PACIFIC LINGUISTICS

Series D - No.8

INFORMATION STORAGE AND RETRIEVAL: A DICTIONARY PROJECT

by

Adrianne Lang
Katharine E.W. Mather
Mary L. Rose

Department of Linguistics
Research School of Pacific Studies

THE AUSTRALIAN NATIONAL UNIVERSITY

PACIFIC LINGUISTICS is published by the *Linguistic Circle of Canberra* and consists of four series:

SERIES A - OCCASIONAL PAPERS
SERIES B - MONOGRAPHS
SERIES C - BOOKS
SERIES D - SPECIAL PUBLICATIONS.

ALL CORRESPONDENCE concerning *PACIFIC LINGUISTICS*, including orders and subscriptions, should be addressed to:

The Secretary,
PACIFIC LINGUISTICS,
Department of Linguistics,
School of Pacific Studies,
The Australian National University,

Canberra, A.C.T. 2600.
Australia.

First published 1972.

The editors are indebted to the Australian National University for help in the production of this series.

This publication was made possible by an initial grant from the Hunter Douglas Fund.

National Library of Australia Card number and ISBN 0 85883 087 6

ACKNOWLEDGEMENTS

The Enga Dictionary Project was organised and implemented while the authors were at The Australian National University: Adrianne Lang in the Department of Linguistics, Research School of Pacific Studies, Katharine E.W. Mather and Mary L. Rose in the Programming Section, Research School of Social Sciences and Pacific Studies. The Australian National University Computer Centre was used for all computing. We would like to express our appreciation to The Australian National University for having made this research possible. We would also like to acknowledge the assistance of Mrs C.A. Townsend and the Data Processing Unit of the A.N.U. in the tedious task of punching the 30,000 cards used in the project.

Adrianne Lang

Katharine E.W. Mather

Mary L. Rose

TABLE OF CONTENTS

Page

Page

1.0 DESCRIPTION OF THE DATA, by Adrianne Lang

1.1 *PURPOSE AND METHOD*

The goal of this project was to create a dictionary data file to allow the user maximum flexibility in storing and retrieving language data. For this purpose we used The Australian National University IBM 360/50. All data was coded (see also 1.4) and punched on standard 80 column cards. From these cards disk files were formed and the resulting master file functioned as a monolingual ethnographic dictionary of Enga, in which all information was with reference to the Enga entry. For each Enga entry the information was subdivided into 33 "items", each item containing a specific kind of information. Each item was assigned a specific space on the punched cards, in order to allow the computer to deal with each kind of information separately.

1.2 *SCOPE*

At present the dictionary master file contains 5545 entries in several mutually-intelligible dialects of Enga. Source materials for the Enga master file included word lists and other relevant documents published by the missions in the Enga area, as well as field data collected by A. and R. Lang[1]. In the future it is hoped to code an extensive English-Kyaka Enga Dictionary by the Australian Baptist Missionary Society (Draper n.d.). The Kyaka Enga entries will occupy entry numbers from 5545 upwards, and it will be possible either to sort the Kyaka Enga directly into the present file (using the alphabetic sort) to produce an expanded Enga Dictionary, or to use these entries as a sub-set of the main file and thus form a Kyaka Enga dictionary on which all other programs could be run separately. Further extensions could include the

---

[1] For a complete breakdown of all sources used, see Lang (forthcoming).

Lang, A. "Description of the Data". In Lang, A., Mather, K.E.W. and Rose, M.L. editors, *Information storage and retrieval: A dictionary project.* D-8:2-18. Pacific Linguistics, The Australian National University, 1972. DOI:10.15144/PL-D8.2

coding of Ipili (a language related to Enga on the sub-family level), Nete (another sub-family level language), and possibly Kewa (related to Enga on the family level). All of this data, in combination with data already present in the main file could then be used for comparative work, e.g. the reconstruction of Proto-Engan.

1.3 *DESCRIPTION OF THE ITEMS*

The items are described in the order in which they follow each other on the punched cards (cf. 2.2.5 Card Format). Each item may be referred to by the final digits of the title, i.e. 1.3.1 Entry Number, is referred to elsewhere as "Item 1". The items include both fixed and variable length data, with the fixed length data occupying items 1 through 12, and the variable length data occyping items 12 through 33.

1.3.1 Entry Number

A unique number was assigned each entry at the time it was coded. Entries are numbered consecutely in the order in which they are entered, and the entry number uniquely identifies every element on the coding sheets, the IBM cards (back-up storage), and the disk master file. At present six columns are used: the first four are the actual entry number and the final two are the card number within the entry.

1.3.2 Enga Entry

The Enga entry presented in this portion of the file is the item to which all the other information in the entry refers. The Enga entry is a word, phrase, or a bound morpheme. Phrasal entries included are those used in Bible translation, such as betésa ándenge dóko *'yeast' ("the thing that makes bread increase")* and anda maki maki katengé yuú *'city' ("place where houses stand in rows")*. The bound morphemes (one, the negative prefix, and the remainder suffixes) are included for the convenience of interested users and the entry information (items 15 through 33) is restricted for these entries. Since the hyphen sorts to the beginning in the alphabetic sort, these morphemes may be ignored if desired.

The computer keyboard and printer characters limited the transcription of Enga entries in only two ways: the tone mark followed the vowel over which it occurred, and the symbol ŋ was represented by the digraph ng. At the onset of the project (October 1968) the orthography followed the conventions agreed upon by the missions at the Enga Orthography Conference of 1966; at the June 1969 Orthography Conference it was decided to write the prenasals in Enga. This was accomplished in the master file during updating and re-creation, as the file was read onto the disk. Thus, ŋ is presently represented as nng in the master file.

Originally twenty-five columns were allotted for the Enga entry, and this number was adequate for all Enga words and phrases. However, with the prenasals written in, some entries would have exceeded 25 characters. These entries were left unchanged in the master file during creation, and the orthographic changes done later by hand.

1.3.3 Existential Verb

Every noun in Enga co-occurs with a classificatory verb which denotes the habitual state of existence for that noun. These existential verbs formally mark the noun classes and correspond to the English copula *'to be'*. The existential verbs were coded by abbreviation of the verb root.

1.3.4 Incomplete Entries

Entries with incomplete information in items 1 through 13 were coded with a question mark in this field, if the question mark was not already coded in the item field. Other entries wanted for reference were coded with an asterisk in the same space. After completion of the first stage of the project (when all incomplete entries would be checked and corrected), this field would be available for whatever other information might be desired.

1.3.5 Semantic Domain

This item contained information clasifying the main entry into semantic domains or fields. During the first stage this was done using large groupings such as kinship terms, colour terms, artifacts, verbs of motion, verbs of breaking, plants, animal names, etc. Future work will include the sub-grouping of these semantic domains.

1.3.6 Source of Entry

This item was intended to account for the source of each main entry: the user could pinpoint dialectal or idiosyncratic biases of the originator. This item also allowed direct reference to published materials from which entries were taken.

1.3.7 Source of Tone

This item provided the informant's name and the date the main entry was toned in cases where the original entry had been coded without its tone. When the original main entry was toned, the Source of Tone field was left blank, as the information was already given in item 6.

1.3.8 Notebook and Tape

For the majority of entries which were originally elicited in alphabetic order, this item was left blank. Secondary and feedback items have coded in this field the notebook where elicited; the notebook number refers to the tape used in eliciation.

1.3.9 Thesaurus

This item is to be used in the future as an extension of item 5, Semantic Domain, for the generation of an Enga thesaurus.

1.3.10 Grammatical Class

This item provided a tentative classification into conventional categories (noun, verb, determiner, etc.). The space allotted was large to allow for future sub-categorisation and flexibility. Default entries were left blank.

1.3.11 Dialect

With eight dialects in Enga, entries positively identified as to dialect were coded in this field. Retrieval of this information would produce dialect-specific word lists or dictionaries, as desired. This item also shows the phonological and morphological changes occurring in the Enga dialect change: asáné *'father'* (Laiapo dialect), apáné *'father'* (Mae dialect); asingí *'old'* (Laiapo), atíngí *'old'* (Mae).

1.3.12 Language and Source

If the main entry was a loan word, this item gave the source language and source word, when known. Marked for either Pidgin or English, it is most probable that all loan words are directly from Pidgin. This item allows for all loan words to be ignored or printed out separately (cf. Lang, forthcoming, in which loan words are marked with an asterisk).

1.3.13 English Gloss

This item contains the English gloss of the Enga entry; the number of English synonyms given was limited to only a few, since they would be redundant to native speakers of English (cf. Gleason 1955). This is the first item with variable length information, and the field allotted for English glosses and definitions is potentially limitless. Originally the longest English gloss was 120 characters, but this was increased to 160 characters to allow such kinship definitions as *'husband's male in-laws except brother-in-law (wife's male relatives of other than her own and -2 generations (male ego); husband's cognates of -1 generation)'*. All information in this item was coded by placing the head word first to facilitate the reversal of the master file into English-Enga (see 4.3).

Multiple glosses were separated by commas or semicolons for the reversal to the English-Enga index; any information in parentheses (as in the kinship definition given above) was ignored by the computer during the reversal. If alternate English glosses occurred with an entry, the initials of the original source were added following the gloss: *'to grow large (of trees)* (LDH); *to pay for magic used at marriage* (RL)*'*. Tentative selectional features were included when possible: *'handsome (of men)'*. Polysemous words (or possible homophones) were given as separate entries, in order to assign them to the appropriate semantic domains: kaná$_1$ *'shilling'*, kaná$_2$ *'moon'*, kaná$_3$ *'stone'*, kaná$_4$ *'month'*.

No sub-entries were coded (i.e. all entries were treated as main entries, cf. Mathiot 1967). This was for ease of retrieval and simplification (i.e. the lexicographer was not forced to make subjective decisions as to whether or not a given entry was the sub-entry of another entry).

1.3.14 Cross-Reference

This item was used to cross-reference the Enga entry with other entries of similar semantic domains and glosses. An example of this item is:

asáné *'father'*
cross-references: asá, apáné, áya, tagánge

Another project using this item would be the listing of all cross-referenced items into groups with other desired information on each entry. This would permit the comparison of the glosses, features, etc. of the entries in each group of cross-references. Thus, for the example above, the information desired would be listed for the members of the group:

asáné..., asá..., apáné..., áya..., takánge...

1.3.15 Class Inclusion[2]

This item was used whenever the main entry was defined in the form, *"X is a member of the class Y"*.

1.3.16 Attributive

X was defined with respect to one or more distinctive or characteristic attributes Y.

1.3.17 Function

X was defined as the means of effecting Y.

1.3.18 Contingency

X was defined with relation to a usual or necessary antecedent or concomitant Y.

1.3.19 Spatial

X was oriented spatially with respect to Y.

1.3.20 Operational

X was defined with respect to an action Y of which it is a characteristic goal or recipient.

1.3.21 Comparison

X was defined in terms of its similarity and/or contrast with Y.

1.3.22 Exemplification/Instrumental[3]

Exemplification: X was defined by citing an appropriate co-occurrent, Y.

Instrumental: X was defined as occurring by or with a specific instrument.

1.3.23 Provenience/Objective

Provenience: X was defined with respect to its source Y.

Objective: X was defined as occurring with a specific object.

---

[2]Items 15 through 30 are based on the semantic relationships given in the folk definitions by Enga informants (cf. Casagrande and Hale 1967 and Lang 1971, Appendix B).

[3]The data contained only a few examples of Exemplification (item 22) and Provenience (item 23), so that their fields were utilised for the more frequently occurring Instrumental and Objective relationships.

1.3.24 Grading

X was defined with respect to its placement in a series or spectrum that also included Y.

1.3.25 Time and Duration

X was defined as occurring at a specific time and often as being of a specific duration.

1.3.26 Explicative

X was defined as a reason for Y.

1.3.27 Subjective[4]

X was defined as the subject of Y.

1.3.28 List

X was defined as a member of a list Y.

1.3.29 Antonymy

X was defined as the negation of Y, its opposite.

1.3.30 Ostensive

X was defined ostensively (by pointing, example, etc.).

1.3.31 Method

This item contained the method employed for some activity, such as processing salt, a ceremony, or a magical spell.

1.3.32 Miscellaneous Information

This item was used for the following data: (1) informant's comments or intuitions about the main entry, i.e. *"we don't say it that way here, we say X, they talk that way in Wabag"*; (2) secondary and tertiary sources, and additional information on the source that was not placed in items 6 or 7; (3) references in published or unpublished articles to the main entry.

---

[4] The relationships given in items 22 (Instrumental), 23 (Objective), 25 (Time and Duration), 26 (Explicative) and 27 (Subjective) are not used by Casagrande and Hale, and are not definitions in the usual sense, but contain information of use for an ethnographic dictionary of Enga.

1.3.33 Illustrative Citation

This item shows the context of the main entry and gives an example of its use. Preference is given to quotations from fables, myths and stories whenever possible; the aim being to illustrate Enga ethnography, customs and beliefs whenever possible. The item includes the citation and its source; at present most of these are from Rev. Dave Hauser, who made available his Enga dictionary extracted primarily from his study of Enga fables.

After the production of an Enga concordance based on a large corpus of natural conversation (transcribed by R. Lang), as well as the published materials and texts of fables, it is planned to complete this item for all entries. The possibilities for illustrating syntax and semantics are obvious.

1.4 *CODING*

The actual coding of the data onto coding sheets, an extremely tedious task, is also the most crucial operation performed by the linguist (assuming that data collection and elicitation have been rigorous). The computer cannot check the data content, but can only make mechanical checks: if a continuation is marked, does one actually occur; are the entry and card numbers in consecutive order, etc. The problem of methodology in lexicography has largely been ignored by lexicographers, and is beyond the scope of this work; nevertheless its importance must be stressed. In performing the actual coding, notes were taken of the decisions made while coding, but these only emphasised that the methodological problems need much further work. A sample coding sheet is included in 1.7.1; the details of the card format are presented in 2.2.5.

1.5 *UTILISATION OF THE DICTIONARY*

1.5.1 The Master File

A printout of the master file after the alphabetic sort and the orthographic changes have taken place is illustrated in 1.7.2. Tones follow the vowel on which they occur, and the trigraph nng represents ŋ. The present master file is unedited and presented here only in illustration of the data elements stored. Since interrogation of the master file can be performed on a single item, a sub-item, or a combination of items, the wide range of applications of the dictionary format is apparent.

Some of the major lexicographic tasks performed by the computer are the alphabetic sort (4.1), the cross reference check (4.2), and the reversal (4.3). A description of the uses of several of the more interesting of the retrieval programs follows.

1.5.2 The Word List

This is a reduced version of the master file which contains only items 1, 2, 3, 10, 11, 13 and 14. This listing was produced in the desired format and, after editing, was ready for publication (see Lang, forthcoming). This allowed the publication of an extensive Enga Word List, which filled a much needed spot in Enga studies. Another use for this reduced file might be for a linguist interested in a small language population or in a less intensive study, who could utilise this format to produce a computerised word list, rather than an extensive monolingual ethnographic dictionary. The Word List printout is illustrated in 1.7.3.

1.5.3 The Reversal

The Reversal program (cf. 4.3) was used to create an English-Enga index. Again, a reduced format was used, with item 13 as the entry, followed by items 2, 3, 10 and 11; this was utilised, after editing, in Lang (forthcoming). The two main advantages in the computerised reversal were firstly, that it provided a check that all Enga items did appear in the English Index, and secondly, that the time saving was considerable (in contrast to the normal methods of generating a bilingual index by hand) - the reversal was accomplished by the computer in twelve minutes running time. 1.7.4 illustrates the Reversal.

1.5.4 Like Words

This program gives a printout of all orthographically-like pairs, triplets, etc. This results in multiple entries for the polysemous entries, but it also produces all minimal pairs. The printout is illustrated in 1.7.5.

1.5.5 Tone Patterns

This program produces groups of all similar tone patterns, subgrouped by syllable length. This was used for tone testing and to determine minimal tonal pairs. One sub-grouping of the Tone Patterns is illustrated in 1.7.6.

1.5.6 Cross Reference Check

This program provided an accuracy check on item 14. The program listed these items in three ways: (1) Enga feedback entries which did not occur as main entries but only as item 14 to some closely related entry (thus allowing these feedback entries to be added to the master file); (2) main entry A, which occurred as item 14 with entry B, but B was not given as an item 14 with the entry A; and (3) main entry A had given B as item 14, but, entry B did not have A as item 14. The print-out is illustrated in 1.7.7.

1.5.7 Descriptive Statistics

A survey program was used for statistical correlations on the hypothesis that certain grammatical classes would usually have certain types of folk definitions associated with them (i.e. that nouns are usually defined by item 15 (class inclusion), then item 16 (attributive); cf. Casagrande and Hale 1967). Additional work on this would include retrieving the items in combinations including the sub-categorisations.

1.5.8 Incomplete Entries

Various retrievals were used to produce printouts of incomplete entries (item 4 had been marked) and those entries with blank fields: i.e. untoned main entries, or those with no English gloss, no existential verb, no grammatical classification. These printouts were immediately available for elicitation during later field work.

1.6 *UPDATING AND MAINTENANCE*

Updating was originally accomplished by the re-creation of the entire master file. As a time and effort saving device, the present updating program (cf. 2.3) was created. The entries which have items to be corrected or added (to date there have been no deletions to the file) are coded in the usual manner, with only the necessary information to be corrected or added being coded. Cards are punched from the coding sheets, sorted, and then the updating program processes them.

The advantages of the updating are that it keeps the master file corrected and current, and the use of the computer produces far fewer errors than would occur with a human lexicographer-updater. The other point is that the new, updated information is written over the original data, so that the original data is no longer present in the master file; this can also be a disadvantage, since the erased data would be difficult to trace. Security is maintained by keeping the dated master file printouts, as well as the coding sheets used for the updating, and this would allow for the comparison between the original and updated data, if desired.

1.7 PRINTOUTS

1.7.1 Sample Coding Sheet

```
186401KUNI' EDA'KI                E   55N071068             1718
186402009                               RIVER CARRYING DEBRIS & TOPSOIL
186403KUNI' I'PA
186405KUNI' I'PA,KONE' PI'PAE EPEGE'
186414KULI LO'GO' LA'O IPATA'MO KADAO' DO'KO - YANAEPI ITA'PI PITAKA' DU'P *
186414A LO'GO' LA'O SOO' EPEGE'
```

5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80

1.7.2 Master Printout

```
AI'YE                    ?  001        53 CB                                                                                   213
     2DECORATION(HEAD)OF CASSOWARY FEATHERS
     4YATI' ME'NDE'                    5LA'IMA I'TI' O'NGO IKI' -TAMBUA'KA,PUMBU'TI',PIASO NAPENGE(?)                    14MA'L
   I LA'LANYA GI'I DOKO'PA AKA'LI PINGI'.E'NDA NA'PINGI-AKA'LI DU'PAMEE'NDA DU'PA SAKAMAII'NGI,E'NDA DU'PAME SI'NGI.MEE' LA
   'IMA PYA'PALA I'TI'YAKO'PAE SILYA'MO GI'I DOKO'PA' PINGI'.               16AKA'LI DU'PAME WASINGI.BAA'  TA'NGEME BAANYA' KA
   IMINI'NGIPI TAKA'NGEPI.LA'IMA YAKA'NE' MENDA'I=_ MENDA'I,LA'IMA ANDA'KE=LA'PO'
AIYOKO KOO' MINA'O TE'E      0080001      54 LSTOTIK195                                                                        211
     2PAYMENT FOR DEATH ON BRIDGE
    20C.F. STOTIK 165
AIYOKO'O                 K  001        21 R 9,56      N061368    1332                                                          212
     2NAME-MAN
     5KOPOA'KO TA'NGE DO'KO,WA'MBA MEMBA'A INJO'O KA'LYANYA MA'SI'A'
AJI'KI' AJI'KI' PIGI'        0080001 L        LH                                                                               221
     2TO COAX/CAJOLE,TO PERSUADE
     3ANDI'KI' PINGI',ANDI'KI' ANDI'KI' PINGI',ANJI'KI' PINGI'                     7ENDAKA'LI ME'NDE'ME BAA' NEEPI' PINJU'PI
    MUNI'PI MEE' DINA' LA'O ANDIKI ANDIKI PII' LAMAII'NGI.MA'NA E'PE' ME'NDE' AUU' PI'PAE. DINA' LA'O TEE LA'O NYI'NGI = MA
   'NA E'PE'(WAA' NYI'NGI = MA'NA KOO') DI'I PII' PITAKA' DU'PA=_ PI'PAE:ANDIKI ANDIKI PYO'O PISI'NGI = E'PE'          18PII'
    KOO' PE' TALA LENGE',PII' PU'U LA'O LENGE'
AKA'IPU WA'I             PO 001        45 LH          LB                                                                       230
     2MAGIC(WHITE),LOVE POTION
     7"AKA'LI ME'NDE' KUMAPE'NGE INJE'TALA YAL'NA' KOO' ME'NDE' NA'PALA KUMATA'MO KANDAO' GI'I DOKO'PA' AKA'LI BAA' DO'PA
   PI'PULI MA'SINGI ME'NDE' NYETA LYANATAMI ENDAKA'LI DU'PAME.LYANYE'PALA PITUU' MENA' ME'NDE' PI'NGI.MENA' PYA'PALA PITUU'
    DOKO'PA' AKA'LI PI'PULI DO'KOME DEE' BAANYA' I'TA' MENDA'PU TOKA'PALA KETAMBU PE'TALA DOKO'NYA' DA'PA MINA'O KAKOPE'TAI
   A I'TA' LYANAE DU'PA KAKA'TALA PITUU' DOKO'NYA' MENA' YAWENGE'. MENA' AYO'MBA TAMOPA MENA' KONJE'PAE MENDATU'PA YAWENGE'
   .YAWA'TALA DOKO'PA' MENA' YAO LOPANYE'TAE DOKO'NYA' TA'NUPI AKAI'PU WAIPI POKENGE'- DO'PA POKENGE'."TIMO'NGO DU'PAME MAI
   YO'O YAWENGE' - TIMO'NGO DU'PA MENA' TUNDUMA' IKI' NENA' LA'TALA.PO'O PI'NGI AKA'LI DO'KO YO'LE' MAII'NGI- MENAPI' MUNI'
   PI MAII'NGI
AKAI'PU                  P  001        08 LH                                                                                   229
     2CORDYLINE
     3TO'PYA,YAKAIPU                        4TA'TA DU'PA:MONGALO,YUKU,KOLOKOLA,AMBA'KO',AMBANO,KA'LYA',KAKU,TAMO,LAIKI,KEKELO
   NDE.                               6A'IPUMA PINGIMA'PE':II' KA'ITA DO'KO SA'NGA' PYAO' PINGI';E'LYA KAENGE' DOKO'NYA'
   PINGI'.NDU'PA TA'NGEME (WANA'KU,E'NDA)_ AUU' KAYA'PALA PINGI'.E'NDA AKA'LI NYI'NGI DU'PAME E'NDA WA'NE' MANDENGE' DU'PAM
   E MALI' LA'LANYA GI'I DOKO'PA'_ PINGI'.WA'NE' A'NE DU'PAME KO'TEAKA LA'O _ PINGI'.ENDAKA'LI DU'PAME _ POKENGE.EE,A'NDA'
    PATAKI'SA,EE' KAME' DU'PANYA WASINGI.AKA'LI BAA' TA'NGEME BAANYA' _ POKENGE';BAANYA' KAIMINI'NGI NA'KATATA'MO' KANDAO'
   DO'KO BAAME MAII'NGI._ MEE' A'NDALA NA'ENGE-ENDAKA'LI WASINGI O'NGO IKI' A'NDENGE.                          16AKA'LI M
   E'NDE' _ NA'PITA'MO' KANDAO' DO'KO BAA' KYAKENGE'.
AKAITA'                     005                LF                                                                              231
     2THESE,THIS DIRECTION
     3AKOITA,ATU'PA,DAKAITA',DATU'PA,DOKAITA'                                      8_=KOPETESA,TOMBIAME;OMOKAITA=KASSAPU,WAK
   IME                               18OMOKAITA                           20ONOTENA=WAKALE
AKA'LI                   K  001        20 LF                                   5                                               232
     2MAN
     5ANDA'KE,ANGAE'TI KATENGE',YAMBA'LE PINGI',MATA'PU PINGI',UAA' MANDENGE',A'NDU NA'KATENGE,PONGO' MANDENGE'           6
   AKA'LI DU'PAME:KAITI'NI' PENGE',A'NDA' PINGI',KALA'I PINGI',EE' POKENGE',LUU' PALENGE',KA'ITA PAENGE',NEE' NENGE',MO'NA
   PALENGE',PII' LENGE',ENDA'KI' TO'KO PINGI,YANDA' TU'KUPI DU'PA MINI'NGI,YANDATE' MINI'NGI,KONALI MINI'NGI,MENAPI' YA'NAP
   I MINI'NGI,PI'NJU' SE'TENGE,AKA'LYANDA PALENGE',SANGA'I SE'TENGE,MA'LI LENGE',TE'E PINGI',YANDA' PI'NGI,SA'A PI'NGI,A'NG
   A LYINGI'
AKA'LI ANDA'KE           K  009        20 BT 210                               15                                              233
     2MAN - IMPORTANT,MAN - LARGE IN SIZE
     5AKA'LI A'NGI' ME'NDE',AKA'LI A'NDAPAE ME'NDE',AKA'LI LYANGA' ME'NDE'      20KAMO'NGO = KU'KILI WAKA'
```

1.7.3 Word List Printout

NE'NE PT 1
INSECT(GENERIC),ARTHROPODA(ALL MEMBERS) AND SNAILS

NE'NE ANDA S 1
MAGGOT

NENE PINGI' 8
TO PLAY
MA'LE LENGE',MA'LE PINGI'

NENE' PINGI' B
TO SHOW DISGUST

NE'NE' PINGI' (L) B
TO HUM/BUZZ

NE'NGE K 1
HORNS/TUSK,TUSK,TEETH

NENGE' 2
TO EAT/BITE/GRAZE,TO CONSUME

NENGE' S 1
TOP OF MOUNTAIN RANGE

NE'NGE KA'ITA S 9
MOUTH

NE'NGE PI'NGI B
TO SHARPEN(WITH FILE OR STONE)
NA'NGA PI'NGI,NA'NGA SI'NGI

NE'NGELYA'PI K 1
FACE
LE'NGELYA'PI

NEPAEPA PL 1
SHOULDER-MIDDLE OF
NEPI'PA'

NEPE'NGE 2
TO DISCARD/THROW AWAY/REMOVE
ANGO'NGE,NEMBE'NGE

NEPI'PA' K 1
NECK(BELOW BASE OF)
NEPAEPA

NEPO PINGI' B
TO PLAY AT FIGHTING
KOPIO NENE PINGI'

NE'TE' 10
EDGE,BORDER
LIMBA,LIMBASA,MATENGE,NETE'SA

NE'TE' KI'PA' KI'PA' 3
SURROUNDING AREA
NETE'SA

NETE'SA 10
BORDER,EDGE
KAPAKATA,KILIKI'LI',LE'MBA,LYA'MBA',MA'TE,MATENGE,MATESA,NEMBANGE',NE'TE',NE'TE' KI'PA' KI'PA'

1.7.4 Reversal Printout

BLOOD/PLASMA	TANJE'NA'		001		4060
BLOW	POO' LENGE'		008004		3616
BLOW FIRE	POO' PINGI'	(M)	0080001M		3618
BLOW NOSE	MA'NJO' NNGAI' LENGE'		0080004		2538
BLUE	SA'KAPAE	(M)	003 M		5058
BLUE-GRAY ?	LIMBI LIMBI PIAPE		003		2202
BLUE/PURPLE/AQUA	WENE' PYA'PAE		020		4731
BLUNT	LOLA'TA		003		2281
BOARD	*PALA'NNGA		001	P- PLANK	3304
BOAST LOUDLY	MAKU' LENGE'		0080004		2574
BOAST/BE FORWARD	YANDAI'TA' LENGE'		0080004		4770
BOAT	*SI'PI		001	P- SIP	3926
BODY	YANO'NGE'		001		4922
BODY	YANU'		001		4925
BODY	YANU'NGI'		001		4929
BODY	YO'NGE'		001		4976
BOG	MANDAU'WA		001		2489
BOG	TA'KE'		001		4072
BOIL	AMU'NGI'	(M)	001 M		346

1.7.5 Like Words Printout

31191	NNGILINNGALI LENGE'	TO WRANGLE
31181	NNGILINNGALI LENGE'	TO GROWL/RUMBLE(OF STOMACH)
30741	NO'LE	NAME-MAN'S:NOLE(RUTTLIOGE)
30751	NOLE'	MARK - SMALL
30871	NUU' PINGI'	NETBAG - TO MAKE A
30861	NUU' PINGI'	TO SWELL
31001	NYOKO' NYI'NGI	TO TAKE/PULL BACK
30991	NYOKO' NYI'NGI	TO TAKE/PULL BACK
31031	NYOKONYI'NGI	TO TAKE SELECTIVELY
31021	NYOKONYI'NGI	TO ORAW BACK/OUT
31601	O'LYA'	NUT
31591	O'LYA'	MOSQUITO
31631	O'MO	THAT
31641	O'MO'	OVER THERE(FURTHER AWAY)
31731	O'NYA'	
31721	O'NYA'	YAM
31711	O'NYA'	NAME-MAN'S
31791	O'PA	THUS
31781	O'PA	SWEET POTATO
31801	OPA'KA	TREE
31811	OPA'KA'	VEGETABLE: SPINACH
32271	PAENGE	THIGH
32261	PAENGE'	TO WALK AROUNO,TO GO ALONG,TO FLY(OF BIROS)
32421	PA'I	THIGH
32431	PAI'	TREE-CHESTNUT(CASTANOPSIS ACUMINATISSIMA)
32571	PA'KA	FEAR
32601	PAKA'	FORKED (POST/TREE/FINGER)
32631	PA'KA'	VERY,PLENTY
32611	PAKA' PI'NGI	TO BRACE (BANANA) TREE
32591	PA'KA PINGI'	AWESOME
32781	PAKENGE'	TO BE AFRAID,TO FEAR
32771	PA'KENGE'	TO TEAR,TO CLAW
32861	PA'KINGI	NAME-MAN'S
32851	PAKINGI'	TO TAKE APART(?)
32841	PAKI'NGI'	TO BREAK OFF
32881	PA'KO	SHIRKER (LIT. "DOG THAT DOESN'T HUNT GAME MAMMALS")
32891	PA'KO'	DOG THAT DOESN'T HUNT GAME MAMMALS
32951	PA'KUNGI	TREE SLIPS DOWN
32941	PA'KUNGI	TO CHANGE ONE'S MIND
32931	PA'KUNGI	TO ACQUIT (IN LITIGATION)
32921	PA'KUNGI	TO TRAP LIVE

