DOCUMENT RESUME

ED 124 183 IR 003 576

AUTHOR Kuczmarski, Tom
TITLE WISE: Building an On-Line Database; Reference Manual for the 1110.
INSTITUTION Wisconsin Univ., Madison. Academic Computing Center.
PUB DATE Apr 76
NOTE 28p.; Information Handling Series

EDRS PRICE MF-$0.83 HC-$2.06 Plus Postage.
DESCRIPTORS Bibliographic Coupling; *Computer Programs; Computers; Computer Storage Devices; *Data Bases; Data Processing; Information Processing; *Information Retrieval; Information Science; Information Sources; Information Storage; Input/Output Devices; *Manuals; On Line Systems; Search Strategies
IDENTIFIERS Computerized Searches; Interactive Computer Systems; *WISE

ABSTRACT

WISE is an on-line interactive information retrieval system designed particularly for bibliographic data bases, but it is potentially useful for any keyword-oriented data base. The WISE user searches a data base by entering and combining keywords that are appropriate to the data base and the subject area of interest. The user can also enter certain command statements to refine the results of the search and to control the generation of output. The WISE system has five absolute programs and two sets of subroutines for creating, updating, and searching a data base. This document deals with the subroutines and programs used to initially create and then update an on-line data base. The reader is assumed to be familiar with the WISE SEARCH program, FORTRAN V, FORTRAN subroutines, and simple EXEC-8 runstreams. (CH)

ED124183

WISE

building an on-line database

Reference Manual for the 1110

April, 1976

IR003576

MACC
academic computing center
the university of wisconsin - madison
1210 west dayton street
madison, wisconsin 53706

AGKNOWLEDGEMENTS

The WISE system was designed and implemented at MACC (Madison Academic Computing Center), University of Wisconsin-Madison by Tom Kuczmarski.

Special thanks go to the following people for their suggestions and contributions during the development of the system: Roy Tally, Tom Olson, Don McIsaac, Dennis Spuck.

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

WISE is an online interactive information retrieval system designed to operate on keyword-oriented databases. The system was designed particularly with bibliographic databases in mind, but is potentially useful for any keyword-oriented database. The WISE user searches a database by entering and combining keywords that are appropriate to the database and the subject area of interest. The user can also enter certain command statements to refine the results of the search, and to control the generation of output. In addition, certain features of WISE make it very convenient to perform SDI (Selective Dissemination of Information) applications.

The WISE system as a whole consists of five absolute programs and two sets of subroutines for creating, updating, and searching a database. The MACC document entitled *WISE Search Program* describes how to search an already existing database. This document deals with the subroutines and programs used to initially create and then update an online database.

The reader is assumed to be familiar with the WISE SEARCH Program, the FORTRAN V programming language implemented for Univac 1100 series computers, FORTRAN subroutines, and simple EXEC-8 runstreams.

Chapter 2

CHARACTERISTICS OF THE ORIGINAL INFORMATION SOURCE

WISE databases can be created from a large variety of information sources. This document assumes that the original information source has the following properties.

1) It exists on some computer-readable medium such as tapes, drum file, cards, etc.

2) It is logically composed of a set of information records each of which contains:

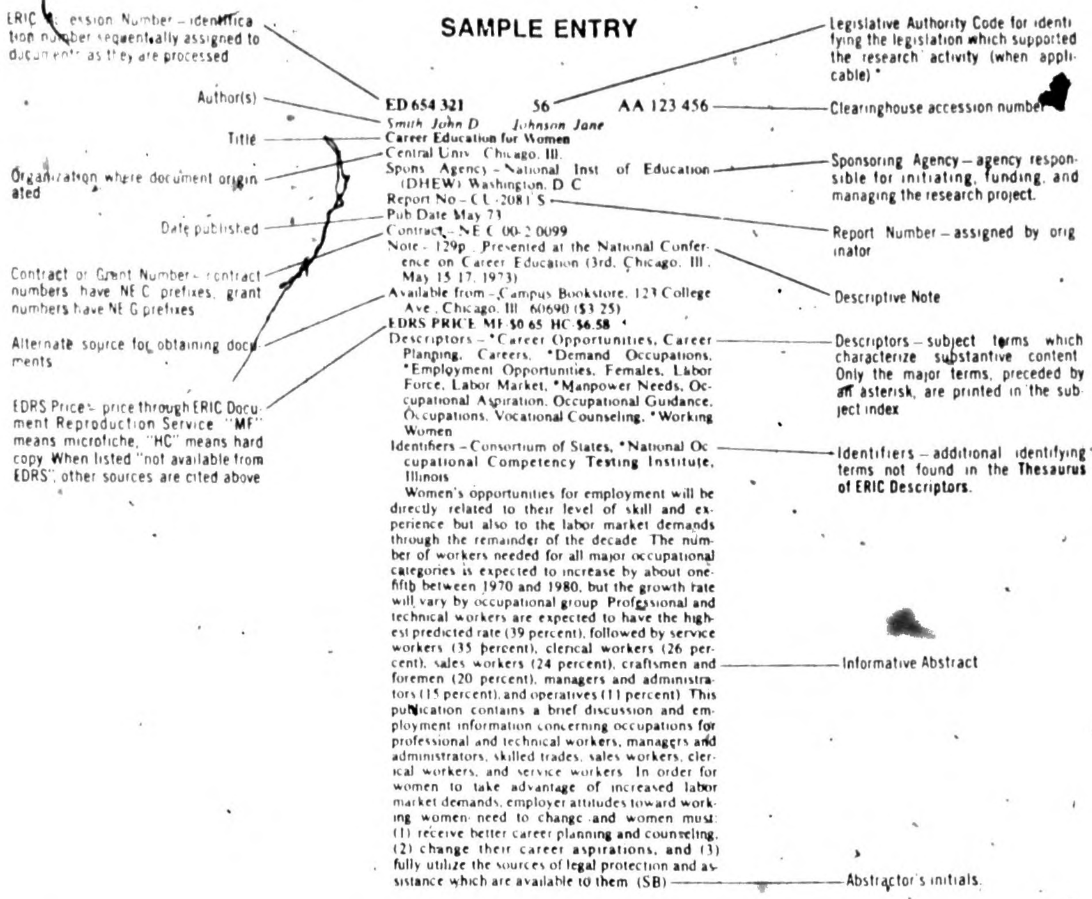
   a) a unique identification number, henceforth called a citation number;

   b) a set of fields which are to be used as retrieval keywords;

   c) other fields, some or all of which may be displayed online.

