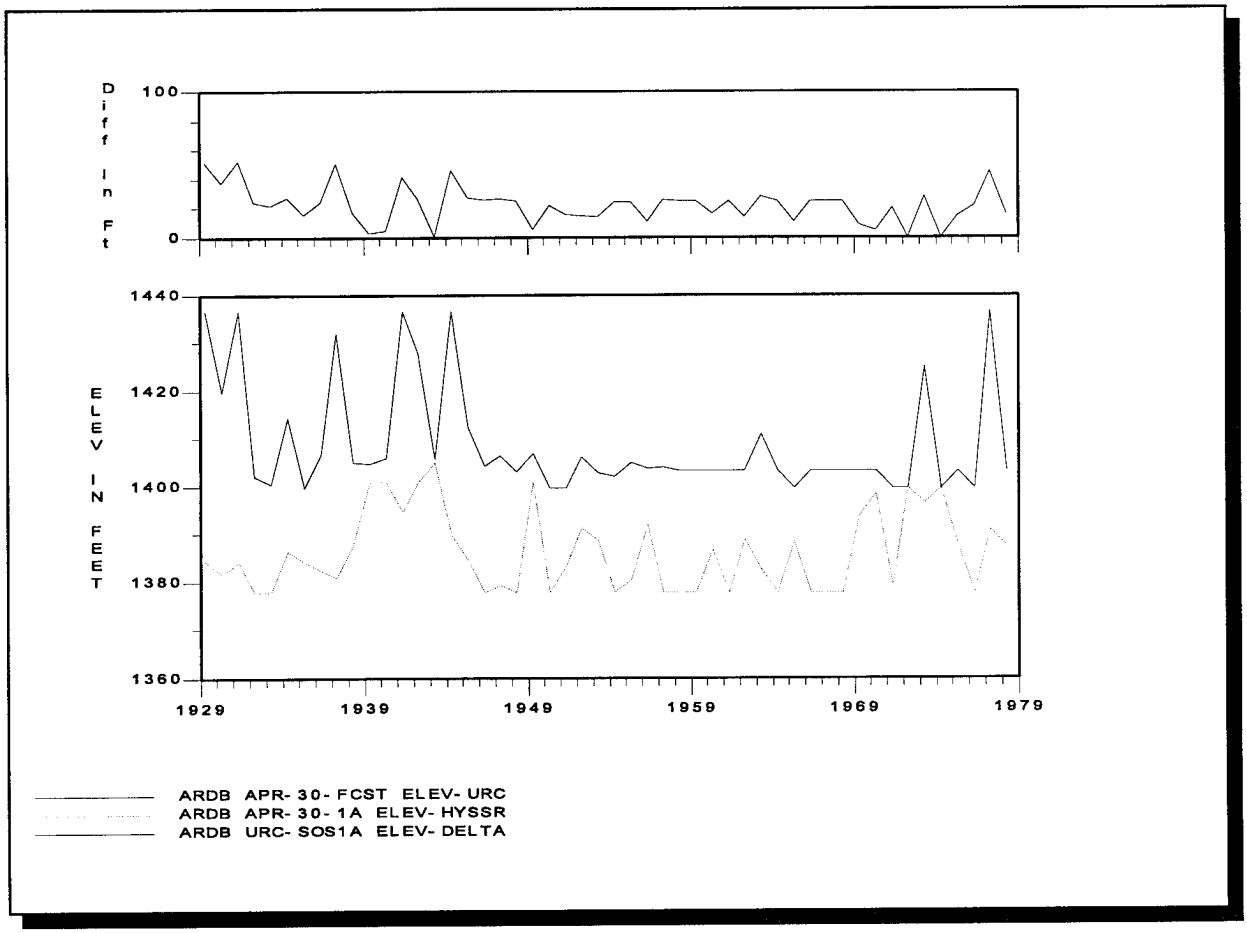
Macro:

```
MACRO TSELURC
** Using SELECT to compare URC to SOR's 30 apr drawdown pts.
CL ALL
TI 01JAN1929 2400 30SEP1978 2400
GET URCF=FIXED.DSS://ARDB/ELEV-URC//1DAY/FCST/
CO URCFA=SELECT(URCF,MONTH,APR,INCLUDE,0)
CO URCFAE=SELECT(URCFA,DAYMONTH,LAST,INCLUDE,0)
CO URC=URCFAE
SD URC U=FEET T=INST-VAL
PUT.A URC=COMPSOR.DSS://ARDB/ELEV-URC//IR-DECADE/APR-30-FCST/
CL URCF URCFA URCFAE
GET S1AHSR=1A.DSS://ARDB/ELEV-HYSSR//IR-DECADE/1A/
CO S1AHS=SELECT(S1AHSR,MONTH,APR,INCLUDE,0)
CO S1AH=SELECT(S1AHS,DAYMONTH,LAST,INCLUDE,0)
CO S1A=S1AH
SD S1A U=FEET T=INST-VAL
PUT.A S1A=COMPSOR.DSS://ARDB/ELEV-HYSSR//IR-DECADE/APR-30-1A/
CL S1AHSR S1AHS S1AH
CO DIFF=URC-S1A
SD DIFF U=FEET T=INST-VAL
PUT.A DIFF=COMPSOR.DSS://ARDB/ELEV-DELTA//IR-DECADE/URC-SOS1A/
TAB.F URC S1A DIFF
$CONTINUE
ENDMACRO
```

Execution:

```
I>!R TSELURC
      -----DSS---ZOPEN:  Existing File Opened,  File: FIXED.DSS
                         Unit:  71;  DSS Version: 6-IC
 -----DSS--- ZREAD Unit  71; Vers.    1:  //ARDB/ELEV-URC/01JAN1929/1DAY/FCST/
 -----DSS---ZOPEN:  Existing File Opened,  File: COMPSOR.DSS
                         Unit:  72;  DSS Version: 6-IC
 -----DSS---ZWRITE Unit  72; Vers.    4:  //ARDB/ELEV-URC/01JAN1920/IR-DECADE/APR-30-FCST/
 URCF CLEARED
 URCFA CLEARED
 URCFAE CLEARED
  -----DSS---ZOPEN:  Existing File Opened,  File: 1A.DSS
                         Unit:  73;  DSS Version: 6-IC
 -----DSS---ZWRITE Unit  72; Vers.    4:  //ARDB/ELEV-HYSSR/01JAN1920/IR-DECADE/APR-30-1A/

 S1AHSR CLEARED
 S1AHS CLEARED
 S1AH CLEARED
 -----DSS---ZWRITE Unit  72; Vers.    4:  //ARDB/ELEV-DELTA/01JAN1920/IR-DECADE/URC-SOS1A/
```

Variables ----	URC	S1A	DIFF
Units ----	FEET	FEET	FEET
Time Date			
2400 30APR1929	1436.700	1384.400	52.300
2400 30APR1930	1419.800	1381.900	37.900
2400 30APR1931	1436.700	1384.200	52.500
2400 30APR1932	1402.200	1377.900	24.300
2400 30APR1933	1400.500	1377.900	22.600
2400 30APR1934	1414.500	1386.700	27.800
2400 30APR1935	1399.900	1384.100	15.800
2400 30APR1936	1406.900	1382.600	24.300
2400 30APR1937	1432.100	1381.200	50.900
2400 30APR1938	1405.100	1387.900	17.200
2400 30APR1939	1404.900	1401.100	3.800
2400 30APR1940	1406.200	1401.100	5.100
2400 30APR1941	1436.700	1394.700	42.000
2400 30APR1942	1428.000	1401.100	26.900
2400 30APR1943	1406.000	1405.200	0.800
2400 30APR1944	1436.700	1389.900	46.800
2400 30APR1945	1412.600	1384.600	28.000
2400 30APR1946	1404.400	1377.900	26.500
2400 30APR1947	1406.700	1379.400	27.300
2400 30APR1949	1407.000	1401.100	5.900
2400 30APR1950	1399.900	1377.900	22.000
2400 30APR1951	1399.900	1383.500	16.400
2400 30APR1952	1406.400	1391.500	14.900
2400 30APR1953	1403.100	1388.900	14.200
2400 30APR1954	1402.400	1377.900	24.500
2400 30APR1955	1405.100	1380.500	24.600
2400 30APR1956	1403.900	1392.200	11.700
2400 30APR1957	1404.100	1377.900	26.200
2400 30APR1958	1403.400	1377.900	25.500
2400 30APR1959	1403.400	1377.900	25.500
2400 30APR1960	1403.400	1386.900	16.500
2400 30APR1961	1403.400	1377.900	25.500
2400 30APR1962	1403.400	1389.000	14.400
2400 30APR1963	1411.200	1382.800	28.400
2400 30APR1964	1403.400	1377.900	25.500
2400 30APR1965	1399.900	1388.500	11.400
2400 30APR1966	1403.400	1377.900	25.500
2400 30APR1967	1403.400	1377.900	25.500
2400 30APR1968	1403.400	1377.900	25.500
2400 30APR1969	1403.400	1394.100	9.300
2400 30APR1970	1403.400	1398.700	4.700
2400 30APR1971	1399.900	1379.500	20.400
2400 30APR1972	1399.900	1399.800	0.100
2400 30APR1973	1425.300	1396.700	28.600
2400 30APR1974	1399.900	1399.800	0.100
2400 30APR1975	1403.400	1387.800	15.600
2400 30APR1976	1399.900	1377.900	22.000
2400 30APR1977	1436.700	1391.200	45.500
2400 30APR1978	1403.400	1387.700	15.700
2400 30APR1948	1403.300	1377.900	25.400



Name: **SHIFT** **Shift adjustment**
Use: CO TY=SHIFT(TS,TX)

Generate a time series of shift adjustments TY with times at TX and actual shifts TS, possibly at other times. TX and TY cannot be the same variables. TS values are interpolated at TX times. The interpolation is linear between TS values, unless TX time is greater than last TS time. Then, the last value of TS is held constant for the remaining TX times. If no previous TS value brackets TY, then TY is set to zero. An IF condition has no effect.

Example: The following example uses the SHIFT and PERCON functions to demonstrate how they can be used in conjunction with an irregular time series variable of gage shift values to create a regular time series of shift adjustments. Note that the SHIFT function interpolates while the PERCON function holds them constant for the period.

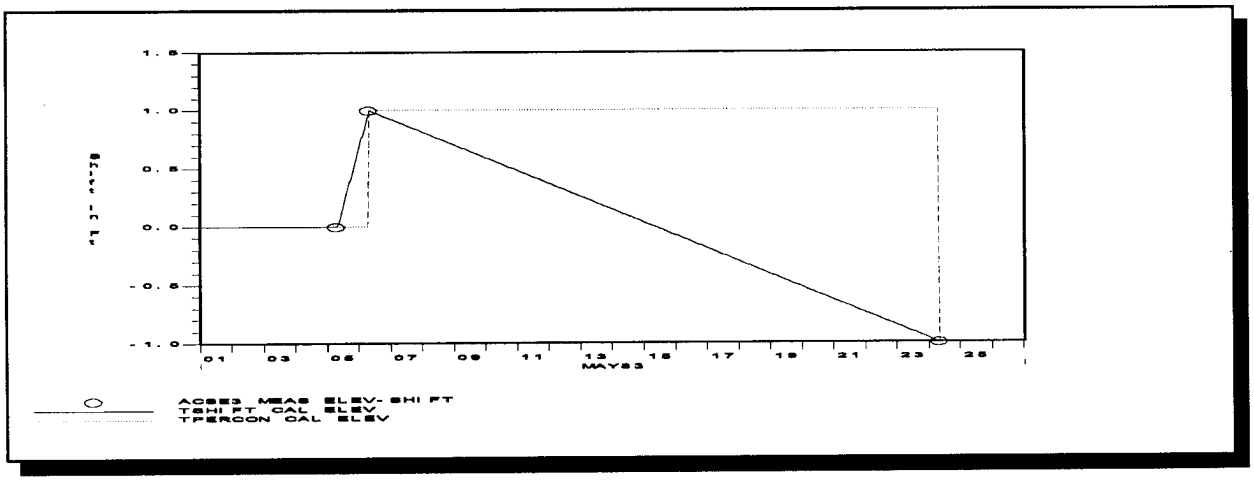
Macro:

```
MACRO TSHIFT
CL ALL
TI 0100 01MAY83 2400 26MAY83
GET ELEV=testdb.dss:/SCIOTO/ACSE3/ELEV/01MAY1983/1HOUR/OBS/
GET  SHFT=testdb.dss:/SCIOTO/ACSE3/ELEV-SHIFT/01JAN1983/IR-YEAR/MEAS/
GET ELEV=testdb.dss:/SCIOTO/ACSE3/ELEV/01MAY1983/1HOUR/OBS/
COM TES=SHIFT(SHFT,ELEV)
COM TEP=PERCON(SHFT,ELEV)
PUT.A TES=B=TSHIFT,F=CAL
PUT.A TEP=B=TPERCON,F=CAL
$CONTINUE INCASE OF ERROR
ENDMACRO
```

Execution:

```
I>!R TSHIFT
ALL VARIABLES HAVE BEEN CLEARED
    -----DSS---ZOPEN:  Existing File Opened,  File: TESTDB.DSS
                       Unit:  71;  DSS Version: 6-IA
 -----DSS--- ZREAD Unit  71; Vers.   22:  /SCIOTO/ACSE3/ELEV/01MAY1983/1HOUR/OBS/
 ELEV   CLEARED
 -----DSS--- ZREAD Unit  71; Vers.   22:  /SCIOTO/ACSE3/ELEV/01MAY1983/1HOUR/OBS/
 -----DSS---ZWRITE Unit  71; Vers.    1:  /SCIOTO/TSHIFT/ELEV/01MAY1983/1HOUR/CAL/
 -----DSS---ZWRITE Unit  71; Vers.    1:  /SCIOTO/TPERCON/ELEV/01MAY1983/1HOUR/CAL/
```

Variables ----	SHFT	ELEV	TES	TEP
Units ----	FEET	FT-MSL	FEET	UNK
Time Date				
0100 01MAY1983	-	888.900	.000	M
0200 01MAY1983	-	889.000	.000	M
0300 01MAY1983	-	M	.000	M
0400 01MAY1983	-	M	.000	M
0500 01MAY1983	-	M	.000	M
0600 01MAY1983	-	M	.000	M
0700 01MAY1983	-	889.430	.000	M
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
0600 05MAY1983	-	892.770	.000	M
0700 05MAY1983	-	892.770	.000	M
0800 05MAY1983	.000	892.730	.000	.000
0900 05MAY1983	-	892.660	.042	.000
1000 05MAY1983	-	892.520	.083	.000
1100 05MAY1983	-	892.540	.125	.000
1200 05MAY1983	-	892.520	.167	.000
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
0500 06MAY1983	-	892.150	.875	.000
0600 06MAY1983	-	892.160	.917	.000
0700 06MAY1983	-	892.100	.958	.000
0800 06MAY1983	1.000	892.060	1.000	1.000
0900 06MAY1983	-	892.050	.995	1.000
1000 06MAY1983	-	892.040	.991	1.000
1100 06MAY1983	-	891.980	.986	1.000
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
0500 24MAY1983	-	888.290	-.986	1.000
0600 24MAY1983	-	888.280	-.991	1.000
0700 24MAY1983	-	888.280	-.995	1.000
0800 24MAY1983	-1.000	888.280	-1.000	-1.000
0900 24MAY1983	-	888.270	-1.000	-1.000
1000 24MAY1983	-	888.270	-1.000	-1.000
1100 24MAY1983	-	888.270	-1.000	-1.000
1200 24MAY1983	-	888.260	-1.000	-1.000
1300 24MAY1983	-	888.260	-1.000	-1.000
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
2300 26MAY1983	-	888.170	-1.000	-1.000
2400 26MAY1983	-	888.170	-1.000	-1.000



Name: **SQRT** **Square root function**
Use: CO TY=SQRT(TX)

Compute the square root of TX and store in TY. TX and TY can be the same variable. TX can be scalar, time series or paired data. If a value of TX is undefined, resulting values of TY are undefined. If a concurrent IF condition is not satisfied, TY is unchanged. If a value in TX is negative, the value in TY will be set to missing.

Name: **SS** **Straddle Stagger routing function**
Use: CO TY=SS(TX,NAVG,LAG,NR)

Route the uniform time series variable TX by the Straddle Stagger hydrologic routing method and store it in variable TY. Note: Variables TY and TX cannot be the same variable and variable TX is permanently modified by being lagged by the SS function. Variable NAVG is the number of ordinates to average. Variable LAG is the number of ordinates to lag hydrograph. Variable NR is the number of subreaches to use. The "IF" compute option has no effect on this function.

Name: **SSW** **Wilmington District Straddle Stagger routing function**
Use: CO TY=SSW(TX,NAVG,LAG,NR)

Route the uniform time series variable TX by the Wilmington District Straddle Stagger hydrologic routing method and store it in variable TY. Note: Variables TY and TX cannot be the same variable. This function is similar to function SS except that LAG is usually zero and the result of averaging NAVG values is stored in the NAVGed value. Variable NAVG is the number of ordinates to average. Variable LAG is the number of ordinates to lag hydrograph. Variable NR is the number of subreaches to use. For example, if NAVG is 7, the results for ordinates 13 through 19 are stored in ordinate 19. For the same example, if LAG is set to 2, the result is stored in ordinate 21. The "IF" compute option has no effect on this function.

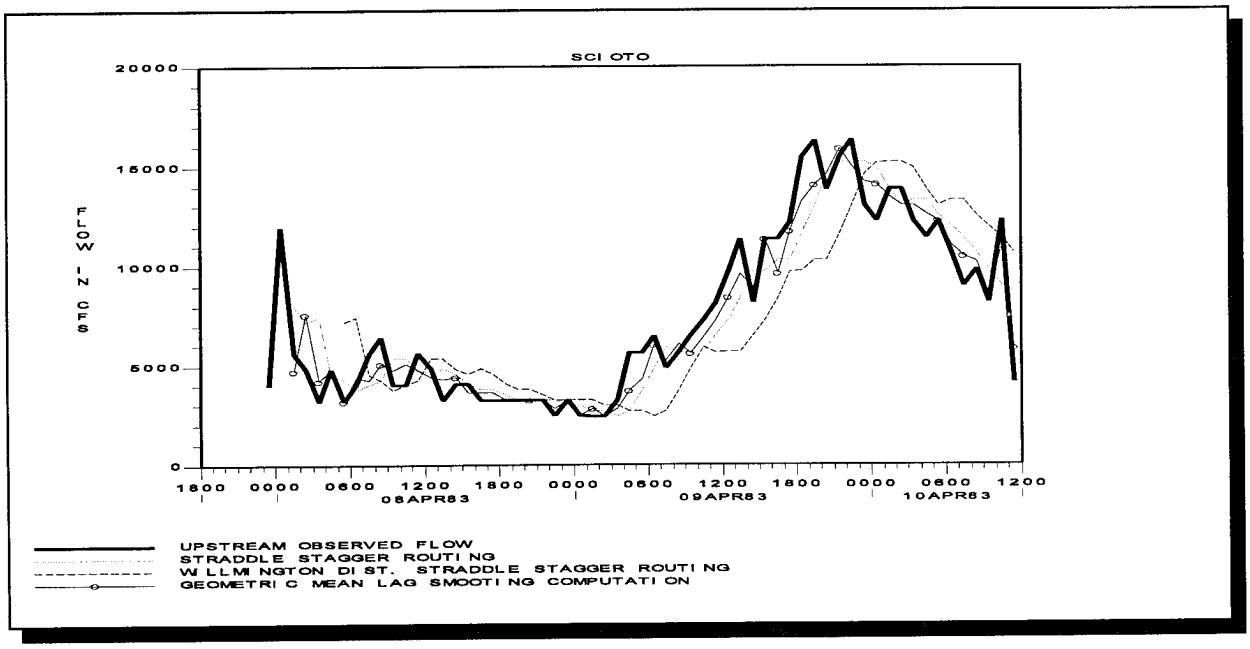
Example: The following example uses the TSHIFT and SQRT functions to compute the lagged geometric mean of an observed flow. The same observed flow is then routed using the Straddle Stagger functions SS and SSW. Note: These examples are not real life applications and only serve to demonstrate the use and results of the functions.

Macro:

```
MACRO TSS
CL ALL
TIME 07APR1983 0100 10APR1983 1200
GET FLOW=testdb.dss:/SCIOTO/ACSE3/FLOW/01APR1983/1HOUR/OBS/
COMP TX=TSHIFT(FLOW,2H)
COMP TX=FLOW*TX
COMP TX=SQRT(TX)
PUT.A TX=B=TXSS,F=CAL
** COMPUTE A STRADDLE STAGGER ROUTING
COMP RFLOW=SS(FLOW,3,2,1)
PUT.A RFLOW=B=TSS,F=CAL
** COMPUTE A WILMINGTON DIST. STRADDLE STAGGER ROUTING
COMP RWFLOW=SSW(FLOW,3,2,1)
PUT.A RWFLOW=B=TWSS,F=CAL
$CONTINUE INCASE OF ERROR
ENDMACRO
```

Execution:

```
I>!R TSS
ALL VARIABLES HAVE BEEN CLEARED
   -----DSS---ZOPEN:  Existing File Opened,  File: TESTDB.DSS
                      Unit:  71;  DSS Version: 6-IA
-----DSS--- ZREAD Unit  71; Vers.    3:  /SCIOTO/ACSE3/FLOW/01APR1983/1HOUR/OBS/
WARNING STARTING TIMES ARE NOT THE SAME
WARNING -- EXISTING VARIABLE RE-DEFINED
WARNING -- EXISTING VARIABLE RE-DEFINED
-----DSS---ZWRITE Unit  71; Vers.    2:  /SCIOTO/TXSS/FLOW/01APR1983/1HOUR/CAL/
Warning - Variable:  FLOW        Has been lagged
-----DSS---ZWRITE Unit  71; Vers.    7:  /SCIOTO/TSS/FLOW/01APR1983/1HOUR/CAL/
Warning - Variable:  FLOW        Has been lagged
-----DSS---ZWRITE Unit  71; Vers.    7:  /SCIOTO/TWSS/FLOW/01APR1983/1HOUR/CAL/
```



Name: **TS1** **Interpolate data at regular intervals**
Use: CO TY=TS1(TX,DT,TOFF)

Derive a new, regular-interval time series TY from TX. TY will have a time span defined by the current time window, a time interval of DT, and a time interval offset of TOFF. TOFF is time from the beginning of the standard interval to the actual time of the data. DT and TOFF are expressed as units of time as nnT, where T is M for minutes, H for hours, and D for days. TX may be a regular or irregular time series. Units and type are preserved. An IF condition has no effect.

Name: **TS2** **Period averages at regular intervals**
Use: CO TY=TS2(TX,DT,TOFF)

Derive a new, regular-interval time-series TY from the instantaneous values in TX with a time span defined by the current time window, a time interval of DT, and an interval offset of TOFF. DT and TOFF are expressed as units of time as nnT, where T is M for minutes, H for hours, and D for days. TOFF is time from the beginning of the standard interval to the actual time of the data. TX may be an irregular or regular time-series, but must be of type INST-VAL. TY will be typed PER-AVER. Example of use: deriving daily average flow from hourly observed values. An IF condition has no effect.

Name: **TS3** **Period mins or maxs at regular intervals**
Use: CO TY=TS3(TX,DT,TOFF,EXT)

Finds either the minima or maxima in TX at DT intervals, with interval offsets at TOFF, and puts the result in TY. EXT is used to indicate the extreme of interest: 'MIN' or 'MAX'. DT and TOFF are expressed as units of time as nnT, where T is M for minutes, H for hours, and D for days. TOFF is time from the beginning of the standard interval to the actual time of the data. TY is typed PER-EXTR. Example of use: finding daily minima and maxima from hourly instantaneous observations. An IF condition has no effect.

Example: The following examples use the TS1, TS2, and TS3 functions to derive a new 6-hour regular interval from a 1-hour regular time interval based on linear interpolation, period average, and maximum and minimum values. Note: These examples are not real life applications and only serve to demonstrate the use and results of the functions.

Macro:

```
MACRO TTS1-3
CL ALL
TIME 09APR1983 1200 10APR1983 1200
GET FLOW=testdb.dss:/SCIOTO/ACSE3/FLOW/01APR1983/1HOUR/OBS/
COMP TS1=TS1(FLOW,6H,0M)
COMP TS2=TS2(FLOW,6H,0M)
COMP TS3MX=TS3(FLOW,6H,0M,MAX)
COMP TS3MN=TS3(FLOW,6H,0M,MIN)
TAB.F FLOW TS1 TS2 TS3MX TS3MN
$CONTINUE INCASE OF ERROR
ENDMACRO
```

Execution:

```
I>!R TTS1-3
  ALL VARIABLES HAVE BEEN CLEARED
    -----DSS---ZOPEN:  Existing File Opened,  File: TESTDB.DSS
                       Unit:  71;  DSS Version: 6-IA
 -----DSS--- ZREAD Unit  71; Vers.    3:  /SCIOTO/ACSE3/FLOW/01APR1983/1HOUR/OBS/
```

Variables ----	FLOW	TS1	TS2	TS3MX	TS3MN
Units ----	CFS	CFS	CFS	CFS	CFS
Time Date					
1130 09APR1983	8146.000	-	-	-	-
1200 09APR1983	-	8953.501	M	M	M
1230 09APR1983	9761.000	-	-	-	-
1330 09APR1983	11380.000	-	-	-	-
1430 09APR1983	8169.000	-	-	-	-
1530 09APR1983	11398.000	-	-	-	-
1630 09APR1983	11403.000	-	-	-	-
1730 09APR1983	12217.000	-	-	-	-
1800 09APR1983	-	13841.500	10755.380	12217.000	8169.000
1830 09APR1983	15466.000	-	-	-	-
1930 09APR1983	16282.000	-	-	-	-
2030 09APR1983	13872.000	-	-	-	-
2130 09APR1983	15501.000	-	-	-	-
2230 09APR1983	16321.000	-	-	-	-
2330 09APR1983	13093.000	-	-	-	-
2400 09APR1983	-	12685.500	15004.500	16321.000	13093.000
0030 10APR1983	12278.000	-	-	-	-
0130 10APR1983	13902.000	-	-	-	-
0230 10APR1983	13908.000	-	-	-	-
0330 10APR1983	12287.000	-	-	-	-
0430 10APR1983	11479.000	-	-	-	-
0530 10APR1983	12294.000	-	-	-	-
0600 10APR1983	-	11481.500	12674.460	13908.000	11479.000
0630 10APR1983	10669.000	-	-	-	-
0730 10APR1983	9043.000	-	-	-	-
0830 10APR1983	9863.000	-	-	-	-
0930 10APR1983	8234.000	-	-	-	-
1030 10APR1983	12324.000	-	-	-	-
1130 10APR1983	4156.000	-	-	-	-
1200 10APR1983	-	M	M	M	M

Name: **TS4** **Interpolate data at irregular intervals**
Use: CO TY=TS4(TX,TZ)

Derive a new, irregular-interval time series TY from TX at the times for TZ (ie., TZ provides the time pattern). Units and type are preserved. An IF condition has no effect.

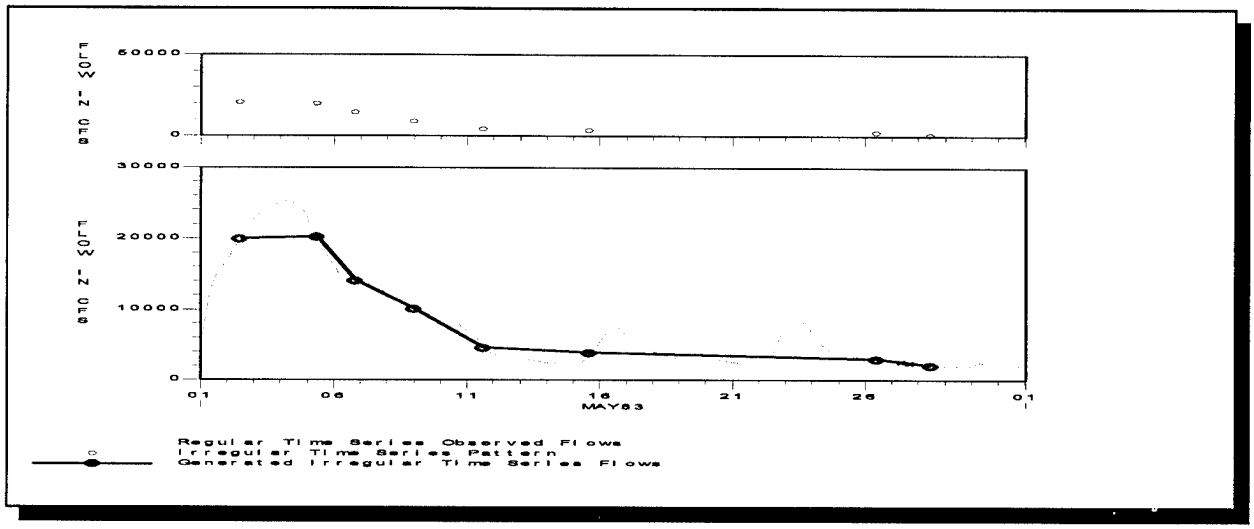
Example: The following example uses the TS4 function to generate an irregular time series data set from a regular interval data set based on a pattern irregular time series data set. Note: This example is not a real application and only serves to demonstrate the use and results of the function.

Macro:

```
MACRO TTS4
CL ALL
TI 01MAY1983 0100 31MAY83 2400
GET TX=testdb.dss:/SCIOTO/CISG3/FLOW/01MAY1983/1HOUR/OBS/
GET TZ=testdb.dss:/SCIOTO/CISG3/FLOW/01JAN1983/IR-YEAR/MEAS/
COM TS4=TS4(TX,TZ)
PUT.A TS4=B=TTS4,F=CAL
$CONTINUE INCASE OF ERROR
ENDMACRO
```

Execution:

```
I>!R TTS4
  ALL VARIABLES HAVE BEEN CLEARED
    -----DSS---ZOPEN:  Existing File Opened,  File: TESTDB.DSS
                       Unit:  71;  DSS Version: 6-IA
 -----DSS--- ZREAD Unit  71; Vers.   22:  /SCIOTO/CISG3/FLOW/01MAY1983/1HOUR/OBS/
 -----DSS---ZWRITE Unit  71; Vers.    2:  /SCIOTO/TTS4/FLOW/01JAN1983/IR-YEAR/CAL/
```



Name: **TSCYCL** **Time series cyclic analysis**
Use: CO XX=TSCYCL(TX,TREAT,STATFILE,DSSFILE)

Derive a set of statistics from cyclic regular time series TX. TX must be regular data at a 1HOUR, 1DAY, or 1MONTH interval. The time series TX may be, for example, 30 years of 1DAY data. For each interval, (e.g., 1st day of the year, 2nd day of the year, ... , 365th day of the year), statistics would be determined. The 14 statistics determined for each interval are: maximum, minimum, average, standard deviation, 5%, 10%, 25%, 50% (median), 75%, 90%, 95% percentiles, date of maximum, date of minimum, and number of values processed. These results are written to a specified DSS file, and two text output files. The variable XX is ignored.

TREAT controls the treatment of newly generated statistic values when missing data occur in the interval. It must be expressed in either the form nn# or nn%. If nn# is used nn gives the number of missing data values in the interval that will be accepted in the computation of the new value. If nn% is used nn gives the percent of missing data values in the interval that will be accepted in the computation of the new value. If the criteria is met a data value will be generated, if not a missing value will occur.

STATFILE is the base name, not more than 6 characters, of two files that will be written containing the same data written to the DSS file. The first file will contain all of the statistics generated. It will be in the form of a wide table. Its name will be 'statfile.sts' or 'statfile.t' where 'statfile' is the portion of the name the user specifies. This file is wide, so it is hard to view on some systems. The second file is a subset of the first named 'statfile.sum', or 'statfile.s'. The second file is limited to 80 columns wide for easy viewing. The second file contains: maximums with their dates, minimums with their dates, average, and the 50% percentile(median) values.

DSSFILE is the DSS file into which the cyclic results will be written. The results will be, for example, a record of all the maximums for each day of the year over the 30 year period. Fourteen records will be added to the DSS file specified, each containing one of the derived statistics. The records will use the pseudo year 3000 for storing the statistics.

Example: The following example uses the TSCYCL function to compute flow statistics of daily regular interval data at a reservoir gage. The amount of statistics generated is quite large and only a select sample of output is presented. Both statistics files "--.sts" and "--.sum" are presented with a couple of statistics plotted. This example was executed on the CD-4300 workstation because of the large amount of data being processed.