1.7.6 Tone Patterns Printout

TONE ON 1ST	
A'KI	WHAT
A'KO	MIST
A'LA	BRACELET, ANKLET (PLAITED)
A'MU	PIG CALL, TO CALL PIGS
A'NA	DOWN THERE
A'NE	BOY, HUMAN MALE
A'OA	SPINACH - NATIVE (CARDAMINE)
A'PA	LIKE THIS
A'SA	HERE (MOTION TOWARDS SPEAKER)
A'ENE	IRON
A'INA	SWEET POTATO (GENERIC)
A'LUA	FROG
A'LYA	UP HERE (CLOSE BY)
A'LYO	NAME - RIVER
A'MBO	MANY, A GROUP
A'MUU	PIG CALL
A'NDA	LARGE
A'NOU	BREAST, MILK
A'NGA	PANDANUS TREE, PANDANUS NUT
A'NGE	THIS (AT HAND)
A'UKI	TREE
A'ANGI	?
A'IMBU	GRASSHOPPER
A'KITA	WHAT
A'SULI	
A'TUKU	TERM OF ADDRESS - TO MALES
A'IPUMA	WHY
A'IPUSA	NAME - PLACE
A'NDATE	ONE HUNDRED
A'NGALE	NAME - MAN'S
A'IPYAMO	WHY
A'NDAPAE	WIDE (IN GIRTH)
A'NDENGE	TO INCREASE, TO GROW LARGE
A'NDENGE	TO INCREASE, TO GROW
A'NGENYA	HERE, AT THIS PLACE OR TIME
A'IPLYAPE	HOW
A'IPUMOSA	WHY
A'IPYANYA	WHY
A'ISAPELA	EIGHT PELLA, EIGHT
A'MBELYAO	ACCIDENTALLY, UNKNOWINGLY
A'NDUINGI	SHOOT, SUCKER
A'NENGAPU	HANDCUFFS
A'NGEWANE	TREE - LIGHT WOOD
A'MBIMANGE	THISTLE POD - EDIBLE

1.7.7 Cross References Printout

5001	BA'LO PALENGE'	IS NOT A SYN OF LANGA LENGE'	AND SHOULD BE
53891	BANA	SHOULD HAVE KOPAO' LI'NGI	AS A SYN
53891	BANA	IS NOT A SYN OF KOPA LINGI	AND SHOULD BE
5061	BASAKE'SA MA'NDI	IS NOT A SYN OF GULI' MA'NDI	AND SHOULD BE
5101	BA'TO'	SHOULD HAVE WI'LIMBATO	AS A SYN
5101	BA'TO'	IS NOT A SYN OF WILIMBATO	AND SHOULD BE
SYN FOR 53181	TAMO'PAE	= BEE PYA'PAE	BUT SYN NOT DIC WORD
SYN FOR 54151	EE PYA'PAE	= BEE PYA'PAE	BUT SYN NOT DIC WORD
SYN FOR 4331	ATI'NGI'	= BEE PYA'PAE	BUT SYN NOT DIC WORD
5151	BEE' PYA'PAE	IS NOT A SYN OF ATI'NGI'	AND SHOULD BE
5151	BEE' PYA'PAE	IS NOT A SYN OF EE PYA'PAE	AND SHOULD BE
5151	BEE' PYA'PAE	IS NOT A SYN OF TAMO'PAE	AND SHOULD BE
5191	BETA' PI'NGI	SHOULD HAVE KEPA' TI'NGI	AS A SYN
5191	BETA' PI'NGI	IS NOT A SYN OF KEPA' TINGI	AND SHOULD BE
SYN FOR 6121	OIL IDALO' LEGE' ME'DE'	= BETO'	BUT SYN NOT DIC WORD
SYN FOR 40711	TA'KE	= BIAKA	BUT SYN NOT DIC WORD
5231	BIA'KA	IS NOT A SYN OF TA'KE	AND SHOULD BE
5271	BI'SA	SHOULD HAVE MUNDU'MA'	AS A SYN
5331	BO'I	SHOULD HAVE KINDU'TA	AS A SYN
5331	BO'I	IS NOT A SYN OF KINDUTA	AND SHOULD BE
5331	BO'I	IS NOT A SYN OF BU'I	AND SHOULD BE
5331	BO'I	IS NOT A SYN OF KINDUTA	AND SHOULD BE
5411	BO'ME	IS NOT A SYN OF KU'MBU PINGI'	AND SHOULD BE
SYN FOR 52201	KOPIO NENE PINGI'	= BOO' MA'LE PINGI	BUT SYN NOT DIC WORD
5441	BOO' MA'LE PINGI'	IS NOT A SYN OF KOPIO NENE PINGI'	AND SHOULD BE
5511	BU'I	SHOULD HAVE KINDU'TA	AS A SYN
5511	BU'I	IS NOT A SYN OF BO'I	AND SHOULD BE
5511	BU'I	IS NOT A SYN OF KINDUTA	AND SHOULD BE
5561	BULU'	IS NOT A SYN OF NNGOLO	AND SHOULD BE
SYN FOR 31221	NNGOLO	= BULU' B	BUT SYN NOT DIC WORD
5731	DAA' LA'TAE	SHOULD HAVE NEE' NYE'TAE	AS A SYN
5731	DAA' LA'TAE	IS NOT A SYN OF NEE NYE'TAE	ANO SHOULD BE
5791	DAKAITA'	SHOULD HAVE AKAITA'	AS A SYN
5791	DAKAITA'	IS NOT A SYN OF AKAITA	AND SHOULD BE
5811	DALA'	IS NOT A SYN OF DALANYA'	AND SHOULD BE
5811	DALA'	IS NOT A SYN OF DALASA'	AND SHOULD BE
SYN FOR 5B11	DALA'	= DALANYA'	BUT SYN NOT DIC WORD
SYN FOR 5811	DALA'	= DALASA'	BUT SYN NOT DIC WORD
5851	DA'MA'	IS NOT A SYN OF AMOTENA	AND SHOULD BE
53961	DAMA'SA	IS NOT A SYN OF AMOTENA	ANO SHOULD BE
53971	DAMATE'NA'	IS NOT A SYN OF AMATE'NA'	AND SHOULD BE
53951	DAMBI'SA	SHOULD HAVE AMBISA'	AS A SYN
SYN FOR 41381	TA'PA' PINGI'	= DEPA'	BUT SYN NOT DIC WORD
SYN FOR 23531	LYA'A LENGE'	= DEPA' LENGE'	BUT SYN NOT DIC WORD
6001	DEPA'(NOEPA') LENGE'	IS NOT A SYN OF LYA'A LENGE'	AND SHOULD BE
6001	DEPA'(NDEPA') LENGE'	IS NOT A SYN OF TA'PA' PINGI'	AND SHOULD BE
SYN FOR 6101	OII' TUMBENGE	= OII' KAMBENGE	BUT SYN NOT DIC WORD
SYN FOR 6091	OII' TA'MBENGE	= OII' KAMBENGE	BUT SYN NOT DIC WORD
6051	OII' KA'MBENGE	IS NOT A SYN OF OII' TA'MBENGE	AND SHOULO BE
6051	OII' KA'MBENGE	IS NOT A SYN OF OII' TUMBENGE	AND SHOULD BE
6091	OII' TA'MBENGE	SHOULD HAVE OII' KA'MBENGE	AS A SYN
6091	OII' TA'MBENGE	IS NOT A SYN OF OII' KAMBENGE	AND SHOULD BE
6101	OII' TUMBENGE	SHOULD HAVE OII' KA'MBENGE	AS A SYN
6101	OII' TUMBENGE	IS NOT A SYN OF OII' KAMBENGE	AND SHOULD BE
6121	DILIDALO' LEGE' ME'DE'	IS NOT A SYN OF BETO'	ANO SHOULD BE
6121	DILIDALO' LEGE' ME'DE'	IS NOT A SYN OF KEAU LENGE' ME'NDE'	AND SHOULD BE
SYN FOR 34231	PE'ENGE	= DI'NGI	BUT SYN NOT DIC WORD
SYN FOR 47421	WININDO'A	= DOA	BUT SYN NOT DIC WORD
6191	DO'A	IS NOT A SYN OF PYAKA'NAKALI	AND SHOULD BE
6191	DO'A	IS NOT A SYN OF WININDO'A	AND SHOULD BE

2.0 FILE DESIGN AND CREATION, by Katharine E.W. Mather

2.1 *DISK FILE STRUCTURE*

Each entry consisted of the Enga word, a number of constant elements (items 1.3.3 through 1.3.12), the English gloss (1.3.13), and up to nineteen variable length items (items 1.3.14 through 1.3.33). Not all of the items existed for each entry. The information pertaining to each dictionary entry was punched on cards for input to the computer. The cards were edited and the information transferred to disk files. The structure of the files was designed to minimise the storage required and the access time during retrieval of information.

2.1.1 Variable Length Data

For ease and speed of retrieval, direct access was used. Since direct access files on the 360/50 must contain fixed length records, nineteen files (each having a different record length) were used to store the variable length items and the English gloss. The record sizes ranged from 40 to 760 characters, in multiples of 40 characters. A file containing records 800 characters long was also established, but has not yet been required. Each item was stored in the shortest record length possible and assigned to the file which contained records of that length. The record length file format is shown in 2.1.7.

2.1.2 Fixed Length Data

The fixed length information relating to each Enga item was stored in a master file, which contained one record per Enga word. Each record in the master file contained an Enga item, the related fixed length information, and the addresses of the storage locations of the English gloss and any items associated with the Enga item. The address area for items which did not exist was set to zero. The master file record format is shown in 2.1.6.

Since the dictionary file was to be sorted, space was left in the master file record format for the address of the next word in the sorted order. After sorting, these addresses were written into the file (see 4.1). Thus, the file could be accessed both sequentially and in alphabetical order.

2.1.3 Record Addresses

Each entry was given a unique sequential identification number which served as its address in the master file. Records in the other nineteen files were stored sequentially as they occurred. At the beginning of each run, control cards were read, which contained counts of the number of records stored in each of the files. These counts were updated during the run, the new record count serving as the address for the new record. The control cards for the succeeding run were punched at the end of processing.

2.1.4 Record Retrieval

This organisation of the file made it possible to access directly any part of the information associated with each dictionary entry. During nearly all retrieval operations, the master only needed to be read for all entries. Only after an entry had satisfied the retrieval criteria were any associated items required for the printout read from the relevant files. Thus unnecessary data transmission was avoided.

2.1.5 File Sizes

The lengths of the twenty files used were as follows. File number seven was the master file. The other files contained variable length information relating to the main Enga entries.

File No.	*No. of characters*	*No. of words*
7	265	
13	40	10
14	80	20
15	120	30
16	160	40
17	200	50
18	240	60
19	280	70
20	320	80
21	360	90

(continued)

File No.	*No. of characters*	*No. of words*
22	400	100
23	440	110
24	480	120
25	520	130
26	560	140
27	600	150
28	640	160
29	660	170
30	720	180
31	760	190

2.1.6 Master File Record Format

Character positions	*No. of characters*	*Contents*	
1-5	5	Word No.	
6	1	Card No. (=1)	
7-11	5	Address of next word	
12-36	25	Enga word	
37-38	2	Verb for Noun (exist. v.)	
39	1	?	
40-43	4	Codes (semantic domain)	
44-53	10	Tone code	
54-83	30	Source	
84-93	10	Thesaurus	
94-100	7	Code-Pt. of speech	
101-102	2	Dialect	
103-125	23	Loan word	
126-127	2	File No.	02 English
128-132	5	Address	
133-134	2	File No.	03 Synonyms
135-139	5	Address	
140-141	2	File No.	04 Class inclusion
142-146	5	Address	

(continued)

Character positions	*No. of characters*	*Contents*		
147-148	2	File No.	05	Attributive
149-153	5	Address		
154-155	2	File No.	06	Function
156-160	5	Address		
161-162	2	File No.	07	Contingency
163-167	5	Address		
168-169	2	File No.	08	Spatial
170-174	5	Address		
175-176	2	File No.	09	Operational
177-181	5	Address		
182-183	2	File No.	10	Comparison
184-188	5	Address		
189-190	2	File No.	11	Exemplification
191-195	5	Address		
196-197	2	File No.	12	Provenience
198-202	5	Address		
202-204	2	File No.	13	Grading
205-209	5	Address		
210-211	2	File No.	14	Time when
212-216	5	Address		
217-218	2	File No.	15	Explicative (why)
219-223	5	Address		
224-225	2	File No.	16	Constituent
226-230	5	Address		
231-232	2	File No.	17	Circularity
233-237	5	Address		
238-239	2	File No.	18	Antonyms, Ost. Def.
240-244	5	Address		
245-246	2	File No.	19	How, When do it
247-251	5	Address		
252-253	2	File No.	20	Comments of inform.
254-258	5	Address		
259-260	2	File No.	21	Quotation & source
261-265	5	Address		

2.1.7 Record Length File Format

Character position	*No. of characters*	*Contents*
1-6	6	Word number
7-8	2	Card number
9+		Record

This format was used for all records stored in the recorded length files, except those containing an English gloss. In these, positions 9-16 were left blank. It was originally intended to sort the file on the English gloss and store the address of the next record in sort order within the record (in characters 9-16), as was done with the Enga sort (4.1). However, this was not possible, since some items had more than one English gloss. A special program had to be written for the English sort and English-Enga print (see 4.3 below), and the space left in the English gloss records was not used.

2.2 *DISK FILE CREATION*

2.2.1 Card Identification

Twenty-one card types were used; the card format is shown in 2.2.5. The first six columns of each card contained the item identification number. Columns 7 and 8 contained the card numbers which defined the nature of the information on the card.

2.2.2 Contents of Card Types

Card types 1 and 2 contained the Enga item and all the fixed length information (items 1-12). The English gloss (item 13) started in column 41 of card 2, continuing onto one or more cards, if necessary. Card types 3 to 21 contained the variable length items (items 14-33).

2.2.3 Continuation

Card types 2 through 21 could have up to ten continuation cards, numbered 1 to 10 in columns 79-80. (The data when coded required at most ten continuation cards. The number could be increased to accommodate longer character strings.) An asterisk in column 80 of the first card of a card type indicated that a continuation card should follow. The last card of the card type was blank in columns 79-80.

2.2.4 Input Requirements

The only requirements during input were that the cards for each entry be in ascending numerical order and that card types 1 and 2 must exist. The entries themselves did not need to be ordered, since the word number served as the disk file address. Any number of entries could be entered in the master file during each run.

2.2.5 Card Input Format

Card No.: Columns	*Item No.*	*Content*
All cards: 1-6	1	Entry number
All cards: 7-8		Card number
1: 9-33	2	Enga entry
1: 34-35	3	Existential verb
1: 36	4	Incomplete items
1: 37-40	5	Semantic domain
1: 41-50	6	Source of entry
1: 51-60	8	Notebook and tape
1: 61-70	7	Source of tone
1: 71-80	9	Thesaurus
2: 9-15	10	Grammatical class
2: 16-17	11	Dialect
2: 18-40	12	Language & source (if loan word)
2: 41-76	13	English gloss
All other cards: 80		Continuation asterisk *, numerals
3:	14	Cross References
4:	15	Definitions types: class inclusive
5:	16	Attributive
6:	17	Function
7:	18	Contingency
8:	19	Spatial
9:	20	Operational
10:	21	Comparison
11:	22	Exemplification & Instrumental
12:	23	Provenience & Objective
13:	24	Grading
14:	25	Time & Duration
15:	26	Explicative

(continued)

Card No.: Columns	*Item No.*	*Contents*
16:	27	Subjective
17:	28	List
18: 9-40	29	Antonyms
18: 41-76	30	Ostensive
19:	31	'How do they do it'
20:	32	Miscellaneous Information
21:	33	Illustrative Quotation/ Citation

2.2.6 Description of Creation Program

At the beginning of each run, a number of initialisation procedures were carried out, which included reading and printing control cards containing counts of the number of records stored in each of the record length files and counts of the number of records for which space had been allowed in each file.

The data cards were read one at a time and edited for sequence within each dictionary item. The fixed length information on cards 1 and 2 was stored as it was read in to be written later to the master file. Numeric items were checked for numeric validity and the orthography of the main Enga item was changed, unless the change made the item more than twenty-five characters long (see 1.3.2).

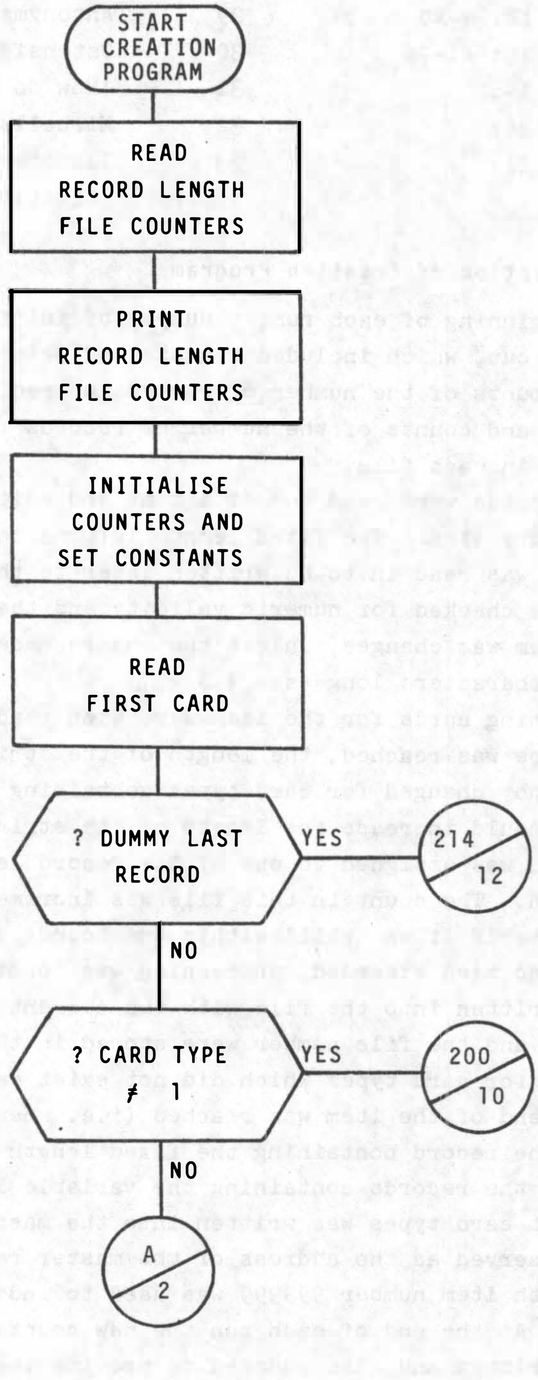
The remaining cards for the item were then read. When the end of each card type was reached, the length of the string was checked and the orthography changed for card types containing Enga (types 3-21); this change could increase the length of the string (record).

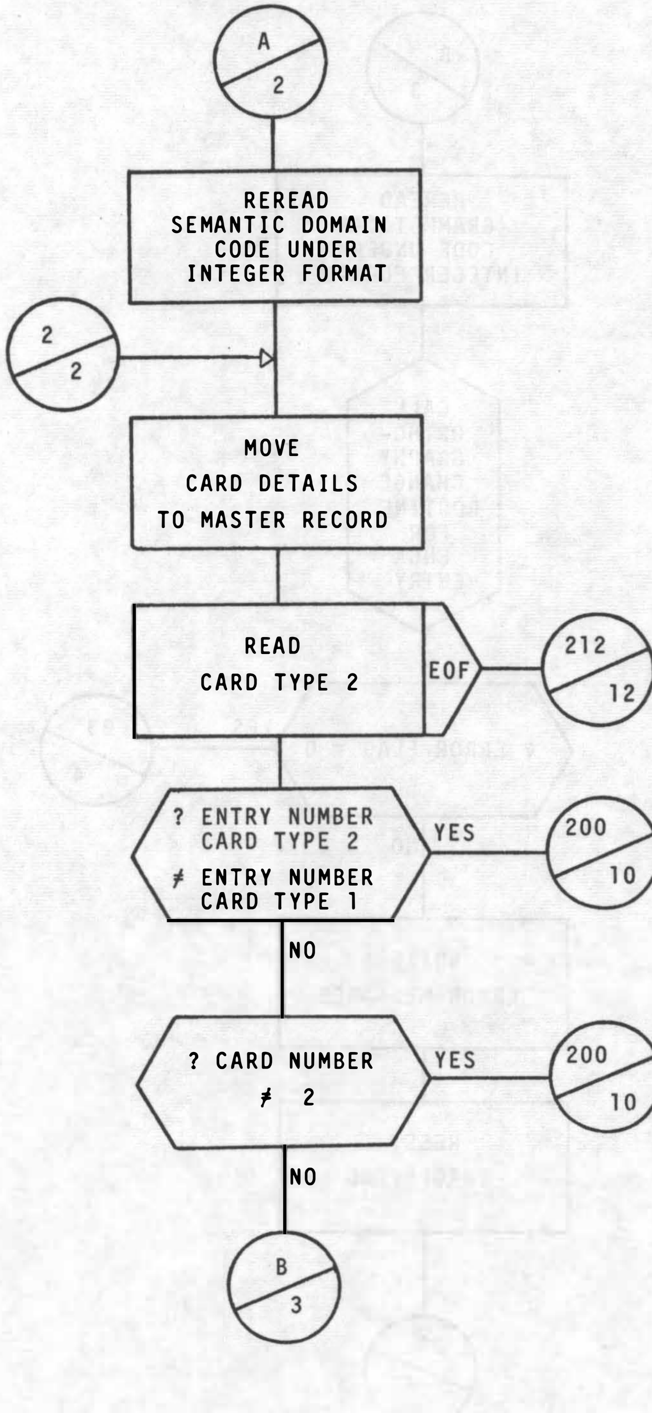
The record was assigned to one of the record length files according to its length. The count in this file was incremented by one and checked to see if it was still within the bounds of the file area. If the bounds had been exceeded, processing was terminated. Otherwise the record was written into the file with the current count as its address; this address and the file number were stored in the master record. The address area for card types which did not exist was set to zero.

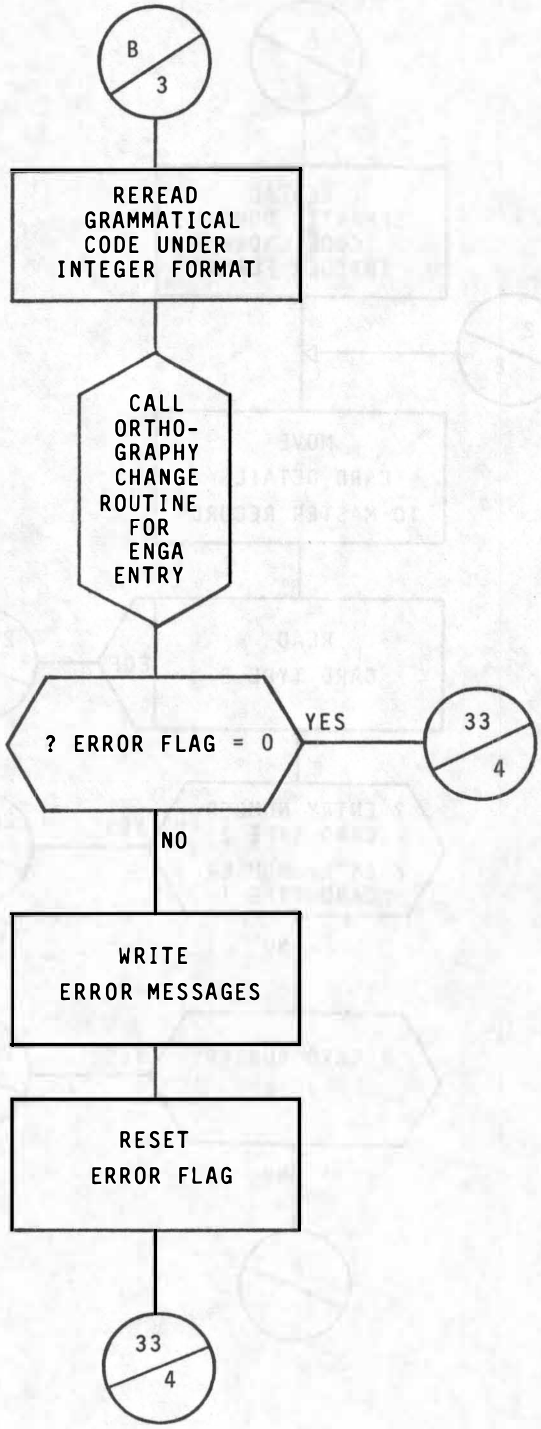
When the end of the item was reached (i.e. when another card type 1 was read), the record containing the fixed length information and the addresses of the records containing the variable length information of the different card types was written into the master file. The number of the item served as the address of the master record.

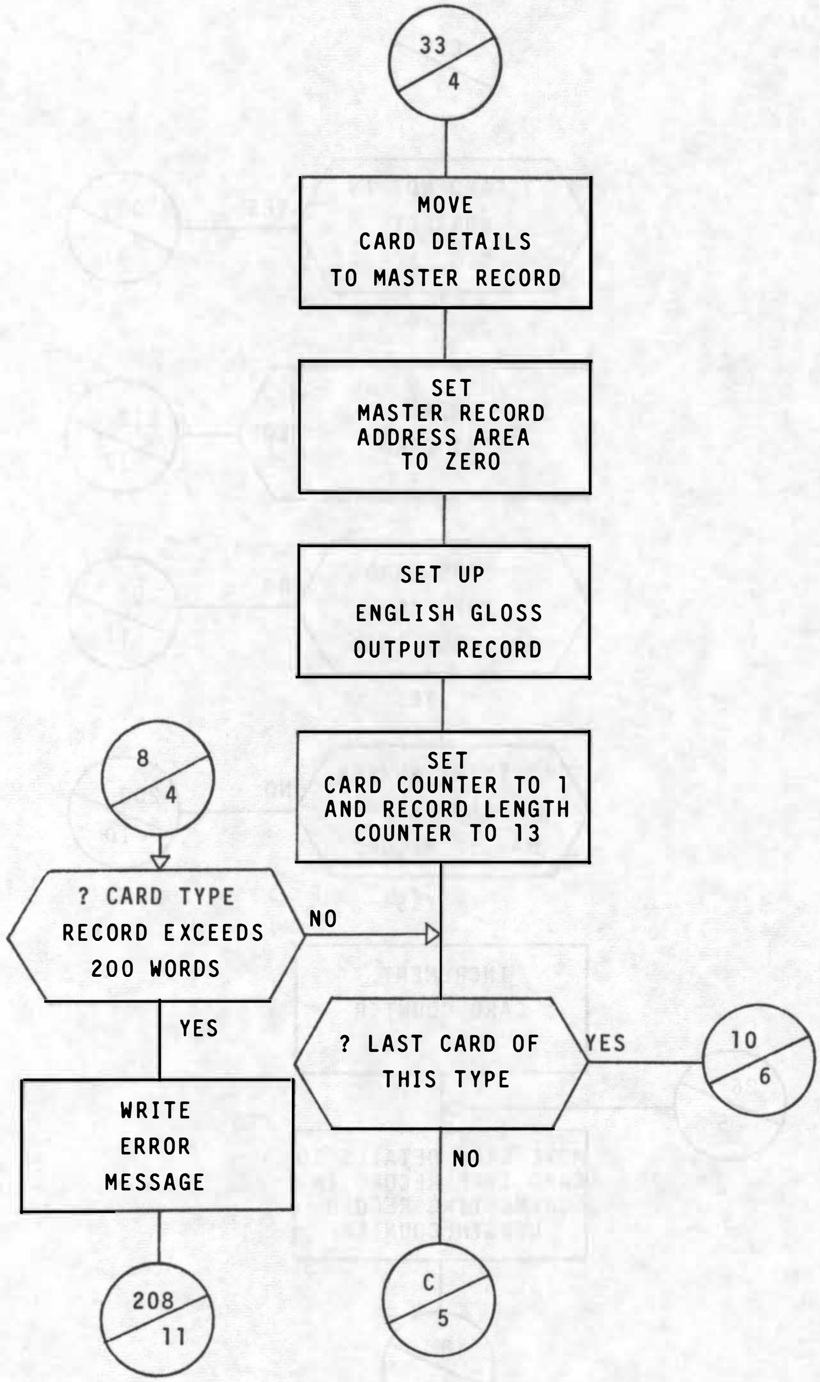
A card with item number 999999 was used to indicate the end of the card input. At the end of each run the new counts for the record length files were printed and also punched to provide the control cards for the following run.

