AD-A042 212 DIRECTORATE OF AEROSPACE STUDIES KIRTLAND AFB N MEX F/G 5/2
FORTRAN SOFTWARE FOR CREATING AND MAINTAINING A LIBRARY CATALOG--ETC(U)
JUL 77 C A FEUCHTER

UNCLASSIFIED DAS-TR-77-2 NL

| OF |
AD A042212

END

DATE FILMED
8-77

DAS-TR-77-2

DAS-TR-77-2

FORTRAN SOFTWARE FOR CREATING AND MAINTAINING A LIBRARY CATALOGING SYSTEM ON SCIENTIFICALLY ORIENTED COMPUTERS:

VOLUME I, PROGRAM ADM - THE DATA BASE

MARCH 1977

CHRISTOPHER A. FEUCHTER

DIRECTORATE OF AEROSPACE STUDIES
DCS/DEVELOPMENT PLANS, HQ AFSC
KIRTLAND AFB, NEW MEXICO 87117

The usefulness of a collection of reference documents is highly dependent upon the ease of identifying which documents are of potential interest in any given instance. Increasing the scope and flexibility of the document cataloging system is one method of making this identification easier. With this as motivation, the Directorate of Aerospace Studies undertook a computerization of the cataloging system for its document center in 1975. The principles of the system and the required software were developed inhouse over a period of months. They have since been refined and have proved themselves in use for two years. This documentation was written to describe the software and its use in enough detail to permit its adoption by other groups seeking the benefits coming from computerized cataloging; be it of documents, books, records - anything which can be categorized by various types of information.

The author would like to acknowledge the assistance of Mary C. Kennedy in designing the principles of the system. I am only sorry that it has not yet been possible to implement all her suggestions.

Christopher C. Feuchter

CHRISTOPHER A. FEUCHTER
Study Director

This report has been reviewed and is approved for publication.

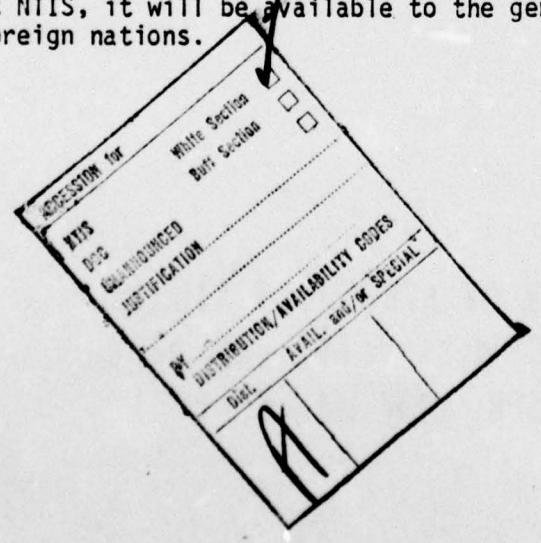
Harry L. Gogan.

HARRY L. GOGAN
Technical Director

James P. Ditz

JAMES P. DITZ, Lt Colonel, USAF
Director of Aerospace Studies

This report has been reviewed by the Information Office (OI) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.



UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

(14) 1. REPORT NUMBER: DAS-TR-77-2

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

(6) 4. TITLE (and Subtitle): FORTRAN SOFTWARE FOR CREATING AND MAINTAINING A LIBRARY CATALOGING SYSTEM ON SCIENTIFICALLY ORIENTED COMPUTERS. VOLUME I. PROGRAM ADM - THE DATA BASE.

5. TYPE OF REPORT & PERIOD COVERED

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)

(10) Christopher A. Feuchter

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION NAME AND ADDRESS

Directorate of Aerospace Studies
Kirtland AFB NM 87117

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS

(11) 12. REPORT DATE: July 1977

13. NUMBER OF PAGES: 81

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)

(12) 71 p.

15. SECURITY CLASS. (of this report): Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release
Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Automation
Computers
Libraries
Computer Programs
Computer Program Documentation
FORTRAN

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The cataloging system of the document center of the Directorate of Aerospace Studies has been computerized on the CDC 6600 computers of the Air Force Weapons Laboratory. The system is unusual in that catalog listings of the holdings may be arbitrary in informational content, i.e., with regard to what types of information appear and their order of appearance. Although its initial application was to library cataloging it may be used to catalog any collection of objects which can be described by at most 20 categories of information.

This document describes ADM, the FORTRAN program which originally creates

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

(cont)

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

409 237

20. ABSTRACT (Cont) (p1473A)

and then maintains the data base from which catalog listings are made. It is a user's manual and a programmer's manual. ADM is described in sufficient detail to permit its adoption to other high level computer languages and/or other large, scientifically oriented computers.

1473B

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

1. INTRODUCTION

Program ADM which is described in this volume and Program LIBLIST which is described in volume II of this report were designed and developed to automate the cataloging operation of the document center of the Directorate of Aerospace Studies (DAS), Kirtland AFB, NM. These programs satisfy the specific needs of that center. Additionally, the software is designed around the Control Data Corporation (CDC) 6600 computers of the Air Force Weapons Laboratory at Kirtland AFB. Nevertheless it is believed that many groups requiring cataloging services can easily adapt these programs to their needs. This two-volume set should provide all the information necessary to determine the suitability of applying ADM and LIBLIST to particular situations. The discussion of the programs is believed to be detailed enough to allow conversion of the FORTRAN software to different computers and/or other programming languages.

The DAS document center maintains a collection of between 5000 and 10,000 holdings. The majority of these holdings are unclassified and classified technical documents. While the size of the collection is reasonably static, the composition is in a constant state of flux. New documents are acquired and old documents having a limited life or of limited interest are purged. Until ADM and LIBLIST were written these documents were cataloged on cards by title, corporate author, report number, accession number, and key words. The decision to automate was based on a single consideration - to increase the accessibility of information about the collection. The programs do this through: (1) increasing the number of types of catalogs maintained, (2) standardizing the format of the catalogs, (3) enhancing the physical ease of using the catalog (computer listings can be scanned more readily than individual catalog cards), (4) providing flexibility by allowing variations in catalog content (including the production of special purpose catalogs on a onetime basis), and (5) allowing easy dissemination of the catalogs through the production of multiple copies.

Program ADM performs the functions of initially creating and then maintaining a data base of information describing the documents. Input to the program is by punch card. Output consists of printed material used as an aid to detecting errors in the input, and/or a new or updated data base stored on magnetic tape or disk. Program LIBLIST distills data from the data base into catalog listings whose content is specified by the user.

Maintaining a data base is basically the process of changing it so that it continues to contain an accurate description of the collection holdings. This means that it must be possible to add information to the data base, delete information, or modify existing information. The program takes its name from these three functions: A(dd) D(elete) M(odify).

Each holding may be described by up to 20 categories of information. These categories are the same for all holdings described by the data base. The list of categories as currently used at DAS is given in table 1. These selections are in no way indigenous to the program coding except that each category is identified on the input cards by a unique two-character code. As the program now stands these codes are suggestive of the categories of table 1 that they represent. They could easily be changed to suggest different categories.[1] This and the more important fact that, with one exception, all categories of data are treated identically in the computer, makes the content of the data base very flexible. This in turn allows ADM to maintain a data base for any set of entities which can be described by at most 20 categories of information.

The equal treatment of categories of information in the data base is used by LIBLIST to allow the creation of catalogs whose entries are arbitrary in content. For example (using the categories of table 1) LIBLIST can produce a catalog by title, followed in order by corporate author, document date, classification, AD number, and accession number. Just as easily it could produce a catalog by title, followed by requestor, date received, AD number, and accession number. This flexibility is a useful feature of this cataloging system.

Another convenient feature of the ADM program is the input scheme. It was designed to make the initial punching of cards and any subsequent correction of errors as easy as possible. Information to be punched need not be entered on coding forms which specify card columns. Depending upon the skills of the keypunch operator, punching may be done from typed forms, handwritten forms, or the

---

1. Having the codes suggest the category of information is merely a convenience to the program user.

Table 1

DATA ITEMS

	ITEM (TWO CHARACTER CODE)	REMARKS
1.	Accession Number (AN)	A unique alphanumeric document identifier assigned by the Directorate of Aerospace Studies document librarian to all documents, except microfiche.
2.	AD Number (AD)	An alphanumeric identifier assigned by the Defense Documentation Center (DDC) to all copies of any document they process.
3.	Title (TI)	Document title.
4.	Copy Number (CN)	Copies of classified documents are numbered sequentially at printing.
5.	Author (PA)	Personal author(s)
6.	Corporate Author (CA)	The organization(s) responsible for authoring the document.
7.	Report Number (RN)	A number assigned to a document by the corporate author.
8.	Document Date (DD)	The nominal date of document publication.
9.	Date Received (DR)	The date the document was received at the Directorate of Aerospace Studies.
10.	Requestor (RE)	The individual initially requesting the document.
11.	Requesting Study (RS)	The study for which the document was acquired.
12.	Classification (CL)	The security classification of the document.
13.	Special Access (SA)	Special restrictions on distribution of the document.
14.	Downgrading Information (DI)	The date(s) of downgrading a classified document to a lower classification.
15.	Document Status (DS)	e.g., on loan, loaned, etc.
16.	Comments (CM)	Anything not covered by other categories.
17.	Format (FM)	Document format, e.g., book, technical report, vugraph, etc.
18.	Key Words (KW)	Words, mnemonics, or expressions characterizing the document.
19.	Abstract (AB)	This category is not currently used.
20.	Not used	

documents themselves. In large part this is the result of there being no requirement as to the order in which information is entered upon the cards, nor any practical restriction[2] on the length of the information associated with a particular category.

Section 2 discusses the structure of the data base in depth. This is followed by a discussion in section 3 of the concept of alphanumeric ordering used in ADM (and LIBLIST) and its implications to program users. Input and output are treated in sections 4 and 5, respectively. Section 6 discusses input errors and the best methods for correcting them. The maintenance of the data base is the subject of section 7, while the final section contains descriptions of the functions of the main program and the three subroutines. The report concludes with two appendices. Appendix A contains an alphabetical list and description of all important program variables. This is followed by a well-commented listing of the FORTRAN program in appendix B.

2. There is a restriction of about 500 characters on the total of the lengths of all the individual categories. This could easily be increased by a factor of four or even six on the CDC 6600 used at Kirtland AFB.

2. THE DATA BASE

A data base is the total of all information about a set of documents that is maintained on a mass storage file.[3] Within a data base each document is represented by a data set. This in turn is composed of data items and a table indicating where the data items are stored within the data set, and the lengths of the items in number of computer words.

2.1 DATA ITEMS

A list of the data items was given in table 1. Data items represent categories of information, e.g., title, personal author, key words. For a given document some of these categories may not be represented. There may be no personal author, for example. Some categories may require multiple entries which need to be treated independently when forming a catalog listing. An obvious example is key words. And whereas the title for one document may occupy a single computer word, that for another document may require 15 computer words.

To handle these variations the data item was conceived as having a substructure and an indefinite length. The substructure allows multiple entries in a given data item to be given independent status by linking them with plus (+) signs.[4] (This feature eliminates the plus sign from the otherwise usable character set.) A data item has zero length if there is no information about a document in that category. The upper limit for its length is determined only by the restriction that the sum of the lengths of all data items in a data set should not exceed 54 computer words.

Data items in the data base are stored in display code. The CDC printer characters and display codes are given in table 2. All but three may be used in data items as themselves. The exceptions are the colon (:), the plus sign (+), and the blank. The colon has a display code of 00. This means that in the alphanumeric ordering

---

3. This data base may be called an electronic data base to distinguish it from the collection of data cards from which it was derived - the card data base. Throughout this report data items and data sets not yet part of the data base are referred to as input data items or input data sets to distinguish them from those in the data base.

4. Because the purpose of the accession number (or AD number if an accession number is not used) is to provide the data set with a unique identification, multiple entries in these data items make no sense.

Table 2

THE PRINTER CHARACTER SET AND DISPLAY CODES

CHARACTER	DISPLAY CODE*	CHARACTER	DISPLAY CODE	CHARACTER	DISPLAY CODE
:	00	V	26	=	54
A	01	W	27	Blank	55
B	02	X	30	,	56
C	03	Y	31	.	57
D	04	Z	32	≡	60
E	05	0	33	[	61
F	06	1	34	]	62
G	07	2	35	%	63
H	10	3	36	≠	64
I	11	4	37	→	65
J	12	5	40	∨	66
K	13	6	41	∧	67
L	14	7	42	↑	70
M	15	8	43	↓	71
N	16	9	44	<	72
O	17	+	45	>	73
P	20	-	46	≤	74
Q	21	*	47	≥	75
R	22	/	50	¬	76
S	23	(	51	;	77
T	24	)	52		
U	25	$	53		

*The display code is the octal representation in the machine.

process (see section 3) it plays the role of a blank (leaving no role for the blank). The plus sign has the special function just discussed. The remaining 61 characters are available to be used in any manner in a data item with two exceptions. First two consecutive colons (blank substitutes) may not appear imbedded in the information of a data item, since two consecutive colons indicate the end of a data item during certain processes within the computer. (In the data base two consecutive colons must appear at the end of every data item, even if an extra computer word must be added to provide one or both of them.) Second, all punctuation characters are automatically spaced at the time ADM creates the data base.[5] Thus, for example, commas are always followed by a colon (blank substitute), but never preceded by one. This information is given in table 3.

Table 3

PROGRAMMED PUNCTUATION SPACING

No blank precedes

+ - * / =)] , . ; $ %

No blank follows

([+ - * / =

A blank precedes

([

A blank follows

)] , . ; $ %

Should two characters with conflicting rules be adjacent to one another, that applying to the second character will prevail, e.g.,) and - is spaced)-.

Each data item occupies an integral number of computer words. Thus, for example when Rand Corporation is used as a corporate author it would be stored in the computer in octal notation as the two computer words[6]

2201160400031722201|2201241117160000000

5. The automatic spacing of these characters places the burden of punctuation spacing on the computer rather than on those preparing input for ADM. This is an important asset in achieving the correct order of entries in listings produced by LIBLIST. See section 3.

6. There are 10 six bit characters in a 60 bit computer word.

which when converted to characters is

RAND:CORPO|RATION::::

Notice that this data item is terminated by at least two consecutive "blanks," i.e., colons. Were the corporate author Bigdome Corporation, it would appear in character representation as

BIGDOME:CO|RPORATION:|::::::::::

where the extra word of "blanks" has been added to provide the second "blank" of the pair required to indicate the end of the data item.

2.2 DATA SET

To form a data set, the data items are assembled in consecutive locations in the data base in the order given in table 1.

Some method must be used for specifying the locations of each of the data items. An easy way is to include at the beginning of the data set a table giving the location of the first word of each data item (numbering the first word of the data set 1) and the number of words comprising it. This has been done by packing this information for the 20 data items into five computer words, four data items to a word. Since a CDC 6600 word has 60 bits, this means that the length and location information for a data item must be presented in 15 bits. The first six are used to give the length; the last nine the location. This is shown in figure 1.

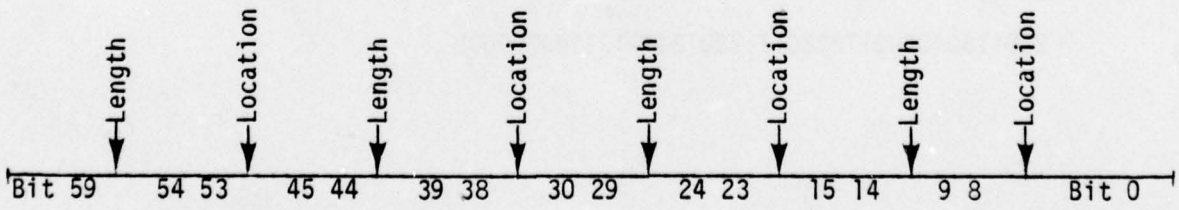


Figure 1. Format of a Word of the Length-Location Table

The correspondence between data item, the computer words in the table, and bit numbers is given in table 4. The maximum expressable length of a data item is 77 octal (63 decimal) computer words. The maximum location is 777 octal (511 decimal) words.

