AD-A037 738. PARKE MATHEMATICAL LABS INC CARLISLE MASS F/G 5/2
A COMPUTER STORAGE AND RETRIEVAL SYSTEM FOR SCIENTIFIC DATA.(U)
SEP 76 T B BARRETT, I B JARVIS F19628-74-C-0031
UNCLASSIFIED RADC-TR-76-344 NL

1 OF 2
AD
A037738

(12)

RADC-TR-76-344
Technical Report
September 1976

A COMPUTER STORAGE AND RETRIEVAL SYSTEM FOR SCIENTIFIC DATA

Parke Mathematical Laboratories, Inc.
One River Road, Carlisle, Mass. 01741

Approved for public release
distribution unlimited.

ROME AIR DEVELOPMENT CENTER
AIR FORCE SYSTEMS COMMAND
GRIFFISS AIR FORCE BASE, NEW YORK 13441

Drs. Theodore B. Barrett and Stanford P. Yukon are the co-responsible investigators for this contract. Carl A. Pitha (ETSS), is the RADC Project Engineer.

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

This technical report has been reviewed and approved for publication.

APPROVED:

Carl A. Pitha

CARL A. PITHA
Project Engineer

[illegible]

APPROVED: ROBERT M. BARRETT
Director
Solid State Sciences Division

FOR THE COMMANDER: [illegible]

Plans Office

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER RADC-TR-76-344	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A COMPUTER STORAGE AND RETRIEVAL SYSTEM FOR SCIENTIFIC DATA.		5. TYPE OF REPORT & PERIOD COVERED Final Report 9/73 - 7/76
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Theodore B./Barrett Ingrid B./Jarvis		8. CONTRACT OR GRANT NUMBER(s) F19628-74-C-0031 new
9. PERFORMING ORGANIZATION NAME AND ADDRESS Parke Mathematical Laboratories, Inc. One River Road - Carlisle, Mass. 01741		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 62601F 56210804 08
11. CONTROLLING OFFICE NAME AND ADDRESS Deputy for Electronic Technology (RADC/ETSS) Hanscom AFB, Massachusetts 01731 Contract Monitor: Carl Pitha		12. REPORT DATE Sep~~tember~~ 1976
		13. NUMBER OF PAGES 176
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Final rept. Sep 73 - Jul 76,		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) 171p.		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) DATA STORAGE; SCIENTIFIC DATA; DATA RETRIEVAL		

18 19 6 10 15 16 17 11 9 12

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

A computer storage and retrieval system for use with a CDC6600 computer as it is configured at AFGL, Hanscom AFB was developed. The information stored and retrieved falls into three categories: source citation (bibliographic), data category (keyword) and scientific data (usually quantitative measurements on various properties of a selected class of materials). This report gives a general and specific description of this system which consists of computer subroutines, files and operational procedures. In addition it provides a detailed user's guide for operating the system.

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

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

276 450

LB

SECURITY CLASSIFICATION OF THIS PAGE(*When Data Entered*)

SECURITY CLASSIFICATION OF THIS PAGE(*When Data Entered*)

EVALUATION

1. This report is the Final Report on the contract. The effort carried out under this contract was in support of an extensive program by the Air Force Weapons Laboratory on the development of high power infrared lasers. The contractor provided expertise in particular aspects of mathematical theory, the mathematical analysis of the optical properties of dielectric and semiconductor materials, the development of computer programs in support of in-house portions of the program, and the development of a computer storage and retrieval system for scientific data. Contractor supplemented in-house capabilities in crucial areas and made possible the development of an adequate output window for the AFWL high power infrared laser.

2. Since this contract was devoted to AFWL work only and predates the establishment of RADC/ET, no RADC TPO exists with which it may be associated.

Carl A. Pitha

CARL A. PITHA
Project Engineer

Preface

This final report provides a description and user's manual of the data storage and retrieval system which was started under contract F19628-71-C-0142 and continued under contract F19628-74-C-0031. For an overview of this system refer to Chapter I. Chapters II and III provide technical details for system users and for those who may wish to modify the system.

Additional work performed under this contract included the following:

1) Multiphonon infrared absorption - basic theory; testing of the correlation function theory for various "model" crystalline and amorphous materials; development of a soluble Hamiltonian model.

   This work is summarized in Tech. Memorandum 25, "Multiphonon Infrared Absorption in Ionic and Semiconducting Solids", S. Yukon, August 1976. This report also includes reprints of journal articles which were a partial outgrowth of work done under this contract as well as the predecessor contract.

2) Selected data reduction - computer reduction of calorimetry data from plotted temperature rise and fall curves; computer reduction of plotted spectrometer data; computer reduction of "computer produced" spectrometer data.

   This work is summarized in Tech. Memorandum 26, "Various Computer Subroutines for Data Reduction of Calorimetry and Spectrometer Data", T. Barrett, August 1976. This work includes user's guides to various subroutines used in the data reduction and various procedures which should be followed in doing the data reduction.

3) Thermal lensing in infrared windows - solution of the heat equation with source in cylindrical coordinates; intensity of a Gaussian source through a thermally stressed window.

This work, which involved the development of several computer subroutines is summarized in RADC report RADC-TR-76-269, "Computer Solutions to the Heat and Diffraction Equations in High Energy Laser Windows", P. Gianino, B. Bendow, N. G. Parke, T. Barrett, - November 1976.

The following individuals at Parke Mathematical Laboratories, Inc. have contributed to results produced under contract F19628-74-C-0031:

T. B. Barrett

L. Calabi

B. M. Langworthy

I. B. Jarvis

N. G. Parke

N. Quinn

TABLE OF CONTENTS

Figures: (continued)

CHAPTER I INTRODUCTION TO THE BIBDIS SYSTEM

Section 1 - Overview and History

In spite of the recent proliferation in computer abstracting services, it is still useful for the individual scientist or group of scientists to be able to store, retrieve and manipulate information for which they have a particular need or expertise. Consider the following quotation from Reference 7 (page 2):

"The other type of component is the individual scientist (or small group of collaborators) who produces a "one-shot" compilation or critical review as a part of what he regards as his normal scientific activity. Many valuable data compilations have been produced in this way. Such individuals do not consider themselves part of a formal data center, and there is generally no commitment for continuity or updating. The rapid growth of the scientific literature makes it increasingly difficult for an individual to do this type of compilation. However, the continuing data centers can serve a useful function by providing bibliographic back-up for individual scientists in other locations who wish to write critical reviews or do critical compilations of limited scope."

The system of data storage and retrieval (which is called BIBDIS) described herein grew from such a need. The origins of this need go back to the experience of compiling, "by hand" so to speak, a compendium of useful physical properties of a restricted class of of materials (see Reference 1).

The initial phase of this work was one of merely generating and maintaining a computer file of bibliographic entries (author, Journal, title) selected by the user. These entries referred to articles, books, etc. which contained information on specific quantitative data germaine to the compendium. A set of Fortran subroutines was written to sort this file in various ways specified by the user so that it could be displayed in various forms such as "by journal" or "by author". These subroutines proved to be quite time consuming so the CDC utility package sort-merge was finally used for the task.

The next phase of the task was to code each bibliographic entry (or actually some of the entries) with "key words" which described briefly the contents of each entry. These key words were put into a separate computer file. Subroutines were subsequently written to list the bibliography "by material" and/or "by property", etc.

The final phase of the task was one of including provisions for storing quantitative data from the journals, etc. referred to in the bibliography. This task included, of course, that of retrieving the data and properly displaying it. Again another file was added to the set of files which constitute the "data base" of the system Thus a "bibliographic file", a "keyword file" and "data file" were successively brought into the system.

Some of the initial work was started under contract F19628-71-C-0142. The effort for the most part has been "low level" and initially at least one without much formal structure to it e.g., the task at hand was accomplished as expeditiously as possible with little concern for the development of the most efficient system for the "overall" task. In fact the overall task has changed according to changing needs of the user.

An attempt was made during the closing phases of the original contract (0142) to formalize the quantitative data storage and retrieval problem (see Reference 2). In fact a trip was made to the National Bureau of Standards Office of Standard Reference Data, Data Systems Design in an attempt to obtain data storage and retrieval "standards" and perhaps subroutines which would ease the writing of the necessary storage and retrieval programs. It became apparent that such standards did not exist and that the available subroutines were either too general or specialized to be of much use in our work. Thus we continued to make our own rules of operation and, more or less intuitively, gradually assembled together a system of files, subroutines, standards, rules of operation, etc. which together constitute the present data storage and retrieval system reported herein. As mentioned previously this system has grown as the need for various capabilities of the system arose; it also has grown and become more sophisticated with increased knowledge of the capabilities of the CDC "operating system" a system, by the way, which has grown considerably in accessibility and utility to the non-systems programmer

This effort has culminated in what is referred to in TM19 (Reference 3) as the "BIBDIS Operating System" which, more or less, consists of a set of general purpose and specialized subroutines which can easily be added onto as the need arises and which together provide the retrieval part of the task. Implicitly this system includes a set of general rules which govern some of the structure of subroutines within the "BIBDIS set".

The overall system of storage and retrieval which has been put together is far from optimum or perfect. It has evolved with need, knowledge, system changes and time and, like most things which do not blossom forth in an instant, retains vestigal elements of its evolution. It works, however, and seems to perform its required duties reasonably well and with a minimum of help from the user (assuming the computer is working as advertised).

Section 2 - Capabilities

With this brief historical introduction we next briefly outline the major elements of the system (which are, in fact, common to many computer information storage and retrieval systems) and how we have implemented these elements. The major constraint has been, of course, that the system must work on the CDC 6600 as it is presently configured at AFGL including the usual user I/O devices, which include punched card input, teletype input, output, and high speed printer (64 character) output.

The first logical element is that of data entry. As mentioned before, for one reason or another, the data base consists of three separate, but inter-related files: bibliographic, keyword and quantitative data (or, for convenience, quantity). Data entry into all files is by punched card; the cards themselves become (or should become) the primary back up "files". Corresponding to each file is a set of two subroutines which are used to preprocess the data and to add entries to the files. Much of the data can be punched on the cards in field free form. The usual procedure is to produce a sizeable number of cards, store them on "permanent" disc file by using a utility copy program, then run the card images through the preprocessor which does as many validity checks on the data as possible. Data correction is then usually done on line through INTERCOM and EDITOR. The presumably correct data is then restructured and added to its appropriate file using the second preprocessor. One of the features of this data entry scheme is a "Standards" file or files which is used to check certain data elements. For example, for the bibliographic file, the standard includes a list of standard journal abbreviations to which journal entries must conform.

The second logical element is the set of working files which constitute the data base. As mentioned above, three separate "main" files are used in the system, partly because of historical reasons, but partly because they can serve as independent useful files. For example the bibliographic file can be sorted and displayed in various ways to produce useful information about the sources of quantitative data. These main files are frequently used in some secondary form by the BIBDIS retrieval system; for example the bibliographic file which is "basically" a sequential file, is most often stored and used as a word addressable file. The quantitative data file is the "decode" files to translate code words in the master files and "standards" files for data entry checks. File form or type (such as sequential, word addressable, indexed sequential) is dictated by various factors such as where the data is to be stored, storage efficiency, speed of data access, how the data is to be accessed. Among other considerations is the fact that CDC Sort-Merge will only work on sequential files thus intermediate files must often be created in sequential form. In addition, for back up purposes files must usually be stored on magnetic tape in sequential form.

The third logical element is that of data base "maintenance" which here means corrections (including deletions) and protection from catastrophic loss. At present, not much work has been done on this element. Some special purpose subroutines have been written from time to time to make massive corrections such as one to standardize journal abbreviations (prior to this some journals appeared in half a dozen variants). Standard techniques can of course be utilized such as the periodic creation of a "transaction file" to amend the "master" file. The quantity file, incidentally, is an indexed sequential file which is amenable to more direct methods of correction.

The fourth element, and perhaps the most important is that of information retrieval since this really tells what the system does. Basically it does what any good information retrieval system does: it produces lists of associated data from the data base which fall into certain categories specified by the user. Perhaps the best way of indicating exactly what this means for this system is to give a table of LISTB directive parameters. The LISTB directive is one of several which can be used to control the BIBDIS "operating system". It is the principal directive for information retrieval.

Table I - LISTB Parameters

1. "Order"Parameter (N) see note 2

Value	Meaning
N	accession number
A	author
J	journal
M	material
Q	quantity
PM	property - material
MP	material - property
PQ	property - quantity
MQ	material - quantity
AY	author - year
AJ	author - journal
JA	journal - author
JY	journal - year

2. First Lower Delimiter (first possible)

c a character string of up to 10 characters appropriate to the first "order" parameter

/ See note 1

3. First Upper Delimiter (last possible)

c a character string of up to 10 characters appropriate to the first "order" parameter

4. Bibliographic Citation Code (F)

Value	Meaning
F	full entry
N	accession number only
AJY	author, journal, year
AY	author, year

5. Contents Code (N)

Value	Meaning
N	none
M	material
P	property
MP	material - property
PM	property - material
D	all data

6. Second Lower Delimiter (first possible)

c a character string of up to 10 characters appropriate to the second "order" parameter

7. Second Upper Delimiter (last possible)

c a character string of up to 10 characters appropriate to the second "order" parameter

Note 1 - if / is inserted for the first lower delimiter of an "order" parameter, all entries are listed which do not include the "order" parameter. This is used mainly to determine what data needs to be added to the data base.

Note 2 - a question mark (?) may be used in place of any parameter value causing a default value to be used. This default value is indicated in parenthesis beside each parameter.

In general the "order" parameter controls the order in which data is listed while all other parameters control the amount of data listed. In words LISTB will: (numbers refer to parameter number in table 1)

List the contents of the data base by 1 (first part) then by 1 (second part) from 2 to 3 and from 6 to 7. The amount of "source" is given by 4 while the amount of "data" is given by 5.

It may be possible to recover the same information using different parameters but, in general, this information will be displayed in slightly different forms.

To give some examples:

LISTB, A, BALLARD, BALLARD, N, D

Lists all data published by authors named Ballard (or more exactly authors whose last names start with Ballard). The bibliographic citation is by accession number only.

LISTB, PM, ELA, ELA, F, D, GASE, GASE

Lists data on the elastic properties of GaSe, full bibliographic citation.

LISTB, MP, GASE, GASE, F, D, ELA, ELA

Same as above.

LISTB, MP

The entire data base is listed alphabetically by material (actually material code), and under each material by property (actually property code). A full bibliographic citation is given but no quantities (or quantitative data is listed.

The technical details of the system are given in the following chapters. However it should be stressed that this system was designed for use by the individual scientist wherein he can collect his own data from any sources whatsoever (including his own laboratory), store it in a data bank and later retrieve it in various useful forms. Of primary importance to such an individual is the way physical data is entered, stored and treated for display purposes. This is done below briefly. Of secondary importance is the way source-of-data (bibliographic citation) and keyword(s) are handled. This will also be done below in a somewhat abstract form since "source" and "keyword(s)" can be considered to be abstract quantities apart from the way they are used in the particular data base which is presently used with the system described.

The physical data (quantitative data, quantity) is assumed to have some or all of the following characteristics. The system will display these characteristics in a "nice" format.

(1) it has a name (quantity code)

(2) it applies to a specific physical material and can be categorized as having the attributes of a specific "physical property" - in general this physical property is not as specific as the quantity itself but they may be identical. (materials code, property code)

(3) it has physical units (units code)

(4) it comes from some source (accession number)

(5) it may be associated with other quantities from the same source. (Thus each quantity from a fixed source is given a reference number which may be used to establish its interdependancy).

(6) it is categorized as being an "independent" or "dependent" quantity. In other words "wavelength" or "temperature" which are usually "independent quantities are treated in the same way as "dependent" quantities such as "index-of-refraction".

(7) it is associated with an environmental condition.

(8) the quantity itself is either a descriptor (an alpha numeric character string) or a list of one or more "numbers".

(9) the "numbers" may appear in one of several forms; e.g. f, $f \pm e$, $\sim f$, $f_1 < f < f_2$, etc. where f is in general a real number in "scientific notation". On occasion we might also find abs (quantity) = f.

(10) the numbers may have a multiplier associated with them.

The actual format for entering data into the system is given in Chapter III, section 3. There are a few important facts however, which are worthwhile to stress here.

(1) Every code word (quantity, material, property, units) is limited to a maximum of 10 characters. Every code word must appear in a code word file along with its translation <u>before</u> other data referring to this code is entered. For display purposes the full translation is used.

(2) In general inter-related quantities are displayed together. In particular tables of values are printed in adjacent columns with appropriate headings. The order in which they are printed out is governed by the reference numbers (see (5) above). A place has been left to store a "display code" which can be used to specify a particular form of display where the "usual" form cannot apply.

(3) It is possible to access the data base by "quantity". This may not make sense in some instances, for example, all sources which contain the quantity "temperature" could be listed even though temperature is usually an independent variable or "condition" under which another physical quantity

was obtained.

The "source" (e.g. bibliographic source as represented by a bibliographic entry and associated accession number) is assumed to be entered in a particular form which permits various orders of presentation e.g. by "author", by "journal", etc. More specifically data entry of each source is done by preparing exactly 4 cards (some of which may be blank except for card no.) with card number as the last item on the card. (See Chapter III, section 1 for the exact format). Card 1 contains "author"; for search purposes the first 10 characters only of author are used. Card 2 contains journal code, journal volume, pages, and year. Journal code must be the first item on the card and year the next to last item (card number is last). Journal volume and pages are not used for search purposes but if they exist must appear in a certain order (see Chapter III, section 1). Note that journal code, etc. may actually be a report or book citation. Cards 3 and 4 contain the title of the source: usually the title of some article or as much of it as can appear in two 80 column cards. Each source is coded by "accession number", an integer between 1 and 99999 inclusive. This code is not provided by the user, rather it is appended to each entry by a computer subroutine as the entry is added onto the master bibliographic file. (Accession number is the same as sequence number in the master file.)

Although "key words" are important elements of the system, they need not concern the user if "source" and "quantities" are added to the data base more or less simultaneously since "key words" are obtained by the system from the "quantity cards" and added to the key word file. The key word file contains the codes: accession number, property, material, quantity. Its primary purpose is to provide a succinct description of the "content" of its source. The user may, of course, add to the contents of the keyword file without creating entries for the quantity file. The rules for doing so are given in Chapter III, section 2 .

Although LISTB is the primary directive of the BIBDIS system other directives are available. For example the directive "SIZE, file" will give the current size of any file in the data base. For a complete list of current directives see Chapter III, section 4 .

In concluding this summary of the BIBDIS system we give below a few comments on some of the technical aspects of the system.

1. The "executive" portion of BIBDIS consists of a very simple main program (calld MAIN) which is made up of an "introduction", a "computed go to" statement, calls to major task entry points (see below) and a closing statement. In addition to these executable elements, the main program is used to declare several common storage areas and four files (INPUT, OUTPUT, DATST and PRINT) which are referenced by the usual Fortran READ, WRITE statements. (all other files are referenced through Record Manager calls - see below). The introductory part of MAIN is a call to subroutine INTERP which reads each BIBDIS directive from INPUT and interprets this directive.

2. The directives of BIBDIS are carried out by calling one or more task entry points corresponding to directive numbers. These entry points are referred to below as BIBSUBs.

3. The structure of BIBDIS will allow for segmentation (see reference 4)

in that each BIBSUB can be overlaid on top of any other BIBSUB during execution. At the present stage of development it is not necessary to use segmentation, but by providing for it, BIBDIS can essentially become infinitely large and still be executable on a finite sized computer. All communications among various BIBSUB's must be done through common storage blocks.

4. The root segment (including MAIN) contains several "utility" subroutines which can be called by all BIBSUB's and MAIN.

5. File INPUT is used only for feeding directives into BIBDIS. See Chapter III, section 4 for the form of these directives.

6. File OUTPUT is used by BIBDIS only for outputting informative messages such as error messages, having to do with the state of BIBDIS. It is also used by the Scope operating system to inform the user of troubles.

7. File PRINT is reserved for use as the major working output file of BIBDIS.

8. Files used by BIBDIS (not including program files) can be put into one of three major classes (within a class there may be a further subdivision of file types).

   Class I - consist of the "utility files" INPUT, OUTPUT, DATST, and PRINT. These are Fortran standard "formatted" files mentioned above.

   Class II - consists of the files which contain all information required by BIBDIS. These are called "primary" files and must conform to certain rules listed below. They are assumed to reside on a private disk pack.

Class III - consists of all other data files such as "tag" files or word addressable files which may be required for the easy extraction of information from the other files. These files are called "secondary" and may or may not be stored between jobs. (The number of secondary files which can be stored depends on the amount of space available.)

9. All files in Class II and Class III are accessed through Record Manager calls.

10. Directives are put on cards (or card images in file INPUT (or a file equivalent to INPUT) using the rules given in Chapter III, section 4.

11. Any number of directives may be placed in INPUT. BIBDIS will terminate under one of the following conditions:

   a) when a fatal error is detected by BIBDIS in interpreting or carrying out a directive. At the present time mis-punched directive codes are not considered fatal, BIBDIS merely goes to the next directive.
   b) when a fatal error is detected by the operating system;
   c) when the last directive has been completed;
   d) when CP time limit is reached.

12. Parameters are passed to each BIBSUB in character string form through common storage area /PARAM/. One of the first tasks of each primary subroutine is to interpret these parameters. Various utility subroutines are available in the root segment to assist in this interpretation. For example subroutine INT translates a character string to binary integer form or returns a code indicating that such a translation cannot be made. Parameters are left justified. A ? indicates that the default is to be used except when a blank parameter is found, this and all following are default.

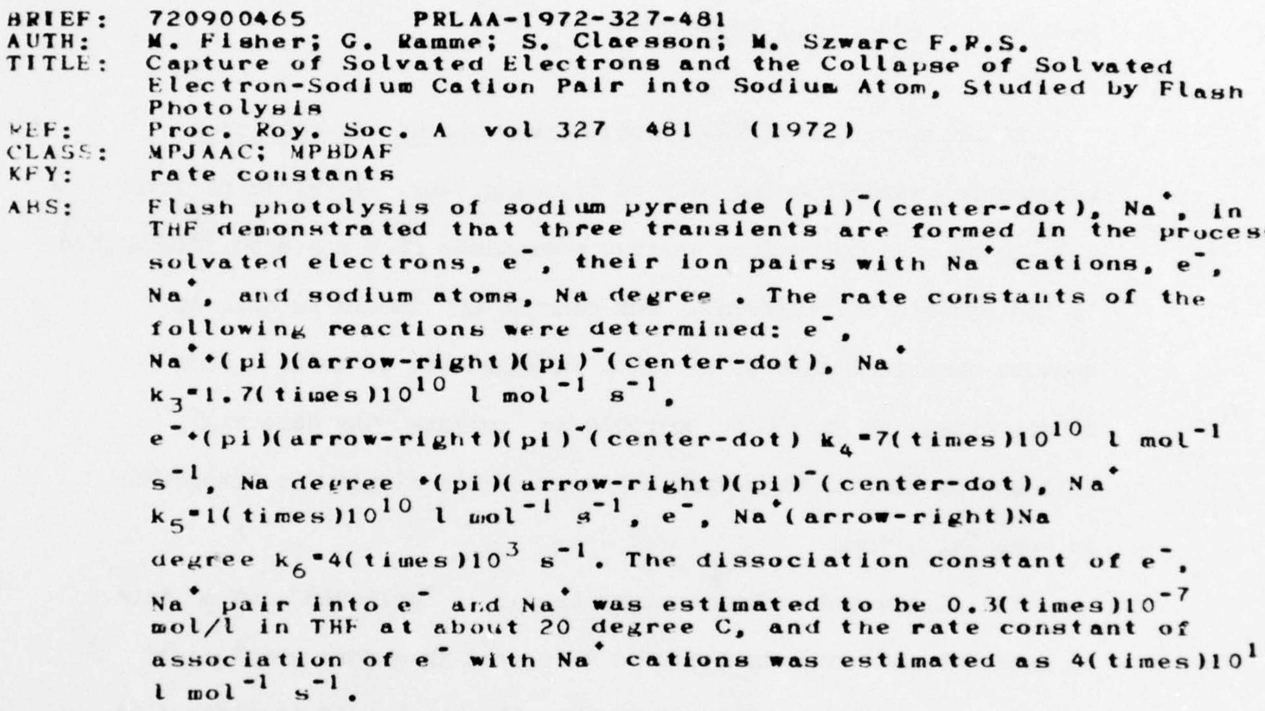
A list of the present BIBSUBs and other "utility" subroutines along with a brief description of their functions are given in Chapter II, section 2 .

Besides the structure of the BIBDIS system, its most noteworthy feature is the extensive usage of Record Manager calls (see Reference 5, Reference 6) for file manipulation rather than the standard Fortran I/O statements. Although more care must be exercized in using this "non-standard" I/O procedure there are several benefits not the least of which is the ability to save considerable storage space by overlaying FIT and buffer areas.

Section 3 - Additional Desiderata

At present, the BIBDIS system merely stores and retrieves information without doing much of anything else. It would be desirable to be able to perform various operations on the stored information in the process of retrieval. For example one should be able to convert numerical data to a common standard set of units. Once so converted, it should be possible to "collate" the data e.g., collect the similar data from all sources and display it altogether in some "nice"form.

The ultimate desiderata might be such a "collated" set of data by material and property which is displayed in a form which could go directly to print. This means that the present limitation of 64 characters fromprint out of information should be increased to a number which at least can embrace upper and lower case letters in addition to sub and superscripts. An example of such an output (done with 256 character IBM print chain) is shown in Figure I.1. It is still an awkward presentation in that greek letters and some other commonly used symbols are not available.



```
BRIEF:   720900465        PRLAA-1972-327-481
AUTH:    M. Fisher; G. Ramme; S. Claesson; M. Szwarc F.R.S.
TITLE:   Capture of Solvated Electrons and the Collapse of Solvated
         Electron-Sodium Cation Pair into Sodium Atom, Studied by Flash
         Photolysis
REF:     Proc. Roy. Soc. A  vol 327  481   (1972)
CLASS:   MPJAAC; MPBDAF
KEY:     rate constants
ABS:     Flash photolysis of sodium pyrenide (pi)^-(center-dot), Na^+, in
         THF demonstrated that three transients are formed in the process:
         solvated electrons, e^-, their ion pairs with Na^+ cations, e^-,
         Na^+, and sodium atoms, Na degree . The rate constants of the
         following reactions were determined: e^-,
         Na^+ +(pi)(arrow-right)(pi)^-(center-dot), Na^+
         k_3=1.7(times)10^10 l mol^-1 s^-1,
         e^- +(pi)(arrow-right)(pi)^-(center-dot) k_4=7(times)10^10 l mol^-1
         s^-1, Na degree +(pi)(arrow-right)(pi)^-(center-dot), Na^+
         k_5=1(times)10^10 l mol^-1 s^-1, e^-, Na^+(arrow-right)Na
         degree k_6=4(times)10^3 s^-1. The dissociation constant of e^-,
         Na^+ pair into e^- and Na^+ was estimated to be 0.3(times)10^-7
         mol/l in THF at about 20 degree C, and the rate constant of
         association of e^- with Na^+ cations was estimated as 4(times)10^11
         l mol^-1 s^-1.
```

Figure I.1 256 Character Print Chain Output

CHAPTER II TECHNICAL DESCRIPTION

The bibliographic entry and display system has been written in Fortran for theCDC 6600. It has been developed in such a way as to require a minimum amount of computer know-how for the user. This was accomplished in part by simplifying input parameters and by using the sort-merge package within the Fortran routines rather than requiring the user to supply the sort-merge parameters himself. Also Record-Manager calls are used extensively in the system to allow the many different files which may be needed to share the same buffer area and record blocke as well as File Information Table. Use of Record-Manager also precludes the need to specify files needed on the Program card.

An overall description of this system is given in the First Section of this chapter. This is followed by a description of all the files used by the system. The final section of this chapter gives a detailed description of each program and subroutine written and the interrelationship of the programs and files.

Section 1.- Modus Operandi of the System

Establishment of a Bibliographic Display System first requires an easy and logical method of entering information. This has boen accomplished by dividing the information to be entered into 3 parts:

1) bibliographic entry itself,
2) keywords from the article and,
3) physical data.

After deciding that information from an article will be entered into the system, one would first want to enter on punched cards the author(s), journal, page(s), and title. Each such entry originates as a set of four 80 column cards as follows:

card 1

Authors

card 2

Journal, month/issue no. (optional), vol. no.,

First page, Last page, Year

card 3 - card 4

Title

Each card contains a card number (1 - 4) in column 80. An entry will always consist of 4 cards even though 1 or more of these cards might be blank except for the card number. The last 10 characters of card 4 are ignored except for the card number, so the title should end by column 70 of this card. A detailed description of the format of these cards can be found in Chapter III, Figure III.1.