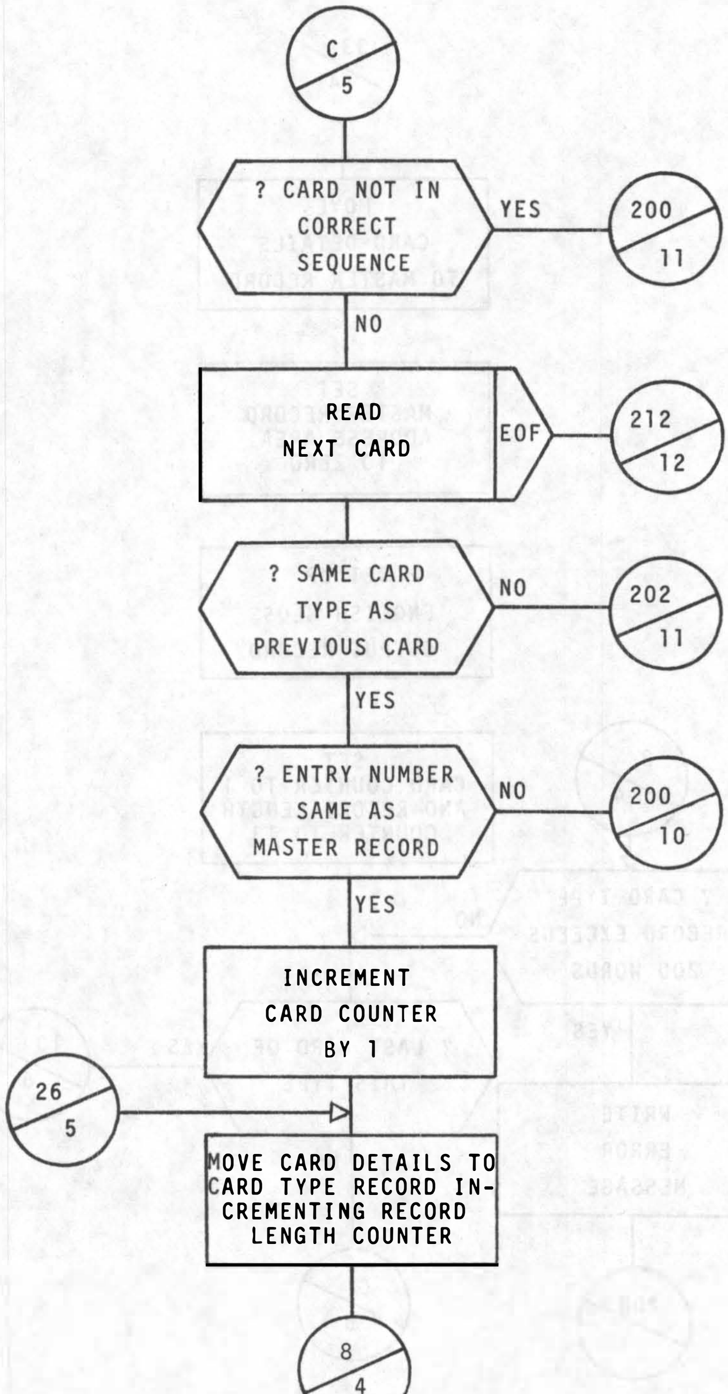
2.2.7 Flowchart

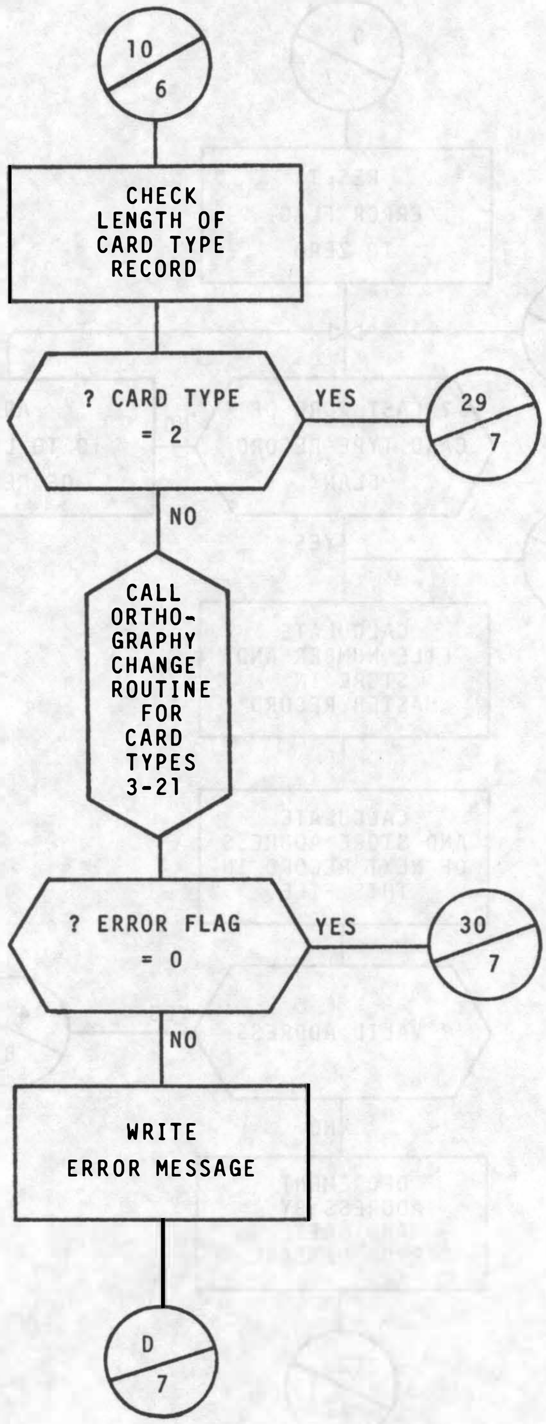


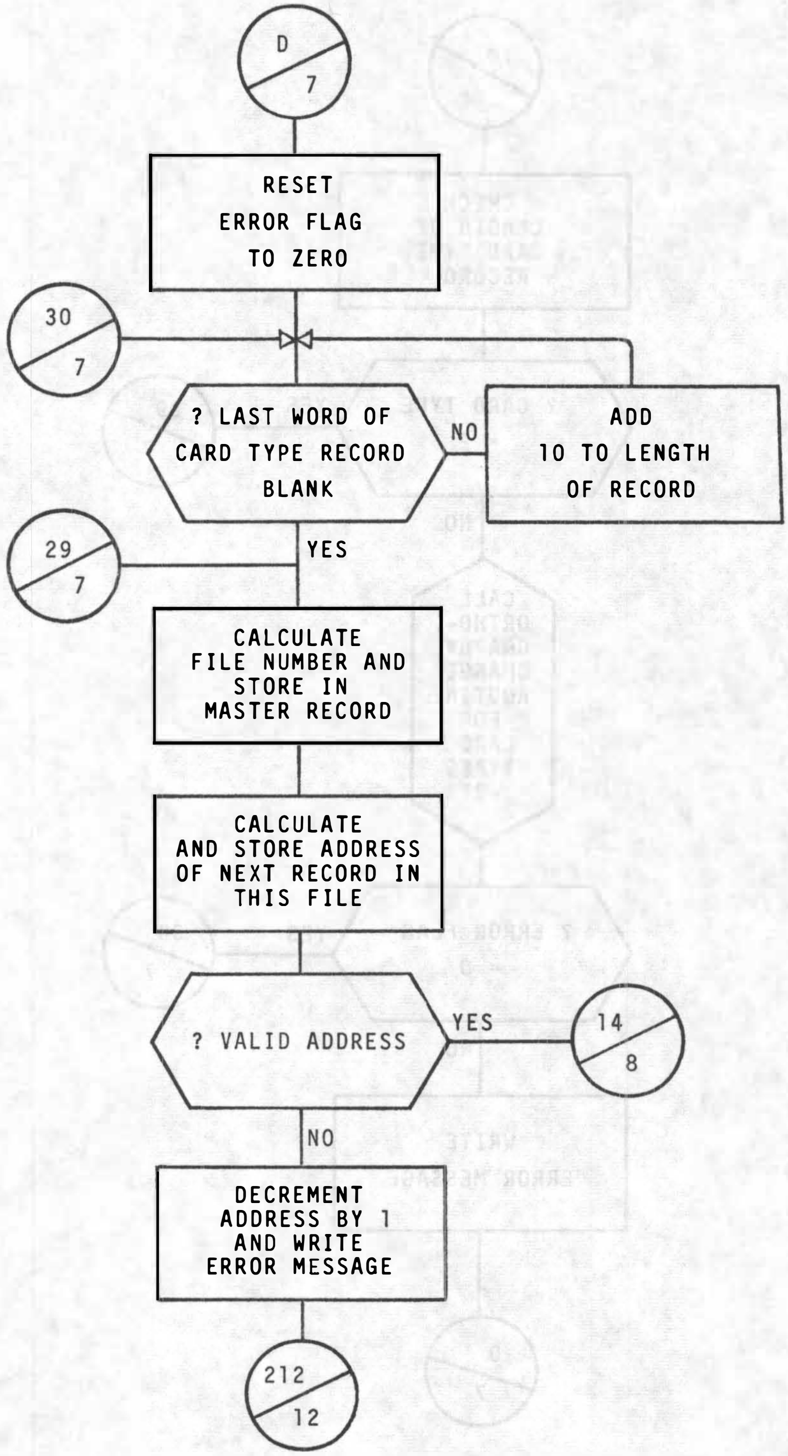

A
2
REREAD
SEMANTIC DOMAIN
CODE UNDER
INTEGER FORMAT
2
2
MOVE
CARD DETAILS
TO MASTER RECORD
READ
CARD TYPE 2
EOF
212
12
? ENTRY NUMBER
CARD TYPE 2
≠ ENTRY NUMBER
CARD TYPE 1
YES
200
10
NO
? CARD NUMBER
≠ 2
YES
200
10
NO
B
3

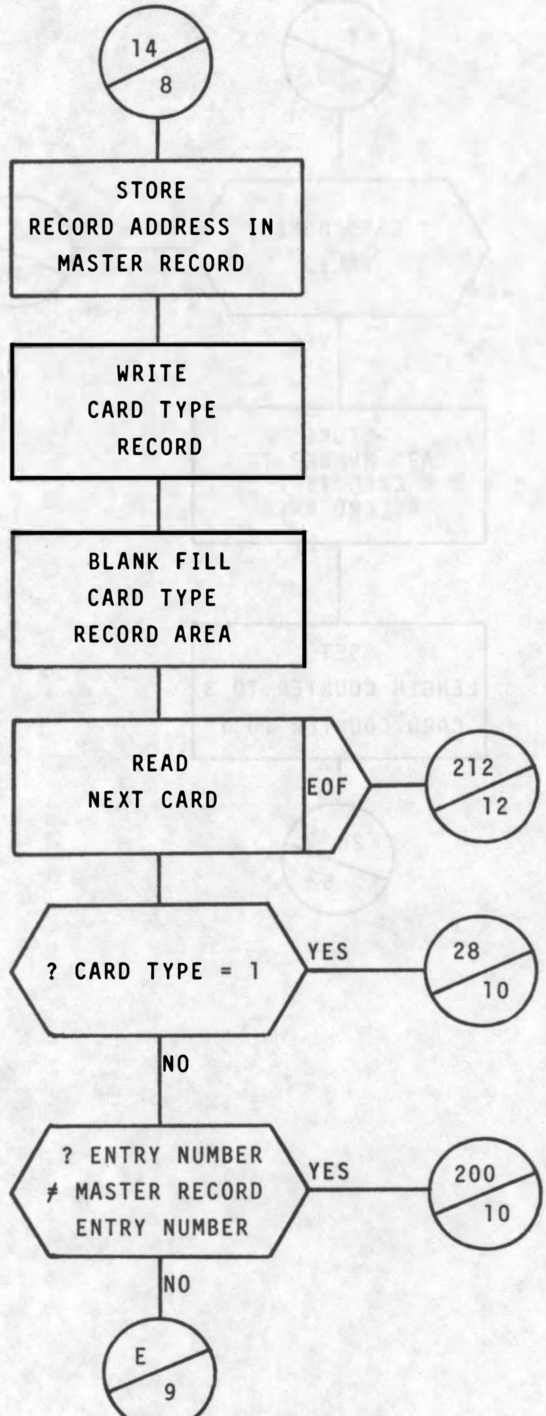

B
3
REREAD GRAMMATICAL CODE UNDER INTEGER FORMAT
CALL ORTHO-GRAPHY CHANGE ROUTINE FOR ENGA ENTRY
? ERROR FLAG = 0
YES
33
4
NO
WRITE ERROR MESSAGES
RESET ERROR FLAG
33
4

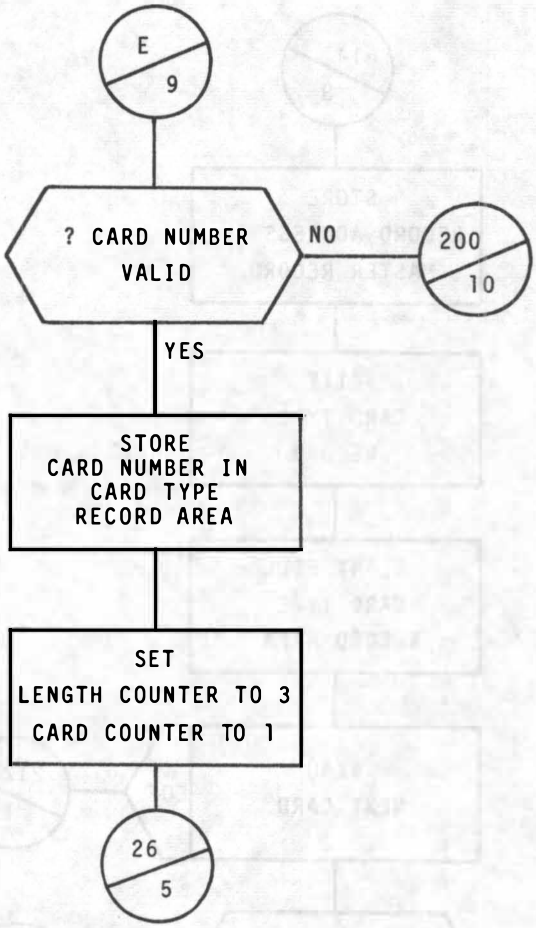

33
4
MOVE
CARD DETAILS
TO MASTER RECORD
SET
MASTER RECORD
ADDRESS AREA
TO ZERO
SET UP
ENGLISH GLOSS
OUTPUT RECORD
SET
CARD COUNTER TO 1
AND RECORD LENGTH
COUNTER TO 13
8
4
? CARD TYPE
RECORD EXCEEDS
200 WORDS
NO
YES
WRITE
ERROR
MESSAGE
208
11
? LAST CARD OF
THIS TYPE
YES
10
6
NO
C
5

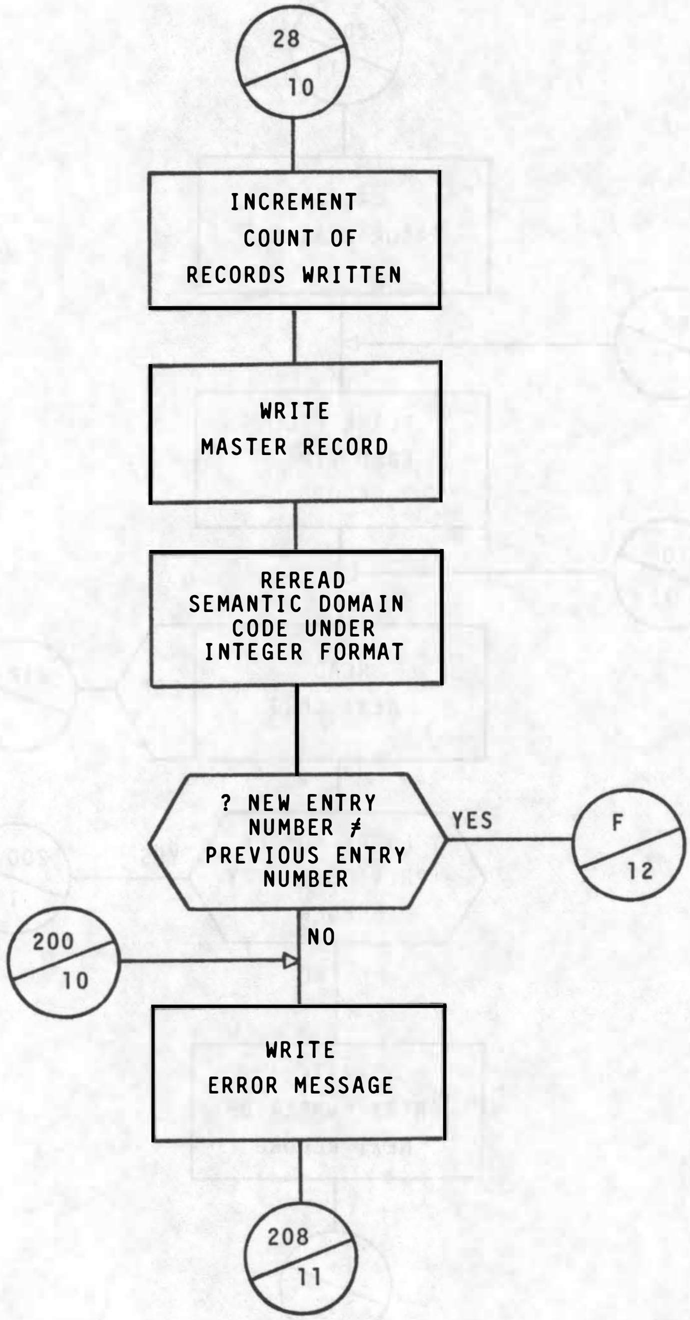

C
5
? CARD NOT IN CORRECT SEQUENCE
YES
200
11
NO
READ NEXT CARD
EOF
212
12
? SAME CARD TYPE AS PREVIOUS CARD
NO
202
11
YES
? ENTRY NUMBER SAME AS MASTER RECORD
NO
200
10
YES
INCREMENT CARD COUNTER BY 1
26
5
MOVE CARD DETAILS TO CARD TYPE RECORD INCREMENTING RECORD LENGTH COUNTER
8
4

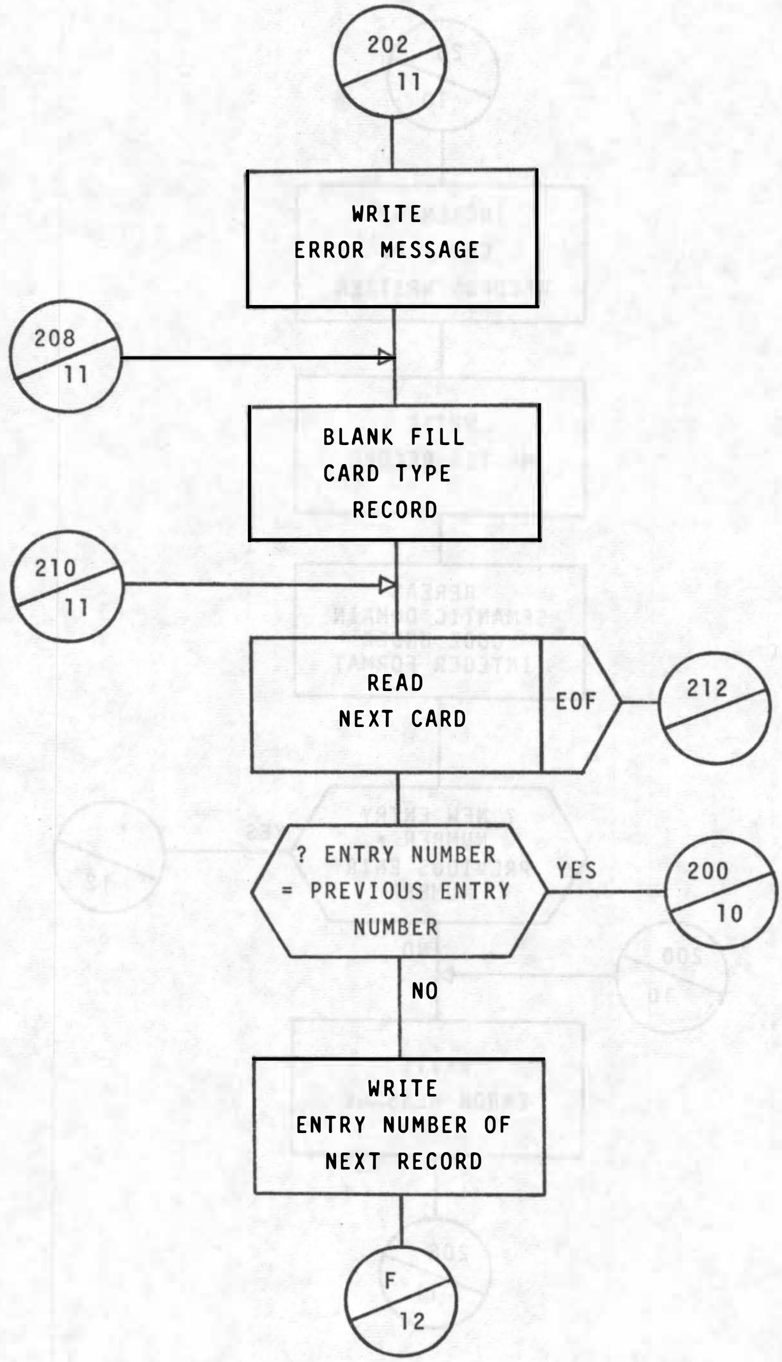

10
6
CHECK
LENGTH OF
CARD TYPE
RECORD
? CARD TYPE
= 2
YES
29
7
NO
CALL
ORTHO-
GRAPHY
CHANGE
ROUTINE
FOR
CARD
TYPES
3-21
? ERROR FLAG
= 0
YES
30
7
NO
WRITE
ERROR MESSAGE
D
7

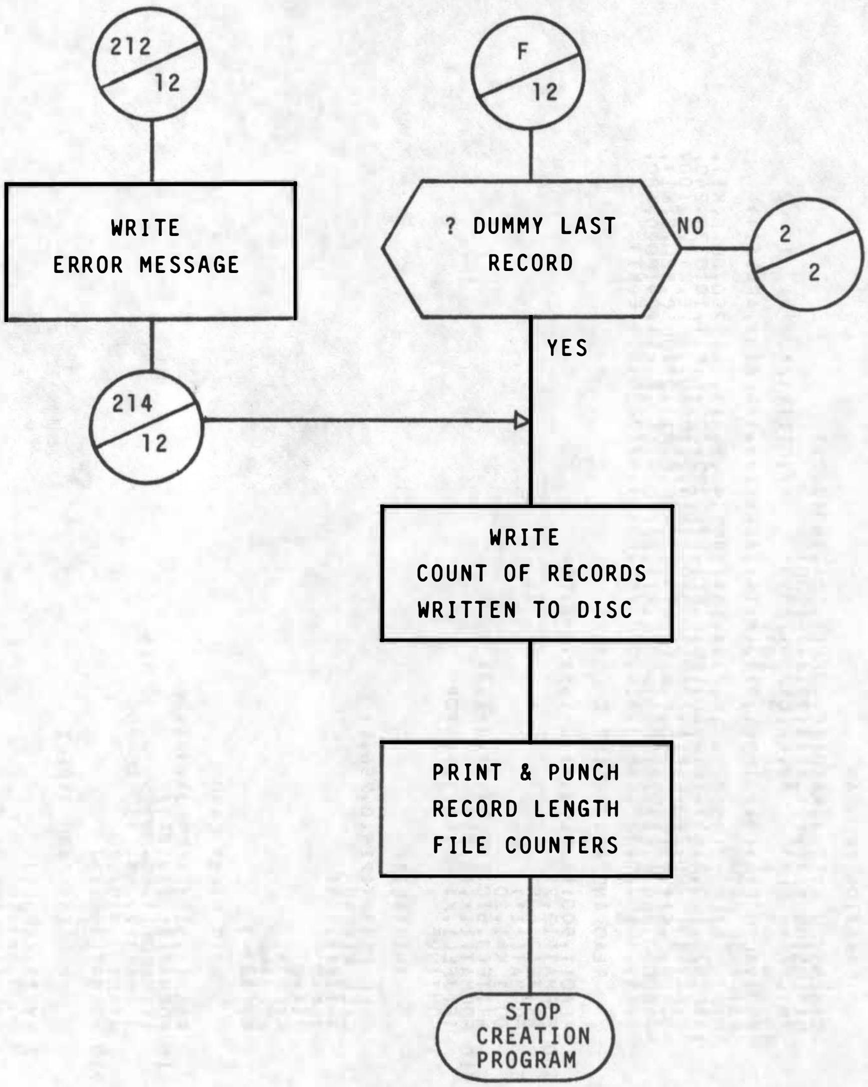

D
7
RESET
ERROR FLAG
TO ZERO
30
7
? LAST WORD OF
CARD TYPE RECORD
BLANK
NO
ADD
10 TO LENGTH
OF RECORD
YES
29
7
CALCULATE
FILE NUMBER AND
STORE IN
MASTER RECORD
CALCULATE
AND STORE ADDRESS
OF NEXT RECORD IN
THIS FILE
? VALID ADDRESS
YES
14
8
NO
DECREMENT
ADDRESS BY 1
AND WRITE
ERROR MESSAGE
212
12


14
8
STORE
RECORD ADDRESS IN
MASTER RECORD
WRITE
CARD TYPE
RECORD
BLANK FILL
CARD TYPE
RECORD AREA
READ
NEXT CARD
EOF
212
12
? CARD TYPE = 1
YES
28
10
NO
? ENTRY NUMBER
≠ MASTER RECORD
ENTRY NUMBER
YES
200
10
NO
E
9


E
9
? CARD NUMBER VALID
NO
200
10
YES
STORE CARD NUMBER IN CARD TYPE RECORD AREA
SET LENGTH COUNTER TO 3 CARD COUNTER TO 1
26
5


28
10
INCREMENT
COUNT OF
RECORDS WRITTEN
WRITE
MASTER RECORD
REREAD
SEMANTIC DOMAIN
CODE UNDER
INTEGER FORMAT
? NEW ENTRY
NUMBER ≠
PREVIOUS ENTRY
NUMBER
YES
F
12
NO
200
10
WRITE
ERROR MESSAGE
208
11


202
11
WRITE
ERROR MESSAGE
208
11
BLANK FILL
CARD TYPE
RECORD
210
11
READ
NEXT CARD
EOF
212
? ENTRY NUMBER
= PREVIOUS ENTRY
NUMBER
YES
200
10
NO
WRITE
ENTRY NUMBER OF
NEXT RECORD
F
12


212
12
F
12
WRITE
ERROR MESSAGE
? DUMMY LAST
RECORD
NO
2
2
YES
214
12
WRITE
COUNT OF RECORDS
WRITTEN TO DISC
PRINT & PUNCH
RECORD LENGTH
FILE COUNTERS
STOP
CREATION
PROGRAM

