UMTRI-82-47

# ADAAS

## AUTOMATED DATA ACCESS AND ANALYSIS SYSTEM

## PROGRAM DOCUMENTATION MANUAL

John A. Green

DECEMBER 1982

UMTRI The University of Michigan Transportation Research Institute

On September 16, 1982, the Regents of The University of Michigan changed the name of the Highway Safety Research Institute to the University of Michigan Transportation Research Institute (UMTRI).

**Technical Report Documentation Page**

| 1. Report No. | 2. Government Accession No. | 3. Recipient's Catalog No. |
|---|---|---|
| | | |

**4. Title and Subtitle**

ADAAS
Automated Data Access and Analysis System
Program Documentation Manual

**5. Report Date**
December 1982

**6. Performing Organization Code**
UMTRI-82-47

**7. Author(s)**
John A. Green

**8. Performing Organization Report No.**

**9. Performing Organization Name and Address**

Transportation Research Institute
2901 Baxter Road
Ann Arbor, Michigan 48109

**10. Work Unit No. (TRAIS)**

**11. Contract or Grant No.**

**12. Sponsoring Agency Name and Address**

Motor Vehicle Manufacturers Association
320 New Center Building
Detroit, Michigan 48202

**13. Type of Report and Period Covered**

**14. Sponsoring Agency Code**

**15. Supplementary Notes**

**16. Abstract**

This manual documents operation of the Automated Data Access and Analysis System (ADAAS), a set of computer programs developed by the Transportation Research Institute. The manual is not descriptive of analysis techniques, but is intended as a detailed guide to system operation.

The two primary functions of ADAAS are: 1) to access accident data maintained by UMTRI, and 2) to perform simple analyses of the data. This manual provides an overview of the system, fully describes each valid command, and documents special features of the system such as selective filtering and recoding operations. Examples of both user input and program output appear throughout the text.

| 17. Key Words | 18. Distribution Statement |
|---|---|
| | |

| 19. Security Classif. (of this report) | 20. Security Classif. (of this page) | 21. No. of Pages | 22. Price |
|---|---|---|---|
| Unlimited | Unlimited | 115 | |

Report Number UMTRI-82-47

# A D A A S

## Automated Data Access and Analysis System

## Program Documentation Manual

by

John A. Green

December 1982

The University of Michigan
Institute of Science and Technology
Transportation Research Institute
Ann Arbor, Michigan

ACKNOWLEDGEMENTS

In the late 1960's, the Transportation Research Institute (the Highway Safety Research Institute at that time) obtained an early version of the OSIRIS system from the U-M Institute for Social Research and converted this batch-oriented system of analysis and data-manipulation programs for terminal-mode operation on the University of Michigan computer under the MTS operating system. Elements of this system formed the core for a user-oriented library management and analysis system denoted by the mnemonic ADAAS that was designed and developed by John Green. The assistance of many people over the years (e.g., David Wood, Carole Hafner, Marianne Stover, Hank Golomb, Elliot Noma, and Chris Ford) has had significant impact in producing the current version of ADAAS.

The format of this manual and much of its implementation in TEXTEDIT are due to Christopher Ford.

Primary funding for the development and maintenance of ADAAS, as well as the wealth of motor vehicle traffic accident information accessible through the system, has come from the Motor Vehicle Manufacturers Association. The continuing support of that agency is gratefully acknowledged.

TABLE OF CONTENTS

# Section 1
# INTRODUCTION AND OVERVIEW

ADAAS, the Automated Data Access and Analysis System, is a set of computer programs maintained by the University of Michigan's Transportation Research Institute (UMTRI). The programs are written in FORTRAN and 370-Assembler and are resident on the University of Michigan Computer operating under the Michigan Terminal System. ADAAS is not a statistical analysis package (like SAS or SPSS for example) although it does possess some statistical capability, but is intended as a means of accessing the large library of transportation-related data sets maintained by the Transportation Data Center and providing the means for performing preliminary analysis tasks on selected data as an integrated part of the system. Considerable effort has been expended to make the system "user-friendly" in the parlance of today's buzz words, and to provide as much information as possible when errors occur.

ADAAS performs four major functions for system users. First, and probably most important, access to the library of data sets maintained by UMTRI is provided through the entry of a eight-character mnemonic "data set descriptor." On-line techniques for finding the desired descriptor are available in a DESCRIBE function that is hierarchical in operation, directing a request through grouped descriptors to find the desired one. Data access is provided without requiring a knowledge of how to mount magnetic tapes or open disk files, thereby removing a prime source of user frustration.

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

After the analytic operations have been performed, the generated output may be printed on the user's terminal or, in some cases, on the XEROX 9700 or line printer located at the computer center.

In many cases, the problem at hand requires more statistical computing power than is available in ADAAS. In these situations, an interface with many other systems that will accept ADAAS data sets is provided. Currently, the widely distributed BMD, OSIRIS, and SPSS packages are available, as well as the powerful MIDAS system developed at the University. In addition, there are a host of other statistical programs to satisfy almost every need.

This document describes the operation of ADAAS. To obtain a copy of this manual on the X9700 printer located at the Computer Center in Ann Arbor, Michigan:

```
$SOURCE HSRI:DOCUMENT
```

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 as well as in batch, 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 MTS command:

```
$RUN HSRI:ADAAS
```

and is terminated by the ADAAS command "STOP."

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

1) Data commands to provide access to data sets in the system library or in private libraries.

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 of use to the user.

A summary of the commands available in ADAAS is given in the table on the next page. Six of the commands are "analysis" or "data manipulation" commands (ANOVA, BIVAR, LIST, SUBSET, TWOWAY, and UNIVAR). They require the input of additional information following the command statement such as a filter, recode, or title statement. A description of the information required by the analysis commands is presented in the Analysis Command Overview in Section 4.

SUMMARY OF ADAAS COMMANDS

| COMMAND | DESCRIPTION |
|---|---|
| ANOVA | Generates analysis of variance tables |
| BIVAR | Produces bivariate tables (see also TWOWAY) |
| CODES | Displays labels for a given data set |
| DATA | Provides access to system or private data sets |
| DESCRIBE | Displays on-line documentation |
| DRIVES | Displays tape-drive information |
| FILES | Displays data set file information |
| HELP | Directs user to the DESCRIBE command |
| LIBRARY | Allows for the management of a private user library of data sets |
| LIST | Produces a listing of a data set |
| MESSAGE | To send a message to the Transportation Data Center |
| MIDAS | Interfaces ADAAS with MIDAS |
| MTS | Temporarily returns ADAAS to MTS |
| NEWS | Displays information on recent system updates |
| PRINT | Copies output to terminal or line printer |
| RELEASE | Releases the current data set |
| SET | Controls various aspects of program operation |
| SPSS | Interfaces ADAAS with SPSS |
| STATUS | Provides a printout of the system status |
| STOP | Terminates program execution |
| SUBSET | Produces a subset of a data set |
| TAPE | Provides for external use of ADAAS tapes |
| TWOWAY | Produces bivariate tables |
| UNIVAR | Generates univariate frequency distributions and statistics |
| XREF | Generates a cross reference dictionary listing |

## 1.2) DATA ACCESS

A primary feature of ADAAS is the ability to access information maintained by the Transportation Data Center, an extensive collection of motor vehicle traffic accident records and transportation-related information. Over 350 separate data sets from many different sources currently comprise the data base.

Data sets are accessed through ADAAS with the DATA command. The only user-supplied information required by the system, however, is a "data set descriptor" that identifies the particular data set desired. All information needed to

mount tapes, assign and open files, etc., is internal to the system. Complete documentation of the current data set descriptors is available on-line using the DESCRIBE command.

Most of the data sets are documented by detailed codebooks. Each codebook contains, among other things, the names and characteristics of variables, the meanings of separate variable values, and frequency counts for the values. A complete listing of available data sets, as well as codebooks printed to date, is contained in the MTS file HSRI:FILES.

## 1.3) DATA STRUCTURE

In ADAAS, a "data set" is defined as a combination 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. These files can be MTS disk files or, more usually, files on magnetic tape. A data set can also include a disk label file that is used for labeling data values in output produced by many of the commands.

System data sets (i.e., data sets maintained by the Transportation Data Center) are used most often. However, there are provisions to enter user-defined data sets that have been created by ADAAS commands such as SUBSET, or by 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 in addition to plots displaying the frequency distributions in a histogram format. Tables displaying bivariate frequency distributions can be produced with the TWOWAY or BIVAR commands. The ANOVA command can be used to generate analysis of variance statistics. A formatted list of selected variables for selected cases 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 via the LIBRARY command. The LIST and SUBSET commands also have the capability of generating deterministic or pseudo-random selections of cases 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 MTS. OSIRIS, a collection of analysis and data-management programs maintained by the U-M Institute for Social Research, readily accepts ADAAS dictionary and data files as input (ADAAS dictionaries correspond to OSIRIS "type 1" dictionaries). MIDAS, a statistical program developed and maintained by the U-M Statistical Research Laboratory, is another statistical package that can use ADAAS data sets as input. The MIDAS command in ADAAS simplifies this interface. And finally, SPSS is also available, with the SPSS command in ADAAS smoothing the interface.

## 1.6) SETUP FILES

An efficient way to operate ADAAS interactively is with the use of a "setup" file. This technique involves the creation and use of a disk file selected by the user that contains input information required by the programs. Using the MTS file editor, the user inserts lines in the file that correspond to the form and order of statements as they would be entered in ADAAS. The file is then "input" into the system with the SET command. The use of setup files has these advantages:

1) The actual analysis program is not loaded into computer memory while the slow job of manually typing in information is performed. That is, the operation is more cost-effective.

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

3) If the user has entered a line in error, it can be easily corrected with the file editor and executed a second time. This capability will be appreciated by anyone who has manually typed in a complicated filter or recode statement only to type it in again after an input error occurs.

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 and Section 5.

## 1.7) ATTENTION INTERRUPT PROCESSING

Attention interrupts (i.e., Break, Attention, Attn, etc.) may be issued at any time during execution of the program to stop the action currently underway, and to return the user to a point of the program where new control input may be entered. In most situations, the program will print the message "+ATTN+" or "ATTN" on receipt of an interrupt and transfer the user to a different section of the program. In general, an attention interrupt issued during the execution of a command will abort the command -- but not always. Issuing an attention break at the command input point will return the user to the MTS command mode. The program may be restarted by issuing the MTS command "$RESTART."

If a data set is being interrogated at the time the attention is issued, the message

$XX.XX DOLLARS USED. CONTINUE? (Y/N/MTS)

will be printed. The cost figure indicated shows the total amount of money spent since the time of signon. If "Y" is entered in response to this prompt, processing will continue as if the interrupt had not been issued. If "N" is entered, the interrupt is treated as valid and processing is terminated. If "M" is entered, program operation is suspended and control is returned to MTS. If $RESTART is issued in MTS, execution of ADAAS is restarted and processing resumes at the point of the interrupt UNLESS a command that involves that loader was issued in MTS.

## 1.8) ERROR RECOVERY

In many situations, error recovery is possible while operating ADAAS interactively. A command spell-checking feature, for instance, is often able to detect misspelled commands. Keywords and modifiers entered in error cause ADAAS to seek replacements. These error replacement prompts also work when input commands are being issued from a setup file. Thus in many cases, errors that are 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 word "CANCEL" may be entered to abort the line entirely, or an empty line may be entered (a return with no other characters typed) to simply delete the offending modifier or keyword while retaining the rest of the line.

Example of ADAAS Error Recovery
(* = user supplied input)

```
  @Command
* ?DATTA K=SFQ8:TESTDATA
  @Do you mean "DATA" ?
* ?Y
  @"TESTDATA": Access is established

  @Command
* ?UNIVAR NOGOOD=BADFILE WRONG
  @"NOGOOD=BADFILE" IS INVALID
  @Enter replacement, "CANCEL", or Return to Ignore
* ?OUT=RESULTS
  @"WRONG" IS INVALID
  @Enter replacement, "CANCEL", or Return to Ignore
* ?CANCEL

  @Command
  ?
```

Certain catastrophic failures in the operation of the program result in the occurrence of a "PROGRAM INTERRUPT." These interrupts are most often caused by a failure in program logic that has never been detected. Should such an condition occur, the operation that is currently underway will be aborted and the user will be returned to the "Command" prompt. A notification of the interrupt is sent to the Transportation Data Center by means of the MTS message system. Please save all the documentation from this computer session (terminal printout and OUTPUT file contents) and mail this material to UMTRI. Cooperation in reporting these errors is essential to their ultimate elimination.

## 1.9) DESCRIBE FEATURE

By entering DESCRIBE in response to the command prompt, a considerable amount of on-line program documentation is made available to the user. The information is structured hierarchically, so that the desired information can be obtained by a number of DESCRIBE operations, with each step defined by the one before. In most cases, the desired information is supplied directly by the DESCRIBE command. In cases where the documentation is lengthy, reference is made to an appropriate file where the information is located. In these cases, the documentation can be printed at the terminal, or routed to a line printer, using the PRINT command.

The basic DESCRIBE command (with no modifiers) prints a list of the available commands and directs the user to more extensive documentation. With appropriate modifiers, the command may be used to give a brief description of all commands, to give a detailed description of any selected command, to print a description of the FILTER or RECODE statement syntax, or to obtain information on the data set descriptors that are used to access each data set. The descriptors are arranged by groups (FEDERAL, OTHER, etc.) so that the desired one can can be easily obtained by a simple top down search.

## 1.10) 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 1980 is used to determine the relationship between FIRST HARMFUL EVENT and roadway TA-1 CLASS for dry and wet weather accidents.

Example of ADAAS Operation
(* = user supplied input)

```
* #$RUN HSRI:ADAAS
  #EXECUTION BEGINS

  @   A D A A S
  @Transportation Research Institute
  @OCT  1, 1981  at  15:44:50
  @VERSION 08-02-81

  @Command
* ?DATA KEY=FARS80AC
  @"FARS80AC": A magnetic tape will be mounted next
  #*HSRI* (C2376D): Mounted on T907
  @"FARS80AC": Access is established

  @Command
* ?TWOWAY OUTPUT=SAVE.TABLE CREATE
  @Analysis for data set "FARS80AC"
  @An output file "SAVE.TABLE" has been created

  @Filter (or Recode or Title):
* ?HARMFUL EVENT BY TA-1 CLASS FOR DRY AND WET

  @Enter Parameters:
* ?CV=15 RV=17 ALL% FV1=27:1 ID=DRY
* ?SAME FV1=27:2 ID=WET DONE
  @Processing begins
  @Case Read = 5269
  @Bulkfile table and line numbers
  @    1    19
  @Processing completed

  @Command
* ?PRIN FI=SAVE.TABLE ROUTE=CNTR X9700
  #*PRINT* ASSIGNED RECEIPT NUMBER 615525
  #*PRINT* 615525 RELEASED TO CNTR, 7 PAGES
  @ROUTE=CNTR COPIES=1 MODE=C PRINTER=X9700
  @FILE=SAVE.TABLE

  @Command
* ?RELEASE
  #*HSRI*: T907 Released
  @"FARS80AC": The active data set is released

  @Command
* ?STOP
  #EXECUTION TERMINATED
```

# 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. The basic structure 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 a necessary item of background information for the successful analysis of data sets maintained by the Transportation Data Center.

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 code value descriptions to label output information generated from the data file in a user readable (i.e., English) format.

## 2.1) DICTIONARIES AND CODEBOOKS

The definition of all the information contained in the data set is contained in a CODEBOOKCARDS file. This file (rarely accessed by system users) contains a set of formatted records of different types. One record type, called a T-record (or Tcard) 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 codes 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 Tcards alone are extracted from the codebookcards file, they may be processed to form a dictionary file. It is this file that is read by the system programs to obtain the information necessary to read the data file. The major items of information contained in the Tcard for a given variable (i.e., data element) are shown below, followed by an explanation of each item.