3) It will be subject to periodic updates (addition of new records, not the modification or deletion of existing records).

The following is an example of an information record from the ERIC, RIE information source (Educational Resources Information Center, Resources in Education). It is typical of the type of record which WISE was designed to process.



SAMPLE ENTRY

ED 654 321 56 AA 123 456

Smith John D Johnson Jane

Career Education for Women

Central Univ Chicago, Ill.

Spons Agency – National Inst of Education (DHEW) Washington, D C

Report No – CL-2081 S

Pub Date May 73

Contract – NE C 00-2 0099

Note – 129p. Presented at the National Conference on Career Education (3rd, Chicago, Ill., May 15-17, 1973)

Available from – Campus Bookstore, 123 College Ave., Chicago, Ill 60690 ($3 25)

EDRS PRICE MF-$0 65 HC-$6.58

Descriptors – *Career Opportunities, Career Planning, Careers, *Demand Occupations, *Employment Opportunities, Females, Labor Force, Labor Market, *Manpower Needs, Occupational Aspiration, Occupational Guidance, Occupations, Vocational Counseling, *Working Women

Identifiers – Consortium of States, *National Occupational Competency Testing Institute, Illinois

Women's opportunities for employment will be directly related to their level of skill and experience but also to the labor market demands through the remainder of the decade. The number of workers needed for all major occupational categories is expected to increase by about one-fifth between 1970 and 1980, but the growth rate will vary by occupational group. Professional and technical workers are expected to have the highest predicted rate (39 percent), followed by service workers (35 percent), clerical workers (26 percent), sales workers (24 percent), craftsmen and foremen (20 percent), managers and administrators (15 percent), and operatives (11 percent). This publication contains a brief discussion and employment information concerning occupations for professional and technical workers, managers and administrators, skilled trades, sales workers, clerical workers, and service workers. In order for women to take advantage of increased labor market demands, employer attitudes toward working women need to change and women must: (1) receive better career planning and counseling, (2) change their career aspirations, and (3) fully utilize the sources of legal protection and assistance which are available to them. (SB)

ERIC Accession Number – identification number sequentially assigned to documents as they are processed

Author(s)

Title

Organization where document originated

Date published

Contract or Grant Number – contract numbers have NE C prefixes, grant numbers have NE G prefixes

Alternate source for obtaining documents

EDRS Price – price through ERIC Document Reproduction Service. "MF" means microfiche, "HC" means hard copy. When listed "not available from EDRS", other sources are cited above

Legislative Authority Code for identifying the legislation which supported the research activity (when applicable) *

Clearinghouse accession number

Sponsoring Agency – agency responsible for initiating, funding, and managing the research project.

Report Number – assigned by originator

Descriptive Note

Descriptors – subject terms which characterize substantive content. Only the major terms, preceded by an asterisk, are printed in the subject index

Identifiers – additional identifying terms not found in the **Thesaurus of ERIC Descriptors**.

Informative Abstract

Abstractor's initials.

Before creating a database, the user should answer the following questions:

1) Which fields are to be used as retrieval keywords? In the example above, natural choices for retrieval keywords would be descriptors, identifiers, and perhaps author names.

2) If several different fields are to be retrieval keywords, is there any spelling overlap among them? For example, suppose that author last names and descriptors were chosen to be retrieval keywords. Would it be likely that the spelling of many (any) author last names would be the same as the spelling of descriptors? If so, author last names

could be defined to be used as auxiliary keywords (see Section 3.1.2). This would resolve the ambiguity.

3) Which fields should be grouped as main keywords? For example, if there were no spelling overlap between author last names and descriptors, both of these fields could be grouped together as main keywords. (Main keywords are described in the WISE SEARCH program document.)

4) Which (if any) of the fields should be treated as auxiliary keywords. When doing a search with the SEARCH program, auxiliary keywords are specified by using a special function name. Special functions are described in the WISE SEARCH program document.

5) Precisely what textual information will be available online?

6) In what format should this textual information be displayed?

Items 5) and 6) refer to what is called additional information (AI) in the WISE SEARCH program document.

The following is an example of one possible AI format. (Note: the information was taken from record 98271 and not record 654321 which is shown above.)

```
    ED   98271
SMITH, KATHLEEN, ED.
DESEGREGATION/INTEGRATION: PLANNING FOR SCHOOL CHANGE. A TRAINING
PROGRAM FOR INTERGROUP EDUCATORS.
74 / UD
```

The format we have chosen displays the following information.

- citation number
- author(s)
- title
- date of publication/the first two alphabetic characters from the Clearinghouse accession number.

Obviously, a wide variety of other formats is also possible.

When all of the above questions have been answered, initial creation of the database may proceed.

Chapter 3

PROCESSING THE ORIGINAL INFORMATION SOURCE

Figure 1 depicts the overall process required to create an online WISE database. There are three programs involved: XX-READ, MERGE, and BUILD. XX-READ requires the most explanation because when you are building a new database, you must code the part of XX-READ that reads the original information source.

3.1 XX-READ

XX-READ is really a generic name which refers to a set of programs, each of which is coded to receive input from a particular information source. For example, RIE-READ is a program which reads the ERIC RIE tapes. CIJE-READ reads the ERIC CIJE tapes (Current Index to Journals in Education). You can name your own XX-READ program anything you like.

Functionally, XX-READ must be capable of reading the input medium on which the original information is stored, isolating the specific data fields of interest within each logical record, and writing two different output tape files. These output files will be referred to as the KEYWORD file and the AI (additional information) file. These output files are written in a standard WISE internal format and once they are created, all other processes for database creation are performed by the already-existing programs, MERGE and BUILD.

Briefly, the KEYWORD file contains information necessary to retrieve the citation numbers for those records containing a given keyword. The AI file contains the textual information which can be displayed online.