2.2.8 Listing

```
C          CREATION PROGRAM
C
      DIMENSION KY(70),KARD(20),JFL(3,20),IND(2,21)
      DIMENSION ARRDV(2),ARRA(7),ARRB(200)
      INTEGER*4 BL/'    '/,KREC(209)/209*'    '/,CT(11)/'   *   1   2
     13   4   5   6   7   8   9  10'/
      EQUIVALENCE(KY(31),IND(1,2)),(KY(4),ARRA(1)),(KREC(3),ARRB(1))
      EXTERNAL ORTHO
      DEFINE FILE  7(5500,265,E,KAA),13(11800,40,E,K1),14(5250,80,E,K1),
     115(1520,120,E,K1),16(0960,160,E,K1),17(0390,200,E,K1),18(0330,240,
     2E,K1),19(0200,280,E,K1),20(0180,320,E,K1),21(0080,360,E,K1),22(007
     30,400,E,K1),23(0070,440,E,K1),24(0060,480,E,K1),25(0060,520,E,K1),
     426(0050,560,E,K1),27(0025,600,E,K1),28(0025,640,E,K1),29(0020,680,
     5E,K1),30(0020,720,E,K1),31(0020,760,E,K1),32(0020,800,E,K1)
C
C          READ AND WRITE FILE COUNTERS
C
      READ(1,500)((JFL(J,K),J=1,3),K=1,20)
  500 FORMAT(I2,2I8)
      WRITE(3,505)
  505 FORMAT('1')
      DO 1 K=1,20
      WRITE(3,510)(JFL(J,K),J=1,3)
  510 FORMAT(11X,I2,2I8)
      IF(JFL(1,K).NE.(K+12))STOP
    1 CONTINUE
C
C          INITIALIZE
C
      CALL ERRSET(215,0,256,1)
      CALL REREAD
      WRITE(3,505)
      IER=0
      KTA=0
      KY(2)=1
      KY(3)=0
      KY(12)=0
C
C          READ FIRST CARD
C
      READ(1,515)(KARD(J),J=1,20)
  515 FORMAT(I6,I2,18A4)
      IF(KARD(1).EQ.999999)GO TO 214
      IF(KARD(2).NE.1)GO TO 200
      READ(99,910)JAJ
  910 FORMAT(36X,I4)
C
C          PROCESS CARD TYPE 1
C
    2 KY(1)=KARD(1)
      DO 3 J=3,10
    3 KY(J+1)=KARD(J)
      DO 4 J=11,20
    4 KY(J+2)=KARD(J)
C
C          READ CARD TYPE 2
C
      READ(1,515,END=212)(KARD(J),J=1,20)
      IF(KARD(1).NE.KY(1))GO TO 200
      IF(KARD(2).NE.2)GO TO 200
```

```
      READ(99,915)JAJ,JBJ
  915 FORMAT(8X,I3,I4)
C
C          CALL ORTHOGRAPHY CHANGE ROUTINE
C
      JVB=0
      IF(JAJ.EQ.2.OR.JAJ.EQ.8)JVB=1
      CALL CSDV(ARRA,ARRDV,25,25)
      CALL XTOPLI(ORTHO,ARRDV,JVB,IER,1)
C
C          WRITE ERROR MESSAGES
C
      IF(IER.EQ.0)GO TO 33
      GO TO(31,32),IER
   31 WRITE(3,540)KY(1),(ARRA(J),J=1,7)
  540 FORMAT('0EXTENSION GT 25 - ',I4,4X,6A4,A1)
      IER=0
      GO TO 33
   32 WRITE(3,545)KY(1),KY(2),(ARRA(J),J=1,6)
  545 FORMAT('0SYNTAX ERROR - ',I4,I2,4X,25A4/(2X,31A4))
      IER=0
C
C          MOVE DATA FROM CARD 2 TO MASTER RECORD
C          SET MASTER RECORD ADDRESS AREA TO ZERO
C
   33 DO 5 J=3,10
    5 KY(J+20)=KARD(J)
      DO 6 J=33,70
    6 KY(J)=0
C
C          SET UP ENGLISH GLOSS OUTPUT RECORD
C
      KREC(1)=KY(1)
      KREC(2)=2
      KREC(3)=BL
      KREC(4)=BL
      DO 7 J=11,20
    7 KREC(J-6)=KARD(J)
      KNT=14
      KR=1
C
C          PROCESS CARD TYPES 2-21
C
    8 IF(KREC(201).NE.BL)GO TO 206
      IF(KARD(20).EQ.BL)GO TO 10
      IF(KARD(20).NE.CT(KR))GO TO 202
      READ(1,515,END=212)(KARD(J),J=1,20)
      IF(KARD(2).NE.KREC(2))GO TO 202
      IF(KARD(1).NE.KY(1))GO TO 200
      KR=KR+1
   26 DO 9 J=3,20
      KREC(KNT)=KARD(J)
    9 KNT=KNT+1
      KNT=KNT-1
      GO TO 8
C
C          CHECK LENGTH OF RECORD FOR CARD TYPES 2-21
C
   10 DO 11 J=11,201,10
      IF(KREC(J).EQ.BL)GO TO 27
```

```
   11 CONTINUE
      WRITE(3,520)
  520 FORMAT('1',10X,'EXECUTION ERROR')
      GO TO 212
   27 J=J-1
C
C         CALL ORTHOGRAPHY CHANGE ROUTINE FOR CARD TYPES 3-21
C
      IF(KREC(2).EQ.2)GO TO 29
      J=J*4-8
      CALL CSDV(ARRB,ARRDV,800,J)
      CALL XTOPL1(ORTHO,ARRDV,0,IER,0)
      J=J/4
      IF(IER.EQ.0)GO TO 30
      WRITE(3,545)KREC(1),KREC(2),(ARRB(K),K=1,J)
      IER=0
   30 IF(KREC(J).EQ.BL)GO TO 29
      J=J+10
      GO TO 30
C
C         WRITE RECORD FOR CARD TYPES 2-21
C
   29 LF=J/10
      LU=LF+12
      IND(1,KREC(2))=LU
      JFL(3,LF)=JFL(3,LF)+1
      IF(JFL(3,LF).LE.JFL(2,LF))GO TO 14
      JFL(3,LF)=JFL(3,LF)-1
      WRITE(3,525)LU
  525 FORMAT('1',10X,'NO. OF RECORDS EXCEEDED IN FILE ',I2)
      GO TO 212
   14 IND(2,KREC(2))=JFL(3,LF)
      K1=JFL(3,LF)
      WRITE(LU'K1,530)(KREC(K),K=1,J)
  530 FORMAT(I6,I2,198A4)
   24 DO 25 K=11,J
   25 KREC(K)=BL
C
C         READ CARD TYPES 3-21
C
      READ(1,515,END=212)(KARD(J),J=1,20)
      IF(KARD(2).EQ.1)GO TO 28
      IF(KARD(1).NE.KY(1))GO TO 200
      IF(KARD(2).GT.21.OR.KARD(2).LE.KREC(2))GO TO 200
      KREC(2)=KARD(2)
      KNT=3
      KR=1
      GO TO 26
C
C         ADD TO COUNTER AND WRITE MASTER RECORD
C
   28 KTA=KTA+1
      KAA=KY(1)
      WRITE(3,905)KTA,KAA
  905 FORMAT(' ',I4,2X,I5)
      WRITE(7'KAA,535)(KY(J),J=1,70)
  535 FORMAT(I5,I1,I5,8A4,I10,18A4,20(I2,I5))
      IF(KARD(1).EQ.999999)GO TO 214
      READ(99,910)JAJ
      IF(KARD(1).EQ.KY(1))GO TO 200
```

```
      GO TO 2
C
C         ERROR MESSAGES
C
  200 WRITE(3,201)KARD(1),KARD(2)
  201 FORMAT('0',90X,'IDENT ERROR:',I6,I3)
      GO TO 208
  202 WRITE(3,203)KARD(1),KARD(2),KARD(20)
  203 FORMAT('0',90X,'CONTINUATION ERROR:',I6,I3,A4)
      GO TO 208
  206 WRITE(3,207)KARD(1),KARD(2)
  207 FORMAT('0',90X,'RECORD SIZE GT 200:',I6,I3)
  208 DO 209 J=11,201
  209 KREC(J)=BL
  210 READ(1,515,END=212)(KARD(J),J=1,20)
      IF(KARD(2).NE.1)GO TO 210
      IF(KARD(1).EQ.KY(1))GO TO 200
      WRITE(3,211)KARD(1)
  211 FORMAT('0',90X,'NEXT RECORD:',I6)
      IF(KARD(1).EQ.999999)GO TO 214
      GO TO 2
C
C         END OF RUN PROCEDURES
C
  212 WRITE(3,213)KY(1)
  213 FORMAT('0',10X,'RECORD NOT WRITTEN:',I6)
  214 WRITE(3,215)KTA
  215 FORMAT('1',10X,I6,' RECORDS FILED TO DISK'/1X)
C
C         WRITE AND PUNCH FILE COUNTERS
C
      DO 216 K=1,20
      WRITE(3,510)(JFL(J,K),J=1,3)
  216 WRITE(2,217)(JFL(J,K),J=1,3)
  217 FORMAT('V',I2,2I8)
      WRITE(3,505)
      STOP
      END
```

```
ORTHO: PROCEDURE(SB,JVB,IFLA,JM);
/* THIS SUBROUTINE OPERATES ON ENGA CARDS EXCEPT CARD TWO (ENGLISH).
   IT PERFORMS FOUR TYPES OF ORTHOGRAPHY CHANGES ON ALL WORDS PRESENTED
   TO IT INCLUDING ANY ENGLISH ON CARDS 3 - 21.
   TYPE 1:   LABIALIZATION.
             ANY CONSONANT + W GOES TO CONSONANT + U
             EG  BWAA -> BUAA
                 KWAA -> KUAA   ETC.
   TYPE 2:   VOWEL CLUSTERS.
```

```
        LAST TWO VOWELS OF THREE VOWEL SET WHICH ARE NOT IDENTICAL
        ARE CHANGED AS FOLLOWS:
        V2=I -> IY
        V2=U -> UW
        V2=E -> Y
       V2=O -> W
        EG  KAIA -> KAIYA
            KOEA -> KOYA   ETC.
 TYPE 3    VOWEL CLUSTERS IN VERB STEMS:
        IN VERBS IF  V1 V2 V3 PRECEDES -GI OR -GE, AND IF V2 + V3 ARE
        THE SAME V3 IS REMOVED.
        EG  MAIIGI -> MAIGI
 TYPE 4    PRENASALIZATION.
        IN THE MIDDLE OF A WORD ONLY (NOT AT THE BEGINNING)
        B ->MB
       D -> ND
        G -> NG
        J -> NJ
        EG KADEGE -> KANDENGE
           AJA -> ANJA
 THE SUBROUTINE IS IN THREE SECTIONS.
 SECTION 1  REMOVES BLANKS FROM END OF STRING SUPPLIED.
 SECTION 2 SPLITS OFF EACH ENGA WORD TO PASS ACROSS TO INTERNAL
           SUBROUTINE CHANGE.
 SECTION 3 (SUBROUTINE CHANGE) PERFORMS THE FOUR TYPES OF ORTHOGRAPHY
           CHANGES.
 PARAMETERS
 SB - STRING CONTAINING ENGA WORDS
 JVB = 1 IF MAIN ENTRY IF A VERB
     = 0 OTHERWISE
```

```
     IFLA = 0 NO ERRORS IN PROCESSING
          = 1 MAIN ENTRY WOULD EXPAND TO GREATER THAN 25 CHARACTERS
          = 2 TONE CODE HAS COME AFTER A SEMI-VOWEL (Y OR W)
     JM = 1 MAIN ENTRY
        = 0 CARDS 3-21
     OTHER IMPORTANT VARIABLES.
     JPOS - STARTING CHARACTER OF NEXT WORD.
     SA,SD,SR,S INTERMEDIATE STRINGS TO HOLD WORDS TEMPORARILY.
     SUB - USED TO EXAMINE ONE CHARACTER AT A TIME
     ICOUNT - COUNTS NO OF VOWELS FOUND IN SEQUENCE
     L - HOLDS POSITIONS OF V2 AND V3
     FOUND - BRANCH TAKEN IF 3 VOWELS FOUND FOR TYPE 2.
     PROC - BRANCH TAKEN IF WORD ENDS IN GI OR GE FOR TYPE 3.
     R - BRANCH TAKEN IF 3 VOWELS FOUND FOR TYPE 3.                   */
DCL SB CHAR(800) VARYING,
IFLA BIN FIXED (31,0),
(JVB,NWD,JM) BIN FIXED (31,0);
/*  SECTION 1 - REMOVES BLANKS FROM END OF STRING                     */
JPOS=1;
DO I=LENGTH(SB) TO 1 BY -1;
TEST: IF SUBSTR(SB,I,1)¬=' ' THEN
/*  SECTION 2 - CHECKS FOR ERROR TYPE 1 AND SPLITS OFF NEXT WORD      */
(STRINGRANGE): ONE: BEGIN;
DCL SA CHAR(I),
SUB CHAR(1),
SD CHAR(800) VARYING;
IF JM=1 & I>20 THEN DO;
IFLA=1;
RETURN;
END;
KOUNT=1;
SA=SB;
IF JM=0 THEN SB=(800)' ';
SB='';
/*  START PROCESSING AT FIRST NON-BLANK CHARACTER.                    */
DO I=1 TO LENGTH(SA);
IF SUBSTR(SA,I,1)¬=' ' THEN DO;
JPOS=I;
SB=SB||SUBSTR(SA,1,I-1);
GO TO LOOP;
END;
END;
/*  SEARCH FOR NEXT WORD TERMINATOR AND CALL SUBROUTINE CHANGE        */
LOOP: DO I=JPOS TO LENGTH(SA);
SUB=SUBSTR(SA,I,1);
J=0;
IF SUB=';' | SUB=',' | SUB=' ' THEN DO;
```

```
DO J=1 TO LENGTH(SA)-I;
IF SUBSTR(SA,I+J,1)¬=' ' THEN
GO TO CALL;
END;
J=J-1;
END;
END;
CALL: N=I+J;
CALL CHANGE(SUBSTR(SA,JPOS,I-JPOS),IFLAG,SD);
/* TEST ERROR FLAG - RETURN ORIGINAL STRING IF ERROR. BLANK OUT
    REST OF STRING AND RETURN..                                          */
JPOS=N;
IFLA=IFLAG;
IF IFLAG¬=0 THEN DO;
SB=SA;
RETURN;
END;
SB=SB||SD||SUB;
KOUNT=KOUNT+1;
IF JPOS<=LENGTH(SA) THEN GO TO LOOP;
IF SUBSTR(SB,LENGTH(SB)-1,1)=SUBSTR(SB,LENGTH(SB),1)
THEN SB=SUBSTR(SB,1,LENGTH(SB)-1);
IF JM=1 THEN LEN=25;
ELSE LEN=800;
SD=(800)' ';
SB=SB||SUBSTR(SD,1,(LEN-LENGTH(SB)));
RETURN;
END ONE;
END;
IFLA=2;
RETURN;
CHANGE: PROCEDURE(S,IFL,SR);
DCL S CHAR(*),
SR CHAR(800) VARYING,
SUB CHAR(1),
L(2),
VOWEL(5) CHAR(1) INITIAL('A','E','I','O','U');
/* TYPE 1 CHANGE. SEARCH FOR CONSONANT + W AND CHANGE TO U.             */
IFL=0;
DO I=1 TO LENGTH(S)-1;
SUB=SUBSTR(S,I,1);
IF SUB<'A' | SUB>'Z' THEN GO TO END;
DO J=1 TO 5;
IF SUB=VOWEL(J) THEN GO TO END;
END;
IF SUBSTR(S,I+1,1)='W' THEN SUBSTR(S,I+1,1)='U';
END: END;
/* TYPE 4 CHANGE. SEARCH FOR B,D,G,J AND INSERT REQUIRED CHARACTER
    BEFORE COPYING STRING INTO INTERMEDIATE STRING SR.                   */
SR=SUBSTR(S,1,1);
DO I=2 TO LENGTH(S);
SUB=SUBSTR(S,I,1);
IF SUB='B' THEN SR=SR||'M';
IF SUB='D'|SUB='G'|SUB='J' THEN SR=SR||'N';
SR=SR||SUB;
END;
/* TYPE 2 CHANGE. SEARCH BACKWARDS THROUGH WORD FOR THREE VOWELS IN A
    ROW OR SEPARATED ONLY BY TONES. STORE POSITIONS OF THESE THREE.     */
I=LENGTH(SR);
ICOUNT=0;
```

```
LOOP: IF I<3 THEN GO TO NXT;
DO J=I TO I-10 BY -1;
IF J<1 THEN GO TO NXT;
SUB=SUBSTR(SR,J,1);
IF SUB='''' THEN GO TO ROUND;
DO K=1 TO 5;
IF SUB=VOWEL(K) THEN DO;
ICOUNT=ICOUNT+1;
IF ICOUNT=3 THEN GO TO FOUND;
L(ICOUNT)=J;
GO TO ROUND;
END;
END;
FIX: ICOUNT=0;
I=I-1;
GO TO LOOP;
ROUND: END;
GO TO FIX;
/* CHECK FINAL TWO VOWELS ARE NON-IDENTICAL AND THEN PERFORM PRESCRIB-
    ED CHANGES PAYING SPECIAL ATTENTION TO TONE CODES.                  */
FOUND: IF SUBSTR(SR,L(2),1)=SUBSTR(SR,L(1),1) THEN GO TO FIX;
SUB=SUBSTR(SR,L(2),1);
IF SUB='A' THEN GO TO FIX;
IF SUB='E' THEN DO;
SUBSTR(SR,L(2),1)='Y';
IF SUBSTR(SR,L(2)+1,1)='''' THEN SR=SUBSTR(SR,1,J    )||''''||'Y'||
SUBSTR(SR,L(2)+2);
GO TO FIX;
END;
IF SUB='O' THEN DO;
SUBSTR(SR,L(2),1)='W';
IF SUBSTR(SR,L(2)+1,1)='''' THEN SR=SUBSTR(SR,1,J    )||''''||'W'||
SUBSTR(SR,L(2)+2);
GO TO FIX;
END;
IF SUBSTR(SR,L(2)+1,1)='''' THEN L(2)=L(2)+1;
IF SUB='I' THEN DO;
SR=SUBSTR(SR,1,L(2))||'Y'||SUBSTR(SR,L(2)+1);
I=L(2);
ICOUNT=0;
GO TO LOOP;
END;
IF SUB='U' THEN DO;
SR=SUBSTR(SR,1,L(2))||'W'||SUBSTR(SR,L(2)+1);
I=L(2);
ICOUNT=0;
GO TO LOOP;
END;
/* TYPE 3 CHANGE. CHECK FOR VERB AND LAST SYLLABLE BEING GI OR GE.   */
NXT: IF JVB=0 THEN GO TO RET;
LN=LENGTH(SR);
IF SUBSTR(SR,LN-3)='NGI'''|SUBSTR(SR,LN-3)='NGE''' THEN DO;
KK=LN-4;
GO TO PROC;
END;
IF SUBSTR(SR,LN-2)='NGI'|SUBSTR(SR,LN-2)='NGE' THEN DO;
KK=LN-3;
GO TO PROC;
END;
ELSE GO TO RET;
```

```
/* SEARCH BACKWARDS THROUGH WORD FOR THREE VOWELS                          */
PROC: ICOUNT=0;
DO I=KK TO KK-10 BY -1;
 IF I<1 THEN GO TO RET;
SUB=SUBSTR(SR,I,1);
 IF SUB='''' THEN GO TO ND;
DO J=1 TO 5;
 IF SUB=VOWEL(J) THEN DO;
 ICOUNT=ICOUNT+1;
 IF ICOUNT=3 THEN GO TO R;
L(ICOUNT)=I;
GO TO ND;
END;
END;
 ICOUNT=0;
ND: END;
GO TO RET;
/* CHECK IF FINAL TWO EQUAL AND DELETE LAST ONE.                           */
R: IF SUBSTR(SR,L(2),1)¬=SUBSTR(SR,L(1),1)THEN GO TO RET;
SR=SUBSTR(SR,1,L(1)-1)||SUBSTR(SR,L(1)+1);
RET: IF JM=1 & LENGTH(SR)>25 THEN IFL=1;
DO I=1 TO LENGTH(SR)-1;
IF SUBSTR(SR,I,1)='Y'|SUBSTR(SR,I,1)='W' THEN
IF SUBSTR(SR,I+1,1)='''' THEN DO;
IFL=2;
RETURN;
END;
END;
RETURN;
END CHANGE;
END ORTHO;
```

2.3 *UPDATING*

2.3.1 Description of Updating Program

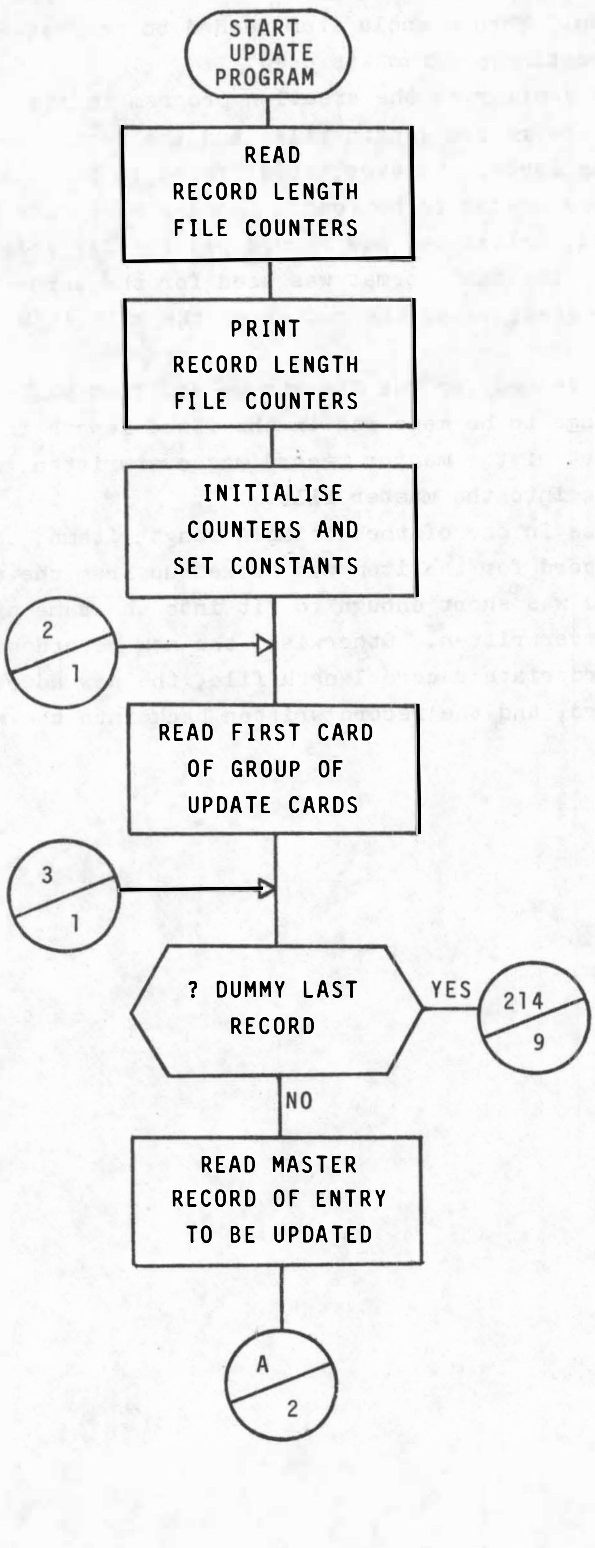
This program was used when information on one card type only required modification. When a whole item needed to be changed or a new item added, the creation program was used.

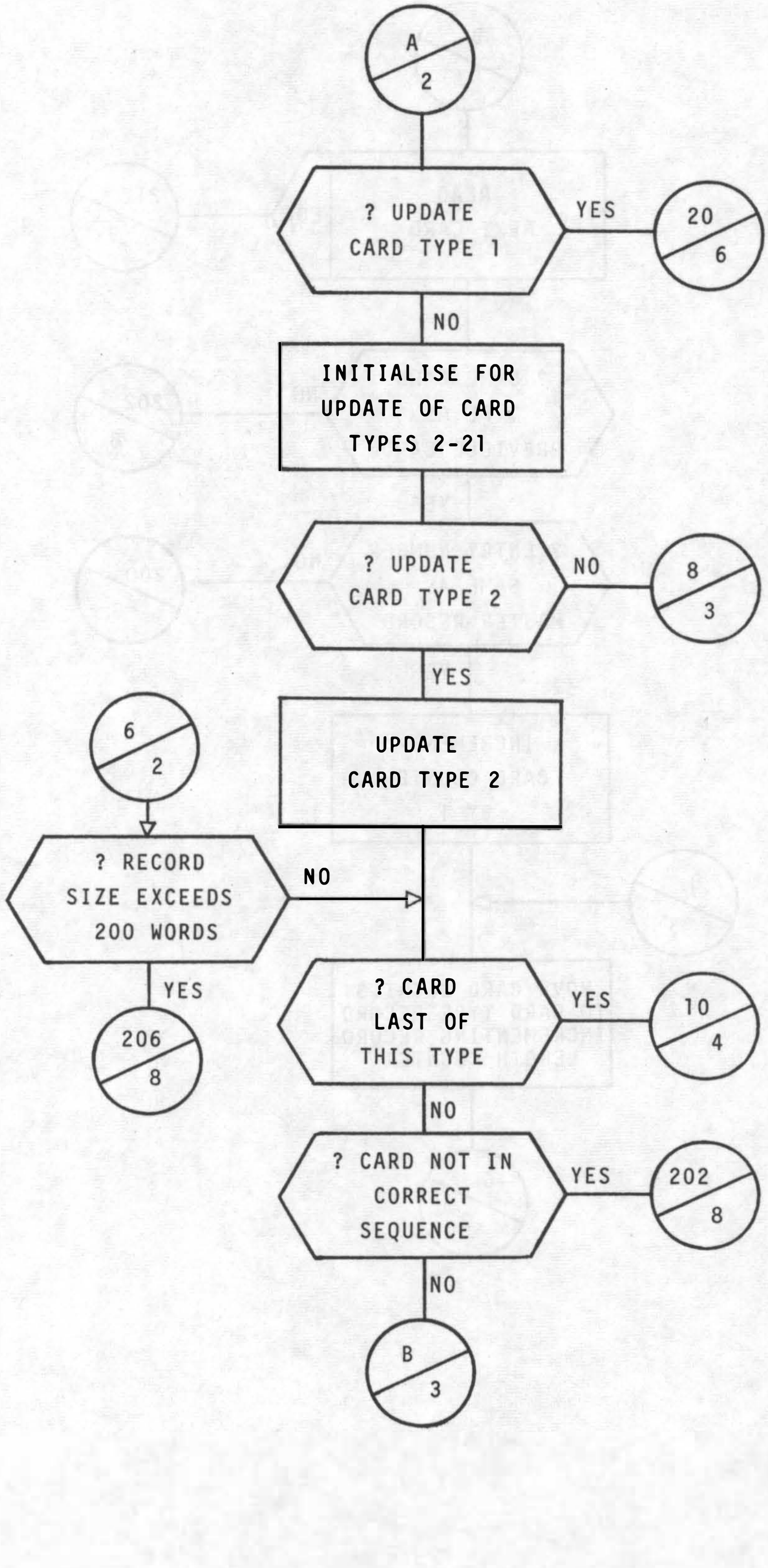
The program was similar to the creation program in its use of control cards for the record length files and the method used for reading and editing the cards. However, it differed in that only the card type to be corrected needed to be read. Changes were made by overwriting the existing record, unless the new record was too large for the area available. Exactly the same format was used for the cards as was used in the creation program; thus, the number of the main item appeared on every card.

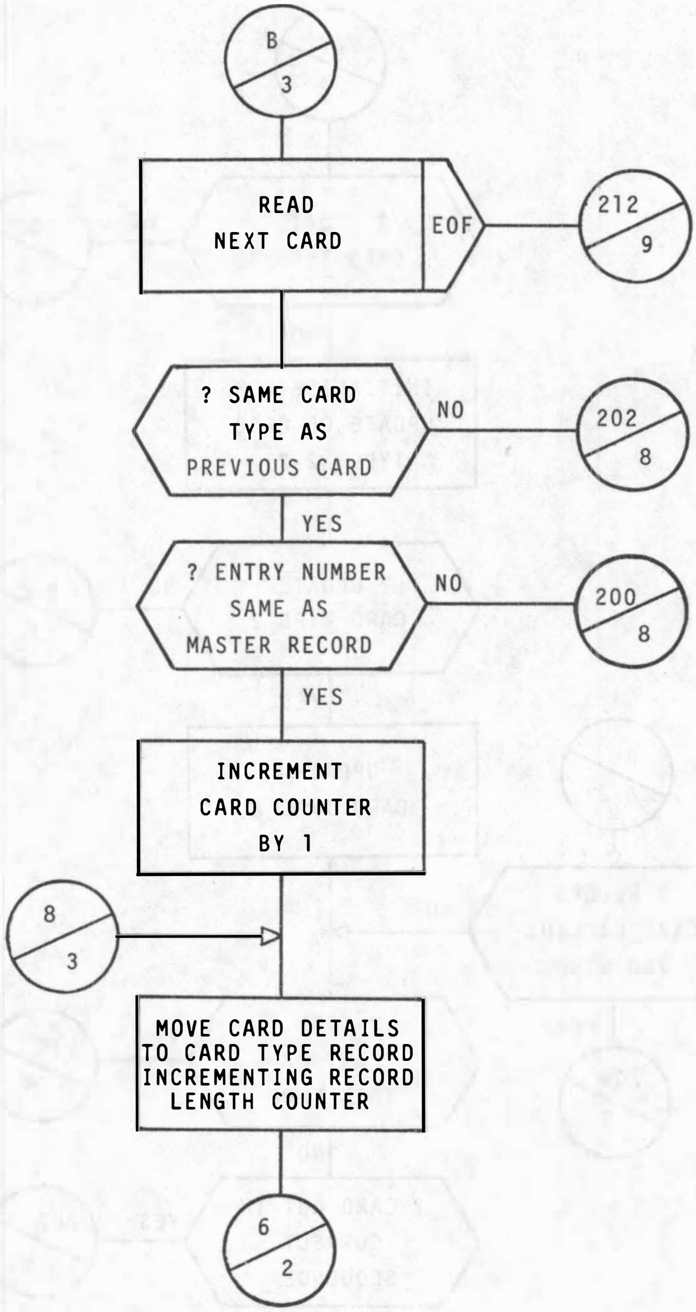
The master file record for the item to be modified was read into core store. If the change to be made was in the fixed length information, the appropriate part of the master record was overwritten, and the record written back into the master file.

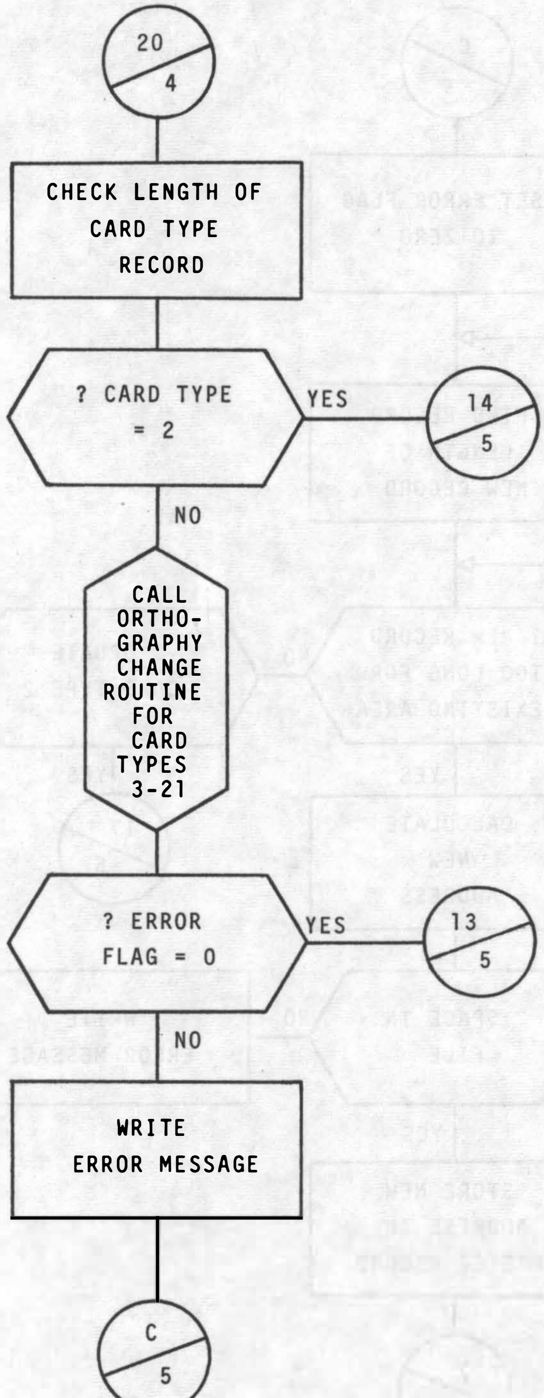
If the change was in one of the variable length items, the address of the existing record for the item was picked up from the master record. When the new record was short enough to fit into the same area, the existing record was overwritten. Otherwise, the new record was added to the end of the appropriate record length file, the new address stored in the master record, and the record written back into the master file.

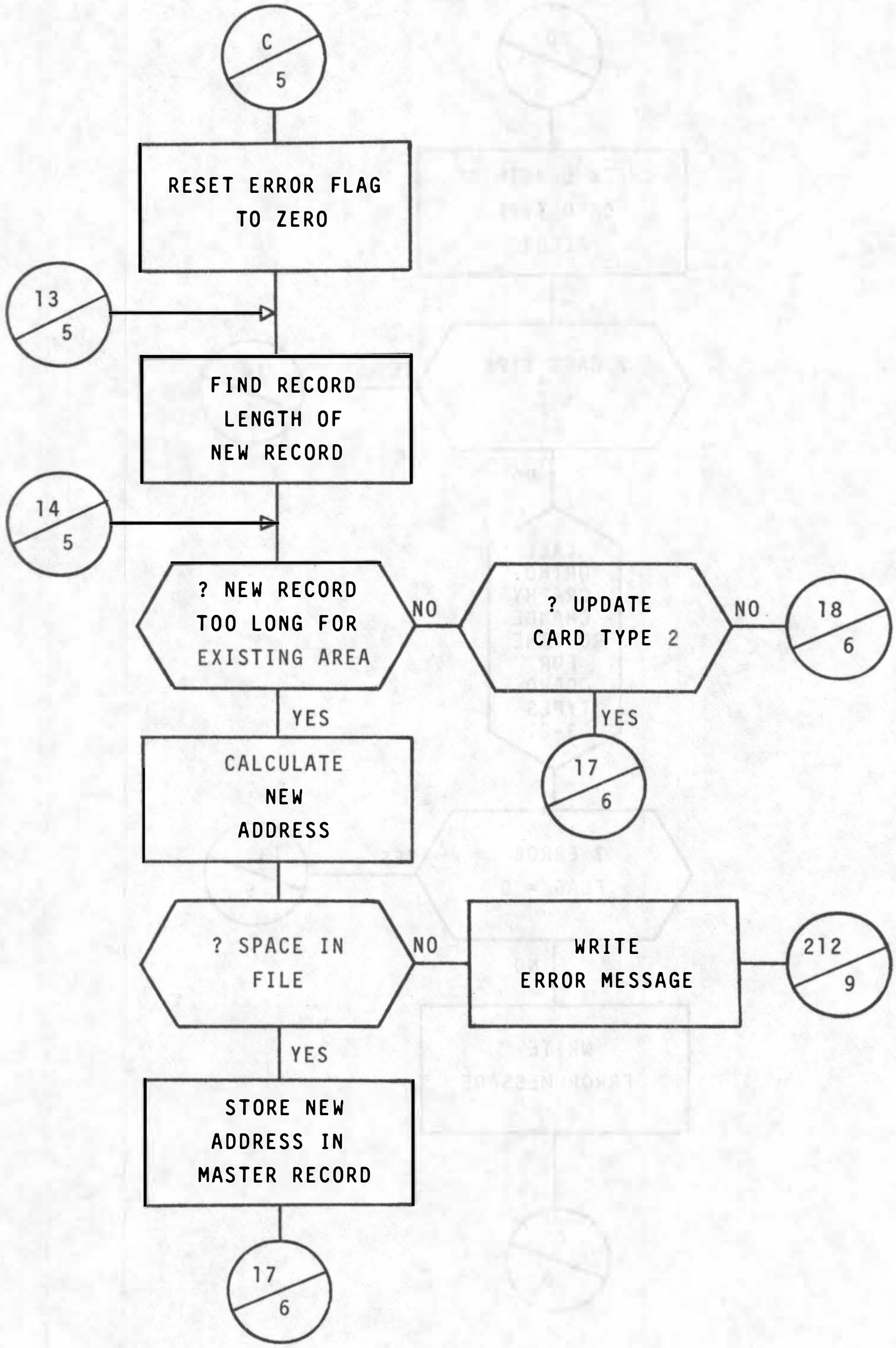
2.3.2 Flowchart

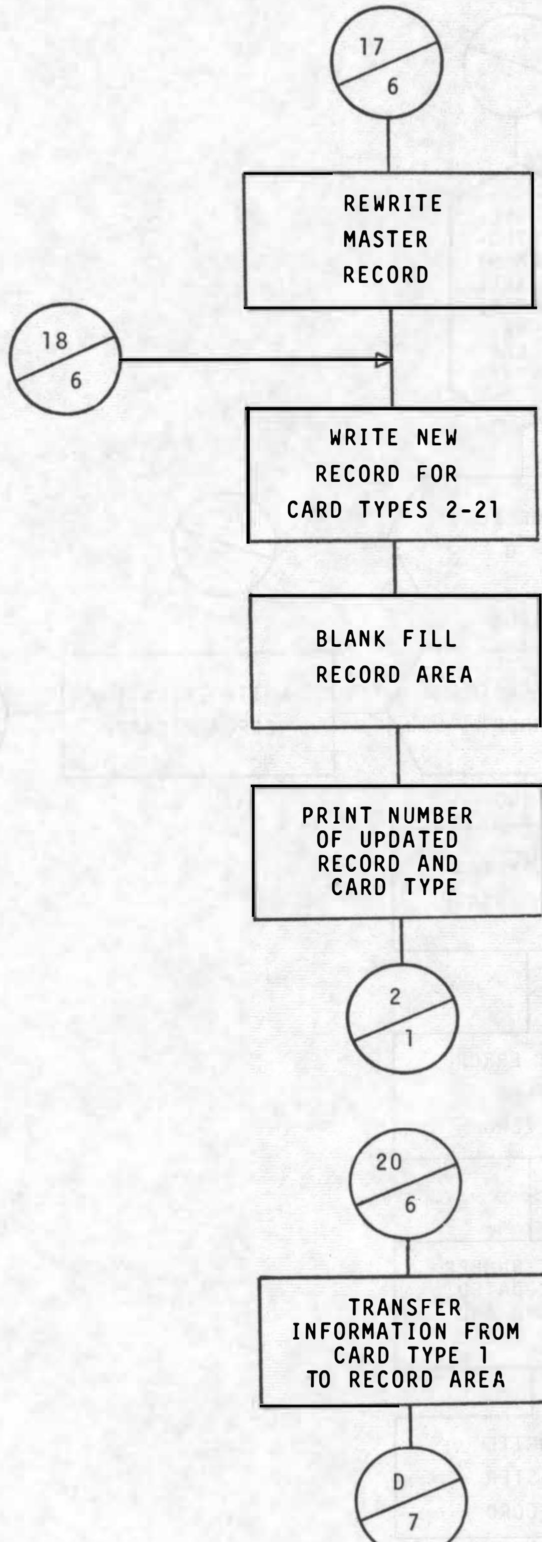


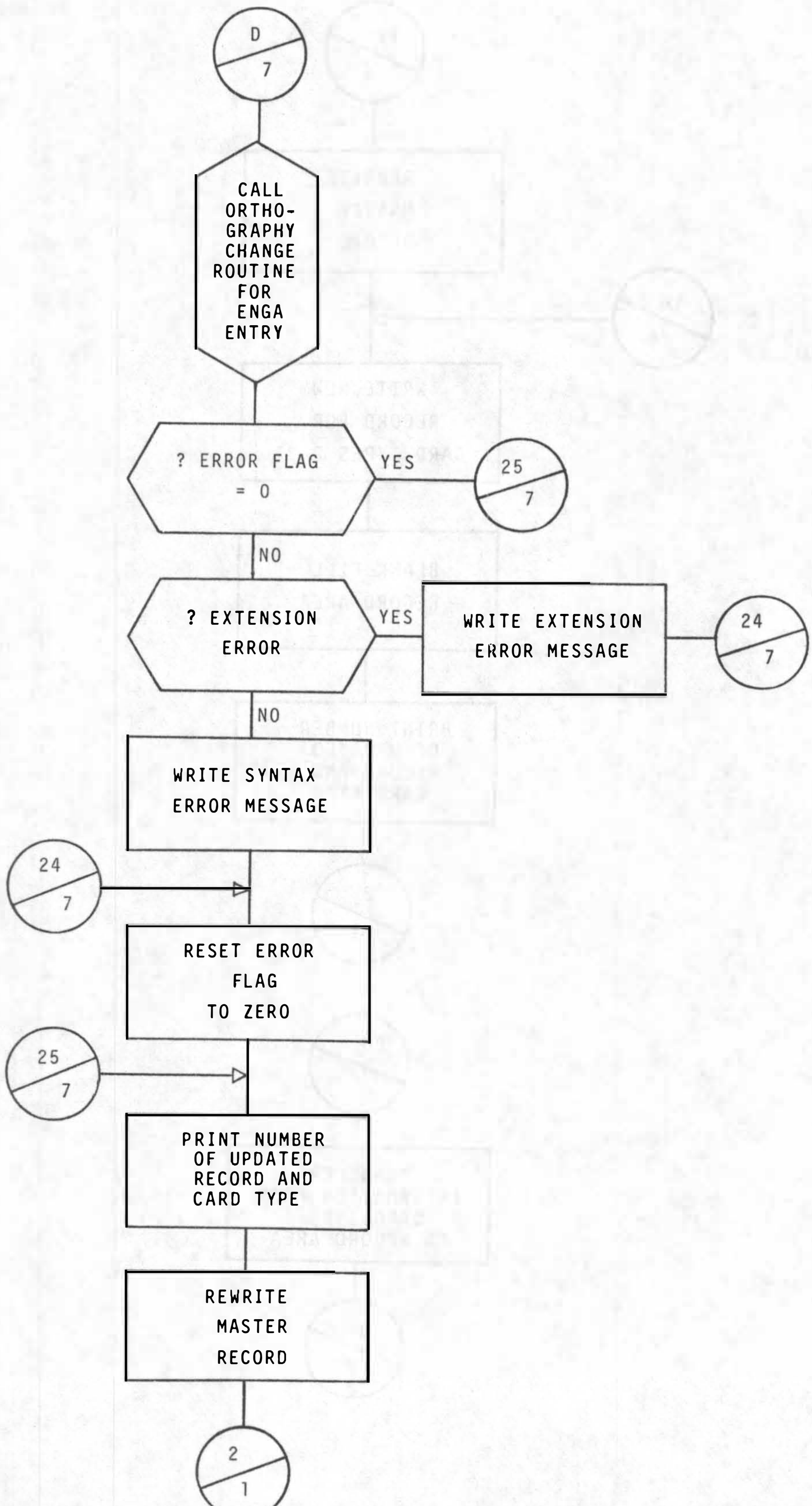

A 2
? UPDATE CARD TYPE 1
YES
20 6
NO
INITIALISE FOR UPDATE OF CARD TYPES 2-21
? UPDATE CARD TYPE 2
NO
8 3
YES
UPDATE CARD TYPE 2
6 2
? RECORD SIZE EXCEEDS 200 WORDS
NO
YES
206 8
? CARD LAST OF THIS TYPE
YES
10 4
NO
? CARD NOT IN CORRECT SEQUENCE
YES
202 8
NO
B 3