Macro:

```
MACRO BB
!RUN B1    LEHIGH  WALTER FLOW  1927  1975
$CONTINUE INCASE OF ERROR
ENDMACRO
MACRO B1 $BASIN $LOC $PARM $BYR $EYR
CL ALL
TIME 01JAN$BYR 2400 31DEC$EYR 2400
GE TX=lhprdss.dss:/$BASIN/$LOC/$PARM/01JAN1990/1DAY/OBS/
COM SZ = TSCYCL ( TX,10%, $LOC, $LOC )
$CONTINUE INCASE OF ERROR
ENDMACRO
```

Execution:

```
I>!R BB
  ALL VARIABLES HAVE BEEN CLEARED
    -----DSS---ZOPEN:  Existing File Opened,  File: LHPRDSS.DSS
                       Unit:  71;  DSS Version: 6-GJ
 -----DSS--- ZREAD Unit  71; Vers.    1:  /LEHIGH/WALTER/FLOW/01JAN1927/1DAY/OBS/
 -----DSS--- ZREAD Unit  71; Vers.    1:  /LEHIGH/WALTER/FLOW/01JAN1928/1DAY/OBS/
 -----DSS--- ZREAD Unit  71; Vers.    1:  /LEHIGH/WALTER/FLOW/01JAN1929/1DAY/OBS/
 -----DSS--- ZREAD Unit  71; Vers.    1:  /LEHIGH/WALTER/FLOW/01JAN1930/1DAY/OBS/
 -----DSS--- ZREAD Unit  71; Vers.    1:  /LEHIGH/WALTER/FLOW/01JAN1931/1DAY/OBS/
 -----DSS--- ZREAD Unit  71; Vers.    1:  /LEHIGH/WALTER/FLOW/01JAN1932/1DAY/OBS/

 -----DSS--- ZREAD Unit  71; Vers.    1:  /LEHIGH/WALTER/FLOW/01JAN1975/1DAY/OBS/
  ------ Begin CYCLIC analysis ------
           Interval: 1DAY
         Begin date: 01JAN1927
     Number of data:        3180
  Missing permitted:           0
     Table filename: walter.sts
           Pathname: /LEHIGH/WALTER/FLOW/01JAN1990/1DAY/OBS/
    -----DSS---ZOPEN:  Existing File Opened,  File: WALTER.DSS
                       Unit:  72;  DSS Version: 6-IB
 -----DSS---ZWRITE Unit  72; Vers.    2:
/LEHIGH/WALTER/COUNT-FLOW/01JAN3000/1DAY/OBS[01JAN1927-31DEC1975]/
 -----DSS---ZWRITE Unit  72; Vers.    2:
/LEHIGH/WALTER/FLOW-AVER/01JAN3000/1DAY/OBS[01JAN1927-31DEC1975]/
 -----DSS---ZWRITE Unit  72; Vers.    2:
/LEHIGH/WALTER/FLOW-MAX/01JAN3000/1DAY/OBS[01JAN1927-31DEC1975]/
 -----DSS---ZWRITE Unit  72; Vers.    2:
/LEHIGH/WALTER/FLOW-MIN/01JAN3000/1DAY/OBS[01JAN1927-31DEC1975]/
 -----DSS---ZWRITE Unit  72; Vers.    2:
/LEHIGH/WALTER/FLOW-SD/01JAN3000/1DAY/OBS[01JAN1927-31DEC1975]/
 -----DSS---ZWRITE Unit  72; Vers.    2:
/LEHIGH/WALTER/FLOW-P05/01JAN3000/1DAY/OBS[01JAN1927-31DEC1975]/
 -----DSS---ZWRITE Unit  72; Vers.    2:
/LEHIGH/WALTER/FLOW-P10/01JAN3000/1DAY/OBS[01JAN1927-31DEC1975]/
 -----DSS---ZWRITE Unit  72; Vers.    2:
/LEHIGH/WALTER/FLOW-P25/01JAN3000/1DAY/OBS[01JAN1927-31DEC1975]/
 -----DSS---ZWRITE Unit  72; Vers.    2:
/LEHIGH/WALTER/FLOW-P50/01JAN3000/1DAY/OBS[01JAN1927-31DEC1975]/
 -----DSS---ZWRITE Unit  72; Vers.    2:
/LEHIGH/WALTER/FLOW-P75/01JAN3000/1DAY/OBS[01JAN1927-31DEC1975]/
 -----DSS---ZWRITE Unit  72; Vers.    2:
/LEHIGH/WALTER/FLOW-P90/01JAN3000/1DAY/OBS[01JAN1927-31DEC1975]/
 -----DSS---ZWRITE Unit  72; Vers.    2:
/LEHIGH/WALTER/FLOW-P95/01JAN3000/1DAY/OBS[01JAN1927-31DEC1975]/
 -----DSS---ZWRITE Unit  72; Vers.    2:
/LEHIGH/WALTER/DATE-FLOW-MAX/01JAN3000/1DAY/OBS[01JAN1927-31DEC1975]/
 -----DSS---ZWRITE Unit  72; Vers.    2:
/LEHIGH/WALTER/DATE-FLOW-MIN/01JAN3000/1DAY/OBS[01JAN1927-31DEC1975]/
```

**Statistics File "WALTER.STS"

```
* Table of Statistics ***
*** Pathname> /LEHIGH/WALTER/FLOW/01JAN1990/1DAY/OBS/
*** Time window> 01JAN1927 - 31DEC1975
*** Number of bins>   365
```

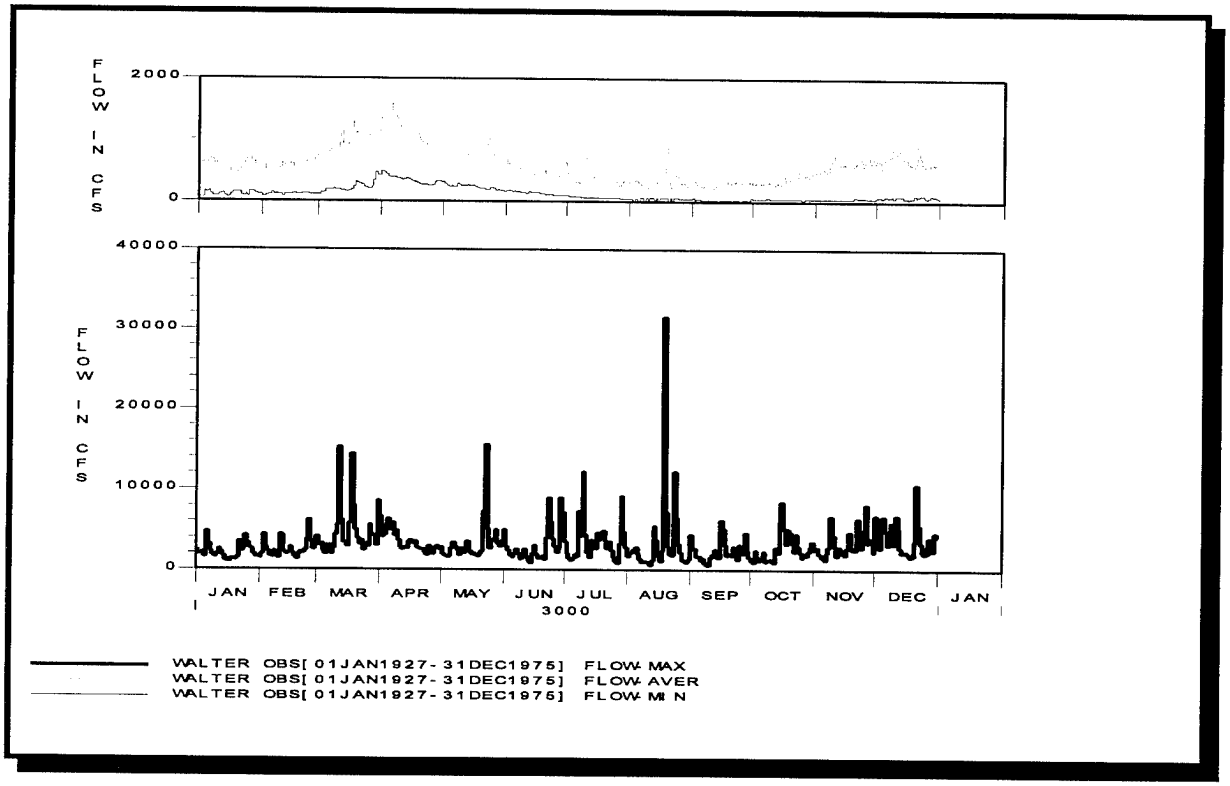
Bin	Co	Sum	Aver	Xmax	Xmin	SD	P05	P10	P25	P50	P75	P90	P95	Dmax	Dmin	Dmax	Dmin	Qflag	Use	Rej
1	48	31005	645.937	2557	69	578.305	90	183	331	441	780	1250	2471	17898	11323	01JAN1949	01JAN1931	0	48	1
2	48	29907	623.062	1966	65	434.971	113	183	317	473	730	1211	1746	26665	11324	02JAN1973	02JAN1931	0	48	1
3	48	30326	631.792	1970	78	443.625	159	201	296	470	754	1433	1539	21917	11325	03JAN1960	03JAN1931	0	48	1
4	48	28673	597.354	2250	159	406.050	169	229	320	450	784	1155	1263	21918	11326	04JAN1960	04JAN1931	0	48	1
5	48	27070	563.958	1610	156	363.564	174	219	319	427	719	1030	1469	21919	11692	05JAN1960	05JAN1932	0	48	1
6	48	31279	651.646	4724	165	679.607	165	210	332	429	831	1111	1290	17903	11328	06JAN1949	06JAN1931	0	48	1
7	48	34332	715.250	3233	110	641.665	156	210	345	509	832	1263	2183	17904	11329	07JAN1949	07JAN1931	0	48	1
8	48	31813	662.771	2290	87	526.880	156	232	322	478	768	1093	1957	17905	11330	08JAN1949	08JAN1931	0	48	1
9	48	29521	615.021	1808	83	396.767	147	230	325	507	744	1178	1379	17906	11331	09JAN1949	09JAN1931	0	48	1
10	48	29384	612.167	1603	83	381.140	147	210	325	529	750	1134	1478	16811	11332	10JAN1946	10JAN1931	0	48	1
11	48	28261	588.771	1968	110	393.346	138	229	301	464	741	1021	1388	27404	11333	11JAN1975	11JAN1931	0	48	1
12	48	26442	550.875	2643	138	409.048	147	256	306	450	655	904	1209	27405	14621	12JAN1975	12JAN1940	0	48	1
13	48	24871	518.146	2246	129	345.814	147	238	310	459	581	739	1120	27406	11335	13JAN1975	13JAN1931	0	48	1
14	48	24198	504.125	1688	92	305.915	151	238	300	437	581	788	1084	27407	11336	14JAN1975	14JAN1931	0	48	1
15	48	24525	510.937	1171	74	262.069	147	238	290	437	635	904	976	27408	11337	15JAN1975	15JAN1931	0	48	1
16	48	24513	510.687	1302	92	269.332	147	243	300	455	680	869	965	22661	11338	16JAN1962	16JAN1931	0	48	1
17	48	22599	470.812	1003	110	217.819	142	238	281	409	627	741	856	22662	11339	17JAN1962	17JAN1931	0	48	1
.																				
351	49	30342	619.224	2156	65	431.526	135	183	300	489	818	1102	1458	20805	11308	17DEC1956	17DEC1930	0	49	0
352	49	29144	594.776	1692	78	363.753	142	179	315	491	818	1200	1247	20806	11309	18DEC1956	18DEC1930	0	49	0
353	49	28986	591.551	1808	117	362.247	125	167	301	617	777	988	1263	20076	11310	19DEC1954	19DEC1930	0	49	0
354	49	31525	643.367	3730	88	562.783	111	210	283	528	852	1120	1352	21173	11311	20DEC1957	20DEC1930	0	49	0
355	49	44876	915.837	10700	95	1793.876	120	210	280	528	849	1012	1531	21174	11312	21DEC1957	21DEC1930	0	49	0
356	49	35687	728.306	5622	112	960.743	119	210	276	509	720	940	1729	27019	23732	22DEC1973	22DEC1964	0	49	0
357	49	31383	640.469	3485	110	616.542	110	221	280	491	680	895	2033	27020	11314	23DEC1973	23DEC1930	0	49	0
358	49	29063	593.122	2377	87	472.976	109	220	298	473	647	1102	1746	27021	11315	24DEC1973	24DEC1930	0	49	0
359	49	27560	562.449	2138	74	416.612	147	196	301	473	581	949	1450	15334	11316	25DEC1941	25DEC1930	0	49	0
360	49	28690	585.510	2240	92	457.691	130	200	305	463	638	1164	1460	21179	11317	26DEC1957	26DEC1930	0	49	0
361	49	32719	667.735	4051	124	730.906	129	219	328	471	629	1204	1924	27024	11683	27DEC1973	27DEC1931	0	49	0
362	49	30526	622.980	3349	98	567.862	141	210	319	469	722	1006	1728	27025	11684	28DEC1973	28DEC1931	0	49	0
363	49	29548	603.020	2487	108	459.409	110	201	319	456	715	1218	1560	27026	11685	29DEC1973	29DEC1931	0	49	0
364	49	35359	721.612	4644	92	841.856	116	203	301	496	667	1388	1937	17896	11321	30DEC1948	30DEC1930	0	49	0
365	49	34865	711.531	4272	78	834.734	100	220	274	455	707	1260	1764	17897	11322	31DEC1948	31DEC1930	0	49	0

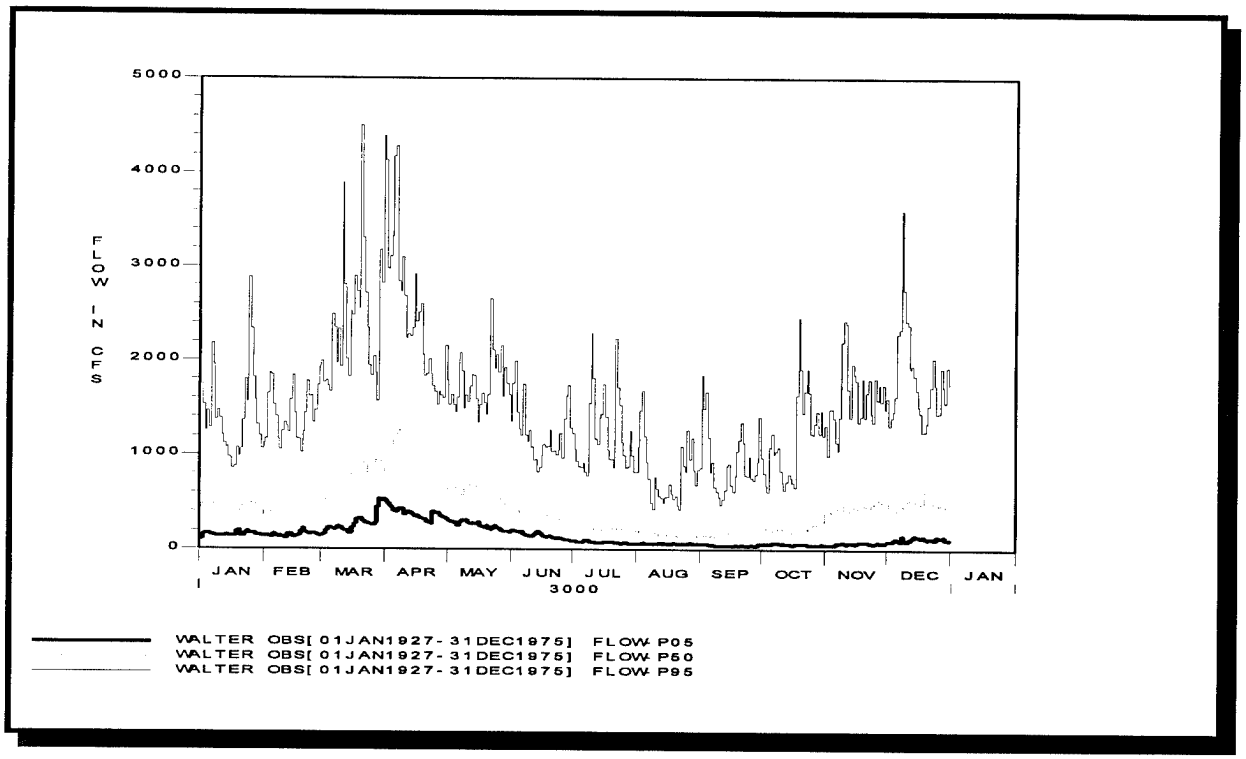
END OF FILE

**Statistics Summary File "WALTER.SUM"

```
*** Table of Statistics ***
*** Pathname> /LEHIGH/WALTER/FLOW/01JAN1990/1DAY/OBS/
*** Time window> 01JAN1927 - 31DEC1975
```

Bin	Max	Date	Min	Date	Aver	P50
1	2557.000	01JAN1949	69.000	01JAN1931	645.937	441.000
2	1966.000	02JAN1973	65.000	02JAN1931	623.062	473.000
3	1970.000	03JAN1960	78.000	03JAN1931	631.792	470.000
4	2250.000	04JAN1960	159.000	04JAN1931	597.354	450.000
5	1610.000	05JAN1960	156.000	05JAN1932	563.958	427.000
6	4724.000	06JAN1949	165.000	06JAN1931	651.646	429.000
7	3233.000	07JAN1949	110.000	07JAN1931	715.250	509.000
8	2290.000	08JAN1949	87.000	08JAN1931	662.771	478.000
9	1808.000	09JAN1949	83.000	09JAN1931	615.021	507.000
10	1603.000	10JAN1946	83.000	10JAN1931	612.167	529.000
11	1968.000	11JAN1975	110.000	11JAN1931	588.771	464.000
12	2643.000	12JAN1975	138.000	12JAN1940	550.875	450.000
13	2246.000	13JAN1975	129.000	13JAN1931	518.146	459.000
14	1688.000	14JAN1975	92.000	14JAN1931	504.125	437.000
15	1171.000	15JAN1975	74.000	15JAN1931	510.937	437.000
16	1302.000	16JAN1962	92.000	16JAN1931	510.687	455.000
17	1003.000	17JAN1962	110.000	17JAN1931	470.812	409.000
.						
348	2263.000	14DEC1927	76.000	14DEC1930	718.673	516.000
349	2308.000	15DEC1956	65.000	15DEC1930	691.367	479.000
350	2352.000	16DEC1956	60.000	16DEC1930	646.918	491.000
351	2156.000	17DEC1956	65.000	17DEC1930	619.224	489.000
352	1692.000	18DEC1956	78.000	18DEC1930	594.776	491.000
353	1808.000	19DEC1954	117.000	19DEC1930	591.551	617.000
354	3730.000	20DEC1957	88.000	20DEC1930	643.367	528.000
355	10700.000	21DEC1957	95.000	21DEC1930	915.837	528.000
356	5622.000	22DEC1973	112.000	22DEC1964	728.306	509.000
357	3485.000	23DEC1973	110.000	23DEC1930	640.469	491.000
358	2377.000	24DEC1973	87.000	24DEC1930	593.122	473.000
359	2138.000	25DEC1941	74.000	25DEC1930	562.449	473.000
360	2240.000	26DEC1957	92.000	26DEC1930	585.510	463.000
361	4051.000	27DEC1973	124.000	27DEC1931	667.735	471.000
362	3349.000	28DEC1973	98.000	28DEC1931	622.980	469.000
363	2487.000	29DEC1973	108.000	29DEC1931	603.020	456.000
364	4644.000	30DEC1948	92.000	30DEC1930	721.612	496.000
365	4272.000	31DEC1948	78.000	31DEC1930	711.531	455.000


FLOW IN CFS
2000
0
40000
30000
20000
10000
0
JAN FEB MAR APR MAY JUN JUL AUG SEP OCT NOV DEC JAN
3000
WALTER OBS[01JAN1927-31DEC1975] FLOW-MAX
WALTER OBS[01JAN1927-31DEC1975] FLOW-AVER
WALTER OBS[01JAN1927-31DEC1975] FLOW-MIN


FLOW IN CFS
5000
4000
3000
2000
1000
0
JAN FEB MAR APR MAY JUN JUL AUG SEP OCT NOV DEC JAN
3000
WALTER OBS[01JAN1927-31DEC1975] FLOW-P05
WALTER OBS[01JAN1927-31DEC1975] FLOW-P50
WALTER OBS[01JAN1927-31DEC1975] FLOW-P95

Name: **TSHIFT** **Shift time series in time**
Use: CO TY=TSHIFT(TX,DT)

Derive a time series TY by shifting times in TX by DT. TX may be regular or irregular. DT is specified in the form nnT, where nn is the number of time units T and T may be M for minutes, H for hours, or D for days. Units and type are preserved. An IF condition has no effect.

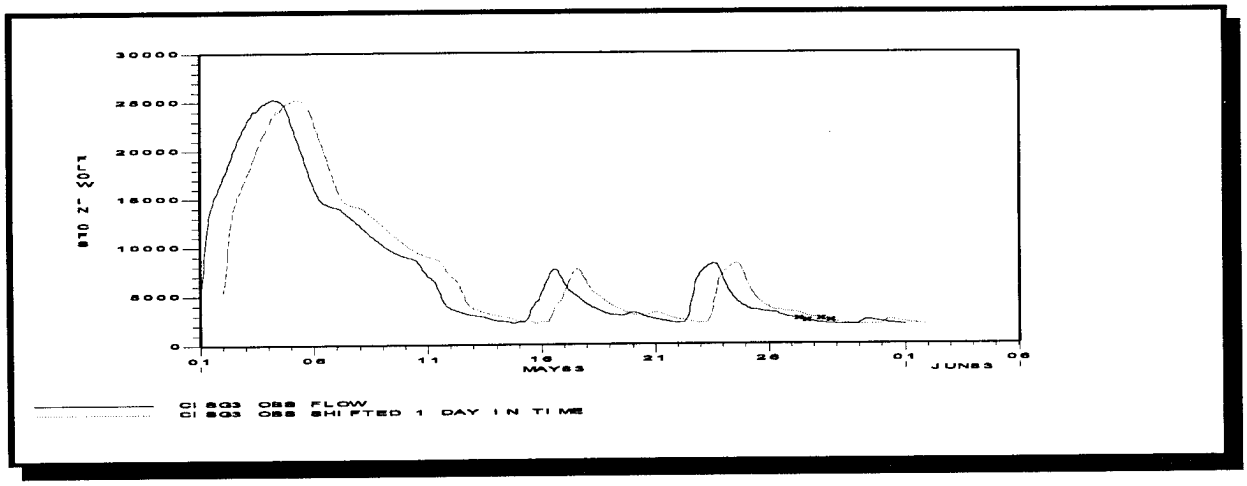
Example: The following example uses the TSHIFT function to shift a regular time series record in time by one day.

Macro:

```
MACRO TTSHIFT
!=R TTSHIFT
CL ALL
TI 01MAY1983 0100 31MAY83 2400
GET T1=testdb.dss:/SCIOTO/CISG3/FLOW/01MAY1983/1HOUR/OBS/
COMP TS=TSHIFT(T1,1D)
PUT.A TS=B=TTSHIFT,F=CAL
$CONTINUE INCASE OF ERROR
ENDMACRO
```

Execution:

```
I>!R TTSHIFT
  ALL VARIABLES HAVE BEEN CLEARED
    -----DSS---ZOPEN:  Existing File Opened,  File: TESTDB.DSS
                       Unit:  71;  DSS Version: 6-IA
 -----DSS--- ZREAD Unit  71; Vers.   22:  /SCIOTO/CISG3/FLOW/01MAY1983/1HOUR/OBS/
 -----DSS---ZWRITE Unit  71; Vers.    2:  /SCIOTO/TTSHIFT/FLOW/01MAY1983/1HOUR/CAL/
 -----DSS---ZWRITE Unit  71; Vers.    2:  /SCIOTO/TTSHIFT/FLOW/01JUN1983/1HOUR/CAL/
```



Name: **TSNAP** **Snap Irregular times to nearest Regular period**
Use: CO TY=TSNAP(TX,DT,TOFF,TBACK,TFORWARD)

Derive a regular time series from existing time series TX. TX may be regular or irregular. The new time series will be at a time interval of DT, interval offset of TOFF, look back of TBACK, and look forward of TFORWARD. Parameters: DT, TOFF,TBACK, and TFORWARD are expressed as units of time "nnT", where "nn" is a number and "T" is MIN (minutes), H (hours), D (days), W (weeks), MON(months), or Y (years). An IF condition has no effect.

Example: The following example uses the TSNAP function to convert irregular time series data that is reported close to the hour to regular time series at the hour. Take notice of the two irregular time series data values that do not fall within five minutes from the hour.

Macro:

```
MACRO TTSNAP
CL ALL
TIME 01MAY1983 0100 02MAY1983 0100
GET FLOW=testdb.dss:/SCIOTO/BMRI2/FLOW/01MAY1983/IR-MONTH/OBS-I/
COM FLOWR=TSNAP(FLOW,1H,0MIN,05MIN,05MIN)
$CONTINUE INCASE OF ERROR
ENDMACRO
```

Execution:

```
I>!R TTSNAP
  ALL VARIABLES HAVE BEEN CLEARED
    -----DSS---ZOPEN:  Existing File Opened,  File: TESTDB.DSS
                       Unit:  71;  DSS Version: 6-IA
```

Variables ----		FLOW	FLOWR
Units	----	CFS	CFS
Time	Date		
0100	01MAY1983	100.735	100.735
0104	01MAY1983	100.735	-
0200	01MAY1983	-	100.085
0204	01MAY1983	100.085	-
0300	01MAY1983	-	99.436
0304	01MAY1983	99.436	-
0400	01MAY1983	-	M
0500	01MAY1983	-	98.140
0504	01MAY1983	98.140	-
0600	01MAY1983	-	M
0700	01MAY1983	-	M
0730	**01MAY1983**	**96.844**	**-**
0800	01MAY1983	-	96.196
0804	01MAY1983	96.196	-
0900	01MAY1983	-	95.873
0904	01MAY1983	95.873	-
1000	01MAY1983	-	M
1100	01MAY1983	-	97.491
1104	01MAY1983	97.491	-
1200	01MAY1983	-	102.035
1204	01MAY1983	102.035	-
1300	01MAY1983	-	106.597
1304	01MAY1983	106.597	-
1400	01MAY1983	-	M
1500	01MAY1983	-	109.211
1504	01MAY1983	109.211	-
1600	01MAY1983	-	111.504
1604	01MAY1983	111.504	-
1700	01MAY1983	-	112.815
1704	01MAY1983	112.815	-
1800	01MAY1983	-	M
1900	01MAY1983	-	115.443
1904	01MAY1983	115.443	-
2000	01MAY1983	-	M
2014	**01MAY1983**	**115.443**	**-**
2100	01MAY1983	-	114.457
2104	01MAY1983	114.457	-
2200	01MAY1983	-	M
2300	01MAY1983	-	112.815
2304	01MAY1983	112.815	-
2400	01MAY1983	-	112.487
0004	02MAY1983	112.487	-
0100	02MAY1983	-	M

Name: **TTSR** **Transform time series to regular**
Use: CO TY=TTSR(TX,DT,TOFF,FUNCT,TREAT,TYPE)

Derive a new regular time series from existing time series TX. TX may be regular or irregular. The new time series will be at a time interval of DT, and an interval offset of TOFF. DT and TOFF must each be expressed in the form nnT, where T may be MIN, HOUR, DAY, WEEK, TRI, MON, or YEAR (e.g. 1DAY). FUNCT is one of the following:

INT	-	Interpolation at end of interval
MAX	-	Maximum over interval
MIN	-	Minimum over interval
AVE	-	Average over interval
ACC	-	Accumulation over interval
ITG	-	Integration over interval
NUM	-	Number of valid data over interval

TREAT controls the treatment of the new generated data value when missing data occur in the interval. It must be expressed in either the form nn# or nn%. If nn# is used nn gives the number of missing data values in the interval that will be accepted in the computation of the new value. If nn% is used nn gives the percent of missing data values in the interval that will be accepted in the computation of the new value. If the criteria is met a data value will be generated, if not a missing value will occur.

TYPE provides the user control to override how the interpolation and processing of data occurs. TYPE must be one of the following: DEFAULT, INST-VAL, PER-AVER, or PER-CUM. If DEFAULT is used, processing depends on the data type stored in the DSS data record. Otherwise processing will be performed as if the data type were as given by TYPE. Data type INST-VAL considers the data to change linearly from the previous data value to the current data value over the interval. Data type PER-AVER considers the data to be constant at the current data value over the interval. Data type PER-CUM considers the data to increase from zero (0.0) up to the current value over the interval.

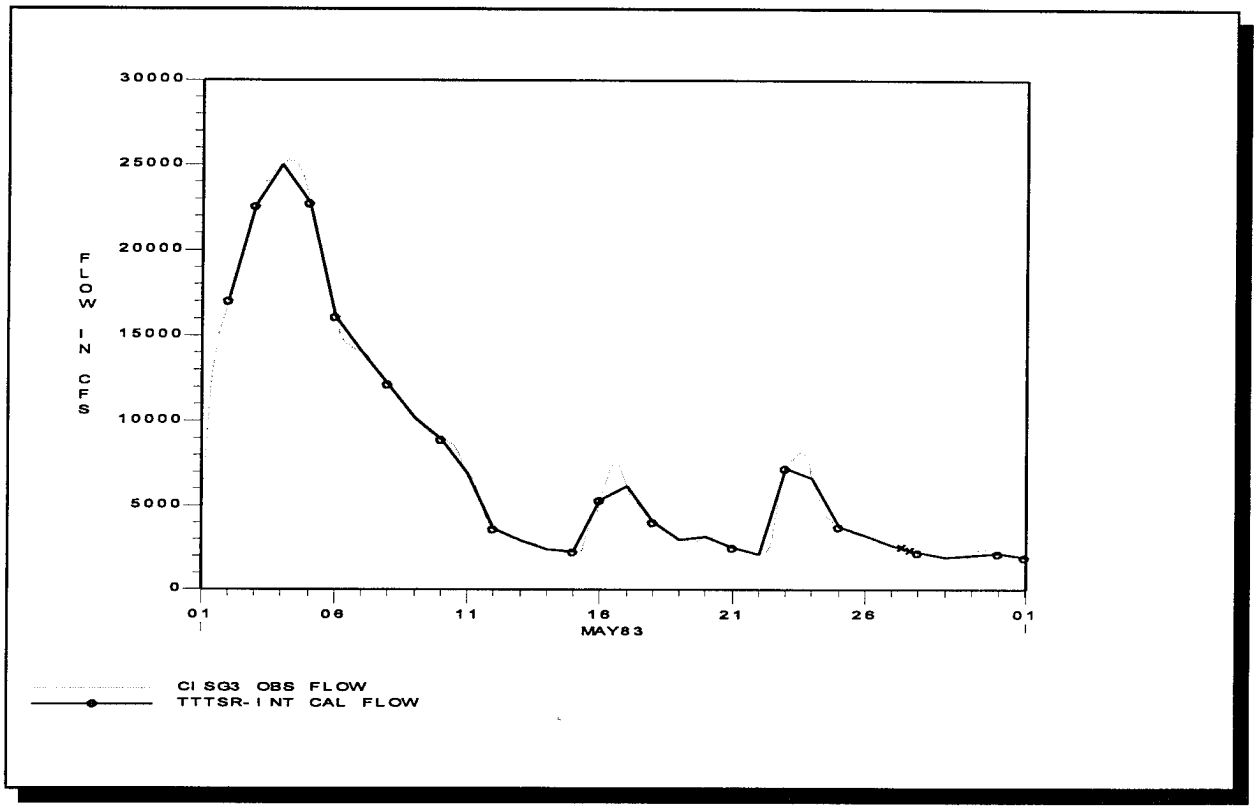
Example: The following example uses the TTSR function to derive a new regular time series from an existing time series at one hour intervals to a daily interval using six of the FUNCT parameters. Note: This example is not a real application and only serves to demonstrate the use and results of the function.

