Automated Data Access and Analysis System
Program Manual
Version 12-19-2001

Section 1
INTRODUCTION AND OVERVIEW

ADAAS, the Automated Data Access and Analysis System, is a set of
computer programs maintained by the Transportation Data Center (TDC) at
the University of Michigan's Transportation Research Institute (UMTRI).
The programs are written in FORTRAN 77, and are resident on the
Transportation Data Center Computer operating under the Sun Microsystems
Solaris UNIX operating system. ADAAS is not a statistical analysis
package like SAS or SPSS for example although it does possess some
rudimentary statistical capability. It is intended instead as a means of
accessing the large collection of transportation-related data sets
maintained by the Transportation Data Center; and as a tool for
performing preliminary analysis tasks on selected data as an integrated
part of the system.

ADAAS performs four major functions for system users. First, and
probably most important, access to the collection of data sets
maintained by the Data Center is provided through the entry of an eight-
character "data keyword." On-line techniques for finding the desired
keyword are available in a HELP command. Data access is provided without
any requirement for information about file names or directories.

With data set access accomplished, several manipulation operations can
be performed on the selected information. Historically popular
operations are the two-way table (TWOWAY) and the data set list (LIST)
that provide preliminary looks at complex relationships that may exist.

After the analytic operations have been performed, the generated output
may be downloaded and printed on the user's computer or, in some
situations, on the printer located at TDC.

Since many problems require statistical computing power not available in
ADAAS, a method of reducing data sets to a manageable size (SUBSET) is
provided. Output from the subset program, in ASCII format, can be
loaded into most statistical packages.

This manual describes the operation of ADAAS. To obtain a copy, call
the Transportation Data Center at (734)-763-3230, or send your email
request to tdc-umtri@umich.edu. A Microsoft WORD document file
containing the latest version of the manual is also available in the
file "$ADAASHOME/doc/adaas.doc.doc".

The remainder of this section presents a brief overview of system design
and capabilities. In Section 2, the component files that comprise an
ADAAS data set are discussed in detail. Section 3 presents the
techniques for operating ADAAS from the terminal and elaborates on the
useful technique of setup files. Section 4 presents the detailed
syntactical requirements for the analysis commands, including the use of
FILTER and RECODE statements. Finally, Section 5 presents comprehensive
documentation for each of the available ADAAS commands.
1.1) PROGRAM EXECUTION AND CONTROL

Execution of ADAAS is initiated by the UNIX operating system command "adaas" and is terminated by the ADAAS command "STOP". The UNIX operating system is case sensitive and the program name "adaas" must be typed in lower case. Once ADAAS is executing, most commands and command options may be entered in either upper or lower case with the exception of system library data keywords, which must be upper case (e.g., "CDS94INJ").

Program control is achieved by entering a command statement in response to the program request. There are four general groups of commands:

1) Data commands to provide, or release, access to data sets in the system library, or in any other properly permitted library.

2) Analysis or data manipulation commands to derive information from the selected data sets.

3) Information commands to provide on-line documentation of program input requirements and data set characteristics.

4) Miscellaneous commands to provide utilities to the user.

A summary of the commands available in ADAAS is given in the following table. Four of the commands are "analysis" or "data manipulation" commands (LIST, SUBSET, TWOWAY, and UNIVAR). They require the input of additional information following the command statement such as a CASEVAR, DUPVAR, FILTER, RECODE, TITLE, and OPTIONS statement. A description of the information required by the analysis commands is presented in Section 4.

<table>
<thead>
<tr>
<th>COMMAND</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>CODES</td>
<td>Display code value labels (i.e., on-line codebook)</td>
</tr>
<tr>
<td>DATA</td>
<td>Provide access to system or private data sets</td>
</tr>
<tr>
<td>FILES</td>
<td>Display data set parameters (e.g., file names, paths, etc.)</td>
</tr>
<tr>
<td>HELP</td>
<td>Display documentation and system information</td>
</tr>
<tr>
<td>LIST</td>
<td>List selected records and variables from a data set</td>
</tr>
<tr>
<td>NEWS</td>
<td>Display current changes to the system or to data sets</td>
</tr>
<tr>
<td>RELEASE</td>
<td>Release access to the current data set</td>
</tr>
<tr>
<td>SET</td>
<td>Change parameters of program operation</td>
</tr>
<tr>
<td>STOP</td>
<td>Terminate program execution and return to the UNIX OS</td>
</tr>
<tr>
<td>SUBSET</td>
<td>Generate a new data set with selected records and variables</td>
</tr>
<tr>
<td>TWOWAY</td>
<td>Generate bivariate (twoway or crosstab) frequencies</td>
</tr>
<tr>
<td>UNIVAR</td>
<td>Generate univariate frequencies and simple statistics</td>
</tr>
</tbody>
</table>

1.2) DATA ACCESS

A primary feature of ADAAS is the ability to access data sets maintained by the Transportation Data Center - an extensive collection of motor vehicle traffic accident records and transportation-related information. Approximately 700 separate data sets from many different sources are currently available. Data sets in this category are stored in a "system library" that is accessible to users; but can only be modified by the Data Center.
In addition, there may be special purpose libraries maintained by the Transportation Data Center, or private libraries maintained by users, that provide access to customized data sets of transient or special interest. Data sets in these "non-system" libraries may be produced by the ADAAS SUBSET command or by special programs (i.e., other statistical packages or special purpose data processing programs). The LIBRARY program (executed by typing "library" at the UNIX command prompt) provides management functions for the maintenance of libraries. Documentation of this program may be found in Appendix A. The ADAAS program defaults to the system library when execution begins, but other libraries (to which the user has UNIX system READ access) may be accessed by entering a SET command with LIBPATH and LIBNAME options specifying location and name of the desired library. The SET command may be entered in program command mode; or may be placed in a program initialization file to automatically specify this library at each program execution.

The DATA command is used to access data sets defined in any properly permitted library. The only user-supplied information required is a "data keyword" that identifies the particular data set desired. For example, FARS95OC represents the FARS occupant level data set for calendar year 1995. All information needed to assign and access files is internal to the system. A complete list of data keywords for the current library is available on-line with the "data list" command. Documentation of system or private library contents is provided by the LIBRARY program's LIST command.

Most system data sets are documented by detailed codebooks. Each codebook contains a description of variable (i.e., information element) characteristics; English text definitions of the (generally) numeric values for each variable; frequency counts for the values; and other descriptive items. A new codebook is produced for each year or format version of each data set (e.g. FARS 1994 for the 1994 FARS files, UMIVOR 16 for the 16th update of the UMIVOR file, etc.). A comprehensive list of available data sets and corresponding codebooks may be obtained from the Data Center.

1.3) DATA STRUCTURE

In ADAAS, a "data set" is defined as a group of files that together make up a data entity that can be accessed, analyzed, manipulated, or used as input to other programs. A data set consists of at least two files: a dictionary file and a data file. A system data set usually also includes a label file that is used for substituting text for numeric data values in output produced by many of the commands. System data sets (i.e., data sets maintained by the Transportation Data Center and stored in the system library) are used most often by users; however, there are provisions to enter user-defined data sets into a user's own library. Such data sets may have been created by ADAAS commands such as SUBSET, or other means.

1.4) DATA ANALYSIS

Once a data set has been accessed, a number of simple data retrieval and analysis functions can be performed in ADAAS. Univariate frequency distributions and statistics can be generated with the UNIVAR command. Tables displaying bivariate frequency distributions can be produced with
the **TWOWAY** command. A formatted list of selected variables for selected records can be produced with the **LIST** command. A subset of the data set in compatible format can be generated with the **SUBSET** command for subsequent entry into the system (in the user’s private library) via the **LIBRARY** program. The **LIST** and **SUBSET** commands also have the capability of generating deterministic or pseudo-random selections of records from the data.

### 1.5) DATA INTERFACE

For more experienced users and more demanding analyses, the data accessed through ADAAS can be interfaced with other statistical packages on the TDC UNIX system; or they may be downloaded for import into other statistical packages. The dictionary format is a modified OSIRIS “type 5” and consists of one record for each variable containing definitions of variable name, field width, data type, starting position in the data record, and missing data value. New dictionaries are written by the **SUBSET** command and contain only the selected subset variables and reflect the new order and position of these variables. These dictionaries can be invaluable in interfacing data subsets with user analysis programs. The dictionary contains only ASCII text entries and is thus directly readable by the user.

The **SUBSET** and **TWOWAY** commands also have special output formats that may be used to facilitate the transfer of data to other programs. The DELIMITED option in **SUBSET** will cause variables to be separated by a special character. Tables generated by **TWOWAY** may also be written in a comma-delimited format.

### 1.6) SETUP FILES

An efficient way to operate ADAAS interactively is to use a "setup" file. This technique involves the creation and use of a command file that contains input information required by the program. Using a UNIX file editor, statements are inserted into a file in the form and order they would be entered into ADAAS from the terminal. Then the **SET** command’s INPUT option is used to direct ADAAS to use commands from this file instead of from the terminal. Alternately, and more efficiently, users may create the setup file on their own system (generally a PC), and transfer it to their home directory on the TDC UNIX system via FTP or some other transfer program. The use of setup files has these advantages:

1) Potential errors in the input information can be discovered and corrected by editing the setup file before it is executed.

2) The setup file can be exercised with the **SET** command’s TEST option in effect to access the setup without actually reading the data set.

3) If a line has been entered in error, it can be corrected with a file editor and executed a second time.

4) Subsequent tasks that involve only slight modifications to the setup may be easily performed.

5) A copy of the setup file can be maintained to document what was done and to provide a basis for new setup files.
Many of the examples used in this documentation are simply the listings of files input into the system as setup files. For more information concerning their use, see "Using Setup Files" in Section 3, and the description of the SET command in Section 5.

1.7) ATTENTION INTERRUPT PROCESSING

Attention interrupts may be issued at any time during execution to stop the action currently underway, and to return the user to the UNIX operating system. The interrupt key is normally CONTROL-C, but depends on how your keyboard is configured. In most situations, the message "TERMINATING adaas" will be displayed on receipt of an interrupt, and control will be returned to UNIX, releasing the active data set and aborting any processing in progress. The user must restart ADAAS and reissue all input commands.

1.8) ERROR RECOVERY

Problems can arise during the operation of ADAAS from two sources: invalid user entries, and program failures. When errors occur, one or more messages are printed to indicate the nature of the problem, and hopefully, to point out the solution. A comprehensive list of system error messages together with their explanation is given in Appendix C.

Error recovery is often possible while operating ADAAS interactively. Invalid program options cause ADAAS to seek replacements from the terminal whether the input is entered from the terminal or comes from a setup file. Consequently, errors present in a pre-prepared setup file can be replaced directly from the terminal as the program executes without interrupting the command and fixing the setup directly. The example below displays these features.

In the example, the user has three options when an incorrect keyword or modifier is entered. A replacement modifier or keyword (both keyword and value) may be entered; the letter "C" may be entered to abort the line entirely; or the letter "D" may be entered to simply delete the offending modifier or keyword while retaining the rest of the line.

Example of ADAAS Error Recovery
("***" at far right = user supplied input)

Command ?
datta k=FARS940C
"datta" is not a recognizable command.
Enter a replacement (OR "C" for cancel, "H" for help).
data
/usr/umtri/tdc/apps/adaas/doc/library.dat/FARS94OC: Access is established

Command ?
univar nogood=bad wront
"nogood=bad" is not a recognizable option.
Enter a replacement (OR "C" for cancel, "D" for delete, "H" for help).
out=results
"wront" is not a recognizable option.
Enter a replacement (OR "C" for cancel, "D" for delete, "H" for help).
d
Filter, Recode, Dupvar, Casevar, or Title
(Title, null line, or End-Of-File terminates entry)

Certain catastrophic failures in the operation of the program result in the occurrence of a "SEGMENTATION FAULT." These interrupts are most often caused by a failure in program logic that has never been detected. Should such a condition occur, the operation that is currently underway will be aborted and the user will be returned to the UNIX system prompt. Your help in notifying TDC of any program failures that occur, and in retaining as much documentation of the event as possible is greatly appreciated.

1.9) HELP FEATURE

By entering HELP in response to the command prompt, a collection of on-line program documentation is made available to the user. In most situations, the desired information is supplied directly by the HELP command. In situations where the documentation is lengthy, reference is made to an appropriate file where the information is located.

The basic HELP command (with no options) displays a list of the available commands and directs the user to more extensive documentation. With appropriate options, the command may be used for the following tasks: to display a brief description of all commands; to display a detailed description of any selected command; to display a description of the FILTER or RECODE statement syntax; or to obtain information on the data keywords that are used to access each data set. The keywords are arranged by groups (FEDERAL, OTHER, etc.) so that the desired keyword can be easily obtained by a simple top down search.

1.10) ADAAS INITIALIZATION FILE

ADAAS will read the contents of the file "adaas.ini" at execution if it exists in the user's home directory. This file may contain any valid SET command to configure the program as required. These commands may be "set libpath=~ libname=mylib" to always access the personal data library "mylib" on the user's home directory, or "set printwid=60" if narrow output is desired from all ADAAS analysis commands. Documentation may also be inserted by adding comment lines that begin with a semi-colon as the first non-blank character.

1.11) EXAMPLE OF OPERATION

The following example shows a complete session for a typical simple problem. The FARS (NHTSA'S Fatal Accident Reporting System) data set for 1993 is used to determine the relative frequency of fatal accidents on and off the road by road curvature in wet weather accidents. A setup file (called "setupfilename") was prepared beforehand and contains the following lines:

data k=FARS93AC
twoway output=farstable
include v33=2
recode v14 (01)=01 (02-98)=02
FARS 1993 - Wet Road Crashes
cv=30 rv=26 fv1=14:02 id=multivehicle
done

The contents of the setup file are executed with the INPUT option of the SET command.

Command ?
set input=setupfilename
/usr/umtri/tdc/apps/adaas/doc/library.dat/FARS93AC: Access is established
Analysis for data set /usr/umtri/tdc/apps/adaas/doc/library.dat/FARS93AC
Program: TWOWAY
Variables = 3
Processing Begins
Records Read = 5002
Processing Completed
End-Of-File on input
Input returned to the terminal

Command ?
;***Note: Here’s a way to see the output stored in the file “farstable”

Command ?
$more farstable

Program: TWOWAY (Bivariate Table)
Mon Sep 23 09:15:51 1996

Data Key = /usr/umtri/tdc/apps/adaas/doc/library.dat/FARS93AC

Filter = INCLUDE V33=2

Recode = RECODE V14 (01)=01 (02-98)=02

Title = FARS 1993 - WET ROAD CRASHES

Listing of Filter Variables Requested:
Group Numbr Name                       Loc Wid T  Dec     Missing
33 ROADWY SURFACE CONDITION    69   1 C                9

Listing of Recode Variables Requested:
Group Numbr Name                       Loc Wid T  Dec     Missing
14 NUMBER OF VEHICLE FORMS     30   2 C

Global Parameters:
LABELS = /usr/umtri/tdc/data/fars/fs93lab

Table Variables:
TN            CV          RV          WV
1            30          26

Local Filters
TN #         VAR         MIN         MAX
1 1          14           2           2

Percentage Switches & ID Tag
TN RO1W% COL% TOT%  MD% FREQ  MD  ID
1     F    F    F    T    T    T multivehicle

Variables = 3
Table Number 1  multivehicle  
Mon Sep 23 09:16:51 1996  
FARS 1993 - WET ROAD CRASHES  
Bivariate Frequencies  
Column Variable 30:ROADWAY ALIGNMENT  
Row Variable 26:RELATION TO ROADWAY  
FV1 Variable 14:NUMBER OF VEHICLE FORMS - Range 2 To 2  
Options:  FREQ  NO%  MD%  MD

<table>
<thead>
<tr>
<th>St</th>
<th>Ustraight</th>
<th>Curve</th>
<th>nknown</th>
<th>1</th>
<th>2</th>
<th>9</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>On roadway</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1863.</td>
<td>486.</td>
<td>10.</td>
</tr>
<tr>
<td>Shoulder</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>13.</td>
<td>6.</td>
<td>0.</td>
</tr>
<tr>
<td>Median</td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>13.</td>
<td>4.</td>
<td>0.</td>
</tr>
<tr>
<td>Roadside</td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>16.</td>
<td>12.</td>
<td>0.</td>
</tr>
<tr>
<td>Outside rt-of-wy</td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td>1.</td>
<td>0.</td>
<td>0.</td>
</tr>
<tr>
<td>Off rdwy-loc unk</td>
<td></td>
<td></td>
<td></td>
<td>6</td>
<td>9.</td>
<td>5.</td>
<td>0.</td>
</tr>
<tr>
<td>In parking lane</td>
<td></td>
<td></td>
<td></td>
<td>7</td>
<td>1.</td>
<td>0.</td>
<td>0.</td>
</tr>
<tr>
<td>Gore</td>
<td></td>
<td></td>
<td></td>
<td>8</td>
<td>1.</td>
<td>0.</td>
<td>0.</td>
</tr>
<tr>
<td>TOTAL</td>
<td>1917.</td>
<td>513.</td>
<td>10.</td>
<td>2440.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Command ?

Section 2  
DATA SETS AND DATA STRUCTURE

Because ADAAS began life as an offshoot of the OSIRIS system, the structure of the data sets is essentially that of its parent. Much of this similarity has been kept intact over the years in order to realize the compatibility gained through the widespread acceptance of this structure by many other currently available analysis packages. Significant differences do exist, however, and this section will present the structure of the ADAAS data sets as necessary background information for the successful analysis of data sets maintained by the Transportation Data Center using other software packages.

There are three components of an ADAAS data set: a DATA file that contains the actual stored information, a DICTIONARY file that provides
the computer with a machine-readable description of the information stored in the data file, and optionally, a LABEL file that contains English equivalents for numeric code values to label output information generated from the data file in a readable format.

2.1) DICTIONARIES AND CODEBOOKS

All information describing data sets is contained in a CODEBOOKCARDS file. This file (not available to system users) contains a set of formatted records of different types. One record type, called a T-record (or T-card) contains the information defining the characteristics of each information element (or VARIABLE) in the data set. A second record, called the C-card defines the code values for each permitted level of the variable that it relates to. The remaining record types are used in the generation of the data set codebook - the printed document that is distributed to users, and which is necessary for most analysis operations.

If the T-cards alone are extracted from the CODEBOOKCARDS file, they may be processed to form a dictionary. It is this file that is used by the system programs to obtain the information necessary to access the required data. The major items of information contained in the T-card for a given variable (i.e., data element) are shown below, followed by an explanation of each item.

