UMTRI-83-18

ADAAS

AUTOMATED DATA ACCESS AND ANALYSIS SYSTEM

CODEBOOK PROCESSOR PROGRAM PROGRAM DOCUMENTATION MANUAL

Christopher R. Ford

APRIL 1983

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.
UMTRI-83-18		

4. Title and Subtitle	5. Report Date
ADAAS Automated Data Access and Analysis System Codebook Processor Program Program Documentation Manual	April 1983
	6. Performing Organization Code
7. Author(s) Christopher R. Ford	8. Performing Organization Report No. UMTRI-83-18
9. Performing Organization Name and Address Transportation Research Institute The University of Michigan 2901 Baxter Rd. Ann Arbor, MI 48109	10. Work Unit No. (TRAIS)
	11. Contract or Grant No. 1133
	13. Type of Report and Period Covered
12. Sponsoring Agency Name and Address Motor Vehicle Manufacturers Association 320 New Center Building Detroit, MI 48202	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

The University of Michigan Transportation Research Institute (UMTRI) has developed a significant transportation-related data base and an integrated set of computer programs (ADAAS) to easily access each data set.

An important part of the utility of this data system is the set of comprehensive codebooks that document each data set in detail. These codebooks describe the characteristics of each element in the data set, including a full definition of all code values and their frequencies of occurrence.

This manual documents the operation of a computer program to manipulate these codebooks and is an adjunct to the ADAAS program documentation manual.

17. Key Words	18. Distribution Statement

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	58	

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

Report Number UMTRI-83-18

A D A A S

Automated Data Access and Analysis System

Codebook Processor Program
Program Documentation Manual

by

Christopher R. Ford

April 1983

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

ACKNOWLEDGEMENTS

The Transportation Research Institute (UMTRI) at the University of Michigan has been involved for more than a decade in the collection, reformatting, and dissemination of transportation-related data with a particular emphasis on motor-vehicle traffic accident data. As a result, an integrated set of computer programs (ADAAS) and a significant data base has been developed. These on-going efforts at the Institute are under the direction of Dr. John A. Green.

Primary funding for the development and maintenance of ADAAS and its associated programs, as well as the extensive library of data sets that comprise the UMTRI Data Center system has come from the Motor Vehicle Manufacturer's Association. The continuing support of that agency is gratefully acknowledged.

TABLE OF CONTENTS

INTRODUCTION

COMMAND DESCRIPTIONS

APPENDICES

INDEX

INTRODUCTION

The Codebook Processor is an interactive utility program that provides for the management of codebook card files, the generation of printable codebooks from these files, and the creation of dictionaries. The program is invoked with the command:

```
$RUN HSRI:CODEBOOK PAR=codecardfile
```

where "codecardfile" is the name of the codebook card file to be worked on. The program will prompt for this name if it is omitted.

Program Operation

The codebook program uses a right bracket ("]") as a prefix character when it is waiting for user input. Command statements may be entered at this point. The commands may be those unique to the codebook program or any valid MTS editor command (see MTS Volume 18, "The MTS File Editor", for a complete description of the editor). In addition, input beginning with "$" is treated as an MTS system command.

Eight unique commands are recognized by the program. The purpose of each command is summarized in the table on the next page. (Commands not recognized by the program are processed as if they were MTS editor commands.) The operation of each command is fully documented on the pages that follow.

Several topics of special interest are covered in appendices at the end of this documentation. These topics include descriptions of the codebook card "types" that make up a codebook card file, a more detailed description of "Tcards", the most important card type, and, finally, a discussion of "frequency files" and how they can be produced with the program HSRI:FREQ.

Command Statements

Each codebook command is described by a "prototype" that has the form:

```
command  lpar  options
```

The designation "command" represents one of the recognized codebook commands. Each command has an acceptable minimum abbreviation indicated by an underlined

Codebook Command Summary

Command	Purpose
ALIGN	To align Tcards on matching file line numbers.
CHECK	To check a codebook card file for format errors.
DICT	To generate dictionaries and produce summary listings of variable elements.
GENERATE	To generate a codebook from the codebook cards.
LABELSET	To produce label setup lines to be used as input to HSRI:LABGEN.
SMOOTH	To sequentially renumber, or "smooth", Tcard variable numbers.
TCONVERT	To perform Tcard conversions.
VREGION	To define "variable" editor regions.

portion of each one's prototype.

"Lpar" is the line-number parameter that specifies the portion of the file to be acted upon by the command. The parameter may consist of a single line number, two line numbers forming a line-number "range", or any combination of either form. The single line number may be either an explicit number or any one of the special editor symbols "*", "*F", or "*L". The range may consist of any two single line numbers or either of the two predefined editor regions "/F" or "/T". In addition, the regions generated by the codebook VREGION command are also recognized. One restriction on line-number ranges is that they cannot be in descending order (i.e., backwards in the file is not allowed). If omitted, the current line is taken as the line-number parameter.

"Options" consists of a collection of keywords and modifiers unique to each codebook command. These options control various aspects of each command's operation such as where output should be written or how the output should be formatted. Options are described in detail with the commands they operate with. Some of the commands have several options, many others have none. Where allowed, they

may always be omitted, in which case default actions will take place.

Examples of Some Valid Codebook Command Statements

Command	Lpar	Options
ALIGN	/F	
CHECK	1 100	NUMBER=2
DICT	/T	LIST OUT=VEHDICT
generate	0 99	output=-book noshift
LAB	4, 5.001 5.2	VERIFY
smooth	9 12.99,14 *L	
TCONV		PROMPT
vreg		

Operation In Batch

Though the codebook program has been designed primarily to be run interactively at a terminal, it may also be run in batch. The operation of the program in batch mode is identical to its operation in terminal mode, with the exception that any error in batch will cause the program to terminate with a return code of 4.

Implicit Concatenation

Implicit concatenation, the use of "$CONTINUE WITH filename RETURN" as a line in the codebook card file to "tie" it together with another file, is a powerful way to achieve special effects while processing a codebook card file. This would be one way, for example, to place more than 999 lines between any two Tcards that reside on adjacent whole line numbers. With the exception of ALIGN and TCONVERT, all of the codebook commands recognize implicit concatenation.

ALIGN Command

Purpose: To align Tcards on matching file line numbers.

Prototype: ALIGN lpar

Action: The cards in the given line-number range "lpar" are renumbered so that Tcards occur on the integer file line numbers corresponding to their variable numbers. Cards positioned between Tcards will occur on fractional line numbers evenly distributed between the Tcards.

A restriction on the value of "lpar" is that it cannot be a multiple line range (for example, "1 4, 6 8").

Options: None.

Problems: Possible problems include: no initial Tcard found in the given line parameter range, Tcard variable number translation trouble, two variables not in ascending order, more than 999 cards found between two Tcards, or an inconsistent alignment configuration encountered.

More often than not, the most common problem with the ALIGN command will be an "inconsistent alignment configuration." Often, for example, the line where a Tcard needs to be written is already occupied by another card. As a general rule, cards to be aligned need to be at line numbers greater than or equal to the line numbers they currently reside on. In any case, alignment cannot take place when duplicate or non-increasing line numbers would occur. By investigating the alignment accomplished up to the point of any error, the problem can usually be solved.

Comments: Unlike the MTS file editor's RENUMBER command, the ALIGN command does not accept a new starting line number or a renumber increment. Because of this inflexibility, the command may require some extra effort to be useful.