A set of subroutines is available for writing the KEYWORD file and another set is available for writing the AI file. The following sections describe how to use them.

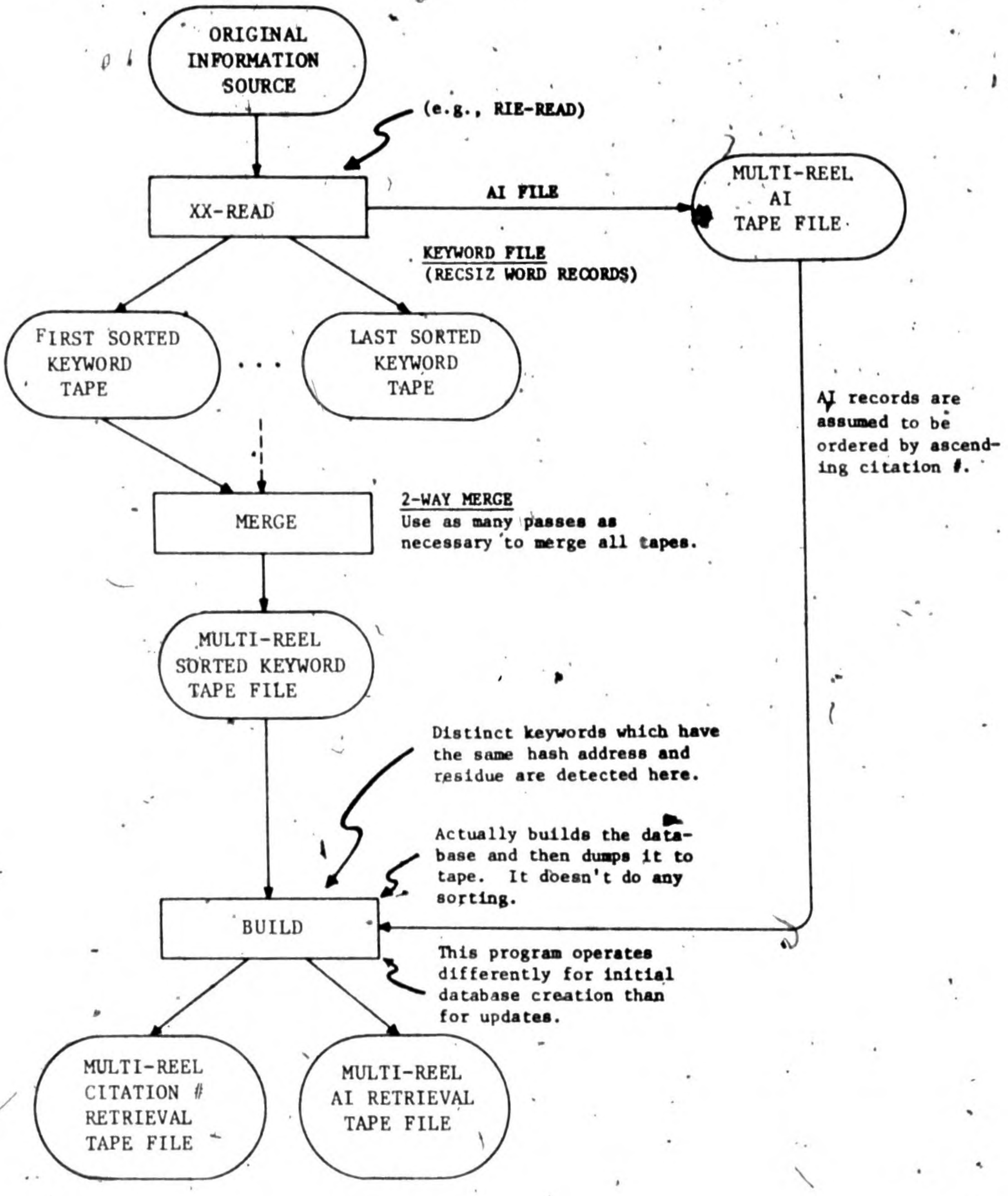

ORIGINAL INFORMATION SOURCE
(e.g., RIE-READ)
XX-READ
AI FILE
MULTI-REEL AI TAPE FILE
KEYWORD FILE (RECSIZ WORD RECORDS)
FIRST SORTED KEYWORD TAPE
. . .
LAST SORTED KEYWORD TAPE
AI records are assumed to be ordered by ascending citation #.
MERGE
2-WAY MERGE Use as many passes as necessary to merge all tapes.
MULTI-REEL SORTED KEYWORD TAPE FILE
Distinct keywords which have the same hash address and residue are detected here.
Actually builds the database and then dumps it to tape. It doesn't do any sorting.
BUILD
This program operates differently for initial database creation than for updates.
MULTI-REEL CITATION # RETRIEVAL TAPE FILE
MULTI-REEL AI RETRIEVAL TAPE FILE

Figure 1 – INITIAL ONLINE DATABASE CREATION

3.1.1 Writing the KEYWORD File

A set of three FORTRAN-callable subroutines is provided for writing the KEYWORD file.

KEYOPN (open the KEYWORD file)
KEYOUT (output KEYWORD records)
KEYCLS (close the KEYWORD file)

The calling sequences are as follows:

```
CALL KEYOPN(KEYREC,KEYSIZ,INLEN,RECSIZ,NUMBIT,HEAD,HEADL)
```

where:

KEYREC	- the name of the FORTRAN array which will contain the keyword records (see Section 3.1.2).
KEYSIZ	- the maximum length, in characters, of a keyword (suggested value is 60).
INLEN	- (KEYSIZ + 5)/6 + 3. This is the size of the array KEYREC.
RECSIZ	- the length of a new record created internally by subroutine KEYOUT based upon the information supplied in array KEYREC. This record contains a variable amount of diagnostic information based upon its length. RECSIZ <u>must</u> be $\geq$ 3 (suggested value is 6).
NUMBIT	- the number of bits in the hash address which is produced by a hashing function called internally by subroutine KEYOUT (suggested value is 13).
HEAD	- the name of a FORTRAN array containing control information for the database (see Section 3.1.2).
HEADL	- the length of the array HEAD. Currently, the value of HEADL should be 11. This may be expanded in the future if more control information is found to be desirable.