Dictionary (T-card) record contents

<table>
<thead>
<tr>
<th>Number</th>
<th>Variable designator (1 to 32767)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>1 to 24 character name</td>
</tr>
<tr>
<td>Type</td>
<td>Data storage mode</td>
</tr>
<tr>
<td></td>
<td>C = Character numeric</td>
</tr>
<tr>
<td></td>
<td>A = Alphabetic</td>
</tr>
<tr>
<td>Location</td>
<td>Column location of field</td>
</tr>
<tr>
<td>Field Width</td>
<td>Number of characters in field</td>
</tr>
<tr>
<td>Decimal Places</td>
<td>Number of implied decimal places</td>
</tr>
<tr>
<td>MD Code #1</td>
<td>Missing data #1 (blank=none)</td>
</tr>
</tbody>
</table>

Number
Each element in the data set is assigned a number that is not necessarily sequential. This number is used as a surrogate for the variable in all program specifications. Thus a variable numbered 134 would be referred to as V134 in a Filter or Recode statement (e.g., INCLUDE V134=1) or simply as 134 in other places (e.g., CV=134, FV1=134, etc.).

Name
This name is used to document generated tables, etc. with English variable names.

Type
The type designator indicates whether the data is stored as a character numeric number (i.e., as a string of ASCII numeric characters), or as an
alphabetic string. Alphabetic variables may be used as filter or recode variables, and may be used in the LIST or SUBSET programs, but are not accepted in the TWOWAY or UNIVAR programs unless they have been recoded to numeric values.

**Location**
Indicates the column location of the first character in the data record containing the information for this variable. This value is useful when special programs are written to process the data file.

**Field width**
The number of characters needed to represent the value of this variable. The maximum field width for numeric variables is 11 and for alphabetic variables is 127.

**Decimal Places**
The variable value is assumed to have an implicitly defined decimal place located from the right side of the data field. A data value of "01" with an implied decimal place of "1" is interpreted as "0.1". With an implied decimal place of "-6" the value would be interpreted as "1000000".

**MD Code #1**
All data values equal to this value are treated as missing and are handled specially in certain analysis operations. An entirely blank field indicates there is no missing data code.

ADAAS dictionaries use the OSIRIS Type 5 format (without the leading header information). ADAAS dictionary records are simply T-cards with the actual column location information for each data element inserted. A listing of the dictionary information for the currently active data set may be generated by the FILES command.

In general, users will find that system-supplied dictionaries are adequate for the majority of purposes. Sometimes, however, problems arise that can be handled most conveniently by a dictionary change. Suppose for example that the system-supplied dictionary referenced a "VEHICLE IDENTIFICATION NUMBER" variable as a data element having eleven columns and it was necessary to access the characters as distinct variables. A modified dictionary with variables defining single column alphabetic fields could be produced. Contact the Data Center for techniques to generate new dictionaries.

A printed codebook for each data set contains all the dictionary information presented above together with the code values for each variable and the occurrence frequency for each code value in the data set.

2.2) **DATA STRUCTURE**

ADAAS does not support structured files so that the file structure is referred to as "flat" or "rectangular." This means that each record, or entry, in the data file consists of a single record of fixed length that is independent of all other entries. Since ADAAS is resident on a UNIX machine, the data characters are stored in ASCII.

Since accident information systems are generally hierarchical in format with accident, vehicle, driver, person, and non-motorist record types, some means of flattening this structure must be employed. This is
accomplished by generating ACCIDENT, VEHICLE, and PERSON data sets with redundantly coded information. The ACCIDENT data set contains all information from the accident-level descriptor so that there is one entry in this file for each event (or accident). The VEHICLE data set contains all the accident information contained in the accident data set together with vehicle and driver information so that there is one entry for each vehicle in each accident. Note that accident information is redundantly coded in multiple-vehicle accidents. The PERSON data set contains all the information in the vehicle data set together with the person information so that there is one entry for each person in each vehicle in each accident. Here, the accident and vehicle information is redundantly coded for multiple vehicle, multiple occupant accidents. With this structure, it is evident that accident level univariate frequencies should not be derived from the PERSON-level data sets, since this information will be weighted by the number of persons in each accident.

Some of the deficiencies resulting from the use of rectangular data sets have been remedied by DUPVAR and CASEVAR options. DUPVAR permits the deletion of sequentially occurring records that are duplicates in a defined sense, while CASEVAR permits the FILTER operation to act on all records that are defined as a “case”. These features are defined in more detail in Section 4.

There is nothing sacred in this decomposition of the original data hierarchy into a flat file structure. Many other equally useful structures can be designed. The structures defined above have been chosen as the default type since they have been shown by experience to be the most generally useful configurations. Many data sets are maintained in their original hierarchical format so users who feel that a different structure would be of advantage should contact the Data Center.

2.3) DICTIONARY AND DATA FILES

The DICTIONARY and DATA files are stored on disk. The information is stored using one line per record (unblocked). To provide record length information, each DATA file is preceded by a one-line HEADER record and the actual data records begin on the second line of the disk file.

<table>
<thead>
<tr>
<th>Column</th>
<th>Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 3</td>
<td>&quot;00F&quot;</td>
</tr>
<tr>
<td>4 - 8</td>
<td>record length (with leading zeros)</td>
</tr>
</tbody>
</table>

2.4) LABEL FILES

The final (optional) component of the ADAAS set is the LABEL file containing code value equivalents for applicable variables. The label file is accessed by a set of system subroutines that replace the numerical codes stored in the data file with readable information. Label files are generated by the Data Center as part of the normal data set generation process.
Section 3
OPERATING PROCEDURES

Using the ADAAS system to generate meaningful results from one or more of the available data sets usually involves a fairly standardized set of operations. In this section, techniques that are commonly used will be discussed in some detail. Complete documentation of all the system commands is presented in Section 5. Operation in interactive mode is the most common method of execution and consequently receives the major emphasis. The use of ADAAS in batch mode, however, is a possible technique for reducing connection time and the means of doing this will be presented as an extension of the use of setup files.

It is impossible to discuss the operation of ADAAS without referring at times to UNIX, the operating system within which ADAAS executes. Documentation of the UNIX operating system is beyond the scope of this manual and the user is encouraged to obtain one of the many readily available general manuals that are available.

3.1) INPUT CONSIDERATIONS

All information entered into the program for control purposes is processed by a common set of routines that provide a consistent input protocol. The maximum length of any input statement is determined by the value of the SET command’s INPUTLEN option. The default maximum is 1,000 characters, but may be changed with the SET command. If the command statement is too long for convenient entry with the input device, it may be entered in segments by terminating each segment with a dash (i.e., "-“) and continuing the statement on the next line. Note that the dash must be the last character on the line and must not be followed by any other characters (including blanks).

Null lines (i.e., a RETURN with no preceding characters), blank lines, and any line whose first non-blank character is a semi-colon are generally ignored and result in a repeat of the current prompt for information. Two exceptions to this rule are the “Filter, Recode, Dupvar, Casevar, or Title” prompt where a null line terminates entry; and the “RETURN” for more; “Q” for quit” prompt where anything but “Q” results in the display of more output.

Using a semi-colon as the first non-blank character in a line permits the insertion of comments in command files, initialization files, and logged program output for documentation purposes.

3.2) CONTROL COMMANDS

ADAAS is controlled by entering a command statement in response to the program request. All prompts for user-supplied input are preceded by an indication of the type of information required. If the user fails to enter information required by the command, default values are used if possible. Where this is not possible, the request is restated, possibly with an error message describing the missing elements, or the command is aborted.

The command statement request is:

Command ?
The command statement itself consists of a valid ADAAS command word that must be the first non-blank character, followed by any valid combination of keyword phrases and/or modifiers specific to the command. "Modifiers" are single words that modify the default operation of the command, while "keywords" are words used to assign specific values for the operation by means of the assignment KEYWORD=VALUE. Blanks are used as a separation character after the command and between all keyword phrases and modifiers. Although the command word must begin the line, modifiers and keyword phrases can be in any order.

Command Prototype

| COMMAND MOD2 KEY1=VALUE1 KEY2=VALUE2 MOD1 |

In addition to program commands, any valid UNIX command may be issued to ADAAS for processing provided that it is preceded by a dollar sign ("\$") in the first non-blank position of the input field. This can be a time saver in system operation as the command

$more <filename>

may be issued in ADAAS to display the contents of <filename> without terminating and restarting ADAAS.

3.3) ACCESSING DATA SETS

After execution of ADAAS has been initiated, the first major task is to select the data set desired for analysis, and to make this the ACTIVE data set - that is, the one on which subsequent operations will be performed. Each data set in the system library, as well as in every private library, is identified by a "data keyword" - a word containing one to eight printing characters. All keywords in the system library use upper case letters, but there is no requirement for this in general. The document "LIST OF SYSTEM DATA SETS", available from the Data Center, contains a complete list of data sets in the system library together with their characteristics. A list of keywords in the current library (the system library by default) may also be displayed with the DATA command:

```
data list
or
data key=*```

For private libraries, the library must first be made the "current" library with the SET command LIBPATH and LIBNAME options. Private libraries must also be properly permitted for others to access it.

When the data keyword is known, access is generated by the DATA command. If the current library has not been changed and is the default system library, the command for the NASS 1980 system accident data set is, for example

```
DATA KEY=NASS80AC```

If the current library is NOT the system library, then the appropriate commands would be
Similarly, for the private data set "NS80TEMP", located in a private library file of user "tdczzz" called "nasslib" the appropriate commands are:

```
SET LIBPATH=-tdczzz LIBNAME=nasslib
DATA KEY=NS80TEMP
```

No other knowledge of the data set is necessary.

Note that the DATA command need only be issued once for all operations on a given data set. For example, any number of analysis commands can be issued after a DATA command and all will use the ACTIVE data set that it accessed. A subsequent use of the DATA command is only necessary when a new data set is required.

### 3.4) ANALYSIS OUTPUT

ADAAS writes all output results into a file - the default output file when none is specified is "print.tmp" written in the user’s home directory. When the analysis command operation is complete, the output file may then be copied to the user terminal or to another output device for viewing. ADAAS commands that produce output accept the keyword assignment "OUTPUT=<filename>", where filename will be created automatically if it does not exist; or if it does exist, will be over written unless the "APPEND" modifier is used. For example:

```
TWOWAY OUTPUT=TABLE.3 APPEND
```

places the results of the twoway analysis at the end of the file TABLE.3.

The output file can be displayed by means of the UNIX "more" command or viewed in a file editor (In the example above "$MORE TABLE.3"). If the default output file "print.tmp" were used for the analysis operation, then it is only necessary to issue the "$more print.tmp" command.

### 3.5) DATA INTERFACE PROCEDURES

Once the desired data set has been accessed by the DATA command, there is no requirement that subsequent analysis be performed with ADAAS programs. Users will commonly find the system programs useful to perform preliminary overviews of the data, but will require more specialized packaged systems or even special programs to resolve their problems. For special user programs, it only necessary to use the FILES command to identify the names of the files that comprise the data set followed by STOP to terminate ADAAS. Information pertinent to file structure that is required for direct access can be found in Section 2.

### 3.6) USING SETUP FILES

Although the ADAAS program can be controlled by entering input commands or parameter lines as they are requested, it is also possible to place all input necessary to accomplish the desired objective in a UNIX file
and then later direct ADAAS to take its input commands from this file. There are several reasons why this procedure is cost and time effective.

1) The user may create the setup files on their PC and transfer them using a file transfer program (such as ftp) or cut and paste from a Windows clipboard.

2) The setup file can be checked for errors before it is used. Many potential problems can be easily located and fixed in advance.

3) If a mistake is found while executing the setup file, it is usually a simple job to fix the error with a UNIX file editor (such as "pico"), and to execute the task a second time. This technique is of special importance when the commands are long and complicated and their entry is subject to typing errors.

4) Repeated analysis operations with only slight modifications to the program input are easily accommodated.

The example below shows the contents of a setup file called "NASSLIST" containing four commands to perform a data set list on the NASS 1980 non-motorist data set.

\begin{verbatim}
data k=NASS80NM
list output=nasslist.report append
include v413=5
var=1,2,6 record label
\end{verbatim}

To use this setup file in ADAAS it is only necessary to issue the ADAAS \texttt{SET} command "\texttt{SET INPUT=NASSLIST}". This directs ADAAS to read all subsequent input from the file specified until another \texttt{SET INPUT} command is encountered, or until an end of file is encountered in the setup (i.e., no more input lines). ADAAS resumes reading commands from the terminal when the input file is exhausted. The execution of the setup file in the earlier example is shown below.

\begin{verbatim}
Command ?
set input=NASSLIST
/usr/umtri/tdc/apps/adaas/doc/library.dat/NASS80NM: Access is established
Analysis for data set /usr/umtri/tdc/apps/adaas/doc/library.dat/NASS80NM
Program: LIST
Processing Begins
Records Read = 3
Records Listed = 3
Processing Completed
End-Of-File on input
Input returned to the terminal
Command ?
\end{verbatim}

Note that when ADAAS is reading input from a source other than the terminal, the commands that it reads are not echoed back. The messages "End-Of-File on input" and "Input returned to the terminal" indicate that no more command lines were available in NASSLIST, and program control is consequently returned to the terminal.
Section 4
ANALYSIS COMMAND OVERVIEW

The amount of control information required by the data manipulation or analysis commands (i.e., LIST, SUBSET, TWOWAY, and UNIVAR) is too extensive to be included as part of the command statement. Consequently, use of one of these commands needs additional control information – two required and five optional statements:

1) A COMMAND Statement (Required)
2) A FILTER Statement (Optional)
3) A RECODE Statement (Optional)
4) A CASEVAR Statement (Optional)
5) A DUPVAR Statement (Optional)
6) A TITLE Statement (Optional)
7) An OPTION Statement (Required)

Each of these statements is discussed in detail in the sections that follow. After a legal analysis command statement has been entered at a command prompt, a prompt sequence begins with the request:

Filter, Recode, Dupvar, Casevar, or Title (Title, null line, or End-Of-File terminates entry)

At this point, one of the specified statements may be entered. The prompt is repeated until a title statement, a null line (i.e., RETURN with no preceding characters), or an end-of-file character is entered.

Only one filter statement is permitted. A maximum of ten recode statements may be entered.

If a problem is encountered while processing the input statement, an error replacement prompt is issued. It has the form:

Replacement Filter, Recode, Dupvar, Casevar, or Title (or "C" for cancel)

A replacement for the erroneous statement, or a new statement of a different type may be entered. If the letter "C" is entered, command execution is aborted and control is returned to the adaas command prompt.

After the Filter, Recode, etc. prompt has been completed, a prompt, unique to each command, is made for a command option statement. It has the form:

<command.name> Options (or STOP to terminate):

If “STOP” is entered, command execution is aborted and control is returned to the adaas command prompt.

A sample run illustrating the prompt sequence for the UNIVAR command appears below.

Example of Analysis Command Operation
("***" at far right = user supplied input)

Command ?
univar output=nass80.rpt  ***
Analysis for data set /usr/umtri/tdc/apps/adaas/doc/library.dat/NASS80VH

Filter, Recode, Dupvar, Casevar, or Title
>Title, null line, or End-Of-File terminates entry
include v108=01-09

Filter, Recode, Dupvar, Casevar, or Title
recode v112 (01-30)=01 (31-50)=02 (51-96)=03

Filter, Recode, Dupvar, Casevar, or Title
NASS 1980 - Passenger Cars - Object Contacted (weighted to Nat. Est.)

UNIVAR Options (or STOP to terminate)
var=112 freq% wv=49
Variables = 2
Processing Begins
Records Read = 3299
Processing Completed

4.1) COMMAND STATEMENT

The analysis COMMAND statement, in addition to specifying the analysis
command to be utilized (e.g., LIST, TWOWAY, etc.), also directs where
the program output will be written. The keyword and modifiers that may
optionally accompany the command on the same line are described here.

OUTPUT=<file name>  EXAMPLE: OUT=TEMP
Output produced by the command (e.g., tables, listings, etc.) are
written to the file specified in the directory from which ADAAS was run.
For example, specifying "OUT=MYFILE" would write the output to the file
MYFILE. If MYFILE does not exist, ADAAS creates the file, if MYFILE
exists the contents will be overwritten.
DEFAULT: The file "print.tmp" is created in the user’s home
directory and the analysis results are written to it. If print.tmp
already exists in the user’s home directory it is emptied first and then
the results of the current command are written to it.

APPEND  If the file specified by the OUTPUT keyword does not exist, the
file is created (and the APPEND modifier has no affect). If the file
already exists the APPEND modifier causes ADAAS to write the analysis
output to the end of the file, thus preserving the contents.
DEFAULT: Existing files are overwritten

4.2) FILTER STATEMENT

The FILTER statement selects those records (or "cases" containing
selected records if CASEVAR is also specified) from the input data set
that are to be included in, or excluded from, the analysis operation.
For a definition of the term "case", see the CASEVAR statement
description. These records are selected by a logically defined
combination of variable code values. In English, a typical filter
specification might be: "Include only those records involving a
passenger car in which the driver was over 45 years old and where the
accident occurred at night." The filtering affects all the operations
of a specific command (e.g., all the tables produced by the TWOWAY
command) and is therefore "global" in nature. This is in distinction to
"local" filters used in TWOWAY that affect only the contents of a
particular table.
A FILTER statement begins with either the word "INCLUDE" or the word "EXCLUDE" and is followed by a number of variable code value specifications that are combined logically by "AND" and/or "OR" operators. The FILTER statement may consequently include desired records or exclude undesired records, but cannot perform both actions. Combined actions are handled, for example, by including everything but an undesired value, thereby performing an exclude function. Each component part of the statement (i.e., INCLUDE, EXCLUDE, AND, OR, or a variable code value specification) MUST be separated from the others by a blank or blanks, but should not contain any imbedded blanks. **WARNING:** care should be taken to spell INCLUDE or EXCLUDE correctly; for if they are misspelled the desired filter statement will be interpreted as a title statement and no filtering will take place.

In the filter statement, each variable code value specification consists of a variable number, an equal sign, and a code value list. Variables are denoted by the letter "V" followed by the variable number (e.g., V23, V271). By default, a total of 50 variables may be used; but this limit can be changed with the SET command’s MAXFVARS option. Variable code value lists are expressed as single values (e.g., V9=1); as single values separated by commas (e.g., V45='CHEV','FORD'); as a range of consecutive values whose limits are separated by a dash (e.g., V23=10-15); or as a combination of these conventions (e.g., V23=1,2,4-9,35). The code value list for each variable may specify a maximum of 255 distinct values (for this code value count, the list "1-256" contains two values, not 256.) The total value count for filter and recode statements has a default limit of 2048.

Code values for alphabetic variables must be enclosed in primes. Blanks are permitted in alphabetic code values, but not in any other part of the code value specification. If fewer characters than required by the field width of the variable are specified, then the characters that are supplied are assumed to be LEFT justified in the field and are padded with blanks to the right. For example, using a variable describing color that has a field width of six, the value 'RED' is equivalent to the value 'RED   ' (but not to ' RED  ', '  RED ', or '   RED').