The data set is completed by preceding the length-location table with a single computer word giving the total length of the data set. The final data set thus has the form shown in figure 2.

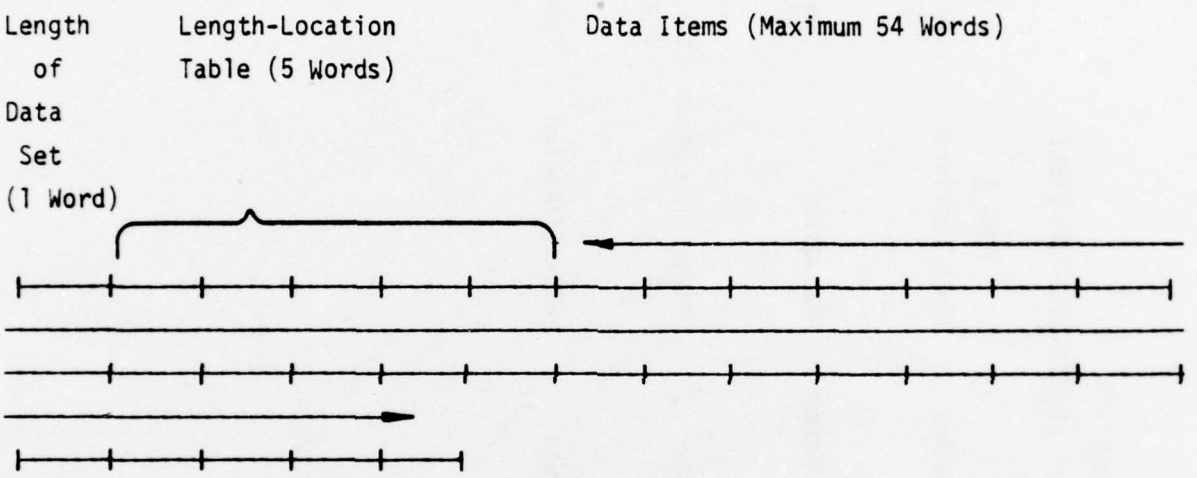


Figure 2. Format of a Data Set

2.3 DATA BASE

The data base is composed of the collection of all data sets. It is advantageous to have these data sets ordered in some manner in the data base. This is brought about by the need to be able to delete or modify data base data sets efficiently with Program ADM. Suppose, for example, that 100 data sets are to be deleted from a data base of 1000 data sets because the corresponding documents are no longer part of the collection. If there is no order to the data sets in the data base, the list of data sets to be removed will have to be searched over and over again as the data sets in the data base are examined one by one looking for matches. If, however, both lists are in order, each list will need to be searched only once to identify the 100 data sets to be removed.

The ability to order the data sets depends upon each data set having a unique identifier. For microfiche with AD numbers, this identifier is the AD number.

Table 4

THE STRUCTURE OF THE LENGTH-LOCATION TABLE

BITS	WORD 2	WORD 3	WORD 4	WORD 5	WORD 6
Length 59-54 Location 53-45	Accession Number	Personal Author	Date Received	Special Access	Format
Length 44-39 Location 38-30	AD Number	Corporate Author	Requestor	Downgrading Information	Keywords
Length 29-24 Location 23-15	Title	Report Number	Requesting Study	Document Status	Abstract
Length 14-9 Location 8-0	Copy Number	Document Date	Classification	Comments	Not Used

For all other documents it is the accession number assigned to it at the Directorate of Aerospace Studies.

3. ALPHANUMERIC ORDERING

Ordinary alphabetical ordering is accomplished by comparing two words (or word groups) beginning with the left-most characters of each and preceding character by character to the right until a difference is found. The two corresponding but different characters then determine which word or word group comes first by which is "alphabetically smaller" according to the sequence blank, A, B, ... Z. Punctuation is generally ignored in determining alphabetical order.

The ordinary extension of alphabetic ordering to include numbers - alphanumeric ordering - is not consistent with this process. Instead of beginning with the left-most digits of the two strings of digits being compared and proceeding character by character until a difference is found, the numbers are compared as entities. Thus nine precedes 10, a situation which everyone expects. This is, of course, a convention, and one that does not lend itself as efficiently to alphanumerically computer ordering as another which will be described.

For efficiency in computer ordering it is desirable to allow the ordering to proceed in the nonconventional way of comparing numbers exactly as letters are compared - character by character from the left. Using this scheme, very nonconventional ordering takes place, e.g., 10 precedes 9; 1000 precedes 11, etc. This too is a convention, but not one everyone will readily adapt to. A partial solution is to preface numbers with zero. Thus 09 will indeed precede 10.[7] This works as long as the two numbers being compared have the same number of digits. In most cases this is a practical way to obtain the traditional ordering.[8]

There is one further difference between traditional alphanumeric ordering and the computer ordering appearing in ADM. Punctuation marks are considered as characters just as are the blank, letters, and the digits 0-9. The alphanumeric order

---

7. Roman numerals provide another problem in computer ordering. For example C(100) will order before I (1), IX (9) before V (5), etc. It will generally pay to replace Roman numerals with Arabic numbers.

8. As currently used, for example, all accession numbers are entered as five digit numbers, the necessary leading zeros appearing as necessary.

of all the characters is that given previously in table 2. The order corresponds to the numerical order of the display codes. This consideration of punctuation in the ordering process makes consistency of punctuation essential if listings are to have the desired order at all times.[9] Below are some pairs of expressions ordered traditionally and as the ADM/LIBLIST programs order them.

	TRADITIONAL	ADM/LIBLIST
1.	BASE	BASE
	BASES	BASES
2.	ALPHA9 ALPHA09	ALPHA10 ALPHA09
	or	or
	ALPHA10 ALPHA10	ALPHA9 ALPHA10
3.	1812	18027
	18027	1812
4.	01812	01812
	18027	18027
5.	STATISTICS, VOL. 1	STATISTICS, VOL 2
	STATISTICS, VOL 2	STATISTICS, VOL. 1

These ordering characteristics should be understood and planned for before the first input data set is ever keypunched.

9. Consideration is being given to changing the program logic to treat strings of digits as entities and to ignore punctuation in the ordering process.

4. INPUT

The first card of every data deck for Program ADM defines the three quantities OLDFILE, NEWFILE, and IPRINT. The card is read under a 3I5 format, and each of the quantities can take the values zero or unity with the following implications:

OLDFILE	= 0	An old data base file is not to be input to the program.
	= 1	An old data base file is to be input to the program to be integrated with input data sets to create a new data base file.
NEWFILE	= 0	A new data base file is not to be created by the run. The run serves only to provide a check of the format and content of the input data sets.
	= 1	A new data base file is to be created by the run.
IPRINT	= 0	Do not list input data sets using an expanded format (see section 5).
	= 1	List input data sets in expanded format.

A new data base file may be created entirely from input data sets, or the input data sets may be combined with an existing data base file to produce a new data base file. In the first case OLDFILE = 0, NEWFILE = 1; in the second OLDFILE = 1, NEWFILE = 1. The combination OLDFILE = 1, NEWFILE = 0 makes no sense in terms of useful program operation.

Following this initial card are the input data sets. An input data set consists of one or more cards which specify the document to be processed, how it is to affect the data base, and the data items that are to be treated. This information is contained in columns 1-72 of successive cards. Columns 73-80 are used for visual identification of the cards only and may contain any information desired (e.g., the document identification number).

The beginning of a data set is indicated by the first three columns of the first card containing *A*, *D*, or *M*. The *A* indicates that the data set will be added to the data base. The *D* indicates that the document identified in the input data set is to be deleted from an already existing data base, while *M* tells the program that the input data set is to modify the specified data set from an

existing data base. The end of a data set is signaled by two successive blank columns appearing anywhere in columns 1-72 of a card, or appearing in column 72 of one card and column 1 of the succeeding card. (Within a data set, the program regards column 1 of one card as being adjacent to column 72 of the previous card.) Nothing should be punched after the second blank on the last data card of a set except in columns 73-80. The program would regard additional information before column 73 as an error, as two successive blanks followed by more information might represent an error in punching the cards. In those cases that the actual data associated with the input data set ends in columns 71 or 72 of a card, an additional card with blanks in columns 1-72 must be added to provide the required blank(s) signifying the end of the input data set.

The two-character input data identifiers (see table 1) appear within an input data set between a pair of asterisks (*), for example *TI*. They are followed directly by the data they identify. Thus, an individual identifier and its data might appear as

*TI*ALICE IN WONDERLAND

If multiple (up to ten) entries are associated with an identifier they should be joined by plus (+) signs, e.g.,

*TI*ALICE IN WONDERLAND + THROUGH THE LOOKING GLASS

This will cause Program LIBLIST, when printing a title listing, to enter this document in the listing under both ALICE IN WONDERLAND and THROUGH THE LOOKING GLASS. An input data set is composed of one or more of these units following the initial three characters *A*, *D*, or *M*. The example below, figure 3, illustrates a typical input data set. The circled letters with their accompanying arrows serve to point out particular items for discussion. It is important to note that the order of the input data items within the input data set is immaterial.

Ⓐ This asterisk does double duty, acting as the third character of *A* and at the same time serving as the first asterisk of the pair indicating the identifier AN.

Ⓑ The program recognizes the end of one input data item by encountering the next identifier (or two successive blank columns). Note that no blank column appears here.

Ⓒ Note as indicated before that column 72 of one card is considered by the program to be adjacent to column one of the next.

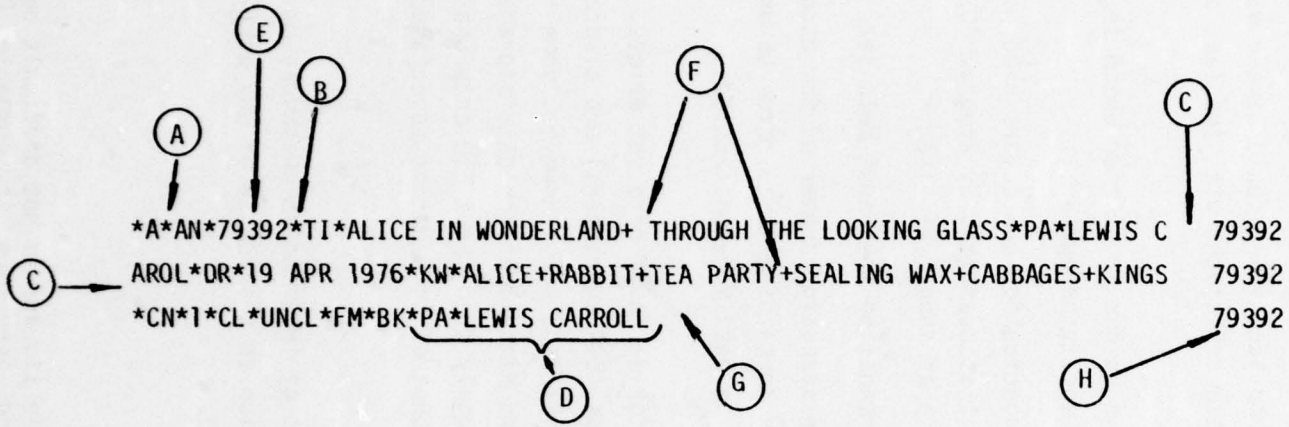


Figure 3. An Example of a Card Input Data Set

Ⓓ The second appearance of an identifier and input data item causes the initially defined information to be replaced by the newly defined data. This feature allows an error in the first appearance of a data item to be corrected without necessitating repunching the cards including and following the error (as would be required by a correction resulting in a change of length of the input data item as is the case here). The same identifier could appear within a data set more than twice. In such a case the data associated with the last appearance is always used.

Ⓔ Every input data set has a unique identifier which is the accession number or the AD number when there is no accession number.

Ⓕ Spacing relative to all punctuation characters need not be a concern in an input data set. This spacing is standardized by Program ADM. These are also examples of multiple entries within an input data item.

Ⓖ The two blank columns terminating the input data set.

Ⓗ Columns 73-80 contain the accession number of the document described by the input data set. Should a deck containing these cards be dropped, for example, this allows the input data set to be easily reconstructed.

A maximum acceptable length for an input data set exists, but it is not easily defined. Fifty-four computer words (540 characters) are allocated in the computer for the input data items of an input data set. However, some of the 540 characters are automatically blank filled (and hence lost for data storage) in the process of storing the input data items. Roughly then about 500 data characters can be stored per input data set - about the amount of information associated with seven or eight input cards.[10]

This limitation on the amount of information which can be stored about a particular document is the only limitation on the length of any particular input data item.

---

10. An expansion of this available storage is not difficult but it naturally increases the size of Program ADM and, even more, its companion Program LIBLIST. Superceded input data items associated with repeated two-character identifiers do not count in the approximate seven-eight card limit.

That is, as long as all the input data items can be stored in the allocated 54 words, the lengths of the individual input data items are immaterial.[11]

When an input data set is being used to delete a document from an existing data base, it needs only to have *D* followed by the unique document identifier as

*D*AN*35802

or

*D*AD*947692

When the modification of data items of a document of an existing data base is specified, the input data items present are only those being changed plus either the accession number, or if one does not exist, the AD number. In the modification of a data set, the input data items replace the existing data items if they exist. If no data item exists in the data set corresponding to the one being added, the input data item is simply added to the data set. It is impossible to modify the accession number (or AD number if an accession number does not exist) by this method. These types of modification are done by deleting the data set from the data base and reintroducing it with a different accession or AD number. To delete an existing data item from the data set, the two-character identifier should be followed by a single minus sign (-). For example to delete the corporate author use *CA*-.

Of the 64 printer characters, three are subject to restrictions when used in input data items. The plus sign (+) is restricted to linking together multiple data items, the colon if used will be stored as a blank in the data base, and the asterisk must appear as a double asterisk (**) each time it is to appear in a data item. The doubling of the asterisk prevents the program from interpreting either of the asterisks as a delimiter. The first of the pair is simply discarded during processing.

The end of input data is indicated by *E* appearing in the first three columns of the last data card.

---

11. While the length of the unique identifier of the input data set (accession number or AD number) is as arbitrary as the length of any other input data item, proper program operations can not be expected unless the first ten characters uniquely define that document. Alphanumeric ordering of documents by accession or AD number is based only on the first ten characters (first word).

5. OUTPUT

The usefulness of any cataloging system is dependent upon the accuracy of the information it contains. Misfiled catalog cards or inaccuracies causing entries on computer printouts to be misordered can cause the "loss" of documents in many cases. With a manual filing system improperly filed cards can be the result of informational errors on the file cards or human error in integrating the cards into the rest of the system. This latter error is inherently absent from computer generated catalogs. "Lost" documents occur only because of errors of informational content.

To minimize errors of informational content in the data base, and hence in the computer generated listings, Program ADM has an output option that lists and identifies the input data items of each input data set on a separate page. An example is shown in figure 4. This format is easy to read and provides a reasonable way to check for keypunch transcription errors. Whatever is shown in this listing is what will be placed in the data base. In addition to showing what information is present, it also indicates which input data items are not present. The action indicating character (A, D, or M) is printed at the top of the page. This output option is controlled by a parameter read from the first input card of a data deck (see section 4). If the option is selected (IPRINT=1) a page is produced for every recognized input data set in the run. If it is not selected no printout of this type is produced.

One other type of printout is produced by the program, a card image by card image listing as shown in figure 5. In addition to reproducing all input data set cards, this listing also contains error messages detailing card format errors detected by the program. These error messages are discussed in section 6. The messages appear as close to the card containing the error as program logic will allow. This listing is not written directly to the output file because selecting the option of the page per document printout would cause it to be interspersed with the card image listing. To avoid this[12] the card images and error messages are written to TAPE3 and copied to output after program execution has been completed.

---

12. Some might find interspersing the card images with the page per document printout to be desirable. The program could be easily modified to give this.