The subroutine, KEYOPN, should be called before the first call to the subroutine KEYOUT.

```
CALL KEYOUT
```

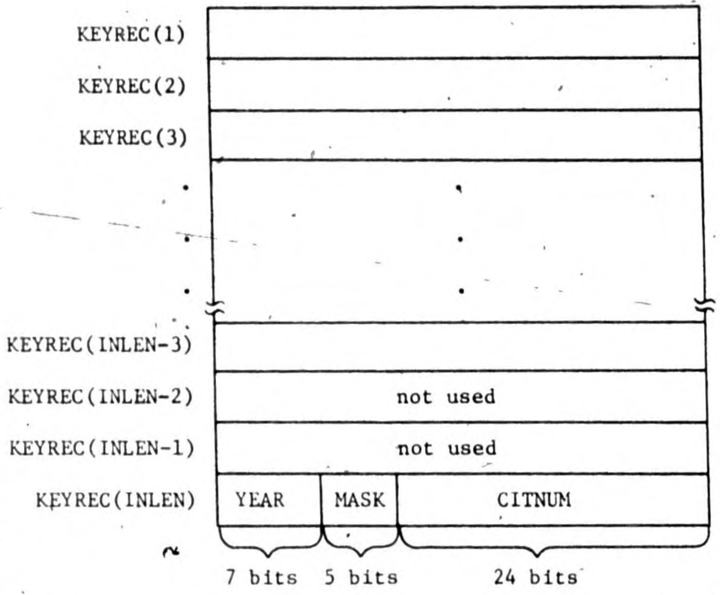
Each time KEYOUT is called, it sends information based on the next keyword record (assumed to be in array KEYREC) to the KEYWORD file.

CALL KEYCLS

This subroutine should be called when there are no more keyword records to pass to KEYOUT.

3.1.2 KEYWORD File Data Formats

Array KEYREC should look like this upon entry to KEYOUT.



Array elements KEYREC(1) through KEYREC(INLEN-3) inclusive are reserved for holding the Fieldata text of a keyword. The keyword must be left-justified and space-filled starting in element KEYREC(1).

Example:

If you choose the maximum size of a keyword to be 60 characters (i.e., KEYSIZ = 60), then INLEN = (KEYSIZ + 5)/6 + 3 = 13, and INLEN - 3 = 10. Therefore there are 10 array elements available for holding a keyword. This is just enough because exactly six Fieldata characters fit in each array element for FORTRAN V implemented for a UNIVAC 1100 series computer. If the keyword contains fewer than KEYSIZ - 5 characters, the array element immediately following the end of the keyword must contain six Fieldata spaces. This is very important!

YEAR - 7-bit field which contains a binary number representing the last two digits of the year of publication for a given citation. YEAR is primarily useful for bibliographic databases. If it is meaningless for your database, just use zero.

MASK - 5-bit field that contains the bit settings which indicate that the keyword is to be used as a main keyword or one of the four auxiliary keywords. (The leftmost bit in a word (array element) is called bit 35.)

bit 28 = 1 - means main keyword.

bit 27 = 1 - means first auxiliary keyword.

⋮

bit 24 = 1 - means fourth auxiliary keyword.

At least one bit must always equal 1. However, it is sometimes meaningful to have more than one bit equal 1.

CITNUM - 24-bit field which contains the binary for the citation number.

NOTE: The array element KEYREC(INLEN) is broken into three odd-sized pieces, 7 bits, 5 bits, and 24 bits. Those programmers who are not familiar with performing partial word manipulations in FORTRAN V are advised to read about the FLD function in section 4.3.2 of the *FORTRAN V Compiler Reference Manual for the 1108.*

Hint: Put the following six statements somewhere before the first executable statement of your program.

```
PARAMETER KEYSIZ = 60
PARAMETER INLEN  = (KEYSIZ + 5)/6 + 3
DIMENSION KEYREC(INLEN)
DEFINE YEAR      = FLD(0,7,KEYREC(INLEN))
DEFINE MASK      = FLD(7,5,KEYREC(INLEN))
DEFINE CITNUM    = FLD(12,24,KEYREC(INLEN))
```

Then later on in your program if you want to set the YEAR field to 74 for example, just use the statement

```
YEAR = 74.
```

Similarly, the statement

    MASK = 1

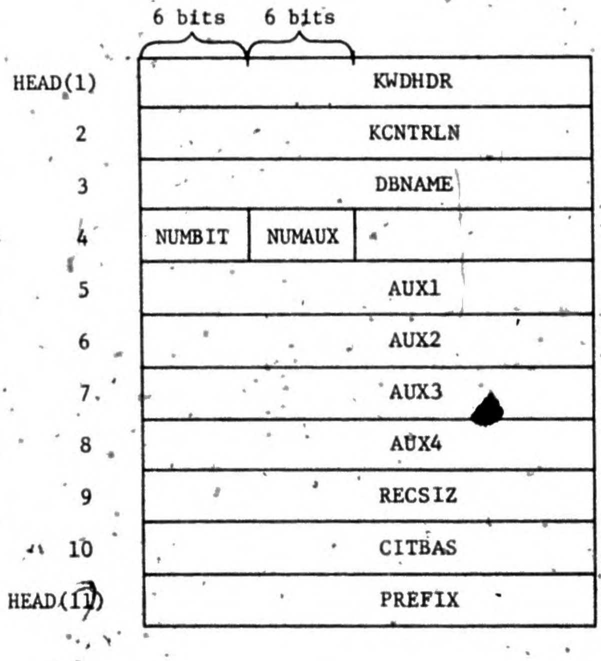
sets the rightmost bit in the MASK field and clears the other four bits. The statement

    CITNUM = 9923

puts the value 9923 into the CITNUM field without changing the current values in the YEAR and MASK fields.

Note: KEYREC(1) through KEYREC(INLEN-1) are changed by KEYOUT. Only KEYREC(INLEN) remains unchanged.

The array HEAD should look like this upon entry to KEYOPN.