Only those records in which the designated variable number has one of the possible code values in the list will pass the filter (i.e., be included or excluded). Individual specifications are combined logically by the connectives "AND" and/or "OR" to provide the desired filter action. Each connective must be separated from other parts of the FILTER statement by one or more blanks. Use of "AND" implies that a record will pass the filter only if both adjacent variable code value specifications are satisfied. Use of "OR" implies that a record will pass the filter if either adjacent specification is satisfied. The "AND" term has precedence over "OR" (i.e., "AND" terms are evaluated first). Since nested parentheses are not permitted, this implies, for example, that the expression

\[ A \text{ AND } B \text{ AND } C \text{ OR } D \]

means

\[ (A \text{ AND } B \text{ AND } C) \text{ OR } D \]

If

\[ (A \text{ AND } B) \text{ AND } (C \text{ OR } D) \]

had been intended, the statement should be
A AND B AND C OR A AND B AND D

Some examples of FILTER statements for typical applications are shown below.

A) INCLUDE V66=5,22 AND V52=4

This includes those records for which variable 66 has a code of 5 or 22 and variable 52 has a code of 4.

B) INCLUDE V16=5 OR V14=9

This includes those records for which either variable 16 is coded 5, or for which variable 14 is coded 9, or both.

C) INCLUDE V16=5 AND V14=9 OR V16=6

This includes those records for which either V16 is coded 5 and V14 is coded 9, or for which V16 is coded 6.

D) INCLUDE V16=5 AND V14=9 AND V66=5 OR -
   V16=5 AND V14=9 AND V52=4

If the user desires those records where V16=5 and V14=9 and either V66=5 or V52=4, enter the above filter. Remember that "AND" takes precedence over "OR" so "INCLUDE V16=5 AND V14=9 AND V66=5 OR V52=4" would not select the records desired, e.g., all records where V52=4 would be included. Note also that the statement is continued to a second line by a terminating dash on the first line.

E) INCLUDE V54='RED','BLUE' AND V129='FORD'

This includes only those records where V54 is coded 'RED' or 'BLUE' and V129 is coded 'FORD'.

F) EXCLUDE V16=5 AND V14=1,7-9

This excludes those records for which both variable 16 is coded 5 and variable 14 is coded 1, 7, 8, or 9.

G) EXCLUDE V78='CHEU'-'CHEZ'

This excludes all records where V78 is coded 'CHEU', 'CHEV', 'CHEW', 'CHEX', 'CHEY', or 'CHEZ'.

4.3) RECODE STATEMENT

The RECODE statement permits code values for any variable to be temporarily (or permanently for SUBSET) modified for the duration of the command. The RECODE option can be used to convert alphabetic variables to numeric values, thus permitting the use of analysis commands on otherwise unusable variables. Alphabetic code values cannot be recoded
to other alphabetic values, however. RECODE is convenient for grouping variable values, as in bracketing age groups, or for changing non-consecutive values into a sequential range. For instance, code values for a variable that documents driver age could be translated by the rules

\[
\begin{align*}
0-9 & \rightarrow 1, \\
10-19 & \rightarrow 2, \\
\text{etc.},
\end{align*}
\]

to provide bracketed age groups.

Note that the RECODE operation is permanent for the output of the SUBSET command and generates a new data set with a modified data file. For the remaining commands it is only temporary and affects the command output only - the input data set is not altered in any way. Note also that code value labels are not altered during the recoding process (i.e., they remain associated with the values to which they were originally assigned within the system). For this reason, code value labeling is generally turned off for recoded variables. For special applications, labeling may be enabled by the LABLRECV option in the SET command.

A RECODE statement begins with the word "RECODE" (or "recode") in the first column of the statement followed by the variable to be recoded, and a set of translate specifications that give specific instructions for the code value translation. Each component part of the RECODE statement (i.e., RECODE, variable specification, or translate specification) must be separated from the others by one or more blanks, but must not contain any imbedded blanks. WARNING: Care should be taken to spell RECODE correctly; for if it is misspelled the desired recode statement will be interpreted as a title statement and no recoding of this variable will take place.

In the statement, the variable to be recoded is denoted by the letter "V" and the variable number (e.g., "V23" or "V271"). Following the variable number is a set of translate specifications of the form "(LIST)=VALUE" where LIST represents the code values to be translated and VALUE specifies the single value to which each element of LIST is translated. The LIST may be specified as a single value (e.g., 1); as a set of single values separated by commas (e.g., 'CHEV','FORD'); as a range of consecutive values whose limits are separated by a dash (e.g., 10-15); as a combination of these conventions (e.g., 1,2,4-9,35); or as the term "ELSE." The "ELSE" operand is used to represent all code values that have not been explicitly identified in a translate specification. If "ELSE" is not used, any code value not represented in LIST will not be recoded, i.e., they will retain their original values. The "ELSE" term is optional for numeric variables, but is required for alphabetic variables and, if used for numeric or alphabetic variables, should appear as the last translate specification.

Note that code values for alphabetic variables must be enclosed in primes. Blanks are permitted in alphabetic variable code values, but not in any other part of the translate specification. If fewer characters than required by the field width of the variable are specified, then the characters that are supplied are assumed to be LEFT justified in the field and are padded with blanks to the right. For example, using a variable describing color that has a field width of six the value 'RED' is equivalent to the value 'RED   '.

Up to ten variables may be recoded for a single analysis command. Each RECODE statement may contain up to 50 translate specifications of the
form "(LIST)=VALUE" and each "LIST" in each specification may contain up to 255 distinct code values. (For this code value count, the list "1-256" contains two values, not 256.) The total count for all filter and recode values must not exceed 2048

Some examples of typical RECODE statements are shown below.

A)  RECODE V87  (0-9)=1 (10-19)=2 (20-29)=3 -
(30-39)=4 (40-49)=5 (ELSE)=6

This example brackets the original code values below 50 into groups of 10 and lumps all values of 50 or greater into a single value.

B)  RECODE V13 ('WHITE')=1 ('YELLO')=2 -
('GOLD')=3 (ELSE)=9

Alphabetic color information is converted to numeric values. Note that 'GOLD ' will be recoded to the value "3" but ' GOLD' will be converted to "9."

C)  RECODE V33 (0,3,6-8)=9

Code values 0, 3, 6, 7, and 8 are converted to code value 9. Since no ELSE term is specified, code values 1, 2, 4, 5, and 9 are left unchanged.

D)  RECODE V591 (9)=1 (30)=2 (31)=3 (32)=4 -
(33)=5 (34)=6 (35)=7 (50)=8 (ELSE)=9

This recode converts the non-consecutive values 9, 30, 31, 32, 33, 34, 35, and 50 into consecutive values for use, as an example, as a column variable in a twoway table.

4.4) DUPVAR STATEMENT

In many data searches, it is useful to retain only the first occurrence of a sequential set of records that have duplicate values for a set of selected variables. As an example of how this need might arise, one might be interested in finding the MAKE/MODEL frequency distribution for vehicles in which at least one occupant was ejected. Since the determination of ejection must be done at an OCCUPANT level, the occupant level data set must be employed. If, however, more than one occupants are ejected in any vehicle, this vehicle would be over-represented in the count of vehicles with ejection. What is needed in this example is a way to prevent any vehicle that has more than one ejected occupant from being counted more than once. This can be done simply by rejecting any subsequent records that have accident IDs, and vehicle numbers that are identical to the record processed last. This is accomplished via a DUPVAR statement.

The DUPVAR statement allows the specification of a set of variables to be used for the elimination of duplicate records. Any record that has the same values as the previous record for each of specified variables is skipped. By specifying the CASEID and VEHICLE NUMBER variables in the above example, duplicate vehicle records would be eliminated resulting in a true vehicle count even though the OCCUPANT data is accessed.
The DUPVAR statement has the form:

```
DUPVAR=<list of up to 10 variable numbers>
```

For example, "DUPVAR=1,2,104". This statement would result in the elimination of any record for which variables 1, 2, and 104 had the same values as the previously processed record.

Unfortunately, since ADAAS is a sequential processing system, only sequentially occurring duplicates will be found and removed. This implies that for successful operation, the data sets must be sorted by the variables used for the DUPVAR keyword. This is generally true for the record type variables (i.e., CASEID, VEHICLE #, OCCUPANT #), but is not true for most variables.

The DUPVAR option strongly interacts with the CASEVAR option (described in Section 4.5 below.) Consequently the use of CASEVAR and DUPVAR in the same program setup is not permitted.

### 4.5) CASEVAR STATEMENT

The CASEVAR statement specifies a set of variables to be used for identifying a "case" - defined as a sequential set of records from the data set that have identical values for the specified CASEVAR variables. Each input record that has the same values as the previous record for the specified variables (and is not deleted by a DUPVAR specification) is considered to be part of the current case. When all records for a case have been input, each record is checked to see if it passes the specified FILTER criteria. If *ANY* record in the case passes the filter, all records in the case are output to the analysis programs. If no FILTER statement is specified then the CASEVAR option has no effect and a warning message indicating this is printed at the terminal. A maximum of ten variables may be specified.

While most sets of variables in system data sets are not sequentially arranged in sort order, accident number, vehicle number, occupant number, etc. generally are making CASEVAR useful in selecting accidents, vehicles in an accident, or occupants in a vehicle in an accident. For example, using accident number and vehicle number as CASEVAR variables in a occupant level data set, the filter statement could specify driver alcohol usage to obtain injury levels for all occupants in the vehicle.

If an error is detected while processing the CASEVAR specification, a request will be made for a replacement. A replacement CASEVAR statement; or a filter, recode, dupvar, or title may be entered at this point; or "C" may be entered to cancel the replacement and quit the analysis command.

The CASEVAR option interacts with some program options that control the selection of records. For example, if STOP=N is specified, the stop count may be reached before all records in the current case are processed. For this option, the STOP count is taken to be a lower limit and all records in the current case are processed before the command terminates. This is indicated by printing the possible range of STOP values at the terminal, and in the command output file. Other options such as RAND% and SKIP may affect case handling and should be used with care. The DUPVAR option (described in Section 4.4 above) strongly interacts with the CASEVAR option. Consequently the use of CASEVAR and DUPVAR in the same program setup is not permitted.
The CASEVAR selection is performed before any variables are filtered or recoded.

Example: CASEVAR=5,8,25

4.6) TITLE STATEMENT

The TITLE statement is a 124-character alphanumeric string that is used to identify program output. It is printed near the beginning of the program output, or near the beginning of each table. Any desired characters may be entered as part of this title except that the first word must not be "CASEVAR", "DUPVAR", "EXCLUDE", "INCLUDE", or "RECODE." If one of these five words is used at the beginning, the title will be taken as a casevar, dupvar, filter, or recode statement.

Some examples of typical TITLE statements are shown below.

A) COMPACTS REARENDED BY HEAVY TRUCKS

B) LIST OF OCCUPANTS WITH MINOR HEAD INJURIES

The table producing program TWOWAY permits the specification of a 24-character ID that may be used as a short title for each specific table in addition to the global title specified by the TITLE statement.

4.7) OPTIONS STATEMENT

The OPTIONS statement is a string of keywords and modifiers that control available program options. Since the content of the statement is unique for each analysis command, the keywords and modifiers making up the statement are described in full detail as a part of the individual command descriptions given in Section 5.

The LIST, SUBSET, and UNIVAR commands require only a single parameter line, while TWOWAY allows multiple lines. When multiple option lines are permitted, the DONE modifier terminates parameter input. Table input is also automatically terminated when the maximum number of tables is entered. The entry of multiple tables may be greatly simplified by using the SAME modifier. If SAME is specified in a parameter statement, then all keywords and modifiers that were specified on the previous entry are carried forward to the current entry and only values which are different need be specified. As an additional consideration in the use of multiple parameter lines, some modifiers or keywords need only be given once and may be given on any of the input lines. Such parameters (e.g., LABEL) are identified as **GLOBAL** in Section 5.

In addition to the "global" filter (i.e., the FILTER statement described previously), the TWOWAY command permits the entry of "local" filters with each table parameter statement. These filter specifications allow the selection of records that are entered into a given table and act in conjunction with the global filter. Use of more than one local filter implies an AND condition between the filter variables that are used. There are three forms of the local filter syntax:

FV1=12:3
FV1=12:3-6
FV1=NONE
The first form specifies that only records for which variable 12 has the value 3 are to be included in the table. Correspondingly, the second form specifies the inclusion of records for which variable 12 has the values 3, 4, 5, or 6. The third form is used to disable local filtering when the SAME modifier is used. For example, the options statement

```
CV=5 RV=8 FV1=12:1-3 FV2=37:4
SAME FV2=NULL
```

would result in local filtering on variables 12 and 37 for the first table, but only on variable 12 for the second table. NONE should not be used if that local filter has not been specified in a previous table. Please note that only single values or value ranges may be specified. A local filter entered as "FV1=12:1,3-5", for example, would be illegal.

4.8) PROCESSING ORDER

It is often important to know the order in which the various filter, recode, etc. operations specified in Section 4 take place when a new record is processed. The following list shows this order:

1) A new record is read from the data file.

2) The record is checked to see if it is part of a case defined by the casevar statement. If it is, it is added to a case buffer, and control returns to step 1). If there is no casevar statement, the record is added to the case buffer.

3) All records in the case are subjected to the filter criteria. If the case does not contain any record that passes the filter, it is rejected and control returns to step 1).

4) The record is checked to see if it satisfies the dupvar criteria. If the record is a duplicate of the preceding record, it is rejected and control returns to step 1).

5) Specified variables for every record in the case buffer are recoded to specified values.

6) Records in the case buffer are passed to the analysis program (list, twoway, etc.) sequentially until the buffer is exhausted.

This order implies, for example, that global filters must always be specified with data set (codebook) values while local filters for twoway, for example, which use recoded variables must be specified in terms of the recoded values.

Section 5
COMMAND DESCRIPTIONS

This section describes the function of each ADAAS command and gives a detailed description of all the modifiers and keywords that control the program operation. The syntax of the commands is presented in Section 3 and the special requirements of the analysis commands are presented in Section 4.
CODES Command
Command Form: CODES
Keywords: OUTPUT, VALUES, VARIABLE
Modifiers: APPEND, FORMAT

The CODES command displays variable names and code values for the active data set, provided this data set includes a label file. The active data set is the one most recently accessed by the DATA command. The command is valuable as a means of generating on-line data set documentation - especially code values for selected variables. The document "LIST OF SYSTEM DATA SETS" available from the Data Center provides information on label file availability.

CODES Keywords and Modifier

APPEND
EXAMPLE: OUTPUT=myfile APPEND
If the file specified by the OUTPUT keyword ALREADY exists, the output of the codes command is appended to the end of the file. If the specified file does not exist it is created.
DEFAULT: The specified file is emptied if it already exists.

FORMAT
EXAMPLE: CODES FORMAT OUTPUT=myfile var=1-10
If the FORMAT modifier is specified, the listing is preceded by a header consisting of a Data Keyword identification and the date. In addition, spacing is added between variables.
DEFAULT: No identification of data set or spacing between variables.

OUTPUT=<file or device name> EXAMPLE: OUT=MYFILE
Specifies the output file or device name where the command results will be written.
DEFAULT: Standard output (usually the screen)

VALUES=<value list>/NONE/ALL EXAMPLE: VAL=1000-1099
Specifies the code values to be printed for each variable. Ranges of values are designated with a hyphen. Individual values or ranges are separated by commas. No more than 100,000 values may be specified with this keyword. If NONE is entered, no code values are printed (i.e., only variable names are listed). The value ALL indicates all code values.
DEFAULT: All code values will be printed.

VARIABLE=<variable list> EXAMPLE: VAR=101-112,115
Specifies the variable numbers whose code values are to be listed. Ranges of variable numbers are designated with a hyphen. Individual numbers, or ranges of numbers, are separated by commas.
DEFAULT: Code values will be printed for all variables.

Example of the CODES Command
("***" at far right = user supplied input)

Command?
data k=CDS88OCC
/usr/umtri/tdc/apps/adaas/doc/library.dat/CDS88OCC: Access is established

Command?
codes var=716-718
Variable 716: EJECTION MEDIUM
Type = C Width = 1 Imp Decs = 0 Missing = 9
DATA Command

Command Form: DATA
Keywords: KEY
Modifiers: LIST

The DATA command generates access to any data set that is defined in the current library (defined by the SET command LIBPATH and LIBNAME options, or the system library by default). The data set processed by the DATA command becomes the ACTIVE data set that is used in all subsequent data manipulations until released explicitly by the RELEASE command, or implicitly by another DATA command. Data sets are selected by assigning a "data keyword" with the "KEY" option. These keywords are acronyms for data sets that are one to eight printing characters in length (e.g., FARS81PR, MI96VEH, SAVE1, etc.). The "HELP KEYWORD" command, or the wild card facility described below, may be used for assistance in locating the desired keyword.

Note that the DATA command need only be issued once for all operations on the active data set. If access to another data set is desired, it is not necessary to explicitly release the active data set with the RELEASE command before issuing a new DATA command.

Data sets that are not accessed sufficiently often to justify keeping them on-line are ARCHIVED and cannot be accessed via the DATA command. If an attempt is made to access an archived data set, a warning message is displayed.

The keywords and modifier accompanying the DATA command are described below.

DATA Keywords and Modifier

LIST  EXAMPLE: DATA List
This option displays a list of all available data keywords in the current library. By default the current library is the SYSTEM library; but this may be changed at any time to a properly permitted private
library with the **SET** command **LIBPATH** and **LIBNAME** options. The number of files in the system library is quite large and it may be more desirable to use the **KEY** option with wild card characters to delimit the list (see below).

**KEY=<data set keyword>**  
**EXAMPLE:** K=MIF77ACC  
Identifies the data set in the current library to be accessed. The **SET** command **LIBPATH** and **LIBNAME** options may be used to make any properly permitted library file the current library.

The common UNIX/DOS wild card characters "?" and "*" may be used with the "KEY" option to select keywords that are members of the set defined by the pattern supplied. If only one data set that matches the pattern is located in the library, then access to that data set is generated. If more than one data set is located, a list of the keywords that match the pattern is displayed and the **DATA** command terminates. The desired keyword may be selected from the list and used in a subsequent **DATA** command.