A		
AN	ACCESSION NUMBER	36065
AD	AD NUMBER	
TI	TITLE	MISSION ANALYSIS FOR MISSILE AND NUDET SURVEILLANCE (U) VOLUME 3. MISSILE SURVEILLANCE SYSTEMS. ANNEXES
CN	COPY NUMBER	
PA	PERSONAL AUTHOR	
CA	CORPORATE AUTHOR	AF SPACE AND MISSILE SYSTEMS ORGANIZATION
RN	REPORT NUMBER	SAMSO-TR-75-83-VOLUME 3-ANNEXES
DD	DOCUMENT DATE	DEC 1975
DR	DATE RECEIVED	21 JAN 1976
RE	REQUESTOR	
RS	REQUESTING STUDY	TFR
CL	CLASSIFICATION	SECRET/XGDS/CAT1+2+3
SA	SPECIAL ACCESS	
DI	DOWNGRADING INFO	2005
DS	DOCUMENT STATUS	
CM	COMMENTS	
FM	FORMAT	TR
KW	KEY WORDS	NUDET+MISSILE SURVEILLANCE
AB	ABSTRACT	

Figure 4. Sample of Page Per Document Output

```
CARD    1    *A*RE*KUERSCHNER*DR*15 JAN 1976*RS*TFR*DD*16 JUL 1975*CL*UNCL*AN*TFR-482   TFR482
CARD    2    *FM*TR*CA*MCDONNELL DOUGLAS CORPORATION*PA*VETTER, H.C.+CHRISTENSEN, H.E   TFR482
CARD    3    .+KREIGER, R.J.*TI*PLANETARY/DOD ENTRY TECHNOLOGY FLIGHT EXPERIMENTS(MID   TFR482
CARD    4    -TERM REPORT)*KW*PLANETARY ENTRY EXPERIMENTS*RN*MDC-NAS2-8678*AN*35766*R   TFR482
CARD    5    S*MIS+TFR                                                                  TFR482
CARD    6    *A*RE*KUERSCHNER*DR*14 JAN 1976*RS*TFR*DD*MAY 1975*CL*UNCL*AN*TFR-483*AD   TFR483
CARD    7    *8003992*FM*TR*CA*AF SPACE AND MISSILE SYSTEMS ORGANIZATION*TI*DOD SPACE   TFR483
CARD    8     TRANSPORTATION SYSTEM(STS)PAYLOAD INTERFACE STUDY FY 75 EXTENSION*KW*ST   TFR483
CARD    9    S*PAYLOAD INTERFACE*RN*SAMSO-TR-75-136*AN*36060                            TFR483
CARD   10    *A*RE*KUERSCHNER*DR*8 JAN 1976*RS*TFR*DD*FEB 1975*CL*SECRET/RD*AN*TFR-48   TFR484
CARD   11    4*AD*C001043*FM*TR*CA*AF WEAPONS LABORATORY*PA*HIGGINS, D.F.*TI*DESIGN G   TFR484
CARD   12    UIDELINES FOR SATELLITE SYSTEM GENERATED ELECTROMAGNETIC PULSE HARDENING   TFR484
CARD   13    (U), VOLUME 1*KW*EMP+HARDENING+SATELLITE*RN*AFWL-TR-74-42 VOLUME 1         TFR484
CARD   14    *A*RE*HIGGINS*DR*19 JAN 1976*RS*TFR*DD*SEP 1975*CL*SECRET/XGDS/CAT 3*AN*   TFR485
CARD   15    TFR-485*CN*030*FM*TR*DI*1989*CA*TRW SYSTEMS GROUP*TI*ADVANCED METRIC SYS   TFR485
CARD   16    TEM EVALUATION FINAL REPORT(U)*KW*MISSILE TRACKING+LASER TRACKING*RN*TRW   TFR485
CARD   17    -26946-6009-TE-01*AN*36061                                                 TFR485
CARD   18    *A*RE*BOMBER*DR*19 JAN 1976*RS*TFR*DD*7 SEP 1973*CL*SECRET/XGDS/CAT 3*AN   TFR486
CARD   19    *TFR-486*FM*ROC*CA*AF AEROSPACE DEFENSE COMMAND*TI*REQUIRED OPERATIONAL    TFR486
CARD   20    CAPABILITY(ROC)AIR DEFENSE AIR-TO-AIR MISSILE(U)*KW*ROC+MISSILE, AIR TO    TFR486
CARD   21    AIR*RN*ADC-ROC-12-73*AN*35767*RS*MIS+TFR                                   TFR486
CARD   22    *A*RE*KUERSCHNER*DR*14 JAN 1976*RS*TFR*DD*DEC 1975*CL*SECRET/GDS*AN*TFR4   TFR487
CARD   23    87*CN*129*FM*TR*DI*1983*CA*RAND CORPORATION*PA*FELDMAN, N.E.*TI*PHYSICAL   TFR487
CARD   24    LY SURVIVABLE BACKUP FOR THE NAVSTAR GLOBAL POSITIONING SYSTEM(U)*KW*NAV   TFR487
CARD   25    STAR+GLOBAL POSITIONING SYSTEM*RN*RAND-R-1780-ARPA*AN*TFR-487              TFR487
CARD   26    *A*RE*KUERSCHNER*DR*26 JAN 1976*RS*TFR*DD*24 JAN 1975*CL*SECRET/XGDS/CAT   TFR488
CARD   27    3*AN*TFR-488*CN*2-19*RN*ADC-ROC-1-75*FM*ROC*DI*2005*CA*AF AEROSPACE DEFE   TFR488
CARD   28    NSE COMMAND*TI*REQUIRED OPERATIONAL CAPABILITY (ROC) FOR AN IMPROVED SPA   TFR488
CARD   29    CE SURVEILLANCE SYSTEM*KW*ROC+SPACE SURVEILLANCE SYSTEM*AN*35768*RS*MIS+   TFR488
CARD   30    TFR                                                                        TFR488
CARD   31    *A*RE*HIGGINS*DR*26 JAN 1976*RS*TFR*DD*JUN 1975*CL*SECRET/XGDS/CAT 3*AN*   TFR489
CARD   32    TFR-489*CN*033*FM*TR*DI*1989*CA*AF SPACE AND MISSILE SYSTEMS ORGANIZATI   TFR489
CARD   33    ON + TRW SYSTEMS GROUP*PA*RODEMS,R.C.*TI*STUDY ON THE USE OF THE GLOBAL    TFR489
CARD   34    POSTIONING SYSTEM(GPS) TO SUPPORT MINUTEMAN POSTFLIGHT ANALYSIS*KW*GPS +   TFR489
CARD   35    MINUTEMAN*CM*XEROX COPY*CA*AF SPACE AND MISSILE SYSTEMS ORGANIZATION*AN*   TFR489
CARD   36    36062                                                                      TFR489
CARD   37    *A*RE*COOPER*DR*26 JAN 1976*RS*TFR*DD*MAY 1974*CL*SECRET/XGDS/CAT 3*AN*   TFR490
CARD   38    TFR-490*AD*530384L*RN*ASD-TR-74-3VOLUME 1*FM*TR*DI*1987*CA*AF AERONAUTIC   TFR490
CARD   39    AL SYSTEMS COMMAND*TI*DRONE CONTROL AND DATA RETRIEVAL SYSTEM (DCDRS) PR   TFR490
CARD   40    ELIMINARY DESIGN STUDY FINAL REPORT VOLUME 1 EXECUTIVE SUMMARY*KW*REMOTE   TFR490
CARD   41    LY PILOTED VEHICLES+DRONE CONTROL*RN*ASD-TR-74-3 VOLUME 1*AN*36130         TFR490
CARD   42    *A*RE*HIGGINS*DR*26 JAN 1976*RS*TFR*DD*DEC 1973*CL*SECRET/XGDS/CAT 3*AN*   TFR491
CARD   43    TFR-491*AD*530646*RN*GRC-RM-1878*FM*ME*CA*GENERAL RESEARCH CORPORATION*P   TFR491
CARD   44    A*FORD,C.T.*TI*BOMBER DEFENSE SYSTEM REQUIREMENTS*KW*BOMBER DEFENSE + AI   TFR491
CARD   45    R-TO-AIR MISSILES*CA*GENERAL RESEARCH CORP*AN*36063                        TFR491
CARD   46    *A*DR*21 JAN 1976*RS*TFR*DD*DEC 1975*CL*SECRET/XGDS/CAT 1+2+3*AN*TFR-492   TFR492
CARD   47    492*FM*TR*DI*2005*RN*SAMSO-TR-75-83-VOLUME 3*CA* AF SPACE AND MISSILE SY   TFR492
CARD   48    STEMS ORGANIZATION*TI*MISSION ANALYSIS FOR MISSILE AND NUDET SURVEILLAN    TFR492
CARD   49    CE (U) VOLUME 3, MISSILE SURVEILLANCE SYSTEMS*KW*NUDET + MISSILE SURVEIL   TFR492
CARD   50    LANCE*AN*TFR-492*AN*36064                                                  TFR492
CARD   51    *A*DR*21 JAN 1976*RS*TFR*DD*DEC 1975*CL*SECRET/XGDS/CAT1+2+3*AN*TFR-493    TFR493
CARD   52    *RN*SAMSO-TR-75-83-VOLUME 3-ANNEXES*CA*AF SPACE AND MISSILE SYSTEMS ORGA   TFR493
CARD   53    IZATION*TI*MISSION ANALYSIS FOR MISSILE AND NUDET SURVEILLANCE (U) VOLUM   TFR493
CARD   54    E 3, MISSILE SURVEILLANCE SYSTEMS, ANNEXES*KW*NUDET + MISSILE SURVEILLAN   TFR493
CARD   55    CE*FM*TR*DI*2005*CA*AF SPACE AND MISSILE SYSTEMS ORGANIZATION*AN*36065     TFR493
CARD   56    *A*RS*TFR*DD*FEB 1975*CL*UNCL*AN*TFR-494*FM*TR*CA*AF 4950TH TEST WING*TI   TFR494
CARD   57    *WRIGHT PATTERSON STORY*KW*WRIGHT-PATTERSON AFB                            TFR494
CARD   58    *A*RE*ERB + HIGGINS*DR*20 OCT 1975*RS*TFR*DD*SEP 1975*CL*UNCL*AN*TFR-495   TFR495
CARD   59    *FM*PA*CA*AF AERONAUTICAL SYSTEMS DIVISION*TI*STATEMENT OF WORK FOR HYPE   TFR495
CARD   60    RSONIC VEHICLE APPLICATIONS STUDY AND TASK SUMMARY SHEET P.E. 63101F PRE   TFR495
CARD   61    LIMINARY DESIGN AND DEVELOPMENT*KW*HYPERSONIC VEHICLES                     TFR495
CARD   62    *E*
```

Figure 5. Example of Card Image Output

6. CORRECTING INPUT DATA SET ERRORS

In dealing with input data sets two types of errors will be encountered. The first type are errors in informational content - data transcription errors. The second type are those detected by the program - format errors. Both types of errors can always be corrected by brute force - correcting the error and (usually) repunching all the data following the error because of an accompanying insertion or deletion of characters. This discussion considers alternatives to the brute force technique.

In general, errors of the first type are more straightforward to correct as there are but two possible techniques involved. First, if incorrect characters can be replaced one-for-one with correct characters, there is no problem. The length of the input data item will not have changed and there will be no impact on the remainder of the input data set. However, it will commonly occur that the correction will entail increasing or decreasing the number of characters in the input data item, i.e., changing its length. Changing the length of an input data item on a card requires shifting everything which follows it either to take up the created space or provide the needed space. In this case where the correction shortens or lengthens an input data item the preferred method of correction is to append to the end of the input data set the appropriate two-character identifier and the corrected input data item. As discussed earlier this will cause the incorrect input data item to be replaced by the corrected one in the input data set assembled in the computer (see the discussion associated with figure 3 in section 4).

Format errors are basically corrected by one of the two methods described for informational content errors. However, the procedure for making the format corrections can in cases be more involved. The error messages associated with each of the format errors are given below along with a discussion of the meaning and a method for correcting the error. Format error messages appear in the card image listing as close to the card containing the error as practical.

1. CARD XXXX BEGINS A NEW DATA SET FOLLOWING AN IMPROPERLY TERMINATED DATA SET.

The data for an input data set end in columns 71 or 72 and are not followed

by a blank card to supply the required blank column(s) indicating the end of the input data set.

This error is corrected by inserting a card blank in columns 1-72. The message can also indicate missing or out of order cards in the deck.

2. CARD XXXX CONTAINS TWO SUCCESSIVE BLANK COLUMNS FOLLOWED BY ADDITIONAL DATA. FOLLOWING CARDS WILL BE IGNORED UNTIL A NEW DATA SET IS INITIATED.

Two (or more) successive blank columns inadvertently appear imbedded in an input data set. The problem is corrected by filling the excess blank columns with arbitrary characters (thus eliminating the format error) and adding the correct two-character identifier and input data item to the end of the input data set.

3. ONE CHARACTER IDENTIFIERS ARE NOT PERMITTED. DATA PRECEDING NEXT VALID IDENTIFIER IGNORED.

One of the characters of a two-character identifier has been omitted, e.g., *T* appears instead of *TI*. Again the correction process is accomplished in two steps. First create the desired two-character identifier at the point of the problem by usurping the position of the first character of the input data item.

```
*T*ALICE IN WONDERLAND
*TI*LICE IN WONDERLAND
```

Then append the desired identifier and input data item to the end of the input data set.

4. CARD XXXX CONTAINS TWO SUCCESSIVE IDENTIFIERS. THE FIRST ONE IS IGNORED.

An example of this is *TI**CA*. This type of error is corrected by replacing the two asterisks of the second identifier with arbitrary characters other than asterisks and appending a new title and corporate author at the end of the input data set (with the appropriate identifiers, of course).

5. CARD XXXX IS IGNORED BECAUSE IT DOES NOT BEGIN A NEW DATA SET.

This message is printed whenever a new input data set is not begun on a card following a card with two successive blank columns. It is most likely to occur in

conjunction with the format error numbered 2 above. It may also occur because of an error in indicating the beginning of a new data set or because of missing or misordered cards. If it occurs in conjunction with format error 2, fixing that error will also correct this error.

6. CARD XXXX CONTAINS AN IMPROPERLY TERMINATED IDENTIFIER. DATA PRECEDING NEXT VALID IDENTIFIER IGNORED.

The program expected to find an asterisk terminating a two-character identifier and it did not. This is corrected by supplying the expected asterisk and adding the same identifier and its input data item at the end of the input data set, e.g.,

```
*TIALICE IN WONDERLAND
*TI*LICE IN WONDERLAND
```

etc.

7. DATA SET HAS NO ACCESSION NUMBER OR AD NUMBER

Every data set must have a unique accession number (or AD number if no accession number is used) before it can be properly processed. The previous input data set has none. One must be supplied.

8. XY IS NOT RECOGNIZED. THE FOLLOWING DATA CAN NOT BE PROCESSED

XY are two characters which should be a two-character identifier which is recognized by the program but is not. The error is corrected by replacing one or both of the characters with correct characters as required.

The basic technique used to correct errors 2-6 is applicable to situations not explicitly discussed that can arise from compound errors. First, fix the format error so that the program will not consider it an error. Second, add the needed identifiers and input data items to the end of the input data set.

7. MAINTAINING THE DATA BASE

A data base is not generally static. The collection of documents it describes gains and loses members and individual data items require changes to correct errors, or to add or delete information. Program ADM specifically performs all these functions, viz.,

1. new data sets may be added
2. old data sets may be deleted
3. individual data items within a data set may be added, deleted or replaced

Maintaining the data base consists of using these functions to update an existing data base. All three functions may be performed on the data base in any run.

When an input data set is to be added to the data base the action indicating character is A. The rules governing the addition of input data sets are easily stated.

1. The input data set must have an accession number (or AD number if no accession number is used) which is not already in the data base.

2. The input data set may additionally contain any other combination of input data items, provided.

3. That total storage for the resulting data set does not exceed 60 computer words (54 for data items, six preceding the data items).

If an attempt is made to add an input data set to the data base and its accession number (or AD number if no accession number is used) is a duplication of one existing in the data base, the input data set is rejected with the error message[13]

---

13. The error messages discussed in this section appear in the card by card output of the input data sets. They can be printed only when a new data base is being constructed. The error message will follow the listing of the applicable input data set, but there may be as many as 127 other data sets listings in between because the error can not be discovered when the input data set is initially processed (see section 8.1).

