cc-69-2

*KWOC - Automatic Indexing by Keyword

by

G. Rose

and

Fran Spigai

OSU

COMPUTER CENTER

Oregon State University
Corvallis, Oregon 97331

*KWOC

Automatic Indexing by Keyword

by

G. Rose
Fran Spigai

cc-69-2
Computer Center
Oregon State University
Corvallis, Oregon 97331

February, 1969

1. INTRODUCTION

Webster's Third New International Dictionary defines an index as: "a usually alphabetical list that includes all or nearly all items (as topics, names of people and places) considered of special pertinence and fully or partially covered or merely mentioned in a printed or written work (as a book, catalog, or dissertation)..."

Subject indexing is conventionally an intellectual effort that relies on knowledge of a subject area and the use of a structured vocabulary from which to choose index terms (e.g. glossary, thesaurus, classification scheme, etc.). Thus, manual subject indexing requiring training and ability, results in a time lag between receipt of a document and its subsequent availability through an index or catalog.

Concordances utilized an indexing technique which required only that significant words in the text of a document be identified and arranged in an alphabetical array as the index to the document. H. P. Luhn was the first to utilize this technique with the aid of a computer to produce KWIC (keyword-in-context) indexes to documents. The title of a document and an ID code are the input; the output is a series of permuted titles, one for each "keyword" (articles, conjunctions, etc. are suppressed).

The KWOC (keyword-out-of-context) index is a variation of the KWIC index and was developed at Stanford Research Institute.[1]

The variations, advantages, disadvantages, and myriad applications of KWIC-type indexes can be explored in other publications.[2,3]

Some advantages can be cited briefly here:

No content analysis is necessary, all indexing is fully automatic, thus cutting indexing time and costs. Index terms are directly representative of the author's terminology and thus are not constrained by obsolete and restrictive indexing schemes. KWIC-type indexes provide good depth of indexing using an automatic method (e.g. each title would be indexed by as many keywords as it contains) thereby simultaneously providing a coordinate index of title "keywords" and a crude associative index. The resultant index is simple to use and requires no training.

2. PROGRAM DESCRIPTION

A KWOC processor is available in the OS-3 operating system. The program is filed under the name *KWOC. This program may be used to generate a "permuted index" to a set of document titles, or, more generally, to a set of variable length records stored in some file. Output from the program is a file that consists of an alphabetically ordered set of keywords, each of which is followed immediately by an alphabetically ordered list of all records containing these keywords. A condensed list of keywords is also output. Since not all words in a title are information-bearing keywords (e.g. and, the, etc.), *KWOC also accepts as input a file of words that are ignored during indexing.

Although *KWOC is extensively parameterized, default options do exist which satisfy most cases. The program is called by the control statement *KWOC followed by a string of parameters. Parameters that may be used are as follows:

I or Input = file name or logical unit (lun) of input file. If no file exists under the given name or if the specified lun is not equipped, the program will abort. If the I parameter is not present in the parameter string, lun 60 is assumed.

O or Output = file name or lun of output file. If no file exists under the given name, one will be created; if no lun exists, one will be equipped. Format of the output file is: KWOC Index, EOF, Keyword List, EOF. If the O parameter is not present in the parameter string, lun 61 is assumed.

S or Suppress = file name or lun of a file that contains any words that should not be used as keywords. This file should contain one such word per record with a file mark at the end of the file. A file of about 150 commonly suppressed words is available under the name *SUPPRES. If the S parameter is not present, all words will be treated as keywords.

L = the maximum line length, in characters, of output records. Since each record is indented with respect to the keyword (see examples), the length of each output record will be twelve characters greater than the declared line size, i.e. twelve leading blanks are inserted in each line. If line size is not specified as a parameter, output records will have a maxiumum length of 110 characters (a convenient size for printed output).

B or Break = An internal BCD code used to break an output line if the output record length exceeds the current line size. Output will then occur in two or more consecutive lines. A table of characters and their internal BCD codes is given in Appendix I. If the value of B exceeds $(100)_8$, the line will be broken exactly at its maximum length. Otherwise, the line will be scanned from right to left for the first occurrence of the specified BCD character, and the line broken at that point. The break character itself will not appear in the output. If the break character is not present in a line, the line will be broken at its maximum length. If the Break parameter is not present in the parameter string, the program will break oversized lines at a space = (60_8).

