

# **PROGRAMMERS MANUAL FOR THE WATERSHED DATA MANAGEMENT (WDM) SYSTEM**

By John L. Kittle, Jr., AQUA TERRA Consultants  
Kathleen M. Flynn, U.S. Geological Survey  
Paul R. Hummel, AQUA TERRA Consultants, and  
Alan M. Lumb, U.S. Geological Survey

---

This is a draft document. It should not be quoted or cited as a publication. The WDM system works essentially as documented here. There may be some errors in this documentation. In addition, there may be corrections and refinements made to the program before final publication.

U.S. GEOLOGICAL SURVEY  
Water-Resources Investigations Report 91-xxxx

Reston, Virginia

1991

DEPARTMENT OF THE INTERIOR

MANUEL LUJAN, JR., Secretary

U.S. GEOLOGICAL SURVEY

Dallas L. Peck, Director

---

For additional information  
write to:

Chief, Office of Surface Water  
U.S. Geological Survey  
415 National Center  
12201 Sunrise Valley Drive  
Reston, Virginia 22092

Copies of this report can be  
purchased from:

U.S. Geological Survey, Books  
and Open-File Reports Section  
Box 25425, Federal Center  
Building 810  
Denver, Colorado 80225

## CONTENTS

ABSTRACT .....
ACKNOWLEDGMENTS .....
INTRODUCTION .....
DEFINITION.....
WHY USE WDM.....
MANUAL CONTENTS.....
HOW TO USE THIS MANUAL.....
OVERVIEW .....
BACKGROUND.....
OVERVIEW OF WDM FILE AND WDM TOOLKIT.....
SYSTEM REQUIREMENTS AND CONVENTIONS.....
LIMITATIONS.....
RELATED SOFTWARE AND DOCUMENTATION.....
AIDE .....
ANNIE .....
ANNIE-WDM Library Reference Manual .....
PROGRAMS USING WDM.....
BASIC CONCEPTS .....
WDM FILE CHARACTERISTICS AND STRUCTURE.....
Data Sets .....
Attributes .....
Groups .....
Blocks .....
TYPES OF DATA STORED.....
Time-series Data Sets.....
Space-time Data Sets.....
Vector Data Sets.....
Text and Tables Template Data Sets .....
Table Data Sets .....
Attribute Characteristic Data Sets .....
RECORD BUFFERS.....
POINTERS.....
Data Set Pointers .....
Record Pointers .....
Group Pointers .....
CASE STUDIES .....
PREREQUISITES TO RUNNING THE CASE STUDY.....
GENERAL WDM HOUSEKEEPING.....
TIME-SERIES DATA.....
VECTOR DATA.....
TEXT AND TABLES TEMPLATE DATA.....
TABLE DATA.....
ATTRIBUTE CHARACTERISTIC DATA.....
PROGRAMMERS TOOLKIT .....
WDM FILE MANIPULATION.....
Open a WDM File .....
Close a WDM File .....
DATA SET MANIPULATION.....
Add a Label for a New Data Set .....

Add a Label for a New Data Set With Special Sizing .....
Check Data Set Existence and Type .....
Check Data Set Existence and Return First Record .....
Copy Old Data Set Label Into New Data Set Label .....
Renumber a Data Set .....
Delete a Data Set .....
<b>ATTRIBUTE MANIPULATION.....</b>
Add Character Search Attribute .....
Add Integer Search Attribute .....
Add Real Search Attribute .....
Get Values for Character Search Attribute .....
Get Values for Integer Search Attribute .....
Get Values for Real Search Attribute .....
Delete a Search Attribute .....
<b>TIME SERIES OPERATIONS.....</b>
Get Time-series Data .....
Put Time-series Data .....
Increment a Time Array .....
<b>SPACE-TIME OPERATIONS .....</b>
Add a Space Time Group .....
Summarize a Space Time Group .....
Get Real Space Time Data .....
Put Real Space Time Data .....
<b>VECTOR OPERATIONS.....</b>
Get Vector Data .....
Summarize Vector Data .....
<b>TEXT AND TABLE TEMPLATES OPERATIONS.....</b>
Get First Block Control Word in a Group .....
Split a Message Block Control Word .....
Get one Record of Text From WDM File.....
Move to Next Data Position in Text Data Set .....
Skip Within Text Data Set .....
<b>TABLE OPERATIONS.....</b>
Put Table Template on WDM File .....
Get Table Data From WDM File .....
Convert Table Data to Internal Format .....
Put Table Data Into a Table Data Set .....
Get Table Label Information From a Table Data Set .....
Determines Pointer to Specified Table .....
Split Table Dimension Variable .....
Split Table Identifier Variable .....
Delete Table From a Table Data Set .....
<b>ATTRIBUTE CHARACTERISTICS OPERATIONS.....</b>
Find Search Attributes Which Match Name .....
Get General Information About Search Attribute .....
Get Search Attribute Description .....
Get Search Attribute Help Location .....
Get Search Attribute Default Value .....
<b>REFERENCES .....</b>
<b>APPENDICES</b>
<b>A. WDM FILE FORMATS .....</b>
A.1 File Definition Record .....
A.2 Directory Record .....
A.3 Time-Series Data Set .....
A.4 Table Data Set .....
A.5 Vector Data Set .....
A.6 Space-Time Data Set .....
A.7 Attribute Characteristics Data Set .....
A.8 Text and Tables Template Data Set .....

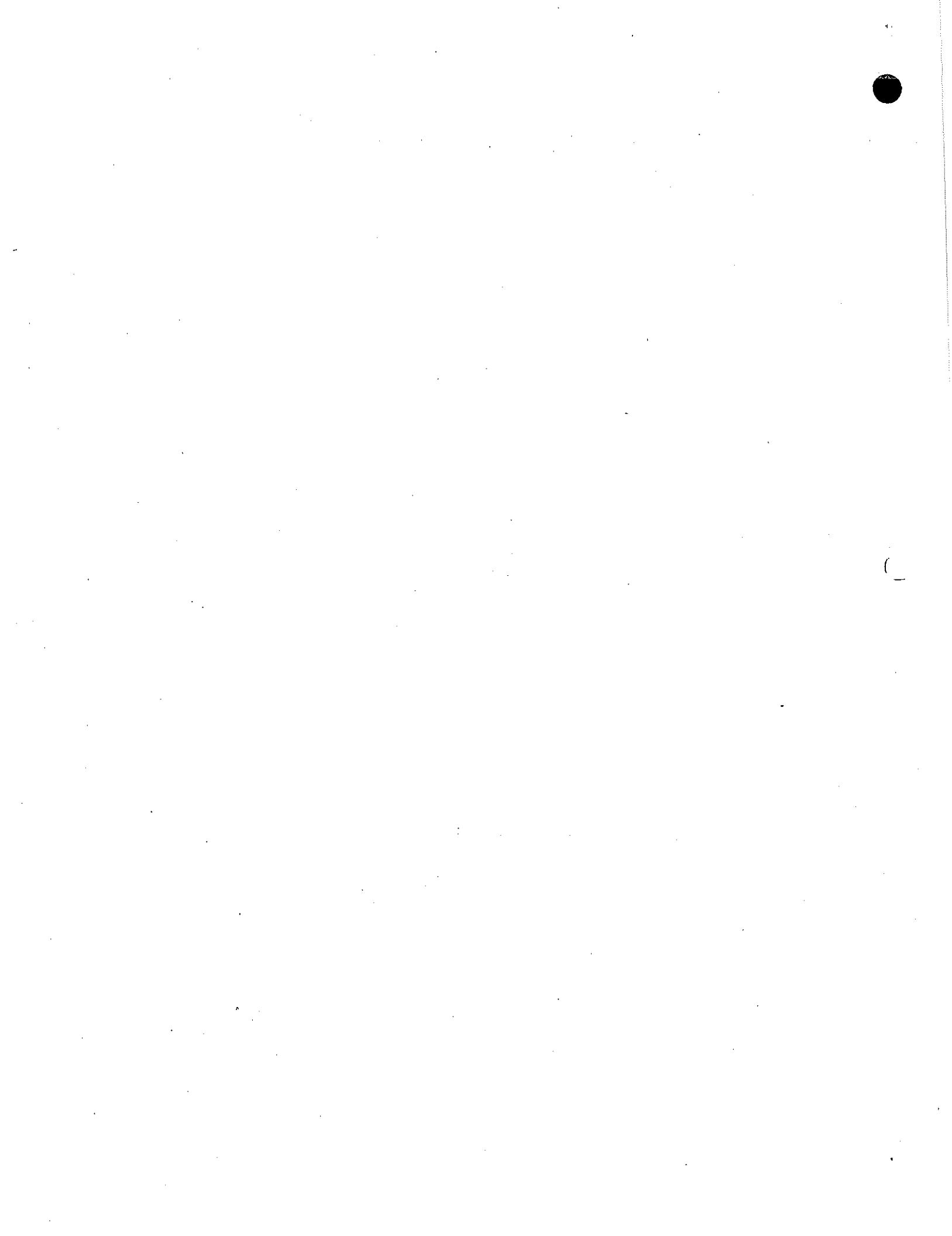
B.	IMPORT/EXPORT SEQUENTIAL FILE FORMATS .....
C.	ATTRIBUTES .....
C.1	Attributes and Their Characteristics.....
C.2	Attribute Names Sorted by Index Number.....
C.3	Required Attributes by Data Set Type.....
D.	DESCRIPTION OF REFERENCED SUBROUTINES .....
E.	DATA STRUCTURES - COMMON BLOCKS .....
F.	RETURN CODES .....
G.	WDM FILE MAINTENANCE PACKAGE .....
H.	INDEX TO SUBROUTINES .....
	GLOSSARY .....

## FIGURES

3.1	Basic Structure of the WDM File .....
-----	---------------------------------------

## TABLES

## ACKNOWLEDGMENTS



# **PROGRAMMERS MANUAL FOR THE WATERSHED DATA MANAGEMENT (WDM) SYSTEM**

By John L. Kittle<sup>1</sup>, Kathleen M. Flynn<sup>2</sup>, Paul R. Hummelt<sup>1</sup>, and Alan M. Lumb<sup>2</sup>

## **ABSTRACT**

(Original) This document is a guide to the computer software called WDMS - the Watershed Data Management System. The system includes a binary, direct access file for storage of hydraulic, hydrologic, and water-quality data and a toolkit of subroutines that enables expedient input, update and output of stored data. The WDM storage file features the ability to store the of data needed to perform water resources investigations including time series data, tables, text and vectors in a single storage mechanism. WDM files allow comprehensive specification of data attributes, multiple time steps in single time-series data sets, user-defined formats for data stored as tables, compression of data, improved speed and flexibility of interaction between a model and its data base, and automatic file maintenance as the user adds, modifies or deletes files. Use of the WDM file and the WDM toolkit subroutines provides an efficient means for storing the output from one model and providing it as input to a second model. WDMS allows multiple models, requiring different data types and data with different time steps, to use a common data base.

The programmers manual and its appendices provide the application programmer with all the information needed to use the WDM file and toolkit to provide data base management and interaction for any process or model code written in Fortran or capable of linking Fortran subroutines.

(KF) The Watershed Data Management (WDM) System provides a mechanism for managing the kinds of data needed to perform water resources investigations including time-series data, tables, text, and vectors in a single storage mechanism. The WDM file is a binary, direct access file for the storage of hydraulic, hydrologic, and water-quality data. The WDM toolkit of routines enables expedient input, update, and output of stored data. WDM files allow comprehensive specification of data attributes, multiple time steps in single time-series data sets, user-defined formats for data stored as tables, compression of data, improved speed and flexibility of interaction between a model and its data base, and automatic file maintenance as the user adds, modifies or deletes files. Use of the WDM file and the WDM toolkit of routines provides an efficient means for storing the output from one model and providing it as input to a second model. The WDM system allows multiple models, requiring different data types and data with different time steps, to use a common data base.

This document provides a programmer with all of the information needed to use the WDM system. The subroutines and functions contained in the WDM toolkit are documented and their arguments are defined. The purpose and use of the many commonly used routines are described with example programs and explanations. The format of the WDM file is described in detail.

---

<sup>1</sup> Aqua Terra Consultants

<sup>2</sup> U.S. Geological Survey

# INTRODUCTION

(Original) Data storage and retrieval is a critical step in the efficient application of most water resources models. Satisfying the data needs within and between models can be one of the most difficult and time-consuming components of model application. Given the scientific and engineering complexity of many environmental problems, analysis often warrants application of one or more data-intensive models, with each model having unique data requirements. Various types of data (e.g., time series, vectors, tables) are commonly used by models, and each data type has different requirements for efficient storage and model use. The Watershed Data Management (WDM) file provides the means to store in a single file the data needed to perform environmental analyses. Use of the WDM file allows multiple models, requiring different data types and data with different time steps, to use a common data base.

In this introductory section, the WDM file and WDM toolkit are defined; potential users of the WDM system are identified; reasons for using the WDM file and toolkit are outlined; the manual's contents are previewed; and instructions for effective use of the manual are provided.

(KF) The scientific and engineering complexity of many environmental problems often warrant the application of one or more data-intensive models, with each model having unique data requirements. Data storage and retrieval is a critical step in the efficient application of most water-resources models. Satisfying the data needs within and between models can be one of the most difficult and time-consuming components of model application. Various types of data (eg., time series, vectors, tables) are commonly used by models. Each data type has different requirements for efficient storage and retrieval. The Watershed Data Management (WDM) system provides the WDM file for data storage and a library of routines called the WDM toolkit for writing to and reading from the file. Use of the WDM toolkit and file allows multiple models, requiring different data types and data with different time steps, to use a common data base

This introductory section defines the WDM file and toolkit. The expected programming audience is identified. Reasons for using, or not using, the WDM system are outlined. The manual's contents are previewed and an approach for the effective use of the manual is provided.

## DEFINITION

(Original) The Watershed Data Management (WDM) file and associated subroutines provide a systematic approach to the storage and retrieval of data required to operate hydrologic, hydraulic, and water-quality models. The system uses a well-defined binary, direct-access file structure accessed through a set of subroutines, called the Watershed Data Management (WDM) toolkit, to create a file, add data to the file, replace data in the file, get data from the file and delete data from the file. Four categories of data may be stored in a WDM file: time series, tables, text, or vectors.

(AL) The Watershed Data Management (WMD) System comprises the WDM file structure and WDM toolkit.

(KF) The Watershed Data Management (WDM) system uses a well-defined binary,

direct-access file structure accessed through a toolkit of routines to create a file, add data to the file, replace data in the file, get data from the file and delete data from the file. Four categories of data may be stored in a WDM file: time series, tables, text, or vectors. The file and toolkit provide a systematic approach to the storage and retrieval of data required to operate hydrologic, hydraulic, and water-quality models.

## WHO SHOULD USE WDM

The WDM system can be used to support process or model codes that are either written in Fortran or can link Fortran routines. It is clearly advantageous to use the WDM System in conjunction with models and process codes that provide input to, or receive output from, other models and processes that already use the WDM file. Software that currently uses the WDM file is described later in the section on related software and documentation (p xxx). In addition, there are strong reasons for using the WDM file and WDM toolkit to provide data storage and interaction for new models or models not directly connected to previous WDM applications. Programmers faced with one or more of the following needs or situations should consider using the WDM file and the WDM toolkit:

**Management of large volumes of data**

When large volumes of data are needed by the application program, the WDM file becomes most useful. For small amounts of data, ASCII flat files or spread-sheet software may be advantageous.

**Storage of, and interaction with, multiple data types**

When the application requires a mix of data types (e.g., tabular data for model parameter specification, time-series data for process modeling, and vector data for spatial representation), the WDM file can provide a single storage mechanism for all data types.

**Multiple data or model time steps**

The WDM file allows storage of data of a single type (e.g., stream flow), but of multiple time steps, within a single data set. Time-series data stored in the WDM file can be automatically aggregated or disaggregated to any time interval (seconds, minutes, hours, days, months, years) needed to satisfy the input requirements of a model retrieving data from the WDM file. Hence, one WDM data set, containing data of various time intervals, can be used to provide input to several models requiring the same input at different time intervals. Note that the aggregation and disaggregation procedures do not alter the contents of the stored data set; the original data values at their original time step always remain in storage.

**Anticipated use of multiple models**

Because of its capability to (1) store different data types, (2) store data of variable time steps, spatial dimensions, or table formats, and (3) aggregate or disaggregate time-series data to any time step required for use as input to a model, the WDM file excels as the central storage mechanism for heterogeneous data needed for application of numerous models for one or more water resources investigation(s).

## WHY USE WDM

(Original) Use of the WDM file and toolkit offer the following advantages:

(KF) In addition to the reasons mentioned above, there are a number of factors that can make the WDM file and toolkit advantageous to use. The expected user audience for a program and their expected computing environment may make the WDM system an attractive choice. Some of these factors are described here:

<b>Comprehensive storage</b>	Different types of data needed for water resources investigations can be stored in one data management structure.
<b>Ease of file use and maintenance</b>	WDMS features dynamic space allocation for new or expanding data sets and automatic file maintenance. Disk space for the WDM file is allocated as needed. Data can be added, modified, or deleted without restructuring the data in the file. Space from deleted data sets within a WDM file is reused. Thus, the WDM file does not require special maintenance processing.
<b>Direct access speed</b>	Compared to ASCII sequential files, use of binary direct access files significantly reduces the time required to read and process data.
<b>Portability</b>	Because the WDM toolkit utilizes strict coding conventions, the machine dependency of programs generated by its use is kept to a minimum, resulting in a high degree of portability to various hardware. The residual machine dependencies are identified in Section 2.4.
<b>Modularity and consistency</b>	WDMS utilizes a library of subroutines to perform basic data base management functions. Repeated use of modular subroutines decreases the likelihood of introducing programming errors. By approaching data base management as a repetitive application of a limited number of tools, a higher degree of programming consistency is maintained both within and between resulting programs.
<b>Compatibility with supporting software</b>	(Original) Preprocessing and postprocessing software for data stored in WDM data sets is already available in ANNIE (Lumb et al., 1990). Graphics software is available to plot much of the data in a WDM file (citation?).  (AL) Preprocessing and postprocessing software for data stored in a WDM file is already available in the program ANNIE (Lumb et al., 1990). ANNIE allows the user to interactively list, table, add, update, and plot data from any WDM file. Traditional hydrologic statistical analyses of time-series data is also available in ANNIE. Thus use of WDMS for a model reduces or eliminates the need to develop software for pre- and post-processing.
<b>Public domain software</b>	The WDM system requires no license for distribution. If the model or process to be developed will be widely distributed, avoiding licensing restrictions can become a considerable advantage.

## MANUAL CONTENTS

The next section of this manual provides background on the development of the WDMS; an overview of the WDM file and WDM subroutine toolkit; a description of the requirements and limitations of using WDMS; a brief description of related software that supports the use of WDMS; and a summary of software products already using the WDM file and WDMS toolkit subroutines. Section 3 describes the basic concepts of the WDM file structure, including details on data set types, data set organization, data attribute specification, and the necessary system of buffers and pointers that enables efficient data storage and retrieval. In Section 4, case studies are presented to illustrate applications of the WDM system. Each case study focuses on one of the six data set types that can be stored in the WDM file. Section 5 provides functional descriptions of the subroutines contained in the WDM toolkit. Information provided for each subroutine includes when to use it, what it does, prerequisites, examples, and input, modify, and output subroutine argument specifications.

The main body of the programmers manual is supported by substantial appendices (Appendices A through H). The appendices contain file formats for WDM records and data set types; file formats for sequential import and export files; definition of data set attributes; a summarial description of referenced subroutines; common blocks; subroutine return codes; a description of the WDM file maintenance package; and an index to WDM toolkit subroutines. A glossary of technical terms used in the programmers manual is provided at the end of the manual directly after the appendices.

## HOW TO USE THIS MANUAL

(AL) For the novice, the overview and basic concepts, Chapters 2 and 3, should be read first. Next the case studies, chapter 4, should be read. Initially attention should be given to sections 4.1, 4.2, and 4.3. As particular subroutines are referenced in those sections, they should be located in Chapter 5 and reviewed. The experienced user will tend to use just the Appendices with occasional references to chapters 4 and 5.

## MANUAL CONTENTS AND HOW BEST TO USE IT

(KF) A brief description of the various section in this manual is contained in table. xx.

Overview	Background on the development of novice the WDM system
Basic Concepts	details on data-set types, data-set organization, attribute specification, system buffers and pointers
Case Studies	Illustrates applications of the WDM system focusing on each of the 6 data-set types

Programmers Toolkit	Functional descriptions of the functions and subroutines in the WDM toolkit. Includes routine function, prerequisites, examples, and argument specifications	experienced user
Appendices	Detailed descriptions of file formats, attributes, toolkit, data structures, and return codes	experineced user

## OVERVIEW

---

This section provides background on the development of the WDM file and WDM toolkit; an overview of the system software; a description of requirements, conventions, and limitations; a brief description of related software that supports the use of WDM; and a summary of software products already using the WDM file and WDM toolkit subroutines.

## BACKGROUND

(AL) An ad hoc group of modelers from Federal agencies met in Denver in the spring of 1984 to discuss coordination of modeling activities for water-supply forecasts. One of the recommendations of the group was the development of a common data storage and retrieval system for hydrologic and hydraulic models. Four Federal agencies have expressed strong interest, and three agencies (USGS, EPA, and SCS) have contributed resources to design and implement the WDM file structure and software utilities toolkit.

## WDM FILE AND WDM TOOLKIT

(AL) The Watershed Data Management (WDM) file and the associated subroutine toolkit are designed to be a standard system to store, update, and retrieve time-series, drainage network, and basin characteristics data for hydrologic, hydraulic, and water-quality models.

A major premise for data management for water-resources models is that data comes in groups such as 10 years of daily streamflow, a grid of elevations for a watershed, a set of coordinates for a channel cross section, or a table of hydraulic properties for a channel. All or parts of one or more group may be needed by a model. And the groups must be identified for easy and logical retrieval by the user. The groups are called data sets, and the data set identifiers are called attributes.

Both the WDM file and associated software toolkit are portable and useful on most microcomputers with a minimum 10 megabytes hard disk, as well as on super microcomputers, minicomputers, and mainframe computers. The file system is space efficient with less than 50 percent overhead for most data. A WDM file has the potential for up to 30,000 data sets per file, yet it is also efficient and useful for only a few data sets. Each data set has a unique data-set number from 1 to 30,000 for pointing to the location of the data set without sequential searching.

For identification and retrieval, a WDM file stores from one to several hundred attributes for each data set.. Attributes may be easily added, modified, or deleted.

The user of time-series data has the option to place flags on the data to indicate quality of the data, whether the data is measured or estimated, periods of missing record, and so forth. Formats for constant or multiple time steps are available. Time steps from 1 second to 1 year are supported. Any string of time-series data with identical or near-identical values can be compressed on the file depending on a tolerance specified as an attribute for the data set. Pointers within a time-series data set can be set for century, year, month, day, or hour as appropriate.

## SYSTEM CONVENTIONS

All code used in the WDM toolkit has been developed with a coding convention adopted by the Office of Surface Water (Flynn et al., xxxx). If the applications programmer follows the coding conventions, the programmer's tool SYSDOC (Flynn et al., 1989) can be used to generate computer documentation that: (1) identifies all subroutines, functions, intrinsics, and common blocks; (2) lists the purpose of each; (3) defines each argument of each subroutine; (4) for each subroutine and function, identifies which subroutines use it and which subroutines it uses; and (5) identifies which common blocks and variables are used.

## LIMITATIONS

\*\*\*\*need functional limitations here\*\*\*\*

\*\*\*\*need hardware dependencies, if any, here\*\*\*\*

If the input and output time required by the application must be extremely fast, the WDM file may not be appropriate. Specialized real-time applications that must process large volumes of data may not find the WDM file adequate.

## RELATED SOFTWARE AND DOCUMENTATION

Related software that supports or supplements the use of WDMS includes the AIDE software for developing interactive user interfaces, the ANNIE data manipulation, display and analysis software, and the ANNIE-WDM library reference manual. The utility of each of these product is summarized below.

AIDE	The WDM toolkit routines contain no interactive capabilities. If interactive programming is desired, the WDM file and WDM toolkit can be used in conjunction with AIDE, the ANNIE Interaction Development Environment (Kittle et al., 1989). AIDE combines a toolkit of utility routines for building individual interactive screens with instructions for developing two parallel products: a file containing all text, questions, and messages used in interactive communication, and a Fortran program containing the control strategy and sequencing instructions for interactions. The AIDE user interaction toolkit uses WDM data sets to store all text, questions, and messages used in interactive communication between the program and user.
------	---

- ANNIE** By using the WDM file for hydrologic applications, all the functionality in ANNIE becomes available for interactively loading, modifying, updating, checking, listing, plotting, and tabling data. The ANNIE users manual (Lumb et al., 1990) describes the available functions.
- ANNIE-WDM Library Reference Manual** All the subroutines, functions and common blocks used by WDMS, AIDE and ANNIE have been documented in detail. The resulting document is entitled "Programmers Documentation for ANNIE: a Fortran Library for Interactive Hydrologic Analyses and Data Management" (Kittle et al., 1990).

## PROGRAMS USING WDM

Many models and statistical analysis software supported by the Office of Surface Water, U.S. Geological Survey, use the WDM file. Included among these software products are procedures for frequency analysis, flow-duration analysis, trend analysis, simple least squares, generalized least squares, data-collection network analysis, n-day high-flow and low-flow statistics, and base-flow separation. Current models utilizing the WDM file perform one-dimensional flow routing, convolution routing, rainfall-runoff, snowmelt, nonpoint pollution, sediment detachment and routing, reservoir routing, and instream water-quality modeling and transport. Interactive software is available for data adjustment and correction, file archiving, data plotting and tabling, and data queries.

In addition to the software supported by the USGS Office of Surface Water, a number of EPA software products also utilize the WDM file. The current version of HSPF, the Hydrological Simulation Program - Fortran (Johanson et. al., 1984) uses the WDM file for storage of input and output time-series data and table data sets containing values for input parameters. As mentioned in Section 2.5.1, the AIDE user interaction toolkit uses WDM data sets to store all text, questions, and messages used in interactive communication between a program and its user. AIDE has been used to produce interactive interfaces for the following EPA products: QUAL2E-UNCAS (Brown and Barnwell, 1987), a stream wasteload allocation program; MMM (citation?), a multimedia chemical fate and transport model; and DBAPE (Imhoff et al., 1990), an interactive framework for analyzing a national agricultural soils data base and estimating parameter values for unsaturated flow models.

# **BASIC CONCEPTS**

This section describes the basic concepts of the WDM file structure, including details on data-set types, data-set organization, data attribute specification, and the necessary system of buffers and pointers that enables efficient data storage and retrieval.

## **WDM FILE CHARACTERISTICS AND STRUCTURE**

WDM files are unformatted, direct-access files with fixed length records of 2,048 bytes. Export and import routines are available to convert a WDM file to an ASCII formatted, sequential file for archiving and transferring between computer systems. The current limit on the maximum size of the file is 8000 M bytes, and is based on the structure of the code. Physical device and system limitations usually determine the maximum size. Initial WDM file size starts at 20 records and expands in 20 record (40,960 byte) increments as needed.

Records in a WDM file are organized by data sets. Figure 3.1 illustrates the basic structure of the WDM file. As the figure indicates, six different types of data sets can be used to store data; in addition, two types of special records (file definition and directory) must be present in the file. The arrows in the figure represent the basic pointer system that structures the file. Details of each data set type and the two types of special records are described below.

\*\*\*\*\* Insert figure 3.1 near here.\*\*\*\*\*

### **Data Sets**

All data in a WDM file are placed in data sets. Each data set is assigned a unique number from 1 to 32000. The record location of a data set within a WDM file is stored by using two special records: the file definition record and the directory record. The **file definition record** is the first record of a WDM file and indicates characteristics of the file such as number of records, pointers to directory records, and file identification name and date. (\*\*\*\*\* date--is this date created, date last accessed, date of last summary, what? \*\*\*\*\*) The file definition record has 64 positions available to locate directory records. A **directory record** contains pointers to individual records in the file based on the assigned data-set number. There are 500 positions available in each directory record to store the record number of the first record number of a data set. The position of the record identification number within a directory record is determined by the data-set number. If the data set number is 117, then in the 117th position of the first directory record an integer number is assigned that identifies the record that starts data set 117. Since 64 positions are available in the file definition record to locate directory records and 500 data sets can be located per directory record, a maximum of 32,000 data sets can be accommodated. (\*\*\*\*\* there are 4 pointers at the beginning of a directory record, do we need an here, or should the description be less specific here, and exact back in the appendix? \*\*\*\*\*)

A data set contains a homogenous set of information and attributes describing the information. A time-series data set might contain a time series of precipitation for a station, the descriptive attributes might include the time step, latitude and longitude of the station, elevation of the recording instrument

**Attributes** An attribute is a name, number, or set of characters that describe the characteristics of data in a data set. Attributes can be real or integer numbers or character strings; all attributes must be predefined and stored in an attribute type data set in a WDM file before they can be used to characterize other types of WDM data sets. (See below and appendices A.7 and C for more information on attribute type data sets) Each attribute is assigned a code number and a six-character name. Subroutines are available to the application programmer to retrieve attribute information, such as type, length, valid range, and definition, from a WDM attribute characteristic file. Subroutines are also available to add, modify, or delete attribute values from the user's WDM file. The maximum number of attributes for a data set and the total amount of space available for the attributes is specified at the time a data set is created. Both the attribute code number and value are stored on the first record of a data set. Some attributes can only be used in certain data sets and are identified as such in the group that defines the attribute.

**Groups** A group is a string of closely related data contained in a record that has been located with a pointer. A group usually contains a set of blocks. Division of data into groups depends on the data-set type. For time-series data sets, groups are organized in units of time, such as a month, day, year, or century. For tables data sets, each table is a group. For space-time data sets, a group is a period of uniform time step. For some data sets, such as time series, the groups are defined by the applications programmer. The type or size of groups can be used to increase the efficiency of data storage and retrieval by reducing sequential reading of records or sequential scanning of blocks in a record.

\*\*\*\*any instructions on how to maximize WDM efficiency by adjusting user-defined groups? if not, people will most likely use defaults- should we tell them to do this? ANNIE manual page 10, 165 - info on timeseries attributes

\*\*\*\*\*can we be specific about what type of control over what type of groups is allowed? is a list of group control words possible and appropriate?

**Blocks**

A block is a set of closely related data within a group of data in a record that has been located with a pointer. Blocks further reduce the sequential scanning of words in a record. All groups do not have blocks. One of the two most significant uses of blocks are in time-series data sets that allow data compression. If that is the case, the block is a string of compressed or uncompressed values. The first word of a compressed or uncompressed block contains the length of the block and whether it is compressed or uncompressed. Basically, blocks are used whenever a variable amount of data is needed. A table stored in a table data set is a group, but since it has a preset size, blocks are not used. (\*\*\*\*\* what is the other significant use of blocks? \*\*\*\*\*)

The application programmer does not have any control over the creation, modification or deletion of blocks. This is an internal WDM function. A list of block control words used for each data set type is found in Appendix A - WDM File Formats.

## POINTERS

Pointers are used to efficiently move to a desired location within a WDM file. Their use is transparent to the applications programmer. However, application performance may be enhanced by specification of attributes like TGROUP whose adjustment may trigger the creation of additional pointers.

**Data Set Pointers** Data set pointers link data sets of the same type. Use of the file definition and directory data sets was explained in Section 3.1.1 as the primary mechanism to point to a data set. In addition, data sets of the same type are linked with forward and backward pointers that are found on the first two words of the first record of each data set. Subroutines in the WDMS toolkit that add or delete data sets automatically update these pointers; consequently, data set pointers are transparent to the application programmer.

**Record Pointers**

Records within a data set also have forward and backward pointers. By using these pointers, the size of a data set does not need to be predetermined and the records do not need to be contiguous. These pointers can greatly reduce the required disk space and also minimize the need for file maintenance. Subroutines in the WDMS toolkit that add or delete data from data sets automatically update these pointers; consequently, record pointers are transparent to the application programmer.

### **Group Pointers**

Pointers to the beginning of each group in a data set are placed on the first record of the data set. The pointer contains the record number and word location within the record for the first word of a group. When a data set is created, the maximum number of groups is set. A word (4 bytes) is reserved on the first record of a data set for the pointer for each possible group. The default for the maximum number of groups is set to 100. The only limitation for this number is the number of words on a record (512) minus the words used for attributes and other pointers. By limiting the number of attributes, the allowable number of groups could be as large as 200 to 250. Group pointers are set by the subroutines that put data in the data set. For tables data sets, a table is a group and must be given a number from one to the maximum number of groups. For a time-series data set with yearly groups, the group number is the year plus one minus the value for the attribute BGYR, the first year of data. As is the case with data set and record pointers, group pointers are automatically updated by WDMS toolkit subroutines, and hence are transparent to the application programmer.

## **RECORD BUFFERS**

Record buffers are used to reduce the number of file reads required to use the WDM file. A cache of the most recently read records are kept in memory at all times. The number of records for the buffer is set by a Fortran PARAMETER statement and the buffer is contained in a common block. Usually the size is set to 10 records. If an additional use of one of the records in memory is required, a read from disk is saved. In particular, this is quite helpful for file definition records and directory records which must be read to find the location of a desired data set. The record buffer is transparent to the applications programmer.

## **TYPES OF DATA STORED**

WDM allows the storage of six distinct types of data:

- time-series data,
- space-time data,
- vector data,
- text and tables template data,
- table data, and
- attribute characteristic data.

The characteristics of each data-set type, as defined by the WDM file structure, are described below.

### **Time-series Data Sets**

There are several types and forms of time-series data. While most time-series data have specific dates when they are measured or calculated, some time series, such as design storm hydrographs, are synthesized and hence are independent of a specific date.

\*\*\*\*\*how do you enter "design" data that is independent of a specific date?

Some data are measured or calculated on a uniform time step, while other data are collected at random intervals. Water-quality samples are often collected at monthly, quarterly, or random intervals. By using microprocessors at data-collection sites, data are being collected whenever a change in the measured variable occurs.

Some models use data of two or more time steps. For example, a nonpoint loading model may require precipitation data on a daily time step in between storm events and on a 15-minute time step for the duration of storm events.

Within a time-series data set, data are organized in groups. Each group contains data of a uniform time unit, such as a month, day, year, or century. Each group has an associated group pointer (Section 3.4.3) that is located in the first record of the data set and contains the record number and word location within the record for the first word of the group. Attributes are used to specify the length of time of a group (hour, day, month, year, or century), and the beginning time for the first group of data. An attribute is also used to indicate whether the data is an average or total over a time step or an instantaneous value at the end of the timestep.

The data in each group is divided into blocks if any of the following conditions occurs: (1) the time step changes, (2) a switch occurs between compressed and uncompressed data, or (3) the end of a record occurs. Each block is preceded with a block control word. The block control word is an integer number (32 bits) with a bit pattern to represent five variables:

Number of values	0-32767	15 bits
Time step for the block	0-63	6 bits
Time-step units code	0-7	3 bits
Compression code	0-3	2 bits
Quality of data code	0-31	5 bits

Time-unit codes are as follows:

Seconds	1
Minutes	2
Hours	3
Days	4
Months	5
Years	6

The compression code is nonzero if the data in the block is compressed by one of three schemes:

- all values the same,
- range linearly between first and last value, or
- range nonlinearly with the type of nonlinear function defined in the data block.

Some time-series data need flags to indicate whether the data were measured or estimated data, how the data were estimated, or periods of missing record. When the quality of the data is important, the quality code of the data is set to a nonzero value. (The quality code meanings are defined by the application programmer). Quality codes can identify missing data, several forms of estimated data, or data with missing time distributions.

Table xx.--Block control word for a time-series data set

Description	valid range	number of bits
Number of values	0-32767	15
Time step for the block	0-63	6
Time-step units code	0-7	3
1-seconds   4-days		
2-minutes   5-months		
3-hours     6-years		
Compression code	0-3	2
0-uncompressed		
1-compressed		
Quality of data code	0-31	5

???? time code range os 0-7, valid 1-6--what gives? Also, how can you have a 0 time step and what is a time step of 63? And how can you have 0 values? and what about compressions codes 2 and 3????

The compression code is nonzero if the data in the block is compressed. Only one scheme is currently implemented in the WDM file. When adjacent values in a time series are the same, or within some user-defined tolerance, only the value and the number of similar adjacent values are stored. The attribute TOLR is used to define a non-zero tolerance.

Some time-series data need flags to indicate whether the data were measured or estimated data, how the data were estimated, or periods of missing record. When the quality of the data is important, the quality code of the data is set to a nonzero value. (The quality code meanings are defined by the application programmer). Quality codes can identify missing data, several forms of estimated data, or data with missing time distributions.

## Space-time Data Sets

Space-time data sets are similar to grid data with an additional dimension of time. Finite-difference and finite-element models use this data set type when values for all nodes or elements are needed at a point in time. Time-series data sets could be used, but retrievals would be inefficient. Data are placed in blocks of uniform time steps. Up to 200 time-step changes can be stored per data set, but the number of uniform time steps is only limited by the capacity of the computer and disk. The number of nodes is constant and cannot be greater than 10,000 per data set. Retrievals specify a time, starting node, and skipping factor.

## Vector Data Sets

The purpose of the vector data set is to reference and link arcs, polygons, nodes, with other data sets. Arcs are strings of x-z coordinates that could define a location of a road, center line of a stream, political boundary, etc. Often boundaries are shared, such as the center line of a river and a political boundary. A polygon is an arc or a set of arcs that close, such as a watershed boundary. Nodes are points in space. The type of coordinates used to express the vector data are defined by attributes. The coordinates

are stored as real numbers.

Vector data sets are used to display spacial information for use in a modeling or data management effort. This has included state and county boundaries, watershed divides, and stream channel locations.

Data in USGS optional DLG format (NCIC, 1985) can be converted to WDM vector format through a stand alone conversion program called CNVDLG. CNVDLG is described in detail in the case study.

Table xx.--Block control word for a vector data set

Description	valid range	number of bits
block type	1-15	4
1 - header		
2 - data		
data type	1-3	2
1 - node		
2 - arc		
3 area		
internal number	1-4095	12
length of block	1-8191	13

## Text and Tables Template Data Sets

For the application programmer, text and tables template data sets serve two functions. First, they can be used to store text to be written to the users terminal or a file. Second, they can be used to store templates for table data sets (Section 3.2.5). This data set type is used to store screen definitions and directives for the ANNIE Interactive Development Environment (AIDE) (Kittle et al., 1989).

Table xx.--Block control word for a text and tables template data set

Description	valid range	number of bits
class of data in block	1-5	4
1 - 1 dimensional parm		
2 - 2 dimensional parm		
3 - text		
4 - menu		
5 - file		
block identifier	0-63	6
order of block info	0-63	6
length of block	1-32767	15

## Table Data Sets

Table data sets store one or more, related or unrelated, tables. Each table in a data set is a group and has a group pointer based on the table number.

The tables stored in a table data set can have identical or different formats; the only requirement is that each format used is described in a WDM tables template data set. Included in this description are header information, order, type, and space allocation. Tables can have up to 30 fields of any mix of integer, real, double precision, or character data. Character fields must be in increments of words (4 bytes). A restriction to 30 fields is currently needed for the software to place the table on the display terminal for processing. The maximum number of rows is system dependent and is based on the buffer dimensions for table input and output.

Table data sets are not useful until table templates and specific application programs have been developed, although the only requirement to producing table data sets is that the table template is stored on the message file. A special interactive program called MESSIE (Section 4.5) is used to create the template specification on the message file. If the AIDE interaction environment is used, once the table is created, only one subroutine call is needed to let the user interactively edit data in the table.

Searches such as those that can be performed on attributes for data sets (Section XX) cannot be done on table header information unless the application programmer creates one table per data set that stores, as data set attributes, the table header information.

Application programs have been written that use WDM table data sets to store annual series of peak flows, hydraulic properties of a cross section, and HSPF model parameters.

### **Attribute Characteristic Data Sets**

\*\*\*\*describe attribute data sets\*\*\*\*

## CASE STUDIES

In this section a case study developed to illustrate applications of the WDMS toolkit is described. The case study focuses on both general WDM housekeeping tasks and the six data set types that can be stored in a WDM file. A brief description and discussion of each section of the case study is provided along with the pertinent Fortran code and resulting output. The seven sections of the case study are:

- (1) general WDM housekeeping
- (2) Time-series data
- (3) Space-time data
- (4) Vector data
- (5) Text and tables template data
- (6) Table data
- (7) Attribute characteristic data

For each section of the case study, part of the complete and functional Fortran program is broken into sequential code segments, and the function of each code segment is described sequentially. For application programmers desiring to use the entire program as a template for producing similar programs, the case study program is included on the WDM distribution disks.

If more information about what routines are available to call for a specific purpose is needed, see functional summaries in Chapter 5. For full definitions of all arguments and return codes, see the detailed summaries of each toolkit routine in Appendix D. Detailed information about attributes is found in Appendix C.

(KF) Note that the case studies are intended as examples. There is no error checking of input files that might normally be found in a finished production version of a program. Input and output formats were kept simple. We have tried to keep the program logic simple and straight forward and still describe a useful application.

### PREREQUISITES TO RUNNING THE CASE STUDY

(AL) The software for the case study is provided on the WDM system distribution disks or tapes. First the code should be compiled and linked. Next the MESSAGE.WDM file should be created using the program WDMIMEX, which is also provided. Next use WDMIMEX to import the table template and DLG data. More information is provided on the READ.ME file on the distribution disks or tapes.

# CASE STUDIES--driver

The program CASES is the driver for the case studies. It does some housekeeping tasks and calls the example subroutines for the various WDM data-set types. Table xx contains a summary of the WDM toolkit routines called by CASES.

Table xx.--WDM toolkit routines called by case study driver.

Routine	How used
WDBOPN	open a WDM file
CSTS	time-series case study
CSST	space-time case study
CSDLG	digital line graph (DLG) case study
CSTXT	text and table template case study
CSTA	table data case study
CSATR	attribute data case study
WDFCLC	close a WDM file

## Assumptions

Note that the case studies are intended as examples. There is little error checking of the input files that would normally be found in a finished, production program. This has been done in an attempt to keep the program logic as simple and straightforward as possible.

## enhancements

### Specifications

Standard conventions have been used to declare local variables and externals. The unit number of the attribute message file is defined in the include file PMESFL.INC. File unit numbers for the other files have been defined in a data statement.

```
C
C
C      PROGRAM  CASES
C
C      + + + PURPOSE + + +
C      Driver program to demonstrate use of WDM library.
C      This is a program that you can drive. Not just a program
C      that runs around a little track in a groove.
C
C      + + + PARAMETERS + + +
C      INCLUDE 'PMESFL.INC'
C
C      + + + LOCAL VARIABLES + + +
C      INTEGER      RONWFG, RETCOD, WDMSFL, DSN, OFL, NDSN, DSNTYP,
C      >           IFL, INP, NORO, OPTN
C      CHARACTER*64 WDNAME, DWDNAM, PRTFIL, INFILE
C
C      + + + EXTERNALS + + +
C      EXTERNAL     WDBOPN, WDFCLC, WDDSDL, WDDSRN, WDCKDT
C                  EXTERNAL    CSTS, CSTA, CSST, CSTXT, CSDLG, CSATR
C
C      + + + DATA INITIALIZATIONS + + +
C      DATA WDMSFL, OFL, IFL, INP
C      >      / 60,   98,   1,   99 /
```

## Input and Output Formats

Input formats are in the range 1000 - 1999. Output formats are in the range 2000 - 2999.

```
C      + + + INPUT FORMATS + + +
1000 FORMAT ( A64 )
1010 FORMAT ( Y1, IX, A64 )
C      + + + OUTPUT FORMATS + + +
2000 FORMAT ( ' Error on open of attribute message file',
    >      /,'   file name: ', A64,
    >      /,'   RETCOD: ', I5 )
2001 FORMAT ( ' Successful open of attribute message file',
    >      /,'   file name: ', A64 )
2010 FORMAT ( ' Error, could not open wdm file as new',
    >      /,'   file name: ', A64,
    >      /,'   RETCOD: ', I5 )
2011 FORMAT ( ' Successful open of new wdm file',
    >      /,'   file name: ', A64 )
2012 FORMAT ( ' Error, could not open wdm file as old',
    >      /,'   file name: ', A64,
    >      /,'   RETCOD: ', I5 )
2013 FORMAT ( ' Successful open of existing wdm file',
    >      /,'   file name: ', A64 )
2100 FORMAT ( ' Finished processing case studies' )
2101 FORMAT ( ' Processing case study option: ', I3,
    >      /,'   file name: ', A64 )
2200 FORMAT ( ' Closing WDM files' )
2210 FORMAT ( ' Error closing ', A64,
    >      /,'   RETCOD: ', I5 )
2220 FORMAT ( ' Closed ', A64 )
C      + + + END SPECIFICATIONS + + +
C      get name for file of summary information and open the file
READ (IFL,1000) PRTFIL
OPEN(UNIT=OFL,FILE=PRTFIL)

C      open message file containing attributes
INCLUDE 'FMESSG.INC'
C      attributes want to be read-only
RONWFG= 1
CALL WDBOPN (MESSFL,WDNAME,RONWFG,
O      RETCOD)
IF (RETCOD.NE.0) THEN
    error opening file
    WRITE (OFL,2000) WDNAME, RETCOD
ELSE
    successful open of file
    WRITE (OFL,2001) WDNAME
END IF

C      open wdm file for case studies data
IF (RETCOD.EQ.0) THEN
    get status and name of data wdm file
    READ (IFL,1010) NORO, DWDNAM
    IF (NORO.EQ. 1) THEN
        create a new wdm file
        RONWFG= 2
        CALL WDBOPN (WDMMSFL,DWDNAM,RONWFG,
                    RETCOD)
        IF (.RETCOD.NE.0) THEN
            problem opening new wdm file
            WRITE (OFL,2010) DWDNAM, RETCOD
        ELSE
            successfully opened new wdm file
            WRITE (OFL,2011) DWDNAM
        END IF
    ELSE
        open an existing wdm file
        RONWFG= 0
        CALL WDBOPN (WDMMSFL,DWDNAM,RONWFG,
                    RETCOD)
        IF (RETCOD.NE.0) THEN
            problem opening wdm as old
            WRITE (OFL,2012) DWDNAM, RETCOD
        ELSE
            successfully opened existing wdm file
            WRITE (OFL,2013) DWDNAM
        END IF
    END IF
END IF
```

## Open output print file

A file for general output is opened. The Fortran unit number was specified in the data statement above.

## Open attribute WDM file

Prerequisite to performing virtually all WDM operations, including creating a WDM file, is to open the message file containing all the data attributes that may be used by the data sets in the WDM file. The include file named FMESSG.INC provides the name of the attribute file, which can be system or application specific. In this example, the name of the file is CASEMS.WDM. In a system with multiple users, the attribute file can be shared, so it is opened as "read only" by specifying RONWFG=1. The Fortran unit number for the message file was provided above in the specifications in the include file PMESFL.INC.

## Open new or existing WDM file

The Fortran unit number for the WDM file was specified in the data statement above. The flag NORA and the name of the WDM file are read from the input file. If NORO is 1, WDBOPN is called with RONWFG=2 to indicate a new WDM file is to be created. If NORO is not 1, WDBOPN is called with RONWFG=0, indicating an existing WDM file is to be opened. A return code of 0 indicates the open was successful.

### select case study option

The case study option (OPTN) and the name of the data input file (INFILE) are read from the input file. OPTN=0 signals the end of the case studies.

```
C      IF (RETCOD.EQ.0) THEN
C          begin loop for case studies options
C          CONTINUE
C              get case study option and input file name
C              OPTN: 0 - done
C                  1 - time series 3 - dlg           5 - table
C                  2 - space time   4 - text & table 6 - attribute
C              READ (IFL,1010) OPTN, INFILE
C              IF (OPTN .LE. 0 .OR. OPTN. GT. 6) THEN
C                  done with case studies
C                  WRITE (OFL,2100)
C                  OPTN = 0
C              ELSE
C                  valid case study, open input file
C                  OPEN (UNIT=INP,FILE=INFILE)
C                  WRITE (OFL,2101) OPTN, INFILE
C                  run selected case study
C                  GO TO (110,120,130,140,150,160) OPTN
C                      CONTINUE
C                          time-series case study
C                          CALL CSTS(MESSFL,WDMMSFL,OFL,INP)
C                          GO TO 190
C      120      CONTINUE
C                  space time case study
C                  CALL CSST(MESSFL,WDMMSFL,OFL,INP)
C                  GO TO 190
C      130      CONTINUE
C                  dig case study
C                  CALL CSDLG(MESSFL,OFL,INP)
C                  GO TO 190
C      140      CONTINUE
C                  text and table template case study
C                  CALL CSTXT(MESSFL,OFL,INP)
C                  GO TO 190
C      150      CONTINUE
C                  table data case study
C                  CALL CSTA(MESSFL,WDMMSFL,OFL,INP)
C                  GO TO 190
C      160      CONTINUE
C                  attribute data case study
C                  CALL CSATR(MESSFL,WDMMSFL,OFL,INP)
C      190      CONTINUE
C                  CLOSE (UNIT=INP)
C              END IF
C              IF (OPTN .NE. 0) GO TO 100
C          END IF
C
C          IF (RETCOD.EQ.0) THEN
C              WRITE (OFL,2200)
C              CALL WDFLCL (WDMMSFL,
C                           RETCOD)
C          O      IF (RETCOD.NE.0) THEN
C                  WRITE (OFL,2210) WDNAME, RETCOD
C          ELSE
C                  WRITE (OFL,2220) WDNAME
C          END IF
C
C          close WDM file containing attributes
C          CALL WDFLCL (MESSFL,
C                           RETCOD)
C          IF (RETCOD.NE.0) THEN
C              WRITE (OFL,2210) WDNAME, RETCOD
C          ELSE
C              WRITE (OFL,2220) WDNAME
C          END IF
C      END IF
C
C      close output log file
C      CLOSE (UNIT=OFL)
C
C      STOP
C  END
```

### close WDM and output files

After the case studies have been run, WDFLCL is called to close the data WDM file and to close the attribute WDM file. The output file is closed with the standard Fortran close.