ACCESSION NUMBER OR AD NUMBER IS CURRENTLY IN THE DATA BASE
AND CAN NOT BE DUPLICATED

To delete a data set from the data base the action indicating character is D. The only input data item needed on the card is the accession number (or AD number if no accession number is used). If an attempt is made to delete a nonexistent data set, one of the following error messages is printed as appropriate.

ACCESSION NUMBER _____ IS NOT IN DATA BASE
AND CAN NOT BE DELETED

AD NUMBER _____ IS NOT IN DATA BASE AND CAN
NOT BE DELETED

Input data sets designed to modify existing data sets in the data base use M as the action indicating character. A data set may be modified by exchanging one or more input data items for the corresponding data items already existing in the data base. One or more data items may also be added to the data set or existing data items may be deleted. Deletion is accomplished by following the two-character input data item identifier by a minus sign, as *TI*-, which would delete the title. Additions and exchanges are both accomplished simply by including the input data item and its two character identifier in the input data set. The data set to be modified is identified in the input data set by the appearance of the accession number (or the AD number if no accession number is used).

In the event that an attempt is made to modify a data set not in the data base, one of the following error messages is printed as appropriate.

ACCESSION NUMBER _____ IS NOT IN DATA
BASE AND CAN NOT BE MODIFIED

AD NUMBER _____ IS NOT IN DATA BASE AND
CAN NOT BE MODIFIED

Should any of the format errors numbered 2-8 in section 6 be detected in the course of creating a new data base the following error message is printed

THIS DATA SET MUST BE CORRECTED BEFORE IT WILL MODIFY THE
DATA BASE IN THE MANNER DESIRED

This is simply a reminder that the input data set containing the error will in no way be reflected in the new data base. In the event that a format error of the type numbered 1 in section 6 is encountered this message is not printed, but it must be kept in mind that the incorrectly terminated input data set referred to in the error message also will not be reflected in the new data base. It is best to insure that no such errors remain in the input data sets when they are used to create a new data base.

8. PROGRAM ADM

In its present configuration Program ADM requires about 57,000 octal words of core in which to execute. It requires six files: INPUT, OUTPUT, an auxiliary output file (TAPE3), a file for the old data base (TAPE1 = OLDFIL), a file for the new data base (TAPE2 = NEWFIL), and a scratch file (TAPE4). ADM is written entirely in CDC FORTRAN Extended Version 4. It uses the .AND. (logical product) and .OR. (logical sum) masking expressions as well as the SHIFT function to do left circular bit shifting operations. Additionally much of the coding is necessarily predicated on 60 bit words and a particular set of six bit character representations. However, an experienced programmer should be able to convert the program to another language and/or computer without excessive difficulty in most cases.

8.1 THE MAIN PROGRAM

The main program reads a single card at the beginning of each run which determines if a new data base is to be created, if an old data base exists as input, and whether or not the page per document printout is to be generated. If the run is not to generate a new data base, the program simply proceeds to call subroutine DECODE enough times for the subroutine to read and check the format of all the input data sets, and print the specified output. DECODE is called once for each recognizable input data set. A maximum of 128 input data sets may be processed in the loop calling DECODE before the main program must recycle and reinitialize the loop.[14]

In the event that a new data base is to be generated and no old data base is input to the program for updating, the main program stores the first 128 input data sets read by DECODE[15] in the exact form in which they will appear in the data base.

---

14. The limit of 128 is related to the maximum number of input data sets which the program can store before having to dump them on to a mass storage device. Input data sets are not stored unless a new data base is to be created, but the limit is always in force.

15. If fewer than 128 input data sets are being processed, then it uses only that number.

The next step is to order the stored input data sets alphanumerically by their accession numbers (or AD number if no accession number is used). This is done by subroutine SORT. Once ordered, the input data sets are written on NEWFIL (=TAPE2) in alphanumeric order. If more input data sets are processed in the same run, the program continues by treating the original 128 as an old data base to be updated.[16]

When a new data base is to be generated by updating an existing data base, the procedure is necessarily different from that just described. The differences arise, however, only after the 128 input data sets have been stored and ordered by subroutine SORT. The updating of the old data base then takes place by merging the input data sets with the old data base data sets. This is accomplished by always comparing the alphanumerically smallest unprocessed member of each list to determine what actions are to be taken. These actions depend upon which member is smaller, if either, and what the nature of the action specifying character of the input data set being examined is. The possible actions are:

1. If the unique data set identifier from the old data base is alphanumerically smaller than the one from the input data set, the old data set is transferred to the new (intermediate or final) data base.

2. If the input data set identifier is smaller than the old data set identifier and the input data set is to be added to the new data base, it is transferred to the new data base. If it is to have deleted or modified an existing old data set an error has occurred as that data set was not present in the old data base. Such an error removes the input data set from further consideration without aborting the formation of the new data base. An appropriate error message is printed (see section 7).

3. If the input data set and the old data set are alphanumerically equal and the action to be performed by the input data set is deletion of an old data set, then the old data set is deleted and nothing is added to the new data base. If the action to be performed is modification rather than deletion, then the requested modifications are made and the modified old data set is transferred to the new data

16. Before the final new data base is created, new intermediate data bases are created for every 128 input data sets processed.

base. An error occurs when the action specified by the input data set is addition to the new data base, since the data base already contains a document with that identifier. The error does not abort the creation of the new data base, but that input data set is removed from further consideration. An appropriate error message is printed (see section 7).

In general one of the lists will become exhausted before the other. When this occurs the program simply proceeds to run through the other list to its conclusion. When both lists have been completely processed, a new intermediate or final data base exists. It is final if no more input data sets remain to be processed. Otherwise it is an intermediate data base and the program will cycle additional times to incorporate all the remaining input data sets. At each additional cycle the newly created intermediate data base takes the role of the old existing data base to be merged with the next group of input data sets.

When the final new data base has been constructed, a check is made to determine if it is residing on the correct file to be saved (TAPE2 = NEWFIL). If it is not, it is copied from the scratch file (TAPE4) to TAPE2.

8.2 SUBROUTINE DECODE

Subroutine DECODE reads the cards of the input data sets. It is called from the main program each time a data set is to be read, and it reads and processes cards one at a time until an input data set has been completed. Each card image is written on TAPE3 after being read, later to be copied to the output file.

Each card that is read is examined character by character to determine each character's significance to the program. A character may be part of an input data item, part of a two-character identifier, the action indicating character (A, D, M, or E), or an asterisk or colon serving in the role of a delimiter (see section 4). Characters which are part of input data items or two-character identifiers are stored appropriately as is the action indicating character. Delimiting asterisks or colons cause the input data items, two-character identifiers and the action indicating character to be correctly stored.

As each input data item is completed its two-character identifier is checked for validity. Valid identifiers were given earlier in table 1. (They can be

changed simply by changing the data statement for the NAME array.) An invalid identifier, one not matching any allowed by the program, causes the input data item to be discarded. A valid identifier causes further processing of the input data item. This processing is composed of making the punctuation spacing conform to the set of rules given previously in table 3, and moving the item into the IDATA array. Provision has been made for introducing format tests for individual input data items, but no such tests are currently performed. Its position in the IDATA array and the number of computer words required for storage are stored in the LENLOC array according to the two-character identifier. LENLOC is a 2x20 array. The first index takes the value "1" when the length of the input data item is being specified; "2" when the location of the first word of the input data item in the IDATA array is being specified. The second index signifies the nature of the input data according to the numbers used in table 1.

Should an identifier be used more than once in the same input data set, the input data items related to each usage are stored in the IDATA array but only the position and length of the last to be encountered are retained. This permits easier corrections of errors in input data set cards than would otherwise be possible (see section 4). After all input data items in a data set have been stored in the IDATA array, they are rearranged in the order of table 1 in the IODATA array. The LENLOC array is adjusted to reflect this new order. During this rearrangement any superseded input data items are discarded.

The page per document output array is printed from subroutine DECODE.

8.3 SUBROUTINE ERROR

Subroutine ERROR is primarily a source of error messages related to machine detectable errors. Associated with the detection of certain errors is a requirement for reinitialization of some quantities. This is also handled in subroutine ERROR. Error messages are written on TAPE3 and hence appear in the card image listing.

8.4 SUBROUTINE SORT

Subroutine SORT is used to order groups of input data sets by their accession

number (or AD number if no accession number is used)[17] so that they may be integrated into the existing data base efficiently. The ordering is done alphanumerically (see section 3) and only the first word of the accession number or AD number affects the ordering process.[18] SORT is designed to order a maximum of 128 input data sets in any call to the subroutine.[19]

At the beginning of the ordering process each entry in the LIST array is treated as a one accession number subset of the set of 128 accession numbers being ordered. During the first step of the ordering process, the first and second subsets of one accession number each (as they occur in the LIST array) are merged to form a new ordered subset containing two accession numbers. Likewise the third and fourth, fifth and sixth, etc., are merged resulting in 128/2 = 64 ordered subsets of two accession numbers each. In this context "merging" is the process of combining two ordered lists into a single ordered list by repeatedly comparing the smallest remaining element of each list and choosing the smallest of these as the next element of the combined list.

Step two causes the first and second two-member subsets to be merged forming a four-member ordered subset. Again the remaining pairs (third and fourth, etc.) of two-member subsets are similarly merged to form 64/2 = 32 ordered subsets of four accession numbers. The general process is continued until a single subset consisting of 128 ordered accession numbers results. The process is illustrated in the "tournament chart" of figure 6.

In actual practice no array is ever created which contains the elements of the subsets of the LIST array in proper alphanumeric order. Instead the order is kept in the INAD and INADS arrays. The order from the previous ordering step is

---

17. For convenience, the use of the words "accession number" during the remainder of the discussion of subroutine SORT will be taken to imply the phrase "or AD number if no accession number is used."

18. A good reason for using no more than 10 characters (one computer word) for the accession number.

19. This maximum could easily be increased by powers of two at the expense of increasing the program size. A decrease in running time for many cases could be expected. Currently about 17,000 octal storage locations are involved in the ordering part of the program.

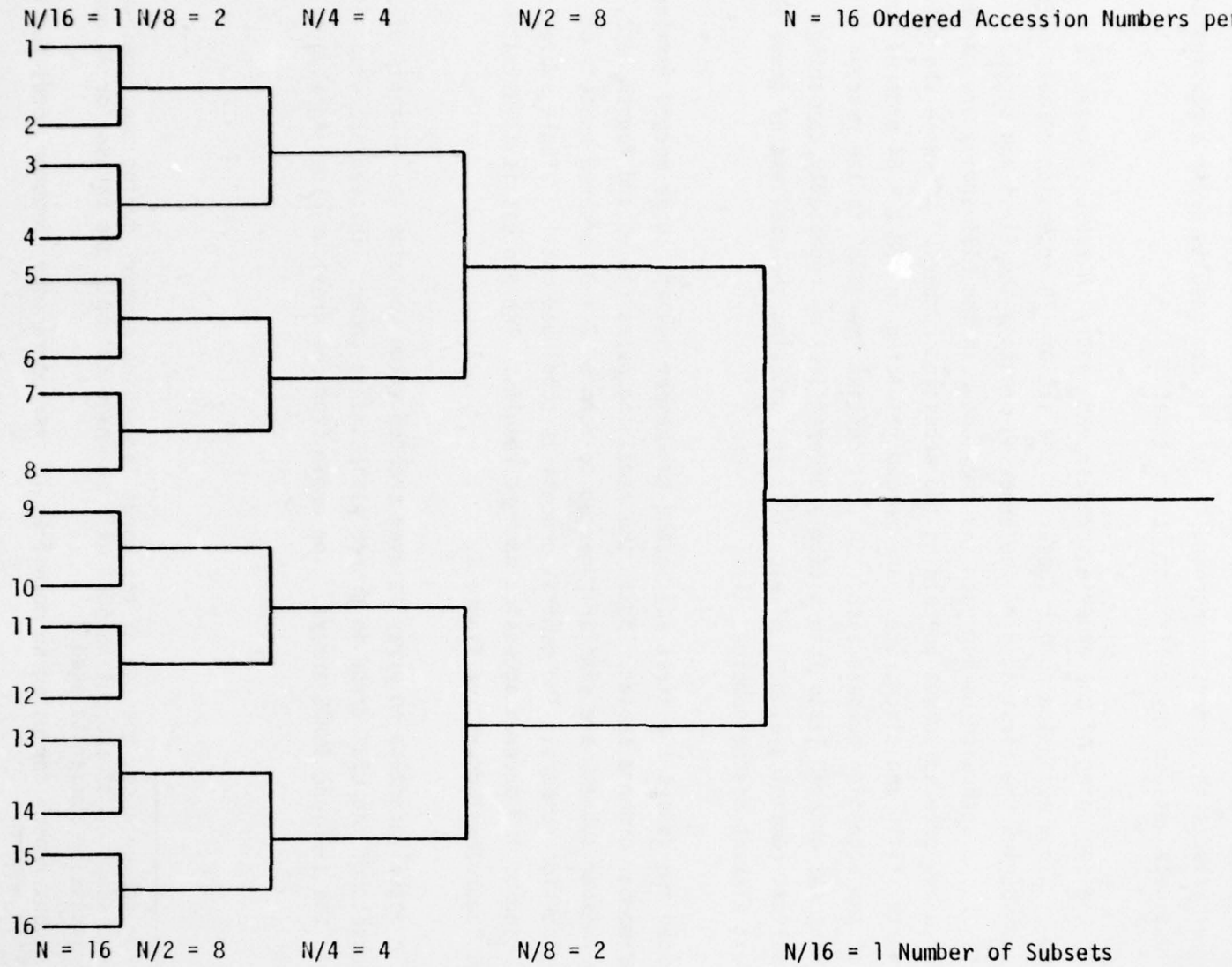


Figure 6. An Illustration of the Ordering Process Used in Subroutine SORT

kept in INADS, while the order for the current step is formed in INAD. At the conclusion of the entire ordering process, INADS(1) indicates the position of the alphanumerically smallest element of the LIST array; INADS(2) of the next smallest; etc. At the conclusion of preceding ordering steps the INADS array contains similar information for each subset. Thus for the next to the last ordering step, INADS(1) contains the position of the smallest element of the first 64 element subset of LIST, INADS(2) the second smallest, etc. At the same time INADS(65) contains the position of the smallest element of the second 64 element subset of LIST, INADS(66) the second smallest, etc.

The LIST array is preset in the main program to 10H;;^{bbbbbbbb} (^{b} = blank) before the accession numbers to be ordered are determined. In the event that fewer than 128 accession numbers are being ordered, say N, the preset value constitutes the final 128-N entries in the LIST array. They always order last, and as only the first N entries will be considered elsewhere in the program, they play no further role.

The ordering process which has just been described requires only that the total number of elements being ordered is expressable as an integral power of 2. When the SORT subroutine is initially called, the minimum power of 2, greater or equal to the number of accession numbers being ordered (and $\leq$ 128) is determined, and only that number of elements of the LIST array are ordered. This increases the efficiency of the subroutine when ordering less than 65 accession numbers.[20]

20. The subroutine as used in this program gains very little from this feature. However, SORT can be used as a general ordering routine and in other applications the increased efficiency may be desirable.

APPENDIX A
PROGRAM VARIABLE DEFINITIONS

IACT	Equivalent to IACTION
IACTION(I) (used without a subscript in subroutine DECODE)	Indicates the action required relative to the I-th input data set being processed: = 1RA (R indicates a right justified Hollerith constant with zero fill), add the document to the data base; = 1RD, delete the document from the data base; = 1RM, modify the document in the existing data base; and = 1RE, signifies the end of input.
IAN	The location in IODATA of the accession number (or AD number if no accession number exists) of the data set most recently read from the old data base.
IBLANK(I)	An array whose I-th member ($1 \leq I \leq 9$) contains (10-I) right justified blanks with zero fill.
IBUFF	A word in which characters from an input data item are assembled one by one.
ICHAR(I)	The 80 columns of an input card which has been read are stored in ICHAR, one character per word, right justified with zero fill.
ICHARPR	The previously examined character of the current input data set.
ICNTCRD	The number of input cards read.
ICONT	If ICONT = 1 there are additional characters to be examined from the current input data set. If ICONT = 0 the examination of the characters is terminated.
ICOUNT	An index used to specify the input data set currently eligible to interact with the old data set.
ICT	A word containing a single character of an input data set, all other bits zero.