X = delimiting character, not an alphanumeric, where scanning is to commence in each input record. X is given as an internal BCD code. If no X is given in the parameter string, scanning will begin at the first character of each input record.

Y = delimiting character, not an alphanumeric, where scanning is to stop in each input record. Y is an internal BCD code. If no Y is specified, scanning will continue until the end of every input record.

T or Table = file name or logical unit of a file which contains a table used to define those character strings that are to be construed as words. For example, the program in normal operation will never try to use a comma as a keyword, and any keyword followed by a comma is, in effect, delimited by that comma. In particular, a word is defined as an alphabetic character followed by a string of consecutive alphabetic or numeric characters. Thus, "CDC3300" is a word but "3300CDC" is not. All characters that are not alphabetics or numerics are delimiters except a space, which is a delimiter of a special sort. The Table parameter allows the user to redefine the set of recognized alphabetic characters. The class of any character may be changed to an alphabetic with the exception of the space. If T is used, then any characters that are not declared to be alphabetic become delimiters, except numbers. Numbers remain as numbers unless they were redeclared to be alphabetics. The format of the file specified by the T parameter should be one character per record with a file mark at the end of the file. If the T parameter is not present in the parameter string, then the set of characters used as alphabetics consists of the 26 letters of the alphabet together with the dash (-).

3. CONSTRAINTS

The maximum size of a keyword is forty characters. Any keyword that exceeds this length will be truncated. In addition, the maximum size of an input record is 1000 characters.

The program uses various logical units as scratch files. These are in the range 50-59. If the program requires a lun, it will unequip any logical unit in this range found to be equipped.

The program is highly space consuming. This can easily be seen since if an input record contains ten keywords, then that input record will occur ten times in the output file. In addition, scratch space of at least twice the size of the output file is required internally in order to sort the output file. Since most of the time required to produce a large index is spent sorting the output file into alphabetic order, the user is referred to the SORT/MERGE manual (cc-68-37) for timing considerations.

4. EXAMPLES

The first example will deal with a file named TEST:

```
KWOC INDEXING, ITS USES AND ABUSES
LITTLE KNOWN FACTS ABOUT KWOC INDEXING
MARY HAD A LITTLE LAMB
```

The KWOC index for this file will utilize 3 parameters: Input, Output and Line Size; lines break on a blank; no terms are suppressed; scanning begins with the first term and continues through the end of line, since neither X nor Y is specified; and, since the parameter T is not referenced, words are defined as character strings beginning with the characters A through Z and the dash, and followed by the characters A through Z, the dash, or the numbers 0 through 9.

The KWOC parameter string will look like this:

```
*KWOC,I=TEST,O=EXAMPLE,L=35
```

The output, EXAMPLE, follows:

```
A
         MARY HAD A LITTLE LAMB

ABOUT
         LITTLE KNOWN FACTS ABOUT KWOC
         INDEXING

ABUSES
         KWOC INDEXING, ITS USES AND ABUSES


AND
         KWOC INDEXING, ITS USES AND ABUSES


FACTS
         LITTLE KNOWN FACTS ABOUT KWOC
         INDEXING

HAD
         MARY HAD A LITTLE LAMB
```

```
INDEXING
         KWOC INDEXING, ITS USES AND ABUSES


         LITTLE KNOWN FACTS ABOUT KWOC
         INDEXING

ITS
         KWOC INDEXING, ITS USES AND ABUSES


KNOWN
         LITTLE KNOWN FACTS ABOUT KWOC
         INDEXING

KWOC
         KWOC INDEXING, ITS USES AND ABUSES


         LITTLE KNOWN FACTS ABOUT KWOC
         INDEXING

LAMB
         MARY HAD A LITTLE LAMB

LITTLE
         LITTLE KNOWN FACTS ABOUT KWOC
         INDEXING

         MARY HAD A LITTLE LAMB

MARY
         MARY HAD A LITTLE LAMB

USES
         KWOC INDEXING, ITS USES AND ABUSES


A
ABOUT
ABUSES
AND
FACTS
HAD
INDEXING
ITS
KNOWN
KWOC
LAMB
LITTLE
MARY
USES
```

It can be seen from the first example that many "keywords" are not useful and it would be desirable to have these words suppressed. A sample of common terms to be suppressed which are contained in the file *SUPPRES follows:

```
A
ABOUT
AD
ALL
AMONG
AN
AND
ANOTHER
ARE
AS
AT
BASED
BE
BETWEEN
BY
.
.
.
```