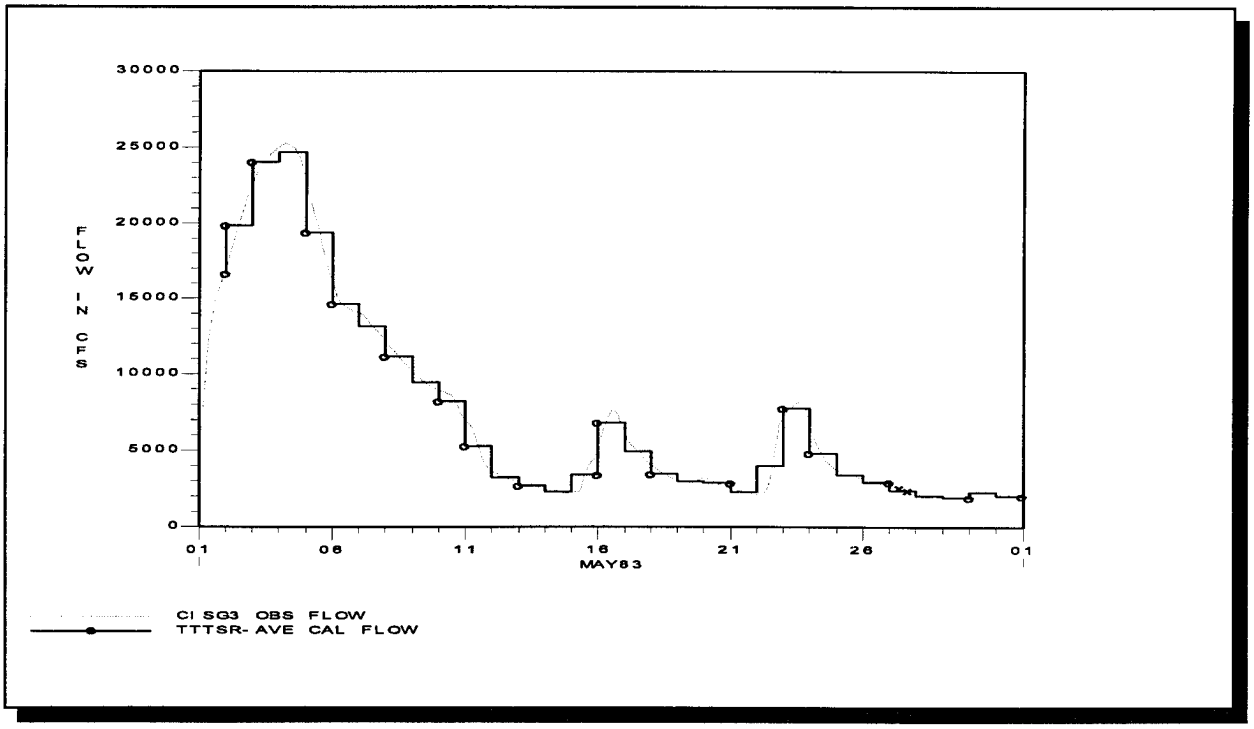
Macro:

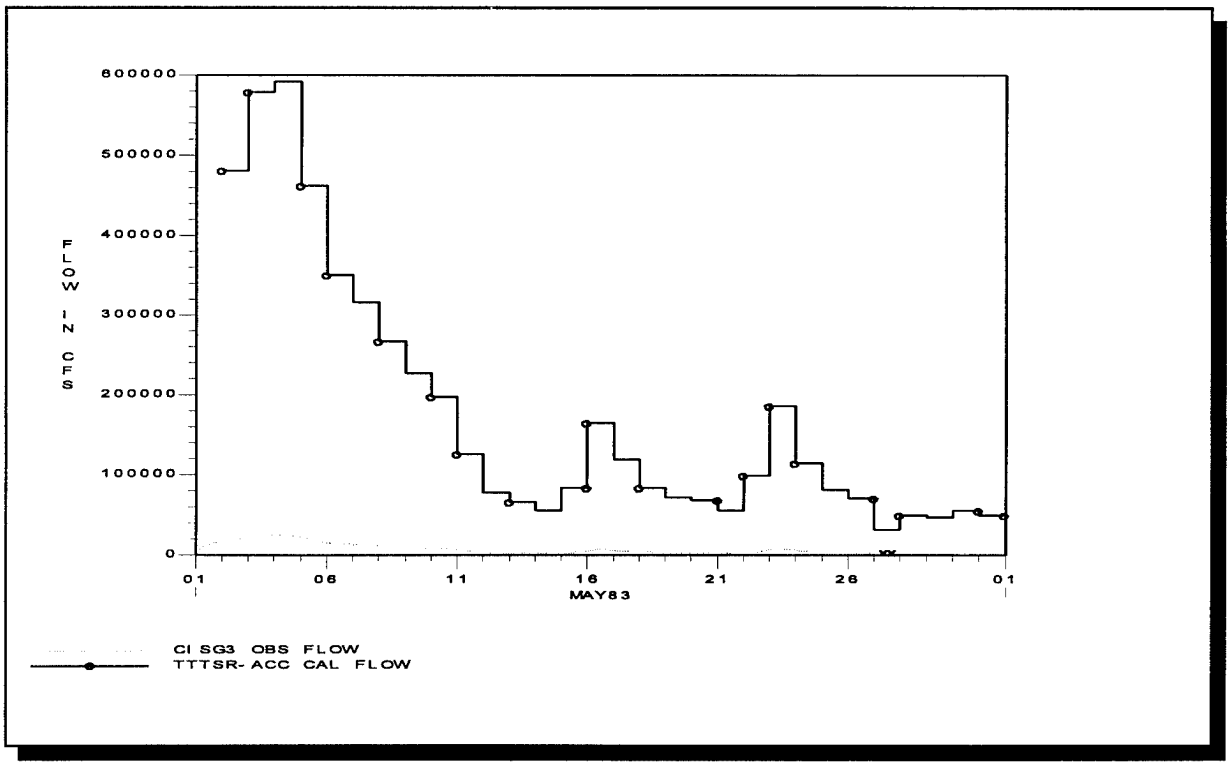
```
MACRO TTTSR
CL ALL
TI 01MAY83 0100 31MAY83 2400
GET TX=testdb.dss:/SCIOTO/CISG3/FLOW/01MAY1983/1HOUR/OBS/
COMP TS=TTSR(TX,24H,0H,INT,20#,DEFAULT)
PUT.A TS=B=TTTSR-INT,E=1DAY,F=CAL
COMP TS=TTSR(TX,24H,0H,AVE,20#,DEFAULT)
PUT.A TS=B=TTTSR-AVE,E=1DAY,F=CAL
COMP TS=TTSR(TX,24H,0H,ACC,20#,DEFAULT)
PUT.A TS=B=TTTSR-ACC,E=1DAY,F=CAL
COMP TS=TTSR(TX,24H,0H,ITG,20#,DEFAULT)
PUT.A TS=B=TTTSR-ITG,E=1DAY,F=CAL
COMP TS=TTSR(TX,24H,0H,MAX,20#,DEFAULT)
PUT.A TS=B=TTTSR-MAX,E=1DAY,F=CAL
COMP TS=TTSR(TX,24H,0H,MIN,20#,DEFAULT)
PUT.A TS=B=TTTSR-MIN,E=1DAY,F=CAL
$CONTINUE INCASE OF ERROR
ENDMACRO
```

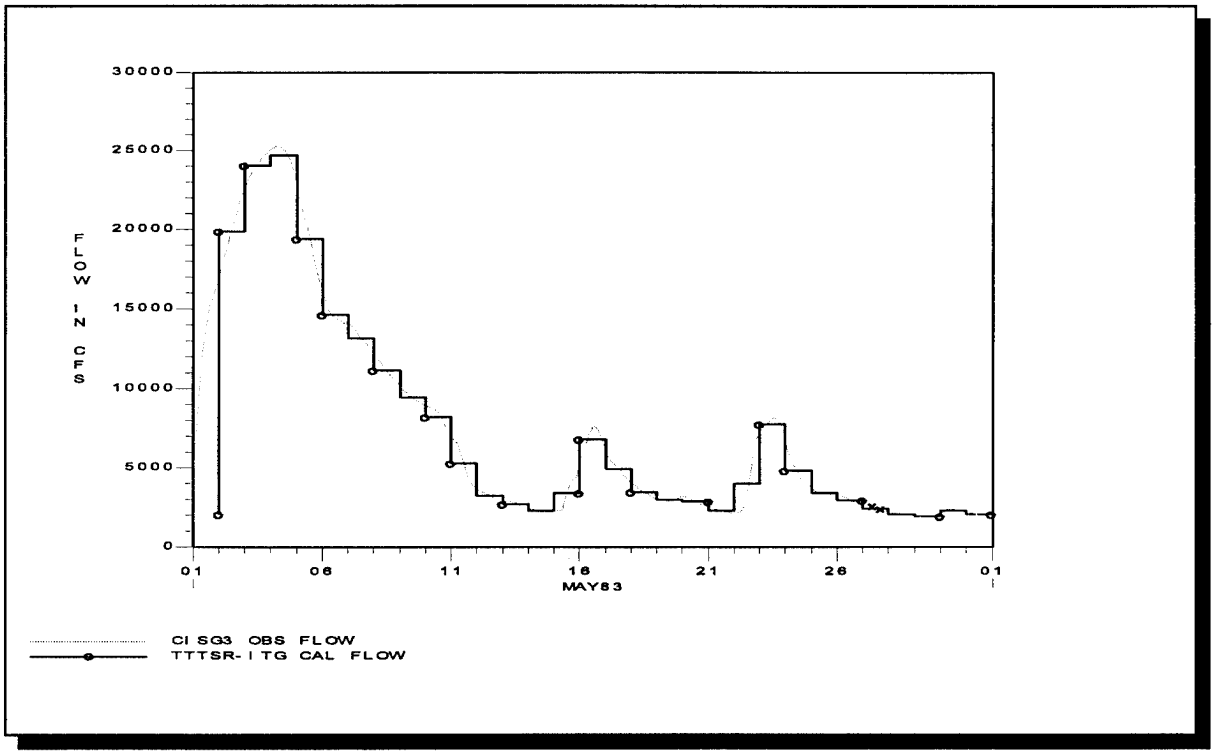
Execution:

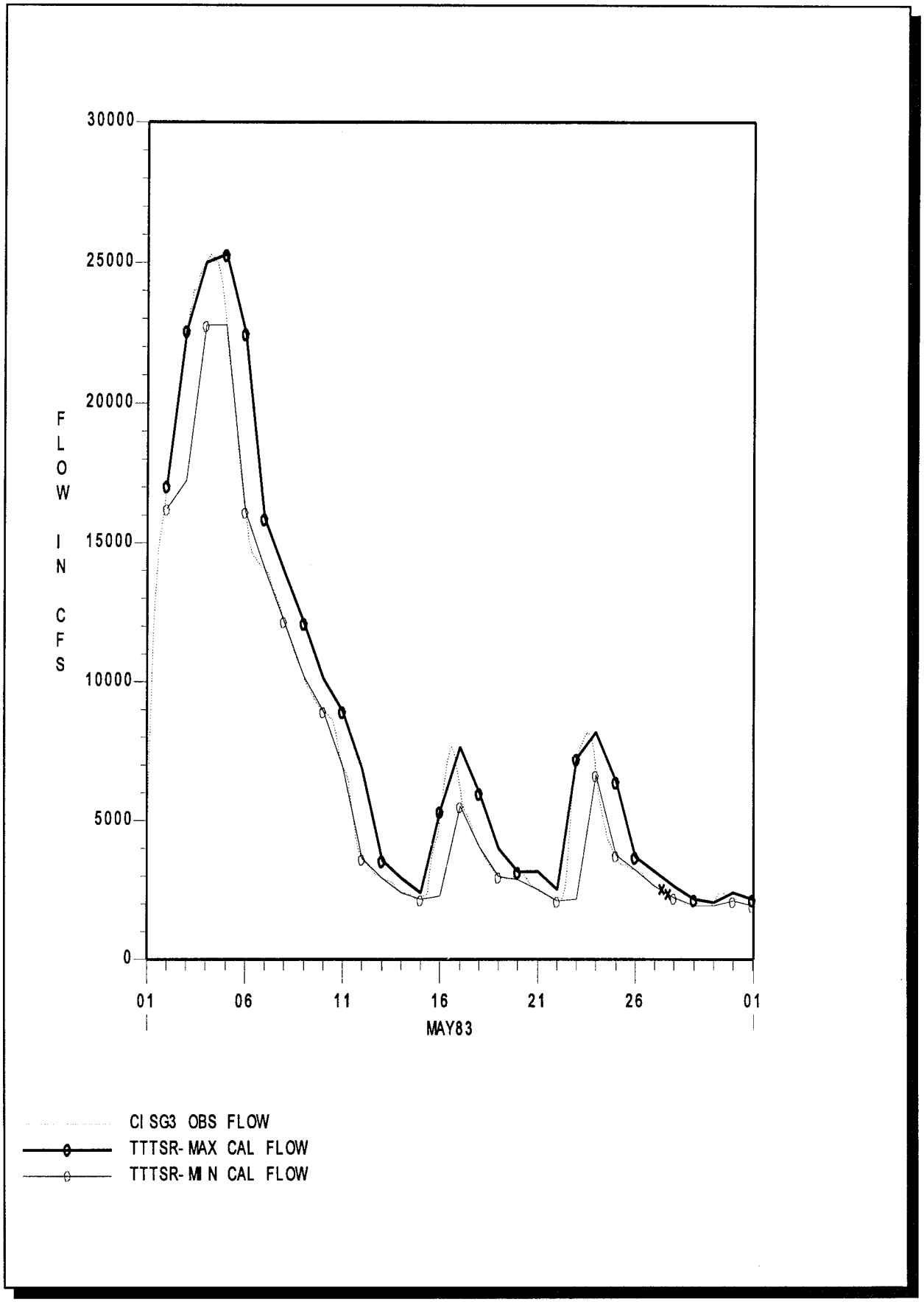
```
I>!R TTTSR
  ALL VARIABLES HAVE BEEN CLEARED
    -----DSS---ZOPEN:  Existing File Opened,  File: TESTDB.DSS
                       Unit:  71;  DSS Version: 6-IA
 -----DSS--- ZREAD Unit  71; Vers.   22:  /SCIOTO/CISG3/FLOW/01MAY1983/1HOUR/OBS/
 -----DSS---ZWRITE Unit  71; Vers.    4:  /SCIOTO/TTTSR-INT/FLOW/01JAN1983/1DAY/CAL/
 WARNING -- EXISTING VARIABLE RE-DEFINED
 -----DSS---ZWRITE Unit  71; Vers.    4:  /SCIOTO/TTTSR-AVE/FLOW/01JAN1983/1DAY/CAL/
 WARNING -- EXISTING VARIABLE RE-DEFINED
 -----DSS---ZWRITE Unit  71; Vers.    5:  /SCIOTO/TTTSR-ACC/FLOW/01JAN1983/1DAY/CAL/
 WARNING -- EXISTING VARIABLE RE-DEFINED
 -----DSS---ZWRITE Unit  71; Vers.    4:  /SCIOTO/TTTSR-ITG/FLOW/01JAN1983/1DAY/CAL/
 WARNING -- EXISTING VARIABLE RE-DEFINED
 -----DSS---ZWRITE Unit  71; Vers.    4:  /SCIOTO/TTTSR-MAX/FLOW/01JAN1983/1DAY/CAL/
 WARNING -- EXISTING VARIABLE RE-DEFINED
 -----DSS---ZWRITE Unit  71; Vers.    4:  /SCIOTO/TTTSR-MIN/FLOW/01JAN1983/1DAY/CAL/
```


30000
25000
20000
15000
10000
5000
0
FLOW IN CFS
01
06
11
16
21
26
01
MAY83
CI SG3 OBS FLOW
TTTSR-INT CAL FLOW


30000
25000
20000
15000
10000
5000
0
FLOW IN CFS
01
06
11
16
21
26
01
MAY83
CI SG3 OBS FLOW
TTTSR-AVE CAL FLOW


600000
500000
400000
300000
200000
100000
0
FLOW IN CFS
01
06
11
16
21
26
01
MAY83
CI SG3 OBS FLOW
TTTSR-ACC CAL FLOW


30000
25000
20000
15000
10000
5000
0
FLOW IN CFS
01
06
11
16
21
26
01
MAY83
CI SG3 OBS FLOW
TTTSR-ITG CAL FLOW


30000
25000
20000
15000
10000
5000
0
FLOW IN CFS
01
06
11
16
21
26
01
MAY83
CI SG3 OBS FLOW
TTTSR-MAX CAL FLOW
TTTSR-MIN CAL FLOW

Name: **TTSI** **Transform time series to irregular**
Use: CO TY=TTSI(TX,TZ,FUNCT,TREAT,TYPE)

Derive a new irregular time series from existing time series TX. TX may be regular or irregular. TZ is an existing time series at the desired new spacing of TY. All other parameters are as shown for the function TTSR.

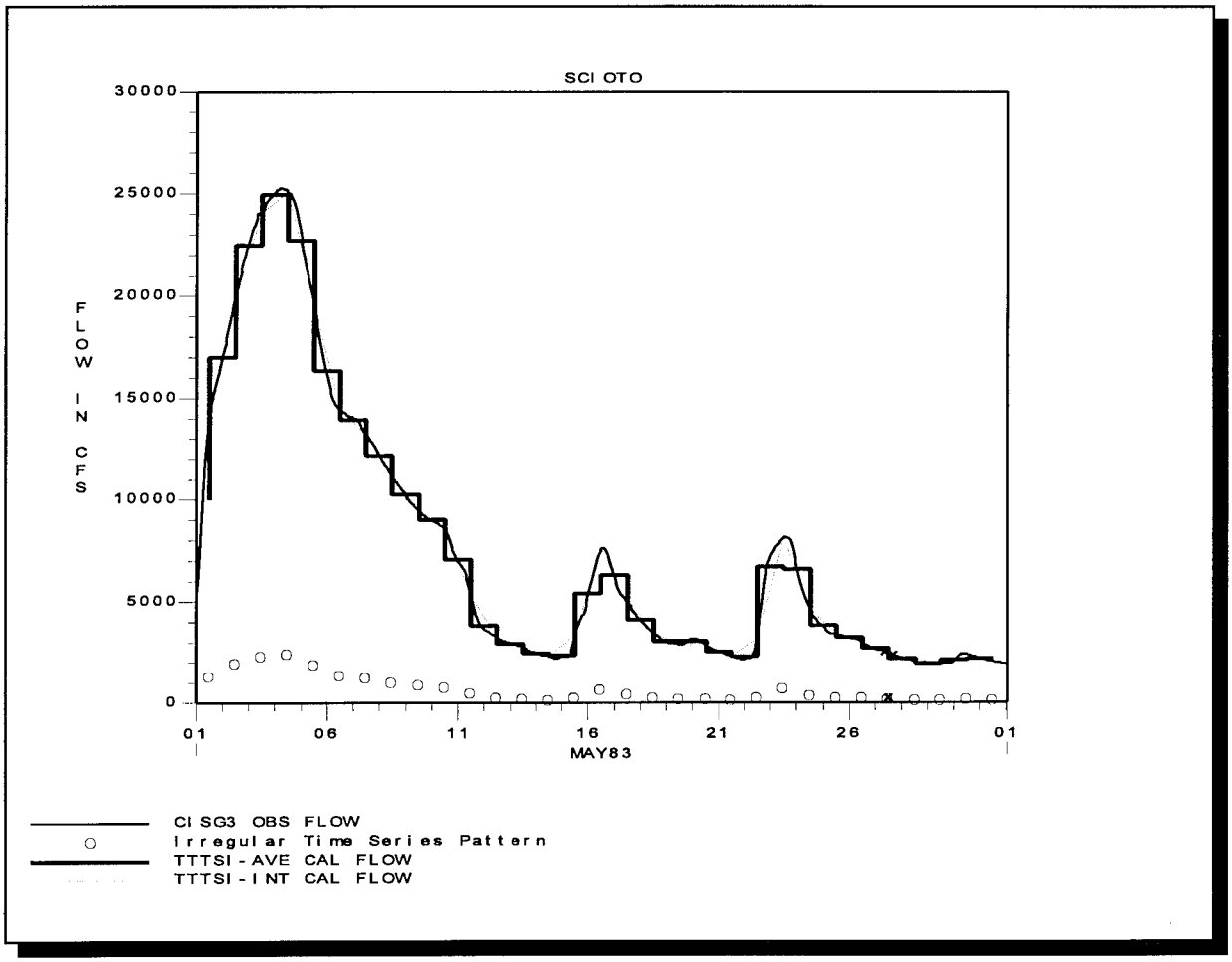
Example 1: The following example uses the TTSI function to take hourly regular data and generate an irregular time series based on an irregular time series pattern record. Note: This example is not a real application and only serves to demonstrate the use and results of the function.

Macro:

```
MACRO TTTSI
!R TTTSI5 AVE
!R TTTSI5 INT
ENDMACRO

MACRO TTTSI5 $FUN
CL ALL
TI 01MAY1983 0100 31MAY83 2400
GET TX=testdb.dss:/SCIOTO/CISG3/FLOW/01MAY1983/1HOUR/OBS/
GET TZ=/TZ/TTTS-PATTERN/FLOW/01MAY1983/IR-MONTH/NEW/
COMP TS=TTSI(TX,TZ,$FUN,12#,DEFAULT)
PUT.A TS=B=TTTSI-$FUN,E=IR-MONTH,F=CAL
$CONTINUE INCASE OF ERROR
ENDMACRO
```

Execution:

```
I>!R TTTSI
  ALL VARIABLES HAVE BEEN CLEARED
    -----DSS---ZOPEN:  Existing File Opened,  File: TESTDB.DSS
                       Unit:  71;  DSS Version: 6-IA
 -----DSS--- ZREAD Unit  71; Vers.   22:  /SCIOTO/CISG3/FLOW/01MAY1983/1HOUR/OBS/
 -----DSS---ZWRITE Unit  71; Vers.    4:  /TZ/TTTSI-AVE/FLOW/01MAY1983/IR-MONTH/CAL/
  ALL VARIABLES HAVE BEEN CLEARED
 -----DSS--- ZREAD Unit  71; Vers.   22:  /SCIOTO/CISG3/FLOW/01MAY1983/1HOUR/OBS/
 -----DSS---ZWRITE Unit  71; Vers.    3:  /TZ/TTTSI-INT/FLOW/01MAY1983/IR-MONTH/CAL/
```



TIME	DATE	CISG3 OBS FLOW CFS INST-VAL	TTTSI-AVE CAL FLOW CFS PER-AVER	TTTSI-INT CAL FLOW CFS INST-VAL
0100	01MAY1983	5432.000	-	-
0200	01MAY1983	5840.000	-	-
0300	01MAY1983	6710.000	-	-
0400	01MAY1983	7747.000	-	-
0500	01MAY1983	8783.000	-	-
0600	01MAY1983	9819.000	-	-
0700	01MAY1983	10855.000	-	-
0800	01MAY1983	11892.000	-	-
0900	01MAY1983	12497.000	-	-
1000	01MAY1983	13077.000	-	-
1100	01MAY1983	13526.000	-	-
1200	01MAY1983	13946.000	10039.550	13946.000
1300	01MAY1983	14298.000	-	-
1400	01MAY1983	14617.000	-	-
1500	01MAY1983	14903.000	-	-
1600	01MAY1983	15191.000	-	-
1700	01MAY1983	15408.000	-	-
1800	01MAY1983	15626.000	-	-
1900	01MAY1983	15845.000	-	-
2000	01MAY1983	16064.000	-	-
2100	01MAY1983	16285.000	-	-
2200	01MAY1983	16544.000	-	-
2300	01MAY1983	16767.000	-	-
2400	01MAY1983	17065.000	-	-
0100	02MAY1983	17253.000	-	-
0200	02MAY1983	17479.000	-	-
0300	02MAY1983	17705.000	-	-
0400	02MAY1983	17933.000	-	-
0500	02MAY1983	18200.000	-	-
0600	02MAY1983	18436.000	-	-
0700	02MAY1983	18674.000	-	-
0800	02MAY1983	18952.000	-	-
0900	02MAY1983	19231.000	-	-
1000	02MAY1983	19552.000	-	-
1100	02MAY1983	19794.000	-	-
1200	02MAY1983	20037.000	17033.900	20037.000

Example 2: The following example of the TTSI function is provided by NPD. The requirement is to average a daily regular time series of flows into an irregular time series of 14 periods per year required for a power simulation model (HYSSR) used in NPD. The "irregular" periods are 1) the complete months of Jan, Feb, Mar, May, Jun, Jul, Sep, Oct, Nov, and Dec and 2)split months of April and August, i.e., the first through the fifteenth days of April and August and then the remaining days of the two months. A date-time pattern (irregular time series) must be provided to tell TTSI at what points to "roll-up" the daily flows into averages. The pattern consists of 14 date-time stamps (and some dummy flows which are not used) for each of the years to be analyzed. For example, the pattern for 1928 would be:

```
31JAN1928 2400, 1000
28FEB1928 2400, 1000      !* Change to 29FEB for leap year
31MAR1928 2400, 1000
15APR1928 2400, 1000
30APR1928 2400, 1000
31MAY1928 2400, 1000
30JUN1928 2400, 1000
31JUL1928 2400, 1000
15AUG1928 2400, 1000
31AUG1928 2400, 1000
30SEP1928 2400, 1000
31OCT1928 2400, 1000
30NOV1928 2400, 1000
31DEC1928 2400, 1000
```

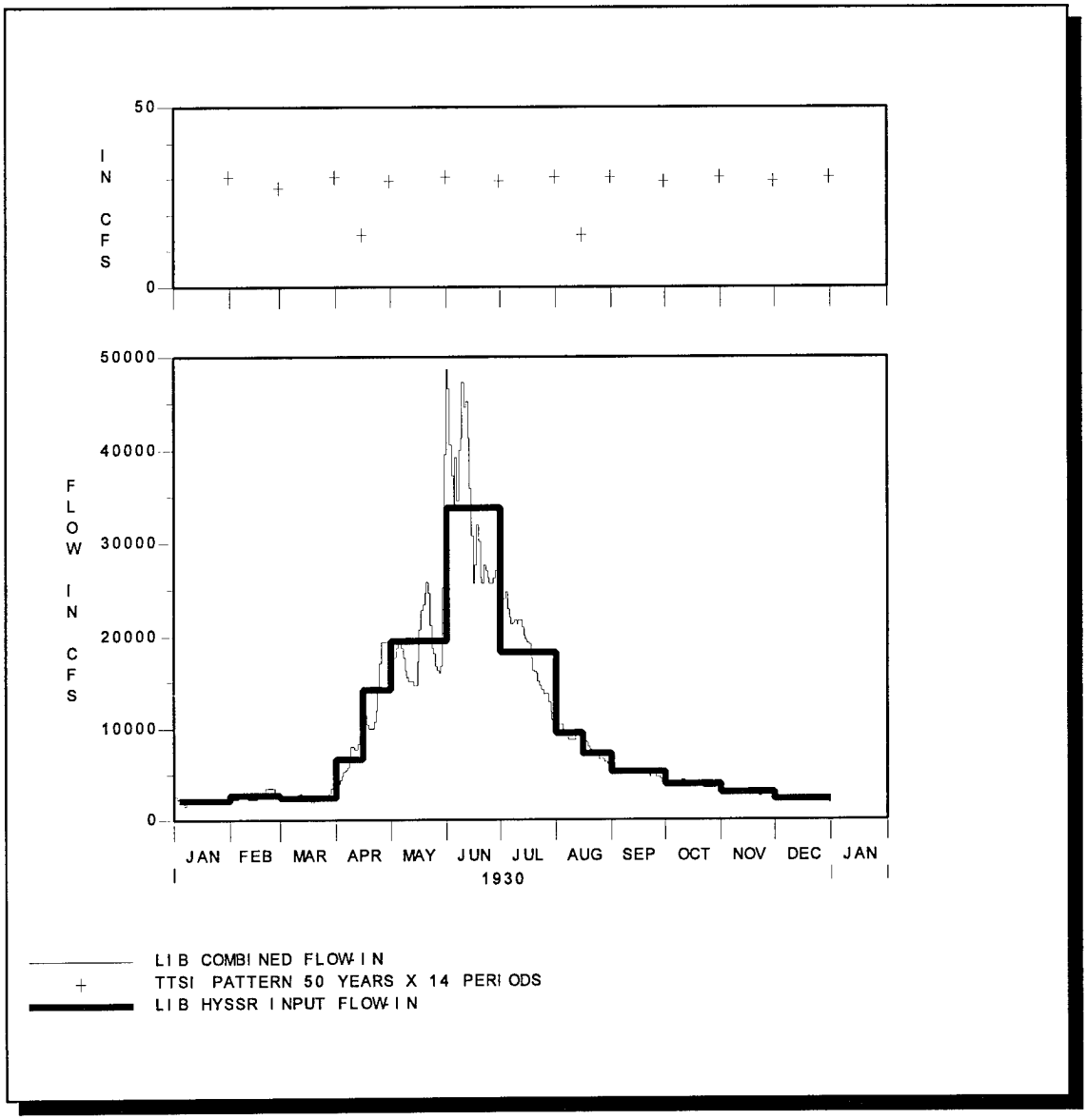
This pattern is then duplicated for all of the required years and read into the study DSS file (assigned to the PATRN variable in the example below). Executing TTSI would average the daily flows for each of the Jan-Mar months and store the monthly averages, average the flows for the first 15 days of April and compute an average for the last 15 days of April, etc.

Macro:

```
MACRO TTSI-NPD
LEAR ALL
ti 01jan1930 2400 31dec1935 2400
ge libin=npd://lib/flow-in//1day/combined/
ge patrn=npd://TTSI PATTERN//01JAN1920/IR-DECADE/50 YEARS X 14 PERIODS/
co libtst=ttsi(libin,patrn,ave,10%,per-aver)
sd libtst t=per-aver
put libtst=npd://lib/flow-in//ir-decade/hyssr input/
$CONTINUE
ENDMACRO
```

Execution:

```
I>!R TTSI-NPD
  ALL VARIABLES HAVE BEEN CLEARED
    -----DSS---ZOPEN:  Existing File Opened,  File: NPD.DSS
                       Unit:  71;  DSS Version: 6-IG
 -----DSS--- ZREAD Unit  71; Vers.    1:  //LIB/FLOW-IN/01JAN1930/1DAY/COMBINED/
 -----DSS--- ZREAD Unit  71; Vers.    1:  //LIB/FLOW-IN/01JAN1931/1DAY/COMBINED/
 -----DSS--- ZREAD Unit  71; Vers.    1:  //LIB/FLOW-IN/01JAN1932/1DAY/COMBINED/
 -----DSS--- ZREAD Unit  71; Vers.    1:  //LIB/FLOW-IN/01JAN1933/1DAY/COMBINED/
 -----DSS--- ZREAD Unit  71; Vers.    1:  //LIB/FLOW-IN/01JAN1934/1DAY/COMBINED/
 -----DSS--- ZREAD Unit  71; Vers.    1:  //LIB/FLOW-IN/01JAN1935/1DAY/COMBINED/
 -----DSS---ZWRITE Unit  71; Vers.    3:  //LIB/FLOW-IN/01JAN1930/IR-DECADE/HYSSR INPUT/
```


50
IN CFS
0
50000
40000
30000
20000
10000
0
FLOW IN CFS
JAN FEB MAR APR MAY JUN JUL AUG SEP OCT NOV DEC JAN
1930
LIB COMBINED FLOW-IN
TTSI PATTERN 50 YEARS X 14 PERIODS
LIB HYSSR INPUT FLOW-IN

REPGEN

Hydrologic Engineering Center
Report Generator

User's Manual

Version 3.2
March 1995

Hydrologic Engineering Center
U.S. Army Corps of Engineers
609 Second Street
Davis, California 95616-4687
(916) 756-1104

REPGEN

Table of Contents

Chapter **Page**

1. Introduction 1
    1.1 Purpose 1
    1.2 Acknowledgements 1
    1.3 Differences From Previous Versions 1

2. Description 3

3. Program Use 7

4. Input Description 9
    4.1 Input Format 9
    4.2 Report Form 11
    4.3 Variable Definitions 11
        4.3.1 Time Specification 12
            4.3.1.1 Time Expressions 13
        4.3.2 Predefined Variables 13
        4.3.3 Time Series Variables from DSS 14
        4.3.4 Variables from Text Files 15
        4.3.5 Variables From Mathematical Formulae and Functions 16
        4.3.6 Mathematical Functions 17
        4.3.7 Time-Related Functions 19
        4.3.8 Grouping of Variables 21
        4.3.9 Subscripted Time Series Variables 21
        4.3.10 Missing Values 23
        4.3.11 Display Formats for Variables 23
        4.3.12 Row and Column Displays 27
        4.3.13 Conditional Messages 28
    4.4 Function Characters 28

5. Examples 29

6. Obsolete Features 33

FIGURES

Page

1. Example Report 3
2. Input File 4
3. Example Gage Record 30

Chapter 1

Introduction

1.1 Purpose

The report generator, REPGEN, can be used to simplify and automate the production of routine reports in a variety of settings. REPGEN provides for the retrieval and presentation of data from a Data Storage System (DSS) or text file in a pre-specified, user-designed, fixed format. The format is the equivalent of a blank form onto which variable information is entered in designated locations. REPGEN could be used, for example, in a water control operations setting to automatically produce routine reports showing the current stage and flow at selected locations in a river basin.

1.2 Acknowledgements

Development of computer program REPGEN was supported by the Rock Island and St. Louis Districts, Corps of Engineers, in support of their water management activities. The basis for and structure of the program were conceived by Art Pabst. The program was designed and written by Paul Ely, Consultant. Substantial testing of the program was performed by the Sacramento District, which resulted in suggestions for a number of program improvements. Many of these improvements are incorporated in the present version of the program. The User's Manual was written by Paul Ely with contributions from Art Pabst, Dennis Huff, Al Montalvo, Carl Franke and John Peters. Bill Eichert was Director of the Hydrologic Engineering Center during development of the initial version of the program; Darryl Davis is the current Director.

1.3 Differences From Previous Versions

Arithmetic operations can be performed using multivalued-time-series variables.

Functions to accumulate time-series values and compute differences between successive values of a time series have been added.

Functions to round values to number of significant digits or to a designated place have been added.

Each moment in time is a single entity. Date and time-of-day are treated as a unit. Time can be adjusted using multi-term expressions. Time expressions can be used wherever time is specified. Functions to get/set specified components of time have been added.

A function to set the date for end-of-month for a given date has been added.

Function DATATIME returns a list of times for the designated variable. This list can be displayed in rows or columns like a time series.

New PICTURE characters are used to indicate which components of time will be displayed in the report.

When a variable which appears in the report is redefined, the last value will be displayed in the report. In previous versions of REPGEN the first-defined value was displayed.

An instruction to display intermediate variable values has been added to aid in debugging reports.

The method for computing a time window using DURATION and STIME or ETIME has been changed.

Chapter 2

Description

REPGEN essentially fills in designated locations in a user-defined form with variable text or numeric information that is retrieved from an external source, such as an operational, hydrologic database. In particular, data values can be entries from a DSS time series, numeric values from a text file, or blocks of text from a text file. REPGEN also provides for conditional text messages, such as missing data flags, that depend on data values.

Figure 1 is an example of a report created with REPGEN.

```
*****************************************

  Date: 01 JUNE 1985          Time: 0800

  Gage              Stage     Discharge

  Rescue            115.20    26450

        WARNING: ABOVE FLOOD STAGE

*****************************************
```

FIGURE 1. Example Report

Figure 2 shows input for creating the report in Figure 1. Program input includes the report form, data definitions and data display formats. The input data set is divided into 'form' and 'definitions' sections by the lines #FORM, #ENDFORM, #DEF, and #ENDDEF.