ICTPR	The previously examined character in the process of correctly spacing punctuation, right justified with zero fill.
IDATA(I)	Intermediate storage of all punctuation processed data items (including duplicates) relative to the input data set being processed.
IDELETE(I)	The accession number (or AD number if no accession number exists) of the I-th data set deleted from the data base.
IDENT	A word in which the two-character identification codes are assembled right justified with zero fill.
IENDOLD	If IENDOLD = 1 all data sets on the old data base have been read. Otherwise unread data sets remain.
IFLAG	A flag which is set to unity when an error is detected by the program which prohibits an input data set from being added to a new data base.
IMASK(I)	For I = 1, ... , 10, a mask of 6 consecutive bits set to 1 (111111 or 77 octal), occupying bit positions 59-6(I-1) to 54-6(I-1).
INCR	The difference in length between a data item modifying the data base and the data item it is modifying.
INEW (Main Program)	The logical unit on which the new data base is written.
INEW (Subroutine Decode)	If INEW = 1 the card most recently read is the first card of an input data set. Otherwise the most recently read card is a continuation of an already begun input data set.
INFO(I)	Intermediate storage for the input data item being examined.
INPEND	If INPEND = 1 all input data sets have been read. Otherwise input data sets remain to be read.

IODATA(I)	Contains the I-th (modulo 128) input data set as processed by subroutine DECODE. IODATA(1) = length of processed input data set; IODATA(2) - IODATA(6) = table giving lengths and locations within IODATA of processed data items.
IOLD	The logical unit on which the old data base resides.
IORD	Equivalent to IORDER.
IORDER(I)	The accession number (or AD number if no accession number exists) of the I-th input data set processed this run, modulo 128.
IPOS	A counter which indicates where within a word a character should be stored.
IPRINT	IPRINT = 1 causes a one-page printout for each input data set processed by the program (see section 6). Otherwise this form of output is omitted.
IREPEAT	The sum of the processed lengths of all data items from the input data set being examined which have been replaced because of multiple use of the same two-character identifier within the input data set (see section 8.2).
IRUN	A variable which controls the placement of input data items in the IDATA and IODATA arrays.
ISTAR	A variable whose value is used in identifying the two-character identifiers in an input data set. ISTAR = 0 indicates that the program is searching for the first asterisk setting off an identifier (or the first of an adjacent pair - see section 5). ISTAR = 1 indicates that the search just described has been successful for the character just examined. ISTAR = 2, 3, and 4 simply count the characters examined from the time ISTAR became 1.
ISUM	An upper limit to various do-loops based on the length of the input data item being processed.

ISUMS	A printer control.
ITAPE(I, J)	Stores up to I = 128 IODATA arrays for use in modifying an existing data base or creating a new data base. All input data sets in ITAPE interact with the data base in one pass through the data base.
IWORD	Counts the number of computer words in the input data item being processed.
JCOUNT	Indicates the number of data sets deleted from the old data base.
JNFO(I)	Used as intermediate storage for the correct punctuation-spaced input data item being considered.
JPOS	A counter which indicates where within a word a character should be stored.
JWORD	Counts the number of words in the input data item being processed.
LAST	The length of the input data set being processed.
LELO(I, J)	Similar to LENLOC, LELO is used when a data set is being modified and two such arrays are required at the same time. LELO refers to the input data set containing the modifications.
LENLOC(I, J)	LENLOC(1, J) contains the number of computer words in the J-th data item. LENLOC(2, J) indicates the location of the first computer word of the data item in the IODATA array.
LISTORD(I)	The I-th word of LISTORD gives the location in the IORDER array of the I-th alphanumerically smallest element of that array. For all J, ITAPE (LISTORD (I), J) corresponds to IORDER (LISTORD(I)).
NAME(I)	An array containing all the two-character identification codes, one per word, right justified with zero fill.

NCHK	If NCHK = 1 the input data set being processed has at least an accession number or an AD number. If NCHK = 0 it has neither.
NEWFIL	If NEWFIL = 1 a new data base is to be created during the run. Otherwise a new data base is not created.
NWORD	The number of input data sets processed this run, modulo 128.
OLDFIL	If OLDFIL = 1 on old data base is being updated during the run. Otherwise no old data base is used.

APPENDIX B

PROGRAM LISTING

```
      PROGRAM ADM(INPUT,OUTPUT,OLDFIL,NEWFIL,TAPE3,TAPE4,TAPE1=OLDFIL,TA
     1PE2=NEWFIL)
C
C     + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
C
C     COMMENTS SET OFF BY ASTERISKS ARE IN FORCE FOR ALL STATEMENTS
C     FOLLOWING THE COMMENTS UP TO THE NEXT GROUP OF SUCH COMMENTS. A
C     COMMENT FOLLOWED BY A NUMBER IN PARENTHESES APPLIES TO THE
C     FOLLOWING STATEMENTS THROUGH THE STATEMENT NUMBERED WITH THAT
C     NUMBER. FOR THE COMPLETE DOCUMENTATION OF PROGRAM ADM SEE
C     DAS-TR-76-12 FORTRAN SOFTWARE FOR CREATING AND MAINTAINING A
C     LIBRARY CATALOGING SYSTEM ON SCIENTIFICALLY ORIENTED COMPUTERS
C     VOLUME I, PROGRAM ADM - THE DATA BASE.
C
C     PROGRAM ADM USES THE FORTRAN FUNCTIONS MASK(I) AND SHIFT(I,J).
C     MASK(I) FORMS A MASK OF I ONES IN BIT POSITIONS 59 THROUGH (60-I).
C     BIT POSITIONS ARE NUMBERED LEFT TO RIGHT FROM 59 TO 0. SHIFT(I,J)
C     AS USED IN THIS PROGRAM (J POSITIVE) RESULTS IN A LEFT END-AROUND
C     SHIFT OF THE WORD I BY J BITS. THE LOGICAL .AND. AND .OR.
C     OPERATORS ARE ALSO USED. THEY DO A BIT-BY-BIT LOGICAL SUM OR
C     PRODUCT OF THE QUANTITIES THEY CONNECT.
C
C     + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
C
      DIMENSION ITAPE(128,60), LENLOC(2,20), LISTORD(128), IORDER(128),
     1IACTION(128), LELO(2,20), IDELETE(128)
C
      COMMON /A/ IBUFF,ICHAR(80),IDENT,IPOS,IRUN,ISTAR,IWORD,NCHK,ICHARP
     1R,INFO(100),ICNTCRD,IORD,IODATA(100),IPRINT
      COMMON /B/ IFLAG,IMASK(10)
C
      INTEGER OLDFILE
C
      DATA IOLD/2/,INEW/1/,INPEND/0/
C
      I=I-1
C         DEFINE IMASK ARRAY (10)
      DO 10 I=1,10
      IMASK(I)=SHIFT(77B,(10-I)*6)
   10 CONTINUE
      WRITE (3,530)
      READ 520, OLDFILE,NEWFILE,IPRINT
      JCOUNT=0
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     INITIALIZATION REQUIRED BEFORE PROCESSING THE NEXT 128 (OR FEWER)
C     INPUT DATA SETS
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

```
C
   20 CONTINUE
      IF (INPEND.EQ.1) GO TO 480
      DO 30 I=1,128
      IORDER(I)=10H;;
   30 CONTINUE ,
      IENDOLD=0
C          DEFINE THE CORRECT LOGICAL UNIT FOR FILES OLDFIL AND NEWFIL.
C          OLDFIL IS INITIALLY TAPE1, LATER ALTERNATES BETWEEN TAPE2 AND
C          TAPE4. NEWFIL IS INITIALLY TAPE2, LATER ALTERNATES BETWEEN
C          TAPE4 AND TAPE2.
      ITEMP=IOLD
      IOLD=INEW
      INEW=ITEMP
      IF (OLDFILE.EQ.1) REWIND IOLD
      IF (NEWFILE.EQ.1) REWIND INEW
      NWORD=0
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     BEGIN LOOP TO PROCESS INPUT DATA SETS
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
      DO 110 I=1,128
   40 CONTINUE
C          CALL SUBROUTINE DECODE TO READ AND ORGANIZE ONE INPUT DATA SET
      CALL DECODE (IACTION(I),LENLOC)
C          HAS LAST INPUT DATA SET BEEN READ
      IF (IACTION(I).EQ.1RE) INPEND=1
      IF (IACTION(I).EQ.1RE.AND.NEWFILE.EQ.0) STOP 1
C          SKIP REMAINDER OF LOOP IF A NEW DATA BASE IS NOT BEING CREATED
      IF (NEWFILE.EQ.0) GO TO 110
C          CHECK IF ERRORS DETECTED BY THE PROGRAM IN THE INPUT DATA SET
C          ELIMINATE IT FROM FURTHER CONSIDERATION
      IF (IFLAG.NE.1) GO TO 50
      WRITE (3,540)
      GO TO 40
   50 CONTINUE
C           IF LAST INPUT DATA SET HAS BEEN READ AND NEW DATA BASE IS TO
C          BE CREATED, MERGE OLD DATA BASE AND INPUT DATA SETS
      IF (IACTION(I).EQ.1RE.AND.NEWFILE.EQ.1) GO TO 120
      NWORD=NWORD+1
C          REPLACE BLANKS WITH ZERO BINARY
      LAST=IODATA(1)
C          NO DATA SET MAY EXCEDE 60 WORDS
      IF (LAST.LE.60) GO TO 60
      PRINT 550
      IF (NEWFILE.EQ.1) STOP 2
   60 CONTINUE
```

```
C          REPLACE BLANKS WITH COLONS (ZERO BINARY) (70)
      DO 70 J=7,LAST
      DO 70 K=1,10
      ICT=SHIFT(IODATA(J),(K-1)*6).AND.IMASK(1)
      IF (ICT.EQ.1L ) IODATA(J)=IODATA(J).AND.SHIFT(MASK(54),(10-K)*6)
   70 CONTINUE
C          STORE INPUT DATA SET ACCESSION NUMBER (OR AD NUMBER) FOR LATER
C          ORDERING OF INPUT DATA SETS
      IORDER(I)=IODATA(LENLOC(2,1))
      IF (LENLOC(2,1).EQ.0) IORDER(I)=IODATA(LENLOC(2,2))
C          ENCODE LENLOC ARRAY (90)
      DO 90 K=1,5
      LLL=0
      DO 80 L=1,4
      KL=(K-1)*4+L
      LEN=LENLOC(1,KL).AND.77B
      LOC=LENLOC(2,KL).AND.7777B
      LLL=LLL.OR.(SHIFT(LEN,9+(4-L)*15).OR.SHIFT(LOC,(4-L)*15))
   80 CONTINUE
      IODATA(K+1)=LLL
   90 CONTINUE
C          STORE INPUT DATA SET JUST PROCESSED IN ITAPE ARRAY (100)
      DO 100 J=1,LAST
      ITAPE(I,J)=IODATA(J)
  100 CONTINUE
  110 CONTINUE
      IF (NEWFILE.EQ.0) GO TO 20
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     A NEW DATA BASE WILL BE PRODUCED. ORDER ALPHANUMERICALLY THE
C     INPUT DATA SETS STORED IN ITAPE TO FACILITATE THEIR INTEGRATION
C     INTO THE OLD DATA BASE OR FOR THE INITIAL CONSTRUCTION OF A DATA
C     BASE. WHEN INITIALLY CONSTRUCTING A DATA BASE, AN OLD DATA BASE
C     IS CONSIDERED TO EXIST AFTER THE FIRST 128 INPUT DATA SETS HAVE
C     BEEN PROCESSED.
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
  120 CONTINUE
      ICOUNT=1
C          ORDER NEW DATA SETS ALPHANUMERICALLY
      NWORDP=NWORD+1
      IF (NWORDP.GT.128) GO TO 140
      DO 130 I=NWORDP,128
      IORDER(I)=10L;;
  130 CONTINUE
  140 CONTINUE
      CALL SORT (NWORD,IORDER,LISTORD)
C          IF OLD DATA BASE EXISTS, MERGE INPUT DATA SETS JUST PROCESSED
```

```
C          WITH OLD DATA BASE
      IF (OLDFILE.EQ.1) GO TO 160
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     NO OLD DATA BASE EXISTS. WRITE THE ORDERED INPUT DATA SETS ON
C     LOGICAL UNIT INEW. NOTE THAT THIS FILE WILL BE CONSIDERED AN OLD
C     DATA BASE IF MORE INPUT DATA SETS ARE PROCESSED DURING THIS RUN.
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
      DO 150 I=1,NWORD
      LAST=ITAPE(LISTORD(I),1)
      WRITE (INEW) (ITAPE(LISTORD(I),J),J=1,LAST)
  150 CONTINUE
      OLDFILE=1
      IOLD=4
      GO TO 20
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     AN OLD DATA BASE EXISTS. MERGE THE ORDERED INPUT DATA SETS WITH
C     THE DATA SETS FROM THE OLD DATA BASE TO PRODUCE A NEW ORDERED DATA
C     BASE ON LOGICAL UNIT INEW.
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
  160 CONTINUE
C          HAVE ALL DATA SETS BEEN READ FROM OLDFIL
      IF (IENDOLD.EQ.1) GO TO 190
C          READ A DATA SET FROM OLDFIL
      READ (IOLD) N,(IODATA(L),L=2,N)
C          WAS END OF FILE READ
      IF (EOF(IOLD)) 170,180
C          END OF FILE WAS READ
  170 CONTINUE
      IENDOLD=1
      IAN=1
      IODATA(1)=777700000000000000008
      GO TO 190
C          END OF FILE WAS NOT READ, DATA SET WAS READ FROM OLDFIL
  180 CONTINUE
      IODATA(1)=N
C          DETERMINE LOCATION, IF ANY, OF OLD DATA SET ACCESSION NUMBER
C          IN IODATA
      IAN=IODATA(2).AND.SHIFT(MASK(9),54)
      IAN=SHIFT(IAN,15)
      IF (IAN.NE.0) GO TO 190
C          DATA SET HAS NO ACCESSION NUMBER, DETERMINE LOCATION OF AD
C          NUMBER
```

```
      IAN=SHIFT(IAN,30)
  190 CONTINUE
C         HAVE ALL DATA SETS FROM THE OLD DATA BASE AND ALL INPUT DATA
C         SETS BEEN PROCESSED
      IF (ICOUNT.GT.NWORD.AND.IENDOLD.EQ.1) GO TO 480
C         HAVE ALL INPUT DATA SETS BEEN PROCESSED
      IF (ICOUNT.GT.NWORD) GO TO 290
C         WILL CURRENTLY CONSIDERED INPUT DATA SET BE TESTED FOR
C         ADDITION TO, DELETION FROM, OR MODIFICATION OF THE DATA BASE
C         AT THIS TIME
      IACT=IACTION(LISTORD(ICOUNT))
      IORD=IORDER(LISTORD(ICOUNT))
C         IS INPUT DATA SET TO BE TESTED FOR ADDITION TO THE DATA BASE
C         AT THIS TIME
      IF (IACT.NE.IRA) GO TO 220
C         FIND THE APPROPRIATE TEST. DATA SET WITH SMALLEST ALPHANUMERIC
C         IDENTIFIER WILL BE WRITTEN ON LOGICAL UNIT INEW.
      IF (IORD.GE.0.AND.IODATA(IAN).GE.0) GO TO 200
      IF (IORD.LT.0.AND.IODATA(IAN).LT.0) GO TO 200
      IF (IORD.GE.0) GO TO 300
      GO TO 290
C       SMALLEST ALPHANUMERIC IDENTIFIER FOUND BY SUBTRACTION
  200 CONTINUE
      IF (IORD-IODATA(IAN)) 300,210,290
C         TWO DATA SETS IN DATA BASE MAY NOT HAVE THE SAME IDENTIFIER.
C         DO NOT CONSIDER INPUT DATA SET FURTHER.
  210 CONTINUE
      CALL ERROR (13)
      ICOUNT=ICOUNT+1
      GO TO 290
C         IS INPUT DATA SET TO BE TESTED FOR DELETION FROM THE DATA BASE
C         AT THIS TIME.
  220 CONTINUE
      IF (IACT.NE.IRD) GO TO 250
C         FIND THE APPROPRIATE TEST. WRITE DATA SET FROM OLD DATA BASE
C         ON LOGICAL UNIT INEW UNLESS DATA SET IDENTIFIERS MATCH, THEN
C         DELETE.
      IF (IORD.GE.0.AND.IODATA(IAN).GE.0) GO TO 230
      IF (IORD.LT.0.AND.IODATA(IAN).LT.0) GO TO 230
      IF (IORD.GE.0) GO TO 240
      GO TO 290
C         EQUAL IDENTIFIERS FOUND BY SUBTRACTION
  230 CONTINUE
      IF (IORD-IODATA(IAN)) 240,470,290
C         DATA SET TO BE DELETED IS NOT IN DATA BASE. DO NOT CONSIDER
C         INPUT DATA SET FURTHER.
  240 CONTINUE
      CALL ERROR (14)
      ICOUNT=ICOUNT+1
      GO TO 190
```