Dictionary (Tcard) record contents

| | |
|---|---|
| Number | Variable designator (1 to 32767) |
| Name | 1 to 24 character name |
| Type | Data storage mode<br>C = Character numeric<br>A = Alphabetic |
| Location | Column location of field |
| Field Width | Number of characters in field |
| Decimal Places | Number of implied decimal places |
| Responses | Number of possible responses |
| MD Code #1 | Missing data #1 (blank=none) |
| MD Code #2 | Missing data #2 (blank=none) |

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
The name is used by the programs to document the output with an english designator.

Type
The type designator indicates whether the data is stored as a number or as an alphabetic string. Alphabetic variables may be used as filter or recode variables, but are not generally accepted in analysis 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. Useful when special programs are written to process the data file.

Field width
The number of characters in the field for this variable.

Decimal Places
The variable value is assumed to have an implicitly defined decimal place located from the right side of the data field.

Responses
The variable has a number of possible values specified by this number. Multiple responses are accumulated by the programs and treated like multiple occurrences.

MD Code #1
All data values equal to this value are treated as missing and are handled specially in certain analysis operations. No MD code is designated by an entirely blank field.

MD Code #2
All data values equal to or greater than this value are treated as missing and are handled as for MD#1. Again, no MD code is designated by an entirely blank field.

Two type of dictionaries, designated as Type 1 and Type 5, are used in ADAAS. Both are generated from Tcards. In a Type 1 dictionary, which is the default format, information is stored in both binary and character mode. In a Type 5 dictionary, records are effectively Tcards with the location information for the data set inserted. Although Type 1 is the default format, ADAAS will read either Type 1 or Type 5 dictionaries.

When a DATA command is issued to generate user access to a specific data set, a Type 5 dictionary is copied into a temporary file named "-ADASDICT" on the user's CCID for subsequent use by the system programs. This on-line dictionary may be accessed at any time for dictionary information, but must not be modified in any way.

In general, users will find that the 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 (as in the FARS 1980 data sets) referenced a "ACCIDENT RELATED FACTORS" variable as a data element having three responses and it was desirable to access the three fields as distinct variables. Contact UMTRI 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 case, 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 an IBM-compatible machine, the data characters are stored in EBCDIC.

Since accident record 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.

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 that users who feel that a different structure would be of advantage should contact UMTRI.

## 2.3) TAPE AND DISK FILES

Both DICTIONARY files and DATA files may be stored on either magnetic tapes, or on disk files. Because of the size (i.e., number of records) of most accident collections, magnetic tape is the most common storage media for these files. For convenience, dictionary files are commonly put on the same tape as the data but this is not a requirement.

If magnetic tapes are used they are 9-track, IBM standard labelled, with a recording density of either 1600 or 6250 BPI. To conserve space, the files are usually blocked with a fixed block (FB) format and a block size of approximately 28000.

If disk files are employed, the information is stored using one line per record (unblocked). To provide record length and blocking information, each disk file is preceded by a one-line HEADER record. Thus the actual dictionary (or data as the case may be), begins on the second line of the disk file.

Disk File HEADER Record Format

| Column | Item |
|---|---|
| 1 - 3 | "HDR" |
| 4 | blank |
| 5 | Blocking Format ("F") |
| 6 - 10 | Blocksize |
| 11 - 15 | Logical Record Length |
| 16 - 52 | blank |
| 53 - 56 | "HSRI" |
| 57 - 76 | Time and date of creation |
| 77 - 80 | "0000" |

## 2.4) LABEL FILES

The final (optional) component of the ADAAS set is the LABEL file containing code value labels for applicable variables. The label file is accessed by a set of system subroutines to label the output data with readable information that replaces the numerical codes stored in the data file. Label files are normally generated by UMTRI as part of the normal data set generation process, but may be easily generated by users for special data sets entered through the LIBRARY command. In addition, it may sometimes be desirable to use special code values in certain situations such as the recoding of a variable in the SUBSET command. Instructions for generating a label file are contained in the Appendix.

# 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, even though the variations on these procedures is very large. 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 terminal mode is the most common method of execution and consequently receives the major emphasis. The use of ADAAS in batch mode, however, is an important technique for reducing costs 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 the Michigan Terminal System (MTS), the operating system within which ADAAS executes. File handling, tape mounting, and cost accounting features are all handled by MTS. Since it is beyond the scope of this manual to treat MTS systems, the user is referred to the manuals that are available.[1]

## 3.1) CONTROL COMMANDS

Operation of ADAAS is controlled by entering a command statement in response to the program request. All program prompts for user-supplied input are prefixed by a question mark. If the user fails to enter information required by the command, default values are used if possible. Where this is not possible, the command is aborted.

The command statement request is:

```
@Command
?
```

The command statement itself consists of a valid ADAAS command word that must begin in the first position after the question mark, 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 separation after the command and between all keyword phrases and

---

[1] See for example, Introduction to MTS, an excellent introduction to MTS and the file editor by the Computing Center

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
```

Most commands, keywords, and modifiers have a minimum acceptable abbreviation that may be used. These minimum abbreviations are indicated in this documentation by an underlined portion of the command, keyword, or modifier. Single-line command statements may be up 511 characters in length. If the command statement is too long for the input device, the line may be terminated with a dash (i.e., "-") and continued 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). An example of a long command that uses both keywords and modifiers is shown below.

Command example
(illustrating line continuation)

```
@Command
?PRINT FILE=SAVE.BIVAR3X(1062,1095)+-PRINT(24) -
?COPIES=3 ROUTE=CNTR X9700
```

In addition to program commands, any valid MTS command may be issued to ADAAS for processing provided that it is preceded by a dollar sign ("$") in the first position of the input field. As a note of caution, any MTS command that invokes the loader (i.e., $RUN, $LOAD, $DEBUG, etc.) will result in the termination of ADAAS. This is a real time saver in system operation as will be seen in Section 3 where the command

$EDIT <filename>

may be issued in ADAAS to permit correction of errors found in a command setup file without terminating and restarting ADAAS.

## 3.2) ACCESSING DATA SETS

After execution of ADAAS has been initiated (see Section 1.1), the first major task is to select the data set desired for analysis, and to make this data set the ACTIVE

data set - that is, the one on which subsequent operations will be performed. Data sets descriptors may be located by means of the DESCRIBE command, or by using a copy of HSRI:FILES. To locate the descriptor for the 1980 NASS data sets with the DESCRIBE command, for example, one would use the commands

```
DESCRIBE
DESCRIBE KEY
DESCRIBE KEYGROUP=FEDERAL
```

where in actual operation, the program output from the first DESCRIBE command defines the second command, and the output of the second defines the third. Once the proper data set descriptor is known, access is accomplished by the command

DATA KEY=<a data set descriptor>

For the NASS 1980 accident data set for example, the command is

DATA KEY=NASS80AC

No other knowledge of the data set is necessary. If the data set dictionary and data files are both disk files, access is generated immediately and the user is notified. If either of the component files is on tape, a message that a magnetic tape must be mounted is issued. Since the tapes must be physically located in the racks at the computing center and placed on a drive, a few minutes may be required to generate access in this case. If there are no tape drives available at the time that the DATA request is issued, the task is placed in a mount queue until there is a free drive. If such queueing is not desired, an attention interrupt will abort the data command and provide a return to the command mode.

By default, ADAAS mounts all magnetic tapes with the pseudo-device name *HSRI*. If a magnetic tape to be used in ADAAS is already mounted as the result of previous tasks with a different name (e.g., *T*), or if it is desired to specify an alternate name, the PDN keyword may be appended to the DATA command. For example:

DATA KEY=NASS80AC PDN=*THISISMYTAPE*

If the PDN keyword is used, it is possible to have a number of tapes mounted simultaneously, with only one being used. Care should be taken to avoid this situation unless it is desired. The DISMOUNT modifier may be appended to the DATA command to automatically dismount unused tapes. The STATUS command may be used to determine if any magnetic tapes are mounted.

The DATA command need only be issued once for all sequential 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.

Since costs are incurred for mounting tapes and for tape drive usage, the DATA command should not be issued until setup files have been prepared and analysis operations are ready for execution. When no further usage of data sets is required, the RELEASE command should be used to dismount tapes.

## 3.3) ANALYZING DATA SETS

After access to the desired data set has been obtained, the user may analyze the data using the operations built into ADAAS, or the active data set may be used with one of the available statistical analysis packages.

A complete description of analysis command operation is given in Section 4, while the techniques for using the data in external systems is given in Section 3.5.

## 3.4) PRINTING THE ANALYSIS OUTPUT

In order that the program is not burdened with the task of displaying its results on a slow printer while it is performing the expensive task of reading the data, ADAAS writes all output results into a file. 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.

Although the output file can be displayed by means of the MTS $COPY command or the file editor PRINT command, the simplest way is to use the ADAAS PRINT command. If the default output file "-PRINT" were used for the analysis operation, then it is only necessary to issue the PRINT command. If another file were assigned by the OUTPUT keyword, then this file can be printed by using the FILE keyword. For example, if the file "SAVE.LIST" were used to store the output of a LIST command, and the LIST command indicated that the actual list started in the output file at line 19, then the command

```
PRINT FILE=SAVE.LIST(19)
```

would print the tabular results only without documentation of the input parameters found in lines one through eighteen. Normally the formatted printout of the PRINT command is

desired. At times, however, it is essential to know the line number of each line in the output file. This would be necessary, for example, if only a part of the output were to be copied for some special purpose. If the LIST modifier is appended to the PRINT command, then the line number is printed together with the entire line contents.

The PAGE modifier may be appended to the PRINT command to simulate the action of a line printer and format the program output as it would appear on a batch printer. If the PAGE modifier is used, a prompt is made to position the terminal at the top of the next physical top-of-form (i.e., the top of the next sheet) and hit RETURN. From this point on, the program will keep track of the number of lines on each sheet, and start new tables on a new page.

Users who reside in the vicinity of a MTS batch output station (not necessarily the Ann Arbor, MI area) may use the PRINT command to route output to the output station with the ROUTE keyword. A list of the important route code designators in given in the PRINT command documentation in Section 5. For the Computer Center in Ann Arbor, the X9700 modifier may be specified in addition to route print jobs to the XEROX 9700 page printer.

## 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. In the case of 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. Since the ADAAS files are OSIRIS based as explained earlier, ADAAS data sets can be also be accessed in OSIRIS in this way.

Since both MIDAS and SPSS can access OSIRIS data sets, the interface with these systems is also direct. For these systems, a MIDAS and SPSS command is available to provide what interfacing is necessary and to provide the user with information on how to access the data in the selected system. Use of the MIDAS command gives the user the form of the MIDAS FILE keyword to access the data set. Use of the SPSS command provides the user with the proper MTS $RUN command to execute the SPSS program and to access the selected data set.

## 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 the input necessary to accomplish the desired objective in an MTS file and then later to direct ADAAS to take its input commands from this file. There are several reasons why this procedure is cost and time effective.

1) The setup file can be proofread for errors before it is used. Many potential errors can be easily located and fixed before they are issued to ADAAS.

2) The setup file can be prepared without having data access. Thus there is no need to have tapes mounted while the clerical task of typing the input commands is completed.

3) If a mistake is found while executing the setup file, it is usually a simple job to fix the error with the MTS line file editor $EDIT, and to execute the task a second time. This technique is of special importance when the input commands are long and complicated and their input is subject to typing errors.

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

Setup files are created by means of the MTS $CREATE command, and prepared using the MTS line file editor $EDIT. For information on these systems, see the MTS Manual "INTRODUCTION TO MTS." The example below shows the creation of a setup file called "NASSLIST" and the entry of four input commands to perform a data set list on the NASS 1980 non-motorist data set.

Three errors have been purposely entered into this setup file to illustrate the available methods of error correction. One is fixed by proofreading the file. In the example above, not that the INCLUDE statement on line 2 of the setup file is incorrectly spelled, The editor ALTER command is used to correct the misspelling and the file is listed a second time for final proofreading. The remaining errors are not as obvious and are not discovered by this visual check, but will be detected by the LIST program when the setup file is executed.

Preparing a Setup File (with errors)
(* = user supplied input)

```
* #$CREATE NASSLIST
  #File "NASSLIST" has been created.
* #$EDIT NASSLIST
* :INSERT
* ?LIST
* ?INCLIDE V413=5
* ?VAR=1,2,6 COLLUMN NARROW
* ?SET INPUT=ME
* ?
* :PRINT/FILE
  :      1      LIST
  :      2      INCLIDE V413=5
  :      3      VAR=1,2,6 COLLUMN NARROW
  :      4      SET INPUT=ME
* :ALTER 2 ;LID;LUD;
  :      2      INCLUDE V413=5
* :PRINT/FILE
  :      1      LIST
  :      2      INCLUDE V413=5
  :      3      VAR=1,2,6 COLLUMN NARROW
  :      4      SET INPUT=ME
* :STOP
  #
```

As a practical matter, it usually wise not to include DATA commands in the setup file unless there is a number of different data set accesses required.

To use the setup file in ADAAS it is only necessary to issue the ADAAS command

SET INPUT=<the setup file name>

This directs ADAAS to read all subsequent input from the file specified until another SET INPUT command is encountered, or until an end of file is encountered in the setup (i.e., no more input lines). In the example, the setup file ends with the command "SET INPUT=ME" directing ADAAS to take further commands from the terminal. The execution of the setup file created in the earlier example is shown below. In this example the TEST option is also turned ON. In TEST mode ADAAS will perform all operations specified with the exception of actually reading the data file. Thus errors in the setup can be checked and no data read costs are incurred.

Executing a Setup File
(* = user supplied input)

```
  @Command
* ?SET INPUT=NASSLIST TEST=ON

  @Command
  @>>>>LIST
  @Analysis for data set "NASS80NM"
  @Warning:  ** the TEST option is ON  **

  @Filter (or Recode or Title)
  @>>>>INCLUDE V413=5

  @Recode (or Title)
  @>>>>VAR=1,2,6 COLLUMN NARROW

  @Enter Parameters:
  @>>>>SET INPUT=ME
  @"SET" IS INVALID
  @Enter Replacement, "CANCEL", or Return to Ignore
  ?CANCEL
  @Input Returned to Terminal

  @Command
* ?$EDIT NASSLIST
  #$EDIT NASSLIST
* :PRINT 2
  :     2     INCLUDE V413=5
* :INSERT 2 ;LIST OF INJURED - SEVERITY UNKNOWN;
  :     2.5   LIST OF INJURED - SEVERITY UNKNOWN
* :STOP

  @Command
  ?
```

Note that when ADAAS is reading commands from a source other than the terminal, the commands that it reads are echoed back for notification with the prefix ">>>>." If desired this echo-back may be disabled by the command "SET TERSE=ON." As the LIST command executes, a warning message that the system is operating in TEST mode is printed, the filter is read and then the trouble begins when the parameter statement is read in place of the TITLE because no TITLE statement was included in the setup file. As a result of this omission, the "SET INPUT=ME" command is read as s parameter statement and the modifier SET is found to be invalid for the LIST command. At this point CANCEL is entered to abort the input line thereby terminating the SET INPUT sequence and returning command input to the terminal.

In the remainder of the example the editor is used to insert a title after the filter statement.

Confident that all the errors have been detected (even though the parameter line has not been checked), the job can be re-executed as shown in the next example with TEST=OFF.

Executing the Corrected Setup File
(* = user supplied input)

```
  @Command
* ?SET INPUT=NASSLIST TEST=OFF

  @Command
  @>>>>LIST
  @Analysis for data set "NASS80NM"

  @Filter (or Recode or Title)
  @>>>>INCLUDE V413=5

  @Recode (or Title)
  @>>>>LIST OF INJURED - SEVERITY UNKNOWN

  @Enter Parameters:
  @>>>>VAR=1,2,6 COLLUMN NARROW
  @"COLLUMN" IS INVALID
  @Enter Replacement, "CANCEL", or Return to Ignore
* ?COLUMN
  @Processing Begins
  @The list begins at output file line number 20
  @Cases Read  =       3
  @Cases Listed=       3
  @Processing Completed