Where:

KWDHDR - the six Fieldata characters: KWDHDR.

KCNTRLN - the length of array HEAD, currently = 11.

DBNAME - six Fieldata characters specifying the database name (you choose it yourself).

NUMBIT - the binary representation for the number of bits in the hash address (suggested value is 13).

NUMAUX - the binary representation for the number of auxiliary keywords defined for this database (should be between 0 and 4 inclusive).

AUX1 - six Fieldata characters, left-justified, space-filled, for the special function that denotes auxiliary key #1. Use all Fieldata spaces if this word is not used.

AUX2 - the same as AUX1 but for auxiliary key #2.

AUX3 - the same as AUX1 but for auxiliary key #3.

AUX4 - the same as AUX1 but for auxiliary key #4.

RECSIZ - the length of the internal record produced by KEYOUT (suggested value is 6).

CITBAS - the base value which is added to all the "relative" citation numbers referenced in this document in order to compute the actual citation number. The fields allocated for the citation number in both the KEYWORD records and the AI records are 24 bits long. Therefore the highest citation number that can be placed into these fields is $2^{24} - 1 = 16,777,215$. Most databases have citation numbers which range between numbers much smaller than $2^{24} - 1$. For these databases we use CITBAS = 0, and we put the actual citation number into the citation number fields. $(a + 0 = a)$, see: Stephen C. Kleene, *Introduction to Metamathematics*, (1950), p. 186. If your database starts with very large citation numbers and the largest numbers exceed $2^{24} - 1$, then you can still use WISE even though your citations numbers won't fit into a 24-bit field. Just set CITBAS equal to the lowest citation number -1. Then call your first citation number 1, the second number 2, etc.

The WISE system will always add CITBAS to compute the actual citation number.

Note: CITBAS does not allow us to have more than $2^{24} - 1$ distinct citation numbers. It just allows us to have very large citation numbers.

PREFIX - six Fieldata characters to be printed just before the citation number when additional information is not requested. For example, for the RIE database, PREFIX =^^^^ED

3.1.3 Writing the AI File

A set of three FORTRAN V callable subroutines is provided for writing the AI file.

AIOPEN (open the AI file)
AIOUT (output AI records)
AICLS (close the AI file)

The calling sequences are as follows.

```
CALL AIOPEN(AHEAD,AHEADL)
```

where:

AHEAD - the name of a FORTRAN array containing control information for the database (see Section 3.1.4).

AHEADL - the length of the array HEAD. Currently, the value of AHEADL should be 4. This may be expanded in the future if more control information is found to be desirable.

```
CALL AIOUT(AIREC)
```

AIOUT assumes that each time it is called a new AI record (see Section 3.1.4) has been placed into the array AIREC. Records should be presented to AIOUT in order of increasing citation number.

```
CALL AICLS
```

This subroutine should be called when there are no more AI records to pass to AIOUT.

3.1.4 AI File Data Formats

The array AHEAD should look like this upon entry to AIOPEN.

AHEAD(1)	AIHDR
(2)	ACNTRLN
(3)	not currently used
AHEAD(4)	ANUMBIT

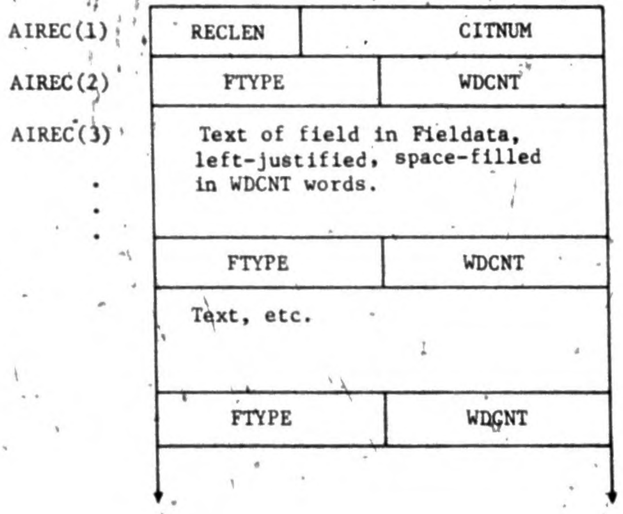
where:

AIHDR - the six Fieldata characters: AIHDR$

ACNTRLN - the length of the array AHEAD, currently = 4.

ANUMBIT - a value pertaining to an internal table used by the SEARCH program to retrieve additional information (suggested value is 13).

The array AIREC should look like this upon entry to AIOUT.



where:

RECLEN - a 12-bit field which contains the binary representation for the total record length (in words) of this AI record.

CITNUM - a 24-bit field which contains the binary representation for the citation number for this record.

(FTYPE and WDCNT comprise a print field control word.)

FTYPE - an 18-bit field containing the field type in binary.

WDCNT - an 18-bit field containing the number of text words in this field (it may be 0).

The additional information records placed into the array AIREC can be variable in length. The maximum length is $2^{12} - 1 = 4095$ words.

The fields are arranged in the order they will be printed, not necessarily in the order they are found in the original records. The print fields may be composites or subsets of the fields in the original data records. Therefore the field types may sometimes be internally generated, (by XX-READ).

Printing conventions for AI records (see p. 2-3 for an example):

The following printing conventions are followed by the print routine in the SEARCH program. The printer or terminal is spaced before printing the first line. The first line consists of the prefix word and the citation number. Field type 100 indicates the prefix word. This consists of exactly six Fieldata characters which are to be printed on the first line immediately preceding the citation number. The citation number is printed in I8 format. The text of the next field prints on the line immediately following the citation number. The text of a new field always starts printing on a new line. If a field is very long, it will be broken into as many lines as are necessary to print all of it. Any field in the AI record , including the prefix word may be omitted from an AI record.

3.1.5 Rules for Coding and Mapping XX-READ

1) All variables referred to in this document must be declared as type integer. This can be easily accomplished by using the statement

```
IMPLICIT INTEGER(A-Z)
```

as the first FORTRAN statement in XX-READ.