Since it is important for retrieval of information that the same journal name always be entered with the same spelling and abbreviations, a standard Journal Abbreviation File has been established according to guidelines in "Bibliographic Guide for Editors and Authors" (Reference 8). A Journal Word Abbreviation File has also been established which is used by the system to abbreviate particular words within the Journal name, so that theywill agree with the words in the

Journal Abbreviation File. Consequently it is not necessary for the user to enter the Journal name exactly as it is in the Standard Journal Abbreviation File as the system will attemptto abbreviate it properly. Two programs, CRJWA and CRJAB, have been written to create the Journal Word Abbreviation File and the Journal Abbreviation File, respectively. The structure of these 2 abbreviation files can be seen in the next section of this Chapter, pages 31 and 32. The card format used to produce these files, and the current list of standards in the files can be found in the User's Guide, i.e., Chapter III, pages 80 thru 91.

Program BREAD1 has been written to check the bibliographic entry cards. This program corrects as many inconsistencies as possible and flags all cards with errors. Program BREAD2 is then used to add the corrected set of cards on to the Primary Bibliographic File. When adding these new cards to the file, BREAD2 assigns an accession number to each bibliographic entry, i.e. each entry will be given a unique, successive number. The Primary Bibliographic File is an unformatted sequential file of 32 words/record as described on page 25.

Keywords which describe the content of any bibliographic entry by property and/or material may be entered next. Again it is necessary to set up standard codes for these keywords so that any property or material will always be abbreviated in the same manner. These codes have been put in a Data Standards File (DATST) along with codes for quantities and units which will be discussed later. This file is described and listed in the User's Guide, page 125.

Keyword cards, see Figure III.5., contain an accession number, property code and up to 6 material codes. Continuation cards with the same accession no. and property code may follow, with up to 29 material codes for the same accession no. and property. Program CREAD1 checks these cards for validity and, when they are corrected, program CREAD2 adds then to the PK File as described on page 28. Program CREAD3 reads this Primary Keyword File and breaks it into records which include accession no., property code and material code. These records are sorted in increasing order of accession no., property code and material code and any duplication is eliminated. This program writes the final File called the DIR which will later become a directory File pointing to the Index Sequential Data File. The File is described on page 29.

The last set of information to be entered is the physical data presented in the article. In forming the data base, articles previously entered into the bibliographic system are reviewed by the problem originator and quantities of importance are red-penciled. For identification purposes each set of data to be entered must include the accession no. of the article from which it came and the property, material and quantity (physical property) to which the data applies. A data type code must also be entered to identify the data as numeric, absolute value, or alpha, etc. These data type codes are detailed in the User's Guide, page 111. String data, environmental conditions, units code, multipliers, and data in singular or tabular form may be entered. Once again, it is necessary to standardize codes. Consequently, acceptable quantity and units codes have also been stored in the Data Standards File. All this data information is punched on cards, in field free format, separated by semi-colons, as described in the User's Guide.

Program DREAD1 checks these cards for errors and program DREAD2 accepts the corrected cards, and adds the information to both the directory file (DIR) and the indexed sequential sorted data file (ISD) as described on pages 29 and 30. This program eliminates any duplication in the DIR file which may occur due to keywords previously entered from the same article as the data now entered. Since one must always enter the appropriate keywords along with the data, it is not necessary to enter keywords alone with programs CREAD1 and CREAD2 if one intends to also enter data.

The final part of this system is referred to as the BIBDIS Operating System. It allows the user to easily retrieve and display, i.e. print, any information previously entered into the system by means of simple directives. Since many of the files created are quite large, they have all been stored on a private device set. Because of this and the size of the programs involved, especially the sorting program, BIBDIS must be run in the batch mode. This program consists of a main program to read and interpret input cards and pass control to one of the many subroutines which actually implement the input directives. These subroutines are explained in detail in pages 45 thru 63.

Several secondary files may be needed during execution of certain directives. These include a Word-addressable Bibliographic File, (PBR), page 26, Tag files pointing to the PBR in many different orders such as Author or Journal, page 27. The Directory File may also need to be sorted in many different orders such as by property or material. The contents of these files is described in the User's Guide, page 137.

As many of these secondary files as will fit on the device set, may be stored there, so that they will not need to be created whenever they are needed. (The system checks first to see if they exist and creates them only if they do not.)

The following pages give a detailed description of all the files used by the system. The abbreviations used in these tables are defined as follows:

FO (File Organization);

SQ (Sequential);

WA (Word-addressable);

IS (Indexed Sequential);

RT (Record Type);

W (Control Word);

U (Undefined);

RL (Record Length).

Further information on file structures may be found in the Record Manager Manual, Reference 5.

"Word" refers to each 60 bit word within the record. "Type" refers to the method of converting the data from input to storage, i.e. A (10 characters, each in 6 bit display code) and I (integer). Further information on word storage may be found in the Fortran Manual, Reference 13.

PB - Primary Bibliographic File

FO = SQ (sequential)

RT = W (control word)

RL = 32

Order: By accession no.

Word		Type
1	Accession no.	A
2-9	Author	A
1∅-14	Journal	A
15-16$\frac{1}{2}$	Volume & Pages	A
16$\frac{1}{2}$-	Year	A
17-32	Title	A

PBR - Word-addressable Bibliographic File

FO = WA (Word-addressable)

RT = W (Control word)

RL = 32

Order: By accession no.

Same Record structure as PB

PBxxT - Tag Files

FO = SQ (Sequential)

RT = W (Control word)

RL = 63

Order: By xx

Word			Type
1-63	Tag name (5 chars.)	repeated	A
	Accession no.(5 chars.)		

PK - Primary Key Word File

FO = SQ (Sequential)

RT = W (Control word)

RL = 32

Order: By accession no. property

Word		Type
1	Accession no.	A
2	Property Code	A
3-31	Material Codes	A
32	Record Number	I

DIR (NPM)-Sorted Keyword and Directory File

FO = SQ (Sequential)

RT = W (Control Word)

RL = 4

Order: By Accession no., Porperty, Material

Word		Type
1	Property code (5 chars.)	A
	Accession no. (5 chars.)	
2	Material Code	A
3	Quantity Code	A
	(= ∅ if no data entered)	
4	Index no. (Acc. no. *1000 + Ref. no.)	I
	Pts. to file ISD	
	(= ∅ if no data entered)	

ISD - Sorted Data File

FO = IS (indexed Sequential)

RT = U (Undefined)

RL = Variable

Order: Index

Word		Mnem.	Type
1	Index (Acc.no*1000+Ref.)	KEY	I
2	Quantity Code	QUAN	A
3	Data Type		A
4-end	String (if Data Type = 5) All other Data Types		A
4	of associated references	NR	I
5,6...NR+4	List of assoc. references		I
NR+5	of independent variables (or * if independent var.)	NI	I or A
NR+6.... NR+NI+5	List of independent var.		I
NR+NI+6	Length of environmental cond.	NLE	I
NR+NI+7... NR+NI+NLE+6	Environmental condition		A
NR+NI+NLE+7	Display Code		A
NR+NI+NLE+8	Units Code		A
NR+NI+NLE+9	Multiplier		A
NR+NI+NLE+1∅	Length of data	NLD	I
NR+NI+NLE+11.... NR+NLE+NLD+1∅	Data (1 word/item)		A

JWABRE - Journal Word Abbreviation File

FO = SQ (Sequential)

RT = W (Control Word)

RL = 12

No.of records = 127

Word		Format
1	4 letters of word	A4
2	No.of abbreviations	I1
3-12	List of abbreviations	1ØA

JABREV - Journal Abbreviation File

FO = SQ (Sequential)

RT = W (Control Word)

RL = 4

No.ofrecords = 256

Word		Format
1-4	Journal Abbreviation	4A1Ø

DATST - Data Standards File

Formatted File

Record	Characters	Description	Format
1	1-4	No. of material codes (m)	I4
2,3,etc.	1-9	1st 9 chars. of mat. code	8A10
	10	"$" if to be continued	
	11-29	Continuation of code	
	30-80	Explanation if needed	
m+2	1-4	No. of property codes (p)	I4
m+3,m+4,etc.	1-5	Property code	8A10
	6-10	Ignored	
	11-30	Translation of code	
	31-80	Any other explanation if needed	
m+p+3	1-4	No. of quantity codes (q)	I4
m+p+4, m+p+5,etc.	1-10	Quantity code	8A10
	11-30	Translation of code	
	31-80	Any other explanation	
m+p+q+4	1-4	No. of units codes (u)	I4
m+p+q+5, m+p+q+6,etc.	1-10	Units code	8A10
	11-30	Translation of code	
	31-80	Any other explanation	

Section 2. - Program Description

CRJWA

This short program creates the Journal Word Abbreviation File. It reads the word abbreviations from cards, attached as TAPE1, and copies them to an intermediate file named AB. Records with all words equal to zero are added to this file to make 127 records. The last card read from TAPE1 is printed on OUTPUT. The main program then calls BIBSUB1 to sort the intermediate file is ascending numerical order on the first 4 characters of each record. Since the display code for the alphabet is in ascending numerical order, this results in an alphabetical sort. The output file of the sort is JWABRE.

CRJAB

This program creates the Journal Abbreviation File in a similar fashion. A 256 record file, 24 words each, is written on AB consisting of the journal abbreviation cards from TAPE1 followed by 0 word records. AB is then sorted by BIBSUB1 in ascending order on character positions 1, 11, 21, and 31 with 10 characters each. The output sorted file is called JABREV.

BREAD1

The purpose of program BREAD1 is to verify original bibliographic entries on punched cards. By using this program it is possible to enter data in field free format (essentially this applies to the journal card). It also attempts to abbreviate journal titles and checks to see if the resultant is a correct title by searching a journal abbreviation table. There are three output files,two of which are print files for user information. The third is a corrected bibliographic entry file which may be further modified in EDITOR or transferred directly into the primary bibliographic file using BREAD2. The following is a list of the functions which BREAD1 performs:

1) It searches for a card number (1 through 4) as the last item on a card (not necessarily in column 80). Cards with missing numbers are flagged.

2) It attempts to put author cards in correct form. The correct form is best explained by an example:

   LOESCHER,DH, DETRY, RJ, CLAUSER, MJ

   i.e. Last name followed by comma followed by initials with no blanks or periods.

   The program will abbreviate first names and put other abbreviation forms into the correct form.

3) It attempts to find correct abbreviations for all words in a journal title.

4) It searches a journal file for a match with the resultant abbreviated journal.

5) It checks for journal number, first and last page number including a check to make sure that the first page number is less than the last page number. It also checks for year and makes sure that $1800 \leq \text{year} \leq \text{current year}$.

6) It lists the original input data as it was punched along with a card sequence number so that the user can locate bad cards easily if corrections are to be made to the original deck.

7) It lists the corrected entries along with the original card sequence number, a number the card will have in EDITOR, a flag indicating

missing card numbers (1-4), a "change made" flag, and a code number beside cards which may still have errors.

8) It outputs a corrected file which may be further modified or fed directly into BREAD2.

The algorithm for abbreviating journal words is worth mentioning.

First the word is tested for length. If greater than or equal to four, the first four letters are used to form an element which is used in a binary search of a file (JWABRE) for a matching element. This file is arranged in ascending numerical order by display code equivalence to 4 letter abbreviation codes. If a match is found the corresponding record is searched for a possible abbreviation as follows. (Actually a sort by integer is done in arranging the code file).

Each record in JWABRE consists of the 4 letter search code mentioned above followed by a number indicating the number of possible abbreviations corresponding to the code, followed by the abbreviations in descending order by length. The word to be abbreviated is then matched letter by letter to abbreviation words until all letters match (up to the end of the abbreviation). When (or if) such a match is found, the abbreviation is accepted. There are some instances where this procedure does not work. For example ELEC may lead to the abbreviation ELECTRON. (for ELECTRONICS) or ELECTRON. To handle this situation, the abbreviation word ELECTRONI$ is used. The symbol $ signifies replace the preceeding symbol with a period. There are other examples (few) where abbreviation is not so straight forward and a special convention is used for these. For example, JAPANESE is abbreviated

JAP. while JAPAN is abbreviated JPN. For this the series JAPANE$$$ JAPAN*JPN. is used where $$$ means place the period 3 characters back and * means use the abbreviation that follows. A list of records in JWABRE as it currently stands is shown on pages 84 thru 86.

If the word is less than 4 letters long it is tested for a match in an "article" list. If a match is found the word is removed, otherwise it remains the same. Words which are one letter long are automatically removed unless they are letters and beyond position 2. (This allows for SER A or PHYS. REV. B)

<u>BREAD2</u>

This program adds on to the Primary Bibliographic File (PB), (page 25), up to 4 boxes of bibliographic entries. It is assumed that the boxes (TAPES 1 thru 4) of cards, (page 74), have previously been checked by BREAD1 and any errors have been corrected. The present PB is attached as TAPE9 and the new PB should be requested as a permanent file on TAPE10. This file should be stored on the private device set. TAPE5 is used to list the new bibliographic entries and the number of records in the original and new files. It should be copied to OUTPUT after execution. Subroutine BREAD1 only checks that the cards are numbered repeatedly from 1 to 4 in column 80. If not, the program stops. Subroutine ADDE removes this card number and adds the accession number as the first word of the record.

CREAD1

This program performs the initial check of a 'box' of keyword cards (TAPE4), (page 102). The Data Standards File, (page 125), is attached as TAPE11. Each card is checked for an accession no. between 1 and 99999 and acceptable property and material codes. Three output files are produced. TAPE5 contains a list of all bad cards (by sequence number + 1). TAPE3 contains a list of all cards with bad cards flagged by asterisks. TAPE7 is a copy of the input tape, TAPE4, except for the first card which is an update directive.

CREAD2

This program adds on to the Primary Keyword File (PK), (page 28), up to 4 boxes of keyword cards (TAPE1-TAPE4). The present PK must be attached as TAPE9 and the new PK should be requested as a permanent file on TAPE10. Sburoutine CREAD1 does a check fo the input cards in the same way as program CREAD1 but only writes TAPE5, a list of bad cards. If any bad cards are found, the program stops. Subroutine ADDK reads the input cards putting all those entries with the same accession no. and property into one record and adding the record no. as the last word. TAPE5 is used to list all the records added, the number of cards read, the last accession no. and the total number of records in the new keyword file. TAPE10 is usually stored on a magnetic tape.

CREAD3

This program uses Record Manager for all I/O except for the OUTPUT file. The Primary Keyword File (PK) as created by CREAD2 is read and each record in broken up into 4 word records containing accession no., property code (packed into 1 word) and material code and 0 in words 3 and 4 of each record. These records are written to an intermediate file called PK1 which is then sorted by subroutine BIBSUB1 in accession no., property and material order. This sorted file, named PK1S, is then read, any duplicate records are eliminated and the final file NPM is written. This file should be made permanent, and catalogued on the private device set. OUTPUT contains the no. of records read, the no. of 4 word records created from them, the no. of duplicate entries and the no. of records in the final file.

DREAD1

This program performs the initial check of a 'box' of data cards (TAPE4) as described in the USER'S GUIDE, page 109. The Data Standards File is attached as TAPE1. Each card is checked for an accession no. between 1 and 99999, index no. between 0 and 9999, valid material, property, quantity and units codes, data type code between -6 and 6, number of associated references less than 51, environmental condition of up to 40 characters, multiplier that is decodable by an E10.0 format, either an * or an integer less than 10 for the number of independent variables, independent variable index numbers that are included in the list of associated reference numbers and an ampersand to end the entry. Subroutine NXCRD saves the last word of the previous card if it is to be continued, reads a new card and separates each character into a computer word. Subroutine WORDSEM finds the starting position and length of each "word" on the card where a "word" is defined as any number of characters separated by semi-colons. The last "word" on a card is assumed to continue to the next card unless it ends with a semi-colon or $ sign. Trailing blanks on a card are ignored if they follow immediately after a semi-colon or $ sign. A $ sign is used to end the information for each data entry.

DREAD2

This program reads the data cards previously checked by DREAD1 and corrected from TAPE4. This program also needs the Data Standards File attached as TAPE1. A permanent file named DIR should be requested. The existing Directory File must be attached as ODIR (old directory). This file is copied onto IDIR (intermediate directory) so that new records may be added to it. The program first reads from INPUT the option 'C' or 'U' meaning "create" or "update". If no data file yet exists, this option should be 'C' and a permanent file named ISD should be requested. If the data file already exists and should be added to, this option should be 'U' and the existing data file should be attached as ISD. Subroutine DREAD1 is called to re-check the data cards. If an error is found the program stops.

Subroutine SETREC is called to read the data cards and set up from these cards both a directory record and a data record as described on pages 29 and 30 . If only the keywords were entered but no data, then no data record is written. The directory record is written to file IDIR. The data record is written to file SDFI (intermediate sequential data file) under the create option, and is put directly on file ISD under the update option. ISD is an Indexed Sequential File with the 1st word of each record used as its index. Program ESTMATE was previously run for an Indexed Sequential File of 10000 records, key size of 10 characters, minimum record size of 3 words, maximum record size of 512 words and it suggested a buffer size of 2866 words which is used in this program. The system default values of 5110 characters, 5% padding for the index block size, 2 average size records for the data block size and 1 index level are used.

After this program has been run many times and several thousand data records have been put in the file, it might be advantageous to write a small program to copy ISD to a new indexed sequential file with a larger index block size. This could bring the number of index levels back to 1, thereby decreasing the random access retrieval time.

After all the cards have been read, the intermediate directory (IDIR) is sorted with subroutine BIBSUB1 by accession no., property and material. This sorted file (IDIRS) is then read and all duplicate records or keyword records which have been supplanted by their corresponding data pointer record, are eliminated while copying this file to the final directory file (DIR).

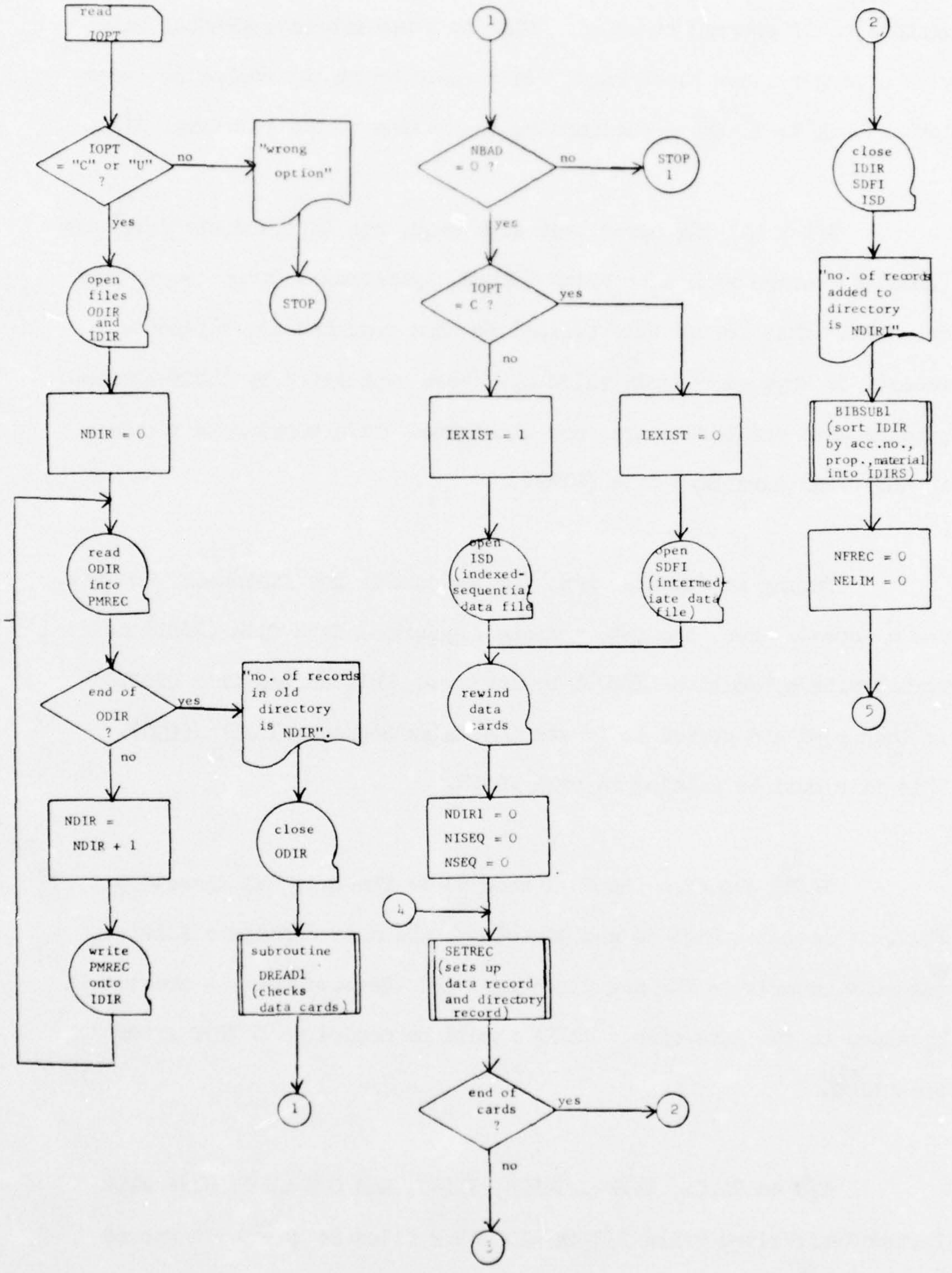
During an "update" run, the program is now finished. However, for a "create" run, the intermediate sequential data file (SDFI) is sorted with subroutine BIBSUB1 by index no. This sorted file (SDFIS) is then read and copied to create the Index Sequential File (ISD). This file must be catalogued with FO=IS.

TAPE5 contains the no. of records in the original directory, the no. of records added to the directory, the no. of duplicate entries, the no. of records in the new directory, and the no. of records created in or added to the data file. TAPE5 should be copied to OUTPUT after execution.

I/O on TAPE1, TAPE4, TAPE5, INPUT, and OUTPUT is done with Fortran Read/Writes while I/O on all other files is done with Record Manager in this program. A flow-chart of this program follows.

IBM **Flowcharting Worksheet**

Programmer: Ingrid Jarvis Program No.: ______ Date: 8/3/76 Page: 1 of 3
Chart ID: ____ Chart Name: ______ Program Name: DREAD2

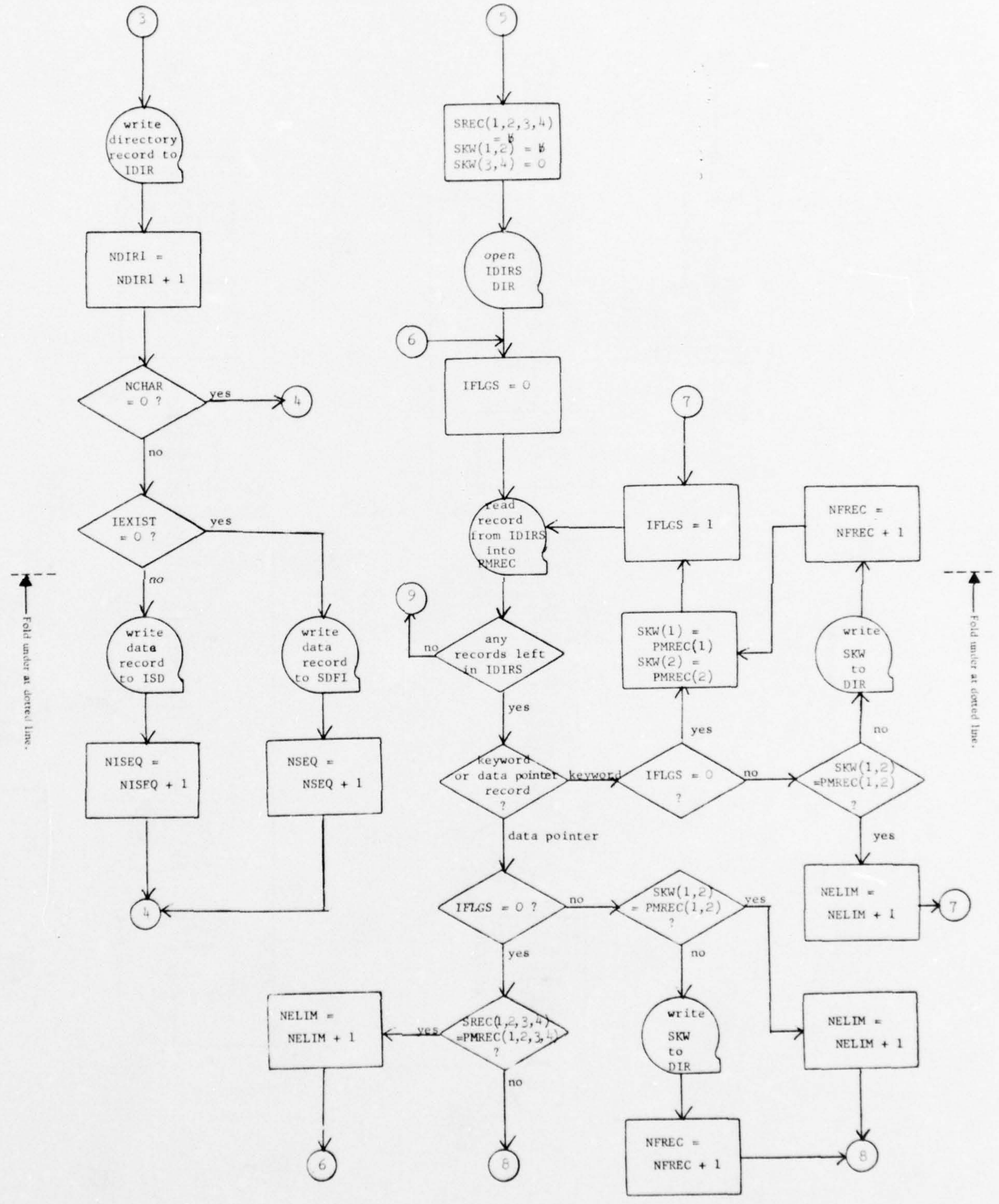


Fold under at dotted line.

Fold under at dotted line.

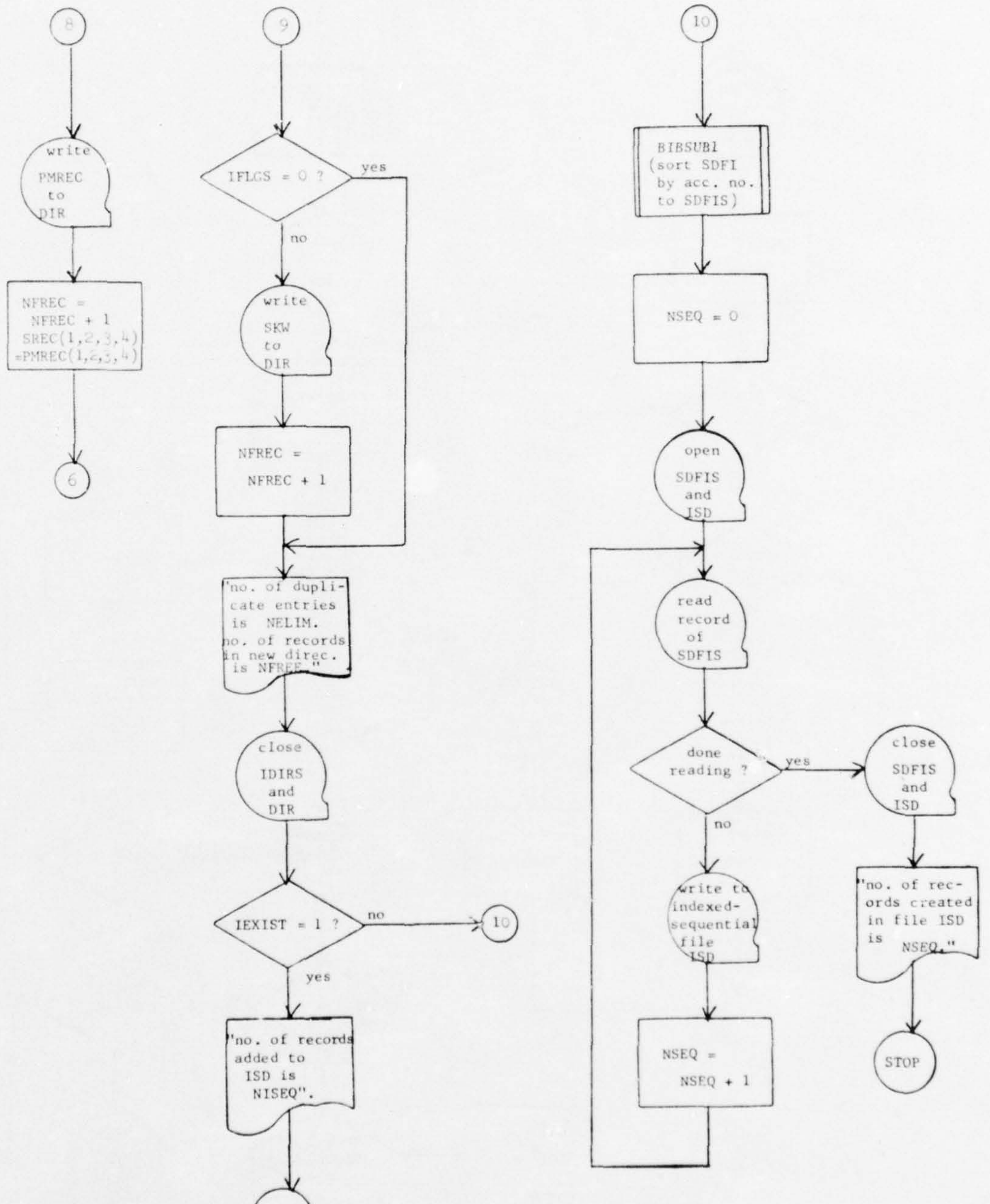
Flowcharting Worksheet

Programmer: Ingrid Jarvis	Program No.:	Date: 8/3/76	Page: 2 of 3
Chart ID:	Chart Name:	Program Name: DREAD2	



IBM Flowcharting Worksheet

Programmer: Ingrid Jarvis	Program No.:	Date: 8/3/76	Page: 3 of 3
Chart ID:	Chart Name:	Program Name: DREAD2	



8
write PMREC to DIR
NFREC = NFREC + 1 SREC(1,2,3,4) =PMREC(1,2,3,4)
6
9
IFLGS = 0 ?
yes
no
write SKW to DIR
NFREC = NFREC + 1
"no. of duplicate entries is NELIM. no. of records in new direc. is NFREC."
close IDIRS and DIR
IEXIST = 1 ?
no
10
yes
"no. of records added to ISD is NISEQ".
STOP
10
BIBSUB1 (sort SDFI by acc. no. to SDFIS)
NSEQ = 0
open SDFIS and ISD
read record of SDFIS
done reading ?
yes
no
write to indexed-sequential file ISD
NSEQ = NSEQ + 1
close SDFIS and ISD
"no. of records created in file ISD is NSEQ."
STOP

Fold under at dotted line.