  @Command
  @>>>>SET INPUT=ME

  @Command
  ?
```

In processing the parameter statement, the COLUMN modifier was incorrectly spelled and could not be recognized by the program. The replacement was entered directly from the terminal (even though commands are being read from NASSLIST) and processing proceeds.

The use of setup files is encouraged for all but the simplest jobs because of the time savings that can accrue due to error correction.

## 3.7) BATCH MODE OPERATION

Terminal mode operation in MTS provides the generally most productive method since the results of previous analysis can be used to direct the course of the current task, and any errors that occur can be corrected immediately. However, batch mode provides the capability of using the much lower rates that occur at night when the computer is less fully utilized. By using terminal mode to create setup files and check them with TEST=ON, and batch mode to perform the actual execution, the best of both worlds can be obtained. The most serious drawback to batch operation is, of course, the turnaround time: a day instead of a few minutes.

Batch mode operation can be accomplished through an extension of the setup file technique. It is only necessary to add signon parameters and (in the case of remote users) to arrange for storing the results of the run. One important point to note, however, is that ADAAS will accept lines that are up to 511 characters long, but the MTS batch processor has a 255 character limit. The MTS $FILESTATUS command may be used to check the maximum line length of the setup file. The format of a prototype batch setup file is shown in the following example.

Batch Mode Setup File Format
(* = For remote users only)

```
  $SIGNON CCID T=<time> PRIO=<level> ROUTE=<code>
  $CONTINUE WITH *MSOURCE* RETURN
* $CREATE <filename>
* $SINK <filename>
  $RUN HSRI:ADAAS
  .
  .   Any valid commands as in a setup file
  .
```

The parameters used in this format have the following meaning.

T=<time>
In batch mode, central processor unit (CPU) time that can be expended by the task is limited to the specified value. The time can be a a simple number (e.g., 60) in which case it is the CPU time in seconds, or an "M" can be appended (e.g., 5M) in which case it represents the CPU time in minutes. For most cases five minutes (i.e., T=5M) is sufficient for tasks which access data with case counts on the 30,000. Ten minutes or more might be required for larger data sets. It is best to err on the high side since the penalty for this

choice is the loss of the total amount of money specified in the remote case of a serious program error. On the other hand, if the amount of time specified is too small, the run may not be completed and the total cost is lost. Since the costs for a given amount of CPU time depend upon the user class and the signon priority, no exact figures can be given here. A current copy of the MTS rate structure can be obtained by the command

$COPY *RATES

PRIO=<level>
The University computer operates under different rate schedules at different times of the day during designated priority periods. Rates during the day in NORMAL priority are highest. LOW, DEFERRED, and MINIMUM are increasingly lower priority designations with correspondingly lower operating costs. Specifying PRIO=MINIMUM causes the job to be held in an execution queue until minimum priority is in effect. At the time of publication, this is from 4:00 a.m. to 7:00 a.m. on weekdays, and from 6:00 p.m. to 7:00 a.m. on Saturdays and Sundays. In minimum priority, the CPU rate is only one-fifth of the daytime rate. See the file *RATES discussed above for the latest definition of the exact priority periods and corresponding rates.

ROUTE=<code>
Specifies the batch station where the output from the batch job will be printed. The applicable route codes can be found in the PRINT command description in Section 5. If ROUTE is not specified the default is CNTR - the computing center in Ann Arbor. Remote users have no need of the printout and should not specify ROUTE.

$CREATE <filename>
$SINK <filename>
If the batch job is entered from a remote site so that the printout normally routed to the center is not available, these commands may be employed to direct the output from the run into a file specified here by <filename>. The $SINK command directs MTS to place all the printed output into the designated file. If the run executes without error, <filename> may be printed the next day.

An alternative approach is to save the program output in a file with the OUTPUT keyword, and to forget about the rest of the program run documentation. As long as no errors occur, this is a perfectly acceptable method. It should be remembered that the analysis programs do not route the output results to a file in batch if the OUTPUT keyword is not specified. OUTPUT should consequently be left off in batch unless the output is to be saved in a file.

Once the batch command file has been prepared, it may be submitted for later execution as shown in the following example.

Submitting a Batch job
(* = User supplied input)

```
* #$COPY BATCH.SET *BATCH*
  >*BATCH* assigned receipt number 663832
* >PASSWORD
* >$ENDFILE
  #*BATCH* 663832 released.
* #$SYSTEMSTATUS QUEUE 663832
  -Job 663832 is awaiting exec, route=CNTR, prio=6D
  -waiting for jobs(prio):  4(8L) 7(6D) = 11(total)
  #
```

On most terminals a CONTROL-C may be used in place of $ENDFILE. The $SYSTEMSTATUS (or $SYS) command is not a necessary part of this procedure, but is included as a way of finding out the status of a batch job whose job number is known.

# Section 4
# ANALYSIS COMMAND OVERVIEW

The amount of control information required by the data manipulation or analysis commands (i.e., ANOVA, BIVAR, LIST, SUBSET, TWOWAY, and UNIVAR) is too extensive to be included as part of the command statement. Consequently, use of one of these commands requires that it be followed by other control lines. Each of the analysis commands accepts three required and two optional statements:

| | |
|---|---|
| 1) A COMMAND Statement | (Required) |
| 2) A FILTER Statement | (Optional) |
| 3) A RECODE Statement | (Optional) |
| 4) A TITLE Statement | (Required) |
| 5) A PARAMETER Statement | (Required) |

After a legal analysis command statement has been entered at a command prompt, a prompt sequence begins with the request:

```
@Filter (or Recode or Title)
?
```

At this point, either a filter, recode, or title statement may be entered. If a filter is entered, the next request is:

```
@Recode (or Title)
?
```

Up to ten recodes may be entered, after which the next request would be:

```
@Title
?
```

A title describing the run would then be entered. At some point in the filter-recode-title prompt sequence, a title statement is required, but whether it is at the initial "@Filter (or Recode or Title)" prompt, or after one filter and ten recodes have been entered, is entirely up to the user. After the title has been entered, a prompt is finally made for the parameter statement unique to each command. It has the form:

```
@Enter Parameters:
?
```

A sample run using an analysis command appears on the next page. Following this, the five statements are each described in detail.

Example of Analysis Command Operation
(* = user supplied input)

```
  @Command
* ?UNIV OUT=-RESULTS
  @Analysis for data set "BIKEACC "

  @Filter (or Recode or Title)
* ?INCLUDE V19=1-15

  @Recode (or Title)
* ?RECODE V24 (2,4-7)=1,(ELSE)=2

  @Recode (or Title)
* ?FREQUENCY DISTRIBUTION / YOUNG BIKE RIDERS

  @Enter Parameters:
* ?VAR=5,8-9 FREQ% WV=24
  @Processing Begins
  @Cases Read=    627
  @Bulkfile Variable & Line Numbers for Frequencies:
  @    5   45
  @    8   51
  @    9  102
  @Processing Completed

  @Command
  ?
```

4.1) COMMAND Statement

The analysis COMMAND statement, in addition to specifying the analysis command to be utilized (e.g., LIST or ANOVA), 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.

CREATE If the file specified by the OUTPUT keyword does not exist, the file is created with a size of 10 pages. Notification is given if the creation takes place. The modifier is ignored if the file already exists. The CREATE modifier cannot be used with device names or with files when a line number range has been appended to the file name. DEFAULT: No attempt is made to create the specified file.

EMPTY Causes the file specified by the OUTPUT keyword (or the file "-PRINT" otherwise) to be emptied.

The EMPTY modifier cannot be used with device names or with files when a line number range has been appended to the file name.
DEFAULT: The output file is not emptied.

OUTPUT=<file or device name> EXAMPLE: OUT=-TEMP
Output produced by the analysis command (e.g., tables, graphs, listings, etc.) are written into the file or device specified. A line number range may accompany the file name. For example, specifying "OUT=-MYFILE(*L+1)" would write the output at the end of the file MYFILE, thus preserving any information already written into it. The CREATE and EMPTY modifiers, however, may not be used when a line number range accompanies the file name or with devices (e.g., *PRINT*).
DEFAULT: The file "-PRINT" is used in terminal mode. In batch, output is written on *SINK*.

## 4.2) FILTER Statement

The FILTER statement selects those records from the input data set that are to be included in, or excluded from, the analysis operation. This statement instructs ADAAS to perform the designated operations on only certain specified cases from the input data set. These cases are selected by a logically defined combination of variable code values. In English, a typical filter specification might be: "Include only those cases involving a passenger car where the driver was over 45 years old and 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 so is "global" in nature.

A FILTER statement begins with the word "INCLUDE" or "EXCLUDE" and is followed by a number of variable code value specifications that are combined logically by "AND" and/or "OR" operators. Care should be taken to spell include or exclude correctly for if they are misspelled the desired filter statement will be read as a title statement.

In the 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). Variable code values 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). Note that code values for alphabetic variables must be enclosed in primes. 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 '.

Each individual variable code value specification supplies the requirement that only those cases 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 "AND" and/or "OR" to provide the desired filter action. Use of "AND" implies that a case will pass the filter only if both adjacent variable code value specifications are satisfied. Use of "OR" implies that a case 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 AND B AND C OR D

means

(A AND B AND C) OR D

If

(A AND B) AND (C OR D)

had been intended, the statement would be

A AND B AND C OR A AND B AND D

Filter statements may be of effectively unlimited length. For data entry purposes, a FILTER statement may be entered as several physical input lines by terminating each input line (except the last) with a dash. The length of each input line, however, must be 511 characters or less.

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

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

This includes those cases 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 cases 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 cases 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 AN-
D V14=9 AND V52=4

If the user desires those cases 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 cases desired, e.g., all cases where V52=4 would be included. Note also that the statement is continued to a second line by a terminating dash.

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

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

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

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

Most of the variables that make up a single case are "single-response" variables (i.e., each case has only one associated value for each variable). Some variables, however, are "multiple-response" variables where each case has more than one value for a variable (a variable OBJECT CONTACTED, for example, could allow up to five contacted objects to be coded for a single case). When using a multiple-response variable in a global filter, a case will pass the filter (that is, be included or excluded) if any response satisfies the filter condition. This means that some uses of filter statements for single-response variables are inappropriate for multiple-response variables. Hence careful consideration must be given to the exact filter action. Some examples of filter action are shown below for a two-response variable (denoted by variable number VN).

Filtering Action
On a Hypothetical Multiple-Response Variable

| Filter Statement | Case Values for VN | | Action |
|---|---|---|---|
| | 1st Resp | 2nd Resp | |
| INCLUDE VN=5,7 | 1 | 8 | Excluded |
| INCLUDE VN=5,7 | 1 | 5 | Included |
| INCLUDE VN=5,7 | 7 | 3 | Included |
| EXCLUDE VN=1-3,5-9 | 4 | 2 | Excluded |
| EXCLUDE VN=1-3,5-9 | 8 | 4 | Excluded |
| EXCLUDE VN=1-3,5-9 | 4 | 4 | Included |

## 4.3) RECODE Statement

The RECODE statement permits code values for any variable to be temporarily modified for the duration of the command. This operation 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 correspondence

```
 0- 9 --> 1,
10-19 --> 2,
   etc.
```

to provide bracketed age groups. In other situations, the RECODE capability provides a means of circumventing certain program restrictions. For example, the BIVAR command prints up to 10 consecutive code values for the column variable. Non-consecutive code values spanning a range greater than 10 can be used in BIVAR by recoding the desired values into the range 0-9.

Another important use of the RECODE option is the conversion of alphabetic code values to numeric values, thus permitting the use of analysis commands on otherwise unusable variables.

Note that the RECODE operation is temporary and affects the output of the analysis command 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).

A RECODE statement begins with the word "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. Care should be taken to spell recode correctly for a misspelling will result in the statement being interpreted as a title.

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 that each element of LIST is translated to. 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, should appear as the last translate specification. Note that code values for alphabetic variables must be enclosed in primes. 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 '. Any number of translate specifications may be employed in a RECODE statement by using a comma to delimit individual terms.

Up to ten variables may be recoded with a single analysis command and each RECODE statement may be of effectively unlimited length. For data entry purposes, a RECODE statement may be entered as several physical input lines by terminating each input line (except the last) with a dash. The length of each input line, however, must be 511 characters or less.

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 bivariate table.

A multiple-response variable must not be used in a RECODE statement. If RECODE is used for a multiple-response variable, only the first response is recoded: the remaining responses are unaltered by the operation. Accidental use of RECODE can consequently produce incorrect or misleading results.

## 4.4) TITLE Statement

The TITLE statement is an 132-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 "INCLUDE", "EXCLUDE", or "RECODE." If one of these three words is used at the beginning, the title will be taken as a 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

## 4.5) PARAMETER Statement

The PARAMETER 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. Single-line parameter statements may be up 511 characters in length. If the parameter statement is too long for the input device, the line may be terminated with a dash (i.e., "-") and continued 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).

Some commands (LIST, SUBSET, and UNIVAR) require only a single parameter line, while other commands (ANOVA, BIVAR, and TWOWAY) require a multiple lines. When multiple parameter lines are required, the DONE modifier terminates parameter input unless input is automatically terminated by entering the maximum number of tables. The entry of multiple tables is greatly simplified in many cases 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 line. 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), each of the commands that process multiple tables (ANOVA, BIVAR, TWOWAY) permit the entry of "local" filters with each table parameter statement. These filter specifications allow the selection of cases 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 cases for which variable 12 has the value 3 are to be included in the table. Correspondingly, the second form specifies the inclusion of cases 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 parameter statement

```
CV=5 RV=8 FV1=12:1-3 FV2=37:4
SAME FV2=NONE
```

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=1,3-5", for example, would be illegal. Note also that, if the filter variable happens to be a multiple-response variable, all responses are checked.

# 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.

The input prompting sequence for the analysis commands that follows that command line and output file specification is summarized in the table below.

Analysis Command Input

| Program Request | User Response |
|---|---|
| @Filter (or Recode or Title)<br>?<br>@Recode (or Title)<br>?<br>@Title<br>? | Enter a filter, recode, or title at each point as requested. The filter or recode may be omitted as long as the order is preserved. |
| @Enter Parameters:<br>? | Enter a set of program keywords and modifiers. |

## ** ANOVA Command **

Command Form: ANOVA

Keyword: OUTPUT
Modifiers: CREATE, EMPTY

The ANOVA command calculates up to 99 one-way analysis of variance tables that statistically describe the dependence of a DEPENDENT variable upon a CONTROL variable. CONTROL variables with a field width less than or equal to four may be employed in the program. Tables of quantities ancillary to the analysis are produced as part of the output together with the F-Test value and corresponding significance level.

### ** ANOVA Parameter Keywords and Modifiers **

Certain keywords and modifiers are global in nature and apply to the entire run rather than to a single table. Identified by "**GLOBAL**" below, these parameters need be entered only once for a set of tables.

Required parameters:
- CV - Control variable
- DV - Dependent variable
- DONE - Terminates table entry

CV=<variable list> EXAMPLE: CV=12-15,56
Control variable number(s). For each control variable listed, a separate table is generated.
DEFAULT: None.

DESCRIBE
A list of available keywords and modifiers is printed at the terminal. If present, all other parameters entered on the same line are ignored.
DEFAULT: No description is printed.

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.

DV=<variable list> EXAMPLE: DV=25
Dependent variable number(s). For each dependent variable listed, a separate table is generated.
DEFAULT: None.

FV1=<variable:values> EXAMPLE: FV1=35:1-24
First local filter and range. Only those cases where the specified variable has a value in the indicated range are included in the table. The Analysis Command Overview's description of the parameter statement provides full information on the use of local filters.
DEFAULT: All cases are used.

FV2=<variable:values> EXAMPLE: FV2=56:4
Second local filter and range (see FV1).

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

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

LABELS=<label file> EXAMPLE: LAB=SP65:NCSS2LAB
Code value labels for the data set being analyzed are found in the file specified. See the Appendix. for label file construction procedures.
**GLOBAL**
DEFAULT: The label file name is supplied by the system if labels are implemented for the active data set.

MD Cases for which the dependent variable has a missing data value are included in the table.
DEFAULT: The default option is NOMD.

NODICT A dictionary listing of the variables accessed is not printed in the output file.
**GLOBAL**
DEFAULT: A dictionary listing is printed.

NOLABEL No code value labels are printed.
**GLOBAL**
DEFAULT: Code value labels are printed.

NOMD Cases for which the dependent variable has a missing data value are not included in the table.
DEFAULT: This is the default option.

NOTERM For conversational mode only. No table numbers or beginning output file line numbers are printed at the terminal.
**GLOBAL**

DEFAULT: The table number(s) and output file line number(s) are printed on-line.

SAME All parameters not explicitly specified on the current parameter input line are set equal to those used in the preceding table.
DEFAULT: All parameters desired must be specified.

TERM For conversational mode only. The ETA statistic, F-ratio, and its degree of freedom and significance level are printed at the terminal for each table along with the table's number and beginning output file line number.
**GLOBAL**
DEFAULT: Only the table number(s) and output file line number(s) are printed on-line.

TN=<table number list> EXAMPLE: T=100
Optional table number. A list of numbers may be assigned for a sequential set of tables.
DEFAULT: Tables are numbered sequentially 1-99.

WV=<variable number> EXAMPLE: WV=56
Weight variable number. Due to interpretation problems, no significance computations are performed when a weight variable is utilized.
DEFAULT: No weighting is performed.

** ANOVA Input and Output Examples **

Example of the ANOVA Parameter Statement
(* = user supplied input)