Examples:

```
ALIGN 107 124.999
```

The above example will align the Tcards within the given line number range on whole line numbers corresponding to their variable numbers. Other

cards will be placed on fractional line numbers following each Tcard.

```
RENUMBER
MATCH@A /F :I:
ALTER@A /M :I:T:
ALIGN /F
ALTER@A 0, 100, 200 :T:I:
```

The above example assumes that there are three variable "sections" in the codebook card file (for example, an accident section that includes variables 1 through 49, a vehicle section that includes 101 through 170, and an occupant section that includes 201 through 212). It also assumes that at the beginning of each section is an Icard (or "index" card) and that the first Icard has the number "0" right-justified in columns 2 through 5, the second has "100", and the third has "200". The file is first renumbered to insure that it begins on line 1 and has no fractional line numbers. The Icards are then identified and changed into "dummy" Tcards. The ALIGN command then puts all Tcards in the file on whole line numbers corresponding to each one's variable number, with other cards on fractional line numbers in between. Finally, the dummy Tcards are changed back into Icards. (Note that the variable numbers of the dummy Tcards are placed, right-justified, in columns 2 through 5, not 2 through 6. The ALIGN command, noting that these "Tcards" do not have the Type-5 Tcard format, will expect the variable numbers to occur there.)

CHECK Command

Purpose: To check a codebook card file for format errors.

Prototype: CHECK lpar options

Action: The cards in the given line-number range "lpar" are checked for possible format errors or problems that may occur when the GENERATE or other codebook commands are used. These problems may be serious (an untranslatable field-width value, for instance), moderate (an out-of-sequence Ccard code value that won't interrupt generation but will cause unpredictable results), or minor (the improper use of a page or blank card). A complete list of error messages produced by the CHECK command is provided at the end of this description, along with more detailed explanations of their possible causes.

Options: NOPERCNT

Allows the CHECK command to assume that percentages will not be calculated in the final generation of the codebook. By default, it is assumed that percentages will be calculated.

NUMBER=n

Specifies the number of frequency files that the CHECK command should assume will be used in the final generation of the codebook. The maximum number is 6. The default is 1.

OUTPUT=fdname

Specifies the output file or device where the error messages will be written. If not specified, messages are written on *SINK*.

WIDTH=n

Specifies the frequency insertion width that the CHECK command should assume will be used in the final generation of the codebook. The minimum width value is 6, the maximum is 12. The default width is 6 when a single set of frequencies will be inserted, and 8 with multiple frequency insertions.

XCHAR=n

Specifies the Xerox 9700 character that the CHECK command should assume will be used in the final generation of the codebook. The character value can be 1, 2, 4, or 5. The default character is 1.

Problems: Interrupts caused by non-numeric characters in fields undergoing character to arithmetic conversions are vigorously guarded against as part of the entire checking procedure. They may still rarely occur, however.

Comments: The CHECK command investigates all card types, including Tcards. Because of their special nature, however, a thorough check of Tcards can only be guaranteed with the use of either the DICT or TCONVERT commands.

Four of the codebook card types (H, M, N, and S) have maximum lengths dependent on the Xerox character to be used in the final generation of the codebook, along with possibly the NUMBER of frequency files to be used, the WIDTH of frequency insertions, and whether or not percentages are to be calculated. (If the frequency-insertion field resulting from these option settings exceeds the maximum allowed for the particular Xerox character being used, card checking will not proceed.) See the comments section of the GENERATE command for a discussion of the interaction between these options, as well as the limits of the frequency-insertion field.

Many of the error messages produced by the CHECK command may not point directly to the format problem causing the error. For example, Type-5 Tcards are recognized by the command by the presence of five zero characters ("00000") in columns 76 though 80 of the card. If they happen to be out of place for some reason, an error message may appear saying that the Tcard variable number is invalid. This would be due to the fact that the Tcard was not identified as Type-5 and the placement of the variable number was misinterpreted.

Other format problems may produce compound error messages. For example, a single Tcard may go unrecognized because of a small-case "t" in column one. A multitude of error messages may result from the valid cards that follow.

Examples: CHECK 20 30

The above example checks all the codebook cards between lines 20 and 30 for possible format errors. Any that are found are reported at the terminal, along with the line number of the offending card.

```
CHECK /F XCHAR=4 NOPERC
```

The above example checks all the codebook cards through the entire file. Xerox character "4" will be used in the final generation of the codebook. No percentages will be calculated. The CHECK command takes these assumptions into consideration when checking the lengths of H, M, N, and Scards.

Messages: The following is a complete list of all possible error messages produced by the CHECK command, along with more detailed descriptions of their meanings.

Bcard follows a Pcard and has no effect

Blanks that are encountered immediately after a new codebook page has begun are not printed out.

Ccard exceeds 120 characters

Ccards over this length will be truncated.

Ccard too short

Ccards must be at least 11 characters long, plus the field width of the variable they document.

Ccard without preceding Tcard

By definition, Ccards document the code values of the Tcard they follow.

Code value out of sequence

Ccard code values must be in ascending, numerical order.

E, M, Bcards better choice than Xcard

Xcards were formerly used to insert text in a codebook. E, M, and Bcards have taken over this function.

Ecard exceeds 120 characters

Ecards over this length will be truncated.

Hcard lacking text?

No text was found on this Hcard.

Hcard not preceded by Bcard (or blank)

Except for the first code value heading, every Hcard should have a Bcard before it.

Hcard too long

The maximum length of an Hcard is dependent on the NUMBER of frequency files being accessed, the WIDTH of each frequency insertion field, and the XCHAR character being used.

Hcard too short

Hcards must be at least 7 characters long.

Invalid character in column 6

For C and Kcards, only a blank or an "at" sign ("@") should appear in column 6.

Invalid delimiter after code: FW problem?

After each C or Kcard code value a delimiter is expected. (This delimiter is usually a period ("."), but it can be any character chosen by the user. The delimiter found on the first C or Kcard is determined to be this character.) A valid delimiter was looked for in this instance where it should have occurred but was not found. Field width problems can often be the cause of this problem. For example, if the Tcard that the Ccard documents indicates that the variable's field width is two, Ccard code values such as "1." or "001." would produce this message.

Kcard exceeds 120 characters

Kcards over this length will be truncated.

Kcard too short

Kcards must be at least 11 characters long, plus the field width of the variable they document.

Kcard without preceding Tcard

Like Ccards, Kcards document the code values of the Tcard they follow.

Mcard too long

The maximum length of an Mcard is dependent on the XCHAR character being used.

Ncard before this Tcard won't be used

The notational text of Ncards appears in the codebook only when C, K, or H cards are also encountered. One could force an Ncard to be printed by following it with a single Kcard suppressed with an "at" sign in column 6.

Ncard follows first C, K, or Hcard

Ncards should appear immediately after Tcards. Those encountered after the first C, K, or Hcard will not be used.

Ncard lacking text?

No text was found on this Ncard.

Ncard too long

The maximum length of an Ncard is dependent on the NUMBER of frequency files being accessed, the WIDTH of each frequency insertion field, and the XCHAR character being used.

Ncard too short

Ncards must be at least 7 characters long.

Non-blank found after code value delimiter

At least one blank should appear after a C or Kcard's code value delimiter.

Non-blank found in column 6

Except for Ccards, Kcards, and some Tcards, column 6 in all codebook cards should be blank.

Non-blank found in range specification

A C or Kcard range should have a dash in column 11 and blanks up to the code value delimiter.

Non-blank(s) found between cols 7 and 10

C or Kcards should have blanks between columns 7 and 10.

Non-numeric found in code value field

This Ccard's code value has a non-numeric character appearing in it.

Non-numeric found in field-width field

This Tcard's field width setting is invalid.

Non-numeric found in variable number field

This Tcard's variable number is invalid.

Only E, M, or Bcards allowed within box

Only E, M, or Bcards will be properly formatted within a box.