```
  250 CONTINUE
C          IS INPUT DATA SET TO BE TESTED FOR MODIFICATION OF THE DATA
C          BASE AT THIS TIME
      IF (IACT.NE.IRM) GO TO 280
C          FIND THE APPROPRIATE TEST WRITE DATA SET FROM OLD DATA BASE
C          ON LOGICAL UNIT INEW UNLESS DATA SET IDENTIFIERS MATCH, THEN
C          MODIFY BEFORE WRITING.
      IF (IORD.GE.0.AND.IODATA(IAN).GE.0) GO TO 260
      IF (IORD.LT.0.AND.IODATA(IAN).LT.0) GO TO 260
      IF (IORD.GE.0) GO TO 270
      GO TO 290
C          EQUAL IDENTIFIERS FOUND BY SUBTRACTION
  260 CONTINUE
      IF (IORD-IODATA(IAN)) 270,310,290
C          DATA SET TO BE MODIFIED IS NOT IN DATA BASE. DO NOT CONSIDER
C          INPUT DATA SET FURTHER.
  270 CONTINUE
      CALL ERROR (15)
      ICOUNT=ICOUNT+1
      GO TO 190
  280 CONTINUE
  290 CONTINUE
C          HAVE ALL DATA SETS FROM THE OLD DATA BASE AND ALL INPUT DATA
C          SETS BEEN PROCESSED
      IF (ICOUNT.GT.NWORD.AND.IENDOLD.EQ.1) GO TO 480
C          HAVE ALL DATA SETS FROM OLD DATA BASE BEEN READ
      IF (IENDOLD.EQ.1) GO TO 190
C          WRITE THE DATA SET FROM THE OLD DATA BASE ON LOGICAL UNIT INEW
      N=IODATA(1)
      WRITE (INEW) N,(IODATA(L),L=2,N)
      GO TO 160
C          WRITE THE INPUT DATA SET ON LOGICAL UNIT INEW
  300 CONTINUE
      N=ITAPE(LISTORD(ICOUNT),1)
      WRITE (INEW) N,(ITAPE(LISTORD(ICOUNT),L),L=2,N)
      ICOUNT=ICOUNT+1
      GO TO 190
C          MODIFY THE DATA SET FROM THE OLD DATA BASE AS INDICATED BY THE
C          INPUT DATA BASE (460)
C
C          DECODE DATA ITEM LENGTH AND LOCATION DATA FOR DATA SET FROM
C          OLD DATA BASE (320)
  310 CONTINUE
      DO 320 K=1,5
      DO 320 L=1,4
      KL=(K-1)*4+L
      LEN=SHIFT(IODATA(K+1),(L-1)*15).AND.MASK(6)
      LOC=SHIFT(IODATA(K+1),6+(L-1)*15).AND.MASK(9)
      LENLOC(1,KL)=SHIFT(LEN,6)
      LENLOC(2,KL)=SHIFT(LOC,9)
```

```
  320 CONTINUE
C          DECODE DATA ITEM LENGTH AND LOCATION DATA FOR INPUT DATA SET
C          (330)
      DO 330 K=1,5
      DO 330 L=1,4
      KL=(K-1)*4+L
      LEN=SHIFT(ITAPE(LISTORD(ICOUNT),K+1),(L-1)*15).AND.MASK(6)
      LOC=SHIFT(ITAPE(LISTORD(ICOUNT),K+1),6+(L-1)*15).AND.MASK(9)
      LELO(1,KL)=SHIFT(LEN,6)
      LELO(2,KL)=SHIFT(LOC,9)
  330 CONTINUE
C          DELETE OR MODIFY K-TH DATA ITEM IF NECESSARY. ACCESSION NUMBER
C          AND AD NUMBER (IF NO ACCESSION NUMBER) ARE EXCLUDED (440)
      DO 440 K=2,20
      IF (K.EQ.2.AND.LELO(1,1).EQ.0) GO TO 440
      IF (LELO(1,K).EQ.0) GO TO 440
      IF (ITAPE(LISTORD(ICOUNT),LELO(2,K)).EQ.1L-) LELO(1,K)=0
      INCR=LELO(1,K)-LENLOC(1,K)
      IF (INCR.EQ.0) GO TO 420
      N=N+INCR
C          FIND LOCATION OF FIRST WORD OF FIRST REPRESENTED DATA ITEM
C          AFTER K-TH DATA ITEM
      M=0
  340 CONTINUE
      M=M+1
      INITIAL=LENLOC(2,K+M)
      IF (INITIAL.EQ.0.AND.K+M.LE.20) GO TO 340
      IF (K+M.GT.20) GO TO 380
      LAST=IODATA(1)
      IF (INCR.LT.0) GO TO 360
C          INCR.GT.0, MOVE DATA ITEMS FOLLOWING DATA ITEM TO BE ADDED OR
C          MODIFIED TO MAKE APPROPRIATE STORAGE AVAILABLE
      DO 350 L=INITIAL,LAST
      IODATA(LAST+INITIAL+INCR-L)=IODATA(LAST+INITIAL-L)
  350 CONTINUE
      GO TO 380
C          INCR.LT.0, MOVE DATA ITEMS FOLLOWING DATA ITEM TO BE MODIFIED
C          TO REMOVE UNNEEDED STORAGE
  360 CONTINUE
      DO 370 L=INITIAL,LAST
      IODATA(L+INCR)=IODATA(L)
  370 CONTINUE
  380 CONTINUE
C          ARRANGEMENT OF DATA ITEMS IN IODATA TO ACCOMMADATE ADDITION OF
C          OR MODIFICATION TO K-TH DATA ITEM IS COMPLETE
      LENLOC(1,K)=LELO(1,K)
      IF (LENLOC(2,K).NE.0) GO TO 400
C          DATA ITEM IS BEING ADDED WHERE NONE PREVIOUSLY EXISTED. SET
C          LOCATION OF NEWLY ADDED DATA ITEM APPROPRIATELY
      KM=K-1
```

```
      DO 390 L=1,KM
      IF (LENLOC(1,L).NE.0) LENLOC(2,K)=LENLOC(2,L)+LENLOC(1,L)
  390 CONTINUE
C         INCR.NE.0, LOCATIONS OF ALL DATA ITEMS FOLLOWING MODIFIED DATA
C         ITEM MUST BE ADJUSTED
  400 CONTINUE
      IF (LELO(1,K).EQ.0) LENLOC(2,K)=0
      KP=K+1
      DO 410 L=KP,20
      IF (LENLOC(2,L).NE.0) LENLOC(2,L)=LENLOC(2,L)+INCR
  410 CONTINUE
C         REPLACE DATA ITEM FROM OLD DATA BASE WITH DATA ITEM FROM INPUT
C         DATA SET
  420 CONTINUE
      LAST=LELO(1,K)
      IF (LAST.EQ.0) GO TO 440
      DO 430 L=1,LAST
      IODATA(LENLOC(2,K)+L-1)=ITAPE(LISTORD(ICOUNT),LELO(2,K)+L-1)
  430 CONTINUE
  440 CONTINUE
C         REENCODE LENLOC
      DO 460 K=1,5
      LLL=0
      DO 450 L=1,4
      KL=(K-1)*4+L
      LEN=LENLOC(1,KL).AND.77B
      LOC=LENLOC(2,KL).AND.777B
      LLL=LLL.OR.(SHIFT(LEN,9+(4-L)*15).OR.SHIFT(LOC,(4-L)*15))
  450 CONTINUE
      IODATA(K+1)=LLL
  460 CONTINUE
      WRITE (INEW) N,(IODATA(K),K=2,N)
      ICOUNT=ICOUNT+1
      GO TO 160
C         DELETE THE LAST DATA SET READ
  470 CONTINUE
      IDELETE(JCOUNT)=IORDER(LISTORD(ICOUNT))
      ICOUNT=ICOUNT+1
      JCOUNT=JCOUNT+1
      GO TO 160
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     PROCESSING OF THE LAST-READ GROUP OF INPUT DATA SETS IS COMPLETE.
C     READ MORE INPUT DATA SETS IF APPROPRIATE OR INSURE NEW, COMPLETED
C     DATA BASE RESIDES ON CORRECT LOGICAL UNIT FOR CATALOGING.
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
  480 CONTINUE
```

```
      IF (OLDFILE.EQ.1) REWIND IOLD
      IF (NEWFILE.EQ.1) REWIND INEW
      IF (IOLD.EQ.1) IOLD=4
C         ARE THERE MORE INPUT DATA SETS TO BE READ
      IF (INPEND.NE.1) GO TO 20
C         IS NEW DATA BASE ON TAPE2
      IF (INEW.EQ.2) GO TO 510
C         COPY TAPE4 (SCRATCH) TO TAPE2 (NEWFIL) FOR CATALOGING
      DO 500 L=1,100000
      READ (INEW) N,(IODATA(K),K=2,N)
      IF (EOF(INEW)) 510,490
  490 CONTINUE
      WRITE (IOLD) N,(IODATA(K),K=2,N)
  500 CONTINUE
C         NEW DATA BASE RESIDES ON TAPE2
  510 CONTINUE
      REWIND INEW
      REWIND IOLD
C
  520 FORMAT (3I5)
  530 FORMAT (1H1)
  540 FORMAT (/56H THIS DATA SET MUST BE CORRECTED BEFORE IT WILL MODIFY
     1 T,35HHE DATA BASE IN THE MANNER DESIRED./)
  550 FORMAT (1H1,6H *****,32H INPUT DATA SET EXCEEDS 60 WORDS)
      END
```

```
      SUBROUTINE DECODE (IACTION,LENLOC)
C
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     SUBROUTINE DECODE READS ONE INPUT DATA SET EVERY TIME IT IS CALLED
C     BY PROGRAM ADM. CARDS ARE READ UNDER AN 80R1 FORMAT AND EXAMINED
C     CHARACTER BY CHARACTER TO DETERMINE WHICH DATA ITEM IS BEING
C     PROCESSED, ITS CONTENTS AND LENGTH. DATA ITEMS ARE IDENTIFIED BY
C     TWO-CHARACTER CODES WHICH APPEAR ON THE CARDS AS*XY* WHERE XY
C     REPRESENTS ONE OF THE ITEMS IN ARRAY NAME. ERRORS IN THE PLACEMENT
C     OR CONTENT OF INFORMATION ON THE CARDS ARE CHECKED FOR AND NOTED
C     BY THE PROGRAM AS SPECIFIED BY THE CODING.
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C
      DIMENSION IDATA(200), LENLOC(2,20), NAME(20), IBLANK(9), JNFO(100)
C
      COMMON /A/ IBUFF,ICHAR(80),IDENT,IPOS,IRUN,ISTAR,IWORD,NCHK,ICHARP
     1R,INFO(100),ICNTCRD,IORD,IODATA(100),IPRINT
      COMMON /B/ IFLAG,IMASK(10)
C
      DATA NAME/2RAN,2RAD,2RTI,2RCN,2RPA,2RCA,2RRN,2RDD,2RDR,2RRE,2RRS,2
     1RCL,2RSA,2RDI,2RDS,2RCM,2RFM,2RKW,2RAB/
      DATA IBLANK/9R          ,8R         ,7R        ,6R       ,5R      ,
     14R     ,3R    ,2R   ,1R /
      DATA ICNTCRD/0/
C
C
C
C          INITIALIZE PARAMETERS FOR NEW INPUT DATA SET
      DO 10 I=1,100
      INFO(I)=JNFO(I)=0
   10 CONTINUE
      IBUFF=0
      ICHARPR=1R
      ICONT=0
      IDENT=0
      IFLAG=0
      IPOS=0
      IREPEAT=0
      IRUN=7
      ISTAR=0
      IWORD=1
      NCHK=0
      DO 20 I=1,40
      LENLOC(I)=0
   20 CONTINUE
C          INITIALIZE PARAMETERS FOR CONTINUATION OF OLD DATA SET
```

```
   30 CONTINUE
      INEW=0
      ICOL=1
      READ 530, (ICHAR(I),I=1,80)
      ICNTCRD=ICNTCRD+1
      WRITE (3,760) ICNTCRD,(ICHAR(I),I=1,80)
C          DOES THE PREVIOUSLY READ CARD BEGIN A NEW INPUT DATA SET
      IF (ICHAR(1).EQ.1R*.AND.ICHAR(3).EQ.1R*) INEW=1
      IF (INEW.EQ.0) GO TO 40
      IACTION=ICHAR(2)
C          HAS END OF DATA INDICATOR BEEN READ
      IF (INEW.EQ.1.AND.ICHAR(2).EQ.1RE) RETURN
      IF (ICHAR(2).NE.1RA.AND.ICHAR(2).NE.1RD.AND.ICHAR(2).NE.1RM) CALL
     1ERROR (8)
   40 CONTINUE
      IF (INEW.EQ.1.AND.ICONT.EQ.1) CALL ERROR (1)
      IF (INEW.EQ.0.AND.ICONT.EQ.0) GO TO 50
      GO TO 60
C          A NEW INPUT DATA SET WAS EXPECTED BUT NOT FOUND
   50 CONTINUE
      CALL ERROR (5)
      GO TO 30
C          NEW INPUT DATA SET INDICATED OR OLD ONE CONTINUES AS EXPECTED
   60 CONTINUE
      ICONT=1
      IF (INEW.EQ.1) ICOL=3
C
C        EXAMINE CARD CHARACTER BY CHARACTER, IDENTIFYING AND BUILDING
C          INPUT DATA ITEMS (480)
C
      DO 480 I=ICOL,72
C          TWO SUCCESSIVE BLANKS TERMINATE CURRENT DATA SET
      IF (ICHARPR.EQ.1R .AND.ICHAR(I).EQ.1R ) ICONT=0
      IF (ICONT.EQ.0) GO TO 90
C          CURRENT INPUT DATA SET NOT YET TERMINATED
      ICHARPR=ICHAR(I)
      IF (ISTAR.EQ.0.AND.ICHAR(I).NE.1R*) GO TO 130
      IF (ISTAR.EQ.1.AND.ICHAR(I).NE.1R*) GO TO 70
C          TWO SUCCESSIVE * ARE USED TO PLACE AN * IN AN INPUT DATA ITEM
      IF (ISTAR.EQ.1.AND.ICHAR(I).EQ.1R*) GO TO 130
C          HAS FIRST * OF A PAIR JUST BEEN ENCOUNTERED
      IF (ISTAR.EQ.0.AND.ICHAR(I).EQ.1R*) ISTAR=1
      IF (ISTAR.EQ.3.AND.ICHAR(I).NE.1R*) GO TO 80
      IF (ISTAR.EQ.3.AND.ICHAR(I).EQ.1R*) CALL ERROR (3)
C          HAS TWO-CHARACTER IDENTIFIER BEEN DEFINED
      IF (ISTAR.EQ.4.AND.ICHAR(I).EQ.1R*) ISTAR=0
      IF (ISTAR.EQ.4.AND.ICHAR(I).NE.1R*) CALL ERROR (6)
      GO TO 480
C          END OF DATA ASSOCIATED WITH CURRENT TWO-CHARACTER IDENTIFIER
   70 CONTINUE
```