```
  @Enter Parameters:
* ?CV=10 DV=5 T=100 FV1=34:3 ID=PROPERTY.DAMAGE
* ?SAME FV1=34:4-6 ID=INJURY/FATAL
* ?DONE
```

Two ANOVA tables are produced with the example above. Both use variable 10 as the control variable and variable 5 as the dependent variable. The first table, numbered 100 and tagged "PROPERTY.DAMAGE", selects only those cases for which variable 34 has a value of 3. The second table, numbered 2 and tagged "INJURY/FATAL", selects only those cases where variable 34 has a value of 4 through 6.

Example of an ANOVA Setup File

```
DATA K=NCSS1COC
ANOVA OUTPUT=-OUT E
VEHICLE WEIGHT AS A FACTOR IN INJURY SEVERITY
CV=1014,1032 DV=110 DONE
```

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 tables such as a dictionary listing). The categories for the two output tables are defined as follows:

| | |
|---|---|
| LABEL | The control variable code value labels |
| CODES | The control variable code values |
| N | The number of cases for each control value |
| WEIGHT-SUM | The weighted number of cases for each value |
| % | The percentage that 'N' is of all cases |
| MEAN | The DEPENDENT variable's mean values |

Although not shown in the output example, the tables also have three other categories of statistics that describe the dependent variable values. They are:

| | |
|---|---|
| S.D.(ESTIM.) | The estimated standard deviation |
| SUM OF X | The sum of the code values |
| SUM OF X-SQUARE | The sum of the squared code values |

Example of ANOVA Output
(produced from setup file on preceding page)

Table No. 1
Date: SEP 23, 1982 at 09:35:17
VEHICLE WEIGHT AS A FACTOR IN INJURY SEVERITY
Control Variable = Var #1014 : INJURY SEVERITY -POLICE
Depend. Variable = Var # 110 : VEHICLE WEIGHT

| Label | Codes | N | Weight-Sum | % | Mean |
|---|---|---|---|---|---|
| Killed | 1 | 441 | 441 | 3.2 | 33.440 |
| Incapacitated | 2 | 2644 | 2644 | 19.3 | 33.307 |
| Not incapacitatd | 3 | 3480 | 3480 | 25.4 | 33.450 |
| Possible injury | 4 | 2332 | 2332 | 17.0 | 34.554 |
| Not injured | 5 | 4825 | 4825 | 35.2 | 34.568 |
| TOTAL | | 13722 | 13722 | 100.0 | 34.003 |

TOTAL SUM OF SQUARES = 0.8908899E+06
FOR 5 GROUPS , ETA = 0.7287413E-01
BETWEEN MEANS SUM OF SQUARES = 0.4731196E+04
WITHIN GROUPS SUM OF SQUARES = 0.8861587E+06
F( 4,13717) = 18.309
SIGNIFICANCE LEVEL = 0.00000

Table No. 2
Date: SEP 23, 1982 at 09:35:26
VEHICLE WEIGHT AS A FACTOR IN INJURY SEVERITY
Control Variable = Var #1032 : OIC1-AIS SEVERITY
Depend. Variable = Var # 110 : VEHICLE WEIGHT

| Label | Codes | N | Weight-Sum | % | Mean |
|---|---|---|---|---|---|
| Minor | 1 | 3135 | 3135 | 52.2 | 33.981 |
| Moderate | 2 | 1142 | 1142 | 19.0 | 33.514 |
| Severe | 3 | 769 | 769 | 12.8 | 33.293 |
| Serious | 4 | 227 | 227 | 3.8 | 33.176 |
| Critical | 5 | 197 | 197 | 3.3 | 32.538 |
| Maximum | 6 | 106 | 106 | 1.8 | 32.547 |
| Inj, unk sev | 8 | 428 | 428 | 7.1 | 33.276 |
| TOTAL | | 6004 | 6004 | 100.0 | 33.651 |

TOTAL SUM OF SQUARES = 0.3834786E+06
FOR 7 GROUPS , ETA = 0.4963069E-01
BETWEEN MEANS SUM OF SQUARES = 0.9445868E+03
WITHIN GROUPS SUM OF SQUARES = 0.3825340E+06
F( 6, 5997) = 2.468
SIGNIFICANCE LEVEL = 0.02183

## ** BIVAR Command **

Command Form: BIVAR

Keyword: OUTPUT
Modifiers: CREATE, EMPTY

The BIVAR command is a restricted version of the TWOWAY command that produces tables displaying the bivariate frequency distribution for up to 50 pairs of variables.

BIVAR will generate tables that have a restricted maximum dimension of 10 for the column variable and 100 for the row variable (that is, 10 code values are tabulated across the top of the table, and 100 code values down the side of the table). The ranges normally default to 0 through 9, and 0 through 99, but may be reset to any value by the CVR and RVR keywords. Values that lie outside the table dimensions are accumulated in a "WILD" code category.

In addition, BIVAR will disregard the implied decimal place specification for all variables. Row, column, and weight variables will be treated as if there were no implied decimal place. Thus, row and column variable values will print in the tables with no decimal place even though the correct digits are displayed. Similarly, weighted cell frequencies are printed without the implied decimal places corresponding to the weight variable. Please note that the values shown are correct when the appropriate decimal place is supplied.

BIVAR may be used in place of TWOWAY in applications where processing costs are of importance (i.e., in analyzing very large data sets) and the variable range restrictions are not a problem.

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.

### ** BIVAR Parameter Keywords and Modifiers **

Certain keywords and modifiers are global in nature and apply to the entire run rather than to a single table. Identified by "**GLOBAL**" below, these parameters need be entered only once for a set of tables.

Required parameters:
- CV or CVR - Column variable
- RV or RVR - 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.

CELLED The table cells are printed in a bordered (celled) format.
**GLOBAL**
DEFAULT: Uncelled.

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. Only variable values within the range 0 to 9 are displayed in the table. All other values are placed in a "WILD" category.
DEFAULT: No default. Either the keyword CV or CVR must be used.

CVR=<variable:valuerange> EXAMPLE: CVR=25:10-15
Column variable with a range. Only those cases where the variable has a value in the indicated range are included in the table. The range itself must consist of a single value, or two values separated by a hyphen, the second greater than the first. If the range is more than 10, only the 10 lowest values are entered in the table. If no range is specified, no selection is performed, and the column values begin with 0. If the CV and CVR keywords are entered on the same parameter input line, the CV keyword is ignored.
DEFAULT: No default. Either the keyword CV or CVR must be used.

DESCRIBE
A list of available keywords and modifiers is printed at the terminal. If present, all other parameters entered on the same line are ignored.
DEFAULT: No description is printed.

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.

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 cases where the specified variable has a value in the indicated range are included in the table. The Analysis Command Overview's description of the parameter statement provides full information on the use of local filters.
DEFAULT: All cases 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).

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.
DEFAULT: No ID is used.

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

LABELS=<label file> EXAMPLE: LAB=SP65:NCSS2LAB
Code value labels for the data set being analyzed are found in the file specified. See the Appendix. for label file construction procedures.
**GLOBAL**
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.

NODICT A dictionary listing of the variables accessed is not printed in the output file.
**GLOBAL**

DEFAULT: A dictionary listing is printed.

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

NOLABEL No code value labels are printed.
**GLOBAL**
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.

NOTERM For conversational mode only. No table numbers or beginning output file line numbers are printed at the terminal.
**GLOBAL**
DEFAULT: The table number(s) and output file line number(s) are printed on-line.

NOZEROS Only rows and columns with at least one non-zero element are printed in the table.
**GLOBAL**
DEFAULT: This is the default.

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. Only variable values within the range 0 to 99 are displayed in the table. All other values are placed in a "WILD" category.
DEFAULT: No default. Either the keyword RV or RVR must be used.

RVR=<variable:valuerange> EXAMPLE: RVR=10:100-199
Row variable with a range. Only those cases where the variable has a value in the indicated range are included in the table. The range itself must consist of a single value, or two values separated by a hyphen, the second greater than the first. If the range is more than 100, only the 100 lowest values are entered in the table. If no range is specified, no selection is performed, and the row values begin with 0. If the RV and RVR keywords

are entered on the same parameter input line, the RV keyword is ignored.
DEFAULT: No default. Either the keyword RV or RVR must be used.

SAME All parameters not explicitly specified on the current parameter input line are set equal to their value defined on the preceding table. If CV and/or RV lists were specified on the preceding line, then the entire set of tables are respecified.
DEFAULT: All parameters desired must be specified.

TN=<table number list> EXAMPLE: T=100
Optional table number. A list of numbers may be assigned for a sequential set 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.

UNCELLED
The table cells are printed in an unbordered (uncelled) format.
**GLOBAL**
DEFAULT: This is the default option.

WRITE=<file name> EXAMPLE: WRITE=BISAVE
The tables are saved in binary in the specified file for use as input to special purpose programs.
**GLOBAL**
DEFAULT: Tables are not saved in binary.

WV=<variable number> EXAMPLE: WV=102
WV=<variable:valuerange> EXAMPLE: WV=102:1-3
Weight variable number. A variable with a value range may optionally be specified, causing only those cases that have a value within the specified range to be included. If no range is specified, only cases for which the weight variable has a non-missing data value are included in the table. To be meaningful, this variable should contain interval data (i.e., values that have a direct correspondence to the information they represent).
DEFAULT: No weighting is performed.

ZEROS All rows and columns (including those with all zero elements) are printed in the table.
**GLOBAL**
DEFAULT: Only rows and columns with at least one non-zero element are printed in the table.

## ** BIVAR Input and Output Examples **

Example of the BIVAR Parameter Statement
(* = user supplied input)

```
  @Enter Parameters:
* ?CVR=45:10-19 RV=102,105 COL%
* ?DONE
```

Two BIVAR tables are produced with the example above. The first uses the values 10 through 19 of variable 45 for the column variable values, and the values 0 through 99 of variable 102 for the row variable values. Printed in each table cell is the frequency of the intersection of each pair of values, along with the percentage this count represents of the column total. The second table is the same as the first except that values 0 through 99 of variable 105 are used for the row variable values.

Example of a BIVAR Setup File

```
DATA K=FARS78PR
BIVAR OUT=TABLES EM
EXCLUDE V201=00
INJURY SEVERITY VS. ACTIVE RESTRAINT USE
RV=215 CVR=206:0-3 COL% NOMD
DONE
```

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 tables such as a dictionary listing).

Example of BIVAR Output
(produced from setup file on preceding page)

Table No. 1
Date: SEP 23, 1982 at 09:35:17

INJURY SEVERITY VS. ACTIVE RESTRAINT USE
BIVARIATE FREQUENCIES AND PERCENTAGES
COLUMN VAR. NO. 206 : ACTIVE RESTRAINT ;RANGE 0 TO 3
ROW VAR. NO. 215 : INJURY SEVERITY
OPTIONS: FREQ COL% MD% NOMD

| | * None us*ed or NA* (0)* | Shoul*der belt* (1)* | *Lap belt*Lap belt* (2)* | *Lap & sh*ldr belt* (3)* | * (TOTAL)* |
|---|---|---|---|---|---|
| No injury (0) | 13989 | 30 | 577 | 266 | 14862 |
| (COL %) | 16.1 | 25.9 | 33.0 | 29.4 | 16.6 |
| Possible inj (1) | 5284 | 8 | 206 | 99 | 5597 |
| (COL %) | 6.1 | 6.9 | 11.8 | 10.9 | 6.2 |
| Nonincap inj (2) | 11085 | 14 | 249 | 120 | 11468 |
| (COL %) | 12.8 | 12.1 | 14.3 | 13.3 | 12.8 |
| Incapac inj (3) | 17085 | 15 | 315 | 170 | 17585 |
| (COL %) | 19.7 | 12.9 | 18.0 | 18.8 | 19.6 |
| Fatal injury (4) | 39273 | 48 | 399 | 250 | 39970 |
| (COL %) | 45.2 | 41.4 | 22.9 | 27.6 | 44.6 |
| Inj/sev unk (5) | 96 | 1 | 0 | 0 | 97 |
| (COL %) | 0.1 | 0.9 | 0.0 | 0.0 | 0.1 |
| Died prior (7) | 5 | 0 | 0 | 0 | 5 |
| (COL %) | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| (TOTAL) | 86817 | 116 | 1746 | 905 | 89584 |
| (COL %) | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 |

## ** CODES Command **

Command Form: CODES

Keywords: KEY, OUTPUT, VALUES, VARIABLE
Modifiers: CREATE, EMPTY, FORMAT

The CODES command displays variable names and code values for any data set, provided that the data set includes a label file. The command can be used either as a means of on-line data set documentation or as way to produce a codebook for off-line use.

The data set for which variable and code value information is to be displayed need not be the active data set (i.e., need not be first accessed via the DATA command), but it is necessary that it have an associated label file. However, if the active data is selected, additional dictionary information that is available is printed (i.e., the field width, missing data codes, etc.). The list of system data sets in HSRI:FILES provides information on label file availability.

The KEY keyword is used to specify the desired data set. If omitted, information on the active data set is displayed and the full complement of dictionary information that is available for each variable is listed. A subset of variables may be specified with the VARIABLE keyword, and a subset of code values with the VALUES keyword. The OUTPUT keyword may be used to route the information produced by the command to a file or to a printer. The FORMAT modifier adds carriage control and title information to produce a formatted pseudo-codebook.

### ** CODES Keywords and Modifier **

CREATE If the file specified by the OUTPUT keyword does not exist, it is created with a size of 10 pages. The modifier is ignored if the file already exists, or if the file specified is not a simple file.
DEFAULT: No attempt is made to create the specified file.

EMPTY Causes the file specified by the OUTPUT keyword to be emptied. The modifier is ignored if the file specified is not a simple file.
DEFAULT: The output file is not emptied.

FORMAT Produces a formatted output for the listing of code values and variable names. A title consisting of the active data set name and the

date is printed at the top of the listing, and spacing between successive variables is added.
DEFAULT: The formatting features are not implemented.

KEY=<data set descriptor> EXAMPLE: KEY=FARS79AC
Identifies the data set for which variable and code value information will be displayed.
DEFAULT: The active data set.

OUTPUT=<file or device name> EXAMPLE: OUT=BOOK
Specifies the output file or device name where the variable and value information will be written.
DEFAULT: *SINK*

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 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. No more than 100 variables may be specified with this keyword.
DEFAULT: Code values will be printed for all variables.

Example of the CODES Command
(* = user supplied input)