Fold under at dotted line.

BIBDIS

This is the main program of the BIBDIS Operating System. The size of all COMMON storage areas are defined here. The File Information Table for the indexed sequential data file (ISD) is initialized here. Subroutine RDCOD is called to read the data inventory codes into an array. Subroutine INTERP reads cards from INPUT, interprets the directive code and stores the directive parameters. For directive, MAKER, the main program calls BIBSUB2 to make the requested random-access file. For the other directives, subroutines LISTB, SIZE & LISTF are called respectively. When all directives are completed, the indexed sequential data file is closed, if it has been opened by another routine, and execution stops. All the following program descriptions are for subroutines used by BIBDIS. A flow chart of BIBDIS is included.

RDCOD

This subroutine reads the data inventory file into arrays for use by the other subroutines. Only those codes that have a translation are saved.

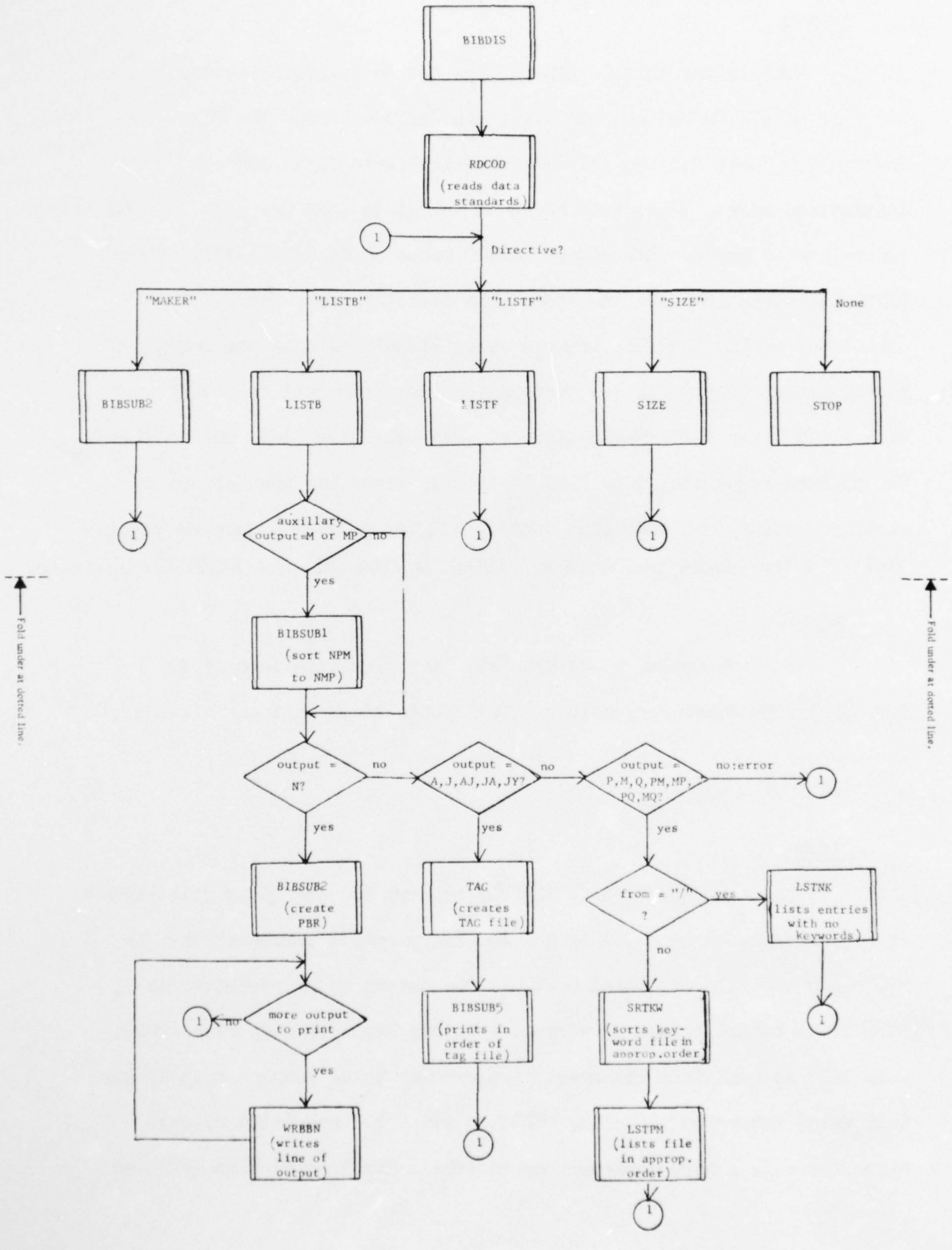
LISTB

This subroutine calls BIBSUB1 to sort the directory file, NPM, by accession no.and material if the auxillary output parameter is M or MP. This new file is called NMP. If the output order parameter is N, BIBSUB2 is called to create a word-addressable bibliographic file (FBR). This file is read from the starting parameter value requested until the last value requested and calls WRBBN to print the requested output. Since there is 1 more word per record (the control word) than is given

IBM Flowcharting Worksheet

PRINTED IN U.S.A.

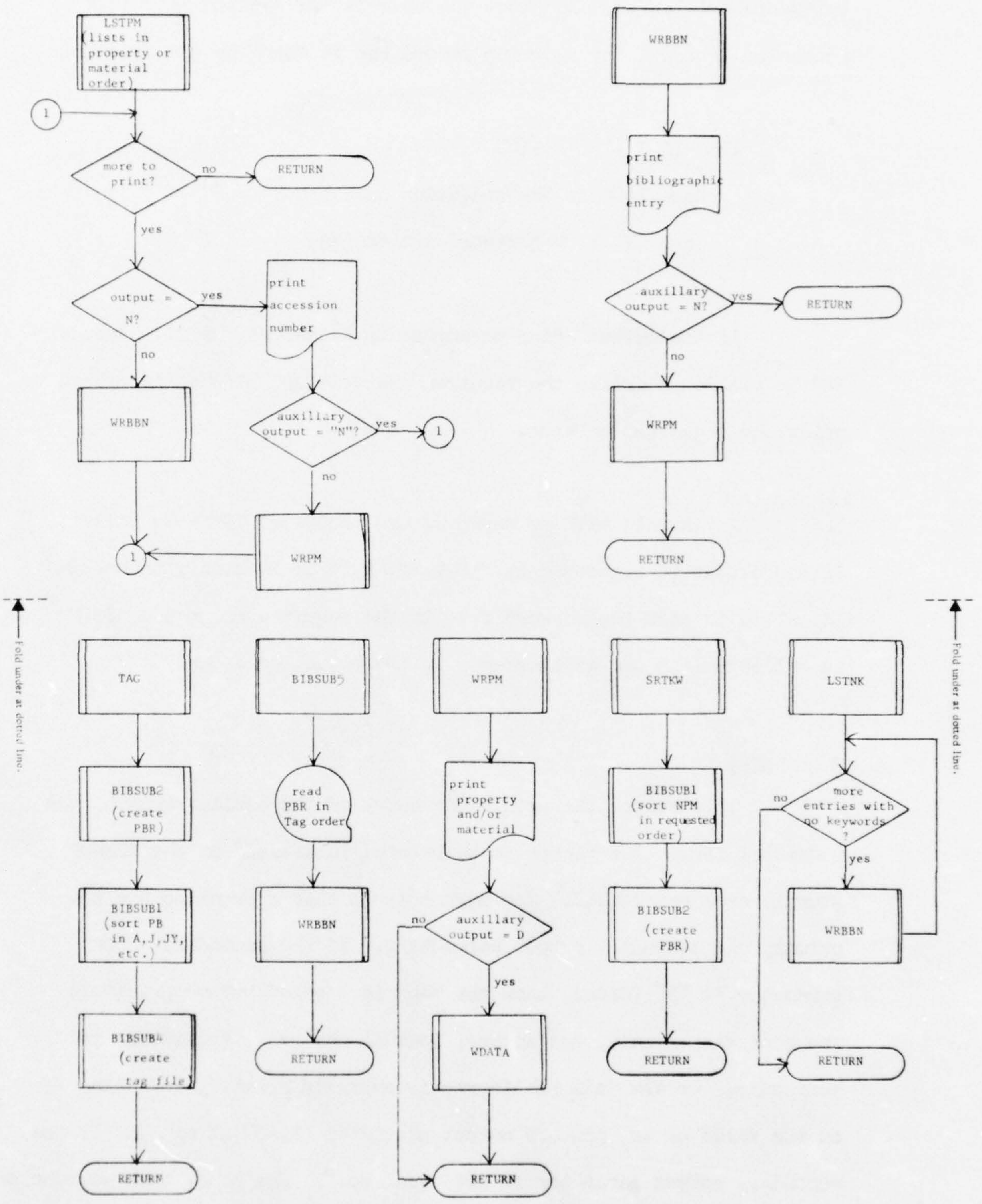
Programmer:	Program No.:	Date: 10/1/76	Page: 1 of 2
Chart ID:	Chart Name:	Program Name: BIBDIS	



BIBDIS
RDCOD (reads data standards)
1
Directive?
"MAKER"
"LISTB"
"LISTF"
"SIZE"
None
BIBSUB2
LISTB
LISTF
SIZE
STOP
1
auxillary output=M or MP
no
yes
BIBSUB1 (sort NPM to NMP)
Fold under at dotted line.
Fold under at dotted line.
output = N?
no
output = A,J,AJ,JA,JY?
output = P,M,Q,PM,MP,PQ,MQ?
no:error
yes
BIBSUB2 (create PBR)
TAG (creates TAG file)
from = "/" ?
LSTNK (lists entries with no keywords)
more output to print
BIBSUB5 (prints in order of tag file)
SRTKW (sorts key-word file in approp.order)
WRBBN (writes line of output)
LSTPM (lists file in approp. order)

IBM **Flowcharting Worksheet**

Programmer: ______ Program No.: ______ Date: 10/1/76 Page: 2 of 2
Chart ID: ____ Chart Name: ______ Program Name: BIBDIS



by the record length, and since the records are written in unique accession no.order, the starting record can be found by the formula:

IREC=IACC*(IRL+1)-IRL

where IACC is the accession no.

and IRL is the record length (32)

If the output order parameter is A,J,AJ,JA, or JY, subroutine TAG is called to create the required TAG file and BIBSUB5 is called to print the requested entries.

If entries with no keywords are requested LSTNK is called. If the output is requested in P,M,Q,PM,MP,PQ or MQ order, then SRTKW is called to sort the keyword file in the required order and LSTPM is called to list the bibliography in the requested order.

WRBBN

This subroutine prints one entry of the bibliography in the requested form. The record to be printed is assumed in the COMMON storage area named OREC. The arguments to this subroutine are the primary and secondary output parameters. If the secondary output parameter is "D" (data), then the page is ejected before printing. The accession no.only, the author, journal and year, the author and year only, or the full bibliographic entry is printed, depending on the value of the primary output parameter (N,AJY,AY or F). If the secondary output parameter is not equal to "N" for none, then subroutine WRPM is called to finish printing.

LSTNK

The arguments to this subroutine are the order of output and the primary and secondary output parameters. If the order of output is "P", "M" or "PM" all bibliographic entries for which no keyword exist are printed. If the order of output is "Q", "D", "PQ" or "MQ", then all bibliographic entries with no associated data are printed to accomplish this, both the primary bibliographic file, PB, and the directory file, NPM, are opened. Both files are read sequentially and any accession no. without an entry in NPM is printed for output order P, M or PM. Any accession no. without an entry in NPM that includes a pointer to the indexed sequential data file is printed if the order of output is Q, D, PQ or MQ. WRBBN is again used to do the actual printing.

TAG

This subroutine sets up the parameters to create a needed TAG file. The input argument to the subroutine is the order of output, either A, J, JY, JA, AJ or AY. The output argument is the name of the TAG file created. A word-addressable bibliographic file (PBR) is created, if it does not already exist, by calling BIBSUB2. This file would be required whenever a TAG file is used. Parameters are then set-up for BIBSUB1 to sort the Primary Bibliographic File (PB) in the appropriate order. BIBSUB4 is then called to read the sorted file and create a TAG file of accession nos. in the sorted order.

BIBSUB4

This subroutine, called from TAG, produces a "TAG" file which allows a word-addressable file to be read in TAG order, i.e. a sequential read of the TAG file produces a sorted read of the word-addressable file. The parameters required in COMMON storage area named PARAM are:

1) sorted sequential file name
2) output TAG file name (default is input file name concatenated with T)
3) word location in each record for the tag word (default is word 1 for the accession no.)
4) word location in each record for the key origin of the sort (default is word 2 for the author' name)

The sorted sequential file and the tag file are opened. If the tag file already exists nothing is done. The input file is read sequentially and a one word tag is created for each record. This word consists of the first five characters of the key origin (usually either the author or journal name) followed by 5 characters of the tag word (the accession no.). Sixty-three of these words are written in each record. The last record could be filled with ∅'s.

BIBSUB5

This subroutine is called by LISTB after TAG has been called. It is used to print the bibliography in author or journal order. COMMON storage area PARAM must contain:

1) name of file created by TAG
2) blank or PBR
3) "from" of the 1st delimiter
4) "to" of the 1st delimiter
5) "from" of the 2nd delimiter, if there is one, and
6) "to" of the 2nd delimiter.

The arguments to this subroutine include the order of output and the primary and secondary output parameters. The program first prints the appropriate heading on a new page. Then files PBR and the TAG file are opened for input. The TAG file is read sequentially and whenever the tag name falls within the limits requested by the "from" and "to" parameters, the corresponding record in the word-addressable bibliographic file is read. Since a word-addressable file has one more word per record than is given by the record length, the word address of a particular accession no. can be found by the following formula:

$$\text{Word address} = (\text{record length} + 1)*(\text{acc. no.} - 1)+1$$

If the bibliographic entry falls between the "from" and "to" parameters, it is printed by subroutine WRBBN. Whenever the value of the output parameter changes, it is printed as a sub-heading.

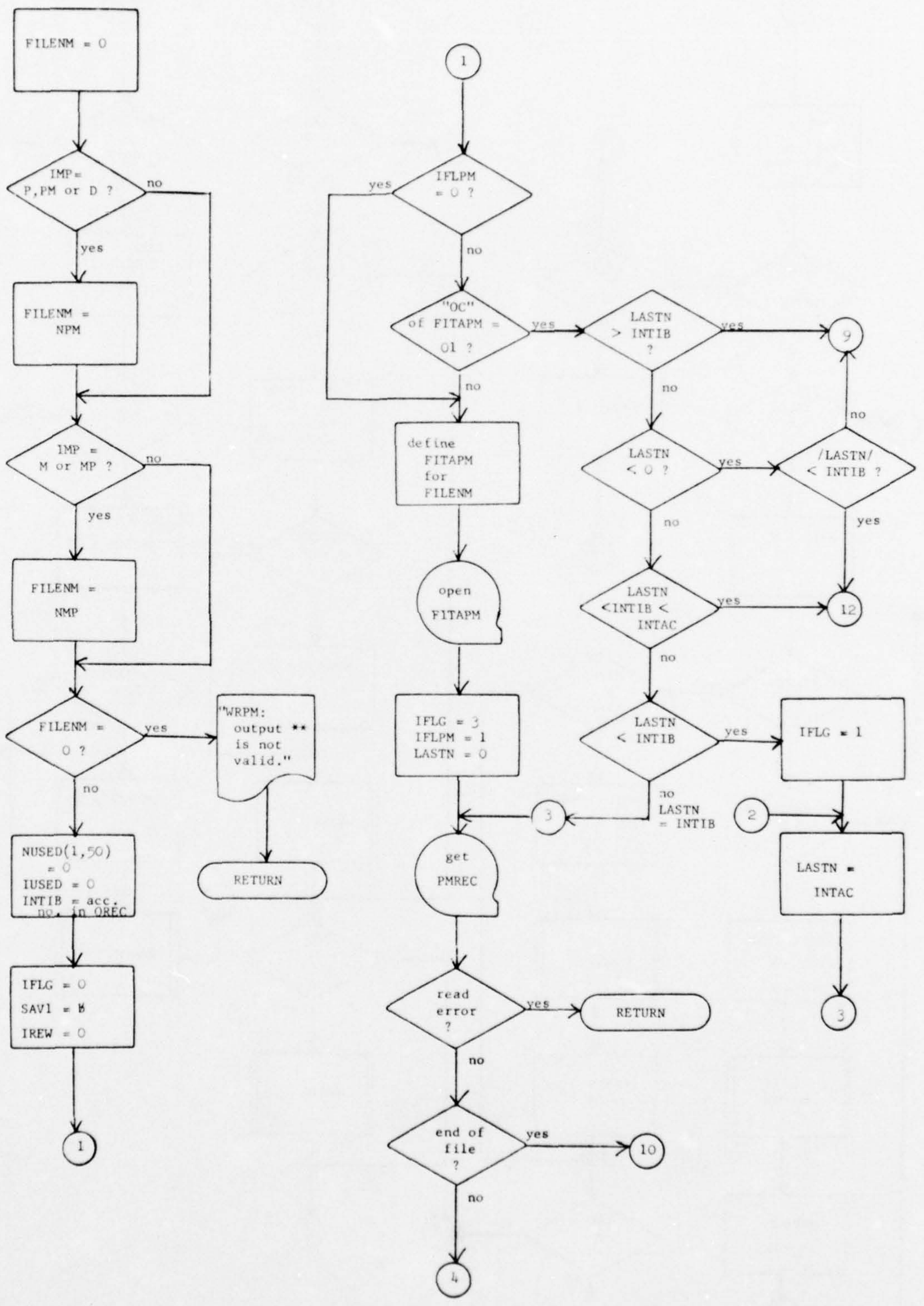
WRPM

This subroutine searches the directory file for all keywords associated with a given accession no. The requested accession no. is found in the COMMON storage area OREC. The argument to the subroutine is the secondary output parameter. If the value of this parameter is P, PM or D, the file searched is NPM. If the value is M or MP, then file NMP is searched. The first time through the program the file information table is set up and the file opened. Subsequent entries into this program start reading the file from wherever it was last read. When the first entry for the given accession no.is found, the requested output is printed (either property, material or both) by subroutines WPROP and WMATER. These values are also saved so that, as other entries are found, they need only be printed if they are different from the previous one printed. If the output requested was D for data, subroutine WDATA is called to find and print the data. A flow chart of WRPM is included here.

IBM **Flowcharting Worksheet**

Programmer: Ingrid Jarvis Program No.: Date: 8/17/76 Page: 1 of 3

Chart ID: Chart Name: Program Name: WRPM



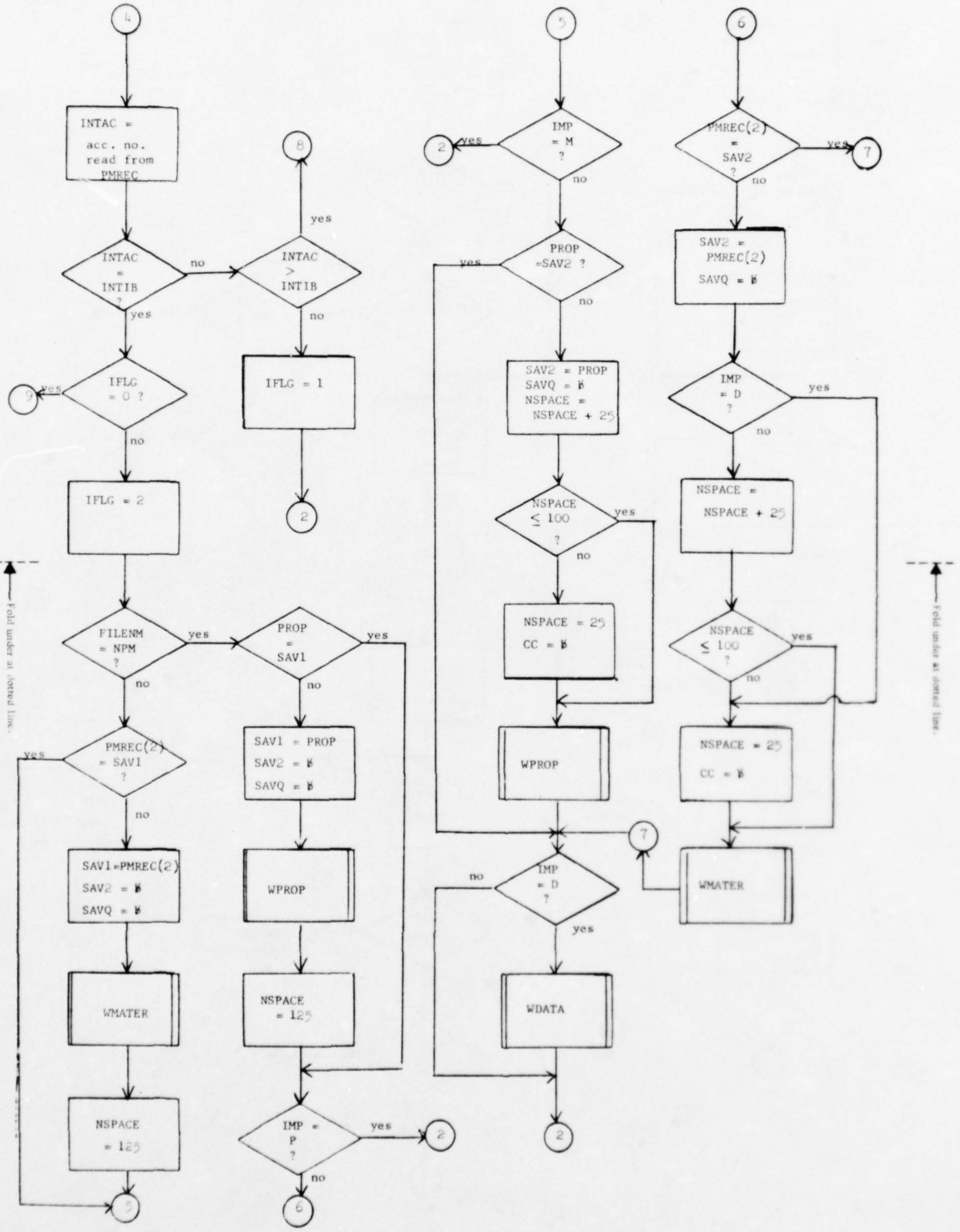
Fold under at dotted line.

Fold under at dotted line.

IBM Flowcharting Worksheet

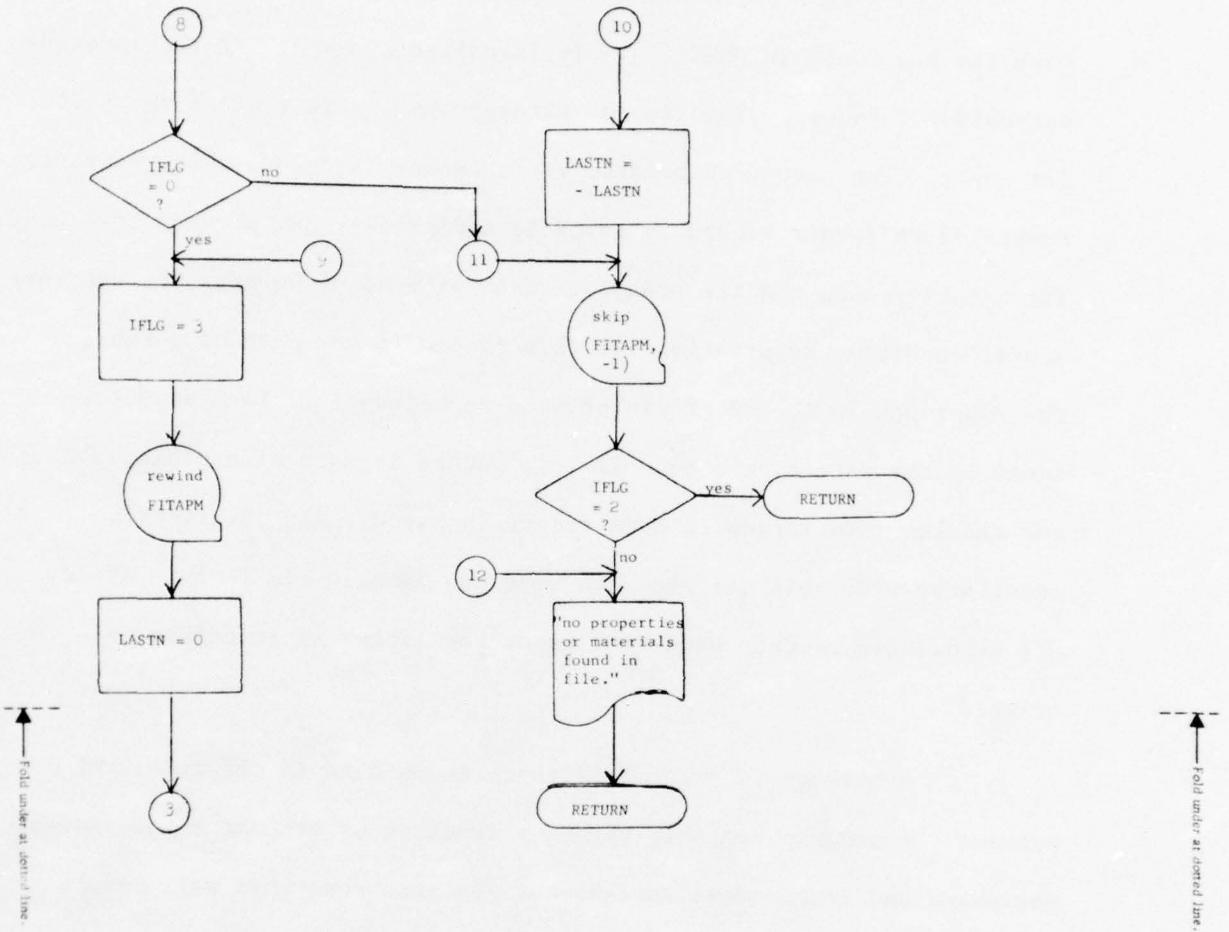
PRINTED IN U.S.A.