## CASE STUDIES--time series

The CSTS routine reads time-series data from a formatted ASCII file and performs a number of operations related to data-set creation, data input, and data manipulation. Table xx contains a summary of the WDM toolkit routines called by CSTS.

Table xx.--WDM toolkit routines called by time-series case study.

Routine	How used
WDBCRL	create a data set
WDBSAC	add character attributes to the data set
WDBSAI	add integer attributes to the data set
WDBSAR	add real attributes to the data set
WDTPUT	write time-series data to the data set
WDDSCL	create new data set, copying attributes from existing data set
WTFNDT	find period of record for time series in data set
WDTGET	read time-series data from data set
TIMDIF	compute number of time steps between two dates

Table yy.--Attributes important to time-series data sets

Attribute	Explanation
VBTIME (85)	variable time step option flag 1 - all data in the data set must be at the same, constant time step 2 - time step of the data may vary over time (default)
TCODE (17)	time units code of the data set 1 - seconds                          4 - days 2 - minutes                            5 - months 3 - hours                             6 - years
TSSTEP (33)	time step of the data in TCODE units
TGROUP (34)	unit for data group pointers 3 - hours                            6 - years (default) 4 - days                             7 - centuries 5 - months
	Depending on the time step of the data, may effect the speed of data retrievals. Will define the maximum amount of data that can be stored in a data set.
TSBYR (27)	starting year of data
TSFORM (84)	form of the time-series data 1 - mean over the time step (default) 2 - total over the time step 3 - instantaneous at time (end of time step) 4 - minimum over the time step 5 - maximum over the time step This may be important when you retrieve data at a time step other than the time step it was stored at.
COMPFG (83)	data compression option flag 1 - data are compressed (default) 2 data are not compressed compressed data will take up less space in a WDM file, but may require special actions to update or modify the data values.

### Assumptions

There will be no partial days in the input file. Time-series data in the input file will be in chronological order. Time step of the input data can be evenly aggregated or disaggregated to a 1 day time step. The input file contains the correct information in the correct columns. A WDM file is open when CSTS is called. The specified data-set number does not exist in the WDM file prior to the run.

### enhancements

New data could be added to an existing data set by calling \*\*\*\*\* to if the specified data set exists. Add additional error checking to be sure input data is valid. Check that time series is in chronological order.

### Specifications

Standard conventions have been used to declare dummy arguments, local variables, functions, and externals. Data-set type has been initialized to time series (DSTYPE=1). Attribute names and index numbers have been initialized in data statements for all of the attributes used in this subroutine. Attribute lengths are initialized for the character attributes.

```
C
C
C      SUBROUTINE  CSTS
I          (MESSFL,WDMSFL,OFL,IFL)
C
C      + + + PURPOSE + + +
C      Case study example for time-series data sets.
C
C      + + + DUMMY ARGUMENTS + + +
INTEGER  MESSFL,WDMSFL,OFL,IFL
C
C      + + + ARGUMENT DEFINITIONS + + +
C      MESSFL - Fortran unit number of WDM file containing attribute
           information
C      WDMSFL - Fortran unit number of WDM file used for case study data
C      OFL   - Fortran unit number of output text file
C      IFL   - Fortran unit number of input data file
C
C      + + + LOCAL VARIABLES + + +
INTEGER  SAINDC(3), SALEN(3), SAINDR(3), SAINDI(7), SALEN,
>          SAVALI(7), DSN, DSNN, NVAL, DATE(6), DATEEND(6),
>          TSSTEP, TCODE, VBTIME, TRNSFM(5), I, J, JJ, RETCOD,
>          GPFLG, TDTSFRC, DTRAN, DSTYPE, DTOVWR, QUALEFG
REAL    SAVALR(3), RVAL(600)
CHARACTER*1 SAVALC(68)
CHARACTER*6 SANAMC(3), SANAMR(3), SANAMI(7)
C
C      + + + EXTERNALS + + +
EXTERNAL  WDBCR1, WDBSAC, WDBSAI, WDBSAR, WDTPUT
EXTERNAL  WDDSCL, WTPNDT, TIMDIF, WDTGET
C
C      + + + DATA INITIALIZATIONS + + +
DATA DSTYPE / 1 /
DATA SANAMC, SAINDC, SALEN
>          /'TSTYPE', 'STAID ', 'DESCRP',
>          1,          2,          45,
>          4,          16,         48 /
DATA SANAMR, SAINDR
>          /'LATDEG', 'LNGDEG', 'DAREA ',
>          8,          9,          11 /
DATA SANAMI, SAINDI
>          /'TSSTEP', 'TCODE ', 'TGROUP', 'TSBYR ', 'TSFORM', 'COMPPG', 'VBTIME',
>          33,         17,         34,         27,         84,         83,         85 /
C
C      tsform: mean total inst min max
C      dtran: ave,same sum,div min,max
C      DATA TRNSFM / 0,      1,      0,      3,      2 /

```

## Input and Output formats

Input formats are in the range 1000 - 1999. Output formats are in the range 2000 - 2999.

```
C      + + + INPUT FORMATS + + +
1001 FORMAT ( I5, 1X, 4A1, 1X, 16A1, 1X, 48A1 )
1002 FORMAT ( 5X, 7I5, 3F8.0 )
1003 FORMAT ( I4,5I3, I5, I3 )
1004 FORMAT ( 11F7.0 )
1005 FORMAT ( 5I5 )

C      + + + OUTPUT FORMATS + + +
2101 FORMAT ( ' Creating time-series data set', I5 )
2102 FORMAT ( ' Error, unable to create dsn', I5, ', RETCOD:', I5 )
2200 FORMAT ( ' Adding attributes to new data set',
    >   /, ' retcod attribute value
    >   /, '                                ')
2210 FORMAT ( I5, 6X, A6, 3X, 48A1 )
2220 FORMAT ( I5, 6X, A6, 3X, I5 )
2230 FORMAT ( I5, 6X, A6, 3X, F10.2 )
2301 FORMAT ( I5, ' values added to dsn starting', I5,5I3 )
2302 FORMAT ( I5, ' values failed add starting' I5,5I3, 'RETCOD:', I5 )
2300 FORMAT ( ' Adding time-series data to data set' )
2400 FORMAT ( ' Copy data set and modify time step' )
2401 FORMAT ( ' Error copying label from dsn', I5, ' to dsn', I5,
    >   /, ' RETCOD:', I5 )
2402 FORMAT ( ' Copied label from dsn', I5, ' to dsn', I5,
    >   /, ' and modified attributes',
    >   /, ' retcod attribute value
    >   /, '                                ')
2410 FORMAT ( I5, 6X, A6, 3X, I5 )
2450 FORMAT ( ' Error getting time series, RETCOD:', I5 )
2460 FORMAT ( ' Error putting time series in dsn', I5, ' RETCOD:', I5 )
2480 FORMAT ( I5,5I3, I5 )
2481 FORMAT ( 10F8.2 )

C      + + + END SPECIFICATIONS + + +
C      begin loop for new data set
100 CONTINUE
C      get dsn and station description
    READ (IFL,1001,ERR=110,END=110) DSN, SAVALC
    IF (DSN .GT. 0) THEN
        add general part of data-set label using default space
        WRITE (OFL,2101) DSN
        CALL WDBCRL ( WDMMSFL, DSN, DSTYPE,
            RETCOD )
        IF (RETCOD .NE. 0) THEN
            unable to create time-series dsn
            WRITE (OFL,2102) DSN, RETCOD
            END IF
        ELSE
            end of input time-series data
            RETCOD = 1
        END IF
        GO TO 120
    110 CONTINUE
    C      end for file or read error
    RETCOD = -1
    120 CONTINUE
```

## Create a data set

The data-set type is defined as time series (DSTYPE=1) in a data statement above. The data-set number for a new data set and some descriptive information about the data set are read from the input file. The general label for the new data set (data-set DSN) is written to the WDM file using subroutine WDBCRL. (WDBCRL uses defaults for space allocation in a new data set. For some applications subroutine \*\*\*\*\* may be a more appropriate choice.) A return code (RETCOD) of 0 indicates the label has been successfully added. A return code of -71 indicates that the data set already exists. See the detailed summaries of each routine in Appendix D for full definitions of all arguments and return codes.

### Add data-set attributes

The descriptive character attributes TSTYPE, STAID, and DESCRP were read in with the data set number above. The attributes describing the time step of the data and how the data is stored in the WDM file are read in and added to the WDM file using WDBSAI. Attributes describing teh location and size of the basin are read in and added to the WDM file using WDBSAR. A more detailed description of some of the time-series specific attributes can be found in table yy.

```

IF (RETCOD .EQ. 0) THEN
    data-set label successfully added now add general attributes
    J= 1
    WRITE (OFL,2200)
    DO 210 I = 1, 3
        add descriptive character attributes read above
        CALL WDBSAC ( WDMFL, DSN, MESSFL,
                      SAINDC(I), SALEN(I), SAVALC(J),
                      RETCOD )
        WRITE (OFL,2210) RETCOD, SANAMC(I),
                          (SAVALC(JJ), JJ=J, J+SALEN(I)-1)
        J= J + SALEN(I)
    CONTINUE
    read rest of attributes
    READ (IFL,1002) (SAVALI(I), I=1,7), (SAVALR(I), I=1,3)
    SALEN= 1
    DO 220 I = 1, 7
        integer attributes
        CALL WDBSAI ( WDMFL, DSN, MESSFL,
                      SAINDI(I), SALEN, SAVALI(I),
                      RETCOD )
        WRITE (OFL,2220) RETCOD, SANAMI(I), SAVALI(I)
    CONTINUE
    DO 230 I = 1, 3
        real attributes
        IF (SAVALR(I) .GT. 0.0) THEN
            add valid latitude, longitude, or drainage area
            CALL WDBSAR ( WDMFL, DSN, MESSFL,
                          SAINDR(I), SALEN, SAVALR(I),
                          RETCOD )
            WRITE (OFL,2230) RETCOD, SANAMR(I), SAVALR(I)
        END IF
    CONTINUE
    assume attributes good, so ignore error flags
    RETCOD= 0
END IF
IF (RETCOD .EQ. 0) THEN
    process the time-series data
    TSSTEP= SAVALI(1)
    TCODE = SAVALI(2)
    DTOVWR= 1
    WRITE (OFL,2300)
CONTINUE
read start date, number of values, and quality code
READ (IFL,1003) (DATE(I), I = 1, 6), NVAL, QUALEFG
IF (NVAL .GT. 0) THEN
    read time series data
    READ (IFL,1004) (RVAL(I), I = 1, NVAL)
    add time-series data to data set
    CALL WDTPUT (WDMFL, DSN, TSSTEP, DATE, NVAL, DTOVWR,
                  QUALEFG, TCODE, RVAL,
                  RETCOD)
    IF (RETCOD .EQ. 0) THEN
        values successfully written
        WRITE (OFL,2301) NVAL, DATE
    ELSE
        error writing values
        WRITE (OFL,2302) NVAL, DATE, RETCOD
    END IF
END IF
IF (RETCOD .EQ. 0 .AND. NVAL .GT. 0) GO TO 300
END IF
IF (RETCOD .EQ. 0) GO TO 100
IF (RETCOD .GE. 0) THEN
    begin loop to transform time step
    WRITE (OFL,2400)
    CONTINUE
    read in old dsn, new dsn, and new time step
    READ (IFL,1005) DSN, DSNN, TSSTEP, TCODE, DTRAN
    IF (DSN .LE. 0) THEN
        done copying data sets
        RETCOD= 1
    ELSE
        create new data set like old data set
        CALL WDDSC ( WDMFL, DSN, DSNN,
                      RETCOD )
        IF (RETCOD .NE. 0) THEN
            error copying label
            WRITE (OFL,2401) DSN, DSNN, RETCOD
    END IF
END IF

```

### Read data from ASCII file and write data to WDM file

The time step and time code of the input data is defined by the values read in earlier. On the first pass through this routine, the number of values (NVAL) to be read and the starting date (DATE) read earlier are used. NVAL values are read from the input file. The subroutine WDTPUT is used to write the values to the data set (DSN) in the WDM file. The data over write switch (DTOVWR) is set to allow overwriting. \*\*\*\* This may be necessary when the group pointer (TGROUP) in the data set is a time period much greater than the amount of time-series data written at each call to WDTPUT. An example might be writing a day of 5 minute data to a data set that has annual group pointers. By default, the program will pad the data set to the end of the year with THE missing data filler (TSFILL). When you try to add a subsequent day, you will be overwriting the missing values. \*\*\*\* A return code of 0 indicates the data was successfully added to the data set. The meaning of non-zero return codes is described in appendix D.

### Create new data set, copying attributes from existing data set

The subroutine WDDSC is used to copy the attributes data-set type, and space allocations of an existing, source data set to a new data set. The time-series data is not copied in this operation. The user specifies the existing data set (DSN) and the requested new data set (DSNN). A return code (RETCOD) of 0 indicates the copy operation was successful. A return code of -61 indicates the source data set did not exist and -62 indicates that the new data set already existed.

### Set time step to daily and force to constant time step

The time step and units for this new data set are read in above with the data-set numbers. WDBSAI is called to modify the original values of TSSTEP and TCODE with these new ones in the new data set. The data sets is also set to a constant time step (VBTIME=1). These attributes can only be modified if no time-series data has been stored in the data set. Once data has been added to the data-set these storage descriptive attributes can not be modified.

### Get period of record from original data set determine total number of days

The subroutine WTFNDT is called to determine the period of record for the time-series data in the original data set. Then TIMDIF is called to determine the number of days between the beginning (DATE) and end (DATEND). Note that TCODE and TSSTEP were defined as 1 day in the step above.

### Read original data set at new time step

#### Write data to second data set

The time-series data is read from the first data set (DSN) at a new time step and then written to the new data set (DSNN). Return code of 0 indicates successful read/write. Non-zero returns include "\*\*\*\*". The data is written to the output file at the new time step.

```

C      ELSE
C          label copied successfully
C          WRITE (OFL,2402) DSN, DSNN
C          modify attributes to 1 day in new data set
C          VBTIME= 1
C          CALL WDBSAI ( WDMSFL,DSNN,MESSFL,SAINDI(1),SALEN,TSSTEP,
C                         RETCOD )
C          WRITE (OFL,2410) RETCOD, SANAMI(1), TSSTEP
C          CALL WDBSAI ( WDMSFL,DSNN,MESSFL,SAINDI(2),SALEN,TCODE,
C                         RETCOD )
C          WRITE (OFL,2410) RETCOD, SANAMI(2), TCODE
C          CALL WDBSAI ( WDMSFL,DSNN,MESSFL,SAINDI(7),SALEN,VBTIME,
C                         RETCOD )
C          WRITE (OFL,2410) RETCOD, SANAMI(7), VBTIME
C          ignore return code for attributes
C          RETCOD = 0
C      END IF
C  END IF
C  IF (RETCOD .EQ. 0) THEN
C      get period of record from original data set
C      GPFLG = 0
C      CALL WTFNDT ( WDMSFL, DSN, GPFLG,
C                     TDSFRC, DATE, DATEND, RETCOD )
C      determine # of days
C      CALL TIMDIF ( DATE, DATEND, TCODE, TSSTEP,
C                     NVAL )
C      get data from first data set
C      QUALFG= 0
C      CALL WDTGET ( WDMSFL,DSN,TSSTEP,DATE,NVAL,OTRAN,QUALFG,TCODE,
C                     RVAL,RETCOD )
C      IF (RETCOD .NE. 0) THEN
C          Error getting time-series data
C          WRITE (OFL,2450) RETCOD
C      ELSE
C          put data in new data set
C          QUALFG = 0
C          CALL WDTPUT ( WDMSFL, DSNN, TSSTEP, DATE, NVAL,
C                         DTOVWR, QUALFG, TCODE, RVAL,
C                         RETCOD )
C          IF (RETCOD .NE. 0) THEN
C              error putting time-series data
C              WRITE (OFL,2460) DSNN, RETCOD
C          ELSE
C              successfully added data, print results
C              WRITE (OFL,2480) DATE, NVAL
C              WRITE (OFL,2481) (RVAL(I), I = 1, NVAL)
C          END IF
C      END IF
C      IF (RETCOD .EQ. 0) GO TO 400
C  END IF
C
C  RETURN
C

```

## CASE STUDIES--space time

The CSST routine reads space-time data from an ASCII file and performs a number of operations related to data-set creation, data input, data summary, and data retrieval. Table xx contains a summary of the WDM toolkit routines called by the subroutine.

Table xx.--WDM toolkit routines called by space-time case study.

Routine	How used
WDCKDT	check WDM file for existence of a data set
WDLBAX	create a data set
WDBSAC	add character attributes to the data set
WDBSAI	add integer attributes to the data set
WSTAGP	allocate space for group for data set
WSTPTR	write space-time data to data set
WSTGSU	get summary information about data in space-time data set
WSTGTR	read space-time data from data set
TIMADD	increment a date by some number of time steps

Table yy.--Attributes important to space-time data sets

Attribute	Explanation
STDTYPE (265)	type of data in the space-time data set **** INTE - integer values REAL - real values DBLE - double precision values
STDIMX (266)	space-time x-dimension
STDIMY (267)	space-time y-dimension
STDIMZ (268)	space-time z-dimension ****

### assumptions

Expects the space-time data to be type REAL. For storing data, the maximum number of points (x-dimension nodes \* y-dimension values) is 300. Five nodes with 3 variables are always retrieved. Could read in additional attributes to describe the data set (Note--be sure there is enough space for attributes, check arguments in call to WDLBAX, NDN+NUP+NSA+NSASP+NDP <= \*\*\*\*)

### enhancements

Could change calls to WSTAGP and WSTGTR to their integer or double precision equivalents (WSTPTI, WSTGTI or WSTPTD, WSTGTD). Or, additional logic could be added to include calls to all three types of data, possibly based on value for the attribute STDTYPE (SAVALC(1-4)).

### Specifications

Standard conventions have been used to declare dummy arguments, local variables, functions, and externals. Data set type has been initialized to space time (DSTYPE=7). Attribute names and index numbers have been initialized in data statements for all of the attributes used in this subroutine. Attribute lengths are initialized for the character attributes.

```

C
C
C      SUBROUTINE  CSST
I          (MESSFL,WDMFL,OFL,IFL)
C
C      + + + PURPOSE + +
C      case study example for space time data sets
C      Stores data in a space time data set. Prints out a summary
C      at selected locations.
C
C      + + + DUMMY ARGUMENTS + +
INTEGER      MESSFL,WDMFL,OFL,IFL
C
C      + + + ARGUMENT DEFINITIONS + +
C      MESSFL - Fortran unit number of WDM file containing attribute info
C      WDMFL - Fortran unit number of WDM file used for case study data
C      OFL  - Fortran unit number of output text file
C      IFL  - Fortran unit number of input file
C
C      + + + LOCAL VARIABLES + +
INTEGER      DSN, RETCOO, PSA, GRPIND, NORO, LEN1, LEN6,
1           TUN, TST, NOV, NDN, NUP, NSA, NSASP, NDP, SALEN,
2           DSTYPE, NVAL, NDIM, NUMXY(2), BASN(2), SKPN(2),
3           I, J, K, M, SDDAT(6), SEDAT(6), M1, MN, IY, JJ, NDELT,
4           SAINDC(2), SAINDI(2), SALEN(2)
REAL        RBUFF(300), XBUFF(5,3,10), FRAC
CHARACTER*1  SAVALC(52)
CHARACTER*6  SANAMC(2), SANAMI(2)
C
C      + + + FUNCTIONS + +
INTEGER      WDCKDT
C
C      + + + EXTERNALS + +
EXTERNAL     WDCKDT, WDLBAX, WDBSAC, WDBSAI
EXTERNAL     WSTAGE, WSTPTR, WSTGTR, WSTGSU, COPYI, TIMADD
C
C      + + + DATA INITIALIZATIONS + +
DATA DSTYPE, NDN, NUP, NSA, NSASP, NDP, LEN1, LEN6
> /      7,   2,   2,  15,   40, 150,   1,   6 /
DATA SANAMC, SANAMI / 'STDTYPE', 'STANAM', 'STDIMX', 'STDIMY' /
DATA SAINDC, SAINDI /   265,       45,       266,      267 /
DATA SALENC      /      4,       48 /

```

## Input and Output formats

Input formats are in the range 1000 - 1999. Output formats are in the range 2000 - 2999.

```
C      + + + INPUT FORMATS + + +
1000 FORMAT ( 3I5, 1X, 4A1, 1X, 48A1 )
1001 FORMAT ( I4,5I3, 3I5, 4I5 )
1300 FORMAT ( I4,5I3, 3I5 )
1340 FORMAT ( 10F8.0 )
C
C      + + + OUTPUT FORMATS + + +
2001 FORMAT ( ' Found existing space-time dsn', 15 )
2002 FORMAT ( ' Error', I4, ' found existing dsn', 15,
>           /, ' It is not a space-time data set.' )
2003 FORMAT ( ' Label added for dsn', 15, ', adding attributes:',
>           /, ' retcod attribute value
>           /, '          '
2210 FORMAT ( I5, 6X, A6, 3X, 48A1 )
2220 FORMAT ( I5, 6X, A6, 3X, I5 )
2240 FORMAT ( '      # time time time
>           /, ' added steps step unit      starting date      # of',
>           /, '          values',
>           /, '-----'
2250 FORMAT ( ' dummy', I6, 2I5, I6,5I3 )
2251 FORMAT ( ' Error', I4, ' when trying to add dummy group' )
2301 FORMAT ( ' Error', I4, ' problem adding group',
>           /, ' date', I5,4I3,
>           /, ' time step and units', 2I5,
>           /, ' number of values', I5 )
2302 FORMAT ( ' group', I6, 2I5, I6,5I3 )
2341 FORMAT ( ' data', 6X, 2I5, I6,5I3, I6 )
2342 FORMAT ( ' Error', I4, ' putting space time data',
>           /, ' date', I5,5I3,
>           /, ' dimensions', I2, ', values', I5 )
2343 FORMAT ( ' X: numn', I4, ', basn', I4, ', skpn', I4,
>           /, ' Y: numn', I4, ', basn', I4, ', skpn', I4 )
2400 FORMAT ( ' Summary of space-time data-set', I5,
>           /, '      time time no. of      period of record ',
>           /, ' group step unit values fraction      begins / ends
>           /, '-----'
2401 FORMAT ( I5,1X, 2I5, I6, F9.2, 4X, I4,5I3, /, 35X, I4,5I3 )
2402 FORMAT ( ' Error', I4, ' summarizing groups in dsn', I5,
>           /, ' group index', I5 )
2403 FORMAT ( ' Finished summarizing groups in dsn', I5 )
2551 FORMAT ( ' Error', I4, ' reading time step', I4,
>           /, ' start date', I5,5I3,
>           /, ' ndim', I5,
>           /, ' numxy', 2I5,
>           /, ' basn', 2I5,
>           /, ' skpn', 2I5,
>           /, ' nval', I5 )
2560 FORMAT ( ' Successful read of', I4, ' intervals of data' )
2561 FORMAT ( 3F10.3 )
2900 FORMAT ( ' Space-Time case study complete.' )
C
C      + + + END SPECIFICATIONS + + +
C      READ (IFL,1000) DSN, (NUMXY(I), I = 1, 2), SAVALC
C      check data set existence and type
RET COD = WDCKDT ( WDMSFL, DSN )
NORO = 2
IF (RETCOD .EQ. DSTYPE) THEN
  data set already exists and is a space time data set
  WRITE (OFL,2001) DSN
  RETCOD= 0
ELSE IF (RETCOD .NE. 0) THEN
  data set already exists but is not a space time data set
  WRITE (OFL,2002) RETCOD, DSN
ELSE
```

## Check WDM for data set

The data set type is defined as space time (DSTYPE=7) in a data statement above. The data-set number for the space time data is read from the input file. The function WDCKDT is called to check the WDM file for the requested data set. If the data set does not exist 0 is returned, if the data set exists and is a space time type data set 7 is returned.

### Create data set Add attributes

If the data set does not exist, WDLBAX is used to add the general label for the new data set. WDLBAX is used instead of WDBCRL so that the number of data group pointers can be increased from the default of 100 to 150 (at the same time the amount of space for attribute information is decreased). WDBSAC is called to add the character attributes station name (STANAM) and space time data type (STDTYPE). WDBSAI is called to add the integer attributes for the maximum x and y dimensions for the space time data (STDIMX and STDIMY).

```

C      data set does not exist, add it
NORO = 1
C      add general part of the label, lots of space for data pointers
CALL WDLBAX (WDMSFL,DSN,DSTYPE,NDN,NUP,NSA,NSASP,NDP,
O          PSA)
        WRITE (OFL,2003) DSN
J = 1
DO 210 I = 1, 2
C      add descriptive character attributes read above
CALL WDBSAC (WDMSFL, DSN, MESSFL,
I          SAINDC(I), SALEN(I), SAVALC(J),
O          RETCOD)
        WRITE (OFL,2210) RETCOD, SANAMC(I),
>          (SAVALC(JJ),JJ=J,J+SALEN(I)-1)
J = J + SALEN(I)
210  CONTINUE
C      add x-dimension and y-dimension attributes
SALEN = 1
DO 220 I = 1, 2
    CALL WDBSAI (WDMSFL,DSN,MESSFL,SAINDI(I),SALEN,NUMXY(I),
O          RETCOD)
    WRITE (OFL,2220) RETCOD, SANAMI(I), NUMXY(I)
220  CONTINUE
C      ignore non-zero attribute return code
RETCOD = 0
END IF
C      IF (RETCOD .EQ. 0) THEN
C          get space-time descriptives
READ (IFL,1001) STDAT, TST, TUN, NDELT,
>          (BASN(I), SKPN(I), I = 1, 2)
C          write header
WRITE (OFL,2240)
IF (NORO .EQ. 1) THEN
    new data set needs dummy starting group
    CALL WSTAGP ( WDMSFL, DSN, STDAT, TUN, TST, NDELT,
>          RETCOD )
    IF (RETCOD .EQ. 0) THEN
        dummy group added
        WRITE (OFL,2250) NDELT, TST, TUN, STDAT
    ELSE
        error adding dummy group
        WRITE (OFL,2251) RETCOD
    END IF
END IF
IF (RETCOD.EQ.0) THEN
    set dimensions and how many values in a group (X*Y)
    NDIM = 2
    NVAL = NUMXY(1) * NUMXY(2)
C      begin loop for new space time group
300  CONTINUE
    C      get group start date, time step and number of time steps
    READ (IFL,1300) STDAT, TST, TUN, NDELT
    IF (NDELT .EQ. 0) THEN
        end of data
        RETCOD = 1
    ELSE
        add the group
        CALL WSTAGP (WDMSFL,DSN,STDAT,TUN,TST,NDELT,
O          RETCOD)
        IF (RETCOD.NE.0) THEN
            error adding group
            WRITE (OFL,2301) RETCOD, STDAT, TST, TUN, NDELT
        END IF
    END IF

```

### Create space-time groups

The starting date, the time step and time units, and the number of time steps is read from the input file. Subroutine WSTAGP is called to allocate space in the data set for the space time data to be added.



### Add space time data

The array of space-time data is read from the input file. The data is arranged in the file as "....". The data is arranged in the array RBUFF as "....". The space-time data are written to the data set using subroutine WSTPTR.

```
C          IF (RETCOD .EQ. 0) THEN
C              get the space-time data from input file and write to dsn
C              WRITE (OFL,2302) NDELT, TST, TUN, STDAT
C              DO 350 J= 1, NDELT
C                  read data values for this time step
C                  M1 = 1
C                  MN = NUMXY(1)
C                  DO 340 IY = 1, NUMXY(2)
C                      for variable iy, read all nodes
C                      READ (IFL,1340) (RBUFF(M), M = M1, MN)
C                      M1 = MN + 1
C                      MN = MN + NUMXY(1)
C 340          CONTINUE
C              put the data
C              CALL WSTPTR (WDMMSFL, DSN, STDAT, NDIM, NUMXY, BASN, SKPN,
C                           I           NVAL, RBUFF,
C                           O           RETCOD)
C              IF (RETCOD.EQ.0) THEN
C                  space time data successfully added
C                  WRITE (OFL,2341) TST, TUN, STDAT, NVAL
C                  increment date
C                  CALL TIMADD (STDAT, TUN, TST, LEN1,
C                               O           SEDAT)
C                  CALL COPYI (LEN6, SEDAT,
C                               O           STDAT)
C              ELSE
C                  error adding space-time data
C                  WRITE (OFL,2342) RETCOD, STDAT, NDIM, NVAL
C                  WRITE (OFL,2343) (NUMXY(I), BASN(I), SKPN(I), I=1,NDIM)
C              END IF
C 350          CONTINUE
C          END IF
C          IF (NDELT .GT. 0 .AND. RETCOD .EQ. 0) GO TO 300
C          end loop for new space-time groups
C      END IF
C      summarize what's on data set
C      RETCOD = 0
C      GRPIND= 0
C      write header
C      WRITE (OFL,2400) DSN
C 400          CONTINUE
C          GRPIND= GRPIND+ 1
C          CALL WSTGSU (WDMMSFL, DSN, GRPIND,
C                           O           STDAT, SEDAT, TUN, TST, NVAL, FRAC, RETCOD)
C          IF (RETCOD.EQ.0) THEN
C              print summary
C              WRITE (OFL,2401) GRPIND, TST, TUN, NVAL, FRAC, STDAT, SEDAT
C          ELSE IF (RETCOD.EQ.-49) THEN
C              finished summarizing groups
C              WRITE (OFL,2403) DSN
C          ELSE
C              error trying to retrieve summary data
C              WRITE (OFL,2402) RETCOD, DSN, GRPIND
C          END IF
C          IF (RETCOD.EQ.0) GO TO 400
C          accept end of data return code
C          IF (RETCOD .EQ. -49) RETCOD = 0
```

### Summarize contents of space-time data set

WSTGSU is called to get general information about each group of data in the space-time data set. The group information includes the begin and end dates of the group, time units and time step of the group, number of values at each time step, and the fraction of the group containing data. A -49 return code indicates the requested group does not exist, in this case we assume it to mean there are no more groups in the data set.

**retrieve selected nodes**

\*\*\*Need description of what is going on here \*\*\*\*

```
C      IF (RETCOD .EQ. 0) THEN
C          at 5 selected nodes, retrieve 3 selected variables
C          NUMXY(1) = 5
C          NUMXY(2) = 3
C          number of values = nodes * variables
C          NVAL = NUMXY(1) * NUMXY(2)
500      CONTINUE
C          define time period and time step, select data values
C          READ (IFL,1001) STDAT, TST, TUN, NDELT,
C                           (BASN(I), SKPN(I), I = 1, 2)
C          >      IF (NDELT .EQ. 0) THEN
C              finished retrieving data
C              RETCOD = 1
C          ELSE
C              retrieve NVAL values nodes for each time step
C              I= 0
550      CONTINUE
C              I= I+ 1
C              get a block(or part) of real data from a group
C              CALL WSTGTR(WDMMSFL,DSN,STDAT,NDIM,NUMXY,BASN,SKPN,NVAL,
C                           XBUFF(1,1,I),RETCOD)
C              O      IF (RETCOD.NE.0) THEN
C                  error reading data
C                  WRITE (OFL,2551) RETCOD, I, STDAT,NDIM,NUMXY,BASN,
C                                   SKPN, NVAL
C              >
C              ELSE
C                  next time step, calc ending time and make it start time
C                  CALL TIMADD (STDAT,TUN,TST,LEN1,
C                               SEDAT)
C                  CALL COPYI (LEN6,SEDAT,
C                               STDAT)
C                  END IF
C                  IF (I .LT. NDELT .AND. RETCOD .EQ. 0) GO TO 550
C              END IF
C              IF (RETCOD.EQ.0) THEN
C                  WRITE (OFL,2560) NDELT
C                  WRITE (OFL,2561) (((XBUFF(I,J,K),J=1,3),I=1,5),K=1,NDELT)
C              END IF
C              IF (RETCOD .EQ. 0) GO TO 500
C          END IF
C          WRITE (OFL,2900)
C
C          RETURN
C
END
```

# CASE STUDIES--text and template

The CSTXT routine reads text and table template records from a WDM file. Table xx contains a summary of the WDM toolkit routines called by CSTXT.

Table xx.--WDM toolkit routines called by text and table template case study.

Routine	How used
MSFBC	get location of information in WDM file
MSBCS	splits up block control word to get information about the group
MSGTE	get a record of text of the group
MSSKB	position pointers to end of current data block
DNXDV	moves to next data position and returns integer equivalent of the data value

Table yy.--parameters used by WDM toolkit routines for text and table template data sets.  
\*\*\*\*\* needs some work! \*\*\*\*\*

Parameter	Explanation
QWORD	block control word, contains CLASS, ID, ORDER, and TLEN, described below
CLASS	class of information 1 - 1-dimensional parameter    4 - menu 2 - 2-dimensional parameter    5 - file 3 - text
ID	identification for portion of group *****needs defn*****
ORDER	order of information
DREC	record number of data on WDM file
DPOS	position of data on data record DREC
GLEN	counter to keep track of when to read off WDM file, initialized to 0 for first call for group.
MLEN	number of characters retrieved so far. Initialized to 0 for first call for group

## Assumptions

Assumes that the requested cluster and group exist in the WDM file.

## Enhancements

### Specifications

Standard conventions have been used to declare dummy arguments, local variables, and externals.

```
C
C
C          SUBROUTINE  CSTXT
I          (MESSFL,OFL,IFL)
C
C          + + + PURPOSE + + +
C          case study example for text and table template datasets
C
C          + + + DUMMY ARGUMENTS + + +
C          INTEGER      MESSFL,OFL,IFL
C
C          + + + ARGUMENT DEFINITIONS + + +
C          MESSFL - Fortran unit number of WDM file containing attribute info
C          OFL   - Fortran unit number of output text file
C          IFL   - Fortran unit number of input file
C
C          + + + LOCAL VARIABLES + + +
C          INTEGER      SCLU,SGRP,DREC,DPOS,QWORD,CLASS,ID,ORDER,TXTLEN,
I          I80,GLEN,MLEN,OLEN,MORE,I, RETCOD
CHARACTER*1 TXBUFL(80)
C
C          + + + EXTERNALS + + +
EXTERNAL     WMSFBC,WMSBCS,WDNXDV,WMSGTE,WMSKBD
```

## Input and Output Formats

Input formats are in the range 1000-1999. Output formats are in the range 2000-2999.

```

C   + + + INPUT FORMATS + +
C   1000 FORMAT ( 2I5 )
C
C   + + + OUTPUT FORMATS + +
2001 FORMAT ( ' Reading from MESSFL', I3,
    >     /, ' cluster', I4, ', group', I4 )
2110 FORMAT ( ' Error, invalid id', I5, ', from BCW', I10 )
2120 FORMAT ( ' Text table block:'
    >     /, ' Class', I5, ', ID', I3, ', Order', I4,
    >     /, ' Txtlen', I5, ', Glen', I5, ', Mlen', I5,
    >     /, ' Olen', I5, ', More', I5,
    >     /, ' text:', 40A1, /, 11X, 40A1 )
2121 FORMAT ( ' Numeric table block:'
    >     /, ' Class', I5, ', ID', I3, ', Order', I4,
    >     /, ' Txtlen', I5 )
2200 FORMAT ( ' Text and table template case study completed.' )
C
C   + + + END SPECIFICATIONS + +
C
C   I80 = 80
C begin loop for new cluster and group
RETCOD = 0
100 CONTINUE
C   get cluster number and group number
READ (IFL,1000) SCLU, SGRP
IF (SCLU .EQ. 0) THEN
    finished reading text and table template data sets
    RETCOD = 1
ELSE
    get address where text or table template info starts
    CALL WMSFBC (MESSFL,SCLU,SGRP,
                  DREC,DPOS,QWORD)
    IF (QWORD.LE.0) THEN
        group does not exist in cluster
        WRITE (OFL,2001) MESSFL, SCLU, SGRP

```

## C1 get block control word

WMSFBC is called to get the record number of the block control word on the WDM file (DREC), the position in record DREC of the block control word (DPOS), and the block control word (QWORD) of the text or table template. A QWORD of 0 indicates the group does not exist in the cluster. \*\*\*\* the documentation states that a STOP is encountered if the group does not exist \*\*\*\*

## get parameters

WMSBCS is called to split up the block control word (QWORD) to determine the class of information (CLASS), identification for portion of group (ID), the order of information (ORDER), and the total number of characters in this block (TXTLEN). Table yy contains more detailed information on these parameters. This loop is repeated until all blocks in the group have been processed. A zero or negative value of ID indicates an invalid block control word.

```

C
C   110
C
C   ELSE
C   process all text/table blocks .
CONTINUE
    get text/table parameters
    CALL WMSBCS (QWORD,
                  CLASS, ID, ORDER, TXTLEN)
    IF (ID .LE. 0) THEN
        invalid ID computed from QWORD
        WRITE (OFL,2110) QWORD, ID

```

## get a record of text

WMSGTE is called to read the text, one record at a time. It is called until all records have been read. A value of 1 for MORE indicates that there is more text in the group, a value of 0 indicates there is no more text available \*\*\*\* does 0 mean nothing retrieved this time or this is that last of the text? The documentation is not clear. \*\*\*\* The counters GLEN and MLEN are initialized to zero each time WMSGTE is called for a new group and cluster. OLEN is the actual number of characters retrieved and the characters are contained in TXBUF1.

## skip data block

WMSSKB is called to update the pointers DREC and DPOS to the end of the numeric table block.

```

C   >
C   ELSE IF (ID.EQ.16 .OR. ID.EQ.17 .OR. ID.EQ.2 .OR.
           ID.EQ.9 .OR. ID.EQ.20) THEN
    header, table name
    GLEN= 0
    MLEN= 0
CONTINUE
    get a record of text from WDM file
    CALL WMSGTE (MESSFL,TXTLEN,I80,
                  DREC,DPOS,GLEN,MLEN,
                  OLEN,TXBUF1,MORE)
    WRITE (OFL,2120) CLASS, ID, ORDER, TXTLEN,GLEN,MLEN;
                      OLEN,MORE, (TXBUF1(I), I=1,OLEN)
    IF (MORE.NE.0) GO TO 120

```

```

C   ELSE
    numeric table block, skip
    WRITE (OFL,2121) CLASS, ID, ORDER, TXTLEN
    CALL WMSSKB (MESSFL,TXTLEN,
                  DREC,DPOS)
END IF
END IF

```

**get next table block control word**  
WDNXDV is called to update the position pointers and  
get the block control word (QWORD) for the next block  
of text/table information. A negative or zero QWORD  
indicates the end of the text or template..

C get next table block control word  
CALL WDNXDV (MESSFL,  
M DREC,DPOS,  
O QWORD)  
IF (QWORD.GT.0) GOTO 110  
END IF  
C WRITE (OFL,2200)  
C RETURN  
END

# CASE STUDIES--attribute

The CSATR routine gets information from the attribute message file about requested attributes.

Table xx.--WDM toolkit routines called by attribute case study.

Routine	How used
WDSAFI	search attribute data sets for a requested attribute, returns the index number and the full name of the attribute.
WDSAGY	gets general information about an attribute, including name, type, and length.

## Assumptions

Attribute data sets exist in the message WDM file. Up to 10 attribute names or partial names can be requested.

## Enhancements

Add logic to list general information about all of the attributes in the attribute data sets.

## Specifications

Standard conventions have been used to declare dummy arguments, local variables, equivalences, and externals.

```
C          C
C          C
C          C          SUBROUTINE  CSATR
C          I          (MESSFL,WDMSFL,OFL,IFL)
C          C          + + + PURPOSE + +
C          C          case study example for attribute data sets
C          C          + + + DUMMY ARGUMENTS + +
C          INTEGER      MESSFL,WDMSFL,OFL,IFL
C          C          + + + ARGUMENT DEFINITIONS + +
C          C          MESSFL - Fortran unit number of WDM file containing attribute info
C          C          WDMSFL - Fortran unit number of WDM file used for case study data
C          C          OFL   - Fortran unit number of output text file
C          C          IFL   - Fortran unit number of input text file
C          C          + + + LOCAL VARIABLES + +
C          INTEGER      I,SAIND,DBTR,SATYP,SALEN,SARQWD,SAUPFG,RETCOD,DSN,
C          1          SAIVAL(1),SARET,TLEN,DREC,DPOS,NSA,J
C          REAL         SARVAL(1),ATRDEF
C          CHARACTER*1  SAINAM(6),SAONAM(6),SANAM(6),BINK,SADESC(47),
C          >          SANAMS(6,10)
C          C          + + + EQUIVALENCES + +
C          EQUIVALENCE (ATRDEF,ATIDEF)
C          INTEGER      ATIDEF
C          C          + + + EXTERNALS + +
C          EXTERNAL     ZIPC, WDSAFI, WDSAGY, WDBSGC, WDBSGI, WDBSGR, WDBSAD
C          EXTERNAL     WADGDF, WADGDS, WADGHL
```

## Input and output formats

Input formats are in the range 1000-1999. Output formats are in the range 2000-2999.

```
C      + + + INPUT FORMATS + + +
C      1000 FORMAT ( I4, 10( 1X,6A1 ) )
C
C      + + + OUTPUT FORMATS + + +
C      2000 FORMAT ( ' Looking for matches and near matches to',
C                  >     I3, ' attributes',
C                  >     //, ' look <- found attribute ->',
C                  >     /, ' for name index type len ',
C                  >     '---- description and additional information -->',
C                  >     /, '----- ', 47('`') )
C      2101 FORMAT ( ' Error', I4, ' , no attributes in WDM file', I4 )
C      2102 FORMAT ( ' Notice, did not find attribute ', 6A1 )
C      2110 FORMAT ( 2(1X,6A1), 3I5, 1X, 47A1,
C                  >     /, 30X, I10, ' - DPTR',
C                  >     /, 30X, I10, ' - SARQND    ', I10, ' - SAUPFG' )
C      2112 FORMAT ( 30X, I10, ' - TLEN',
C                  >     /, 30X, I10, ' - DREC    ', I10, ' - DPOS  ' )
C      2113 FORMAT ( 30X, ' No help available' )
C      2114 FORMAT ( 30X, I10, ' - default value' )
C      2115 FORMAT ( 30X, F10.2, ' - default value' )
C      2800 FORMAT ( ' Attribute case study completed.' )
```

```
C      + + + END SPECIFICATIONS + + +
```

```
C      get attributes to look for
```

```
READ (IFL,1000) NSA, ((SANAMS(I,J),I=1,6),J=1,NSA)
```

```
WRITE (OFL,2000) NSA
```

```
J = 0
```

```
100 CONTINUE
```

```
C      begin loop to look for match to each SANAMS.
J = J + 1
```

## Look for match

WDSAIFI is called to see find information on the attribute SANAMS(). If SANAMS contains a complete attribute name, only one match will be found. If SANAMS contains a partial name, the first attribute, beginning at SAIND, that begins with the requested name will be found. SAIND will be modified to the current attribute index number. A return code of 0 indicates there are no possible additional matches, -112 indicates there may be additional matches, a -111 indicates no match was found. WDSAIFI is called until all attributes have been checked for a match.

## Get information about attribute

WDSAGY is called to get information about the attribute. This includes the complete attribute name, a pointer to additional information, type, length. WADGDS is called to get the description of the attribute.

## Help Information

WADGHL is called to get the location of the help information and the total number of characters in the help information. If there is no help information, the length is returned as 0.

```
110 SAIND= 1
C      CONTINUE
C      begin loop to look for matches to SANAMS (J)
CALL WDSAIFI (MESSFL,SANAMS(1,J),
M           SAIND,
O           SANAM,RETCOD)
IF (RETCOD .EQ. -110) THEN
      there are no attribute data sets in WDM file MESSFL
      WRITE (OFL,2101) MESSFL
ELSE IF (RETCOD .EQ. -111) THEN
      no match found for attribute SANAMS(J)
      WRITE (OFL,2102) (SANAMS(I,J),I=1,6)

C      ELSE IF (RETCOD.EQ.0 .OR. RETCOD.EQ.-112) THEN
      found a match, get attribute characteristics
      CALL WDSAGY (MESSFL,SAIND,
C           SANAM,DPTR,SATYP,SALEN,SARQND,SAUPFG)
      get description of the attribute
      CALL WADGDS (MESSFL,DPTR,
O           SADESC)
      WRITE (OFL,2110) (SANAMS(I,J),I=1,6), SANAM, SAIND,
>           SATYP, SALEN, SADESC,
>           DPTR, SARQND, SAUPFG
      get location of help information
      CALL WADGHL (MESSFL,DPTR,
O           TLEN,DREC,DPOS)
      IF (TLEN .GT. 0) THEN
          help available
          WRITE (OFL,2112) TLEN, DREC, DPOS
      ELSE
          no help available
          WRITE (OFL,2113)
      END IF
```

**get attribute default value**

WADGDF is called to get the default value for the attribute from the attribute data set. The value is returned as a real number, if the attribute is an integer, use the equivalenced value ATIDEF.

```
C      get default value for the attribute
      CALL WADGDF (MESSFL,DPTR,SATYP,
                  ATRDEF)
      O
C      IF (SATYP.EQ.1) THEN
          write out default for integer attribute
          WRITE(OFL,2114) ATIDEF
      ELSE IF (SATYP.EQ.2) THEN
          write out default for real attribute
          WRITE(OFL,2115) ATRDEF
      END IF
      END IF
C      up for another partial match to SANAMS(J)
      IF (RETCOD .EQ. -112) GO TO 110
C      up for new attribute
      IF (J .LT. NSA .AND. RETCOD .NE. -110) GO TO 100
C      WRITE (OFL,2800)
C
      RETURN
      END
```

## CASE STUDIES--table data

This example does the following:

- create a table data set in an existing WDM file,
- add attributes for the data set,
- put table template on WDM file
- read a flat file that has 50 rows of data with a character string in the first field, real numbers in the next two fields, followed by a double precision number, and an integer number,
- add the first 40 rows of data to the data set,
- add the remaining 10 rows to the data set,
- retrieve the two real number fields, create a third field as a product of the two, and write the results as a new table in a second data set, retrieve rows 20 to 40, multiply the real numbers by 2, and write the results to a third table in a third data set,
- summarize contents of a table data set, and
- delete a table.

To perform these tasks the following toolkit subroutines are used:

WDBCRL, WDDSCL, WDTBTM, WTBPUS, WDTBSU, WDBSAC, WTBGET,  
WTBCOD, WDTBFX, WDTBDL, WTBISP, WTBDSP

An excerpt (i.e., the first ten of 50 rows of data) from the external file that provides the initial data for this example is shown below. The data were synthesized to allow demonstration of the input and manipulation of different data types in a table data set. The fields of the external file, from left to right, contain a column of character data, two columns of real data, a column of double precision data, and a column of integer data.

Data for Table Data Set	A0	0.0	49.0	0.00000	1
	A1	1.0	48.0	0.00001	2
	A2	2.0	47.0	0.00002	3
	A3	3.0	46.0	0.00003	4
	A4	4.0	45.0	0.00004	5
	A5	5.0	44.0	0.00005	6
	A6	6.0	43.0	0.00006	7
	A7	7.0	42.0	0.00007	8
	A8	8.0	41.0	0.00008	9
	A9	9.0	40.0	0.00009	10

The following pages contain code segments and descriptions for a subroutine named CSTA that performs various operations related to input and manipulation of data in WDM table data sets.