```
  @Command
* ?CODE KEY=NCSS1ACC VAR=1,2
  @VARIABLE    1: TEAM NUMBER (7)
  @      1  Calspan
  @      2  HSRI
  @      3  Indiana U
  @      4  U of Kentucky
  @      5  U of Miami
  @      6  SWRI
  @      7  Dyn Science
  @VARIABLE    2: YEAR
  @      7  1977
  @      8  1978
  @      9  1979

  @Command
* ?CODES K=NCSS1ACC OUT=-CODEBOOK FORMAT

  @Command
  ?
```

## ** DATA Command **

Command Form: DATA

Keywords: KEY, PDN
Modifier: DISMOUNT

The DATA command generates access to any data set available in the system library or to any data set in a properly permitted private user library. All operations necessary to access the data are performed by this command. The files accessed become the active data set and are used in all data manipulations until implicitly released by an ensuing DATA command or explicitly released by the RELEASE command. 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, the command should be given specifying the new data set without specifically releasing the current data set with the RELEASE command.

In the process of generating system access to the requested data set, the DATA command makes a copy the data set dictionary for the active file in the temporary file "-ADASDICT" for subsequent use in other facets of operation. This dictionary file should obviously not be modified during system operation. However, the existence of an on-line dictionary can be used to quickly obtain dictionary information. For instance, an unformatted list of the entire dictionary can be obtained with the command "PRINT FILE=-ADASDICT. The dictionary is in OSIRIS V format and external applications may specify "-ADASDICT(0)" as a system compatible data set dictionary file.

At a minimum, use of the DATA command requires the KEY keyword to specify the data set desired. The keywords and modifier accompanying the DATA command on the same line are described below.

### ** DATA Keywords and Modifier **

DISMOUNT
: When a data set that resides on disk storage is accessed after using a magnetic tape, or if a second magnetic tape is mounted using a different pseudo-device name, the first magnetic tape in use by the system may become inactive. Using the DISMOUNT modifier with the DATA command will cause the automatic dismounting of inactive magnetic tapes.
DEFAULT: The inactive tape will remain mounted.

KEY=<data set descriptor> EXAMPLE: K=MIF77ACC
: Identifies the data set to be accessed. For system data sets, the descriptor is of the form "KEY=MI76ACC." For data sets in a private library

on account CCID, the descriptor is of the form "KEY=CCID:DSNAME." Note that the CCID prefix is required, even when the user accessing the private data set is signed on to that particular ccid. DEFAULT: None.

PDN=<tape pseudo-device name> EXAMPLE: PDN=*IN*
Data sets that reside on magnetic tape are normally mounted on pseudo-device *HSRI*. However, if a magnetic tape is already mounted in MTS (without the program's knowledge) on a pdname other than *HSRI*, or if another pdname is desired for a tape being mounted, the default pdname can be reset through the PDN keyword. DEFAULT: *HSRI*

Example of the DATA Command
(* = user supplied input)

```
  @Command
* ?DATA KEY=MIF77ACC
  @"MIF77ACC": A magnetic tape will be mounted next
  #*HSRI* (C3476A): Mounted on T901
  @"MIF77ACC": Access is established

  @Command
  ?

  .
  .  (Analysis operation(s) on the active data set)
  .


  @Command
* ?DA K=SFQ8:NEWDATA
  @"NEWDATA ": A magnetic tape will be mounted next
  #*HSRI* (C3476A): Dismounted
  #*HSRI* (C4106C): Mounted on T901
  @"NEWDATA ": Access is established

  @Command
  ?
```

## ** DESCRIBE Command **

Command Form: DESCRIBE

Keyword: KEYGROUP
Modifiers: COMMANDS, FILTER, KEY, RECODE, (any valid command name)

The DESCRIBE command displays on-line documentation of system commands and data set descriptors. Use of DESCRIBE alone gives 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 each command may be described in more detail by using DESCRIBE followed by the command name. To obtain information on the descriptors for all data sets that can be accessed via the DATA command, use DESCRIBE KEY.

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

### ** DESCRIBE Keyword and Modifiers **

COMMANDS
: A brief description of available commands is printed.

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

KEY
: A description of groups of data set descriptors is printed together with the code used with the KEYGROUP keyword.

KEYGROUP=<group letter-code> EXAMPLE: K=FED
: A listing of data set descriptors for the group defined by the code is printed at the terminal. The code for each available group may be obtained with the KEY modifier. Only the first letter of the code is necessary

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

Example of the DESCRIBE Command
(* = user supplied input)

```
  @Command
* ?DESC MTS
  @The "MTS" command provides a return to the
  @MTS command mode.  Providing that nothing is
  @done to unload the system, ADAAS may be
  @reentered via the MTS command  "$RESTART."

  @Command
  ?
```

## ** DRIVES Command **

Command Form: DRIVES

The DRIVES command displays information on the current usage of the 9-track and 7-track magnetic tape drives at the Computing Center. Since most ADAAS data sets reside on tape, a free 9-track drive must be available before access to the data set can be established. For a variety of reasons, this command indicates only that a drive may be available, not that one is, in fact, actually available.

Example of the DRIVES Command
(* = user supplied input)

```
  @Command
* ?DRIV


  @At this moment there are AT MOST
  @6 9-Track and 1 7-Track drives free.
  @   3 6250/1600-BPI
  @   3  800/1600-BPI

  @There is one user waiting in the mount queue.

  @

  @Command
  ?
```

## ** FILES Command **

Command Form: FILES Keyword: KEY

The FILES command displays the names of the dictionary, data, and label files for any ADAAS data set, along with the appropriate volume name if any of the files reside on magnetic tape. With the KEY keyword, the file make-up of any system or private data set to which the user has access may be ascertained. Without the KEY keyword, file information for the currently active data set will be displayed (a good way to determine the active data set if there is doubt as to what it is).

## ** FILES Keyword **

KEY=<data set descriptor> EXAMPLE: K=FARS80PR
Identifies the data set for which file information is to be displayed. For system data sets, the descriptor is of the form "KEY=MI76ACC." For data sets in a private library on account CCID, the descriptor is of the form "KEY=CCID:DSNAME."
DEFAULT: The currently active data set.

Example of the FILES Command
(* = user supplied input)

```
  @Command
* ?FILE
  @Data Set Description for "TXS77VEH"
  @Dictionary File Name = TXS77VEHDIC  Volume = TXS77S1
  @      Data File Name = TXS77VEHDAT  Volume = TXS77S1
  @     Label File Name = SP65:TX75LAB

  @Command
  ?
```

## ** LIBRARY Command **

Command Form: LIBRARY

Commands: ADD, DELETE, END, LIST, MODIFY, PERMIT

Command Keywords: ACCESS, DATA, DICT, KEY, LABELS, TAPE

The LIBRARY command provides for the management of data sets that are maintained in a private data set library. With commands given in the Library command mode, users can define the characteristics of their own private data sets that subsequently may be accessed via the DATA command.

When the LIBRARY command itself is entered in ADAAS, program control is transferred to the Library command mode. Any number of Library commands may then be entered. As with other commands, each Library command begins in the first position of the line, with any associated keywords following on the same line. A blank separates the command from the keywords and each keyword from other keywords. Lines may be concatenated by ending each command segment with a dash. The END command returns program control back to the ADAAS command mode.

The first use of the LIBRARY command creates a library file called "ADFILE" on the user's account. Subsequent Library commands use this file to define the make-up of the specified data set: dictionary file, data file, label file (optional), magnetic tape parameters if applicable, and access information.

Though access to any data set may be controlled to some extent by the ACCESS designation given to it (see below), control may also be exercised through the permit status of the file ADFILE. The Library PERMIT command, for example, will give READ access through ADAAS to all other users. A greater degree of control can be achieved with the MTS $PERMIT command. For instance, giving the MTS command "$PERMIT ADFILE, RW CCID" would allow, besides the owner, user CCID to access and manage the library.

If dictionary and data files on disk are to be accessed by other CCID's, they should be entered into the library file with their own CCID prefix attached. In addition, for others to access them, the files themselves must be permitted READ to these users with the MTS $PERMIT command.

The Library commands are described below with any associated keywords listed after each description. Following this, full documentation on the keywords is provided.

## ** LIBRARY Commands **

ADD A new data set is added to the private library. The characteristics of the data set must be specified by an accompanying set of keywords. At a minimum, the keywords KEY, DICT, and DATA must be used, along with TAPE if either, or both, the dictionary and data files reside on magnetic tape. If both reside on tape, it must be the same tape.
KEYWORDS: ACCESS, DATA, DICT, KEY, LABELS, TAPE

DELETE The data set specified by the accompanying KEY keyword is deleted from the private library.
KEYWORD: KEY

END Returns program control to the ADAAS command mode.

LIST Prints out a listing of all data set definitions and tape specifications in the user's library.

MODIFY The data set specified by the accompanying KEY keyword is modified to reflect the values of the other keyword(s) given.
KEYWORDS: ACCESS, DATA, DICT, KEY, LABELS, TAPE

PERMIT Permits the owner's library file ADFILE to be read by all other users through ADAAS. This will allow others to access data sets stored in this (otherwise) private library. Access may still be controlled to some extent with the ACCESS keyword.

## ** LIBRARY Command Keywords **

ACCESS=<user group code(s)> EXAMPLE: ACC=*
Designates the "user group" that the data set will be restricted to. There are ten user groups plus an unrestricted category currently defined:

H - Transportation Data Center
L - UMTRI
N - DOT/NHTSA
M - MVMA
G - Government (other that DOT)
C - Commercial
U - University
1 - Special #1
2 - Special #2
3 - Special #3
* - Unrestricted

One letter is entered for each group desired.

DEFAULT: The owner's access code. If the library is PERMITted, only users with the same group designation as the owner may access the data set.

DATA=<filename,volume,file#> EXAMPLE: DA=PEDDATA,PED1,5
Parameters for the data file are assigned by this keyword. "Filename" is the name of the file. "Volume" is the volume name if the file is on tape. "File#" is the (optional) file number if the file is on tape.
DEFAULT: None when using the ADD command.

DICT=<filename,volume,file#> EXAMPLE: DI=PEDDICT
Parameters for the dictionary file are assigned by this keyword. "Filename" is the name of the file. "Volume" is the volume name if the file is on tape. "File#" is the (optional) file number if the file is on tape.
DEFAULT: None when using the ADD command.

KEY=<data set descriptor> EXAMPLE: K=SFRM:FARS8SUB
Identifies the data set to be ADDed, DELETEd, or MODIFYed. The descriptor is of the form "CCID:DSETNAME." Note that the user-id prefix is required (even when the data set being managed is on the current signon) and that the name itself is limited to 8 characters.
DEFAULT: None

LABELS=<file name>/NONE EXAMPLE: L=SP65:MI78LAB
Specifies the name of the file where code value labels are stored. Most system data sets have label files associated with them. Any that are compatible with the private data set being managed may be specified. If the value "NONE" is specified, the label file name is deleted from the data set specification.
DEFAULT: The data set will have no associated label file.

TAPE=<rack#,vol,extlabel> EX: T=C1232D,PED1,HS0324
TAPE=KEY=<data set descriptor>
Specifies the tape parameters of the magnetic tape where the data file and/or dictionary file may be found. "Rack#" is the Computing Center rack number, "vol" is the tape volume name, and "extlabel" is the external tape label (without enclosing primes). If the tape specification to be used is the same as that of a previously defined data set in the same library, the form "TAPE=KEY=CCID:DSETNAME" may be used.
DEFAULT: None if either the data file or the dictionary file reside on magnetic tape.

Example of the LIBRARY Command
(* = user supplied input)

```
  @Command
* ?LIB
  @LIBRARY Ready. Use "END" to terminate
* ?ADD K=SPPN:80FATALS DICT=80FATALDICT,VOL1 -
* ?DATA=80FATALDATA,VOL1 TAPE=C1234A,VOL1,HS0001
* ?ADD K=SPPN:80ACC DIC=SPNN:ACCDIK DAT=SPNN:ACCDAT
* ?END

  @Command
* ?DATA K=SPNN:80ACC
  @FILE DOES NOT EXIST: SPNN:ACCDIK

  @Command
* ?LIBRARY
  @LIBRARY Ready. Use "END" to terminate
* ?$FILE
  #$FILE
   ACCDAT         ACCDIC        TAPEMOUNT
* ?MOD K=SPNN:80ACC DIC=SPNN:ACCDIC
* ?E

  @Command
* ?DATA K=SPNN:80ACC
  @"80ACC    ": Access is established

  @Command
* ?LIST E
  @Analysis for data set "80ACC    "

  @Filter (or Recode or Title)
  ?
```

## ** LIST Command **

Command Form: LIST

Keyword: OUTPUT
Modifiers: CREATE, EMPTY

The LIST command retrieves and lists a subset of cases and/or variables from the active data set in one of several optional formats. The command is therefore of particular utility when a large number of variables, or alphabetic information, is needed on a small number of cases.

Three output formats are available, designated by the CASE, COLUMN, and PACKED modifiers. The PACKED format produces a listing of all variables for each case as a packed string of characters (useful for data processing). The CASE format produces a listing of selected variables in an unpacked case format (useful for individual case studies). The default COLUMN format produces a listing of the values for selected variables in columns, one case per line (useful for tabular searches). Users should be aware that multiple data passes may occur with the COLUMN format in effect, causing the listing to be printed in sections as the print width allows. The TEST option of the SET command, however, can be used to predict whether or not this will occur before the run is made if there is any doubt.

The LIST command can also be used for producing formatted dictionary listings of data sets whose variable names and characteristics are not known. The parameter statement for doing this is "PACKED VARIABLE=ALLV STOP=0."

### ** LIST Parameter Keywords and Modifiers **

Required parameters:
VARIABLE - Variables to be listed

CASE The variables to be listed are printed in an unpacked case format. The number of each variable is printed with its corresponding value. The variable numbers and values are listed across the page within the selected print width. The LABEL modifier causes the variable name to accompany the variable number, along with the label of the corresponding value if one exists for that particular variable. Normally, with the LABEL option in effect, the variables are listed two across. With the NARROW modifier also specified, the variables are listed one to a line.

DEFAULT: COLUMN format.

COLUMN The variables to be listed are printed in a columnar format, one case per line, with all variable values in columns. The LABEL modifier will cause the values to be replaced with labels if possible. (PLEASE NOTE: only those variables that can be printed within the selected print width are listed in one pass over the data. Use of the modifiers LABEL or NARROW while specifying a large number of variables increases the probability that multiple reads of the data will be required to complete the list. This can add considerably to the cost of the command when processing large data sets).
DEFAULT: This is the default format.

DESCRIBE
A listing of the available program modifiers and keywords is printed at the terminal. If this modifier is present, all other parameters are ignored, and a new parameter statement may be entered.
DEFAULT: No description is printed.

LABEL With the CASE or COLUMN formats, labels, instead of values, are printed for each variable (with the exception of multiple-response variables) if they exist for that variable.
DEFAULT: Values are printed.

LABELS=<label file> EXAMPLE: LAB=SP65:NCSS2LAB
Code value labels for the data set being analyzed are found in the file specified. See the Appendix. for label file construction procedures.
DEFAULT: The label file name is supplied by the system if labels are implemented for the active data set.

NARROW With the CASE or COLUMN formats, the maximum print width is set equal to 70 characters.
DEFAULT: A print width of 130 characters.

NODICT A dictionary listing of the variables accessed is not printed in the output file.
DEFAULT: A dictionary listing is printed.

PACKED Requested variables are listed in a packed case format, with no spaces between variable values. Each listing is printed with a heading that indicates column positions in the data string. If the total record length is greater than 100 characters or the print width (whichever is smaller), the listing is printed in segments. The

PACKED modifier takes precedence over COLUMN or CASE if all happen to be specified on the same parameter line. Keywords and modifiers not applicable to the PACKED format (such as LABEL or PAGE) are also ignored if specified.
DEFAULT: COLUMN format.

PAGE With the COLUMN format, the user-supplied label is printed at the top of each page, along with a heading that specifies the variable number for each column.
DEFAULT: The title and heading are printed only at the top of the first page.

RAND%=<percent> EXAMPLE: RAND=5
A pseudo-random sample of cases that satisfy the global filter criteria are included in the listing. "Percent" is a positive integer number between 1 and 100.
DEFAULT: All cases that satisfy the global filter are included.

SKIP=<number> EXAMPLE: SK=15
Every Nth case (where N is a positive, integer number) that satisfies the global filter criteria is included in the listing.
DEFAULT: All cases that satisfy the global filter are included.

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

STOP=<number> EXAMPLE: STOP=0
Processing is terminated after the specified number of cases have been listed.
DEFAULT: All cases that satisfy the global filter are listed.

VARIABLE=<variable list>/ALLV/ALLNV EXAMPLE: V=1,3,10
Specifies the variables to be included in the listing. The option ALLV specifies that all variables are to be listed. The option ALLNV specifies that all numeric variables are to be listed.
DEFAULT: No default.

** LIST Input and Output Examples **

Examples of LIST Parameter Statements
(* = user supplied input)

```
  @Enter Parameters:
* ?VAR=55,3,1,55,112-115 LABEL
```

The example above produces a listing, in COLUMN format, of variables 55, 3, 1, 55 again, and 112 through 115. Variable 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, thus allowing the user considerable control over the exact format of the listing.