By adding the S parameter,

```
(*KWOC,I=TEST,O=XAMPLE,L=50,S=*SUPPRES)
```

the following output is the result:

```
ABUSES
         KWOC INDEXING, ITS USES AND ABUSES
FACTS
         LITTLE KNOWN FACTS ABOUT KWOC
         INDEXING
HAD
         MARY HAD A LITTLE LAMB
INDEXING
         KWOC INDEXING, ITS USES AND ABUSES

         LITTLE KNOWN FACTS ABOUT KWOC
         INDEXING
KNOWN
         LITTLE KNOWN FACTS ABOUT KWOC
         INDEXING
KWOC
         KWOC INDEXING, ITS USES AND ABUSES

         LITTLE KNOWN FACTS ABOUT KWOC
         INDEXING
```

```
LAMB
         MARY HAD A LITTLE LAMB
LITTLE
         LITTLE KNOWN FACTS ABOUT KWOC
         INDEXING

         MARY HAD A LITTLE LAMB

MARY
         MARY HAD A LITTLE LAMB
USES
         KWOC INDEXING, ITS USES AND ABUSES

ABUSES
FACTS
HAD
INDEXING
KNOWN
KWOC
LAMB
LITTLE
MARY
USES
```

One can create a suppression list through COPY, *TVCOPY, *TVE or EDIT by inputting the desired words, one word per input record. It is more likely that one would want to expand the list of words found in *SUPPRES. For example, one might want the words HAD, KNOWN, LITTLE and USES suppressed, in addition to A, ABOUT, AND and ITS in the file TEST. The file *SUPPRES can be copied, and through the INSERT or APPEND commands in EDIT, these additional terms can be added.

Another file to be used for input, which contains citations for 3 technical reports and one book is LBJ:

```
  BLACKWELL, FREDERICK W. *ON-LINE COMPUTER SYMBOLIC MANIPULATION.*
$1966$ =QA76-B55=
  BORKO, HAROLD. *THE BOLD (BIBLIOGRAPHIC ON-LINE DISPLAY) SYSTEM.*
$1967$ =AD-632473=
  KELLOGG, C.H. *ON-LINE TRANSLATION OF NATURAL LANGUAGE QUESTIONS
INTO ARTIFICIAL LANGUAGE QUERIES.* $1967$ =AD-643494=
  BORKO, HAROLD, ET AL. *ON-LINE INFORMATION RETRIEVAL USING
ASSOCIATIVE INDEXING.* $1968$ =AD-670195=
```

Assuming you wish only the title portion to be scanned in your index, and, in no case should the call number or report number be broken up in the resultant index, the parameter string would look like this:

```
*KWOC,I=LBJ,S=HHH,L=40,X=54,Y=54,B=13,O=RMN
```

(Where HHH is a suppression file you have created, line size= 40 characters, scanning begins with data following the first asterisk (BCD code 54) and ends with the next asterisk encountered, the break is on the equal sign (BCD code 13), and your output file is RMN.)

RMN will appear as follows:

```
ARTIFICIAL
        KELLOGG, C.H. *ON-LINE TRANSLATION OF NA
        TURAL LANGUAGE QUESTIONS INTO ARTIFICIAL
         LANGUAGE QUERIES.* $1967$ =AD-643494=


ASSOCIATIVE
        BORKO, HAROLD, ET AL. *ON-LINE INFORMATI
        ON RETRIEVAL USING ASSOCIATIVE INDEXING.
        * $1968$ =AD-670195=

BIBLIOGRAPHIC
        BORKO, HAROLD. *THE BOLD (BIBLIOGRAPHIC
        ON-LINE DISPLAY) SYSTEM.* $1967$ =
        AD-632473=

BOLD
        BORKO, HAROLD. *THE BOLD (BIBLIOGRAPHIC
        ON-LINE DISPLAY) SYSTEM.* $1967$ =
        AD-632473=

COMPUTER
        BLACKWELL, FREDERICK W. *ON-LINE COMPUTE
        R SYMBOLIC MANIPULATION.* $1966$ =
        QA76-B55=

DISPLAY
        BORKO, HAROLD. *THE BOLD (BIBLIOGRAPHIC
        ON-LINE DISPLAY) SYSTEM.* $1967$ =
        AD-632473=
```