## Specifications

```
C
C
C
      SUBROUTINE CSTA
      I          (MESSFL,WDMFL,OFL)
C
C      + + + PURPOSE + +
C      case study example for table datasets
C
C      + + + DUMMY ARGUMENTS + +
      INTEGER      MESSFL,WDMFL,OFL
C
C      + + + ARGUMENT DEFINITIONS + +
C      MESSFL - Fortran unit number of WDM file containing attribute info
C      WDMFL - Fortran unit number of WDM file used for case study data
C      OFL    - Fortran unit number of output text file
C
C      + + + PARAMETERS + +
      INTEGER      MXTBTL,MAXTAB
      PARAMETER (MXTBTL=1000,MAXTAB=10)
C
C      + + + LOCAL VARIABLES + +
      INTEGER      I,J,RETCOD,DSTYPE,
      1          DATFIL,ENDFG,ODSN,DSN,SAIND,SALEN,
      2          TCLU,TGRP,TABIND,NROW,TFLDS,TNUM(4),TLEN(6),TCOL(6),
      3          TSPA,TGRPPT,AFLDS,ANUM(4),ALEN(6),ACOL(6),
      4          ASPA,ACLU,AGRP,DATFLG,FROW,FSPA,
      5          TABBAS,TABCNT,TABID(MAXTAB),TABDIM(MAXTAB),
      6          PDATVL(MAXTAB),LREC,NCOL,NEXT
      REAL        RBUFF(8,50),NWBUFF(50)
      CHARACTER*1  CBUFF(80,50),TTYP(6),ATYP(6),MFID(2),
      1          TABNM(16,MAXTAB)
      CHARACTER*16 TABNAM
C
C      + + + EQUIVALENCES + +
      EQUIVALENCE (SABUFF,CBUFF)
      CHARACTER*80 SABUFF
C
C      + + + EXTERNALS + +
      EXTERNAL     WDBCRL, WDDSCL, WDTBTM, WTBPUT, WDTBSU
      EXTERNAL     WDBSAC, WTBGET, WTBCOD, WDTBFX, WDTBDL
      EXTERNAL     WTBISP, WTBDSP
C
C      + + + DATA INITIALIZATIONS + +
      DATA MFID/'X','X'
C
C      + + + INPUT FORMATS + +
      1000 FORMAT (80A1)
C
C      + + + OUTPUT FORMATS + +
      2000 FORMAT (10X,16A1,1X,2A1)
      2005 FORMAT (10X,7I8)
      2010 FORMAT (' table details for: ',A16,I8)
      2020 FORMAT (' ',2I8,1X,2A1,1X,4I8)
C
C      + + + END SPECIFICATIONS + + +
```

Standard coding conventions have been used to declare local variables, specify parameters, equivalences, functions, externals, data initializations, and formats.

**Create Table Data Set**

```

C      set dataset type for table
DSTYPE= 2
C      add a new dataset label
DSN= 20
      WRITE(OFL,*) 'creating table dataset',DSN
C      add the general part of the label
      CALL WDBCR (WDMSFL,DSN,DSTYPE,
O          RETCOD)
      IF (RETCOD .NE. 0) THEN
          WRITE(OFL,*) 'table dataset',DSN,' not created, retcod:',RETCOD

```

First, the data set type is specified as a table (DSTYPE= 2). Next, data set number 20 is specified as a new data set in the WDM file. Then the general label of the new data set is assigned using subroutine WDBCR. If an error is encountered, a non zero return code (RETCOD) is returned. See Appendix D for definitions of possible return codes from this routine.

**Add Attribute to Table Data Set**

```

C      now add attribute
SAIND = 10
SALEN = 80
SABUFF= 'Dataset for table data case study.'
CALL WDBSAC (WDMSFL,DSN,MESSFL,SAIND,SALEN,CBUFF,
O          RETCOD)

```

Next, a character type attribute is added for data set 20 by using subroutine WDBSAC. Input attributes for WDBSAC include the WDM file unit number (WDMSFL), the data set number (DSN), the attribute file unit number (MESSFL), the attribute index number in the attribute file (SAIND), the attribute length (SALEN), and the attribute value (CBUFF). The values for SAIND and SALEN are specified as 10 and 80, respectively. The value of the attribute (CBUFF) is the alphanumeric string 'Dataset for table data case study', as assigned to the buffer SABUFF and equivalenced to CBUFF.

**Put Table Template on WDM File**

```

C      put table template on wdm file
TCLU = 12
TGRP = 1
TABIND= 1
NROW = 50
CALL WDTBTM (MESSFL,MFID,TCLU,TGRP,WDMSFL,DSN,TABIND,NROW,
O          TFLDS,TNUM,TTYP,TLEN,TCOL,TSPA,TABNAM,TGRPPT,
O          AFLDS,ANUM,ATYP,ALEN,ACOL,ASPA,ACLU,AGRP,
O          RETCOD)

```

Next, table template information is put onto the WDM file by using subroutine WDTBTM. The subroutine requires values for eight input arguments: the unit number of the attribute file (MESSU), the message file identification (MFID), the data set number of the text and table template data set in which the template is stored (TCLU), the table group (TGRP), the unit number for the WDM file (WDMSFL), the number of the data set in which the data will be stored (DSN), the table index (TABIND) and the number of table rows (NROW). Values for MESSU, WDMSFL and DSN are assigned earlier in the program. The value for MFID is provided in a data statement, and the values of TCLU, TGRP, TABIND and NROW are assigned immediately before calling the subroutine. The table template is stored as table group 1 of text and tables data set number 12. Refer to Section 4.5 to review the specifications for this table.

\*\*\*\*what does the subroutine actually do?

**Create Table Data Set**

```

C      set dataset type for table
DSTYPE= 2
C      add a new dataset label
DSN= 20
        WRITE(OFL,*) 'creating table dataset',DSN
C      add the general part of the label
        CALL WDBCRL (WDMFL,DSN,DSTYPE,
O              RETCOD)
        IF (RETCOD .NE. 0) THEN
            WRITE(OFL,*) 'table dataset',DSN,' not created, retcod:',RETCOD

```

First, the data set type is specified as a table (DSTYPE= 2). Next, data set number 20 is specified as a new data set in the WDM file. Then the general label of the new data set is assigned using subroutine WDBCRL. If an error is encountered, a non zero return code (RETCOD) is returned. See Appendix D for definitions of possible return codes from this routine.

**Add Attribute to Table Data Set**

```

C      now add attribute
SAIND = 10
SALEN = 80
SABUFF= 'Dataset for table data case study.'
CALL WDBSAC (WDMFL,DSN,MESSFL,SAIND,SALEN,CBUFF,
O              RETCOD)

```

Next, a character type attribute is added for data set 20 by using subroutine WDBSAC. Input attributes for WDBSAC include the WDM file unit number (WDMFL), the data set number (DSN), the attribute file unit number (MESSFL), the attribute index number in the attribute file (SAIND), the attribute length (SALEN), and the attribute value (CBUFF). The values for SAIND and SALEN are specified as 10 and 80, respectively. The value of the attribute (CBUFF) is the alphanumeric string 'Dataset for table data case study', as assigned to the buffer SABUFF and equivalenced to CBUFF.

**Put Table Template on WDM File**

```

C      put table template on wdm file
TCLU = 12
TGRP = 1
TABIND= 1
NROW = 50
CALL WDTBTM (MESSFL,MFID,TCLU,TGRP,WDMFL,DSN,TABIND,NROW,
O             TFLDS,TNUM,TTYP,TLEN,TCOL,TSPA,TBNAME,TGRPPT,
O             AFLDS,ANUM,ATYP,ALEN,ACOL,ASPA,ACLU,AGR,
O             RETCOD)

```

Next, table template information is put onto the WDM file by using subroutine WDTBTM. The subroutine requires values for eight input arguments: the unit number of the attribute file (MESSU), the message file identification (MFID), the data set number of the text and table template data set in which the template is stored (TCLU), the table group (TGRP), the unit number for the WDM file (WDMFL), the number of the data set in which the data will be stored (DSN), the table index (TABIND) and the number of table rows (NROW). Values for MESSU, WDMFL and DSN are assigned earlier in the program. The value for MFID is provided in a data statement, and the values of TCLU, TGRP, TABIND and NROW are assigned immediately before calling the subroutine. The table template is stored as table group 1 of text and tables data set number 12. Refer to Section 4.5 to review the specifications for this table.

\*\*\*\*what does the subroutine actually do?

**Read Table Data  
From a Formatted  
ASCII File**

```
C      open data file
DATFIL= 61
OPEN (UNIT=DATFIL,FILE='CSTA.DAT',STATUS='OLD',ERR=20)
GO TO 30
CONTINUE
C      get here on open error
      WRITE (OFL,*) ' ERROR, could not open file CSTA.DAT'
      RETCOD= -1
30     CONTINUE
      IF (RETCOD.EQ.0) THEN
C      now read data
      ENDFG= 0
      DO. 50 I= 1,50
          IF (ENDFG.EQ.0) THEN
              continue reading data
              READ (DATFIL,1000,ERR=40,END=40) (CBUFF(J,I),J=1,80)
              GO TO 45
40     CONTINUE
C      get here on read error or end of file
      ENDFG= 1
      CLOSE (UNIT=DATFIL)
45     CONTINUE
      END IF
50     CONTINUE
```

First, the external file containing the input data, in this case named CSTA.DAT, is opened by using a standard Fortran OPEN statement. An excerpt from the contents of the data set are provided at the introduction of this example. After a check has been performed to assure that the data set has been successfully opened, a standard formatted Fortran READ statement is used to perform the reading operation. Rows of data are read into a buffer called CBUFF one at a time until all fifty rows have been read. Since the data being read is a mixture of data types (character and real), a character type buffer is used that can store all the data types. When the program encounters an end-of-file condition in the external data file, the program exits the monthly loop and reading of data ceases.

**Convert  
Character Buffer  
Into Real Data  
Buffer**

```
C      convert character buffer into real data buffer
CALL WTBCOD (TFLDS,NROW,TSPA,TLEN,TTYP,TCOL,
I
CBUFF,MXTBTL,
O
RBUFF,RETCOD)
```

Regardless of actual data type, all data is stored as real numbers in WDM table data sets. Subroutine WTBCOD is used to convert the character type buffer CBUFF to a real number type buffer called RBUFF. With the exception of CBUFF, NROW and MXTBTL, the values for the subroutine's input arguments originate in the table template definition in the sequential message file and are passed to the program by WDTBTM, the subroutine used to put table template information into the WDM file. The value of NROW is assigned earlier in the subroutine, and the value of MXTBTL is set using a parameter statement in the specifications section. Argument definitions for WTBCOD are available in Appendix D.

**Put Rows of Data  
Into Table Data  
Set**

```
C      now put first 40 rows of data to wdm file
I      DATFLG= 1
O      FROW = 1
NROW = 40
FSPA = 1
CALL WTBPPUT (WDMMSFL,DSN,TABNAM,TABIND,DATFLG,
I      FROW,NROW,FSPA,TSPA,RBUFF,
O      RETCOD)
C      IF (RETCOD.EQ.0) THEN
I      successful put of table data
O      WRITE (OFL,*) NROW,' rows of data successfully put to',
I      ' dataset',DSN
1      ELSE
I      WRITE (OFL,*) ' ERROR putting data to dataset',DSN,
1      ', retcod:',RETCOD
C      END IF
I      now put the last 10 rows of data to wdm file
O      FROW = 41
NROW = 10
FSPA = 1
CALL WTBPPUT (WDMMSFL,DSN,TABNAM,TABIND,DATFLG,
I      FROW,NROW,FSPA,TSPA,RBUFF,
O      RETCOD)
C      IF (RETCOD.EQ.0) THEN
I      successful put of table data
O      WRITE (OFL,*) NROW,' rows of data successfully put to',
I      ' dataset',DSN
1      ELSE
I      WRITE (OFL,*) ' ERROR putting data to dataset',DSN,
1      ', retcod:',RETCOD
C      END IF
```

Subroutine WTBPPUT is used to copy table data as real numbers from RBUFF to the WDM data set, in this case data set 20. To demonstrate an additional capability, the example program copies the data contained in RBUFF into the table data set by using two calls to WTBPPUT: initially, the first 40 rows of data are copied, and then the final 10 rows are copied. For the first call, values for arguments specify that the data type (DATFLG) is a main table, writing of data should begin at the first row and continue for 40 rows, and writing should begin at the first available data space (FSPA= 1). Values for the table name (TABNAM), table identifier number (TABIND) and the space for data in each row (TSPA) originate in the template definition and are provided to the program by subroutine WDTBTM. For the second call to WTBPPUT, the argument values specify that the writing of data should begin at the 41st row and continue for 10 rows, and writing should begin at the first available data space.

**Get Real Data from Table Data Set**

```
C      get two real data fields
FROW= 1
NROW= 50
FSPA= 4
TSPA= 2
CALL WTBGET (WDMSFL,DSN,TABNAM,TABIND,DATFLG,
I      FROW,NROW,FSPA,TSPA,
O      RBUFF,RETCOD)
IF (RETCOD.EQ.0) THEN
C      successful get of real fields
      WRITE (OFL,*) NROW,' rows of data for',TSPA,' fields',
I      ' successfully read from dataset',DSN
ELSE
      WRITE (OFL,*) ' ERROR getting data from dataset',DSN,
I      ', retcod:',RETCOD
END IF
```

The next operation performed in the example is focused on the two fields of real data contained in the second and third columns of the table (see excerpt of data at the beginning of this section).

The purpose of this portion of the program is to demonstrate the reading of data from a table data set, followed by manipulation of the data to produce a new collection of data, that is in turn stored in a new table data set. Subroutine WTBGET is used to read the two fields of real number data from data set 20 into the buffer RBUFF. Values for the subroutine arguments specify that reading begin at the first row and continue for 50 rows. In each row reading begins at the fourth space (FSPA= 4) and continues for 2 spaces (TSPA= 2). A "space" can store 4 alphanumeric characters or one real number; two spaces store a double precision number. Hence, in each row of table data set 20, the first three spaces store the 12 character alphanumeric field, the fourth and fifth spaces store the two real numbers, the sixth and seventh spaces store the double precision number, and the integer number is stored in the eighth space.

**Generate New Field of Data**

```
C      generate a new field of data
DO 60 I= 1,50
      NWBUFF(I)= RBUFF(1,I)*RBUFF(2,I)
60      CONTINUE
```

Next, the product of the two real numbers in each of the 50 rows of data that has been read into RBUFF from data set 20 is computed and put into a new buffer named NWBUFF.

**Create Data Set for New Data**

```
C      create a new dataset for the new field
ODSN = DSN
DSN = DSN+ 1
C      copy 1st dataset label to 2nd dataset
CALL WDDSCL (WDMSFL,ODSN,DSN,
O      RETCOD)
IF (RETCOD.EQ.0) THEN
C      successful copy of label
      WRITE (OFL,*) ' label copied from dataset',ODSN,
I      ' to dataset',DSN
ELSE
      WRITE (OFL,*) 'error on label copy, retcod:',RETCOD
END IF
```

Next, a new table data set, data set 21, is defined by copying the data set label from data set 20 using subroutine WDDSCL.

**Put Template for New Table In New Data Set**

```

C      put new field on new dataset
      TCLU= 12
      TGRP= 2
      NROW= 50
C      first put template
      CALL WDTBTM (MESSFL, MFID, TCLU, TGRP,
I          WDMSFL, DSN, TABIND, NROW,
O          TFLDS, TNUM, TTYP, TLEN, TCOL, TSPA,
O          TABNAM, TGRPPT, AFLDS, ANUM, ATYP,
O          ALEN, ACOL, ASPA, ACLU, AGRP, RETCOD)

```

After data set 21 has been successfully created, subroutine WDTBTM is used to put the table template for the new data field into the new data set. The template information is retrieved from the second table group (TGRP) in text and tables template data set (TCLU) number 12. Refer to Section 4.5 to review the specification for the table.

**Put New Data In Data Set**

```

C      now put data
      FROW= 1
      FSPA= 1
      CALL WTBPUT (WDMSFL, DSN, TABNAM, TABIND, DATFLG,
I          FROW, NROW, FSPA, TSPA, NWBUFF,
O          RETCOD)
      IF (RETCOD.EQ.0) THEN
C          successful put of table data
          WRITE (OFL,*) NROW, ' rows of data successfully put to',
I              ' dataset', DSN
      ELSE
          WRITE (OFL,*) ' Problem putting data to dataset', DSN,
I              ', retcod:', RETCOD
      END IF

```

Finally, the values contained in NWBUFF are put into data set 21 by using WTBPUT. All the values are written to the data set beginning at the first space (FSPA= 1) of the first row (FROW= 1).

**Get Additional Data**

```

C      get 20 rows starting at row 21
      FROW= 21
      NROW= 20
      CALL WTBGET (WDMSFL, DSN, TABNAM, TABIND, DATFLG,
I          FROW, NROW, FSPA, TSPA,
O          NWBUFF, RETCOD)
      IF (RETCOD.EQ.0) THEN
C          successful get of real fields
          WRITE (OFL,*) NROW, ' rows of data for', TSPA, ' fields',
I              ' successfully read from dataset', DSN
      ELSE
          WRITE (OFL,*) ' Problem getting data from dataset', DSN,
I              ', retcod:', RETCOD
      END IF

```

The program now performs a similar sequence of operations involving the reading of data from a table data set, followed by manipulation of the data to produce a new collection of data, that is in turn stored in a new table data set. This example, however, demonstrates the retrieval and manipulation of data starting at a location in a table data other than its beginning.

Again, subroutine WTBGET is used to read the data from the table data set. For this example, the data is read from rows 21 through 40 of data set 21 into the first 20 spaces of NWBUFF.

**Generate New Field of Data**

```
C      generate another field of data
      DO 70 I= 1,NROW
           NWBUFF(I+20)= 2* NWBUFF(I)
70      CONTINUE
```

Next, another field of data is computed by multiplying each of these values by 2. These new values are written to spaces 21 through 40 of NWBUFF.

**Create Data Set for Additional Data**

```
C      create a new dataset for the new field
      ODSN= DSN
      DSN = DSN+ 1
      C      copy 1st dataset label to 2nd dataset
           CALL WDDSCL (WDMSFL,ODSN,DSN,
O          RETCOD)
           IF (RETCOD.EQ.0) THEN
C             successful copy of label
           WRITE (OFL,*) ' label copied from dataset',ODSN,
1               ' to dataset',DSN
           ELSE
               WRITE (OFL,*) 'error on label copy, retcod:',RETCOD
           END IF
```

Next, a new table data set, data set 22, is defined by copying the data set label from data set 21 using subroutine WDDSCL.

**Put Template In New Data Set**

```
C      put new field on new dataset
      TCLU  = 12
      TGRP  = 3
      NROW  = 20
      C      first put template
           CALL WDTBTM (MESSFL,MFID,TCLU,TGRP,
I              WDMSFL,DSN,TABIND,NROW,
O              TFLDS,TNUM,TTYP,TLEN,TCOL,TSPA,
O              TABNAM,TGRPPT,AFLDS,ANUM,ATYP,
O              ALEN,ACOL,ASPA,ACLU,AGR,P,RETCOD)
```

After data set 22 has been successfully created, subroutine WDTBTM is used to put the table template for the new data field into the new data set. The template information is retrieved from the third table group (TGRP) in text and tables template data set (TCLU) number 12. Refer to Section 4.5 to review the specification for the table.

**Put Data In New Table Data Set**

```
C      now put data
      FROW= 1
      FSPLA= 1
      CALL WTBPPUT (WDMSFL,DSN,TABNAM,TABIND,DATFLG,
I          FROW,NROW,FSPLA,TSPA,NWBUFF,
O          RETCOD)
      IF (RETCOD.EQ.0) THEN
C        successful put of table data
           WRITE (OFL,*) NROW,' rows of data successfully put to',
1               ' dataset',DSN
           ELSE
               WRITE (OFL,*) ' Problem putting data to dataset',DSN,
1                   ', retcod:',RETCOD
           END IF
```

Finally, the values contained in NBUFF are put into data set 22 by using WTBPPUT. All the values are written to the data set beginning at the first space (FSPLA= 1) of the first row (FROW= 1).

**Specs**

<u>order</u>	<u>name</u>	<u>declaration</u>	<u>status</u>	<u>explanation</u>
1	WDMSFL	I*4	I	watershed data management file unit number
2	DSN	I*4	I	data-set number
3	DSTYPE	I*4	I	type of data set
4	NDN	I*4	I	number of down pointers
5	NUP	I*4	I	number of up pointers
6	NSA	I*4	I	number of search attributes
7	NSASP	I*4	I	amount of search attribute space
8	NDP	I*4	I	number of data pointers
9	PSA	I*4	O	pointer to search attribute space

**Check Data Set Existance and Type****WDCKDT****When to use** when information about a data set is required**What it does** Check data set for existance and type, returns:

0 - data set does not exist  
 or data-set type  
 1 - time series        6 - raster  
 2 - table              7 - space-time  
 3 - schematic          8 - attribute  
 4 - project            9 - message  
 5 - vector

**Prerequisites** an open WDM file**Example**

```

      WRITE(OFL,*) 'check existance of datasets'
      DO 10 DSN= 10,15
        DSNTYP= WDCKDT(WDMSFL,DSN)
        IF (DSNTYP.NE.0) THEN
          WRITE(OFL,*) 'dsn',DSN,' exists and is type',DSNTYP
        ELSE
          WRITE(OFL,*) 'dsn',DSN,' does not exist'
        END IF
10      CONTINUE
    
```

**Specs**

<u>order</u>	<u>name</u>	<u>declaration</u>	<u>status</u>	<u>explanation</u>
1	WDMSFL	I*4	I	Fortran unit\name of WDM file
2	DSN	I*4	I	data-set number to be checked

**Check Data Set Existance and Return First Record****WDDSCK****When to use** \*\*\*\*\***What it does** Check data set for existance and return record number of first record in data set (contains label)**Prerequisites** \*\*\*\*\***Example**

```

      DSN      = 30
      CALL WDDSCK (WDMSFL,DSN,
      O           DSNFRC,RETCOD)
      IF (RETCOD .EQ. 0) THEN
        WRITE(OFL,*) 'space time dataset',DSN,' already exists'
        RETCOD= 0
    
```

**Specs**

<u>order</u>	<u>name</u>	<u>declaration</u>	<u>status</u>	<u>explanation</u>
1	WDMFL	I*4	I	Fortran unit number of WDM file
2	DSN	I*4	I	data-set number to be checked
3	DREC	I*4	O	record number of first record in data set
4	RETCOD	I*4	O	return code 0 - data set exists -81 - data set does not exist -84 - data set number out of range

**Copy Old Data Set Label Into New Data Set Label****WDDSCL****When to use**

\*

**What It does**

copies an old data-set label into a new data-set label

**Prerequisites**

\*

**Example**

```

C      create a new dataset
      NDSN= 11
C      copy 1st dataset label to 2nd dataset
      CALL WDDSCL (WDMFL,DSN,NDSN,
O          RETCOD)
      IF (RETCOD.EQ.0) THEN
      successful copy of label
      WRITE(OFL,*) 'copied label from dsn',DSN,' to dsn',NDSN
      ELSE
      WRITE(OFL,*) 'ERROR creating dataset:',NDSN,' retcod:',RETCOD
      END IF

```

**Specs**

<u>order</u>	<u>name</u>	<u>declaration</u>	<u>status</u>	<u>explanation</u>
1	WDMFL	I*4	I	watershed data management file unit number
2	ODSN	I*4	I	old data-set number
3	NDSN	I*4	I	new data-set number
4	RETCOD	I*4	O	return code 0 - copy complete -61 - old data set doesn't exist -62 - new data set already exists

**Renumber a Data Set****WDDSRN****When to use**

\*

**What It does**

routine to renumber data sets with no user interaction

**Prerequisites**

\*

**Example**

```

      WRITE(OFL,*) 'renumber a dataset'
      DSN = 10
      NDSN= 12
      CALL WDDSRN (WDMFL,DSN,NDSN,
O          RETCOD)
      IF (RETCOD.EQ.0) THEN
      WRITE(OFL,*) 'DSN ',DSN,' renumbered to DSN',NDSN
      ELSE
      WRITE(OFL,*) 'ERROR renumbering DSN',DSN,' retcod:',RETCOD
      END IF

```

**Specs**

<u>order</u>	<u>name</u>	<u>declaration</u>	<u>status</u>	<u>explanation</u>
1	WDMSFL	I*4	I	Fortran unit number for WDM file
2	DSN	I*4	I	old data-set number
3	NDSN	I*4	I	new data-set number
4	RETCOD	I*4	O	return code 0 - renumber successfully completed -72 - old data set does not exist -73 - new data set already exists

**Delete a Data Set****WDDSDL****When to use**

\*

**What It does**

routine to delete a data set from the WDMSFL with no user interaction

**Prerequisites**

\*

**Example**

```

      WRITE(OFL,*) 'delete a dataset'
      DSN= 11
      CALL WDDSDL(WDMSFL,DSN,
      O      RETCOD)
      IF (RETCOD.EQ.0) THEN
          WRITE(OFL,*) 'DSN ',DSN,' deleted'
      ELSE
          WRITE(OFL,*) 'ERROR deleting DSN',DSN,' retcod:',RETCOD
      END IF
    
```

**Specs**

<u>order</u>	<u>name</u>	<u>declaration</u>	<u>status</u>	<u>explanation</u>
1	WDMSFL	I*4	I	Fortran unit number of WDM file
2	DSN	I*4	I	dataset number to be deleted
3	RETCOD	I*4	O	return code 0 - data set successfully deleted -81 - data set does not exist

## ATTRIBUTE MANIPULATION

\*\*\*\*describe this functional category\*\*\*\*

### Add Character Search Attribute

**WDBSAC**

**When to use**

**What it does** adds (or modifies) character search attribute on given dsn

**Prerequisites**

**Example**

```
C      now add attributes
C      timeseries type
      SAIND = 1
      SALEN = 4
      SABUFF= 'FLOW'
      CALL WDBSAC (WDMSFL,DSN,MESSFL,SAIND,SALEN,SABUF1,
O          RETCOD)
      WRITE(OFL,*) 'TSTYPE added, retcod:',RETCOD
```

**Specs**

order	name	declaration		
		type size	status	explanation
1	WDMSFL	I*4	I	watershed data management file unit number
2	DSN	I*4	I	data-set number being modified
3	MESSFL	I*4	I	message file unit number
4	SAIND	I*4	I	index number of attribute or highest attribute number if printing
5	SALEN	I*4	I	length of attribute
6	SAVAL	C*1 (V)	I	value of attribute
7	RETCOD	I*4	O	return code indicating if add or mod was successful 0 - successful -81 - data set does not exist -101 - incorrect character value for attribute -103 - no room on label for attribute -104 - data present, can't update attribute -105 - attribute not allowed for this type data set

### Add Integer Search Attribute

**WDBSAI**

**When to use**

**What it does** adds (or modifies) integer search attribute on given dsn

**Prerequisites**

**Example**

```
C      tcode
      SAIND= 17
      SALEN= 1
      SAVAL(1)= 4
      CALL WDBSAI (WDMSFL,DSN,MESSFL,SAIND,SALEN,SAVAL,
O          RETCOD)
      WRITE(OFL,*) 'TCODE added, retcod:',RETCOD
```

**Specs**

<u>order</u>	<u>name</u>	<u>declaration</u>	<u>status</u>	<u>explanation</u>
1	WDMFL	I*4	I	watershed data management file unit number
2	DSN	I*4	I	data-set number being modified
3	MESSFL	I*4	I	message file unit number
4	SAIND	I*4	I	index number of attribute or highest attribute number if printing
5	SALEN	I*4	I	length of attribute
6	SAVAL	I*4 (V)	I	value of attribute
7	RETCOD	I*4	O	return code indicating if add or mod was successful 0 - successful -81 - data set does not exist -103 - no room on label for attribute -104 - data present, can't update attribute -105 - attribute not allowed for this type data set -108 - incorrect integer value for attribute

**Add Real Search Attribute****WDBSAR****When to use**

\*

**What it does**

adds (or modifies) real search attribute on given dsn

**Prerequisites**

\*

**Example**

```
C      stddev
      SAIND= 15
      SALEN= 1
      SARVAL(1)= 1.25
      CALL WDBSAR (WDMFL,DSN,MESSFL,SAIND,SALEN,SARVAL,
O          RETCOD)
      WRITE(OFL,*) 'STDDEV added, retcod:',RETCOD
```

**Specs**

<u>order</u>	<u>name</u>	<u>declaration</u>	<u>status</u>	<u>explanation</u>
1	WDMFL	I*4	I	watershed data management file unit number
2	DSN	I*4	I	data-set number being modified
3	MESSFL	I*4	I	message file unit number
4	SAIND	I*4	I	index number of attribute or highest attribute number if printing
5	SALEN	I*4	I	length of attribute
6	SAVAL	R*4 (V)	I	value of attribute
7	RETCOD	I*4	O	flag indicating if modification or addition was successful 0 - successful -81 - data set does not exist -103 - no room on label for attribute -104 - data present, can't update attribute -105 - attribute not allowed for this type data set -109 - incorrect real value for attribute

**Get Values for Character Search Attribute****WDBSGC**

**When to use \***

**What it does** Get the values of character search attribute for a data set.

**Prerequisites \***

**Example**

```

C      look for an character attribute on a data set
      IF (SALEN .GT. 48) THEN
C      only have room for part of attribute
      SALEN= 48
END IF
CALL WDBSGC (WDMSFL,DSN,SAIND,SALEN,
O          SACVAL,SARET)
IF (SARET.EQ.0) THEN
      WRITE(OFL,2020) DSN,(SACVAL(I),I=1,SALEN)
END IF

```

**Specs**

<u>order</u>	<u>name</u>	<u>declaration</u>	<u>status</u>	<u>explanation</u>
1	WDMSFL	I*4	I	watershed data management file unit number
2	DSN	I*4	I	data-set number to add
3	SAIND	I*4	I	index number of attribute
4	SALEN	I*4	I	length of attribute
5	SAVAL	C*1 (V)	O	value of attribute
6	RETCOD	I*4	O	return code,
				0 - attribute value returned
				-81 - data set does not exist
				-107 - attribute not present on this data set

**Get Values for Integer Search Attribute**

**WDBSGI**

**When to use \***

**What It does** gets the values of integer search attribute for a dsn

**Prerequisites \***

**Example**

```

IF (SATYP.EQ.1) THEN
C      look for an integer attribute on a data set
      CALL WDBSGI (WDMSFL,DSN,SAIND,SALEN,
O          SAIVAL,SARET)
IF (SARET.EQ.0) THEN
      WRITE(OFL,' value on DSN',DSN,' is ',SAIVAL
END IF

```

**Specs**

<u>order</u>	<u>name</u>	<u>declaration</u>	<u>status</u>	<u>explanation</u>
1	WDMSFL	I*4	I	watershed data management file unit number
2	DSN	I*4	I	data-set number to add
3	SAIND	I*4	I	index number of attribute
4	SALEN	I*4	I	length of attribute
5	SAVAL	I*4 (V)	O	value of attribute
6	RETCOD	I*4	O	return code,
				0 - attribute value returned
				-81 - data set does not exist
				-107 - attribute not present on this data set

## Get Values for Real Search Attribute

WDBSGR

**When to use**

**What it does**

Get the values of real search attribute for a data set.

**Prerequisites**

**Example**

```
C      look for a real attribute on a data set
CALL WDBSGR (WDMFL,DSN,SAIND,SALEN,
              SARVAL,SARET)
O
IF (SARET.EQ.0) THEN
  WRITE(OFL,*) ' value on DSN',DSN,' is ',SARVAL
END IF
```

**Specs**

order	name	declaration		
		type size	status	explanation
1	WDMFL	I*4	I	watershed data management file unit number
2	DSN	I*4	I	data-set number to add
3	SAIND	I*4	I	index number of attribute
4	SALEN	I*4	I	length of attribute
5	SARVAL	R*4 (V)	O	value of attribute
6	RETCOD	I*4	O	return code 0 - attribute value returned -81 - data set does not exist -107 - attribute not present on this data set

## Delete a Search Attribute

WDBSAD

**When to use**

**What it does**

deletes search attribute on given dsn

**Prerequisites**

**Example**

```
C      try to delete it
CALL WDBSAD (WDMFL,DSN,SAIND,
              SAUPFG,SARQWD,SALEN,
              RETCOD)
I
O
IF (RETCOD.EQ.0) THEN
  WRITE(OFL,*) ' delete of attribute complete'
ELSE IF (RETCOD.EQ.-104) THEN
  WRITE(OFL,*) ' can not delete this attribute, data is present'
ELSE IF (RETCOD.EQ.-106) THEN
  WRITE(OFL,*) ' can not delete this attribute, it is required'
ELSE IF (RETCOD.EQ.-107) THEN
  WRITE(OFL,*) ' attribute not present on dataset'
ELSE
  WRITE(OFL,*) ' error on attribute delete, retcod:',RETCOD
END IF
```

**Specs**

<u>order</u>	<u>name</u>	<u>declaration</u>	<u>status</u>	<u>explanation</u>
<u>type</u>	<u>size</u>			
1	WDMFL	I*4	I	watershed data management file unit number
2	DSN	I*4	I	data-set number being modified
3	SAIND	I*4	I	index number of attribute
4	SAUPFG	I*4	I	update allowed if data present flag(0=yes)
5	SARQWD	I*4	I	search attribute required word
6	SALEN	I*4	I	length of attribute
7	RETCOD	I*4	O	flag indicating if deletion successful 0 - deletion successful -81 - data set does not exist -104 - data present, can't delete attribute -106 - attribute reqd. for this type data set, can't delete -107 - attribute not present on this data set

## TIME SERIES OPERATIONS

\*\*\*\*describe this functional category\*\*\*\*

### Get Timeseries Data

WDTGET

When to use \*

What It does gets timeseries information from the WDMSFL

Prerequisites \*

Example

```
C      read data from 1st dataset at monthly time steps
      TUNITS= 5
C      average values
      DTRAN = 0
      IDATES(1)= SYR
      IDATES(2)= SMO
      CALL WDTGET (WDMSFL,DSN,TSTP,IDATES,NMONS,
      I           DTRAN,QUALFG,TUNITS,
      O           RVAL,RETCOD)
      IF (RETCOD.NE.0) THEN
         WRITE(OFL,*) 'ERROR getting timeseries, retcod:',RETCOD
      ELSE
         WRITE(OFL,*) NMONS,' months of data successfully read'
```

Specs

order	name	declaration	status	explanation
1	WDMSFL	I*4	I	watershed data management file unit number
2	DSN	I*4	I	data-set number
3	DELT	I*4	I	time step for get
4	DATES	I*4 (6)	I	starting date
5	NVAL	I*4	I	number of values
6	DTRAN	I*4	I	transformation code 0 - ave,same 1 - sum,div 2 - max 3 - min
7	QUALFG	I*4	I	allowed quality code
8	TUNITS	I*4	I	time units for get
9	RVAL	R*4 (V)	O	array to place retrieved values in
10	RETCOD	I*4	O	return code 0 - everything O.K. -8 - invalid date -14 - date specified not within valid range for data set -20 - problem with one or more of following: GPFLG, DXX, NVAL, QUALVL, LTSTEP, LTUNIT -21 - date from WDM doesn't match expected date -81 - data set does not exist -82 - data set exists, but is wrong DSTYP -84 - data set number out of range

### Put Timeseries Data

WDTPUT

When to use \*

What It does Puts time series data into a WDM file. This routine traps the problem with overwriting existing data.

## Prerequisites

**Example**

```
C      put month of daily data on wdm file dataset
C      allow overwrite of existing data
C      DTOVWR= 1
C      assign a quality code to data
C      QUAFLG= 1
C      writing daily data
C      TUNITS= 4
C      one day timestep
C      TSTP = 1
I      CALL WDTPUT (WDMFL,DSN,TSTP,IDATES,N DAYS,
O                  DTOVWR,QUAFLG,TUNITS,RVAL,
I                  RETCOD)
IF (RETCOD.NE.0) THEN
    WRITE (OFL,*) 'ERROR putting timeseries, retcod:',RET COD
1
ELSE
    WRITE (OFL,*) 'wrote month of timeseries to WDM',
1      IDATES(1),IDATES(2)
END IF
```

## Specs

order	name	declaration	status	explanation
1	WDMFL	I*4	I	watershed data management file unit number
2	DSN	I*4	I	data-set number
3	DELT	I*4	I	time step for put
4	DATES	I*4 (6)	I	starting date
5	NVAL	I*4	I	number of values
6	DTOVWR	I*4	I	data overwrite flag, 0 - dont overwrite 1 - overwrite O.K.
7	QUAFLG	I*4	I	allowed quality code
8	TUNITS	I*4	I	time units for put
9	RVAL	R*4 (V)	I	array for writing out values
10	RETCOD	I*4	O	return code 0 - everything is O.K. -8 - invalid date -9 - data not present in current group -10 - no data in this group -11 - no non missing data, data has not started yet -14 - date specified not within valid range for data set -15 - VBTIME=1 and DELT,TUNITS do not agree with the data set -20 - problem with one or more of following: DTOVWR, NVAL, QUAFLG, TUNITS, DELT -21 - date from WDM doesn't match expected date -81 - data set does not exist -82 - data set exists, but is wrong DSTYP -84 - data set number out of range -85 - trying to write to a read-only data set

## Increment a Time Array

**TIMADD**

### When to use

### What it does

Add NVALS time steps to first date to compute second date. The first date is assumed to be valid.

## Prerequisites \*

### Example

```
C      calc ending time
      CALL TIMADD (STDAT,TUN,TST,NOV,
O          SEDAT)
```

### Specs

declaration					
<u>order</u>	<u>name</u>	<u>type</u>	<u>size</u>	<u>status</u>	<u>explanation</u>
1	DATE1	I*4	(6)	I	starting date
2	TCODE	I*4		I	time units
				1 - second	5 - month
				2 - minute	6 - year
				3 - hour	7 - century
				4 - day	
3	TSTEP	I*4		I	time step in TCODE units
4	NVALS	I*4		I	number of time steps to be added
5	DATE2	I*4	(6)	O	new date

## SPACE-TIME OPERATIONS

\*\*\*\*describe this functional category\*\*\*\*

### Add a Space Time Group

WSTAGP

When to use \*

What It does add a group to a space time data set, physically allocate space for

Prerequisites \*

Example

```
C      add the dummy group
      TUN= 4
      TST= 1
      NOV= 1
      CALL WSTAGP (WDMSFL,DSN,STDAT,TUN,TST,NOV,
      RETCOD)
      IF (RETCOD.EQ.0) THEN
         WRITE(OFL,*) 'dummy group added'
      ELSE
         WRITE(OFL,*) 'error',RETCOD,' when trying to add dummy group'
      END IF
```

Specs

order	name	declaration	status	explanation
1	WDMSFL	I*4	I	Fortran unit number of WDM file
2	DSN	I*4	I	Data-set number
3	NGPDAT	I*4 (6)	I	Starting date for group
4	NGPTUN	I*4	I	Time units for group
5	NGPTST	I*4	I	Time step for group
6	NGPNOV	I*4	I	Number of timesteps in group
7	RETCOD	I*4	O	Return code           0 - group added           -42 - overlap an existing group           -43 - can't add another space time group           -46 - bad space time group specification           parameter           -81 - data set does not exist           -82 - data set exists, but is wrong DSTYP           -84 - data set number out of range

## Summarize a Space Time Group

WSTGSU

**When to use**

\*

**What It does**

summarize a group in a space time data set

**Prerequisites**

\*

**Example**

```
C      summarize whats on dataset
      GRPIND= 0
90    CONTINUE
      GRPIND= GRPIND+ 1
      CALL WSTGSU (WDMSFL,DSN,GRPIND,
      0           STDAT,SEDAT,TUN,TST,NOV,FRAC,RETCOD)
      IF (RETCOD.EQ.0) THEN
          WRITE(OFL,*) 'summary of group',GRPIND
          WRITE(OFL,*) '      start date',STDAT
          WRITE(OFL,*) '      end date',SEDAT
          WRITE(OFL,*) '      other',TUN,TST,NOV,FRAC,RETCOD
      ELSE IF (RETCOD.NE.-49) THEN
          WRITE(OFL,*) 'error summarizing grp',GRPIND,' retcod:',RETCOD
      ELSE
          WRITE(OFL,*) 'completed summary of groups on dsn',DSN
      END IF
      IF (RETCOD.EQ.0) GO TO 90
```

**Specs**

declaration					
order	name	type	size	status	explanation
1	WDMSFL	I*4		I	Fortran unit number of WDM file
2	DSN	I*4		I	Data set number
3	GRPIND	I*4		I	Index of group to summarize
4	GSDAT	I*4	(6)	O	Start date of group
5	GEDAT	I*4	(6)	O	End date of group
6	GTUN	I*4		O	Group time units
7	GIST	I*4		O	Group time steps
8	GNOV	I*4		O	Number of values in group
9	GFRAC	R*4		O	Fraction of group containing data
10	RETCOD	I*4		O	Return code 0 - group summarized -49 - group doesn't exist -81 - data set does not exist -82 - data set exists, but is wrong DSTYP -84 - data set number out of range

## Get Real Space Time Data

WSTGTR

When to use \*

What it does get real space time data

Prerequisites \*

Example

```

I= 0
70    CONTINUE
      I= I+ 1
C      get a block(or part) of real data from a group
      CALL WSTGTR(WDMSFL,DSN,STDAT,NDIM,NUMN,BASN,SKPN,NOV,
O          RBUFF(I),RETCOD)
      IF (RETCOD.NE.0) THEN
        WRITE (OFL,*) 'error reading data:',RETCOD
      ELSE
C        next time step, calc ending time
        CALL TIMADD (STDAT,TUN,TST,I1,
                     SEDAT)
C        make ending time next starting time
        LEN= 6
        CALL COPYI (LEN,SEDAT,
O                      STDAT)
      END IF
      IF (I.LT.J .AND. RETCOD.EQ.0) GO TO 70

```

Specs

order	name	declaration	status	explanation
1	WDMSFL	I*4	I	Fortran unit number of WDM file
2	DSN	I*4	I	Data-set number
3	STDAT	I*4 (6)	I	Date of data to get
4	NDIM	I*4	I	Number of dimensions specified
5	NUMN	I*4 (V)	I	Number of values to get in each dimension
6	BASN	I*4 (V)	I	Base value in each dimension
7	SKPN	I*4 (V)	I	Skip value in each dimension
8	NVAL	I*4	I	Total number of values to get
9	RBUFF	R*4 (V)	O	Buffer to put values in
10	RETCOD	I*4	O	Return code <ul style="list-style-type: none"> <li>0 - data retrieved</li> <li>-36 - missing needed following data for a get</li> <li>-37 - no data present</li> <li>-38 - missing part of time required</li> <li>-39 - missing data group</li> <li>-40 - no data available</li> <li>-41 - no data to read</li> <li>-42 - overlap an existing group</li> <li>-44 - trying to get/put more data than in block</li> <li>-45 - types don't match</li> <li>-81 - data set does not exist</li> <li>-82 - data set exists, but is wrong DSTYP</li> <li>-84 - data set number out of range</li> </ul>

**Put Real Space Time Data****WSTPTR****When to use****What it does** put real space time data**Prerequisites**

**Example** C      put the data

```

CALL WSTPTR(WDMSFL, DSN, STDAT, NDIM, NUMN, BASN, SKPN,
             NVAL, RBUFF,
             RETCOD)
IF (RETCOD.EQ.0) THEN
    WRITE(OFL,2020) NVAL, STDAT
ELSE
    WRITE(OFL,*) 'error', RETCOD, ' putting space time data'
    WRITE(OFL,*) WDMSFL, DSN, STDAT
    WRITE(OFL,*) NDIM, NOV
    WRITE(OFL,*) NUMN, BASN, SKPN
END IF

```

**Specs**

<u>order</u>	<u>name</u>	<u>declaration</u>		<u>status</u>	<u>explanation</u>
1	WDMSFL	I*	4	I	Fortran unit number of WDM file
2	DSN	I*	4	I	Data-set number
3	STDAT	I*	4 (6)	I	Date of data to get
4	NDIM	I*	4	I	Number of dimensions specified
5	NUMN	I*	4 (V)	I	Number of values to get in each dimension
6	BASN	I*	4 (V)	I	Base value in each dimension
7	SKPN	I*	4 (V)	I	Skip value in each dimension
8	NVAL	I*	4	I	Total number of values to get
9	RBUFF	R*	4 (V)	I	Buffer to write values from
10	RETCOD	I*	4	O	Return code 0 - data written -36 - missing needed following data for a get -37 - no data present -38 - missing part of time required -39 - missing data group -40 - no data available -41 - no data to read -42 - overlap an existing group -44 - trying to get/put more data than in block -45 - types don't match -81 - data set does not exist -82 - data set exists, but is wrong DSTYP -84 - data set number out of range

# VECTOR OPERATIONS

\*\*\*\*describe this functional category\*\*\*\*

## Get Vector Data

WDLGET

When to use \*

What it does retrieve DLG header or coordinate pairs from WDM file

Prerequisites \*

Example

```
C      header first
      ID= 1
      CALL WDLGET (MESSFL,DSN,ATTYP(M),ATT1TM(M),ATT2TM(M),MAXBUF,
M           ID,
O           NPTS,DLGBUF,RETCOD)
IF (RETCOD.EQ.0) THEN
      WRITE(OFL,*) 'dlg type:',ATTYP(M),' header info:',NPTS
      WRITE(OFL,*) '          data:',DLGBUF(1),DLGBUF(2),DLGBUF(3)
ELSE IF (RETCOD.NE.2) THEN
      WRITE(OFL,*) 'ERROR in dlg get, retcod:',RETCOD
END IF
```

Specs

order	name	declaration			
			type	size	status explanation
1	WDMSFL	I*4	I		Fortran unit number for WDM file
2	DSN	I*4	I		data-set number on WDM file
3	ITYPE	I*4	I		type of DLG info (1- LINE, 2- AREA, 3- NODE)
4	ATT1	I*4	I		major attribute value
5	ATT2	I*4	I		minor attribute value
6	LEN	I*4	I		max length of information being retrieved (4 byte words)
7	ID	I*4	M		id of information retrieved (0-either, 1-header, 2-data)
8	OLEN	I*4	O		actual length of output buffer
9	DLGBUF	R*4 (V)	O		buffer of information being retrieved
10	RETCOD	I*4	O		return code 2 - no more data in this group 1 - more of current id remaining 0 - DLG data retrieved successfully -81 - data set does not exist -82 - data set exists, but is wrong DSTYP -50 - major and minor attributes not found on this data set

## Summarize Vector Data

WDLLSU

When to use \*

What it does summarize label information for DLG data set

Prerequisites \*

Example

```
DSN= 90
C   summarize dlg's in dataset
CALL WDLLSU(MESSFL,DSN,MAXATTR,
O           ATTUSE,ATTTYPE,ATT1TM,ATT2TM,RETCOD)
IF (RETCOD.EQ.0) THEN
  WRITE (OFL,*) ' attributes found in dlg data set',DSN
  DO 20 M= 1,ATTUSE
    WRITE (OFL,*) ''
    WRITE (OFL,*) 'attribute ',M,ATTTYPE(M),ATT1TM(M),ATT2TM(M)
```

Specs

declaration					
order	name	type	size	status	explanation
1	MESSFL	I*4		I	Fortran unit number for WDM file
2	DSN	I*4		I	data-set number on WDM file
3	ILEN	I*4		I	maximum size of information buffers
4	OLEN	I*4		O	actual amount of information returned (<= ILEN)
5	TYPE	I*4 (V)		O	buffer of information types on DSN
6	ATT1	I*4 (V)		O	buffer of major attributes on DSN
7	ATT2	I*4 (V)		O	buffer of minor attributes on DSN
8	RETCOD	I*4		O	return code 1 - more groups on DSN 0 - label summary returned successfully -81 - data set does not exist -82 - data set exists, but is wrong DSTYP

## TEXT AND TABLE TEMPLATES OPERATIONS

\*\*\*\*describe this functional category\*\*\*\*

### Get First Block Control Word in a Group

WMSFBC

**When to use**

\*

**What It does**

Get first block control word and its position for group GNUM in a message data set. A STOP is encountered when group GNUM does not exist.

**Prerequisites**

\*

**Example**

```
C      cluster/group containing table template
      SCLU= 12
      SGRP= 1
C      get address where text or table template info starts
      CALL WMSFBC (MESSFL,SCLU,SGRP,
      O      DREC,DPOS,QWORD)
C
      IF (QWORD.LE.0) THEN
         WRITE(OFL,*)'ERROR, no text or table tmplte for:',MESSFL,
      1      SCLU,SGRP.
```

**Specs**

<u>order</u>	<u>name</u>	<u>declaration</u>	<u>status</u>	<u>explanation</u>
1	WDMSFL	I*4	I	Fortran unit number for WDM file
2	DSN	I*4	I	data-set number
3	GNUM	I*4	I	group number
4	DREC	I*4	O	record number of block control word on WDM file
5	DPOS	I*4	O	position of block control word on DREC
6	BCWORD	I*4	O	block control word

### Split a Message Block Control Word

WMSBCS

**When to use**

\*

**What It does**

Split up a block control word for a message type data set into its components.

**Prerequisites**

\*

**Example**

```
C      get text/table parameters.
      CALL WMSBCS (QWORD,
      O      CLASS, ID, ORDER, TXTLEN)
      IF (ID.GT.0) THEN
```

## Get one Record of Text From WDM File

WMSGTE

## When to use

☆

## **What It does**

Get one record of text off WDM file.