Open box flows into a Tcard

Another box card is needed to close this open box.

Pcard follows a Pcard and will NOT page

Because the Pcard sets a value within the program, the new page occurs when the next card is encountered. If the next card is a Pcard, it is ignored.

Range must be closed by C or Kcard

A range has C or Kcards before and after it.

Range not preceded by C or Kcard

A range has C or Kcards before and after it.

Scard too long

The maximum length of an Scard is dependent on the XCHAR character being used.

Tcard field width setting invalid

This Tcard's field width setting is less than or equal to zero.

Two ranges in sequence ?

A range has C or Kcards before and after it.

Unrecognizable card type " "

This card will be ignored by the program.

Variable number does not match line number

As a matter of style, Tcards should appear on the integer line number that corresponds to their variable number. The ALIGN command can be used to achieve this effect.

1st Hcard does not need Bcard before it

Every code value heading except the first should be preceded by a Bcard.

DICT Command

Purpose: To generate dictionaries and produce summary listings of variable elements.

Prototype: DICT lpar options

Action: The Tcards in the given line-number range "lpar" are used to generate a data dictionary. This dictionary may be either an OSIRIS type "1" or type "5" dictionary. An optional listing may also be produced that summarizes the elements of all the variables. The user may choose, in fact, to not generate a dictionary at all, in which case only a summary listing will be produced.

The DICT command relies heavily on the formats of Tcards to operate properly. Appendix 2 should be consulted for a complete discussion of Tcards.

Options: FILENO=n

Specifies the file number of the generated dictionary when it is written to tape. The value "0" indicates that the file is to be written at the end of the tape. It will have this value if the option is omitted and the dictionary is being written to tape.

LIST=fdname

Specifies the file or device where the summary listing of variable elements will be written. If not specified and a dictionary is being generated, no listing will be produced. If not specified and no dictionary is being generated, the listing will be produced on *SINK* (i.e., the terminal or printer).

LIST

This is the equivalent of specifying "LIST=*SINK*" (i.e., it produces the summary listing of variable elements on the terminal or printer).

OUTPUT=filename

Specifies the file where the generated dictionary will be written. The file may be either a disk file or a tape file. If it is a tape file, the

name should not be greater than 17 characters. In addition, if it is a tape file, the option must be accompanied by the VOLUME option. If omitted, no dictionary is generated.

PDN=pseudodevice-name

Specifies the pseudodevice name of the tape being used when the generated dictionary is being written to tape. If omitted and the dictionary is being written to tape, the pdn "*HSRI*" is used.

TITLE=string

Specifies a string that will be used as a title for the summary listing of variable elements, along with the time and date. The string must not contain embedded blanks. Strings longer than 24 characters will be truncated. If not specified and a dictionary is being generated, the output file name will be used as the title. If not specified and no dictionary is being generated, the title will consist merely of the time and date.

TYPE1

Causes the generated dictionary to be an OSIRIS type "1" dictionary. This is the default dictionary type.

TYPE5

Causes the generated dictionary to be an OSIRIS type "5" dictionary. If omitted, the dictionary will be a type "1" dictionary.

VOLUME=tape-volume

Specifies the volume label of the tape being used when the generated dictionary is being written to tape. This option is required when a tape is used. If it is omitted, it is assumed that the dictionary is being written to a disk file.

Problems: All Tcard elements are thoroughly checked while being processed to insure that they not only contain valid data (all numeric characters, for example), but also that the values they translate to lie within valid limits (variable numbers between 1 and 9999, for example). Any errors that are encountered will immediately stop command processing.

Comments: As stated above, an actual dictionary need not be generated with the DICT command. If one is not generated, only a summary variable listing is produced. This feature is useful if one only wishes to investigate the attributes of any of the Tcards in the file more closely.

If a dictionary is to be generated, it may be written either to a disk file or a tape file. If to a tape file, the VOLUME option is required, while the FILENO and PDN options may or may not be used. In addition, the tape itself must be mounted with its "write-enable" ring in prior to the command being issued.

Examples:

```
DICT 1 10
```

The above example produces a summary listing of variables 1 through 10, writing it out on the terminal or printer.

```
DICT /F OUTPUT=NEWDICT TYPE5
```

The above example generates an OSIRIS type "5" dictionary using all the Tcards in the codebook card file and writes it into the disk file "NEWDICT". No summary listing of the variable elements is produced.

```
$MOUNT
C1234A 9TP *HSRI* VOL=CRF WRITE=YES 'HS0001'
$ENDFILE
DICT /F OUT=VEH83DICT VOL=CRF LIST=-TEMP
```

The above example generates an OSIRIS type "1" dictionary using all the Tcards in the codebook card file. The dictionary is written to the tape file "VEH80DICT", situated at the end of the tape ("FILENO=0"). As can be seen, the tape was mounted previously with the pseudodevice name "*HSRI*" and the specification "WRITE=YES". A summary listing of the variable elements processed by this command is written into the temporary file "-TEMP".

GENERATE Command

Purpose: To generate a codebook from the codebook cards.

Prototype: GENERATE lpar options

Action: The cards in the given line-number range "lpar" are used to generate the pages of a formatted, printable codebook. The action of the numerous card types serve to provide variable information, supplemental information, code value documentation, titles, tables of contents, etc., in the final codebook. Appendix 1 should be consulted for a complete description of the action of each of the card types.

For each Ccard, frequency of occurrences from up to six different sources are inserted beside the code value in the codebook, along with an optional percentage. The frequencies themselves are produced by other computer programs and stored in separate disk files (see Appendix 3). Depending on the value of the NUMBER option (see below), the user will be prompted for however many frequency files are required. The file name of each should be entered when requested without a line-number range. A null response ("return" with nothing else) may be entered if no frequency file is to be used. Along with each file name, a character string "title" may also be entered to be used as a heading for that file's particular column of frequencies. This title can be separated from the file name by a comma, a blank, or both. Strings longer than one less than the value of the WIDTH option will be truncated. Lower case letters are not converted to upper case. A default string "FREQ" will be used if no title is entered.

Command processing begins after the frequency file names have been entered by the user. To allow the progress of the generation to be monitored, the number of every page divisible by 10 is printed at the terminal when encountered, along with the final page number.

Upon completion of command processing, the generated codebook may be used as input to the MTS program *PAGEPR for printing on the Xerox 9700 page printer.

Options: ERROR=undefined.value.string

Specifies the text to be used for the labels of any code value that occurs in a frequency file but is found to have no associated code card. Embedded blanks are not allowed. Lower case letters will be converted to upper case. Strings longer than 30 characters will be truncated. The default value is a blank string.

NOERROR

Suppresses the reporting of insertion errors (i.e., the occurrence of code values in frequency files with no matching code cards). By default, errors are reported with a generated code value in the codebook.

NOPERCNT

Suppresses the calculation and insertion of frequency percentages. By default, percentages are calculated.

NOSHIFT

Disables page-shifting for the codebook. By default, codebooks are shifted for subsequent back-to-back printing of the pages.

NUMBER=n

Specifies the number of files whose frequencies will be inserted in the codebook. The maximum number is 6. The default is 1.

OUTPUT=fdname

Specifies the output file or device where the generated codebook will be written. If not specified, output is written on *SINK* in the TN format.

PAGE=n

Specifies the beginning page number of the codebook. The specification "PAGE=*" will cause the beginning page number to be one more than the last generated codebook's final page number. This is useful when a codebook is being built in pieces with more than one use of the GENERATE command. Note, however, that this use of the option may not work if the last codebook command issued was not a GENERATE command. The default

beginning page number is 1.

TN

Causes the codebook to be formatted without Xerox 9700 control lines and with dashes, rather than horizontal bars, in certain structures. By default, codebooks are formatted with these special 9700 features when the OUTPUT option is being used.