INDEXING

BORKO, HAROLD, ET AL. *ON-LINE INFORMATI
ON RETRIEVAL USING ASSOCIATIVE INDEXING.
* 1968 =AD-670195=

INFORMATION

BORKO, HAROLD, ET AL. *ON-LINE INFORMATI
ON RETRIEVAL USING ASSOCIATIVE INDEXING.
* 1968 =AD-670195=

LANGUAGE

KELLOGG, C.H. *ON-LINE TRANSLATION OF NA
TURAL LANGUAGE QUESTIONS INTO ARTIFICIAL
LANGUAGE QUERIES.* 1967 =AD-643494=

MANIPULATION

BLACKWELL, FREDERICK W. *ON-LINE COMPUTE
R SYMBOLIC MANIPULATION.* 1966 =
QA76-B55=

NATURAL

KELLOGG, C.H. *ON-LINE TRANSLATION OF NA
TURAL LANGUAGE QUESTIONS INTO ARTIFICIAL
LANGUAGE QUERIES.* 1967 =AD-643494=

ON-LINE

BLACKWELL, FREDERICK W. *ON-LINE COMPUTE
R SYMBOLIC MANIPULATION.* 1966 =
QA76-B55=

BORKO, HAROLD. *THE BOLD (BIBLIOGRAPHIC
ON-LINE DISPLAY) SYSTEM.* 1967 =
AD-632473=

BORKO, HAROLD, ET AL. *ON-LINE INFORMATI
ON RETRIEVAL USING ASSOCIATIVE INDEXING.
* 1968 =AD-670195=

KELLOGG, C.H. *ON-LINE TRANSLATION OF NA
TURAL LANGUAGE QUESTIONS INTO ARTIFICIAL
LANGUAGE QUERIES.* 1967 =AD-643494=

QUERIES

KELLOGG, C.H. *ON-LINE TRANSLATION OF NA
TURAL LANGUAGE QUESTIONS INTO ARTIFICIAL
LANGUAGE QUERIES.* 1967 =AD-643494=

QUESTIONS

KELLOGG, C.H. *ON-LINE TRANSLATION OF NA
TURAL LANGUAGE QUESTIONS INTO ARTIFICIAL
LANGUAGE QUERIES.* 1967 =AD-643494=

```
RETRIEVAL
        BORKO, HAROLD, ET AL. *ON-LINE INFORMATI
        ON RETRIEVAL USING ASSOCIATIVE INDEXING.
        * $1968$ =AD-670195=


SYMBOLIC
        BLACKWELL, FREDERICK W. *ON-LINE COMPUTE
        R SYMBOLIC MANIPULATION.* $1968$ =
        QA76-B55=


SYSTEM
        BORKO, HAROLD. *THE BOLD (BIBLIOGRAPHIC
        ON-LINE DISPLAY) SYSTEM.* $1967$ =
        AD-632473=


TRANSLATION
        KELLOGG, C.H. *ON-LINE TRANSLATION OF NA
        TURAL LANGUAGE QUESTIONS INTO ARTIFICIAL
         LANGUAGE QUERIES.* $1967$ =AD-643494=


ARTIFICIAL
ASSOCIATIVE
BIBLIOGRAPHIC
BOLD
COMPUTER
DISPLAY
INDEXING
INFORMATION
LANGUAGE
MANIPULATION
NATURAL
ON-LINE
QUERIES
QUESTIONS
RETRIEVAL
SYMBOLIC
SYSTEM
TRANSLATION
```

The pre-coordination (or "linking") of terms, concepts and names can be accomplished by treating multiple words as a character string. For example, in file LBJ, if scanning begins with the beginning of each record, the authors name(s) will be picked up in addition to title words. It would be desirable to have the complete name kept together in a lengthy list (e.g. SMITH,JOHN-J. or SMITH,J.J. instead of SMITH) for sorting purpose, and will also prevent a printout of JOHN and J separately. If table T is created to include the letters A-Z, the hyphen (-), the period (.) and the comma (,), the name Smith, J.J. will be recognized as one character string, and will print out as one keyword.

Likewise, terms considered concepts (i.e. information-retrieval, heat-transfer) and other types of names (e.g. corporate names: Control-Data-Corp., or journal names: SOFTWARE-AGE) will provide a more meaningful and economical index if kept together. Only a small amount of extra preparation on input is required.

[1] Stevens, Mary Elizabeth. Automatic Indexing: A State-of-the-Art Report. N.B.S. Monograph 91. 1965. p. 48.