## Prerequisites

☆

### Example

2

```
C           header, table name
GLEN= 0
MLEN= 0
20      CONTINUE
        CALL WMSGTE (MESSFL,TXTLEN,I80,
M             DREC,DPOS,GLEN,MLEN,
O             OLEN,TXBUF1,MORE)
        WRITE(OFL,*) '    text table block:',CLASS,ID,ORDER,
1             TXTLEN,GLEN,MLEN,OLEN,MORE
        WRITE (OFL,2000) (TXBUF1(I),I=1,OLEN)
IF (MORE.NE.0) GO TO 20
```

## Specs

**declaration**

<u>order</u>	<u>name</u>	<u>type</u>	<u>size</u>	<u>status</u>	<u>explanation</u>
1	WDMSFL	I*4		I	Fortran unit number for WDM file
2	TLEN	I*4		I	total length of text (may be more than one record)
3	LLEN	I*4		I	maximum size of record to get
4	DREC	I*4		M	record number of data on WDM file
5	DPOS	I*4		M	position of data on data record
6	GLEN	I*4		M	counter to keep track of when to read off WDM file should be initialized to 0 for first call
7	MLEN	I*4		M	number of characters retrieved so far (must be <= TLEN) should be initialized to 0 for first call
8	OLEN	I*4		O	actual size of record retrieved
9	OBUFF	C*1 (V)		O	array of size LLEN containing OLEN characters retrieved
10	CONT	I*4		O	indicator flag for text 0 - no more text available 1 - more text available

## Move to Next Data Position In Text Data Set

WDNXDV

**When to use**

\*

**What It does**

Move to the next data position and return the integer equivalent of the data value.

**Prerequisites**

\*

**Example**

C            get next table block control word  
               CALL WDNXDV (MESSFL,  
                   M            DREC, DPOS,  
                   O            QWORD)

**Specs**

declaration					
<u>order</u>	<u>name</u>	<u>type</u>	<u>size</u>	<u>status</u>	<u>explanation</u>
1	WDMFL	I*4		I	Fortran unit number for WDM file
2	DREC	I*4		M	record number of data on WDM file
3	DPOS	I*4		M	position of data on data record (both DREC and DIND)
4	DVAL	I*4		O	data value on WDM file

## Skip Within Text Data Set

WMSSKB

**When to use**

\*

**What It does**

Position DREC and DPOS at the end of the current data block. DREC and DPOS are assumed to be input as the start of the block.

**Prerequisites**

\*

**Example**

C            skip block  
               WRITE(OFL,\*) 'numeric table block:',CLASS, ID, ORDER, TXTLEN  
               CALL WMSSKB (MESSFL, TXTLEN,  
                   M            DREC, DPOS)

**Specs**

declaration					
<u>order</u>	<u>name</u>	<u>type</u>	<u>size</u>	<u>status</u>	<u>explanation</u>
1	WDMFL	I*4		I	Fortran unit number for WDM file
2	TLEN	I*4		I	total number of characters to skip
3	DREC	I*4		M	record number on WDM file
4	DPOS	I*4		M	position on record DREC

## TABLE OPERATIONS

\*\*\*\*describe this functional category\*\*\*\*

### Put Table Template on WDM File

**WDTBTM**

**When to use**

\*

**What It does**

put WDM table template on WDM file, return parameters about table

**Prerequisites**

\*

**Example**

```
C      put table template on wdm file
      TCLU = 12
      TGRP = 1
      TABIND= 1
      NROW = 50
      CALL WDTBTM (MESSFL, MFID, TCLU, TGRP, WDMSFL, DSN, TABIND, NROW,
      O           TFLDS, TNUM, TTYP, TLEN, TCOL, TSPA, TABNAM, TGRPPT,
      O           AFLDS, ANUM, ATYP, ALEN, ACOL, ASPA, ACLU, AGRP,
      O           RETCOD)
```

**Specs**

declaration					
order	name	type	size	status	explanation
1	MESSFL	I*4		I	Fortran unit number of WDM file containing table definition
2	MFID	C*1 (2)		I	Message file name id of MESSFL
3	TCLU	I*4		I	Message file cluster containing table template
4	TGRP	I*4		I	Group number containing table template
5	WDMSFL	I*4		I	Fortran unit number of WDM file to put table template in
6	DSN	I*4		I	Data-set number to put table template in
7	TABIND	I*4		I	Table identifier number of new table
8	NROW	I*4		I	Number of rows in new table
9	TFLDS	I*4		O	Number of fields in table
10	TNUM	I*4 (4)		O	Number of each variable type in table(I-1,R-2,C-3,D-4)
11	TTYP	C*1 (30)		O	Type of each field in table(I,R,C,D)
12	TLEN	I*4 (30)		O	Length of each field in table(characters)
13	TCOL	I*4 (30)		O	Starting column for each field
14	TSPA	I*4		O	Space required for each table row(words)
15	TBNAM	C*16		O	Name of table from message file
16	TGRPPT	I*4		O	Pointer to group within DSN
17	AFLDS	I*4		O	Number of fields in table extension
18	ANUM	I*4 (4)		O	Number of each variable type in table extension(see 10)
19	ATYP	C*1 (30)		O	Type of each field in table extenision
20	ALEN	I*4 (30)		O	Length of each field in table extenision
21	ACOL	I*4 (30)		O	Starting column for each associated field
22	ASPA	I*4		O	Space required for table extenision
23	ACLU	I*4		O	Associated table cluster number
24	AGRP	I*4		O	Associated table group number
25	RETCOD	I*4		O	Return code

## Get Table Data From WDM File

WTBGET

**When to use**

\*

**What it does**

get the main table data

**Prerequisites**

\*

**Example**

```
C      get two real data fields
FROW= 1
NROW= 50
FSPLA= 4
TSPA= 2
CALL WTBGET (WDMSFL,DSN,TABNAM,TABIND,DATFLG,
I      FROW,NROW,FSPLA,TSPA,
O      RBUFF,RETCOD)
IF (RETCOD.EQ.0) THEN
C      successful get of real fields
      WRITE (OFL,*) NROW,' rows of data for',TSPA,' fields',
I      ' successfully read from dataset',DSN
ELSE
      WRITE (OFL,*) ' ERROR getting data from dataset',DSN,
I      ', retcod:',RETCOD
END IF
```

**Specs**

		declaration			
order	name	type	size	status	explanation
1	WDMSFL	I*4		I	Watershed data management file
unit number					
2	DSN	I*4		I	table data-set number
3	TABNAM	C*16		I	table name
4	TABIND	I*4		I	table identifier number
5	DATFLG	I*4		I	data type
					1 - main table
					2 - extension
6	FROW	I*4		I	first row of data to read from
7	NROW	I*4		I	number of rows of data to read
8	FSPLA	I*4		I	first data space to read from
9	NSPA	I*4		I	space for data in each row
10	TBRBUF	R*4 (V,V)		O	buffer for data to get from WDM
file					
11	RETCOD	I*4		O	return code

## Convert Table Data to Internal Format

WTBCOD

**When to use**

\*

**What it does**

convert data from full screen buffer into WDM internal format

**Prerequisites**

\*

**Example**

```
C      convert character buffer into real data buffer
CALL WTBCOD (TFLDS,NROW,TSPA,TLEN,TTYP,TCOL,
I      CBUFF,MXTBTL,
O      RBUFF,RETCOD)
```



## Get Table Label Information From a Table Data Set

WDTBSU

**When to use**

\*

**What it does**

get WDM table label info from WDM file table data set

**Prerequisites**

\*

**Example**

```

DSN      = 20
TABBAS= 1
WRITE(OFL, *)
WRITE(OFL,*) 'summary of tables stored in dataset',DSN
CALL WDTBSU (WDMSFL,DSN,MAXTAB,TABBAS,
O           TABCNT,TABNMX,TABID,TABDIM,PDATVL,RETCOD)
IF (RETCOD.EQ.0) THEN
  WRITE(OFL,*) TABCNT,' tables found'

```

**Specs**

<u>order</u>	<u>name</u>	<u>declaration</u>	<u>status</u>	<u>explanation</u>
1	WDMSFL	I*4	I	Fortran unit number of WDM file
2	DSN	I*4	I	WDM table data-set number
3	TABMAX	I*4	I	Maximum number of tables to get info about
4	TABBAS	I*4	I	Table base pointer, first group to get info about
5	TABCNT	I*4	O	Total number of tables found
6	TABNM	C*1 (16,V)	O	Name of table
7	TABID	I*4 (V)	O	Id of table
8	TABDIM	I*4 (V)	O	Dimensions of table
9	PDATVL	I*4 (V)	O	Pointer to table data values
10	RETCOD	I*4	O	Return code, (+) if more tables than TABMAX

## Determines Pointer to Specified Table

WDTBFX

**When to use**

\*

**What it does**

determines pointer to the specified table, also returns its message file cluster and group number

**Prerequisites**

\*

**Example**

```

WRITE(OFL, *)
WRITE(OFL,*) 'more info about first table'
TABNAM= 'TEST1.TAB'
TABIND= 1
CALL WDTBFX (WDMSFL,DSN,TABIND,TABNAM,
O           TABCNT,LREC,TGRPPT,MFID,TCLU,TGRP,NROW,RETCOD)

```

**Specs**

<u>order</u>	<u>name</u>	<u>declaration</u>	<u>status</u>	<u>explanation</u>
1	WDMFL	I*4	I	Fortran unit number of WDM file
2	DSN	I*4	I	table data-set number
3	TABIND	I*4	I	table identifier number
4	TABNAM	C*16	I	name of table
5	TBCNT	I*4	O	total number of tables in data set
6	LREC	I*4	O	label record number
7	TGRPPT	I*4	O	table group pointer
8	MFID	C*1 (2)	O	message file name id
9	TCLU	I*4	O	table message file cluster number
10	TGRP	I*4	O	table message file group number
11	NROW	I*4	O	number of rows in table
12	RETCOD	I*4	O	return code
				0 - table found, pointer and other information returned

**Split Table Dimension Variable****WTBDSP****When to use**

\*

**What It does**

Split up dimension variable for table type data set into its components.

**Prerequisites**

\*

**Example**

```
CALL WTBDSP(TABDIM(I),
             TABIND,NROW,NCOL,NEXT)
```

**Specs**

<u>order</u>	<u>name</u>	<u>declaration</u>	<u>status</u>	<u>explanation</u>
1	TABDIM	I*4	I	table dimension variable
2	TABIND	I*4	O	table index number
3	NROW	I*4	O	number of rows in table
4	NCOL	I*4	O	space for each column(words)
5	NEXT	I*4	O	amount of table extension space (words)

**Split Table Identifier Variable****WTBISP****When to use**

\*

**What It does**

Split up an identifier for a table type data set into its components.

**Prerequisites**

\*

**Example**

```
CALL WTBISP(TABID(I),
              MFID,TCLU,TGRP)
```

**Specs**

<u>order</u>	<u>name</u>	<u>declaration</u>	<u>status</u>	<u>explanation</u>
1	TABID	I*4	I	table identifier
2	MSFLID	C*1 (2)	O	message file name id
3	MCLU	I*4	O	message file cluster for table
4	MGRP	I*4	O	message file group number for table

## Delete Table From a Table Data Set

**WDTBDL**

**When to use**

\*

**What It does**

delete WDM table from WDM file

**Prerequisites**

\*

**Example**

```
      WRITE(OFL,*) 'delete first table'
      CALL WDTBDL (WDMSFL,DSN,TABNAM,TABIND,
      O           RETCOD)
      IF (RETCOD.EQ.0) THEN
          WRITE(OFL,*) 'table deleted'
      ELSE
          WRITE(OFL,*) 'error on table delete, retcod:',RETCOD
      END IF
```

**Specs**

order	declaration				explanation
	name	type	size	status	
1	WDMSFL	I*4		I	Fortran unit number of WDM file
2	DSN	I*4		I	Table data-set number
3	TABNAM	C*16		I	Table data-set name
4	TABIND	I*4		I	Table identifier
5	RETCOD	I*4		O	Return code

## ATTRIBUTE CHARACTERISTICS OPERATIONS

\*\*\*\*describe this functional category\*\*\*\*

### Find Search Attributes Which Match Name

WDSAFI

When to use \*

What It does

Given the attribute name SAFNAM, starting at attribute index SAIND, find the first attribute name which matches SAFNAM. Return the index of the next attribute which also matches SAFNAM, if one exists.

Prerequisites \*

Example

```
C      look for all attributes starting with 'ST'  
SAINAM(1)= 'S'  
SAINAM(2)= 'T'  
WRITE(OFL,2010) SAINAM  
SAIND= 1  
C      loop to look for more attributes  
10    CONTINUE  
      CALL WDSAFI (MESSFL,SAINAM,  
M          SAIND,  
O          SAONAM, RETCOD)  
      IF (RETCOD.EQ.0 .OR. RETCOD.EQ.-112) THEN  
C          match found  
          WRITE(OFL,*)  
          WRITE(OFL,2000) SAONAM,SAIND
```

Specs

order	declaration				explanation
	name	type	size	status	
1	MESSFL	I*4		I	Fortran unit number for message file
2	SAFNAM	C*1 (6)		I	character array containing attribute name to search for
3	SAIND	I*4		M	index of next matching attribute, if one exists, otherwise, index of matching attribute
4	SAONAM	C*1 (6)		O	character array of first attribute name matching SAFNAM
5	RETCOD	I*4		O	return code -110 - attributes not found on message file -111 - attribute name not found (no match) -112 - more attributes exist which match SAFNAM

### Get General Information About Search Attribute

WDSAGY

When to use \*

What It does

gets general detail information about specified attribute

Prerequisites \*

Example

```
C      get its characteristics  
CALL WDSAGY (MESSFL,SAIND,  
O          SANAM,DPTR,SATYP,SALEN,SARQWD,SAUPFG)  
          WRITE(OFL,2005) DPTR,SATYP,SALEN,SARQWD,SAUPFG
```

**Specs**

<u>order</u>	<u>name</u>	<u>type</u>	<u>size</u>	<u>status</u>	<u>declaration</u>	<u>explanation</u>
1	MESSFL	I*4		I		message file unit number
2	SAIND	I*4		I		attribute index number
3	SANAM	C*1 (6)		O		name of search attribute
4	DPTR	I*4		O		pointer to other details of attribute
5	SATYP	I*4		O		type of attribute
6	SALEN	I*4		O		length of attribute
7	SARQND	I*4		O		word containing attribute requirements by dsn type
8	SAUPFG	I*4		O		attribute update flag

**Get Search Attribute Description****WADGDS****When to use**

\*

**What it does**

get the description for an attribute off the message file

**Prerequisites**

\*

**Example**

```

C      get the description
CALL WADGDS (MESSFL,DPTR,
              SACVAL)
WRITE(OFL,2030) SACVAL

```

**Specs**

<u>order</u>	<u>name</u>	<u>type</u>	<u>size</u>	<u>status</u>	<u>declaration</u>
1	MESSFL	I*4		I	Fortran unit number for message file
2	DPTR	I*4		I	pointer to start of details for this attribute
3	SADESC	C*1 (47)		O	description for attribute

**Get Search Attribute Help Location****WADGHL****When to use**

\*

**What it does**get the length and starting record/pos of the help info for an attribute  
off the message file**Prerequisites**

\*

**Example**

```

CALL WADGHL (MESSFL,DPTR,
              TLEN,DREC,DPOS)
IF (TLEN.GT.0) THEN
  WRITE(OFL,*) ' help is ',TLEN,' long at location',DREC,DPOS
ELSE
  WRITE(OFL,*) ' no help available'
END IF

```

**Specs**

<u>order</u>	<u>name</u>	<u>type</u>	<u>size</u>	<u>status</u>	<u>declaration</u>
1	MESSFL	I*4		I	Fortran unit number for message file
2	DPTR	I*4		I	pointer to start of details for this attribute
3	TLEN	I*4		O	length of help info (0 - no help)
4	DREC	I*4		O	record on which help info starts
5	DPOS	I*4		O	position on record where help starts

## Get Search Attribute Default Value

WADGDF

**When to use**

\*

**What it does**

get the default value for an attribute off the message file

**Prerequisites**

\*

**Example**

```
C      get the default value
CALL WADGDF (MESSFL,DPTR,SATYP,
              O          ATRDEF)
IF (SATYP.EQ.1) THEN
C      integer
      WRITE(OFL,*) ' default is',ATIDEF
ELSE IF (SATYP.EQ.2) THEN
C      real
      WRITE(OFL,*) ' default is',ATRDEF
END IF
```

**Specs**

		declaration			
order	name	type	size	status	explanation
1	MESSFL	I*4		I	Fortran unit number for message file
2	DPTR	I*4		I	pointer to start of details for this attribute
3	ATTYP	I*4		I	attribute type
4	ATDEF	R*4		O	default value for attribute

## REFERENCES

- Brown, L.C. and T.O. Barnwell. 1987. The Enhanced Water Quality Models QUAL2E and QUAL2E-UNCAS: Documentation and User Manual. U.S. Environmental Protection Agency, Athens, GA. EPA/600/3-87/007.
- Imhoff, J.C., R.F. Carsel, J.L. Kittle, Jr. and P.R. Hummel. 1990. Data Base Analyzer and Parameter Estimator (DBAPE) Interactive Computer Program User's Manual. U.S. Environmental Protection Agency, Athens, GA. EPA/600/3-89/083.
- Johanson, R.C., J.C. Imhoff, J.L. Kittle, Jr. and A.S. Donigian, Jr. 1984. Hydrological Simulation Program - Fortran (HSPF): User's Manual for Release 8.0. U.S. Environmental Protection Agency, Athens, GA. EPA/600/3-84/066.
- Kittle, J.L., Jr., P.R. Hummel and J.C. Imhoff. 1989. ANNIE-IDE, A System for Developing Interactive User Interfaces for Environmental Models (Programmers Guide). U.S. Environmental Protection Agency, Athens, GA. EPA/600/3-89/034.
- Kittle, J.L., Jr., K.M. Flynn, A.M. Lumb and P.R. Hummel. 1990. Programmers Documentation for ANNIE; a Fortran Library for Interactive Hydrologic Analyses and Data Management. U.S. Geological Survey, Reston VA. Draft Open File Report.
- Lumb, A.M., J.L. Kittle, Jr. and K.M. Flynn. 1990. Users Manual for ANNIE, a Computer Program for Interactive Hydrologic Analyses and Data Management. U.S. Geological Survey, Reston, VA. Water Resources Investigations Report 89-4080.
- Sharpe-Hansen, S., C. Travers, P. Hummel and T. Allison. 1990. A Subtitle D Landfill Application Manual for the Multimedia Exposure Assessment Model (MULTIMED). U.S. Environmental Protection Agency, Athens, GA. EPA/600/.
- Coding Conventions and Sysdoc - Flynn 1989
- NCIC- 1985 DLG formats

## GLOSSARY

**Application programs** - Subroutines or programs that use the WDM file by calling the set of subroutines that get, put, and modify data on WDM data sets.

**Application programmer** - Person that uses the utility subroutines in their own program to get, put, and modify data on the WDM file.

**Attribute** - A name, number, or set of characters that describe data in a data set.

**Block** - String of closely related data within a group of closely related data in a record that has been located with a pointer.

**Block control word** - An integer number (32 bits) with a bit

pattern to represent five variables: number of values, time step for the group, time-step units code, compression code, and quality of data code.

**Cluster** - Text or table data set

**Data set** - Collection of related records in a file that contain data on a time series, a segment of drainage basin, a watershed schematic, a project definition, the file definition, or the data set pointers. The data set includes attributes and pointers as well as the basic data.

**Data set buffer** -

**Data set number** - Unique index number from 1 to 200,000 assigned to each data set.

**Data set pointer** -

**Date array** -

**Directory record** - Contains pointers to records in the file based on the assigned data set number.

**Disk** - Physical device for storing a large volume of data. As used herein, a hard disk with a capacity of at least 10 megabytes.

**File definition record** - First record of a WDM file indicating characteristics of the file such as number of records, pointers to directory records, and file identification name and date.

**File (direct access file)** - Space on a computer disk where the user stores a collection of related data.

**Group** - String of closely related data in a record that has been located with a pointer. Usually contains a string of blocks.

**Group pointer** -

**Label** - First part of a data set that includes attributes of the data set and pointers to strings of data within the data set.

**Pointer** - Integer number of a record in a direct access file or the physical position in words within a record.

**Position pointer** - Integer number on a record that is another position within that record.

**Record** - Physical space in a file on a disk. Its size is usually given in number of bytes or characters. A collection of records makes a file.

**Record buffer** -

**Record pointer** - Integer number on a record that is the number of another record.

**Schematic data** - Data that references segment characteristic data sets in an upstream-downstream link.

**System WDM file** -

**Time-series data** - Data for a specific location that varies with time such as streamflow, rainfall, and temperature.

**User WDM file** -

**Watershed Data Management System (WDMS)** - Combination of a direct access file of hydrologic data in the specific format described herein and the software ANNIE-ES to process the data.

# APPENDIX A. WDM FILE FORMATS

\*\*\*\*\* write an introduction here describing how this appendix will work

## APPENDIX A.1 File Definition Record

Variable	Position	Type	Default	Explanation
PPRBKR	1	I4	-998	Primary backwater record pointer (always -998 in this record) (previous version was -999)
PPRFWR	2	I4	1	Primary forward record pointer (always 0 in this record)
PSCBKR	3	I4	0	Secondary backward record pointer (always 0 in this record)
PSCFWR	4	I4	0	Secondary forward record pointer (always 0 in this record)
-	5-28	-	-	< not in use >
LSTREC	29	I4	20	File size in records
-	30	-	-	< not in use >
FREREC	31	I4	3	Next free record in file
DSCNT(1)	32	I4	0	Count of time-series data sets
DSFST(1)	33	I4	0	Data set number of most recently created time-series data set
DSCNT(2)	34	I4	0	Count of table data sets
DSFST(2)	35	I4	0	Data set number of most recently created table data set
DSCNT(3)	36	I4	0	Count of schematic data sets
DSFST(3)	37	I4	0	Data set number of most recently created schematic data set
DSCNT(4)	38	I4	0	Count of project description data sets
DSFST(4)	39	I4	0	Data set number of most recently created project description data set
DSCNT(5)	40	I4	0	Count of vector data sets
DSFST(5)	41	I4	0	Data set number of most recently created vector data set
DSCNT(6)	42	I4	0	Count of raster data sets
DSFST(6)	43	I4	0	Data set number of most recently created raster data set
DSCNT(7)	44	I4	0	Count of space-time data sets
DSFST(7)	45	I4	0	Data set number of most recently created space-time data set
DSCNT(8)	46	I4	0	Count of attribute data sets

## **APPENDIX A.1 File Definition Record--continued**

---

<u>Variable</u>	<u>Position</u>	<u>Type</u>	<u>Default</u>	<u>Explanation</u>
DSFST(8)	47	I4	0	Data set number of most recently created attribute properties data set
DSCNT(9)	48	I4	0	Count of message data sets
DSFST(9)	49	I4	0	Data set number of most recently created message data set
-	50-112	-	-	< not in use >
DIRPNT(64)	113-176	64I4	0	Record pointers to directory data sets
-	177-512	-	-	< not in use >

## **APPENDIX A.2 Directory Record**

---

<u>Variable</u>	<u>Position</u>	<u>Type</u>	<u>Default</u>	<u>Explanation</u>
PPRBKR	1	I4	0	Primary backwater record pointer (always 0 in this record)
PPRFWR	2	I4	0	Primary forward record pointer (always 0 in this record)
PSCBKR	3	I4	0	Secondary backwater record pointer (always 0 in this record)
PSCFWR	4	I4	0	Secondary forward record pointer (always 0 in this record)
RECLOC(500)	5-504	500I4	0	Record location for each data set number, positioned by data set number. Data set number 3 would have the record location in the seventh position of the record.
-	505-512	-	-	< not in use >

## **APPENDIX A.3 Time-Series Data Set**

---

<u>Variable</u>	<u>Position</u>	<u>Type</u>	<u>Default</u>	<u>Explanation</u>
PPRBKR	1	I4	0	Primary backward record pointer to previous time-series data set (always zero for first data set)
PPRFWR	2	I4	0	Primary forward record pointer to next time-series data set (always zero for last data set)
PSCBKR	3	I4	0	Secondary backward record pointer (always zero for first record data set)
PSCFWR	4	I4	0	Secondary forward record pointer to next record within data set (always zero for last record in data set)
DSN	5	I4	none	Unique data set number
DSTYPE	6	I4	1	Data set type (always set to 1 for time-series)
-	7	-	-	< not in use >
PDP	8	I4	0	Position of pointers to other data sets

## APPENDIX A.3 Time-Series Data Set--continued.

Variable	Position	Type	Default	Explanation
PUP	9	I4	0	Position of pointers from other data sets
PSA	10	I4	0	Position pointer to search attributes
PDAT	11	I4	0	Position pointer to group pointers
PDATV	12	I4	0	Pointer to start of groups
If PDP > 0 (pointers to other data sets)				
DNCNT	PDP	I4	0	Number of 'to' pointers (must be <(PUP-PDP-1))
DNVAL(n)	PDP+n	I4	0	Data set number of data set referenced (n=1,DNCNT)
If PUP > 0 (pointers from other data sets)				
UPCNT	PUP	I4	0	Number of 'from' pointers (must be < (PSA-PUP-1))
UPVAL(n)	PUP+n	I4	0	Data set number of data set referencing this data set (n=1,UPCNT)
If PSA > 0 (search attributes exist)				
SACNT	PSA	I4	0	Current count of search attributes (must be < (PSASTR-PSA-1)/2)
PSASTR	PSA+1	I4	none	Position pointer to start of search attribute values
For each search attribute, the following pair occur				
SAIND(n)	PSA+(2*n)	I4	none	Index number of search attribute on the read-only MESSAGE.WDM file (n=1,SACNT)
PSAVAL(n)	above+1	I4	none	Position pointer from PSASTR to search attribute value (n=1,SACNT)
Search attributes				
SAVAL(n)	PSASTR+PSAVAL(n)	*	*	Search attribute value - type and default are shown in Appendix C by search attribute index number (n=1,SACNT)
DPCNT	PDAT	I4	none	Count of group pointers (must be < (PDATV-PDAT-1))
FREPOS	PDAT+1	I4	none	Pointer to first free data position with the following bit template  record (PREC) 22 bits 1-2097151 position within record (POFF) 9 bits 0-511
PDATVL(n)	PDAT+1+n	I4	none	Pointer to a group of time series data with the same bit template as FREPOS (n is number of groups offset from base date calculated using attributes TSBYR, TSBMO, TSBDY, TSBHR, and TGROUP)

## APPENDIX A.3 Time-Series Data Set--continued.

<u>Variable</u>	<u>Position</u>	<u>Type</u>	<u>Default</u>	<u>Explanation</u>
-----------------	-----------------	-------------	----------------	--------------------

For each group of data, there is date stamp followed by a series of a block control words followed by a block of data.

DATWRD	PDATVL	I4	none	Date stamp for the beginning of the group with the following bit template
				Year (DAT(1)) 17 bits 1-131071
				Month (DAT(2)) 4 bits 1-12
				Day (DAT(3)) 5 bits 1-31
				Hour (DAT(4)) 5 bits 1-24
BCW	PDATVL+m	I4	none	Block control ( $m > 0$ ) with the following bit template
				Number of values (NOV) 15 bits 0-32767
				Time step (TSTEP) 6 bits 0-63
				Units (TCODE) 3 bits 0-7
				Compression code (COMPDC) 2 bits 0-3
				Quality of data code (QUALCD) 5 bits 0-31
RVAL(n)	above+n	R4	none	Data values for the block ( $n=1, NOV$ )

## APPENDIX A.4 Table Data Set

<u>Variable</u>	<u>Position</u>	<u>Type</u>	<u>Default</u>	<u>Explanation</u>
PPRBKR	1	I4	0	Primary backward record pointer to previous tables data set (always zero for first data set)
PPRFWR	2	I4	0	Primary forward record pointer to next tables data set (always zero for last data set)
PSCBKR	3	I4	0	Secondary backward record pointer (always zero for first record data set)
PSCFWR	4	I4	0	Secondary forward record pointer to next record within data set (always zero for last record in data set)
DSN	5	I4	none	Unique data set number
DSTYPE	6	I4	2	Data set type (always set to 2 for tables)
-	7	-	-	< not in use >
PDP	8	I4	0	Position of pointers to other data sets
PUP	9	I4	0	Position of pointers from other data sets
PSA	10	I4	0	Position pointer to search attributes
PDAT	11	I4	0	Position pointer to group pointers
PDATV	12	I4	0	Pointer to start of groups
If PDP > 0 (pointers to other data sets)				
DNCNT	PDP	I4	0	Number of 'to' pointers (must be <(PUP-PDP-1))
DNVAL(n)	PDP+n	I4	0	Data set number of data set referenced ( $n=1, DNCNT$ )

## APPENDIX A.4 Table Data Set--continued.

Variable	Position	Type	Default	Explanation
If PUP > 0 (pointers from other data sets)				
UPCNT	PUP	I4	0	Number of 'from' pointers (must be < (PSA-PUP-1))
UPVAL(n)	PUP+n	I4	0	Data set number of data set referencing this data set (n=1,UPCNT)
If PSA > 0 (search attributes exist)				
SACNT	PSA	I4	0	Current count of search attributes (must be < (PSASTR-PSA-1)/2)
PSASTR	PSA+1	I4	none	Position pointer to start of search attribute values
For each search attribute, the following pair occur				
SAIND(n)	PSA+(2*n)	I4	none	Index number of search attribute on the read-only MESSAGE.WDM file (n=1,SACNT)
PSAVAL(n)	above+1	I4	none	Position pointer from PSASTR to search attribute value (n=1,SACNT)
Search attributes				
SAVAL(n) PSASTR+PSAVAL(n)	*	*		Search attribute value - type and default are shown in Appendix C by search attribute index number (n=1,SACNT)
TBCNT	PDAT	I4	none	Count of tables (groups) present
FREPOS	PDAT+1	I4	none	Pointer to first free data position with the following bit template
				record (PREC) 22 bits 1-2097151 position within record (POFF) 9 bits 0-511
(for each table, PTAB = PDAT+2+(n-1)*7, n=1,TBCNT)				
TABNAM	PTAB	4A4	none	Table name for summary and check with MESSAGE.WDM file
TABID	PTAB+4	I4	none	Table identifier on MESSAGE.WDM file with the following bit template
				File ID (MSFLID) 16 bits (2 characters) cluster (MCLU) 8 bits (1-255) group (MGRP) 7 bits (1-127)
TABDIM	PTAB+5	I4	none	Table dimensions with the following bit template
				table index number (TABIND) 7 bits (1-127) number of rows (NROW) 9 bits (1-511) column space(words) (NCOL) 7 bits (1-127) extension space (NEXT) 8 bits (1-255)
PDATVL	PTAB+6	I4	none	Pointer to table data, has the same bit template as FREPOS

## APPENDIX A.4 Table Data Set--continued.

<u>Variable</u>	<u>Position</u>	<u>Type</u>	<u>Default</u>	<u>Explanation</u>
(for each table)				
TABCHK	PDATVL	I4	none	Table identifier check (copy of TABID and TABDIM)
EXTVAL	PDATVL+2	*	none	Array of header values NEXT words long
TABVAL	PDATVL+NEXT+2	*	none	Array of table values NROW*NCOL words long

\* type based on table template found at TABID in MESSAGE.WDM file

## APPENDIX A.5 Vector Data Set

<u>Variable</u>	<u>Position</u>	<u>Type</u>	<u>Default</u>	<u>Explanation</u>
PPRBKR	1	I4	0	Primary backward record pointer to previous vector data set (always zero for first data set)
PPRFWR	2	I4	0	Primary forward record pointer to next vector data set (always zero for last data set)
PSCBKR	3	I4	0	Secondary backward record pointer (always zero for first record data set)
PSCFWR	4	I4	0	Secondary forward record pointer to next record within data set (always zero for last record in data set)
DSN	5	I4	none	Unique data set number
DSTYPE	6	I4	5	Data set type (always set to 5 for vector)
-	7	-	-	< not in use >
PDP	8	I4	0	Position of pointers to other data sets
PUP	9	I4	0	Position of pointers from other data sets
PSA	10	I4	0	Position pointer to search attributes
PDAT	11	I4	0	Position pointer to group pointers
PDATV	12	I4	0	Pointer to start of groups
If PDP > 0 (pointers to other data sets)				
DNCNT	PDP	I4	0	Number of 'to' pointers (must be <(PUP-PDP-1))
DNVAL(n)	PDP+n	I4	0	Data set number of data set referenced (n=1,DNCNT)
If PUP > 0 (pointers from other data sets)				
UPCNT	PUP	I4	0	Number of 'from' pointers (must be < (PSA-PUP-1))
UPVAL(n)	PUP+n	I4	0	Data set number of data set referencing this data set (n=1,UPCNT)
If PSA > 0 (search attributes exist)				
SACNT	PSA	I4	0	Current count of search attributes (must be < (PSASTR-PSA-1)/2)
PSASTR	PSA+1	I4	none	Position pointer to start of search attribute values

## APPENDIX A.5 Vector Data Set--continued.

<u>Variable</u>	<u>Position</u>	<u>Type</u>	<u>Default</u>	<u>Explanation</u>
For each search attribute, the following pair occur				
SAIND(n)	PSA+(2*n)	I4	none	Index number of search attribute on the read-only MESSAGE.WDM file (n=1,SACNT)
PSAVAL(n)	above+1	I4	none	Position pointer from PSASTR to search attribute value (n=1,SACNT)
Search attributes				
SAVAL(n)	PSASTR+PSAVAL(n)	*	*	Search attribute value - type and default are shown in Appendix C by search attribute index number (n=1,SACNT)
VECCNT	PDAT	I4	none	Count of group pointers
FREPOS	PDAT+1	I4	none	Pointer to first free data position with the following bit template
				record (PREC) 22 bits 1-2097151 position within record (POFF) 9 bits 0-511
(for each vector group, PVEC = PDAT+2+(n-1)*2, n=1,VECCNT)				
DLWORD	PVEC	I4	none	Vector identifier with the following bit template
				data type (ITYPE) 2 bits 1-3 1 - node 2 - arc 3 - polygon major attribute (ATT1) 10 bits 0-1023 minor attribute (ATT2) 11 bits 0-2047 block count (BLCNT) 8 bits 1-255
PDATVL	(PDAT+1)	I4	none	Pointer to vector data, has the same bit template as FREPOS)
(for each vector group, repeat BLCNT times)				
BWORD	PDATVL	I4	none	Vector block identifier with the following bit template
				block type (ID) 4 bits (1-15) 1 - header 2 - data data type (DTYPE) 2 bits (1-3) 1 - node 2 - arc 3 - area internal number (INUM) 12 bits (1-4095) length of block (ILEN) 13 bits (1-8191)
(for block type 1 - header)				
MATR	PDATVL+1	I4	none	major attribute code
XVAL	PDATVL+2	R4	none	X coordinate value for marker
YVAL	PDATVL+3	R4	none	Y coordinate value for marker
GRNAME	PDATVL+4	nA4	none	name of group - up to 60 characters (15 words) long

## APPENDIX A.5 Vector Data Set--continued.

Variable	Position	Type	Default	Explanation
(for block type 2 - data, n=1,ILEN/2)				
XVAL	PDATVL-1+(n*2)	R4	none	X coordinate value for point on arc or area
YVAL	PDATVL+(n*2)	R4	none	Y coordinate value for point on arc or area

## APPENDIX A.6 Space-Time Data Set

Variable	Position	Type	Default	Explanation
PPRBKR	1	I4	0	Primary backward record pointer to previous space-time data set (always zero for first data set)
PPRFWR	2	I4	0	Primary forward record pointer to next space-time data set (always zero for last data set)
PSCBKR	3	I4	0	Secondary backward record pointer (always zero for first record data set)
PSCFWR	4	I4	0	Secondary forward record pointer to next record within data set (always zero for last record in data set)
DSN	5	I4	none	Unique data set number
DSTYPE	6	I4	7	Data set type (always set to 7 for space-time)
-	7	-	-	< not in use >
PDP	8	I4	0	Position of pointers to other data sets
PUP	9	I4	0	Position of pointers from other data sets
PSA	10	I4	0	Position pointer to search attributes
PDAT	11	I4	0	Position pointer to group pointers
PDATV	12	I4	0	Pointer to start of groups
If PDP > 0 (pointers to other data sets)				
DNCNT	PDP	I4	0	Number of 'to' pointers (must be <(PUP-PDP-1))
DNVAL(n)	PDP+n	I4	0	Data set number of data set referenced (n=1,DNCNT)
If PUP > 0 (pointers from other data sets)				
UPCNT	PUP	I4	0	Number of 'from' pointers (must be < (PSA-PUP-1))
UPVAL(n)	PUP+n	I4	0	Data set number of data set referencing this data set (n=1,UPCNT)
If PSA > 0 (search attributes exist)				
SACNT	PSA	I4	0	Current count of search attributes (must be < (PSASTR-PSA-1)/2)
PSASTR	PSA+1	I4	none	Position pointer to start of search attribute values
For each search attribute, the following pair occur				
SAIND(n)	PSA+(2*n)	I4	none	Index number of search attribute on the read-only MESSAGE.WDM file (n=1,SACNT)

## APPENDIX A.6 Space-Time Data Set--continued.

<u>Variable</u>	<u>Position</u>	<u>Type</u>	<u>Default</u>	<u>Explanation</u>
PSAVAL(n)	above+1	I4	none	Position pointer from PSASTR to search attribute value (n=1,SACNT)
<b>Search attributes</b>				
SAVAL(n) PSASTR+PSAVAL(n)	*	*		Search attribute value - type and default are shown in Appendix C by search attribute index number (n=1,SACNT)
GRPCNT	PDAT	I4	none	count of group pointers
FREPOS	PDAT+1	I4	none	Pointer to first free data position with the following bit template  record (PREC) 22 bits 1-2097151 position within record (POFF) 9 bits 0-511
(for each space time group, PST = PDAT+2+(n-1)*3, n=1,GRPCNT)				
DATWRD	PST	I4	none	Date of the beginning of the group with the following bit template  Year (DAT(1)) 17 bits 1-131071 Month (DAT(2)) 4 bits 1-12 Day (DAT(3)) 5 bits 1-31 Hour (DAT(4)) 5 bits 1-24
GCW	PST+1	I4	none	Group control word with the following bit template  Minutes (STMIN) 6 bits 0-60 Seconds (STSEC) 6 bits 0-60 Time units (TUNITS) 3 bits 1-7 Time step (TSTEP) 6 bits 1-63 Number of values (NOV) 10 bits 1-1023
PDATVL	PST+2	I4	none	Pointer to vector data, has the same bit template as FREPOS)
(for each space time group, n=PDATVL, PDATVL+(DIMX*DIMY*DIMZ*DIMDAT*NOV)-1,DIMDAT)				
IVAL	n	I4	none	Integer data value  or
RVAL	n	R4	none	Real data value  or
DVAL	n	R8	none	Double precision data value

## APPENDIX A.7 Attribute Characteristics Data Set

<u>Variable</u>	<u>Position</u>	<u>Type</u>	<u>Default</u>	<u>Explanation</u>
PPRBKR	1	I4	0	Primary backward record pointer to previous attribute characteristics data set (always zero for first data set)
PPRFWR	2	I4	0	Primary forward record pointer to next attribute characteristics data set (always zero for last data set)

## APPENDIX A.7 Attribute Characteristics Data Set--continued.

<u>Variable</u>	<u>Position</u>	<u>Type</u>	<u>Default</u>	<u>Explanation</u>
PSCBKR	3	I4	0	Secondary backward record pointer (always zero for first record data set)
PSCFWR	4	I4	0	Secondary forward record pointer to next record within data set (always zero for last record in data set)
DSN	5	I4	none	Unique data set number
DSTYPE	6	I4	8	Data set type (always set to 8 for attribute)
-	7	-	-	< not in use >
PDP	8	I4	0	Position of pointers to other data sets
PUP	9	I4	0	Position of pointers from other data sets
PSA	10	I4	0	Position pointer to search attributes
PDAT	11	I4	0	Position pointer to group pointers
PDATV	12	I4	0	Pointer to start of groups
If PDP > 0 (pointers to other data sets)				
DNCNT	PDP	I4	0	Number of 'to' pointers (must be <(PUP-PDP-1))
DNVAL(n)	PDP+n	I4	0	Data set number of data set referenced (n=1,DNCNT)
If PUP > 0 (pointers from other data sets)				
UPCNT	PUP	I4	0	Number of 'from' pointers (must be < (PSA-PUP-1))
UPVAL(n)	PUP+n	I4	0	Data set number of data set referencing this data set (n=1,UPCNT)
If PSA > 0 (search attributes exist)				
SACNT	PSA	I4	0	Current count of search attributes (must be < (PSASTR-PSA-1)/2)
PSASTR	PSA+1	I4	none	Position pointer to start of search attribute values
For each search attribute, the following pair occur				
SAIND(n)	PSA+(2*n)	I4	none	Index number of search attribute on the read-only MESSAGE.WDM file (n=1,SACNT)
PSAVAL(n)	above+1	I4	none	Position pointer from PSASTR to search attribute value (n=1,SACNT)
Search attributes				
SAVAL(n)	PSASTR+PSAVAL(n)	*	*	Search attribute value - type and default are shown in Appendix C by search attribute index number (n=1,SACNT)
GRPCNT	PDAT	I4	none	count of attribute group pointers

## APPENDIX A.7 Attribute Characteristics Data Set--continued.

Variable	Position	Type	Default	Explanation
FREPOS	PDAT+1	I4	none	Pointer to first free data position with the following bit template  record (PREC) 22 bits 1-2097151 position within record (POFF) 9 bits 0-511
(for each attribute , PAT= PDAT+2+(n-1)*4, n=1,GRPCNT)				
ANAM1	PAT	I4	none	Attribute name (1st 4 characters) stored as an integer
AWRD	PAT+1	I4	none	Attribute detail word with the following bit pattern  Attr. name (last 2 chars) (ANAM2) 16 bits (2 characters) Attr. index (ATTIND) 9 bits 1-511
APTWRD	PDAT+2	I4	none	Pointer to attribute details, has the same bit template as FREPOS
ADTWRT	PDAT+3	I4	none	Attribute details with the following bit pattern  Type (ATTYP) 3 bits 0-7 1 - integer 2 - real 3 - character Length (ATLEN) 7 bits 0-127 (repeat next variable 10 times) Data set usage flags (ATUSE) 2 bits 0-2 0 - not allowed 1 - optional 2 - required Update flag (ATUPD) 1 bits 0-1 0 - don't update if data exists 1 - update always allowed
(for each attribute repeat until ID=0, ABPOS starts at APTWRD, increments by TLEN+1)				
ABKWRD	ABPOS	I4	none	Block identifier with the following bit template  Information ID (ID) 16 bits 0,3-7 0 - all done 3 - range 4 - valid responses 5 - default 6 - description 7 - help Info length (TLEN) 9 bits 1-511
(for ID = 3, ATTYP = 1)				
IMIN	ABPOS+1	I4	none	Minimum value allowed for attribute
IMAX	ABPOS+2	I4	none	Maximum value allowed for attribute
(for ID = 3, ATTYP = 2)				
RMIN	ABPOS+1	R4	none	Minimum value allowed for attribute
RMAX	ABPOS+2	R4	none	Maximum value allowed for attribute

## **APPENDIX A.7 Attribute Characteristics Data Set--continued.**

<u>Variable</u>	<u>Position</u>	<u>Type</u>	<u>Default</u>	<u>Explanation</u>
(for ID = 4, 6 or 7, n=1,TLEN)				
STR(n)	ABPOS+n	A4	none	String associated with ID
(for ID = 5, same as ID = 3 but has only 1 value for Default)				

## **APPENDIX A.8 Text and Tables Template Data Set**

<u>Variable</u>	<u>Position</u>	<u>Type</u>	<u>Default</u>	<u>Explanation</u>
PPRBKR	1	I4	0	Primary backward record pointer to previous text and tables template data set (always zero for first data set)
PPRFWR	2	I4	0	Primary forward record pointer to next text and tables template data set (always zero for last data set)
PSCBKR	3	I4	0	Secondary backward record pointer (always zero for first record data set)
PSCFWR	4	I4	0	Secondary forward record pointer to next record within data set (always zero for last record in data set)
DSN	5	I4	none	Unique data set number
DSTYPE	6	I4	9	Data set type (always set to 9 for text and tables template)
-	7	-	-	< not in use >
PDP	8	I4	0	Position of pointers to other data sets
PUP	9	I4	0	Position of pointers from other data sets
PSA	10	I4	0	Position pointer to search attributes
PDAT	11	I4	0	Position pointer to group pointers
PDATV	12	I4	0	Pointer to start of groups
If PDP > 0 (pointers to other data sets)				
DNCNT	PDP	I4	0	Number of 'to' pointers (must be <(PUP-PDP-1))
DNVAL(n)	PDP+n	I4	0	Data set number of data set referenced (n=1,DNCNT)
If PUP > 0 (pointers from other data sets)				
UPCNT	PUP	I4	0	Number of 'from' pointers (must be < (PSA-PUP-1))
UPVAL(n)	PUP+n	I4	0	Data set number of data set referencing this data set (n=1,UPCNT)
If PSA > 0 (search attributes exist)				
SACNT	PSA	I4	0	Current count of search attributes (must be < (PSASTR-PSA-1)/2)
PSASTR	PSA+1	I4	none	Position pointer to start of search attribute values

## APPENDIX A.8 Text and Tables Template Data Set--continued

<u>Variable</u>	<u>Position</u>	<u>Type</u>	<u>Default</u>	<u>Explanation</u>
-----------------	-----------------	-------------	----------------	--------------------

For each search attribute, the following pair occur

SAIND(n)	PSA+(2*n)	I4	none	Index number of search attribute on the read-only MESSAGE.WDM file (n=1,SACNT)
----------	-----------	----	------	--

PSAVAL(n)	above+1	I4	none	Position pointer from PSASTR to search attribute value (n=1,SACNT)
-----------	---------	----	------	--

Search attributes

SAVAL(n) PSASTR+PSAVAL(n)	*	*		Search attribute value - type and default are shown in Appendix C by search attribute index number (n=1,SACNT)
---------------------------	---	---	--	--

DPCNT	PDAT	I4	none	Count of group pointers (must be < (PDATV-PDAT-1))
-------	------	----	------	--

FREPOS	PDAT+1	I4	none	Pointer to first free data position with the following bit template
--------	--------	----	------	---

record	(PREC)	22 bits	1-2097151
position within record	(POFF)	9 bits	0-511

PDATVL(n)	PDAT+1+n	I4	none	Pointer to a group of text and table template data with the same bit template as FREPOS (n = 1,DPCNT)
-----------	----------	----	------	---

For each group of data, there are blocks containing a block control word and data as appropriate. The group is terminated by block control word with a value of 0. BLKPOS starts at PDATVL, increments by TLEN + 1.