Programmer: Ingrid Jarvis	Program No.:	Date: 8/17/76	Page: 2 of 3
Chart ID:	Chart Name:	Program Name: WRPM	



IBM **Flowcharting Worksheet**

Programmer: Ingrid Jarvis	Program No.:	Date: 8/17/76	Page: 3 of 3
Chart ID:	Chart Name:	Program Name: WRPM	



WDATA

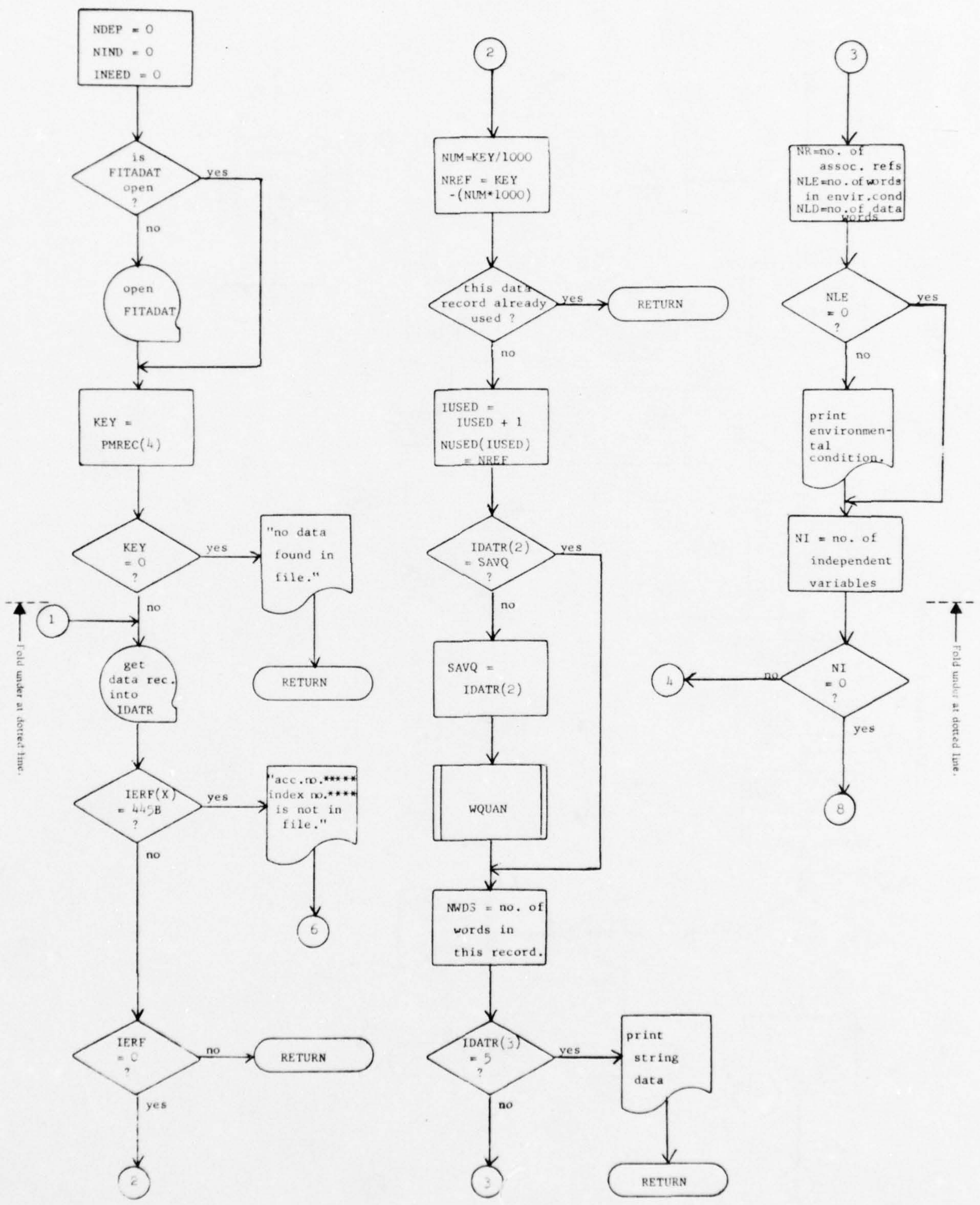
This subroutine reads and prints all the data associated with the key found in PMREC(4). It is called by WRPM. If the indexed-sequential data file (ISD) is not already open, this routine opens it for input. The record associated with the key is found. The reference number of this data record is saved to indicate it has already been used. The quantity code and its translation is printed by WQUAN. The environmental condition is printed. If this record is not part of a table, the data found in the record is printed in appropriate form as determined by the data type code. If this record is part of a table, all the tabular information in this record is saved. All the records associated with this one are also read and handled similarly. After all associated records have been read, the table, if it exists, is printed.

A message is printed if there is no data in the file. At present the program can only handle a function of at most 2 independent variables and 5 dependent variables. Any more than this will result in an error message. A flow chart for this program follows.

IBM **Flowcharting Worksheet**

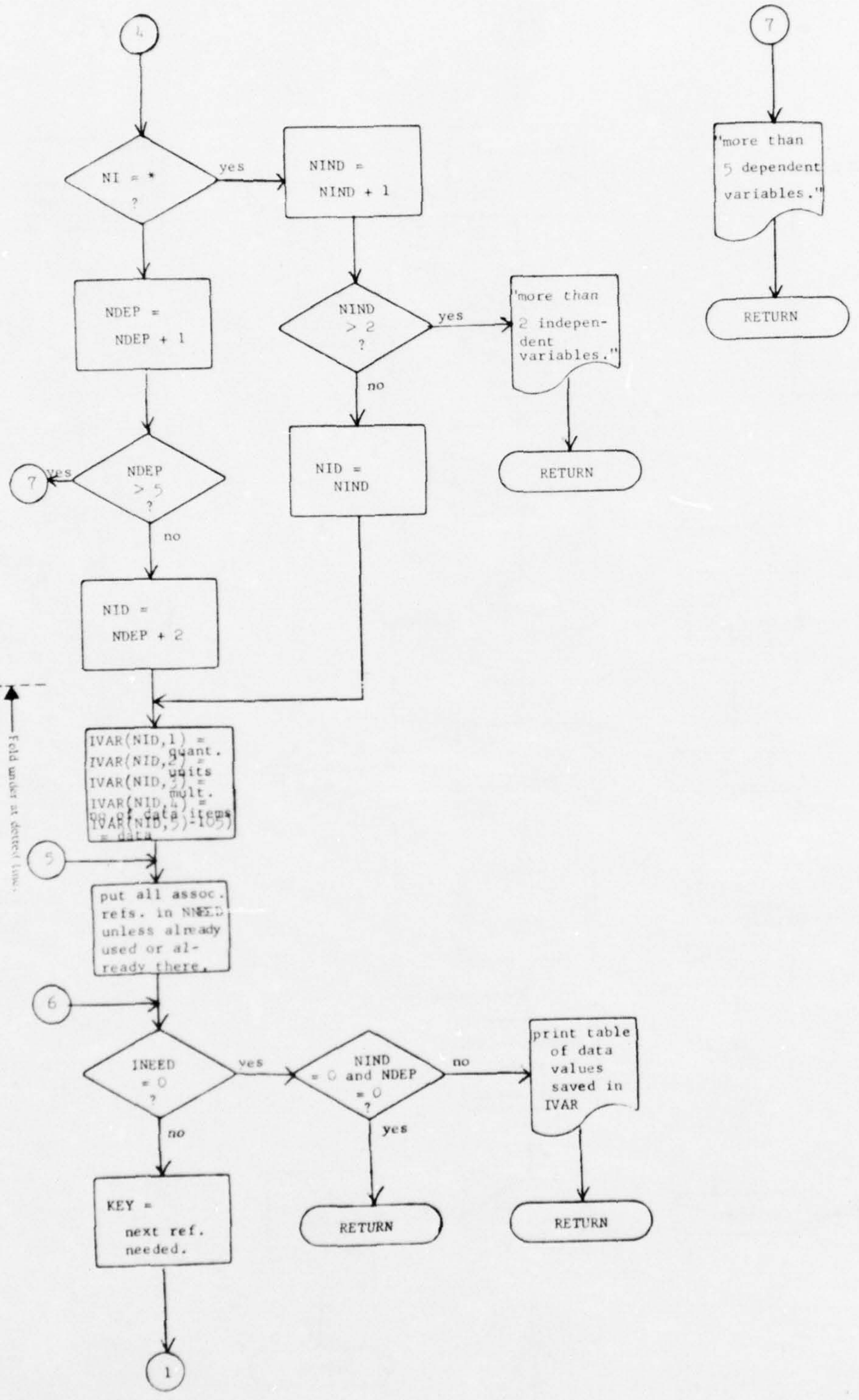
PRINTED IN U.S.A.

Programmer: Ingrid Jarvis	Program No.:	Date: 8/18/76	Page: 1 of 3
Chart ID:	Chart Name:	Program Name: WDATA	



IBM Flowcharting Worksheet

Programmer: Ingrid Jarvis	Program No.:	Date: 8/18/76	Page: 2 of 3
Chart ID:	Chart Name:	Program Name: WDATA	

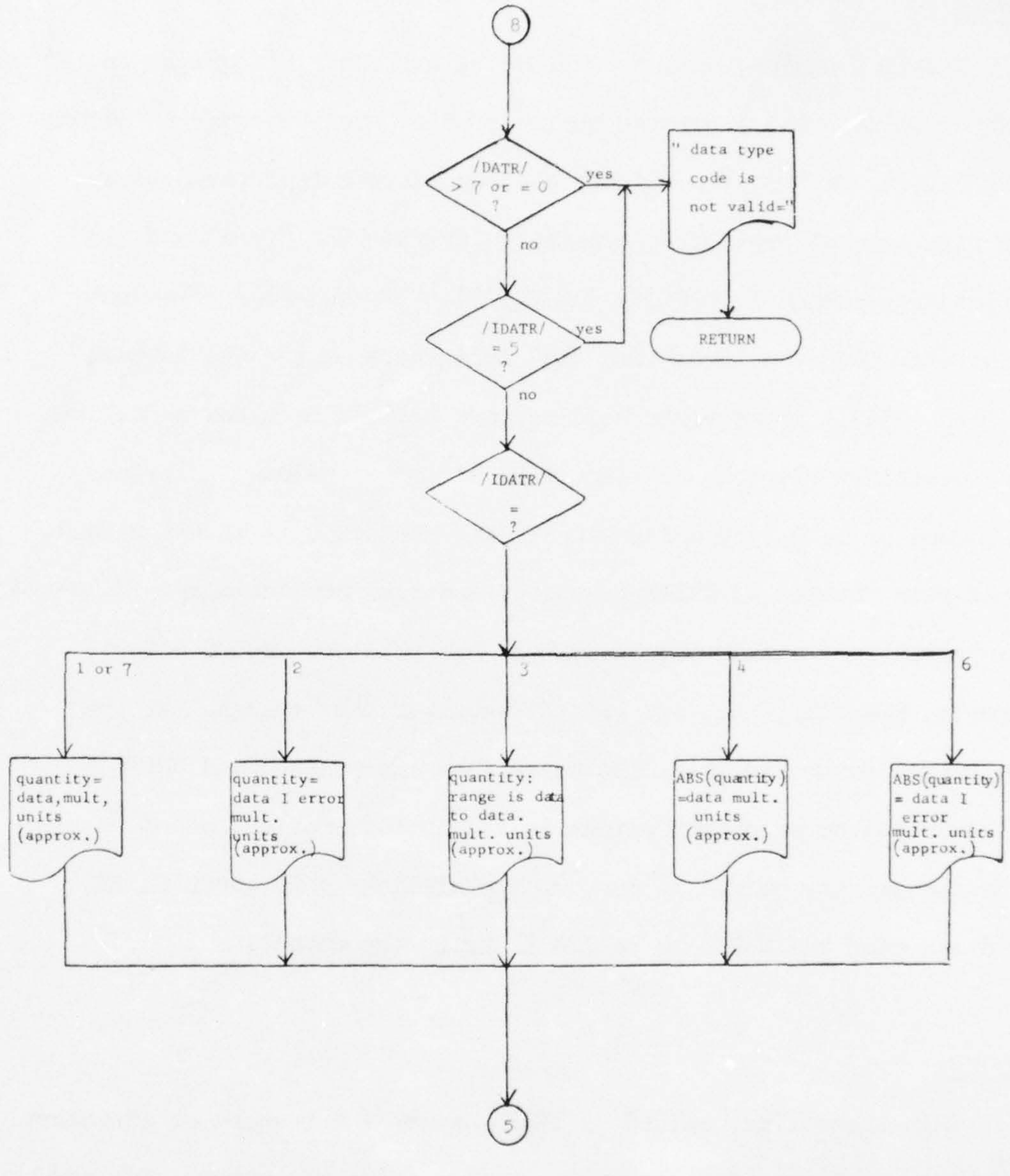


Fold under at dotted line.

Fold under at dotted line.

IBM **Flowcharting Worksheet**

Programmer: Ingrid Jarvis	Program No.:	Date: 8/18/76	Page: 3 of 3
Chart ID:	Chart Name:	Program Name: WDATA	


8
/DATR/ > 7 or = 0 ?
yes
no
" data type code is not valid="
RETURN
/IDATR/ = 5 ?
yes
no
/IDATR/ = ?
1 or 7
2
3
4
6
quantity= data,mult, units (approx.)
quantity= data I error mult. units (approx.)
quantity: range is data to data. mult. units (approx.)
ABS(quantity) =data mult. units (approx.)
ABS(quantity) = data I error mult. units (approx.)
5

Fold under at dotted line.

Fold under at dotted line.

LSTPM

This subroutine, called by LISTB, lists the bibliography in order of keywords. The arguments to the routine are the order of output (P,M,PM,MP,Q,PQ, or MQ), the primary and secondary output parameters and the file name as previously created by SRTKW. The "from" and "to" delimiters are assumed in COMMON, PARAM(1&2). If the order of output has 2 letters, then the "from" and "to" delimiters of the 2nd letters are in PARAM(3&4). Appropriate headings are written on a new page. The word-addressable bibliographic file (PBR) is opened unless the primary output parameter is N (accession no.only) in which case it is not needed. The input file created by SRTKW is opened and read sequentially. All accessions nos. whose keywords fall within the delimiters are printed. Sub-headings of property, material, etc. are printed whenever they change. If the primary output parameter is N, the accession no.is printed and WRPM is called unless no secondary output is requested. For any other value of the primary output parameter, the bibliographic entry in PBR is found and read and WRBBN is called to print the output.

SRTKW

This subroutihe, called by LISTB, sorts the keyword or directory file (NPM) in various orders depending on the order of output. The name of the sorted file is saved for use by LSTPM. If the order of output is P, then file PNM is created by sorting in order of property and accession no. If the order requested is M, file MNP is created in order of materials and accession no.

Similarly:

Order	File Created	Sorted by
PM	PMN	property and material
MP	MPN	material and property
Q	QNP	quantity and accession no.
PQ	PQN	property and quantity
MQ	MQN	material and quantity

The sorting is actually done by calling BIBSUB1. BIBSUB2 is also called to create a word-addressable bibliographic file (PBR) since it will be needed whenever the sorted keyword file is read.

LISTF

This subroutine is called by BIBDIS to list the auxillary file named in PARAM(1). If the file requested is the Data Standards File (DATST), it is rewound and read using standard Fortran I/O and listed. If the file is the Journal Word Abbreviation File (JWABRE) or the Journal Abbreviation File (JABREV), Record Manager calls are used for reading and standard Fortran I/O is used for printing.

SIZE

This subroutine is called by BIBDIS to print the size in words of the files PB,NPM,PBAT,PBJT,JWABRE,JABREV,DATST or ISD. The first 6 of these files are read with Record Manager and the record length in characters divided by 10 and multiplied by the number of records to find the size in words. File DATST is rewound and read using standard Fortran I/O.

This file has a record length of 8 words and the number of records is the sum of the numbers of material, property, quantity and units codes. File ISD is the only file with variable length records. Consequently, it's size is the sum of the number of words in each record. However, since it is an indexed sequential file, it actually requires more of disk storage than is given by SIZE because of Index Blocks.

BIBSUB1

This subroutine is used to create a sorted sequential file from an unsorted one. If the sorted file already exists, nothing is done. The input parameters which must be passed through the COMMON storage area named PARAM are:

1) sequential input file name
2) sorted sequential output file name
3) number of sort keys
4) key origin
5) key length
6) coding ID
7) order
8) display code sequence

repeat of last 5 parameters for each key.

This routine opens the files and calls SMSORT, SMFILE, SMKEY and SMEND within the SORT-MERGE package to actually do the sorting.

BIBSUB2

This subroutine creates a word-addressable file from a sequential file. If the file name to be created already exists, it does nothing. The input parameters which must be passed through COMMON storage area PARAM are:

1)sequential input file name

2) word-addressable output file name

If this parameter is blank, the file name will be the input file name concatenated with R. The 2 files are opened and a call to STOREF is used to put a 1 in WA so that writing will start at the first word of the file. All records are read sequentially from the input file and written on the output file.

The following chart shows the interrelationship of the files and the programs. (Figure II.1)

Files / Programs	PB	PBR	PK	DIR	(IDIR)	(SDFI)	(SDFIS)	ISD	DATST	JABREV	JWABRE	(PBXX)	PBXXT	MNP	PMN	QNP	PQN	MQN
CRJWA											W							
CRJAB										W								
BREAD1										R	R							
BREAD2	W									R	R							
CREAD1									R									
CREAD2			W						R									
CREAD3			R	W														
DREAD1									R									
DREAD2				R/W	W/R	W/R	W/R	W	R									
BIBDIS:																		
LISTB		R																
LSTNK	R			R														
WRPM				R												R		
RDCOD									R									
WDATA								R										
LSTPM		R		R												R		
LISTF									R	R	R							
SIZE	R			R				R	R	R	R		R					
BIBSUB5		R											R					
BIBSUB1	R			R								W				W		
BIBSUB2	R	W																
BIBSUB4												R	W					

where:

R - Files read
W - Files written
( )- Intermediate Files used only within program

Figure II.1 - Programs Use of Files

CHAPTER III USERS' GUIDE

This system of programs and data files has been developed to allow the user to maintain and retrieve bibliographic information and associated physical data on various materials using a simple set of computer "commands".

The first three sections of this chapter describe the creation and maintenance of the bibliographic file, the keyword file and the data file respectively. The final section describes the retrieval of information entered into these files. Consequently anyone interested only in retrieving information need only skim the first three sections before reading section 4 in detail.

It should be noted here that many different mediums for I/O are used within this system. All information is initially entered on punched cards. These cards may be copied onto permanent files for use by the programs or they may be run directly from cards. The files as well as the programs used by the display system will always reside on a private device set so that the user need not worry about available space or retention time. A back-up of these files is stored on magnetic tape.

Section 1.- Bibliographic Storage and Maintenance

General Comment on Source Citation

The primary purpose of the bibliographic file is to provide a repository for sources of information. It consists of "4 card" entries wherein each entry is coded by "accession number". This number is assigned by the software at data entry time and becomes the number which essentially ties the 3 main files: bibliographic, keyword and data together. The bibliographic file has several purposes, the main one being that of providing enough information so that any user may, if he desires and has the resources, eventually obtain the source material be it journal article, book or perhaps some industrial "flyer". Secondarily this file should provide a synopsis of source contents (mainly through "title"). It also should be structured in some form so that sources can be easily ordered in various forms e.g., by author or by year, etc.

Most entries (probably 90 °/o or more) are from journals, so in the later detailed description of entry preparation, it is assumed that the source is a journal article. However other sources do abound and provision should be made to "standardize" their entry as well as those of journal articles into the primary bibliographic file (PB). Therefore the following rules, or more exactly, guide lines are set forth in an attempt to provide some degree of standardization. It should always be remembered that the primary purpose of each entry is to provide an unambiguous guide to the original source.

A. General Rules - (card refers to an 80 column punch card)

a) The first card contains author or authors

b) The second card contains sufficient information (as a rule) to obtain the original source material

c) The third/fourth cards describe the material in the source (article title, etc.)

B. Fixed Rules - (mainly for search/display)

a) The year of publication of the source must appear as the last bit of information on card 2

b) All non-journal sources must be prefaced by an asterisk, dollar sign or number symbol (see below) on card 2

c) 4 cards must always be used with card number in column 80

C. Guide Lines for Various Source Types

1. <u>Journals</u>

Journal sources are covered in detail under Preparation and Verification of Bibliographic Sources. It should be pointed out that normally issue number (or month) need not be given since most scientific journals number pages consecutively throughout a volume. However, if this is not done issue should be included. If a journal comes out in more than one series or parts with the same volume number, then the identifier for such a subdivision (usually a letter or Roman numeral) should be included. Space for this information is provided in columns 41-50 on card 2.

2. <u>Conference Proceedings, Supplements, etc.</u>

If a journal publishes special issues during the year with a volume number, then such issues should be cited as a journal with appropriate notation in columns 41-50 (card 2). For example SUPPL., CONF., SYMP., PROC. followed by an appropriate number if there is one. Otherwise such sources should be treated as books (see below) using the full title of the source as given by the publisher.

3. Books

As a general rule, a book is one entity by one or more authors or a collection of parts written by several authors with one or more editors. In either case, author(s) should appear on card 1 as for journals. Editor(s) should appear starting in column 51 of card 1 and be prefaced with the abbreviation ED-. Occasionally no author appears. In particular this may happen with a government publication. If so, in lieu of author(s)/editor(s), cite the publisher e.g. U. S. Government Printing Office.

The book title, including volume number, should appear (in full, if possible) on card 2 between columns 2 and 65. If abbreviations must be used try to follow standards which have been adopted for journal word abbreviations (see ref. 8 or the journal word abbreviation file described in this chapter). The year of publication should appear in columns 66-70. An asterisk must appear in column 1.

Cards 3 and 4 should be used for part title if the book is a collection of parts by several authors. It may be used for chapter title (include chapter number) if this appears to be appropriate to further describe the source material.

Example: (not formatted)

card 1:

```
    STROKE,GN                      ED- FLUGGE,S          1
```

card 2:

```
    *ENCYCLOPEDIA OF PHYSICS    VOL. XXIX   1967         2
```

card 3:

```
    DIFFRACTION GRATINGS                                 3
```

card 4:

```
    blank                                                4
```

4. Government Reports (not appearing as books)

Some government reports may appear without author. If so, leave card 1 blank.

Card 2 should cite the report identifier (start in column 2), which should include the acronym of the reporting government office. The year of publication, only, should appear in columns 66-70. It may on occasion be necessary to further identify the government laboratory by either including the full name of the lab and/or its location. See references 10, 11 for a list of recognized acronyms. If the report identifier does not include the office acronym, the acronym should appear starting in column 2, followed by the report number.

A dollar sign should appear in column 1 of card 2 to indicate that this is a government report.

Cards 3 and 4 should contain the full title of the report.

Example:

card 1:

```
 SAHAGIAN,CS,PITHA,CA                                     1
```

card 2:

```
 $AFCRL-72-0170                          1972             2
```

card 3,4:

```
 COMPENDIUM ON HIGH POWER INFRARED LASER WINDOW
 MATERIALS (LQ-10 PROGRAM)
```

5. Industrial Reports

Author card as for government reports.

Card 2 should start with the symbol "#" followed immediately by the company name and location (city, state only). Use standard post office abbreviations for states. This should be followed by the report identifier assigned by the company. The year of publication should appear as always in columns 66-70. If (and only if) the company does not assign a report identifier per se, then include any information which might help in identifying the source. For example, the report might be a Final Report for a government contract in which case the document identifier might appear as:

```
#PARKE MATH.LABS.,INC.,CARLISLE,MA FIN.RPT.,CONT.#F1968-73-C-0125
                                                1975    2
```

Cards 3 and 4 contain the title of the report as usual.

In some instances the source document is not in the form of a report, for example it may be a brochure or advertisement giving information on the physical properties of some material which the company manufactures. In this case include on card 2 the company name/address as above followed by any company identifier (code words and/or numbers). Cards 3 and 4 should then contain the title of the brochure or, lacking that, a description of the brochure.

5. <u>University Reports</u> (not including theses)

These should be treated in the same manner as industrial reports. The names of the better known universities may be abbreviated, e.g. MIT, but location should be included. If the report does not have a university report number, it may be necessary to cite the source laboratory. For example, if the Radiophysics Laboratory at Dartmouth College publishes a report with no report number, card 2 might appear as

```
#DARTMOUTH C., HANOVER,NH  (RADIOPHYSICS LAB.)  1973    2
```

6. <u>Theses</u>

Card 1 contains the author as usual.

Card 2 starts with the university name and address as for university reports followed by the type of thesis, e.g. PHD THESIS, followed by year of thesis appearance. As for university and industrial reports, # should appear in column 1.

Cards 3 and 4 contain the thesis title.

D. Guide Lines for Author Citation.

Author citation (if not editor) starts in column 1 of card 1. The name of each author should appear with last name first (in full) followed by a comma and initials with no space or other punctuation mark between initials. The names of authors should be seperated from one another by a single comma. The author's name should not include such parts as JR. (for junior) or III (for the third). If an author's last name consists of more than one part such as "Van Vleck", the two parts should be joined by a dash to become, for example, VAN-VLECK.

Frequently it is necessary to include the names of editors, particularly for book citations. Names of editors should appear following the code "ED-" which starts at column 51. The rules for citing editors' names are the same as those given for authors.

Occasionally it may not be possible to list all authors in the space available. If so, the abbreviation "ET AL" should appear after the last name listed with a blank as the only separator.

The following example illustrates the above rules:

```
PETIT,RH,FERRE,J,BADOZ,J                         ED-GRAHAM,CD ET AL
```

Preparation and Verification of Bibliographic Entries

The primary bibliographic file (PB) consists of fixed length records of 32 10-character words each. Each such record holds a complete bibliographic entry consisting of the author(s), journal, page(s), title and accession number. Each such entry originates as a set of four 80 column cards as shown in Figure III.1.

Each card contains a card number (1-4) in column 80. It should be noted that an entry always consists of 4 cards even though 1 or more cards may be blank except for card number. The accession number is added to each entry during the creation of the PB. A typical set of 4 cards is shown in Figure III.2. A PB record can be imagined by putting the 4 cards in a row, replacing each card number by a blank space and placing the (right justified) accession number in the front.

When gathering information for the bibliographic file, it soon becomes clear that it is necessary to standardize the abbreviations of acceptable journal names. This is done by keeping a file of Journal Abbreviations, JABREV. To help in abbreviating owrds within the journal name a Journal Word Abbreviation File (JWABRE) is also kept. These files have already been created and stored and their present contents are shown on pages 84 and 90. If you wish to add to or change the contents of these files refer to pages 80-91 of this section for details. An * in column 1 of the journal card is used to indicate that the source citation is not a journal and should therfore not be checked against the file.

The first step in either creating or updating the PB is to process one or more "conveniently" sized sets of card entries using program BREAD1. A convenient size is usually one box of approximately 2000 cards so that below we refer to the input file to BREAD1 as a BOX or BOXn where n is an integer. This grouping of entries into boxes has significance also in the fact that, since the card entries do not have accession number, it is convenient (and prudent) to store the cards in boxes with first and last accession number of entries in the box recorded somewhere (like on the box itself). Then, if it is ever necessary to recreate the PB from cards; they can be put back-in in the correct sequence. (Do not depend on getting a box of cards back from the computer center in the same box however).

Program BREAD1 checks and attempts to correct as far as possible these bibliographic entry cards. Three output files are produced by the program. Two of these are print files for user information and the third is a corrected bibliographic entry file which may be further modified in EDITOR or transferred directly into the primary bibliographic file using BREAD2. The following is a list of the functions which BREAD1 performs:

1) It searches for a card number (1 through 4) as the last item on a card (not necessarily in column 80). Cards with missing numbers are flagged.

2) It attempts to put author cards in correct form. The correct form is best explained by an example:

   LOESCHER,DH,DETRY,RJ,CLAUSER,MJ

   i.e. last name followed by comma followed by initials with no blanks or periods.

   The program will abbreviate first names and put other abbreviation forms into the correct form.