2) If your XX-READ program contains any READ statements, it must perform all of its READ's <u>before</u> the <u>first</u> call to KEYOUT. The most likely reason for doing a READ would probably be to find out the number of input tape reels.

3) Assuming that you are working with a file called A, the following two statements will properly map XX-READ for you.

```
@MAP,S NEW*WISE.XX-READ-MAP,A.XX-READ
IN A.XX-READ
```

You must substitute your own program name for the underlined occurrences of XX-READ above.

3.1.6 Debugging Aids for XX-READ

MACC has developed some utility routines which are useful for examining the contents of the two tape files produced by XX-READ. These are available on an informal basis from the MACC staff member responsible for WISE.

Chapter 4

INITIAL DATABASE CREATION

At this point, we assume that the appropriate XX-READ program has been coded, mapped and debugged. This portion of the document deals with the initial creation of a database (see Figure 1).

4.1 The Runstream for XX-READ

The following is a sample runstream for an XX-READ execution, assuming that the original information source is a set of 7-track tapes. XX-READ is assumed to be in file A.

```
    @RUN,/R runid,project,user,dollar limit
    @ASG,T  7.,T,<tape reel ID's corresponding to the original information
                 source tapes>
    @ASG,T  8.,T,<tape reel ID's for tapes to receive the KEYWORD file>
    @ASG,T  9.,T,<tape reel ID's for tapes to receive the AI file>
    @ASG,T  XB.,F/1800/TRK/2149
    @ASG,T  XA.,F/250/TRK/307
    @XQT,<options> A.XX-READ
    <Data images for any READ's performed prior to calling KEYOUT>
(1) <Data images to be read by the system sort routine only if the P
    option is specified on the @XQT statement>
    @FIN
```

Note: The tape reel ID's for units 8. and 9. must be preceded by $'s because they are output tapes. Full 2400 foot tape reels must be used for file 8. Short reels cause XX-READ to terminate in error. The files XB. and XA. are used by the system sort routine (called internally by KEYOUT). The file sizes were computed for RECSIZ $\leq$ 6.

Summary of XQT options:

NONE - The output tape files are produced in their standard format.

E - A software EOF (end of file) record is not written on the last tape of the AI file. This is used when an initial database creation must be split into several executions (e.g., because of elapsed time consideration). In this case, all executions <u>except</u> the <u>last</u> one should use the E XQT option.

H - The AI header record is not written when AIOPEN is called. This is also necessary when multiple executions are required. Every execution except the first one should use the H option.

P - If the P option is specified, the system sort routine (called by subroutine KEYOUT) reads information from the runstream in an attempt to optimize its sorting. (See Section 4.5 of the MACC manual, *File Sorting from FORTRAN*.) A pair of BIAS and @EOF cards should be placed at (1) above, for each tape in the KEYWORD output file, (FILE 8. above). A bias value of 1.9 is recommended.

4.2 Using MERGE

If XX-READ produces only one KEYWORD tape during an initial database creation, then the MERGE program is not necessary. The single, sorted keyword tape may be directly input to the BUILD program. However, if more than one KEYWORD tape was produced, the tapes must be merged before input to BUILD. This merging is necessary because each individual tape is the result of sorting only part of the total set of keyword records. MERGE is coded to perform two-way merges. Therefore, if n tapes were produced, then n - 1 MERGE runs are necessary to get them all merged.

You can see from Figure 2 that MERGE can read files written both by XX-READ and previous MERGE runs.

EXAMPLE:

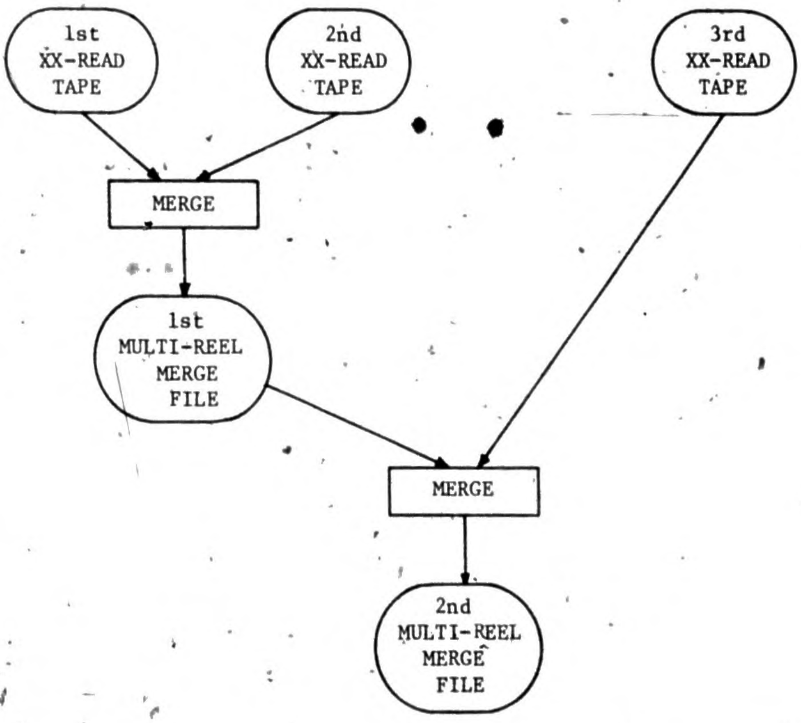


Figure 2. Merging

.2.1 The Runstream for MERGE

The following should do the trick.

```
@RUN,/R runid,project,user,dollar limit
@ASG,T  7.,T,<tape reel ID's for first input file>
@ASG,T  8.,T,<tape reel ID's for second input file>
@ASG,T  9.,T,<tape reel ID's for output file>
@XQT    NEW*WISE.MERGE
@FIN
```

4.3 Using BUILD

During an initial creation, BUILD reads KEYWORD tapes written by MERGE (or XX-READ) and AI tapes written by XX-READ. It then creates a two part database which it nicely dumps onto two multi-reel tape files:

1) citation number retrieval structure,
2) AI retrieval structure.

The input AI tapes may be missing, depending on the XQT options specified. Of course, input tape files must be consistent with the options specified.

XQT Options:

None - initial build, only keyword tapes are expected.