BCW	BLKPOS	I4	none	Block control (m > 0) with the following bit template
-----	--------	----	------	---

Class of data in block (CLASS)	4 bits	1-5
1 - 1 dim parm		
2 - 2 dim parm		
3 - text		
4 - menu		
5 - file		
Block identifier (ID)	6 bits	0-63
Order of block info (ORDER)	6 bits	0-63
Length of block (TLEN)	15 bits	0-32767

----

(for 1 or 2 dimensional parameters (CLASS = 1 or 2) blocks may contain the following

(for ID = 1,2,4,5,6,7,9,11,12,13,14,15,16,17,19,20 with n = 1,TLEN)

STR(n)	BLKPOS+n	A4	none	String associated with ID
--------	----------	----	------	---------------------------

(for ID = 8,10,18)

IVAL	BLKPOS+1	I4	none	Integer split with the following bit template
------	----------	----	------	---

First integer value (IV1)	16 bits	0-65535
Second integer value (IV2)	15 bits	0-32767

## APPENDIX A.8 Text and Tables Template Data Set--continued

Variable	Position	Type	Default	Explanation
(for ID = 3)				
IP2VAL	BLKPOS+1	I4	none	Field parameter split with the following bit template
				Field type (FTYP) 4 bits 1-4 1 - integer 2 - real 3 - double precision 4 - character
				Field length (FLEN) 7 bits 1-127
				Field order (FORDER) 2 bits 0-2 0 - no ordering 1 - ascending order 2 - descending order
				Field protection (FPROT) 2 bits 0-2 0 - no protection 1 - value must be in specified range 2 - value may not be modified
				Field starting column (FCOL) 7 bits 1-127
(for ID = 5, FTYP = 1)				
IMIN	BLKPOS+1	I4	none	Minimum value allowed for field
IMAX	BLKPOS+2	I4	none	Maximum value allowed for field
(for ID = 5, FTYP = 2 or 3)				
RMIN	BLKPOS+1	R4	none	Minimum value allowed for field
RMAX	BLKPOS+2	R4	none	Maximum value allowed for field
----				
(for text (CLASS = 3) blocks may contain the following				
(for ID = 1,2,19 with n = 1,TLEN)				
STR(n)	BLKPOS+n	A4	none	String associated with ID
----				
(for menu (CLASS = 4) blocks may contain the following				
(for ID = 1,3,4,5,6,19 with n= 1,TLEN)				
STR(n)	BLKPOS+n	A4	none	String associated with ID
(for ID = 2)				
IMNVAL	BLKPOS+1	I4	none	Menu parameter with the following bit template
				Default response value (IDEF) 6 bits 1-63
				Response length (ILEN) 6 bits 1-63
				No number option (INNU) 1 bit 0-1 0 - no numbers 1 - numbers
				Column width (IWID) 7 bits 1-127
				Length of columns (ICOL) 4 bits 1-15

## APPENDIX A.8 Text and Tables Template Data Set--continued

<u>Variable</u>	<u>Position</u>	<u>Type</u>	<u>Default</u>	<u>Explanation</u>
---				
(for file (CLASS = 5) blocks may contain the following				
(for ID = 1,2,3,4,5,6,19 with n= 1,TLEN)				
STR(n)	BLKPOS+n	A4	none	String associated with ID
(for ID = 7)				
IVAL	BLKPOS+1	I4	none	Record length on file

## **APPENDIX B. IMPORT / EXPORT SEQUENTIAL FILE FORMATS**

\*\*\*\* talk about why import/export needed

\*\*\*\* describe WDIMEX

\*\*\*\* format overview

### **GENERAL INFORMATION**

**always appears at beginning of file**

DATE RECORD			1 record
IDENTIFIER	(1:4)	'DATE'	
TEXT	(6:80)	(any text)	
WDMSFL RECORD			1 record
IDENTIFIER	(1:6)	'WDMSFL'	
TEXT	(8:80)	(any text)	
SYSTEM RECORD			1 record
IDENTIFIER	(1:6)	'SYSTEM'	
TEXT	(8:80)	(any text)	
COMMENT HEADER RECORD			1 record
IDENTIFIER	(1:7)	'COMMENT'	
COMMENT MAIN RECORD			repeat as needed
TEXT	(3:80)	(any text)	
COMMENT TRAILER RECORD			1 record
IDENTIFIER	(1:11)	'END COMMENT'	

1		8
1 34 678	1	0
DATE		
WDMSFL		
SYSTEM		
COMMENT		
This is the first line of comment information		
This is the second line of comment information		
This is the last line of comment information.		
END COMMENT		
1		8
1 34 678	1	0

# DATA SET INFORMATION

repeats for each data set

DATA SET HEADER RECORD		1 record
IDENTIFIER	(1:3)	'DSN'
DATA SET NUMBER	(11:16)	(integer)
FIELD IDENTIFIER 1	(21:24)	'TYPE'
DATA SET TYPE	(27:30)	(character string, valid: TIME, TABL, SCHE, PROJ, VECT, RAST, SPTI, ATTR, MESS)
FIELD IDENTIFIER 2	(34:36)	'NDN'
NUMBER OF DOWN POINTERS	(38:40)	(integer $\geq 0$ )
FIELD IDENTIFIER 3	(44:46)	'NUP'
NUMBER OF UP POINTERS	(48:50)	(integer $\geq 0$ )
FIELD IDENTIFIER 4	(54:56)	'NSA'
NUMBER OF SEARCH ATTRIBUTES	(58:60)	(integer $\geq 0$ )
FIELD IDENTIFIER 5	(64:66)	'NSP'
AMOUNT OF ATTRIBUTE SPACE	(68:70)	(integer $\geq 0$ )
FIELD IDENTIFIER 6	(74:76)	'NDP'
AMOUNT OF DATA POINTER SPACE	(78:80)	(integer $\geq 0$ )
ATTRIBUTE HEADER RECORD		1 record
IDENTIFIER	(3:8)	'LABEL'
ATTRIBUTE SPECIFICATION RECORD		repeat for each attribute
ATTRIBUTE NAME	(5:10)	(character string, see Appendix C for valid values)
INTEGER ATTRIBUTE VALUE OR	(11:20)	(integer)
REAL ATTRIBUTE VALUE OR	(11:20)	(real)
CHARACTER ATTRIBUTE VALUE	(13:80)	(left justified character string)
ATTRIBUTE TRAILER RECORD		1 record
IDENTIFIER	(3:11)	'END LABEL'
DATA HEADER AND DETAIL RECORDS		see each data set type for details
DATA TRAILER RECORD		1 record
IDENTIFIER	(3:10)	'END DATA'
DATA SET TRAILER RECORD		1 record
IDENTIFIER	(1:7)	'END DSN'

11	1	22	2	2	3	3	3	4	4	4	5	5	5	5	6	6	6	6	7	7	7	8		
1	3	5	78	01	3	6	01	4	7	0	4	6	0	4	6	0	4	6	8	0	4	6	8	0
DSN																								
LABEL	12	TYPE	TIME	NDN	2	NUP	2	NSA	30	NSP	??	NDP	100											
ISTAID	6714310																							
TSTYPE	PREC																							
TSFILL	0.																							
END LABEL																								
END DATA																								
END DSN																								
11	1	22	2	2	3	3	3	4	4	4	5	5	5	5	6	6	6	6	7	7	7	8		
1	3	5	78	01	3	6	01	4	7	0	4	6	0	4	6	0	4	6	8	0	4	6	8	0

# TIMESERIES DATA

DATA HEADER RECORD		1 record
IDENTIFIER	(3:6)	'DATA'
FIELD IDENTIFIER 1	(14:20)	'STARTS:'
START YEAR	(22:25)	(integer)
START MONTH	(27:28)	(integer 1-12)
START DAY	(30:31)	(integer 1-31)
START HOUR	(33:34)	(integer 0-24)
START MINUTE	(36:37)	(integer 0-60)
START SECOND	(39:40)	(integer 0-60)
FIELD IDENTIFIER 2	(43:47)	'ENDS:'
END YEAR	(49:52)	(integer)
END MONTH	(54:55)	(integer 1-12)
END DAY	(57:58)	(integer 1-31)
END HOUR	(60:61)	(integer 0-24)
END MINUTE	(63:64)	(integer 0-60)
END SECOND	(66:67)	(integer 0-60)
BLOCK HEADER RECORD		repeat for each block
BLOCK START YEAR	(5:8)	(integer)
BLOCK START MONTH	(10:11)	(integer 1-12)
BLOCK START DAY	(13:14)	(integer 1-31)
BLOCK START HOUR	(16:17)	(integer 0-24)
BLOCK START MINUTE	(19:20)	(integer 0-60)
BLOCK START SECOND	(22:23)	(integer 0-60)
BLOCK TIME UNITS	(27:29)	(integer 1-6)
BLOCK TIME STEP	(30:32)	(integer 1-60)
BLOCK QUALITY CODE	(33:35)	(integer 0-31)
NUMBER OF VALUES IN BLOCK	(36:41)	(integer)
BLOCK COMPRESSION CODE	(42:44)	(integer 0-1)
FIRST BLOCK DATA VALUE	(53:63)	(real)
BLOCK SUPPLEMENTAL RECORDS		repeat as needed to complete block
BLOCK DATA VALUES	(5:15)	(real)
	(17:27)	(real)
	(29:39)	(real)
	(41:51)	(real)
	(53:63)	(real)
	(65:75)	(real)

11	1	22	2	2	3	3	3	4	4	4	5	5	5	5	6	6	6	6	7	7	7	8		
1	3	5	78	01	3	6	01	4	7	0	4	6	0	4	6	0	4	6	8	0	4	6	8	0
<b>*****</b>																								
<b>***** need sample here *****</b>																								
<b>*****</b>																								
11	1	22	2	2	3	3	3	4	4	4	5	5	5	5	6	6	6	6	7	7	7	8		
1	3	5	78	01	3	6	01	4	7	0	4	6	0	4	6	0	4	6	8	0	4	6	8	0

## TABLE DATA

DATA HEADER RECORD		repeat for each table
IDENTIFIER	(3:6)	'DATA'
TABLE NAME IDENTIFIER	(9:13)	'NAME'
TABLE NAME	(15:30)	(character string)
TABLE INDEX IDENTIFIER	(33:35)	'IND'
TABLE INDEX NUMBER	(36:40)	(integer > 0)
MESSAGE FILE ID IDENTIFIER	(43:45)	'MID'
MESSAGE FILE ID	(48:49)	(two character string)
TABLE TEMPLATE CLUSTER IDEN.	(52:54)	'CLU'
TABLE TEMPLATE CLUSTER NUMBER	(55:59)	(integer > 0)
TABLE TEMPLATE GROUP IDEN.	(62:64)	'GRP'
TABLE TEMPLATE GROUP NUMBER	(65:69)	(integer > 0)
NUMBER OF ROWS IDENTIFIER	(72:74)	'NRW'
NUMBER OF ROWS	(75:79)	(integer > 0)
EXTENSION DATA IDENTIFIER RECORD		optional 1 record
IDENTIFIER	(5:18)	'EXTENSION DATA'
EXTENSION DATA HEADER RECORD		optional, repeat as needed
EXTENSION DATA HEADER	(1:80)	(character string from associated table template; '****' inserted to indicate record is a comment)
EXTENSION DATA RECORD		optional 1 record
EXTENSION DATA	(1:80)	(character and numeric values in format defined by associated table template; table index inserted at start of record)
EXTENSION DATA END RECORD		optional 1 record
IDENTIFIER	(5:22)	'END EXTENSION DATA'
MAIN DATA IDENTIFIER RECORD		1 record
IDENTIFIER	(5:13)	'MAIN DATA'
MAIN DATA HEADER RECORD		optional, repeat as needed
MAIN DATA HEADER	(1:80)	(character string from table template; '****' inserted to indicate record is a comment)
MAIN DATA RECORD		repeat for each row
MAIN DATA	(1:80)	(numeric and character values in format defined by a table template)
MAIN DATA END RECORD		1 record
IDENTIFIER	(5:17)	'END MAIN DATA'

11	1	22	2	2	3	3	3	4	4	4	5	5	5	5	6	6	6	7	7	7	8			
1	3	5	7	8	0	1	3	6	0	1	4	7	0	4	6	0	4	6	8	0	4	6	8	0
*****	***** need sample here *****	*****	*****	*****	*****	*****	*****	*****	*****	*****	*****	*****	*****	*****	*****	*****	*****	*****	*****	*****	*****	*****	*****	*****
11	1	22	2	2	3	3	3	4	4	4	5	5	5	5	6	6	6	7	7	7	8			
1	3	5	7	8	0	1	3	6	0	1	4	7	0	4	6	0	4	6	8	0	4	6	8	0

## VECTOR DATA

DATA HEADER RECORD			1 record
IDENTIFIER	(3:6)	'DATA'	
GROUP HEADER RECORD			repeat for each group
IDENTIFIER	(1:5)	'GROUP'	
FIELD IDENTIFIER 1	(8:11)	'TYPE'	
GROUP TYPE	(13:16)	(character string, valid: LINE, AREA, NODE)	
FIELD IDENTIFIER 2	(21:28)	'MAJ ATTR'	
MAJOR ATTRIBUTE	(29:33)	(integer)	
FIELD IDENTIFIER 3	(36:43)	'MIN ATTR'	
MINOR ATTRIBUTE	(44:48)	(integer)	
BLOCK HEADER RECORD			1 record
IDENTIFIER	(3:3)	'1'	
HEADER LENGTH	(4:8)	(integer > 0)	
MAJOR ATTRIBUTE	(9:14)	(integer)	
X MARKER COORDINATE	(15:26)	(real) only required if HEADER LENGTH > 1	
Y MARKER COORDINATE	(27:38)	(real) only required if HEADER LENGTH > 1	
BLOCK HEADER CONTINUE RECORD			1 record needed only if HEADER LENGTH > 3
GROUP NAME	(1:80)	(character string)	
BLOCK DATA RECORD			repeat as needed to complete group
IDENTIFIER	(3:3)	'2'	
DATA LENGTH	(4:8)	(integer > 0)	
BLOCK DATA CONTINUE RECORD			repeat as needed to complete block
X COORDINATE	(1:12)	(real)	
Y COORDINATE	(13:24)	(real)	
X COORDINATE	(25:36)	(real)	
Y COORDINATE	(37:48)	(real)	
X COORDINATE	(49:60)	(real)	
Y COORDINATE	(61:72)	(real)	

11	1	22	2	2	3	3	3	4	4	4	5	5	5	5	6	6	6	7	7	7	8			
1	3	5	78	01	3	6	01	4	7	0	4	6	0	4	6	0	4	6	8	0	4	6	8	0
*****																								
***** need sample here *****																								
*****																								
11	1	22	2	2	3	3	3	4	4	4	5	5	5	5	6	6	6	7	7	7	8			
1	3	5	78	01	3	6	01	4	7	0	4	6	0	4	6	0	4	6	8	0	4	6	8	0

# SPACE TIME DATA

DATA HEADER RECORD		1 record
IDENTIFIER	(3:6)	'DATA'
BLOCK HEADER RECORD		repeat for each block
BLOCK START YEAR	(5:8)	(integer)
BLOCK START MONTH	(10:11)	(integer 1-12)
BLOCK START DAY	(13:14)	(integer 1-31)
BLOCK START HOUR	(16:17)	(integer 0-24)
BLOCK START MINUTE	(19:20)	(integer 0-60)
BLOCK START SECOND	(22:23)	(integer 0-60)
BLOCK TIME UNITS	(27:29)	(integer 1-6)
BLOCK TIME STEP	(30:32)	(integer 1-60)
NUMBER OF VALUES IN BLOCK	(36:41)	(integer)
BLOCK SUPPLEMENTAL RECORDS		repeat as needed to complete block
BLOCK DATA VALUES	(5:15)	(real)
	(17:27)	(real)
	(29:39)	(real)
	(41:51)	(real)
	(53:63)	(real)
	(65:75)	(real)

11	1	22	2	2	3	3	3	4	4	4	5	5	5	5	6	6	6	6	7	7	7	8		
1	3	5	78	01	3	6	01	4	7	0	4	6	0	4	6	0	4	6	8	0	4	6	8	0
*****																								
***** need sample here *****																								
*****																								
11	1	22	2	2	3	3	3	4	4	4	5	5	5	5	6	6	6	6	7	7	7	8		
1	3	5	78	01	3	6	01	4	7	0	4	6	0	4	6	0	4	6	8	0	4	6	8	0

## ATTRIBUTE DATA

DATA HEADER RECORD			1 record
IDENTIFIER	(3:6)	'DATA'	
FIELD IDENTIFIER	(10:12)	'DSN'	
DATA SET NUMBER	(14:18)	(integer >= 0)	
GROUP HEADER RECORD			repeat for each group
IDENTIFIER	(1:10)	'#ATTRIBUTE'	
ATTRIBUTE NAME	(12:17)	(character string, see Appendix C for values)	
FIELD IDENTIFIER 1	(21:25)	'INDEX'	
INDEX NUMBER	(26:31)	(integer > 0)	
TYPE RECORD			1 record
IDENTIFIER	(1:5)	'\$TYPE'	
TYPE	(8:23)	(character string, valid: INTEGER,REAL,CHARACTER,DOUBLE PRECISION	
LENGTH RECORD			1 record
IDENTIFIER	(1:7)	'\$LENGTH'	
LENGTH	(9:12)	(integer, generally 1)	
REQUIRED RECORD			optional 1 record
IDENTIFIER	(1:9)	'\$REQUIRED'	
REQUIRED FOR DATA SET TYPES	(11:80)	(character string, valid TIMESERIES, TABLE, VECTOR, SPACE-TIME, ATTRIBUTE, MESSAGE, multiple allowed, separated by commas	
OPTIONAL RECORD			optional 1 record
IDENTIFIER	(1:9)	'\$OPTIONAL'	
OPTIONAL FOR DATA SET TYPES	(11:80)	(character string, same as REQUIRED RECORD)	
UPDATE RECORD	optional	1 record if updates to attribute not allowed	
IDENTIFIER	(1:7)	'\$UPDATE'	
RANGE RECORD			optional 1 record
allowed if TYPE is INTEGER, REAL, or DOUBLE			
IDENTIFIER	(1:6)	'\$RANGE'	
MINIMUM ALLOWED VALUE	(8:17)	(integer or real depending on TYPE)	
SEPARATOR	(18:18)	:	
MAXIMUM ALLOWED VALUE	(19:28)	(integer or real depending on TYPE)	
DEFAULT RECORD			optional 1 record
allowed if TYPE is INTEGER, REAL, or DOUBLE			
IDENTIFIER	(1:8)	'\$DEFAULT'	
DEFAULT VALUE	(9:18)	(integer or real depending on TYPE)	
VALID RECORD			optional 1 record
IDENTIFIER	(1:6)	'\$VALID'	
VALID VALUES	(8:80)	(character strings of size <= LENGTH delimited by commas)	
DESCRIPTION RECORD			optional 1 record
IDENTIFIER	(1:5)	'\$DESC'	
DESCRIPTION	(7:80)	(character string)	
HELP RECORD			optional 1 record
IDENTIFIER	(1:5)	'\$HELP'	
HELP CONTINUATION RECORD			repeat as needed
HELP	(1:78)	(character string)	

1	3	5	7	8	0	1	3	6	0	1	2	2	2	3	3	3	4	4	4	5	5	5	5	6	6	6	6	7	7	7	8
*****																															
***** need sample here *****																															
*****																															
1	3	5	7	8	0	1	3	6	0	1	2	2	2	3	3	3	4	4	4	5	5	5	5	6	6	6	6	7	7	7	8

## TEXT AND TABLE TEMPLATE DATA

DATA HEADER RECORD		1 record
IDENTIFIER	(3:6)	'DATA'
FIELD IDENTIFIER	(10:12)	'DSN'
DATA SET NUMBER	(14:18)	(integer >= 0)
GROUP HEADER RECORD		repeat for each group
IDENTIFIER	(1:6)	'#GROUP'
GROUP NUMBER	(8:12)	(integer > 0)
TYPE IDENTIFIER	(14:17)	'TYPE'
TYPE	(19:22)	(character string, valid: TEXT, MENU, PRM1, PRM2, FILE)
FOR TYPE TEXT:		
TEXT RECORD		1 record
IDENTIFIER	(1:5)	'\$TEXT'
TEXT CONTINUATION RECORD		repeat as needed
TEXT	(1:78)	(character string)
HELP RECORD		optional, 1 record
IDENTIFIER	(1:5)	'\$HELP'
HELP CONTINUATION RECORD		repeat as needed
HELP	(1:78)	(character string)
WINDOW RECORD		optional, 1 record
IDENTIFIER	(1:7)	'\$WINDOW'
WINDOW NAME	(9:56)	(character string)

11	1	22	2	2	3	3	3	4	4	4	5	5	5	5	6	6	6	6	7	7	7	8		
1	3	5	78	01	3	6	01	4	7	0	4	6	0	4	6	0	4	6	8	0	4	6	8	0
*****																								
***** need sample here *****																								
*****																								
11	1	22	2	2	3	3	3	4	4	4	5	5	5	5	6	6	6	6	7	7	7	8		
1	3	5	78	01	3	6	01	4	7	0	4	6	0	4	6	0	4	6	8	0	4	6	8	0

FOR TYPE MENU

TITLE RECORD  
 IDENTIFIER (1:6) '\$TITLE'  
 TITLE (8:80) (character string)  
 optional, 1 record

DEFAULT RECORD  
 IDENTIFIER (1:8) '\$DEFAULT'  
 DEFAULT (10:12) (integer > 0 but <= number of menu options)  
 1 record

LENGTH RECORD  
 IDENTIFIER (1:7) '\$LENGTH'  
 LENGTH (9:11) (integer > 0 and <= 63)  
 optional, 1 record

WIDTH RECORD  
 IDENTIFIER (1:6) '\$WIDTH'  
 WIDTH (8:10) (integer > 0 and <= 78, default = 78)  
 optional, 1 record

COLENGTH RECORD  
 IDENTIFIER (1:9) '\$COLENGTH'  
 COLENGTH (11:13) (integer > 0 and <= 8, default = number of options)  
 repeat for each menu option

OPTION RECORD  
 IDENTIFIER (1:7) '\$OPTION'  
 OPTION (9:71) (character string, length must be <= LENGTH)  
 optional, 1 record

DESCRIPTION RECORD  
 IDENTIFIER (1:5) '\_DESC'  
 DESCRIPTION (7:80) (character string, length must be < WIDTH - LENGTH - 3)

OPTION HELP RECORD  
 IDENTIFIER (1:5) '\_HELP'  
 optional, 1 record

OPTION HELP CONTINUATION RECORD  
 OPTION HELP (1:78) (character string)  
 repeat as needed

>> RECORDS 'HELP', 'HELP CONTINUATION' AND 'WINDOW' SAME AS TEXT

11	1	22	2	2	3	3	3	4	4	4	5	5	5	5	6	6	6	6	7	7	7	8	
1	3	5	78	01	3	6	01	4	7	0	4	6	0	4	6	0	4	6	8	0	4	6	8
*****																							
***** need sample here *****																							
*****																							
11	1	22	2	2	3	3	3	4	4	4	5	5	5	5	6	6	6	6	7	7	7	8	
1	3	5	78	01	3	6	01	4	7	0	4	6	0	4	6	0	4	6	8	0	4	6	8

FOR TYPE PRM1:

SCREEN RECORD IDENTIFIER (1:7) '\$SCREEN' 1 record

SCREEN CONTINUATION RECORD SCREEN (1:78) repeat as needed up to 16 times (character string)

FIELD RECORD IDENTIFIER (1:6) repeat as needed for up to 30 fields '\$FIELD'

FIELD NAME (8:80) (character string, must match field name in SCREEN text)

TYPE RECORD IDENTIFIER (1:5) '\_TYPE' 1 record

TYPE (8:23) (character string, valid: INTEGER, REAL, CHARACTER, DOUBLE PRECISION) optional 1 record

RANGE RECORD allowed if TYPE is INTEGER, REAL or DOUBLE PRECISION

IDENTIFIER (1:6) '\_RANGE'

MINIMUM ALLOWED VALUE (8:17) (integer or real depending on TYPE)

SEPARATOR (18:18) ':'

MAXIMUM ALLOWED VALUE (19:28) (integer or real depending on TYPE)

DEFAULT RECORD IDENTIFIER (1:8) '\_DEFAULT' optional 1 record

DEFAULT VALUE (9:18) (integer, real or character depending on TYPE)

VALID RECORD IDENTIFIER (1:6) '\_VALID' optional 1 record

VALID VALUES (8:80) (character strings of size <= size of field delimited by commas)

INVALID RECORD IDENTIFIER (1:6) '\_INVALID' optional 1 record

INVALID VALUES (8:80) (character strings of size <= size of field delimited by commas)

PROTECT RECORD IDENTIFIER (1:8) '\_PROTECT' optional 1 record

PROTECTION (10:18) (character string, valid: NONE(default), CORRECT, PROTECTED)

FIELD HELP RECORD IDENTIFIER (1:5) '\_HELP' optional, 1 record

FIELD HELP CONTINUATION RECORD FIELD HELP (1:78) (character string) repeat as needed

>> RECORDS 'HELP', 'HELP CONTINUATION' AND 'WINDOW' SAME AS TEXT

11	1	22	2	2	3	3	3	4	4	4	5	5	5	5	6	6	6	7	7	7	8			
1	3	5	7	8	0	1	3	6	0	1	4	7	0	4	6	0	4	6	8	0	4	6	8	0
*****																								
***** need sample here *****																								
*****																								
11	1	22	2	2	3	3	3	4	4	4	5	5	5	5	6	6	6	7	7	7	8			
1	3	5	7	8	0	1	3	6	0	1	4	7	0	4	6	0	4	6	8	0	4	6	8	0

FOR TYPE PRM2:

HEADER RECORD		1 record
IDENTIFIER	(1:7)	'\$HEADER'
HEADER CONTINUATION RECORD		repeat as needed up to 15 times
SCREEN HEADER	(1:78)	(character string)
FIELD RECORD		repeat as needed for up to 30 fields
IDENTIFIER	(1:6)	'\$FIELD'
FIELD NAME	(8:80)	(character string)
TYPE RECORD		1 record
IDENTIFIER	(1:5)	'_TYPE'
TYPE	(8:23)	(character string, valid: INTEGER, REAL, CHARACTER, DOUBLE PRECISION)
WIDTH RECORD		1 record
IDENTIFIER	(1:6)	'_WIDTH'
FIELD WIDTH	(8:11)	(integer > 0 and < 78)
ORDER RECORD		optional, 1 record
IDENTIFIER	(1:6)	'_ORDER'
FIELD ORDERING	(8:17)	(character string, valid: RANDOM(default), ASCENDING, DESCENDING)
COLUMN RECORD		1 record
IDENTIFIER	(1:7)	'_COLUMN'
STARTING COLUMN OF FIELD	(9:12)	(integer > 0)

>> RECORDS 'FIELD HELP', 'FIELD HELP CONTINUATION', 'RANGE',  
'VALID' AND 'INVALID' SAME AS PRM1                                  'DEFAULT',

PARAMETER NAME RECORD		optional 1 record
IDENTIFIER	(1:5)	'_PNAME'
TABLE PARAMETER NAME	(7:18)	(character string)
PARAMETER CODE RECORD		optional 1 record
IDENTIFIER	(1:5)	'_PCODE'
TABLE PARAMETER CODE	(7:11)	(integer value)
UNITS CODE RECORD		optional 1 record
IDENTIFIER	(1:5)	'_UCODE'
TABLE UNITS CODE	(7:11)	(integer value)
TABLE NAME RECORD		optional 1 record
IDENTIFIER	(1:6)	'\$TNAME'
TABLE NAME	(8:80)	(character string)

>> RECORDS 'HELP', 'HELP CONTINUATION' AND 'WINDOW' SAME AS TEXT

11	1	22	2	2	3	3	3	4	4	4	5	5	5	5	6	6	6	6	7	7	7	8	
1	3	5	78	01	3	6	01	4	7	0	4	6	0	4	6	0	4	6	8	0	4	6	8
*****																							
***** need sample here *****																							
*****																							
11	1	22	2	2	3	3	3	4	4	4	5	5	5	5	6	6	6	6	7	7	7	8	
1	3	5	78	01	3	6	01	4	7	0	4	6	0	4	6	0	4	6	8	0	4	6	8

**FOR TYPE FILE:**

>> RECORDS 'WINDOW', 'SCREEN', AND 'HELP' SAME AS PRM1

NAME RECORD

1 record

IDENTIFIER (1:5) '\$NAME'  
FILE/FIELD NAME (7:80) (character string: name of file)  
(\$SCREEN not allowed) or field name in  
\$SCREEN text

STATUS RECORD

optional 1 record

IDENTIFIER (1:7) '\$STATUS'  
FILE STATUS (9:15) (character string, valid:  
OLD(default), NEW, UNKNOWN, SCRATCH)  
optional 1 record

ACCESS RECORD

IDENTIFIER (1:7) '\$ACCESS'  
FILE ACCESS (9:18) (character string, valid:  
SEQUENTIAL(default) or DIRECT)  
optional 1 record

FORMAT RECORD

IDENTIFIER (1:7) '\$FORMAT'  
FILE FORMAT (9:19) (character string, valid:  
FORMATTED(default) or UNFORMATTED)  
optional 1 record

RECORD LENGTH RECORD

IDENTIFIER (1:5) '\$RECL'  
FILE RECORD LENGTH (7:11) (integer > 0)

1	3	5	78	01	3	6	01	4	7	0	4	6	0	4	6	0	4	6	8	0	4	6	8	0	4	6	8	0
*****																												
***** need sample here *****																												
*****																												
1	3	5	78	01	3	6	01	4	7	0	4	6	0	4	6	0	4	6	8	0	4	6	8	0	4	6	8	0

## **APPENDIX C. ATTRIBUTES**

\*\*\*\*\* describe what attributes are

\*\*\*\*\* indicate where to find in seq form, what dsn's are reserved

\*\*\*\*\* reference wdimex section

\*\*\*\*\* describe app c.1 - attributes and their characteristics

\*\*\*\*\* describe app c.2 - attribute names sorted by index number

\*\*\*\*\* describe app c.3 - required attributes by dataset type

\*\*\*\*\* discuss user defined attributes - coordination with USGS

### **Appendix C.1 Attributes and Their Characteristics**

ATTRIBUTE NAME	INDEX NUMBER	TYPE	LENGTH	UPDATE	DESCRIPTION
ACODE	19	INT	1	YES	Area units code, user defined.
AGENCY	40	CHAR	8	YES	Agency code. See WATSTORE users manual, volume 1, chapter 3.
AQTYPE	48	CHAR	4	YES	Aquifer type. See WATSTORE users manual, volume 1, chapter 3. U - unconfined single aquifer N - unconfined multiple aquifers C - confined single aquifer M - confined multiple aquifers X - mixed multiple aquifers
AZMUTH	95	REAL	1	YES	Azimuth, in decimal degrees from north of a straight line connecting points 85- and 10-percent of distance from gage to divide. Basin and streamflow characteristic no 18, AZIMUTH. See WATSTORE users manual, Appendix.
BASEQ	49	REAL	1	YES	Base discharge, in cubic feet per second. See WATSTORE user manual, volume 1, chapter 3.
BLNGTH	87	REAL	1	YES	Stream length, in miles, from gage to end of defined channel, blue line on topographic map. Basin and streamflow characteristic no 6, BLENGTH. See WATSTORE users manual, Appendix.
BRANCH	261	INT	1	YES	Integer ID number of a channel segment.
BSLOPE	86	REAL	1	YES	Average basin slope, in feet per mile. Measured by grid sampling method. Basin and streamflow characteristic no 4, BSLOPE. See WATSTORE users manual, Appendix.
CHEAT	82	INT	1	YES	Pointer to an associated data set. Used to associate quality flags for peak flow data, needed for J407.

## **Appendix C.1**

### **Attributes and Their Characteristics--continued**

ATTRIBUTE	INDEX				
NAME	NUMBER	TYPE	LENGTH	UPDATE	DESCRIPTION
COCODE	6	INT	1	YES	County or parish code. See WATSTORE users manual, Appendix C.
COMPFG	83	INT	1	NO	Compression flag 1 - yes, data are compressed (default) 2 - no, data are not compressed compressed data will take up less space in the WDM file, but will require a COPY operation to update data values.
CONTDA	43	REAL	1	YES	Drainage area, in square miles, that contributes to surface runoff.
DAREA	11	REAL	1	YES	Total drainage area, in square miles, including non-contributing areas.
DATE	50	INT	6	YES	Date string. User defined use.
DATUM	264	REAL	1	YES	Reference elevation, to mean sea level.
DCODE	22	INT	1	YES	Attribute DCODE
DEPTH25	210	REAL	1	YES	Flow depth, in feet. Corresponding to the difference between the 25 percent flow duration gage height and point of zero flow. Basin and streamflow characteristic no 168, DEPTH25. See WATSTORE users manual, Appendix.
DEPTH	259	REAL	1	YES	Sampling depth, in feet, at which observation was made.
DESCRP	10	CHAR	80	YES	Data-set description. Might be name and/or location, or some anecdotal information.
DSCODE	42	INT	1	YES	State code of the Geological Survey office that operates the station. Usually the same as the state code (STFIPS). See WATSTORE Users manual, Appendix B.
EL1085	89	REAL	1	YES	Average of channel elevations, in feet above mean sea level, at points 10- and 85-percent of stream length upstream from gage. Basin and streamflow characteristic no 9, ELV10,85. See WATSTORE users manual, Appendix.
EL5000	90	REAL	1	YES	Percent of basin above elevation 5000 feet, mean sea level. Basin and streamflow characteristic no 10, EL5000. See WATSTORE users manual, Appendix.
EL6000	91	REAL	1	YES	Percent of basin above elevation 5000 feet, mean sea level. Basin and streamflow characteristic no 11, EL6000. See WATSTORE users manual, Appendix.
ELEV	7	REAL	1	YES	Elevation (mean sea level).

## Appendix C.1

### Attributes and Their Characteristics--continued

ATTRIBUTE NAME	INDEX NUMBER	TYPE	LENGTH	UPDATE	DESCRIPTION
FOREST	61	REAL	1	YES	Forested area in percent of contributing drainage area, measured by the grid sampling methods. Basin and streamflow characteristic no 14, FOREST. See WATSTORE users manual, Appendix.
FROST	129	REAL	1	YES	Mean frost depth on February 28, in inches. From U.S. Weather Bureau, "Climates of States". Basin and streamflow characteristic no 72, FROST. See WATSTORE users manual, Appendix.
GCODE	23	INT	1	YES	Angle (slope) code, user defined.
GLACIER	93	REAL	1	YES	Area of glaciers in percent of contributing drainage area. Basin and streamflow characteristic no 15, GLACIER. See WATSTORE users manual, Appendix.
GRPNAM	263	CHAR	8	YES	Six character name for a cluster of message type data-set groups.
GUCODE	46	CHAR	12	YES	Geologic unit code. See WATSTORE users manual, Appendix F.
H01002	175	REAL	1	YES	Annual maximum 1-day mean discharge, in cubic feet per second for 2-year recurrence interval, defined by log-Pearson Type III fitting in ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 136, V1,2. See WATSTORE users manual, Appendix.
H01005	176	REAL	1	YES	Annual maximum 1-day mean discharge, in cubic feet per second for 5-year recurrence interval, defined by log-Pearson Type III fitting in ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 137, V1,5. See WATSTORE users manual, Appendix.
H01010	177	REAL	1	YES	Annual maximum 1-day mean discharge, in cubic feet per second for 10-year recurrence interval, defined by log-Pearson Type III fitting in ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 138, V1,10. See WATSTORE users manual, Appendix.
H01020	178	REAL	1	YES	Annual maximum 1-day mean discharge, in cubic feet per second for 20-year recurrence interval, defined by log-Pearson Type III fitting in ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 139, V1,20. See WATSTORE users manual, Appendix.
H01025	179	REAL	1	YES	Annual maximum 1-day mean discharge, in cubic feet per second for 25-year recurrence interval, defined by log-Pearson Type III fitting in ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 140, V1,25. See WATSTORE users manual, Appendix.

## **Appendix C.1**

### **Attributes and Their Characteristics--continued**

ATTRIBUTE NAME	INDEX NUMBER	TYPE	LENGTH	UPDATE	DESCRIPTION
H01050	180	REAL	1	YES	Annual maximum 1-day mean discharge, in cubic feet per second for 50-year recurrence interval, defined by log-Pearson Type III fitting in ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 141, V1,50. See WATSTORE users manual, Appendix.
H01100	181	REAL	1	YES	Annual maximum 1-day mean discharge, in cubic feet per second for 100-year recurrence interval, defined by log-Pearson Type III fitting in ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 131, V1,100. See WATSTORE users manual, Appendix.
H03002	182	REAL	1	YES	Annual maximum 3-day mean discharge, in cubic feet per second for 2-year recurrence interval, defined by log-Pearson Type III fitting in ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 142, V3,2. See WATSTORE users manual, Appendix.
H03005	183	REAL	1	YES	Annual maximum 3-day mean discharge, in cubic feet per second for 5-year recurrence interval, defined by log-Pearson Type III fitting in ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 143, V3,5. See WATSTORE users manual, Appendix.
H03010	184	REAL	1	YES	Annual maximum 3-day mean discharge, in cubic feet per second for 10-year recurrence interval, defined by log-Pearson Type III fitting in ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 144, V3,10. See WATSTORE users manual, Appendix.
H03020	185	REAL	1	YES	Annual maximum 3-day mean discharge, in cubic feet per second for 20-year recurrence interval, defined by log-Pearson Type III fitting in ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 145, V3,20. See WATSTORE users manual, Appendix.
H03025	186	REAL	1	YES	Annual maximum 3-day mean discharge, in cubic feet per second for 25-year recurrence interval, defined by log-Pearson Type III fitting in ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 146, V3,25. See WATSTORE users manual, Appendix.
H03050	187	REAL	1	YES	Annual maximum 3-day mean discharge, in cubic feet per second for 50-year recurrence interval, defined by log-Pearson Type III fitting in ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 147, V3,50. See WATSTORE users manual, Appendix.
H03100	188	REAL	1	YES	Annual maximum 3-day mean discharge, in cubic feet per second for 100-year recurrence interval, defined by log-Pearson Type III fitting in ANNIE

---

## **Appendix C.1**

### **Attributes and Their Characteristics--continued**

---

ATTRIBUTE	INDEX				
NAME	NUMBER	TYPE	LENGTH	UPDATE	DESCRIPTION

or WATSTORE program A969. Basin and streamflow characteristic no 148, V3,100. See WATSTORE users manual, Appendix.

## **Appendix C.1**

### **Attributes and Their Characteristics--continued**

ATTRIBUTE NAME	INDEX NUMBER	TYPE	LENGTH	UPDATE	DESCRIPTION
H07002	189	REAL	1	YES	Annual maximum 7-day mean discharge, in cubic feet per second for 2-year recurrence interval, defined by log-Pearson Type III fitting in ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 149, V7,2. See WATSTORE users manual, Appendix.
H07005	190	REAL	1	YES	Annual maximum 7-day mean discharge, in cubic feet per second for 5-year recurrence interval, defined by log-Pearson Type III fitting in ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 150, V7,5. See WATSTORE users manual, Appendix.
H07010	191	REAL	1	YES	Annual maximum 7-day mean discharge, in cubic feet per second for 10-year recurrence interval, defined by log-Pearson Type III fitting in ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 151, V7,10. See WATSTORE users manual, Appendix.
H07020	192	REAL	1	YES	Annual maximum 7-day mean discharge, in cubic feet per second for 20-year recurrence interval, defined by log-Pearson Type III fitting in ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 152, V7,20. See WATSTORE users manual, Appendix.
H07025	193	REAL	1	YES	Annual maximum 7-day mean discharge, in cubic feet per second for 25-year recurrence interval, defined by log-Pearson Type III fitting in ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 153, V7,25. See WATSTORE users manual, Appendix.
H07050	194	REAL	1	YES	Annual maximum 7-day mean discharge, in cubic feet per second for 50-year recurrence interval, defined by log-Pearson Type III fitting in ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 154, V7,50. See WATSTORE users manual, Appendix.
H07100	195	REAL	1	YES	Annual maximum 7-day mean discharge, in cubic feet per second for 100-year recurrence interval, defined by log-Pearson Type III fitting in ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 155, V7,100. See WATSTORE users manual, Appendix.
H15002	196	REAL	1	YES	Annual maximum 15-day mean discharge, in cubic feet per second for 2-year recurrence interval, defined by log-Pearson Type III fitting in ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 156, V15,2. See WATSTORE users manual, Appendix.

## **Appendix C.1**

### **Attributes and Their Characteristics--continued**

ATTRIBUTE INDEX					
NAME	NUMBER	TYPE	LENGTH	UPDATE	DESCRIPTION
H15005	197	REAL	1	YES	Annual maximum 15-day mean discharge, in cubic feet per second for 5-year recurrence interval, defined by log-Pearson Type III fitting in ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 132, V15,5. See WATSTORE users manual, Appendix.
H15010	198	REAL	1	YES	Annual maximum 15-day mean discharge, in cubic feet per second for 10-year recurrence interval, defined by log-Pearson Type III fitting in ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 157, V15,10. See WATSTORE users manual, Appendix.
H15020	199	REAL	1	YES	Annual maximum 15-day mean discharge, in cubic feet per second for 20-year recurrence interval, defined by log-Pearson Type III fitting in ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 158, V15,20. See WATSTORE users manual, Appendix.
H15025	200	REAL	1	YES	Annual maximum 15-day mean discharge, in cubic feet per second for 25-year recurrence interval, defined by log-Pearson Type III fitting in ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 159, V15,25. See WATSTORE users manual, Appendix.
H15050	201	REAL	1	YES	Annual maximum 15-day mean discharge, in cubic feet per second for 50-year recurrence interval, defined by log-Pearson Type III fitting ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 160, V15,50. See WATSTORE users manual, Appendix.
H15100	202	REAL	1	YES	Annual maximum 15-day mean discharge, in cubic feet per second for 100-year recurrence interval, defined by log-Pearson Type III fitting ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 161, V15,100. See WATSTORE users manual, Appendix.
H30002	203	REAL	1	YES	Annual maximum 30-day mean discharge, in cubic feet per second for 2-year recurrence interval, defined by log-Pearson Type III fitting ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 162, V30,2. See WATSTORE users manual, Appendix.
H30005	204	REAL	1	YES	Annual maximum 30-day mean discharge, in cubic feet per second for 5-year recurrence interval, defined by log-Pearson Type III fitting ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 133, V30,5. See WATSTORE users manual, Appendix.

# **Appendix C.1**

## **Attributes and Their Characteristics--continued**

ATTRIBUTE NAME	INDEX NUMBER	TYPE	LENGTH	UPDATE	DESCRIPTION
H30010	205	REAL	1	YES	Annual maximum 30-day mean discharge, in cubic feet per second for 10-year recurrence interval, defined by log-Pearson Type III fitting ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 163, V30,10. See WATSTORE users manual, Appendix.
H30020	206	REAL	1	YES	Annual maximum 30-day mean discharge, in cubic feet per second for 20-year recurrence interval, defined by log-Pearson Type III fitting ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 134, V30,20. See WATSTORE users manual, Appendix.
H30025	207	REAL	1	YES	Annual maximum 30-day mean discharge, in cubic feet per second for 25-year recurrence interval, defined by log-Pearson Type III fitting ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 164, V30,25. See WATSTORE users manual, Appendix.
H30050	208	REAL	1	YES	Annual maximum 30-day mean discharge, in cubic feet per second for 50-year recurrence interval, defined by log-Pearson Type III fitting ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 165, V30,50. See WATSTORE users manual, Appendix.
H30100	209	REAL	1	YES	Annual maximum 30-day mean discharge, in cubic feet per second for 100-year recurrence interval, defined by log-Pearson Type III fitting ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 135, V30,100. See WATSTORE users manual, Appendix.
HUCODE	4	INT	1	YES	Hydrologic unit code (8 digits). These codes are given in the U.S. Geological Survey map series "State Hydrologic Unit Maps," Open File Report 84-708.
I24-2.	63	REAL	1	YES	Precipitation intensity, 24-hour rainfall, in inches, expected on the average of once each 2 years. Basin and streamflow characteristic no 33, I24,2. See WATSTORE users manual, Appendix.
I24010	99	REAL	1	YES	Precipitation intensity, 24-hour rainfall, in inches, expected on the average once each 10 years. Estimated from U.S. Weather Bureau technical Paper 40 except for western states where NOAA Atlas 2 exists). Basin and streamflow characteristic no 34, I24,10. See WATSTORE users manual, Appendix.
I24025	100	REAL	1	YES	Precipitation intensity, 24-hour rainfall, in inches, expected on the average once each 25 years. Estimated from U.S. Weather Bureau technical Paper 40 except for western states where NOAA Atlas 2 exists). Basin and streamflow

## **Appendix C.1**

### **Attributes and Their Characteristics--continued**

ATTRIBUTE	INDEX					DESCRIPTION
	NAME	NUMBER	TYPE	LENGTH	UPDATE	
characteristic no 35, I24,25. See WATSTORE users manual, Appendix.						
I24050	101	REAL		1	YES	Precipitation intensity, 24-hour rainfall, in inches, expected on the average once each 50 years. Estimated from U.S. Weather Bureau technical Paper 40 except for western states where NOAA Atlas 2 exists). Basin and streamflow characteristic no 36, I24,50. See WATSTORE users manual, Appendix.
I24100	102	REAL		1	YES	Precipitation intensity, 24-hour rainfall, in inches, expected on the average once each 100 years. Estimated from U.S. Weather Bureau technical Paper 40 except for western states where NOAA Atlas 2 exists). Basin and streamflow characteristic no 37, I24,100. See WATSTORE users manual, Appendix.
ISTAID	51	INT		1	YES	Integer identification number, as an integer.
J407BQ	273	REAL		1	YES	Base gage discharge (Bulletin 17B frequency analysis).
J407BY	278	INT		1	YES	Year to begin analysis, used to identify subset of available record (Bulletin 17B frequency analysis).
J407EY	279	INT		1	YES	Year to end analysis, used to identify subset of available record (Bulletin 17B frequency analysis).
J407GS	272	REAL		1	YES	Generalized skew (Bulletin 17B frequency analysis).
J407HO	270	REAL		1	YES	High outlier discharge criterion (Bulletin 17B frequency analysis).
J407HP	277	INT		1	YES	Historic peak option (Bulletin 17B frequency analysis): 1 - include historic peaks 2 - exclude historic peaks
J407LO	269	REAL		1	YES	Low outlier discharge criterion (Bulletin 17B frequency analysis).
J407NR	274	INT		1	YES	Number of historic peaks (Bulletin 17B frequency analysis).
J407SE	275	REAL		1	YES	Root mean square error of generalized skew (Bulletin 17B frequency analysis).
J407SO	271	INT		1	YES	Generalized skew option (Bulletin 17B frequency analysis): -1 - station skew 0 - weighted skew 1 - generalized skew

## Appendix C.1 Attributes and Their Characteristics--continued

ATTRIBUTE NAME	INDEX NUMBER	TYPE	LENGTH	UPDATE	DESCRIPTION
J407UR	276	INT	1	YES	Include urban-regulated peaks (Bulletin 17B frequency analysis): 1 - no 2 - yes
JANAVE	122	REAL	1	YES	Mean monthly temperature for January, in degrees F. From U.S. Weather Bureau, "Climates of States". Basin and streamflow characteristic no 61, JANAV. See WATSTORE users manual, Appendix.
JANMIN	64	REAL	1	YES	Mean minimum January temp, in degrees F. Basin and streamflow characteristic no 60, JANMIN. See WATSTORE users manual, Appendix.
JULAVE	125	REAL	1	YES	Mean monthly temperature for July, in degrees F. From U.S. Weather Bureau, "Climates of States". Basin and streamflow characteristic no 64, JULYAV. See WATSTORE users manual, Appendix.
JULMAX	124	REAL	1	YES	Mean maximum July temperature, in degrees F. From U.S. Weather Bureau, "Climates of States". Basin and streamflow characteristic no 63, JULYMAX. See WATSTORE users manual, Appendix.
KENPLV	284	REAL	1	YES	P-level for Kendahl Tau statistic.
KENSLP	285	REAL	1	YES	Median slope of time-series trend for Kendahl Tau statistic.
KENTAU	283	REAL	1	YES	Kendahl Tau statistic for time-series data.
L01002	156	REAL	1	YES	Annual minimum 1-day mean discharge, in cubic feet per second, for 2-year recurrence interval, defined by log-Pearson Type III fitting in ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 112, M1,2. See WATSTORE users manual, Appendix.
L01010	157	REAL	1	YES	Annual minimum 1-day mean discharge, in cubic feet per second, for 10-year recurrence interval, defined by log-Pearson Type III fitting in ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 113, M1,10. See WATSTORE users manual, Appendix.
L01020	158	REAL	1	YES	Annual minimum 1-day mean discharge, in cubic feet per second, for 20-year recurrence interval, defined by log-Pearson Type III fitting in ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 114, M1,20. See WATSTORE users manual, Appendix.
L03002	159	REAL	1	YES	Annual minimum 3-day mean discharge, in cubic feet per second, for 2-year recurrence interval, defined by log-Pearson Type III fitting in ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 115, M3,2. See WATSTORE users manual, Appendix.

## **Appendix C.1**

### **Attributes and Their Characteristics--continued**

ATTRIBUTE NAME	INDEX NUMBER	TYPE	LENGTH	UPDATE	DESCRIPTION
L03010	160	REAL	1	YES	Annual minimum 3-day mean discharge, in cubic feet per second; for 10-year recurrence interval, defined by log-Pearson Type III fitting in ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 116, M3,10. See WATSTORE users manual, Appendix.
L03020	161	REAL	1	YES	Annual minimum 3-day mean discharge, in cubic feet per second, for 20-year recurrence interval, defined by log-Pearson Type III fitting in ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 117, M3,20. See WATSTORE users manual, Appendix.
L07002	162	REAL	1	YES	Annual minimum 7-day mean discharge, in cubic feet per second, for 2-year recurrence interval, defined by log-Pearson Type III fitting in ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 118, M7,2. See WATSTORE users manual, Appendix.
L07005	163	REAL	1	YES	Annual minimum 7-day mean discharge, in cubic feet per second, for 5-year recurrence interval, defined by log-Pearson Type III fitting in ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 119, M7,5. See WATSTORE users manual, Appendix.
L07010	164	REAL	1	YES	Annual minimum 7-day mean discharge, in cubic feet per second, for 10-year recurrence interval, defined by log-Pearson Type III fitting in ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 120, M7,10. See WATSTORE users manual, Appendix.
L07020	165	REAL	1	YES	Annual minimum 7-day mean discharge, in cubic feet per second, for 20-year recurrence interval, defined by log-Pearson Type III fitting in ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 121, M7,20. See WATSTORE users manual, Appendix.
L14002	166	REAL	1	YES	Annual minimum 14-day mean discharge, in cubic feet per second, for 2-year recurrence interval, defined by log-Pearson Type III fitting in ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 122, M14,2. See WATSTORE users manual, Appendix.
L14010	167	REAL	1	YES	Annual minimum 14-day mean discharge, in cubic feet per second, for 10-year recurrence interval, defined by log-Pearson Type III fitting in ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 123, M14,10. See WATSTORE users manual, Appendix.
L14020	168	REAL	1	YES	Annual minimum 14-day mean discharge, in cubic feet per second, for 20-year recurrence interval, defined by log-Pearson Type III fitting in ANNIE

## **Appendix C.1**

### **Attributes and Their Characteristics--continued**

---

ATTRIBUTE	INDEX				
NAME	NUMBER	TYPE	LENGTH	UPDATE	DESCRIPTION

or WATSTORE program A969. Basin and streamflow characteristic no 124, M14,20. See WATSTORE users manual, Appendix.