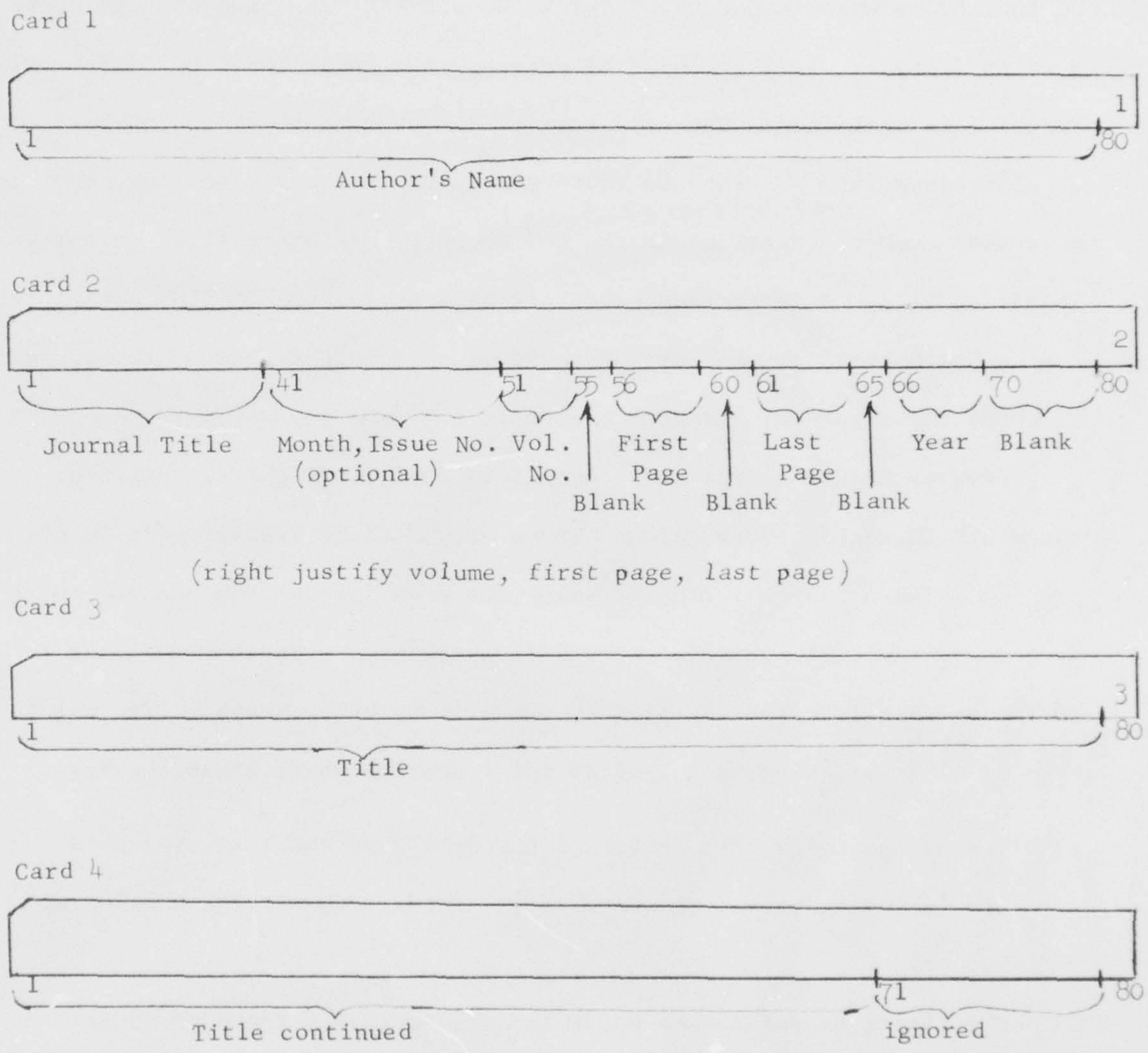
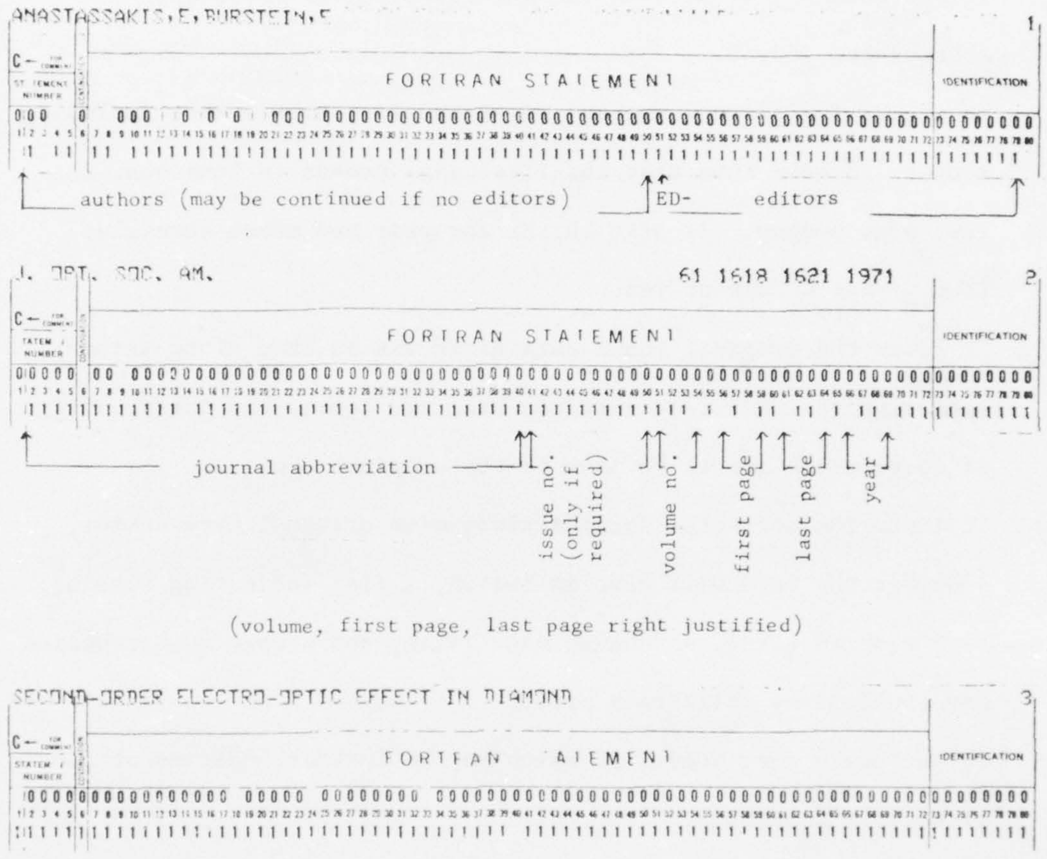


Figure III.1 - Bibliographic Entry Cards


FORTRAN STATEMENT
IDENTIFICATION
authors (may be continued if no editors)
ED-
editors
journal abbreviation
issue no. (only if required)
volume no.
first page
last page
year
(volume, first page, last page right justified)

```
ANASTASSAKIS,E,BURSTEIN,E                                    1
J. OPT. SOC. AM.                        61 1618 1621 1971    2
SECOND-ORDER ELECTRO-OPTIC EFFECT IN DIAMOND                 3
```

Card 4 not shown but should be included with 4 in column 80.

Figure III.2 - Sample Bibliographic Entry Cards

3) It attempts to find correct abbreviations for all words in a journal title.

4) It searches a journal file for a match with the resultant abbreviated journal.

5) It checks for journal number, first and last page number including a check to make sure that the first page number is less than the last page number. It also checks for year and makes sure that $1800 \leq \text{year} \leq \text{current year}$.

6) It lists the original input data as it was punched along with a card sequence number so that the user can locate bad cards easily if corrections are to be made to the original deck.

7) It lists the corrected entries along with original card number, a number the card will have in EDITOR, a flag indicating missing card numbers (1-4), a "change made" flag, and a code number beside cards which may still have errors.

8) It outputs a corrected file which may be further modified or fed directly into BREAD2.

The following flagging conventions are used to assist the user in evaluating the output of BREAD1. These flags appear on file TAPE4 which lists the corrected cards along with card sequence numbers and EDITOR sequence numbers.

1) If a card number is missing the card is flagged with ***** as are cards following it until card 1 appears.

2) If any change is made to the original card it is flagged with <<<<<.

3) If a card is questionable, the card is flagged with one or more error code numbers with the following significance:

1	bad year (less than 1800 or greater than current year)
2	bad last page number (not a number)
3	page number too long (>4 characters)
4	first page > last page
5	bad volume number (The volume "number" does not contain any numbers.)
6	bad author configuration (can not be put into standard form)
7	bad or non-journal (could not be found in journal abbreviation file)
8	no author.

Whenever new journal titles are entered, the journal abbreviation files (JABREV) and journal word abbreviation files (JWABRE) should be updated. The output of BREAD1 can only assist the user in obtaining a valid bibliographic entry. Each entry should be checked for correct spelling, etc. which the computer can not do.

File Descriptions:

```
BREAD1(TAPE1,TAPE2,TAPE3,TAPE4,JWABRE,JABREV,OUTPUT)
```

TAPE1	-	formatted input file containing the bibliography cards or card images
TAPE2	-	formatted output file. This is a printable file containing a list of the input cards with card sequence numbers included.
TAPE3	-	formatted output file. This file contains the corrected card images. It is non-printable (unless COPYSBF is used) but suitable for EDITOR. When this file is brought into EDITOR using the command `EDIT,TAPE3,S` lines will have the same edit sequence number which appears on TAPE4.
TAPE4	-	formatted output file. This is a printable file which lists the corrected cards with card sequence numbers, edit sequence numbers and flags and error codes.
JWABRE	-	unformatted journal word abbreviation file. This file contains the journal word abbreviation records (sorted "numerically"). See page 80 for details on creating this file.
JABREV	-	unformatted journal abbreviation file. This file contains the journal abbreviations (sorted "numerically"). See page 87 for details on creating this file.

OUTPUT - system output file. Also entry count and last entry in JABREV are printed.

Note: Files 1 - 4 are rewound by the program at job initiation; files 2 and 4 are rewound at program termination.

A sample job might be:

```
BARBR,CM60000,T30.
ATTACH(BRB,BREAD1BX3296,ID=PITHA)
ATTACH(JABREV,JABREVX3296,ID=PITHA)
ATTACH(JWABRE,JWABREX3296,ID=PITHA)
ATTACH(TAPE1,BX4X3296,ID=PITHA,CY=1)
REQUEST(TAPE3,*PF)            (if TAPE3 is to be stored)
BRB.
COPY(TAPE2,OUTPUT)
COPY(TAPE4,OUTPUT)
CATALOG(TAPE3,BX4CX3296,ID=PITHA)    (if TAPE3 is to be stored)
```

The above job assumes that the input has been previously stored on a permanent file. The same set of commands can be used on INTERCOM except replace the job card with CONNECT(OUTPUT).

If the input is in the input stream i.e. control deck is placed in front of the deck containing the initial entries, remove ATTACH(TAPE1,...) and replace BRB. with BRB(INPUT).

Creation of Journal Word Abbreviation File

This file is created from cards as described in Figure III.3 consisting of the 4 letter search code followed by a number indicating the number of possible abbreviations corresponding to the code, followed by the abbreviations in descending order by length. If all the abbreviations corresponding to a code cannot fit on 1 card, 3 asterisks are to be punched in the last 3 columns of the card and continue up to 3Ø columns on the next card. The word to be abbreviated is then matched letter by letter to abbreviation words until all letters match (up to the end of the abbreviation). When (or if) such a match is found, the abbreviation is accepted. There are some instances where this procedure does not work. For example ELEC may lead to the abbreviation ELECTRON. (for ELECTRONICS) or ELECTRON. To handle this situation, the abbreviation word ELECTRONI$ is used. The symbol $ signifies replace the preceeding symbol with a period. There are other examples (few)where abbreviation is not so straight forward and a special convention is used for these. For example, JAPANESE is abbreviated JAP. while JAPAN is abbreviated JPN. For this the series JAPANE$$$ JAPAN*JPN. is used where $$$ means place the period 3 characters back and * means use the abbreviation that follows.

These cards are then processed with program CRJWA. JWABRE, the output of this program, is the sorted sequential (unformatted) word abbreviation file.

File Descriptions:

CRJWA(TAPE1,OUTPUT)

TAPE1 - formatted input file containing the journal word abbreviations, up to 127 cards.

OUTPUT - system output file. Also contains the last card read from TAPE1.

A sample job for creating JWABRE might be:

```
job,CM60000,T20.
REQUEST(JWABRE,*PF)
FTN.
LIBRARY(COBOL)
RFL(60000)
LGO(INPUT)
CATALOG(JWABRE,JWABREX3296,ID=PITHA,RP=999)
7/8/9
   [programs CRJWA and BIBSUB1]
7/8/9
   [journal word abbreviations - up  to 127 cards]
6/7/8/9
```

After this file has been created satisfactorily, it may be copied to the device set by the following job.

```
BARCJ,CM12000,T30.
PAUSE. THIS JOB USES DEVICE SET VSN PITHA
MOUNT(VSN=PITHA,SN=BIBLIO)
REQUEST(JWABRE,*PF,SN=BIBLIO)
ATTACH(JW,JWABREX3296,ID=PITHA)
COPY(JW,JWABRE)
CATALOG(JWABRE,ID=PITHA)
6/7/8/9
```

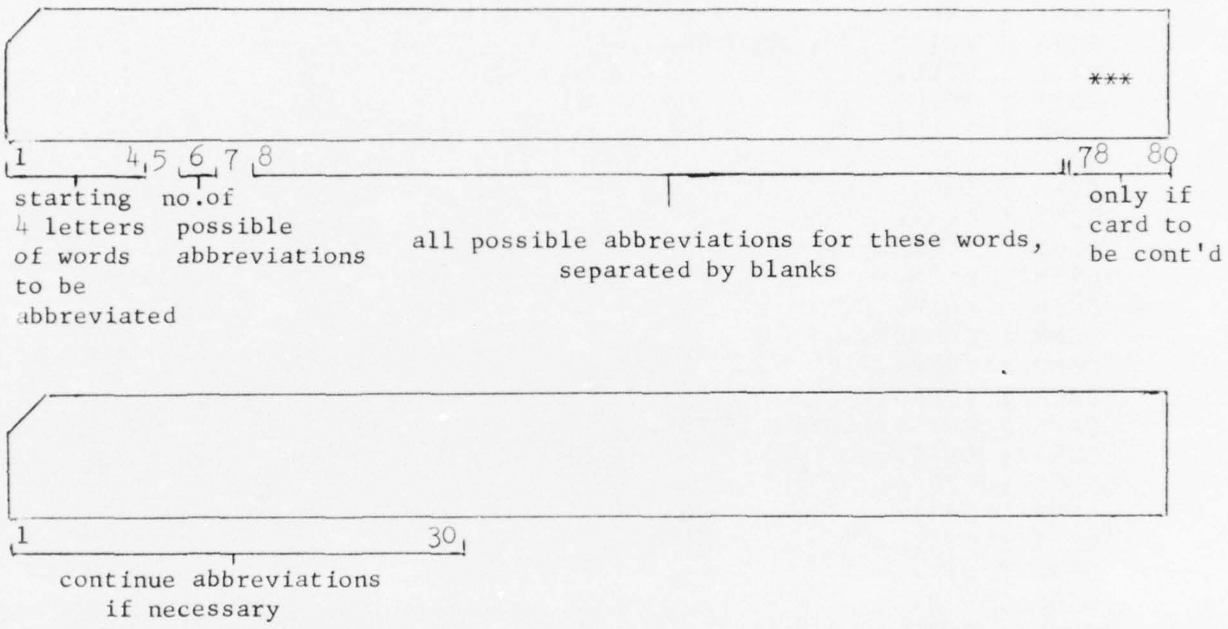

***
1
4
5
6
7
8
78
80
starting 4 letters of words to be abbreviated
no.of possible abbreviations
all possible abbreviations for these words, separated by blanks
only if card to be cont'd
1
30
continue abbreviations if necessary

Figure III.3 - Journal Word Abbreviation Cards

```
JWABRE: JOURNAL WORD ABBREVIATION FILE
ACAD 1 ACAD.
ACOU 1 ACOUST.
ADVA 1 ADV.
AKAD 1 AKAD.
AKUS 1 AKUST.
ALLG 1 ALLG.
AMER 1 AM.
ANNA 1 ANN.
ANNU 1 ANNU.
ANOR 1 ANORG.
APPL 1 APPL.
ARKI 1 ARK.
ASTR 2 ASTROPHYS. ASTRON.
BIOL 1 BIOL.
BRIT 1 BR.
BULL 1 BULL.
BURE 1 BUR.
CANA 1 CAN.
CENT 1 CENT.
CERA 1 CERAM.
CHEM 1 CHEM.
CHIM 1 CHIM.
COMM 1 COMMUN.
COMP 1 COMPT.
COPE 1 COPEN.
CRYS 2 CRYSTALLOGR. CRYST.
CURR 1 CURR.
CZEC 1 CZECH.
DANS 1 DAN.
DESI 1 DES.
DEVE 1 DEV.
DIGE 1 DIG.
DOKL 1 DOKL.
ECON 1 ECON.
EKSP 1 EKSP.
ELEC 3 ELECTROCHEM. ELECTRO-OPT. ELECTRONI$
ELEK 2 ELEKTROCHEM. ELEKTROTECH.
ENGI 1 ENG.
ENGL 1 ENGL.
EXPE 1 EXP.
FENN 1 FENN.
FIZI 1 FIZ.
FYSI 1 FYS.
GEOG 1 GEO.
GEOL 1 GEOL.
GRAD 1 GRAD.
HELV 1 HELV.
INDU 1 IND.
INFO 1 INFO.
INOR 1 INORG.
INST 2 INSTRUM. INST.
INTE 1 INT.
IZVE 1 IZV.
```

```
JAPA 2 JAPANESE  JAPAN*JPN.
JOUR 1 J.
KHIM 1 KHIM.
KRIS 5 KRISTALLOGRAPHIA KRISTALLGEOM. KRISTALLPHYS. KRISTALLOGR. KRIST.
LETT 1 LETT.
MAGA 1 MAG.
MAGN 1 MAGN.
MATE 2 MATER. MAT.
MATH 1 MATH.
MECH 1 MECH.
MEKH 1 MEKH.
MEMO 1 MEM.
META 2 METALL. MET.
MINE 1 MINERAL.
MODE 1 MOD.
NATI 1 NAT.
NATU 2 NATURFORSCHER NATURFORSCH.
NEOR 1 NEORG.
NON- 1 NON-CRYST.
NOUV 1 NOUV.
NUCL 1 NUCL.
OBZO 1 OBZ.
OPTI 2 OPTIK OPT.
PAPE 1 PAP.
PHYS 2 PHYSICOL. PHYS.
POLO 1 POL.
PRIB 1 PRIB.
PROC 1 PROC.
PROG 1 PROG.
PROM 1 PROM*PROMST.
REFE 1 REF.
REND 1 R.
REPO 1 REP.
RESE 1 RES.
REVI 1 REV.
ROYA 1 R.
RUSS 1 RUSS.
SCAN 1 SCAND.
SCIE 1 SCI.
SECT 1 SECT.
SELS 1 SELSK.
SEMI 1 SEMICOND.
SERI 1 SER.
SOCI 1 SOC.
SOVI 1 SOV.
SPEC 1 SPEC.
SPEC 3 SPECTROCHIM. SPECTRA SPECTR.
SPEK 1 SPEKTROSK.
STAN 1 STAND.
STAT 3 STATUS STATE STAT.
STRU 1 STRUKT.
SUPL 1 SUPL.
SUPP 1 SUPPL.
```

```
SYST 1 SYST.
TECH 2 TECHNOL. TECH.
TEKH 1 TEKHN.
TEOR 1 TEOR.
THEO 1 THEORET.
TRAN 2 TRANSL. TRANS.
TVER 1 TVERD.
UCHE 1 UCHEBN.
ULTR 1 ULTRASON.
UNIV 1 UNIV.
VACU 1 VAC.
VIDE 1 VIDENSK.
VYSS 1 VYSSH.
ZAVE 1 ZAVED.
ZEIT 1 Z.
ZHUR 1 ZH.
```

Creation of Journal Abbreviation File

This file is created from cards containing one journal abbreviation per card as described in Figure III.4. These cards are processed with program CRJAB and then sorted to form the final journal abbreviation file (JABREV).

File Descriptions:

CRJAB(TAPE1,OUTPUT)

TAPE1	- formatted input file containing up to 255 journal abbreviations.
OUTPUT	-system output file. Also contains the last card read from TAPE1.

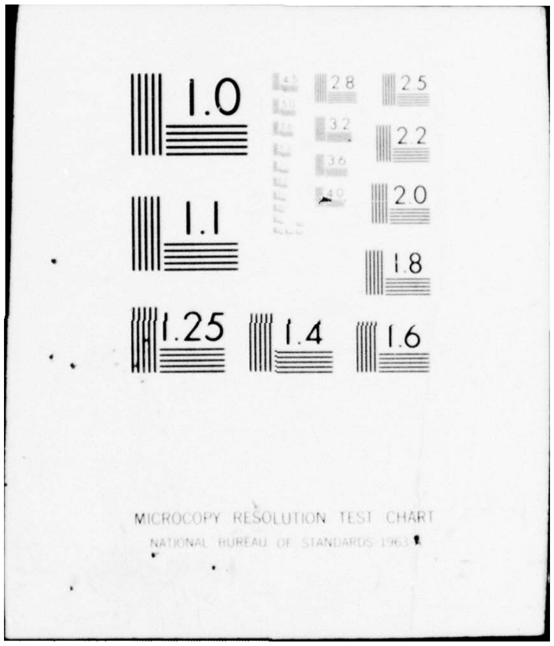
A sample job for creating JABREV might be:

```
job,CM600000,T20
REQUEST(JABREV,*PF)
FTN.
LIBRARY(COBOL)
RFL(600000)
LGO(INPUT)
CATALOG(JABREV,JABREVX3296,ID=PITHA,RP=999)
7/8/9
  [programs CRJAB and BIBSUB1]
7/8/9
  [journal title abbreviations - up to 255 cards]
6/7/8/9
```

AD-A037 738. PARKE MATHEMATICAL LABS INC CARLISLE MASS F/G 5/2
A COMPUTER STORAGE AND RETRIEVAL SYSTEM FOR SCIENTIFIC DATA.(U)
SEP 76 T B BARRETT, I B JARVIS F19628-74-C-0031
UNCLASSIFIED RADC-TR-76-344 NL

2 OF 2
AD A037738

END
DATE
FILMED
4-77


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

This file may be copied to the device set by the following job.

```
BARCJ,CM12000,T30.
PAUSE.  THIS JOB USES VSN=PITHA
MOUNT(VSN=PITHA,SN=BIBLIO)
REQUEST(JABREV,*PF,SN=BIBLIO)
ATTACH(JA,JABREVX3296,ID=PITHA)
COPY(JA,JABREV)
CATALOG(JABREV,ID=PITHA)
6/7/8/9
```

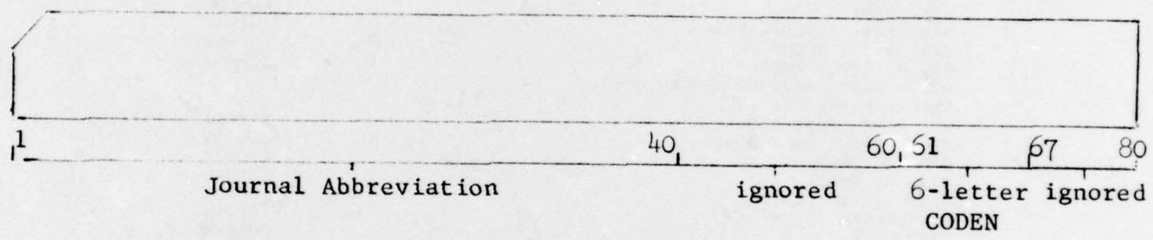


Figure III.4 - Journal Abbreviation Cards

JOURNAL ABBREVIATIONS AND CODEN

ACTA CHEM. SCAND. SER. A	ACAPCT
ACTA CHEM. SCAND. SER. B	ACBOCV
ACTA CHEM. SCAND.	ACSAA4
ACTA CRYSTALLOGR. SECT. A	ACACBN
ACTA CRYSTALLOGR. SECT. B	ACBCAR
ACTA CRYSTALLOGR.	ACCRA9
ACTA ELECTRON.	ACELAZ
ACTA PHYSIOL. POL. ENGL. TRANSL.	APHPAH
ACTA PHYSIOL. POL. SUPL.	APPSCY
ACTA PHYSIOL. POL.	APYPAY
ADV. PHYS.	ADPHAH
AKUST. ZH.	AKZHAE
AM. J. PHYS.	AJPIAS
AM. MINERAL.	AMMIAY
ANNU. REV. MATER. SCI.	ARMSCX
ANNU. REV. PHYS. CHEM.	ARPLAP
ANN. ACAD. SCI. FENN SER. A1 MATH. PHYS.	AFMPA6
ANN. ACAD. SCI. FENN SER. A2	AAFCAX
ANN. ACAD. SCI. FENN SER. A3	AAFGAB
ANN. ACAD. SCI. FENN SER. A4	AAFBAU
ANN. ACAD. SCI. FENN SER. A5	AFMAAT
ANN. ACAD. SCI. FENN SER. A6	AAFPA4
ANN. ACAD. SCI. FENN SER. A	ASFABI
ANN. CHIM. PHYS.	ACPHAA
ANN. PHYS. LEIPZIG	ANPYA2
ANN. PHYS. PARIS	ANPHAJ
APPL. OPT. SUPPL.	APOSAR
APPL. OPT.	APOPAI
APPL. PHYS. LETT.	APPLAB
ARK. FYS.	AFYSA7
ARK. KEMI	ARKEAD
ARK. MAT. ASTRON. FYS.	AMAFAX
BELL SYST. TECH. J.	BSTJAN
BR. J. APPL. PHYS.	BJAPAJ
BULL. AKAD. NAUK SSSR SER. PHYS.	
BULL. AM. CERAM. SOC.	
BULL. AM. PHYS. SOC.	BAPHA6
BULL. CHEM. SOC. JPN.	BCSJA8
CAN. J. CHEM.	CJCHAG
CAN. J. PHYS.	CJPHAD
CHEM. PHYS. LETT.	CHPLBC
CRYOGENICS	CRYOAX
CURR. SCI.	CUSCAM
CZECH. J. PHYS.	CZYPAO
DAN. VIDENSK. SELSK. COPENHAGEN	
DOKL. AKAD. NAUK SSSR SER. A	DASABO
DOKL. AKAD. NAUK SSSR SER. BIOL.	DASBAQ
DOKL. AKAD. NAUK SSSR SER. GEOL.	DASGA7
DOKL. AKAD. NAUK SSSR SER. KHIM.	DASKAJ
DOKL. AKAD. NAUK SSSR SER. MATER. FIZ.	DAMFA8
DOKL. AKAD. NAUK SSSR	DANKAS
ECON. GEOL.	ECGLAL
ELECTRON. DES.	ELODAW
ELECTRO-OPT. SYST. DES.	EOSDA5
ELEKTROTECH. OBZ.	EKOBAJ
EXP. MECH.	EXMCAZ
FIZ. TEKH. POLUPROVODN	FIPPA4
FIZ. TVERD. TELA KHARKOV	FZTTAA
FIZ. TVERD. TELA LENINGRAD	FTVTAC
FOREIGN SCI. BULL.	
FREQUENCY	
GEOL. SOC. AM. MEM.	GSAMAQ
GEOL. SOC. AM. SPEC. PAP.	GSAPAZ
GLASS IND.	GLINAK
GROWTH CRYST. USSR	GRCRAA
HELV. PHYS. ACTA SUPPL.	HPYSAI
HELV. PHYS. ACTA	HPACAK
HEWLETT-PACKARD J.	
IBM J. RES. DEV.	IBMJAE
IEEE J. QUANTUM ELECTRON.	IEJQA7
IEEE TRANS. ELECTRON DEVICES	IETDAI
IEEE TRANS. MICROWAVE THEORY TECH.	IETMAB
IEEE TRANS. SONICS ULTRASON.	IESUAU
INDIAN J. PHYS.	IJPYAS
INDIAN J. PURE APPL. PHYS.	IJOPAU
INFRARED PHYS.	INFPAD
INORG. CHEM.	INOCAJ
INORG. MATER. USSR	INOMAF
INORG. NUCL. CHEM. LETT.	INUCAF
INSTRUM. CONTROL SYST.	INCSA7
INSTRUM. EXP. TECH. USSR	INETAK
INT. J. ELECTRON.	IJELAR
IRE TRANS. ULTRASON. ENG.	IRUEAF
IZV. AKAD. NAUK. SSSR. NEORG. MATER.	IVNMAW
IZV. VYSSH. UCHEBN. ZAVED. FIZ.	IVUFAC
JETP. LETT.	JTPLA2
JPN. J. APPL. PHYS.	JJAPA5
J. ACOUST. SOC. AM.	JASMAN
J. AM. CERAM. SOC.	JACTAW
J. AM. CHEM. SOC.	JACSAT
J. APPL. PHYS.	JAPIAU
J. CHEM. PHYS.	JCPSA6
J. CHEM. SOC.	JCSOA9
J. CRYST. GROWTH	JCRGAE

J. ELECTROCHEM. SOC.	JESOAN
J. ELECTRON. MATER.	JECMAS
J. GRAD. RES. CENT.	JGFLAV
J. INDIAN INST. SCI.	JIISAD
J. NON-CRYST. SOLIDS	JNCSBJ
J. OPT. SOC. AM.	JOSAAH
J. PHYS. C	JPSOAW
J. PHYS. CHEM. REF. DATA	JPCRBU
J. PHYS. CHEM. SOLIDS	JPCSAW
J. PHYS. CHEM.	JPCH[illegible]
J. PHYS. D	[illegible]
J. PHYS. E	[illegible]IAE
J. PHYS. PARIS	JPQCAK
J. PHYS. RADIUM	JPRAAJ
J. PHYS. SOC. JAP.	JUPSAU
J. RES. NATL. BUR. STAND. SECT. A	JNBAAR
J. RES. NATL. BUR. STAND. SECT. C	JNBCAX
J. RES. NATL. BUR. STAND.	JRNBAG
J. SCI. INSTRUM.	JVSTAL
J. VAC. SCI. TECHNOL.	JVSTAL
KRISTALLOGRAFIA	KRISAJ
KRIST. TECH.	KRTSAW
LASER TECHNOL.	
MAGN. MAGN. MATER. DIGEST	MMMDAF
MATER. RES. BULL.	MRBUAC
MATER. RES. STAND.	MTRSAW
MINERAL. MAG.	MNLMBB
NATURE LONDON	NATUAS
NATURFORSCHER	NAFSAK
NOUV. REV. OPT.	NVROBC
NUCL. INSTRUM. METHODS	NUIMAL
NUOVO CIMENTO	NUCIAD
OPTIK STUTTGART	OTIKAJ
OPT. ACTA	OPACAT
OPT. COMMUN.	OPCOB8
OPT. MEKH. PROMST.	OPMPAQ
OPT. SPECTRA	OPTSA7
OPT. SPEKTROSK.	OPSPAM
PHYSICA THE HAGUE	PYSIA7
PHYSICA UTRECHT	PHYSAG
PHYS. CHEM. GLASSES	PCGLA6
PHYS. CHEM. SOLIDS	PCSOA7
PHYS. LETT.	PHLTAM
PHYS. REV. B	PLRBAQ
PHYS. REV. LETT.	PRLTAO
PHYS. REV.	PHRVAO
PHYS. STATUS SOLIDI	PHSSAK
PHYS. STATUS SOLIDI A	PSSABA
PHYS. STATUS SOLIDI B	PSSBBD
PHYS. TODAY	PHTOAD
PRIB. TEKH. EKSP.	PRTEAJ
PROC. AM. ACAD. ARTS SCI.	PAAAAV
PROC. INDIAN ACAD. SCI. SECT. A	PISAA7
PROC. INDIAN ACAD. SCI. SECT. B	PISBAA
PROC. NATL. ACAD. SCI. INDIA	NAIPAQ
PROC. NATL. ACAD. SCI. USA	PNASA6
PROC. PHYS. SOC. LONDON	PPSOAU
PROC. R. SOC. LONDON	PRSLAZ
PROG. OPT.	POPTAN
REP. PROG. PHYS.	RPPHAG
REV. MOD. PHYS.	RMPHAT
REV. SCI. INSTRUM.	RSINAK
RUSS. J. PHYS. CHEM.	RJPCAR
SOLID STATE COMMUN.	SSCOA4
SOLID STATE ELECTRON.	SSELA5
SOLID STATE PHYS.	SSPHAE
SOV. J. OPT. TECHNOL.	SJOTBH
SOV. J. QUANTUM ELECTRON.	SJQEAF
SOV. PHYS.-CRYSTALLOGR.	SPHCA6
SOV. PHYS.-JETP.	SPHJAR
SOV. PHYS.-SEMICOND.	SPSEBY
SOV. PHYS.-SOLID STATE	SPSSA7
SOV. PHYS.-TECH. PHYS.	SPTRA3
SPECTRA	
SPECTROCHIM. ACTA	SPACA5
SURFACE SCI.	SUSCAS
THEOR. CHIM. ACTA	TCHAAM
THIN FILMS	THFIA3
THIN SOLID FILMS	THSFAP
TRANS. METALL. SOC. AIME	TMSAAB
ULTRASONICS	ULTRA3
ZH. EKSP. TEOR. FIZ. PISMA RED	ZFPRAU
ZH. STRUKT. KHIM.	ZSTKAI
Z. ANORG. ALLG. CHEM.	ZAACAB
Z. ANORG. CHEM.	ZACMAH
Z. CHEM.	ZECEAL
Z. ELEKTROCHEM.	ZEELAI
Z. KRISTALLOGR. KRISTALLGEOM. KRISTALLPHYS.	ZEKGAX
Z. NATURFORSCH.	ZNTFAZ
Z. PHYS. CHEM. LEIPZIG	
Z. PHYS.	