WIDTH=n

Specifies the character width of each frequency insertion field. The minimum width value is 6, the maximum is 12. The default width is 6 when a single set of frequencies is being inserted, and 8 with multiple frequency insertions.

XCHAR=n

Specifies the Xerox 9700 character that the format of the generated codebook will be built around. Xerox character 1 has a pitch of 12. Xerox characters 2 and 4 have pitches of 10. Xerox character 5 has a pitch of 14 and can include over 20 more lines per page than the other characters. The default character is 1.

Problems: Character to binary translations of the Tcard variable number, field width, or number of responses will all interrupt command processing if non-numeric characters are encountered. Command processing will also stop if any Ccard code value translation trouble occurs. These and many other problems may be detected beforehand with the CHECK command. Errors resulting from the internal format of frequency files are less easily anticipated. The user should refer to Appendix 3 if any occur.

Comments: The amount of space available to the user for frequency insertions is a function of the NUMBER of frequency files being processed, the WIDTH of each insertion field, and whether or not percentages are being calculated. A formula may be used to express this size:

SIZE = NUMBER * (WIDTH + PERCENTFACTOR)

where PERCENTFACTOR is 6 if percentages are to be calculated, and 0 if not. For an XCHAR value of 1, the value of SIZE cannot exceed 36. For an XCHAR value of 2 or 4, SIZE cannot exceed 30.

For an XCHAR value of 5, SIZE cannot be greater than 42.

Examples:

```
GENERATE /F OUTPUT=-TEMP
1982FREQ
STOP
$RUN *PAGEPR SCARDS=-TEMP
```

The above example generates a printable codebook using all the codebook cards in the file. Frequencies are taken from the file "1982FREQ". After the codebook program is stopped, the formatted codebook is printed on the Xerox 9700 with the *PAGEPR program.

```
GEN *F 299 NOPERC WIDTH=8 OUT=-BOOK NUMBER=3
TXS81FREQ 5%SAMP
TXT81FREQ TRUCK
TXF81FREQ FATAL
GEN 300 *L NOPERC WIDTH=8 OUT=-BOOK(*L+1) P=*
TXP81FREQ PED/CYC
STOP
$RUN *PAGEPR SCARDS=-BOOK
```

The above example generates a single codebook with two uses of the GENERATE command. Both uses suppress insertion of frequency percentages alongside the code value frequencies, and both set the frequency columns to a width of 8. The first use inserts frequencies from three files: "TXS81FREQ", "TXT81FREQ", and "TXF81FREQ". Frequency titles are provided for each. The second use inserts frequencies from only one file, "TXP81FREQ", but also provides a title. Notice, too, how the second use outputs to the end of the temporary file "-BOOK" with "(*L+1)" and causes the beginning page number to be one more than the final page of the first use with the specification "P=*". After the program is stopped, the formatted codebook is printed on the Xerox 9700 with the *PAGEPR program.

LABELSET Command

Purpose: To produce label setup lines to be used as input to HSRI:LABGEN.

Prototype: LABELSET lpar options

Action: The Tcards and Ccards in the given line-number range "lpar" are transformed into label "setup" lines. These setup lines can then be input into the UMTRI label-generation program HSRI:LABGEN to produce an ADAAS label file.

Options: OUTPUT=fdname

Specifies the output file or device where the setup lines will be written. By default, setup lines (if any) are written on *SINK*.

STRIP

Causes the setup lines to be generated without regard to labels that exceed 16 characters, in effect, "stripping" the codebook card file. By default, labels are verified while they are generated.

VERIFY

Causes the lengths of all Ccard labels to be verified that they do not exceed 16 characters without generating any actual setup lines. By default, labels are generated while they are verified.

Problems: Besides verifying that Ccard labels do not exceed 16 characters, the command will also verify that all Ccard code values are in sequential numeric order. Certain errors will interrupt command processing, however. For example, if a Tcard field-width setting contains any non-numeric characters, a conversion error will occur. In addition, if more than 4096 labels are encountered for a single variable, processing will also be interrupted. One should note, however, that two potential problems are not checked for: non-numeric characters in the Tcard variable number, and non-numeric characters in the Ccard code value. The CHECK command can be used to locate these errors.

Comments: An appendix to the ADAAS documentation entitled "Generating Label Files" should be consulted for a complete description of the HSRI:LABGEN program.

Examples:

```
LABELSET /F OUTPUT=-LABTEMP
STOP
$RUN HSRI:LABGEN
MYLABELFILE
$CONTINUE WITH -LABTEMP
STOP
```

The above example generates label setup lines for the entire file and places them in the temporary file "-LABTEMP". After running HSRI:LABGEN, a "$CONTINUE WITH -LABTEMP" within the program causes the setup lines to be input into the LABGEN program as commands, thus generating ADAAS-compatible labels that are stored in the file "MYLABELFILE".

```
LAB 30 39.999 VERIFY
```

The above example verifies the Ccard label lengths of variables 30 through 39 without generating any label setup lines.

SMOOTH Command

Purpose: To sequentially renumber, or "smooth", Tcard variable numbers.

Prototype: SMOOTH lpar

Action: The Tcards in the given line-number range "lpar" are renumbered in ascending, sequential order, beginning with the first Tcard encountered. In addition, for reference purposes, each new variable number is placed, right-justified, in columns 2 through 5 of the non-Tcards that immediately follow each Tcard.

Options: None.

Problems: Two problems may occur: no initial Tcard found within the given line parameter range or, if one is found, a non-numeric character encountered while translating its variable number.

Comments: One should note that this command does not affect the file line numbers that Tcards reside on. Rather, it only affects the variable number of each Tcard (occurring in columns 3 through 6 on Type-5 Tcards, or 2 through 5 on SRS Tcards). To renumber Tcard file line numbers, the ALIGN command may be used.

Examples:

```
SMOOTH /F
```

The above example will sequentially renumber the Tcard variable numbers through the entire file using the variable number of the first Tcard as the initial number. If there were, for example, 40 Tcards and the first Tcard documented variable 10, the resulting Tcards would range from variable 10 to 49, inclusive.

```
SMOOTH 1 99
SMOOTH 101 199
SMOOTH 201 299
```

The above example assumes that there are three variable "sections" in the codebook card file (for example, an accident, vehicle, and occupant section). It also assumes that the first accident Tcard is variable 1 at file line number 1, the first vehicle Tcard is variable 101 at line 101, and the first occupant variable is

variable 201 at line 201. If several of the variables have had their relative positions changed by additions, deletions, or simple exchanges, the above commands will insure that the file's Tcards are once more numbered contiguously within each section.

TCONVERT Command

Purpose: To perform Tcard conversions.

Prototype: TCONVERT lpar options

Action: The Tcards in the given line-number range "lpar" are converted into another Tcard format according to the option(s) specified.

Tcards may be converted from an SRS format to a Type-5 format, from Type-5 to SRS, and from a "free format" to either SRS or Type-5. Both the SRS and Type-5 formats are described in detail in Appendix 2. Free-format Tcards are described below.

Because "lpar" applies only to the codebook card file as input, if either the INPUT, PROMPT, or SOURCE options are utilized, "lpar" is effectively ignored.

Options: INPUT=fdname

Specifies the file or device where Tcards to be converted are read from. If not specified and "prompting" is not taking place (see below), the codebook card file is used for Tcard input.

OUTPUT=filename

Specifies the file where converted Tcards will be written. If not specified, the codebook card file is used for Tcard output.