## Appendix C.1

### Attributes and Their Characteristics--continued

ATTRIBUTE NAME	INDEX NUMBER	TYPE	LENGTH	UPDATE	DESCRIPTION
L30002	169	REAL	1	YES	Annual minimum 30-day mean discharge, in cubic feet per second, for 2-year recurrence interval, defined by log-Pearson Type III fitting in ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 125, M30,2. See WATSTORE users manual, Appendix.
L30010	170	REAL	1	YES	Annual minimum 30-day mean discharge, in cubic feet per second, for 10-year recurrence interval, defined by log-Pearson Type III fitting in ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 126, M30,10. See WATSTORE users manual, Appendix.
L30020	171	REAL	1	YES	Annual minimum 30-day mean discharge, in cubic feet per second, for 20-year recurrence interval, defined by log-Pearson Type III fitting in ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 127, M30,20. See WATSTORE users manual, Appendix.
L90002	172	REAL	1	YES	Annual minimum 90-day mean discharge, in cubic feet per second, for 2-year recurrence interval, defined by log-Pearson Type III fitting in ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 128, M90,2. See WATSTORE users manual, Appendix.
L90010	173	REAL	1	YES	Annual minimum 90-day mean discharge, in cubic feet per second, for 10-year recurrence interval, defined by log-Pearson Type III fitting in ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 129, M90,10. See WATSTORE users manual, Appendix.
L90020	174	REAL	1	YES	Annual minimum 90-day mean discharge, in cubic feet per second, for 20-year recurrence interval, defined by log-Pearson Type III fitting in ANNIE or WATSTORE program A969. Basin and streamflow characteristic no 130, M90,20. See WATSTORE users manual, Appendix.
LAKE	92	REAL	1	YES	Area of lakes and ponds in percent of contributing drainage area. Measured by the grid sampling method. Basin and streamflow characteristic no 13, LAKE. See WATSTORE users manual, Appendix.
LATCTR	96	REAL	1	YES	Latitude of center of basin, decimal degrees. Basin and streamflow characteristic no 19, LAT. See WATSTORE users manual, Appendix.
LATDEG	8	REAL	1	YES	Latitude in decimal degrees.
LATDMS	54	INT	1	YES	Latitude in degrees, minutes, seconds (dddmmss).
LCODE	18	INT	1	YES	Length units code, user defined.
LENGTH	26	REAL	1	YES	Channel length, units user defined.

# Appendix C.1

## Attributes and Their Characteristics--continued

ATTRIBUTE NAME	INDEX NUMBER	TYPE	LENGTH	UPDATE	DESCRIPTION
LKEVAP	127	REAL	1	YES	Mean annual lake evaporation, in inches. From U.S. Weather Bureau, Technical Paper 37. Basin and streamflow characteristic no 70, EVAP. See WATSTORE users manual, Appendix.
LNGCTR	97	REAL	1	YES	Longitude of center of basin, decimal degrees. Basin and streamflow characteristic no 20, LONG. See WATSTORE users guide, Appendix.
LNGDEG	9	REAL	1	YES	Longitude in decimal degrees.
LNGDMS	55	INT	1	YES	Longitude in degrees, minutes, seconds (dddmmss).
LOESS	94	REAL	1	YES	Depth of surficial loess, in feet. From Soil Conservation Service. Basin and streamflow characteristic no 17, LOESS. See WATSTORE users manual, Appendix.
MARMAX	123	REAL	1	YES	Mean maximum March temperature, in degrees F. From U.S. Weather Bureau, "Climates of States". Basin and streamflow characteristic no 62, MARMAX. See WATSTORE users manual, Appendix.
MAXVAL	13	REAL	1	YES	Maximum value in data set, general use.
MEANND	280	REAL	1	YES	Mean of the logarithms, base 10, of annual n-day high-flow or low-flow statistic.
MEANPK	74	REAL	1	YES	Mean of the logarithms, base 10, of systematic annual peak discharges from Bulletin 17B frequency analysis or WATSTORE program J407. Basin and streamflow characteristic no 83, MEANPK. See WATSTORE users manual, Appendix.
MEANVL	14	REAL	1	YES	Mean of values in data set, general use.
MINVAL	12	REAL	1	YES	Minimum value in data set, general use.
NONZRO	286	INT	1	YES	Number of non-zero values in the time series.
NUMZRO	287	INT	1	YES	Number of zero values in time series.
P1.25	65	REAL	1	YES	Annual flood peak, in cubic feet per second, 1.25-year recurrence interval. Basin and streamflow characteristic no 75, P1.25. See WATSTORE users manual, Appendix.
P10.	68	REAL	1	YES	Annual flood peak, in cubic feet per second, 10-year recurrence interval. Basin and streamflow characteristic no 78, P10. See WATSTORE users manual, Appendix.
P100.	71	REAL	1	YES	Annual flood peak, in cubic feet per second, 100-year recurrence interval. Basin and streamflow characteristic no 81, P100. See WATSTORE users manual, Appendix.

## Appendix C.1

### Attributes and Their Characteristics--continued

ATTRIBUTE NAME	INDEX NUMBER	TYPE	LENGTH	UPDATE	DESCRIPTION
P2.	66	REAL	1	YES	Annual flood peak, in cubic feet per second, 2-year recurrence interval. Basin and streamflow characteristic no 76, P2. See WATSTORE users manual, Appendix.
P200.	72	REAL	1	YES	Annual flood peak, in cubic feet per second, 200-year recurrence interval. Basin and streamflow characteristic no 82, P200. See WATSTORE users manual, Appendix.
P25.	69	REAL	1	YES	Annual flood peak, in cubic feet per second, 25-year recurrence interval. Basin and streamflow characteristic no 79, P25. See WATSTORE users manual, Appendix.
P5.	67	REAL	1	YES	Annual flood peak, in cubic feet per second, 5-year recurrence interval. Basin and streamflow characteristic no 77, P5. See WATSTORE users manual, Appendix.
P50.	70	REAL	1	YES	Annual flood peak, in cubic feet per second, 50-year recurrence interval. Basin and streamflow characteristic no 80, P50. See WATSTORE users manual, Appendix.
P500.	73	REAL	1	YES	Annual flood peak, in cubic feet per second, 500-year recurrence interval. Basin and streamflow characteristic no 178, P500. See WATSTORE users manual, Appendix.
PARMCD	56	INT	1	YES	Parameter code, see WATSTORE users manual, Appendix D.
PNEVAP	128	REAL	1	YES	Mean annual Class A pan evaporation, in inches. From U.S. Weather Bureau, Technical Paper 37. Basin and streamflow characteristic no 71, EVAPAN. See WATSTORE users manual, Appendix.
PRCAPR	109	REAL	1	YES	April mean monthly precipitation, in inches. Basin and streamflow characteristic no 47, PR4. See WATSTORE users manual, Appendix.
PRCAUG	113	REAL	1	YES	August mean monthly precipitation, in inches. Basin and streamflow characteristic no 51, PRC8. See WATSTORE users manual, Appendix.
PRCDEC	105	REAL	1	YES	December mean monthly precipitation, in inches. Basin and streamflow characteristic no 43, PRC12. See WATSTORE users manual, Appendix.
PRCFEB	107	REAL	1	YES	February mean monthly precipitation, in inches. Basin and streamflow characteristic no 45, PRC2. See WATSTORE users manual, Appendix.
PRCJAN	106	REAL	1	YES	January mean monthly precipitation, in inches. Basin and streamflow characteristic no 44, PRC12. See WATSTORE users manual, Appendix.

# Appendix C.1

## Attributes and Their Characteristics--continued

ATTRIBUTE NAME	INDEX NUMBER	TYPE	LENGTH	UPDATE	DESCRIPTION
PRCJUL	112	REAL	1	YES	July mean monthly precipitation, in inches. Basin and streamflow characteristic no 50, PRC7. See WATSTORE users manual, Appendix.
PRCJUN	111	REAL	1	YES	June mean monthly precipitation, in inches Basin and streamflow characteristic no 49, PRC6. See WATSTORE users manual, Appendix.
PRCMAR	108	REAL	1	YES	March mean monthly precipitation, in inches Basin and streamflow characteristic no 46, PR3. See WATSTORE users manual, Appendix.
PRCMAY	110	REAL	1	YES	May mean monthly precipitation, in inches Basin and streamflow characteristic no 48, PRC5. See WATSTORE users manual, Appendix.
PRCNOV	104	REAL	1	YES	November mean monthly precipitation, in inches. Basin and streamflow characteristic no 42, PRC11. See WATSTORE users manual, Appendix.
PRCOCT	103	REAL	1	YES	October mean monthly precipitation, in inches. Basin and streamflow characteristic no 41, PRC10. See WATSTORE users manual, Appendix.
PRCSEP	114	REAL	1	YES	September mean monthly precipitation, in inches Basin and streamflow characteristic no 52, PRC9. See WATSTORE users manual, Appendix.
PRECIP	58	REAL	1	YES	Mean annual precipitation, in inches, from U.S. Weather Bureau Series "Climates of States;" grid sampling methods used if isohyetal map is available, otherwise anomaly map constructed (Water-Supply Paper 1580-d). Basin and streamflow characteristics no 32, PRECIP. See WATSTORE users manual, Appendix.
QANN	130	REAL	1	YES	Mean annual discharge, in cubic feet per second, from WATSTORE flow variability program W4422. Basin and streamflow characteristic no 86, QA. See WATSTORE users manual, Appendix.
QAPR	138	REAL	1	YES	Mean discharge, in cubic feet per second, for April, from WATSTORE flow variability program W4422. Basin and streamflow characteristic no 94, Q4. See WATSTORE users manual, Appendix.
QAUG	142	REAL	1	YES	Mean discharge, in cubic feet per second, for August, from WATSTORE flow variability program W4422. Basin and streamflow characteristic no 98, Q8. See WATSTORE users manual, Appendix.
QDEC	134	REAL	1	YES	Mean discharge, in cubic feet per second, for December, from WATSTORE flow variability program W4422. Basin and streamflow characteristic no 90, Q12. See WATSTORE users manual, Appendix.
QEX10P	217	REAL	1	YES	Discharge, in cubic feet per second, exceeded 10 percent of time. Defined by daily flow duration, WATSTORE program A969. Basin and

## **Appendix C.1**

### **Attributes and Their Characteristics--continued**

<b>ATTRIBUTE NAME</b>	<b>INDEX NUMBER</b>	<b>TYPE</b>	<b>LENGTH</b>	<b>UPDATE</b>	<b>DESCRIPTION</b>
					streamflow characteristic no 177, D10. See WATSTORE users manual, Appendix.
QEX25P	216	REAL	1	YES	Discharge, in cubic feet per second, exceeded 25 percent of time. Defined by daily flow duration, WATSTORE program A969. Basin and streamflow characteristic no 176, D25. See WATSTORE users manual, Appendix.
QEX50P	215	REAL	1	YES	Discharge, in cubic feet per second, exceeded 50 percent of time. Defined by daily flow duration, WATSTORE program A969. Basin and streamflow characteristic no 175, D50. See WATSTORE users manual, Appendix.
QEX70P	214	REAL	1	YES	Discharge, in cubic feet per second, exceeded 70 percent of time. Defined by daily flow duration, WATSTORE program A969. Basin and streamflow characteristic no 174, D70. See WATSTORE users manual, Appendix.
QEX75P	213	REAL	1	YES	Discharge, in cubic feet per second, exceeded 75 percent of time. Defined by daily flow duration, WATSTORE program A969. Basin and streamflow characteristic no 173, D75. See WATSTORE users manual, Appendix.
QEX90P	212	REAL	1	YES	Discharge, in cubic feet per second, exceeded 90 percent of time. Defined by daily flow duration, WATSTORE program A969. Basin and streamflow characteristic no 172, D90. See WATSTORE users manual, Appendix.
QEX95P	211	REAL	1	YES	Discharge, in cubic feet per second, exceeded 95 percent of time. Defined by daily flow duration, WATSTORE program A969. Basin and streamflow characteristic no 171, D95. See WATSTORE users manual, Appendix.
QFEB	136	REAL	1	YES	Mean discharge, in cubic feet per second, for February, from WATSTORE flow variability program W4422. Basin and streamflow characteristic no 92, Q2. See WATSTORE users manual, Appendix.
QJAN	135	REAL	1	YES	Mean discharge, in cubic feet per second, for January, from WATSTORE flow variability program W4422. Basin and streamflow characteristic no 91, Q1. See WATSTORE users manual, Appendix.
QJUL	141	REAL	1	YES	Mean discharge, in cubic feet per second, for July, from WATSTORE flow variability program W4422. Basin and streamflow characteristic no 97, Q7. See WATSTORE users manual, Appendix.
QJUN	140	REAL	1	YES	Mean discharge, in cubic feet per second, for June, from WATSTORE flow variability program W4422. Basin and streamflow characteristic no 96, Q6. See WATSTORE users manual, Appendix.

## Appendix C.1

### Attributes and Their Characteristics--continued

ATTRIBUTE INDEX					
NAME	NUMBER	TYPE	LENGTH	UPDATE	DESCRIPTION
QMAR	137	REAL	1	YES	Mean discharge, in cubic feet per second, for March, from WATSTORE flow variability program W4422. Basin and streamflow characteristic no 93, Q3. See WATSTORE users manual, Appendix.
QMAY	139	REAL	1	YES	Mean discharge, in cubic feet per second, for May, from WATSTORE flow variability program W4422. Basin and streamflow characteristic no 95, Q5. See WATSTORE users manual, Appendix.
QNOV	133	REAL	1	YES	Mean discharge, in cubic feet per second, for November, from WATSTORE flow variability program W4422. Basin and streamflow characteristic no 89, Q11. See WATSTORE users manual, Appendix.
QOCT	132	REAL	1	YES	Mean discharge, in cubic feet per second, for October, from WATSTORE flow variability program W4422. Basin and streamflow characteristic no 88, Q10. See WATSTORE users manual, Appendix.
QSDANN	131	REAL	1	YES	Standard deviation of mean annual discharge, in cubic feet per second, from WATSTORE flow variability program W4422. Basin and streamflow characteristic no 87, QSDANN. See WATSTORE users manual, Appendix.
QSDAPR	150	REAL	1	YES	Standard deviation, in cubic feet per second, of mean discharge for April. From flow variability computer program no. W4422. Basin and streamflow characteristic no 106, SDQ4. See WATSTORE users manual, Appendix.
QSDAUG	154	REAL	1	YES	Standard deviation, in cubic feet per second, of mean discharge for August. From flow variability computer program no. W4422. Basin and streamflow characteristic no 110, SDQ8. See WATSTORE users manual, Appendix.
QSDDEC	146	REAL	1	YES	Standard deviation, in cubic feet per second, of mean discharge for December. From flow variability computer program no. W4422. Basin and streamflow characteristic no 102, SDQ12. See WATSTORE users manual, Appendix.
QSDFEB	148	REAL	1	YES	Standard deviation, in cubic feet per second, of mean discharge for February. From flow variability computer program no. W4422. Basin and streamflow characteristic no 104, SDQ2. See WATSTORE users manual, Appendix.
QSDJAN	147	REAL	1	YES	Standard deviation, in cubic feet per second, of mean discharge for January. From flow variability computer program no. W4422. Basin and streamflow characteristic no 103, SDQ1. See WATSTORE users manual, Appendix.
QSDJUL	153	REAL	1	YES	Standard deviation, in cubic feet per second, of mean discharge for July. From flow variability computer program no. W4422. Basin and streamflow

## **Appendix C.1**

### **Attributes and Their Characteristics--continued**

---

<u>ATTRIBUTE</u>	<u>INDEX</u>				
<u>NAME</u>	<u>NUMBER</u>	<u>TYPE</u>	<u>LENGTH</u>	<u>UPDATE</u>	<u>DESCRIPTION</u>
					characteristic no 109, SDQ7. See WATSTORE users manual, Appendix.
QSDJUN	152	REAL	1	YES	Standard deviation, in cubic feet per second, of mean discharge for June. From flow variability computer program no. W4422. Basin and streamflow characteristic no 108, SDQ6. See WATSTORE users manual, Appendix.

## Appendix C.1 Attributes and Their Characteristics--continued

ATTRIBUTE NAME	INDEX NUMBER	TYPE	LENGTH	UPDATE	DESCRIPTION
QSDMAR	149	REAL	1	YES	Standard deviation, in cubic feet per second, of mean discharge for March. From flow variability computer program no. W4422. Basin and streamflow characteristic no 105, SDQ3. See WATSTORE users manual, Appendix.
QSDMAY	151	REAL	1	YES	Standard deviation, in cubic feet per second, of mean discharge for May. From flow variability computer program no. W4422. Basin and streamflow characteristic no 107, SDQ5. See WATSTORE users manual, Appendix.
QSDNOV	145	REAL	1	YES	Standard deviation, in cubic feet per second, of mean discharge for November. From flow variability computer program no. W4422. Basin and streamflow characteristic no 101, SDQ11. See WATSTORE users manual, Appendix.
QSDOCT	144	REAL	1	YES	Standard deviation, in cubic feet per second, of mean discharge for October. From flow variability computer program no. W4422. Basin and streamflow characteristic no 100, SDQ10. See WATSTORE users manual, Appendix.
QSDSEP	155	REAL	1	YES	Standard deviation, in cubic feet per second, of mean discharge for September. From flow variability computer program no. W4422. Basin and streamflow characteristic no 111, SDQ9. See WATSTORE users manual, Appendix.
QSEP	143	REAL	1	YES	Mean discharge, in cubic feet per second, for September, from WATSTORE flow variability program W4422. Basin and streamflow characteristic no 99, Q9. See WATSTORE users manual, Appendix.
RFOOT	260	REAL	1	YES	Distance from mouth of river, in feet.
RMILE	25	REAL	1	YES	Distance from basin outlet, in miles.
RWFLAG	35	INT	1	YES	Data Read/Write flag: 0 - read and write, default 1 - read only ***** probably not implemented *****
SDND	281	REAL	1	YES	Standard deviation of logarithms, base 10, of annual n-day high-flow or low-flow statistic.
SDPK	75	REAL	1	YES	Standard deviation of logarithms, base 10, of systematic annual peak discharges, from Bulletin 17B frequency analysis or WATSTORE program J407.. Basin and streamflow characteristic no 84 SDPK. See WATSTORE users manual, Appendix.
SEASBG	256	INT	1	YES	Beginning month of a user defined season. Will start on first day of the month. Used with attribute SEASND to define a specific time period, usually a year. January is month 1 and December is month 12.

## Appendix C.1 Attributes and Their Characteristics--continued

ATTRIBUTE NAME	INDEX NUMBER	TYPE	LENGTH	UPDATE	DESCRIPTION
SEASND	257	INT	1	YES	Ending month of a user defined season. Will end on last day of the month. Used with attribute SEASBG to define a specific time period, usually a year. January is month 1 and December is month 12.
SITECO	44	CHAR	4	YES	Site code, see WATSTORE users manual, volume 1, chapter 3. SW - stream SP - spring ES - estuary GW - well LK - lake or reservoir ME - meteorological
SKEWCF	16	REAL	1	YES	Skew coefficient of values in data set, general use.
SKWND	282	REAL	1	YES	Skew of logarithms, base 10, of annual n-day high-flow or low-flow statistic.
SKWPK	76	REAL	1	YES	Skew of logarithms, base 10, of systematic annual peak discharges, from Bulletin 17B frequency analysis or WATSTORE program J407. Basin and streamflow characteristic no 85, SKEWPK. See WATSTORE users manual, Appendix.
SLOPE	24	REAL	1	YES	Slope, units are user defined.
SN002	118	REAL	1	YES	Maximum water equivalent, in inches, of snow cover as of March 15, 2-year recurrence interval. From U.S. Weather Bureau, Technical Paper 50. Basin and streamflow characteristic no 56, SN2. See WATSTORE users manual, Appendix.
SN010	119	REAL	1	YES	Maximum water equivalent, in inches, of snow cover as of March 15, 10-year recurrence interval. From U.S. Weather Bureau, Technical Paper 50. Basin and streamflow characteristic no 57, SN10. See WATSTORE users manual, Appendix.
SN025	120	REAL	1	YES	Maximum water equivalent, in inches, of snow cover as of March 15, 25-year recurrence interval. From U.S. Weather Bureau, Technical Paper 50. Basin and streamflow characteristic no 58, SN25. See WATSTORE users manual, Appendix.
SN100	121	REAL	1	YES	Maximum water equivalent, in inches, of snow cover as of March 15, 100-year recurrence interval. From U.S. Weather Bureau, Technical Paper 50. Basin and streamflow characteristic no 59, SN100. See WATSTORE users manual, Appendix.
SNOAPR	117	REAL	1	YES	Mean water equivalent, in inches, of snow cover as of April 30. From U. S. Weather Bureau, Technical Paper 50. Basin and streamflow characteristic no 55, SNOAPR. See WATSTORE users manual, Appendix.

# Appendix C.1

## Attributes and Their Characteristics--continued

ATTRIBUTE NAME	INDEX NUMBER	TYPE	LENGTH	UPDATE	DESCRIPTION
SNOFAL	115	REAL	1	YES	Mean annual snowfall, in inches. From U. S. Weather Bureau, "Climates of States". Basin and streamflow characteristic no 53, SNOFALL. See WATSTORE users manual, Appendix.
SNOMAR	116	REAL	1	YES	Mean water equivalent, in inches, of snow cover as of March 1. From U. S. Weather Bureau, Technical Paper 50. Basin and streamflow characteristic no 54, SNOMAR. See WATSTORE users manual, Appendix.
SOILIN	62	REAL	1	YES	Soils index, in inches, a relative measure of potential infiltration (soil water storage), from Soil Conservation Service. Basin and streamflow characteristic no 16, SOIL INF.
STAID	2	CHAR	16	YES	Alpha-numeric station id.
STANAM	45	CHAR	48	YES	Short name or description of the data set.
STATCD	57	INT	1	YES	Statistics code, see WATSTORE users manual, Appendix E.
STCODE	3	CHAR	4	YES	Standard 2-character post office abbreviation, includes DC - Washington, District of Columbia PR - Puerto Rico VI - Virgin Islands GU - Guam PI - Pacific Trust Territories Use NON for no state abbreviation.
STDDEV	15	REAL	1	YES	Standard deviation of values in data set, general use.
STDIMX	266	INT	1	NO	Space time dimension in X direction.
STDIMY	267	INT	1	NO	Space time dimension in Y direction.
STDIMZ	268	INT	1	NO	Space time dimension in Z direction.
STDTYP	265	CHAR	4	NO	Type of space time data: ***** ***** *****
STFIPS	41	INT	1	YES	State FIPS code, see WATSTORE users manual, Appendix B.
STORAG	59	REAL	1	YES	Area of lakes, ponds, and swamps in percent of contributing drainage area, measured by the grid sampling methods. Basin and streamflow characteristics no 12, STORAGE. See WATSTORE users manual, Appendix.
SUBHUC	5	INT	1	YES	Extension to hydrologic unit code (HUCODE). See the U.S. Geological Survey map series "State Hydrologic unit maps," Open File Report 84-708.

## Appendix C.1 Attributes and Their Characteristics--continued

ATTRIBUTE NAME	INDEX NUMBER	TYPE	LENGTH	UPDATE	DESCRIPTION								
TCODE	17	INT	1	YES	<p>Time units code.</p> <table style="margin-left: 20px;"> <tr><td>1 - seconds</td><td>4 - days</td></tr> <tr><td>2 - minutes</td><td>5 - months</td></tr> <tr><td>3 - hours</td><td>6 - years</td></tr> </table> <p>Used in combination with TSSTEP.</p>	1 - seconds	4 - days	2 - minutes	5 - months	3 - hours	6 - years		
1 - seconds	4 - days												
2 - minutes	5 - months												
3 - hours	6 - years												
TGROUP	34	INT	1	NO	<p>Unit for group pointers, depending on the time step of the data, may effect the speed of data retrievals. The default group pointer is 6 (years). For timeseries data with a timestep of an hour or less, monthly or even daily group pointers may be more efficient.</p> <table style="margin-left: 20px;"> <tr><td>3 - hours</td><td>6 - years</td></tr> <tr><td>4 - days</td><td>7 - centuries</td></tr> <tr><td>5 - months</td><td></td></tr> </table>	3 - hours	6 - years	4 - days	7 - centuries	5 - months			
3 - hours	6 - years												
4 - days	7 - centuries												
5 - months													
TMTOPK	98	REAL	1	YES	<p>Time, in hours, measured as time difference between center of mass of total rainfall and peak discharge. Basin and streamflow characteristic no 21, TIMETOPK. See WATSTORE users manual, Appendix.</p>								
TMZONE	262	INT	1	YES	<p>Time zone. Each time zone is represented as the number of hours to be added to, or subtracted from, Greenwich time:</p> <table style="margin-left: 20px;"> <tr><td>-4 - Atlantic Standard</td><td>-8 - Pacific Standard</td></tr> <tr><td>-5 - Eastern Standard</td><td>-9 - Yukon Standard</td></tr> <tr><td>-6 - Central Standard</td><td>-10 - Alaska-Hawaii Standard</td></tr> <tr><td>-7 - Mountain Standard</td><td>Standard</td></tr> </table>	-4 - Atlantic Standard	-8 - Pacific Standard	-5 - Eastern Standard	-9 - Yukon Standard	-6 - Central Standard	-10 - Alaska-Hawaii Standard	-7 - Mountain Standard	Standard
-4 - Atlantic Standard	-8 - Pacific Standard												
-5 - Eastern Standard	-9 - Yukon Standard												
-6 - Central Standard	-10 - Alaska-Hawaii Standard												
-7 - Mountain Standard	Standard												
TOLR	36	REAL	1	YES	<p>Data compression tolerance. Data values within +/- of TOLR will be considered the same value and compressed in the data set. Once data has been compressed, the original values can not be retrieved.</p>								
TSBDY	29	INT	1	NO	<p>Starting day for time-series data in a data set. Defaults to day 1.</p>								
TSBHR	30	INT	1	NO	<p>Starting hour for time-series data in a data set. Defaults to hour 1.</p>								
TSBMO	28	INT	1	NO	<p>Starting month for time-series data in a data set. Defaults to month 1 (January).</p>								
TSBYR	27	INT	1	NO	<p>Starting year for time-series data in a data set. Defaults to year 1900.</p>								
TSFILL	32	REAL	1	NO	<p>Time-series filler value. This value will be used for missing values. The default is 0.0.</p>								
TSFORM	84	INT	1	NO	<p>Form of data</p> <table style="margin-left: 20px;"> <tr><td>1 - mean over the timestep (default)</td></tr> <tr><td>2 - total over the timestep</td></tr> <tr><td>3 - instantaneous @ time (end of timestep)</td></tr> <tr><td>4 - minimum over the timestep</td></tr> <tr><td>5 - maximum over the timestep</td></tr> </table>	1 - mean over the timestep (default)	2 - total over the timestep	3 - instantaneous @ time (end of timestep)	4 - minimum over the timestep	5 - maximum over the timestep			
1 - mean over the timestep (default)													
2 - total over the timestep													
3 - instantaneous @ time (end of timestep)													
4 - minimum over the timestep													
5 - maximum over the timestep													

## Appendix C.1 Attributes and Their Characteristics--continued

ATTRIBUTE NAME	INDEX NUMBER	TYPE	LENGTH	UPDATE	DESCRIPTION
TSPREC	31	INT	1	NO	New group, new record flag: 0 - start new group at the end of the last group (default) 1 - start new group at the beginning of a record.
TSPTAD	60	INT	1	YES	Timeseries put aggregation/disaggregation code.
TSSTEP	33	INT	1	NO	Time step, in TCODE units (used in combination with TCODE).
TSTYPE	1	CHAR	4	YES	User-defined four-character descriptor. Used to describe the contents of the data set, for example: PRCP, RAIN, SNOW - Precipitation FLOW, DISC, PEAK - discharge TEMP, TMIN, TMAX - temperature EVAP, PET - evapotranspiration Some models and application programs may require a specific TSTYPE for data sets they use.
UBC024	220	REAL	1	YES	Defined by user or application. Basin and streamflow characteristic no 24 See WATSTORE users manual, Appendix.
UBC025	221	REAL	1	YES	Defined by user or application. Basin and streamflow characteristic no 25 See WATSTORE users manual, Appendix.
UBC026	222	REAL	1	YES	Defined by user or application. Basin and streamflow characteristic no 26 See WATSTORE users manual, Appendix.
UBC027	223	REAL	1	YES	Defined by user or application. Basin and streamflow characteristic no 27 See WATSTORE users manual, Appendix.
UBC028	224	REAL	1	YES	Defined by user or application. Basin and streamflow characteristic no 28 See WATSTORE users manual, Appendix.
UBC029	225	REAL	1	YES	Defined by user or application. Basin and streamflow characteristic no 29 See WATSTORE users manual, Appendix.
UBC030	226	REAL	1	YES	Defined by user or application. Basin and streamflow characteristic no 30 See WATSTORE users manual, Appendix.
UBC031	227	REAL	1	YES	Defined by user or application. Basin and streamflow characteristic no 31 See WATSTORE users manual, Appendix.
UBC038	228	REAL	1	YES	Defined by user or application. Basin and streamflow characteristic no 38 See WATSTORE users manual, Appendix.

## Appendix C.1 Attributes and Their Characteristics--continued

ATTRIBUTE NAME	INDEX NUMBER	TYPE	LENGTH	UPDATE	DESCRIPTION
UBC039	229	REAL	1	YES	Defined by user or application. Basin and streamflow characteristic no 39 See WATSTORE users manual, Appendix.
UBC040	230	REAL	1	YES	Defined by user or application. Basin and streamflow characteristic no 040 See WATSTORE users manual, Appendix.
UBC066	231	REAL	1	YES	Defined by user or application. Basin and streamflow characteristic no 66 See WATSTORE users manual, Appendix.
UBC067	232	REAL	1	YES	Defined by user or application. Basin and streamflow characteristic no 67 See WATSTORE users manual, Appendix.
UBC068	233	REAL	1	YES	Defined by user or application. Basin and streamflow characteristic no 68 See WATSTORE users manual, Appendix.
UBC069	234	REAL	1	YES	Defined by user or application. Basin and streamflow characteristic no 69 See WATSTORE users manual, Appendix.
UBC073	235	REAL	1	YES	Defined by user or application. Basin and streamflow characteristic no 73 See WATSTORE users manual, Appendix.
UBC074	236	REAL	1	YES	Defined by user or application. Basin and streamflow characteristic no 74 See WATSTORE users manual, Appendix.
UBC166	237	REAL	1	YES	Defined by user or application. Basin and streamflow characteristic no 166 See WATSTORE users manual, Appendix.
UBC167	238	REAL	1	YES	Defined by user or application. Basin and streamflow characteristic no 167 See WATSTORE users manual, Appendix.
UBC169	239	REAL	1	YES	Defined by user or application. Basin and streamflow characteristic no 168 See WATSTORE users manual, Appendix.
UBC170	240	REAL	1	YES	Defined by user or application. Basin and streamflow characteristic no 170 See WATSTORE users manual, Appendix.
UBC182	241	REAL	1	YES	Defined by user or application. Basin and streamflow characteristic no 182 See WATSTORE users manual, Appendix.
UBC183	242	REAL	1	YES	Defined by user or application. Basin and streamflow characteristic no 183 See WATSTORE users manual, Appendix.
UBC184	243	REAL	1	YES	Defined by user or application. Basin and streamflow characteristic no 184 See WATSTORE users manual, Appendix.

## Appendix C.1 Attributes and Their Characteristics--continued

ATTRIBUTE NAME	INDEX NUMBER	INDEX TYPE	LENGTH	UPDATE	DESCRIPTION
UBC185	244	REAL	1	YES	Defined by user or application. Basin and streamflow characteristic no 185 See WATSTORE users manual, Appendix.
UBC186	245	REAL	1	YES	Defined by user or application. Basin and streamflow characteristic no 186 See WATSTORE users manual, Appendix.
UBC187	246	REAL	1	YES	Defined by user or application. Basin and streamflow characteristic no 187 See WATSTORE users manual, Appendix.
UBC188	247	REAL	1	YES	Defined by user or application. Basin and streamflow characteristic no 188 See WATSTORE users manual, Appendix.
UBC189	248	REAL	1	YES	Defined by user or application. Basin and streamflow characteristic no 189 See WATSTORE users manual, Appendix.
UBC190	249	REAL	1	YES	Defined by user or application. Basin and streamflow characteristic no 190 See WATSTORE users manual, Appendix.
UBC191	250	REAL	1	YES	Defined by user or application. Basin and streamflow characteristic no 191 See WATSTORE users manual, Appendix.
UBC192	251	REAL	1	YES	Defined by user or application. Basin and streamflow characteristic no 192 See WATSTORE users guide, Appendix.
UBC193	252	REAL	1	YES	Defined by user or application. Basin and streamflow characteristic no 193 See WATSTORE users guide, Appendix.
UBC194	253	REAL	1	YES	Defined by user or application. Basin and streamflow characteristic no 194 See WATSTORE users guide, Appendix.
UBC195	254	REAL	1	YES	Defined by user or application. Basin and streamflow characteristic no 195 See WATSTORE users guide, Appendix.
UBC200	255	REAL	1	YES	Defined by user or application. Basin and streamflow characteristic no 200 See WATSTORE users guide, Appendix.
VALLGH	88	REAL	1	YES	Valley length, in miles, measured along general path of flood plain from gage to basin divide. Basin and streamflow characteristic no 7, VALLGH. See WATSTORE users manual, Appendix.
VBTIME	85	INT	1	NO	Variable time-step option for the data set 1 - all data are at the same time step 2 - time step may vary (default)
VCODE	20	INT	1	YES	Volume units code, user defined.

## Appendix C.1

### Attributes and Their Characteristics--continued

ATTRIBUTE NAME	INDEX NUMBER	TYPE	LENGTH	UPDATE	DESCRIPTION
VLCODE	21	INT	1	YES	Velocity units code, user defined.
WELLDP	47	REAL	1	YES	Depth of well, in feet. The greatest depth at which water can enter the well. See WATSTORE users manual, volume 1, chapter 3.
WEMAR2	126	REAL	1	YES	Water equivalent, in inches, of snow cover as of the first week in March, 2-year recurrence interval. Data compiled by the New Your District USGS. Basin and streamflow characteristic no 65, WE MAR2. See WATSTORE users manual, Appendix.
WRCMN	78	REAL	1	YES	WRC mean of logarithms, base 10, of annual peak discharge after outlier and historic-peak adjustments, from Bulletin 17B frequency analysis or WATSTORE program J407. Basin and streamflow characteristic no 180, WRC MEAN. See WATSTORE users manual, Appendix.
WRCSD	79	REAL	1	YES	WRC standard deviation of logarithms, base 10, of annual peak discharge after outlier and historic-peak adjustments, from Bulletin 17B frequency analysis or WATSTORE program J407. Basin and streamflow characteristics no 181, WRC SD. See WATSTORE users manual, Appendix.
WRCSKW	77	REAL	1	YES	WRC skew of logarithms, base 10, of annual peak discharge after outlier and historic-peak adjustments and generalized skew weighting, from Bulletin 17B grequency analysis or WATSTORE program J407. Basin and streamflow characteristic no 179, WRC SKEW. See WATSTORE users manual, Appendix.
XSECLC	258	REAL	1	YES	Cross-section locator, distance in feet from left bank (as determined by facing downstream).
YRSDAY	218	REAL	1	YES	Number of years of daily-flow record, from WATSTORE flow variability program W4422. Basin and streamflow characteristic no 198, YRSDAY. See WATSTORE users manual, Appendix.
YRSHPK	81	REAL	1	YES	Number of consecutive years used for historic-peak adjustment to flood-frequency data used in Bulletin 17B frequency analysis or WATSTORE program J407. Basin and streamflow characteristic no 197, YRSHISPK. See WATSTORE users manual, Appendix.
YRSLOW	219	REAL	1	YES	Number of years of low-flow record. Basin and streamflow characteristic no 199, YRSLOW. See WATSTORE users manual, Appendix.
YRSPK	80	REAL	1	YES	Number of years of systematic peak flow record, used in Bulletin 17B frequency analysis or WATSTORE program J407. Basin and streamflow characteristic no 196, YRSPK. See WATSTORE users manual, Appendix.

# APPENDIX C.2

## Attribute names Sorted by Index Number

1	TSTYPE	64	JANMIN	127	LKEVAP	190	H07005
2	STAID	65	P1.25	128	PNEVAP	191	H07010
3	STCODE	66	P2.	129	FROST	192	H07020
4	HUCODE	67	P5.	130	QANN	193	H07025
5	SUBHUC	68	P10.	131	QSDANN	194	H07050
6	COCODE	69	P25.	132	QOCT	195	H07100
7	ELEV	70	P50.	133	QNOV	196	H15002
8	LATDEG	71	P100.	134	QDEC	197	H15005
9	LNGDEG	72	P200.	135	QJAN	198	H15010
10	DESCRP	73	P500.	136	QEHB	199	H15020
11	DAREA	74	MEANPK	137	QMAR	200	H15025
12	MINVAL	75	SDPK	138	QAPR	201	H15050
13	MAXVAL	76	SKWPK	139	QMAY	202	H15100
14	MEANVL	77	WRCSKW	140	QJUN	203	H30002
15	STDDEV	78	WRCMN	141	QJUL	204	H30005
16	SKEWCF	79	WRCSD	142	QAUG	205	H30010
17	TCODE	80	YRSPK	143	QSEP	206	H30020
18	LCODE	81	YRSHPK	144	QSDOCT	207	H30025
19	ACODE	82	CHEAT	145	QSDNOV	208	H30050
20	VCODE	83	COMPPG	146	QSDDEC	209	H30100
21	VLCODE	84	TSFORM	147	QSDJAN	210	DEPH25
22	DCODE	85	VBTIME	148	QSDFEB	211	QEX95P
23	GCODE	86	BSLOPE	149	QSDMAR	212	QEX90P
24	SLOPE	87	BLNGTH	150	QSDAPR	213	QEX75P
25	RMILE	88	VALLGH	151	QSDMAY	214	QEX70P
26	LENGTH	89	EL1085	152	QSDJUN	215	QEX50P
27	TSBYR	90	EL5000	153	QSDJUL	216	QEX25P
28	TSBMO	91	EL6000	154	QSDAUG	217	QEX10P
29	TSBDY	92	LAKE	155	QSDSEP	218	YRSDAY
30	TSBHR	93	GLACER	156	L01002	219	YRSLOW
31	TSPREC	94	LOESS	157	L01010	220	UBC024
32	TSFILL	95	AZMUTH	158	L01020	221	UBC025
33	TSSTEP	96	LATCTR	159	L03002	222	UBC026
34	TGROUP	97	LNGCTR	160	L03010	223	UBC027
35	RWFLAG	98	TMTOPK	161	L03020	224	UBC028
36	TOLR	99	I24010	162	L07002	225	UBC029
37	HELP	100	I24025	163	L07005	226	UBC030
38	DONE	101	I24050	164	L07010	227	UBC031
39	ALL	102	I24100	165	L07020	228	UBC038
40	AGENCY	103	PRCOCT	166	L14002	229	UBC039
41	STFIPS	104	PRCNOV	167	L14010	230	UBC040
42	DSCODE	105	PRCDEC	168	L14020	231	UBC066
43	CONTDA	106	PRCJAN	169	L30002	232	UBC067
44	SITECO	107	PRCFEB	170	L30010	233	UBC068
45	STANAM	108	PRCMAR	171	L30020	234	UBC069
46	GUCODE	109	PRCAPR	172	L90002	235	UBC073
47	WELLDP	110	PRCMAY	173	L90010	236	UBC074
48	AQTYPE	111	PRCJUN	174	L90020	237	UBC166
49	BASEQ	112	PRCJUL	175	H01002	238	UBC167
50	DATE	113	PRCAUG	176	H01005	239	UBC169
51	ISTAID	114	PRCSEP	177	H01010	240	UBC170
52	START	115	SNOFAL	178	H01020	241	UBC182
53	END	116	SNOMAR	179	H01025	242	UBC183
54	LATDMS	117	SNOAPR	180	H01050	243	UBC184
55	LNGDMS	118	SN002	181	H01100	244	UBC185
56	PARMCD	119	SN010	182	H03002	245	UBC186
57	STATCD	120	SN025	183	H03005	246	UBC187
58	PRECIP	121	SN100	184	H03010	247	UBC188
59	STORAG	122	JANAVE	185	H03020	248	UBC189
60	TSPTAD	123	MARMAX	186	H03025	249	UBC190
61	FOREST	124	JULMAX	187	H03050	250	UBC191
62	SOILIN	125	JULAVE	188	H03100	251	UBC192
63	I24-2.	126	WEMAR2	189	H07002	252	UBC193

---

## **APPENDIX C.2**

### **Attribute names Sorted by Index Number**

---

253 UBC194  
254 UBC195  
255 UBC200  
256 SEASBG  
257 SEASND  
258 XSECLC  
259 DEPTH  
260 RFOOT  
261 BRANCH  
262 TMZONE  
263 GRPNAM  
264 DATUM  
265 STDTYP  
266 STDIMX  
267 STDIMY  
268 STDIMZ  
269 J407LO  
270 J407HO  
271 J407SO  
272 J407GS  
273 J407BQ  
274 J407NH  
275 J407SE  
276 J407UR  
277 J407HP  
278 J407BY  
279 J407EY  
280 MEANND  
281 SDND  
282 SKWND  
283 KENTAU  
284 KENPLV  
285 KENSLP  
286 NONZRO  
287 NUMZRO

## Appendix C.3 Required Attributes by Data Set Type

### TIMESERIES data sets

TSTYPE - 1  
TCODE - 17  
TSBYR - 27  
TSSTEP - 33  
TGROUP - 34  
STANAM - 45  
TSFORM - 84  
VBTIME - 85

### TABLE data sets

(none)

### VECTOR data sets

TSPREC - 31

### SPACE-TIME data sets

TSPREC - 31  
STDYP - 265  
STDIMX - 266

### MESSAGE data sets

GRPNAM - 263

## **APPENDIX D. DESCRIPTION OF REFERENCED SUBROUTINES**

The WDM toolkit is designed to allow the application programmer to utilize the capabilities of over 200 utilities while dealing directly with approximately 50 "lead" subroutines. In Section 5 the lead subroutines are identified and their functions are explained. This appendix provides additional programmer-oriented information including: the purpose of each module, arguments and their definitions, use of common blocks, routines called by the module, and routines that call the module.

In addition to the lead subroutines, subordinate subroutines contained in WDM toolkit which have been used in the case study in Section 4 are also documented in this appendix to allow better understanding of how WDM can implement the data management functions which have been illustrated. Parallel documentation for all of the approximately 200 subordinate routines contained in the WDM toolkit is available in a file called SYSDOC.OUT which is included with the WDM distribution disks.

### **COPYI**

### **COPYI**

This SUBROUTINE is number 5 in file UTNUMB.

Copy the integer array ZIP of size LEN to the integer array X.

#### ARGUMENTS:

		declaration		
<u>order</u>	<u>name</u>	<u>type</u>	<u>size</u>	<u>status</u>
1	LEN	I*4		I      size of arrays
2	ZIP	I*4	(V)	I      input array of size LEN
3	X	I*4	(V)	O      output array of size LEN

#### COMMON USAGE:

none

#### CALLS:

none

#### CALLED BY:

##### group    routine

CASES	CSST
UTDATE	TIMCHK        TIMDIF
WDMRX	CHKTIM
WDTMS2	WTDATE

### **DAYMON**

### **DAYMON**

This INTEGER FUNCTION is number 5 in file UTDATE.

Return the number of days in the given month for the given year, with leap year taken into account. For an invalid month, -1 is returned. For an invalid year and a valid month, the correct number of days is returned, with February = 28.

#### ARGUMENTS:

		declaration		
<u>order</u>	<u>name</u>	<u>type</u>	<u>size</u>	<u>status</u>
1	YR	I*4		I      year, valid range is 1 - 2080
2	MON	I*4		I      month, valid range is 1 - 12

COMMON USAGE:

none

CALLS:

routine

MOD

CALLED BY:

group routine

CASES	CSTS						
UTCHAR	DATLST						
UTDATE	DATCHK	DATNXT	NUMPTS	TIMADD	TIMBAK	TIMCNV	TIMCVT
	TIMDIF						
WDTMS2	WTGRP	WTSGRP					

TIMADD

TIMADD

This SUBROUTINE is number 9 in file UTDATE.

Add NVALS time steps to first date to compute second date. The first date is assumed to be valid.

ARGUMENTS:

order	name	declaration type size	status	explanation
1	DATE1	I*4 (6)	I	starting date
2	TCODE	I*4	I	time units
				1 - second
				2 - minute
				3 - hour
				4 - day
5	TSTEP	I*4	I	time step in TCODE units
4	NVALS	I*4	I	number of time steps to be added
5	DATE2	I*4 (6)	O	new date

COMMON USAGE:

none

CALLS:

routine

DAYMON

CALLED BY:

group routine

CASES	CSST						
UTDATE	TIMDFX	TIMDIF					
UTEXPT	PRWMTE						
UTIMPT	PRWMFI						
WDMRX	CHKTIM						
WDSPTM	WSTAGP	WSTFGP	WSTGSU	WSTSCP			

## **WADGDF**

## **WADGDF**

This SUBROUTINE is number 9 in file WDATMS.

get the default value for an attribute off the message file

### ARGUMENTS:

<u>order</u>	<u>name</u>	<u>declaration</u>	<u>type</u>	<u>size</u>	<u>status</u>	<u>explanation</u>
1	MESSFL	I*4	I			Fortran unit number for message file
2	DPTR	I*4	I			pointer to start of details for this attribute
3	ATTYP	I*4	I			attribute type
4	ATDEF	R*4	O			default value for attribute

### COMMON USAGE:

none

### CALLS:

routine

WATWDS      WDNXDV      WDPRPS      WDPTSP      WMSSKB

### CALLED BY:

group    routine

CASES : CSATR

## **WADGDS**

## **WADGDS**

This SUBROUTINE is number 10 in file WDATMS.

get the description for an attribute off the message file

### ARGUMENTS:

<u>order</u>	<u>name</u>	<u>declaration</u>	<u>type</u>	<u>size</u>	<u>status</u>	<u>explanation</u>
1	MESSFL	I*4	I			Fortran unit number for message file
2	DPTR	I*4	I			pointer to start of details for this attribute
3	SADESC	C*1 (47)	O			description for attribute

### COMMON USAGE:

none

### CALLS:

routine

WATWDS      WDNXDV      WDPRPS      WDPTSP      WMSGTE      WMSSKB

### CALLED BY:

group    routine

CASES : CSATR

## WADGHL

## WADGHL

This SUBROUTINE is number 11 in file WDATMS.

get the length and starting record/pos of the help info for an attribute off the message file

### ARGUMENTS:

<u>order</u>	<u>name</u>	<u>declaration</u>	<u>type</u>	<u>size</u>	<u>status</u>	<u>explanation</u>
--------------	-------------	--------------------	-------------	-------------	---------------	--------------------

1	MESSFL	I*4	I			Fortran unit number for message file
2	DPTR	I*4	I			pointer to start of details for this attribute
3	TLEN	I*4	O			length of help info (0 - no help)
4	DREC	I*4	O			record on which help info starts
5	DPOS	I*4	O			position on record where help starts

### COMMON USAGE:

none

### CALLS:

routine

WATWDS      WDNXDV      WDPRPS      WDPTSP      WMSSKB

### CALLED BY:

group    routine

CASES      CSATR

## WDBCRL

## WDBCRL

This SUBROUTINE is number 1 in file WDBTCH.

add a label to a wdmsfl

### ARGUMENTS:

<u>order</u>	<u>name</u>	<u>declaration</u>	<u>type</u>	<u>size</u>	<u>status</u>	<u>explanation</u>
--------------	-------------	--------------------	-------------	-------------	---------------	--------------------

1	WDMSFL	I*4	I			watershed data management file unit number
2	DSN	I*4	I			data-set number to add
3	DSTYPE	I*4	I			type of data set, 1- timeseries, 2-table, ...
4	RETCOD	I*4	O			return code, 0 - label added -71 - data set already exists

### COMMON USAGE:

none

### CALLS:

routine

WDDSCK      WDLBAD

### CALLED BY:

group    routine

CASES      CSTA      CSTS

## WDBOPN

## WDBOPN

This SUBROUTINE is number 1 in file USYSPR.

Open a WDM file. File is opened as new or old, depending on the value of RONWFG. The common block related to the WDM record buffer are initialized the first time this routine is called.