```
  @Enter Parameters:
* ?CASE V=1,101,201-218 LABEL
```

The example above produces a listing, in CASE format, of variables 1, 101, and 201 through 218. Variable labels, rather than values, are used in the listing. Because all the variables selected for each case are listed before the next case is processed, the user need not be concerned with the possibility of multiple reads of the data.

The following three examples demonstrate the three formats available with the LIST command. All three examples used the BMCS 1979 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. In addition, the PACKED output lines have been trimmed to fit on the page.

Example of LIST Output
(Statement: "PACKED VAR=ALLV STOP=4" )

```
Data Set List - PACKED mode
Date: SEP 23, 1982  at  09:35:17
LIST OF FOUR 1979 BMCS CASES USING "PACKED" FORMAT

   1   1        10        20        30        40        50
       340703921042050957901032879802001019898231420004
       912330000000001000010000048000201112614 13

   2   1        10        20        30        40        50
       344571711182050557912292915809002020913998124402 08
       922330000000000000000000007620201111244 13

   3   1        10        20        30        40        50
       347594221183030527901082880322302070322998123200 02
       621330000000000010000010000180020112262423

   4   1        10        20        30        40        50
       348064211171030517901102880512002040698981233060 7
       622330000000000000000000000260020112511313

Cases Read  =      4
Cases Listed=      4
```

Example of LIST Output
(Statement: "COLUMN LABEL V=1,3-4,11-12,14 STOP=4" )

```
Data Set List - COLUMN mode
Date: JUN 15, 1982  at  14:13:58
LIST OF FOUR 1979 BMCS CASES USING "COLUMN" FORMAT

VAR.#=     1          3              4    11  12         14
    1  34070  Authorized  Over-the-road  Jan  03   2:00 am
    2  34457     Private  Over-the-road  Dec  29   9:00 am
    3  34759  Authorized  Over-the-road  Jan  08  10:00 pm
    4  34806     Private  Over-the-road  Jan  10      Noon

Cases Read  =      4
Cases Listed=      4
```

Example of LIST Output
(Statement: "CASE LABEL NARROW V=1,3-4,11-12,14 STOP=4" )

```
Data Set List - CASE mode
Date: JUN 15, 1982  at  14:13:58
LIST OF FOUR 1979 BMCS CASES USING "CASE" FORMAT

CASE#      1
   V1:RECORD ID NUMBER          = 34070
   V3:TYPE OF CARRIER           = Authorized
   V4:TYPE OF TRIP              = Over-the-road
  V11:MONTH OF CRASH            = Jan
  V12:DAY OF CRASH              = 03
  V14:HOUR OF CRASH             = 2:00 am

CASE#      2
   V1:RECORD ID NUMBER          = 34457
   V3:TYPE OF CARRIER           = Private
   V4:TYPE OF TRIP              = Over-the-road
  V11:MONTH OF CRASH            = Dec
  V12:DAY OF CRASH              = 29
  V14:HOUR OF CRASH             = 9:00 am

CASE#      3
   V1:RECORD ID NUMBER          = 34759
   V3:TYPE OF CARRIER           = Authorized
   V4:TYPE OF TRIP              = Over-the-road
  V11:MONTH OF CRASH            = Jan
  V12:DAY OF CRASH              = 08
  V14:HOUR OF CRASH             = 10:00 pm

CASE#      4
   V1:RECORD ID NUMBER          = 34806
   V3:TYPE OF CARRIER           = Private
   V4:TYPE OF TRIP              = Over-the-road
  V11:MONTH OF CRASH            = Jan
  V12:DAY OF CRASH              = 10
  V14:HOUR OF CRASH             = Noon

Cases Read  =      4
Cases Listed=      4
```

## ** MESSAGE Command **

Command Form: MESSAGE

The MESSAGE command permits a message to be sent to the Transportation Data Center. After the command is entered the prompt "Text:" will be printed and the message may be entered a line at a time in response to the "?" sign. To terminate text entry, enter an empty line (i.e., a RETURN with no characters typed), a CONTROL-C, or $ENDFILE. Replies to any questions will be sent to the user through the MTS message system.

Example of the MESSAGE Command
(* = user supplied input)

```
  @Command
* ?MESSAGE
   Text:
* ?Please send a copy of the new NASS 1980 codebook.
* ?Also, do you have any idea when the 1983 FARS data
* ?will be available?
* ?Thanks - Jim S.
* ?
   Message forwarded from mailbox HSRI to SKD7.
   Message sent (114588).

  @Command
  ?
```

## ** MIDAS Command **

Command Form: MIDAS

The MIDAS command controls the active data set (accessed by the DATA command) for use by the Michigan Interactive Data Analysis System (MIDAS). Because ADAAS dictionary and data files are similar in structure to the dictionary and data files of OSIRIS, the MIDAS "OSIRIS" command can be used to input HSRI data sets into MIDAS for use by that system. When the MIDAS command is given in ADAAS, the proper form of the MIDAS/OSIRIS command's "FILE" keyword is printed at the terminal and ADAAS program execution is terminated.

Example of the MIDAS Command
(* = user supplied input)

```
  @Command
* ?DATA K=FARS78AC
  @"FARS78AC": A magnetic tape will be mounted next
  #*HSRI* (C0867C): Mounted on T903
  @"FARS78AC": Access is established

  @Command
* ?MIDAS
  @The MIDAS/OSIRIS command "FILE" keyword for
   "FARS78AC" is:
  @    FILE=*HSRI* FARSCRA8DIC;*HSRI* FARSCRA8DAT
  @Note: a magnetic tape is mounted.
  #EXECUTION TERMINATED
  #
```

## ** MTS Command **

Command Form: MTS

The MTS command returns control to MTS without unloading ADAAS. If no commands that invoke the loader are issued while in MTS (i.e., $RUN, $LOAD, etc.), the command $RESTART may be used to reenter ADAAS at the point where the MTS command was issued.

Example of the MTS Command
(* = user supplied input)

```
  @Command
* ?MTS
* #F
   ADAAS.CONFER BATCHSETUP   FARSOUTPUT   MYFILE
   TRUCKRUN     UNIVARIATES
* #RES

  @Command
  ?
```

## ** NEWS Command **

Command Form: NEWS

The NEWS command displays a listing, by date, of recent changes to the programs or additions to the accident data base.

Example of the NEWS Command
(* = user supplied input)

```
  @Command
* ?NEWS
  @Changes to the HSRI Accident Data System are listed
  @below in reverse chronological order.

  @10-03-80
  @The System library was updated at 08:02.  FARS 76
  @and 77 updated to Versions 25A and 21A, respec-
  @tively.  Washington Census pedestrian data sets
  @reformatted.  CPIR Version C Light Truck Data Sets
  @added.  See ADAAS NEWSLETTER, Volume 7, Number 5
  @for details.

  @09-22-80
  @A new version of the DATA command was implemented
* @at !
  @+ATTN+

  @Command
  ?
```

## ** PRINT Command **

Command Form: PRINT

Keywords: COPIES, FILE, ROUTE
Modifiers: LIST, TN, X9700

The PRINT command causes the contents of an MTS file to be either copied or listed on the user's terminal or, alternatively, on a printer at a batch print station. By default, the specified file is copied rather than listed. This may be overridden with the LIST modifier. If the COPIES or ROUTE keyword is used, or the TN or X9700 modifier is included, the output is routed to a batch print station. Otherwise, the file contents print out on the terminal. The PAGE modifier may be used to simulate printer output on a terminal so that lines that begin a new page on the printer will also begin a new page on the terminal.

### ** PRINT Keywords and Modifiers **

COPIES=<number> EXAMPLE: COP=2
Designates the number of copies of the file to be printed. Up to 9 copies may be specified.
DEFAULT: One copy will be printed.

FILE=<file name> EXAMPLE: F=-TABLE
Specifies the file whose contents will be copied or listed. Note: more than one file may be printed by means of concatenation. For example, specifying the files "-OUT1+-OUT2+-OUT3" would print out a copy of each.
DEFAULT: The file "-PRINT."

LIST Causes the file to be listed rather than copied. Listing results in file line numbers being printed along with the contents of the lines, and effectively disables the action of carriage control characters.
DEFAULT: The file is copied.

PAGE Enables printer-like carriage control on hard-copy terminals. The action of this modifier is based on an 11 inch paper size and a vertical print pitch of six lines per inch. After issuing the PRINT command with the PAGE modifier, the user is prompted to position the print head at the top of page and hit RETURN. After this action, carriage control is simulated by printing extra lines.
DEFAULT: Standard terminal carriage control is used.

ROUTE=<4-letter route code> EXAMPLE: R=NHSB

Designates the batch print station where the output is to be sent. Valid route codes for some important stations are:

| | |
|---|---|
| CNTR | U-M Computing Center - Ann Arbor |
| DRBN | U-M Computing Center - Dearborn, MI |
| FORD | Ford Motor Company - Dearborn, MI |
| NHSB | NHTSA/Nassif Bldg - Washington, DC |
| NUBS | U-M Central Campus Station - Ann Arbor |
| SWRI | Southwest Res Inst - San Antonio, TX |

DEFAULT: The default route is CNTR.

X9700 Output will be printed on the Xerox 9700 laser printer at the U-M North Campus Computing Center. This printer produces high-quality upper/lower case output on 8 1/2 by 11 paper. The ROUTE keyword and TN modifier are ignored if present when the X9700 modifier is used.
DEFAULT: Output will be routed to a line printer if batch printing is indicated.

Example of the PRINT Command
(* = user supplied input)

```
  @Command
* ?P
  @FILE=-PRINT




  >TIME: 09:00:06
  >DATE: SEP 18, 1980

  >FILTER:
  >INCLUDEV11=1

* !
  >ATTN!

  @Command
* ?P F=-FIRSTOUT+-PRINT ROUTE=DRBN
  #*PRINT* ASSIGNED RECEIPT NUMBER 656249
  #*PRINT* 656249 RELEASED TO DRBN, 10 PAGES
  @ROUTE=DRBN COPIES=1 MODE=C PRINT=PN
  @FILE=-FIRSTOUT+-PRINT

  @Command
  ?
```

## ** RELEASE Command **

Command Form: RELEASE

The RELEASE command removes access to the current data set. If the data set resides on magnetic tape, then the tape is dismounted. This command should not be used between successive DATA commands, but rather whenever no further data manipulation is contemplated on any data set.

Example of the RELEASE Command
(* = user supplied input)

```
  @Command
* ?REL
  #T901 RELEASED
  @"NEWDATA ": The active data set is released

  @Command
  ?
```

## ** SET Command **

Command Form: SET Keywords: INPUT, TERSE, TEST TIME

The SET command controls several aspects of program operation. With the TERSE keyword, a terse mode may be specified so that abbreviated messages are issued by the system. With the TIME keyword, a timer interrupt may be set so that the program will indicate when certain amounts of CPU time have been used. With the TEST keyword, the program may be put in a "test" mode to debug analysis setup routines. And with the INPUT keyword, the command input stream may be directed to a file that contains previously prepared commands so that they may be executed automatically.

The INPUT keyword is especially useful. For ADAAS runs that are done frequently or that involve a good deal of forethought, the user can place all the necessary commands, filters, recodes, and parameter statements as they would normally be entered from the terminal into a file, check it for errors, and input it into the system for quick processing. Not only can this reduce the time spent in the system, but if any errors are encountered, the file need only be edited and input again. The input file may also be delimited with a line-number range. The command "SET INPUT=SETUPFILE(3)", for example, would begin reading the ADAAS commands in the user file SETUPFILE from line 3 rather than line 1.

### ** SET Command Keywords **

INPUT=<file/device name>/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 input "*MSOURCE*" (the terminal or card reader).
DEFAULT: Input is read from *MSOURCE*

TERSE=ON/OFF EXAMPLE: TERSE=ON

If TERSE is set ON, input read from a file designated by the INPUT keyword is not echoed on the terminal.
DEFAULT: TERSE=OFF

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 parameter statements, but not do any actual processing. This mode is useful for

checking analysis setup routines for possible errors before doing the actual run.
DEFAULT: TEST=OFF

TIME=<integer number N> EXAMPLE: TIME=15
Sets the timer interrupt interval to N seconds of CPU time (N is a positive integer). One second of CPU time costs $0.30 at current normal priority rates for University and government users, about $0.65 for industrial users.
DEFAULT: 100 seconds of CPU time.

Example of the SET Command's TIME Feature
(* = user supplied input)

```
  @Command
* ?SET TIME=1

  @Command
* ?DATA KE=WAC80PED
  @"WAC80PED": A magnetic tape will be mounted next
  #*HSRI* (C0171C): Mounted on T908
  @"WAC80PED": Access is established

  @Command
* @UNIVAR OUT=-TEMP EM
  @Analysis for file "WAC80PED"

  @Filter (or Recode or Title)
* @1980 WASHINGTON PEDESTRIAN STATISTICS

  @Enter Parameters:
* @V=301-315 STAT
  @Processing Begins
  @ 13:34:25:     1.19 DOLLARS USED.
  @ 13:34:29:     1.60 DOLLARS USED.
  @Cases Read=   3974
  @Bulkfile Line Numbers for Statistics:
  @Start=   32  End=   49
  @Processing Completed

  @Command
  ?
```

Example of the SET Command's INPUT Feature
(* = user supplied input)

```
  @Command
* ?PRINT F=FATALSETUP
   DATA K=FARS79VH
   BIVAR OUT=-TABLE E
   INCLUDE V105=1-9
   RECODE V103 (1-13)=71,(14-17,21,24)=72,-
   (18-20,22,23,26)=73,(ELSE)=74
   DOMESTIC VS. FOREIGN PASS VEHICLE FATAL COUNTS
   CVR=103:71-74 RV=155 DONE

  @Command
* ?SET IN=FATALSETUP

  @Command
  @>>>>DATA K=FARS79VH
  @"FARS79VH": A magnetic tape will be mounted next
  #*HSRI* (C0870D): Mounted on T903
  @"FARS79VH": Access is established

  @Command
  @>>>>BIVAR OUT=-TABLE E
  @Analysis for file "FARS79VH"

  @Filter (or Recode or Title)
  @>>>>INCLUDE V105=1-9

  @Recode (or Title)
  @>>>>RECODE V103 (1-13)=71,(14-17,21,24)=72,-
  @     (18-20,22,23,26)=73,(ELSE)=74