PROMPT

Causes the program to prompt the user for "free-format" Tcard input. As many free-format Tcards as desired may be entered. Prompting will continue until either a "$ENDFILE" is issued or an input error occurs. If omitted and the INPUT option is not used, the codebook card file is used for Tcard input.

SOURCE

This is a synonym for the PROMPT option. Using either one is the equivalent of specifying "INPUT=*SOURCE*".

SRS

Causes the Tcards to be converted using the SRS format. The specification "TYPE1" is a synonym for this option. The SRS format is the default conversion format.

TYPE5

Causes the Tcards to be converted using the Type-5 format. If omitted, the SRS format is used.

Problems: All Tcard elements are thoroughly checked while being processed to insure that they not only contain valid data (all numeric characters, for example), but also that the values they translate to lie within valid limits (variable numbers between 1 and 9999, for example). Any errors that are encountered will immediately stop command processing.

Another less obvious problem may occur if either the INPUT or OUTPUT option is used, or if prompting is taking place. When these options are used, converted Tcards are written out on file line numbers corresponding to their variable numbers (the Tcard for variable 3 on line 3, for example). Before any new Tcard is written out, the line where it will be written is first checked to insure that it is vacant. If it is not, the card that resides there is then checked to insure that it is indeed a Tcard and that it describes a variable with the same number as the Tcard that is about to take its place. If either of these conditions is not met, command processing is interrupted. This checking provides a measure of security against accidental damage of the output file.

Comments: As stated above, when either the INPUT or OUTPUT options is used, or when prompting is taking place, converted Tcards are written out on file line numbers corresponding to their variable numbers. Otherwise, the default input and output file is the codebook card file itself, and converted Tcards are written back out on the same lines where they were found.

Examples: TCONVERT /F TYPE5

The above example converts all the Tcards in the file into Type-5 Tcards. The Tcards to be converted can be in any format, including the

free-format Tcard described below. Free-format cards must begin with a "T", however, in order to be recognized. Any Type-5 Tcards that exist in the file prior to the command being issued will be left in the Type-5 format, effectively only being checked by the command.

```
TCO PROMPT
1,ACCIDENT NUMBER,6,C, ,
2,DAY OF WEEK,,,9
3,ACCIDENT TYPE,2,A
$ENDFILE
```

The above example causes the program to prompt for free-format Tcards. Three are entered, along with "$ENDFILE" to stop the prompting. The resulting Tcards are in the SRS format. They are displayed in the example that shows typical codebook card types in Appendix 1.

Free-Format Tcards

Free-format Tcards have been designed to facilitate the entry of finished Tcards. They may either be entered by hand at the prompt when using the PROMPT or SOURCE options, or placed in the input file with a "T" in column 1. When being entered by hand, the program produces the prompt:

Var,Name,Width,Type,MD1,MD2,Resp,Dec,Location

These nine elements, or items, may be entered on a single line in the order shown, separated from one another by commas. Except in the variable name specification, blanks should be avoided.

Every element has a default value that will be assigned to it when no value has been entered. Default values are assigned by simply omitting the item to be defaulted. For example, "2,DAY OF WEEK,,,9" will cause variable 2 to have a field width of 1 and a character numeric storage type. Default values are also assigned when elements remain after the end of the Tcard line. For example, "2,DAY OF WEEK,,,9" will have no second missing data code, will be a single response variable with no implied decimal places, and will have no specified starting location.

Normally, the default value for the missing data elements is "none", or no missing data code. Another default technique has been provided, however. For these elements, a single blank will cause the missing data code to be a field of nines for that variable. For example, "1,ACCIDENT NUMBER,6,C, ," assigns the value "999999" as the first missing data code for the variable.

Any value entered in the ninth position is usually interpreted as an absolute starting column location for the variable. If the value begins with a star ("*"), however, the value will be taken to be an offset. See Appendix 2 for a further discussion of locations and offsets.

The default values and limits for all the free-format Tcard elements are provided in the table below.

Free-Format Tcard Elements

Item	Default Value	Value Limits
Variable number	1 for first Tcard -- one more than last for subsequent Tcards	1 to 9999
Variable name	"VARIABLE n" (where "n" is variable number)	24 characters (no commas)
Field width (FW)	1	C: 1 to 15 A: 1 to 999
Storage type	"C" (character numeric)	C, A, I, F, P, Z, or B
Missing data 1	No missing data 1 (if blank, "nines")	-999999 to 9999999 (if FW >= 7)
Missing data 2	No missing data 2 (if blank, "nines")	-999999 to 9999999 (if FW >= 7)
Response number	1 (single response)	0 to 99
Decimal places	0 (no decimal places)	0 to 9
Location	No starting location	1 to 32767

VREGION Command

Purpose: To define "variable" editor regions.

Prototype: VREGION lpar

Action: Every whole number in the given line-number range "lpar" is used to define an editor region that corresponds to all of a variable's codebook cards. For example, for a variable 130, the region is defined as ranging from file line number 130 to 130.999 and specified by the designator "/130". A single use of the command will define up to a maximum of 100 regions.

Options: None.

Problems: None.

Comments: Use of this command assumes that Tcards reside on whole line numbers corresponding to their variable numbers (see the ALIGN command).

Examples:

```
VRE
PRINT /3
```

The above example defines a single variable region, based on the value of the current line. The region is then printed out (assuming that the current line happened to be something like 3.4, or, for that matter, any line number between 3 and 3.999).

```
VREGION 20 22
MOVE /22, /21, /20 TO 20
SMOOTH 19 23
ALIGN 19 23
```

The above example defines regions for variables 20 through 22. The subsequent editor MOVE command serves to invert the order of the three variables. The SMOOTH command is then used to renumber the variable numbers, while the ALIGN command is used to insure that the Tcards once again reside on whole line numbers corresponding to their variable numbers.

APPENDIX 1

Codebook Card Types

UMTRI codebooks are generated from a collection of what have traditionally been called codebook "cards". These cards actually correspond to lines in a disk file. Currently there are 15 recognizable card types that provide, among other things, variable documentation, code value descriptions, titles, tables of contents, etc., in the final printable codebook. This appendix describes each of the card types.

The type of the card, usually an upper-case letter, always appears in card column 1. Columns 2 through 5 for all cards usually contain the variable number of the preceding Tcard, though this is not a requirement. For most cards, column 6 should be blank. The remainder of the card format varies from card to card. No single card, however, should be longer than 120 characters.

T Tcards describe the characteristics of the data set variables. These "variable descriptors" document, variables. These "variable descriptors" document, among other things, the number and name of each variable, the field widths, and the missing data codes. As a matter of style, each Tcard should appear on the file line number corresponding to its variable number. Currently, there are two types of Tcard formats: an SRS format that comes from the early days of the HSRI Statistical Research System, and a Type-5 format that closely follows the current format of OSIRIS variable descriptor records. These formats are both documented in Appendix 2. Because the format of any Tcard is quite rigid, Tcards are best generated by the TCONVERT command.

C Ccards document the variable's code values. Column 6 is usually blank, except for a print suppression indicator (described below). Columns 7 through 10 must be blank. The numeric code value itself must begin in column 11, with leading zeros inserted to the correct field width. A delimiter immediately follows the code value. (Traditionally, this delimiter has been a period ("."), but it may, in fact, be any character.) The card may end here, or it may continue with text. If it does continue, the delimiter should be followed by at least one blank. In the columns that remain, a label documenting the code value may be entered. Text beyond column 120 is ignored. Text should not be continued on subsequent cards.

A range of values may be accommodated with a "range" card, a Ccard with a dash in column 11, blanks in the remainder of the value field, a delimiter, blank, and any label. The range card is placed between two ascending Ccards.