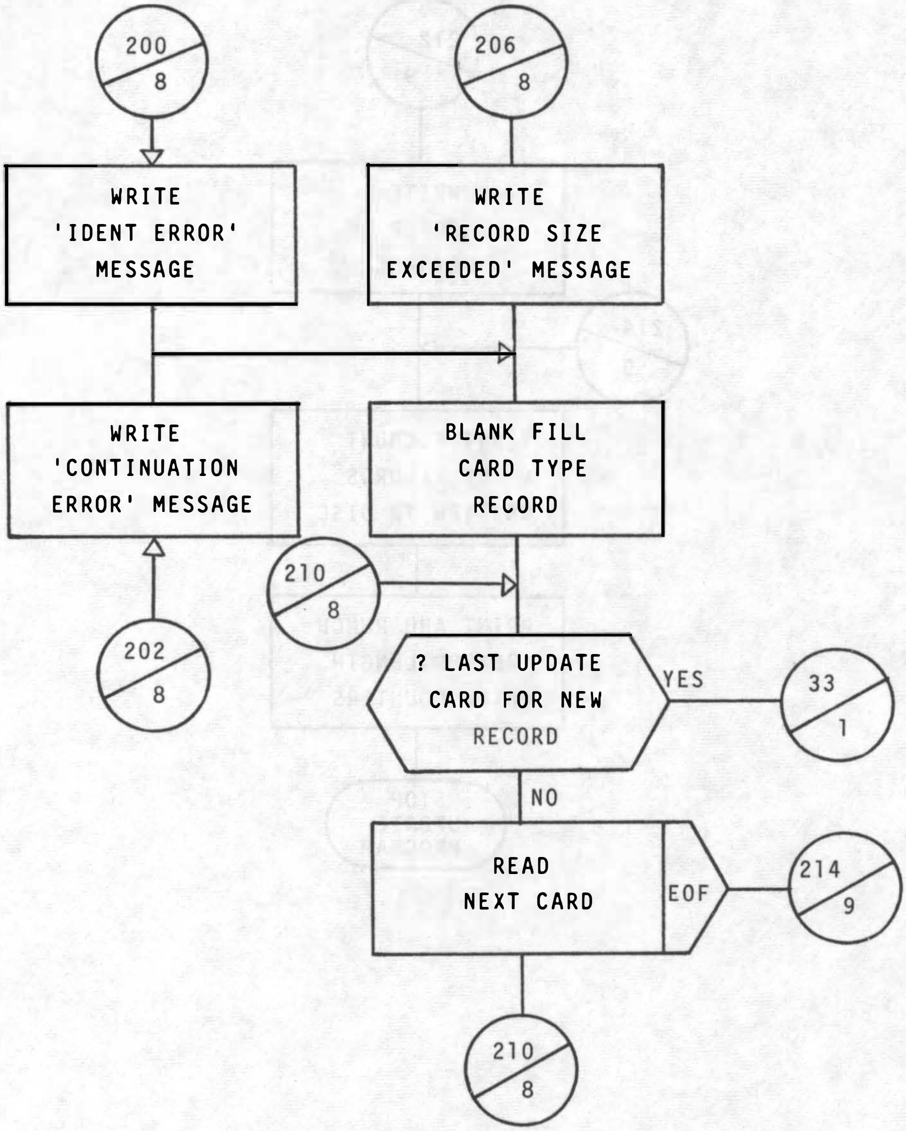

B
3
READ
NEXT CARD
EOF
212
9
? SAME CARD
TYPE AS
PREVIOUS CARD
NO
202
8
YES
? ENTRY NUMBER
SAME AS
MASTER RECORD
NO
200
8
YES
INCREMENT
CARD COUNTER
BY 1
8
3
MOVE CARD DETAILS
TO CARD TYPE RECORD
INCREMENTING RECORD
LENGTH COUNTER
6
2

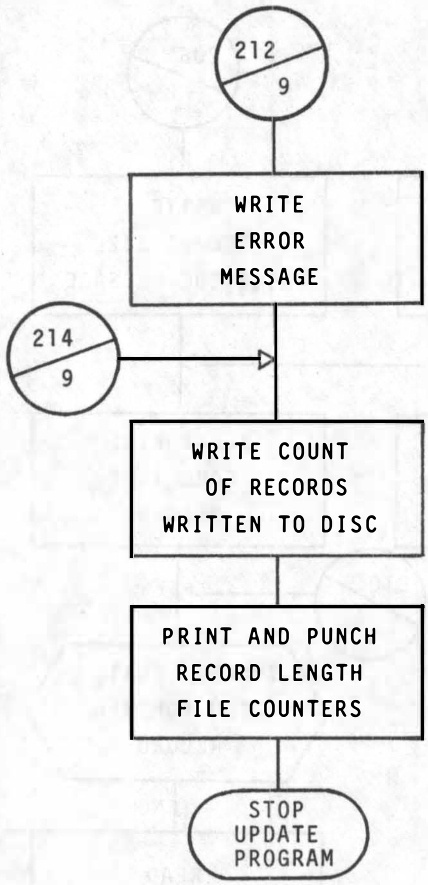

20
4
CHECK LENGTH OF CARD TYPE RECORD
? CARD TYPE = 2
YES
14
5
NO
CALL ORTHOGRAPHY CHANGE ROUTINE FOR CARD TYPES 3-21
? ERROR FLAG = 0
YES
13
5
NO
WRITE ERROR MESSAGE
C
5


C
5
RESET ERROR FLAG TO ZERO
13
5
FIND RECORD LENGTH OF NEW RECORD
14
5
? NEW RECORD TOO LONG FOR EXISTING AREA
NO
? UPDATE CARD TYPE 2
NO
18
6
YES
17
6
YES
CALCULATE NEW ADDRESS
? SPACE IN FILE
NO
WRITE ERROR MESSAGE
212
9
YES
STORE NEW ADDRESS IN MASTER RECORD
17
6


17
6
REWRITE MASTER RECORD
18
6
WRITE NEW RECORD FOR CARD TYPES 2-21
BLANK FILL RECORD AREA
PRINT NUMBER OF UPDATED RECORD AND CARD TYPE
2
1
20
6
TRANSFER INFORMATION FROM CARD TYPE 1 TO RECORD AREA
D
7


D
7
CALL ORTHOGRAPHY CHANGE ROUTINE FOR ENGA ENTRY
? ERROR FLAG = 0
YES
25
7
NO
? EXTENSION ERROR
YES
WRITE EXTENSION ERROR MESSAGE
24
7
NO
WRITE SYNTAX ERROR MESSAGE
24
7
RESET ERROR FLAG TO ZERO
25
7
PRINT NUMBER OF UPDATED RECORD AND CARD TYPE
REWRITE MASTER RECORD
2
1


200
8
206
8
WRITE
'IDENT ERROR'
MESSAGE
WRITE
'RECORD SIZE
EXCEEDED' MESSAGE
WRITE
'CONTINUATION
ERROR' MESSAGE
BLANK FILL
CARD TYPE
RECORD
210
8
202
8
? LAST UPDATE
CARD FOR NEW
RECORD
YES
33
1
NO
READ
NEXT CARD
EOF
214
9
210
8


212
9
WRITE
ERROR
MESSAGE
214
9
WRITE COUNT
OF RECORDS
WRITTEN TO DISC
PRINT AND PUNCH
RECORD LENGTH
FILE COUNTERS
STOP
UPDATE
PROGRAM

```
C
C          CALCULATE NEW ADDRESS AND STORE IN MASTER RECORD
C
   15 JFL(3,LF)=JFL(3,LF)+1
      IF(JFL(3,LF).LE.JFL(2,LF))GO TO 16
      JFL(3,LF)=JFL(3,LF)-1
      WRITE(3,530)LU
  530 FORMAT('1',10X,'NO. OF RECORDS EXCEEDED IN FILE ',I2)
      GO TO 212
   16 IND(1,KREC(2))=LU
      IND(2,KREC(2))=JFL(3,LF)
C
C          REWRITE MASTER RECORD
C
   17 KAA=KY(1)
      WRITE(7'KAA,520)(KY(M),M=1,70)
C
C          WRITE NEW RECORD FOR CARD TYPES 2-21
C
   18 K1=IND(2,KREC(2))
      WRITE(LU'K1,535)(KREC(K),K=1,J)
  535 FORMAT(I6,I2,198A4)
      DO 19 K=11,J
   19 KREC(K)=BL
      KTA=KTA+1
      WRITE(3,540)KTA,KREC(1),KREC(2)
  540 FORMAT(' ',I4,2X,I5,2X,I2)
      GO TO 2
C
C          UPDATE OF CARD TYPE 1
C
   20 READ(99,910)JAJ
  910 FORMAT(36X,I4)
      DO 21 J=3,10
   21 KY(J+1)=KARD(J)
      DO 22 J=11,20
   22 KY(J+2)=KARD(J)
      CALL CSDV(ARRA,ARRDV,25,25)
      CALL XTOPL1(ORTHO,ARRDV,JVB,IER,1)
      IF(IER.EQ.0)GO TO 25
      IF(IER.EQ.2)GO TO 23
      WRITE(3,545)KY(1),(ARRA(J),J=1,7)
  545 FORMAT('0EXTENSION GT 25 - ',I4,4X,6A4,A1)
      GO TO 24
   23 WRITE(3,525)KY(1),KY(2),(ARRA(J),J=1,6)
   24 IER=0
   25 KTA=KTA+1
      KAA=KY(1)
      KAB=1
      WRITE(3,540)KTA,KAA,KAB
      WRITE(7'KAA,520)(KY(J),J=1,70)
      GO TO 2
```

```
C
C           ERROR MESSAGES
C
  200 WRITE(3,201)KARD(1),KARD(2)
  201 FORMAT('0',20X,'IDENT ERROR:',I6,I3)
      GO TO 208
  202 WRITE(3,203)KARD(1),KARD(2),KARD(20)
  203 FORMAT('0',20X,'CONTINUATION ERROR:',I6,I3,A4)
      GO TO 208
  206 WRITE(3,207)KARD(1),KARD(2)
  207 FORMAT('0',20X,'RECORD SIZE GT 200:',I6,I3)
  208 DO 209 J=11,KNT
  209 KREC(J)=BL
  210 IF(KARD(1).NE.KREC(1))GO TO 3
      READ(1,515,END=214)(KARD(J),J=1,20)
      GO TO 210
C
C           END OF RUN PROCEDURES
C
  212 WRITE(3,213)KREC(1),KREC(2)
  213 FORMAT('0',20X,'RECORD NOT WRITTEN:',I6,I3)
  214 WRITE(3,215)KTA
  215 FORMAT('1',10X,I6,' RECORDS FILED TO DISK'/1X)
C
C           WRITE AND PUNCH FILE COUNTERS
C
      DO 216 K=1,20
      WRITE(3,510)(JFL(J,K),J=1,3)
  216 WRITE(2,217)(JFL(J,K),J=1,3)
  217 FORMAT('V',I2,2I8)
      WRITE(3,505)
      STOP
      END
```

```
ORTHO: PROCEDURE(SB,JVB,IFLA,JM);
/* THIS SUBROUTINE OPERATES ON ENGA CARDS EXCEPT CARD TWO (ENGLISH).
   IT PERFORMS FOUR TYPES OF ORTHOGRAPHY CHANGES ON ALL WORDS PRESENTED
   TO IT INCLUDING ANY ENGLISH ON CARDS 3 - 21.
   TYPE 1:    LABIALIZATION.
           ANY CONSONANT + W GOES TO CONSONANT + U
           EG  BWAA -> BUAA
               KWAA -> KUAA   ETC.
   TYPE 2:    VOWEL CLUSTERS.
           LAST TWO VOWELS OF THREE VOWEL SET WHICH ARE NOT IDENTICAL
           ARE CHANGED AS FOLLOWS:
           V2=I -> IY
           V2=U -> UW
           V2=E -> Y
         V2=O -> W
           EG  KAIA -> KAIYA
               KOEA -> KOYA   ETC.
   TYPE 3     VOWEL CLUSTERS IN VERB STEMS:
           IN VERBS IF  V1 V2 V3 PRECEDES -GI OR -GE, AND IF V2 + V3 ARE
```

```
             THE SAME V3 IS REMOVED.
             EG  MAIIGI -> MAIGI
      TYPE 4   PRENASALIZATION.
             IN THE MIDDLE OF A WORD ONLY (NOT AT THE BEGINNING)
             B ->MB
            D -> ND
             G -> NG
             J -> NJ
             EG KADEGE -> KANDENGE
                AJA -> ANJA
      THE SUBROUTINE IS IN THREE SECTIONS.
      SECTION 1  REMOVES BLANKS FROM END OF STRING SUPPLIED.
      SECTION 2 SPLITS OFF EACH ENGA WORD TO PASS ACROSS TO INTERNAL
                SUBROUTINE CHANGE.
      SECTION 3 (SUBROUTINE CHANGE) PERFORMS THE FOUR TYPES OF ORTHOGRAPHY
                CHANGES.
      PARAMETERS
      SB - STRING CONTAINING ENGA WORDS
      JVB = 1 IF MAIN ENTRY IF A VERB
          = 0 OTHERWISE
      IFLA = 0 NO ERRORS IN PROCESSING
           = 1 MAIN ENTRY WOULD EXPAND TO GREATER THAN 25 CHARACTERS
           = 2 TONE CODE HAS COME AFTER A SEMI-VOWEL (Y OR W)
      JM = 1 MAIN ENTRY
         = 0 CARDS 3-21
      OTHER IMPORTANT VARIABLES.
      JPOS - STARTING CHARACTER OF NEXT WORD.
      SA,SD,SR,S INTERMEDIATE STRINGS TO HOLD WORDS TEMPORARILY.
      SUB - USED TO EXAMINE ONE CHARACTER AT A TIME
      ICOUNT - COUNTS NO OF VOWELS FOUND IN SEQUENCE
      L - HOLDS POSITIONS OF V2 AND V3
      FOUND - BRANCH TAKEN IF 3 VOWELS FOUND FOR TYPE 2.
      PROC - BRANCH TAKEN IF WORD ENDS IN GI OR GE FOR TYPE 3.
      R - BRANCH TAKEN IF 3 VOWELS FOUND FOR TYPE 3.                  */
   DCL SB CHAR(800) VARYING,
   IFLA BIN FIXED (31,0),
   (JVB,NWD,JM) BIN FIXED (31,0);
   /*  SECTION 1 - REMOVES BLANKS FROM END OF STRING                  */
   JPOS=1;
   DO I=LENGTH(SB) TO 1 BY -1;
   TEST: IF SUBSTR(SB,I,1)¬=' ' THEN
   /*  SECTION 2 - CHECKS FOR ERROR TYPE 1 AND SPLITS OFF NEXT WORD   */
   (STRINGRANGE): ONE: BEGIN;
   DCL SA CHAR(I),
   SUB CHAR(1),
   SD CHAR(800) VARYING;
   IF JM=1 & I>20 THEN DO;
   IFLA=1;
   RETURN;
   END;
   KOUNT=1;
   SA=SB;
   IF JM=0 THEN SB=(800)' ';
   SB='';
   /*  START PROCESSING AT FIRST NON-BLANK CHARACTER.                  */
   DO I=1 TO LENGTH(SA);
```

```
IF SUBSTR(SA,I,1)¬=' ' THEN DO;
JPOS=I;
SB=SB||SUBSTR(SA,1,I-1);
GO TO LOOP;
END;
END;
/*  SEARCH FOR NEXT WORD TERMINATOR AND CALL SUBROUTINE CHANGE        */
LOOP: DO I=JPOS TO LENGTH(SA);
SUB=SUBSTR(SA,I,1);
J=0;
IF SUB=';' | SUB=',' | SUB=' ' THEN DO;
DO J=1 TO LENGTH(SA)-I;
IF SUBSTR(SA,I+J,1)¬=' ' THEN
GO TO CALL;
END;
J=J-1;
END;
END;
CALL: N=I+J;
CALL CHANGE(SUBSTR(SA,JPOS,I-JPOS),IFLAG,SD);
/* TEST ERROR FLAG - RETURN ORIGINAL STRING IF ERROR. BLANK OUT
     REST OF STRING AND RETURN.                                        */
JPOS=N;
IFLA=IFLAG;
IF IFLAG¬=0 THEN DO;
SB=SA;
RETURN;
END;
SB=SB||SD||SUB;
KOUNT=KOUNT+1;
IF JPOS<=LENGTH(SA) THEN GO TO LOOP;
IF SUBSTR(SB,LENGTH(SB)-1,1)=SUBSTR(SB,LENGTH(SB),1)
THEN SB=SUBSTR(SB,1,LENGTH(SB)-1);
IF JM=1 THEN LEN=25;
ELSE LEN=800;
SD=(800)' ';
SB=SB||SUBSTR(SD,1,(LEN-LENGTH(SB)));
RETURN;
END ONE;
END;
IFLA=2;
RETURN;
CHANGE: PROCEDURE(S,IFL,SR);
DCL S CHAR(*),
SR CHAR(800) VARYING,
SUB CHAR(1),
L(2),
VOWEL(5) CHAR(1) INITIAL('A','E','I','O','U');
/* TYPE 1 CHANGE. SEARCH FOR CONSONANT + W AND CHANGE TO U.            */
IFL=0;
DO I=1 TO LENGTH(S)-1;
SUB=SUBSTR(S,I,1);
IF SUB<'A' | SUB>'Z' THEN GO TO END;
DO J=1 TO 5;
IF SUB=VOWEL(J) THEN GO TO END;
END;
```

```
IF SUBSTR(S,I+1,1)='W' THEN SUBSTR(S,I+1,1)='O';
END: END;
/* TYPE 4 CHANGE. SEARCH FOR B,D,G,J AND INSERT REQUIRED CHARACTER
   BEFORE COPYING STRING INTO INTERMEDIATE STRING SR.                  */
SR=SUBSTR(S,1,1);
DO I=2 TO LENGTH(S);
SUB=SUBSTR(S,I,1);
IF SUB='B' THEN SR=SR||'M';
IF SUB='D'|SUB='G'|SUB='J' THEN SR=SR||'N';
SR=SR||SUB;
END;
/* TYPE 2 CHANGE. SEARCH BACKWARDS THROUGH WORD FOR THREE VOWELS IN A
   ROW OR SEPARATED ONLY BY TONES. STORE POSITIONS OF THESE THREE.     */
I=LENGTH(SR);
ICOUNT=0;
LOOP: IF I<3 THEN GO TO NXT;
DO J=I TO I-10 BY -1;
IF J<1 THEN GO TO NXT;
SUB=SUBSTR(SR,J,1);
IF SUB='''' THEN GO TO ROUND;
DO K=1 TO 5;
IF SUB=VOWEL(K) THEN DO;
ICOUNT=ICOUNT+1;
IF ICOUNT=3 THEN GO TO FOUND;
L(ICOUNT)=J;
GO TO ROUND;
END;
END;
GO TO FIX;
ROUND: END;
FIX: ICOUNT=0;
I=I-1;
GO TO LOOP;
/* CHECK FINAL TWO VOWELS ARE NON-IDENTICAL AND THEN PERFORM PRESCRIB-
   ED CHANGES PAYING SPECIAL ATTENTION TO TONE CODES.                  */
FOUND: IF SUBSTR(SR,L(2),1)=SUBSTR(SR,L(1),1) THEN GO TO FIX;
SUB=SUBSTR(SR,L(2),1);
IF SUB='A' THEN GO TO FIX;
IF SUB='E' THEN DO;
SUBSTR(SR,L(2),1)='Y';
IF SUBSTR(SR,L(2)+1,1)='''' THEN SR=SUBSTR(SR,1,J    )||''''||'Y'||
SUBSTR(SR,L(2)+2);
GO TO FIX;
END;
IF SUB='U' THEN DO;
SUBSTR(SR,L(2),1)='W';
IF SUBSTR(SR,L(2)+1,1)='''' THEN SR=SUBSTR(SR,1,J    )||''''||'W'||
SUBSTR(SR,L(2)+2);
GO TO FIX;
END;
IF SUBSTR(SR,L(2)+1,1)='''' THEN L(2)=L(2)+1;
IF SUB='I' THEN DO;
SR=SUBSTR(SR,1,L(2))||'Y'||SUBSTR(SR,L(2)+1);
I=L(2);
ICOUNT=0;
GO TO LOOP;
END;
```

```
IF SUB='U' THEN DO;
SR=SUBSTR(SR,1,L(2))||'W'||SUBSTR(SR,L(2)+1);
I=L(2);
ICOUNT=0;
GO TO LOOP;
END;
/* TYPE 3 CHANGE. CHECK FOR VERB AND LAST SYLLABLE BEING GI OR GE.      */
NXT: IF JVB=0 THEN GO TO RET;
LN=LENGTH(SR);
IF SUBSTR(SR,LN-3)='NGI'''|SUBSTR(SR,LN-3)='NGE''' THEN DO;
KK=LN-4;
GO TO PROC;
END;
IF SUBSTR(SR,LN-2)='NGI'|SUBSTR(SR,LN-2)='NGE' THEN DO;
KK=LN-3;
GO TO PROC;
END;
ELSE GO TO RET;
/* SEARCH BACKWARDS THROUGH WORD FOR THREE VOWELS                        */
PROC: ICOUNT=0;
DO I=KK TO KK-10 BY -1;
IF I<1 THEN GO TO RET;
SUB=SUBSTR(SR,I,1);
IF SUB='''' THEN GO TO ND;
DO J=1 TO 5;
IF SUB=VOWEL(J) THEN DO;
ICOUNT=ICOUNT+1;
IF ICOUNT=3 THEN GO TO R;
L(ICOUNT)=I;
GO TO ND;
END;
END;
ICOUNT=0;
ND: END;
GO TO RET;
/* CHECK IF FINAL TWO EQUAL AND DELETE LAST ONE.                         */
R: IF SUBSTR(SR,L(2),1)¬=SUBSTR(SR,L(1),1)THEN GO TO RET;
SR=SUBSTR(SR,1,L(1)-1)||SUBSTR(SR,L(1)+1);
RET: IF JM=1 & LENGTH(SR)>25 THEN IFL=1;
DO I=1 TO LENGTH(SR)-1;
IF SUBSTR(SR,I,1)='Y'|SUBSTR(SR,I,1)='W' THEN
IF SUBSTR(SR,I+1,1)='''' THEN DO;
IFL=2;
RETURN;
END;
END;
RETURN;
END CHANGE;
END ORTHO;
```

3.0 FORTRAN RETRIEVAL TECHNIQUES, by Katharine E.W. Mather

The retrieval technique was essentially the same for all the required printouts. The whole of the master file was read, sequentially if a count was being performed, or in sort order for all listings. If a record satisfied the retrieval criterion being applied (e.g. no tones in Enga item or no English gloss), the Enga main item and any other necessary information from the master record were printed. If cross references or other card types were required to be printed, the master file record was checked to see if these records existed. If so, their addresses were picked up from the master record, and they were read and printed.

3.1 *SINGLE-PHASE RETRIEVALS*

Two basic types of retrieval problems existed. In some cases one sub-group only had to be selected and printed according to the values in a particular field or fields. Examples of this single phase retrieval are the lists of words with no tones, no English gloss, no existential verb or no grammatical class. For these the master file was read and the required information printed.

3.2 *MULTI-PHASE RETRIEVALS*

The second type of retrieval problem involved those cases in which a large number of sub-groups existed within the file. There were, for instance, thirty complex verb sub-groups and over eighty semantic domain sub-groups.

To produce listings of the groups without reading the master file again and again checking the same field each time, the master file was read only once and lists of addresses of words of each sub-group were produced. These lists were then stored in direct access disk files.

A second program was used to read and print the information for any or all sub-groups. The list of addresses for the required group was first read from the disk file. Then the master record for each entry and any associated items required were read and printed as described above (3.0).