  @Recode (or Title)
  @>>>>DOMESTIC VS. FOREIGN PASS VEHICLE FATAL COUNTS

  @Enter Parameters:
  @>>>>CVR=103:71-74 RV=155 DONE
  @Processing Begins
  @Cases Read=  34372
  @Bulkfile Table & Line Numbers:
  @     1     46
  @Processing Completed

  @Command
  @END OF FILE - Input returned to Terminal

  @Command
  ?
```

## ** SPSS Command **

Command Form: SPSS Keywords: DATA, DICT

The SPSS command controls the currently active data set (accessed via the DATA command) for use by the Statistical Package for the Social Sciences (SPSS) program. Dictionary files, and data files that are stored on disk, are copied into SPSS-compatible files on the user's signon ID. The proper MTS commands to execute the SPSS program are printed at the terminal, and ADAAS program execution is terminated. The SPSS "OSIRIS VARS" command can then be used to input the data set into SPSS for use by that system.

## ** SPSS Keywords **

DATA=<file name> EXAMPLE: DAT=WA78SPSSDAT
Specifies the file where the SPSS-compatible data will be written. Not needed (or recommended) if the original data file resides on magnetic tape.
DEFAULT: The file "-SPSSDATA" if the original data file is a disk file.

DICT=<file name> EXAMPLE: DIC=WA78SPSSDIC
Specifies the file where the SPSS-compatible dictionary will be written.
DEFAULT: The file "-SPSSDICT."

Example of the SPSS Command
(* = user supplied input)

```
  @Command
* ?DA K=BMCS-78
  @"BMCS-78 ": A magnetic tape will be mounted next
  #*HSRI* (C0170C): Mounted on T900
  @"BMCS-78 ": Access is established

  @Command
* ?SPSS
  @TO RUN SPSS, ISSUE THE FOLLOWING COMMANDS:
  @
  @     $CONTROL *HSRI* POSN BMCS78
  @     $RUN UNSP:SPSS 7=-SPSSDICT 8=*HSRI*
  @
  #EXECUTION TERMINATED
  #
```

## ** STATUS Command **

Command Form: STATUS

The STATUS command displays a number of parameters shown in the table below that are relevant to operation of ADAAS.

Example of the STATUS Command
(* = user supplied input)

```
  @Command
* ?STATUS
  @Status of user CCID on OCT 25, 1982 at 13:00:00
  @System is operating in terminal mode
  @Total cost ($)  0.16
  @Remaining funds ($) 183.50
  @The active data set is FARS80PR
  @The data set access code is G
  @Test mode is OFF
  @Terse mode is OFF
  @Timer interrupt interval is 100 sec.
  @Program is reading input from *MSOURCE*
  @Tapes mounted: *HSRI* (C3437B) on T902
  @               *WORK* (N1539D) on T906

  @Command
  ?
```

## ** STOP Command **

Command Form: STOP

The STOP command terminates program operation and returns control to MTS. This command should only be issued when all desired operations are completed. A warning message is printed if a tape is mounted at the time that the command is issued.

Example of the STOP Command
(* = user supplied input)

```
  @Command
* ?STOP
  @Note: A magnetic tape is mounted.
  #EXECUTION TERMINATED
  #
```

## ** SUBSET Command **

Command Form: SUBSET Keyword: OUTPUT
Modifiers: CREATE, EMPTY

The SUBSET command creates a new data set from the active data set that includes only those cases and variables selected by the user. Cases to be included are determined by the FILTER statement. Variables are selected in the parameter statement. The new data set may be written into disk files or onto magnetic tape, may be subsequently entered into the user's private library with the LIBRARY command, and may then be accessed like any other data set with the DATA command. The RECODE statement may be used to effect a permanent recode of variables. For example, an alphabetic variable may be converted into a numeric variable with no change in field width.

When not otherwise specified, the data and dictionary files making up the new data set are written into the temporary files "-DATA" and "-DICT", respectively.

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

### ** SUBSET Parameter Keywords and Modifiers **

Required parameters:
ALLV or ALLNV or VARIABLE - Variable list

ALLV All variables in the active data set are included in the new data set.
DEFAULT: No default. Either the ALLV or ALLNV modifier or the VARIABLE keyword must be used.

ALLNV All numeric variables in the active data set are included in the new data set.
DEFAULT: No default. Either the ALLV or ALLNV modifier or the VARIABLE keyword must be used.

DATA=<filename> EXAMPLE: DAT=NEW80DATA
DATA=<filename,pdn,file#,vol> EX: DA=ACCDATA,*T*,0,TAPE4
The data produced by the subset is written into the file specified. If the data is to be written onto magnetic tape, "pdn" specifies the pseudo-

device name of the user-mounted tape, "file#" indicates the file number of the data, and "vol" specifies the tape volume name. If the file number is given as "0" (or zero), the file is written at the end-of-tape. If the file number is greater than zero, the new file will be written at the file position indicated, effectively destroying any previously-written data after the file on the tape.
DEFAULT: The temporary file "-DATA."

DATAFMT=<lrecl,blkfac> EXAMPLE: DATAF=198,165
Specifies the blocking parameters for the output data file. Need only be used if special blocking is required. "Lrecl" is an integer representing the logical record length of each case, while "blkfac" is an integer representing the blocking factor such that "lrecl * blkfac" is an integer between 1 and 32767.
DEFAULT: The logical record length defaults to the value for the output data. The blocking factor defaults to 1 for disk files and a value such that "lrecl * blkfac" is less than or equal to 28000 for tape files.

DESCRIBE
A listing of the available program modifiers and keywords is printed at the terminal. If this modifier is present, all other parameters are ignored, and a new parameter statement may be entered.
DEFAULT: No description is printed.

DICT=<filename> EXAMPLE: DIC=NEW80DICT
DICT=<filename,pdn,file#,vol> EX: DI=ACCDICT,*T*,0,TAPE4
The dictionary produced by the subset is written into the file specified. If the dictionary is to be written onto magnetic tape, "pdn" specifies the pseudo-device name of the user-mounted tape, "file#" indicates the file number of the dictionary, and "vol" specifies the tape volume name. If the file number is given as "0" (or zero), the file is written at the end-of-tape. If the file number is greater than zero, the new file will be written at the file position indicated, effectively destroying any previously-written data after the file on the tape.
DEFAULT: The temporary file "-DICT."

DICTFMT=<lrecl,blkfac> EXAMPLE: DICTF=80,200
Specifies the blocking parameters for the output dictionary file. Need only be used if special blocking is required. "Lrecl" is an integer representing the logical record length of each

case, while "blkfac" is an integer representing the blocking factor such that "lrecl * blkfac" is an integer between 1 and 32767.
DEFAULT: The logical record length defaults to the value for the output data. The blocking factor defaults to 1 for disk files and a value such that "lrecl * blkfac" is less than or equal to 28000 for tape files.

NODICT A listing of the new data set dictionary is not printed in the documentation file.
DEFAULT: A listing is printed.

PRESERVE
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 cases that satisfy the global filter criteria make up the new data set. "Percent" is a positive integer number between 1 and 100.
DEFAULT: All cases 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.

SKIP=<number> EXAMPLE: SKIP=100
Every Nth case (where N is a positive, integer number) that satisfies the global filter criteria is included in the new data set.
DEFAULT: All cases 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=<number> EXAMPLE: STOP=9000
Processing is terminated after the specified number of cases have been written into the data file.

DEFAULT: All cases that satisfy the global filter are included.

TYPE1

The dictionary produced by the SUBSET command is an OSIRIS Type 1 dictionary.
DEFAULT: This is the default.

TYPE5

The dictionary produced by the SUBSET command is an OSIRIS Type 5 dictionary.
DEFAULT: A OSIRIS Type 1 dictionary is produced.

VARIABLE=<variable list>/ALLV/ALLNV EXAMPLE: VA=1,24

Specifies the variables to be included in the new output data set. The option ALLV is equivalent to using the modifier ALLV, and the option ALLNV is equivalent to using the modifier ALLNV.
DEFAULT: No default. Either the ALLV or ALLNV modifier or the VARIABLE keyword must be used.

** SUBSET Input and Output Examples **

Example of a SUBSET Parameter Statement
(* = user supplied input)

```
  @Enter Parameters:
* ?VAR=1-10,12,20-30 DICT=CARDIC,*OUT*,0,MYTAPE -
* ?DATA=CARDAT,*OUT*,0,MYTAPE
```

The new data set produced with the example above includes variables 1 through 10, 12, and 20 through 30 from the active data set. The data set is put onto tape: the dictionary into a file called "CARDIC" and the data into a file called "CARDAT", the dictionary in front of the data file at the end of the tape (file number "0"). The tape was mounted prior to the run with the command:

```
$MOUNT CnnnnA 9TP *OUT* WRITE=YES VOL=MYTAPE 'extlabel'
```

where "CnnnnA" is the tape rack number and "extlabel" is the tape's external label.

Example of a SUBSET Setup File

```
DATA KEY=WAC79PED
SUBSET OUT=-DICTLIST
INCLUDE V307=4 AND V26=0-43,45,47-99
WASHINGTON 1979 BICYCLE SUBSET
VAR=3,6-7,19,20,23-25,37,102,129-130,134,137,301,-
303-306,308-312,314 DI=WA79BIKEDIC DA=WA79BIKEDAT
LIB
ADD K=CCID:WA79BIKE DICT=CCID:WA79BIKEDIC -
    DATA=CCID:WA79BIKEDAT -
    LAB=SP65:WA79LAB ACC=*
PERMIT
END
```

The file above, run with the "SET INPUT=" feature of ADAAS, produced a subset of the 1979 Washington Pedestrian data set. Included were cases where variable 307 equaled 4 and variable 26 did not equal 44 or 46. The new data set included a total of 25 variables from the original data set. The dictionary was written into the disk file "WA79BIKEDIC", while the data went into the disk file "WA79BIKEDAT." Both these files were explicitly created by the user prior to the run. The file size of the resulting output dictionary was 1 page, while the output data file was 17 pages (it consisted of a little over 1000 cases, each with a logical record length of 51). The data set was then added to a private library with the LIBRARY command.

The data set created with this setup file could be subsequently accessed by the user with the command "DATA KEY=CCID:WA79BIKE." Because the files making up the data set resided on direct-access disk, the data set could be analysed at considerable savings over a data set residing on magnetic tape. Even a frequently accessed portion of a large data set, however, could be analysed more cost effectively by subsetting it onto a user-provided tape than by reading and filtering through the large data set each time a run was made.

## ** TAPE Command **

Command Form: TAPE

The TAPE command controls the active data set (resident on magnetic tape and accessed via the DATA command) for use external to ADAAS. The command will print out the file names of the currently active data set and terminate program operation.

Example of the TAPE Command
(* = user supplied input)

```
  @Command
* ?DATA KEY=WAC78VEH
  @"WAC78VEH": A magnetic tape will be mounted next
  #*HSRI* (C1616C): Mounted on T908
  @"WAC78VEH": Access is established

  @Command
* ?TAPE
  @Data Set Description for "WAC78VEH"
  @Dictionary File Name = WA78VEHDIC    Volume = WA78S1
  @      Data File Name = WA78VEHDAT    Volume = WA78S1
  @     Label File Name = SP65:WACLAB
  @Note: a magnetic tape is mounted.
  #EXECUTION TERMINATED
  #
```

## ** TWOWAY Command **

Command Form: TWOWAY

Keyword: OUTPUT
Modifiers: CREATE, EMPTY

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. Tables with virtually unlimited dimensions (e.g., 10 by 10, 100 by 1000, etc.) may be generated with only the amount of core storage available from the computer acting as an upper limit. Users should be aware, however, of large memory and processing costs involved in the utilization of very large tables. Only columns and rows with non-zero elements are printed.

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.

### ** TWOWAY Parameter Keywords and Modifiers **

Certain keywords and modifiers are global in nature and apply to the entire run rather than to a single table. Identified by "**GLOBAL**" below, these parameters need be entered only once for a set of tables.

Required parameters:
- CV or CVR - Column variable
- RV or RVR - 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. Either the keyword CV or CVR must be used.

CVR=<variable:valuerange> EXAMPLE: CVR=25:10-15
Column variable with a range. Only those cases where the variable has a value in the indicated range are included in the table. The range itself should consist of a single value, or two values, separated by a hyphen, the second greater than the first. If no range is specified, no selection is performed. If the CV and CVR keywords are entered on the same parameter input line, the CV keyword is ignored.
DEFAULT: No default. Either the keyword CV or CVR must be used.

DESCRIBE
A list of available keywords and modifiers is printed at the terminal. If present, all other parameters entered on the same line are ignored.
DEFAULT: No description is printed.

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 cases where the specified variable has a value in the indicated range are included in the table. The Analysis Command Overview's description of the parameter statement provides full information on the use of local filters.
DEFAULT: All cases 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).

ID=<24-character tag> EXAMPLE: ID=ROLLOVER
A table identifier that consists of an twenty-four character tag with no imbedded blanks. It may be supplied for each table.

DEFAULT: No ID is used.

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

LABELS=<label file> EXAMPLE: LAB=SP65:NCSS2LAB
Code value labels for the data set being analyzed are found in the file specified. See the Appendix. for label file construction procedures.
**GLOBAL**
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.

NODICT A dictionary listing of the variables accessed is not printed in the output file.
DEFAULT: A dictionary listing is printed.

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

NOLABEL 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.

NOTERM For conversational mode only. No table numbers or beginning output file line numbers are printed at the terminal.
DEFAULT: The table number(s) and output file line number(s) are printed on-line.

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. Either the keyword RV or RVR must be used.

RVR=<variable:valuerange> EXAMPLE: RVR=10:100-199
Row variable with a range. Only those cases where the variable has a value in the indicated range are included in the table. The range itself should consist of a single value, or two values, separated by a hyphen, the second greater than the first. If no range is specified, no filtering is performed. If the RV and RVR keywords are entered on the same parameter input line, the RV keyword is ignored.
DEFAULT: No default. Either the keyword RV or RVR must be used.

SAME All parameters not explicitly specified on the current parameter input line are set equal to their value defined on the previous line. If CV and/or RV lists were specified on the preceding line, then the entire set of tables are respecified.
DEFAULT: All parameters desired must be specified.

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
WV=<variable:valuerange> EXAMPLE: WV=102:1-3
Weight variable number. A variable with a value range may optionally be specified, causing only those cases that have a value within the specified range to be included. If no range is specified, only cases for which the weight variable has a non-missing data value are included in the table. To be meaningful, this variable should contain interval data (i.e., values that have a direct correspondence to the information they represent).
DEFAULT: No weighting is performed.

** TWOWAY Input and Output Examples **

Example of the TWOWAY Parameter Statement
(* = user supplied input)

```
  @Enter Parameters:
* ?RV=103 CV=105
* ?SAME WV=145
* ?DONE
```

Two TWOWAY tables are produced with the example above. The first uses variable 103 for the row variable values and variable 105 for the column variable values. If variable 103 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 145 is used to weight the distribution. If variable 103 was a vehicle make-model code and variable 105 was a vehicle model year code, the number in each cell of table one would represent the occurrence frequency of a particular make-model for a particular model year. If variable 145 was a counter indicating the total number of persons killed in the vehicle, the number in each cell of table two would represent the total killed for a particular make-model for a particular model year.

Example of a TWOWAY Setup File