[2] Borko, Harold, ed. Automated Language Processing: the State of the Art. Wiley: 1967.

[3] Fischer, Marguerite. The KWIC Index Concept: A Retrospective View. AMERICAN DOCUMENTATION 17: 57-70, April 1966.

Appendix I

Delimiters available on the CRT, teletype, and keypunch and their BCD Codes

	Characters		BCD Code	Characters	
	Keypunch	Teletype		CRT	
colon		:	12	:	
equal	=	=	13	=	
ampersend		&	15	≤	less than or equal to
percent		%	16	%	
left bracket		[	17	[	
plus	+	+	20		
less than		<	32	<	
period	.	.	33	.	
right paren	)	)	34	)	
sharp		#	35	≥	more than or equal to
quotation mark		"	36	-	(carriage return)
semicolon		;	37	±	plus or minus
exclamation		!	52	∨	upside down caret
dollar sign	$	$	53	$	
asterisk	*	*	54	*	
North arrow		↑	55	↑	
more than		>	57	>	
blank	space	space	60	space	
slash	/	/	61	/	
right bracket		]	72	]	
comma	,	,	73	,	
left paren	(	(	74	(	
question mark		?	77	∧	caret

The BCD codes 13, 33, 34, 53, 54, 60, 61, 73, and 74 are the compatible codes on all 3 input devices.

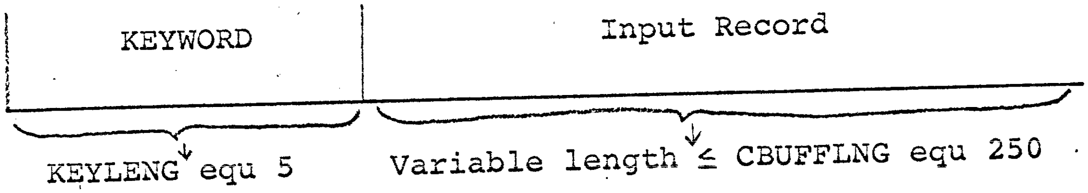
*KWOC Overview

The KWOC program, stored in a file named *KWOC as an overlay, consists of:

1. KWOC1: KWOC and PARPROC
2. SORTX

and 3. KWOC2

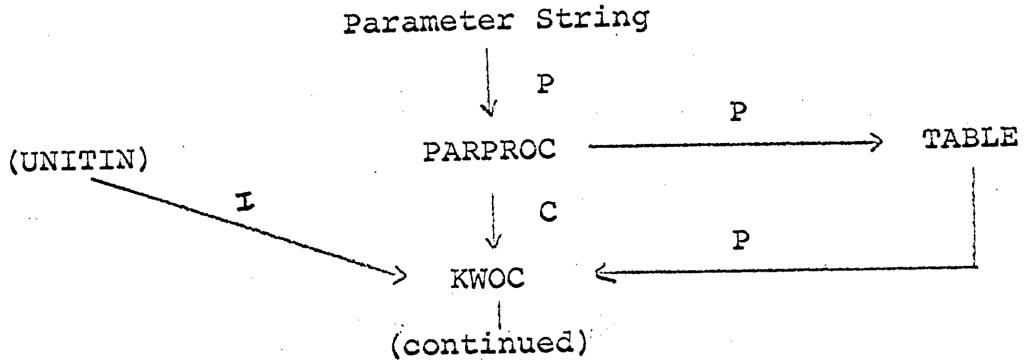
PARPROC accepts the parameter string from the user and leaves parameters in a table called TABLE. Control is then passed to KWOC. KWOC reads variable length input records from the lun contained in the word UNITIN and writes output records on TEMPOUT (equ 50). Format of each output record is:



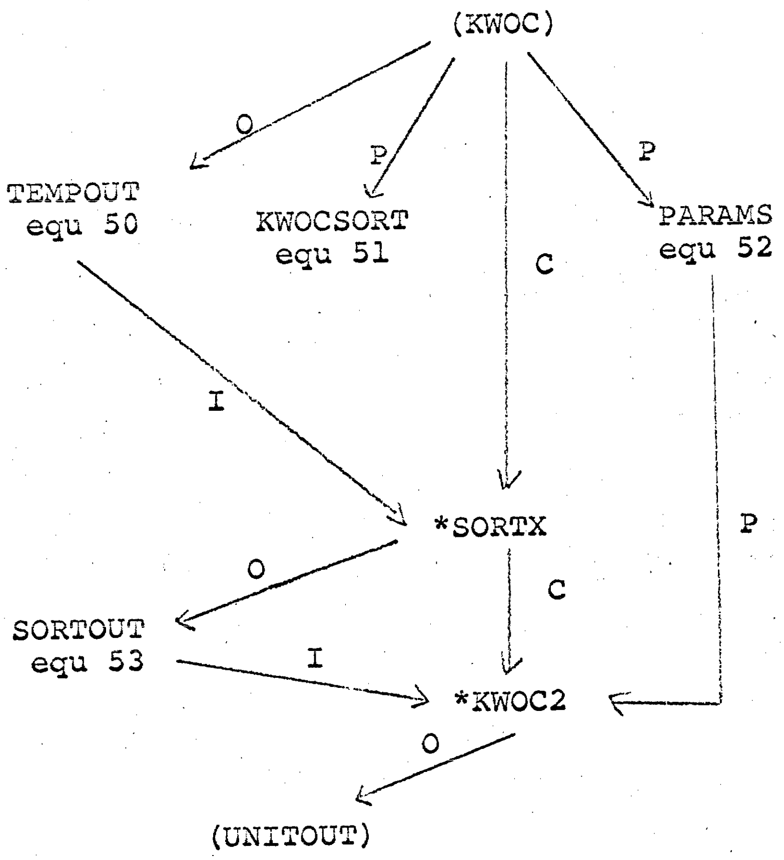
Program then reads in *SORTX (a modified version of *SORT) as an overlay and output records are sorted by keyword, then by the rest of the record.

*SORTX exits to *KWOC2 which is read in as an overlay. The program outputs sorted records to the lun contained in UNITOUT, writing each unique keyword as a hanging head. After all records are processed in this way, a file mark is written on (UNITOUT) followed by the list of keywords that were used and a second file mark.

Transfer of Parameters, Input, and Output During Execution



C = Control
P = Parameters
O = Output
I = Input



KWOC Operation

1. Equips lun's 50-54 as scratch files with *SORTX on 54.

2. Changes Break, Begncode, and ENDCode back to octal. They were assumed to be decimal numbers by PARPROC which is lun oriented, but were really BCD codes.

3. Writes contents of TABLE on PARAMS to be used by *KWOC2.

4. Reads in list of suppressed words from (SUPPRESS) and stores them in SLIST. If (SUPPRESS) contained a not accessed bit, this operation is ignored. Format of SLIST is

   ```
   <WORD>|00000000|<WORD>..|00000000|...
              Word Boundary
   ```

   Length of SLIST stored in SLLENG.

5. Read a record into CARDBUFF and set scanner to start scanning from beginning of that buffer.

6. Set jump exits for a special character where scanning is to begin or end, as specified by parameters BEGNCODE and ENDCODE.

7. Scan off a symbol into ACCRUBUF.

8. Branch on type of symbol contained in TYPE. Types are:

```
                5 = alphabetic string
               11 = numeric string
              BCD = BCD code of any special character
                    is its type.

                5 → 9)
               11 → 7)
            Other → 7)
```

If a BEGNCODE was specified, the program scans each symbol in turn, but no further processing occurs until TYPE = (BEGNCODE)

If an ENDCODE was specified, the program reads a new record when TYPE = (ENDCODE).

9. Check if word contained in SLIST

```
            yes → 7)
             no → 10)
```

10. Write (ACCRUBUF) + (CARDBUF) onto TEMPOUT equ 50.

11. →7)

All parameters are taken from TABLE. In the order given there, parameters are

UNITIN	lun of input file
UNITOUT	lun of output file
SUPPRESS	lun of words not to be used as keywords. Format of this file is one word per record.
LINESIZE	Maximum length, in characters, of output records.
BREAK	BCD code where an output record is to be split if record length exceeds L. If (BREAK)>$(77)_8$ the line will be broken exactly at L characters.
BEGNCODE	BCD code of character in record that will cause keyword searching to begin.
ENDCODE	BCD code of character in record that will cause keyword searching to terminate.

Operation of Scanner

The SCANNER is a three state processor that is used to scan and accumulate symbol strings from the input record stored in CARDBUFF. Strings are of three types:

1. <alphanumeric>::=<alphabetic><alphanumeric>
2. <numeric>::=<number>|<number><numeric>
3. <other>::=any other characters except space comprises a one character string

The SCANNER exits with the string in ACCRUBUF, the string length in LENGTH, and the string type in TYPE.