**DEFAULT:** None.

---

**Examples of the **DATA** Command**  
("***" at far right = user supplied input)

**Command ?**  
**data** key=FARS*AC  
There are 23 keywords in the current library matching FARS*AC

Valid keywords are:
FARS75AC  FARS76AC  FARS77AC  FARS78AC  FARS79AC  FARS80AC  
FARS81AC  FARS82AC  FARS83AC  FARS84AC  FARS85AC  FARS86AC  
FARS87AC  FARS88AC  FARS89AC  FARS90AC  FARS91AC  FARS92AC  
FARS93AC  FARS94AC  FARS95AC  FARS96AC  FARS97AC  FARS98AC  
**Command ?**  
**DATA** K=FARS85AC  
**/usr/umtri/tdc/apps/adaas/doc/library.dat/FARS85AC:** Access is established

**Command ?**  
**set libpath=~/jdsmythe libname=subset.lib**

**Command ?**  
**data k=fs89sub1**  
**/net/tdcnet2.umtri.umich.edu/nfsvol/home/jdsmythe/subset.lib/fs89sub1:** Access is established

**FILES** **Command**
**Command Form:** **FILES**
**Keyword:** **OUTPUT**
**Modifiers:** **NODICT, APPEND**

The **FILES** command displays the names of the dictionary, data, and label files as well as other items of information for the ACTIVE data set. This information, including a complete dictionary listing, is displayed.
on the standard output device (a good way to determine the active data set if there is doubt as to what it is). The OUTPUT keyword directs the results of the `FILES` command to the specified output file (at the end, if APPEND is specified). The NODICT modifier suppresses the variable listing so that only the file names, paths, and characteristics of the data set are listed.

`FILES` Keyword and Modifiers

APPEND

Example: OUTPUT=myfile APPEND

If the file specified by the OUTPUT keyword already exists, the output of the codes command is appended to the end of the file. If the specified file does not exist it is created.

DEFAULT: The specified file is emptied if it already exists.

NODICT

If the NODICT modifier is used with the `FILES` command, the dictionary listing is suppressed.

DEFAULT: A dictionary listing of the data set variables is made.

OUTPUT=<file or device name>

Example: OUT=MYFILE

Specifies the output file or device name where the command results will be written.

DEFAULT: Standard output (usually the screen)

Example of the `FILES` Command

("**" at far right = user supplied input)

Command ?
files

Data Descriptor = GES94ACC
Total variables = 50
Numeric variables = 50
Alpha variables = 0
First variable = 1
Last variable = 50
Record length = 87
Archive status = 0
Compression = U
Library path = /usr/umtri/tdc/apps/adaas/doc/
Name length = 30
Library name = library.dat
Name length = 11
Label file name = /usr/umtri/tdc/data/nass/ges94lab
Dictionary file = /usr/umtri/tdc/data/nass/ges94acc.dic
Num of data files = 1
Active file number = 0
Data file 1 name = /usr/umtri/tdc/data/nass/ges94acc.may0595

Listing of internal dictionary

<table>
<thead>
<tr>
<th>Group Numbr</th>
<th>Name</th>
<th>Loc</th>
<th>Wid</th>
<th>T</th>
<th>Dec</th>
<th>Missing</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CASE NUMBER</td>
<td>1</td>
<td>8</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>ACCIDENT DATE - MONTH</td>
<td>9</td>
<td>2</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>ACCIDENT DATE - YEAR</td>
<td>11</td>
<td>2</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>ACCIDENT DAY OF WEEK</td>
<td>13</td>
<td>1</td>
<td>C</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>ACCIDENT TIME - HOUR</td>
<td>14</td>
<td>2</td>
<td>C</td>
<td>99</td>
<td></td>
</tr>
</tbody>
</table>

(List truncated)

HELP Command
Command Form: HELP
Keywords: EXAMPLE, KEYWORD
The HELP command displays on-line documentation of system commands and data keywords. Use of HELP alone with no options displays a list of system commands and instructions on how to obtain further information.

A brief description of all available commands may be obtained with the COMMANDS modifier, or more detailed information may be obtained by using HELP followed by the command name. To obtain information on the keywords for all data set that can be accessed, use HELP KEYWORD.

A description of the FILTER and RECODE statement syntax requirements may be obtained by using the FILTER or RECODE modifiers, respectively.

HELP Keyword and Modifiers

COMMANDS
A brief description of available commands is displayed.

CASEVAR
A description of the use of the CASEVAR option is displayed.

DUPVAR
A description of the use of the DUPVAR option is displayed.

EXAMPLES
A list of available program operation example names is displayed. These names may be used with the EXAMPLE=<example name> option (see below).

EXAMPLE=<an example name>
The specified example is displayed. Example names can be determined with the HELP EXAMPLES command.

FILTER
A description of the FILTER statement syntax is displayed together with examples of typical statements.

HELP
A description of HELP keywords that may be entered for more information is displayed.

KEYWORD
A description of groups of data keywords is displayed together with the "group code" used with the KEYWORD=<group code> option (see below).

KEYWORD=<group code>
A listing of data keywords for the group defined by the code is displayed at the terminal. The code for each available group may be obtained with the KEY modifier.

RECODE
A description of the RECODE statement syntax is displayed together with examples of typical statements.

Example of the HELP Command
("***" at far right = user supplied input)

Command ?
HELP HELP

**
The HELP command prints on-line documentation of system operation.

Use one of the following HELP commands for more information:

- Help COMmands - Brief description of system commands
- Help {command} - More detailed description of each command (e.g., HELP UNIVAR)
- Help CASEvar - Description of CASEVAR statement syntax
- Help DUPvar - Description of DUPVAR statement syntax
- Help EXamples - List of program examples that may be displayed.
- Help FILTER - Description of FILTER statement syntax
- Help INIfiles - Use of an "adaas.ini" initialization file
- Help Keyword - Categories of available system data sets that can be accessed via the DATA command
- Help RECode - Description of RECODE statement syntax

Command ?

LIST Command
Command Form: LIST
Keyword: OUTPUT
Modifiers: APPEND

The LIST command retrieves and lists a subset of records and/or variables from the active data set in one of two optional formats: it is therefore of particular utility when a large number of variables, or alphabetic information, is needed on a small number of records. Records are selected with a FILTER statement; variables by the LIST option "VARIABLE".

After the LIST command is entered (with possible options OUTPUT or APPEND), the user is prompted for global options with the prompt:

Filter, Recode, Dupvar, Casevar, or Title
(Title, null line, or End-Of-File terminates entry)

The entry of a title ends the sequence of prompts for global parameters (each on a separate line or using a "-" to concatenate multiple lines) and the user sees:

LIST Options (or STOP to terminate)

At this point the user may specify the contents and layout of the analysis output. Two output formats are available, designated by the COLUMN and RECORD modifiers entered as LIST Options.

The default COLUMN format produces a tabular listing of the values for selected variables in columns using one line per record (useful for tabular searches). The number of columns is determined by the available print width. If there are too many variables to fit on a single line, the list is printed in segments (i.e., all records for the first group of variables to fit in the print width, followed by all records for the next group of variables, etc.). If any alphabetic variable has a field width greater than (PRINTWID – 8), the variable value is truncated and the truncation is indicated by a terminating character "|". This length limit has a value of 71 for the default print width of 79. A full width list of all variables (including full width values of long alphabetic variables) from the last LIST command that used the COLUMN mode option.
may be found in the file "ADAAS.LIST.colmode.tmp" on the users home directory.

The RECORD format produces a non-columnar listing. Every selected variable for a record is displayed within the available print width using as many lines of output as required. All information for a single record is therefore available at a single location.

The print width of the output list can be set by the PRINTWID option, or by the global "SET PRINTWID=<width>" command that affects all programs. See the SET command description in Section 5 for more information on the global option.

LIST Option Keywords and Modifiers

Note: Minimum abbreviations are indicated by the use of capitalization of the minimum portion of the command, keyword, or modifier needed.

Keywords:
- Labels - Label file name (labels=-user/labelfile)
- Printwid - Set print width of list (print=140 - max 1000)
- Rand% - Pseudo-random sample percentage (r=5 - 5% sample)
- SEed - Start value for RAND% random sequence
- Skip - Subset every n'th record (skip=3)
- STop - Stop after specified number of records (stop=10)
- Variable - List of variables to include in the list, or "ALLV" (var=allv: all variables)
- "ALLNV" (var=allnv: all numeric variables)

Modifiers:
- COLUMN - Display records in COLUMN format
- HELP - List keywords and modifiers
- LABEL - Use labels for numeric code values
- LIstdict - List the input dictionary parameters
- RECord - Display records in RECORD format
- SPace - Double space the list
- STOP - Terminate the LIST command

COLUMN

The variables to be listed are displayed in a columnar format using one line per record. The LABEL modifier will cause the values to be replaced with labels, if possible. It is important to note that the use of code value labels in COLUMN mode (i.e., specifying the LABEL modifier or the LABELS keyword) will significantly increase the number of segments necessary to print all variables.

DEFAULT: This is the default format.

HELP

A listing of the available program modifiers and keywords is displayed at the terminal. If this modifier is present, all other options (except STOP) are ignored, and a new option statement may be entered.

DEFAULT: No description is displayed.

LABEL

Labels, instead of numeric values, are used for each variable if they exist for that variable.

DEFAULT: Values are displayed.

LABELS=<label file> EXAMPLE: LAB=-tdc8aa/NASS82LAB

Code value labels for the data set being analyzed are found in the file specified.

DEFAULT: The label file name is supplied by the system if labels are implemented for the active data set.
LISTDICT
List dictionary records for requested variables.
DEFAULT: No dictionary records for variables are listed.

PRINTWID(TH)=<number>                  EXAMPLE: print=160
Sets the print width for the list generated by the program. The print
width has a maximum value of 1000 and a minimum value that depends on
the options specified.
DEFAULT: The default print width is the setting of the global PRINTWID
option (which defaults to a value of 79).

RAND%=<percent>                        EXAMPLE: RAND=5
A pseudo-random sample of records that satisfy the global filter
criteria is included in the listing. The percentage must be a positive
integer number between 1 and 99. The random sample begins with an
initial value (or "seed") that depends upon the time of day unless a
value is given explicitly by the SEED keyword. The RAND% option
interacts with the CASEVAR option (see Section 4.5) and should be used
with care in conjunction with CASEVAR.
DEFAULT: All records that satisfy the global filter are included.

RECORD
The variables to be listed are displayed in a non-tabular format. The
number of each variable is displayed with its corresponding value. The
LABEL modifier will cause the values to be replaced with labels, if
possible.
DEFAULT: COLUMN format.

SEED=<number>                          EXAMPLE: SEED=999
The random number generator used in the record selection specified by
the RAND% keyword is initialized with the value supplied. This number
must be an odd, positive integer with a value between 1 and
2,147,483,647. Two pseudo-random samples made with the same SEED value
will be identical.
DEFAULT: A seed value is computed from the time of day.

SKIP=<number>                          EXAMPLE: SK=15
Every Nth record (where N is a positive, integer number) that satisfies
the global filter criterion is included in the listing. The SKIP option
interacts with the CASEVAR option (see Section 4.5) and should be used
with care in conjunction with CASEVAR.
DEFAULT: All records that satisfy the global filter are included.

SPACE
The output listing is double-spaced.
DEFAULT: The output listing is single-spaced.

STOP
Command execution is terminated with no processing.

STOP=<number>                          EXAMPLE: STOP=10
Processing is terminated after the specified number of records has been
listed. If the CASEVAR option is in effect, the STOP count is a lower
limit, and all records in the current case are used before processing is
terminated.
DEFAULT: All records that satisfy the global filter are listed.

VARIABLE=<variable list>/ALLV/ALLNV    EXAMPLE: VAR=1,3,10-12
Specifies the variables to be included in the listing. The value “ALLV”
specifies that all variables are to be listed. The value “ALLNV”
specifies that all numeric variables are to be listed.
DEFAULT: No default. A variable list must be specified.

LIST Input and Output Examples

Examples of LIST Parameter Statements
("***" at far right = user supplied input)

LIST Options (or STOP to terminate)
var=1-10,12,21-23,12 label

Processing Begins

The example above produces a listing of variables 1-10, 12, and 21-23 in COLUMN format. Value labels, rather than values, are used in the listing. Notice that the variable list does not have to be in ascending order and that individual variables may be repeated, allowing the user considerable control over the exact format of the listing.

LIST Options (or STOP to terminate)
var=allv label record

Processing Begins

The preceding example produces a listing of all variables in the active data set in RECORD format. Value labels, rather than values, are used in the listing.

The following examples demonstrate the two formats available with the LIST command. Each example used the first four records in the FARS 1994 data set. Above each is the parameter statement that produced the output. Not shown is global information that precedes each table (such as a dictionary listing).

Example of LIST Output

LIST Options (or STOP to terminate)
var=1,2,9,10 label stop=4

Data Set List - COLUMN mode
Tue Sep 24 16:13:26 1996
FARS 1994

Rec. #  1      2  9 10
1  Alabama  1   January  1
2  Alabama  2   January  1
3  Alabama  3   January  3
4  Alabama  4   January  7

Records Read = 4
Records Listed = 4

Example of LIST Output

LIST Options (or STOP to terminate)
var=1,2,9,10 stop=4 record

Data Set List - RECORD mode
Sat Aug  2 11:00:36 1997
FARS 94

33
Record # 1
V1 = 1  V2 =  1  V9 = 1  V10 = 1

Record # 2
V1 = 1  V2 =  2  V9 = 1  V10 = 1

Record # 3
V1 = 1  V2 =  3  V9 = 1  V10 = 3

Record # 4
V1 = 1  V2 =  4  V9 = 1  V10 = 7

Records Read   = 4
Records Listed = 4

Example of LIST Output

LIST Options (or STOP to terminate)
var=1,2,9,10 label stop=4 record

Data Set List - RECORD mode
Tue Sep 24 16:18:12 1996
FARS 1994

Record # 1
V1:CASE STATE               = Alabama
V2:CASE NUMBER              =     1
V9:ACCIDENT DATE - MONTH    = January
V10:ACCIDENT DATE - DAY      =   1

Record # 2
V1:CASE STATE               = Alabama
V2:CASE NUMBER              =     2
V9:ACCIDENT DATE - MONTH    = January
V10:ACCIDENT DATE - DAY      =   1

Record # 3
V1:CASE STATE               = Alabama
V2:CASE NUMBER              =     3
V9:ACCIDENT DATE - MONTH    = January
V10:ACCIDENT DATE - DAY      =   3

Record # 4
V1:CASE STATE               = Alabama
V2:CASE NUMBER              =     4
V9:ACCIDENT DATE - MONTH    = January
V10:ACCIDENT DATE - DAY      =   7

Records Read   = 4
Records Listed = 4

NEWS Command
Command Form: NEWS <news item name>

The NEWS command displays current information about changes to the
system or updates to selected data set groups. Use "NEWS" with no
options to obtain a current list of news items available for viewing.
Use "NEWS FARS", for example, for news about Fatal Accident Reporting System data sets.

**RELEASE Command**  
Command Form: **RELEASE**

The **RELEASE** command removes access to the current data set. This command need not be used between successive **DATA** commands, but only when no further data manipulation is contemplated on any data set. Terminating ADAAS also releases the active data set.

**SET Command**  
Command Form: **SET**  
Keywords: INPUT, INPUTLEN, LABLRECV, LIBNAME, LIBPATH, MAXFVARS, MAXRECS, PAGE, PRINTWID, TEST  
Modifiers: VERIFY

The **SET** command controls several aspects of program operation. The **INPUT** option redirects the input stream to a file that contains previously prepared commands so that they may be executed automatically. The **INPUTLEN** option may be used to change the length of the program input buffer when long FILTER or RECODE statements are required. **LABLRECV** enables the labeling of recoded variables. The **LIBNAME** and **LIBPATH** options are used to access data keywords in a library other than the system library. **MAXFVARS** sets the maximum number of variables that may be used in a FILTER statement. **MAXRECS** sets the maximum number of records in a case when the **CASEVAR** option is in effect. **PAGE** controls paging in analysis program outputs. The **PRINTWID** option may be used to set the output print width to something other than the system default. The **TEST** option is used to put in the system in a "test" mode to debug analysis setup routines.

The **SET** command may alternatively be issued with NO options to display the current value of each of the parameters that may be changed with the command. The **VERIFY** modifier may also be used with any **SET** command to provide the same display and to verify the parameter settings.

The **INPUT** keyword is especially useful. For ADAAS operations that are performed frequently, or that involve a good deal of forethought, the user can employ an editor to enter all the necessary commands, filters, recodes, and parameter statements into a file as they would normally be entered from the terminal. After checking the file for errors, it can be input into the system for quick processing. This not only reduces the time pent in the system, but also provides the capability of editing the file before re-entry if any errors are encountered.

**SET Command Keywords**

**INPUT={<filename>|ME}**  
EXAMPLE: IN=ADAASRUN  
All ensuing commands will be read from the specified file or device until an end of file condition is sensed, or another "SET INPUT=" specification is read. The value "ME" designates the system "standard input" (the terminal).  
DEFAULT: Input is read from standard input (stdin).

**INPUTLEN={a number}**  
EXAMPLE: INPUTLEN=2000  
The length of the program buffer (i.e., the maximum length of an input line that the program will accept) is set to the value indicated. The
default buffer length is usually sufficient for most applications, but a longer length may sometimes be required when complicated FILTER or RECODE statements are entered. The default length may be specified with "INPUTLEN=*". Since this option modifies the input buffer, other options entered on the same line may be lost and spurious error messages may be displayed. The INPUTLEN option must consequently not be entered with any other SET command options.
DEFAULT: The input buffer length is 1,000 characters.

LABELRECV={ON|OFF} EXAMPLE: LABL=ON
Labeling is normally turned off for variables that are recoded since the correspondence between code values and labels is broken. For special situations, however, labeling of recoded variables may be enabled by setting LABLRECV=ON. Labels are not modified in any way, however, so labels will still be assigned as if the value were not recoded. This is a global switch that remains in effect until turned off and should consequently be used with care.
DEFAULT: LABLRECV=OFF (Recoded variables are NOT labeled)

LIBNAME=<a file name>|* EXAMPLE: libname=nasslib
The name of the current library file is set to the name supplied. The system library name "library.dat" may be specified by entering "*".
DEFAULT: The default name is the system library name "library.dat".

LIBPATH=<a path name>|* EXAMPLE: libpath=~tdcaaa/adaaslibs
The path name for the current library file is set to the name supplied. The system file path "$ADAASHOME/doc" may be specified by entering "*".
DEFAULT: The default path points to the system library.

MAXFVARS=<a number> EXAMPLE: MAXFVARS=200
Specifies the maximum number of variables allowed in a filter statement.
DEFAULT: The default number of variables is 50.

MAXRECS=<a number> EXAMPLE: MAXREC=50
Specifies the maximum number of records in a case when the CASEVAR option is in effect.
DEFAULT: The default number of records is 10.

PAGE={ON|OFF} EXAMPLE: PAGE=OFF
The PAGE option controls paging of the output from the LIST, TWOWAY, and UNIVAR commands. If PAGE=ON, output is divided into pages of approximately 60 lines with an ASCII form feed character (CNTRL-L) and header at the beginning of each page. If PAGE=OFF, no form feed characters are used and a header is only printed at the beginning of the output. Note that documentation of programs options and dictionary list, etc. that occur at the beginning of the programs output are not affected by paging.
DEFAULT: PAGE=ON

PRINTWID=<a number> EXAMPLE: PRINT=130
Specifies the print width for program output.
DEFAULT: The default print width is 79.