Locations where data is to be inserted are indicated by variable labels beginning with '%'. In Figure 2 the labels are %BASDATE, %BTM, %STG, %FLOW, and %MESSAGE.

```
#FORM
      ******************************************

      Date: %BASDATE                    Time: %BTM

      Gage                Stage               Discharge

      Rescue              %STG                %FLOW

      %MSG


      ******************************************
#ENDFORM
#DEF
%BASDATE
PICTURE = DDBAAABYYYY
%BTM = TIME (0800)
PICTURE = ZZZT
%STG
FILE = GAGEDATA  TYPE = DSS  A=MUDDY  B=RESCUE
C=STAGE  E=1HOUR  F=OBS
TIME = %BTM
PICTURE = NNN.ZZ
%FLOW
C = FLOW   PICTURE = NNNNN
#IF %STG > 100
%MSG = "WARNING: ABOVE FLOOD STAGE"
#ELSE
%MSG = ""
#ENDIF
#ENDDEF
```

FIGURE 2. Input File

A variable definition starts with the label as the first item in the line and includes all lines up to the next line beginning with a label. In Figure 2 the definition of %STG is:

```
%STG
FILE = GAGEDATA TYPE = DSS A=MUDDY B=RESCUE
C=STAGE E=1HOUR F=OBS
TIME = %BTM
PICTURE = NNN.ZZ
```

FILE is the file where stage data is located. TYPE shows the file to be a DSS file, and A, B, C, E and F set parts of the DSS pathname. TIME sets the time associated with the data retrieved from the DSS record.

PICTURE establishes the display format of the data. There is a one-to-one correspondence between PICTURE characters and characters placed in the report. The first character of the displayed data (which may be a blank) will be placed in the report where the % of

the corresponding variable label is located.

The report time is specified by variables %BASDATE and %BTM. Values for these variables can be set in the program execution command, or they will default to the current date and time when the program is run.

REPGEN provides capabilities for retrieving time series data from DSS. An element of the time series can be referenced individually or as a group of contiguous elements referenced collectively.

(This page intentionally left blank)

Chapter 3

Program Use

REPGEN was designed to use parameters on the execution line. Files and report time can be set using parameters in the program execution command as follows:

```
REPGEN [-option] [parameter]...
```

"parameter" has the form keyword=value

When option character D is used, values of variables named on #SHOW instructions will be written to the output file.

File names and program parameter values used by the program are:

Parameter	Meaning	Default
IN	Input filename	Standard input
OUT	Output filename	Standard output
REPORT	Report filename	report
FUNFILE	PREAD functions filename	repfun
MACFILE	PREAD macro filename	repmac
DATE	Date for intrinsic time variables %BASDATE and %BTM	today's date
TIME	Time for intrinsic time variables %BASDATE and %BTM	current time

DATE establishes the date for the variables %BASDATE and %BTM. It is defined with a 9-character date of the form DDMMMYYYY, where DD is the day of month, MMM are the first three letters of the month name, and YYYY is the year; e.g., 01MAY1985. Similarly, TIME establishes the time-of-day for the variables %BASDATE and %BTM. It is defined with a 4-character time from a 24-hour clock of the form HHMM, where HH is hour and MM is minutes, e.g., 0800. %BASDATE and %BTM are described in a following section.

Error messages are written to the terminal if the job is interactive; otherwise, they are written to batch job output. Several work files not listed here are also used; a complete list can be obtained with the execution command "REPGEN ?". Other files are used by REPGEN when they are named in the variable definitions.

On an IBM PC, the filenames and parameter values are:

Parameter	Meaning	Default
IN	Input filename	CON
OUT	Output filename	CON
REPORT	Report filename	REPORT
FUNFILE	PREAD functions	REPFUN
MACFILE	PREAD macros	REPMAC
DATE	Date for intrinsic time variables %BASDATE and %BTM	today's date
TIME	Time for intrinsic time variables %BASDATE and %BTM	current time

Chapter 4

Input Description

4.1 Input Format

The words 'line' and 'card' in this input description are used interchangeably to refer to a line in an input file. In parameter definitions, items enclosed in angle brackets, <>, are generic names; items enclosed in brackets, [], are optional; and a bar, |, is used to separate items where one or the other is to be used.

Lines beginning with # are used to divide the input file into the report form and variable definitions and to control processing. The '#' character must be in column one. Data is read from the input file in the following sequence:

```
        Optional Comments
#FORM
        Body of Form (no comments allowed)
#ENDFORM
        Optional Comments
#DEF
        Variable Definitions
#ENDDEF
        Optional Comments
```

#Preview

If a `#PREVIEW` line appears before the `#FORM` line, the report will show the PICTURE strings for each variable. Each line of the form will be printed twice: first the original line; then the same line with the variable label replaced by its PICTURE.

#SHOW `<variable> [PICTURE = <string>]`

The `#SHOW` instruction can be used to display intermediate values of a variable. Place the `#SHOW` instruction after the variable definition and the value of the variable will be written to the output file. Subscripts can be used to display selected elements of a time-series variable. An optional picture string can be used for numeric values. Note that values are not written to output unless option character D is used on the program execution command.

#FORM and #ENDFORM

`#FORM` and `#ENDFORM` lines define the limits of the report form. Comments cannot be included in the report form.

#DEF and #ENDDEF

`#DEF` and `#ENDDEF` lines define the limits of the variable definitions. However, comments may be used within this portion of the input file.

#IF-#ELSEIF-#ELSE-#ENDIF

`#IF`, `#ELSEIF`, `#ELSE`, and `#ENDIF` are used to skip lines in the variable definitions depending on the condition on a `#IF` or `#ELSEIF` line.

```
#IF <condition>
      variable definitions
#ELSEIF <condition>
      variable definitions
#ELSE
      variable definitions
#ENDIF
```

`#ELSEIF` and `#ELSE` lines are optional, but `#IF` and `#ENDIF` lines must be paired.

`<condition>` is a logical expression which can be evaluated to be true or false. A logical expression is two arithmetic expressions combined by a comparison operator.

Allowable comparison operators are:

=	equal
<>	not equal
<	less than
>	greater than
<= or =<	less than or equal
>= or =>	greater than or equal

Logical expressions may be joined or negated by logical operators. Allowable logical operators are:

NOT	negation
AND	logical and
OR	logical or

Example:

```
#IF MONOFYR(%date) >= 5 AND MONOFYR(%date) <= 10
%LEVEL = 11.7
#ELSE
%LEVEL = 12.9
#ENDIF
```

Comment

Lines with a # in the first column and a blank in the second column are ignored for report generation and can be used for comments.

Comments may not be included in the report form. Further, any line occurring before the `#FORM` line, between the `#ENDFORM` and `#DEF` lines, or after the `#ENDDEF` line will be ignored, and can therefore be used for comments.

4.2 Report Form

The report form is a literal image of the report, except that locations where data are to be inserted are indicated by variable labels. The number of characters occupied by a data value and the position of the field relative to the label position are specified in the display field definitions. If the specified field width is less than the width of the variable label, the line will be compressed to compensate for the difference in the number of characters in the variable label and the field width. If the field width is greater than the width of the variable label, adjacent characters will be overwritten.

4.3 Variable Definitions

Variable definitions describe the assignment of values represented by labels and their display formats. The value of a variable may be defined by: reference to a file; a direct assignment to another variable; assignment to the result of a mathematical expression; or assignment to a function.

Six types of variables can be used: numbers, times, alphanumeric strings, text, list, and paired-data. Numbers are read from DSS files or text files, or result from evaluating a mathematical expression. Numbers can be used in mathematical or logical expressions. Time consists of date and time-of-day. Alphanumeric strings are missing-data strings or other strings assigned in a statement. Text is read from a file. A single variable may consist of a block of one or more lines of text. List variables are used as parameters in math functions and cannot be displayed. Paired-data are read from DSS, and are used to interpolate values for number variables. Paired-data cannot be used in a report, but the values can be displayed using `#SHOW`.

A variable definition is entered in free form on one or more lines. The definition begins with a label line which has a label as the first item in the line. A variable label is two to eight characters beginning with a '%'. Only numerals (0, 1, ..., 9) and letters (A, B, ..., Z) may be used besides the initial '%'. No spaces are allowed in variable labels. A '%' in the report form may be used to indicate percent when it is followed by a space. The variable definition continues until a % line is entered for the next variable or an `#IF`, `#ELSEIF`, `#ELSE` or `#ENDIF` statement is encountered. Comments can be included in a variable definition by placing a '#' in the first column of the comment line. Any line within the variable definition portion of the input file which does not have a # or % as the first non-blank character is assumed to contain parameters defining the variable. Blank lines are ignored.

Variables defined more than once always have the value last defined at any point in the input sequence. Variables whose value is displayed in a report should not be defined more than once, but if they are the last-defined value will appear in the report. A warning message will be printed when a report variable is redefined. If too many report variables are redefined, REPGEN may run out of memory.

There is no distinction between upper and lower case characters in the variable definitions. Lower case characters will be interpreted as upper case, except for characters enclosed within double quotes, ", or the single character following an up arrow, ^.

4.3.1 Time Specification

Time consists of two parts, date and time-of-day. These two values are used together to indicate a single moment in time.

Time is specified as <date> <time_of_day> where

> <date> is a 9-character date of the form ddmmmyyyy (using 2 digits for year implies year in the 1900's), e.g., 21MAR1982, 03JAN88;
>
> <time-of-day> is a 4-character time in the form hhmm based on a 24-hour clock, e.g., 1630, 0400.

4.3.1.1 Time Expressions

Time expressions can be used wherever time is specified in REPGEN.

A time expression consists of

```
<time> [+/- <increment>] [+/- <increment>] ...
```

```
where <time> can be expressed as
      a time variable, or
      <date>[ <time_of_day>], or
      [<date> ]<time_of_day>;

+/- means plus or minus;
```

`<increment> is` expressed as an integer, n, combined with a unit of time: nMINUTE, nHOUR, nDAY, nMONTH, nYEAR. HOUR, DAY, MONTH, and YEAR can be abbreviated using their first letter. MINUTE can be abbreviated as MIN.

4.3.2 Predefined Variables

There are five variables which are defined when REPGEN begins: %CURDATE, %CTM, %BASDATE, %BTM, and %MISSDTA.

%CURDATE, %CTM, %BASDATE and %BTM are reference times. They may be used as arguments for time in a function or may be displayed in the report.

%CURDATE and %CTM are initially set to the same time. Both variables are defined to maintain compatibility with older reports.

%BASDATE and %BTM are initially set to the same time. Both variables are defined to maintain compatibility with older reports.

%BASDATE - time set using execution line parameters DATE and TIME. If DATE or TIME is not defined, the current date and/or time-of-day are used.

%BTM - same as %BASDATE.

%CURDATE - time when REPGEN begins running.

%CTM - same as %CURDATE.

%MISSDTA is a numeric value which is used to indicate missing values in a time series and may be defined by the user. The string set by parameter MISSTR will be displayed in the report for values equal to %MISSDTA.

4.3.3 Time Series Variables from DSS

Time series variables are read from DSS. A variable can be a single value or a series of values occurring within a time window.

Parameter TIME is used to read a single value from a time series. Parameters STIME, ETIME, and DURATION are used to set a time window to read a series of values. Up to 1000 values may be included in the window. Once the window is defined, any element or series of elements in the series may be referenced by a subscript or by the ELEMENT function. If a report uses several values from a time series, it is most efficient to define a time window which includes those values and then use subscripts or functions to identify the subsets or individual elements of the series in the definitions immediately following in the input sequence.

Parameters defining a time series variable are:

```
FILE = <file name>

TYPE = DSS

A = <pathname part A> .... F = <pathname part F>
```

Used to specify parts of the DSS pathname. If a blank is to be included in a pathname, the pathname part must be enclosed in double quotes, e.g., A="WHITE RIVER". Pathname parts remain in effect for all variable definitions until they are changed.

```
STIME = <time_expression>
```

Sets start time for a time series to be read from DSS.

```
ETIME = <time_expression>
```

Sets end time for a time series to be read from DSS.

```
DURATION = <duration>
```

Sets the duration of a time window. Duration is used with STIME or ETIME. <duration> can be nHOUR, nDAY, nWEEK, nMONTH, or nYEAR, where n is an integer. For example:

```
DURATION = 1MONTH
DURATION = 15DAY
```

REPGEN computes STIME or ETIME using the following formulas:

```
ETIME = STIME + DURATION - 1 minute
STIME = ETIME - DURATION + 1 minute
```

```
TIME = <time_expression>
```

Sets time for locating a data value in a DSS time series.

REPGEN gets the last entry at or before this time. The actual time of an entry can be obtained with the function DATATIME.

```
VALUE = <value type>
```

Sets allowable type of data value read from DSS. Default value is MISSOK. When VALUE = MISSOK, values representing missing data are 'acceptable'. That is, if the last entry at or before the time set by TIME represents a missing value, that value will be displayed as MISSTR. If the entry is a valid number, it will be displayed as defined by PICTURE.

If VALUE = NOMISS, the last entry, at or before the time set by TIME, not representing a missing value will be returned from DSS. The function DATATIME may be used to obtain the actual time of the data value.

If VALUE = EXACT, only a value which occurs at the time set by TIME will be returned from DSS. If no value is recorded at this time, the value will be undefined, and the string set for UNDEF will be displayed in the report.

VALUE is ignored when STIME and ETIME are used to define a time window.

4.3.4 Variables from Text Files

Text or numbers can be read from a text file. A text or number variable can be defined by setting parameters which specify the location of a piece of data in a text file. This file reference is defined by setting several parameters. Once a parameter is set, its value will be used for all future variables until it is redefined.

Parameters used for a text file reference are:

```
FILE = <file name>

TYPE = TEXT

START = <start string>
```

`<start string>` is an optional character string. A text file will be searched until <start string> is found. The line containing this string will be line zero. START should be null ("" or blank) if absolute line numbers are to be used.

```
END = <end string>
```

`<end string>` is an optional character string. Text lines will be copied until this string is encountered. The line containing this string will not be copied. END should be null ("" or blank) if absolute line numbers are to be used.

```
LINE =  <first line> [- [<last line>]]
```

This parameter sets the lines of a text file which will be copied to the report file. <first line> is the number of the first line to be copied. <last line> is the number of the last line to be copied. If START is not null, line numbers will be relative to the line containing <start string>.

Examples:

`LINE = 10`	will copy line 10
`LINE = 10-30`	will copy lines 10 through 30.
`LINE = 10-`	will copy from line 10 to end-of-file (or line containing <end string> if <end string> is not null).

```
COL = <first column> [- [<last column>]]
```

This parameter sets the columns of a text file which will be copied to the report file.

Examples:

`COL = 10`	will copy column 10
`COL = 10-30`	will copy columns 10 through 30
`COL = 10-`	will copy columns 10 to end-of-line

A text block (from a text file) is printed as it appears in the text file. The first line and column of the text block is printed where the first character (%) of the variable label appears in the report form. If the text block contains more than one line, additional lines will be added to the report as required by the text.

When a text-block variable appears on a report line, no other variables can be placed to the right of that variable on the line.

A number may be read from a text file only if these conditions are met:

1) Only one line is read from the file, and
2) the characters in the specified columns can be converted to a number.

4.3.5 Variables From Mathematical Formulae and Functions

A formula is a mathematical expression beginning with an equals sign, '='. The arithmetic operations (and operators) are addition (+), subtraction (-), multiplication (*), and division (/). Multiplication and division are performed before addition and subtraction. Parentheses may be used to modify the order in which operations are performed. Spaces in a mathematical expression are ignored.

If any term or factor in a mathematical formula has a value indicating missing data, the result of the formula will be a value indicating missing data.

Example:

```
%CHANGE = %NEW - %OLD
%A = (1.2*%P1 + 1.4*%P2 + 1.1*%P3) / 3
```

If one of the variables to the right of the equals sign is a missing data value, the variable to the left of the equals sign will be set to %MISSDTA.

A time series can be used in a mathematical expression. In that case the designated operation is carried out on each element of the time series. When two time series are combined using an arithmetic operation, the operation is performed using corresponding elements from each time series. Both time series must have the same number of elements and the corresponding values must occur at the same time. Each calculation is treated individually, i.e., if one time series contains a missing value, the result will have a missing value at the corresponding time, but values at other times will not be affected by the missing value. Two irregular-interval time series cannot be combined with an arithmetic operation.

4.3.6 Mathematical Functions

The following functions can be used in a mathematical expression:

`SUM (<treat>,<list>)` - summation of parameter list

`MAX (<treat>,<list>)` - find maximum value in parameter list

`MIN (<treat>,<list>)` - find minimum value in parameter list

`AVERAGE (<treat>,<list>)` - compute average of values in parameter list

`COUNT (<list>)` - number of valid data values in list

`<list>` consists of a one or more variables separated by commas. Spaces are ignored. Subscripted variables (see Section 4.3.9) can be used in a list.

`<treat>` indicates treatment of missing data:

`ZERO`	-	use zeros for missing data;
`IGNORE`	-	do not use missing data and decrease number of values used to compute average;
`MISS`	-	if missing data occurs in list, return missing data value for function.

For example:

Assume -901 indicates missing data, and
let %a = 10, %b = -901, %c = 20, then
MIN (ZERO, %a, %b, %c) is 0, and
MIN (IGNORE, %a, %b, %c) is 10, and
MIN (MISS, %a, %b, %c) is -901.

`ACCUM (<treat>,<variable>)` - generates a time series consisting of the cumulative amount from the beginning of the time series, `<variable>`, to the current value.

`<treat>` indicates treatment of missing data:

`ZERO`	-	use zeros for missing data;
`IGNORE`	-	do not use missing data. Corresponding value is missing, but remaining values are not affected;
`MISS`	-	if missing data occurs in time series, remaining cumulative values are missing.

`DIFF (<treat>,<variable>)` - generates a time series consisting of the differences between the current value and the preceding value. The difference for the first value is undefined.

`<treat>` indicates treatment of missing data:

`ZERO`	-	use zeros for missing data;
`IGNORE`	-	do not use missing data. Corresponding difference is undefined;
`MISS`	-	if data is missing, resulting difference is missing.

`RNDDIG (<variable>,ndig)` - makes a copy of <variable> with the values rounded to ndig significant digits.

`RNDPOS (<variable>,place)` - makes a copy of <variable> with the values rounded to place where place designates a position as a power of 10, e.g., -1 specifies rounding to the nearest tenth (0.1).

`INTERP (<ts-variable>,<pd-variable>,<n>` - uses linear interpolation to compute a value for each value in a time series. Result is a time series.

<ts-variable> is the time-series variable,
<pd-variable> is a variable containing paired-data values, and
<n> is 1 or 2, indicating that the first or second value in the data pairs is the dependent value.

For example: if %STG contains stage data, and %STGFLO contains a stage-flow rating table, with stage first in each pair, then INTERP (%STG,%STGFLO,2) will compute flows corresponding to the stages in %STG.

Subscripted variables can be used in the preceding functions.

4.3.7 Time-Related Functions

These functions are used to define the value of variables which refer to time.

<time_string> a string including time-of-day or date or both separated by a space

<time_variable> a variable which points to a date and time-of-day

<time_expression> an expression which begins with a <time_string> or <time_variable> which can be followed by modifying increments of time. (See paragraph on time expressions above.)

`DATATIME ()` - returns the actual time for the last data value read from a DSS time series. Assign result to a time variable. (Obsolete: use following form of DATATIME.)

`DATATIME (<time_series>)` - returns a time list for <time_series>. Assign result to a time variable.

`DAY (<time_expression>)` - returns two-digit day of month as a number.

`DAYOFWK (<time_expression>)` - returns name for day of week. Day of week is left-justified in a 9-character string. Use PICTURE to mask characters to be displayed.

`DAYOFYR (<time_expression>)` - returns day of year as a number.

`DMY2DATE (day, month, year)` - gets date from integers for day, month, and year. Time-of-day is assumed to be 2400 hours. Assign result to a time variable.

`EOM (<time-expression>)` - returns date of end-of-month for given date. Time-of-day is the time-of-day from the time expression, or 2400 if time-of-day cannot be determined from the expression.

`HOUR (<time_expression>)` - returns hour from time_expression as a number.

`LASTHOUR (<time-expression>)` - time-of-day is truncated to the last whole hour. Date is the same as the date from the time expression, or undefined if date cannot be determined from the expression.

`MINUTE (<time_expression>)` - returns minute from time_expression as a number.

`MONOFYR (<time_expression>)` - returns month of year as a number.

`MONTH (<time_expression>)` - returns month name. Month name is left-justified in a 9-character string. Use PICTURE to mask characters to be displayed.

`NDAYS (<time_expression>)` - returns the number of days in the month given by time expression.

`NEARHOUR (<time_expression>)` - time-of-day is rounded to nearest whole hour. See LASTHOUR.

`NEXTHOUR (<time_expression>)` - time-of-day is set to next whole hour when minutes is greater than zero. See LASTHOUR.

`SETTIME (<time_expression>,<component>,<value>)` - begin with time from <time_expression>, then change the designated component of time to the given value. When <component> is YEAR, MONTH, DAY, HOUR, or MINUTE, <value> is a number or a variable with a numeric value. If <component> is DATE or TIME, <value> is a time_expression, and date or time-of-day is changed to the value from the <value> time_expression. Assign result to a time_variable.

`TIME (<time_string>)` - identifies a string as a time string. Assign result to a time variable, e.g.,

%btm = TIME (0400 12JAN88)

`TIMEWHEN (<time_series>, <op>, <value>)` - finds the time when a time series reaches a specified value. <time_series> is a time-series variable, <op> is one of the operators:

LT	-	less than
LE	-	less than or equal
EQ	-	equal
GE	-	greater than or equal
GT	-	greater than

<value> is a number or a variable.

`YEAR (<time_expression)` - returns year as a four-digit number.

`YEAR2 (<time_expression>)` - returns year as a two-digit number.

`YMDHM2T (<year>,<month>,<day>,<hour>,<minute>)` - gets a time (date and time-of-day) from integers for year, month, day, hour, and minute. Assign result to a time variable.

`SELECT (FIRST | CENTER | LAST, <time_series>, <start_time>,<end_time>)`
The SELECT function selects one value from a time series for a time window. FIRST, CENTER, LAST indicate which value to return. FIRST means the result will be the first value encountered in the time window. CENTER indicates the value nearest the center of the time window, and LAST means the last value in the time window. If there are no data in the time window the result is a missing value. <time_series> is a time-series variable. The resulting value is selected from this time series. <start_time> is a time expression defining the beginning of the time window. <end_time> is a time expression setting the

end of the time window.

`SELECT (NEAREST, <time>, <time_series>, <start_time>, <end_time>)`
NEAREST indicates that the values nearest <time> will be returned. <time> is a time expression. <time_series> is a time-series variable. The resulting value is selected from this time series. <start_time> is a time expression defining the beginning of the time window. <end_time> is a time expression setting the end of the time window.

`SNAP (<time_series>, <interval>, <offset>)` The SNAP function creates a regular-interval time series from a time series. SNAP may be used to convert irregular-interval data to regular intervals, or to convert regular-interval data to another time interval. <time_series> is a time-series variable . <interval> is a time interval specified as nMINUTE, nHOUR, or nDAY. <offset> is a time string indicating the offset from a series of intervals beginning at midnight. For example, 6HOUR data is normally reported at 0600, 1200, 1800, and 2400; but setting <offset> to 0700 moves the times to 0700, 1300, 1900 and 0100.

4.3.8 Grouping of Variables

`GROUP (list)` - can be used to assign a label to a list of variables for use in the mathematical functions described previously.

`IGROUP (<index string>, n1, n2, <index label>)` - used to define a group using an index to reference the variables. n1 and n2 are integers; n2 > n1; the index string takes on the values of n1 through n2 and replaces that string in <index label>. Example: IGROUP (xx,1,31,%flowxx) will create a group of variable labels consisting of %flow1, %flow2, ..., %flow31.

Example:
The summation of variables can be computed by:

%SUM = %15 + %42 + %23 + %85

or %SUM = SUM (MISS,%15,%42,%23,%85)

or %A = GROUP (%15,%42,%23,%85)
%SUM = SUM (MISS,%A)

4.3.9 Subscripted Time Series Variables

A numeric variable may be defined from individual elements or a subset of a time series defined within a time window either by assignment or through a mathematical expression or function. The subset is similar to a group. The element or subset may be referenced by position

or by time using subscript notation or using the ELEMENT function.

Subscript notation is:

%var(<elem>)

where <elem> is:

position	-	a number indicating position in the time series
	-	START, indicating the first element of the time series
	-	END or LAST, indicating the last element of the time series
time	-	indicating the element occurring at time where time is a time expression.

For example, if a time series, %X, begins at 02JUL1987,0400 and ends at 06JUL1987,0300 with time interval of 1HOUR, then

%X(5)	is the element occurring at 02JUL1987,0800
%X(START)	is the element occurring at 02JUL1987,0400
%X(END)	is the element occurring at 06JUL1987,0300

If %DAT1 is 02JUL1987 and %TIM1 is 0400, then

%X(%DAT1,%TIM1)	is the element occurring at 02JUL1987,0400
%X(%DAT1,%TIM1+24H)	is the element occurring at 03JUL1987,0400
%X(04JUL1987,1200)	is the element occurring at 04JUL1987,1200

A subset of a time series can be referenced by position, times for the beginning and end of the subset, or by a block using the following notation:

%VAR(<range>) where range is

position1, position2	-	beginning position and end position
time1,time2	-	beginning time and end time
block	-	a month or year

Examples of block specification are:

JAN87	-	refers to a time series for January 1987
JAN	-	refers to a time series for January; the year is based on the parent time series
MONTH	-	uses month and year from %BASDATE
YEAR	-	uses year from %BASDATE

For example, if a time series, %Y, starts on 01OCT1986 and ends on 30SEP1987 with time interval of 1DAY, then

%Y(NOV) - refers to the time series for November 1986

The ELEMENT function provides reference to an element in a time series:

```
ELEMENT (<time_series>,BEFORE|AFTER|AT,<time>,<missval>)
```

<time_series> is a time-series variable,
<time> is a time_expression, and
<missval> indicates the action to take when the nearest data element is a missing value. If <missval> is MISSOK the missing value will be returned. If <missval> is NOMISS the nearest non-missing value will be returned if one exists in the time series.
BEFORE and AFTER indicate the direction from time to look for the nearest data element. When AT is specified, the resulting value will be undefined if there is not a value at the given time.

4.3.10 Missing Values

`MISSVAL (<list>)` - used to define the numeric values with which missing data is indicated. Up to 10 numbers can be included in <list>. Use: %MISSDTA = MISSVAL (-901,-902). The variable %MISSDTA is set to the first value in the list.

`%MISSDTA` - set to the first missing data value listed in the MISSVAL function. %MISSDTA is initially set to -901. If a missing data value is encountered while a mathematical formula is being evaluated, the result of the formula will be set to %MISSDTA. A special string, set by the MISSTR parameter, will be displayed in the report when a variable is equal to one of the missing data values.

`KNOWN (<variable>)` - used in a logical expression to determine if the value of <variable> is known. If the value is missing or undefined, KNOWN is false.

4.3.11 Display Formats for Variables

Display parameters determine how the value of a variable will be displayed in the report. Once a parameter is set, its value will be used for subsequent variables until the parameter is redefined.

```
DIGITS = <number of significant digits>
```

Digits sets the number of significant digits (n) to be displayed for a variable. The displayed number will be rounded to n significant digits. Non-significant digits will be shown as zeroes or blanks. All digits will be considered significant if n = 0.

PICTURE = <picture>

The appearance of the display field is shown in a picture. The picture consists of one or more of the characters described below. The number of characters in the picture is the maximum number of characters in the display field. If the number of digits in a number is greater than the field width, the field will be filled with asterisks (*). If an alphanumeric value has more characters than the field width, the left-most characters will be shown, and the right-most characters will be truncated.

A numeric display is defined by PICTURE characters S, N, Z, B, comma, and period (see below). Numeric fields are assumed to have an implied decimal point to the right of the field if a decimal point does not appear in the picture.

Time displays are defined by picture characters A, B, D, M, Y, T, and Z. Characters D, M, Y, and T select the component of time to be displayed. An A is used to indicate that a month name is to be displayed instead of the month number.

Alphanumeric values will be left-justified in the field. Day of week, and month of year use PICTURE to determine how they will be displayed. Other strings such as set by MISSTR and IF-THEN-ELSE statements appear exactly as they are given. Only picture characters A and B have special meaning for displaying alphanumeric fields.

PICTURE is not used to format a text block.

PICTURE is initialized to be "?PICTUR?", so it will be necessary for the user to define a picture for all variables.

S - Forces the sign of a number to be displayed. This must be the first character of a picture if it is used.

N - A numeric digit appears in the display field. Non-significant zeroes are printed as blanks, except where the number of significant digits requires trailing zeroes to be shown.

Z - A non-significant zero is to be displayed. This will override the number of significant digits.

. - A decimal point (.) is displayed. Only one decimal point may be used in a picture.

, - A comma (,) is displayed. More than one comma may be used. Commas preceding the first significant digit will appear as blanks.

A - An alphanumeric character will be displayed.

B - A blank will be displayed.

D - A numeric digit from the day component of time will be displayed.

M - A numeric digit from the month component of time will be displayed.

Y - A numeric digit from the year component of time will be displayed.

T - A numeric digit from the time-of-day component of time will be displayed.

F - A flag character will be displayed. Flag characters will be based on flag values from DSS. Note: This feature does not yet exist in DSS. Until it does, a blank will be printed for the letter F.

H - A numeric digit from the hour portion of time-of-day will be displayed.

Any other character which does not have a special meaning defined above may be used in a picture. This character will appear in its corresponding position in the display field.

Examples:

```
      Value                    Picture                  Display
      25.40                        NNN.NN                    24.4
      25.40                       SNNN.NN                   +25.4
      25.40                        NNN.ZZ                    25.40
      25.40                        ZZZ.ZZ                   025.40
    07JAN1985                   AABAAABAAAA              07 JAN 1985
      abcde                       AABABAA                  ab c de
     100000                      N,NNN,NNN                 100,000

Assume DIGITS = 3:
        1.20                   NNNBNNN.NN                     1.20
       12.04                   NNNBNNN.NN                    12.0
      120.45                   NNNBNNN.NN                   120.
     1204.5                    NNNBNNN.NN                 1 200.
    12045                      NNNBNNN.NN                12 000.
  1234567                      NNNBNNN.NN             **********
       12.34                   NNNBNNN                       12
```

```
MISSTR = <missing data string>
```

This parameter sets a string which will be displayed in the data field whenever a variable has a value equal to a value which represents missing data. MISSTR is initialized to be "?MISSTR?".

```
UNDEF = <undefined variable string>
```

This parameter sets a string which will be displayed for elements of a time series which are undefined. When a time series has more positions in the report form than values in the time series, this string will be displayed for the extra positions.

UNDEF defaults to the variable label.

```
JUSTIFY = L | C | R
```

This parameter sets how alphabetic and time strings will be justified within the display field. The string can be left justified (L), centered (C), or right-justified (R).

Alphabetic strings are normally left-justified.

Time is displayed using the PICTURE characters D, M, Y, H and T for day, month, year, hour, and time-of-day respectively. These characters can be preceded by Z's where leading zeros are to be displayed, but the last character of each component must be a D, M, Y, H or T. The letter A can be used to indicate that the alphabetic month is to be displayed instead of a numeric month. The month name will be right-justified in the space reserved by for month name. If the month name is longer than the reserved space, the right-most characters will be truncated.

Examples:

```
Picture                      Display
DDbAAAbYYYY                  4 JUL 1987
ZDbAAAbYYYY                  04 JUL 1987
MM/DD                        7/4
MM/DD/YYbbHH                 7/4/87  8
ZM/ZD/YYbbHH                 07/04/87  08
DDbAAAbYY                    4 JUL 87
ZDbAAAbYY                    04 JUL 87
AAAbDD,bYYYY                 JUL 4, 1987
AAAAAAAAAbDD,bYYYY           JULY 4, 1987

ZZZT                         0830
ZZ:ZT                        08:30
ZT                           30
TZ:ZT                         8:30
HH                            8
ZH                           08

AAAAAAAAAbDD,bYYYYb@bZZZT    JULY 4, 1987 @ 0830
```

Time strings are normally right-justified within the picture. The JUSTIFY parameter can be used to left-justify or center the string.

4.3.12 Row and Column Displays

Time series data can be printed in rows or columns in the report. The variable label is placed wherever a time series value is to appear. As each line of the report form is processed, the line is scanned from left to right, and the labels are replaced by the corresponding data value. That is, the first appearance of a label is replaced by the first value of the time series, the second appearance is replaced by the second value, etc. So, if a label is repeated in a row, the time series will be printed in a row. If the label is repeated in a column, the time series will be printed in a column. A label must be repeated for the number of values in the time series, otherwise all values will not be printed in the report. The UNDEF parameter, previously described, is used to define a string to replace values that may not exist under some circumstances, such as daily data for February 29. See Figure 6 for an example of input for a column-oriented report.

4.3.13 Conditional Messages

The #IF-#ELSEIF-#ELSE-#ENDIF statements may be used to display a message or flag when a condition is met by the data. For example:

```
#IF %stage > 1000
 %warning = "Flood Warning"
#ELSE
 %warning = ""
#ENDIF
```

and

```
#IF %b >= 100 AND %b < 110
 %range = "A"
#ELSEIF %b >= 110 AND %b < 125
 %range = "B"
#ELSEIF %b >= 125 AND %b < 150
 %range = "C"
#ELSEIF %b >= 150
 %range = "D"
#ELSEIF %b = %missdta
 %range = " "
#ENDIF
```

4.4 Function Characters

REPGEN uses PREAD to read and expand special characters which have been defined in the function file referenced by execution line parameter FUNFILE. Function characters are only expanded within the variable definitions. They are not expanded in the report form. Function expansion can be turned off or on by including PREAD instructions in the input file. Function character expansion is initially "on" when REPGEN begins execution. See the PREAD manual included in this document for details.