3.3 *LISTINGS*

3.3.1 Incomplete Item Program: No Card 21

```
C           PRINT OF ITEMS HAVING NO CARD 21 - QUOTATION
C
      DIMENSION KY(9),JR(36)
      INTEGER*4 BL/'    '/
      DEFINE FILE  7(5500,265,E,KAA),13(11800,40,F,K1),14(5250,80,E,K1),
     115(1520,120,E,K1),16(0960,160,E,K1)
C
C           INITIALIZE COUNTER  -  PRINT HEADING
C
      KTR=0
      WRITE(3,500)
  500 FORMAT('1'/' WORDS HAVING NO CARD 21 - QUOTATION'/)
C
C           READ ADDRESS OF FIRST ENTRY IN SORT ORDER
C
      READ(13'1,510)KAA
  510 FORMAT(I6)
C
C           READ MASTER RECORD
C
C           NWA     ADDRESS OF NEXT ENTRY IN SORT ORDER
C           KY      ENGA ENTRY, EXISTENTIAL VERB, GRAMMATICAL CLASS
C           LU,K1   ADDRESS OF ENGLISH GLOSS
C           NQ      ADDRESS OF QUOTATION
C
    1 READ(7'KAA,530)NWA,(KY(J),J=1,9),LU,K1,NQ
  530 FORMAT(6X,I5,6A4,A3,55X,A4,A3,25X,I2,I5,126X,I2)
      IF(NQ.EQ.0)GO TO 3
C
C           CHECK FOR END OF FILE  -  GO BACK TO READ NEXT RECORD
C
      IF(NWA.EQ.99999)GO TO 6
      KAA=NWA
      GO TO 1
C
C           ADD TO COUNTER  -  READ ENGLISH GLOSS
C
    3 KTR=KTR+1
      LF=(LU-12)*10-4
      READ(LU'K1,540)(JR(J),J=1,LF)
  540 FORMAT(16X,36A4)
C
C           PRINT RECORD
C
      IF(LF.GT.22.AND.JR(23).NE.BL)GO TO 4
      LZ=22
      IF(LF.LT.LZ)LZ=LF
      WRITE(3,550)(KY(J),J=1,9),(JR(J),J=1,LZ)
  550 FORMAT('0',31A4)
      IF(NWA.EQ.99999)GO TO 6
      KAA=NWA
      GO TO 1
C
    4 WRITE(3,550)(KY(J),J=1,9),(JR(J),J=1,22)
      WRITE(3,560)(JR(J),J=23,LF)
  560 FORMAT(5X,30A4)
      IF(NWA.EQ.99999)GO TO 6
      KAA=NWA
      GO TO 1
C
```

```
C          END OF FILE  -  PRINT COUNTER
C
    6 WRITE(3,570)KTR
  570 FORMAT('0'/'0',9X,'NUMBER OF WORDS HAVING NO CARD 21 - QUOTATION =
     1 ',I4)
      WRITE(3,500)
      STOP
      END
```

3.3.2 Single Phase Retrieval: Noun and Noun Phrase

```
C          PRINT OF NOUNS AND NOUN PHRASES
C
      DIMENSION KY(8),JR(59)
      INTEGER*4 BL/'    '/,TH/'   3'/
      DEFINE FILE   7(5500,265,E,KAA),13(11800,40,E,K1),14(5250,80,E,K1),
     115(1520,120,E,K1),16(0960,160,E,K1),17(0390,200,E,K1),18(0330,240,
     2E,K1)
C
      CALL ERRSET(215,0,256,1)
      DO 7 NN=1,9,8
C
C          INITIALIZE COUNTER  -  PRINT HEADING
C
      KTR=0
      IF(NN.EQ.1)WRITE(3,500)
  500 FORMAT('1'/' NOUNS'/)
      IF(NN.EQ.9)WRITE(3,505)
  505 FORMAT('1'/' NOUN PHRASES'/)
C
C          READ ADDRESS OF FIRST ENTRY IN SORT ORDER
C
      READ(13'1,510)KAA
  510 FORMAT(I6)
C
C          READ MASTER RECORD
C          NWA        ADDRESS OF NEXT ENTRY IN SORT ORDER
C          KY         ENGA ENTRY, SEMANTIC DOMAIN
C          NDR,NPN    GRAMMATICAL CLASS
C          LU,K1      ADDRESS OF ENGLISH GLOSS
C          LUS,KIS    ADDRESS OF CROSS REFERENCES
C
    1 READ(7'KAA,530)NWA,(KY(J),J=1,8),NDR,NPN,LU,K1,LUS,KIS
  530 FORMAT(6X,I5,6A4,A3,1X,A4,50X,I3,I1,28X,2(I2,I5))
C
C          CHECK IF NOUN, NOUN PHRASE, OR PROPER NOUN
C
      IF(NDR.NE.NN)GO TO 2
      IF(NPN.NE.1)GO TO 3
C
C          CHECK FOR END OF FILE  -  GO BACK TO READ NEXT RECORD
C
    2 IF(NWA.EQ.99999)GO TO 6
      KAA=NWA
      GO TO 1
C
C          ADD TO COUNTER  -  READ ENGLISH GLOSS AND PRINT RECORD
C
    3 KTR=KTR+1
      LF=(LU-12)*10-3
      READ(LU'K1,540)(JR(J),J=2,LF)
  540 FORMAT(16X,36A4)
      IF(LF.GT.23.AND.JR(24).NE.BL)GO TO 4
      LZ=23
      IF(LF.LT.LZ)LZ=LF
      WRITE(3,550)(KY(J),J=1,8),(JR(J),J=2,LZ)
  550 FORMAT('0',7A4,1X,A4,2X,22A4)
      GO TO 5
C
    4 WRITE(3,550)(KY(J),J=1,8),(JR(J),J=2,23)
      WRITE(3,560)(JR(J),J=24,LF)
```

```
  560 FORMAT(5X,30A4)
C
C         READ AND PRINT CROSS REFERENCES IF THEY EXIST
C
    5 IF(LUS.EQ.0)GO TO 2
      KI=KIS
      JF=(LUS-12)*10-1
      READ(LUS'KI,520)(JR(J),J=2,JF)
  520 FORMAT(8X,58A4)
      JR(1)=TH
      WRITE(3,560)(JR(J),J=1,JF)
      IF(NWA.EQ.99999)GO TO 6
      KAA=NWA
      GO TO 1
C
C         END OF FILE  -  PRINT COUNTER
C
    6 IF(NN.EQ.1)WRITE(3,570)KTR
  570 FORMAT('0'/'0',9X,'NUMBER OF NOUNS = ',I4)
      IF(NN.EQ.9)WRITE(3,580)KTR
  580 FORMAT('0'/'0',9X,'NUMBER OF NOUN PHRASES = ',I4)
    7 CONTINUE
      WRITE(3,500)
      STOP
      END
```

2.3.3 Listing

```
C        UPDATE PROGRAM
C
      DIMENSION KY(70),KARD(20),JFL(3,20),IND(2,21)
      DIMENSION ARRDV(2),ARRA(7),ARRB(200)
      INTEGER*4 BL/'    '/,KREC(209)/209*'    '/,CT(11)/'   *   1   2
     13   4   5   6   7   8   9  10'/
      EQUIVALENCE(KY(31),IND(1,2)),(KY(4),ARRA(1)),(KREC(3),ARRB(1))
      EXTERNAL ORTHO
      DEFINE FILE  7(0340,265,E,KAA),13(00590,40,E,K1),14(0290,80,E,K1),
     115(0280,120,E,K1),16(0130,160,E,K1),17(0110,200,E,K1),18(0044,240,
     2E,K1),19(0025,280,E,K1),20(0024,320,E,K1),21(0024,360,E,K1),22(002
     31,400,E,K1),23(0018,440,E,K1),24(0018,480,E,K1),25(0015,520,E,K1),
     426(0015,560,E,K1),27(0015,600,E,K1),28(0012,640,E,K1),29(0012,680,
     5E,K1),30(0012,720,E,K1),31(0012,760,E,K1),32(0009,800,E,K1)
C
C        READ AND WRITE FILE COUNTERS
C
      READ(1,500)((JFL(J,K),J=1,3),K=1,20)
  500 FORMAT(I2,2I8)
      WRITE(3,505)
  505 FORMAT('1')
      DO 1 K=1,20
      WRITE(3,510)(JFL(J,K),J=1,3)
  510 FORMAT(11X,I2,2I8)
      IF(JFL(1,K).NE.(K+12))STOP
    1 CONTINUE
C
C        INITIALIZE
C
      CALL ERRSET(215,0,256,1)
      CALL REREAD
      WRITE(3,505)
      IER=0
      JVB=0
      KTA=0
C
C        READ FIRST OF GROUP OF UPDATE CARDS
C
    2 READ(1,515)(KARD(J),J=1,20)
  515 FORMAT(I6,I2,18A4)
    3 IF(KARD(1).EQ.999999)GO TO 214
C
C        READ MASTER RECORD OF ENTRY TO BE UPDATED
C
      KAA=KARD(1)
      READ(7'KAA,520)(KY(J),J=1,70)
  520 FORMAT(I5,I1,I5,8A4,I10,18A4,20(I2,I5))
C
C        INITIALIZE FOR CARD TYPES 2-21
C
      IF(KARD(2).EQ.1)GO TO 20
      KREC(1)=KARD(1)
      KREC(2)=KARD(2)
      KNT=2
      KR=1
```

```
C
C        UPDATE OF CARD TYPE 2
C
      IF(KARD(2).NE.2)GO TO 8
      KREC(3)=BL
      KREC(4)=BL
      KNT=13
      READ(99,900)JAJ
  900 FORMAT(8X,I7)
      DO 4 J=3,10
    4 KY(J+20)=KARD(J)
      DO 5 J=11,19
    5 KREC(J-6)=KARD(J)
      GO TO 7
C
C        READ AND PROCESS REMAINDER OF GROUP OF UPDATE CARDS
C
    6 IF(KREC(201).NE.BL)GO TO 206
    7 IF(KARD(20).EQ.BL)GO TO 10
      IF(KARD(20).NE.CT(KR))GO TO 202
      READ(1,515,END=212)(KARD(J),J=1,20)
      IF(KARD(2).NE.KREC(2))GO TO 202
      IF(KARD(1).NE.KY(1))GO TO 200
      KR=KR+1
    8 DO 9 J=3,19
      KNT=KNT+1
    9 KREC(KNT)=KARD(J)
      GO TO 6
C
C        CHECK LENGTH OF RECORD FOR CARD TYPES 2-21
C
   10 DO 11 J=11,201,10
      IF(KREC(J).EQ.BL)GO TO 12
   11 CONTINUE
   12 J=J-1
C
C        CALL ORTHOGRAPHY CHANGE ROUTINE FOR CARD TYPES 3-21
C
      IF(KREC(2).EQ.2)GO TO 14
      J=J*4-8
      CALL CSDV(ARRB,ARRDV,B00,J)
      CALL XTOPL1(ORTHO,ARRDV,C,IER,0)
      J=J/4
      IF(IER.EQ.0)GO TO 13
      WRITE(3,525)KREC(1),KREC(2),(ARRB(K),K=1,J)
  525 FORMAT('0SYNTAX ERROR - ',I4,I2,4X,25A4/(2X,31A4))
      IER=0
   13 IF(KREC(J).EQ.BL)GO TO 14
      J=J+10
      GO TO 13
C
C        CHECK IF NEW RECORD TOO LONG FOR EXISTING AREA
C
   14 LF=J/10+1
      LU=LF+12
      IF(IND(1,KREC(2)).LT.LU)GO TO 15
      IF(KREC(2).GT.2)GO TO 18
      GO TO 17
```

3.3.3 Multi Phase Retrieval: Semantic Domains

```
C          CREATE FILE OF ADDRESSES OF ITEMS BELONGING TO EACH SEMANTIC DOMAIN
C
      INTEGER*2 JK(425,100),KT(100)
      DEFINE FILE   7(5500,265,E,KAA),13(11800,40,E,K1)
      DEFINE FILE 8(100,1700,E,KB)
C
C          INITIALIZE FILE AREA AND RECORD COUNTERS
C
      CALL ERRSET(215,0,256,1)
      DO 10 K=1,100
      DO 9 J=1,425
    9 JK(J,K)=0
   10 KT(K)=0
      WRITE(3,500)
  500 FORMAT('1')
C
C          READ ADDRESS OF FIRST ENTRY IN SORT ORDER
C
      READ(13'1,510)KAA
  510 FORMAT(I6)
C
C          READ MASTER RECORD
C          NWD  ADDRESS OF ENTRY
C          NWA  ADDRESS OF NEXT ENTRY IN SORT ORDER
C          KC   SEMANTIC DOMAIN CODE
C
    1 READ(7'KAA,530)NWD,NWA,KC
  530 FORMAT(1X,I4,1X,I5,28X,I4)
C
C          CHECK VALIDITY OF SEMANTIC DOMAIN CODE
C
      IF(KC.EQ.0)GO TO 4
      IF(KC.LT.100)GO TO 8
      WRITE(3,535)KC,NWD
  535 FORMAT('0      KC = ',I4,'      NWD = ',I4)
      GO TO 4
C
C          STORE ADDRESS OF ENTRY IN APPROPRIATE RECORD
C
    8 KT(KC)=KT(KC)+1
      IF(KT(KC).GT.425)GO TO 2
      JK(KT(KC),KC)=NWD
    4 IF(NWA.EQ.99999)GO TO 3
      KAA=NWA
      GO TO 1
C
C          ERROR MESSAGE
C
    2 WRITE(3,540)KC
  540 FORMAT('0',9X,'425 EXCEEDED FOR CODE ',I2/'1')
      STOP
C
C          WRITE FILE TO DISK
C
    3 WRITE(8'1,550)((JK(J,K),J=1,425),K=1,100)
  550 FORMAT(200I4,200I4,25I4)
C
C          PRINT COUNTS OF NUMBER OF ADDRESSES IN EACH RECORD
C
      WRITE(3,500)
```

```
      DO 7 K=1,99
      WRITE(3,560)K
  560 FORMAT('0 CODE ',I2/1X)
      DO 5 J=1,425
      IF(JK(J,K).EQ.0)GO TO 6
      WRITE(3,570)JK(J,K)
  570 FORMAT(10X,I4)
    5 CONTINUE
    6 J=J-1
      WRITE(3,580)J
  580 FORMAT('0',9X,'N = ',I3/'0')
    7 CONTINUE
      WRITE(3,500)
      STOP
      END
```

```
C          PRINT OF ITEMS BELONGING TO EACH SEMANTIC DOMAIN
C          USING FILES OF ADDRESSES CREATED IN A PRIOR RUN
C
      INTEGER*4 NR(20)/'    2    3    4    5    6    7    8    9   10   11   12   13
     1   14   15   16   17   18   19   20   21'/,BL/'    '/
      DIMENSION KY(12),JR(1501),NF(2,20),JK(425),KED(10)
      DEFINE FILE  7(5500,265,E,KAA),13(11800,40,E,K1),14(5250,80,E,K1),
     115(1520,120,E,K1),16(0960,160,E,K1),17(0390,200,E,K1),18(0330,240,
     2E,K1),19(0200,280,E,K1),20(0180,320,E,K1),21(0080,360,E,K1),22(007
     30,400,E,K1),23(0070,440,E,K1),24(0060,480,E,K1),25(0060,520,E,K1),
     426(0050,560,E,K1),27(0025,600,E,K1),28(0025,640,E,K1),29(0020,680,
     5E,K1),30(0020,720,E,K1),31(0020,760,E,K1),32(0020,800,E,K1)
      DEFINE FILE 8(100,1700,E,KB)
C
C          READ CONTROL CARD
C          PRINT INDICATOR (JTY), SEMANTIC DOMAIN CODE (KB), AND HEADING
C
      WRITE(3,500)
  500 FORMAT('1')
    1 READ(1,505)JTY,KB,(KED(J),J=1,10)
  505 FORMAT(I1,I2,10A4)
C
C          PRINT SEMANTIC DOMAIN CODE AND HEADING
C
      WRITE(3,510)KB,(KED(J),J=1,10)
  510 FORMAT(I3,3X,10A4)
C
C          READ RECORD CONTAINING ADDRESSES OF ENTRIES BELONGING TO
C          THE REQUIRED SEMANTIC DOMAIN
C
      READ(8'KB,515)(JK(J),J=1,425)
  515 FORMAT(200I4,200I4,25I4)
C
C          SET COUNTER TO ZERO
C
      NC=0
      GO TO(2,9),JTY
C
C          PRINT REQUIRED OF ALL INFORMATION ASSOCIATED WITH EACH ENTRY
C
C          PICK UP ADDRESS OF NEXT ENTRY  -  READ MASTER RECORD
C
    2 NC=NC+1
      IF(JK(NC).EQ.0)GO TO 13
      KAA=JK(NC)
      READ(7'KAA,520)KY(12),(KY(J),J=1,7),KY(11),(KY(J),J=8,10),((NF(J,K
     1),J=1,2),K=1,20)
  520 FORMAT(I5,6X,8A4,50X,A4,A3,A2,23X,20(I2,I5))
C
C          READ ENGLISH GLOSS
C          PRINT INFORMATION FROM MASTER RECORD AND ENGLISH GLOSS
C
      LU=NF(1,1)
      K1=NF(2,1)
      FIND(LU'K1)
      LF=(LU-12)*10-4
      READ(LU'K1,525)(JR(J),J=1,LF)
  525 FORMAT(16X,36A4)
      LF=LF+1
    3 LF=LF-1
```

```
      IF(JR(LF).EQ.BL)GO TO 3
      WRITE(3,540)(KY(J),J=1,12),(JR(J),J=1,LF)
  540 FORMAT('0',7A4,1X,2A4,A3,A4,I5,'   2',18A4/5X,30A4)
C
C        READ AND PRINT ALL ASSOCIATED INFORMATION
C
      DO 4 KK=2,20
      IF(NF(1,KK).NE.0)GO TO 5
    4 CONTINUE
      GO TO 2
C
    5 JF=0
      LU=NF(1,KK)
      K1=NF(2,KK)
      FIND(LU'K1)
      IF(KK.EQ.20)GO TO 7
      LL=KK+1
      DO 6 JJ=LL,20
      IF(NF(1,JJ).EQ.0)GO TO 6
      JR(JF+1)=NR(KK)
      JG=JF+2
      JF=JF+(LU-12)*10-1
      READ(LU'K1,530)(JR(J),J=JG,JF)
  530 FORMAT(8X,198A4)
      LU=NF(1,JJ)
      K1=NF(2,JJ)
      FIND(LU'K1)
      KK=JJ
    6 CONTINUE
C
    7 JR(JF+1)=NR(KK)
      JG=JF+2
      JF=JF+(LU-12)*10-1
      READ(LU'K1,530)(JR(J),J=JG,JF)
      JF=JF+1
    8 JF=JF-1
      IF(JR(JF).EQ.BL)GO TO 8
      WRITE(3,550)(JR(J),J=1,JF)
  550 FORMAT(5X,30A4)
      GO TO 2
C
C        PRINT REQUIRED OF CROSS REFERENCES ONLY
C
C        PICK UP ADDRESS OF NEXT ENTRY  -  READ MASTER RECORD
C
    9 NC=NC+1
      IF(JK(NC).EQ.0)GO TO 13
      KAA=JK(NC)
      READ(7'KAA,560)(KY(J),J=1,8),LU,K1,NF(1,2),NF(2,2)
  560 FORMAT(11X,6A4,A3,1X,A4,82X,2(I2,I5))
C
C        READ ENGLISH GLOSS
C        PRINT INFORMATION FROM MASTER RECORD AND ENGLISH GLOSS
C
      LF=(LU-12)*10-4
      READ(LU'K1,525)(JR(J),J=1,LF)
      LF=LF+1
   10 LF=LF-1
      IF(JR(LF).EQ.BL)GO TO 10
      WRITE(3,570)(KY(J),J=1,8),(JR(J),J=1,LF)
```

```
  570 FORMAT('0',8A4,'    2',22A4/5X,4A4)
C
C          READ AND PRINT CROSS REFERENCES IF THEY EXIST
C
      IF(NF(1,2).EQ.0)GO TO 9
      LU=NF(1,2)
      K1=NF(2,2)
      JF=(LU-12)*10-1
      READ(LU'K1,530)(JR(J),J=2,JF)
      JR(1)=NR(2)
      JF=JF+1
   12 JF=JF-1
      IF(JR(JF).EQ.BL)GO TO 12
      WRITE(3,550)(JR(J),J=1,JF)
      GO TO 9
C
C          END OF FILE  -  PRINT COUNT OF NUMBER OF ENTRIES
C          BELONGING TO THE SEMANTIC DOMAIN
C
   13 NC=NC-1
      KB=KB-1
      WRITE(3,580)KB,(KED(J),J=1,10),NC
  580 FORMAT('0'/I3,3X,10A4/'ON = ',I4/'1')
C
C          GO BACK TO READ ANOTHER CONTROL CARD
C
      GO TO 1
      END
```

4.0 PL/1 RETRIEVAL TECHNIQUES, by Mary L. Rose

This section deals with those programs which required manipulation of characters, and as this was necessary for any problem which required the reordering of information, most of the programs described in this section involve sorting.

4.1 *SORT OF MASTER FILE*

4.1.1 Description of Program

A threaded list sort/merge was used to create pointers to the next item in sort order. These values were then inserted into the main file (2.1.6) to be used as the direct access keys for accessing the file in order. The address of the first word was stored in the first word of the 40 character length file.

This program also used the position of the tones as part of the sort key, so that positions of the tones within words of like orthography would influence their order.

A maximum of five tones was allowed for in any one Enga item, so five consecutive fields of two characters each were set up to contain the character position of each tone in the item. Table 4.1 illustrates the numbers created and the order into which the items would sort.

The sort key consisted of the Enga item with tones removed plus ten numeric characters containing the position of each tone.

This sort technique was used in other programs where direct access print files were created (**Reversal 4.3.0** and **Tone Patterns 4.5.0**); it was also used in the programs which checked the cross references with the main item (4.2.0).

See Table 4.1 overleaf.

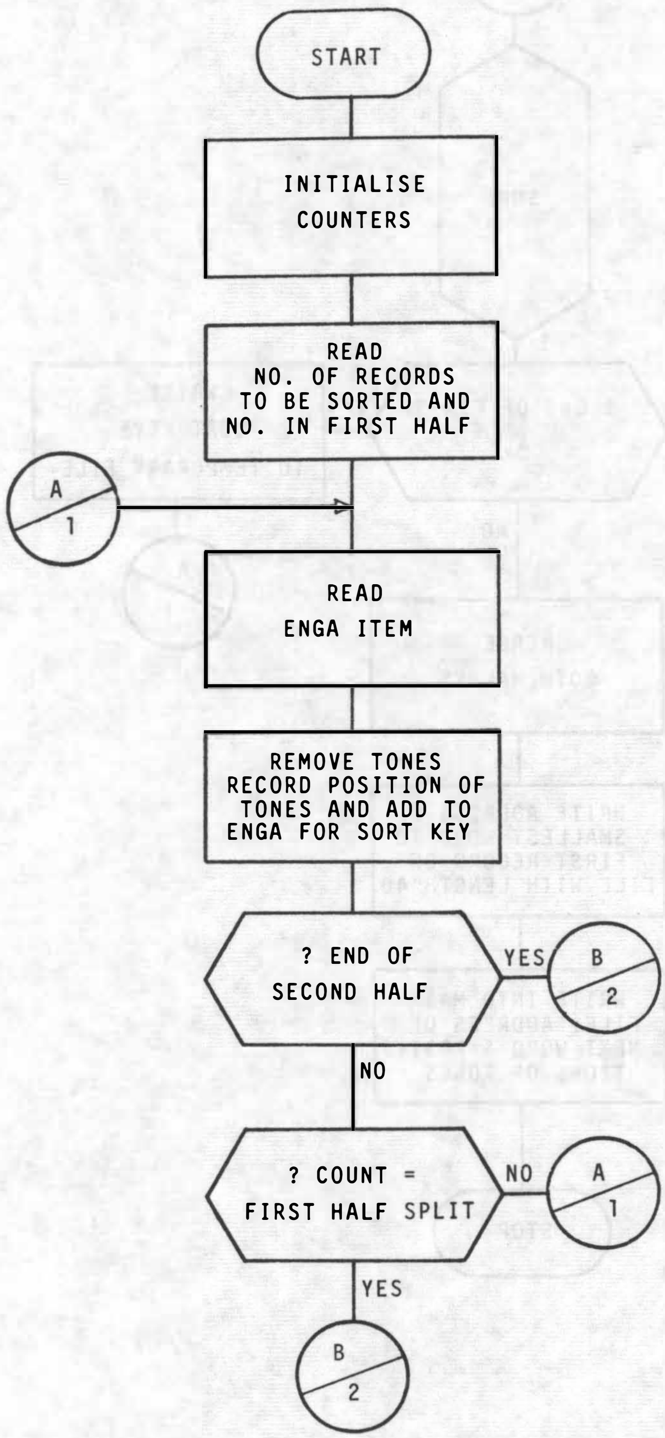
Table 4.1 Numeric Representation of Tone Position

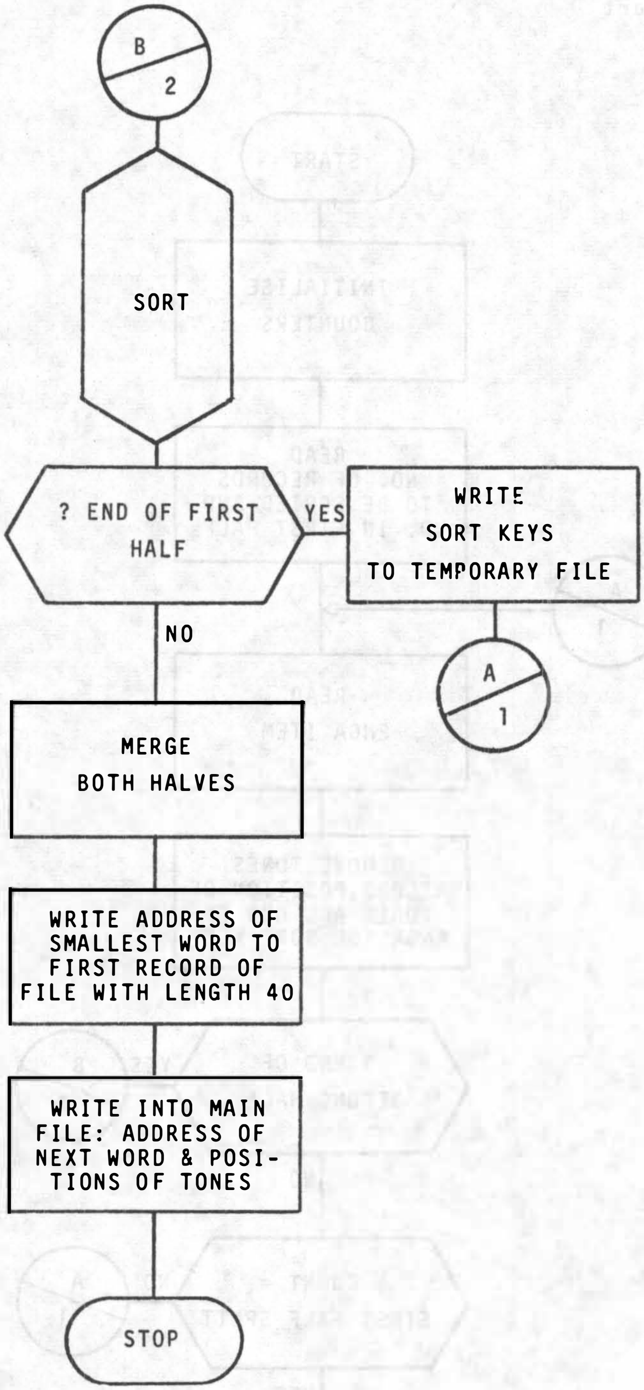
Word	*Tone Position*
A B C	0
A ' B C	2
A B ' C	3
A B C '	4
A ' B ' C	402
A ' B C '	502
A B ' C '	503
A ' B ' C '	60402

In all uses of this sort routine, the arrays for the sort keys and the addresses could not both be kept in core store together, so the addresses and half the sort keys were held in core while the other half were held in a temporary disk file. The direct access facility then made it easy to make a final adjustment of the addresses when merging them.

The flow charts for this program and the sort routine are in 4.1.2 and the listings are in 4.1.3.

4.1.2 Flowchart



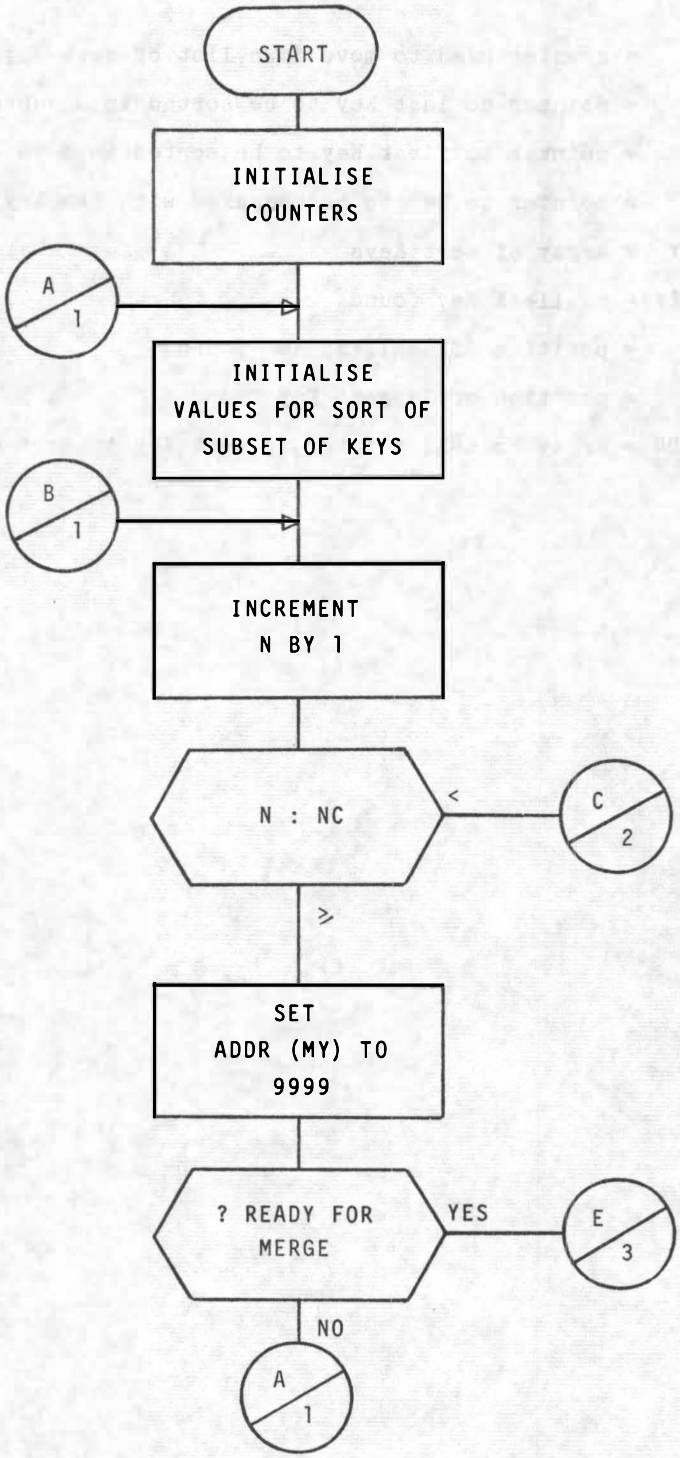

B
2
SORT
? END OF FIRST HALF
YES
WRITE SORT KEYS TO TEMPORARY FILE
A
1
NO
MERGE BOTH HALVES
WRITE ADDRESS OF SMALLEST WORD TO FIRST RECORD OF FILE WITH LENGTH 40
WRITE INTO MAIN FILE: ADDRESS OF NEXT WORD & POSITIONS OF TONES
STOP