TEST={ON|OFF} EXAMPLE: TEST=ON
If TEST is set ON, the program is put into a "test" mode that will enable it to read filters, recodes, titles, and option statements, but not do any actual processing. This mode is useful for checking analysis setups for possible errors before doing the actual run.
DEFAULT: TEST=OFF
Example of the **SET** Command's INPUT and TEST Features
("***" at far right = user supplied input)

In this example, the INPUT and TEST options are used to check the file "nass.setup" for errors before execution. An error in line 9 is located. This error can be edited and the repaired file used with TEST=OFF for error free execution of the setup.

Command ?
$cat nass.setup

```
data
k=CDS94INJ
twoway out=test.rpt append
include v309=01-09 and v342=000-800
RECODE V342 (00-08)=05 (09-17)=10 (18-25)=15 (26-33)=20 (34-42)=25 - 
(43-50)=30 (51-58)=35 (59-67)=40 (68-75)=45 (76-81)=50 (82-92)=55 - 
(93-150)=56
CDS 1994 Injuries - Passenger Cars with known Delta V
rv=342 cv=814 row% col% ID=all.injuries
same fv1=809;1 id=head.injuries
done
```

Command ?

```
set in=nass.setup test=on
```

```
/usr/umtri/tdc/apps/adaas/doc/library.dat/CDS94INJ: Access is 
established
Warning: ** The TEST option is ON **
Analysis for data set /usr/umtri/tdc/apps/adaas/doc/library.dat/CDS94INJ
Program: TWOWAY
Non-numeric character in number
Error occurred at column 4
"fv1=809;1" is not a recognizable option.
Enter a replacement (OR "C" for cancel, "D" for delete, "H" for help).
fv1=809;1
```

Variables = 3
End-Of-File on input
Input returned to the terminal

Command ?

**STOP Command**
Command Form: **STOP**

The **STOP** command terminates program operation and returns control to UNIX. This command should only be issued when all desired operations are completed. The active data set, if present, is released.

**SUBSET Command**
Command Form: **SUBSET**
Keyword: **OUTPUT**
Modifiers: **APPEND**

The **SUBSET** command creates a new data set from the active data set that includes only those records and variables selected by the user. Records to be included are determined by the FILTER statement in conjunction with the record-selecting program options (e.g., RAND%, SKIP, and STOP). Variables are selected with the VARIABLE program option. The new data set is written to disk files ("data.tmp" and "dict.tmp" by default) and may be subsequently entered into the user's private library with the
LIBRARY program where it may then be accessed like any other data set with the DATA command. An example of the procedure for doing this is given at the end of this section. The new data set is also available to the user for downloading (ftp).

The RECODE statement may be used to produce a permanent modification of variable values. For example, an alphabetic variable may be converted into a numeric variable (with no change in field width).

One of the important uses of SUBSET is the generation of data sets for analysis with other systems. For this reason a DELIM option generates data sets that are compatible with many programs. The use of the DELIM=<character> option produces a data set where each variable is separated by the character specified. The special delimiter option DELIM=STD places a comma between each variable and encloses alphabetic variables in primes.

Please note the difference between the documentation output assigned by the OUTPUT keyword on the command statement (referred to in the following descriptions as the documentation file) and the output files assigned in the options statement that make up the new data set itself.

**SUBSET Option Keywords and Modifiers**

**APPEND**
Append output to an existing data set file. Note that this is a DATA SET append supplied with the "SUBSET options" prompt - not the documentation append that is supplied as part of the SUBSET command discussed above.
A file must be specified with the DATA option when APPEND is used. The new data set is appended to the contents of this file with no header. It is the responsibility of the user to insure that the record length and content of the new records match those already contained in the data file.
DEFAULT: The data set file is created or emptied before use.

DATA=<filename> EXAMPLE: DATA=~tdcaaa/new/newdata
Specifies the output file to write the new data into. Emptied if it already exists, created if it does not exist. The example above writes the data file into the subdirectory /new on the home directory of user tdcaaa.
DEFAULT: Data is written to data.tmp in the user's home directory.

DELIMITR=<char>|BLANK|STD EXAMPLE: DEL=BLANK
Specify a character (delimiter) to put between fields in the output file. "delim=," would put a comma between fields. BLANK specifies a blank delimiter character. The special value "STD" causes a comma to be put between fields and alphabetic variables to be enclosed in primes.
When the DELIMITR option is used, the resulting data set is not system compatible and cannot be used with adaas.
DEFAULT: None; the data is packed with no delimiter.

DICT=<filename> EXAMPLE:DICT=newdict
Specifies the output file to write the new dictionary into. Emptied if it already exists, created if it does not exist.
DEFAULT: The dictionary is written to dict.tmp in the user's home directory.

HELP
A list of program options is displayed. Any other options (except STOP) on the same input line are ignored.
LISTDICT
A listing of all selected dictionary records for the input (i.e., the active data set) is written to the documentation file designated with the OUTPUT option or the default file print.tmp.
DEFAULT: Dictionary records are not listed for subset variables.

NOOUTDICT
No listing of the output dictionary records is written to the documentation file.
DEFAULT: A listing of the output dictionary is written to the output file.

PREPARE
The variables in the new data set have the same numbers that they had in the original data set.
DEFAULT: This is the default option.

RAND%=<percent>                   EXAMPLE: RAND=5
A pseudo-random sample of records that satisfy the global filter criteria are included in the listing. The percentage must be a positive integer number between 1 and 99. The random sample begins with an initial value (or "seed") that depends upon the time of day unless a value is given explicitly by the SEED keyword. The RAND% option interacts with the CASEVAR option (see Section 4.5) and should be used with care in conjunction with CASEVAR.
DEFAULT: All records that satisfy the global filter are included.

RENUMBER
The variables in the new data set are renumbered, retaining the sequential ordering of the variables from the original, active data set. The starting variable number may be specified with the START keyword.
DEFAULT: The variable numbers are preserved.

SEED=<number>                     EXAMPLE: SEED=999
The random number generator used in the record selection specified by the RAND% keyword is initialized with the value supplied. This number must be an odd, positive integer with a value between 1 and 2,147,483,647. Two pseudo-random samples made with the same SEED value will be identical.
DEFAULT: A seed value is computed from the time of day.

SKIP=<number>                     EXAMPLE: SKIP=100
Every Nth record (where N is a positive, integer number) that satisfies the global filter criteria is included in the new data set. The SKIP option interacts with the CASEVAR option (see Section 4.5) and should be used with care in conjunction with CASEVAR.
DEFAULT: All records that satisfy the global filter are included.

START=<number>                    EXAMPLE: START=101
If the RENUMBER modifier is included in the parameter statement, variables in the output data set will be sequentially renumbered starting with the variable number specified.
DEFAULT: If the RENUMBER modifier is specified, the default is "START=1."

STOP
The SUBSET command is terminated.

STOP=<number>                     EXAMPLE: STOP=9000
Processing is terminated after the specified number of records have been written into the data file. If the CASEVAR option is in effect, the
STOP count is a lower limit, and all records in the current case are used before processing is terminated.
DEFAULT: All records that satisfy the global filter criteria are included.

VARIABLE=<variable list>|ALLV|ALLNV  
EXAMPLE: VA=1,24
Specifies the variables to be included in the new output data set. The option ALLV specifies that all variables in the active data set are included in the new data set. The option ALLNV specifies that all NUMERIC variables in the active data set are included in the new data set.
DEFAULT: No default.

SUBSET Input and Output Examples

Example of a SUBSET Parameter Statement
(“***” at far right = user supplied input)

SUBSET Options (or STOP to terminate)
VAR=1-45,47 dict=waout.dict data=waout.data stop=10                  ***
Processing Begins
Dictionary processing Completed
Data processing Completed
  Records Read = 10
  Records Written = 10
  Processing Completed

Command ?

The new data set produced with the example above includes variables 1 through 45 and 47 from the active data set. The data set is written for 10 records into the files waout.dict and waout.data.

Example of a SUBSET Setup File and use

The setup file:

data  k=WAC91ACC
subset output=wa91sub.rpt append
  include v26=10-31
  recode v25 (4-6)=4
Washington 1991 Two-vehicle crashes, recode light to 4 levels
  var=1-32,34-35,44 data=wa91.dat dict=wa91.dic delim=, listdic stop=1

Executing the setup file:
Command ?
set input=subset.set                                                 ***
/usr/umtri/tdc/apps/adaas/doc/library.dat/WAC91ACC: Access is established
Analysis for data set /usr/umtri/tdc/apps/adaas/doc/library.dat/WAC91ACC
Program: SUBSET
Processing Begins
Dictionary processing Completed
Data processing Completed
  Records Read = 1
  Records Written = 1
  Processing Completed
REMINDER: The output data set is not system compatible
End-Of-File on input
Input returned to the terminal
Command ?

The preceding file, run with the "SET INPUT=" feature of ADAAS, produced a subset of the Washington 1991 data that is not compatible with the ADAAS system due to the specification of commas delimiting the data. Included were records where variable 26 was in the range 10-26. The new data set included a total of 46 variables from the original data set. The dictionary was written into the disk file "wa91.dic", while the data went into the disk file "wa91.dat."

The documentation file (specified with "output=wa91sub.rpt") contains the following information:

Program: SUBSET
Mon Sep 30 16:09:55 1996

Data Key = /usr/umtri/tdc/apps/adaas/doc/library.dat/WAC91ACC

Filter = INCLUDE V26=10-31

Recode = RECODE V25 (4-6)=4

Title = WASHINGTON 1991 TWO-VEHICLE CRASHES, RECODE LIGHT TO 4 LEVELS

Listing of Filter Variables Requested:
Group Numbr Name Loc Wid T Dec Missing
26 COLLISION TYPE 73 2 C 99

Listing of Recode Variables Requested:
Group Numbr Name Loc Wid T Dec Missing
25 LIGHT CONDITIONS 72 1 C 9

Listing of SUBSET Command Variables Requested:
Group Numbr Name Loc Wid T Dec Missing
1 ACCIDENT REPORT NUMBER 1 6 C
... (dictionary list truncated for this display)
44 WORST INJURY IN ACC 110 1 C 0

Options Specified:
PRESCRIBE
LISTDICT
DATA = wa91.dat
SKIP = 1
STOP = 1
DELIMITER =,
VARIABLE = 1-32,34-35,44

SUBSET Dictionary Listing:
Group Numbr Name Loc Wid T Dec Missing
1 ACCIDENT REPORT NUMBER 1 6 C
... (dictionary list truncated for this display)
44 WORST INJURY IN ACC 123 1 C 0

Output record length = 124

Records Read = 1
Records Written = 1

Note that the record length increased due to the ",," delimiter.
Adding SUBSET Data Sets to a Private Library

Suppose that a SUBSET operation had generated default "data.tmp" and "dict.tmp" files. The simple procedure for adding this new data set to a system library on the user's home directory so that it may be accessed by the DATA command is given below.

After terminating adaas, run the "library" program and enter the following commands:

```plaintext
library
set libpath=-tdczzz libname=* 
add key=tempdata dict=dict.tmp data=data.tmp 
list key=tempdata 
stop
```

The library SET command accesses the library file "library.dat" on the home directory of "tdczzz" or creates one if none exists. The ADD command adds a data set with keyword "tempdata" to this library, and the LIST command verifies the entry. Following this operation, adaas may be re-run and the new data set accessed with the adaas commands:

```plaintext
set libpath=-tdczzz libname=* 
data key=tempdata
```

Of course, other library, dictionary, data or keyword names may be used instead of the ones shown above. See the library program documentation in this manual for more details. The utility of such a "tempdata" keyword, however, is that it need only be entered once. If the default dict and data files are used for SUBSET at any later date, then the data set may be immediately accessed with the tempdata keyword without entering a new data set in the library.

**TWOWAY Command**
Command Form: TWOWAY
Keyword: OUTPUT
Modifier: APPEND

The TWOWAY command produces up to 50 tables showing the bivariate frequency distribution of selected pairs of variables from the active data set, together with a variety of optional percentages. Only those columns and rows with at least one non-zero element are printed. Tables with virtually unlimited dimensions (e.g., 10 by 10, 100 by 1000, etc.) may be generated with only the amount of memory available from the computer acting as an upper limit. Users should be aware, however, of the high memory use and processing time involved in the generation of very large tables.

By default, only frequencies appear in the cells of the table. Percentages based on the column total, the row total, and the grand (or table) total may be optionally calculated and printed.

In addition to the tabular output discussed above, a comma-delimited form of the bivariate frequencies may optionally be written to a file for later input to spreadsheets and other programs that accept this type of data. Output is limited to 100 columns in this format. The format of the comma-delimited is presented in Appendix D.
Some options are global in action and apply to ALL tables. Consequently they need be entered only once for a set of tables. These options are indicated below by the designation "(GLOBAL)".

Required parameters:
- CV - Column variable
- RV - Row variable
- DONE - Terminates table entry

ALL%
Percentages based on the column total (COL%), the row total (ROW%), and the grand (or table) total (TOT%) are all calculated and printed in the table cells.
DEFAULT: No percentages are calculated or printed.

COL%
Percentages based on the column totals are calculated and printed in the table cells.
DEFAULT: No column percentages are calculated.

CV=<variable list>  EXAMPLE: CV=12,34-36
Column variable(s) to be used for the table(s). For each variable listed, a separate table is generated.
DEFAULT: No default. The CV option must be specified.

DELIMITD=<file name>  (GLOBAL)  EXAMPLE: DELIM=special.tmp
Comma-delimited bivariate frequencies are written to the specified file for tables having 100 columns or less. See Appendix D for the output format of the DELIMITD file.
DEFAULT: No comma-delimited output is generated.

DONE
Indicates that all desired tables have been specified. May be entered on the last table specification line, or on a line by itself.
DEFAULT: Additional entries follow. MUST be entered after all parameters have been specified unless the 50-table limit is reached.

FREQ
Frequencies are printed in the table cells.
DEFAULT: This is the default option.

FV1=<variable:values>  EXAMPLE: FV1=23:0-5
First local filter and range. Only those records where the specified variable has a value in the indicated range are included in the table. "fv1=none" may be used if filter 1 had been specified on a previous table and "same" is specified, but no filter 1 is desired in the current table. The Analysis Command Overview's description of the parameter statement provides full information on the use of local filters.
DEFAULT: All records are used.

FV2=<variable:values>  EXAMPLE: FV2=35:1-15
Second local filter and range (see FV1).

FV3=<variable:values>  EXAMPLE: FV3=164:5
Third local filter and range (see FV1).

FV4=<variable:values>  EXAMPLE: FV4=123:25-125
Fourth local filter and range (see FV1).
HELP
A description of options is displayed. If present, all other parameters entered on the same line (except STOP) are ignored.
DEFAULT: No description is displayed.

ID=<24-character tag>         EXAMPLE: ID=ROLLOVER
A table identifier that consists of a twenty-four character tag with no imbedded blanks. It may be supplied for each table. The ID is printed at the beginning of each table.
DEFAULT: No ID is used.

LABEL    (GLOBAL)
Code value labels are printed for each variable if they exist for that variable.
DEFAULT: This is the default option.

LABELS=<label file>    (GLOBAL)    EXAMPLE: LAB=~user/MYNASSLAB
Code value labels for the data set being analyzed are found in the file specified. See Appendix A for label file construction procedures.
DEFAULT: The label file name is supplied by the system if labels are implemented for the active data set.

MD
Missing data values are included in the table.
DEFAULT: This is the default option.

MD%
Missing data values are included in the calculation of any percentages.
DEFAULT: This is the default option.

NO%
No percentages are calculated or printed in the table cells.
DEFAULT: This is the default option.

NOFREQ
Frequencies are not printed in the table cells.
DEFAULT: Frequencies are printed.

NOLABEL    (GLOBAL)
No code value labels are printed.
DEFAULT: Code value labels are printed for each variable if they exist for that variable.

NOMD
Missing data values are not included in the table.
DEFAULT: Missing data values are included.

NOMD%
Missing data values are not included in the calculation of any percentages.
DEFAULT: The calculations include missing data values.

PRINTWID=<columns>    (GLOBAL)    EXAMPLE: PRINT=130
The PRINTWID option designates a value for the number of columns used for program output.
DEFAULT: The default value is set by the SET command PRINTWID option which in turn defaults to 79.

ROW%
Percentages based on the row totals are calculated and printed in the table cells.
DEFAULT: No row percentages are calculated.

RV=<variable list>                EXAMPLE: RV=217
Row variable(s) to be used for the table(s). For each variable listed, a separate table is generated.
DEFAULT: No default. The RV option must be used.

SAME
All parameters not explicitly specified on the current parameter input line are set equal to their values defined on the previous line.
DEFAULT: All parameters desired must be specified.

STOP
The TWOWAY command is terminated.

STOP=<number>         (GLOBAL)    EXAMPLE: STO=1000
Processing is terminated after the specified number of records have been processed. If the CASEVAR option is in effect, the STOP count is a lower limit, and all records in the current case are used before processing is terminated.
DEFAULT: All records that satisfy the global filter are processed.

TN=<table number>                 EXAMPLE: T=100
An optional table number. A list of numbers may be assigned for a sequential list of tables.
DEFAULT: Tables are numbered sequentially 1-50.

TOT%
Percentages based on the grand (or table) total are calculated and printed in the table cells.
DEFAULT: No total percentages are calculated.

WV=<variable number>              EXAMPLE: WV=102
Record counts are weighted by the value of the specified variable. If the value of the weight variable is zero or equal to the missing data value, the record is skipped.
DEFAULT: No weighting is performed.

**TWOWAY** Input and Output Examples

Example of a **TWOWAY** Run
("***" at right = user supplied input)

adaas$ adaas

ADAAS
Transportation Data Center
Mon Nov 18 16:22:05 1996
UNIX Version 11-17-1996
Type "help help" for help information.

Command ?
data k=FARS75AC
/usr/umtri/tdc/apps/adaas/doc/library.dat/FARS75AC: Access is established

Command ?
twoway out=farstable append
Analysis for data set /usr/umtri/tdc/apps/adaas/doc/library.dat/FARS75AC
Filter, Recode, Dupvar, Casevar, or Title
(Title, null line, or End-Of-File terminates entry)
include v3=26

Filter, Recode, Dupvar, Casevar, or Title
recode v19 (01-13)=13 (14-29)=29

Filter, Recode, Dupvar, Casevar, or Title
dupvar=3,4

Filter, Recode, Dupvar, Casevar, or Title
FARS 1975, Michigan crashes (no dup on record number)

**TWOWAY** Table Options (or STOP to terminate)
(DONE terminates table option input)
cv=46 rv=19 row% nomd% id=Michigan.all
same fv1=33:1-5 id=Mich.bad.weather
done

Variables = 3
Processing Begins
Records Read = 1584
Processing Completed

Command ?