Directories and tables are:
ATABLE - a table of actions that point to the following
 α_1 : ACCFET - accrue character and fetch next character.
 α_2 : CHRØUT - exit from scanner with special character.
 α_3 : SKIP - Ignore current character in window.
 α_4 : ACCØUT - exit from scanner with character string.
STABLE - a table for computing the next state. States are
 S_0: initial state
 S_1: stacking an alphanumeric symbol
 S_2: stacking a numeric string
CTABLE - a table that translates a BCD character to its character class. Classes are
 0 - alphabetic
 1 - special character
 2 - decimal digit
 3 - ignore

Counters and lists are
NSP the next state pointer, used to compute the next state.
WINDOW hold character that was input at time t.
CLASS holds class of character in window.
STATE the current state indicator
ACCRUE a buffer used to hold the character string being accrued.
LENGTH the length of string in the ACCRUE buffer.
TYPE the type of string in the ACCRUE buffer. If string is a special character, then type = char.
FCHAR a pointer on the character string being scanned.

Method of operation can be described as follows: As in a finite state machine, the behavior of the scanner is determined by

$$S_t = F(S_{t-1}, I_t)$$

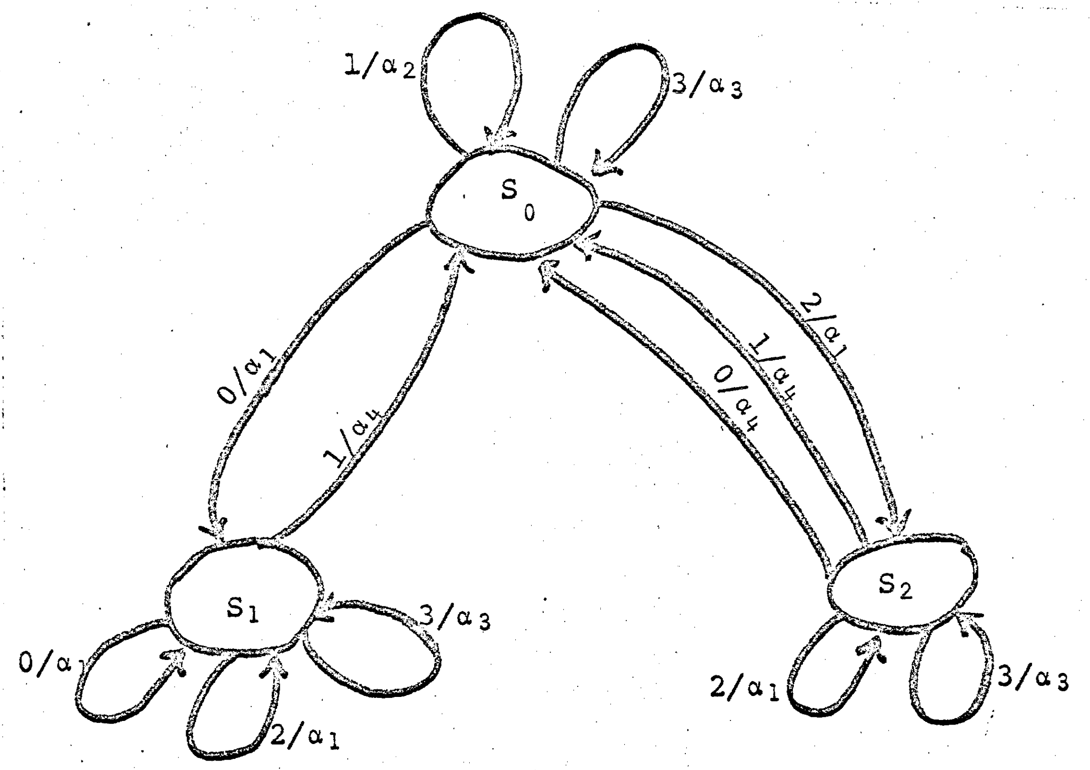
$$\alpha_t = G(S_t, I_t)$$

where S_t is the state at time = t,
 α_t is the action at time = t,
and I_t is the input symbol at time = t.