```
      IF (IPOS.GT.0) IBUFF=IBUFF.OR.IBLANK(IPOS)
      INFO(IWORD)=IBUFF
      IF (INFO(1).EQ.0.AND.IDENT.NE.0) CALL ERROR (4)
      IF (IPOS.EQ.0.AND.IDENT.NE.0) IWORD=IWORD-1
      ISTAR=ISTAR+1
      GO TO 140
C         ADD A CHARACTER TO THE TWO CHARACTER IDENTIFIER
   80 CONTINUE
      IDENT=IDENT.OR.SHIFT(ICHAR(I),(3-ISTAR)*6)
      ISTAR=ISTAR+1
      GO TO 480
C         END OF INPUT DATA SET INDICATED, IF MORE DATA FOLLOWS ON
C         PRESENT CARD THERE IS AN ERROR (100)
   90 CONTINUE
      IF (I.EQ.72) GO TO 110
      IP=I+1
      DO 100 II=IP,72
      IF (ICHAR(II).EQ.1R ) GO TO 100
      CALL ERROR (2)
      GO TO 110
  100 CONTINUE
C         DETERMINE IF IBUFF CONTAINS CHARACTERS WHICH BELONG WITH THE
C         LAST DATA ITEM (THE CHARACTER IN IBUFF FOR IPOS=1 IS A BLANK)
  110 CONTINUE
      IF (IPOS.NE.0.AND.IPOS.NE.1) GO TO 120
      IWORD=IWORD-1
      GO TO 140
C         BLANK FILL REMAINDER OF LAST WORD OF LAST INPUT DATA ITEM
C         ASSOCIATED WITH INPUT DATA SET BEING PROCESSED
  120 CONTINUE
      IBUFF=IBUFF.OR.IBLANK(IPOS)
      INFO(IWORD)=IBUFF
      GO TO 140
C         STORE ICHAR(I) IN IBUFF
  130 CONTINUE
      ISTAR=0
      IPOS=IPOS+1
      IBUFF=IBUFF.OR.SHIFT(ICHAR(I),(10-IPOS)*6)
      IF (IPOS.NE.10) GO TO 480
C         10 CHARACTERS HAVE BEEN STORED IN IBUFF
      INFO(IWORD)=IBUFF
      IBUFF=0
      IPOS=0
      IWORD=IWORD+1
      GO TO 480
C
C         A DATA ITEM HAS BEEN COMPLETED
C
  140 CONTINUE
C         IDENT=0 THE FIRST TWO-CHARACTER IDENTIFIER HAS BEEN FOUND
```

```
      IF (IDENT.EQ.0) GO TO 80
C         DOES PROGRAM RECOGNIZE TWO-CHARACTER IDENTIFIER (160)
      DO 150 J=1,19
      IAN=IODATA(2).AND.SHIFT(MASK(9),39)
      IF (IDENT.EQ.NAME(J)) GO TO 160
  150 CONTINUE
C         PROGRAM DOES NOT RECOGNIZE TWO-CHARACTER IDENTIFIER
      CALL ERROR (10)
      GO TO 460
  160 CONTINUE
C         CHECK CHARACTER SPACING AND INSERT OR DELETE BLANKS AS
C         REQUIRED (240)
      JPOS=0
      ICTPR=18
      DO 240 K=1,IWORD
      DO 230 L=1,10
      ICT=INFO(K).AND.IMASK(L)
      ICT=SHIFT(ICT,L*6)
      IF (ICT.GT.08.AND.ICT.LE.448.OR.ICT.EQ.558) GO TO 190
C         ICT IS NOT A LETTER, NUMERAL, OR BLANK
      IPB=1
      IF (ICT.EQ.518.OR.ICT.EQ.618) IPB=-1
      IF (ICT.GE.458.AND.ICT.LE.508.OR.ICT.EQ.548) IPB=0
      IF (IPB) 170,180,180
  170 CONTINUE
C         CHECK TO SEE THAT A BLANK PRECEDES ( OR [
      IF (ICTPR.EQ.558) GO TO 220
      JPOS=JPOS+1
      ISHIFT=(10-MOD(JPOS,10))*6
      JWORD=(JPOS-1)/10+1
      JNFO(JWORD)=JNFO(JWORD).OR.SHIFT(558,ISHIFT)
      ICTPR=558
      GO TO 220
C         CHECK TO SEE THAT NO BLANK PRECEDES +,-,*,/,=,,),],.,.,;,$,%
C         CHANGE  TO - (SINCE COLONS ARE LATER TREATED AS BLANKS)
  180 CONTINUE
      IF (ICT.EQ.0) ICT=468
      IF (ICTPR.NE.558.OR.JPOS.EQ.0) GO TO 220
      ISHIFT=(10-MOD(JPOS,10))*6
      JWORD=(JPOS-1)/10+1
      JNFO(JWORD)=JNFO(JWORD).AND.SHIFT(MASK(54),ISHIFT)
      JPOS=JPOS-1
      GO TO 220
  190 CONTINUE
C         ICT IS A LETTER, NUMERAL, OR BLANK
      IF (ICTPR.GT.08.AND.ICTPR.LE.448.OR.ICTPR.EQ.558) GO TO 220
      IPB=1
      IF (ICTPR.EQ.518.OR.ICTPR.EQ.618) IPB=-1
      IF (ICTPR.GE.458.AND.ICTPR.LE.508.OR.ICTPR.EQ.548) IPB=0
      IF (IPB) 200,200,210
```

```
  200 CONTINUE
C         CHECK TO SEE THAT NO BLANK FOLLOWS (,),+,-,*,/, OR =
      IF (ICT.EQ.55B) GO TO 230
      GO TO 220
  210 CONTINUE
C         CHECK TO SEE THAT A BLANK FOLLOWS ),],.....:,$, OR %
      IF (ICT.EQ.55B) GO TO 220
      JPOS=JPOS+1
      ISHIFT=(10-MOD(JPOS,10))*6
      JWORD=(JPOS-1)/10+1
      JNFO(JWORD)=JNFO(JWORD).OR.SHIFT(55B,ISHIFT)
  220 CONTINUE
C         STORE CHARACTER IN JNFO
      IF (JPOS.EQ.0.AND.ICT.EQ.55B) GO TO 230
      JPOS=JPOS+1
      ISHIFT=(10-MOD(JPOS,10))*6
      JWORD=(JPOS-1)/10+1
      JNFO(JWORD)=JNFO(JWORD).OR.SHIFT(ICT,ISHIFT)
      ICTPR=ICT
  230 CONTINUE
  240 CONTINUE
C         BLANK FILL LAST WORD OF INPUT DATA ITEM
      IF (MOD(JPOS,10).NE.0) JNFO(JWORD)=JNFO(JWORD).OR.IBLANK(MOD(JPOS,
     110))
C         INSURE AT LEAST TWO BLANKS TERMINATE INPUT DATA ITEM
      ICT=JNFO(JWORD).AND.SHIFT(MASK(12),12)
      IF (ICT.EQ.2R  ) GO TO 250
      JWORD=JWORD+1
      JNFO(JWORD)=10H
      GO TO 260
C         ENTRY TERMINATED BY TWO BLANKS WITHOUT ADDING A WORD OF
C         BLANKS. HOWEVER JNFO(JWORD) MAY BE ALL BLANKS AND UNNEEDED.
  250 CONTINUE
      IF (JWORD.EQ.1) GO TO 260
      ICT=JNFO(JWORD-1).AND.SHIFT(MASK(12),12)
      IF (ICT.NE.2R  ) GO TO 260
      JNFO(JWORD)=INFO(JWORD)=0
      JWORD=JWORD-1
      GO TO 250
C         INPUT DATA ITEM READY TO BE STORED IN IDATA ARRAY WITH LENGTH
C         AND LOCATION INFORMATION STORED IN LENLOC ARRAY (FOR TWO-
C         CHARACTER IDENTIFIERS APPEARING MORE THAN ONCE IN AN INPUT
C         DATA SET, THE LENLOC ARRAY CONTAINS INFORMATION ONLY ABOUT THE
C         LAST APPEARANCE.)
  260 CONTINUE
      IF (LENLOC(1,J).NE.0) IREPEAT=IREPEAT+LENLOC(1,J)
      LENLOC(1,J)=JWORD
      LENLOC(2,J)=IRUN
      ISUM=IRUN+JWORD-1
      DO 270 K=IRUN,ISUM
```

```
      IDATA(K)=JNFO(K+1-IRUN)
  270 CONTINUE
      IRUN=IRUN+JWORD
C         TEST FORMAT OF ACCESSION NUMBER
      IF (IDENT.NE.NAME(1)) GO TO 280
      NCHK=1
  280 CONTINUE
C
C         PERFORM TESTS ON FORMAT OF INPUT DATA ITEM (CURRENTLY NO TESTS
C         ARE PERFORMED, BUT AN AREA FOR SUCH TESTS HAS BEEN DEFINED FOR
C         EVERY INPUT DATA ITEM) (460)
C
C         TEST FORMAT OF AD NUMBER
      IF (IDENT.NE.NAME(2)) GO TO 290
      NCHK=1
  290 CONTINUE
C         TEST FORMAT OF TITLE
      IF (IDENT.NE.NAME(3)) GO TO 300
  300 CONTINUE
C         TEST FORMAT OF COPY NUMBER
      IF (IDENT.NE.NAME(4)) GO TO 310
  310 CONTINUE
C         TEST FORMAT OF PERSONAL AUTHOR
      IF (IDENT.NE.NAME(5)) GO TO 320
  320 CONTINUE
C         TEST FORMAT OF CORPORATE AUTHOR
      IF (IDENT.NE.NAME(6)) GO TO 330
  330 CONTINUE
C         TEST FORMAT OF REPORT NUMBER
      IF (IDENT.NE.NAME(7)) GO TO 340
  340 CONTINUE
C         TEST FORMAT OF DOCUMENT DATE
      IF (IDENT.NE.NAME(8)) GO TO 350
  350 CONTINUE
C         TEST FORMAT OF DATE RECEIVED
      IF (IDENT.NE.NAME(9)) GO TO 360
  360 CONTINUE
C         TEST FORMAT OF REQUESTOR
  370 CONTINUE
      IF (IDENT.NE.NAME(10)) GO TO 370
C         TEST FORMAT OF REQUESTING STUDY
      IF (IDENT.NE.NAME(11)) GO TO 380
  380 CONTINUE
C         TEST FORMAT OF CLASSIFICATION
      IF (IDENT.NE.NAME(12)) GO TO 390
  390 CONTINUE
C         TEST FORMAT OF SPECIAL ACCESS
      IF (IDENT.NE.NAME(13)) GO TO 400
  400 CONTINUE
C         TEST FORMAT OF DOWNGRADING INFORMATION
```

```
      IF (IDENT.NE.NAME(14)) GO TO 410
  410 CONTINUE
C         TEST FORMAT OF DOCUMENT STATUS
      IF (IDENT.NE.NAME(15)) GO TO 420
  420 CONTINUE
C         TEST FORMAT OF COMMENTS
      IF (IDENT.NE.NAME(16)) GO TO 430
  430 CONTINUE
C         TEST FORMAT OF FORMAT
      IF (IDENT.NE.NAME(17)) GO TO 440
  440 CONTINUE
C         TEST FORMAT OF KEY WORDS
      IF (IDENT.NE.NAME(18)) GO TO 450
  450 CONTINUE
C         TEST FORMAT OF ABSTRACT
      IF (IDENT.NE.NAME(19)) GO TO 460
  460 CONTINUE
C         INITIALIZATION REQUIRED FOR PROCESSING NEXT INPUT DATA ITEM
      IDENT=0
      DO 470 J=1,JWORD
      INFO(J)=JNFO(J)=0
  470 CONTINUE
      IBUFF=0
      IPOS=0
      IWORD=1
C         ISTAR=2 LAST PROCESSED INPUT DATA ITEM TERMINATED BY FINDING
C         NEXT TWO CHARACTER IDENTIFIER
      IF (ISTAR.EQ.2) GO TO 80
C         HAS END OF INPUT DATA SET BEEN FOUND
      IF (ICONT.EQ.0) GO TO 490
  480 CONTINUE
C         READ ANOTHER CARD BELONGING TO THIS INPUT DATA SET
      GO TO 30
  490 CONTINUE
C
C         ALL INPUT DATA ITEMS IN INPUT DATA SET HAVE BEEN PROCESSED
C
      IF (NCHK.EQ.0) CALL ERROR (7)
C         DELETE ERRONEOUS, REPEATED INPUT DATA ITEMS FROM IDATA ARRAY
C         TO FORM IODATA ARRAY. ADJUST LENLOC ARRAY. (510)
      IODATA(1)=IRUN-IREPEAT-1
      IRUN=7
      DO 510 K=1,20
      LOC=LENLOC(2,K)
      IF (LOC.EQ.0) GO TO 510
      LENLOC(2,K)=IRUN
      IWORD=LENLOC(1,K)
      ISUM=IRUN+IWORD-1
      DO 500 L=IRUN,ISUM
      IODATA(L)=IDATA(LOC+L-IRUN)
```

```
  500 CONTINUE
      IRUN=IRUN+IWORD
  510 CONTINUE
C         IS PAGE/INPUT DATA SET PRINTOUT TO BE EXERCISED
      IF (IPRINT.EQ.0) RETURN
C
C         PRINT DOCUMENT INFORMATION INPUT DATA ITEM BY INPUT DATA ITEM
C
      PRINT 540
      PRINT 550, IACTION
C         PRINT ACCESSION NUMBER
      LOC=LENLOC(2,1)
      ISUM=LOC+LENLOC(1,1)-1
      IF (LOC.EQ.0) PRINT 560
      IF (LOC.GT.0) PRINT 560, (IODATA(K),K=LOC,ISUM)
C         PRINT AD NUMBER
      LOC=LENLOC(2,2)
      ISUM=LOC+LENLOC(1,2)-1
      IF (LOC.EQ.0) PRINT 570
      IF (LOC.GT.0) PRINT 570, (IODATA(K),K=LOC,ISUM)
C         PRINT TITLE
      LOC=LENLOC(2,3)
      ISUM=LOC+LENLOC(1,3)-1
      IF (LOC.EQ.0) PRINT 580
      IF (LOC.GT.0) PRINT 580, (IODATA(K),K=LOC,ISUM)
C         PRINT COPY NUMBER
      LOC=LENLOC(2,4)
      ISUM=LOC+LENLOC(1,4)-1
      IF (LOC.EQ.0) PRINT 600
      IF (LOC.GT.0) PRINT 600, (IODATA(K),K=LOC,ISUM)
C         PRINT PERSONAL AUTHOR
      LOC=LENLOC(2,5)
      ISUM=LOC+LENLOC(1,5)-1
      IF (LOC.EQ.0) PRINT 610
      IF (LOC.GT.0) PRINT 610, (IODATA(K),K=LOC,ISUM)
C         PRINT CORPORATE AUTHOR
      LOC=LENLOC(2,6)
      ISUM=LOC+LENLOC(1,6)-1
      IF (LOC.EQ.0) PRINT 620
      IF (LOC.GT.0) PRINT 620, (IODATA(K),K=LOC,ISUM)
C         PRINT REPORT NUMBER
      LOC=LENLOC(2,7)
      ISUM=LOC+LENLOC(1,7)-1
      IF (LOC.EQ.0) PRINT 630
      IF (LOC.GT.0) PRINT 630, (IODATA(K),K=LOC,ISUM)
C         PRINT DOCUMENT DATE
      LOC=LENLOC(2,8)
      ISUM=LOC+LENLOC(1,8)-1
      IF (LOC.EQ.0) PRINT 640
      IF (LOC.GT.0) PRINT 640, (IODATA(K),K=LOC,ISUM)
```