Chapter 5

Examples

Figure 3 shows an example input file that could be used to produce an annual streamflow report. This example shows how time series can be printed in columns.

```
#FORM
                    NATURAL FLOWS AT HAWLY
                     %YEAR WATER YEAR

        OCT   NOV   DEC   JAN   FEB   MAR   APR   MAY   JUN   JUL   AUG   SEP

  1     %OCT  %NOV  %DEC  %JAN  %FEB  %MAR  %APR  %MAY  %JUN  %JUL  %AUG  %SEP
  2     %OCT  %NOV  %DEC  %JAN  %FEB  %MAR  %APR  %MAY  %JUN  %JUL  %AUG  %SEP
  3     %OCT  %NOV  %DEC  %JAN  %FEB  %MAR  %APR  %MAY  %JUN  %JUL  %AUG  %SEP
  4     %OCT  %NOV  %DEC  %JAN  %FEB  %MAR  %APR  %MAY  %JUN  %JUL  %AUG  %SEP
  5     %OCT  %NOV  %DEC  %JAN  %FEB  %MAR  %APR  %MAY  %JUN  %JUL  %AUG  %SEP

  6     %OCT  %NOV  %DEC  %JAN  %FEB  %MAR  %APR  %MAY  %JUN  %JUL  %AUG  %SEP
  7     %OCT  %NOV  %DEC  %JAN  %FEB  %MAR  %APR  %MAY  %JUN  %JUL  %AUG  %SEP
  8     %OCT  %NOV  %DEC  %JAN  %FEB  %MAR  %APR  %MAY  %JUN  %JUL  %AUG  %SEP
  9     %OCT  %NOV  %DEC  %JAN  %FEB  %MAR  %APR  %MAY  %JUN  %JUL  %AUG  %SEP
 10     %OCT  %NOV  %DEC  %JAN  %FEB  %MAR  %APR  %MAY  %JUN  %JUL  %AUG  %SEP

 11     %OCT  %NOV  %DEC  %JAN  %FEB  %MAR  %APR  %MAY  %JUN  %JUL  %AUG  %SEP
 12     %OCT  %NOV  %DEC  %JAN  %FEB  %MAR  %APR  %MAY  %JUN  %JUL  %AUG  %SEP
 13     %OCT  %NOV  %DEC  %JAN  %FEB  %MAR  %APR  %MAY  %JUN  %JUL  %AUG  %SEP
 14     %OCT  %NOV  %DEC  %JAN  %FEB  %MAR  %APR  %MAY  %JUN  %JUL  %AUG  %SEP
 15     %OCT  %NOV  %DEC  %JAN  %FEB  %MAR  %APR  %MAY  %JUN  %JUL  %AUG  %SEP

 16     %OCT  %NOV  %DEC  %JAN  %FEB  %MAR  %APR  %MAY  %JUN  %JUL  %AUG  %SEP
 17     %OCT  %NOV  %DEC  %JAN  %FEB  %MAR  %APR  %MAY  %JUN  %JUL  %AUG  %SEP
 18     %OCT  %NOV  %DEC  %JAN  %FEB  %MAR  %APR  %MAY  %JUN  %JUL  %AUG  %SEP
 19     %OCT  %NOV  %DEC  %JAN  %FEB  %MAR  %APR  %MAY  %JUN  %JUL  %AUG  %SEP
 20     %OCT  %NOV  %DEC  %JAN  %FEB  %MAR  %APR  %MAY  %JUN  %JUL  %AUG  %SEP

 21     %OCT  %NOV  %DEC  %JAN  %FEB  %MAR  %APR  %MAY  %JUN  %JUL  %AUG  %SEP
 22     %OCT  %NOV  %DEC  %JAN  %FEB  %MAR  %APR  %MAY  %JUN  %JUL  %AUG  %SEP
 23     %OCT  %NOV  %DEC  %JAN  %FEB  %MAR  %APR  %MAY  %JUN  %JUL  %AUG  %SEP
 24     %OCT  %NOV  %DEC  %JAN  %FEB  %MAR  %APR  %MAY  %JUN  %JUL  %AUG  %SEP
 25     %OCT  %NOV  %DEC  %JAN  %FEB  %MAR  %APR  %MAY  %JUN  %JUL  %AUG  %SEP

 26     %OCT  %NOV  %DEC  %JAN  %FEB  %MAR  %APR  %MAY  %JUN  %JUL  %AUG  %SEP
 27     %OCT  %NOV  %DEC  %JAN  %FEB  %MAR  %APR  %MAY  %JUN  %JUL  %AUG  %SEP
 28     %OCT  %NOV  %DEC  %JAN  %FEB  %MAR  %APR  %MAY  %JUN  %JUL  %AUG  %SEP
 29     %OCT  %NOV  %DEC  %JAN  %FEB  %MAR  %APR  %MAY  %JUN  %JUL  %AUG  %SEP
 30     %OCT  %NOV  %DEC  %JAN   ---  %MAR  %APR  %MAY  %JUN  %JUL  %AUG  %SEP
 31     %OCT   ---  %DEC  %JAN   ---  %MAR   ---  %MAY   ---  %JUL  %AUG   ---

AVG     %A10  %A11  %A12  %A01  %A02  %A03  %A04  %A05  %A06  %A07  %A08  %A09
MAX     %X10  %X11  %X12  %X01  %X02  %X03  %X04  %X05  %X06  %X07  %X08  %X09
MIN     %N10  %N11  %N12  %N01  %N02  %N03  %N04  %N05  %N06  %N07  %N08  %N09

#ENDFORM
```

Figure 3. Example Input for Tabulating Gage Records

```
#DEF

# Set %DATE to 30SEPyear using year from %BASDATE

%Y = YEAR(%BASDATE)
%DATE = DMY2DATE (30, 9, %Y)
%YEAR = YEAR(%DATE)    PICTURE = NNNN

%FLOW     FILE=DSSDLY  TYPE=DSS
  A=LACKAWAXEN B=HAWLY C=FLOW-NAT E=1DAY F=CMP
  ETIME=%DATE  DURATION=1YEAR

%JAN = %FLOW(JAN) PICTURE = NNNN DIGITS = 3 MISSTR = " ---"
%FEB = %FLOW(FEB)  UNDEF = " ---"
%MAR = %FLOW(MAR)
%APR = %FLOW(APR)
%MAY = %FLOW(MAY)
%JUN = %FLOW(JUN)
%JUL = %FLOW(JUL)
%AUG = %FLOW(AUG)
%SEP = %FLOW(SEP)
%OCT = %FLOW(OCT)
%NOV = %FLOW(NOV)
%DEC = %FLOW(DEC)

%A01 = AVERAGE("MISS",%JAN)
%A02 = AVERAGE("MISS",%FEB)
%A03 = AVERAGE("MISS",%MAR)
%A04 = AVERAGE("MISS",%APR)
%A05 = AVERAGE("MISS",%MAY)
%A06 = AVERAGE("MISS",%JUN)
%A07 = AVERAGE("MISS",%JUL)
%A08 = AVERAGE("MISS",%AUG)
%A09 = AVERAGE("MISS",%SEP)
%A10 = AVERAGE("MISS",%OCT)
%A11 = AVERAGE("MISS",%NOV)
%A12 = AVERAGE("MISS",%DEC)

%X01 = MAX("MISS",%JAN)
%X02 = MAX("MISS",%FEB)
%X03 = MAX("MISS",%MAR)
%X04 = MAX("MISS",%APR)

%X05 = MAX("MISS",%MAY)
%X06 = MAX("MISS",%JUN)
%X07 = MAX("MISS",%JUL)
%X08 = MAX("MISS",%AUG)
%X09 = MAX("MISS",%SEP)
%X10 = MAX("MISS",%OCT)
%X11 = MAX("MISS",%NOV)
%X12 = MAX("MISS",%DEC)

%N01 = MIN("MISS",%JAN)
%N02 = MIN("MISS",%FEB)
%N03 = MIN("MISS",%MAR)
%N04 = MIN("MISS",%APR)
%N05 = MIN("MISS",%MAY)
%N06 = MIN("MISS",%JUN)
%N07 = MIN("MISS",%JUL)
%N08 = MIN("MISS",%AUG)
%N09 = MIN("MISS",%SEP)
%N10 = MIN("MISS",%OCT)
%N11 = MIN("MISS",%NOV)
%N12 = MIN("MISS",%DEC)
#ENDDEF
```

Figure 3 Example Input for Tabulating Gage Records (cont.)

NATURAL FLOWS AT HAWLY
1943 WATER YEAR

	OCT	NOV	DEC	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP
1	634	620	682	2840	29	728	742	816	61	7	97	59
2	514	565	3360	154	280	576	69	55	719	71	75	57
3	428	64	190	1200	261	47	675	589	565	80	64	50
4	37	621	1060	952	260	415	525	577	457	78	56	50
5	378	498	842	712	271	380	553	487	377	97	55	55
6	1360	452	705	661	287	370	529	419	310	128	55	55
7	767	420	589	536	331	400	409	402	347	90	49	54
8	535	389	503	502	318	360	442	397	530	73	47	55
9	438	378	457	438	300	330	424	433	385	65	49	5
10	381	369	410	433	291	349	385	640	366	128	4	50
11	343	442	389	415	30	510	343	1070	34	15	49	45
12	304	385	381	37	360	2740	32	164	294	118	46	45
13	277	35	36	392	341	225	366	1830	258	89	44	45
14	26	335	318	385	330	1220	498	1040	240	163	50	42
15	261	290	357	335	311	1310	438	750	223	124	49	42
16	268	290	321	318	300	2060	397	621	201	87	44	43
17	449	284	307	365	290	4100	434	577	181	71	40	45
18	1040	402	300	425	300	3460	750	712	174	62	42	45
19	648	521	290	629	310	2540	1010	1180	159	56	41	4
20	487	397	281	743	350	3990	2290	1700	144	51	4	42
21	462	369	270	720	150	2480	1790	2650	15	11	41	40
22	655	345	301	64	3110	1480	161	358	126	178	40	38
23	577	31	29	570	3890	104	1130	1840	116	247	40	37
24	48	381	281	438	3620	842	878	1200	105	135	39	35
25	415	3110	270	334	3240	793	720	913	97	97	40	34
26	1510	3470	270	343	1600	758	675	1740	90	80	38	32
27	3690	1610	311	321	1160	758	559	1660	90	72	55	30
28	1600	1070	753	289	888	661	577	1070	82	63	202	29
29	1070	816	1860	309	---	530	530	869	82	70	124	2
30	800	800	6800	300	---	541	565	675	82	258	7	28
31	690	---	7140	290	---	962	---	571	---	15	65	---
AVG	680	659	970	531	829	1160	630	903	230	93	53	40
MAX	3690	3470	7140	2840	3890	4100	2290	2650	719	258	202	59
MIN	26	31	29	37	29	47	32	55	15	7	4	2

Figure 3 Example Output Gage Record

Chapter 6
Obsolete Features

This chapter lists features which were used in previous versions of REPGEN. Program changes attempted to minimize impact on current report forms. Most of these features continue to work. But they should not be used in new reports.

* Time window for reading data from DSS is specified using time expressions rather than specifying date and time-of-day separately. SDATE, EDATE, and DATE are no longer used.

SDATE = <date>

Sets start date for a time series to be read from DSS. <date> is a 9-character military date, e.g., 21SEP1987.

EDATE = <date>

Sets end date for a time series to be read from DSS. <date> is a 9-character military date, e.g., 21SEP1987.

DATE = <date>

Sets date for locating a data value in a DSS time series. <date> is 9-character military date, e.g. 03MAR1985.

REPGEN gets the last entry at or before this date. The actual date of an entry can be obtained with the function DATADATE.

* Obsolete Functions:

CLOCK - Use TIME function

DATADATE - Use DATATIME function

DATEWHEN - Use TIMEWHEN function

* Subscripts FIRST and START should not be used with time series variables. Use the number 1 instead.

* Using PICTURE characters A and N for dates and times:

Picture	Display
AABAAABAAAA	04 JUL 1987
NN/NN	7/4
NN/NN/NN	7/4/87
ZN/ZN/NN	07/04/87
NNBAAABNN	4 JUL 87
ZNBAAABNN	04 JUL 87
NNBAAABNNNN	4 JUL 1987
ZNBAAABNNNN	04 JUL 1987
AAABNN,BNNNN	JUL 4, 1987
AAAA	0830
AA:AA	08:30
ZZNN	0830
ZZ:NN	08:30
NN:NN	8:30

DSSTS

Hydrologic Engineering Center
Data Storage System
Regular Interval Time-Series
Data Entry Program

User's Manual

Version 2.8
March 1995

Hydrologic Engineering Center
U. S. Army Corps of Engineers
609 Second Street
Davis, California 95616-4687
(916) 756-1104

DSSTS

1. Introduction

DSSTS is a program for entering regular interval time series data into a HEC-DSS data base file. Regular interval time series data has an implicit date/time associated with each data value, whereas irregular interval time series data uses an explicit date/time for data.

DSSTS is a prompt driven program that requests information from the user. It may be run interactively (input from the keyboard), or in a batch mode with input from a file. To execute DSSTS in a batch mode, the input that would normally be typed interactively are placed into a file, then the program is executed with that file specified as input (e.g., "`DSSTS input=myfile`").

If desired, all information entered at the keyboard can be copied into a "log file" by specifying the log file name on the command line ("`DSSTS logfile=mylog`"). If an abort or some other error should occur, DSSTS may be rerun using the logfile as the input (e.g., "`DSSTS input=mylog`").

If an illegal value is entered during an interactive execution (e.g., a letter instead of a number), the program will re-request the last piece of data. If an illegal value is entered during a batch execution, the program will terminate.

2. Use

2.1 The program is initiated by entering its name (and the directory of where the program is located, if needed):

```
DSSTS
```

2.2 Optional parameters that may be specified on the execution line are:

Name	Default	Description
`INPUT`	standard in	Command input file
`OUTPUT`	standard out	Output file
`DSSFILE`	none	DSS file
`LOGFILE`	`SCRATCH.002`	Copy of input commands

The execution line parameters may be abbreviated to 2 characters (`INPUT` can be `IN`).

If a command input file is specified on the execution line, it should contain DSSTS input as if it were being entered at the keyboard (NOT just data). If a DSS file name is provided on the execution line, the program will not ask for it.

3. Command Input

3.1 DSSTS prompts with "ENTER DSS FILE NAME", whereby the user enters the name of the DSS file to use. If the file does not exist, it will be created.

3.2 "ENTER PATHNAME, OR PATHNAME PART(S), OR FINISH".
The full six part pathname, including slashes (/), may be given, or individual pathname parts may be specified. To enter individual pathname parts, type the part letter (A, B, C, D, E, or F) followed by an equal sign "=" then the part. One to six parts may be entered, separated by a comma or a blank space. If a pathname had been given earlier, then those parts not specified will remain the same as in the earlier pathname. The pathname must follow the regular interval time-series conventions specified in the HEC-DSS Overview, Appendix A. Upon the completion of entering all data, typing "FINISH" at this point will terminate the program.

3.3 "ENTER UNITS OF DATA (E.G. CFS, FEET)".
The units of the data may be specified with up to eight characters.

3.4 "ENTER DATA TYPE (E.G. PER-AVER, INST-VAL)".
One of the following four data types must be given:

```
PER-AVER
PER-CUM
INST-VAL
INST-CUM
```

3.5 "ENTER THE DATE AND TIME FOR THE FIRST DATA VALUE".
The date and time for the first data value are to be provided in a military style format (e.g., 05FEB74). The time follows the date, separated by a space or comma, and is given in 24 hour clock time (e.g., 0830 for 8:30 a.m.).

3.6. "ENTER DATA VALUES.
ENTER END AT THE BEGINNING OF LINE WHEN DONE"
The data is to be entered at this point, corresponding to the date and time given on the prompt. One or more data values may be provided on each line. The data may be given in an integer or real format, but not in scientific notation. A missing data value may be specified by entering the letter "M", or the value -901. The data may cross the pathname date boundary specified by the "D" part of the pathname. If an illegal data value is entered, the program will request that value (and all following values) again.

When the entry of data for this pathname has been completed, type "END" to store the data in

the DSS data file. After this, the program will return to step 2, where a new pathname may be specified, or the program may be terminated by entering "FINISH".

4. Example

```
dssts
ENTER DSS FILE NAME
FILE = datab
                    -----DSS---ZOPEN EXISTING FILE OPENED   71  datab.dss

ENTER PATHNAME, OR PATHNAME PART(S), OR FINISH
I>/SCIOTO/WALDO/FLOW/01JAN1984/1DAY/OBS/
/SCIOTO/WALDO/FLOW/01JAN1984/1DAY/OBS/
ENTER UNITS OF DATA (E.G. CFS, FEET)
I>CFS
ENTER DATA TYPE (E.G. PER-AVER, INST-VAL)
I>INST-VAL
ENTER THE DATE AND TIME FOR THE FIRST DATA VALUE
I>28DEC84, 1320
Enter data values.
Enter END at the beginning of the line when done.
28DEC84, 1320 >932 M 940.5 945
01JAN85, 1320 >950
02JAN85, 1320 >955,956, 958 949 M M M
09JAN85, 1320 >930 928.6 927.4
12JAN85, 1320 >END
-----DSS---ZWRITE FILE  71, VERS.  1  /SCIOTO/WALDO/FLOW/01JAN1984/1DAY/OBS/
-----DSS---ZWRITE FILE  71, VERS.  1  /SCIOTO/WALDO/FLOW/01JAN1985/1DAY/OBS/

ENTER PATHNAME, OR PATHNAME PART(S), OR FINISH
I>B=DUBLIN, C=STAGE
/SCIOTO/DUBLIN/STAGE/01JAN1984/1DAY/OBS/
ENTER UNITS OF DATA (E.G. CFS, FEET)
I>FEET
ENTER DATA TYPE (E.G. PER-AVER, INST-VAL)
I>INST-VAL
ENTER THE DATE AND TIME FOR THE FIRST DATA VALUE
I>02JAN85, 0830
Enter data values.
Enter END at the beginning of the line when done.
02JAN85, 0830 >6.54 6.56 6.61 M M 6.66
08JAN85, 0830 >6.7
09JAN85, 0830 >6.72 6.74 M 6.80
13JAN85, 0830 >M
14JAN85, 0830 >6.84
15JAN85, 0830 >END
-----DSS---ZWRITE FILE 71, VERS.  1  /SCIOTO/DUBLIN/STAGE/01JAN1985/1DAY/OBS/

ENTER PATHNAME, OR PATHNAME PART(S), OR FINISH
I>FINISH

                    -----DSS---ZCLOSE FILE   71
                              NO. RECORDS=      11
                              FILE SIZE=     15819 WORDS,    142 SECTORS
                              PERCENT INACTIVE=  0.00
```

DSSITS

Hydrologic Engineering Center
Data Storage System
Irregular Interval Time-Series
Data Entry Program

User's Manual

Version 3.6
March 1995

Hydrologic Engineering Center
U.S. Army Corps of Engineers
609 Second Street
Davis, California 95616-4687
(916) 756-1104

DSSITS

1. Introduction

DSSITS is a program for entering irregular interval time series data into a HEC-DSS data base file. Irregular interval time series data has a explicit date/time associated with each data value, whereas regular interval time series data uses an implicit date/time for data.

DSSITS is a prompt driven program that requests information from the user. It may be run interactively (input from the keyboard), or in a batch mode with input from a file. To execute DSSITS in a batch mode, the input that would normally be typed interactively are placed into a file, then the program is executed with that file specified as input (e.g., "`dssits input=myfile`").

If desired, all information entered at the keyboard can be copied into a "log file" by specifying the log file name on the command line ("`dssits logfile=mylog`"). If an abort or some other error should occur, DSSITS may be rerun using the logfile as the input (e.g., "`dssits input=mylog`").

Irregular interval time series data must be entered in a sequential order. If an illegal value is entered during an interactive execution (e.g., a letter instead of a number), the program will re-request the last piece of data. If an illegal value is entered during a batch execution, the program will terminate.

2. Use

2.1 The program is initiated by entering its name (and the directory of where the program is located, if needed):

```
dssits
```

2.2 Optional parameters that may be specified on the execution line are:

Name	Default	Description
INPUT	standard in	Command input file
OUTPUT	standard out	Output file
DSSFILE	none	DSS file
LOGFILE	SCRATCH.002	Copy of input commands

The execution line parameters may be abbreviated to 2 characters (`INPUT` can be `IN`).

If a command input file is specified on the execution line, it should contain DSSITS input as if it were being entered at the keyboard (NOT just data). If a DSS file name is provided on the execution line, the program will not ask for it.

3. Command Input

3.1 DSSITS prompts with "ENTER DSS FILE NAME", whereby the user enters the name of the DSS file to use. If the file does not exist, it will be created.

3.2 "ENTER PATHNAME, OR PATHNAME PART(S), OR FINISH".
The full six part pathname, including slashes (/), may be given, or individual pathname parts may be specified. To enter individual pathname parts, type the part letter (A, B, C, D, E, or F) followed by an equal sign "=" then the part. One to six parts may be entered, separated by a comma or a blank space. If a pathname had been given earlier, then those parts not specified will remain the same as in the earlier pathname. The pathname must follow the irregular interval time-series conventions specified in the HEC-DSS Overview, Appendix A. Upon the completion of entering all data, typing "FINISH" at this point will terminate the program.

3.3 "ENTER UNITS OF DATA (E.G. CFS, FEET)".
The units of the data may be specified with up to eight characters.

3.4 "ENTER DATA TYPE (E.G. PER-AVER, INST-VAL)".
One of the following four data types must be given:

```
PER-AVER
PER-CUM
INST-VAL
INST-CUM
```

3.5 "ENTER DATE, TIME, AND VALUE (FREE FORMAT)
ENTER END AT THE BEGINNING OF THE LINE WHEN DONE".
One data value, along with a date and time, is to be given on each line. The date must be seven or nine characters long, in a day, month, year order (e.g. 05FEB74). The time follows the date, separated by a space or comma, and is given in 24 hour clock time (e.g. 0830 for 8:30 a.m.). The data value follows the time, separated by a blank space or comma, and may be given in an integer or real format (not scientific notation). The data may cross the pathname date boundary specified by the "D" part of the pathname. If an illegal data value or date and time is entered, the program will request that value again.

When the entry of data for this pathname has been completed, type "END" to store the data in the DSS data file. After this, the program will return to step 2, where a new pathname may be specified, or the program may be terminated by entering "FINISH".

4. Example

```
dssits
ENTER DSS FILE NAME
FILE = datab
         -----DSS---ZOPEN EXISTING FILE OPENED   71  datab.dss

ENTER PATHNAME, OR PATHNAME PART(S), OR FINISH
I>/SCIOTO/WALDO/FLOW/01JAN1984/IR-YEAR/OBS/
/SCIOTO/WALDO/FLOW/01JAN1984/IR-YEAR/OBS/
ENTER UNITS OF DATA (E.G. CFS, FEET)
I>CFS
ENTER DATA TYPE (E.G. PER-AVER, INST-VAL)
I>INST-VAL
ENTER DATE, TIME, AND VALUE (FREE FORMAT)
ENTER END AT THE BEGINNING OF THE LINE WHEN DONE
I>28DEC84, 1220, 932
I>30DEC84, 1510, 935.4
I>01JAN85, 0830, 938.8
I>02JAN85, 1700, 940
I>END
---DSS--ZWRITE FILE  71, VERS. 1 /SCIOTO/WALDO/FLOW/01JAN1984/IR-YEAR/OBS/
---DSS--ZWRITE FILE  71, VERS. 1  /SCIOTO/WALDO/FLOW/01JAN1985/IR-YEAR/OBS/

ENTER PATHNAME, OR PATHNAME PART(S), OR FINISH
I>B=DUBLIN, C=STAGE
/SCIOTO/DUBLIN/STAGE/01JAN1984/IR-YEAR/OBS/
ENTER UNITS OF DATA (E.G. CFS, FEET)
I>FEET
ENTER DATA TYPE (E.G. PER-AVER, INST-VAL)
I>INST-VAL
ENTER DATE, TIME, AND VALUE (FREE FORMAT)
ENTER END AT THE BEGINNING OF THE LINE WHEN DONE
I>29DEC84 1310 14.4
I>30DEC84 1510 14.5
I>01JAN85 0830 14.5
I>02JAN85 1700 14.8
I>END
---DSS--ZWRITE FILE  71, VERS. 1  /SCIOTO/DUBLIN/STAGE/01JAN1984/IR-YEAR/OBS/
---DSS--ZWRITE FILE  71, VERS. 1  /SCIOTO/DUBLIN/STAGE/01JAN1985/IR-YEAR/OBS/

ENTER PATHNAME, OR PATHNAME PART(S), OR FINISH
I>FINISH

             -----DSS---ZCLOSE FILE   71
                        NO. RECORDS=       7
                        FILE SIZE=    2875 WORDS,   26 SECTORS
                        PERCENT INACTIVE=  0.00
```

DSSPD

Hydrologic Engineering Center
Data Storage System
Paired Data Entry Program

User's Manual

Version 3.4
March 1995

Hydrologic Engineering Center
Water Resources Support Center
U.S. Army Corps of Engineers
609 Second Street
Davis, California 95616-4687
(916) 756-1104

DSSPD

1. Introduction

DSSPD is a program for entering paired data into a HEC-DSS data base file. Paired data is a group of data that represents a two variable relationship. An example is data that makes up a curve, such as a rating table or a flow-frequency curve. Several curves may be stored in the same record if one of the variables is the same. For example, several frequency-damage curves may be stored in the same record, where the curves may be residential, commercial, etc. A scale associated with the variable may be one of three types: linear, logarithmic, or probability.

DSSPD is a prompt driven program that requests information from the user. It may be run interactively (input from the keyboard), or in a batch mode with input from a file. To execute DSSPD in a batch mode, the input that would normally be typed interactively are placed into a file, then the program is executed with that file specified as input (e.g., "`dsspd input=myfile`").

If desired, all information entered at the keyboard can be copied into a "log file" by specifying the log file name on the command line ("`dsspd logfile=mylog`"). If an abort or some other error should occur, DSSPD may be rerun using the logfile as the input (e.g., "`dsspd input=mylog`").

Rating table data may be entered with the "r" option. This allows the entry of additional information that is used by a few programs (such as DSSMATH). This information consists of an offset, shift, and datum, plus whether the transformation is linear or log-log. To initiate the "r" option, enter "`dsspd -r`" when executing dsspd.

2. Use

2.1 The program is initiated by entering its name (and the directory of where the program is located, if needed):

```
dsspd
```

If rating table information is to be entered, use the "r" option"

```
dsspd -r
```

2.2 Optional parameters that may be specified on the execution line are:

Name	Default	Description
INPUT	standard in	Command input file
OUTPUT	standard out	Output file
DSSFILE	none	DSS file
LOGFILE	SCRATCH.002	Copy of input commands

The execution line parameters may be abbreviated to 2 characters (INPUT can be IN).

If a command input file is specified on the execution line, it should contain DSSPD input as if it were being entered at the keyboard (NOT just paired data). If a DSS file name is provided on the execution line, the program will not ask for it.

3. Command Input

3.1 DSSPD prompts with "Enter DSS File Name", whereby the user enters the name of the DSS file to use. If the file does not exist, it will be created.

3.2 "Enter Pathname, or Pathname Part(s), or FINISH".
The full six part pathname, including slashes (/), may be given, or individual pathname parts may be specified. To enter individual pathname parts, type the part letter (A, B, C, D, E, or F) followed by an equal sign "=" then the part. One to six parts may be entered, separated by a comma or a blank space. If a pathname had been given earlier, then those parts not specified will remain the same as in the earlier pathname. Upon the completion of entering all data, typing "FINISH" at this point will terminate the program.

3.3 "Enter the number of curves"
Enter the number of curves this pathname record will contain (typically 1). Some data, such as frequency-damage curves, may have more than one curve stored in the same record.

3.4 "Enter the units of the (independent axis) data (e.g., CFS, FEET)"
Enter the units for the first portion of the "C" part of the pathname. For example, if the "C" part is "STAGE-FLOW", the units may be "FEET". The units may be from zero to eight characters long.

3.5 "Enter the data type for the (independent axis) data (e.g., UNT, LOG)"
Enter the data type for this axis. Typically "UNT" for unitary or "LOG" for logarithmic. The data type may be from zero to four characters long.

3.6 "Enter the units of the (dependent axis) data"
Enter the units for the second portion of the "C" part of the pathname. For example, if the "C" part is STAGE-FLOW, the units would be "CFS". The units may be from zero to eight characters long.

3.7 "Enter the data type for the (dependent axis) data"
Enter the data type for this axis (i.e., the second portion of the "C" part of the pathname). The data type may be from zero to four characters long. If more than one curve is to be entered under this pathname, the units and type will be repeated for each curve.

3.8 "Enter the Offset, Shift, and Datum (or blank for all zeros)"
(Asked when the rating table option is used.) If the rating table has an offset, shift or datum associated with it, enter that information here separated by commas or blanks. A blank line will cause zeros to be entered for these items.

3.9 "Enter the Transformation Type (LINLIN or LOGLOG)"
(Asked when the rating table option is used.) If the rating table is logarithmic, enter "LOGLOG". If it is linear enter "LINLIN".

3.10 "For DSPLAY plots, do you want the (independent axis) data
to be on the X (horizontal) axis, or on the Y (vertical) axis?"
For the DSS graphics program "DSPLAY", if the first portion of the "C" part of the pathname is to be on the horizontal axis, enter a "Y". If the second portion of the "C" part of the pathname is to be on the horizontal axis, enter a "X".

3.11 "Enter a label for the first dependent curve, or
all labels for all curves, or blank to store no labels"
A twelve character label may be specified for providing additional information in the legend in DSPLAY plots. Such a label for a stage-damage curve may be the type of damage, e.g., COMMERCIAL for commercial damage. If no label is desired, just press the carriage return.

3.12 "Enter data in pairs (i.e., X, Y)
Enter END at the beginning of the line when done."
Enter the actual data pairs in sequential order in a free format mode. The data is to be entered with one pair (or set for multiple curves) per line, separated by a comma or a blank. It may be given in an integer or real format, but not in scientific notation. If there is more than one curve, enter the additional Y values following the primary Y value.

When the entry of data for this pathname has been completed, type "END" to store the data in the HEC-DSS data file. After this, the program will return to step 2, where a new pathname may be specified, or the program may be terminated by entering "FINISH".

Example 1

Store a stage-damage curve for the reach named "GLXR7" in the Kanawha river basin. This data was developed on January 5, 1993.

```
>dsspd

     DSSPD:  3.4.0 ;  March, 1995

Enter DSS File Name
File = db
   -----DSS---ZOPEN:  New File Opened,  File: db.dss
                      Unit:  71;  DSS Version: 6-JB

Enter Pathname, or Pathname Part(s), or FINISH
I>/KANAWHA/GLXR7/STAGE-DAMAGE//05JAN93//
/KANAWHA/GLXR7/STAGE-DAMAGE//05JAN93//
Enter the number of DAMAGE curves
Number of Curves: 2
Enter the units of the STAGE data (e.g., CFS, FEET)
Units: FEET
Enter the data type for the STAGE data (e.g., UNT, LOG)
Type: UNT
Enter the units of the DAMAGE data
Units: $1000
Enter the data type for the DAMAGE data
Type: UNT
For DSPLAY plots, do you want the STAGE data
to be on the X (horizontal) axis, or on the Y (vertical) axis?
Enter X or Y:  Y
Enter a label for the first DAMAGE curve, or
all labels for all curves, or blank to store no labels
Label: RESIDENTIAL
Enter a label for DAMAGE curve  2
Label: COMMERCIAL
Enter data in pairs (i.e., STAGE, DAMAGE1, DAMAGE2)
Enter END at the beginning of the line when done.
I>3., 0, 0
I>5.0, 0, 0
I>6.5, .5, 2.
I>7.5, 1., 3.
I>9.0, 12., 5.
I>10.5, 18., 5.
I>12.0, 45., 5.
I>END
-----DSS---ZWRITE:  /KANAWHA/GLXR7/STAGE-DAMAGE//05JAN93//

 Enter Pathname, or Pathname Part(s), or FINISH
I>FINISH

  -----DSS---ZCLOSE Unit:  71,   File: db.dss
             Pointer Utilization:  0.25
             Number of Records:      1
             File Size:     11.5  Kbytes
             Percent Inactive:   0.0
```

Example 2

Store the rating curve for the reach named "GLXR7" in the Kanawha river basin. This data comes from the USGS and was computed on April 2, 1982.

```
>dsspd -r

    DSSPD:  3.4.0 ;  March, 1995
    Rating Table Entry Version.

Enter DSS File Name
File = db
    -----DSS---ZOPEN:  Existing File Opened,  File: db.dss
                       Unit:  71;  DSS Version: 6-JB

Enter Pathname, or Pathname Part(s), or FINISH
I>/KANAWHA/GLXR7/STAGE-FLOW//02APR82/USGS 10/
/KANAWHA/GLXR7/STAGE-FLOW//02APR82/USGS 10/
Enter the number of FLOW curves
Number of Curves: 1
Enter the units of the STAGE data (e.g., CFS, FEET)
Units: FEET
Enter the data type for the STAGE data (e.g., UNT, LOG)
Type: UNT
Enter the units of the FLOW data
Units: CFS
Enter the data type for the FLOW data
Type: UNT
For DSPLAY plots, do you want the STAGE data
to be on the X (horizontal) axis, or on the Y (vertical) axis?
Enter X or Y: Y
Enter the Offset, Shift, and Datum (or blank for all zeros)
I>0, 2.4, 937
Enter the Transformation Type (e.g., LINLIN or LOGLOG)
I>LINLIN
Enter a label for the FLOW curve (or blank to store no label)
Label:
Enter data in pairs (i.e., STAGE, FLOW)
Enter END at the beginning of the line when done.
I>0.6, 250
I>1.,0 85
I>1.,7 300
I>3.0, 570
I>5.0, 930
I>8.,0 1350
I>END
-----DSS---ZWRITE: /KANAWHA/GLXR7/STAGE-FLOW//02APR82/USGS 10/

Enter Pathname, or Pathname Part(s), or FINISH
I>FIN

   -----DSS---ZCLOSE Unit:  71,   File: db.dss
              Pointer Utilization:  0.25
              Number of Records:      2
              File Size:     11.8  Kbytes
              Percent Inactive:   0.0
```

DSSTXT

Hydrologic Engineering Center
Data Storage System
Text Data Entry Program

User's Manual

Version 1.2
March 1995

Hydrologic Engineering Center
U.S. Army Corps of Engineers
609 Second Street
Davis, California 95616-4687
(916) 756-1104

DSSTXT

1. Introduction

DSSTXT is a program for storing and retrieving text data from a DSS file. Text data is defined as generic alpha-numeric lines of text, where each line is preceded by a line feed character and ends with a carriage return character. It does not, at this time, include other types characters, such as those that would be used to create a graphical display.