Character classes:
 0 = alphabetic
 1 = special character
 2 = number
 3 = ignore

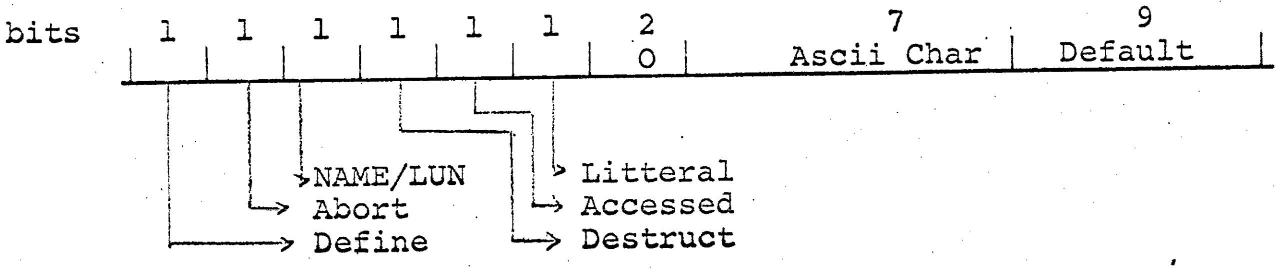
States:
 S_0 = initial state
 S_1 = stacking alphanumeric symbol
 S_2 = stacking a numeric string

Actions:
α_1 = accrue and fetch
α_2 = character out
α_3 = skip
α_4 = accrue buffer out



Operation of PARPROC

PARPROC is a parameter processor used to process the user's parameter string. Parameters are placed in a table name TABLE. Each parameter entry in TABLE contains the following information:



Define 1: Equip lun or save name if parameter not already equipped or saved.

0: Don't equip; don't save.

Abort 1: Abort if lun not equipped or if name not saved and if define = 0.

0: Don't abort.

NAME/LUN 1: Parameter is a file name. Lower nine bits of TABLE word used as a pointer to two word name in NAMETBL.

0: Parameter is a lun.

Destruct 1: Unequip lun after run.

0: Don't unequip lun.

Accessed 1: Parameter not supplied by user. Default value present in word.

0: Parameter supplied by user. Default value erased.

Litteral 1: Parameter is a litteral; don't treat it as a file.

0: Parameter is a lun or file name.

Control proceeds as follows:

1. Read a character. If CR→5)

2. Test if parameter letter
   Yes→3
   No→1

3. Look for equal sign
   No→3
   Yes→4

4. Read next characters. If alphabetic string, store in NAMETBL. If numeric string, store lower 9 bits of TABLE parameter word.

5. Process all parameters as directed by bits in TABLE parameter word.

The contents of TABLE may be changed with the TABLE creation macro, CREATE.

Operation of SORTX

SORTX, stored in *SORTX, is a modified version of *SORT. Modifications are

1. PARUNIT has been changed to default to KWOCSORT (equ 51) for its parameters.

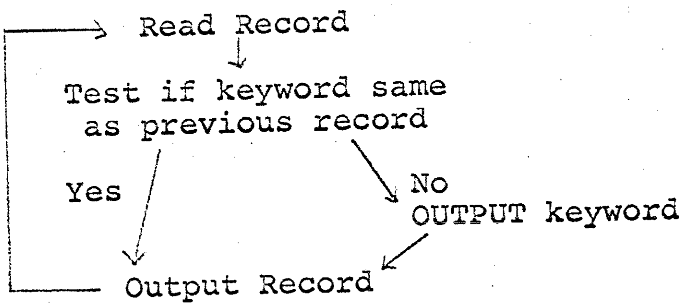
2. The program no longer writes its parameters on lun 61.

3. The exit has been changed to read in *KWOC2 as an overlay and then jump to it.

Operation of POSTKWOC

Program stored as an overlay in file *KWOC2. Reads sorted output from SORTOUT (equ 53) and TABLE created by *KWOC1 from PARAMS (equ 52).

A simple loop is used to output the sorted records:



All output is directed to UNITOUT.

The record is output by a subroutine called JUSTIFY. JUSTIFY breaks up the record into LINESIZE units and outputs it until the entire record has been written. If a BREAK character is given, JUSTIFY scans a LINESIZE record from right to left looking for the BREAK character. If such a character is found, the output record diminished appropriately. If no BREAK character is discovered, the line is equal to LINESIZE.

Each time a keyword is output to UNITOUT by POSTKWOC, that keyword is also written on KWLIST (equ 54). At the end of the program, KWLIST is copied onto UNITOUT.