The printing of values for which no corresponding frequencies are found may be suppressed by placing an "at" sign ("@") in column 6. The printing of the range card may also be suppressed. In contrast, values for which there are frequencies but no corresponding code cards are automatically generated and printed in the codebook (unless the GENERATE command's NOERROR option is used). Up to 100 contiguous, non-matched values can be processed in this manner.

K Kcards are identical in format to Ccards and are handled in much the same way. Frequency insertion, however, is not attempted for Kcards. Because of this fact, a non-numeric code value may appear on the card. Kcards are useful for documenting the values of an alphabetic-type variable, as well as for adding supplementary information to Ccards such as a parallel set of definitions for the same values.

H Hcards are used as headings for C or Kcards. The heading text should begin in column 9, though text beginning anywhere from column 7 or beyond will be accommodated. The heading will be formatted to "hang over" the Ccards that follow by two spaces. The text itself may extend to column 120 on the Hcard, but only the portion able to fit between two spaces before the start of values and the right margin of the codebook will be used. This length is variable and depends on the GENERATE command's NUMBER and WIDTH setting, as well as the XCHAR character being used.

N Ncards contribute "notational" information to the variable Tcards they follow. The text of the (optional) note card can be used to spell out the name of the variable more fully, perhaps, or to otherwise amplify the significance of the variable in some way. The text may begin in column 7 or anywhere beyond. The text will be used on the first "frequency heading" placed above the beginning of C or Kcards for the variable. For this reason, if no C or Kcards follow the variable's Tcard, no frequency heading is printed. Though the frequency heading is distinct from the heading produced by the Hcard, the Ncard text is formatted in an identical manner. Thus, the same text-length limitations apply.

E Ecards allow elaborations, or explanations, to be included in the codebook. Text for Ecards may begin in

column 7 or anywhere beyond, and may extend to column 120. The GENERATE command will automatically format the text of contiguous Ecards so that at least 5 spaces occur between the text and the left and right margins of the finished codebook. Indented paragraphs may be achieved with the use of tildes ("~").

S Scards provide titles for the codebook. The title text may begin in column 7 or anywhere beyond. It will be centered automatically by the program and placed at the top of each codebook page. Scards usually occur in pairs, the first card providing what could be called a global title, with the second card providing a subtitle. Scards can occur anywhere in the codebook card file and take effect on the next page after being encountered. The maximum length of the card text is variable and depends on the XCHAR character being used.

M Mcards, or "middle" cards, are single lines of text that are automatically centered in the finished codebook. The text may begin in column 7 or anywhere beyond. The maximum length of the card text is variable and depends on the XCHAR character being used.

B Bcards generate a blank line in the codebook.

I Icards cause an "index dump" to take place. All summary information gathered for all variables (name, field width, etc.) since the last index-dump (if any) is printed out. With this card, a separate accident, vehicle, and occupant index can be generated. When brought to the front of the codebook by the user, these separate indices become a codebook table of contents. The Icard will also force the next codebook section to begin printing on a new page (odd-numbered if shifting is in effect). Another important action of the Icard is the adding up of a new frequency total for percentage calculation. Because an end-of-file produces the same effect as an Icard, an Icard need not be the last card in the file.

A Acards, or "add" cards, cause a new frequency total to be calculated for percentage calculation. The Icard performs the same function, but the Acard does it without generating an index. These cards should be placed between data "levels" (i.e., between an accident and vehicle section or whenever the value of 'N' changes). They should be used whenever percentages are being included in the finished codebook and the index-dumping of Icards is not desired.

P Pcards cause the information that follows to begin on a new page.

U Ucards, or "upper" cards, cause the information that follows to begin on a new page if it would otherwise occur on the bottom half of the printed page. If it occurs on the top half of the page, no new page is generated.

The "box" card formats a box in the finished codebook. These cards occur in pairs. When first encountered, the card turns on the box format and generates the top of the box. The next box card encountered generates the box bottom and turns off the box format. Any number and combination of E, M, or Bcards may appear between the two box cards. The text generated by them will be printed within the box in the finished codebook.

X Xcards are supported for compatibility with older codebook card files. Any text beginning in column 7 is printed in the finished codebook exactly as it appears. E, M, or Bcards can usually do what the Xcard was formerly called upon to do in a more powerful way. They are therefore better choices than the Xcard.

Examples of Some Typical Codebook Cards
(with column rulers above and below)

```
         1         2         3         4         5         6
1234567890123456789012345678901234567890123456789012345678901234567890

I    0
S    0 NATIONAL ACCIDENT STUDY
S    0 Accident - 1980
#    0
M    0 Accident Variables
B    0
E    0 ~~~~~Variables 1 through 50 describe
E    0 accident level information.  They are included in the
E    0 Accident, Vehicle, and Occupant data sets.
E    0 The Accident data set contains these variables
E    0 and no others.
#    0
T    1 ACCIDENT NUMBER          *      6 C    999999
C    1     000000.
C    1     -      . Accident report number
C    1     999999.
T    2 DAY OF WEEK              *      1 C         9
N    2 DAY OF WEEK OF ACCIDENT
C    2     1. Sunday
C    2     2. Monday
C    2     3. Tuesday
C    2     4. Wednesday
C    2     5. Thursday
C    2     6. Friday
C    2     7. Saturday
C    2@    9. Missing data
T    3 ACCIDENT TYPE            *      2 A
H    3   Single Vehicle Accident
K    3     RO. Rollover
K    3     FX. Struck fixed object
B    3
H    3   Multiple Vehicle Accident
K    3     HD. Head on
K    3     SW. Sideswipe

         1         2         3         4         5         6
1234567890123456789012345678901234567890123456789012345678901234567890
```

The output on the following page was produced from the codebook cards above using the GENERATE command. An XCHAR value of "4" was used as an option, and, as can be seen, no frequencies were inserted.

NATIONAL ACCIDENT STUDY
Accident - 1980

Accident Variables

Variables 1 through 50 describe accident level information. They are included in the Accident, Vehicle, and Occupant data sets. The Accident data set contains these variables and no others.

Variable 1 **ACCIDENT NUMBER** MD1: 999999 FW: 6
MD2: None Numeric

FREQ Prcnt ACCIDENT NUMBER

000000.
- . Accident report number
999999.

Variable 2 **DAY OF WEEK** MD1: 9 FW: 1
MD2: None Numeric

FREQ Prcnt DAY OF WEEK OF ACCIDENT

1. Sunday
2. Monday
3. Tuesday
4. Wednesday
5. Thursday
6. Friday
7. Saturday

Variable 3 **ACCIDENT TYPE** MD1: None FW: 2
MD2: None Alpha

FREQ Prcnt ACCIDENT TYPE

Single Vehicle Accident
RO. Rollover
FX. Struck fixed object

Multiple Vehicle Accident
HD. Head on
SW. Sideswipe

APPENDIX 2

Tcard Formats

Tcards form the basis for data set dictionaries in both ADAAS and OSIRIS. In OSIRIS, dictionaries may also contain other codebook cards, or records, that perform functions similar to those performed by UMTRI codebook cards. In fact, codebook card files have no separate existence apart from dictionary files -- they are one and the same. At one time, UMTRI codebook card files and dictionary files were also the same, but, over the years, codebook card files have gradually come to exist independently from dictionary files. And though the codebook Tcard and dictionary Tcard very often have a similar, if not identical, format, some processing is usually necessary to transform the Tcards into a dictionary file.

The Tcards described in this appendix are "codebook" Tcards as opposed to "dictionary" Tcards. Currently there are two distinct formats recognized by the codebook program.

Type-5 Tcards closely follow the format of the OSIRIS variable descriptor records that make up type "5" dictionaries. These records, as well as structured file description records and codebook records, are documented in the OSIRIS manual[1] on pages 349 through 356.