U (alone) - update, only keyword tapes are expected.

A - AI tapes are also expected.

H - bypass AI header records found on AI tapes other than the first tape. This is for people who forgot to use the H option on the appropriate XX-READ executions. Without the H option, two header records in the same AI file is considered an error and BUILD will perform an error stop because the wrong tapes are probably mounted.

4.3.1 The Runstream for BUILD (Initial Creation)

The following runstream is appropriate for an initial database creation where both keyword tapes and AI tapes are expected as input.

```
@RUN,/R runid,project,user,dollar limit
@ASG,T  7.,T,<tape set 1>/<tape set 2>/<tape set 3>
@XQT,A  NEW*WISE.BUILD
@FIN
```

<tape set 1> = a set of tape reel ID's representing the output from MERGE (or XX-READ)

<tape set 2> = a set of tape reel ID's representing the multi-reel AI tape file produced by XX-READ

<tape set 3> = a set of tape reel ID's representing output tapes that will receive:

1) the multi-reel citation number retrieval tape file
2) the multi-reel AI retrieval tape file.

In general it is a good idea to supply a large number of tapes in tape set 3. BUILD starts writing the multi-reel citation number retrieval structure on the first output tape. It uses as many tapes as are necessary to dump the whole structure. When this process is done, it prints the message:

EOF RECORD WRITTEN ON TAPE REEL # n .

where n is the nth output reel (by this means, you know where the multi-reel file ended). In the example above, an AI retrieval structure is also to be written, so BUILD will switch to the next output tape and start dumping the AI structure. When this process is done you get another EOF message. The value of n is to be interpreted to be the nth tape in tape set 3, not the nth tape within a multi-reel file.

Chapter 5

RUNNING SEARCHES ON NON-MACC DATABASES

Databases not maintained by MACC are referred to as non-standard databases (non-standard in the sense that the tapes on which they reside are not known internally to the SEARCH program). The following computer output demonstrates how to load a non-standard database.

```
@ASG,T 7.,T,10615/10616
READY
@NEW*WISE.SEARCH,N YOUR*DATABASE.
SEARCH - VERSION 01.02   DATE:042676 TIME:170447
DO YOU WANT TO PAY TO LOAD THE DATABASE?  YES
DO YOU ALSO WANT TO LOAD THE ADDITIONAL INFORMATION?  YES
DO YOU WANT TO CATALOG THE DATABASE?  YES
PARDON ME WHILE I COLLECT MY MEMORY
DATABASE LOADED
TO OBTAIN INFORMATION ON HOW TO USE THIS PROGRAM, INPUT:
/HELP
FOR COMMENTS OF GENERAL INTEREST TO WISE USERS, INPUT:
/HELP
NEWS

ACTION

:HIGHER EDUCATION
SET   1   410 ENTRIES

:/QUIT

END SEARCH
@SAVE,S YOUR*DATABASE.
 MACC SAVE PROCESSOR VERSION 6.0406
 YOUR*DATABASE(1) TO BE SAVED INDEFINITELY
 END SAVE...
@FIN
```

There are two important things to remember when loading a non-standard database:

1) A tape assign statement for unit 7., must precede the @NEW*WISE.SEARCH,N processor call statement.

2) The SEARCH program must be called with an N option as shown above. This tells SEARCH that it is loading a non-standard database.

5.1 The Tape Assign Statement

The general form of the tape assign statement is:

```
@ASG,T 7.,<tape set 1>/<tape set 2>
```

<tape set 1> is a set of reel ID's corresponding to a multi-reel citation number retrieval tape file (the first output file from BUILD).

<tape set 2> is a set of reel ID's corresponding to a multi-reel AI retrieval tape file (the second output file from BUILD). This set may be absent if you don't want to load the AI file.

In the above example, tape set 1 consists of exactly one tape, 10615. Tape set 2 consists of exactly one tape, 10616.

5.2 The Processor Call Statement

The general form of the processor call statement for non-standard databases is

```
@NEW*WISE.SEARCH,N  FILENAME.
```

or

```
@NEW*WISE.SEARCH,N  QUALIFIER*FILENAME.
```

The final period is mandatory! FILENAME. and QUALIFIER*FILENAME. are the two most common ways of specifying an EXEC-8 filename. (See the *MACC Computing Handbook*, Section 3.2, for EXEC-8 file naming conventions.)

During an initial load, SEARCH creates a file with the name specified on the processor call statement, and either catalogs it (PUBLIC, READ-ONLY) or makes it a temporary file. (This is determined by the user's response to the question "DO YOU WANT TO CATALOG THE DATABASE?".) It then loads the database from the tapes on unit 7 and internally @FREE's the tape file. SEARCH then enters ACTION mode and you may start searching as described in the WISE SEARCH program document.

SEARCH does not SAVE the file it catalogs. If you want your file to be saved you must do it yourself with the @SAVE command. (See the *MACC Computing Handbook*, Section 3.9.2, and the section on @SAVE in the blue pages.)

A note of caution: WISE database files are cataloged with position granularity. If your file exceeds 23 positions, MACC will not be able to actually save your file for you. However, the @SAVE processor will not notify you of this condition. Although the situation is not very likely because 23 positions represents an extremely large file, it is nevertheless a good idea to check the size of your file with the statement:

```
@PRT,F  YOUR*DATABASE.
```

If you have more than 23 positions, contact MACC in order to make your file a MACC standard database. This will solve the problem.

The following is an example of accessing a non-standard database that was previously loaded and cataloged.

```
@NEW*WISE.SEARCH,N YOUR*DATABASE.
SEARCH - VERSION 01.02   DATE:042676 TIME:171310
TO OBTAIN INFORMATION ON HOW TO USE THIS PROGRAM, INPUT:
/HELP
FOR COMMENTS OF GENERAL INTEREST TO WISE USERS, INPUT:
/HELP
NEWS

ACTION

:HIGHER EDUCATION
SET    1    410 ENTRIES

:INSTRUCTIONAL MATERIALS
SET    2    243 ENTRIES

:1*2
SET    3      9 ENTRIES

:/QUIT

END SEARCH
@FIN
```

Chapter 6

DATABASE UPDATE

Figure 3 is an overview of how to perform a database update. The operation of XX-READ and MERGE should be obvious from the diagram. The runstreams should be about the same as those already shown for initial database creation. The only program that operates differently during an update is BUILD.