Processing of Bibliographic Entries

The final step in creating or adding to the PB is to process the corrected "BOXES" using program BREAD2. This program will process up to 4 "BOXES" at one time (by one run of the program). It is at this stage that an accession number is added to each entry; these accession numbers are added according to the sequence in which the BOXES are processed. It should be pointed out that it is possible to bypass verification of the cards to create or add to the PB by going directly to the final step. This might be done for instance if it is known that the original card entries are correct. However, BREAD2 only checks the card no. from 1 to 4 in column 80 of each entry to the PB since it can (and frequently does) happen that cards are misread (and even destroyed) by the CDC hardware. As soon as a single bad entry is found (as indicated by improper card numbers) the program terminates. BREAD2 indicates what BOXES have been added to the PB and the accession numbers to be associated with each box. It also lists all entries which have been added by accession number. Finally BREAD2 copies the entire bibliography onto a new file, perferably on the device set.

File Descriptions:

BREAD2(TAPE1,TAPE2,TAPE3,TAPE4,TAPE5,TAPE9,TAPE10,OUTPUT)

TAPE1 - TAPE4	- up to 4 formatted input files containing the corrected bibliographic cards or card images.
TAPE5	- formatted output file containing the total no. of records in the new file and a listing of all the new bibliographic entries with their accession number.

TAPE9 - old unformatted bibliographic file.

TAPE10 - new unformatted bibliographic file.

OUTPUT - system output file.

N.B. - Files 1-4 and 9 are rewound by the program during initialization.

Sample Deck Set-up:

```
BARB2,CM60000,T30,TP1.
VSN(TAPE9=CC4277)
ATTACH(BRB,BREAD2BX3296,ID=PITHA)
REQUEST(TAPE9)    (CC4277/NORING)PITHA
REWIND(TAPE9)
ATTACH(TAPE1,BX4CX3296,ID=PITHA)
PAUSE. THIS JOB USES PACK VSN=PITHA
MOUNT(VSN=PITHA,SN=BIBLIO)
REQUEST(TAPE10,*PF,SN=BIBLIO)
BRB.
REWIND(TAPE10,TAPE9,TAPE5)
CATALOG(TAPE10,PB,ID=PITHA)
COPY(TAPE5)
```

The above job assumes that the corrected input file has been stored on permanent files as BX4CX3296. It also assumes the old bibliography is on magnetic tape CC4277 and the new bibliography is to be put on the device set VSN=PITHA,SN=BIBLIO. Since this job uses a magnetic tape and a device set, it cannot be run on intercom, except in the BATCH mode.

Example: (punched cards to primary bibliographic file)

The following is an example of the JOBS which might be used in adding two boxes of bibliographic entries to the existing PB. Each JOB consists partly in a control record which, in turn consists of several control statements. A control statement usually consists of a control word followed by parameters, which modify this word, in parentheses. In the following each control statement is given a reference number so that it may be easily referred to in comments following the JOB. In addition to the control record a JOB may also include one or more data records following the control record. In some of the examples below the data records may be written explicitly or merely indicated by (---).

<u>Step 1</u>

Process two boxes with two JOBS using program BREAD1.

```
BARB1(CM60000,T20)                        (1)
REQUEST(TAPE3,*PF)                        (2)
ATTACH(BRB,BREAD1BX3296,ID=PITHA)         (3)
ATTACH(JWABRE,JWABREX3296,ID=PITHA)       (4)
ATTACH(JABREV,JABREVX3296,ID=PITHA)       (4a)
BRB(INPUT)                                (5)
COPY(TAPE2,TAPE4)                         (6)
CATALOG(TAPE3,BOX5X3296,ID=PITHA)         (7)
7/8/9                                     (8)
 (box of bibliographic entries - box 5)
6/7/8/9                                   (9)
```

(1) The job card consists of the job name, the first 3 letters of which should be the same as the programmer's name. Following the job name are parameters which give the amount of central memory, CM, required and the maximum central processor time, T. Not shown in the above example are two mandatory "comments". A project number (4 digit number in columns 50-53) and the authorized programmer's name starting in column 58.

(2) TAPE3 is the logical file name of the output of BREAD1 which is suitable for use with utility program EDITOR. This file is to be made "permanent" so that another JOB can access the file for corrections if necessary.

(3) This assumes that the binary version of program BREAD1 has been catalogued under the permanent file name BREAD1BX3296 and with the identifier (ID) PITHA. BRB is merely the arbitrary logical file name by which this file can be referred to within the JOB.

(4) These 2 ATTACH cards make the journal abbreviation and journal word abbreviation files available to the program.

(5) This statement causes execution of BREAD1. The parameter INPUT indicates that the input (a box of cards) to BREAD1 is a data record within the JOB. (The default input to BREAD1 is TAPE1.)

(6) The command causes TAPE2 and TAPE4 to be copied onto the printer.

(7) This causes TAPE3 to become a permanent file. At the time of writing a permanent file with no retention parameter (RP) specified will be kept for only 5 days.

(8) 7/8/9 means multi-punch 7,8,9,in column 1. It is the standard end-of-record indicator for the INPUT file.

(9) 6/7/8/9 is the standard end-of-file (or end-of-job) indicator. The symbol means multi-punch 6,7,8,9, in column 1.

(Note that the AFCRL computation center requires that the first card of a JOB be blue and the last one green with the green card such that the upper left corner is not cut-off.)

The second box of cards (say box 6) is processed using an identical JOB. The only difference being that control statement (7) might be replaced by:

```
CATALOG(TAPE3,BOX6X3296,ID=PITHA)
```

<u>Step 2</u>

Correct the BOXES (the two TAPE3's) created above using utility program EDITOR. Program UPDATE could also be used to correct these boxes. It's use is described in Appendix A . After the boxes have been corrected, re-run BREAD1 for a final check.

Step 3

Add the two corrected BOXES to the PB . The JOB control record is completed as follows:

```
VSN(TAPE9=CC4277)                           (1)
ATTACH(BRB,BREAD2BX3296,ID=PITHA)           (2)
REQUEST(TAPE9)  (CC4277/NORING)PITHA        (3)
REWIND(TAPE9)
PAUSE. THIS JOB USES PACK VSN=PITHA         (4)
MOUNT(VSN=PITHA,SN=BIBLIO)                  (5)
REQUEST(TAPE10,*PF,SN=BIBLIO)               (6)
BRB.                                        (7)
REWIND(TAPE5)
COPY(TAPE5)                                 (8)
CATALOG(TAPE10,PB,ID=PITHA)                 (9)
```

(1) A VSN control statement must be used before a magnetic tape file can be referenced later in the JOB. Each magnetic tape is assigned a CC number by the computation center.

(2) BREAD2BX3296 is the permanent file name of the binary version of program BREAD2.

(3) This REQUEST card comments informs the computer operator that the tape is to be mounted without the ring, so that it can only be read by this job and that the owner of the tape is PITHA. The owner's name also appears on a label on the tape reel. If the two names are

not identical, the tape will not be mounted.

(4) This PAUSE card alerts the operator to look for and mount the device set whose VSN (volume serial no.) is PITHA.

(5) MOUNT is a logical operation of associating with this JOB the device set whose VSN is PITHA and whose name is BIBLIO.

(6) This REQUEST card states that this program will create a permanent file on the device set whose name is BIBLIO. This file will contain the entire bibliography in the format described as file PB on page 25.

(7) This command causes execution of the program BREAD2.

(8) File TAPE5 includes any error messages and also gives the last accession number of the old PB as well as the last accession number assigned to each box of new entries and the number of cards in each box.

(9) When execution has been completed successfully the new file will be CATALOG'ed on the device set.

It should be pointed out that BREAD2 will not add a box of cards to the new PB (TAPE10) if any bad cards are found (as indicated by improper sequence numbers 1-4 in column 80 of each card). In fact the program terminates whenever a bad box is found. Thus one should carefully check the TAPE5 messages to make sure that all boxes did indeed get added to the PBF.

Section 2 - Keyword Storage and Maintenance

Description:

Each entry in the bibliography may have certain key-words associated with it, these key-words being the materials for which data is given within the article and the properties of the materials which are discussed therein. The entry, verification of these key-words, as well as the creation of the keyword file, are described in this section.

In order to check the validity of these entries, it is necessary to maintain a file of valid property and material codes. These codes along with quantity and units codes are kept in a Data Standards File (DATST) which is described in Section 3 since it is more widely used in the creation of the data file.

Since these key-words are also entered when data is entered, one may skip this section entirely and use DREAD2, described in Section 3, to enter and maintain the keyword file.

Preparation and Verification of Keyword File, PK

The PK consists of fixed length (unformatted) records of 32 10 character words each. Each such record holds a complete keyword entry consisting of the data: article accession number, property code for a single specific material property for which data is provided in the article, list of materials for which data is available with this property and record number. Since a given article may provide data on several properties there may be several entries associated with a single accession number. Each such entry originates as a set of cards (1 to 5 cards) where each card has the form:

column	1-10	accession no.	
	11-20	properties code	
	21-30	material	up to 6 materials per card
	⋮		
	71-80		

The format of this card is shown in Figure III.5.

The following rules apply:

1) All items are left justified in their respective field.

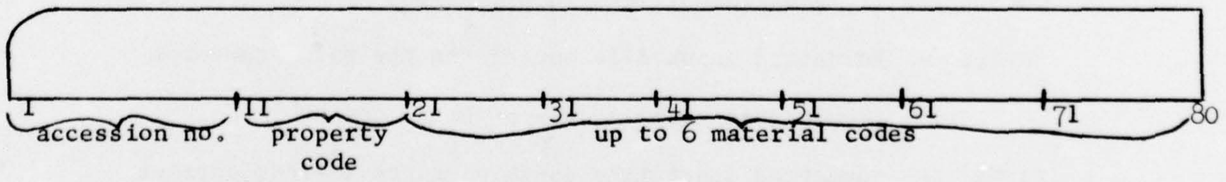
2) Up to 5 cards may be used to include all materials with a given property and accession number. Up to 29 materials may be associated with one accession number, property pair (accession number and property must be included on each card). The last word in each PK record is a record number. (This number is added by CREAD2 and is not included on the original cards.)

3) The current list of "properties" codes is shown on page 132. A blank properties code is acceptable. It is replaced by the code ZZZ in the PK

4) A blank"materials" field is a terminator for the materials list put in a PK record (i.e. any non-blank "materials" fields following the blank one are ignored). If the first materials field is blank (indication that the article does not include data about any specific material) then the code UNSPEC is inserted in the PK record.

5) Material codes (or identifiers) are limited to one CDC word (10 characters) and must be in the Data Standards File (DATST). For materials with longer identifiers, the first 9 characters concatenated with $ should be used. The remaining characters (up to 10) are stored in this look-up table (DATST) which is accessible to the bibliographic display program BIBDIS (see Section 4). This means, of course, that during the punching of the key-words cards that a list is kept of entries to be added to this standard table. The current list of acceptable materials' codes is shown on page 130 .

6) Every PK record has a record number in the last word of the record. This number is put in at the time of creation of the PK (or addition to the PK) and, as a rule, will be different from the accession number. It is very important to note that before the PK can be used, records with identical accession numbers should be grouped together.



All items are left justified in their respective field.

Figure III.5 - Keyword Cards

The PK is created in a 3 step procedure similar to that used in creating the PB. The first step is to process one or more "boxes" of keyword cards using program CREAD1 to check for "bad"cards and to create a file suitable for UPDATE. Like BREAD1, the output of CREAD1 consists of:

1) A list of bad cards (see below)
2) A list of all cards with possible bad cards flagged with asterisks.
3) A file suitable for use with UPDATE.

A "bad card" is one with an illegitimate properties code (i.e. the code is not in the properties code table) and/or illegitimate material code and/or a bad accession number. A bad accession number is a set of characters which can not be interpreted as a positive integer less than a set limit (at the time of writing this limit is 99995). Note that although the accession number is left justified on the keywords card for ease of punching, trailing blanks within the accession number field are not substituted for by zeroes as in the current CDC Fortran. In the PK the accession number is right justified.

File Descriptions:

CREAD1(TAPE11,TAPE4,TAPE5,TAPE3,TAPE7,OUTPUT)

TAPE11	-	Formatted input file containing the data standards (acceptable material and property codes)
TAPE4	-	Formatted input file containing the keyword entries
TAPE5	-	Formatted output file containing a list of all bad cards

TAPE3 - Formatted output file containing a list of all cards with bad cards flagged.

TAPE7 - Formatted output file containing all cards which may be used directly by UPDATE.

OUTPUT - System output file.

N.B. - Only file 11 is rewound at initialization in this program.

Sample deck set-up:

```
BARC1,CM60000,T20
REQUEST(TAPE7,*PF)
ATTACH(CRB,CREAD1BX3296,ID=PITHA)
ATTACH(TAPE11,DATST X3296,ID=PITHA)
CRB(,INPUT)
REWIND(TAPE3,TAPE5)
COPY(TAPE5)
COPY(TAPE3)
CATALOG(TAPE7,BOX1X3296,ID=PITHA)
```

<u>Processing of Keyword Entries</u>:

The second step in creating (or appending to) the PK is to correct all cards flagged by CREAD1 using UPDATE,EDITOR or just correcting the cards themselves. Next CREAD2 is used to add 1 to 4 "BOXES" to the PK. Most of the comments given in Section 1 pertaining to these steps apply here.

File Descriptions:

```
CREAD2(TAPE1,TAPE2,TAPE3,TAPE4,TAPE5,TAPE9,TAPE10,OUTPUT,TAPE11)
```

TAPE1 - TAPE4	- up to 4 formatted input files containg the corrected keyword cards.
TAPE5	- Formatted output file containing the total number of records in the new keyword file.
TAPE9	- Old unformatted keyword file.
TAPE10	- New unformatted keyword file.
TAPE11	- Formatted data inventory file.

N.B. - Files 1,2,3,4,9 and 11 are rewound initially by the program.

Sample Deck Set-up:

```
BARC2,CM60000,T20,TP2.
VSN(TAPE9=CC1045)
VSN(TAPE10=CCXXXX)
REQUEST(TAPE9)          (CC1045/NORING)PITHA
REQUEST(TAPE10)         (CCXXXX/RINGIN)PITHA
REWIND(TAPE10)
ATTACH(TAPE11,DAT ST X3296,ID=PITHA)
ATTACH(TAPE1,BOXiX3296,ID=PITHA)
CRB.
REWIND(TAPE10,TAPE9,TAPE5)
COPY(TAPE5)
```

The above job assumes the corrected box of keyword cards has been stored on permanent file as BOX1X3296. It also assumes the original keyword file is on magnetic tape CC1045 and the new file is to be written on a different magnetic tape. If you wish to put the new file back on the same tape, you need only leave out the VSN and REQUEST card for TAPE10, put the ring in on TAPE9 and COPY TAPE10 to TAPE9 at the end of the job.

The final step in processing the keyword entries is to create a sorted sequential file with only 1 accession no., property code and material code per record. This file is called NPM since it is sorted in accession number, property and material order. It is created by CREAD3 and stored on the device set to be used by the bibliographic display program.

File Descriptions:

CREAD3(OUTPUT)

PK	-	Primary keyword input file, probably on magnetic tape.
NPM	-	Sorted keyword output file to be stored on device set.
OUTPUT	-	System output file, also lists the number of records in the files.

Sample Deck Set-up:

```
BARC3,CM100000,T30,TP1.
VSN(PK=CC1045)
REQUEST(PK)      (CC1045/NORING)PITHA
PAUSE. THIS JOB USES VSN=PITHA
```

```
MOUNT (VSN=PITHA,SN=BIBLIO)
REQUEST(NPM,*PF,SN=BIBLIO)
ATTACH(CRB,CREAD3BX3296,ID=PITHA)
LIBRARY(COBOL)
RFL(100000)
CRB.
CATALOG(NPM,ID=PITHA)
```

The above job cannot be run from intercom except in the batch mode because it uses a private device set and too much core storage.

Example:

The following is an example of the JOBS which might be used in adding 3 boxes of keywords to the existing PK. Refer to page 95 for a detailed explanation of the various control statements and data sequences used. They apply here with obvious changes.

<u>Step 1</u>

Process 3 boxes with 3 JOBS using program CREAD1.

```
BARC1(CM60000,T20)
REQUEST(TAPE7,*PF)
ATTACH(CRB,CREAD1BX3296,ID=PITHA)
ATTACH(TAPE11,DATSTX3296,ID=PITHA)
CRB(,INPUT)
```

```
REWIND(TAPE5)
COPY(TAPE5)
REWIND(TAPE3)
COPY(TAPE3)
CATALOG(TAPE7,BOX1X3296,ID=PITHA)
7/8/9
(box of keyword cards - box no. 1)
6/7/8/9
```

The second and third boxes of cards are processed using identical JOB's except that control statement (7) is replaced by:

```
    CATALOG(TAPE7,BOX2X3296,ID=PITHA)
and CATALOG(TAPE7,BOX3X3296,ID=PITHA)
```

<u>Step 2</u>

Correct the BOXES (the 3 TAPE7's) created above using UPDATE.

```
BARUP(CM60000,T30,TP1)
ATTACH(BOX1,BOX1X3296,ID=PITHA)
UPDATE(N,I=BOX1,C=O,D)
UPDATE(P=NEWPL,C=TAPE1,D,8)
RETURN(BOX1,NEWPL)
ATTACH(BOX2,BOX2X3296,ID=PITHA)
UPDATE(N,I=BOX2,C=O,D)
UPDATE(P=NEWPL,C=TAPE2,D,8)
RETURN(BOX2,NEWPL)
```

```
ATTACH(BOX3,BOX3X3296,ID=PITHA)
UPDATE(N,I=BOX3,C=0,D)
UPDATE(P=NEWPL,C=TAPE3,D,8)
[See Step 3 for the rest of the control record]
7/8/9
*DELETE KYW.512
  (replacement card for 512)
7/8/9
*INSERT KYW.101
  (card to be inserted after card 101)
6/7/8/9
  (In this example there are no corrections to box 3)
```

Step 3

Add the three BOXES to PK. The JOB control record is completed as follows:

```
ATTACH(CRB,CREAD2BX3296,ID=PITHA)
VSN(TAPE9=CC1045)
REQUEST(TAPE9)        (CC1045/RINGIN)PITHA
REWIND(TAPE9)
ATTACH(TAPE11,DATST X3296,ID=PITHA)
CRB.
REWIND(TAPE10,TAPE9,TAPE5)
COPY(TAPE10,TAPE9)
COPY(TAPE5)
```

Step 4

Convert the PK file to a sorted sequential file (NPM) and store it on the private device set as follows:

```
BARC3,CM100000,T30,TP1.
VSN(PK=CC1045)
REQUEST(PK)          (CC1045/NORING)PITHA
PAUSE.  THIS JOB USES VSN=PITHA
MOUNT(VSN=PITHA,SN=BIBLIO)
REQUEST(NPM,*PF,SN=BIBLIO)
ATTACH(CRB,CREAD3BX3296,ID=PITHA)
LIBRARY(COBOL)
RFL(100000)
CRB.
CATALOG(NPM,ID=PITHA)
```

Section 3 - Creation and Maintenance of the Data File

Description

In conjunction with the creation of a bibliographic file of books and articles, a system of storing quantitative measurements or data has been established. This section deals with the creation and updating of this data file.

In order to check the validity of these data entries, it is necessary to maintain a file of valid property, material, quantity and units codes. These codes are stored in a Data Standards File (DATST) which is described later in this section (page 125).

Preparation and Verification of Data Cards

In forming the data base, articles previously entered into the bibliographic system are reviewed by the problem originator and information of importance is red-penciled. This information may include:

Properties for which data is provided in the article, for example, stress or thermal expansion.

Materials for which data is provided, such as lithium fluoride.

Physical property (also called quantity herein) identifying the particular data measurements such as temperature, refractivity, etc.

Data items giving the numerical measurement of the above physical property. (Occasionally the data may be qualitative)

Units in which the data is given, such as, dynes/cm^2.

Multiplier by which all the data items should be multiplied.

Environmental condition under which the measurement applies.

The property, material, quantity and units will always be entered in coded form where the acceptable codes and their translation will be maintained in the Data Standards File (DATST). This is necessary so that standards will be maintained for new entries which can be validated by the checking program, DREAD1. The creation of this file and its current contents are described at the end of this section.

When an article has been so marked, it should already have on it an accession number which was assigned by the bibliographic entry program. This number must be included in the data card since it is used to retrieve the information stored.

The data within an article may be presented in a fairly complicated display since it may "naturally" be printed in conjunction with other data in tabular form as a function of one or more variables (also quantities). Consequently, it is necessary to give each quantity or data entry from a specific article a unique reference number by which other data entries may identify it. One would normally number each entry from a particular article consecutively starting with 1.

Each data entry may have a list of associated reference numbers to other entries within the same article which should be printed along with it.

The data itself may come in many forms for which we have provided a data type code. This code is a positive or negative integer between 1 and 7, as follows:

Data Type Code	Data Form
1	= number
2	= number ± error
3	range is number to number
4	the absolute value = number
5	up to 150 characters of alpha-numeric information
6	the absolute value = number ± error
7	degrees, minutes and seconds

If any of the above codes is prefixed by a minus sign, the measurement is taken to be approximate. A '-5' is invalid.

Data given in tabular form may have any of the above data type codes except 5 and would also need a count of the number of values within the table. The independent variable must be marked with an * and the dependent variable needs a list of the independent variables' reference numbers.

The initial form of data storage is the punched card. These cards are punched in field free format, i.e., elements need not be in a particular column, but do need to be in a particular order. Elements on the data card are separated by semi-colons. The data cards may continue for as many cards as are needed for the entry. However no one element may continue for more than 2 cards. The entire 80 columns on the card need not be completed. An ampersand (&) is needed after the last field of a data entry to signal the end of that entry. Any field which can be omitted, may be done so by putting semi-colons right next to each other.

An example of a data entry may be found on page 115 .

The elements on the data entry card must be in the following order:

1) The accession number of the article from which the data was taken. This should be an integer between 1 and 99999, inclusive. It may not contain a decimal point.

2) The <u>reference number</u> by which this particular data entry may be called. The first data entered from a specific article will probably be reference number 1 and the rest would be numbered successively. Any integer between 1 and 99999, inclusive, and without a decimal point is acceptable.

3) The <u>material code</u> as described on page 126. This is a code of up to 10 characters which has been previously entered in the Data Standards File. If it is not in the file, an error will be noted by DREAD1.

4) The <u>property code</u> also described on page 127. This is a code of up to 5 characters which must also be in the Data Standards File.

5) The <u>quantity code</u> of up to 10 characters, also in the Data Standards File.

6) The <u>data type code</u> as described on page 111. It must be a positive or negative integer between 1 and 6, inclusive.

7) <u>String of alpha-numeric information</u> or the <u>number of associated references</u>. If the data type code in element 6 is a 5, this element contains up to 150 characters of information to be printed exactly as it appears here. Since this would be the

last element if the data type code is 5, it must end with an ampersand. For any other data type code, this field contains the number of associated references (nr) that should be printed at the same time as this data entry. It must be an integer between 0 and 50, inclusive. The next nr elements must contain the <u>reference numbers</u> (between 1 and 9999) associated with this entry.

8+nr) If the data entered was not in tabular form, this element should contain a <u>∅</u>. If the data entered is data for an independent variable, this element should contain an asterisk (*). If the data entered is a function of other variables, this element should contain the <u>number of independent variables</u> (ni). The next ni elements should contain the <u>list of reference numbers of the independent variables</u>. These reference numbers must have been included in the list of associated reference numbers in element 7, or else an error message will occur.

9+nr+ni) <u>Environmental condition</u> under which this data applies. This is a string of up to 40 characters including blanks defining any measurements incidental to the primary one.

10+nr+ni) <u>Display code</u> to be used later to define methods of displaying the data. For now leave this element blank.

11+nr+ni) <u>Units code</u> of the data. This is a code of up to 10 characters which must also be in the Data Standards File.

12+nr+ni) Multiplier by which the data number should be multiplied. This field must be able to be decoded by an E10.0 format statement, that is to say that this element may have up to 10 characters which may include a sign, a decimal point and an E to indicate an exponent. For example x 10^3 would be entered as 1.E3.

13+nr+ni) The number of data items (nd) to be entered. Data type codes 2, 3 and 6 would have 2 data items and codes 1, 4 and 7 would have 1. However, if the data is in tabular form, this element would be the number of items in the table. The next nd elements would contain the data items. Followed by an ampersand to end this data entry. All data items except type 7 should always include a decimal point and be less than or equal to 10. Again an exponent would be indicated by an E, for example 4.68 x 10^{-13} would be entered as 4.68E-13. Data type 7 (degrees, minutes, seconds) is to be keyed with a blank between degrees and minutes, an apostrophe after minutes and a quote after seconds, for example, 45 15'10".

A Sample Data Case

The following example illustrates the above rules. From an article assigned bibliographic accession number 3093 the table given in Figure III.6 is to be entered.