DSSTXT is a prompt driven program that requests information from the user. It may be run interactively (i.e., from the keyboard), or in a batch mode with input commands from a file. Text to be stored may be in a file (regardless whether the program is run interactively or batch) or it can be entered from the keyboard. Text that is retrieved from a DSS file can be placed in a file or displayed on the screen.

There are no definitive size limitations for the number of lines in a DSS text record, but it is recommended that a record contain no more than about 200 lines. The maximum possible length of a line in a DSS text record is 160 characters. A reasonable maximum length is 132 characters. The maximum number of bytes that can be stored in a text record (including begin and end of line characters) is 9600 on DOS machines, and 16,000 on most other machines. The number of lines that can be stored is dependent on the average length of the lines. For example, if the average length is about 40 characters, the maximum number of lines would be about 200 for DOS and about 350 for other computers. Generally, an appropriate number of lines to store in a single record is from 2 to 150 lines. If one desired to store a large amount of text, for example an entire book, the text could be divided into sections consisting of 1 to 3 pages.

There are no conventions set for the structure of a text record's pathname. However, it is recommended that the pathname parts be labeled in a descending order of importance, and that the pathname imply that the record contains text data and not one of the other types of data.

All information entered at the keyboard is copied into a "log file" (scratch file "`SCRATCH.002`" on personal computers). If an abort or some other error should occur, DSSTXT may be rerun using the log file as the input (e.g., `DSSTXT INPUT=SCRATCH.002`).

2. Use

2.1 To store text data into a DSS file, the program is initiated by entering its name (and the directory or qualifier of where the program is located):

```
DSSTXT
```

To retrieve text data from a DSS file, the "DIR=RETRIEVE" parameter must be given on the execution line:

```
DSSTXT DIR=RETRIEVE
```

Text data may not be both retrieved and stored in a single execution.

2.2 Optional parameters that may be specified on the execution line are:

Name	Default	Description
INPUT	standard in	Command input file
OUTPUT	standard out	Output file
DSSFILE	none	DSS file
DIRECTION	STORE	STORE or RETRIEVE text records
TEXTFILE	none	Text input or output file
LOG	SCRATCH.002	Copy of input commands
FUNFILE	GENFUN	PREAD function file
MACFILE	GENMAC	PREAD macro file
SCNFILE	GENSCN	PREAD screen file

The execution line parameters may be abbreviated to 2 characters (INPUT can be IN).

If a command input file is specified on the execution line, it should contain DSSTXT input as if it were being entered at the keyboard (NOT just text data). If the text file name is given on the execution line, text will be placed in (or read from) that file. The text file name will not be asked for in the input. If a DSS file name is provided on the execution line, the program will not ask for it.

3. Command Input

3.1 DSSTXT prompts with "Enter DSS File Name", whereby the user enters the name of the DSS file to use. If the file does not exist, it will be created.

3.2 The next prompt is "Enter Pathname, or Pathname Part(s), or FINISH". The full six part pathname, including slashes (/), may be given, or individual pathname parts may be specified. To enter individual pathname parts, type the part letter (A, B, C, D, E, or F) followed by an equal sign "=" then the part. One to six parts may be entered, separated by a comma or a blank space. If a pathname had been given earlier, then those parts not specified will remain the same as in the earlier pathname. The program may be terminated at this prompt by entering "FINISH".

3.3 If text data is being retrieved, the following prompt is given:

```
"Enter the name of the file to write to, or a "*" to write to your screen"
```

At this point, the user may either enter a asterisk (*) to cause the text to be displayed on the screen, or the name of an ASCII file to put it in. If the file does not exist, it will be created by DSSTXT.

3.4 If text data is being stored, the following prompt is given:

```
"Enter the name of the file to read from (or a "*" to read from your keyboard)"
```

The user may either enter a asterisk (*) to cause the text to be read from the keyboard (interactive use only), or a file name to read the text from. If a file name is given, the file must be an ASCII file that the user has access to. All lines in that file will be stored in a single record with the pathname specified in step 3.2. If an asterisk (*) is entered, DSSTXT will read directly from the keyboard until an end-of-file marker is entered. An end-of-file marker is either a control-D for UNIX computers, or a control-Z for other computers, entered on a separate line and followed by a carriage return. The size of the record to store should not exceed the limits discussed in the introduction.

3.5 After the text record has been stored or retrieved, the program will return to step 3.2 ("`Enter Pathname...`"), where a new pathname may be specified, or the program may be terminated by entering the word FINISH. If text is being retrieved, the same ASCII text (output) file will be used if a blank line is entered for the text file name, when prompted.

4.1 Example

Store weather forecasts received from the National Weather Service. The daily forecast is in file NWS.FOR, and the 3-4 day forecast is in file NWS.EXT.

```
>DSSTXT

   DSSTXT:  1.2.0 ;  June, 1990

Enter DSS File Name
File = WEATHER
   -----DSS---ZOPEN; Created DSS File: WEATHER
   -----DSS---ZOPEN:  New File Opened,  File: WEATHER
                      Unit:  71;  DSS Version: 6-FA

Enter Pathname, or Pathname Part(s), or FINISH
Path: /NWS/SAC/FORECAST/04JUL90///
Pathname: /NWS/SAC/FORECAST/04JUL90///
Enter the name of the file to read from (or a "*" to read from your keyboard)
File: NWS.FOR
-----DSS---ZWRITE Unit  71; Vers.    1:  /NWS/SAC/FORECAST/04JUL90///
   13 lines Stored.

Enter Pathname, or Pathname Part(s), or FINISH
Path: F=EXTENDED
Pathname: /NWS/SAC/FORECAST/04JUL90//EXTENDED/
Enter the name of the file to read from (or a "*" to read from your keyboard)
File: NWS.EXT
-----DSS---ZWRITE Unit  71; Vers.    1:  /NWS/SAC/FORECAST/04JUL90//EXTENDED/
   12 lines Stored.

Enter Pathname, or Pathname Part(s), or FINISH
Path: F=NOTES
Pathname: /NWS/SAC/FORECAST/04JUL90//NOTES/
Enter the name of the file to read from (or a "*" to read from your keyboard)
File: *
Enter Text.  When Complete enter "^Z"
>Communications lost with NWS for 4 hours beginning at 1600 hours.
>July 3rd forecast was 5 degrees lower than recorded temperature.
>^Z
-----DSS---ZWRITE Unit  71; Vers.    1:  /NWS/SAC/FORECAST/04JUL90//NOTES/
    2 lines Stored.

Enter Pathname, or Pathname Part(s), or FINISH
Path: FIN
   -----DSS---ZCLOSE Unit:  71,   File: WEATHER
              Pointer Utilization:  0.29
              Number of Records:      3
              File Size:     16.5  Kbytes
              Percent Inactive:   0.0
STOP
```

4.2 Example

Retrieve weather forecast from a DSS file. Display one of the records on the screen, and place the other in a (new) file named FOR.EXT.

```
>DSSTXT DIR=RET

   DSSTXT:  1.2.0 ;  June, 1990

Enter DSS File Name
File = WEATHER
   -----DSS---ZOPEN:  Existing File Opened,  File: WEATHER
                      Unit:  71;  DSS Version: 6-FA

Enter Pathname, or Pathname Part(s), or FINISH
Path: /NWS/SAC/FORECAST/04JUL90///
Pathname: /NWS/SAC/FORECAST/04JUL90///
Enter the name of the file to write to, or a "*" to write to your screen
File: *
-----DSS--- ZREAD Unit  71; Vers.    1:  /NWS/SAC/FORECAST/04JUL90///
LOWER SACRAMENTO VALLEY FORECAST
NATIONAL WEATHER SERVICE SACRAMENTO CA
9 AM PDT WED JUL 04 1990...DO NOT USE AFTER 3.30 PM PDT WEDNESDAY

LOWER SACRAMENTO VALLEY
.TODAY...MOSTLY SUNNY.  HIGHS MID 80S TO LOW 90S.
 SOUTH WINDS 5 TO 15 MPH WITH SOUTHWEST DELTA WINDS 15 TO 25 MPH.
.TONIGHT...FAIR.  LOWS MID 50S TO LOW 60S.
 SOUTH WINDS 5 TO 15 MPH WITH SOUTHWEST DELTA WINDS 15 TO 25 MPH.
.THURSDAY...MOSTLY SUNNY.  HIGHS MID 80S TO LOW 90S.
 DOWNTOWN SACRAMENTO  87 57 89    MARYSVILLE/YUBA CITY  88 57 90

.END
   13 lines Retrieved.

Enter Pathname, or Pathname Part(s), or FINISH
Path: F=EXTENDED
Pathname: /NWS/SAC/FORECAST/04JUL90//EXTENDED/
Enter the name of the file to write to, or a "*" to write to your screen
File: FOR.EXT
-----DSS--- ZREAD Unit  71; Vers.    1:  /NWS/SAC/FORECAST/04JUL90//EXTENDED/
   12 lines Retrieved.

Enter Pathname, or Pathname Part(s), or FINISH
Path: FINISH
   -----DSS---ZCLOSE Unit:  71,   File: WEATHER
              Pointer Utilization:  0.29
              Number of Records:      3
              File Size:     16.5  Kbytes
              Percent Inactive:   0.0
STOP
```

4.3 Example

Retrieve text from a DSS file in a "batch mode", giving the command input file, the DSS file, the text (output) file, and an output file on the execution line.

DSSTXT is executed by the following line:

```
DSSTXT DIR=RET DSS=WEATHER IN=FPATHS OUT=OUT TEXT=FOR.SUM
```

The INPUT file is a list of the pathnames to retrieve:

```
/NWS/SAC/FORECAST/04JUL90///
/NWS/SAC/FORECAST/04JUL90//EXTENDED/
/NWS/SAC/FORECAST/04JUL90//NOTES/
```

The TEXT file will contain the retrieved data:

```
LOWER SACRAMENTO VALLEY FORECAST
NATIONAL WEATHER SERVICE SACRAMENTO CA
9 AM PDT WED JUL 04 1990...DO NOT USE AFTER 3.30 PM PDT WEDNESDAY

LOWER SACRAMENTO VALLEY
.TODAY...MOSTLY SUNNY.  HIGHS MID 80S TO LOW 90S.
 SOUTH WINDS 5 TO 15 MPH WITH SOUTHWEST DELTA WINDS 15 TO 25 MPH.
.TONIGHT...FAIR.  LOWS MID 50S TO LOW 60S.
 SOUTH WINDS 5 TO 15 MPH WITH SOUTHWEST DELTA WINDS 15 TO 25 MPH.
.THURSDAY...MOSTLY SUNNY.  HIGHS MID 80S TO LOW 90S.
 DOWNTOWN SACRAMENTO  87 57 89    MARYSVILLE/YUBA CITY  88 57 90

.END
 NORTHERN CALIFORNIA EXTENDED FORECAST
NATIONAL WEATHER SERVICE SAN FRANCISCO CA
4.30 AM PDT WED JUL 4 1990

PARTLY CLOUDY AT TIMES OVER THE NORTHERN MOUNTAINS...OTHERWISE FAIR
WITH PATCHY COASTAL LOW CLOUDS AND FOG.
COAST...HIGHS 60S TO LOWER 70S. LOWS MID 40S TO MID 50S.
COASTAL VALLEYS...HIGHS 70S TO LOWER 90S. LOWS MID 40S AND 50S.
INLAND VALLEYS...HIGHS UPPER 80S TO UPPER 90S. LOWS UPPER 50S AND 60S.
MOUNTAINS...HIGHS 70S AND 80S. LOWS UPPER 30S TO MID 50S.

.END
Communications lost with NWS for 4 hours beginning at 1600 hours.
July 3rd forecast was 5 degrees lower than recorded temperature.
```

The output from DSSTXT is:

```
   DSSTXT:  1.2.0 ;  June, 1990

   -----DSS---ZOPEN:  Existing File Opened,  File: WEATHER
                      Unit:  71;  DSS Version: 6-FA
Pathname: /NWS/SAC/FORECAST/04JUL90///
-----DSS--- ZREAD Unit  71; Vers.    1:  /NWS/SAC/FORECAST/04JUL90///
   13 lines Retrieved.
Pathname: /NWS/SAC/FORECAST/04JUL90//EXTENDED/
-----DSS--- ZREAD Unit  71; Vers.    1:  /NWS/SAC/FORECAST/04JUL90//EXTENDED/
   12 lines Retrieved.
Pathname: /NWS/SAC/FORECAST/04JUL90//NOTES/
-----DSS--- ZREAD Unit  71; Vers.    1:  /NWS/SAC/FORECAST/04JUL90//NOTES/
    2 lines Retrieved.
   -----DSS---ZCLOSE Unit:  71,   File: WEATHER
              Pointer Utilization:  0.29
              Number of Records:      3
              File Size:     21.5  Kbytes
              Percent Inactive:   0.0
```

DWINDO

Hydrologic Engineering Center
Interactive Data Entry and Editing

User's Manual

Version 2.1
March 1995

Hydrologic Engineering Center
U.S. Army Corps of Engineers
609 Second Street
Davis, California 95616-4687
(916) 756-1104

DWINDO
Table of Contents

Chapter Page

1. Introduction 1

2. Use 3
   - 2.1 Program Initiation 3
   - 2.2 Global Control 3
   - 2.3 Data Entry 5
     - 2.3.1 Function Keys 5
     - 2.3.2 Commands 8
   - 2.4 Saving Data 9

3. Implementation 11
   - 3.1 Forms Description File 11
     - 3.1.1 Form Image 14
     - 3.1.2 Viewports 14
     - 3.1.3 Data Definitions 15
     - 3.1.4 DSS References 17
   - 3.2 Terminal Definition File 18

Tables

Page

1. Form Control Functions 6
2. Data Definitions 16
3. Intrinsic Variables 16
4. DSS References 17
5. Control Function Sequences 21

Figures

Page

1. Sample Form 1
2. Form Definition Example 12
3. Terminal Definition Example 19

Chapter 1

Introduction

DWINDO is a generalized program which provides interactive entry and editing of HEC Data Storage System (DSS) time-series data in a form displayed on the computer console or remote access terminal. An example is shown in Figure 1. DWINDO requires a display environment that meets the ANSI standards.

Short - Period Reservoir Computations
Friant Reservoir, San Joaquin River, California

Date	Time	Precip inches	Elevation feet	Storage ac-ft	Release cfs	Inflow cfs
12JUN86	0347	0.00	5175.00	195000	0	--
12JUN86	0416	--	5176.21	197000	0	10000
12JUN86	0648	0.00	5176.21	197500	0	16000
12JUN86	0730	0.00	--	--	--	--
12JUN86	0912	0.00	5181.40	198000	400	13000
12JUN86	1000	--	5181.40	198000	400	--
13JUN86	0710	1.42	5179.04	196400	0	--
13JUN86	1200	1.60	5179.02	--	10	800
14JUN86	0640	1.53	5178.88	196320	0	500
14JUN86	2114	0.00	--	--	0	500

Figure 1. Sample Form

DWINDO presents time series data in columns on the form. All columns are of the same length, and each row in a column corresponds to a time for which there is at least one data value among all the columns. Since only a few data observations can be shown in the columns at one time, the columns may be scrolled (moved up or down one row at a time) or paged (moved up or down by a full number of form rows). When scrolling or paging occur, the data in all columns move together.

Data entry and editing in DWINDO is controlled mainly through the use of specially designated function keys. Each key invokes a particular function, such as moving the cursor among the rows and columns.

Cursor movement allows the user to move both horizontally and vertically from one cell to the next on the displayed form. New values are simply input from the keyboard (preferably, the numeric keypad). Existing values may be edited in the cell by a combination of overtyping, insertion and deletion, or may be erased and a substitute entered.

The forms used to present data are designed by the user in order to permit organizations of data which are appropriate to the user's needs. Descriptions of the forms are contained in a reference file. When a particular form is used, the form is filled with data in accordance with a user-defined time window.

One typical application of DWINDO would be entry of real-time operational data, either by office personnel entering data for several field projects on a local computer, or by field personnel entering data pertaining to their specific projects on remote terminals.

Chapter 2

Use

2.1 Program Initiation

DWINDO is initiated from job control by:

```
DWINDO [parameter] ...          [DOS]
dwindo [parameter] ...          [UNIX]
```

"parameter" has the form "keyword=xxx" and may be one or more of the following specifications:

Keyword	Default	Description
INPUT	CON(stdin)	Terminal input
OUTPUT	CON(stdout)	Terminal output
FORMS	forms	Forms description file
QFORMS	qforms	Forms quick reference file
TRMDEF	dwindo.trm	Terminal definitions file
REPORTS	dwindo.rep	File in which to print forms
FUNFILE	dwindo.fun	PREAD functions file
MACFILE	dwindo.mac	PREAD macros file
SCNFILE	dwindo.scn	PREAD screens file

2.2 Global Control

When the DWINDO program is first executed, the user is prompted for the type of display terminal in use and then the "D>" prompt is presented. At this point, commands are used to direct the functioning of the program. Normally, a time window is defined (TIME), a form is loaded (LOAD) and then entry and editing of data begins. Once entry and revision of data on the form is complete, the form is saved by the user and then program control returns back to the "D>" prompt. The program is also terminated from the "D>" prompt using the (FINISH) commnd. The type of terminal being used or emulated must be identified by the program and definitions of control keys for the terminal must be present in the terminal definition file (see section 3 below).

DWINDO has the capabilities of a system known as "PREAD" imbedded in the program executable. This system allows DWINDO to be operated using menu selection screens and predefined macros controlling the program. For more information, refer to PREAD User Interface User's Manual.

TIme start-date start-time end-date end-time

A time window is required to define the time span of the data to be retrieved or stored in the DSS file. For example:

```
TI 01SEP86 0730 31OCT86 0730
```

The argument "T" may be used to designate the current time and date. Further, "T" may be modified to indicate a reference relative to the current time and date. For example:

```
TI T-30D T
```

The example designates a time window which begins 30 days prior to the current date and time and ends with the current date and time. "H" is used to designate hours and "M" minutes in a relative time reference.

The time window determines the number of data values directly available (ie., those which can be scrolled or paged) in DWINDO for a particular form. A time series in DWINDO may contain up to 1000 values. When the possibility exists for inserting more data values, a lesser number must be retrieved in order to provide space for insertion. When regular-interval time series are retrieved, then the number of data consists of all observations of the series which fall within the time window, including ones with missing values. When irregular -interval data are retrieved, all observations within the time window are retrieved.

LOad form-name

The load command causes the form labeled "form-name" to be displayed, filled with data consistent with the current time window. If the time window has not been defined, an error message will result.

UPdate

UP (update) causes creation or update of the forms quick reference file. Forms reference information is read from the file "FORMS" in the current directory and stored in the file "QFORMS". FORMS must be prepared as specified in section 3 below.

STatus

ST (status) results in the display of key file names and the currently defined time window.

FInish

The FI (Finish) command terminates operation of the DWINDO program.

2.3 Data Entry

A successful LO command results in a clearing of the screen and the appearance of the specified form, filled with data found for the time window. The cursor is positioned in the first column on the row of data corresponding to the end of the time window. While data are shown in a form, the operation of DWINDO is controlled by keys on the keyboard or by commands.

2.3.1 Function Keys

The control functions are referred to by their mnemonics. The functions and their mnemonics are summarized in Table 1.

Table 1. Form Control Functions

Mnemonic	Description
EXIT	Prompt to save data, return to "D>"
CMD	Prompt for command
CURU	Move cursor up one row in column
TABF	Move cursor to next column
TABB	Move cursor to previous column
CURD	Move cursor down one row in column
BOL	Move cursor to beginning data in row
EOL	Move cursor to ending data in row
BNL	Move cursor to beginning data in next row
PGUP	Fill form with previous page of data
PGDN	Fill form with next page of data
INSL	Prompt for time, insert row at new time
DELL	Delete irregular-interval data observation
ENTR	Save new data value in memory, move to next column
CURR	Move cursor right one cell
CURL	Move cursor left one cell
INSC	Insert character(s) in cell
DELC	Delete character in cell
DUP	Duplicate previous value in column
ERAS	Erase (blank) cell
REST	Restore cell to value in memory
MISS	Enter missing value flag in cell

Data are entered by moving the cursor to the appropriate cell (column and row) and then entering data values with the numeric keypad. If data is already present in the cell, values can be changed using the insert and delete keys as well as overtyping the existing value.

Movement around the form is controlled mainly by the use of the function keys which control cursor movement: CURU, CURD, CURL, CURR, TABF, TABB, BOL, EOL, BNL, PGUP and PGDN. In addition, the ENTR function causes movement to the next cell in the row.

The order in which the cursor is placed in cells in response to the cursor movement keys is controlled, in turn, by the order in which the data are defined in the forms description file (see Chapter 3). Cells displaying intrinsic variables (dates and times) are skipped as are regular-interval time series cells for which there are no data for the particular time.

When the cursor is positioned at a data cell, the entry of numeric data and INSC, DELC, DUP, ERAS, REST, and MISS key entries imply one of three data change styles: entry, edit or replacement. The style is indicated on a status line at the bottom of the form.

Entry is the style implied when the cell is initially undefined with "M" (missing) displayed for regular-interval time series data or "--" for irregular-interval time series. In entry, the cursor remains at the rightmost position in the cell, and digits in the cell are shifted left as more are entered. A sign is placed at the beginning of the value if none exists; otherwise, it toggles the existing sign.

Entry	Cell
-	M
6	6
9	69
1	691
5	6915

Replacement is implied when the DUP function is used. In replacement, digits entered are interpreted as replacements for the least significant digits of the cell. As many digits of the cell are replaced as entered. For example:

Entry	Cell
<DUP>	98651
6	98656
7	98667

Editing is implied when the cell already contains a value or the INSC or DELC functions are used. Data in the cell are altered by a combination of inserting, deleting and overtyping.

In all three styles of change, the cell being changed is highlighted and the value in the cell no longer is consistent with the value in memory. Keyboard entries change the display until the enter key or a cursor movement key are depressed. At that point, the value displayed is submitted to a data checking process, the result is put in the memory and the memory value is confirmed by display of the memory contents for that cell. Data checking parameters are minimum, maximum, and maximum negative and positive changes and are specified in the forms definition file.

The enter key also causes movement to the next data cell in a regular sequence. If the sequence for a particular row is finished, the cursor is moved to the first in the sequence in the next row. If the next row is not showing on the form, scrolling or paging occurs as appropriate to reach the proper row.

Data entry and editing are made easier with special function keys: duplication of a preceding value (DUP); restoration of a the last previous value in memory (REST); erasure of a cell contents (ERAS); deletion and insertion of characters in a cell (DELC and INSC); and deletion of an irregular-interval time series observation (DELL).

The association of keys with functions is specified in a terminal definition file which is described in section 3 below. The program permits considerable flexibility in the designation of these associations, but the capabilities of the terminal being used may restrict the ease of implementation or use.

2.3.2 Commands

Commands are also used to direct the program when a form is displayed. The key designated for the CMD function is used, and a prompt (>) appears at the bottom of the form. Commands are entered in response to the prompt. Certain commands prompt for additional information as noted below.

GOTO

GOTO is used to move to data associated with a particular date and time in the form. The program prompts for the date and time, which are entered military style (see the TI command above). The desired date and time must be within the range of the time window in effect.

CHECK

CHECK allows a review of all data entered in a session entry to allow correction of the data before it is saved in a DSS file. Data exceeding specified limits are tagged in the entry process. When the CHECK command is given, a search is made of all data entered during the session. When a value is found with an error tag, the appropriate page of data is presented and the cursor is moved to the appropriate data cell to allow corrections. A message appears at the bottom of the screen to indicate the error condition and the value of the respective limit.

Corrections are made with the entry and change procedures described above. If an error condition is found again when the enter key is pressed, the cursor stays at the position and the reason for the error is again displayed at the bottom of the screen. If no error is found, the cursor is moved to the next erroneous value, if any are present. The error check may be overridden (ie., acceptance forced) by use of the TABF function rather than the ENTR key.

Pressing the EXIT key provides an escape from the checking process. Checking may be resumed later and will continue beginning at the same location where an escape was made. Otherwise, when the review process is complete, a message to that effect is displayed at the bottom of the screen, and the form is returned to its state prior to the CHECK command.

PRINT

An image of the screen is recorded in the file assigned to the reports parameter when the DWINDO program is initiated.

TW

The current time window is displayed on the message line at the bottom of the screen.

2.4 Saving Data

The EXIT key is used to terminate transactions in a form. Before the form is erased from the screen, the program inquires about saving the data. The default response is to save the data. If the response so indicates, the data in memory is copied to appropriate files. Data in the files for operative time window are overwritten.

(This page intentionally left blank)

Chapter 3
Implementation

A forms description file and a terminal definition file must exist in order to use DWINDO. The forms description file is used to describe forms and data references. The terminal definition file contains information describing the characteristics of particular terminals and their function key layouts. These files may be created or changed with any text editor.

3.1 Forms Description File

The forms description file FORMS contains information describing forms and their contents. Specifically, a forms description file contains: an image of the form as it is to appear on the screen; definitions of windows (or "viewports") in which data appear; definitions of individual data records which appear in columns in the windows; and locations of the data records in DSS files. A sample form definition is shown in Figure 2.

A processed version of FORMS is actually used by DWINDO to provide more rapid access to individual forms. This processing is performed by DWINDO when the UP (update) command is used. FORMS is assumed to be in the current directory.

All information about a particular form is located contiguously in FORMS in the order described below. Unique labels are used to indicate the separations among forms and among their parts: each section describing an individual form component ends with #END beginning in column 1.

Comments may be placed anywhere within the form definition, except within the form image itself (ie., between #FORM and the #END which terminates the form image). Comment lines begin with a "*".

Form specifications are entered on lines in "free form" unless otherwise noted. "Free form" means an individual specification need not begin in any particular column on a line, but must be provided in the order indicated and separated from other specifications by blanks or commas. A "null" entry (ie., an optional specification) must be indicated by two successive commas.

Certain form descriptors specify a location on the form in terms of rows or columns. A "row" is the same as a horizontal line on the display device and are numbered from 1 at the top of the form increasing downward. "Column," in this context, means a character position in a row, numbered from 1 at the leftmost position.

```
#FORM DATAH ***********************************************************************
 Location Name: ^N     River Mile: ^B     Description: Hydro-Power

 Date      Time     Stage-Upper      Stage-Lower      Hydro     Hydro
(ddmmmyy)  (hhmm)    (ft)    Change   (ft)    Change   Release   Units
 -------   ----     -----   -----    -----   -----    -----     ----
 -------   ----     -----   -----    -----   -----    -----     ----
 -------   ----     -----   -----    -----   -----    -----     ----

 Temperature  Precip  Weather  Tows Waiting  Delay   Tows Locked   Dam
 Air   Water   (in)    Type    Above Below  (hh) (mm)  Up    Down  Condition
 ----  ----   ----    ---     --    --     --   --    --    --    -----
 ----  ----   ----    ---     --    --     --   --    --    --    -----
 ----  ----   ----    ---     --    --     --   --    --    --    -----

    Weather Type:                F1  - Exit or Help          F5  - Override Data
 1 - Fair     4 - Fog            F2  - Insert new date/time
 2 - Cloudy   5 - Snow           F3  - Delete date/time line at cursor position
 3 - Rain     6 - Windy          F4  - Clear data field at cursor position
#END
* ---------------------------------------------------------------------------
*          NUMBER     NUMBER OF   ROW NUM OF   BEG COL    END COL
*            OF       ROWS PER    TOP ROW IN     OF         OF
*         VIEWPORT    VIEWPORT     VIEWPORT    VIEWPORT   VIEWPORT
* ---------------------------------------------------------------------------
#VPORTS       2          3
                                      7          1         80
                                     14          1         80
#END
* ---------------------------------------------------------------------------
* LABEL    VIEWPORT  BEG.   COL.   DATA   MIN    MAX    NEG    POS
*           NUMBER   COL.  WIDTH  FORMAT  VALUE  VALUE  CHANGE CHANGE
* ---------------------------------------------------------------------------
#DEF
@DDMMMYY      1       3      7      A
@HHMM         1      14      4      A
SUP           1      23      5     N1      1.0   99.0
CUP           1      33      5     N1     -9.0    9.0
SLW           1      43      5     N1      1.0   99.0
CLW           1      53      5     N1     -9.0    9.0
HPREL         1      63      5     N1      0.0  100.0
UNITS         1      73      4     N0        0      8
TAIR          2       3      4     N0      -40    120
TWTR          2      10      4     N0       32     90
PREC          2      18      4     N2     0.00    9.0
WEAT          2      27      3     N0        1      6
WAB           2      35      2     N0        0    100
WBL           2      42      2     N0        0    100
DHRS          2      48      2     N0        0    240
DMIN          2      53      2     N0        0     59
LUP           2      59      2     N0        0    100
LDN           2      66      2     N0        0    100
DCON          2      73      5     N1        0  999.0
#END
```

Figure 2 Form Definition Example

```
* ---------------------------------------------------------------------------
* LABEL DATA     DATA    TIME TIME DSS FILENAME:DSS PATHNAME
*       TYPE     UNITS   OFFS SHFT
* ---------------------------------------------------------------------------
#DSS
SUP   ,INST-VAL ,FEET    , , ,DSSNAV.DSS://^B/STAGE-UP//IR-YEAR/OBS/
CUP   ,PER-CUM  ,FEET    , , ,DSSNAV.DSS://^B/STAGE-CHANGE_UP//IR-YEAR/OBS/
SLW   ,INST-VAL ,FEET    , , ,DSSNAV.DSS://^B/STAGE-LO//IR-YEAR/OBS/
CLW   ,PER-CUM  ,FEET    , , ,DSSNAV.DSS://^B/STAGE-CHANGE_LO//IR-YEAR/OBS/
HPREL,INST-VAL ,KCFS    , , ,DSSNAV.DSS://^B/FLOW-HYDRO_POWER//IR-YEAR/OBS/
UNITS,INST-VAL ,COUNT   , , ,DSSNAV.DSS://^B/COUNT-HP_UNITS//IR-YEAR/OBS/
TAIR ,INST-VAL ,DEG-F   , , ,DSSNAV.DSS://^B/TEMP-AIR//IR-YEAR/OBS/
TWTR ,INST-VAL ,DEG-F   , , ,DSSNAV.DSS://^B/TEMP-WATER//IR-YEAR/OBS/
PREC ,PER-CUM  ,INCHES  , , ,DSSNAV.DSS://^B/PRECIP-INC//IR-YEAR/OBS/
WEAT ,INST-VAL ,CODE    , , ,DSSNAV.DSS://^B/CODE-WEATHER//IR-YEAR/OBS/
WAB  ,INST-VAL ,COUNT   , , ,DSSNAV.DSS://^B/COUNT-WAIT_ABOVE//IR-YEAR/OBS/
WBL  ,INST-VAL ,COUNT   , , ,DSSNAV.DSS://^B/COUNT-WAIT_BELOW//IR-YEAR/OBS/
DHRS ,INST-VAL ,HOURS   , , ,DSSNAV.DSS://^B/TIME-DELAY_HOURS//IR-YEAR/OBS/
DMIN ,INST-VAL ,MINS    , , ,DSSNAV.DSS://^B/TIME-DELAY_MINUTES//IR-YEAR/OBS/
LUP  ,PER-CUM  ,COUNT   , , ,DSSNAV.DSS://^B/COUNT-LOCK_UP//IR-YEAR/OBS/
LDN  ,PER-CUM  ,COUNT   , , ,DSSNAV.DSS://^B/COUNT-LOCK_DN//IR-YEAR/OBS/
DCON ,INST-VAL ,FEET    , , ,DSSNAV.DSS://^B/GATE-OPENING//IR-YEAR/OBS/
#END
```

Figure 2 Form Definition Example (cont'd)

3.1.1 Form Image

A form begins with the label #FORM name beginning in column one, where name is the name (one to eight characters) by which the form is retrieved. The lines following #FORM are a character-by-character image of the fixed component of the form as it appears on the screen. The numbers of lines and characters per line should be consistent with the maximum number available on the terminal to be used. The number of characters per line cannot exceed 132. Locations on the form where data are to appear will be overwritten after the form is displayed on the screen.