```
C          PRINT DATE RECEIVED
      LOC=LENLOC(2,9)
      ISUM=LOC+LENLOC(1,9)-1
      IF (LOC.EQ.0) PRINT 650
      IF (LOC.GT.0) PRINT 650, (IODATA(K),K=LOC,ISUM)
C          PRINT REQUESTOR
      LOC=LENLOC(2,10)
      ISUM=LOC+LENLOC(1,10)-1
      IF (LOC.EQ.0) PRINT 660
      IF (LOC.GT.0) PRINT 660, (IODATA(K),K=LOC,ISUM)
C          PRINT REQUESTING STUDY
      LOC=LENLOC(2,11)
      ISUM=LOC+LENLOC(1,11)-1
      IF (LOC.EQ.0) PRINT 670
      IF (LOC.GT.0) PRINT 670, (IODATA(K),K=LOC,ISUM)
C          PRINT CLASSIFICATION
      LOC=LENLOC(2,12)
      ISUM=LOC+LENLOC(1,12)-1
      IF (LOC.EQ.0) PRINT 680
      IF (LOC.GT.0) PRINT 680, (IODATA(K),K=LOC,ISUM)
C          PRINT SPECIAL ACCESS
      LOC=LENLOC(2,13)
      ISUM=LOC+LENLOC(1,13)-1
      IF (LOC.EQ.0) PRINT 690
      IF (LOC.GT.0) PRINT 690, (IODATA(K),K=LOC,ISUM)
C          PRINT DOWNGRADING INFORMATION
      LOC=LENLOC(2,14)
      ISUM=LOC+LENLOC(1,14)-1
      IF (LOC.EQ.0) PRINT 700
      IF (LOC.GT.0) PRINT 700, (IODATA(K),K=LOC,ISUM)
C          PRINT DOCUMENT STATUS
      LOC=LENLOC(2,15)
      ISUM=LOC+LENLOC(1,15)-1
      IF (LOC.EQ.0) PRINT 710
      IF (LOC.GT.0) PRINT 710, (IODATA(K),K=LOC,ISUM)
C          PRINT COMMENTS
      LOC=LENLOC(2,16)
      ISUM=LOC+LENLOC(1,16)-1
      IF (LOC.EQ.0) PRINT 720
      IF (LOC.GT.0) PRINT 720, (IODATA(K),K=LOC,ISUM)
C          PRINT FORMAT
      LOC=LENLOC(2,17)
      ISUM=LOC+LENLOC(1,17)-1
      IF (LOC.EQ.0) PRINT 730
      IF (LOC.GT.0) PRINT 730, (IODATA(K),K=LOC,ISUM)
C          PRINT KEY WORDS
      LOC=LENLOC(2,18)
      ISUM=LOC+LENLOC(1,18)-1
      IF (LOC.EQ.0) PRINT 740
      IF (LOC.GT.0) PRINT 740, (IODATA(K),K=LOC,ISUM)
```

```
C          PRINT ABSTRACT
      LOC=LENLOC(2,19)
      ISUMS=ISUM=LOC+LENLOC(1,19)-1
      IF (LOC.EQ.0) PRINT 750
      IF (ISUM-LOC.GT.8) ISUM=LOC+7
      IF (LOC.GT.0) PRINT 750, (IODATA(K),K=LOC,ISUM)
      IF (ISUMS-LOC.LE.8) GO TO 520
      LOC=LOC+8
      PRINT 590, (IODATA(K),K=LOC,ISUMS)
  520 CONTINUE
      RETURN
C
  530 FORMAT (80R1)
  540 FORMAT (1H1)
  550 FORMAT (2H *,R1,1H*/)
  560 FORMAT (/5H *AN*,4X,17HACCESSION NUMBER,5X,8A10)
  570 FORMAT (/5H *AD*,11X,10HAD NUMBER,5X,8A10)
  580 FORMAT (/5H *TI*,15X,6HTITLE,5X,8A10,5X,(/31X,8A10))
  590 FORMAT (31X,8A10)
  600 FORMAT (/5H *CN*,9X,12HCOPY NUMBER,5X,8A10)
  610 FORMAT (/5H *PA*,5X,16HPERSONAL AUTHOR,5X,8A10)
  620 FORMAT (/5H *CA*,4X,17HCORPORATE AUTHOR,5X,8A10,5X,(/31X,8A10))
  630 FORMAT (/5H *RN*,7X,14HREPORT NUMBER,5X,8A10)
  640 FORMAT (/5H *DD*,7X,14HDOCUMENT DATE,5X,8A10)
  650 FORMAT (/5H *DR*,7X,14HDATE RECEIVED,5X,8A10)
  660 FORMAT (/5H *RE*,11X,10HREQUESTOR,5X,8A10)
  670 FORMAT (/5H *RS*,4X,17HREQUESTING STUDY,5X,8A10)
  680 FORMAT (/5H *CL*,6X,15HCLASSIFICATION,5X,8A10)
  690 FORMAT (/5H *SA*,6X,15HSPECIAL ACCESS,5X,8A10)
  700 FORMAT (/5H *DI*,4X,17HDOWNGRADING INFO,5X,8A10)
  710 FORMAT (/5H *DS*,5X,16HDOCUMENT STATUS,5X,8A10)
  720 FORMAT (/5H *CM*,12X,9HCOMMENTS,5X,8A10)
  730 FORMAT (/5H *FM*,14X,7HFORMAT,5X,8A10)
  740 FORMAT (/5H *KW*,11X,10HKEY WORDS,5X,8A10)
  750 FORMAT (/5H *AB*,12X,9HABSTRACT,5X,8A10)
  760 FORMAT (5H CARD,I4,5X,80R1)
      END
```

```
      SUBROUTINE ERROR (N)
C
CCC   * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
CCC
CCC   SUBROUTINE ERROR HANDLES THE OUTPUT OF ERROR MESSAGES AND THE
CCC   SETTING OF PROGRAM PARAMETERS REQUIRED FOR PROPER CONTINUATION
CCC   OF THE REMAINDER OF THE PROGRAM. ALL OUTPUT FROM SUBROUTINE ERROR
CCC   IS ON TAPE3.
CCC
CCC   * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
      COMMON /A/ IBUFF,ICHAR(80),IDENT,IPOS,IRUN,ISTAR,IWORD,NCHK,ICHARP
     1R,INFO(100),ICNTCRD,IORD,IODATA(100),IPRINT
      COMMON /B/ IFLAG,IMASK(10)
      IFLAG=1
      PRINT 340, N
      GO TO (10,20,30,40,50,60,70,80,90,100,120,130,140,150,180), N

   10 CONTINUE
      WRITE (3,210) ICNTCRD
      IBUFF=0
      ICHARPR=1R
      IDENT=0
      IPOS=0
      IRUN=7
      ISTAR=0
      IWORD=1
      NCHK=0
      IFLAG=0
      RETURN
   20 CONTINUE
      WRITE (3,220) ICNTCRD
      RETURN
   30 CONTINUE
      INVALID=1
      ISTAR=0
      WRITE (3,230)
      RETURN
   40 CONTINUE
      WRITE (3,240) ICNTCRD
      RETURN
   50 CONTINUE
      WRITE (3,250) ICNTCRD
      RETURN
   60 CONTINUE
      ISTAR=0
      INVALID=1
      WRITE (3,260) ICNTCRD
      RETURN
   70 CONTINUE
```

```
      WRITE (3,270)
      RETURN
   80 CONTINUE
      RETURN
   90 CONTINUE
      RETURN
  100 CONTINUE
      IF (INVALID.EQ.0) GO TO 110
      INVALID=0
      RETURN
  110 CONTINUE
      WRITE (3,280) IDENT,(INFO(I),I=1,IWORD)
      RETURN
  120 CONTINUE
      RETURN
  130 CONTINUE
      RETURN
  140 CONTINUE
      WRITE (3,290) IORD
      RETURN
  150 CONTINUE
      DO 160 I=1,10
      ICT=IORD.AND.SHIFT(MASK(6),(11-I)*6)
      IF (ICT.EQ.0) IORD=IORD.OR.SHIFT(1L ,(11-I)*6)
  160 CONTINUE
      IAN=IODATA(2).AND.SHIFT(MASK(9),54)
      IAN=SHIFT(IAN,15)
      IF (IAN.EQ.0) GO TO 170
      WRITE (3,300) IORD
      RETURN
  170 CONTINUE
      WRITE (3,310) IORD
      WRITE (3,310) IORD
      RETURN
  180 CONTINUE
      DO 190 I=1,10
      ICT=IORD.AND.SHIFT(MASK(6),(11-I)*6)
      IF (ICT.EQ.0) IORD=IORD.OR.SHIFT(1L ,(11-I)*6)
  190 CONTINUE
      IAN=IODATA(2).AND.SHIFT(MASK(9),54)
      IAN=SHIFT(IAN,15)
      IF (IAN.EQ.0) GO TO 200
      WRITE (3,320) IORD
      RETURN
  200 CONTINUE
      WRITE (3,330) IORD
      RETURN
C
  210 FORMAT (/5H CARD,I4,44H BEGINS A NEW DATA SET FOLLOWING AN IMPROPE
     1R,22HLY TERMINATED DATA SET/)
```

```
220 FORMAT (/5H CARD,I4,44H CONTAINS TWO SUCCESSIVE BLANK COLUMNS FOLL
   1O,23HWED BY ADDITIONAL DATA./34H FOLLOWING CARDS WILL BE IGNORED U
   2,33HNTIL A NEW DATA SET IS INITIATED./)
230 FORMAT (/55H ONE CHARACTER IDENTIFIERS ARE NOT PERMITTED. DATA PRE
   1C,36HEDING NEXT VALID IDENTIFIER IGNORED.)
240 FORMAT (/5H CARD,I4,44H CONTAINS TWO SUCCESSIVE IDENTIFIERS. THE F
   1I,19HRST ONE IS IGNORED./)
250 FORMAT (/5H CARD,I4,44H IS IGNORED BECAUSE IT DOES NOT BEGIN A NEW
   1 ,9HDATA SET./)
260 FORMAT (/5H CARD,I4,44H CONTAINS AN IMPROPERLY TERMINATED IDENTIFI
   1E,48HR. DATA PRECEDING NEXT VALID IDENTIFIER IGNORED./)
270 FORMAT (/46H DATA SET HAS NO ACCESSION NUMBER OR AD NUMBER/)
280 FORMAT (/2H *,R2,1H*,43H IS NOT RECOGNIZED. THE FOLLOWING DATA CAN
   1 ,16HNOT BE PROCESSED/(10X,8A10))
290 FORMAT (/28H ACCESSION NUMBER/AD NUMBER ,A10,56H IS CURRENTLY IN T
   1HE DATA BASE AND CAN NOT BE DUPLICATED/)
300 FORMAT (/18H ACCESSION NUMBER ,A10,31HIS NOT IN DATA BASE AND CAN
   1NOT,11H BE DELETED/)
310 FORMAT (11H AD NUMBER ,A10,38HIS NOT IN DATA BASE AND CAN NOT BE D
   1EL,4HETED/)
320 FORMAT (/18H ACCESSION NUMBER ,A10,31HIS NOT IN DATA BASE AND CAN
   1NOT,12H BE MODIFIED/)
330 FORMAT (/11H AD NUMBER ,A10,37HIS NOT IN DATA BASE AND CAN NOT BE
   1MO,6HDIFIED/)
340 FORMAT (5H ****,12HERROR NUMBER,I3,4H****)
    END
```

```
      SUBROUTINE SORT (N,LIST,INADS)
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     SUBROUTINE SORT ALPHANUMERICALLY ORDERS THE ENTRIES IN THE LIST
C     ARRAY (WITHOUT ALTERING THE ORDER OF THE ENTRIES) BY PLACING IN
C     INAD(I) THE SUBSCRIPT J SUCH THAT LIST(INAD(I)) GIVES THE I-TH
C     ALPHANUMERICALLY SMALLEST ELEMENT OF LIST. THE PROCEDURE DEPENDS
C     ON IPOWER MERGING CYCLES. IN THE FIRST CYCLE INDIVIDUAL ENTRIES
C     OF LIST ARE MERGED BY PAIRS - 1 AND 2, 3 AND 4, ETC. TO FORM
C     ORDERED PAIRS. IN THE SECOND CYCLE THE FIRST PAIR IS MERGED WITH
C     THE SECOND, THE THIRD WITH THE FOURTH, ETC. TO FORM ORDERED QUADS.
C     THE RESULT OF THE IPOWER-TH CYCLE IS A COMPLETELY ORDERED LIST
C     ARRAY.
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
      DIMENSION LIST(128), INADS(128), INAD(128)
C
C         DETERMINE MINIMUM POWER OF 2 WHICH IS GREATER THAN OR EQUAL TO
C         THE NUMBER OF ENTRIES TO BE ORDERED IN THE LIST ARRAY, N
      LIMUP=1
      DO 10 I=1,7
      LIMUP=LIMUP*2
      IF (LIMUP.LT.N) GO TO 10
      IPOWER=I
      GO TO 20
   10 CONTINUE
C         INITIALIZE INADS
      LIMUPH=LIMUP/2
   20 CONTINUE
      DO 30 I=1,LIMUP
      INADS(I)=I
   30 CONTINUE
C
C         ORDER LIST ALPHANUMERICALLY (150)
C
      DO 150 I=1,IPOWER
      N2=2**(I-1)
      I1S=-N2*2+1
C         MERGE BY PAIRS LIMUPH/N2 SUBGROUPS OF N2 MEMBERS EACH
      DO 130 J=1,LIMUPH,N2
C         I1 AND I2 KEEP TRACK OF THE ELEMENTS BEING COMPARED FROM THE
C         TWO SUBGROUPS
      I1=I1S=I1S+N2*2
      I2=I2S=I1+N2
      INITIAL=I1
      LAST=I2+N2-1
C         MERGE A PAIR OF SUBGROUPS (120)
      DO 120 K=INITIAL,LAST
```

```
C          THE DETERMINATION OF THE ALPHANUMERICALLY SMALLEST OF TWO
C          MEMBERS OF LIST CAN BE MADE BY SUBTRACTION ONLY WHEN BOTH ARE
C          POSITIVE OR NEGATIVE BECAUSE THE SIGN BIT (59) IS NOT USED AS
C          A TRUE SIGN BIT, BUT RATHER AS PART OF THE LEFTMOST
C          ALPHANUMERIC CHARACTER.
       IF (LIST(INADS(I1)).GE.0 AND.LIST(INADS(I2)).GE.0) GO TO 40
       IF (LIST(INADS(I1)).LT.0.AND.LIST(INADS(I2)).LT.0) GO TO 40
       IF (LIST(INADS(I1)).GE.0) GO TO 50
       GO TO 70
    40 CONTINUE
       IF (LIST(INADS(I1))-LIST(INADS(I2))) 50,60,70
C          LIST(INADS(I1)).LT.LIST(INADS(I2))
    50 CONTINUE
       INAD(K)=INADS(I1)
       I1=I1+1
       IF (I1.EQ.I2S) GO TO 80
       GO TO 120
C          LIST(INADS(I1)).EQ.LIST(INADS(I2))
    60 CONTINUE
       GO TO 50
C          LIST(INADS(I1)).GT.LIST(INADS(I2))
    70 CONTINUE
       INAD(K)=INADS(I2)
       I2=I2+1
       IF (I2.GT.LAST) GO TO 100
       GO TO 120
       KP=K
    80 CONTINUE
C          TAKEN IN ORDER
C          I1 SUBSET OF LIST EXHAUSTED, REMAINING ENTRIES OF I2 SUBSET
       DO 90 M=I2,LAST
       KP=KP+1
       INAD(KP)=INADS(M)
    90 CONTINUE
       GO TO 130
   100 CONTINUE
C          I2 SUBSET OF LIST EXHAUSTED, REMAINING ENTRIES OF I1 SUBSET
C          TAKEN IN ORDER
       I1P=I2S-1
       KP=K
       DO 110 M=I1,I1P
       KP=KP+1
       INAD(KP)=INADS(M)
   110 CONTINUE
       GO TO 130
   120 CONTINUE
   130 CONTINUE
C          SAVE INAD ARRAY FOR NEXT CYCLE OF ORDERING PROCESS
       DO 140 IL=1,LIMUP
       INADS(IL)=INAD(IL)
```

```
140 CONTINUE
150 CONTINUE
    RETURN
    END
```