The quantities C and C' required further explanations. The title of this article is "The Dispersion of the Stress Optical Coefficient of Vitreous Silica". The material is vitreous silica which has a material code of SIO2 and the property category is STRESS-OPTICAL CONST. which has a property code STRSS. Enter LAMBDA as the independent variable and C and C' as dependent variables. A further explanation of the quantities C and C' may be defined when entering them in the data inventory file. These explanations will be printed whenever this data entry is printed. Also enter the temperature at which these measurements were made as an environmental condition. Figure III.7 shows the work sheet and the resultant form of punched cards. All the values of the tabular data were not written on the work sheet, since it would be easier to punch these directly from the source. This will then result in the print out given in Figure III.8.

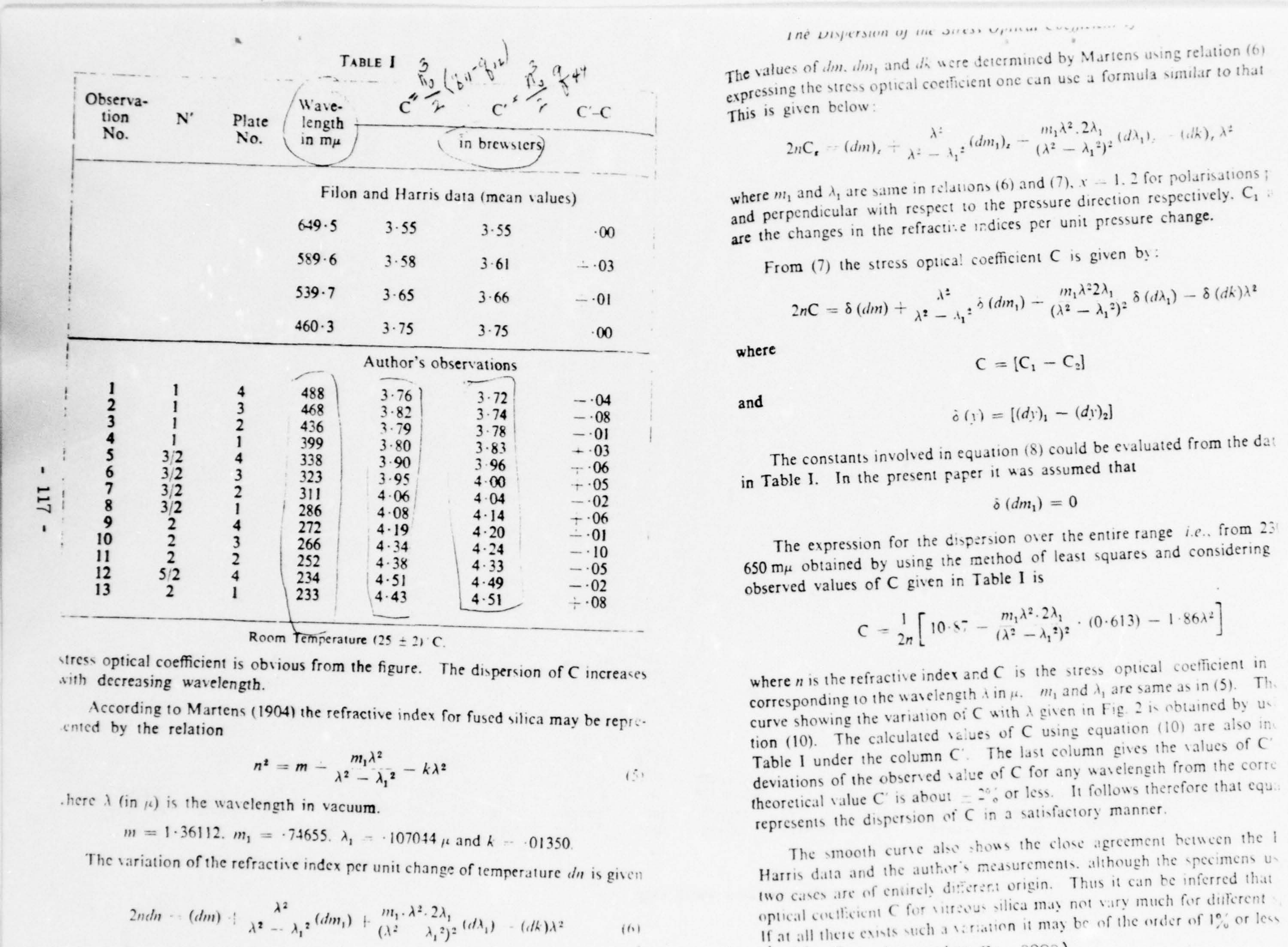


TABLE I

$C'' = \frac{n_0^3}{2}(q_{11}-q_{12})$ $C' = \frac{n_0^3}{2} q_{44}$

Observation No.	N'	Plate No.	Wave-length in mμ	C	C'	C'–C
				in brewsters		
Filon and Harris data (mean values)						
			649·5	3·55	3·55	·00
			589·6	3·58	3·61	+·03
			539·7	3·65	3·66	—·01
			460·3	3·75	3·75	·00
Author's observations						
1	1	4	488	3·76	3·72	—·04
2	1	3	468	3·82	3·74	—·08
3	1	2	436	3·79	3·78	—·01
4	1	1	399	3·80	3·83	+·03
5	3/2	4	338	3·90	3·96	+·06
6	3/2	3	323	3·95	4·00	+·05
7	3/2	2	311	4·06	4·04	—·02
8	3/2	1	286	4·08	4·14	+·06
9	2	4	272	4·19	4·20	+·01
10	2	3	266	4·34	4·24	—·10
11	2	2	252	4·38	4·33	—·05
12	5/2	4	234	4·51	4·49	—·02
13	2	1	233	4·43	4·51	+·08

Room Temperature (25 ± 2) °C.

stress optical coefficient is obvious from the figure. The dispersion of C increases with decreasing wavelength.

According to Martens (1904) the refractive index for fused silica may be represented by the relation

$$n^2 = m - \frac{m_1\lambda^2}{\lambda^2 - \lambda_1^2} - k\lambda^2 \qquad (5)$$

where λ (in μ) is the wavelength in vacuum.

$m = 1{\cdot}36112$, $m_1 = {\cdot}74655$, $\lambda_1 = {\cdot}107044\,\mu$ and $k = {\cdot}01350$.

The variation of the refractive index per unit change of temperature dn is given

$$2n\,dn = (dm) + \frac{\lambda^2}{\lambda^2 - \lambda_1^2}(dm_1) + \frac{m_1\cdot\lambda^2\cdot 2\lambda_1}{(\lambda^2 - \lambda_1^2)^2}(d\lambda_1) - (dk)\lambda^2 \qquad (6)$$

The values of dm, dm_1 and dk were determined by Martens using relation (6) expressing the stress optical coefficient one can use a formula similar to that This is given below:

$$2nC_x = (dm)_x + \frac{\lambda^2}{\lambda^2 - \lambda_1^2}(dm_1)_x - \frac{m_1\lambda^2\cdot 2\lambda_1}{(\lambda^2 - \lambda_1^2)^2}(d\lambda_1)_x - (dk)_x\lambda^2$$

where m_1 and λ_1 are same in relations (6) and (7), $x = 1, 2$ for polarisations and perpendicular with respect to the pressure direction respectively. C_1 are the changes in the refractive indices per unit pressure change.

From (7) the stress optical coefficient C is given by:

$$2nC = \delta(dm) + \frac{\lambda^2}{\lambda^2 - \lambda_1^2}\delta(dm_1) - \frac{m_1\lambda^2 2\lambda_1}{(\lambda^2 - \lambda_1^2)^2}\delta(d\lambda_1) - \delta(dk)\lambda^2$$

where

$$C = [C_1 - C_2]$$

and

$$\delta(y) = [(dy)_1 - (dy)_2]$$

The constants involved in equation (8) could be evaluated from the dat in Table I. In the present paper it was assumed that

$$\delta(dm_1) = 0$$

The expression for the dispersion over the entire range *i.e.*, from 23 650 mμ obtained by using the method of least squares and considering observed values of C given in Table I is

$$C = \frac{1}{2n}\left[10{\cdot}87 - \frac{m_1\lambda^2\cdot 2\lambda_1}{(\lambda^2 - \lambda_1^2)^2}\cdot(0{\cdot}613) - 1{\cdot}86\lambda^2\right]$$

where n is the refractive index and C is the stress optical coefficient in corresponding to the wavelength λ in μ. m_1 and λ_1 are same as in (5). Th curve showing the variation of C with λ given in Fig. 2 is obtained by us tion (10). The calculated values of C using equation (10) are also in Table I under the column C'. The last column gives the values of C' deviations of the observed value of C for any wavelength from the corre theoretical value C' is about ± 2% or less. It follows therefore that equ represents the dispersion of C in a satisfactory manner.

The smooth curve also shows the close agreement between the I Harris data and the author's measurements, although the specimens u two cases are of entirely different origin. Thus it can be inferred that optical coefficient C for vitreous silica may not vary much for different If at all there exists such a variation it may be of the order of 1% or less

Figure III.6 - Data Input Sample (table from Accession No. 3093)

Work Sheet for Entering Quantitative Data

Acc. No.;	Index;	Mat.;	Prop.;	Quant.;	Data Type;	assoc. refs.;	ind. var.;	Envir. cond.;	Display Code;	Units Code;	Multi-plier;	No. of Data Wds.;	Data
3¢95	1	SIO2	STRSS	LAMBDA	1	2;2;3	*	ROOM TEMP (25+-2)C		MILLIMICRN		,3	488. $\longrightarrow$ 233.8
"	2	"	"	C	1	1;1	1;1			BREWSTERS		,3	3.76 $\longrightarrow$ 4.438
"	3	"	"	CPRIME	1	1;1	1;1			"		,3	3.72 $\longrightarrow$ 4.518

Card 1:

```
3093;1;SIO2;STRSS;LAMBDA;1;2;2;3;*;ROOM TEMPERATURE(25+-2)C;
```

Card 2:

```
;MILLIMICRN;;13;488.;468.;436.;399.;338.;323.;311.;286.;272.;
```

Card 3:

```
266.;252.;234.;233. &
```

Card 4:

```
3093;2;SIO2;STRSS;C;1;2;1;3;1;1;BREWSTERS;;13;33.76;
```

Card 5:

```
3.82;3.79;3.80;3.90;3.95;4.06;4.08;4.19;4.34;4.38;4.51;4.43 &
```

Card 6:

```
3093;3;SIO2;STRSS;CPRIME;1;2;1;2;1;1;BREWSTERS;;
```

Card 7:

```
13;3.72;3.74;3.78;3.83;3.96;4.00;4.04;4.14;4.20;4.24;4.33;
```

Card 8:

```
4.49;4.51 &
```

Figure III.7 - Data Cards Sample

```
3093

    JOG,ES

    J. IND. INST. SCI.                                39   93  100  1957

    THE DISPERSION OF THE STRESS OPTICAL COEFFICIENT OF VITREOUS SILICA

                    STRESS-OPTIC CONST.
                        SI02
-------------------------------------------------------------------------------------------
                                                                ROOM TEMPERATURE(25+-2)C
                                          CPRIME          = (N**3)/2*Q44
                                          C               = (N**3)/2*(Q11-Q12)

LAMBDA              CPRIME         C
 MILLIMICRN          BREWSTERS      BREWSTERS

    488.            3.72            3.76
    468.            3.74            3.82
    436.            3.78            3.79
    399.            3.83            3.80
    338.            3.96            3.90
    323.            4.00            3.95
    311.            4.04            4.06
    286.            4.14            4.08
    272.            4.20            4.19
    266.            4.24            4.34
    252.            4.33            4.38
    234.            4.49            4.51
    233.            4.51            4.43
```

Figure III.8 - Output From BIBDIS For Accession Number 3093 With Data

Program DREAD1 is used to check the validity of data cards. When a substantial number of cards has been punched, this program is used to check that the cards have met all the requirements stated on pages 112-115. Two files are created to aid in the correction of errors detected on the data cards.

File Descriptions:

DREAD1(TAPE1,TAPE4,TAPE5,TAPE3,OUTPUT)

TAPE1	- formatted input file containing the data standards.
TAPE4	- formatted input file containing the data entry cards.
TAPE5	- formatted output file containing specific information on the errors within any rejected cards.
TAPE3	- formatted output file containing a complete list of the data entry cards with asterisks flagging those cards with errors.
OUTPUT	- system output file.

N.B. - No files are rewound within the program.

Sample Deck Set-up:

```
BARD1,CM60000,T30.
ATTACH(DRB,DREAD1BX3296,ID=PITHA)
ATTACH(TAPE1,DATSTX3296,ID=PITHA)
DRB(,INPUT)
REWIND(TAPE5,TAPE3)
```

```
    COPY(TAPE5)
    COPY(TAPE3)
    7/8/9
      (Data entry cards)
    6/7/8/9
```

The above job assumes that the binary version of program DREAD1 has been stored as permanent file DREAD1BX3296 and the Data Standards File has been stored as permanent file DATSTX3296. The data entry cards are assumed in the input stream.

This job may also be run interactively on INTERCOM by storing the data entry cards on permanent file. The above set of commands would be used except that the job card would be replaced by CONNECT(OUTPUT) and another attach card would be needed, i.e.,

```
    ATTACH(TAPE4,DATBX3296,ID=PITHA)
```

Replace DRB(,INPUT) with

```
      DRB.
```

Processing of Data Entry Cards

After all errors in the data entry cards have been corrected, program DREAD2, is used to add the new cards to the sorted data file (ISD). This program rechecks these cards and will stop if any error is found. If no errors are found, the program updates or creates the ISD. If the data file does not yet exist, it must be created by punching a "C" in column 1 of the input card. If this file already exists, a "U" for update must be punched in column 1. This program will also put the property and material on the data entry cards into the sorted keyword file (NPM) if they are not already there. Consequently, it is not necessary to run CREAD1, 2 or 3 (Section 2) if you are going to also enter data from any articles.

File Descriptions:

DREAD2(TAPE1,TAPE4,TAPE5,INPUT,OUTPUT)

TAPE1	-	formatted input file containing the data standards.
TAPE4	-	formatted input file containing the corrected data entry cards.
TAPE5	-	formatted output file containing information on errors within rejected cards.
INPUT	-	system input file containing "C" for create or "U" for update in column 1.
OUTPUT	-	system output file containing the number of records in the files.
ODIR	-	Unformatted input file containing the original sorted keyword or directory file, usually called NPM.
DIR	-	unformatted output file containing the new sorted keyword file.
ISD	-	unformatted indexed-sequential data file.

Since the new data file (ISD) and directory file (DIR) should be written on the private device set, this program should be run in the batch mode. It also cannot be run interactively on INTERCOM because it requires too much core storage.

Sample Deck Set-up:

```
BARD2,CM120000,T50
PAUSE. THIS JOB USES DEVICE SET VSN=PITHA
MOUNT (VSN=PITHA,SN=BIBLIO)
ATTACH(DR2B,ID=PITHA,SN=BIBLIO)
ATTACH(TAPE1,DATST ,ID=PITHA,SN=BIBLIO)
ATTACH(TAPE4,DATBX1X3296,ID=PITHA)
ATTACH(ODIR,DIR,ID=PITHA,SN=BIBLIO)
REQUEST(DIR,*PF,SN=BIBLIO)
ATTACH(ISD,ISD,ID=PITHA,SN=BIBLIO)
LIBRARY(COBOL)
RFL(120000)
DR2B.
CATALOG(DIR,ID=PITHA)
REWIND(TAPE5)
COPY(TAPE5)
PURGE(ODIR)
7/8/9
U
6/7/8/9
```

The above job assumes that the data entry cards have been stored on permanent file DATBX1X3296. If they are on cards instead, remove the ATTACH(TAPE4) card and replace the DR2B. with DR2B(,INPUT). The data entry cards would then preceed the 6/7/8/9 card.

If the data file does not yet exist but is to be created, replace the ATTACH(ISD) card with

```
     REQUEST(ISD,*PF,SN=BIBLIO)
```

Put in another CATALOG card after the DR2B., i.e.,

```
     CATALOG(ISD,ID=PITHA,FO=IS)
```

The last parameter is used to indicate to the system that ISD is an indexed sequential file.

You need also replace the 'U' card with a'C' card.

Creation of the Data Standards File

The Data Standards File must be established to collect in one place a record of valid material, property, quantity and units codes. This file is used by programs CREAD1,CREAD2,DREAD1 and DREAD2 to check the validity of the data entries. It is also used by program BIBDIS to fully identify a word when the code is not sufficiently clear. Most importantly, it is necessary to have a standard code for each name so that it is consistently entered in the same manner.

The data standard is entered and updated on punched cards and then copied onto the device set for use by BIBDIS. The format of these cards is shown in Figure III.9. The following rules have been devised for the data standard cards:

Card 1: The number of valid material codes,m,in I4 format, i.e., the number would be punched so that its last digit is in column 4.

Next m cards: Material codes. These codes should be kept in alphabetical order for ease in checking the contents of the file. The first 10 columns contain the material code which will be, as nearly as possible, the chemical formula for the substance. For instance, $(CH_2)_6N_4$ will have a code of (CH2)6N4 and SiO_2 will be SIO2. Some materials have chemical formulas which exceed the ten alpha-numeric characters available. In this case we truncate the formula to 9 characters and insert a dollar sign for the tenth to indicate the truncation. The formula would then be continued in columns 11 through 30. For example, thallium alum would be punched as:

THAL(SO4)$2*12H2O

In other cases materials cannot be described in this manner and material codes such as GLASS-9692 will require a further explanation which would be punched in columns 30 through 80 of the card. When punching the material code on a keyword or data entry card, only the first 10 characters of the code should be used. However for printing purposes, BIBDIS will print out the entire code. The explanation field will only be used when listing the entire data inventory file.

Card m+2: The number of valid property codes, p, in an I4 format. See Card 1 for an explanation.

Next p cards: Property codes. These cards should also be kept in alphabetical order. The property code is a 3 to 5 character abbreviation for some general category into which measurements can be placed. Examples are STRSS and ACOUS which stand for stress-optical properties and accoustical properties respectively. These property codes should be punched in columns 1-5, starting in column 1. Corresponding to each of these codes, a 20 character translation should be punched starting in column 11 which will be used on all print outs. Columns 31-80 may be used for any further explanation of the property.

Card m+p+3: The number of valid quantity codes, q, again in I4 format.

Next q cards: Quantity or Physical Property codes, again kept in alphabetical order. The quantity code identifies a particular data measurement such as temperature, refractivity or shear velocity. Codes for these quantities have 10 characters, starting in column 1 and try, as nearly as possible, to be the appropriate scientific symbol or symbol name, such as LAMBDA for wavelength or ALPHA for α or Q11-Q12 for $q_{11}-q_{12}$. These codes sometimes become less clear and a translation may be punched in columns 11-30. Any further explanation of the quantity may be punched om columns 31-80.

Card m+p+q+4: The number of valid units codes,r,in I4 format.

Next r cards: Units codes, in alphabetical order. Again the first 10 columns contain the code, the next 20 a translation if needed and the final 50 any further explanation.

These m+p+q+r+4 cards constitute the Data Standards File. The present contents of the file are listed at the end of this section. If additions are made to this file, simply insert the new card in the proper section and change the number of entries to reflect the additions.

In keeping these cards in alphabetical order one should maintain the convention used by BIBDIS which is dependent on the display code for each character on the CDC6600. The collating sequence of the valid characters is:

ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789+-*/( )$blank,

This implies that (CH2)6N4 comes after ZNSE and that SI comes after SIO2.

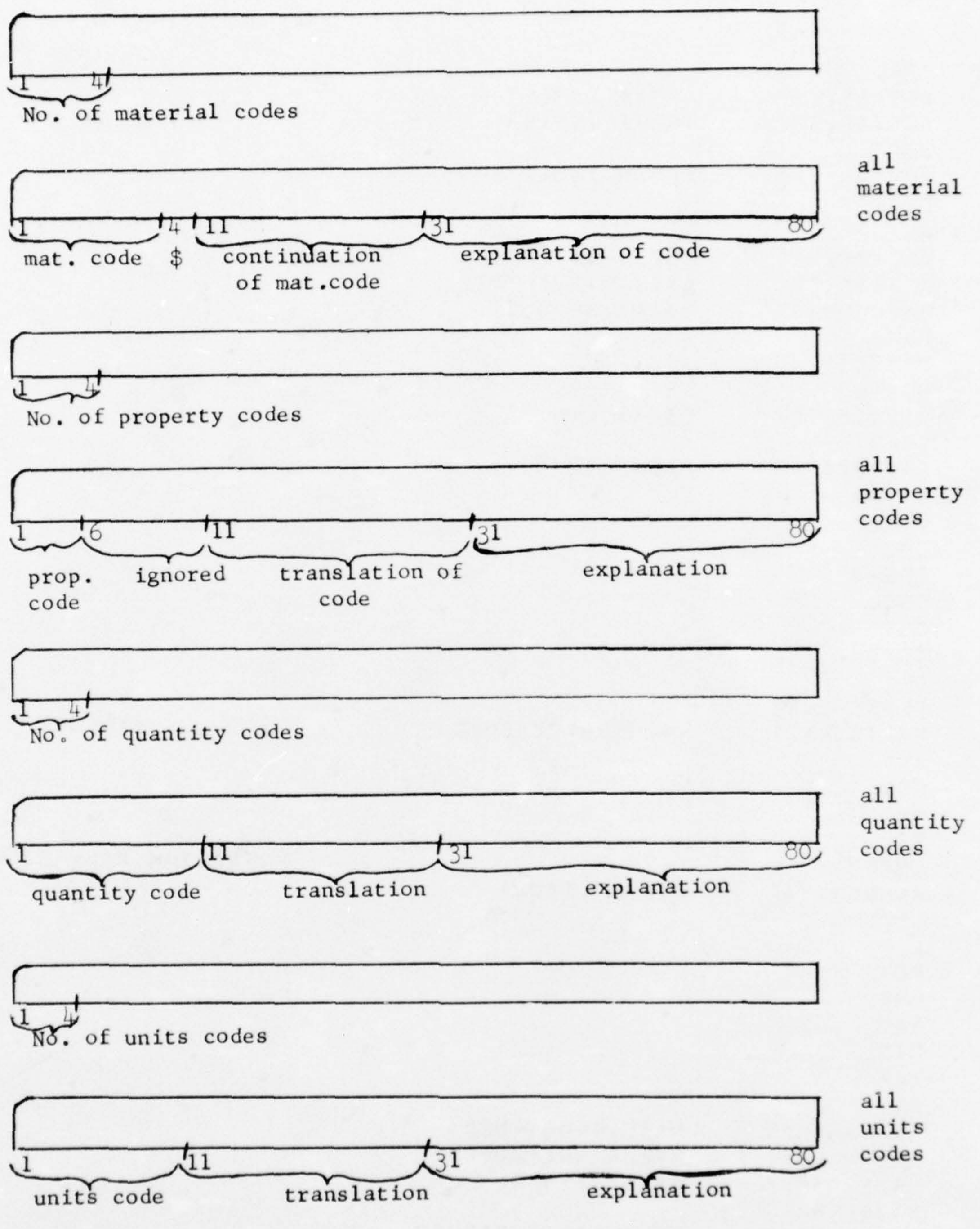


Figure III.9 - Data Standard Cards

DATA STANDARDS FILE

```
MATERIAL CODES
     AG3(AS,SB$        AG3(AS,SB)S3
     AG3(AS,SB$        AG3(AS,SB)S3
     ALF3
     AL(NH4)SO$        AL(NH4)S04*12H20
     AS2SE3
     AS2S3
     AS2(SE,TE$        AS2(SE,TE)3
     A(II)B(IV$        A(II)B(IV)C(V)2
     BA2NANBB5$        BA2NANBB5012
     BAF2
     BA(NO3)2
     BERYL
     BI2(SE,TE$        BI2(SE,TE)3
     CAF2
     CALASCAP:$        CALASOAP:NO
     CAW04
     CDF2
     CDS
     CDS-A
     CSCL
     CSI
     GAAS
     GE
     GLASS-9692
     KAL(SO4)2$        KAL(SO4)2*12H20                        K-ALUM
     KBR
     KCL
     KF
     KI
     KMGF3                                     POTASSIUM MAGNESIUM FLUORIDE
     K(NBO3,TA$        K(NBO3,TAO3)
     LICL
     LIF
     MGF2
     MNF2
     NABR
     NACL
     NAF
     NAI
     NAKC4H406$        NAKC4H406*4H20
     NA2C4H406$        NA2C4H406*2H20
     NA2S203*5$        NA2S203*5H20
     NA3ALF6
     NH4AL(S04$        NH4AL(S04)2*12H20
     NH4CL
     NH4H3(SE0$        NH4H3(SE03)2
     NIC
```

```
PB(NO3)2
PB5GE3O11
RBBR
RBCL
RBH3(SEO3$      RBH3(SEO3)2
RBI
SB2(SE,TE$      SB2(SE,TE)3
SI
SIO2
SRF2
SRTIO3
TE
TEO2
TLAL(SO4)$      TLAL(SO4)2*12H2O                              THALLIUM ALUM
TL3ASS4
TL5TE3
TL(NH4)(S$      TL(NH4)(SO4)2
ZNO
ZNSE
ZNS-A
(AG,SB)(S$      (AG,SB)(S,SE)
(AS,SE)2S$      (AS,SB)2SE3
(AS,SB)2S$      (AS,SB)2S3
(BATIO3,M$      (BATIO3,MNNBO3)
(BATIO3,T$      (BATIO3,TA2O5)
(BA,SR,NA$      (BA,SR,NA)NBO3
(BA,SR)NB$      (BA,SR)NB2O6
(BA,SR)T1$      (BA,SR)T1O3
(BA,SR)TI$      (BA,SR)TIO3
(BI)2(T1O$      (BI)2(T1O3)3
(CA,SR,BA$      (CA,SR,BA)F2
(CH2)6N4
(GAAS,AS2$      (GAAS,AS2SE3)
(GA,AS)2T$      (GA,AS)2TE3
(IN,GA)(A$      (IN,GA)(AS,P)
(IN,SB)2T$      (IN,SB)2TE3
(LA2,LA3)$      (LA2,LA3)TE4
(NH4)AL(S$      (NH4)AL(SO4)2*12H2O                           NH4-ALUM
(SBSI,SB2$      (SBSI,SB2S3)
(SN,SB)(S$      (SN,SB)(S,SE)
(SR,BA)NB$      (SR,BA)NB2O6
(SR,BA)(N$      (SR,BA)(NBO3)2
(TL2SE,AS$      (TL2SE,AS2TE3)
(ZN,CO)3A$      (ZN,CO)3AS2
```

PROPERTY CODES

ABS	ABSORPTION SPECTRA
ACOUS	ACOUSTIC PROPERTIES
COL	COLOR CENTER
CRY	CRYSTAL GROWTH
DAM	DAMAGE(RAD.THERM.)
DEB	DEBYE TEMPERATURE
DEN	DENSITY
DIB	DIELECTRIC BREAKDOWN
DIE	DIELECTRIC CONSTANT
DIEDP	DIELECT.C.(DE/DP)
DIEDT	DIELECT.C.(DE/DT)
DIEDV	DIELECT.C.(DE/DV)
ELA	ELASTIC PROPERTIES
ELADT	ELASTIC C.(DC/DT)
ELE	ELECTRICAL PROPERTIE S
ELL	ELLIPSOMETRY
ELC	ELECTRO-OPTIC
ENE	ENERGY GAP
FEE	FERROELECTRICS
FER	FERRO-ELECTRIC
GRU	GRUNEISEN
HAR	HARDNESS
HEA	HEAT CAPACITY
LAT	LATTICE DYNAMICS
MAG	MAGNETIC
MEL	MELTING POINT
MISC	MISCELLANEOUS
NON	NON-LINEAR OPTIC
OPT	OPTICAL PROPERTIES
PSC	PIESO-OPTICS
PZE	PIEZO-ELECTRIC
REF	REFRACTIVE INDEX
REP	REPORT
REV	REVIEW
SPGR	SPECIFIC GRAVITY
STRSS	STRESS-OPTIC CONST.
STRUD	STRUCTURAL DEFECTS
STRUT	STRUCTURE
SURF	SURFACES
THCFR	THERMAL COEFF.REF.IN
THCO	THERMAL CONDUCTIVITY
THERM	THERMODYNAMIC
THEX	THERMAL EXPANSION
THINF	THIN FILM
VAP	VAPOR PRESSURE
ZZZ	UNKNOWN

```
QUANTITY CODES
    ABSORP
    ALPHA
    ALPHA-19A
    ALPHA-19B
    ALPHA-23A
    ALPHA-23B
    BETA
    CELL DIM A
    CELL DIM B
    CELL DIM C
    CLEAVAGE
    CLINF
    CL
    CPERP
    CPRIME             (N**3)/2*Q44
    CP
    CRYSTAL
    C11
    C12
    C44
    C                  (N**3)/2*(Q11-Q12)
    DEL L/L
    DIFF ASPCT
    DN/DP-EPS
    DN/DP-OHM
    HT
    KNCOP
    K3(100)
    K3(111)
    LAMBDA MU
    LAMBDA
    LAMREV1112
    LAMREV44
    LNGFQ(PER)
    LNGFRQ(AB)
    LNGFRQ(AC)
    LOADING
    LONGFREQ
    LONG VEL
    MELPTAFLCR                               MELTING POINT AFTER HYDROFLOU
    MELT POINT
    MOHS
    NA
    NB
    NC
    NEPSILON
    NOMEGA
    N-EPS
    N-OHM
    N                  REFRACTIVE INDEX
    ORIENT.ANG                               ORIENTATION ANGLE
    POINT GRP
```

```
POLAR-LONG
POLAR-TRAN
P11
P11POL
P11-P12
P12
P12POL
P13
P16
P31
P32
P33
P44
P44POL
P66
Q11
Q11-Q12
Q11-Q13
Q12
Q44
RDN/DR-EPS
RDN/DR-OHM
RESIST
RHO
R41
SHEARVEL
SOFTPT
SPACE GRP
STD DEV
TCLAMDAMUD
TCLAMDAMUE
TDEPLONG
TEMPCOPIE
TEMP
TEMP.COEF.                    TEMPERATURE COEFFICIENT
THETA0
THEXCC
TRANS VEL
TRANSFAR
TRANSVFREQ
TRNFQ(PAR)
TRNFQ(PER)
TRNFRQ(AB)
TRNFRQ(AC)
TSLAMDAMUD
TSLAMDAMUE
V1          VEL.OF  LONG.MODE
V2          VEL.OF  TRANS.MODE
V3          VEL.OF  TRANS.MODE
V4          VEL.OF  LONG.MODE
V5          VEL.OF  TRANS.MODE
V6          VEL.OF  TRANS.MODE
V
```

UNITS CODES

A	ANGSTROM
BREWSTERS	
CAL/MOL K	CALORIES PER MOL KELVIN
CGS	
CM2/DYN	SQUARE CENTIMETERS PER DYNE
CM/SEC	CENTIMETERS PER SECOND
DEG.C	DEGREES CENTIGRADE
DEG.K	DEGREES KELVIN
DEG.MIN.	DEGREES MINUTES
DYN/CM2	DYNES PER CENTIMETER SQUARED
G/CM3	GRAMS PER CENTIMETER CUBED
MILLIMICRN	MILLIMICRON
M/SEC	METERS PER SECOND
/DEG.C	PER DEGREE CENTIGRADE

Deck Set-up for putting the data inventory on the device set:

```
BARDI,CM60000,T10.
PAUSE.  THIS JOB USES PACK VSN=PITHA
MOUNT (VSN=PITHA,SN=BIBLIO)
PURGE (OLDDAT,DATST ,ID=PITHA,SN=BIBLIO)

                    use this card  only if there is already a Data
           Standards File on the device set which you wish to discard.
REQUEST(DATST ,*PF,SN=BIBLIO)
COPY(INPUT,DATST)
CATALOG(DATST ,ID=PITHA)
7/8/9
Data inventory cards
6/7/8/9
```

Section 4 - Bibliographic Display Operating System (BIBDIS)

Description

The primary task of BIBDIS is to retrieve and display bibliographic information and associated physical data using a simple set of directives. Since many of the files containing this information are quite large, it is assumed that they exist on a private device set. Therefore the user must run in the batch mode rather than interact with the computer.