3.1.2 Viewports

The lines following the image define "viewports" and are bracketed by "#VPORTS" and "#END" keywords. Keywords like these must begin in column one (1). A viewport is a region of the form composed of horizontally adjacent columns. Viewports are used to effect rapid scrolling and paging. A viewport may contain as many columns as will fit (but without exceeding the limitations numbers of data records), and at least one and up to 10 viewports may be defined. All viewports scroll or page synchronously. Because they perform synchronously, all viewports have the same number of rows.

The line beginning with "#VPORTS" also specifies two viewport parameters: the number of viewports, and the number of rows per viewport. The subsequent lines define parameters for the viewports, one line per viewport. The parameters are: beginning row in the form and beginning and ending character positions on a row.

3.1.3 Data Definitions

The lines following the viewport definitions and bracketed by "#DEF" and "#END" provide definitions for the data to be displayed in columns in the form on the terminal screen. Each column is part of a time series of data: either expressions of time or data values. The expression of time is defined by the use of "intrinsic" values. Data values are retrieved from and saved in DSS files.

The data definition section of FORMS identifies the various data columns, provides information on where they're displayed on the screen, and, for data values, specifies numeric criteria for checking the values as they are entered. Each definition is specified in "free format" on a single line as described below.

The order in which the definitions appear on the form controls order in which they are accessed with horizontal cursor movements. A top to bottom order in FORMS corresponds to the conventional order of left to right for horizontal cursor movement.

Table 2. Data Definitions

Field	Description
1	Label for data item. Labels beginning with "@" designate intrinsic data --- see Table 3 below.
2	Viewport assignment: viewport 1 would be "1", etc.
3	Beginning character position on form.
4	Width of column for data in characters.
5	Format for data: A - alphanumeric Nn - numeric with n spaces to right of decimal if n is 0, no decimal is shown.
6	Minimum accepted value for screening data. A null entry implies no checking.
7	Maximum accepted value for screening data. A null entry implies no checking.
8	Maximum accepted negative change between the current and previous values. A null entry implies no checking.
9	Maximum accepted positive change between the current and previous values. A null entry implies no checking.

Table 3. Intrinsic Variables

Variable	Description	Example
@DDMMMYY	Military date	01JAN87
@DDMMMYYYY	Military date	01JAN1987
@DD	Day of month	01
@HHMM	Military time	1745

3.1.4 DSS References

The lines following the data definitions and bracketed by the keywords "#DSS" and "#END" specify references to data records in DSS files. The DSS references are specified in free format. The DSS reference is correlated with the data items by a label. Units, type, and offset are used only when the information is not found in a DSS file.

Table 4. DSS References

Field	Description
1	Label, same as corresponding data definition, up to 10 characters.
2	DSS data type default (INST-VAL, PER-AVER, INST-CUM, or PER-CUM). Used when originating a new DSS record or the type is not specified in the old record; otherwise, the type specified in the old record is used.
3	Data units default. Used when originating a new DSS record or the type is not specified in the old record; otherwise, the units specified in the old record are used.
4	DSS regular interval time offset default. Minutes from the beginning of the standard interval: for example, if a daily value is recorded at 0730, the time offset would be 450 (7 x 60 + 30). Used when originating a new DSS record; otherwise, the offset specified in the old record is used. A blank entry implies no time offset (data time is at end of interval).
5	Time shift. Minutes to shift the data forward (+) or backward (-) in time for display on the form. The data will be displayed on the form as follows: actual_time_of_data - time_shift Data entered will be stored in DSS at the actual time by applying the time shift as appropriate. Applies to regular-interval data only. A blank entry implies no shift.
6	DSS file and pathname reference in the form: filename:pathname For example: `MASTDB:/SCIOTO/CLSF4/FLOW/.....`

3.2 Terminal Definition File

ANSI terminals from different vendors tend to have extensive sets of functions which may be controlled by unique and exclusive command sequences from a host computer. As a result, a specific terminal usually must be configured for use with DWINDO through a terminal definition file named "`DWTRM.DEF`". This file contains control sequences used to specify initiation and termination sequences to initiate and restore the terminal and designate function key assignments. "`DWTRM.DEF`" is provided with the DWINDO program and may already contain a satisfactory description of the terminal to be used.

"`DWTRM.DEF`" is an ASCII (text) file divided into sections, one for each terminal type, headed by a line beginning with `TERM` and ending with a line beginning with `ENDTRM`. The line beginning with `TERM` also specifies the identity of the terminal by a 6 character identifier. Each section contains a sequence of: `IM` (initiate message); `RM` (reset message), `NL` (maximum number of lines); `NC` (maximum number of characters); `DE` (redefinition of a command key sequence); or `D2` (alternative definition of a command key sequence). These may appear in any order. An example terminal definition is shown in Figure 3.

TERM name

This line begins a sequence of lines providing configuration of a particular terminal identified as name. <name> may be up to 6 characters long.

NL no_of_lines

no_of_lines specifies the maximum number of lines available on the device screen.

NC no_of_char

no_of_char specifies the maximum number of characters available on one row of the display device.

IM and RM

Initiation messages (IM) and reset messages (RM) each consist of a sequence of up to forty characters which are interpreted into control sequences and sent to the terminal by DWINDO. A control character is represented as a combination of "^" followed by a normal character whose ASCII code is offset from the ASCII code of the control character by 64. For example, "^[" represents an escape (the ASCII code of "[" is 91; an escape is ASCII code 27).

```
TERM HDS
NL 24
NC 132
*  REDEFINE CONTROL KEYS
*
*  TURN ON TRANSPARENT MODE
*IM '^[Q'
*
*  SET TERMINAL TO ANSI
IM '^[<'
*  Set Horizonal scroll ON
*IM '^[[=4h'
*  Set Black Background
IM '^[[?5l'
*  Set normal attributes
IM '^[[0m'
*  Set replacement mode (instead of insert)
IM '^[[4l'
*
*  PROGRAM HDS KEYBOARD
*  Delete Line (ESC [ M)                        F17
IM '^[[17;1;0u*^[[M*'
*  Insert Line (ESC [ L)                        F18
IM '^[[18;1;0u*^[[L*'
*  Beginning of Line (control-D)                F10
IM '^[[10;1;0u*^D*'
*  Beg. of Next Line (ESC-E)                    Shift ENTER
IM '^[[185;1;0u*^[E*'
*  End of Line (control-F)                      F12
IM '^[[12;1;0u*^F*'
*  HELP (CMD?)                                  F20
IM '^[[20;1;0u*^[AA?*'
*  Restore (control-C)                          F21
IM '^[[21;1;0u*^C*'
*  Command  (ESC-A-A)                           F22
IM '^[[22;1;0u*^[AA*'
*  Insert Character Mode (ESC O p)              F1, Shift F1
IM '^[[1;1;0u*^[Op*^[[31;1;0u*^[Op*'
*  Insert Character Mode (ESC O p)              F16
IM '^[[16;1;0u*^[Op*'
*  Delete Character (ESC [ P)                   F13
IM '^[[13;1;0u*^[[P*'
*  Missing Value (ESC [ m)                      F106 (numeric keypad)
IM '^[[106;1;0u*^[[m*'
*  Tab Backward (ESC [ H)                       F203 (shift left arrow)
IM '^[[203;1;0u*^[[H*'
*  Tab Forward (^I)                             F202 (shift right arrow)
IM '^[[202;1;0u*^I*'
```

Figure 3. Terminal Definition Example

```
*  Page Up (ESC [ V)                                F206 (shift PAGE)
IM '^[[206;1;0u*^[[V*'
*  Page Up (ESC [ V)                                F200 (shift up arrow)
IM '^[[200;1;0u*^[[V*'
*  Page Down (ESC [ U )                             F126 (PAGE)
IM '^[[126;1;0u*^[[U*'
*  Page Down (ESC [ U)                              F201 (shift down arrow)
IM '^[[201;1;0u*^[[U*'
*  SCROLL UP Function (ESC O m)                     Shift SCROLL
*IM '^[[205;1;0u*^[Om*'
*  SCROLL DOWN Function (ESC O l)                   SCROLL
*IM '^[[125;1;0u*^[Ol*'
*  Exit Form            (ESC [ O)                   F23
IM '^[[23;1;0u*^[[O*'
*
*  Disable certain keys                             F5, Shift F5
IM '^[[5;6;0u^[[35;6;0u'
*
*  TURN OFF TRANSPARENT MODE
*IM '^[R'
*
*  Reset Numeric Keypad; Clear temporary key definitions
RM '^[>^[[0;8;0u'
ENDTERM
```

Figure 3 Terminal Definition Example (cont'd)

Initiation messages are interpreted and sent to the terminal as soon as they are encountered in the input stream. Reset messages are stored in memory and sent each time the program returns from ANSI (full screen) operation or an abnormal termination occurs.

DE and D2

Each of the keys designated for control actually generates a unique sequence of one or more characters which are transmitted to the host computer when the key is pressed. Each sequence begins with a control character to distinguish the control sequence from literal input. A control character is represented as a combination of a "^" followed by a normal character whose ASCII code is offset from the ASCII code of the control character by 64. For example, "^[" represents an escape (the ASCII code of "[" is 91; an escape is ASCII code 27). A list of the functions is shown in Table 5.

Table 5. Control Function Sequences

Mnemonic Default	Control Function	Control Sequence
BNL	Move to beginning of next line	^[E
BOL	Move to beginning of current line	^D
CMD	Prepare to accept command	^[AA
CURD	Move down one line, same column	^[[B
CURL	Move left one cell	^[[D
CURR	Move right one cell	^[[C
CURU	Move up one line, same column	^[[A
DELC	Delete character	^[[P
DELL	Delete line	^[[M
DUP	Duplicate previous entry in column	^[OR
ENTR	Enter (carriage return)	^M
EOL	Move to end of line	^F
ERAS	Erase entry	^[OQ
EXIT	Exit form	^[[O
INSC	Insert character	^[Op
INSL	Insert line	^[[L
MISS	Enter missing value flag	^[[m
PGDN	Page down	^[[U
PGUP	Page up	^[[V
REST	Restore entry (from memory)	^C
TABB	Move to previous column	^[[H
TABF	Move to next column	^I

The table may be altered by DE and D2 directives in the terminal definition file, with each directive altering the control sequence for a single function. DE substitutes a sequence for the default one. For example, the following sequence designates an alternative sequence for CURR:

```
DE CURR=^[[KJ
```

D2 creates an additional sequence which is recognized for a function to allow more than one key to be designated for the same function. The syntax for a D2 directive is the same as for DE. The following limitations apply in use of DE and D2:

1) Two different functions may not be assigned the same control sequence, including default sequences.

2) Escape sequences may not be more than four characters long, including the escape character.

3) Sequences that do not begin with an escape (^[) are restricted to the following (to avoid conflicts with those which have other significance in communications between the terminal and host computer): A B C D F P R T U V W Y \ ] _

WATDSS

Hydrologic Engineering Center
Data Storage System
Watstore to DSS Data Entry Program

User's Manual

Version 2.4
March 1995

Hydrologic Engineering Center
U.S. Army Corps of Engineers
609 Second Street
Davis, California 95616-4687
(916) 756-1104

WATDSS

Introduction

Program WATDSS extracts daily flow data from a Watstore format 3 file, or data from a Watstore format B file, and stores it in a HEC-DSS data base file. The data is stored in the regular interval time series format. WATDSS is not capable of storing other types of data, nor is it capable of retrieving data from the USGS. The file read by WATDSS may have come from the USGS, or it may have been created from a CD-ROM data base. When creating a data file from CD-ROM, the "card" format is the correct format to use.

All data in the input data file may be stored in the HEC-DSS file, or data for select stations (not selected times) may be stored. The A, B, C, and F pathname parts may be specified, or defaults will be used. If the B (location) parts are not given, the USGS gage numbers from the data file will be used. To specify the B parts, a file must be created that contains the USGS gage number followed by the appropriate B part for the data set for each station to store. The other pathname parts are given as execution line parameters.

Use

All parameters, except for the optional B part names, may be specified on the execution line. The available parameters are:

Name	Default	Description
INPUT	standard in	File containing the Watstore data
OUTPUT	standard out	Output and error messages
DSSFILE	none	DSS file to store the data in
A	WATS	A (basin) part of the pathnames
C	FLOW	C (parameter) part of the pathnames
F	OBS	F (additional identifier) part of the pathnames
SID	none	Name of file containing station ID's and B parts
ALL	YES	Store all data in the input file. If set to NO, store data for only those stations in the station ID (SID) file.

The minimum parameters required are the input file (containing the Watstore data) and the

DSS file name to store the data in. If the DSS file does not exist, it will be created. To specify a parameter on the execution line follow the program name by a space or comma, the parameter name (or abbreviation), an equal sign, then the parameter (with no spaces). For example:

```
watdss input=mydata dssfile=datab
```

In this example, all data in the Watstore file "mydata" (which was created earlier from CD-ROM or retrieved from the USGS) is stored in the DSS file named "datab". The pathnames of the data have an A part of "WATS", a B part of the USGS gage number, a C part of "FLOW", and an F part of "OBS".

Execution line parameter names may be abbreviated to 2 characters. They may be given in any order on the execution line. For example:

```
watdss in=mydata dss=datab a=sacramento f=usgs
```

(Note that the parameter names and the pathname parts may be either in lower or upper case. Pathnames are converted to upper case when the data is stored in the DSS file.)

The B parts of pathnames can be set by the use of a station ID file. The station ID file is a file created by the user that contains the USGS gage number (found in each data line in the Watstore data file), followed by a comma and the B part of the pathname for that station. There should be one line for each station. For example:

```
0178900,CEDAR CREEK
0137430,BATESVILLE
0843900,CLEAR WATER
```

If this file is named "mysids", the execution line might be:

```
watdss in=mydata dss=datab a=sacramento f=usgs sid=mysids all=NO
```

If only selected stations are to be stored in the DSS file (assuming that the Watstore data file contains more stations than you want), a station ID file must be created with the station names of the data to store, and the "ALL" execution line parameter set to "NO". For example:

```
watdss in=mydata dss=datab a=sacramento f=usgs sid=mysids all=NO
```

If the "ALL=NO" parameter is not given, all data in the Watstore file will be stored in the DSS file. Those stations without entries in the station ID file will have a B part of their USGS gage number. Note that there is no capability to specify a time window for data.

Example

Watstore format 3 data was retrieved from CD-ROM, using the "`card`" format. The data was stored in a file given the name "`cache.flo`". It contains the following:

```
Z                                   USGS
H 11452500      38433112148220006061 13SW180201101139.00   0.00 0052.27-99999.00
N 11452500      CACHE CREEK AT YOLO, CALIF.
2 11452500      999999999999   60    3                 ENT
3 11452500      198410 1     18     19     19     19     20     18     16     18
3 11452500      198410 2     21     23     31     34     32     27     22     23
3 11452500      198410 3     26     24     23     20     21     21     21     21
3 11452500      198410 4     23     25     26     26     25     26     27
3 11452500      198411 1     28     22     16     14     14     14     14     20
3 11452500      198411 2     15     18     23     25    104    357    188    176
3 11452500      198411 3    311    219    143    130    113    100     88     93
3 11452500      198411 4    102    151    130   1340    717    416 999999
3 11452500      198412 1    314    272    344    474    334    291    291    318
3 11452500      198412 2    234    243    418    339    275    245    233    227
3 11452500      198412 3    224    213    191    175    166    165    156    153
3 11452500      198412 4    152    151    154    157    152    147    144
3 11452500      1985 1 1    142    140    116     92     86     82     90     92
3 11452500      1985 1 2    104     96     91     88     82     80     78     77
3 11452500      1985 1 3     75     74     71     70     69     67     66     65
3 11452500      1985 1 4     64     65     63     62     61     60     60
```

A (small) file containing the USGS gage number and the selected gagename was created and named "`cache.sid`". It contains the following:

```
11452500,YOLO
```

The execution line to load the Watstore data into DSS is:

```
watdss in=cache.flo dss=cache.dss a=cache sid=cache.sid
```

The output from this execution is:

```
   -----DSS---ZOPEN:  New File Opened,  File: cache.dss
                      Unit:  71;  DSS Version: 6-IG
-----DSS---ZWRITE: /CACHE/YOLO/FLOW/01JAN1984/1DAY/OBS/
-----DSS---ZWRITE: /CACHE/YOLO/FLOW/01JAN1985/1DAY/OBS/
   -----DSS---ZCLOSE Unit:  71,   File: CACHE.DSS
              Pointer Utilization:   .25
              Number of Records:      2
              File Size:     14.6  Kbytes
              Percent Inactive:    .0
```

NWSDSS

Hydrologic Engineering Center
National Weather Service to
Data Storage System Conversion Utility

User's Manual

Version 5.3
March 1995

Hydrologic Engineering Center
U.S. Army Corps of Engineers
609 Second Street
Davis, California 95616-4687
(916) 756-1104

NWSDSS

Table of Contents

Chapter **Page**

1. Purpose 1

2. Description 1

3. Use 2
   3.1 Input Parameters 2
   3.2 Program Options 3

4. Programmer Information 3

Appendices

A. Input Description A-1
B. Sample Output B-1
C. Sample Catalog of Stations C-1

NWSDSS

1. Purpose

NWSDSS is a utility program which extracts precipitation and other meteorological information from National Weather Service files containing National Climatic Data Center (NCDC) formatted data, and stores the data into a Data Storage System (DSS) database file. The program can process several NCDC formats including TD-3240 and TD-9654 (hourly data) as well as TD-3200 and TD-9655 (daily data). Hourly data is stored in the regular-interval time series format; daily is stored in the irregular-interval format. The program may also be used to create a catalog of all stations encountered in the NCDC file.

2. Description

Data is extracted from NCDC formatted data files according to specifications provided by the user in an input file. The specifications include a list of stations, a list of events, and an optional list of desired weather data elements supplied by the user. The list of stations consists of station identifiers in ascending numerical order and their respective alphanumeric identifiers. The list of events contains a beginning year and month and duration in months for each event. The events are specified in chronological order. The weather data element list specifies which of the various possible elements are to be extracted.

Observation times for daily stations may optionally be specified in the input list. These times are recorded in the respective DSS records, overriding the times indicated in the NCDC file. A complete description of user specifications and their formats is presented in Appendix A.

All available weather data elements are extracted for each available event and station and the resultant DSS pathnames are reported in the program's output; stations and events not found are also reported. Sample program output is presented in Appendix B.

A list of all stations encountered in the NCDC file and the starting and ending dates of their records may be optionally cataloged in a separate file. An example of the catalog of stations is shown in Appendix C.

Precipitation amounts accumulated during periods of missing hourly data are available from files containing the TD-3240 format. When they are found during extraction, the accumulated amounts and the times and dates of the accumulation period are written to a separate file which may be used as miscellaneous precipitation input to the program PRECIP. An example of the accumulated amounts is also shown in Appendix C.

3. Use

NWSDSS is normally run as a batch job (DOS) or in a script process (UNIX). Sample executions for each operating system are shown below.

DOS:

```
NWSDSS  INPUT=NWS.LST  OUTPUT=NWS.OUT DSSFILE=NWS.DSS  CDROM=NWS.DAT
```

UNIX:

```
#! /bin/csh
nwsdss  input=nws.lst output=nws.out dssfile=nws.dss cdrom=nws.dat
```

Keywords on the command line can be abbreviated. For example, the following two lines will do the same thing:

```
NWSDSS INPUT=NWS.LST OUTPUT=NWS.OUT  DSSFILE=NWS.DSS  CDROM=NWS.DAT

NWSDSS  I=NWS.LST  O=NWS.OUT  DSS=NWS.DSS  CD=NWS.DAT
```

The input list is read from a file in the format shown in Appendix A. The input list is echoed in the output. Output may be displayed on the console or directed to a file.

3.1 Input Parameters

Input parameters are used to make file assignments and to specify global input variables. Default file assignments are made by the program, but substitutes may be specified by the user on the program execution line.

The parameter names and defaults are:

Name	Default	Description
INPUT	CON(stdin)	Input file
OUTPUT	CON(stdout)	Output file
DSSFILE	nwsdss.dss	DSS file
CDROM	***'no default'***	NCDC data file (**must be specified**)
CAT	nwsdss.cat	Catalog file
ACC	nwsdss.acc	Accumulated amounts file
A	NWSDSS	DSS pathname part A to use for output

3.2 Program Options

Program options are used to select optional program functions. They may be specified when initiating the program as follows:

```
NWSDSS  -OPTIONS  PARAMETERS
```

Example:

```
NWSDSS  -LTD  IN=NWS.LST  OUT=NWS.OUT  DSS=NWS.DSS CD=NWS.DAT
```

Options include:

```
L - show the contents of each data record extracted from the NCDC file
M - show the contents of each data record containing missing data flags
C - generate catalog (list) of all stations found in the NCDC file
T - trace program operation
D - debug program operation
```

4. Programmer Information

NWSDSS is written in FORTRAN and is implemented in both the DOS and UNIX environments. Use of the program presumes that the NWS data has been extracted into NCDC formatted text files from a CD-ROM or some other source of data prior to use of the NWSDSS program.

When extracting data from the CD-ROM, the user typically uses a menu-based system to select time periods, station ID's, time increment and parameter types. The selected data is then read from the CD-ROM and written to a text file using the appropriate NCDC format. This file is then used as the input data file for the NWSDSS program.

Appendix A

Input Description

Description

The input to NWSDSS identifies stations and event time windows for which data is desired. The input file must also specify the type of NCDC format used. Optional comments may be used throughout the list, and a time of observation may be specified for daily values. The input list must not exceed the maximum allowable number of stations (=250), events (=30), and hours of observation (=500).

Each type of input is provided on a separate input line and is labelled with a two-character identifier as follows:

```
TA   NCDC Data Format
**   Comment
ST   Station
EV   Event
ET   Weather Element Types
HO   Time of Observation
```

Order

Comments may appear anywhere. The NCDC data format specification must be the first non-comment record. Stations must appear in ascending order of their index numbers. Event windows must appear in ascending chronological order. Weather element types may appear anywhere. Hour of observation information must appear in ascending order of station index numbers first and then in ascending chronological order according to the effective dates of changes in hour of observations. Station, event and hour of observation information may appear in any sequence relative to each other as long as each type is in its respective order.

** Comment

Field	Columns	Content	Description
1	1 - 2	"**"	Comment identifier
2	3 - 80	a...a	Comment

TA NCDC Format

Field	Columns	Content	Description
1	1 - 2	"TA"	NCDC format information identifier
2	3 -	a...a	"TD32" - TD3200 or TD3240 format "TD9654" - TD9654 format "TD9655" - TD9655 format

ST Station

Field	Columns	Content	Description
1	1 - 2	"ST"	Identifier for station information
2	4 - 9	iiiiii	NCDC station identification number (integer)
3	10 - 41	aaa a	Alphanumeric station label to be used in DSS pathname

HO Hour of Observation[1]

Field	Columns	Content	Description
1	1 - 2	"HO"	Identifier for hour of observation information
2	4 - 9	iiiiii	NCDC station identification number (integer)
3	11 - 12	ii	Hour of observation, 24 hour clock (integer) if zero, hour of observation in NCDC file is used
4	14 - 17	iiii	Year observation time began (integer)
5	19 - 20	ii	Month observation time began (integer)
6	22 - 23	ii	Day observation time began (integer)

EV Event Window

Field	Columns	Content	Description
1	1 - 2	"EV"	Identifier for event window information
2	4 - 7	iiii	Beginning year of event (integer)
3	9 - 10	ii	Beginning month of event (integer)
4	12 - 13	ii	Length of event in months (integer)

[1]Optional, for daily data only, and for observation times prior to December, 1981. Later observation times are considered reliable and, therefore, override HO specifications.

ET Element Type[1]

Field	Columns	Content	Description
	1 - 2	"ET"	Identifier for element type information
2	3 -	aaaa aaaa	Weather element codes, separated by commas or blanks[2]
			PRCP - Precipitation
			DYSW - Weather occurence
			EVAP - Evaporation
			MNPN - Minimum precipitation
			MXPN - Maximum precipitation
			SNOW - Snowfall
			SNWD - Snow depth
			TMAX - Maximum temperature
			TMIN - Minimum temperature
			WDMV - Wind movement
			WTEQ - Snow water equivalent

[1]Optional. Applies only to daily data. Default is PRCP (precipitation).

[2]The following are equivalent:

```
ET SNOW SNWD TMAX TMIN
```

and

```
ET SNOW
ET SNWD
ET TMAX
ET TMIN
```

Example

The following is an example of an input data set:

```
TA TD32
**
** Example Input Data
**
ST 180015 Station #1
ST 468536 Station #2
ST 490000 Station #3
EV 1932 03 02
EV 1948 08 03
EV 1955 01 03
EV 1984 10 03
HO 468536 15 1940 01 04
HO 468536 16 1948 09 06
HO 468536 07 1955 06 01
ET PRCP TMAX TMIN
```

In this example, daily precipitation and temperature minimums and maximums are to be extracted from a file in the TD3200 format for three stations and four events. The first event occurred in March and April 1932. An observation time is specified for station 468536 for three different periods; observation times for the other stations and prior to January 4, 1940 at station 468536 will be the time reported in the NCDC file.

Appendix B

Sample Output

```
 N W S D S S    VERSION  5.3.0   SEPTEMBER 1, 1994

          15SEP94    10:15:01


INPUT DATA
-------------

  1 TA TD9654
  2 ST 361100 STATION # 1
  3 ST 367851 STATION # 2
  4 ST 369999 STATION # 3
  5 EV 1932 07 02
  6 EV 1969 01 01
  7 EV 1969 08 01
  8 EV 1970 07 02
  9 EV 1971 01 03
 10 EV 1977 10 01


              -----DSS---ZOPEN      NEW FILE OPENED   71  NWSDSS.DSS    42
 TAPFMT = TD9654   FREQ = HOURLY
 STATION # 1                       NOT ON TAPE
 MONTH  7 YEAR 1932 NOT AVAILABLE AT STATION # 2
 MONTH  8 YEAR 1932 NOT AVAILABLE AT STATION # 2
 MONTH  1 YEAR 1969 NOT AVAILABLE AT STATION # 2
 -----DSS---ZWRITE FILE  71, VERS.   1   /NWSDSS/STATION #2/PRECIP-ING/
01AUG1969/1HOUR/OBS/
 -----DSS---ZWRITE FILE  71, VERS.   1   /NWSDSS/STATION #2/PRECIP-INC/
01JUL1970/1HOUR/OBS/
 -----DSS---ZWRITE FILE  71, VERS.   1   /NWSDSS/STATION #2/PRECIP-INC/
01AUG1970/1HOUR/OBS/
 -----DSS---ZWRITE FILE  71, VERS.   1   /NWSDSS/STATION #2/PRECIP-INC/
01JAN1971/1HOUR/OBS/
 -----DSS---ZWRITE FILE  71, VERS.   1   /NWSDSS/STATION #2/PRECIP-INC/
01FEB1971/1HOUR/OBS/
 -----DSS---ZWRITE FILE  71, VERS.   1   /NWSDSS/STATION #2/PRECIP-INC/
01MAR1971/1HOUR/OBS/
 MONTH 10 YEAR 1977 NOT AVAILABLE AT STATION # 2
 END OF TAPE ENCOUNTERED
 STATION # 3                       NOT ON TAPE
 END OF JOB
                    -----DSS---ZCLOSE FILE   71
                              NO. RECORDS=       6
                              FILE SIZE=      9700 WORDS,     87 SECTORS
                              PERCENT INACTIVE=  0.00
```

Appendix C

Sample Catalog of Stations

T A P E C A T A L O G

STATION	FROM YR	FROM MO	TO YR	TO MO	EVENT TYPE
367847	1948	11	1948	11	PRCP SNOW SNWD
367851	1969	7	1973	12	PRCP SNOW SNWD TMAX TMIN DYSW
367855	1948	5	1956	3	PRCP TOBS DYSW

Sample Accumulated Amounts

ACCUMULATED PRECIPITATION

STA	LAT LON	AMT IN	DUR HR	DATE	TIME
150031	LAT LON	000.35	11	31JAN49	0800
150031	LAT LON	000.57	10	13JAN51	1000
150031	LAT LON	000.55	21	13MAR51	1100
150031	LAT LON	000.48	12	27OCT53	0600

PREAD User Interface

Hydrologic Engineering Center
Functions, Macros and Screens

User's Manual

March 1995

Hydrologic Engineering Center
U.S. Army Corps of Engineers
609 Second Street
Davis, CA 95616-4687
(916)756-1104

PREAD
Table of Contents

Chapter **Page**

1. Introduction 1
   - 1.1 Function References 1
   - 1.2 Macro Procedures 1
   - 1.3 Screen Selection 2
   - 1.4 Support Capabilities 2
   - 1.5 Extended Capabilities 2
   - 1.6 PREAD Commands 3

2. Commands
   - 2.1 CHAIN 5
   - 2.2 COED 5
   - 2.3 DSPLAY/LIST 6
   - 2.4 ECHO 6
   - 2.5 EDIT 7
   - 2.6 FIND 7
   - 2.7 FUNCTION 8
   - 2.8 HARDCOPY 8
   - 2.9 IF/ELSEIF/ELSE/ENDIF 9
   - 2.10 JCL 10
   - 2.11 KBLINE 10
   - 2.12 LEARN 11
   - 2.13 LOG 11
   - 2.14 PAGE 12
   - 2.15 PAUSE 12
   - 2.16 PRINT 13
   - 2.17 RUN 14
   - 2.18 SCREEN 14
   - 2.19 SET/GET 15
   - 2.20 STATUS 15
   - 2.21 TEACH 15
   - 2.22 WAIT 16
   - 2.23 *COMMAND 16
   - 2.24 ?COMMAND 16

Appendices

A Function File A-1
B Macro File B-1
C Screen File C-1
D Samples D-1

Chapter 1

1. Introduction

The incorporation of the PREAD user interface in an interactive or batch executed program can greatly enhance its ease of use. PREAD allows a program to conveniently take input from the keyboard as would normally occur in an interactive environment, but also allows several input alternatives.

Input alternatives available to any program using PREAD are:

1) Function references,
2) Macro procedures, and
3) Screen selection input.

Function references, macro and screen procedures may be utilized by interactive users at any kind of terminal. PC users may use a mouse to select items from a screen.

1.1 Function References

The function capability allows any ASCII character to be redefined to mean a string of zero or more ASCII characters. For example, the dollar sign "$ "could be redefined to be the string "PLOT" or the pound sign "#" could be redefined to be the string "THIS IS A TEST LINE". Then each use of the $ or # characters would produce the redefined equivalent.

The function capability can be a significant asset for frequently typed strings entered at the keyboard. Function references can also be nested, so that one function reference may reference other functions. Functions may also be referenced by input lines from macros as discussed later. Function definitions are preserved in the function file.

1.2 Macro Procedures

Macro procedures are sets of recurring lines of input. If, for example, an interactive program requires 5 or 6 lines of input to create a certain plot, the 5 or 6 lines could be stored in a macro. When the plot is desired, the user runs the macro to produce the plot. The execution of a macro can make using an interactive program considerably more pleasant for repetitive uses. Lines stored in the macro can contain function references. Since macros can accept parameters at execution time, a macro with a function reference can produce 2 different plots if the function

reference is redefined between its executions. Macros can also be nested. Macro definitions are preserved in the macro file.

1.3 Screen Selection

Screen selection allows the execution of complex programs by novice users, as well as by experienced users. When desired, the user is presented a screen of options from which to select. All entries by the user must come from the list of valid responses. A valid response causes the program to execute one or more operations. The user is prevented from making erroneous requests, and can only do what the selection screen permits. This is very useful for controlling access to high level programs where the user is unfamiliar with the program, or how to use it. Screen definitions are preserved in the screen file.

1.4 Support Capabilities

Each of the above capabilities is controlled by a separate file that describes the functions, macros, and screens available to the user. These files are normal text files that can be edited by the user to change the function definitions, macro procedures, or screens . The format of each of these files is given in the appendices of this document.

The user can define or redefine functions directly without the need to edit the appropriate text file. Similarly, the user can have PREAD create a macro as the steps are executed by the program. Screens can only be created or changed by use of a text editor. Optionally, commands can be echoed to the user terminal as they are executed. This can be helpful when function references are used as input or lines are retrieved from a macro file.

PREAD can also keep a log of all lines entered at the terminal, run from a macro, or selected from a screen. This feature can help reconstruct critical terminal sequences. Input lines are stored in the log file.

1.5 Extended Capabilities

PREAD extended capabilities require special implementation for each computer system used. The CHAIN feature allows the user to interrupt the execution of the program at any program input location and begin execution of any other program. When the user has completed execution of the second program, the user can then return to the first program continuing execution where the interruption first occurred.

The JCL feature allows the user to move into job control at any point in the execution of a program and resume execution when JCL activities are complete. Caution must be exercised while in job control not to destroy files needed to resume execution of the interrupted program.

1.6 PREAD Commands

All lines that contain the command character in the first character position of a line are assumed to be commands to PREAD. The default command character is an exclamation point "!". A minus sign " - " following the command character and preceding the command will cause appropriate commands to take on their opposite meaning. (eg., `!-echo`) An equal sign "=" following the command character and preceding the command will cause appropriate commands to display current definitions. (eg., `!=function`)

Commands may be abbreviated to two or more characters.