SRS Tcards were originally used at UMTRI in conjunction with the Statistical Research System, the predecessor of ADAAS. These Tcards, as well as several early (and outmoded) codebook card types, are fully described in the SRS manual[2] on pages 41 through 47.

In most cases, OSIRIS Type-5 Tcard records may be used without alteration in a codebook card file. Some incompatibilities, however, exist at this time. The GENERATE command, for example, can only recognize character numeric and alphabetic storage types. One should note, too, that the program cannot handle variable numbers greater than 9999 or implied decimal place numbers over 9. One other major difference between the UMTRI Type-5 Tcard and the OSIRIS variable descriptor record is the (optional) use by

---

[1] Survey Research Center, OSIRIS IV User's Manual (Ann Arbor: The University of Michigan Institute for Social Research, October 1982).

[2] David E. Wood and Carole D. Hafner, The Statistical Research System (Ann Arbor: The University of Michigan Highway Safety Research Institute, June 1972).

the UMTRI Tcard of the location field to provide offsets in the dictionary. This use is discussed below.

The SRS Tcard format currently in use is a modified version of an original format used with the Statistical Research System. Some differences should be noted. A blank in the variable's data storage type field, for example, at one time indicated a character numeric variable, while "1" indicated an alphabetic variable. These designations have been changed to "C" and "A", respectively. The codebook program, however, can still recognize the earlier designations. Another major difference, as with the Type-5 Tcard, is the use of the location field to optionally specify offsets. Many older SRS Tcards have "01010" in this field, and the codebook program takes them into consideration.

Both UMTRI Type-5 and SRS Tcards make special use of the field containing the variable's starting column location. If the location field begins with a star ("*"), the number that follows is assumed to be an "offset" (i.e, the variable's starting location will be so much more or less than the current location). For example, if one variable had a starting location of 45 and a field width of 10, the specification "*-10" would cause the next variable to begin at the same starting location, 45, rather than at 55 (the case if the offset had been blank). If no star is present and the field is not blank, whatever value appears in the field is taken as the variable's absolute starting location.

Use of one Tcard format over the other is essentially a personal choice. The Type-5 format has the advantage of being nearly compatible with the OSIRIS variable descriptor format. On the other hand, the SRS format, besides having been in use for several years, is much easier to work with. It more closely matches the format of other UMTRI codebook cards, has more clearly recognizable elements, and is never longer than 59 characters, thus avoiding "wrap-around" when the codebook card file is edited on a terminal with a width less than 80 characters.

All of the codebook commands can handle either Tcard format. Most distinguish a Type-5 Tcard from an SRS Tcard by the presence of five zero characters ("00000") in columns 76 through 80 of the card.

The two tables that follow document the formats of the Type-5 and SRS Tcards. Fields not described are usually blank. Several of the Type-5 fields that are described are not significant to the codebook program.

Type-5 Tcard Fields

Columns	Len	Item	Comments
1	1	Card type	"T"
3- 6	4	Variable number	Leading blanks
8-11	4	Reference number	Leading blanks
12-14	3	Group number	" 0"
15-38	24	Variable name	
39	1	Character type C: Char numeric A: Alphabetic	Data storage type
42-46	5	Location * + Displacement * - Displacement * Absolute location	Location or offset Trailing blanks Trailing blanks Trailing blanks Leading blanks
47-49	3	Field width	Leading blanks
51	1	Decimal places	0/blank if none
52-53	2	Response number	1/blank if single
56-62	7	MD code #1	Leading blanks
65-71	7	MD code #2	Leading blanks
76-80	5	Card designator	"00000"

SRS Tcard Fields

Columns	Len	Item	Comments
1	1	Card type	"T"
2- 5	4	Variable number	Leading blanks
7-30	24	Variable name	
32-36	5	Location * + Displacement * - Displacement * Absolute location	Location or offset Trailing blanks Trailing blanks Trailing blanks Leading blanks
37-39	3	Field width	Leading blanks
40	1	Decimal places	0/blank if none
41	1	Character type C: Char numeric A: Alphabetic	Data storage type
43-44	2	Response number	1/blank if single
45-51	7	MD code #1	Leading blanks
52-58	7	MD code #2	Leading blanks

APPENDIX 3

HSRI:FREQ and Frequency Files

Frequencies inserted in finished, printable codebooks by the GENERATE command are stored in separate disk files. Although a knowledge of the structure of these files is sometimes useful, it is never critical to the operation of the codebook program. More important is knowing how to produce them in the first place. A separate program has been written to perform this task. This program, "HSRI:FREQ", is described below. Following this, the structure of the frequency files themselves is covered in detail.

HSRI:FREQ

HSRI:FREQ is a "frequency generation" program that performs basic one-way tabulations on a data file, counting the number of times that certain data elements occur for any of several variables. The program has several options that may be passed to it on the program's run-time parameter line. For example:

```
$RUN HSRI:FREQ PAR=list of options in any order
```

If the parameter line is omitted, the options are read in on SCARDS until either a "$ENDFILE" is encountered or the specification "END" is processed. For example:

```
$RUN HSRI:FREQ
list of options
in any order
END
```

These options set the input data and dictionary files, the output frequency file, specify the variables to be counted, and perform various other miscellaneous functions.

HSRI:FREQ requires both a data file and a dictionary file in order to be run successfully. The dictionary must be an OSIRIS type "1" dictionary. The variables to be accessed must all store their data in "character numeric" mode (i.e., the numbers 0 to 9 as typed in at a terminal).

To summarize the use of the program's options, the DATA, DICT, and OUTPUT options are all required. If either DATA or DICT specify a tape file, the VOLUME option is also required. In addition, some variables must be also be specified, either with the VARS option, the RV option, or both.

The use of the VARS and RV options deserves special comment. The VARS option is at the same time powerful and tolerant in the manner in which it ignores certain variables. Entire sections of a data file can be specified without worrying about variable gaps, invalid variables, etc. Conversely, the VARS option may at first seem somewhat restrictive in not being able to access variables with field widths greater than four. Full frequency counts, however, are usually not inserted in codebooks for these variables. Rather, they are usually represented by a "range" of values in the codebook card file. The RV option was, in fact, designed for these situations and may easily be used to generate frequencies for these variables.

Options: DATA=filename

Specifies the data file to be used for the frequency generation. The file may be either a disk file or a tape file. If it is a disk file and the dictionary file happens to be on tape, the data file name must be prefixed by its CCID, even if the program is running from that CCID. There is no default data file name. It must be specified.

DICT=filename

Specifies the dictionary file to be used for the frequency generation. The file may be either a disk file or a tape file. If it is a disk file and the data file happens to be on tape, the dictionary file name must be prefixed by its CCID, even if the program is running from that CCID. There is no default dictionary file name. It must be specified.

FILL=ZERO/BLANK

Specifies the character to be used as the "fill" character. Though it can usually be disregarded, the fill character can occasionally affect the operation of data conversion, as well as program filtering. By default, the fill character is ZERO ("0").

FILTER=variable:value

Specifies a single variable and non-negative integer value, separated by a colon, that will be used to selectively filter the data file. Only those cases that have the same value for that variable will be counted by the program. The variable itself cannot have a field width greater than eight. If omitted, no filtering takes

place.

OUTPUT=filename

Specifies the disk file where the final frequency information will be written. The specification "WRITE" is a synonym for "OUTPUT". There is no default output file. One must be specified.

PDN=pseudodevice-name

Specifies the pseudodevice name of the tape where either the dictionary file, the data file, or both, reside. If omitted and a tape is involved, the pdn "*HSRI*" is used.