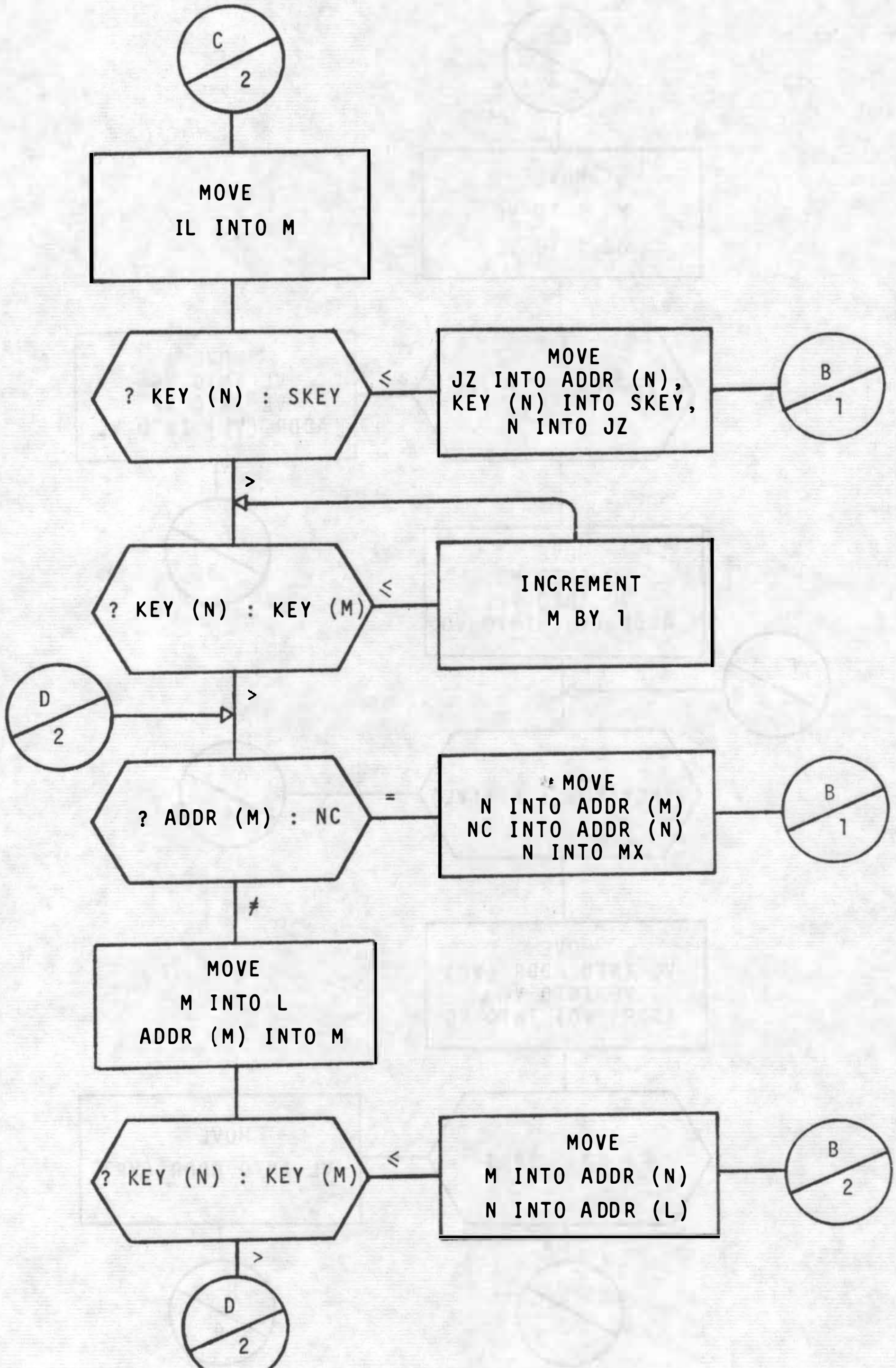
SORT ROUTINE

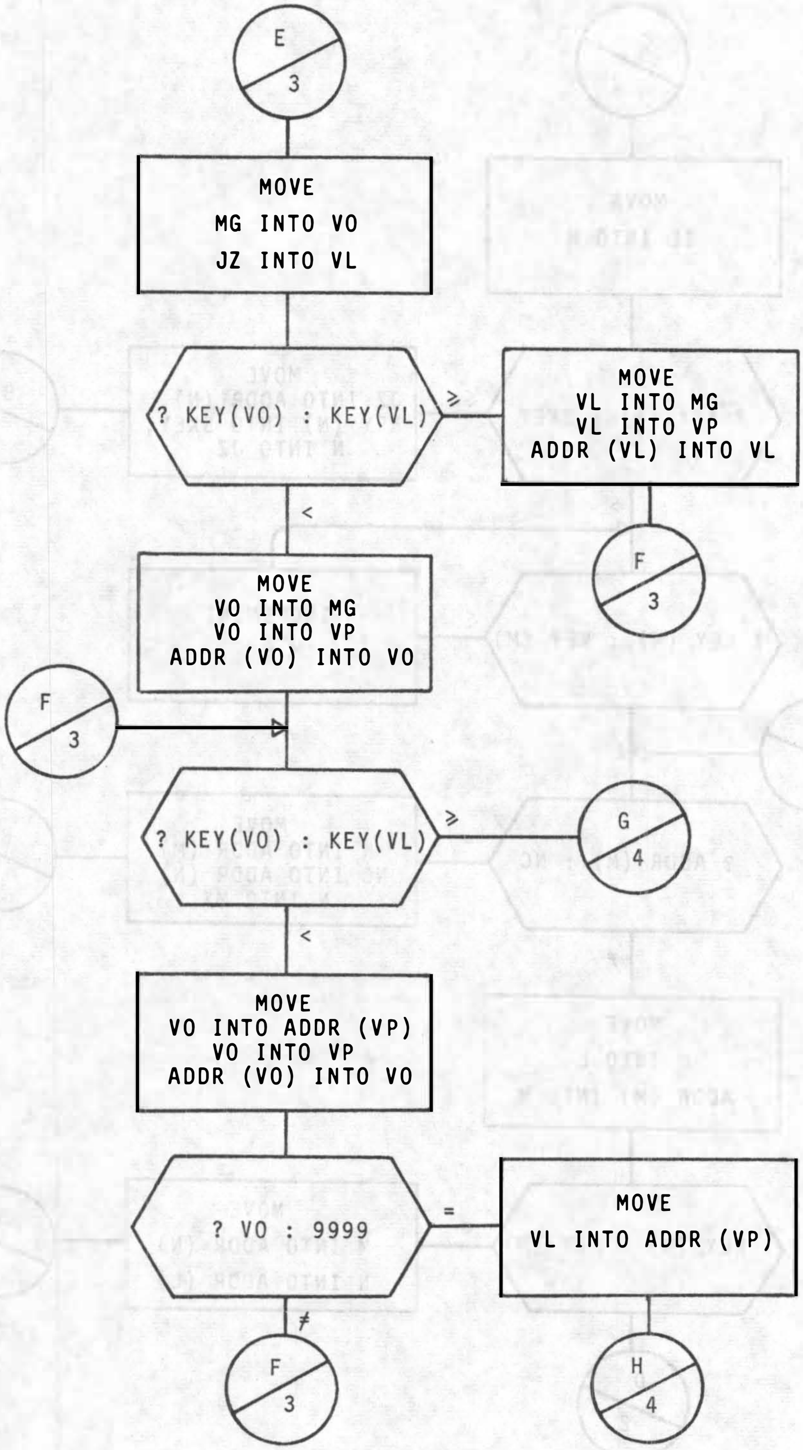
Key

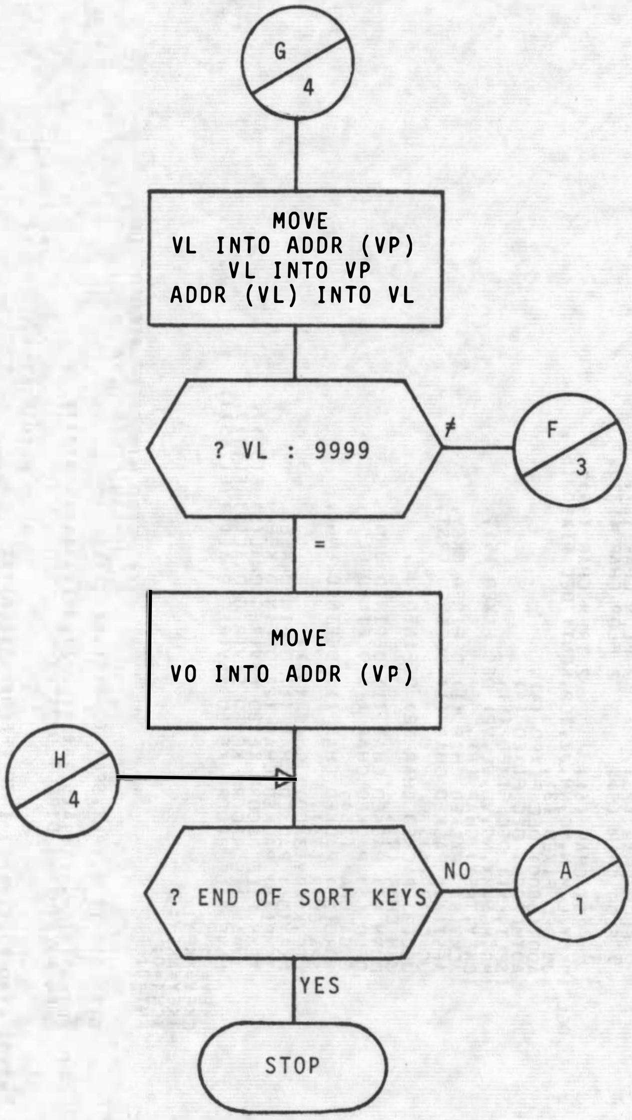
N	= counter used to move down list of sort Keys
NC	= pointer to last Key to be sorted in a subset
IL	= pointer to first Key to be sorted in a subset
M	= pointer to key to be compared with Nth Key
KEY	= array of sort Keys
SKEY	= smallest Key found
JZ	= position of smallest Key found
MX	= position of largest Key found
ADDR	= array to hold pointer to next Key in sort order

SORT ROUTINE




C 2
MOVE
IL INTO M
? KEY (N) : SKEY
≤
MOVE
JZ INTO ADDR (N),
KEY (N) INTO SKEY,
N INTO JZ
B 1
>
? KEY (N) : KEY (M)
≤
INCREMENT
M BY 1
>
D 2
? ADDR (M) : NC
=
MOVE
N INTO ADDR (M)
NC INTO ADDR (N)
N INTO MX
B 1
≠
MOVE
M INTO L
ADDR (M) INTO M
? KEY (N) : KEY (M)
≤
MOVE
M INTO ADDR (N)
N INTO ADDR (L)
B 2
>
D 2


E
3
MOVE
MG INTO VO
JZ INTO VL
? KEY(VO) : KEY(VL)
≥
MOVE
VL INTO MG
VL INTO VP
ADDR (VL) INTO VL
F
3
<
MOVE
VO INTO MG
VO INTO VP
ADDR (VO) INTO VO
F
3
? KEY(VO) : KEY(VL)
≥
G
4
<
MOVE
VO INTO ADDR (VP)
VO INTO VP
ADDR (VO) INTO VO
? VO : 9999
=
MOVE
VL INTO ADDR (VP)
≠
F
3
H
4


G
4
MOVE
VL INTO ADDR (VP)
VL INTO VP
ADDR (VL) INTO VL
? VL : 9999
≠
F
3
=
MOVE
VO INTO ADDR (VP)
H
4
? END OF SORT KEYS
NO
A
1
YES
STOP

4.1.3 Listings

```
PROG: PROC OPTIONS (MAIN);
                                                    /*  SORT MAIN ITEM  */
      DCL 1 RECD,
             2 FLDA CHAR (5),         2 FLDB CHAR (1),
             2 FLDC CHAR (5),         2 FLDD CHAR (25),
             2 FLDE CHAR (2),         2 FLDF CHAR (1),
             2 FLDG CHAR (4),         2 FLDH (5) CHAR (2),
             2 FLDI CHAR (30),        2 FLDJ CHAR (10),
             2 FLDK CHAR (7),         2 FLDL CHAR (2),
             2 FLDM CHAR (23),        2 FLDN CHAR (2),
             2 FLDO CHAR (5),         2 FLDP CHAR (133),
           1 TREC,
             2 FRST CHAR (6),         2 TREM CHAR (34);
      DCL (NKEY,L,M,N,NC,NN,JZ,TOP,KCNT) DEC FIXED (5),
            JN DEC FIXED (3),
            ADDR(3800) DEC FIXED (4),
            ADTM(2000) DEC FIXED (4),
           (SPLT,KNT) DEC FIXED (5),
           (MG,MGA,MX,NG,VO,VL,VP) DEC FIXED (5),
           (CKEY,TKEY) CHAR (8),
            ACKY(8)  PACKED CHAR (1) DEFINED CKEY,
            AFST(6)  PACKED CHAR (1) DEFINED FRST,
            JC CHAR (6),
            JCC(3) PACKED CHAR (2) DEFINED JC,
            JNN CHAR (7),
            DJNN (7) PACKED CHAR (1) DEFINED JNN,
            DFLD (5) PACKED CHAR (1) DEFINED FLDC,
            ENGA CHAR (25),
            WORD(25) PACKED CHAR (1) DEFINED ENGA,
            SPA CHAR (75),
           (KEE,KEARY(3800)) CHAR (36),
            PKEY(36) PACKED CHAR (1) DEFINED KEE,
            TEMP(18) PACKED CHAR (2) DEFINED KEE,
            TEMM FILE RECORD KEYED ENV(REGIONAL(1) F(36)),
            TYPA FILE RECORD KEYED ENV(REGIONAL(1) F(40)),
            MANE FILE RECORD KEYED ENV(REGIONAL(1) F(265));
      NKEY=0;
      CKEY=NKEY;
      TKEY=NKEY;
      KCNT=0;
      KNT=0;
                                                    /*  READ NUMBER OF RECORDS IN
                                                        MAIN FILE    */
      GET EDIT (TOP,SPLT,SPA) (2 F(5),A(70));
      JC='      ';
FA:   FORMAT(A(5),A(1),A(5),A(25),A(2),A(1),A(4),5 A(2));
      OPEN FILE(MANE) DIRECT UPDATE;
                                                                            /*
                                                        READ MAIN FILE RECORD  */
REED: READ FILE(MANE) INTO (RECD) KEY(CKEY);
      IF FLDA<'    1' THEN DO; NKEY=NKEY+1;    KNT=KNT+1;
                                 KEARY(KNT)=SPA;
                                  GO TO AC;    END;
      KCNT=KCNT+1;
      ENGA=FLDD;
                                                                            /*
                                                        FIND POSITION OF EACH TONE AND ADD
                                                        TO SORT KEY    */
      JP=18;
      DO J=1 TO 5;
```

```
        FLDH(J)=' 0';
        END;
        KEE='                                 0 0 0 0 0';
        DO JN=1 TO 25;
        IF WORD(JN)¬='''' THEN GO TO AA;
        JC=JN;
        TEMP(JP)=JCC(3);
        FLDH(JP-13)=JCC(3);
        JP=JP-1;
        IF JP < 14 THEN GO TO AB;
AA:     END;
AB:     K=1;
        IF JP < 16 THEN PUT EDIT (FLDA,ENGA,(FLDH(J) DO J=1 TO 5))
                           (SKIP,A(6),A(30),5 A(2));
                                                                /*
                                        REMOVE TONES FOR SORT KEY
                                                                */
        DO J=1 TO 25;
        IF WORD(J) ¬= '''' THEN DO; PKEY(K)=WORD(J);
                                    K=K+1;        END;
        END;
     REWRITE FILE(MANE) FROM (RECD) KEY (CKEY);
        NKEY=NKEY+1;
        KNT=KNT+1;
        KEARY(KNT)=KEE;
AC:     IF NKEY = TOP THEN GO TO SRT;
        CKEY=NKEY;
        IF NKEY ¬= SPLT THEN GO TO REED;
                                          /*    SORT     */
SRT:    NN=KNT;
        NG=.5*(SQRT(5*NN));
        NC=1;
        MG=0;
CC:     IF NC>NN THEN GO TO CA;
        N=NC;
        MX=NC;
        JZ=NC;
        KEE=KEARY(NC);
        IL=NC;
        NC=NC+NG;
        IF NC>NN THEN DO;  NC=NN+1;   END;
        ADDR(IL)=NC;
AT:     N=N+1;
        IF N=NC THEN GO TO AX;
        IF N>NC THEN GO TO AX;
        M=IL;
        IF KEARY(N)>KEE THEN GO TO AU;
        ADDR(N)=JZ;
        KEE=KEARY(N);
        JZ=N;
        GO TO AT;
AU:     IF KEARY(N)>KEARY(M) THEN GO TO AV;
        M=M+1;
        GO TO AU;
AV:     IF ADDR(M)¬=NC THEN GO TO AW;
        ADDR(M)=N;
        ADDR(N)=NC;
        MX=N;
        GO TO AT;
AW:     L=M;
```

```
        M=ADDR(M);
        IF KEARY(N)>KEARY(M) THEN GC TO AV;
        ADDR(N)=M;
        ADDR(L)=N;
        GO TO AT;
AX:     ADDR(MX)=9999;
        IF MG¬=0 THEN GO TO CB;
        MG=JZ;
        GO TO CC;
                                                      /*   MERGE     */
CB:     VO=MG;
        VL=JZ;
        IF KEARY(VO)<KEARY(VL) THEN GO TO CD;
        MG=VL;
        VP=VL;
        VL=ADDR(VL);
        GO TO CE;
CD:     MG=VO;
        VP=VO;
        VO=ADDR(VO);
CE:     IF KEARY(VO)<KEARY(VL) THEN GO TO CF;
        ADDR(VP)=VL;
        VP=VL;
        VL=ADDR(VL);
        IF VL=9999 THEN GO TO CG;
        GO TO CE;
CF:     ADDR(VP)=VO;
        VP=VO;
        VO=ADDR(VO);
        IF VO=9999 THEN GO TO CH;
        GO TO CE;
CG:     ADDR(VP)=VO;
        GO TO CC;
CH:     ADDR(VP)=VL;
        GO TO CC;
CA:     JZ=MG;
        PUT EDIT ('ADDRESS OF FIRST WORD ',JZ) (SKIP(5),A,F(8));
        IF NKEY ¬= SPLT THEN GO TO AE;
                                                                   /*
                                                  FILE 1ST HALF OF SORK KEYS
                                                                   */
        OPEN FILE(TEMM) DIRECT OUTPUT;
        MGA=JZ;
        DO N=1 TO KNT;
        KEE=KEARY(N);
        TKEY=N;
        WRITE FILE(TEMM) FROM (KEE) KEYFROM (TKEY);
        ADTM(N)=ADDR(N);
        END;
        CLOSE FILE (TEMM);
        KNT=0;
        GO TO REED;
AE:     CLOSE FILE(MANE);
                                                                   /*
                                                  MERGE ADDRESSES OF BOTH HALVES
                                                                   */
        OPEN FILE(TEMM) DIRECT INPUT;
        VO=MGA;
        VL=JZ;
        TKEY=MGA;
```

```
      READ FILE(TEMM) INTO(KEE) KEY (TKEY);
      IF KEE < KEARY(VL) THEN GO TO BD;
      MG=VL+SPLT;
      VP=VL;
      FL=0;
      VL=ADDR(VL);
      GO TO BB;
BD:   MG=VO;
      VP=VO;
      FL=1;
      VO=ADTM(VO);
BE:   MGA=VO;
      TKEY=MGA;
      READ FILE(TEMM) INTO(KEE) KEY (TKEY);
BB:   IF KEE < KEARY(VL) THEN GO TO BF;
      IF FL=0 THEN GO TO BA;
      ADTM(VP)=VL+SPLT;
      GO TO BI;
BA:   ADDR(VP)=VL+SPLT;
BI:   VP=VL;
      FL=0;
      VL=ADDR(VL);
      IF VL=9999 THEN GO TO BG;
      GO TO BB;
BF:   IF FL=0 THEN GO TO BJ;
      ADTM(VP)=VO;
      GO TO BK;
BJ:   ADDR(VP)=VO;
BK:   VP=VO;
      FL=1;
      VO=ADTM(VO);
      IF VO=9999 THEN GO TO BH;
      GO TO BE;
BG:   ADDR(VP)=VO;
      GO TO BC;
BH:   ADTM(VP)=VL+SPLT;
BL:   VP=VL;
      VL=ADDR(VL);
      IF VL=9999 THEN GO TO BC;
      ADDR(VP)=VL+SPLT;
      GO TO BL;
BC:   JZ=MG;
      PUT EDIT ('ADDRESS OF FIRST WORD ',JZ) (SKIP(5),A,F(8));
      OPEN FILE(TYPA) DIRECT UPDATE;
      N=0;
      TKEY=N;
      READ FILE(TYPA) INTO (TREC) KEY (TKEY);
      JN=JZ;
      CKEY=JZ;
      DO J=1 TO 6;
      AFST(J)=ACKY(J+2);
      END;
                                        /*  WRITE ADDRESS OF 1ST WORD INTO
                                            40 CHARACTER FILE  */
      REWRITE FILE(TYPA) FROM (TREC) KEY (TKEY);
      READ FILE(TYPA) INTO (TREC) KEY (TKEY);
      PUT EDIT (FRST,TREM) (SKIP(3),A(6),A(34));
      OPEN FILE(MANE) DIRECT UPDATE;
      M=SPLT+KNT;
      PUT EDIT ('NUMBER SORTED = ',M) (SKIP(2),A,F(7));
```

```
                                                        /*
                                       REWRITE MAIN FILE WITH ADDRESS
                                       OF NEXT WORD IN ORDER   */
      DO N=1 TO M;
      NKEY=N-1;
      CKEY=NKFY;
       READ FILE(MANE) INTO (RECD) KEY(CKEY);
      IF N > SPLT THEN GO TO AH;
      IF ADTM(N)=9999 THEN DO; FLDC='99999';
                         GO TO AG;    END;
      JNN=ADTM(N);
      GO TO AK;
AH:   L=N-SPLT;
      IF ADDR(L)=9999 THEN DO; FLDC='99999';
                         GO TO AG;    END;
      JNN=ADDR(L);
AK:   DO J=1 TO 5;
      DFLD(J) = DJNN(J+2);
      END;
AG:   REWRITE FILE(MANE) FROM (RECD) KEY (CKEY);
      IF N<51 THEN GO TO AI;
      IF N¬=M  THEN GO TO AJ;
AI:    READ FILE(MANE) INTO (RECD) KEY(CKEY);
      PUT EDIT (FLDA,FLDB,FLDC,FLDD,FLDE,FLDF,FLDG,(FLDH(J) DO J=1 TO
        5))  (SKIP,R(FA));
AJ:   END;
      END PROG;
```

4.2 CROSS REFERENCES

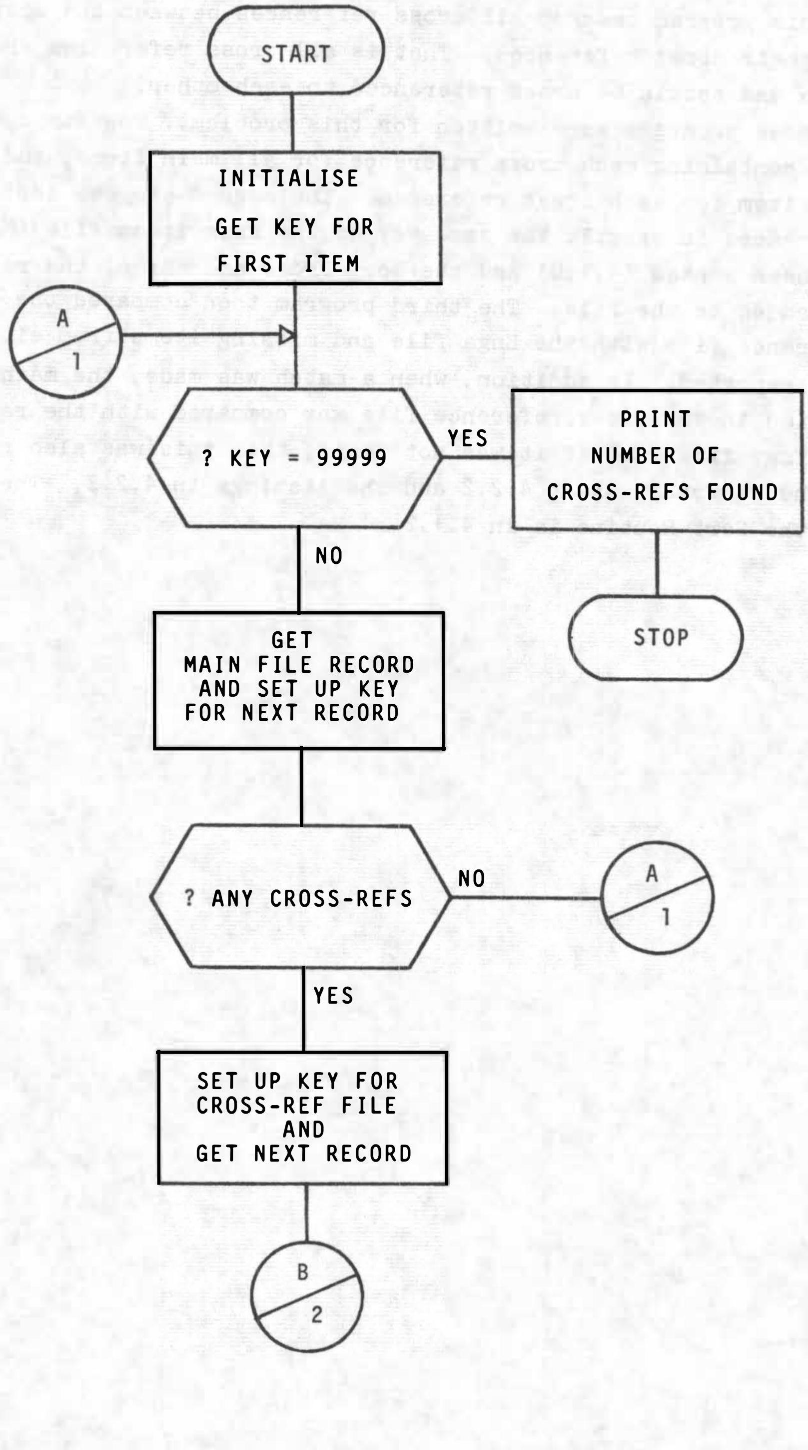
4.2.1 Description of Program

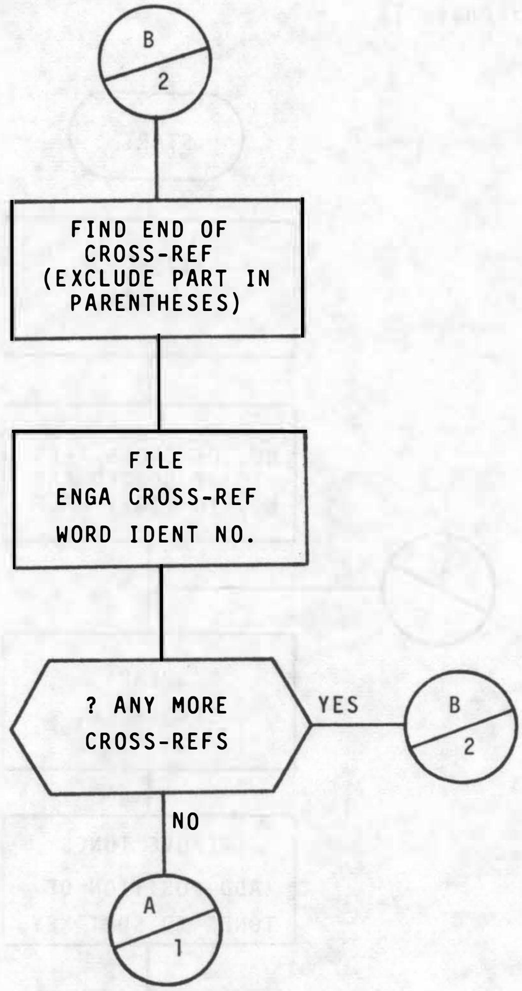
This program checked all cross references between the main items and their cross references. That is all cross references should be main items and should be cross referenced to each other.

Three programs were written for this problem. The first created a file containing each cross reference for all main items, and kept the main item for each cross reference. The second program sorted the references in exactly the same way as the main items file (Master File) had been sorted (4.1.0) and the sort order address of the references was added to the file. The third program then compared the cross reference file with the Enga file and missing items from either file were reported. In addition, when a match was made, the main item carried in the cross reference file was compared with the references of the Enga file and, if it was not found, then this was also reported.

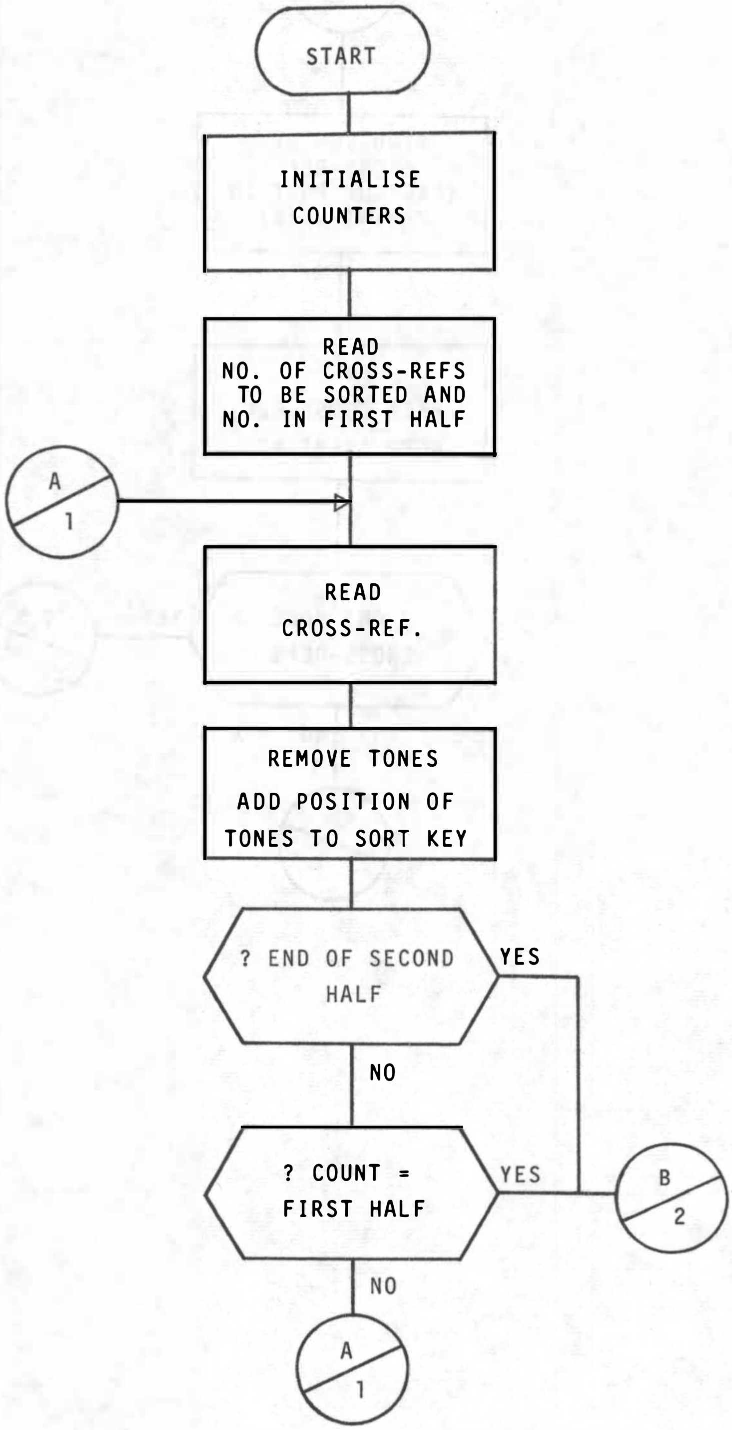
The flowchart is in 4.2.2 and the listings in 4.2.3. The flowchart for the Sort Routine is in 4.1.2.