Two tables are produced with the preceding example. The first uses variable 19 for the row variable values and variable 46 for the column variable values. If variable 19 had a field width of four, for example, up to 10,000 rows may be included in the table. The second table is identical to the first with the exception that variable 33 is used to filter the records for bad weather crashes only. This table will contain one entry for each fatal crash that occurred in Michigan in 1975.

Example of a **TWOWAY** Setup File

This is the form the above example would look like in a file to be used for input to ADAAS:

data k=FARS75AC
twoway out=farstable append
include v3=26
recode v19 (01-13)=13 (14-29)=29
dupvar=3,4
FARS 1975, Michigan crashes (no dup on record number)
cv=46 rv=19 row% nomd% id=Michigan.all
same fv1=33:1-5 id=Mich.bad.weather
done

NOTE: The file above, run with the "SET INPUT=" feature of ADAAS, produced the output below

Example of **TWOWAY** Output
(produced from setup file above)

Program: TWOWAY (Bivariate Table)
Mon Nov 18 16:39:56 1996

Data Key = /usr/umtri/tdc/apps/adaas/doc/library.dat/FARS75AC
Filter = INCLUDE V3=26

Recode = RECODE V19 (01-13)=13 (14-29)=29

Dupvar = 3,4

Title = FARS 1975, MICHIGAN CRASHES (NO DUP ON RECORD NUMBER)

Listing of Filter Variables Requested:
<table>
<thead>
<tr>
<th>Group Numbr Name</th>
<th>Loc</th>
<th>Wid</th>
<th>T</th>
<th>Dec</th>
<th>Missing</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATE</td>
<td>11</td>
<td>2</td>
<td>C</td>
<td></td>
<td>99</td>
</tr>
</tbody>
</table>

Listing of Recode Variables Requested:
<table>
<thead>
<tr>
<th>Group Numbr Name</th>
<th>Loc</th>
<th>Wid</th>
<th>T</th>
<th>Dec</th>
<th>Missing</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIRST HARMFUL EVENT</td>
<td>44</td>
<td>2</td>
<td>C</td>
<td></td>
<td>99</td>
</tr>
</tbody>
</table>

Listing of Duplicate Check Variables Requested:
<table>
<thead>
<tr>
<th>Group Numbr Name</th>
<th>Loc</th>
<th>Wid</th>
<th>T</th>
<th>Dec</th>
<th>Missing</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATE</td>
<td>11</td>
<td>2</td>
<td>C</td>
<td></td>
<td>99</td>
</tr>
<tr>
<td>SEQUENCE ID</td>
<td>13</td>
<td>4</td>
<td>C</td>
<td></td>
<td>9999</td>
</tr>
</tbody>
</table>

Global Parameters:
LABELS = /usr/umtri/tdc/data/fars/fs75lab

Table Variables:
<table>
<thead>
<tr>
<th>TN</th>
<th>CV</th>
<th>RV</th>
<th>WV</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>46</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>46</td>
<td>19</td>
<td></td>
</tr>
</tbody>
</table>

Local Filters
<table>
<thead>
<tr>
<th>TN #</th>
<th>VAR</th>
<th>MIN</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>33</td>
<td>5</td>
</tr>
</tbody>
</table>

Percentage Switches & ID Tag
<table>
<thead>
<tr>
<th>TN</th>
<th>ROIW%</th>
<th>COL%</th>
<th>TOT%</th>
<th>MD%</th>
<th>FREQ</th>
<th>MD</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>Michigan.all</td>
</tr>
<tr>
<td>2</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>Mich.bad.weather</td>
</tr>
</tbody>
</table>

Variables = 3
Records Read = 1584

Table Number 1 Michigan.all
Mon Nov 18 16:40:10 1996
FARS 1975, MICHIGAN CRASHES (NO DUP ON RECORD NUMBER)
Bivariate Frequencies and Percentages
Column Variable 46: TOTAL KILLED IN ACC
Row Variable 19: FIRST HARMFUL EVENT
Options: FREQ ROW% NOMD% MD

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>6</th>
<th>7</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>1025.</td>
<td>83.</td>
<td>19.</td>
<td>3.</td>
<td>0.</td>
<td>0.</td>
<td>1130.</td>
</tr>
<tr>
<td>ROW %</td>
<td>90.7</td>
<td>7.3</td>
<td>1.7</td>
<td>0.3</td>
<td>0.0</td>
<td>0.0</td>
<td>100.0</td>
</tr>
<tr>
<td>29</td>
<td>406.</td>
<td>38.</td>
<td>8.</td>
<td>0.</td>
<td>1.</td>
<td>1.</td>
<td>454.</td>
</tr>
<tr>
<td>ROW %</td>
<td>89.4</td>
<td>8.4</td>
<td>1.8</td>
<td>0.0</td>
<td>0.2</td>
<td>0.2</td>
<td>100.0</td>
</tr>
<tr>
<td>TOTAL</td>
<td>1431.</td>
<td>121.</td>
<td>27.</td>
<td>3.</td>
<td>1.</td>
<td>1.</td>
<td>1584.</td>
</tr>
<tr>
<td>ROW %</td>
<td>90.3</td>
<td>7.6</td>
<td>1.7</td>
<td>0.2</td>
<td>0.1</td>
<td>0.1</td>
<td>100.0</td>
</tr>
</tbody>
</table>
Table Number 2  Mich.bad.weather
Mon Nov 18 16:40:10 1996
FARS 1975, MICHIGAN CRASHES (NO DUP ON RECORD NUMBER)
Bivariate Frequencies and Percentages
Column Variable 46:TOTAL KILLED IN ACC
   Row Variable 19:FIRST HARMFUL EVENT
   FV1 Variable 33:WEATHER/ATMOSPHERE - Range 1 To 5
Options:  FREQ  ROW%  NOMD%  MD

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>TOTAL</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>154.</td>
<td>16.</td>
<td>5.</td>
<td>175.</td>
</tr>
<tr>
<td>ROW %</td>
<td>88.0</td>
<td>9.1</td>
<td>2.9</td>
<td>100.0</td>
</tr>
<tr>
<td>29</td>
<td>66.</td>
<td>7.</td>
<td>1.</td>
<td>74.</td>
</tr>
<tr>
<td>ROW %</td>
<td>89.2</td>
<td>9.5</td>
<td>1.4</td>
<td>100.0</td>
</tr>
<tr>
<td>TOTAL</td>
<td>220.</td>
<td>23.</td>
<td>6.</td>
<td>249.</td>
</tr>
<tr>
<td>ROW %</td>
<td>88.4</td>
<td>9.2</td>
<td>2.4</td>
<td>100.0</td>
</tr>
</tbody>
</table>

**UNIVAR Command**

Keyword: OUTPUT
Modifier: APPEND

The UNIVAR command generates one-way frequency distributions and simple descriptive statistics for selected variables in the active data set. Any number of variables may be used, with any number of levels per variable and frequencies may be weighted by the values of a selected variable. Univariate frequency information produced by the command includes, depending on the options specified, the code value, a label for this code value, the occurrence frequency of the code value, and its percentage of the total frequency for the variable. Simple statistical information that may optionally be generated includes the minimum and maximum code values for the variable, the average value, the standard deviation, and the number of occurrences of missing data.

**UNIVAR Parameter Keywords and Modifiers**

<table>
<thead>
<tr>
<th>Required parameters:</th>
</tr>
</thead>
<tbody>
<tr>
<td>VARIABLE=Variable list</td>
</tr>
</tbody>
</table>

**COLUMN**

Univariate frequencies are printed in a columnar format with one line for each code value.

DEFAULT: Output print width is set by the UNIVAR command PRINTWID option, or by the global SET command PRINTWID setting. A print width of 79 characters is the default.

**FREQ**

Univariate frequencies are calculated for all variable values in the variable list.

DEFAULT: This is the default. When the FREQ modifier is omitted, however, frequencies are not calculated if the STAT modifier is specified.
FREQ%
Univariate frequencies are calculated for all variable values in the variable list, along with the percentage that each value represents of the total records satisfying the global filter criteria.
DEFAULT: No percentages are calculated.

HELP
A description of options is displayed. If present, all other parameters entered on the same line (except STOP) are ignored.
DEFAULT: No description is displayed.

LABELS=</path/label file>  EXAMPLE: LAB=/CDS/MYCDSLAB
Code value labels for the data set being analyzed are found in the file specified. See Appendix A for label file construction procedures.
DEFAULT: The label file name is supplied by the system if labels are implemented for the active data set.

NOLABEL
No code value labels are printed.
DEFAULT: Code value labels are printed for each variable, if the labels exist for that variable.

PAGE
The user-supplied title is printed at the top of each page, along with either the phrase "UNIVARIATE FREQUENCY DISTRIBUTION" or "UNIVARIATE STATISTICS" on the following line.
DEFAULT: No header is printed for new pages.

PRINTWID=<columns>  EXAMPLE: PRINT=130
The PRINTWID option designates a value for the number of columns used for program output.
DEFAULT: The default value is set by the SET command PRINTWID option which in turn defaults to 79.

STAT
Descriptive statistics are calculated for all variables in the variable list. When the STAT modifier is used frequencies are not printed unless the FREQ modifier is also given.
DEFAULT: No statistics are computed.

STOP
The UNIVAR command is terminated.

STOP=<number>  EXAMPLE: STOP=100
Processing is terminated after the specified number of records have been processed. If the CASEVAR option is in effect, the STOP count is a lower limit, and all records in the current case are used before processing is terminated.
DEFAULT: All records that satisfy the global filter are processed.

VARIABLE={<variable list>|ALLV|ALLNV}  EXAMPLE: VA=109-114,204
Specifies the variable(s) for which univariate frequency information or statistics will be generated. The option ALLV specifies that all variables are to be used. The option ALLNV specifies that all numeric variables are to be used. Since frequencies or statistics are not generated for alphabetic variables, ALLV differs from ALLNV only in the inclusion of recoded alphabetic variables.
DEFAULT: None.

WV=<variable>  EXAMPLE: WV=240
Record counts are weighted by the value of the specified variable. If the value of the weight variable is zero or equal to the missing data value, the record is skipped.

DEFAULT: No weighting is performed.

**UNIVAR** Input and Output Examples

Example of the **UNIVAR** Parameter Statement
("***" at far right = user supplied input)

**UNIVAR** Options (or **STOP** to terminate)
var=101,205,311-321 col stat freq stop=100 ***

NOTE: The example above produces one-way frequency of occurrence information for variables 101, 205, and 311 through 321.

Example of a **UNIVAR** Setup File

data k=FARS75AC
univar out=farsfreq append
include v3=26
FARS 1975, Michigan crashes
var=19,33,46 freq% stat printwid=60

The file above, run with the "SET INPUT=" feature of ADAAS, produced the output on the next page (in addition to global information that precedes the output such as a dictionary listing). The categories for the descriptive statistics are defined as follows:

<table>
<thead>
<tr>
<th><strong>VAR</strong></th>
<th>The variable number</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Minimum</strong></td>
<td>The minimum code value</td>
</tr>
<tr>
<td><strong>Maximum</strong></td>
<td>The maximum code value</td>
</tr>
<tr>
<td><strong>MD1</strong></td>
<td>The number of records with a missing data code value</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td>The mean (or average) code value</td>
</tr>
<tr>
<td><strong>Standard Dev</strong></td>
<td>The standard deviation</td>
</tr>
</tbody>
</table>

The Average and Standard Deviation are printed in "scientific notation". In this format, the number is printed as a decimal number between 0.1 and 1 together with an exponent indicating how many places the decimal place must be shifted. The number is printed as "0.ND-MM" or "0.ND-MM" where "D+MM" means shift the decimal MM positions to the right, and "D-MM" means shift the decimal MM positions to the left. "D+00" means do not shift the decimal place. For example:

0.1347980D+02 => 13.4798
0.1347980D+00 => 0.134798
0.1347980D-03 => 0.000134798

The categories for the frequency information are defined as follows:

<table>
<thead>
<tr>
<th><strong>CODE</strong></th>
<th>The variable's code values</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LABEL</strong></td>
<td>The code value's associated label</td>
</tr>
<tr>
<td><strong>FREQ</strong></td>
<td>The code value's frequency of occurrence</td>
</tr>
<tr>
<td><strong>PRCNT</strong></td>
<td>The percentage that this frequency represents of the total number of records (&quot;N&quot;)</td>
</tr>
</tbody>
</table>

Example of **UNIVAR** statistics output
Program: UNIVAR (Univariate Frequencies)
Mon Nov 18 17:06:30 1996

Data Key = /usr/umtri/tdc/apps/adaas/doc/library.dat/FARS75AC

Filter = INCLUDE V3=26

Title = FARS 1975, MICHIGAN CRASHES

Listing of Filter Variables Requested:
Group Numbr Name       Loc Wid T  Dec     Missing
3    STATE             11   2 C               99

Listing of UNIVAR Command Variables Requested:
Group Numbr Name       Loc Wid T  Dec     Missing
19   FIRST HARMFUL EVENT 44   2 C               99
33   WEATHER/ATMOSPHERE  61   1 C                9
46   TOTAL KILLED IN ACC 87   2 C               99

Options Specified:
FREQ
STAT
FREQ%
PRINTWID = 60
LABELS = /usr/umtri/tdc/data/fars/fs75lab
VARIABLE = 19,33,46

Variables = 3
Records Read = 1584

Univariate Frequency Distribution
Mon Nov 18 17:06:44 1996
FARS 1975, MICHIGAN CRASHES
Number of records = 1584

***Variable 19 FIRST HARMFUL EVENT

<table>
<thead>
<tr>
<th>Code</th>
<th>Label</th>
<th>Freq</th>
<th>Prcnt</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Overturn</td>
<td>122</td>
<td>7.7</td>
</tr>
<tr>
<td>3</td>
<td>Immersion</td>
<td>1</td>
<td>0.1</td>
</tr>
<tr>
<td>5</td>
<td>Fell from veh</td>
<td>9</td>
<td>0.6</td>
</tr>
<tr>
<td>7</td>
<td>Other noncollission</td>
<td>4</td>
<td>0.3</td>
</tr>
<tr>
<td>8</td>
<td>Pedestrian</td>
<td>300</td>
<td>18.9</td>
</tr>
<tr>
<td>9</td>
<td>Pedalcycle</td>
<td>63</td>
<td>4.0</td>
</tr>
<tr>
<td>10</td>
<td>Railway train</td>
<td>20</td>
<td>1.3</td>
</tr>
<tr>
<td>11</td>
<td>Animal</td>
<td>5</td>
<td>0.3</td>
</tr>
<tr>
<td>12</td>
<td>Veh in transport</td>
<td>605</td>
<td>38.2</td>
</tr>
<tr>
<td>13</td>
<td>Veh in othr road</td>
<td>1</td>
<td>0.1</td>
</tr>
<tr>
<td>14</td>
<td>Parked vehicle</td>
<td>24</td>
<td>1.5</td>
</tr>
<tr>
<td>15</td>
<td>Othr obj-unfixed</td>
<td>2</td>
<td>0.1</td>
</tr>
<tr>
<td>16</td>
<td>Brdg or overpass</td>
<td>41</td>
<td>2.6</td>
</tr>
<tr>
<td>17</td>
<td>Building</td>
<td>5</td>
<td>0.3</td>
</tr>
<tr>
<td>18</td>
<td>Culvert/ditch</td>
<td>28</td>
<td>1.8</td>
</tr>
<tr>
<td>19</td>
<td>Curb or wall</td>
<td>8</td>
<td>0.5</td>
</tr>
<tr>
<td>20</td>
<td>Divider</td>
<td>1</td>
<td>0.1</td>
</tr>
<tr>
<td>21</td>
<td>Embankment</td>
<td>7</td>
<td>0.4</td>
</tr>
</tbody>
</table>
**Variable 33 WEATHER/ATMOSPHERE**

| Code | Label       | Freq | Prctn | * Code | Label       | Freq | Prctn | *
|------|-------------|------|-------|--------|-------------|------|-------|-----
| 0    | None        | 1333 | 84.2  | * 1    | Rain        | 153  | 9.7   | *   |
| 3    | Snow        | 79   | 5.0   | * 4    | Fog/dust    | 17   | 1.1   | *   |
| 9    | Unk         | 2    | 0.1   | *      |             |      |       |     |

**Variable 46 TOTAL KILLED IN ACC**

| Code | Freq | Prctn | * Code | Freq | Prctn | *
|------|------|-------|--------|------|-------|-----
| 1    | 1431 | 90.3  | * 2    | 121  | 7.6   | *   |
| 3    | 27   | 1.7   | * 4    | 3    | 0.2   | *   |
| 6    | 1    | 0.1   | * 7    | 1    | 0.1   | *   |

Univariate Statistics
Mon Nov 18 17:06:44 1996
FARS 1975, MICHIGAN CRASHES
Number of records = 1584

<table>
<thead>
<tr>
<th>Var</th>
<th>Minimum</th>
<th>Maximum</th>
<th>MD1</th>
<th>Average</th>
<th>Standard Dev</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>1</td>
<td>29</td>
<td>0.01347980D+02</td>
<td>0.7362612D+01</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>0</td>
<td>4</td>
<td>2.02895070D+00</td>
<td>0.7966588D+00</td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>1</td>
<td>7</td>
<td>0.01123106D+01</td>
<td>0.4302186D+00</td>
<td></td>
</tr>
</tbody>
</table>
Appendix A

LIBRARY Program

Version 02-14-1999

The ADAAS program stores all information necessary to access data sets in a "library" file. By storing dictionary and data file names, path names, etc., in this library, only a keyword designation for the data set need be used for access. The LIBRARY program provides for the maintenance of these library files. The program is executed by the UNIX command "library", and is terminated by the library program command "stop". When program execution begins, the current library file managed by the program is the "system library" unless another library is specified in the initialization file "library.ini" (see below).

The maximum length of any program input line is 1000 characters. Input may be entered in shorter segments by ending each segment with a dash ("-") and continuing on the next line (see the example section for the use of this feature). Blank input lines are ignored and comment lines may be entered by using a semi-colon (";") as the first non-blank character.

There is one special library (known as the "system" library) that contains definitions of data sets maintained by the Transportation Data Center. This library is "$ADAASHOME/doc/library.dat". However, there may be other special purpose libraries created by the Transportation Data Center; or private libraries created by other users who may wish to access data sets they have created by subsetting system sets, or by other means. There is no structural difference between the system library and private libraries. The structure of library files is documented at the end of this Appendix.

Due to the organization of the UNIX file system, data, dictionary, or label files names in a library may contain multiple occurrences of very long path names. To facilitate the entry and maintenance of these path names, a cross reference (or XREF) file may be used to translate an alias name used in library files to a full path name contained in the XREF file. The structure of these XREF files is documented at the end of this Appendix.

All libraries are created by the program, if necessary, and are given the name specified by the current path and name values. The LIBRARY program does not perform any file permission, so access to the library file must be controlled with the UNIX chmod command.