### ARGUMENTS:

<u>order</u>	<u>name</u>	<u>type size</u>	<u>status</u>	<u>explanation</u>
1	WDMSFL	I*4	I	Fortran unit number of the WDM file
2	WDNAME	C*64	I	name of the WDM file
3	RONWFG	I*4	I	read only/new file flag 0- normal open of existing WDM file, 1- open WDM file as read only (system dependent), 2- open new WDM file
4	RETCOD	I*4	O	return code 0 - successful open 1 - successful open, but invalid WDM file <0 - error on open, -IOSTAT, compiler specific

### COMMON USAGE:

none

### CALLS:

#### routine

WDBFIN      WDCREA      WDFLCK

### CALLED BY:

#### group    routine

CASES      CASES  
WDIMEX      WDIMEX

## WDBSAC

## WDBSAC

This SUBROUTINE is number 2 in file WDATTRB.

adds (or modifies) character search attribute on given dsn

### ARGUMENTS:

<u>order</u>	<u>name</u>	<u>type size</u>	<u>status</u>	<u>explanation</u>
1	WDMSFL	I*4	I	watershed data management file unit number
2	DSN	I*4	I	data-set number being modified
3	MESSFL	I*4	I	message file unit number
4	SAIND	I*4	I	index number of attribute or highest attribute number if printing
5	SALEN	I*4	I	length of attribute
6	SAVAL	C*1 (V)	I	value of attribute
7	RETCOD	I*4	O	return code indicating if add or mod was successful 0 - successful -81 - data set does not exist -101 - incorrect character value for attribute -103 - no room on label for attribute -104 - data present, can't update attribute -105 - attribute not allowed for this type data set

COMMON USAGE:

block name status

CFBUFF WIBUFF A

CALLS:

routine

CHKSTR WADGVA WDDPAR WDDSCK WRRCGO WRRCUP WDSAGY WDSASP

CALLED BY:

group routine

CASES CSST CSTA CSTS  
WDIMEX PRWMIM

**WDBSAD**

**WDBSAD**

This SUBROUTINE is number 5 in file WDATTRB.

deletes search attribute on given dsn

ARGUMENTS:

			declaration	
<u>order</u>	<u>name</u>	<u>type</u>	<u>size</u>	<u>status</u> <u>explanation</u>
1	WDMMSFL	I*4		I watershed data management file unit number
2	DSN	I*4		I data-set number being modified
3	SAIND	I*4		I index number of attribute
4	SAUPFG	I*4		I update allowed if data present flag(0=yes)
5	SARQWD	I*4		I search attribute required word
6	SALEN	I*4		I length of attribute
7	RETCOD	I*4	O	flag indicating if deletion successful 0 - deletion successful -81 - data set does not exist -104 - data present, can't delete attribute -106 - attribute reqd. for this type data set, can't delete -107 - attribute not present on this data set

COMMON USAGE:

block name status

CFBUFF WIBUFF A

CALLS:

routine

WDDPAR WDDSCK WRRCGO WRRCUP WDSAFL

CALLED BY:

group routine

CASES CSATR

**WDBSAI****WDBSAI**

This SUBROUTINE is number 3 in file WDATRB.

adds (or modifies) integer search attribute on given dsn

## ARGUMENTS:

<u>order</u>	<u>name</u>	<u>declaration</u>	<u>type</u>	<u>size</u>	<u>status</u>	<u>explanation</u>
1	WDMSFL	I*4			I	watershed data management file unit number
2	DSN	I*4			I	data-set number being modified
3	MESSFL	I*4			I	message file unit number
4	SAIND	I*4			I	index number of attribute or highest attribute number if printing
5	SALEN	I*4			I	length of attribute
6	SAVAL	I*4	(V)		I	value of attribute
7	RETCOD	I*4			O	return code indicating if add or mod was successful 0 - successful -81 - data set does not exist -103 - no room on label for attribute -104 - data present, can't update attribute -105 - attribute not allowed for this type data set -108 - incorrect integer value for attribute

## COMMON USAGE:

<u>block</u>	<u>name</u>	<u>status</u>
CFBUFF	WIBUFF	A

## CALLS:

routine

CHKINT	WADGRA	WDDPAR	WDDSCK	WDRCGO	WDRCP	WDSAGY	WDSASP
--------	--------	--------	--------	--------	-------	--------	--------

## CALLED BY:

group    routine

CASES	CSST	CSTS
WDIMEX	PRWMIM	

**WDBSAR****WDBSAR**

This SUBROUTINE is number 4 in file WDATRB.

adds (or modifies) real search attribute on given dsn

## ARGUMENTS:

<u>order</u>	<u>name</u>	<u>declaration</u>	<u>type</u>	<u>size</u>	<u>status</u>	<u>explanation</u>
1	WDMSFL	I*4			I	watershed data management file unit number
2	DSN	I*4			I	data-set number being modified
3	MESSFL	I*4			I	message file unit number
4	SAIND	I*4			I	index number of attribute or highest attribute number if printing
5	SALEN	I*4			I	length of attribute
6	SAVAL	R*4	(V)		I	value of attribute
7	RETCOD	I*4			O	flag indicating if modification or addition was successful 0 - successful

-81 - data set does not exist  
-103 - no room on label for attribute  
-104 - data present, can't update attribute  
-105 - attribute not allowed for this type data set  
-109 - incorrect real value for attribute

COMMON USAGE:

block name status

.CFBUFF WIBUFF A

CALLS:

routine

CHKREA WADGRA WDDPAR WDDSCK WRDRCGO WRDCUP WDSAGY WDSASP

CALLED BY:

group routine

CASES CSTS  
WDIMEX PRWMIM

**WDBSGC**

**WDBSGC**

This SUBROUTINE is number 2 in file WDBTCH.

gets values of character search attribute for a dsn

ARGUMENTS:

		declaration	
<u>order</u>	<u>name</u>	<u>type</u> <u>size</u>	<u>status</u> <u>explanation</u>
1	WDMMSFL	I*4	I watershed data management file unit number
2	DSN	I*4	I data-set number to add
3	SAIND	I*4	I index number of attribute
4	SALEN	I*4	I length of attribute
5	SAVAL	C*1 (V)	O value of attribute
6	RETCOD	I*4	O return code, 0 - attribute value returned -81 - data set does not exist -107 - attribute not present on this data set

COMMON USAGE:

block name status

.CFBUFF WIBUFF A

CALLS:

routine

WDDSCK WRDRCGO WDSAFL

CALLED BY:

group routine

CASES CSATR

## **WDBSGI**

## **WDBSGI**

This SUBROUTINE is number 3 in file WDBTCH.

gets the values of integer search attribute for a dsn

### ARGUMENTS:

<u>order</u>	<u>name</u>	<u>declaration</u>	<u>status</u>	<u>explanation</u>
1	WDMSFL	I*4	I	watershed data management file unit number
2	DSN	I*4	I	data-set number to add
3	SAIND	I*4	I	index number of attribute
4	SALEN	I*4	I	length of attribute
5	SAVAL	I*4 (V)	O	value of attribute
6	RETCOD	I*4	O	return code, 0 - attribute value returned -81 - data set does not exist -107 - attribute not present on this data set

### COMMON USAGE:

block    name    status

CFBUFF    WIBUFF    A

### CALLS:

routine

WDDSCK    WDRCGO    WDSAFL

### CALLED BY:

group    routine

CASES    CSATR

## **WDBSGR**

## **WDBSGR**

This SUBROUTINE is number 4 in file WDBTCH.

Get the values of real search attribute for a data set.

### ARGUMENTS:

<u>order</u>	<u>name</u>	<u>declaration</u>	<u>status</u>	<u>explanation</u>
1	WDMSFL	I*4	I	watershed data management file unit number
2	DSN	I*4	I	data-set number to add
3	SAIND	I*4	I	index number of attribute
4	SALEN	I*4	I	length of attribute
5	SAVAL	R*4 (V)	O	value of attribute
6	RETCOD	I*4	O	return code 0 - attribute value returned -81 - data set does not exist -107 - attribute not present on this data set

### COMMON USAGE:

block    name    status

CFBUFF    WIBUFF    A  
CFBUFF    WRBUFF    I

CALLS:

routine

WDDSCCK WDRCGO WDSAFL

CALLED BY:

group routine

CASES CSATR

**WDCKDT**

**WDCKDT**

This INTEGER FUNCTION is number 11 in file UTWDM.D.

Check data set for existance and type, returns:

0 - data set does not exist	
or data-set type	
1 - time series	6 - raster
2 - table	7 - space-time
3 - schematic	8 - attribute
4 - project	9 - message
5 - vector	

ARGUMENTS:

		declaration	
<u>order</u>	<u>name</u>	<u>type</u>	<u>size</u>
			<u>status</u>
1	WDMSFL	I*	4
2	DSN	I*	4
		I	Fortran unit number of WDM file
		I	data-set number to be checked

COMMON USAGE:

block name status  
CFBUFF WIBUFF I

CALLS:

routine

WDDSCCK WDRCGO

CALLED BY:

group routine

CASES CASES

**WDDSCCK**

**WDDSCCK**

This SUBROUTINE is number 12 in file UTWDM.D.

Check data set for existance and return record number of first record in data set (contains label)

ARGUMENTS:

		declaration	
<u>order</u>	<u>name</u>	<u>type</u>	<u>size</u>
			<u>status</u>
1	WDMSFL	I*	4
2	DSN	I*	4
		I	Fortran unit number of WDM file
		I	data-set number to be checked

3	DREC	I*4	0	record number of first record in data set
4	RETCOD	I*4	0	return code
				0 - data set exists
				-81 - data set does not exist
				-84 - data set number out of range

COMMON USAGE:

block    name    status

CFBUFF    WIBUFF    I

CALLS:

routine

MOD            WDDRRRC            WDRCGO

CALLED BY:

group    routine

CASES	CSST.					
UTWDMD	WDCKDT	WDSCHA				
UTWDMF	WMSFBC					
WDATMS	WADDSI	WADGTL				
WDATRB	WDBSAC	WDBSAD	WDBSAI	WDBSAR		
WDBTCH	WDBCRL	WDBSGC	WDBSGI	WDBSGR	WDDSDL	WDDSRN
WDIMEX	PRWMEX	PRWMIM	WDIMEX			
WDLBLE	WDDSCL	WDFCUP	WDRCDL			
WDSPTM	WSTDGP					
WDTMS1	WTBYFX					

WDDSCL

WDDSCL

This SUBROUTINE is number 1 in file WDLBLE.

copies an old data-set label into a new data-set label

ARGUMENTS:

<u>order</u>	<u>name</u>	<u>declaration</u>				
1	WDMSFL	I*4	I	watershed data management file unit number		
2	ODSN	I*4	I	old data-set number		
3	NDSN	I*4	I	new data-set number		
4	RETCOD	I*4	O	return code		
				0 - copy complete		
				-61 - old data set doesn't exist		
				-62 - new data set already exists		

COMMON USAGE:

block    name    status

CFBUFF    RECNO    A  
CFBUFF    WIBUFF    A

CALLS:

routine

WDDSCK    WDFCUP    WDFDUP    WDPTCL    WDRCGO    WDRCGX    WDRCP

CALLED BY:

group routine

CASES CSTA CSTS

## WDDSDL

## WDDSDL

This SUBROUTINE is number 5 in file WDBTCH.

routine to delete a data set from the WDMSFL with no user interact

ARGUMENTS:

		declaration		
<u>order</u>	<u>name</u>	<u>type</u>	<u>size</u>	<u>status</u> <u>explanation</u>
1	WDMSFL	I*4		I Fortran unit number of WDM file
2	DSN	I*4		I dataser number to be deleted
3	RETCOD	I*4		O return code 0 - data set successfully deleted -81 - data set does not exist

COMMON USAGE:

block name status

CFBUFF WIBUFF I

CALLS:

routine

WDDCK WDFCUP WDFDUP WDRCDL WDRCGO

CALLED BY:

group routine

CASES CASES  
WDIMEX PRWMIM

## WDDSRN

## WDDSRN

This SUBROUTINE is number 6 in file WDBTCH.

routine to renumber data sets with no user interaction

ARGUMENTS:

		declaration		
<u>order</u>	<u>name</u>	<u>type</u>	<u>size</u>	<u>status</u> <u>explanation</u>
1	WDMSFL	I*4		I Fortran unit number for WDM file
2	ODSN	I*4		I old data-set number
3	NDSN	I*4		I new data-set number
4	RETCOD	I*4		O return code 0 - renumber successfully completed -72 - old data set does not exist -73 - new data set already exists

COMMON USAGE:

block name status

CFBUFF WIBUFF M  
CDRLOC PTSNUM I

CALLS:

routine

WDDCK WDFDUP WDRCGO WDRCP

CALLED BY:

group routine

CASES CASES

**WDFLCL**

**WDFLCL**

This SUBROUTINE is number 3 in file UTWDMD.

Remove a WDM file from the open WDM buffer and adjust  
buffer accordingly.

ARGUMENTS:

			declaration	
<u>order</u>	<u>name</u>	<u>type</u>	<u>size</u>	<u>status</u> <u>explanation</u>
1	WDMFL	I*4		I Fortran unit number of WDM file
2	RETCOD	I*4		O return code 0 - everything ok -87 - can't remove message WDM file from buffer

COMMON USAGE:

block name status

CFBUFF WDMCNT M  
CFBUFF WDMFUN O  
CFBUFF WDMOPN O

CALLS:

routine

CALLED BY:

group routine

CASES CASES

**WDLBAX**

**WDLBAX**

This SUBROUTINE is number 6 in file WDLBLE.

add a new data-set label, but no search attributes or data.

ARGUMENTS:

			declaration	
<u>order</u>	<u>name</u>	<u>type</u>	<u>size</u>	<u>status</u> <u>explanation</u>
1	WDMFL	I*4		I watershed data management file unit number
2	DSN	I*4		I data-set number
3	DSTYPE	I*4		I type of data set
4	NDN	I*4		I number of down pointers
5	NUP	I*4		I number of up pointers
6	NSA	I*4		I number of search attributes
7	NSASP	I*4		I amount of search attribute space

8	NDP	I*4	I	number of data pointers
9	PSA	I*4	O	pointer to search attribute space

COMMON USAGE:

block name status

CFBUFF	RECNO	A
CFBUFF	WIBUFF	O

CALLS:

routine

WDFCUP	WDFDUP	WDPTCL	WDRCGO	WDRCGX	WDRCUP
--------	--------	--------	--------	--------	--------

CALLED BY:

group routine

CASES	CSST
WDIMEX	PRWMIM
WDLBLE	WDLBAD

**WDLGET**

**WDLGET**

This SUBROUTINE is number 3 in file WDDLG.

retrieve DLG header or coordinate pairs from WDM file

ARGUMENTS:

			declaration	
order	<u>name</u>	<u>type</u>	<u>size</u>	<u>status</u> explanation
1	WDMSFL	I*4	I	Fortran unit number for WDM file
2	DSN	I*4	I	data-set number on WDM file
3	ITYPE	I*4	I	type of DLG info (1- LINE, 2- AREA, 3- NODE)
4	ATT1	I*4	I	major attribute value
5	ATT2	I*4	I	minor attribute value
6	LEN	I*4	I	max length of information being retrieved (4 byte words)
7	ID	I*4	M	id of information retrieved (0-either, 1-header, 2-data)
8	OLEN	I*4	O	actual length of output buffer
9	DLGBUF	R*4 (V)	O	buffer of information being retrieved
10	RETCOD	I*4	O	return code 2 - no more data in this group 1 - more of current id remaining 0 - DLG data retrieved successfully -81 - data set does not exist -82 - data set exists, but is wrong DSTYP -50 - major and minor attributes not found on this data set

COMMON USAGE:

block name status

CFBUFF	WIBUFF	A
--------	--------	---

CALLS:

routine

WDLBSP	WDLLCK	WDNXPS	WDPTSP	WDRCGO	WDSCHK	"
--------	--------	--------	--------	--------	--------	---

CALLED BY:

group routine

CASES CSDLG  
UTEXPT PRWMDE

## WDLLSU

## WDLLSU

This SUBROUTINE is number 5 in file WDDLG.

summarize label information for DLG data set

ARGUMENTS:

<u>order</u>	<u>name</u>	<u>type</u>	<u>size</u>	<u>status</u>	<u>declaration</u>	<u>explanation</u>
1	WDMMSFL	I*	4	I	Fortran unit number for WDM file	
2	DSN	I*	4	I	data-set number on WDM file	
3	ILEN	I*	4	I	maximum size of information buffers	
4	OLEN	I*	4	O	actual amount of information returned (<= ILEN)	
5	TYPE	I*	4	(V)	O buffer of information types on DSN	
6	ATT1	I*	4	(V)	O buffer of major attributes on DSN	
7	ATT2	I*	4	(V)	O buffer of minor attributes on DSN	
8	RETCOD	I*	4	O	return code	
					1 - more groups on DSN	
					0 - label summary returned successfully	
					-81 - data set does not exist	
					-82 - data set exists, but is wrong DSTYP	

COMMON USAGE:

block name status

CFBUFF WIBUFF I

CALLS:

routine

WDLISP WDRCGO WDSCHK

CALLED BY:

group routine

CASES CSDLG  
UTEXPT PRWMDE

## WDNXDV

## WDNXDV

This SUBROUTINE is number 5 in file UTWDMF.

Move to the next data position and return the integer equivalent of the data value.

ARGUMENTS:

<u>order</u>	<u>name</u>	<u>type</u>	<u>size</u>	<u>status</u>	<u>declaration</u>	<u>explanation</u>
1	WDMMSFL	I*	4	I	Fortran unit number for WDM file	
2	DREC	I*	4	M	record number of data on WDM file	
3	DPOS	I*	4	M	position of data on data record (both DREC and DINP)	
4	DVAL	I*	4	O	data value on WDM file	

COMMON USAGE:

block name status

CFBUFF WIBUFF I

CALLS:

routine

WDNXPS

CALLED BY:

group routine

CASES	CSTXT			
MSEXPT	PRMSFE	PRMSME	PRMSPE	PRMSTE
UTEXPT	PRMSAE			
UTWDMF	WMSIDP			
WDATMS	WADGDF	WADGRA	WADGVA	WADGDS
WDTBLE	WDTBSP			WADGHL

**WDSAFI**

**WDSAFI**

This SUBROUTINE is number 11 in file WDATTRB.

Given the attribute name SAFNAM, starting at attribute index SAIND, find the first attribute name which matches SAFNAM. Return the index of the next attribute which also matches SAFNAM, if one exists.

ARGUMENTS:

		declaration		
<u>order</u>	<u>name</u>	<u>type</u> <u>size</u>	<u>status</u>	<u>explanation</u>
1	MESSFL	I*4	I	Fortran unit number for message file
2	SAFNAM	C*1 (6)	I	character array containing attribute name to search for
3	SAIND	I*4	M	index of next matching attribute, if one exists, otherwise, index of matching attribute
4	SANAM	C*1 (6)	O	character array of first attribute name matching SAFNAM
5	RETCOD	I*4	O	return code -110 - attributes not found on message file -111 - attribute name not found (no match) -112 - more attributes exist which match SAFNAM

COMMON USAGE:

none

CALLS:

routine

ASRTC CHRCHR LENSTR QUPCAS WADDSSI WADGTN

CALLED BY:

group routine

CASES	CSATR
WDATTRB	WDBSGX

## WDSAGY

## WDSAGY

This SUBROUTINE is number 1 in file WDATTRB.

gets general detail information about specified attribute

### ARGUMENTS:

<u>order</u>	<u>name</u>	<u>declaration</u>	<u>type</u>	<u>size</u>	<u>status</u>	<u>explanation</u>
1	MESSFL	I*4	I			message file unit number
2	SAIND	I*4	I			attribute index number
3	SANAM	C*1 (6)	O			name of search attribute
4	DPTR	I*4	O			pointer to other details of attribute
5	SATYP	I*4	O			type of attribute
6	SALEN	I*4	O			length of attribute
7	SARQWD	I*4	O			word containing attribute requirements by dsn type
8	SAUPFG	I*4	O			attribute update flag

### COMMON USAGE:

none

### CALLS:

#### routine

WADDSSI      WADGTL      ZIPC

### CALLED BY:

#### group    routine

CASES	CSATR			
WDATMS	WADGTN			
WDATTRB	WDBSAC	WDBSAI	WDBSAR	WDBSGX
WDIMEX	PRWMEX			

## WDTBDL

## WDTBDL

This SUBROUTINE is number 11 in file WDTBLE.

delete WDM table from WDM file

### ARGUMENTS:

<u>order</u>	<u>name</u>	<u>declaration</u>	<u>type</u>	<u>size</u>	<u>status</u>	<u>explanation</u>
1	WDMSFL	I*4	I			Fortran unit number of WDM file
2	DSN	I*4	I			Table data-set number
3	TABNAM	C*16	I			Table data-set name
4	TABIND	I*4	I			Table identifier
5	RETCOD	I*4	O			Return code

### COMMON USAGE:

#### block    name    status

CFBUFF    WIBUFF    M

### CALLS:

#### routine

WDPTCL WDPTSP WDRCDL WRDCGO WDRCPW WDTBFN WTBDSP

CALLED BY:

group routine

CASES CSTA

### WDTBFX

### WDTBFX

This SUBROUTINE is number 2 in file WDTBLE.

determines pointer to the specified table,  
also returns its message file cluster and group number

ARGUMENTS:

<u>order</u>	<u>name</u>	<u>type</u>	<u>size</u>	<u>status</u>	<u>declaration</u>	<u>explanation</u>
1	WDMFL	I*4		I		Fortran unit number of WDM file
2	DSN	I*4		I		table data-set number
3	TABIND	I*4		I		table identifier number
4	TABNAM	C*16		I		name of table
5	TBCNT	I*4		O		total number of tables in data set
6	LREC	I*4		O		label record number
7	TGRPPT	I*4		O		table group pointer
8	MFID	C*1 (2)		O		message file name id
9	TCLU	I*4		O		table message file cluster number
10	TGRP	I*4		O		table message file group number
11	NROW	I*4		O		number of rows in table
12	RETCOD	I*4		O		return code
						0 - table found, pointer and other information returned

COMMON USAGE:

block name status

CFBUFF WIBUFF A

CALLS:

routine

WRDCGO WDTBFN WTBISP

CALLED BY:

group routine

CASES CSTA

### WDTBSU

### WDTBSU

This SUBROUTINE is number 6 in file WDTBLE.

get WDM table label info from WDM file table data set

ARGUMENTS:

<u>order</u>	<u>name</u>	<u>type</u>	<u>size</u>	<u>status</u>	<u>declaration</u>	<u>explanation</u>
1	WDMFL	I*4		I		Fortran unit number of WDM file
2	DSN	I*4		I		WDM table data-set number
3	TABMAX	I*4		I		Maximum number of tables to get info about

4	TABBAS	I*4	I	Table base pointer, first group to get info about
5	TABCNT	I*4	O	Total number of tables found
6	TABNAM	C*1 (16,V)	O	Name of table
7	TABID	I*4 (V)	O	Id of table
8	TABDIM	I*4 (V)	O	Dimensions of table
9	PDATVL	I*4 (V)	O	Pointer to table data values
10	RETCOD	I*4	O	Return code, (+) if more tables than TABMAX

COMMON USAGE:

block name status

CFBUFF WIBUFF I

CALLS:

routine

COPYC WDRCGO WDSCHK

CALLED BY:

group routine

CASES CSTA

## WDTBTM

## WDTBTM

This SUBROUTINE is number 1 in file WDTBLE.

put WDM table template on WDM file, return parameters about table

ARGUMENTS:

<u>order</u>	<u>name</u>	<u>type</u>	<u>size</u>	<u>declaration</u>	<u>status</u>	<u>explanation</u>
1	MESSFL	I*4		I		Fortran unit number of WDM file containing table definition
2	MFID	C*1 (2)		I		Message file name id of MESSFL
3	TCLU	I*4		I		Message file cluster containing table template
4	TGRP	I*4		I		Group number containing table template
5	WDMSFL	I*4		I		Fortran unit number of WDM file to put table template in
6	DSN	I*4		I		Data-set number to put table template in
7	TABIND	I*4		I		Table identifier number of new table
8	NROW	I*4		I		Number of rows in new table
9	TFLDS	I*4		O		Number of fields in table
10	TNUM	I*4 (4)		O		Number of each variable type in table(I-1,R-2,C-3,D-4)
11	TTYP	C*1 (30)		O		Type of each field in table(I,R,C,D)
12	TLEN	I*4 (30)		O		Length of each field in table(characters)
13	TCOL	I*4 (30)		O		Starting column for each field
14	TSPA	I*4		O		Space required for each table row(words)
15	MTBNAM	C*16		O		Name of table from message file
16	TGRPPT	I*4		O		Pointer to group within DSN
17	AFLDS	I*4		O		Number of fields in table extension
18	ANUM	I*4 (4)		O		Number of each variable type in table extension(see 10)
19	ATYP	C*1 (30)		O		Type of each field in table extension
20	ALEN	I*4 (30)		O		Length of each field in table extension
21	ACOL	I*4 (30)		O		Starting column for each associated field
22	ASPA	I*4		O		Space required for table extension
23	ACLU	I*4		O		Associated table cluster number
24	AGRP	I*4		O		Associated table group number
25	RETCOD	I*4		O		Return code

COMMON USAGE:

block name status

CFBUFF RECNO I  
CFBUFF WIBUFF M

CALLS:

routine

WDPTCL WDPTSP WDRCGO WDRCGX WDRCUP WDTBFN WDTBSP WTBDCL  
WTBICL

CALLED BY:

group routine

CASES CSTA

**WDTGET**

**WDTGET**

This SUBROUTINE is number 1 in file WDTMS1.

gets timeseries information from the WDMMSFL

ARGUMENTS:

		declaration	
<u>order</u>	<u>name</u>	<u>type</u> <u>size</u>	<u>status</u> <u>explanation</u>
1	WDMMSFL	I*4	I watershed data management file unit number
2	DSN	I*4	I data-set number
3	DELT	I*4	I time step for get
4	DATES	I*4 (6)	I starting date
5	NVAL	I*4	I number of values
6	DTRAN	I*4	I transformation code 0 - ave,same 1 - sum,div 2 - max 3 - min
7	QUALFG	I*4	I allowed quality code
8	TUNITS	I*4	I time units for get
9	RVAL	R*4 (V)	O array to place retrieved values in
10	RETCOD	I*4	O return code 0 - everything O.K. -8 - invalid date -14 - date specified not within valid range for data set -20 - problem with one or more of following: GPFLG, DXX, NVAL, QUALVL, LTSTEP, LTUNIT -21 - date from WDM doesn't match expected date -81 - data set does not exist -82 - data set exists, but is wrong DSTYP -84 - data set number out of range

COMMON USAGE:

block name status

CFBUFF WIBUFF A

CALLS:

routine

FLOAT	MOD	WDATCP	WDRCGO	WTDSPX	WTFNDG	WTGTVL
WTPMCK	WTSCSC	ZIPR				

CALLED BY:

group routine

CASES	CSTS
WDSPTM	WSTSCP

## WDTPUT

## WDTPUT

This SUBROUTINE is number 17 in file WDTMS1.

Puts time series data into a WDM file. This routine traps the problem with overwriting existing data.

### ARGUMENTS:

order	name	declaration	type	size	status	explanation
1	WDMMSFL	I*4	I			watershed data management file unit number
2	DSN	I*4	I			data-set number
3	DELT	I*4	I			time step for put
4	DATES	I*4 (6)	I			starting date
5	NVAL	I*4	I			number of values
6	DTOVWR	I*4	I			data overwrite flag, 0 - dont overwrite 1 - overwrite O.K.
7	QUALFG	I*4	I			allowed quality code
8	TUNITS	I*4	I			time units for put
9	RVAL	R*4 (V)	I			array for writing out values
10	RETCOD	I*4	O			return code 0 - everything is O.K. -8 - invalid date -9 - data not present in current group -10 - no data in this group -11 - no non missing data, data has not started yet -14 - date specified not within valid range for data set -15 - VBTIME=1 and DELT,TUNITS do not agree with the data set -20 - problem with one or more of following: DTOVWR, NVAL, QUALFG, TUNITS, DELT -21 - date from WDM doesn't match expected date -81 - data set does not exist -82 - data set exists, but is wrong DSTYP -84 - data set number out of range -85 - trying to write to a read-only data set

### COMMON USAGE:

none

### CALLS:

routine

WDTPFX	WTDEL
--------	-------

### CALLED BY:

group routine

CASES CSTS  
UTIMPT PRWMTI  
WDSPTM WSTSCP

## WMSBCS

## WMSBCS

This SUBROUTINE is number 13 in file UTWDT1.

Split up a block control word for a message type data set into its components.

### ARGUMENTS:

			declaration	
order	<u>name</u>	<u>type</u>	<u>size</u>	<u>status</u> <u>explanation</u>
1	QWORD	I*4		I message type dataset block control word
2	CLASS	I*4		O class of information 1 - 1-dimensional parameter      4 - menu 2 - 2-dimensional parameter      5 - file 3 - text
3	ID	I*4		O id for portion of group examples: CLASS ID Description 1      4 default value for parameter field 4      6 help for menu screen 5      3 <u>status</u> of file
4	ORDER	I*4		O order of information
5	TLEN	I*4		O total number of characters in the block

### COMMON USAGE:

none

### CALLS:

routine

MOD

### CALLED BY:

group routine

CASES	CSTXT				
MSEXPT	PRMSFE	PRMSME	PRMSPE	PRMSTE	PRWMME
UTWDMP	WMSBCX	WMSIDP			
WDMRX	CHKMBS				
WDTBLE	WDTBSP				

## WMSFBC

## WMSFBC

This SUBROUTINE is number 12 in file UTWDMP.

Get first block control word and its position for group GNUM in a message data set. A STOP is encountered when group GNUM does not exist.

### ARGUMENTS:

			declaration	
order	<u>name</u>	<u>type</u>	<u>size</u>	<u>status</u> <u>explanation</u>
1	WDMSFL	I*4		I Fortran unit number for WDM file
2	DSN	I*4		I data-set number
3	GNUM	I*4		I group number
4	DREC	I*4		O record number of block control word on WDM file
5	DPOS	I*4		O position of block control word on DREC
6	BCWORD	I*4		O block control word

COMMON USAGE:

block name status

CFBUFF WIBUFF A

CALLS:

routine

WDDSCX WDPTSP WRDRCG WMSQCK

CALLED BY:

group routine

CASES CSTXT  
MSEXPT PRWMME  
UTWDMF WMSIDP  
WDTBLE WDTBSP

**WMSGTE**

**WMSGTE**

This SUBROUTINE is number 8 in file UTWDMF.

Get one record of text off WDM file.

ARGUMENTS:

<u>order</u>	<u>name</u>	<u>declaration</u>	<u>type</u>	<u>size</u>	<u>status</u>	<u>explanation</u>
1	WDMSFL	I*4	I			Fortran unit number for WDM file
2	TLEN	I*4	I			total length of text (may be more than one record)
3	LLEN	I*4	I			maximum size of record to get
4	DREC	I*4	M			record number of data on WDM file
5	DPOS	I*4	M			position of data on data record
6	GLEN	I*4	M			counter to keep track of when to read off WDM file should be initialized to 0 for first call
7	MLEN	I*4	M			number of characters retrieved so far (must be <= TLEN) should be initialized to 0 for first call
8	OLEN	I*4	O			actual size of record retrieved
9	OBUFF	C*1 (V)	O			array of size LLEN containing OLEN characters retrieved
10	CONT	I*4	O			indicator flag for text 0 - no more text available 1 - more text available

COMMON USAGE:

block name status

CFBUFF WIBUFF I

CALLS:

routine

ICHAR MOD WDNXPS WRDRCG ZIPC

CALLED BY:

group routine

CASES CSTXT

MSEXPT	PRMSTA
UTWDMF	WMSGTO
WDATMS	WADGVA
WDTBLE	WDTBSP

## WMSSKB

## WMSSKB

This SUBROUTINE is number 14 in file UTWDMF.

Position DREC and DPOS at the end of the current data block. DREC and DPOS are assumed to be input as the start of the block.

### ARGUMENTS:

			declaration
order	name	type size	status explanation
1	WDMSFL	I*4	I Fortran unit number for WDM file
2	TLEN	I*4	I total number of characters to skip
3	DREC	I*4	M record number on WDM file
4	DPOS	I*4	M position on record DREC

### COMMON USAGE:

none

### CALLS:

routine

MOD WDNXPS

### CALLED BY:

group routine

CASES	CSTXT				
WDATMS	WADGDF	WADGRA	WADGVA	WADGDS	WADGHL
WDTBLE	WDTBSP				

## WSTAGP

## WSTAGP

This SUBROUTINE is number 3 in file WDSPTM.

add a group to a space time data set, physically allocate space for

### ARGUMENTS:

			declaration
order	name	type size	status explanation
1	WDMSFL	I*4	I Fortran unit number of WDM file
2	DSN	I*4	I Data-set number
3	NGPDAT	I*4 (6)	I Starting date for group
4	NGPTUN	I*4	I Time units for group
5	NGPTST	I*4	I Time step for group
6	NGPNOV	I*4	I Number of timesteps in group
7	RETCOD	I*4	O Return code 0 - group added -42 - overlap an existing group -43 - can't add another space time group -46 - bad space time group specification parameter -81 - data set does not exist -82 - data set exists, but is wrong DSTYP -84 - data set number out of range

COMMON USAGE:

block name status

CFBUFF RECNO I  
CFBUFF WIBUFF A

CALLS:

routine

MOD	TIMADD	TIMCHK	WDATCL	WDATSP	WDPTCL	WDPTSP	WDRCGO
WDRCGX	WDRCP	WDSAV	WDSCHK	WSTGCL	WSTGSP		

CALLED BY:

group routine

CASES CSST

**WSTGSU**

**WSTGSU**

This SUBROUTINE is number 6 in file WDSPTM.

summarize a group in a space time data set

ARGUMENTS:

<u>order</u>	<u>name</u>	<u>declaration</u>	<u>type</u>	<u>size</u>	<u>status</u>	<u>explanation</u>
--------------	-------------	--------------------	-------------	-------------	---------------	--------------------

1	WDMFL	I*4	I			Fortran unit number of WDM file
2	DSN	I*4	I			Data set number
3	GRPIN	I*4	I			Index of group to summarize
4	GSDAT	I*4 (6)	O			Start date of group
5	GEDAT	I*4 (6)	O			End date of group
6	GTUN	I*4	O			Group time units
7	GTST	I*4	O			Group time steps
8	GNOV	I*4	O			Number of values in group
9	GFRAC	R*4	O			Fraction of group containing data
10	RETCOD	I*4	O			Return code
						0 - group summarized
						-49 - group doesn't exist
						-81 - data set does not exist
						-82 - data set exists, but is wrong DSTYP
						-84 - data set number out of range

COMMON USAGE:

block name status

CFBUFF WIBUFF A

CALLS:

routine

FLOAT	TIMADD	WDATSP	WDPTSP	WDRCGO	WDSCHK	WSTGSP
-------	--------	--------	--------	--------	--------	--------

CALLED BY:

group routine

CASES CSST

## WSTGTR

## WSTGTR

This SUBROUTINE is number 9 in file WDSPTM.

get real space time data

### ARGUMENTS:

<u>order</u>	<u>name</u>	<u>type</u>	<u>size</u>	<u>status</u>	<u>declaration</u>	<u>explanation</u>
1	WDMSFL	I*4		I	Fortran unit number of WDM file	
2	DSN	I*4		I	Data-set number	
3	STDAT	I*4	(6)	I	Date of data to get	
4	NDIM	I*4		I	Number of dimensions specified	
5	NUMN	I*4	(V)	I	Number of values to get in each dimension	
6	BASN	I*4	(V)	I	Base value in each dimension	
7	SKPN	I*4	(V)	I	Skip value in each dimension	
8	NVAL	I*4		I	Total number of values to get	
9	RBUFF	R*4	(V)	O	Buffer to put values in	
10	RETCOD	I*4		O	Return code	
					0 - data retrieved	
					-36 - missing needed following data for a get	
					-37 - no data present	
					-38 - missing part of time required	
					-39 - missing data group	
					-40 - no data available	
					-41 - no data to read	
					-42 - overlap an existing group	
					-44 - trying to get/put more data than in block	
					-45 - types don't match	
					-81 - data set does not exist	
					-82 - data set exists, but is wrong DSTYP	
					-84 - data set number out of range	

### COMMON USAGE:

<u>block</u>	<u>name</u>	<u>status</u>
CFBUFF	WRBUFF	I

### CALLS:

#### routine

WSTWNT

### CALLED BY:

#### group    routine

CASES	CSST
WDSPTM	WSTSCP

## WSTPTR

## WSTPTR

This SUBROUTINE is number 12 in file WDSPTM.

put real space time data

### ARGUMENTS:

<u>order</u>	<u>name</u>	<u>type</u>	<u>size</u>	<u>status</u>	<u>declaration</u>	<u>explanation</u>
1	WDMSFL	I*4		I	Fortran unit number of WDM file	

2	DSN	I*4	I	Data-set number
3	STDAT	I*4 (6)	I	Date of data to get
4	NDIM	I*4	I	Number of dimensions specified
5	NUMN	I*4 (V)	I	Number of values to get in each dimension
6	BASN	I*4 (V)	I	Base value in each dimension
7	SKPN	I*4 (V)	I	Skip value in each dimension
8	NVAL	I*4	I	Total number of values to get
9	RBUFF	R*4 (V)	I	Buffer to write values from
10	RETCOD	I*4	O	Return code 0 - data written -36 - missing needed following data for a get -37 - no data present -38 - missing part of time required -39 - missing data group -40 - no data available -41 - no data to read -42 - overlap an existing group -44 - trying to get/put more data than in block -45 - types don't match -81 - data set does not exist -82 - data set exists, but is wrong DSTYP -84 - data set number out of range

COMMON USAGE:

block    name    status

CPBUFF WRBUFF    O

CALLS:

routine

WDRCPUP    WSTWNT

CALLED BY:

group    routine

CASES    CSST  
WDSPTM    WSTSCP

**WTBCOD**

**WTBCOD**

This SUBROUTINE is number 7 in file WDTBLE.

convert data from full screen buffer into WDM internal format

ARGUMENTS:

			declaration
order	<u>name</u>	<u>type</u> <u>size</u>	<u>status</u> <u>explanation</u>
1	XFLDS	I*4	I    Number of fields in table
2	NROW	I*4	I    Number of rows in table
3	XSPA	I*4	I    Number of columns of table data(fields adjusted for size)
4	XLEN	I*4 (V)	I    Length of each field
5	XTYP	C*1 (V)	I    Type(I,R,C,D,A) of each field
6	XCOL	I*4 (V)	I    Starting column for each field
7	TBCBUF	C*1 (80,V)	I    Buffer containing output from full screen routine
8	MXTBTL	I*4	I    Size of buffer to put data in WDM internal format
9	TBRBUF	R*4 (V)	O    Buffer to put data in WDM internal format
10	RETCOD	I*4	O    Return code

COMMON USAGE:

none

CALLS:

routine

CHRDEC      CHRDPR      CHRINT

CALLED BY:

group    routine

CASES      CSTA

**WTBDSP**

**WTBDSP**

This SUBROUTINE is number 10 in file UTWDT1.

Split up dimension variable for table type data set into its components.

ARGUMENTS:

			declaration
order	<u>name</u>	<u>type</u> <u>size</u>	<u>status</u> <u>explanation</u>
1	TABDIM	I*4	I      table dimension variable
2	TABIND	I*4	O      table index number
3	NROW	I*4	O      number of rows in table
4	NCOL	I*4	O      space for each column(words)
5	NEXT	I*4	O      amount of table extension space (words)

COMMON USAGE:

none

CALLS:

routine

MOD

CALLED BY:

group    routine

CASES      CSTA  
WDMRX      CHKTAB  
WDTBLE      WDTBFN      WTBPUT      WDTBDL      WTBGET

**WTBGET**

**WTBGET**

This SUBROUTINE is number 10 in file WDTBLE.

get the main table data

ARGUMENTS:

			declaration
order	<u>name</u>	<u>type</u> <u>size</u>	<u>status</u> <u>explanation</u>
1	WDMSFL	I*4	I      Watershed data management file unit number
2	DSN	I*4	I      table data-set number
3	TABNAM	C*16	I      table name
4	TABIND	I*4	I      table identifier number
5	DATFLG	I*4	I      data type

			1 - main table
			2 - extension
6	FROW	I*4	I first row of data to read from
7	NROW	I*4	I number of rows of data to read
8	FSPA	I*4	I first data space to read from
9	NSPA	I*4	I space for data in each row
10	TBRBUF	R*4 (V,V)	O buffer for data to get from WDM file
11	RETCOD	I*4	O return code

COMMON USAGE:

block name status

CFBUFF WIBUFF I

CALLS:

routine

WDPTSP WDRCGO WDTBFN WTBDSP

CALLED BY:

group routine

CASES CSTA

**WTBISP**

**WTBISP**

This SUBROUTINE is number 8 in file UTWDT1.

Split up an identifier for a table type data set into its components.

ARGUMENTS:

		declaration		
<u>order</u>	<u>name</u>	<u>type</u> <u>size</u>	<u>status</u>	<u>explanation</u>
1	TABID	I*4	I	table identifier
2	MSFLID	C*1 (2)	O	message file name id
3	MCLU	I*4	O	message file cluster for table
4	MGRP	I*4	O	message file group number for table

COMMON USAGE:

none

CALLS:

routine

CHAR MOD

CALLED BY:

group routine

CASES CSTA  
WDMRX CHKTAB  
WDTBLE WDTBFX

**WTBPUT****WTBPUT**

This SUBROUTINE is number 9 in file WDTBLE.

put WDM table data into WDM file

## ARGUMENTS:

<u>order</u>	<u>name</u>	<u>type</u>	<u>size</u>	<u>status</u>	<u>declaration</u>	<u>explanation</u>
1	WDMSFL	I*4		I		Watershed data management file unit number
2	DSN	I*4		I		table data-set number
3	TABNAM	C*16		I		table name
4	TABIND	I*4		I		table identifier number
5	DATFLG	I*4		I		data type 1 - main table 2 - extension
6	FROW	I*4		I		first row to start writing
7	NROW	I*4		I		number of rows of data to write
8	FSPA	I*4		I		first data space to start writing
9	NSPA	I*4		I		space for data in each row
10	TBRBUF	R*4	(V,V)	I		buffer for data to write onto WDM file
11	RETCOD	I*4		O		return code

## COMMON USAGE:

<u>block</u>	<u>name</u>	<u>status</u>
CFBUFF	WIBUFF	I

## CALLS:

routine

WDPTSP      WDRCGO      WDRCUP      WDTBFN      WTBDSP

## CALLED BY:

group    routine

CASES      CSTA

**ZIPC****ZIPC**

This SUBROUTINE is number 31 in file UTCHAR.

Fill the character array X of size LEN with the given value ZIP.

## ARGUMENTS:

<u>order</u>	<u>name</u>	<u>type</u>	<u>size</u>	<u>status</u>	<u>declaration</u>	<u>explanation</u>
1	LEN	I*4		I		size of character array
2	ZIP	C*1		I		character to fill array
3	X	C*1	(V)	O		character array to be filled

## COMMON USAGE:

none

## CALLS:

none

CALLED BY:

group routine

CASES	CSATR					
MSIMPT	PRWMSI					
UTCHAR	DATCHR	DATLST	DECCHR	DPRCHR	INTCHR	PRTLNO
UTWDMF	WMSGTE					
WDATRB	WDSAGY					
WDIMEX	PRWMEX					

## **APPENDIX E. DATA STRUCTURES - COMMON BLOCKS**

---

This appendix provides detailed information on the common blocks used internally by WDM. Included are a general description of the common block's function, a list of parameters used to dimension arrays within the common block, and the attributes of variables and equivalences defined in the common block. Attributes defined are: type (integer, real, etc.), array dimensions, and a list of subroutines where the variable is used. For each subroutine where a common variable is used, the usage is classified as follows.

set - variable value is set

ref - variable is referenced

s/r - variable is both set and referenced

arg - variable is used as an argument to a subroutine

a/r - variable is used as an argument and referenced

asr - variable is used as an argument, referenced and set

### **CDRLOC**

---

This common block contains pointers to critical pieces of information on the file definition record. It is found in the include file named 'CDRLOC.INC'.

PDIRPT pointer to position where record pointer to first directory data set is stored  
set: WDBFIN  
ref: WDDRRC

PFNNAME pointer to position where name on wdm file is stored  
set: WDBFIN  
ref: WDRCAD WDRCDL WDRCGN

PMXREC pointer to position where number of records in wdm file is stored  
set: WDBFIN  
ref: WDCREA WDFLCK WDRCAD

PTSNUM pointer to position where count of timeseries datasets in wdm file is stored  
set: WDBFIN  
ref: PRWMEX WADDSSI WDDSRN WDFCUP

## CFBUFF

This common block contains buffers for in memory storage of records from WDM files. It is found in the include file named 'CFBUFF.INC'.

```
FREPOS next free position on the in memory record buffer
set: WDBFIN
s/r: WDRCGO
MAXREC maximum number of records on each available WDM file
set: WDBFIN WDCREA WDFLCK WDRCAD
ref: WDRCGO
s/r: WDFLCL
NXTPOS forward chain of records in the in memory record buffer
set: WDBFIN
s/r: WDRCGO
PREPOS backward chain of records in the in memory record buffer
set: WDBFIN
s/r: WDRCGO
RECNO WDM file record numbers for each record in the in memory record buffer
set: WDBFIN
ref: WADADI WDDRRC WDDSCL WDLANG WDLBAX WDNXPS WDRCAD WDRCP WDTBTM WMSANG WSTAGP WSTFGP
WTNBK
WTSKVX
s/r: WDRCGO
arg: WDFDUP
WDMCNT count of currently open WDM files
set: WDBFIN
ref: WDRCAD WDRCGO
s/r: WDCREA WDFLCK WDFLCL
WDMFUN Fortran file unit numbers for each record in the in memory record buffer
set: WDBFIN
s/r: WDFLCL WDRCGO
WDMOPN Fortran file unit numbers for each open WDM file
set: WDBFIN
ref: WDRCAD WDRCGO
s/r: WDCREA WDFLCK WDFLCL
WIBUFF integer in memory record buffer
set: WDCREA WDLBAX WDRCGO WMSADI WMSPTE WSTPTD WSTPTI WSTPTR
ref: PRWMEX PRWMTE WADDSSI WADGTL WADQCK WDCKDT WDDSCK WDDSDL WDLGET WDLLCK WDLSU WDNXDV
WDPRPS WDRCP WDSKBK WDTBFN WDTBFX WDTBSP WDTBSU WMSANG WMSBTR WMSGTE WMSQCK WSTFDT
WSTGSU
WSTGTD WSTGTI WSTGTR WSTWNT WTBGET WTGPCK WTGTVN
s/r: WADADI WDDRRC WDDSCL WDDSRN WDFCUP WDFDUP WDFLCK WDLANG WDLPUT WDRCAD WDRCDL WDRCGN
WDTBDL
WDTBTM WSTAGP WSTDGP WTBPW WTDDEL WTPTVL
arg: WDTGET WMSBCX
a/s: WDBSAC WDBSAI WDBSAR
a/r: WDBSGC WDBSGI WDBSGR WDSCHA WDTPFX WMSFBC WTDSCU WTFNDG WTFNDT
asr: WDBSAD WSTFGP WTBYFX WTSKVX
WRBUFF real in memory record buffer
set: WDBSAR WDLPUT WSTPTD WSTPTR WTBPUT WTDDEL WTPTVL
ref: PRWMEX PRWMTE WDBSGR WDLGET WSTGTD WSTGTR WTBGET WTFNDT WTGTVN
s/r: WTSKVX
arg: WTFNDG
```

## CWTSDS

This common block is used to store internal variables used by timeseries management routines. It is found in the include file 'CWTSDS.INC'.