Basically the differences are:

1) BUILD must be called with a U XQT option (and others if appropriate).

2) The tape assign statement for unit 7 is different.

6.1 The Runstream for BUILD (Database Update)

The following runstream would be appropriate for updating a database that contained both a citation number retrieval tape file and an AI retrieval tape file.

```
@RUN,/R runid,project,user,dollar limit
@ASG,T  7.,T,<tape set 1>/.../<tape set 4>
@XQT,UA NEW*WISE.BUILD
@FIN
```

<tape set 1> = the multi-reel sorted keyword tape file from the update MERGE run (output from MERGE in Figure 3).

<tape set 2> = the multi-reel AI retrieval tape file that was created by BUILD during the previous update (or initial creation)

<tape set 3> = the multi-reel AI tape file created by XX-READ during the current update process.

<tape set 4> = a set of tapes to receive the new (updated) multi-reel citation number retrieval tape file, and the new (updated) multi-reel AI retrieval tape file. The output tapes are processed exactly as in an initial database creation.

When you want to load the new database that was written on tape set 4:

1) Delete the old database (@DELETE,C YOUR*DATABASE. will work);

2) Use the SEARCH program to load the updated database from the new tapes.

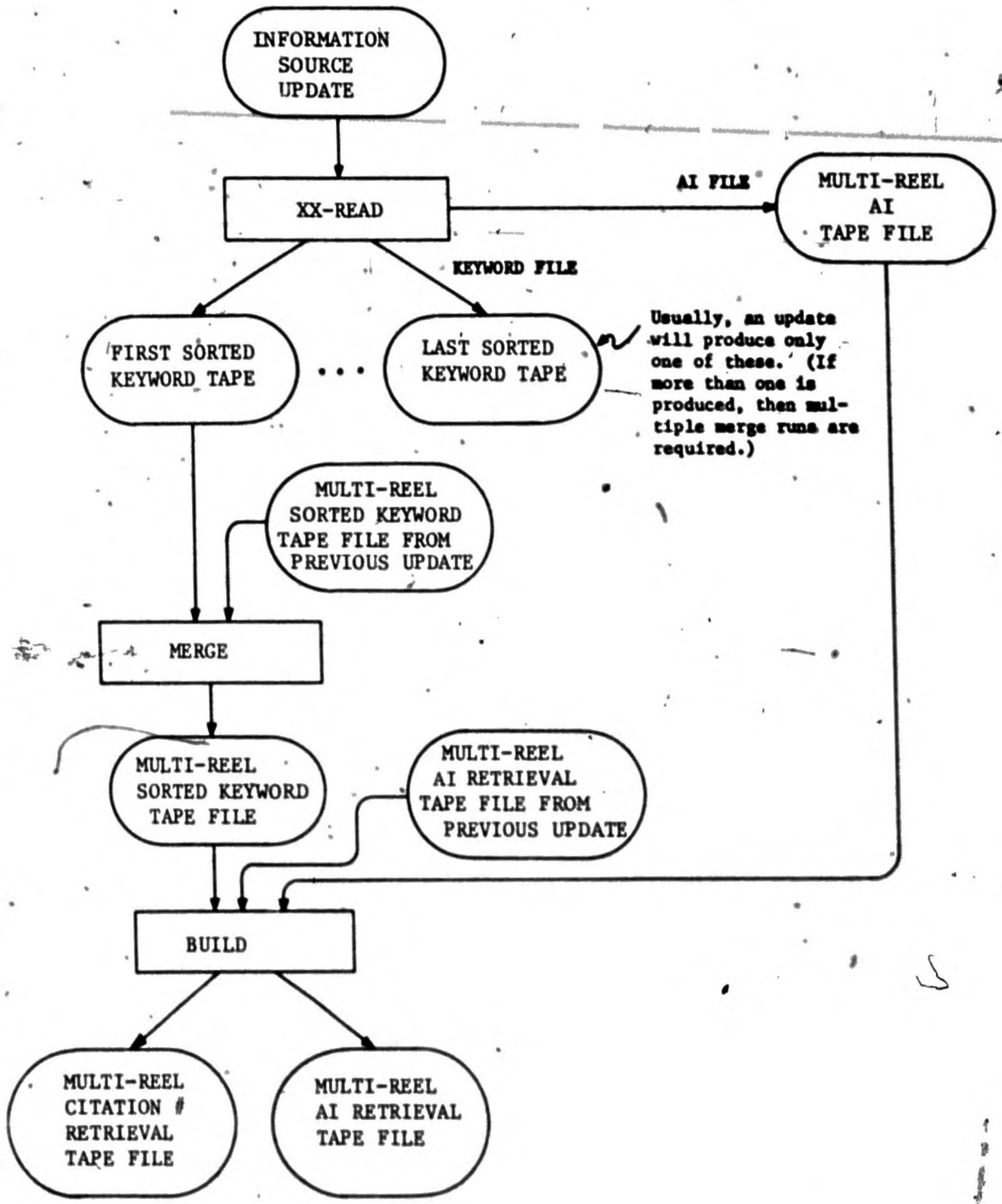

INFORMATION SOURCE UPDATE
XX-READ
AI FILE
MULTI-REEL AI TAPE FILE
KEYWORD FILE
FIRST SORTED KEYWORD TAPE
• • •
LAST SORTED KEYWORD TAPE
Usually, an update will produce only one of these. (If more than one is produced, then multiple merge runs are required.)
MULTI-REEL SORTED KEYWORD TAPE FILE FROM PREVIOUS UPDATE
MERGE
MULTI-REEL SORTED KEYWORD TAPE FILE
MULTI-REEL AI RETRIEVAL TAPE FILE FROM PREVIOUS UPDATE
BUILD
MULTI-REEL CITATION # RETRIEVAL TAPE FILE
MULTI-REEL AI RETRIEVAL TAPE FILE

Figure 3 — ONLINE DATABASE UPDATE