The system has been designed so that the user may operate BIBDIS using a simple set of directives, page 139, and a minimum of job control language, page 145.

Files used by BIBDIS include:

1) System Files:

INPUT	- Used only for feeding directives into BIBDIS. See page 140 for a list of these directives.
OUTPUT	- Used by BIBDIS for outputting informative messages. It is also used by the SCOPE Operating System to inform the user of trouble.
PRINT	- Used for the major output requested by the directives.

2) Primary files assumed on the private device set:

PB	- The Primary Bibliographic file contains information for each bibliographic entry such as accession no., author, journal, year, pages and title. This information has been entered through program BREAD2 as described in Section 1, (page 92).
DIR	- The Keyword Directory file contains the accession no., property code and material code and a pointer to the data file if data has been entered for this accession no. Since this file is in ascending order by accession no., property and material codes, it is also referred to as NPM. This information might have been entered with program CREAD2 (page 103) or DREAD2 (page 122).
ISD	- The Sorted Directory file contains all the data that has been entered through program DREAD2, (page 122). It is sorted in accession no. order and in entry no. order.

DATST	- The Data Standards File contains all the valid property, material, quantity and units codes, (page 125).
JABREV	- The Journal Abbreviation file contains all the acceptable journal abbreviations in "numerical" order, (page 87).
JWABRE	- The Journal Word Abbreviation file contains all the acceptable word abbreviations, (page 80).

3) Secondary files which may be saved on the private device set as space allows or will be created as they are needed:

PBR	- The bibliographic file in random access form is used for all LISTB operations.

PBAT
PBJT
PBAYT
PBAJT
PBJAT
PBJYT

These Tag files pointing to the bibliography are sorted in order of author and accession no., journal and accession no., author and year, author and journal, journal and author, journal and year respectively. They are created and used by LISTB operations in author or journal order.

NMP
PNM
PMN
MNP
MPN
QNP
PQN
MQN

These files contain the same information as DIR but this information has been sorted in various orders by accession no. (N), property (P), material (M) and/or quantity (Q). These files are created and used by LISTB operations in property, material or quantity order.

Directives for the BIBDIS Operating System

Directives are put on cards (or card images) in file INPUT using the following rules:

a) All items in a directive are field free (separated by one comma) but are position dependent.

b) The first item is the directive code or corresponding directive number. It corresponds to the task to be accomplished. The directive code is a mnemonic which describes the task. It consists of up to 10 alphanumeric characters (no blanks). Alternatively each directive code has an acceptable abbreviation (which must of course be unique) which may be used instead; or a directive number may be used.

c) All other items are parameters which modify the directive code. Each parameter consists of up to 10 alphanumeric characters (all characters on the 29 punch are acceptable except comma). Some directives may require no parameters and most directives have default parameters. A default parameter may be indicated by the question character (?) or by blanks.

The system currently has 4 directives available to the user. They may be referred to either by their directive number, directive code or abbreviation. These 4 directives are:

Directive No.	Directive Code	Abbreviation	Description
1	MAKER	MA	Make a random access file
2	LISTB	LB	List bibliography
3	SIZE	SZ	Get size of a file
4	LISTF	LF	List an auxillary file

Directive 1: MAKER - This directive will create a random access (word-addressable) file from a sequential file. You may want to use it to store a new version of file "PBR" on the private device set. It will not create a new file unless no file of that name is attached. Consequently, of an old version of the random access file exists on the device set, it should be purged first.

Parameters needed:

1. File Name of existing sequential file.

   Default: PB

2. File Name of random access file.

   Default: Name in Parameter 1 with an "R" appended.

EXAMPLE:

MAKER

This directives creates the random access file named PBR from sequential file PB.

Directive 2: LISTB - This directive will display data base contents including author, journal, year, properties, materials and data in many forms depending on the parameters' values. A summary of the parameters for LISTB is shown on page 144. The parameters explained in detail are:

1. Order in which to list the output. Acceptable values for this parameter are N,A,J,P,M,Q,PM,PQ,MQ,AY,AJ,JA,JY. These values list the output respectively in order of accession no., author, journal, property, material, quantity, properties of particular materials, properties with particular quantities, materials with particular quantities, authors in chronological order, authors in journal order, journals in author order, journals in chronological order.

   Default: N

2. Starting number or name. Output is listed <u>from</u> this value of parameter 1. For instance if parameter 1 is an N, then parameter 2 would have the accession no. at which to start listing. If parameter 1 is P, this parameter would be the <u>code</u> of the property from which to start. A slash (/) for this parameter would direct the computer to list all entries with <u>no</u> author, journal, property, material or quantity, depending on the letter in Parameter 1.

   Default: First possible value.

3. Ending number or name. Output is listed <u>to</u> this value of parameter 1. In order to list a particular accession no. or author, etc., just set both parameters 2 and 3 to the same value. Remember that only 10 characters may be

included in any parameter, consequently only the first 10 characters or less of the author name or journal name may be entered. If these 10 characters are not unique one may get a little more output than desired. If a slash has been entered for parameter 2, this parameter will be ignored. However, since parameters are position dependent, a space or comma is needed for it.

Default: Last possible value. Consequently, if both parameters 2 and 3 are blank, the entire bibliography will be printed.

4. Type of output. Parameters 4 and 5 tell what part of the data base asked for by parameters 1 through 3 are to be printed. F lists full bibliographic entry including accession no., author, journal, year, pages and title; N, accession no. only; AY, author and year only; AJY, author journal and year only.

   Default: F

5. Auxillary output. N lists no further output; P lists properties; M lists materials; PM lists both properties and materials; and D lists data.

6. Starting number or name for the 2nd letter of parameter 1. If parameter 1 only has 1 letter this and the next field are ignored.

   Default: First possible value.

7. Ending number or name for the 2nd letter of parameter 1.

   Default: Highest possible values.

N.B. - Whenever you enter a property, material or quantity as a parameter, you must enter the code as listed in the Data Standards File for this word. Also if you enter the "from" and "to" values of any word, remember that the "to" must be after the "from" in the alphabetical order maintained by the system which is:

```
    ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789+-*/( )$blank,
```

Consequently if you wish to list all bibliographic entries with quantities of C and CPRIME you may use the following directive:

```
    LISTB,Q,CPRIME,C,F,N
```

Since blank comes after the letters in alphabetical order, C must come after CPRIME.

Summary of Parameters for LISTB

No.	Purpose	Acceptable Values	Explanation	Default
1	Order	N,A,J,P,M,Q,PM, MP,PQ,MQ,AY,AJ, JA,JY	Output is listed in acc. no., author, journal, prop., material, quantity, prop. and material order, etc.	N
2	From	Any value of 1 of the above orders or "/"	Output is listed from this value ("/" lists entries with no author, journal, prop. or mat.)	First possible value
3	To	Any value of 1 of the above orders	Output is listed to this value	Last possible value
4	Type of output	F,N,AJY,AY	Full listing, Acc. no. only, author, journal and year, author and year.	F
5	Auxillary output	M,P,MP,PM,N,D	Also lists properties, materials, or both to which bib. entry refers or all data from this bib. entry.	N(none)
6 7	From To		Used only if 2 letters in the value of Parameter 1, to input the "from" and "to" values of the 2nd letter.	beginning end

Examples:

1) LISTB,N,90,110,F,P
lists all acc. no.'s from 90 to 110, with full listing plus all properties associated with each entry.
2) LISTB,A,BALLARD,BALLARD,AJ,?
lists all bib. entries whose author is BALLARD, **listing only** author and journal
3) LISTB,P,DEN,DEN,N,?
lists only the acc. no.'s of all bib. entries dealing with the property "density".

Directive 3: SIZE - This directive gives the size in records and words of the requested file. The only parameter needed is the name of the file. Any of the files listed on pages 137 and 138 except INPUT, OUTPUT and PRINT are acceptable for this parameter.

EXAMPLE:

```
SIZE,DIR
```

Directive 4: LISTF - This directive lists the contents of the auxillary files used by the system, such as, DATST, JABREV, JWABRE. This file name is the only parameter required.

EXAMPLE:

```
LISTF,DATST
```

JOB CONTROL DECK

When running BIBDIS the following control deck is required:

```
JOBID,CM150000,T70.        PROB. NO. NAME
                                            (1)
PAUSE. THIS JOB USES PACK VSN=PITHA         (2)
MOUNT(VSN=PITHA,SN=BIBLIO)                  (3)
SETNAME(BIBLIO)                             (4)
ATTACH(PB,ID=PITHA)                         (5)
ATTACH(NPM,ID=PITHA)                        (6)
ATTACH(ISD,ID=PITHA)                        (7)
ATTACH(JWABRE,ID=PITHA)                     (8)
ATTACH(JABREV,ID=PITHA)                     (9)
ATTACH(DATST,ID=PITHA)                      (10)
ATTACH(BDBIN,ID=PITHA)              (11)
LIBRARY(COBOL)                      (12)
RFL(150000)                         (13)
LOAD(BDBIN)                         (14)
EXECUTE(BIBDIS)                     (15)
REWIND(PRINT)                       (16)
COPY(PRINT)                         (17)
EXIT.                               (18)
REWIND(PRINT)                       (19)
COPY(PRINT)                         (20)
7/8/9                               (21)
All directives                      (22)
6/7/8/9                             (23)
```

Explanation of some of these control cards:

(2) This PAUSE card alerts the operator to look for and mount the device set whose VSN (volume serial no.) is PITHA.

(3) MOUNT is the logical operation of associating with this JOB the device set whose VSN is PITHA and whose name is BIBLIO.

(4) This SETNAME card informs the system that all the files to be attached are on the device set named BIBLIO.

(5-10) These ATTACH cards allow the files on the device set to be used by the system. If any of the secondary files as described on page 138 have been stored on the device set, they too should be attached at this point.

(11) This ATTACH card allows the system to use the binary version of the BIBDIS program.

(12) The Cobol Library is required in order to use the sort-merge package.

(13) This RFL card prevents automatic reduction of core and is needed for execution of internal sorts.

(14-15) Load and execute the BIBDIS program.

(16-17) Copies the PRINT file which contains the requested listings to the OUTPUT file.

(18-20) These cards also copy the PRINT file if a system error occurs which halts execution.

(22) As many Directive cards (page 139) as required may be put here.

Those secondary files most frequently used by the system such as PBR, may be stored on the device set as space permits. To do this the following cards would be needed:

```
REQUEST(FILENM,*PF,SN=BIBLIO)
                    (between cards 4 and 5)
CATALOG(FILENM,ID=PITHA)
                    (between cards 18 and 19)
```

where FILENM is the name of any of the secondary files named on page 138. One of the directives must be a command which would use this file so that it will be created during the run and permanently stored. Once this file has been stored on the device set, it would then need to be included in the list of files to be ATTACHed by the program. For example, if PBR has been put on the device set, the following card would be needed after card 9.

```
ATTACH(PBR,ID=PITHA)
```

The following control deck would be used to put a new PB file on the private device set. The new file is assumed on tape XXXX and the file presently on the device set will be saved on tape YYYY as a back-up.

```
JOB ID,CM600000,T30,TP1.                PROB. NO.        NAME
VSN(TAPE3=CCYYYY)
VSN(TAPE1=CCXXXX)
PAUSE. THIS JOB USES DEVICE SET(VSN=PITHA)
MOUNT(VSN=PITHA,SN=BIBLIO)
ATTACH(TAPE2,PB,ID=PITHA,SN=BIBLIO)
REQUEST(TAPE3,RING)        (CCYYYY/RING)PITHA
LABEL(TAPE3,W,L=BIBBKUP)
COPY(TAPE2,TAPE3)
UNLOAD(TAPE3)
PURGE(TAPE2)
RETURN(TAPE2)
REQUEST(TAPE2,*PF,SN=BIBLIO)
REQUEST(TAPE1)        (CCXXXX/NORING)PITHA
LABEL(TAPE1,R,L=BIBLIOGRAPHY)
COPY(TAPE1,TAPE2)
CATALOG(TAPE2,PB,ID=PITHA)
```

If you wish to put the control deck on a permanent file so that you can remote batch a job, from intercom, the following steps would be necessary:

1) Copy the control deck to permanent file, with the following job.

```
JOB ID,CM40000,T10.                  PROB. NO.        PITHA
REQUEST(TBB,*PF)
COPY(INPUT,TBB)
CATALOG(TBB,BATBDX3296,ID=PITHA,RP=999)
7/8/9
Control Deck
                (up to the 7/8/9 card)
6/7/8/9
```

2) When this is done, it is necessary to bring the control deck into EDITOR and add on the directives whenever you wish to run. Do this on intercom with the following commands.

```
COMMAND-ATTACH(BATBD,BATBDX3296,ID=PITHA)
COMMAND-EDITOR
..      F 50 58  CH=80
..      EDIT,BATBD,S
```

This S puts line nos. on the control deck for editor's use.

```
..      L,L
```

This command will instruct editor to print out the last line in the control deck with its line number.

Intercom will respond with something like

300=COPY(PRINT)

now add an end of record mark and directives; for instance.

.. 310=*EOR

320=LISTB,N,1,10,F,N

330=LISTF,DATST

Inorder to see a listing of the entire control deck, use

L,A

Save this control deck without line numbers in order to BATCH it.

.. SAV,BAT,N

For any further explanation of the EDITOR, see INTERCOM REFERENCE MANUAL, Chapter 4, Ref. 9.

3) BATCH the job to the INPUT queue. If OUTPUT is to go directly to the printer use the command:

.. BATCH,BAT,INPUT

If the OUTPUT file is to return to INTERCOM, use the command:

.. BATCH,BAT,INPUT,HERE

4) If HERE has been used and the job has finished running, i.e., the COMMAND-FILES indicates that a REMOTE OUTPUT FILE exists, look at this output file with the PAGE COMMAND. First bring the remote output file for LOCAL use, i.e.,

$$\text{COMMAND-BATCH,}\underbrace{*******}_{\text{JOB ID with 2 letters appended by system}}\text{,LOCAL}$$

COMMAND-PAGE,*******

PAGE responds

```
READY..
```

And enter a line no., say

```
2000
```

If this is beyond the number of lines in the output file, PAGE will respond with

```
END OF INFORMATION REACHED
LINE ****
```

To see the day file, subtract about 30 lines from the end of information line no. and enter this number.

```
****
```

Now PAGE will print the next 10 lines of your output file. In order to continue printing after these 10 lines key "+" for the next 10 lines.

```
+
```

Enter another line number or a "+" sign whenever PAGE is through printing. Abort its printing by pressing space % A

In order to leave PAGE key "E".

```
E
```

For a further explanation of PAGE COMMAND see INTERCOM REFERENCE MANUAL, pages 3-26 thru 3-31 (Ref. 9).

5) In order to send this output file to the PRINTER use the command

$$\text{BATCH,}\underbrace{\text{*******}}_{\text{output file name}}\text{,PRINT,}\underbrace{\text{XXXX}}_{\text{identifier}}$$

APPENDIX A DATA ENTRY AND MAINTENANCE NOTES

As noted, the prime method of data entry is to punch the data on cards. These cards are then checked (as much as possible) by a checking program tailored to the data being entered and a printed list is produced which summarizes and flags possible errors. There may still remain many undetected errors which only a human observer can isolate.

There are three basic methods for correcting data "cards" available to the CDC6600 user:

1) Remove incorrect cards, repunch new ones and replace the old ones
2) Correct card images (i.e. copies of the cards on permanent file) from an INTERCOM terminal using EDITOR.
3) Create an UPDATE file from the original cards (or card images) and make corrections using UPDATE directives and a "correction deck".

Each method has advantages and disadvantages. The one which is used may depend on the number of original cards and/or the number of cards to be corrected.

Method 1

Advantages:

- a correct backup data file (the cards) is automatically created.
- no computer time is required to make the corrections.

Disadvantages:

- the card deck or decks must be fed back into the system. Large decks are particularly subject to card reader problems.
- since the original cards are unnumbered it may be time consuming to find the right cards to be corrected.

Method 2

Advantages:

- it is easy to make a large number of similar changes.
- it is the fastest way to make corrections in general.
- it is easy to locate cards to be corrected (BREAD1 for example produces a list of cards with numbers which they would have in EDITOR.)

Disadvantages:

- EDITOR may "bog down" with extremely large files.
- no back up file is automatically created unless the corrected card images are "batched" to the punch.

Method 3

Advantages:

- corrections to the card images and the original cards can be made more or less simultaneously.
- this is the only method of changing card images in the "batch mode". It does not require the use of a remote terminal although UPDATE can be used under INTERCOM.

Disadvantages:

- most time consuming since both directives and correction cards must be punched.

Since data entry is by punched cards it makes sense to maintain back up data files in the form of punched card records stored in correct sequence in boxes or card drawers. Thus, once card images have been corrected, the corresponding cards themselves should be corrected or replaced in toto by the corrected card images using the high speed punch to punch out the new cards. For massive corrections method 2 followed by complete replacement of the original cards is the preferred method. For few corrections to many cards, method 3 is preferred. Either method 1 or 3 should be used for few corrections to moderate card decks.

The following paragraphs outline the procedures for using method 2 and method 3.

method 2 (INTERCOM and EDITOR - see ref. 9)

1) store cards on a permanent file of card images

```
REQUEST(TBB,*pF)
COPY(INPUT,TBB)
CATALOG(TBB,BOX1X3693818,ID=BARRETT)
7/8/9
  [cards]
6/7/8/9
```

or use a file generated by a program such as BREAD1 as the input to EDITOR.

2) log in to a terminal and use the following commands

```
EDITOR
F CH = 80
ATTACH (B,BOX1X3693818,ID=BARRETT)
EDIT,B,S
```

(cards are assigned sequence numbers starting at 100 and incrementing by 10.)

- make changes by:

a) overwriting the card to be changed by typing n = a where n is the card number (assigned by editor)and a is the character string desired.

b) /a/ = /b/, n_1, n_2 where a is the string to be replaced by b between and including card numbers n_1 and n_2. If only one card use only n_1; if all cards set n_1=A for "all".

- make deletions by:

  D,n_1,n_2 where cards between $n_1$and n_2 inclusive are to be deleted.

- make additions (insertions) by typing n = b where n is the card number such that $n_1 < n < n_2$ where n_1 and n_2 are the preceeding and succeeding card numbers reqpectively. Similarly additions can be made wherein $n > n_{last}$.

3) save and catalog the corrected cards for future use

```
REQUEST(TBB,*PF)
SAV,TBB,O
CATALOG(TBB,BOX1X3693181,ID=BARRETT)
```

(note - this will create a new "cycle" of the old permanent file. To get rid of the old permanent file use DISCARD,B)

4) if a completely new deck of cards is to be obtained use

```
BATCH,TBB,PUNCH, identifier
```

where identifier is a character string of 4 alphanumeric characters. The deck of cards will be identified by I identifier XX where XX is replaced by 2 characters assigned by the system.

method 3 (UPDATE - see ref. 12)

The following annotated example illustrates the use of UPDATE to correct 2 "boxes" of key word card images such as might be produced by CREAD1 from the original cards. Of course UPDATE can be used with any set of card images regardless of their source. It is important to note, however, that the first card of the deck of cards to be updated (i.e. used to create the "update program library") must consist of *DECK name, where name is assigned by the user. All cards in this deck are internally numbered in sequence (for UPDATE purposes) starting with 1 for the *DECK card.

```
BARUP(CM60000,T30)                          (1)
ATTACH(BOX5,BOX5X3296,ID=PITHA)             (2)
UPDATE(N,I=BOX5,C=0,D)                      (3)
REQUEST(TAPE1,*PF)
UPDATE(P=NEWPL,C=TAPE1,D,8)                 (4)
CATALOG(TAPE1,BOX5X3296,ID=PITHA)
RETURN(NEWPL)                               (5)
PURGE(BOX5)
ATTACH(BOX6,BOX6X3296,ID=PITHA)             (6)
UPDATE(N,I=BOX6,C=0,D)                      (7)
REQUEST(TAPE2,*PF)
UPDATE(P=NEWPL,C=TAPE2,D,8)                 (8)
CATALOG(TAPE2,BOX6X3296,ID=PITHA)
PURGE(BOX6)
7/8/9
*DELETE KEY.214                             (9)
     (replacement card for 214)
7/8/9                                       (10)
*DELETE KEY.1096
     (replacement card for 1096)
*BEFORE KEY.2
     (insertion card for missing first keyword card)
6/7/8/9
```

(1) Job card should also include problem number and name.

(2) BOX5 is the UPDATE compatible version of the original set of cards

identified as "box5". The first logical record in this file consists of the UPDATE directive: *DECK KEY ("DECK" is an UPDATE code. "KEY" is any user code word.)

(3) This command causes the UPDATE utility program to create a "New Program Library", NEWPL, from the input BOX5. This NEWPL can then be modified on subsequent calls to UPDATE. The significance of the parameters are as follows:

N	A NEWPL is to be created
I = BOX5	The input is on file BOX5
C = 0	Do not create a compile file (0 = "zero")
D =	The input data is contained in 80 columns

(4) This command causes UPDATE to correct NEWPL according to "directives" in the first record following the control record (i.e. the record after the first 7/8/9. The useful output of UPDATE is TAPE1 which contains, so to speak, BOX5 corrected according to the correction cards following the various UPDATE directives. The significance of the parameters is as follows:

P = NEWPL	The program library is NEWPL created in the preceding step.
C = TAPE1	The Compile file is TAPE1. The default logical file names for the BOX inputs to CREAD2 are TAPE1, TAPE2, TAPE3, TAPE4.
8	80 "column" output on the compile file. (if 8 is omitted then the compile file consists of 90 "column" output with sequence numbers placed within the last 10 columns.)

(5) File NEWPL should be "returned" i.e. it is no longer attached to the job since a new NEWPL is to be created. BOX5 is purged since it is no longer needed.

(6) See (2) above except here BOX6 is "box6".

(7) See comments (3) - (4).

(8) See comments (3) - (4).

(9) A set of "directives" in the "input" file to UPDATE may be used to inform this program what task to perform. For correcting "cards" in a file such as a BOX of key word entries one must be able to either: delete, insert, or replace (or correct). These tasks are performed according to the following directives:

*DELETE c	Card c is deleted.
*DELETE c_a, c_b	Cards c_a to c_b are deleted.

If the DELETE directive is followed by some text cards then these cards will replace those that have been deleted. Thus DELETE also functions as a "replace" directive.

*INSERT c (text cards)	Text cards are inserted after card c.
*BEFORE c (text cards)	Text cards are inserted before card c.

Each card in the original BOX is given an identifier c in the form deck-name - sequence-number. (Deck-name is the user code word referred to above.)

For key word entries the deck-name is KEY and is assigned by CREAD1 in the creation of the UPDATE compatible file TAPE7 mentioned on page 107. The sequence-number is the same as the sequence number assigned to eachcard on file TAPE6 by CREAD1. (This number indicates the actual card position in the original box with the first card assigned the number 2.)

(10) This end-of-record card terminates the set of directives for the first correction UPDATE. The set of directives following the EOR are for the second correction UPDATE.

Note on use of UPDATE under INTERCOM

This same sequence of control statements should be used. The only difference is that the set of directives should be created using EDITOR and then saved in a separate "input" file. Then when the correction UPDATE ((4) or (7) above) is called, the "input" file must be indicated by including the parameter I = "input". Thus the following sequence of INTERCOM "commands" might be used to create "input" and run UPDATE.

```
EDITOR
F CH=80
100=*DELETE BIB.214
110= (replacement card)
SAV,C1,N
BYE
UPDATE(P=NEWPL,C=TAPE1,D,8,I=C1)
```

REFERENCES

1. Sahagian, C. S. and Pitha, C. A.
AFCRL - 72-0170 Special Reports, No. 135 1972
COMPENDIUM on High Power Infrared Laser Window
Materials (LQ-10 Program)

2. Barrett, T. B.
Parke Mathematical Laboratories TM11 1973
(Contract F19028-71-C-0142) Task Outline for Data
Storage and Retrieval

3. Barrett, T. B.
Parke Mathematical Laboratories TM19 1975
(Contract F19628-71-C-0031) The BIBDIS Operating System

4. Control Data Corporation 60344200 (no date)
Loader Version 1 Reference Manual

5. Control Data Corporation 60359600 (no date)
Cyber Record Manager Version 1 User's Guide

6. Control Data Corporation 60307300 (no date)
Cyber Record Manager Version 1 Reference Manual

REFERENCES (continued)

7. Rossmassler, S. A. (editor)
NBS Technical Note 747 1972
Critical Evaluation of Data in the Physical Sciences
A Status Report on the National Standard Reference Data System

8. The American Chemical Soc., Wash., DC 1974
Bibliographic Guide for Editors and Authors

9. Control Data Corp., Sunnyvale, Ca. 60307100 (no date)
Intercom Reference Manual Models 72,73,74 Version 4
6000 Version 4

10. Crowley, E. T., Thomas, R. C.
Acronyms Initialisms Dictionary 1975

11. Defense Documentation Center AD/A -006 800
Government Acronyms and Alphabetic Organizational
Designations used in DDC
(DDC - TGD
Cameron Station
Alexandria, VA 22314)

12. Control Data Corp., Sunnyvale, Ca. 60342500 (no year)
Update Reference Manual, 6000 Series Computer Systems

REFERENCES (Continued)

13. Control Data Corp., Sunnyvale, Ca. 60305600
Fortran Extended Version 4 Reference Manual

MISSION
of
Rome Air Development Center

RADC plans and conducts research, exploratory and advanced development programs in command, control, and communications (C^3) activities, and in the C^3 areas of information sciences and intelligence. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.

AMERICAN REVOLUTION BICENTENNIAL 1776-1976

Printed by
United States Air Force
Hanscom AFB, Mass. 01731

FILM

4