CURBKS starting position of current block within current record  
arg: WDTPFX WTGTVL

CURCMP compression code of current block  
arg: WDTPFX WTGTVL

CURCNT current position within block  
arg: WDTPFX  
asr: WTGTVL

CURDAT date of start of current value  
arg: WDTPFX  
a/r: WTGTVL

CURNOV number of values in current block  
arg: WDTPFX WTGTVL

CURPOS position in current block  
arg: WDTPFX WTGTVL

CURQUA quality code of current block  
arg: WDTPFX  
a/r: WTGTVL

CURREC current record number  
arg: WDTPFX WTGTVL

CURTST current time step  
arg: WDTPFX  
a/r: WTGTVL

CURTUN current time units  
arg: WDTPFX  
a/r: WTGTVL

CURVAL current value  
arg: WDTPFX  
a/r: WTGTVL

PREVAL previous value  
arg: WDTPFX WTGTVL

## APPENDIX F.--RETURN CODES

\*\*\*\*\* describe what each one is and where it is set

CODE	SOURCE GROUP	ROUTINE	DESCRIPTION
- (ABS(FERR))	usys**.f77	WDBOPN	system specific file error number
-01	usys**.f77	WDBOPN	non specific error on WDM file open
-04	wdtms2.f77	WTDS CU	copy/update failed due to data overlap problem - part of source needed
-05	wdtms2.f77	WTDS CU	copy/update failed due to data overlap problem
-06	wdtms2.f77	WTFNDT	no data present
-08	wdtms1.f77	WTPMCK	bad dates
-09	wdtms1.f77	WTGPCK	data present in current group
-10	wdtms1.f77	WTSKVX	no data in this group
-11	wdtms1.f77	WTSKVX	no non missing data, data has not started yet
-14	wdtms1.f77	WTFNDG	date specified not within valid range for data set
-15	wdtms1.f77	WDT PFX	time units and time step must match label exactly with VBTIME=1
-20	wdtms1.f77	WTPMCK	problem with one or more of GPFLG, DXX, NVAL, QUALVL, LTSTEP, LTUNIT
-21	wdtms1.f77	WTSKVX	date from WDM doesn't match expected date
-23	wdtbl e.f77	WDTBSP	not a valid table
-24	wdtbl e.f77	WDTBSP	not a valid associated table
-25	wdtbl e.f77	WDTBTM	template already exists
-26	wdtbl e.f77	WDTBTM	can't add another table
-26	utwdmf.f77	WMSANG	no space for another question
-27	wdtbl e.f77	WDTBSU	no tables to return info about
-28	wdtbl e.f77	WDTBFX	table doesn't exist yet
-30	wdtbl e.f77	WTBPUT	more than whole table
-30	wdtbl e.f77	WTBGET	want more than whole table
-31	wdtbl e.f77	WTBPUT	more than whole extension
-31	wdtbl e.f77	WTBGET	want more than whole extension
-32	wdtbl e.f77	WTBPUT	data header doesn't match
-32	wdtbl e.f77	WTBGET	data header doesn't match
-33	wdtbl e.f77	WTBPUT	problems with row/space specs
-33	wdtbl e.f77	WTBGET	problems with row/space specs
-36	wdsptm.f77	WSTFGP	missing needed following data for a get
-37	wdsptm.f77	WSTFGP	no data present
-38	wdsptm.f77	WSTFGP	missing part of time required
-39	wdsptm.f77	WSTFGP	missing data group
-40	wdsptm.f77	WSTFGP	no data available
-41	wdsptm.f77	WSTFGP	no data to read
-42	wdsptm.f77	WSTAGP	overlap an existing group
-43	wdsptm.f77	WSTAGP	can't add another space time group
-44	wdsptm.f77	WSTWNT	trying to get/put more data than in block
-45	wdsptm.f77	WSTWNT	types don't match
-46	wdsptm.f77	WSTAGP	bad space time group specification parameter
-47	wdsptm.f77	WDSTCP	bad direction flag
-48	wdsptm.f77	WDSTCP	conflicting spec of space time dim and number of timeseries data sets
-49	wdsptm.f77	WSTGSU	group doesn't exist
-50	wddlg.f77	WDLGET	requested attributes missing from this data set
-51	wddlg.f77	WDLANG	no space for another DLG
-61	wdbl e.f77	WDD SCL	old data set does not exist
-62	wdbl e.f77	WDD SCL	new data set already exists
-71	wdbtch.f77	WDB CRL	data set already exists
-72	wdbtch.f77	WDD SRN	old data set does not exist
-73	wdbtch.f77	WDD SRN	new data set already exists
-81	utwdmd.f77	WDD SCK	data set does not exist
-82	utwdmd.f77	WDSCHA	data set exists, but is wrong DSTYP

## APPENDIX F.--RETURN CODES--continued

CODE	SOURCE GROUP	ROUTINE	DESCRIPTION
-83	utwdmd.f77	WDCREA	WDM file already open, can't create it
-84	utwdmd.f77	WDDSCK	data set number out of valid range
-85	utwdmd.f77	WDSCHA	trying to write to a read-only data set
-87	utwdmd.f77	WDFLCL	can't remove message WDM file from buffer
-88	utwdmd.f77	WDFLCK	can't open another WDM file
-89	utwdmd.f77	WDFLCK	check digit on 1st record of WDM file is bad
-101	wdatrb.f77	WDBSAC	incorrect character value for attribute
-102	wdatrb.f77	WDSASP	attribute already on label
-103	wdatrb.f77	WDSASP	no room on label for attribute
-104	wdatrb.f77	WDDPAR	data present, can't update attribute
-105	wdatrb.f77	WDDPAR	attribute not allowed for this type data set
-106	wdatrb.f77	WDDPAR	can't delete attribute, its required
-107	wdatrb.f77	WDSA FN	attribute not present on this data set
-108	wdatrb.f77	WDBSAI	incorrect integer value for attribute
-109	wdatrb.f77	WDBSAR	incorrect real value for attribute
-110	wdatrb.f77	WDSA FI	attributes not found on message file
-111	wdatrb.f77	WDSA FI	attribute name not found (no match)
-112	wdatrb.f77	WDSA FI	more attributes exist which match SAFNAM
-121	wdatms.f77	WADADI	no space for another attribute
+01	wddlg.f77	WDLGET	more data available for this DLG group
+01	wddlg.f77	WDLLSU	more groups available to summarize
+01	wdtble.f77	WDTBSU	more tables exist than we have space for
+01	wdsptm.f77	WSTAGP	ok to add group
+01	wdtble.f77	WDTBFN	table template not found
+02	wddlg.f77	WDLGET	no more data available for this DLG group

## **APPENDIX G. WDM FILE MAINTENANCE PACKAGE**

A stand alone utility program called WDMRX (strong medicine for an ailing WDM file) is available for diagnostic use when the integrity of a WDM file is in question. WDMRX checks file internals, writes a summary of the file contents, and optionally attempts to correct major errors. The user must supply the name of the WDM file, the output summary level, and whether the attempt corrections. The following output was produced when summarizing the MESSAGE.WDM file created for use by the case studies described in this manual. Detailed output from WDMRX is always written to file DUMP.OUT with summary information written to the screen. Comments have been added to explain the particular output.

<b>Comment</b>	<b>Output</b>												
<b>summary of user options</b>	OUTPUT LEVEL IS GROUP												
	UPDATE OPTION IS NO												
<b>general information about WDM file</b>	WDM FILE: message.wdm												
	MAXREC : 60												
	FREREC : 42												
<b>summary of data sets by type</b>	COUNT OF DATASETS BY TYPE												
	1-TIMESERIES: 0												
	2-TABLE : 0												
	3-SCHEMATIC : 0												
	4-PROJECT : 0												
	5-VECTOR : 1												
	6-RASTOR : 0												
	7-SPACE-TIME: 0												
	8-ATTRIBUTE : 6												
	9-MESSAGE : 1												
<b>dataset chains by type</b>	CHAIN OF VECTOR	DATASETS:	90										
	CHAIN OF ATTRIBUTE	DATASETS:	10	9	8	7	5	6					
	CHAIN OF MESSAGE	DATASETS:	12										
<b>directory records</b>	LOOKING FOR DIRECTORY RECORDS FOUND DIR REC: 1 ON PHYS REC 3 FOR DSN: 1 - 500 END SEARCH FOR DIRECTORY RECORDS												
<b>free record chain</b>	FREE REC CHN :	42 43 44 45 46 47 48 49 50 51 52 53											
		54 55 56 57 58 59 60											
<b>Comment</b>	<b>Output</b>												
<b>data set 5 details</b>	DSN: 5	TYPE: 8-ATTRIBUTE					LENGTH: 4						
	CHAIN: 6 5 4 2												
	FREE POSITION OF DATA: 2						113						

## APPENDIX G. WDM FILE MAINTENANCE PACKAGE--continued

<b>data set 6 details</b>	DSN:	6	TYPE: 8-ATTRIBUTE	LENGTH:	6						
		CHAIN:	7 8 9 10 11 12								
		FREE POSITION OF DATA:	12 80								
<b>data set 7 details</b>	DSN:	7	TYPE: 8-ATTRIBUTE	LENGTH:	7						
		CHAIN:	15 16 17 18 19 20 21								
		FREE POSITION OF DATA:	21 129								
<b>data set 8 details</b>	DSN:	8	TYPE: 8-ATTRIBUTE	LENGTH:	9						
		CHAIN:	13 14 22 23 24 25 26	27	28						
		FREE POSITION OF DATA:	28 157								
<b>data set 9 details</b>	DSN:	9	TYPE: 8-ATTRIBUTE	LENGTH:	6						
		CHAIN:	29 30 31 32 33 34								
		FREE POSITION OF DATA:	34 200								
<b>data set 10 details</b>	DSN:	10	TYPE: 8-ATTRIBUTE	LENGTH:	3						
		CHAIN:	35 36 37								
		FREE POSITION OF DATA:	37 368								
<b>data set 12 details</b>	DSN:	12	TYPE: 9-MESSAGE	LENGTH:	2						
		CHAIN:	38 39								
		FREE POSITION OF DATA:	39 73								
		CHECKING MESSAGE GROUPS									
		CLUSTER: CSTA									
		GROUP #: 1 IS AT 38 339									
		GROUP #: 2 IS AT 38 497									
		GROUP #: 3 IS AT 39 31									
<b>data set 90 details</b>	DSN:	90	TYPE: 5-VECTOR	LENGTH:	2						
		CHAIN:	40 41								
		FREE POSITION OF DATA:	41 309								
		CHECKING VECTOR GROUPS									
		GROUP #: 1 IS AT 40 461 ATTR: 1 0 0 1									
		GROUP #: 2 IS AT 40 484 ATTR: 1 180 201 1									
		GROUP #: 3 IS AT 41 129 ATTR: 1 180 208 1									
<b>orphan record check</b>	LOOKING FOR MISSING RECORDS IN CHAINS										
	MAP OF RECORD USAGE										
<b>codes:</b>	1 -	10:	-3	5	-1	5	5	5	6	6	6
<b>-3 - fdefn</b>	11 -	20:	6	6	8	8	7	7	7	7	7
<b>-2 - free</b>	21 -	30:	7	8	8	8	8	8	8	9	9
<b>-1 - direct</b>	31 -	40:	9	9	9	9	10	10	10	12	12
<b>0 - orphan</b>	41 -	50:	90	-2	-2	-2	-2	-2	-2	-2	-2
<b>&gt;0 - DSN</b>	51 -	60:	-2	-2	-2	-2	-2	-2	-2	-2	-2
	FOUND NO RECORDS MISSING FROM CHAINS										
<b>data set count check</b>	CHECKING DATASET COUNTS ALL DATASET COUNTS MATCH										

## APPENDIX H.--INDEX TO SUBROUTINES

The following is a complete list of the names of all subroutines contained in the WDM software. A short definition of the function of each subroutine also is included. The purposes of this list are two-fold: first, to prevent the application programmer from duplicating already-used routine names when naming new subroutines developed for the specific application program; and second, to alert the programmer to the capabilities of lower-level subroutines included in the WDM software. The programmer may obtain further details on these subroutines in the SYSDOC.OUT file contained on the system distribution disk. For each subroutine, the appendix provides information on the subroutine type, the file in which it resides on the distribution disk, the order number indicating where the subroutine occurs within the file, and a functional description. The subroutine types are:

E - main program  
S - subroutine  
I - integer function  
R - real function  
D - double precision function

NAME	TYP	GROUP	NUMB	DESCRIPTION
ASRTC	S	UTSORT	1	Sorts character strings.
ASRTI	S	UTSORT	2	Sorts integers.
ASRTIP	S	UTSORT	3	Sorts integers in their array.
ASRTR	S	UTSORT	4	Sorts decimal numbers.
ASRTRP	S	UTSORT	5	Sorts decimal numbers in their array.
BLDWRT	S	WDMEX	4	Write TBUFF to screen for program Messie.
CARVAR	S	UTCHAR	1	Convert a character*1 array of size LENA to a character variable of length LEN.
CASES	E	CASES	1	Demonstrate use of WDM library.
CHAIN	S	WDMRX	2	Determine chain of records.
CHDECE	S	UTCHAR	2	Convert a character array to its real equivalent.
CHINTE	S	UTCHAR	3	Convert a character array to its integer equivalent.
CHKDPR	S	UTNUMB	3	Check the double precision DVAL against the minimum and maximum values.
CHKINT	S	UTNUMB	1	Check the integer IVAL against the minimum and maximum values.
CHKMES	S	WDMRX	5	Check groups and blocks of a message data set.
CHKREA	S	UTNUMB	2	Check the real RVAL against the minimum and maximum values.
CHKSTR	I	UTCHAR	4	Search thru STR2 for a match to the character array STR1.
CHKTAB	S	WDMRX	4	Check groups and blocks of a table data set.
CHKTIM	S	WDMRX	3	Check groups and blocks of a timeseries data set.
CHKVEC	S	WDMRX	6	Check groups and blocks of a vector data set.
CHRCHR	S	UTCHAR	5	Copy LEN characters from array STR1 to array STR2.
CHRDEC	R	UTCHAR	6	Convert a character array to its real equivalent.
CHRDEL	S	UTCHAR	8	Delete the character in array position POS and shift the rest left one position.
CHRDIG	I	UTCHAR	9	Convert a single character to an integer.
CHRDPR	D	UTCHAR	7	Convert a character array to its double precision equivalent.
CHRINS	S	UTCHAR	10	Insert the character CHAR into position COL in the array STRING.
CHRINT	I	UTCHAR	11	Convert a character array to its integer equivalent.
CKDATE	S	UTDATE	1	Determine the calendar order of two dates. The dates are assumed to be valid.
CKNBLK	I	UTCHAR	12	Check character array CBUF for all blanks.
CMPTIM	S	UTDATE	2	Compare one time unit and step to a second time unit and step.
CNVDLG	E	CNVDLG	1	Convert nmd format dlg data to wdm export format.
COPYC	S	UTCHAR	13	Copy the character array ZIP of size LEN to the character array X of size LEN.
COPYD	S	UTNUMB	4	Copy the double precision array ZIP of size LEN to the double precision array X.
COPYI	S	UTNUMB	5	Copy the integer array ZIP of size LEN to the integer array X.
COPYYR	S	UTNUMB	6	Copy the real array ZIP of size LEN to the real array X.
CRINTE	I	UTCHAR	14	Convert a character array to its integer equivalent.
CRINTX	I	UTCHAR	15	Convert a character array to its integer equivalent.
CSATR	S	CASES	7	Case study example for attribute datasets.

## APPENDIX H. INDEX TO SUBROUTINES--continued

NAME	TYP	GROUP	NUMB	DESCRIPTION
CSDLG	S	CASES	4	Case study example for dlg datasets.
CSST	S	CASES	3	Case study example for space time datasets.
CSTA	S	CASES	6	Case study example for table datasets.
CSTS	S	CASES	2	Case study example for timeseries datasets.
CSTXT	S	CASES	5	Case study example for text and table template datasets.
CTRSTR	S	UTCHAR	16	Center the characters within the array TITLE.
CVARAR	S	UTCHAR	17	Convert a character variable of length LENV to a character array of size LENA.
DATCHK	S	UTDATE	3	Check DATE for valid entries.
DATCHR	S	UTCHAR	18	Put a date into a character array.
DATLST	S	UTCHAR	19	Put DATE into the character array DSTRNG in the format 1986 FEB. 14 10:30:00.
DATNXT	S	UTDATE	4	Adds or subtracts the time interval INTRVL from the current date and time DATE.
DAYMON	I	UTDATE	5	Return the number of days in the given month for the given year.
DECCHR	S	UTCHAR	20	Convert a real number to a character array.
DECCHX	S	UTCHAR	22	Right justifies the real number REAIN into the character array STR of size LEN.
DIGCHR	C	UTCHAR	23	Convert a single digit to a character.
DLIMIT	S	UTDATE	6	Depending on the value of FSLS, find earliest or latest date in array of dates.
DPRCHR	S	UTCHAR	21	Convert a double precision number to a character array.
INTCHR	S	UTCHAR	24	Convert the integer number INTIN to a character array.
JDMODY	S	UTDATE	7	Convert a julian day to month and day, leap year taken into account.
LENSTR	I	UTCHAR	25	Return the actual length of the character array, excluding trailing blanks.
LFTSTR	S	UTCHAR	26	Left justify characters in the array TITLE.
NUMPTS	S	UTDATE	8	Calculate the number of time steps between two dates.
PRADIT	S	UTIMPT	5	Put text from any id of attribute information on WDM file.
PRATIM	S	UTIMPT	4	Import attribute data information for an attribute group.
PRMSAE	S	UTEXPT	3	Export attribute type question.
PRMSFE	S	MSEXPT	5	Export file type group.
PRMSFI	S	MSIMPT	5	Import file type group.
PRMSIT	S	MSIMPT	6	Put text from any ID of information on WDM file.
PRMSME	S	MSEXPT	4	Export menu type group.
PRMSMI	S	MSIMPT	4	Import menu type group.
PRMSPA	S	MSIMPT	2	Import 1-dimensional and 2-dimensional type groups.
PRMSPE	S	MSEXPT	2	Export 1-dimensional and 2-dimensional type groups.
PRMSTA	S	MSEXPT	6	Export block with header (CHDR) and information on same line.
PRMSTE	S	MSEXPT	3	Export text type group.
PRMSTI	S	MSIMPT	3	Import text type group.
PRTLIN	S	UTCHAR	27	Convert DATE and an array of real numbers into a character array.
PRTLNO	S	UTCHAR	28	Convert DATE and an array of real numbers into a character array.
PRTSTR	S	UTCHAR	33	Write a character array to the given file unit.
PRWMAE	S	UTEXPT	2	Export attribute data.
PRWMIA	S	UTIMPT	3	Import attribute type question from sequential message file.
PRWMDE	S	UTEXPT	4	Export DLG type datasets.
PRWMDI	S	UTIMPT	2	Import DLG type datasets.
PRWMEX	S	WDIMEX	3	Export data sets (clusters) from WDM file to sequential file.
PRWMIM	S	WDIMEX	2	Import data sets (clusters) from sequential file to WDM file.
PRWMME	S	MSEXPT	1	Export message file clusters (data sets).
PRWMSI	S	MSIMPT	1	Import message file clusters (data sets).
PRWMTE	S	UTEXPT	1	Export timeseries data.
PRWMTI	S	UTIMPT	1	Import WDM timeseries data from sequential file to WDMSFL.
QUPCAS	S	UTCHAR	36	To convert a character string from lower case to upper case.
STRFND	I	UTCHAR	29	Return position in array STR of size LEN that array FSTR of size FLEN occurs.
STRLNX	I	UTCHAR	30	Return the number of characters in the array.
TIMADD	S	UTDATE	9	Add NVALS time steps to first date to compute second date.
TIMBAK	S	UTDATE	10	Subtract one time interval at the given units TCODE from DATE.
TIMCHK	I	UTDATE	11	Determine the calendar order of two dates.
TIMCNV	S	UTDATE	12	Convert date that uses midnight convention of 00:00 to the convention 24:00.
TIMCVT	S	UTDATE	13	Convert date that uses midnight convention of 24:00 to the convention 00:00.
TIMDFX	S	UTDATE	15	Calculate number of values between two dates, including units and time step.
TIMDIF	S	UTDATE	14	Calculate the number of time steps between two dates.
WADADI	S	WDATMS	3	Put attribute data set information on WDM message file.
WADDSI	S	WDATMS	6	Determine the data-set numbers containing the attribute data sets.
WADGDF	S	WDATMS	9	Get the default value for an attribute off the message file.

## APPENDIX H. INDEX TO SUBROUTINES--continued

NAME	TYP	GROUP	NUMB	DESCRIPTION
WADGDS	S	WDATMS	10	Get the description for an attribute off the message file.
WADGHL	S	WDATMS	11	Get length and starting record/pos of help info for attribute off message file.
WADGRA	S	WDATMS	7	Get the min and max values for an attribute off the message file.
WADGTL	S	WDATMS	5	Given attribute index, get type, length, data set usage, etc.
WADGTN	S	WDATMS	4	Given attribute index, return attribute name.
WADGVA	S	WDATMS	8	Get the valid values for an attribute off the message file.
WADNSA	S	WDATMS	2	Given a search attribute index, determine if it exists on the message file.
WADQCK	S	WDATMS	1	Given attribute offset for data set check to see if attribute already exists.
WATTCL	I	UTWDT1	24	Calculate a word containing parameters for attribute type data set.
WATTSP	S	UTWDT1	25	Split up a word containing parameters for attribute type data set.
WATTUS	S	UTWDT1	26	Split up word containing array of indicator flags for dataset attribute usage.
WATWDC	I	UTWDT1	27	Calculate a word containing parameters for attribute type data set.
WATWDS	S	UTWDT1	28	Split up a word containing parameters for attribute type data set.
WBCWCL	I	UTWDT1	4	Calculate a block control word for time-series type data.
WBCWSP	S	UTWDT1	3	Split up block control word for time-series data set to determine info stored.
WBCWSQ	S	UTWDT1	11	Split up block control word for time-series data set.
WDATCL	I	UTWDT1	6	Calculate a date in compressed format from its components (year- hour).
WDATCP	S	WDTMS2	6	Copies an old array date into a new one.
WDATSP	S	UTWDT1	5	Split up a WDMS date word.
WDBCRL	S	WDBTCH	1	Add a label to a wdmsfl.
WDBFIN	S	UTWDMD	13	Initialize pointers and counters for WDM buffer of records.
WDBOPN	S	USYSPR	1	Open a WDM file. File is opened as new or old, depending on value of RONWFG.
WDBSAC	S	WDATRB	2	Adds (or modifies) character search attribute on given dsn.
WDBSAD	S	WDATRB	5	Deletes search attribute on given dsn.
WDBSAI	S	WDATRB	3	Adds (or modifies) integer search attribute on given dsn.
WDBSAR	S	WDATRB	4	Adds (or modifies) real search attribute on given dsn.
WDBSGC	S	WDBTCH	2	Gets values of character search attribute for a dsn.
WDBSGI	S	WDBTCH	3	Gets the values of integer search attribute for a dsn.
WDBSGR	S	WDBTCH	4	Get the values of real search attribute for a data set.
WDBSGX	S	WDATRB	12	Routine gets information about search attribute from message file.
WDCKDT	I	UTWDMD	11	Check data set for existance and type.
WDCREA	S	UTWDMD	14	Adds directory record and 19 empty records to a new WDM file.
WDDPAR	S	WDATRB	10	Determines if either data present and attrib can't update or attrib not allowed.
WDDRRC	I	UTWDMD	8	Determine WDM file directory record number for data-set number DSN.
WDDSCK	S	UTWDMD	12	Check data set existance and return record number of first record in data set.
WDDSCL	S	WDLBLE	1	Copies an old data-set label into a new data-set label.
WDDSDL	S	WDBTCH	5	Routine to delete a data set from the WDMSFL with no user interact.
WDDSRN	S	WDBTCH	6	Routine to renumber data sets with no user interaction.
WDDTFG	I	WDATRB	7	Determines if data is present in a WDMS data set.
WDFCUP	S	WDLBLE	3	Updates file definitions record data set counters and pointers.
WFDUP	S	WDLBLE	2	Updates a WDMS file directory record.
WDFLCK	S	UTWDMD	2	Check directory of WDM for major errors.
WDFLCL	S	UTWDMD	3	Remove a WDM file from the open WDM buffer and adjust buffer accordingly.
WDIMEX	E	WDIMEX	1	Import/export data sets of all types to/from a WDM file.
WDLANG	S	WDDLG	1	Add new DLG group to WDM file.
WDLBAD	S	WDLBLE	5	Add new data set label, but no search attrib or data, use default label sizing.
WLBAX	S	WDLBLE	6	Add new data-set label, but no search attrib or data.
WDLBCV	I	UTWDT1	22	Calculate a block control word for vector (DLG) type data set.
WDLBSP	S	UTWDT1	23	Split up a block control word for vector (DLG) type data set.
WDLGET	S	WDDLG	3	Retrieve DLG header or coordinate pairs from WDM file.
WDLICV	I	UTWDT1	20	Calculate a word containing parameters for a vector (DLG) type data set.
WDLISP	S	UTWDT1	21	Split up a word containing parameters for a vector (DLG) type data set.
WDLLCK	S	WDDLG	4	Search for group with desired data type and attributes.
WDLLSU	S	WDDLG	5	Summarize label information for DLG data set.
WDLPUT	S	WDDLG	2	Store DLG header or coordinate pairs on WDM file.
WDMRX	E	WDMRX	1	Strong medicine for an ailing wdm file.
WDNXDV	S	UTWDMF	5	Move to the next data position and return the integer equivalent of data value.
WDNXPS	S	UTWDMF	4	Get the next data position on a WDM file.
WDPRPS	S	UTWDMF	6	Get the previous data position on a WDM file.
WDPTCL	I	UTWDT1	1	Calculate a pointer value from record number and the offset within the record.
WDPTSP	S	UTWDT1	2	Split up a pointer into record number and offset within record.

# APPENDIX H. INDEX TO SUBROUTINES--continued

NAME	TYP	GROUP	NUMB	DESCRIPTION
WDRCAD	S	UTWDMD	7	Add NUMADD records to the WDM file. Update directory record on the WDM file.
WDRCDL	I	WDLBLE	4	Deletes a record in the WDMSFL and updates pointers as required.
WDRCGN	I	UTWDMD	5	Get the next free record from the WDM file and add it to WDM buffer of records.
WDRCGO	I	UTWDMD	1	Determine index of user requested record in WDM in memory buffer of records.
WDRCGX	I	UTWDMD	6	Get the next free record from the WDM file.
WDRCUP	S	UTWDMD	4	Write record index number RIND from the buffer of records to the WDM file.
WDSAIFI	S	WDATRB	11	Find the first attribute name which matches SAFNAM.
WDSAFL	S	WDATRB	8	Determines where values for particular search attribute start in data-set label.
WDSAGY	S	WDATRB	1	Gets general detail information about specified attribute.
WDSASP	S	WDATRB	6	Adds space for search attribute on a dsn label if not present on it.
WDSASV	I	WDATRB	9	Determines where values for particular search attribute start in data-set label.
WDSCHA	S	UTWDMD	10	Check WDM data set existence, type and ability to update.
WDSCHK	S	UTWDMD	9	Check WDM data set existence and type.
WDSKBK	S	WDTMS1	10	Skips to next WDMSFL block.
WDTBOL	S	WDTBLE	11	Delete WDM table from WDM file.
WDTBFN	S	WDTBLE	3	Determines pointer to specified table.
WDTBFX	S	WDTBLE	2	Determines pointer to specified table and returns its cluster and group number.
WDTBSP	S	WDTBLE	4	Get WDM table specifications from message file.
WDTBSU	S	WDTBLE	6	Get WDM table label info from WDM file table data set.
WDTBTM	S	WDTBLE	1	Put WDM table template on WDM file, return parameters about table.
WDTGET	S	WDTMS1	1	Gets timeseries information from the WDMSFL.
WDTPPF	S	WDTMS1	13	Puts timeseries information into the WDMSFL.
WDTPUT	S	WDTMS1	17	Puts time series data into a WDM file.
WMSADI	S	UTWDMF	3	Add portion of group-number GNUM information to DSN.
WMSANG	S	UTWDMF	2	Check that data-set DSN exists and there is space in data set for new group.
WMSBCS	S	UTWDT1	13	Split up a block control word for a message type data set into its components.
WMSBCV	I	UTWDT1	12	Calculate a block control word for message type data set.
WMSBCX	S	UTWDMF	11	Given pointer to a block control word, return its record number, position, etc.
WMSBTR	S	UTWDMF	9	Back up NREC text records in a text group and reset the pointers for record.
WMSFBC	S	UTWDMF	12	Get first block control word and position for group GNUM in message data set.
WMSGTE	S	UTWDMF	8	Get one record of text off WDM file.
WMSGTO	S	UTWDMF	10	Get and write text to a sequential file until end of text is reached.
WMSIDP	S	UTWDMF	13	Return the record number and position on the WDM file for the given ID.
WMSMNS	S	UTWDT1	15	Split up word containing integer parameters for a message type data-set menu.
WMSMNV	I	UTWDT1	14	Calculate word containing parameters for a message type data-set menu.
WMSP2S	S	UTWDT1	19	Split up word containing integer parameters for message type data-set PRM2 scr.
WMSP2V	I	UTWDT1	18	Calculate word containing parameters for message type data-set PRM2 class scr.
WMSPIS	S	UTWDT1	17	Split up word into two integer values.
WMSPIV	I	UTWDT1	16	Calculate word from two integer values (1st value- 16 bits, 2nd- 15 bits).
WMSPTE	S	UTWDMF	7	Add one record of text to WDM file.
WMSQCK	S	UTWDMF	1	Check to see if a group already exists in a data set on WDM file.
WMSSKB	S	UTWDMF	14	Position DREC and DPOS at the end of the current data block.
WSTAGP	S	WDSPTM	3	Add a group to a space time data set, physically allocate space for it.
WSTDGP	S	WDSPTM	4	Delete group from a space time data set, free up space it uses.
WSTDIM	S	WDSPTM	5	Determine dimensions a space time data set.
WSTFDT	S	WDSPTM	2	Find start position of data from group pointer.
WSTFGP	S	WDSPTM	1	Find space time data within a data set, return pointers to it.
WSTGCL	I	UTWDT1	29	Calculate a block control word for space-time type data set.
WSTGSP	S	UTWDT1	30	Split up a block control word for space-time type data set.
WSTGSU	S	WDSPTM	6	Summarize a group in a space time data set.
WSTGTD	S	WDSPTM	10	Get double precision space time data.
WSTGTI	S	WDSPTM	8	Get integer space time data.
WSTGTR	S	WDSPTM	9	Get real space time data.
WSTPTD	S	WDSPTM	13	Put double precision space time data.
WSTPTI	S	WDSPTM	11	Put integer space time data.
WSTPTR	S	WDSPTM	12	Put real space time data.
WSTSCP	S	WDSPTM	14	Copy from WDM space time data set to timeseries data sets or vv.
WSTWNT	S	WDSPTM	7	Determine next wdm space time data offset (from beginning of data).
WTBCOD	S	WDTBLE	7	Convert data from full screen buffer into WDM internal format.
WTBDCD	S	WDTBLE	8	Convert data from WDM internal format into full screen buffer.
WTBDCL	I	UTWDT1	9	Calculate the table dimension for a table type data set.

# APPENDIX H. INDEX TO SUBROUTINES--continued

NAME	TYP	GROUP	NUMB	DESCRIPTION
WTBDSP	S	UTWDT1	10	Split up dimension variable for table type data set into its components.
WTBGET	S	WDTBLE	10	Get the main table data.
WTBICL	I	UTWDT1	7	Calculate the identifier for a table type data set.
WTBISP	S	UTWDT1	8	Split up an identifier for a table type data set into its components.
WTBPUT	S	WDTBLE	9	Put WDM table data into WDM file.
WTBSPA	S	WDTBLE	5	Calculates space required for a row in table data set and count types of data.
WTBYFX	S	WDTMS1	6	Add base year attribute to a timeseries data set.
WTDATE	S	WDTMS2	7	Find common period with data from a list of time-series data sets on a WDM file.
WTDEL	S	WDTMS2	3	Delete all data following a specified date in the given data set.
WTDS CU	S	WDTMS2	2	Copy from a data set to another any timeseries data between start and end dates.
WTDSPM	S	WDTMS1	5	Obtains values for a variety of TIMSER parms from labels or defaults.
WTDSPX	S	WDTMS1	7	Obtains values for a variety of timeseries parms from labels or defaults.
WTGRP	S	WDTMS2	4	Determines end of group which contains a given date.
WTFNDG	S	WDTMS1	4	Check the data set, computes ending date, start and end group pointers, etc.
WTFN DT	S	WDTMS2	1	Determine starting and ending dates of data in data set.
WTGPCK	S	WDTMS1	14	Checks information related to group, skip to start value, fill in current info.
WTGT NV	S	WDTMS1	12	Routine to get the next value from a WDS timeseries DSN.
WTGTVL	S	WDTMS1	11	Fills in RVAL array with data values from WDMS DSN.
WTNW BK	S	WDTMS1	16	Starts a new WDMS timeseries block, on a new record if req.
WTPM CK	S	WDTMS1	3	Checks the parameters supplied to either WDTPUT or WDTGET.
WTPT VL	S	WDTMS1	15	Writes all or part of a WDMS group into a WDMS timeseries data set.
WTSC SC	S	WDTMS1	2	Converts the given time units and time step to seconds.
WTSG RP	S	WDTMS2	5	Determines start of group which contains given date.
WTSK VL	S	WDTMS1	8	Skips values within a WDMSFL timeseries group.
WTSK VX	S	WDTMS1	9	Skips values within a WDMSFL timeseries group.
ZIPC	S	UTCHAR	31	Fill the character array X of size LEN with the given value ZIP.
ZIPD	S	UTNUMB	7	Fill the double precision array X of size LEN with the given value ZIP.
ZIPI	S	UTNUMB	8	Fill the integer array X of size LEN with the given value ZIP.
ZIPR	S	UTNUMB	9	Fill the real array X of size LEN with the given value ZIP.
ZJUST	S	UTCHAR	32	Remove leading blanks from a character variable.
ZLNTXT	I	UTCHAR	34	Determine length of text in a character variable.
ZTRIM	S	UTCHAR	35	Remove embeded blanks in a character variable.

# APPENDIX I. DLG CONVERSION UTILITY

\*\*\*\*\* need to put stuff describing how to convert optional format dlg data to export format somewhere, an appendix?

The input data for this example are digital line graph data in the National Mapping Division DLG format; the data represent a railroad network in the Chickamauga TN area; the file format for the external data file is illustrated in this excerpt.

## Excerpt

USGS-NMD DLG DATA - CHARACTER FORMAT - 09-29-82 VERSION  
CHICKAMAUGA, GA AL TN 01 1981, 100000. F01  
100K 4059 R01.RR EMC86 4 40 0  
3 1 16 2 0.25400000000D+01 4 0 4 1  
-0.850520300000000D+08 0.340520300000000D+08 0.000000000000000D+00  
0.000000000000000D+00 0.000000000000000D+00 0.000000000000000D+00  
0.000000000000000D+00 0.000000000000000D+00 0.000000000000000D+00  
0.000000000000000D+00 0.000000000000000D+00 0.000000000000000D+00  
0.000000000000000D+00 0.000000000000000D+00 0.000000000000000D+00  
0.10000000000D+01 0.00000000000D+00 0.00000000000D+00 0.00000000000D+00  
SW 34.750000 -86.000000 591534.17 3845580.49  
NW 35.000000 -86.000000 591256.91 3873300.74  
NE 35.000000 -85.750000 614069.78 3873558.63  
SE 34.750000 -85.750000 614415.64 3845836.61  
RAILROADS 0 29 29 010 9 9 010 36 36 1  
N 1 591534.17 3845580.49 2 0 0  
28 -30  
N 2 591256.91 3873300.74 2 0 0  
1 -28  
\*\*\* (records deleted)  
A 5 602818.50 3859568.47 3 0 0 0 0  
4 2 -3  
A 6 602818.50 3859568.47 3 0 0 0 0  
14 -9 -12  
A 7 602818.50 3859568.47 3 0 0 0 0  
12 -10 -11  
A 8 602818.50 3859568.47 3 0 0 0 0  
21 22 20  
A 9 602818.50 3859568.47 3 0 0 0 0  
36 -25 35  
L 1 2 5 1 2 2 0 0  
591256.91 3873300.74 599976.35 3873385.63  
L 2 7 6 3 5 2 2 0  
613107.83 3865320.27 613840.85 3866098.14  
180 201 181 2  
L 3 8 6 5 4 3 1 0  
613479.74 3865680.05 613813.56 3866039.41 613840.85 3866098.14  
180 208  
L 4 8 7 4 5 4 1 0  
613479.74 3865680.05 613206.20 3865382.33 613128.12 3865323.04  
613107.83 3865320.27  
180 208  
L 5 9 7 3 4 2 2 0  
613047.64 3865251.01 613107.83 3865320.27  
180 201 181 2  
L 6 10 9 4 4 6 1 0  
612953.52 3864581.89 612921.96 3864678.07 612932.87 3864838.22  
612982.13 3864975.94 613025.23 3865210.12 613047.64 3865251.01  
180 208  
\*\*\* (records deleted)

The following pages contain code segments and descriptions for a program called CSDLG. The program reads the digital line graph data shown above and transforms the data to a format that will allow subsequent storage in the WDM file.

As described above, the computer program CSDLG produces two new files: (1) a WDM sequential import/export file containing the reformatted DLG data and (2) a file that reports the number of "lines" and "areas" successfully saved in the import/export file. The contents of the two output files are shown below; the import/export file illustration is an excerpt from a larger file.

Program CNVDLG

```
C
C
C
C      PROGRAM  CNVDLG
C
C      + + + PURPOSE + +
C      convert nmd format dlg data to wdm export format
C
C      + + + LOCAL VARIABLES + +
C      INTEGER      FL,I,J,K,NLIN,NATTR,FOU,DSN,
C      2          LINID(2000),AL,AID,IH,IA,IB,IC,
C      3          OLIN,TLIN,LCNT,IAT1,IAT2,ITMP,ILEN,
C      4          LINSTR(11000),IFR,ILS,CPTS,DIR,IX,
C      5          ACNT,AREATR(500),ARECNT(500),AREVAL(25,500),
C      6          AATMIN,AATMAX,LAT1MN,LAT1MX,LAT2MN,LAT2MX,
C      7          LINAT1(11000),LINAT2(11000),AATCNT,LATCNT
C      REAL        TX(4800),TY(4800),XMIN,XMAX,YMIN,YMAX,
C      1          AX(1000),AY(1000)
C      CHARACTER*1 ID
C      CHARACTER*8 FNAME
C      CHARACTER*80 BUFF
C
C      COMMON /SCR/LINX,LINY
C      REAL        LINX(37000),LINY(37000)
C
C      + + + INTRINSICS + +
C      INTRINSIC ABS
C
C      + + + INPUT FORMATS + +
C      1000 FORMAT (A80)
C      1010 FORMAT (12I6)
C      1020 FORMAT (A1,41X,2I6)
C      1030 FORMAT (6F12.2)
C      1060 FORMAT (A8)
C      1070 FORMAT (A1,I5,2F12.2,6X,I6,6X,I6,I6)
C
C      + + + OUTPUT FORMATS + +
C      2000 FORMAT (' XMIN,XMAX:',2F12.0,' YMIN,YMAX:',2F12.0)
C      2010 FORMAT ('GROUP TYPE LINE  MAJ ATTR',I5,' MIN ATTR',I5)
C      2020 FORMAT (' 1',I5,I6,2F12.2)
C      2030 FORMAT (' 2',I5)
C      2040 FORMAT (6F12.2)
C      2070 FORMAT ('DATE',//,'WDM$FL',//,'SYSTEM',//,'COMMENT',//,'END COMMENT',
C      1          //,'DSN      ',I6,' TYPE VECT  NDN  1  NUP  1',
C      2          '      NSA 10  NSP 20  NDP 400',
C      3          //,' LABEL',//,' TSPREC      0',
C      4          //,' END LABEL',
C      5          //,' DATA')
C      2080 FORMAT (' END DATA')
C
```

```

C      + + + END SPECIFICATIONS + + +
C
C      WRITE(*,*) 'enter base file name '
C      READ(*,1060) FNAME
C      FOU= 40
C      OPEN (UNIT=FOU,FILE=FNAME//'.MSG')
C      FL = 41
C      OPEN (UNIT=FL,FILE=FNAME//'.DLG',STATUS='OLD')
C
C      AATMIN= 100000
C      AATMAX= -100000
C      AATCNT= 0
C      LAT1MN= 100000
C      LAT1MX= -100000
C      LAT2MN= 100000
C      LAT2MX= -100000
C      LATCNT= 0
C      TLIN = 0
C      ACNT = 0
50    CONTINUE
      READ (FL,1000,END=60,ERR=50) BUFF
      IF (BUFF(1:1).EQ.'L') THEN
C          a line header, get details
          READ (BUFF,1020) ID,NLIN,NATTR
C          a line
          LCNT= LCNT+ 1
          OLIN= TLIN+ 1
          LINSTR(LCNT)= OLIN
          TLIN= TLIN+ NLIN
          READ (FL,1030) (LINX(I),LINY(I),I=OLIN,TLIN)
          IF (NATTR.GT.0) THEN
C              what are these attributes
              READ (FL,1010) LINAT1(LCNT),LINAT2(LCNT)
              IF (LINAT1(LCNT).GT.LAT1MX) LAT1MX= LINAT1(LCNT)
              IF (LINAT1(LCNT).LT.LAT1MN) LAT1MN= LINAT1(LCNT)
              IF (LINAT2(LCNT).GT.LAT2MX) LAT2MX= LINAT2(LCNT)
              IF (LINAT2(LCNT).LT.LAT2MN) LAT2MN= LINAT2(LCNT)
              LATCNT= LATCNT+ 1
          ELSE
C              no attribute available
              LINAT1(LCNT)= -1
              LINAT2(LCNT)= -1
          END IF
          ELSE IF (BUFF(1:1).EQ.'A') THEN
C              an area header, get details
              ACNT= ACNT+ 1
              READ (BUFF,1070) ID,AID,AX(ACNT),AY(ACNT),ARECNT(ACNT),NATTR
              IF (ARECNT(ACNT).GT.25) THEN
C                  too many lines make up this area
                  WRITE (*,*) '**** ARECNT: ',ARECNT(ACNT)
                  READ (FL,1010) (LINID(I),I=1,ARECNT(ACNT))
                  ACNT= ACNT- 1
                  READ (FL,1010) ITMP,IX
                  WRITE (*,*) '      SKIP: ',ITMP,IX
                  WRITE(FOU,'') '      SKIP: ',ITMP,IX
              ELSE
C                  an valid area header
                  READ (FL,1010) (AREVAL(I,ACNT),I=1,ARECNT(ACNT))
                  IF (NATTR.GT.0) THEN
C                      attribute available
                      READ (FL,1010) AREATR(ACNT),IX
                      IF (IX.NE.0) THEN
C                          whats this second attribute

```

```

        WRITE(FOU,*) 'second attribute:',IX
        WRITE(*,*) 'second attribute:',IX
    END IF
    IF (AREATR(ACNT).LT.AATMIN) AATMIN= AREATR(ACNT)
    IF (AREATR(ACNT).GT.AATMAX) AATMAX= AREATR(ACNT)
    AATCNT= AATCNT+ 1
    ELSE
        C      no attribute
        AREATR(ACNT)= -1
    END IF
    END IF
    GO TO 50
C
    60  CONTINUE
C      all done
    LINSTR(LCNT+1)= TLIN+ 1
    CLOSE(UNIT=FL)
C
    WRITE (FOU,*) 'lines saved:',LCNT,' with attributes:',LATCNT
    WRITE (*,*) 'lines saved:',LCNT,' with attributes:',LATCNT
    WRITE (FOU,*) 'areas saved:',ACNT,' with attributes:',AATCNT
    WRITE (*,*) 'areas saved:',ACNT,' with attributes:',AATCNT
    XMIN = 1.0E30
    YMIN = 1.0E30
    XMAX = -1.0E30
    YMAX = -1.0E30
C
    DO 70 I= 1,TLIN
    IF (LINX(I).GT.XMAX) XMAX= LINX(I)
    IF (LINX(I).LT.XMIN) XMIN= LINX(I)
    IF (LINY(I).GT.YMAX) YMAX= LINY(I)
    IF (LINY(I).LT.YMIN) YMIN= LINY(I)
    70  CONTINUE
C
    WRITE (FOU,2000) XMIN,XMAX,YMIN,YMAX
    WRITE (*,2000) XMIN,XMAX,YMIN,YMAX
C
    OPEN (UNIT=FL,FILE=FNAME//'.EXP',STATUS='NEW',ERR=900)
C      write header
    WRITE (*,*) 'enter dsn for dlg in export format: '
    READ(*,*) DSN
    WRITE (FL,2070) DSN
C
    IF (AATCNT.GT.0) THEN
        C      process areas with attributes
        IAT1= 0
        IAT2= 0
        WRITE (*,*) 'looking for areas:',AATMIN,AATMAX
        DO 190 IH= AATMIN,AATMAX
            IA= 0
            DO 180 IC= 1,ACNT
                IF (AREATR(IC).EQ.IH) THEN
                    IA= IA+ 1
                    IF (IA.EQ.1) THEN
                        IAT1= IH
                        WRITE(*,*) 'attribute:',IAT1,IAT2
C                      we need to write type 1 info
                        WRITE(FL,2010) IAT1,IAT2
                        AL= 3
                        WRITE(FL,2020) AL,IAT1,AX(IC),AY(IC)
                    END IF
                    WRITE(*,*) ' for area:',IC,' with:',ARECNT(IC),' lines'
C

```

```

CPTS= 0
DO 130 I= 1,ARECNT(IC)
      put lines into buffer
      J= AREVAL(I,IC)
      IF (J.NE.0) THEN
          not an island, save points
          IF (J.LT.0) THEN
              go last to first
              J = ABS(J)
              IDIR= -1
              IFR = LINSTR(J+1)- 1
              ILS = LINSTR(J)
          ELSE IF (J.GT.0) THEN
              first first
              IDIR= 1
              IFR = LINSTR(J)
              ILS = LINSTR(J+1)- 1
          END IF
          IF (I.GT.1) THEN
              dont need first point
              IFR = IFR+ IDIR
          END IF
          DO 110 K= IFR,ILS,IDIR
              CPTS= CPTS+ 1
              TX(CPTS)= LINX(K)
              TY(CPTS)= LINY(K)
CONTINUE
110
      END IF
      WRITE (*,*) 'line:',I,LINID(I),J,IFR,ILS,IDIR,CPTS
      IF (J.EQ.0 .OR. I.EQ.ARECNT(IC)) THEN
          IF (CPTS.GT.5) THEN
              enough points to define area
              IFR= 1
CONTINUE
115
              ILEN= CPTS
              IF (ILEN.GT.1200) THEN
                  only write part to fit in a dig buffer
                  ILEN= 1200
                  WRITE(FOU,*) 'part buffer:',IAT1,CPTS
              END IF
              ILS= IFR+ ILEN- 1
              IF (IFR.GT.1) THEN
                  WRITE(FOU,*) 'more buffer:',IFR,ILS,ILEN
              END IF
              write out line coordinates
              WRITE(FL,2030) ILEN*2
              WRITE(FL,2040) (TX(K),TY(K),K=IFR,ILS)
              WRITE(*,*) 'output line:',IAT1,IAT2,I,ILEN
C
              see if more to write
              CPTS= CPTS- ILEN+ 1
              IFR = ILS
              IF (CPTS.GT.1) GO TO 115
ELSE
      skip line, not enough points
      WRITE (FOU,*) 'skip',CPTS
      WRITE (*,*) 'skip',CPTS
END IF
C
      start at new position
      CPTS= 0
END IF
130      CONTINUE
END IF
180      CONTINUE
190      CONTINUE

```

```

        END IF
C
C      IF (LATCNT.GT;0) THEN
C        process lines with attributes
C        WRITE(*,*) 'looking for lines:',LAT1MN,LAT1MX,LAT2MN,LAT2MX,LCNT
C        DO 250 IA=LAT1MN,LAT1MX
C          DO 240 IB= LAT2MN,LAT2MX
C            IX= 0
C            DO 230 IC= 1,LCNT
C              IF (LINAT1(IC).EQ.IA .AND. LINAT2(IC).EQ.IB) THEN
C                got one that matches
C                IX= IX+ 1
C                IF (IX.EQ.1) THEN
C                  first match
C                  WRITE(*,*) 'attribute:',IA,IB
C                  we need to write type 1 info
C                  WRITE(FL,2010) IA,IB
C                  a handle at the mid point of the first line
C                  AL= 3
C                  K = (LINSTR(IC)+ LINSTR(IC+1))/2
C                  WRITE(FL,2020) AL,IA,LINX(K),LINY(K)
C                END IF
C                write out line coordinates
C                WRITE(FL,2030) (LINSTR(IC+1)- LINSTR(IC))* 2
C                WRITE(FL,2040) (LINX(K),LINY(K),
C                                K-LINSTR(IC),LINSTR(IC+1)-1)
C
C              1           END IF
C 230            CONTINUE
C 240            CONTINUE
C 250            CONTINUE
C            END IF
C
C            WRITE (FL,2080)
C            CLOSE (UNIT=FL)
C
C            GOTO 920
C            errors on file opens
C 900            CONTINUE
C            WRITE (*,*) *** ERROR ON OPEN OF OUTPUT DLG FILE ***
C 920            CONTINUE
C
C            STOP
C            END

```

**Export format  
DLG data**

DATE  
WDMSPL  
SYSTEM  
COMMENT  
END COMMENT  
DSN 99 TYPE VECT NDN 1 NUP 1 NSA 10 NSP 20  
NDP 400  
LABEL  
TSPREC 0  
END LABEL  
DATA  
GROUP TYPE LINE MAJ ATTR 0 MIN ATTR 0  
1 3 0 602818.50 3859568.50  
2 16  
591534.19 3845580.50 596116.06 3845629.25 614415.62 3845836.50  
614159.69 3866434.50 614069.75 3873558.75 599976.37 3873385.75  
591256.94 3873300.75 591534.19 3845580.50  
GROUP TYPE LINE MAJ ATTR 180 MIN ATTR 201  
1 3 180 613840.87 3866098.25  
2 4  
613107.81 3865320.25 613840.87 3866098.25  
2 4  
613047.62 3865251.00 613107.81 3865320.25  
2 8  
606622.19 3859245.25 607422.25 3859932.50 611278.00 3863359.00  
613047.62 3865251.00

**Summary Output  
From CNVDLG**

lines saved: 36 with attributes: 29  
areas saved: 9 with attributes: 1  
XMIN,XMAX: 591257. 614416.  
YMIN,YMAX: 3845580. 3873559.