A "library.ini" initialization file may be created on the user's home directory with lines that contain SET commands (e.g.,"set libpath=~/dbx/sysfiles" or "set maxrecs=200"). This command file will be read when the library program is first executed to set the current library (through the LIBPATH and LIBNAME keywords), or the maximum number of library records (through the MAXRECS keyword).

The current library file may be changed at any time with the SET command.
***WARNING***:
All operations made on the library are performed on an image of the records in memory. Any changes that are made are not written to the actual library file until a SAVE or STOP command is issued. Aborting the program may consequently result in the loss of recent changes made to library records. Periodic use of the SAVE command can reduce the potential for loss.

** LIBRARY Commands and Options **

Commands: ADD, ARCHIVE, CHECK, DELETE, HELP, LIST, MODIFY, RESTORE, SAVE, SET, STOP

Keywords: ARCHIVE, COMPRESS, DATA1, DATA2, DATA3, DATA4, DATA5, DATA6, DATA7, DATA8, DATA9, DATAPATH, DICT, KEYWORD, LABELS, LIBNAME, LIBPATH, MAXRECS, OUTPUT, PATH, RANGE

Modifiers: APPEND, NOSAVE

Available Library commands are described below with any applicable keywords or modifiers listed after each description. Following this, full documentation of the keywords and modifiers is provided.

** LIBRARY Commands **

The letters necessary for the unambiguous recognition of the command are shown in upper case.

ADD   A new data set is added to the library with characteristics specified by an accompanying set of keywords.
KEYWORDS: COMPRESS, DATA1, ..., DATA9, DATAPATH, DICT, KEYWORD, LABELS, PATH

ARCHIVE  The data set specified by KEYWORD is flagged as "archived" indicating that it is not directly available for use.
KEYWORDS: KEYWORD

CHECK    Checks the structure and consistency of the library.

DELETE   The data set specified by KEYWORD is deleted from the library.
KEYWORD: KEYWORD

HELP     Prints the available commands, keywords, and modifiers.

LIST     Prints information for all data set entries in the library; or for a data set defined by KEYWORD; or for a group of descriptors selected by RANGE. The list may be printed directly, or routed to a file with the OUTPUT keyword.
KEYWORDS: KEYWORD, OUTPUT, RANGE

MODIFIERS: APPEND

MODIFY   The data set specified by KEYWORD is modified to reflect the new values specified by the keywords supplied.
KEYWORDS: COMPRESS, DATA1, ..., DATA9, DATAPATH, DICT, KEYWORD, LABELS, PATH

RESTORE  The data set specified by KEYWORD is flagged as "on-line"
indicating that it is available for use.
KEYWORDS: KEYWORD

SAve Saves the current image of the library.

SET Sets the value of program parameters. If SET is issued with no options, then the current value of LIBNAME, LIBPATH, and MAXRECS is displayed.
KEYWORDS: LIBNAME, LIBPATH, MAXRECS

STOP Saves the current image of the library if any changes have been made, and terminates the program. If the NOSAVE modifier is used, the image of the library records in memory is NOT saved.
MODIFIERS: NOSAVE

** LIBRARY Command Keywords **

The letters necessary for the unambiguous recognition of the keyword are shown in upper case.

Archive=[O|A] EXAMPLE: A=A
Specifies whether the data files are on-line ('O') or archived ('A').
DEFAULT: Files are on-line ('O')

Compress=[U|C] EXAMPLE: C=U
Specifies whether the data files are uncompressed ('U') or compressed ('C').
DEFAULT: Files are uncompressed ('U')

DATAn=<filename>|NONE EXAMPLE: DA=mi78dat
There are nine keywords (DATA1, ..., DATA9) to specify up to nine data files (without the path) comprising the data set. "DATA1" may be abbreviated "DA"; for all others, the full name must be used. The value "NONE" may be used with DATA2 through DATA9 to delete a data file entry. The maximum data file name length is 64 characters.
DEFAULT: None

DATAPath=<datafile path name> EXAMPLE: DATAPATH=/rmj/data
Defines a common path to all the data files defined by the keywords DATA1 through DATA9. The maximum length of a path name is 256 characters.
DEFAULT: Data files are located on the current directory (i.e., './')

DIct=<filename> EXAMPLE: DI=mi78dic
Specifies the name of the dictionary file (without the path). The maximum length of the dictionary file name is 64 characters.
DEFAULT: None

Keyword=<data set descriptor> EXAMPLE: KEY=fars8sub
Identifies the data set to be processed. The descriptor consists of one to eight printing characters.
DEFAULT: None

LABels=<file name>/NONE EXAMPLE: L=mi78lab
Specifies the name of the file (without the path) where code value labels are stored. Any label file that is compatible
with the data set being managed may be used. If the value "NONE" is specified, the label file name is deleted from the data set specification. The maximum label file length is 64 characters.
DEFAULT: The data set will have no associated label file.

LIBName=<library file name> EXAMPLE: LIBN=fars_subsets
Specifies the file name for the library to be managed. The system library file name (library.dat) may be specified by "*". The maximum length of the library name is 64 characters.
DEFAULT: The current name.

LIBPath=<library path name> EXAMPLE: LIBP=~tdczzz/sysfiles
Specifies the path name to the directory containing the library to be managed. The maximum length of a path name is 256 characters. The system library path ($ADAASHOME/doc) may be specified by "*".
DEFAULT: The current path.

MAXrecs=<an integer>
Specifies the maximum number of library records than can be read and processed by the program.
DEFAULT: The maximum number of records is 100.

Output=<file name> EXAMPLE: OUT=fars8sub.lst
Specifies the name of the file where the LIST command output will be written.
DEFAULT: Output is written to the screen.

Path=<a path name> EXAMPLE: PATH=/home/dbx/data
Specifies the path name for the directory where the data, dictionary, and label files are located. The maximum length of a path name is 256 characters. Use of PATH overrides any DATAPATH specification.
DEFAULT: Files are located on the current directory ('./')

Range=<key1>,<key2> EXAMPLE: RA=F,G
Specifies the range of data set descriptors to be printed in the LIST command output. If only one value is specified, then all descriptors that begin with the letters supplied will be listed. The value "FIRST" may be used for <key1> to indicate the first entry, and "LAST" may be used for <key2> to indicate the last entry.
DEFAULT: Definitions for all descriptors will be listed (i.e., RANGE=FIRST,LAST).
** LIBRARY Command Modifiers **

The letters necessary for the unambiguous recognition of the modifier are shown in upper case.

** Append **
New information is appended to the output file.
DEFAULT: The output file is overwritten.

** Nosave **
The core image of the library file is not saved before program termination.
DEFAULT: The current library image is saved.

** Examples of LIBRARY Commands **

1) set libp=-tdczzz/dbx/sysfiles libn=farslib maxrecs=500

2) set

3) ADD KEY=f86to88 LAB=FARS80.LAB path=/home/dbx/data -
   DICT=86fatdic DATA1=86fatdat -
   DATA2=87fatdat DATA3=88fatdat

4) ADD KEY=80acc DI=/home/dbx/accdik da=accdat datapath=/home/dbx/data

5) DEL k=TESTDATA

6) list OUT=flist.sav RANGE=fars80,fars86

7) MOD KEY=nass1 DI=nassdic DA=nassdat labels=none data2=none

Library File Structure

The structure of records found in library files is shown below.

<table>
<thead>
<tr>
<th>Line</th>
<th>Col</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>DSDESC*8</td>
<td>Eight character data set keyword</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>Blank</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>ARCHIVE*1</td>
<td>Archive flag (O=Online, A=Archived)</td>
</tr>
<tr>
<td>1</td>
<td>11</td>
<td>COMPRESS*1</td>
<td>Compression (U=uncompressed, C=Compressed)</td>
</tr>
<tr>
<td>1</td>
<td>12</td>
<td>SPARE1*1</td>
<td>Spare flag ('-')</td>
</tr>
<tr>
<td>1</td>
<td>13</td>
<td>SPARE2*1</td>
<td>Spare flag ('-')</td>
</tr>
<tr>
<td>1</td>
<td>14</td>
<td>Blank</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>15</td>
<td>NUMFIL*1</td>
<td>Number of data files (1 - 9)</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>LABFIL*320</td>
<td>Full path label file name (or 'None')</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>DICTNAME*320</td>
<td>Full path dictionary file name</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>DATANAME(1)*320</td>
<td>Full path data file #1 name</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>DATANAME(2)*320</td>
<td>Full path data file #2 name</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>DATANAME(3)*320</td>
<td>Full path data file #3 name</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>DATANAME(4)*320</td>
<td>Full path data file #4 name</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>DATANAME(5)*320</td>
<td>Full path data file #5 name</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>DATANAME(6)*320</td>
<td>Full path data file #6 name</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>DATANAME(7)*320</td>
<td>Full path data file #7 name</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>DATANAME(8)*320</td>
<td>Full path data file #8 name</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>DATANAME(9)*320</td>
<td>Full path data file #9 name</td>
</tr>
</tbody>
</table>
Cross Reference Files

A cross reference (XREF) file may be used with the library program to provide for the translation of path name aliases found in a library file. The XREF file should have the same name as the library file it is attached to with the addition of an ".xrf" extension. For example, if the library name is "mylib", then the XREF file must be named "mylib.xrf". If the library file is named "library.dat", however, the XREF file name must be "library.xrf". The XREF file must have the same permit status as the library file.

Each entry in the XREF file consists of a path keyword alias name that begins with a "$" sign in column one, followed by at least one blank, then the path name associated with the keyword.

The library file may contain these alias names at the *beginning* of any label, dictionary, or data name. The DATA command will substitute the path name from the XREF file for the alias.

For instance, if the XREF file contains the entry:

```
$XPATH /net/tdcnet2/nfsvol/home/tdcxxx
```

then the library entries:

```
$XPATH/ivor/ivorlab
$XPATH/ivor/ivordict
$XPATH/ivor/ivordata
```

will be translated by the DATA command to the real paths:

```
/net/tdcnet2/nfsvol/home/tdcxxx/ivor/ivorlab
/net/tdcnet2/nfsvol/home/tdcxxx/ivor/ivordict
/net/tdcnet2/nfsvol/home/tdcxxx/ivor/ivordata
```
Appendix B

Label File Structure
Version 07-18-94

The label file is a BINARY, DIRECT FORTRAN file with a RECL of 20. The first line gives general information on the number of variables, etc. Lines 2 to LASTLINE are variable descriptor records for each variable followed by the code value/code label pairs for this variable.

Line 1
Fullword 1 - NUMVARS - Total number of variables
Fullword 2 - MAXVAR  - Maximum variable number
Fullword 3 - MAXCDNUM - Maximum number of code values for any variable
Fullword 4 - LASTLINE - The line number of the last line in the file

Variable descriptor lines (Line 2 + ...)
Fullword 1 - VARNUM  - Variable number
Fullword 2 - MAXLEN  - Maximum length of code labels (1 - 16)
Fullword 3 - MAXCOD  - Maximum code value
Fullword 4 - NUMCODES - Number of code values

Label descriptor lines:
Fullword 1 - CODVAL  - Code value
Char*16    - LABNAME  - Label for this code value
Appendix C
System Error Messages
Version 10-24-2000

When errors occur during the processing of one of the system commands, a message (or messages) is displayed to alert the user to the nature of the problem. Ideally, these messages should be self-explanatory and should clearly communicate the nature of the problem. Often, however, the problem is sufficiently complex that it cannot be easily explained in a short phrase. In addition, users are (by design) unaware of internal processes, so that the messages, while clearly describing the problem, are often incomprehensible to the user. However, it is hoped that an incomprehensible message is better than no message at all. This appendix contains an explanation of the error messages that are generated by ADAAS to serve as an aid in understanding and correcting the manifold problems that can always occur in the operation of a computer system.

The list of error messages given below is a comprehensive list of possible messages from the ADAAS program and its associated subroutines, but does not represent the full range of messages that may be obtained during program execution. The reason for this apparent paradox is that many messages are generated by UNIX operating system or by other subroutines that are not Data Center programs. If non-ADAAS messages do occur, however, they will generally be accompanied by one of the messages given in this appendix. For the sake of completeness, several warning messages are included. These messages simply warn users of conditions that may not be expected; but which are not errors.

For the most part, messages are listed below in alphabetical order. Several of the messages begin with an error specific text and hence cannot be sorted: these messages are listed first. To conserve space, several groups of messages are listed under a group heading (i.e., "Numeric conversion", "Problems with Input/Output", and "Storage", etc.). Some of the messages come from support subroutines that are called by ADAAS, but are also generally used in a variety of other programs. As a result, many of the possible messages that can be generated by these routines should not ever occur in the ADAAS environment. They have been included in this list for the sake of completeness, and to anticipate the all too frequent penchant of programmers to overlook all the possible consequences of program logic.

As a final distinction, error messages may be classified as those that document user-induced problems (i.e., bad input information), and those that document program logic flaws (i.e., conditions that should never occur but perversely have done so anyway, and decision logic errors). It is hoped that this appendix will give the user sufficient information to correct errors of the first type. Errors of the second type should be reported to the Data Center with as much backup information as possible, since program "bugs" can only be fixed if they are known.
** Unsorted Error Messages **

'<FILENAME> is empty'

Program: opendat.f
The file <FILENAME> that has been specified as an input file does not contain any information.

'<KEYWORD>: Access cannot be established'

Program: datacmd.f
Access to the data set denoted by <KEYWORD> cannot be established. This message should be preceded by other messages that detail the reason.

'"<TEXT>" is invalid'

Program: frtget.f
The characters shown between the quotes are the segment of a FILTER or RECODE statement found to be in error. The exact location of the problem is shown by the pointer "^".

'"<TEXT>" is not a recognizable option.'
'Enter a replacement (OR "C" for cancel, "D" for delete, "H" for help).'

Program: keyscn.f
The program has been unable to recognize a user-supplied input keyword or modifier as a valid parameter. The message is therefore a request to re-enter the incorrect parameter specified by the message. If a replacement is entered, it should be for the entire parameter. For instance, if the message indicates that "KEY=BADFILE" is invalid, then "KEY=GOODFILE" should be entered and not simply "GOODFILE". If "D" is entered, the parameter will be deleted from the input. If "C" is entered, processing of the entire line will be aborted. If "H" is entered, a list of valid keywords and modifiers will be displayed.

'"<TEXT>" is not a recognizable command.'
'Enter a replacement (OR "C" for cancel, "H" for help).'

Program: cmdscn.f
The command entered cannot be recognized as a valid system command. The message is therefore a request to re-enter the command. If "C" is entered, the command will be aborted. If "H" is entered, a list of valid commands will be displayed.

** Sorted Error Messages **

'Actual STOP count will be between <NUMBER1> and <NUMBER2>'

Program: list.f
If the STOP=N option is used in combination with CASEVAR, the actual number of records processed may exceed the specified stop count
value since all records in the last case must be completely processed. 
\(<\text{NUMBER1}>\) and \(<\text{NUMBER2}>\) give the range of the actual STOP count.

\textbf{'adaas.ini: Line ignored - not a \texttt{SET} command'}

Program: adaas.f
A command line in the adaas.ini file was not a \texttt{SET} command and was 
consequently ignored.

\textbf{'Alphabetic recoding requires the (ELSE) operand'}

Program: frtget.f
If alphabetic variables are recoded to numeric values, the ELSE 
operand is required. Use "help recode" for help with the RECODE 
statement and the ELSE operand.

\textbf{'At record Number = <NUMBER>}'
\textbf{'At input record Number = <NUMBER>}'
\textbf{'At output record Number = <NUMBER>}'

Program: list.f, subset.f, twoway.f, univar.f
These messages indicate the number of the input record where a 
problem occurred. This message should be preceded by other messages that 
detail the nature of the problem.

\textbf{'Can’t delete <filename>}'

Program: adaas.f
The specified file could not be deleted. Lack of proper access is 
the most likely cause of this problem.

\textbf{'Can't find <KEYWORD> in the library on <PATHNAME>}'

Program: datacmd.f
The specified keyword could not be located in current library. Make 
sure that you are accessing the proper library and use \texttt{SET} 
\texttt{LIBPATH=<path>"} to change the current library.

\textbf{'CASEVAR cannot be used with DUPVAR'}

Program: frtget.f
The CASEVAR and DUPVAR options interact strongly and cannot both be 
specified in the same operation.

\textbf{'COPYDICT: Error in dictionary Tcard'}

Program: datacmd.f
The input dictionary contains a T card with an invalid value. This 
message should be preceded by other messages that detail the nature of the problem.

\textbf{'COPYDICT: No variables in the dictionary'}

Program: datacmd.f
There are no variables (specified by T cards) in the input dictionary.

'DATACMD: Error in CNVTPATHNAMES'
'DATACMD: Error in GETLRECL'
'DATACMD: Error in USELST'

Program: datacmd.f
An error in one of the indicated subroutines was encountered while processing the data command request. The message should be preceded by additional messages detailing the problem.

'Data file is empty'

Program: datainpt.f
The data file for the current data set is empty.

'Data Set keyword too long'

Program: datacmd.f
The length of the keyword used in the DATA command was greater than eight characters.

'DELIMITD file name too long'

Program: twoway.f
The file name given for the output of the delimited option is too long.

'Delimiter is not a printable character'

Program: subset.f
The delimiter character specified by the SUBSET keyword "DELIMITR" is not a printable ASCII character.

'Dictionary specifies a record length greater than the data file record length'

Program: datacmd.f
Some variables defined in the input dictionary have a record location that is larger than the record length of the data file.

'DUPVAR cannot be used with CASEVAR'

Program: frtget.f
The CASEVAR and DUPVAR options interact strongly and cannot both be specified in the same operation.

"ELSE" must be the last recode set'

Program: frtget.f
If the ELSE operand is used in a RECODE statement to recode all unspecified code values, then it must be placed at the end of the RECODE statement.

'End-Of-File on input'

Program: readin.f
An End-of-File condition (i.e., no more input available) was sensed while reading program input. This usually occurs with a "SET INPUT=<file>" command when the information in <file> is exhausted before a "SET INPUT=ME" is processed.

'Error return from SYSTEM'

Program: adaas.f
Processing of a UNIX command in the SYSTEM subroutine generated an error return from the UNIX operating system. The command may be one specified by the ADAAS command $<UNIXCMD>, or may be generated by the ADAAS program.

'Field width too large for variable'

Program: frtget.f
The number of digits supplied for a code value in a FILTER or RECODE statement is greater than the field width of the variable. That is, the number supplied is too large to be a possible value of the variable referenced.

'File name is too long'

Program: subset.f
The name specified for the SUBSET output dictionary or data file is longer than the current limit of 320 characters.

'A Filter statement has already been entered'

Program: frtget.f
A FILTER statement has been entered previously, and only one statement is permitted.

'GROUP value out of range'

Program: vartobin.f
The GROUP value for a variable is less than zero, or greater than 32,267.

'HELP is not available'

Program: adaas.f
There is no HELP file for the item requested.