RVn=variable:minvalue-maxvalue

Specifies a "range" variable and two non-negative integer values, the first less than the second. Up to 25 different ranges may be specified for one program run, each identified by different values of "n". Frequencies will be generated for each range for data elements less than or equal to the minimum value, and greater than or equal to the maximum value. In essence, the two values specified by the option serve as "end-points" that either define a range of data elements that will be skipped, or ignored, by the program (i.e., those between the minimum and maximum values), or, alternatively, two sets of data elements that will be counted by the program (i.e., the range from the smallest data element found to the minimum value, and the range from the maximum value to the greatest data element found). Up to 100 different data elements will be counted by the program, above and below the range values. If over 100 data elements are encountered for a given range variable, processing for that variable will cease and its frequencies will not be written into the output file. The variable itself cannot have a field width greater than eight. If no RV option is used, variables must be specified with the VARS option.

VARS=variable-list

Specifies the variables whose data elements will be counted by the program. Up to 10,000 variables may be specified, but only the first 250 character numeric variables in the dictionary with a field width of four or less will have frequencies generated. Many variables will

therefore be ignored. These include: variables not in the dictionary, non-numeric variables, variables with field widths greater than four, and any variable also selected with an RV option. If the VARS option is not used, variables must be specified with the RV option.

VOLUME=tape-volume

Specifies the volume label of the tape where either the dictionary file, the data file, or both, reside. This option is required when a tape is involved. If it is omitted, the dictionary and data files are assumed to reside on disk.

Problems: The number of distinct data elements allowed for a single variable processed by the VARS command is limited to 4096 non-negative values. The limit for range variables has been arbitrarily set at 100 positive and negative values. If either of these limits is exceeded, the frequencies for the offending variable are not written in the output file.

Decimal (character numeric) to binary conversion errors will cause processing to cease for the offending variable. Processing continues, however, for all other variables. The case number and field causing the error are also reported.

Problems may also occur when leading blanks, rather than leading zeros, were used to fill out variable fields in the data file being processed. In these situations, decimal to binary conversion errors may result when the blanks are encountered. To overcome this problem, the fill character may be set to blank with the FILL option. One should understand, however, that this will also effect program filtering and the identification of range values. If, for example, the FILTER option were used and it had been set to value "1" for a variable with a field width of two, the value would be represented as "01" when the fill character was zero, but as " 1" when the fill character was blank. The low and high values of a range variable would be similarly affected. Because both filtering and range selection is based on a value's character representation, rather than its binary representation, in certain situations values may be ignored by the program that should otherwise be considered.

Examples:

```
$RUN HSRI:FREQ
DICT=VEHICLE.DIC  DATA=VEHICLE.DAT  VOL=VEHTAP
VARS=101-199  OUTPUT=FREQVEH  END
```

The above example generates frequencies for all character numeric variables with a field width of four or less between variables 101 and 199 as defined by the dictionary file "VEHICLE.DIC" and found in the data file "VEHICLE.DAT". Both these files are located on a tape with the volume label "VEHTAP" that was mounted previously with the pseudodevice name "*HSRI*". Generated frequencies are written into the disk file "FREQVEH". Notice that the options are passed to the program on lines immediately after the program is run, and that they are terminated with the "END" specification.

```
$RUN HSRI:FREQ  PAR=OUT=FREQVEH -
DA=TEMPDATA  DI=NEWTEMPDICT  VARS=1-3,10-40
```

The above example generates frequencies for all character numeric variables with a field width of four or less between variables 1 and 3, as well as 10 and 40. Both the data file ("TEMPDATA") and dictionary file ("NEWTEMPDICT") are disk files. Generated frequencies are written into the file "FREQVEH". Notice that the run-time parameter line is used to pass options to the program and that, because of its length, it is continued onto a second line with a dash ("-").

```
$RUN HSRI:FREQ
DICT=SFQ8:RESULTDICT  DATA=RESULTDATA
VOL=MYTAPE  PDN=*T*
VARS=1-1000  OUTPUT=FREQS
RV1=5:1-99998  RV2=20:1-30257  RV3=142:0-999
END
```

The above example generates frequencies for the first 250 character numeric variables with a field width of four or less between variables 1 and 1000, excluding variables 5, 20, and 142. The data is found in the file "RESULTDATA" located on a tape with volume "MYTAPE", mounted with the pseudodevice name "*T*". The dictionary, however, is found in the disk file "RESULTDICT" which is explicitly identified with its CCID. Generated frequencies are written into the file "FREQS". In addition to the variables specified with the VARS options, three range variables are selected:

```
RV1=5:1-99998  RV2=20:1-30257  RV3=142:0-999
```

These range variable examples demonstrate several uses of the RV option, illustrated hypothetically here. The first example will cause frequencies to be generated for data elements of 1 or less and 99998 or greater for variable 5. Variable 5's missing data code may be 99999, in which case the missing data rate can then be displayed in the codebook. The second example processes data elements for variable 20 in a similar manner. Variable 20 could describe a date in Julian format, however. In this case, not only could the missing data rate be calculated, but frequencies for all dates past January 1, 1983 would also be generated (if less than 100 elements), thus identifying possible wild codes if no dates should occur there. The third range example again works in a similar manner. Variable 64, however, may have values that range from -99 to 999 (as in a "delta-v" measurement, for example). If the VARS option were used to select this variable, the first negative value encountered would cause a conversion error to occur. To avoid this, the variable can be handled with the RV option, as the data elements it processes can be both positive and negative (though the minimum and maximum range values themselves cannot be less than zero). This specification would be one way, then, to generate "end-point" frequencies for a variable with negative values.

Frequency Files

Frequencies inserted in finished, printable codebooks by the GENERATE command are stored in separate disk files. The structure of these files is covered in detail here.

Frequency information for any particular variable is stored on two lines in the disk file. Both of these lines are keyed to the variable's number. Because of this, a single frequency file cannot hold multiple frequency information for a given variable (e.g., frequencies for variable 1 for injury-only accidents and frequencies for variable 1 for fatal accidents). In these cases, multiple frequency files are required.

The first line for a given variable occurs on the integer line number corresponding to its variable number (for example, variable 4 on line 4). This line contains two pieces of information: the variable number in the first two bytes, and, in the next four bytes, the total number of full-word code values and frequency pairs contained on the second line. These two values are stored in binary.

The line that follows occurs at a fractional increment of "0.001" from the first line (for example, line 4.001 for variable 4). This second line contains the actual frequency information: four-byte full-word code values, each immediately followed by its full-word frequency of occurrence, all stored in binary. Because a single MTS file line is limited to a length of 32,767 bytes, this line can hold no more than 4096 separate code values and frequencies. Only code values that have a frequency greater than zero, however, will normally appear. Code values are arranged on the line in ascending numerical order.

Care should be taken if a frequency file is ever modified by hand with the MTS file editor. As can be observed, precise line numbers and line lengths are both critical to the proper insertion of frequencies by the GENERATE command. Because the information in the file is stored in binary, to make any sense of the lines when using the file editor, the lines must be managed with the hexadecimal modifier (for example, "PRINT@X").

When calculating frequency percentages, the GENERATE command determines the value of 'N' by adding up the frequencies appearing on the second line. This operation only takes place, however, for the first variable processed that has frequencies in the codebook card file, along with the first variable with frequencies after either an Icard or an Acard has been encountered. The importance of Icards and Acards can be seen, then, whenever a frequency file is being processed that has more than one value of 'N' (an accident,

vehicle, and occupant value, for example). The implication, too, is that an incorrect value for percentage calculations will be derived if the frequencies for the first variable processed do not equal the value of 'N' (because they were generated by the RV option of the HSRI:FREQ program, for example). The user should be aware of these potential problems.

INDEX