Some commands accept option characters. Options are single characters separated from the command by a period "." (e.g., `!WAIT.X 10`). PREAD commands and options are case insensitive; that is, they may be entered in any combination of lower and upper case. Arguments may or may not be case sensitive depending on their usage.

Syntax:

```
!command [.option]  [arguments,...]
```

Examples:

```
!Teach  #  String
!wait.x  10
!Run  myMac
```

(This page intentionally left blank)

Chapter 2

Commands

2.1 Name: CHAIN

Use: `!CHAIN program name`

Description: The chain command allows any other program to be executed from the current host program. When the new program is terminated the execution of the original program is resumed where it was interrupted.

Example:

```
!CHAIN  DSSUTL
!chain  dsplay macfile=dsp123 dssfile=mydss
```

2.2 Name: COED

Use: `!COED edit file name`

Description: The COED command allows the current host program to be interrupted to edit a file. Upon termination of the edit, control will revert to the host program.

Example:

`!COED MYDATA` - Edit the MYDATA file.
`!coed dspmac` -Edit the dspmac macro file.

2.3 Name: DISPLAY / LIST

Use: `!DI [list file name] [first line of file to be displayed] [first column of file to be displayed]`

Description: The DISPLAY and LIST commands are identical. They each show lines from a text file. If the command is given without a filename, the next series of lines from the file will be displayed. See the examples below.

Options: T - Text only will be output suppressing frame line normally shown at beginning of each page of listing.

Example:

`!LIST MYTEXT`	Display file MYTEXT
`!list.t myfile 200`	Display file MYFILE beginning with Line 200 without header
`!LIST`	Display next series of lines
`!-Display`	Display previous series of lines

2.4 Name: ECHO

Use: `!ECHO`

Description: The line being processed by PREAD may be echoed to the screen. This is often of interest when the user wishes to see the lines of a macro being executed.

Example:

`!-ECHO` - Turn echo off (if previously turned on)
`!echo` - Turn echo on (if previously turned off)

2.5 Name: EDIT

Use: `!EDIT macro-name`

Description: The edit command allows macros to be created, changed, or deleted on-line. The required parameter is the name of the macro to be edited. If the macro does not already exist it will be created. Only simple (I) insert, (D) delete, (R) replace, (N) next, (P) print, and (F) finish commands may be used. Major macro editing activities should be performed by exiting the program and using a normal text editor.

Example:

```
!EDIT PLOTIT
!edit bigmac
```

2.6 Name: FIND

Use: `!FIND [file name] find string`

Description: The FIND command will find a string in a file. The file to be used may be specified, or will be the last referenced DISPLAY/LIST file. A FIND without a string will show the next occurrence of the string in the file.

Example:

```
!FIND MYOUTPUT SUMMARY
!find ERROR
!FIND
```

2.7 Name: FUNCTION

Use: `!FUNCTION`

Description: Turns on or off function mode, or shows current function definitions. The default at program initiation is OFF.

Example:

`!FUNCTION` Turn function mode on (any character that is defined in the function file will be replaced by its function value without the need for a function shift character).

`!-FUNCTION` Turn function mode off (function characters will be treated as normal characters unless preceded by the function shift character).

`!=FUNCTION` Display all defined functions (all currently defined function characters will be displayed with their definition).

If function mode is off (the default) a function reference may be forced by preceding the function reference with the function shift character. The default function shift character is the caret "^" . The function shift character is defined at the beginning of the function file and may be changed there by a text editor. It is generally recommended that the function mode be left in the OFF state and all desired function requests use the function shift character. For example, with automatic function replacement OFF if the pound sign, " # ", has been defined as the string "`123`", the expression "`ABC#DEF^#GHI`" would be expanded to be "`ABC#DEF123GHI`".

2.8 Name: HARDCOPY

Use: `!HARDCOPY`

Description: The hardcopy command will generate the code sequence required to trigger a screen copy for a Tektronix or compatible device.

Example:

```
!Hardcopy
```

2.9 Name: IF/ELSEIF/ELSE/ENDIF

Use:
```
!IF ( 'string1' .EQ. 'string2' ) THEN
!ELSEIF ( 'string3' .EQ. 'string4' ) THEN
!ELSE
!ENDIF
```

Description: The macro file may contain conditional execution lines governed by `IF`, ELSEIF, ELSE, and ENDIF statements. The conditional statements may be nested. Each IF sequence may be followed by any number of optional ELSEIF conditions, not more than one optional ELSE condition, and MUST be terminated by an ENDIF statement. Each of the expressions must be a character string. The string may be a function reference (e.g., `^a, ^M`), a parameter passed into the macro, a literal string (e.g., `'FIRST'`, `'Bone Creek'`), or combinations thereof. The only logical tests permitted on the strings are equal `".EQ."` or not equal `".NE."`.

Example:

```
!IF ( '^a' .EQ. 'FIRST' ) THEN
!RUN FIRSTONE
!ELSEIF ( '^a' .EQ. 'SECOND' ) THEN
!RUN SECONONE
!ELSEIF ( '^a' .NE. '999' ) THEN
!RUN ANYONE
!ELSE
!RUN EVERYONE
!ENDIF
```

```
!IF ( '$PARM' .NE. 'FLOW' ) THEN
!IF ( '$LOC' .EQ. 'BLACK BROOK' ) THEN
!Run Special $LOC
!ELSE
!Run Normal $LOC $PARM
!ENDIF
!ENDIF
```

2.10 Name: JCL

Use: `!JCL [system command]`

Description: The JCL command allows the user to interrupt the current host program and return to the operating system. To resume execution of the interrupted program enter EXIT.

Example:

```
!JCL
PRINT MYFILE
EXIT
```

2.11 Name: KBLINE

Use: `!KBLINE [function character]`

Description: The KBLINE command is used to interrupt a macro to allow the next line of input to be entered from the keyboard. After entry the currently executing macro resumes execution. If the command contains an optional function character, the next line entered from the keyboard will be used to define that function character.

Options:

- `H` - Hold moving to new line and hold displaying prompt.
- `P` - Prompt even when H option in effect.
- `S` - Screen response last selected will be used as the input.

Example:

`!KBLINE`	(Next line from keyboard will be past to program)
`!KBLINE.H $`	(Next line from keyboard will be stored as $, no prompt or newline will be issued)
`!kbline.s #`	(Last selection line from a screen will be stored in #)

2.12 Name: LEARN

Use: `!LEARN new macro name`

Description: Creates a macro by learning (remembering) a series of entries made by the user. Macros may also be created by editing the macro file. Creating a macro by editing the macro file is the most common method to use.

Example:

`!learn abc`	(Begin saving lines in macro ABC. All subsequent input lines will be included in macro ABC)
`!-LEARN`	(End saving lines in current macro)

2.13 Name: LOG

Use: `!LOG`

Description: This command controls the logging of lines to the log file. All lines that are past to the host program may be written to a log file for future reference. The log command may enable or disable this feature.

Example:

`!Log`	- Enable log file (if previously desabled)
`!-LOG`	- Disable log file (if previously enabled)

2.14 Name: PAGE

Use: `!PAGE [page type]`

Description: The page command will generate the code sequence required to trigger a new page (screen clear) for a Tektronix or compatible device. A parameter of GRAPHICS will cause a screen clear of the graphics area. A parameter of PRINTER will generate a formfeed character.

Example:

```
!PAGE
!PAGE GRAPHICS
!page printer
```

2.15 Name: PAUSE

Use: `!PAUSE`

Description: Causes a macro to pause until a carriage return (CR) is entered from the keyboard. This allows a macro to contain a series of plots, or other steps, and allow the user to control when its execution will proceed.

Options:

`H` - Hold moving to new line and hold displaying prompt.
`P` - Prompt even when H option in effect.

Example:

```
!PAUSE
!pause.hp
```

2.16 Name: PRINT

Use: `!PRINT [message to be printed]`

Description: The print command allows a message to be displayed to the user regardless of the echo status. If the .C option is used, the message will be written at the cursor location specified and exactly three fields must follow the command. The cursor line and column fields must be numeric values. If the .C option is used, the message must be enclosed in quotes. If the numeric values are unsigned, absolute screen coordinates are used. If they are signed, relative movements from current location are made. With the .C option, the cursor remains at last position after message. Zero length messages are permitted.

Option:

`C` - Cursor positioning used.

Example:

```
!PRINT -- Press <RETURN> to continue --
!print.c 10,20 "Message prints at line 10, column 20"
!PRINT.C 11,20 "Message prints at line 11, column 20"
!print.c +2,-20 "Message prints 2 lines down and 20 columns left"
```

2.17 Name: RUN

Use: `!RUN macro name`

Description: Runs a macro, displays names of macros, displays the contents of a macro. Macros may be passed parameters at execution time. See Appendix B for examples of macros defined in a file and parameter passing conventions.

Example:

`!run one`	(Execute macro named ONE)
`!=RUN`	(Display names of all defined macros)
`!=RUN TWO`	(Display contents of macro TWO)
`!run PLOTIT RED STAGE OBS`	(Execute macro named PLOTIT, passing three parameters RED, STAGE, and OBS.)

2.18 Name: SCREEN

Use: `!SCREEN screen name`

Description: Invokes the selection screen from which the user may choose a valid response. The screen may provide narrative descriptions of desired actions which may be selected by the simple entry of a one or more character strings. Only valid responses are accepted. PC users may optionally use a mouse to make selections.

Options:

`V` - Force screen as visible regardless of other controls.
`I` - Force screen as invisible regardless of other controls.
(only a prompt will be displayed)
`D` - Delete saved PUF screen name. This option is only valid on the PC.

Example:

`!screen main`	
`!=SCREEN`	List names of available screens
`!screen.v helpscn` `!SCREEN BLUERES`	Only screen prompt line will be displayed
`!Scr fast|`	(MS DOS only) vertical bar as end character of screen name causes screen to be saved for faster display on subsequent uses.
`!screen.d faster|`	(MS DOS only) delete "fasterl" screen from saved area so that changes in imbedded function characters can change screen display

2.19 Name: SET/GET

Use: `!SET STATUS keyword OFF | ON`
`!GET STATUS keyword function character`

Description: The SET/GET commands allow getting and setting various control information. STATUS information on current settings as displayed in the STATUS command may be stored in functions with the GET command. Valid keywords for STATUS requests are: FUNCTION, ECHO, LOG, MENU, AUTOSCREEN, MACRO, BREAK, and LEARN. Changing MACRO or LEARN by use of the SET command is not valid. The SET command can be used to set values of the same control flags to either ON or OFF.

Example:

```
!SET STATUS FUNCTION OFF
!GET STATUS ECHO &
!set STATUS ECHO &
!SET TERMINAL TYPE 4107
```

2.20 Name: STATUS

Use: `!STATUS`

Description: The status command displays the current status of each of the capabilities of PREAD.

Example:

```
!status
```

2.21 Name: TEACH

Use: `!TEACH function string`

Description: Defines, re-defines or removes a function character. Functions of zero length may be defined, whereby all occurrences are replaced with nothing. The maximum length of an individual function string is 30 characters on the PC and 80 characters on other systems.

Example:

`!teach @ Test`	(Define the at sign @ to mean "Test")
`!-TEACH #`	(Remove the definition for the pound # sign)
`!TEACH %`	(Define zero length function for %)

2.22 Name: WAIT

Use: `!WAIT number of seconds`

Description: The wait command forces the macro to wait the specified number of seconds before proceeding. This is useful when a continuous display of information is made and time is needed to allow the screen to be read as the macro executes.

Option: `X` - Exits all macros running if a user enters an "X" character during the wait.

Example:

```
!WAIT 15
!wait.x 10
```

2.23 Name: * Command

Use: `!* [comment]`

Description: An * as the first character following the command character signifies that the line is a comment. Comments will not be passed to the program.

Example:

`!*` This line is a comment

2.24 Name: ? Command

Use: `!?`

Description: The entry of a ? will display a table of allowable PREAD commands.

APPENDIX A

FUNCTION FILE

The function file contains the definition of the function shift character and each function character that has been defined. It is a normal text file in the following format.

Line 1 - the first character of line 1 is the function shift character. The recommended character is the caret. The line is read as an A1 format. The caret character "^" is the default function shift character.

The remaining entries occur in pairs of lines. The first line of the pair contains the character which is to be treated as a function and a numeric integer count of how many characters of the next line are included in the function reference. This line is read in free format. The second line of the pair contains the string of zero or more characters that are the function reference value. Two lines are used for each function to be defined.

Sample Function File

```
^
#     3
ABC
$    14
THIS IS A LINE
@     4
XX
```

Note: The @ character is defined as a 4 character string ending with 2 blanks (ie, "XX ").

APPENDIX B

MACRO FILE

The macro file contains information about the PREAD command character, the initial bootstrap macro and the definition of each macro.

An optional entry to redefine the PREAD command character from its internal default of exclamation point ! to another character may be entered as the first line of the macro file. The format is the characters CC in Columns 1-2, a space in Column 3 and the new PREAD command character in Column 4.

A second optional entry to define a BOOTSTRAP macro may be entered at the beginning of the macro file. If used, it must immediately follow the CC line, or if no CC line is used it must be the first line of the file. The characters BOOTSTRAP must appear beginning in Column 1 and are case sensitive. If present, the line immediately following BOOTSTRAP will be taken as a command line to PREAD and be executed by PREAD when a program first begins execution. Note that only the first line will be executed and if additional lines follow they will be ignored. Most typically this line would be a RUN command, such as, !RUN BEGIN. Macro BEGIN would be executed which may contain several other commands.

The remaining lines of the macro file are macro definitions.

Macro Syntax:

```
MACRO  name [parameter 1, parameter 2, ...]
. . . .
. . . .
. . . .
. . . .
ENDMACRO
```

Each definition begins with the line MACRO name where MACRO must begin in column 1 and the 1 to 8 character name must begin in column 7. Subsequent lines are lines to be executed when the macro is run. The macro definition is terminated by an ENDMACRO line beginning in column. 1. The macro name line may optionally contain one or more parameter strings. If a parameter string is used, each occurrence of the string will be replaced in the macro definition by the parameter used at execution time. The examples below illustrate the use of parameters passed to macros.

The keywords MACRO, ENDMACRO, and the macro name are all case insensitive. Old versions of software may require these to be upper case.

B.1 Sample Macro File with BOOTSTRAP

```
BOOTSTRAP
!run begin

MACRO begin
!-ECHO
TEST
A B C
endmacro

macro aBC
INPUT
25 29 59 10.31
endmacro
```

B.2 Sample Macro with Parameters

1. Macro execution line:

```
!Run Plotit RED STAGE FORECASTED
```

2. Macro definition:

```
Macro PLOTIT $LOCATION $PARAMETER $VERSION
PATH /ARKANSAS/$LOCATION/$PARAMETER/1DAY/01JAN1988/$VERSION/
PATH /ARKANSAS/$LOCATION/$PARAMETER/1DAY/01JAN1988/OBS/
Endmacro
```

3. Macro effect:

```
PATH /ARKANSAS/RED/STAGE/1DAY/01JAN1988/FORECASTED/
PATH /ARKANSAS/RED/STAGE/1DAY/01JAN1988/OBS/
```

APPENDIX C

SCREEN FILE

The screen file defines one or more screens that may be invoked by a program. A screen is composed of the material to be written to the users screen to describe the options available, a list of valid responses and the corresponding operations to perform to accomplish the response. Some additional control information describes the size of the areas, whether or not to clear the screen before and/or after displaying the screen (see action flag below) , where to position the cursor after the screen is displayed, and where to write error messages.

Elements of the screen definition are contained on lines beginning with a '#' sign. Any number of screens may be defined in the same file. The column positions of the entries defining screens must be strictly adhered to. The keywords following the # are case insensitive. Screen names are case insensitive.

The width of screens is limited to 80 columns.

Explicit function references may be placed anywhere in the text portion of the screen. The function value is substituted immediately before the screen is written. Certain control values may also be represented by function references.

C.1 - Screen definition line

Example:

```
|---5----|---15----|---25----|---35----|---45----|---55----|-

        Namexxxx Sc Sl Dl Dc P A Tl Tc E Tim V Al Ai M

#SCREEN ORD      38 18 03 23 1 1 11 50 5 060 5 07 02 2
```

The first line of a screen must be a screen definition line containing;

a) '#SCREEN' (columns 1-7),

b) one to eight character screen name (columns 9-16) If you wish to save the screen window characters and attributes to a PUF file, make certain that the screen name has a "|" as its last character. This option is only valid on the PC and is of value because subsequent reuse of the screen will be restored directly from memory instead of being put out a line at a time. For additional information, see SCREEN command "D" option.

c) Sc - number of columns required to define screen text area (columns 18-19),

d) Sl - number of lines required to define screen text area (columns 21-22),

* e) Dl - physical display line to write screen to (columns 24-25),

* f) Dc - physical display column to write screen to (columns 27-28),

* g) P - priority of screen for retention in buffer (columns 30),

* h) A - action flag, may be hexadecimal sum of items below (column 32)

1=	Before	-Force ANSI, clear screen
2=	After	-Force ANSI, clear screen
4=	Before	-Erase Graphics, Force ANSI, Clear screen
8=	Before	-Force ANSI(ANSI mode)
	After	-Force TEK

NOTE: Because this value may be a hexadecimal number, if a function character is used for this field avoid use of characters A through F as function character here.

i) Tl - total lines in TRANSLATE area (columns 33-35),

j) Tc - maximum columns in TRANSLATE area (columns 37-38).
If this value is zero, screen will be displayed but no response will be read.

* k) E - maximum sequential errors permitted before invoking response #SEQERR. If zero, no limit used. (column 40)

* l) Tim - inactivity time limit before invoking response #INACT (columns 42-44). If zero, no limit used.

* m) V - visibility level of screen (column 46). If zero, screen always visible.
(This feature not currently implemented.)

n) Al - number of lines in ATTMAP area (column 48-49). Lines are taken as same length as screen lines (item c) above.

* o) Ai - column offset index in ATTMAP line for current attributes (columns 51-52). Must point to sequence beginning with = sign. If zero, attributes will not be used.

p) M - mouse flag (column 54). If zero, no mouse areas defined; if 1, mouse areas defined with mouse inactive; if 2, mouse areas defined with mouse active. Mouse capability limited to PC screens only.

* Entry may be represented by explicit function reference. (Any non-numeric value in right position of field treated as function reference, explicit function shift character (i.e., ^) optional).

C.2 - Screen text to be displayed to user

The next lines contain an exact image of the screen that should be written to the users screen. It must be of the width and length given in C.1c and C.1d above.

Example:

```
    Ohio River Division

 H  -  Huntington District

 L  -  Louisville District

 N  -  Nashville District

 P  -  Pittsburgh District


 X  -  Exit

Enter Location or 'X' to Exit:
```

C.3 - Screen attributes to be displayed under text (Optional)

The #ATTRIBUTES defines the beginning of the text attribute area. The next lines contain an exact image of the screen attributes that should be written to the users screen. It must be of identical size and layout as the screen text above. Characters used in this layout area that are not given a definition in the ATTMAP area are ignored.

Example:

```
#ATTRIBUTES
    HHHHHHHHHHHHHHHHHHHH

 SSS   IIIIIIIIIIIIIIIIIIIII

 SSS   IIIIIIIIIIIIIIIIIIIII

 SSS   IIIIIIIIIIIIIIIIIIIII

 SSS   IIIIIIIIIIIIIIIIIIIII


 SSS   IIIIII
```

C.4 - Attribute map area (ATTMAP)

This area defines the meaning of each character used in the attribute layout above. Any character may be used to define an attribute, even the space character. The meaning of that character may be multi-defined. The definition to be used is pointed to by the screen definition line columns 51-52. This allows attributes for monochrome, and several versions of color to be available for any screen. The attribute definition uses ANSI attribute parameters.

ANSI Parameter	Meaning
00	All attributes off
01	Bold on
04	Underscore on
05	Blink on
07	Reverse video on

Foreground	Background	Color
30	40	Black
31	41	Red
32	42	Green
33	43	Yellow
34	44	Blue
35	45	Magenta
36	46	Cyan
37	47	White

Example:

```
#ATTMAP
 =00+44+37+1          =00
H=00+44+37            =00+01
I=00+44+37+1          =00+01
S=00+47+30            =00+07
P=00+41+30            =00+07
N=00+44+37+1          =00
Z=00                  =00
```

C.5 - Prompt location

After the screen is displayed to the user the cursor will be moved to the screen location defined by the `#PROMPT` line (relative to base location of the screen). If attributes are to be used, the attributes for the prompt area and the normal attribute after a screen is exited, follows the prompt position. If a mouse is to be used, its initial position may be given following the normal attribute.

a) '#PROMPT' (column 1-7),

b) row number of prompt cursor (columns 9-10).

c) column number of prompt cursor (columns 12-13).

d) attribute prompt character (column 15),

e) normal attribute character set after leaving screen (column 17),

f) row number of mouse cursor (columns 19-20),

g) column number of mouse cursor (columns 22-23).

Example:

```
#PROMPT 17,34 N Z 05,05
```

C.6 - Message location

If an invalid response is chosen by the user, the error response will be written to the screen location defined (relative to base location of the screen). If attributes are to be used, the message attribute follows the message location.

a) '#MESSAGE' (column 1-8),

b) row number of message location (columns 10-11).

c) column number of message location (columns 13-14).

d) message attribute character (column 16),

Example:

```
#MESSAGE 14,05 P
```

C.7 - Response list and translation area.

The response list and translation area contains a list of all valid responses, and the action to be taken when a response is entered.

The first line of the TRANSLATE area must contain:

a) '#TRANSLATE' (column 1-10),

b) number of total lines in TRANSLATE area (columns 11-13).
(Total lines may now exceed previous limit of 99).

c) optional type-ahead buffer flush indicator, FLUSH (columns 15-19)
If the optional FLUSH parameter appears on this line the system type-ahead buffer will be flushed before accepting response from this screen. This may be used on error recovery, or other critical screens where type-ahead may not be appropriate.

The remaining lines define valid responses:

c) number of lines executed for this response (column 1), must be in range 1 to 9,

d) response control character (column 2), control how rigid response must be,
if ' ' > abbreviation permitted, case sensitive,
if ':' > abbreviation prohibited, case sensitive, (must be exact)
if ',' > abbreviation permitted, case insensitive, (most relaxed)
if ';' > abbreviation prohibited, case insensitive.

e) 'response' one to eight characters (column 3-10),

f) action to 'response' (column 12-mm), where mm= is maximum column given in C.1j

g) optional, if col 54 of `#SCREEN` line is 1 or 2, a rectangular hit area is specified for mouse pick to be equivalent to keyboard response. Format is [r1,c1,r2,c2] where:
r1,c1 define upper left corner of hit box, and
r2,c2 define lower right corner of hit box.

If a response of `#ANY` appears as the last entry in the translate area, it will always be used if no previous keyboard responses match.

If control characters are desired as valid responses they may be represented in the translate area by three character sequences of ^nm, where the nm is the ANSI 2-character representation of the control character. (e.g., `^CR`, `^BL`, `^EC`).

Example:

```
#TRANSLATE 11
1;H         !RUN ORD H      [06,03,06,38]
1;L         !RUN ORD L      [08,03,08,38]
1;N         !RUN ORD N      [10,03,10,38]
1;P         !RUN ORD P      [12,03,12,38]
1,X         !RUN FINISH     [15,03,15,13]
1;EXIT      !RUN FINISH
1:^CR       !RUN FINISH
1:XxYy?     !SC ORD         [01,01,18,38]
1:#SEQERR   !RUN ERRORS
1:#INACT    !RUN TIMEOUT
1;OP        !SC  OPTION1|
#ENDSCREEN
```

C.8 - End of Screen

The end of the screen is defined by '#ENDSCREEN' (columns 1-10).

C.9 - Full Example Screen

```
|---5----|---15----|---25----|---35----|---45----|---55----|-

        Namexxxx Sc Sl Dl Dc P A Tl Tc E Tim V Al Ai M

#SCREEN ORD      38 18 03 23 1 1 11 50 5 060 5 07 02 2
```

```
      Ohio River Division

  H  -  Huntington District

  L  -  Louisville District

  N  -  Nashville District

  P  -  Pittsburgh District


  X  -  Exit

 Enter Location or 'X' to Exit:
```

```
#ATTRIBUTES
```

```
      HHHHHHHHHHHHHHHHHHH

  SSS   IIIIIIIIIIIIIIIIIIII

  SSS   IIIIIIIIIIIIIIIIIIII

  SSS   IIIIIIIIIIIIIIIIIIII

  SSS   IIIIIIIIIIIIIIIIIIII


  SSS   IIIIII
```

```
#ATTMAP
 =00+44+37+01          =00
H=00+44+37             =00+01
I=00+44+37+01          =00+01
S=00+47+30             =00+07
P=00+41+30             =00+07
N=00+44+37+01          =00
Z=00                   =00
#PROMPT 17,34 N Z 05,05
#MESSAGE 14,05 P
#TRANSLATE 11
1;H          !RUN ORD H      [06,03,06,38]
1;L          !RUN ORD L      [08,03,08,38]
1;N          !RUN ORD N      [10,03,10,38]
1;P          !RUN ORD P      [12,03,12,38]
1,X          !RUN FINISH     [15,03,15,13]
1;EXIT       !RUN FINISH
1:^CR        !RUN FINISH
1:XxYy?      !SC ORD         [01,01,18,38]
1:#SEQERR    !RUN ERRORS
1:#INACT     !RUN TIMEOUT
1;OP         !SC  OPTION1|
#ENDSCREEN
```

APPENDIX D

SAMPLES

D.1 Sample macro file

```
BOOTSTRAP
!RUN BEGIN

-----------------------------------------------------------------------

MACRO BEGIN
!IF ( '^K' .EQ. 'COPY-IT' ) THEN
!JCL COPY REPORT ^F >NUL
!TEACH K NULL
!PAGE
!RUN REPVUE ^B ^T CHOICES
!SC ^D
!ENDIF
!SC MAIN
FIN
ENDMACRO

-----------------------------------------------------------------------

MACRO REPORT $DIST
!TEACH D $DIST
!IF ( '^D' .EQ. 'ORP' ) THEN
!TEACH A  P
!SC ORP
!ELSEIF ( '^D' .EQ. 'ORH' ) THEN
!TEACH A  H
!SC ORH
!ELSEIF ( '^D' .EQ. 'ORL' ) THEN
!TEACH A  L
!SC ORL
!ELSE
!SC ORD
!ENDIF
!SC MAIN
ENDMACRO

-----------------------------------------------------------------------

MACRO SITE $L $NAME $ID $T $NXT
!IF ( '$L' .NE. '---' ) THEN
!TEACH L '$L'
!TEACH N '$NAME'
!ENDIF
!IF ( '^L' .NE. '---' ) THEN
!RUN TYPE-^T
!SC CHOICES
!ENDIF
!SC $NXT
ENDMACRO
```

D.2 Example macro file from IA5/MOD5/HEC5 workshop

```
BOOTSTRAP
!RUN BEGIN

MACRO BEGIN
!RUN SAVEPO
!-FUNCTION
^z
TIME 0100 20JAN59 2400 23JAN59
GR,7
!* SH,ON
!-ECHO
!RUN MAP
FINISH
ENDMACRO

MACRO WC
US Worthington (CP #268)
PA  /SCIOTO/WOOE2/FLOW-CHAN CAP/01JAN1959/3HOUR/^r^s^p5^o/
PA  /SCIOTO/WOOE2/FLOW-REG/01JAN1959/3HOUR/^r^s^p5^o/
PA  /SCIOTO/WOOE2/PRECIP-INC/01JAN1959/3HOUR/^r^s^p/
PA  /SCIOTO/WOOE2/FLOW-LOC CUM/01JAN1959/3HOUR/^r^s^p5^o/
PA  /SCIOTO/WOOE2/FLOW-FLOOD RES/01JAN1959/3HOUR/^r^s^p5^o/
TY 1 1 2 1 1
SW BLN TRI
SP,75
PLOT.A
!SC OPTIONS
ENDMACRO

MACRO CC
US Scioto River at Columbus (CP #275)
PA  /SCIOTO/CLSF4/FLOW-CHAN CAP/01JAN1959/3HOUR/^r^s^p5^o/
PA  /SCIOTO/CLSF4/FLOW-REG/01JAN1959/3HOUR/^r^s^p5^o/
PA  /SCIOTO/CLSF4/PRECIP-INC/01JAN1959/3HOUR/^r^s^p/
PA  /SCIOTO/CLSF4/FLOW-LOC CUM/01JAN1959/3HOUR/^r^s^p5^o/
PA  /SCIOTO/CLSF4/FLOW-FLOOD RES/01JAN1959/3HOUR/^r^s^p5^o/
TY 1 1 2 1 1
SW BLN TRI
SP,75
PLOT.A
!SC OPTIONS
ENDMACRO
```

D.3 Example screen file MODCON/IA5/MOD5/HEC5 Workshop

```
#SCREEN MAP      80 20 04 01 1 2 20 35
 -------------  Scioto Basin Locations  ------------------

    Delaware Res (D)
          |
          |
   Worthington (W)     Alum Ck Res (A)
          |                |
          |                |
  Columbus (C)        E. Col-Alum Ck (E)
          |                |
          |                |------ Hoover Res (H)
          |                |
          |------------ Rees (R)
          |                                  Precipitation Alt =  ^p
          |                                  Operations Alt = ^o
  Circleville (CV)                           ( Enter P to change Precipitation, )
          |                                  (    or Operations Alternative     )
          |

 Select location, or X to Exit ==>
#PROMPT 20,36
#MESSAGE 19,02
#TRANSLATE 20
2 D         !RUN DR
            !RUN MAP
2 W         !RUN WC
            !RUN MAP
2 C         !RUN CC
            !RUN MAP
2:CV        !RUN CVC
            !RUN MAP
2 A         !RUN AR
            !RUN MAP
2 E         !RUN EC
            !RUN MAP
2 H         !RUN HR
            !RUN MAP
2 R         !RUN RC
            !RUN MAP
2 X         !RUN RESTORPO
            FINISH
2 P         !RUN OPTSET
            !RUN MAP
#ENDSCREEN
```

```
#SCREEN OPT      79 18 05 01 1 2 09 25
                Change Precipitation or Operations Alternative
                ----------------------------------------------

                P-A    Select Precipitation Alternative 'A'
                P-B    Select Precipitation Alternative 'B'
                P-C    Select Precipitation Alternative 'C'
                P-D    Select Precipitation Alternative 'D'

                O-A    Select Operations Alternative 'A'
                O-B    Select Operations Alternative 'B'
                O-C    Select Operations Alternative 'C'
                O-D    Select Operations Alternative 'D'

                X      Exit


                Select desired option ==>

#PROMPT 17,42
#MESSAGE 16,16
#TRANSLATE 09
1:P-A       !TEACH p A
1:P-B       !TEACH p B
1:P-C       !TEACH p C
1:P-D       !TEACH p D
1:O-A       !TEACH o A
1:O-B       !TEACH o B
1:O-C       !TEACH o C
1:O-D       !TEACH o D
1 X         !PR Exit
#ENDSCREEN
```

```
#SCREEN OPTIONS  79 03 01 01 1 9 24 25
 Enter X, option, or ? for Help
 ==>

#PROMPT 02,06
#MESSAGE 01,35
#TRANSLATE 24
2 ^CR       PLOT
            !SC OPTIONS
3 S         SHADE,ON
            PLOT
            !SC OPTIONS
3 SN        SHADE,OFF
            PLOT
            !SC OPTIONS
3 G         GRID,ON
            PLOT
            !SC OPTIONS
3 GN        GRID,OFF
            PLOT
            !SC OPTIONS
2 W         W
            !SC OPTIONS
3 HELP      !SC HELP
            PLOT
            !SC OPTIONS
3 ?         !SC HELP
            PLOT
            !SC OPTIONS
2 X
            !SC CLEAR
#ENDSCREEN
```

```
#SCREEN HELP     80 20 01 01 1 7 01 25

                          DSPLAY - Help Screen
                       ----------------------------

     After completing the plot the user may exercise several options.

         Option            Remarks
         ------            -------
          X                Exit plot and continue
          S                Shade
          SN               Shade-No
          G                Grid
          GN               Grid-No
          W                Window the plot area by setting cross-hairs
                           (After the cross-hairs are displayed move to
                           the desired left side and press W, then
                           move to the desired right side and press
                           W again.)

          Press <RETURN> to restore original plot.
#PROMPT 20,10
#MESSAGE 21,03
#TRANSLATE 01
1 ^CR      !PAGE GRAPHICS
#ENDSCREEN
```

D.4 Example macros using parameters

```
MACRO RIVDISP APART BPART CPART FPART DOIT NEXT1
RESET
PA /APART/BPART/FLOW/01JAN1988/1DAY/FPART/
PA /APART/BPART/STAGE/01JAN1988/1DAY/FPART/
!RUN DOIT APART BPART
!SCREEN NEXT1
ENDMACRO

MACRO RESDISP APART BPART CPART FPART DOIT NEXT1
RESET
PA /APART/BPART/STORAGE/01JAN1988/1DAY/FPART/
!RUN DOIT APART BPART
!SCREEN NEXT1
ENDMACRO

MACRO HELP NAME NEXT1
REP XINDEX=`H-DSP.X
REP NAME
!SCREEN NEXT1
ENDMACRO

MACRO SETPLOT NEXT1
!TEACH $ PLOTIT
TIME &
!SCREEN NEXT1
ENDMACRO

MACRO SETTAB NEXT1
!TEACH $ TABIT
TIME T-10D T+30D
!SCREEN NEXT1
ENDMACRO

MACRO #ABORT
!PR
!PR - - Aborting - -
!PR
!WAIT 2
ABORT
ENDMACRO
```