```
DATA K=FARS80VH
TWOWAY OUT=TBBL CREATE
INCLUDE V109=1-2
JACKKNIFE OCCURRENCE BY VEHICLE BODY TYPE
RV=104 CV=109 ROW% DONE
```

The file above, run with the "SET INPUT=" feature of ADAAS, produced the output below (in addition to global information that precedes the tables such as a dictionary listing).

Example of TWOWAY Output
(produced from setup file above)

Table No. 1
Date: AUG 12, 1982 at 01:04:37

JACKKNIFE OCCURRENCE BY VEHICLE BODY TYPE
BIVARIATE FREQUENCIES AND PERCENTAGES
COLUMN VAR. NO. 109 : JACKKNIFE
ROW VAR. NO. 104 : BODY TYPE
OPTIONS: FREQ ROW% MD%

| | MD - No 1 | Yes 2 | (TOTAL) |
|---|---|---|---|
| Trktract/1 trlr | | | |
| 57 | 3249 | 342 | 3591 |
| (ROW %) | 90.5 | 9.5 | 100.0 |
| Trktract/2+ trlr | | | |
| 58 | 133 | 21 | 154 |
| (ROW %) | 86.4 | 13.6 | 100.0 |
| Unk body type | | | |
| 99 | 3 | 0 | 3 |
| (ROW %) | 100.0 | 0.0 | 100.0 |
| (TOTAL) | 3385 | 363 | 3748 |
| (ROW %) | 90.3 | 9.7 | 100.0 |

## ** UNIVAR Command **

Command Form: UNIVAR Keyword: OUTPUT
Modifiers: CREATE, EMPTY

The UNIVAR command generates one-way frequency distributions, descriptive statistics, and histograms for selected variables in the active data set. Any number of variables may used, with any number of levels per variable. Users should be aware, however, of the costs involved in processing variables with large numbers of code values.

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. Statistical information includes the minimum value of the code values for the variable, the maximum value, the mean value, the standard deviation, and the number of occurrences of missing data codes one and two (if assigned). The histogram option provides a graphical display of the non-zero code values for each variable together with code values, labels, and frequencies.

### ** UNIVAR Parameter Keywords and Modifiers **

Required parameters:
VARIABLE - Variable list

COLUMN Univariate frequencies are printed in a columnar format with one line for each code value.
DEFAULT: A print width of 132 characters.

DESCRIBE
A listing of the available program modifiers and keywords is displayed at the terminal. If this modifier is present, all other parameters are ignored, and a new parameter statement may be entered.
DEFAULT: No description is printed.

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 either the SKEW or STAT modifiers are 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 cases satisfying the global filter criteria.
DEFAULT: No percentages are calculated.

LABELS=<label file> EXAMPLE: LAB=SP65:NCSS2LAB
Code value labels for the data set being analyzed are found in the file specified. See the Appendix. for label file construction procedures.
DEFAULT: The label file name is supplied by the system if labels are implemented for the active data set.

NARROW Print width for the univariate frequency output is set to 70 characters. This permits printing on most terminals.
DEFAULT: A print width of 132 characters.

NODICT A dictionary listing of the variables accessed is not printed in the output file.
DEFAULT: A dictionary listing is printed.

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

NOTERM Output file line numbers designating the beginning location of statistics and/or frequencies are not printed at the terminal.
DEFAULT: For conversational mode, the output file line numbers are printed on-line. In batch, NOTERM is the default.

PAGE The user-supplied title is printed at the top of each page, along with either the phrase "UNIVARIATE FREQUENCY DISTRIBUTION" or "DESCRIPTIVE STATISTICS" on the following line.
DEFAULT: This is the default. No paging is performed, however, if COLUMN or NARROW have been specified.

PLOT A histogram of the occurrence frequencies for each variable is generated.
DEFAULT: No histograms are produced.

STAT Descriptive statistics (excluding the third and fourth moments) are calculated for all variables in the variable list.
DEFAULT: No statistics are computed.

STOP=<number> EXAMPLE: STOP=100
Processing is terminated after the specified number of cases have been listed.

DEFAULT: All cases that satisfy the global filter are processed.

VARIABLE=<variable list> EXAMPLE: VA=109-114,204
Specifies the variable(s) for which univariate frequency information will be generated. Up to 200 variables of type numeric may be included, each having a frequency of occurrence of no more than 2500 distinct values.
DEFAULT: None.

WV=<variable> EXAMPLE: WV=240
Frequencies are weighted by the specified variable.
DEFAULT: Frequencies are not weighted.

** UNIVAR Input and Output Examples **

Example of the UNIVAR Parameter Statement
(* = user supplied input)

```
  @Enter Parameters:
* ?FREQ VAR=12,34-37
```

The example above produces one-way frequency of occurrence information for variables 12, 34, 35, 36, and 37.

Example of a UNIVAR Setup File

```
DATA K=WAC80ACC
UNIVAR E
INCLUDE V23=4
ACCIDENTS ON ICY ROADS / MISCELLANEOUS ASPECTS
FREQ% STAT V=3,23,24 COLUMN PLOT
```

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:

| | |
|---|---|
| VAR | The variable number |
| # NOMD | The number of cases with no missing data |
| MIN | The minimum code value |
| MAX | The maximum code value |
| MEAN | The mean (or average) code value |

| | |
|---|---|
| S.D. | The standard deviation |
| # MD1 | The number of cases with the first missing data code value |
| # MD2 | The number of cases with the second missing data code value |

The categories for the frequency information are defined as follows:

| | |
|---|---|
| CODE | The variable's code values |
| LABEL | The code value's associated label |
| FREQ | The code value's frequency of occurrence |
| PRCNT | The percentage that this frequency represents of the total number of cases ("N") |

Example of UNIVAR statistics output
(produced from the setup file on the previous page)

```
Univariate Statistics
Date: OCT 28, 1982  at  16:05:13
ACCIDENTS ON ICY ROADS / MISCELLANEOUS ASPECTS

Number of cases = 4553  Weight sum = 4553.

  Var   Min   Max   Mean   S.D.   NoMD    MD1    MD2
  ---   ---   ---   ----   ----   ----    ---    ---
    3     1    12    5.5    5.0 4553.     0.     0.
```

Example of UNIVAR frequency output
(produced from the setup file on the previous page)

```
Univariate Frequency Distribution
Date: OCT 28, 1982  at  16:05:13
ACCIDENTS ON ICY ROADS / MISCELLANEOUS ASPECTS


***Variable 3 MONTH OF ACCIDENT          Cases = 4553

 Code Label       Freq Prcnt *
 ---- -----       ---- ----- *
    1 January    1592.  35.0 *
    2 February   1091.  24.0 *
    3 March        74.   1.6 *
    4 April         5.   0.1 *
    5 May           1.   0.0 *
   10 October      14.   0.3 *
   11 November    299.   6.6 *
   12 December   1477.  32.4 *
```

Example of UNIVAR PLOT output
(produced from the setup file on the previous page)
(scale is compressed to fit the page)

```
Univariate Frequency Histogram
Date: OCT 28, 1982  at  16:05:14
ACCIDENTS ON ICY ROADS / MISCELLANEOUS ASPECTS

Histogram Variable 3 : MONTH OF ACCIDENT

Number of Cases = 4553
Number of cases represented by each X = 64.294


Code Label      Freq  Prcnt ++++++++++++++++++++++++++
                            +
   1 January    1592.  35.0 +XXXXXXXXXXXXXXXXXXXXXXXX
                            +
   2 February   1091.  24.0 +XXXXXXXXXXXXXXXX
                            +
   3 March        74.   1.6 +X
                            +
   4 April         5.   0.1 +
                            +
   5 May           1.   0.0 +
                            +
  10 October      14.   0.3 +
                            +
  11 November    299.   6.6 +XXXX
                            +
  12 December   1477.  32.4 +XXXXXXXXXXXXXXXXXXXXXX
                            +
                            ++++++++++++++++++++++++++
```

## ** XREF Command **

Command Form: XREF

Keywords: CODEBOOK, OUTPUT
SKIPWORD, VARIABLE
Modifiers: CREATE, EMPTY,
SINGLE

The XREF command generates a sorted cross reference list of all words contained in the variable names of the active data set dictionary. Each word that appears in the list is referenced to the variable name and number in which it appears, as well as the data set descriptor and an optional codebook number supplied by the CODEBOOK keyword. For the purposes of the command, a word is assumed to be any string of characters delimited by blanks.

Single letter words are not included in the list by default but may be included by specifying the SINGLE modifier. In addition, an optional set of words to be skipped may be supplied with the SKIPWORD keyword. The cross reference is generated for the entire dictionary unless a first and/or last variable number for the list is supplied by the VARIABLE keyword.

The cross reference listing is printed in the default print file "-PRINT" unless an alternate file is specified by the OUTPUT keyword. If the EMPTY modifier is specified, the output file will be emptied before use.

### ** XREF Keywords and Modifiers **

CREATE If the file specified by the OUTPUT keyword does not exist, it is created with a size of 10 pages. The modifier is ignored if the file already exists, or if the file specified is not a simple file.
DEFAULT: No attempt is made to create the specified file.

CODEBOOK=<A codebook number> EXAMPLE: CODE=82-7
Specifies the identification number for the codebook documenting the active data set.
DEFAULT: No codebook number is used.

EMPTY Causes the file specified by the OUTPUT keyword to be emptied. The modifier is ignored if the file specified is not a simple file.
DEFAULT: The output file is not emptied.

OUTPUT=<a file or device name> EXAMPLE: OUT=FARSXREF
Specifies the output file or device where the cross reference list will be written.

DEFAULT: The list will be written in the default file "-PRINT."

SINGLE Single letter words will be included in the list.
DEFAULT: Single letter words are not included in the output list.

SKIPWORD=<A word list> EXAMPLE: SKIP=ACC,VEH,OR
A list of up to twenty words, separated by commas, that are not to be included in the cross reference may be supplied.
DEFAULT: All words are cross referenced

VARIABLE=<first,last variable> EXAMPLE: VAR=,256
Specifies the first and/or last variables numbers to be used in the listing, separated by a comma. If one number is specified, it is assumed to be the first and the last variable is set to be the last variable in the dictionary. If the list begins with a comma followed by a number, the number is assumed to be the last and the first variable is set to be the first variable in the dictionary.
DEFAULT: A cross reference listing for all variables is generated.

Example of the XREF Command
(* = user supplied input)

```
  @Command
* ?XREF CODE=82-5 VAR=1,5 SKIP=OF
  @Dictionary records read = 5
  @Cross reference records written = 10

  @Command
* ?PRINT
  @FILE=-PRINT

                    Dictionary Cross Reference

              Variable Name     Number  Key        Code
              -------------     ------  ---        ----
   ACCIDENT   MONTH OF ACCIDENT      5  NASS80AC   82-5
   CASE       CASE NUMBER            2  NASS80AC   82-5
   MONTH      MONTH OF ACCIDENT      5  NASS80AC   82-5
   NUMBER     CASE NUMBER            2  NASS80AC   82-5
   NUMBER     PSU NUMBER             1  NASS80AC   82-5
   NUMBER     RECORD NUMBER          3  NASS80AC   82-5
   NUMBER     VERSION NUMBER         4  NASS80AC   82-5
   PSU        PSU NUMBER             1  NASS80AC   82-5
   RECORD     RECORD NUMBER          3  NASS80AC   82-5
   VERSION    VERSION NUMBER         4  NASS80AC   82-5

  @Command
  ?
```

# Appendix
# GENERATING LABEL FILES

While most system data sets have label files that are created for them as a part of the construction process, there are many situations where user-generated labels are needed. For instance:

1) If a system data set does not have labels, a file may be created for only the variables that are needed.

2) Different labels may be desired than those supplied with the system data set.

3) Private data sets that have not been generated from system data sets may need entirely new label files.

The management of label files is handled by the program HSRI:LABGEN. Before using this program, a permanent label file should be created with the MTS $CREATE command.

Executing the Label Management Program
(* = User supplied input)

```
* #$CREATE SPECIAL.LAB
  #File "SPECIAL.LAB" has been created.
* #$RUN HSRI:LABGEN
  #Execution begins


   Label File Maintenance Program
   Transportation Research Institute
   15:21:54
   SEP 17, 1982

   Enter Filename
* ?SPECIAL.LAB

   Command
  ?
```

There are eight commands that may be entered to control the operation of the program.

Summary of LABGEN commands
(Abbreviations Underlined)

| | |
|---|---|
| DICT | - To enter variable names for the ADAAS codes command (optional) |
| ELIMINATE | - To delete labels for a variable |
| FILE | - To specify a new label file |
| LIST | - To list labels for variables |
| MTS | - To return to MTS |
| SAME | - To set the labels of one variable equal to those of another |
| STOP | - To terminate program execution |
| VARIABLE | - To enter labels for a variable |

Each of the commands is described below together with an explanation of the applicable keywords and modifiers.

DICT PN=<PD name> VO=<volume name> C=<file name>
This command will transfer variable names from a Type 1 dictionary that is located in the file name specified. If the dictionary is on tape, then the pseudo-device name and volume serial name must be specified by the PN and VO keywords.

ELIMINATE <number>
All labels for the specified variable number are deleted. If two variables have identical labels as a result of the SAME command, the labels of the second variable are not deleted by the ELIMINATE command.

FILE <label file name>
The file currently being used as the label file is released and the file specified becomes the active label file.

LIST <number 1> <number 2>
A listing of all labels for variable number 1 through variable number 2 is produced. If only one variable number is given, then the labels for this variable are listed. If no variable numbers are given, then the labels for all variables are listed.

MTS <MTS command>
A return to MTS is made and the specified MTS command is executed. If no command is given, a restartable return to MTS is made. The program may be re-entered by the command $RESTART. A listing of all labels may be routed to a file (or to *PRINT*) for documentation by the procedure:

```
MTS $RESTART SPRINT=<file name>
LIST
MTS $RESTART SPRINT=*MSOURCE*
```

SAME <number 2> AS <number 1>
The labels associated with variable number 1 are assigned also to variable number 2.

STOP
Execution of the program is terminated and the active label file is released.

VARIABLE <number> LEN=<length> MAX=<max code value>
This command initializes the entry of code value labels for the variable specified. The maximum length of the labels is given by LEN and must be in the range one to 16 (i.e., 0<LEN<17 ). If LEN is not specified, a default length of 16 is assumed. The maximum code value for the variable is given by MAX. If MAX is not specified, a default value of 99 is assumed. If a variable has been initialized by a prior use of the VARIABLE command, the LEN and MAX parameters may be omitted. This permits the entry of new labels for a variable, or for the replacement of existing labels.

After the VARIABLE command has been issued, the user will be prompted for code values and their corresponding labels by the prompt:

```
Enter Code,Label
```

The entry of labels is terminated by entering $ENDFILE, or CONTROL-C. A code value and its corresponding label may be deleted by entering a right parenthesis ")" in place of the label.

An example of the use of the LABGEN program to enter and list labels for a single variable is shown below.

Example of code value label entry
(* = user supplied input)

```
   Command
* ?VAR 14 LEN=3 MAX=9
   Enter code,label
* ?1,NO
* ?2,YES
* ?3,UNK
* ?$ENDFILE

   Command
* ?LIST

   VARIABLE NO.    14    LEN=   3    MAX=        9
         1. NO
         2. YES
         3. UNK

   Command
* ?VAR 14
   Enter code,label
* ?3,)
* ?9,UNK
* ?$ENDFILE

   Command
* ?MTS $RESTART SPRINT=-TEMP
  #$RESTART SPRINT=-TEMP

   Command
* ?LIST

   Command
* ?STOP
  #
```

In this example, labels for variable 14 are entered into the label file. A maximum label length of three and a maximum code value of nine are specified for this variable. After entering three code values and then listing the contents of the label file for confirmation, the VARIABLE command is used a second time to delete the code value (3) for the Unknown label and to change the value to nine. All labels in the label file are then listed in the file "-TEMP" for documentation.

Once a label file has been prepared by LABGEN, there are two ways that it can be used in ADAAS:

1) If the label file is to be used with a data set that is entered in a private library, then the LABELS keyword may be used with the LIBRARY command to include this file name as part of the data set description. In this case the system will automatically label output in all uses of the data set.

2) If the label file is to be used with a data set that is entered into the system library, then the LABELS keyword may be specified as part of the parameter statement for all data analysis operations with the exception of SUBSET.

Because of their internal structure, label files should be copied (via the MTS $COPY command) with care. If it is necessary to copy the files, the following command should be used.

```
$COPY FILE1(FIRST)@-TRIM FILE2@-TRIM@I
```

Failure to properly copy the file can result in system errors when it is accessed.

# GENERAL INDEX

# KEYWORD/MODIFIER INDEX