'Improper placement of &'
'Improper placement of colon'
'Improper placement of comma'
'Improper placement of dash'
'Improper placement of equal sign'
'Improper placement of parentheses'
'Improper placement of prime'

Program: frtget.f, numlist.f, numlistp, vartobin.f, varlist.f, vrange.f

The character indicated by the message was used in a context that is not proper for the application. If the syntax error occurred as part of a keyword right-hand-side value, the message will be followed by "Error occurred at column N" where column "N" is measured from the beginning of the expression - not from the beginning of the entire input line. If the syntax error occurred as part of a FILTER or RECODE statement, the location of the error will be indicated by a message showing the part of the statement in error together with a flag showing the exact location of the problem.

'Information file not available'

Program: adaas.f
A system information file could not be located or opened for printing. Please notify TDC of this problem.

'INPUTLEN must be positive'

Program: adaas.f
The value of the SET command INPUTLEN option specified must be greater than zero.

'INPUTLEN must not be used with other SET options'

Program: adaas.f
The INPUTLEN option changes the input buffer and must not be used in combination with any other SET options

'Input Line length greater than <NUMBER>'

Program: readin.f
The length of the input line was greater than the input buffer could accommodate. The current length of the buffer is given by <NUMBER>.

'Invalid header expansion bytes'

Program: chkheadr.f
The first two characters of the data file header must be "00".

'Invalid header for <FILENAME>'

Program: opendat.f
The data set file header has an invalid header - an eight byte record specifying the record format and record length. This message
should be preceded by other messages that detail the nature of the problem.

'Invalid header record length'
Program: chkheadr.f
The record length specified by the data file header is not a character numeric number.

'Invalid label file'
'If this a system data set please notify the Data Center'
'If not, check your library'
Program: list.f, twoway.f, univar.f
The label file specified for the current data set is not a valid label file. If you are accessing a data set in the system library, notify the Data Center of this problem.

'Invalid library path name'
Program: adaas.f, datacmd.f
The path name supplied for the library is not a valid path.

'Invalid RAND% value'
'Invalid SKIP value'
Program: subset.f
The values entered by the keywords RAND% or SKIP keywords are not valid for the program. Use "help subset" for proper values for these parameters.

'Invalid variable number'
Program: frtget.f
The variable number specified for a filter or recode variable is less than one, or greater than 32767.

'Label file name too long'
Program: list.f, twoway.f, univar.f
The name specified for the LABELS keyword is longer than the current limit of 320 characters.

'Label file is empty'
Program: list.f, twoway.f, univar.f
The specified label file contains no information.

'LABLRECV value must be "ON" or "OFF"
Program: adaas.f
An invalid value for the LABLRECV keyword was entered.
'LIBNAME length greater than <NUMBER>'
'LIBPATH length greater than <NUMBER>'

Program: adaas.f
The name supplied for the library file name or the library file path were longer that the allowed limit indicated by NUMBER.

'Library path name too long'

Program: datacmd.f
The name specified for the library path is longer than the current limit of 256 characters.

'MAXFVARS must be positive'

Program: adaas.f
The maximum number of filter variables entered with the MAXFVARS option must be a positive number.

'Maximum field width for CASEVAR variables is 16'
'Maximum field width for Dupvar variables is 16'

Program: frtget.f
The maximum field width of alphabetic variables uses in CASEVAR or DUPVAR statements is 16.

'Maximum INPUTLEN value is <NUMBER>,'

Program: adaas.f
The value of the specified SET command INPUTLEN option must not be greater than the value specified by NUMBER.

'Maximum PRINTWID value is <NUMBER>,'

Program: adaas.f, list.f, twoway.f, univar.f
The maximum available print width is specified by <NUMBER>.

'MAXRECS must be positive'

Program: adaas.f
The maximum number of records in a case must be a positive number.

'Minimum value greater than maximum'
'Error occurred at column <NUMBER>,'

Program: vrange.f
A keyword right-hand-side of the form "VAL:MIN-MAX" occurred where the minimum value was greater than the maximum value. The error occurred at the column indicated by <NUMBER> measured from the beginning of the expression - not from the beginning of the entire input string.
'Misplaced "AND"'
'Misplaced "OR"'

Program: frtget.f
The logical connectives "AND" or "OR" were not found in an expected position in a filter statement. Use "help filter" for help with the syntax of filter statements.

'Missing ")=" after code value list'
'Missing connective "AND" or "OR"
'Missing equal sign after "V"
'Missing parentheses'
'Missing variable prefix "V"

Program: frtget.f
The characters indicated were not found in their expected position in a filter or recode statement. Use "help filter" or "help recode" for help with the syntax of filter or recode statements.

'More than <NUMBER> filter variables'
'Use "SET MAXFVARS=N" to redimension for N variables'

Program: frtget.f
Too many filter variables were used in the FILTER statement. The current limit is 50 variables. This limit may be changed by the SET command's MAXFVARS option. Use "help filter" for help with the syntax of filter statements.

'More than <NUMBER> recode sets'
'More than <NUMBER> RECODE statements'

Program: frtget.f
Too many recodes were used; or there were too many recode sets in a recode statement. The current limits are 10 recode statements, and 50 recode sets. Use "help recode" for help with the syntax of recode statements.

'More than <NUMBER> tables specified'
'Processing continues for the first <NUMBER> tables'

Program: twoway.f
Input to the TWOWAY program specified more tables than the program can process. The current maximum number of tables is given by <NUMBER>.

'No code value list supplied'

Program: frtget.f
The logical end of a code value list was detected, but no code value list was found. Use "help filter" or "help recode" for help with the syntax of filter or recode statements.

'No column variable specified'

Program: twoway.f
No column variable was specified for the twoway table whose parameters were previously entered.

'No Data Set keyword specified'
Program: datacmd.f
No data set keyword was specified for the data command. Use "help data" for help with the syntax of the data command.

'No I/O error message available'
Program: ioerrmsg.f
The file name associated with the I/O error could not be determined so that no error message is available.

'No labels available for this data set'
Program: codes.f
There is no label file attached to this data set, so that no labels are available for display.

'No NEWS information available'
Program: adaas.f
There is no news file available for the specified item.

'No recode sets specified'
Program: frtget.f
A recode variable was entered, but it was not followed by any recode sets. Use "help recode" for help with the syntax of recode statements.

'No row variable specified'
Program: twoway.f
No row variable was specified for the twoway table whose parameters were previously entered.

'No signon message available'
Program: adaas.f
The signon message displayed at the start of program execution is not available.

'No tables have been entered'
Program: twoway.f
Table entry for TWOWAY has been terminated (e.g., via the DONE keyword), but no valid tables have been specified.
'No valid <title>'

Program: listpars.f
There are no valid parameters for the indicated title. The text <title> is usually "Commands", "Modifiers", or "Keywords".

'Number of records in a case exceeds the program limit'
'Use "SET MAXRECS=N" to increase this limit to N records'

Program: datainpt.f
When CASEVAR is specified, the allowable number of records permitted in a case is given by the parameter "MAXRECS". This value can be changed with the SET command's MAXRECS option.

'Number of values specified greater than <NUMBER>'
'Error occurred at column <NUMBER>'

Program: numlist.f
The list of values being processed contains more numbers than the maximum allowed value of <NUMBER>. For example, the list "1-5,16-20" contains ten values ("1, 2, 3, 4, 5, 16, 17, 18, 19, 20").

'Number of value ranges specified greater than <NUMBER>'
'Error occurred at column <NUMBER>'

Program: numlistp.f
The list of values being processed contains more range pairs than the maximum allowed value of <NUMBER>. For example, the list "1-5,16-20" contains two range pairs ("1-5" and "16-20").

'Number of variables specified greater than <NUMBER>'
'Error occurred at column <NUMBER>'

Program: varlist.f
The list of variable numbers being processed contains more values than the maximum allowed value of <NUMBER>.

'NUMBER value out of range'

Program: vartobin.f
A variable number is less than one or greater than 32767.

Numeric conversion messages:

'Misplaced minus sign character'
'Misplaced plus sign character'
'Non-numeric character in number'
'Number greater than 2,147,483,647'
'Number is all fill characters'
'Number is only a sign character'
'Number less than -2,147,483,648'
'More than 10 significant digits in number'
'Undocumented error from DECBIN'
Program: decbinms.f
A character numeric number being converted to binary was found to contain one of the problems indicated.

'OUTFIL: Non-printing character in file name'
Program: outfil.f
The file name specified for output contains an invalid non-printing character.

'PAGE value must be "ON" or "OFF"'
Program: adaas.f
An invalid value for the PAGE keyword was entered.

'Premature End-Of-File encountered reading library'
Program: datacmd.f
An End-of-file signal was detected at an unexpected position while reading the library file. This indicates some structural problem with the library. Use the LIBRARY program’s CHECK command to help determine the nature of the problem.

'PRINTWID must be positive'
Program: adaas.f, list.f, univar.f, twoway.f
The value supplied for the PRINTWID keyword must be greater than zero.

'Print width insufficient for COLUMN mode display'
'Minimum print width = 24'
Program: list.f
The print width set by the PRINTWID option of the global SET command or the LIST command is insufficient for a COLUMN mode display. The minimum acceptable value is indicated.

'Print width insufficient for RECORD mode display'
'Minimum print width = 56 (labelling ON)'
'Minimum print width = 31 (labelling OFF)'
Program: list.f
The print width set by the PRINTWID option of the global SET command or the LIST command is insufficient for a RECORD mode display. The minimum acceptable values are indicated.

Print width too small for variable <Number>
This variable will truncated
See "ADAAS.LIST.colmode.tmp" for a full width list
Program: list.f
The value for variable <Number> has a width greater than (PRINTWID – 8) and consequently had to be truncated for the display. A full width list may be found in the file "ADAAS.LIST.colmode.tmp". The PRINTWID option may be used to assign a larger print width value so this variable value can be displayed in the output file.

Problems with Input/Output:

'Problem closing <FILENAME>'
'CLOSE error <NUMBER>,'

'Problem opening <FILENAME>'
'OPEN error <NUMBER>,'

'Problem reading <FILENAME>
'READ error <NUMBER>,'

'Problem writing <FILENAME>'
'WRITE error <NUMBER>,'

'TWREAD: Problem reading the data (from CASE)'
'UNREAD: Problem reading the data (from CASE)'

Program: ioerrmsg.f, twoway.f, univar.f
An INPUT/OUTPUT problem was encountered while processing some file. In most situations the name of the file is given by <FILENAME>; and the exact nature of the problem is indicated by the I/O error number "N". The explanation of numbers is usually displayed along with the number; but in addition the meaning of numbers less than 1000 may be found with the UNIX command "man -s2 intro". The explanation of numbers greater than 1000 may be found with the UNIX command "cat $ADAASHOME/doc/ftnioerrmsg.txt".

'Problem permitting the output data file'

Program: subset.f
The output data file is permitted "rw u" at the end of the subset operation. There was a problem performing the permit operation.

'Problem writing the full subset file header'

Program: subset.f
The full eight byte header could not be written to the output data file due to an I/O error.

'Program error: No terminating blank'

Program: frtget.f
Please notify the Data Center of this error and save all available documentation.

'RAND% value must be in the range (1-99)'

Program: list.f
An invalid value for the RAND% parameter has been entered.

'Region not in ascending order'

Program: frtget.f
Values supplied in ranges must be in ascending order.

'RECODE statement is empty'

Program: frtget.f
The "recode" keyword at the beginning of a recode statement is not followed by any variable or recode set specifications.

'Record length for file <NUMBER> does not agree with earlier files'

Program: datainpt.f
In a data set with multiple data files, the record length of the files are not all identical.

'Record length not positive'

Program: chkheadr.f
The record length given in the data file header is not a positive number.

'Record length too large for input buffer'

Program: chkheadr.f
The record length given in the data file header is too large for the internal buffer. Please notify the Data Center of this error and save all available documentation.

'Record length MAXRECLEN'

Program getlrecl.f
The length of records found in the data file is greater than the maximum permissible system record length.

'Record Number = <NUMBER>'

Program: list.f, subset.f, twoway.f, univar.f
This message indicates the number of the input record where a problem occurred. This message should be preceded by other messages that detail the nature of the problem.

'Record structure not Type <RECTYPE>'

Program: chkheadr.f
The record structure given in the data file header is not <RECTYPE>, as expected by the program.
'Replacement Filter, Recode, Dupvar, Casevar, or Title (or "C" for cancel)'

Program: frtget.f
This is a prompt for a casevar, filter, dupvar, recode, or title statement to replace the invalid one entered previously. You may enter "C" to terminate the request and return control to the ADAAS command prompt.

'SKIP value must be greater than zero'

Program: list.f
The value for the SKIP keyword must be greater than zero.

'Statement contains unpaired primes'

Program: frtget.f
A filter or recode statement using alphabetic variables contains unpaired sets of primes around the code values.

'STOP value must be non-negative'

Program: list.f
The value supplied for the STOP keyword must be non-negative.

Storage messages:

'ALFLST: Size of code value storage area exceeded'
'Can't allocate an input buffer (GETLRECL)'
'COLSUM: Table storage exceeded'
'NUMSLST: Size of code value storage area exceeded'
'STORINIT: Can't allocate storage for array <NUMBER>'
'Too many variables for storage routines'
'TWREAD: Column table storage exceeded'
'TWREAD: Row table storage exceeded'
'Storage capacity exceeded'
'At record Number = <NUMBER>'

Program: getlrecl.f, frtget.f, twoway.f, univar.f
The internal storage required by the program cannot be obtained from the UNIX OS. Please notify the Data Center of this problem and save all available documentation.

'TEST value must be "ON" or "OFF"'

Program: adaas.f
An invalid value for the TEST keyword was entered.

'There are no variables to process'

Program: datainpt.f
There are no variables specified for the analysis.
'There is no active data set'

Program: adaas.f, codes.f, filescmd.f
Commands use the currently active data set and there was none currently defined. Use the DATA command to choose the data set you are interested in working with.

'Too many keywords for internal storage'

Program: datacmd.f
This error occurs when a wildcard search of library keywords is performed and the program has located more than 1200 keywords that match the pattern. Please notify TDC of this problem.

'Unable to access library file'
'Unable to access library XRF file'

Program: datacmd.f
The library file (library.dat) and/or cross reference file (library.xrf) could not be located on the specified path.

'Unit <NAME> attached to <FILENAME>,'
'Unit DATOUT .NE. -1'
'Unit DAUNIT .NE. -1'

Program: adaas.f
Please notify the Data Center if any of these messages occur and save all available documentation.

'Unknown user: <USERNAME>,'

Program: fixtilde.f
An unknown user name was used in the construct ":<username>/filename".

'Values for alphabetic variables must be enclosed in primes'

Program: frtget.f
Code values for alphabetic variables used in FILTER or RECODE statements must be enclosed in primes. Use "help filter" or "help recode" for help with filter or recode statement syntax.

'Variable display too large for print width'

Program: list.f
The display of one of the requested variables in RECORD mode is too small for the available print width.

'***Variable <NUMBER> <NAME>','
'PRINTWID insufficient for display'

Program: univar.f
The assigned value of PRINTWID is too small to permit printing information for the designated variable. Use the PRINTWID keyword to set a larger value.

'A variable list must be specified'

Program: subset.f
No variable list was specified for the subset operation. Use the VARIABLE keyword to specify a list of variables.

'Variable <NUMBER> cannot be located in the dictionary'

Program: frtget.f
The variable number "N" that was specified in a variable list is not in the dictionary for the active data set. Use the FILES command to make sure that you are accessing the proper data set and check the variable listing for the proper variable number.

'Warning: CASEVAR was specified without a filter.'
'It has no effect in this configuration.'

Program: frtget.f
If a CASEVAR statement is entered, but no FILTER statement is entered, the CASEVAR statement has no effect, but may increase processing time needlessly.

'Warning: Problem permitting the output dictionary'

Program: subset.f
The output dictionary is permitted "u+rw" at the end of the subset operation. There was a problem performing the permit operation.

'Warning: ** The TEST option is ON **'

Program: adaas.f
When the SET command TEST option is ON, no records are read from the input data set, but input statements are parsed.

'Your replacement is too long'

Program: keyscn.f
This message may arise from a two causes: 1) your replacement string is too long to fit into the memory storage allotted for input processing (this limit is currently 999 characters) when added to the existing contents of the input; or 2) the replacement string itself is more than 79 characters in length.
Appendix D

**TWOWAY DELIMITD Output Format**

Version 04-27-1998

The TWOWAY command DELIMITD keyword causes a comma-delimited form of the bivariate frequencies generated by the command to be written to a file. The format of this file is presented in this Appendix.

There are four different types of lines in the DELIMITD output file:

1) A brief table header line (one line);
2) A line giving the column code labels or code values (one line);
3) A number of lines giving the row code label or code value, the column frequencies for that row value, and the total frequency for that row value (one line for each row variable value);
4) A line giving the column totals and table total (one line).

The format of each line type is given below. All alphanumeric quantities are enclosed in primes.

1) **Header Line**

   A simple four-element header line is printed for each table consisting of the table number, table ID tag, column variable number, and row variable number. The format is:

   'Table #N','Table_ID',Colvar#,Rowvar#

   The symbol "N" is the number assigned to the table with the TN keyword, or the default number. The Table_ID is the string assigned by the ID keyword, or blank.

2) **Column header values**

   The format for the column header line is:

   ' ','ColLab1','ColLab2', ... ,'ColLabN','Total'

   where ColLabn is the Label for column n if a label exists; or the numeric code value for this column if there is no label. "N" is the total number of columns in the table. The initial blank field is to insure that this line has the same number of entries as the following lines.

3) **Row Frequencies**

   The column frequencies for each row variable value are written in the format:

   'Rowlab',Colfreq1,Colfreq2, ... ,ColfreqN,Rowtotal

   where Rowlab is the Label for the row variable value if a label exists; or the numeric code value for this row if there is no label. "N" is the total number of columns in the table. Rowtotal is the total frequency for this value of the row variable.

4) **Column Totals**

   The final line has the format:

   'Total',Coltotal1,Coltotal2, ... ,ColtotalN,Tabletotal
where Coltotaln is the column total for column n and Tabletotal is the grand total for the table (i.e., the sum of all row or column totals.

An example of the comma-delimited output from the 1993 FARS accident file is shown below. The second line of the output is printed in two lines due to the line length limits of this manual.

```
'Table #1',' ',36,35
',',No hit and run','Hit m.v.transprt','Hit ped/non-motr',
'Hit prkd veh/obj','TOTAL'
'No controls', 26525., 176., 961., 25., 27687.
'Dev n/fncting', 48., 1., 1., 0., 50.
'Dev func impropr', 23., 1., 1., 0., 25.
'Dev func proprly', 7636., 120., 155., 6., 7917.
'Unknown', 64., 0., 4., 0., 68.
'TOTAL', 34296., 298., 1122., 31., 35747.
```