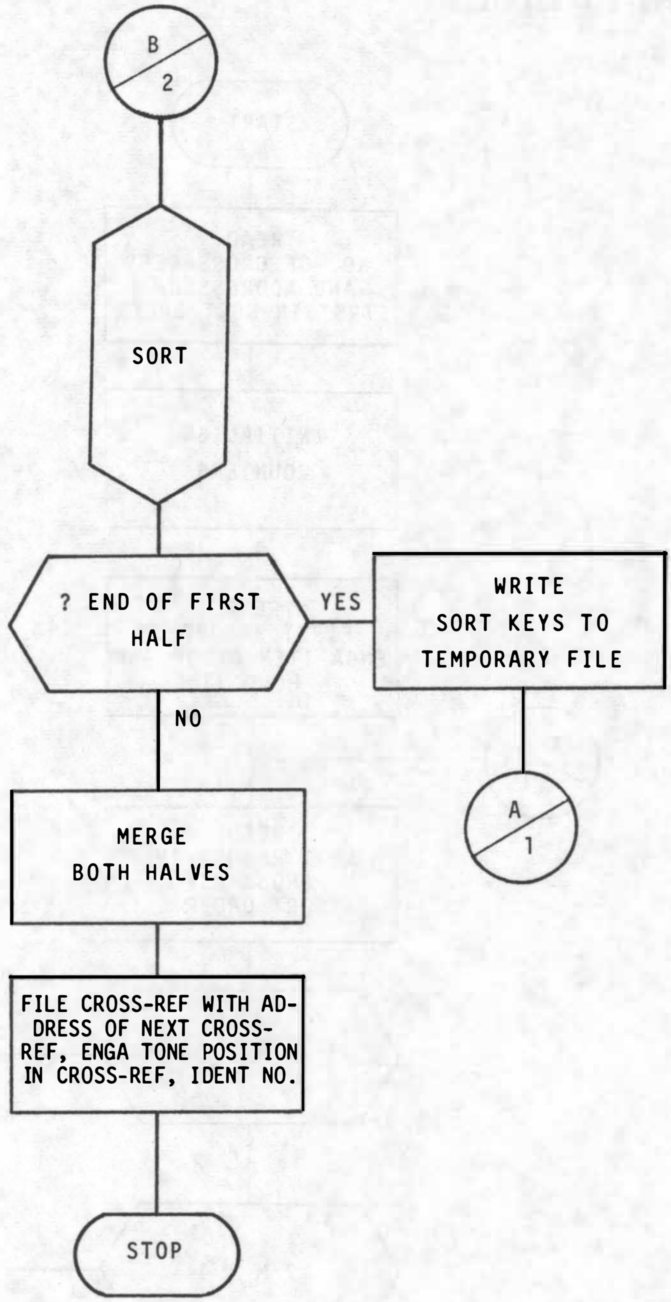
4.2.2 Flowchart: Phase I



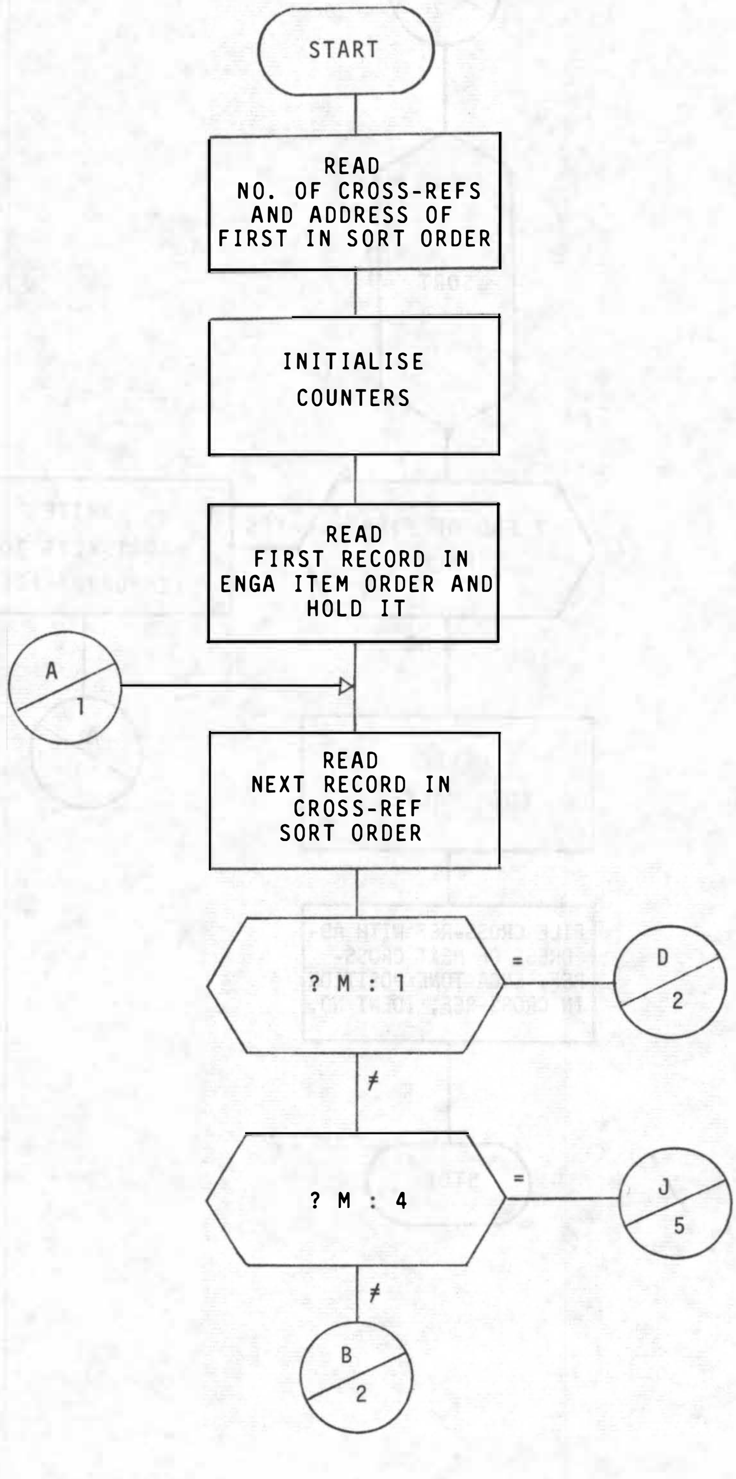

B
2
FIND END OF
CROSS-REF
(EXCLUDE PART IN
PARENTHESES)
FILE
ENGA CROSS-REF
WORD IDENT NO.
? ANY MORE
CROSS-REFS
YES
B
2
NO
A
1

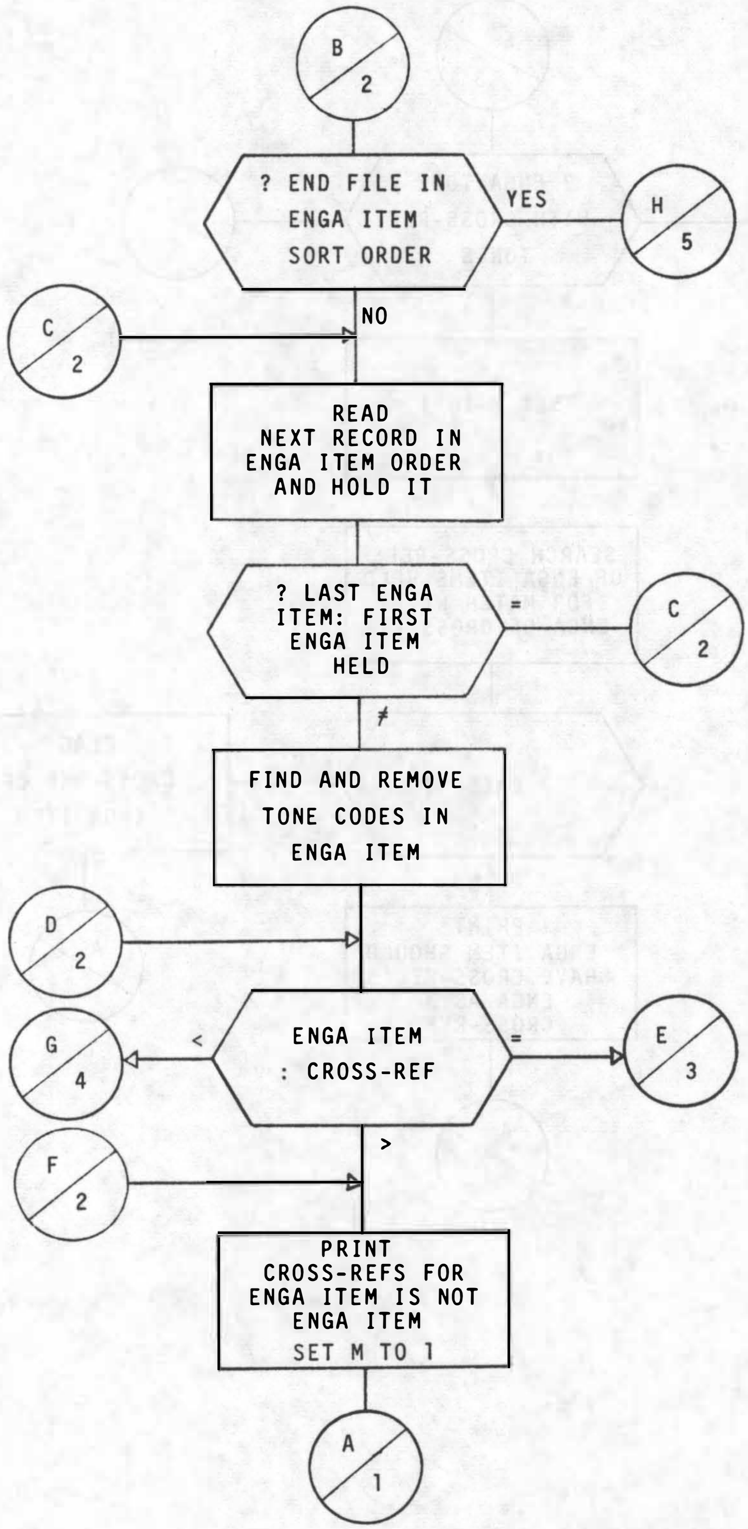
Flowchart: Phase II

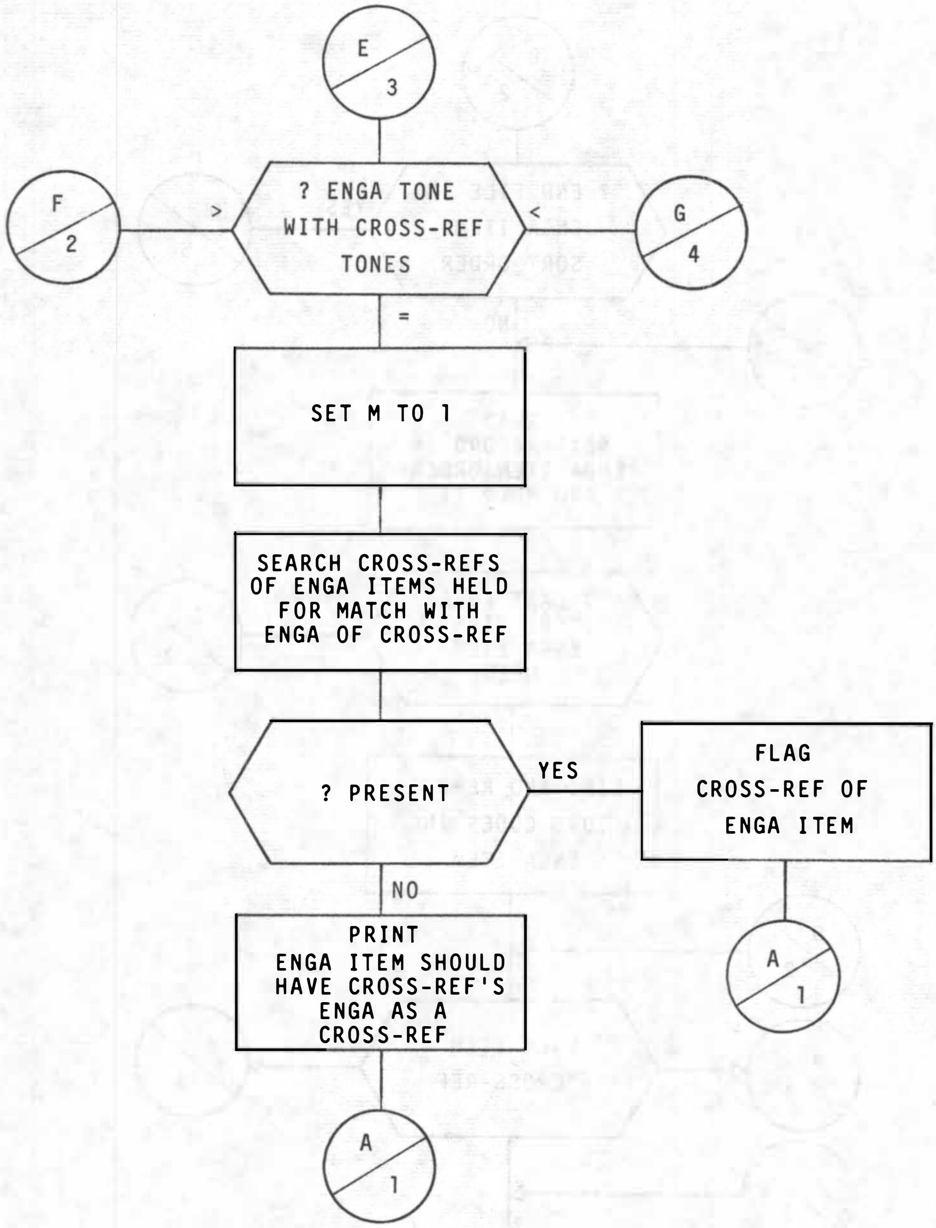


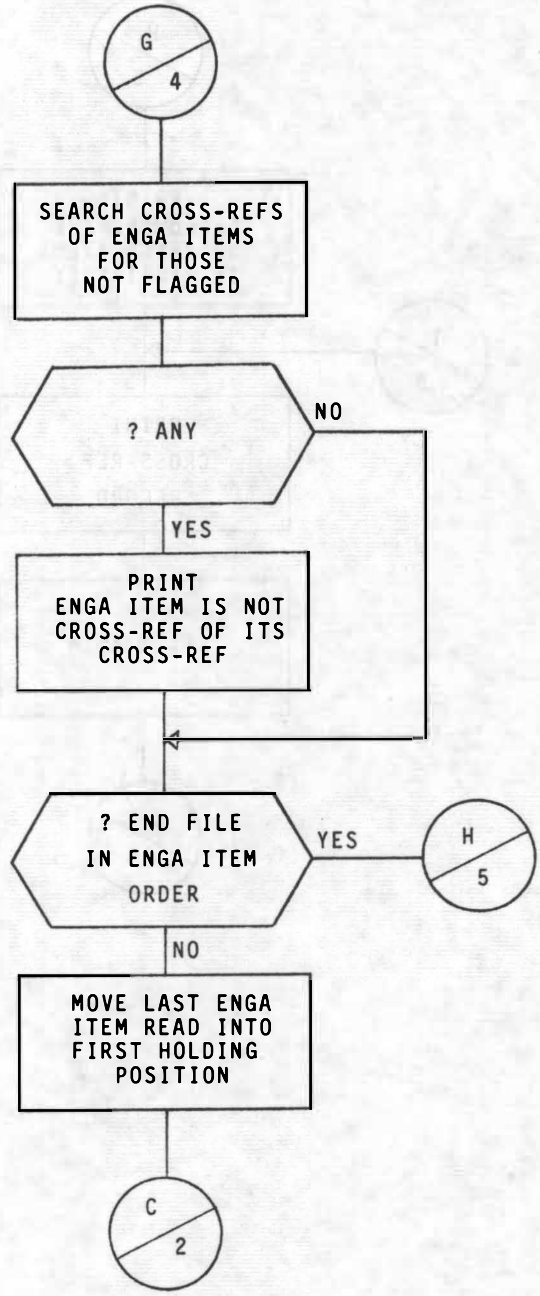

B
2
SORT
? END OF FIRST HALF
YES
WRITE SORT KEYS TO TEMPORARY FILE
A
1
NO
MERGE BOTH HALVES
FILE CROSS-REF WITH ADDRESS OF NEXT CROSS-REF, ENGA TONE POSITION IN CROSS-REF, IDENT NO.
STOP

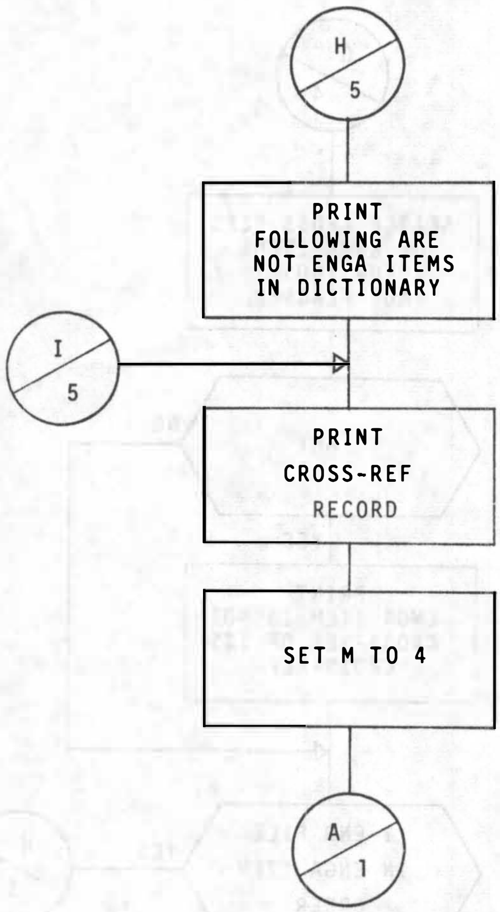
Flowchart: Phase III




B
2
? END FILE IN
ENGA ITEM
SORT ORDER
YES
H
5
NO
C
2
READ
NEXT RECORD IN
ENGA ITEM ORDER
AND HOLD IT
? LAST ENGA
ITEM: FIRST
ENGA ITEM
HELD
=
C
2
≠
FIND AND REMOVE
TONE CODES IN
ENGA ITEM
D
2
ENGA ITEM
: CROSS-REF
<
G
4
=
E
3
>
F
2
PRINT
CROSS-REFS FOR
ENGA ITEM IS NOT
ENGA ITEM
SET M TO 1
A
1


E
3
F
2
G
4
? ENGA TONE WITH CROSS-REF TONES
>
<
=
SET M TO 1
SEARCH CROSS-REFS OF ENGA ITEMS HELD FOR MATCH WITH ENGA OF CROSS-REF
? PRESENT
YES
FLAG CROSS-REF OF ENGA ITEM
NO
PRINT ENGA ITEM SHOULD HAVE CROSS-REF'S ENGA AS A CROSS-REF
A
1
A
1


G
4
SEARCH CROSS-REFS OF ENGA ITEMS FOR THOSE NOT FLAGGED
? ANY
NO
YES
PRINT ENGA ITEM IS NOT CROSS-REF OF ITS CROSS-REF
? END FILE IN ENGA ITEM ORDER
YES
H
5
NO
MOVE LAST ENGA ITEM READ INTO FIRST HOLDING POSITION
C
2


H
5
PRINT
FOLLOWING ARE
NOT ENGA ITEMS
IN DICTIONARY
I
5
PRINT
CROSS-REF
RECORD
SET M TO 4
A
1

4.2.3 Listings

Phase I

```
PROG: PROC OPTIONS (MAIN);
                                                     /*  MAKE CROSS-REF FILE  */
     DCL 1 OREC,
            2 SAD CHAR (5),           2 SYNO CHAR (25),
            2 UPAK CHAR (25),         2 NOFL(5) CHAR (2),
            2 ENGA CHAR (25),         2 MWD CHAR (6);
     DCL (CKEY,EKEY,SKEY,OKEY) CHAR (8),
         (KNT,TOP,IL) DEC FIXED (5),
          CON DEC FIXED (1),
          SPA CHAR (160),
           TEMP CHAR (232),
          MREC CHAR (265),
          SWD CHAR (6),
         (ADDR,SYAD) CHAR (5),
          REST CHAR (96),             EREST CHAR (34),
            LY DEC FIXED (2),
              SYF  CHAR (2),
          TTEM CHAR (126),
          SYN CHAR (232),
              SRA  CHAR (40),
              SRB  CHAR (80),
              SRC  CHAR (120),
              SRD  CHAR (160),
              SRE  CHAR (200),
              SRF  CHAR (240),
                 YY(6)     DEC FIXED (4),
          STEM (232) PACKED CHAR(1) DEFINED TEMP,
          STYN (25)  PACKED CHAR(1) DEFINED SYNO,
          Y(6)  LABEL,
          OUT  FILE RECORD KEYED ENV(REGIONAL(1) F(96)),
          MANE FILE RECORD KEYED ENV(REGIONAL(1) F(265)),
          TYPA FILE RECORD KEYED ENV(REGIONAL(1) F(40)),
          TYPB FILE RECORD KEYED ENV(REGIONAL(1) F(80)),
          TYPD FILE RECORD KEYED ENV(REGIONAL(1) F(160)),
          TYPE FILE RECORD KEYED ENV(REGIONAL(1) F(200)),
          TYPF FILE RECORD KEYED ENV(REGIONAL(1) F(240)),
          TYPC FILE RECORD KEYED ENV(REGIONAL(1) F(120));
     KNT=0;
     CON=1;
     YY=0;
     EKEY=KNT;
     OPEN FILE(MANE) DIRECT INPUT;      OPEN FILE(OUT) DIRECT OUTPUT;
     OPEN FILE(TYPA) DIRECT INPUT;      OPEN FILE(TYPB) DIRECT INPUT;
     OPEN FILE(TYPC) DIRECT INPUT;      OPEN FILE(TYPD) DIRECT INPUT;
     OPEN FILE(TYPE) DIRECT INPUT;      OPEN FILE(TYPF) DIRECT INPUT;
                                                   /*  GET ADDRESS OF 1ST MAIN FILE
                                                       ITEM      */
     READ FILE (TYPA) INTO(SRA)  KEY (EKEY);
     GET STRING (SRA) EDIT (SWD,REST)  (A(6),A(34));
     GET EDIT (TOP,SPA) (A(5),A(75));
     GET EDIT (SPA)  (A(160));                     /*  TWO BLANK CDS */
     SAD=SPA;
     UPAK=SPA;
     NOFL='  ';
     IL=SWD*CON;
     IL=IL-1;
     CKEY=IL;
     ADDR='     ';
                                                   /*  GET NEXT MAIN FILE ITEM */
REED:  IF ADDR ='99999' THEN GO TO NXT;
```

```
       READ FILE(MANE) INTO (MREC) KEY(CKEY);
       GET STRING (MREC) EDIT (MWD,ADDR,ENGA,REST,SYF,SYAD,ITEM)
                (A(6),A(5),A(25),A(96),A(2),A(5),A(126));
                                              /*  SET UP ADDRESS OF NEXT MAIN FILE
                                                  ITEM      */
       IL=ADDR*CON;
       IL=IL-1;
       CKEY=IL;
AB:
                                              /*  SET UP ADDRESS FOR SYNONYM RECORD
                                                  OF MAIN ITEM     */
       IL=SYAD*CON;
       IF IL = 0 THEN GO TO REED;
       IL=IL-1;
       SKEY=IL;
       LY=SYF*CON;
       LY=LY-12;
       IL=LY;
       SYN=SPA;
       YY(IL)=YY(IL)+1;
       IF IL > 6 THEN DO; PUT EDIT ('SYNM TOO LONG',MWD,ENGA)
                          (SKIP,A,A(8),A(30));
                          GO TO AD;  END;
       GO TO Y(IL);
                                              /*  GET SYNONYM RECORD  */
Y(1):  READ FILE(TYPA) INTO (SRA)  KEY (SKEY);
       GET STRING (SRA)  EDIT (SWD,SYN)
            (A(8),A(32));
       GO TO AH;
Y(2):  READ FILE(TYPB) INTO (SRB)  KEY (SKEY);
       GET STRING (SRB)  EDIT (SWD,SYN)
            (A(8),A(72));
       GO TO AH;
Y(3):  READ FILE(TYPC) INTO (SRC)  KEY (SKEY);
       GET STRING (SRC)  EDIT (SWD,SYN)
            (A(8),A(112));
       GO TO AH;
Y(4):  READ FILE(TYPD) INTO (SRD)  KEY (SKEY);
       GET STRING (SRD)  EDIT (SWD,SYN)
            (A(8),A(152));
       GO TO AH;
Y(5):  READ FILE(TYPE) INTO (SRE)  KEY (SKEY);
       GET STRING (SRE)  EDIT (SWD,SYN)
            (A(8),A(192));
       GO TO AH;
Y(6):  READ FILE(TYPF) INTO (SRF)  KEY (SKEY);
       GET STRING (SRF)  EDIT (SWD,SYN)
            (A(8),A(232));
AH:    KA=1;
       KB=0;
       M=0;
       TEMP=SYN;
       IL=SYF*CON;
       IL=(IL-12)*40;
AC:    IF M=0 THEN GO TO AF;
                                              /*  ISOLATE EACH SYNONYM IGNORING
                                                  BRACKETS    */
       DO J=KA TO IL;
       IF STEM(J)=')' THEN GO TO AI;
       K=J;
```

```
      END;
      GO TO REED;
AI:  M=0;
      IF  STEM(K+2)=',' THEN DO; KA=K+3; GO TO AG; END;
      IF  STEM(K+2)=';' THEN DO; KA=K+3; GO TO AG; END;
      IF  STEM(K+2)>' ' THEN GO TO AK;
      IF  STEM(K+3)=' ' THEN GO TO REED;
AK:   KA=K+2;
AG:   IF KA>IL THEN GO TO REED;
AF:   DO J=KA TO IL;
      IF STEM(J)=',' THEN GO TO AA;
      IF STEM(J)=';' THEN GO TO AA;
      IF STEM(J)='(' THEN DO; M=1;  GO TO AA; END;
      IF STEM(J)>' ' THEN GO TO AJ;
      IF STEM(J+1) > ' ' THEN GO TO AJ;
      IF STEM(J+2) > ' ' THEN GO TO AJ;
      KB=3;
      GO TO AA;
AJ:   K=J;
      END;
AA:   N=1;
      KT=K;
      IF (K-KA+1)> 25 THEN DO; PUT EDIT ('***** SYN > 25 CHAR IN ',MWD)
                                 (SKIP(2),A,F(8));
                               KT=KA+24;   END;
      SYNO=SPA;
                                                 /*  MOVE SYNONYM DEFINITION TO
                                                     OUTPUT FIELD   */
      DO J=KA TO KT;
      STYN(N)=STEM(J);
      N=N+1;
      END;
      IF SYNO = SPA THEN GO TO REED;
      KNT=KNT+1;
      OKEY=KNT;
                                                 /*  WRITE TO INTERMEDIATE FILE
                                                                             */
      WRITE FILE(OUT) FROM(OREC) KEYFROM(OKEY);
      IF KNT > 100 THEN GO TO AE;
      PUT EDIT (OREC) (SKIP,A(5),2A(30),5A(2),X(2),A(30),A(6));
                                                                          /*
                                                     CHECK IF ANY MORE SYNONYMS IN
                                                     SYNONYM RECORD   */
AE:   IF KB>2 THEN GO TO REED;
      KA=K+2;
      IF KA>IL THEN GO TO REED;
      GO TO AC;
AD:
      KNT=KNT+1;
      OKEY=KNT;
      WRITE FILE(OUT) FROM (OREC) KEYFROM(OKEY);
      IF KNT > 100 THEN GO TO REED;
      PUT EDIT (OREC) (SKIP,A(5),2A(30),5A(2),X(2),A(30),A(6));
      GO TO REED;
                                                 /*  PRINT OUT COUNT OF NUMBER OF
                                                     SYNONYMS IN INTERMEDIATE FILE
                                                                             */
NXT: PUT EDIT ('SYN FILE CREATE - COUNT = ',KNT) (SKIP(5),A,F(8));
      PUT LIST ('FILE         SYN') SKIP(2);
      DO J=1 TO 6;
      PUT EDIT (J,YY(J)) (SKIP,F(3),F(9));
      END;
      END PROG;
```

```
PROG: PROC OPTIONS (MAIN);                              /* SORT CROSS-REF FILE  */
     DCL 1 RECD,
            2 FLDC CHAR (5),          2 FLDD CHAR (25),
            2 UPAK CHAR (25),         2 NOFL(5) CHAR (2),
            2 FLDI CHAR (25),         2 MWD CHAR (6),
         (NKEY,L,M,N,NC,NN,JZ,TOP,KNT)  DEC FIXED (5),
         (SPLT,MGA) DEC FIXED (5),
         (NG,MG,MX,IL,VO,VL,VP) DEC FIXED (5),
          JN DEC FIXED (3),
          ADDR(3500) DEC FIXED (4),
          ADTM(3500) DEC FIXED (4),
         (CKEY,TKEY) CHAR (8),
          JC CHAR (6),
          JCC(3) PACKED CHAR (2) DEFINED JC,
          JNN CHAR (7),
          DJNN (7) PACKED CHAR (1) DEFINED JNN,
          DFLD (5) PACKED CHAR (1) DEFINED FLDC,
          ENGA CHAR (25),
          WORD(25) PACKED CHAR (1) DEFINED ENGA,
          SPA CHAR (75),
         (KEE,KEARY(3500)) CHAR (36),
          PKEY(36) PACKED CHAR (1) DEFINED KEE,
          TEMP(18) PACKED CHAR (2) DEFINED KEE,
          PAKD(25) PACKED CHAR (1) DEFINED UPAK,
          TEMM FILE RECORD KEYED ENV(REGIONAL(1) F(36)),
          MANE FILE RECORD KEYED ENV(REGIONAL(1) F(96));
     KNT=1;
     NKEY=1;
     CKEY=NKEY;
                                                        /* READ NUMBER OR SYNONYMS IN FILE
                                                                                   */
     GET EDIT (TOP,SPLT,SPA) (2F(5),A(70));
     JC='      ';
     OPEN FILE(MANE) DIRECT UPDATE;
                                                        /* GET NEXT INTERM. FILE SYNONYM
                                                                                   */
REED: READ FILE(MANE) INTO (RECD) KEY(CKEY);
     KEE='                                    0 0 0 0 0';
     IF FLDD=SPA THEN GO TO AF;
     ENGA=FLDD;
                                                                                /*
                                                         FIND POSITION OF EACH TONE AND ADD
                                                         TO SORT KEY     */
     JP=18;
     DO JN=1 TO 25;
     IF WORD(JN)¬='''' THEN GO TO AA;
     JC=JN;
     TEMP(JP)=JCC(3);
     JP=JP-1;
     IF JP < 14 THEN GO TO AB;
AA:  END;
AB:  K=1;
                                                                                /*
                                                         REMOVE TONES FOR SORT KEY
                                                                                */
     DO J=1 TO 25;
     IF WORD(J) ¬= '''' THEN DO; PKEY(K)=WORD(J);
                              K=K+1;          END;
     END;
```

```
AF:
     KEARY(KNT) =KEE;
     KNT=KNT+1;
     VO=NKEY;
     NKEY=NKEY+1;
                                               /*  CHECK IF END OF 1ST OR 2ND HALF
                                                                               */
     IF NKEY > TOP THEN GO TO SRT;
     CKEY=NKEY;
     IF  VO ¬=SPLT THEN GO TO REED;
                                               /*    SORT     */
SRT: NN=KNT-1;
     NG=.5*(SQRT(5*NN));
     NC=1;
     MG=0;
CC:  IF NC>NN THEN GO TO CA;
     N=NC;
     MX=NC;
     JZ=NC;
     KEE=KEARY(NC);
     IL=NC;
     NC=NC+NG;
     IF NC>NN THEN DO;  NC=NN+1;   END;
     ADDR(IL)=NC;
AT:  N=N+1;
     IF N>NC THEN GO TO AX;
     IF N=NC THEN GO TO AX;
     M=IL;
     IF KEARY(N)>KEE THEN GO TO AU;
     ADDR(N)=JZ;
     KEE=KEARY(N);
     JZ=N;
     GO TO AT;
AU:  IF KEARY(N)>KEARY(M) THEN GO TO AV;
     M=M+1;
     GO TO AU;
AV:  IF ADDR(M)¬=NC THEN GO TO AW;
     ADDR(M)=N;
     ADDR(N)=NC;
     MX=N;
     GO TO AT;
AW:  L=M;
     M=ADDR(M);
     IF KEARY(N)>KEARY(M) THEN GO TO AV;
     ADDR(N)=M;
     ADDR(L)=N;
     GO TO AT;
AX:  ADDR(MX)=9999;
     IF MG¬=0 THEN GO TO CB;
     MG=JZ;
     GO TO CC;
                                               /*   MERGE     */
CB:  VO=MG;
     VL=JZ;
     IF KEARY(VO)<KEARY(VL) THEN GO TO CD;
     MG=VL;
     VP=VL;
     VL=ADDR(VL);
     GO TO CE;
CD:  MG=VO;
```

```
      VP=VO;
      VO=ADDR(VO);
CE:   IF KEARY(VO)<KEARY(VL) THEN GO TO CF;
      ADDR(VP)=VL;
      VP=VL;
      VL=ADDR(VL);
      IF VL=9999 THEN GO TO CG;
      GO TO CE;
CF:   ADDR(VP)=VO;
      VP=VO;
      VO=ADDR(VO);
      IF VO=9999 THEN GO TO CH;
      GO TO CE;
CG:   ADDR(VP)=VO;
      GO TO CC;
CH:   ADDR(VP)=VL;
      GO TO CC;
CA:   JZ=MG;
      PUT EDIT ('ADDRESS OF FIRST WORD ',JZ) (SKIP(5),A,F(8));
      VO=NKEY-1;
      IF  VO  ¬= SPLT THEN GO TO AE;
                                                          /*
                                          FILE 1ST HALF OF SORK KEYS
                                                          */
      OPEN FILE(TEMM) DIRECT OUTPUT;
      MGA=JZ;
      DO N=1 TO NN;
      KEE=KEARY(N);
      TKEY=N;
      WRITE FILE(TEMM) FROM (KEE) KEYFROM (TKEY);
      ADTM(N)=ADDR(N);
      END;
      CLOSE FILE (TEMM);
      KNT=1;
      GO TO REED;
AE:   CLOSE FILE(MANE);
                                                          /*
                                          MERGE ADDRESSES OF BOTH HALVES
                                                          */
      OPEN FILE(TEMM) DIRECT INPUT;
      VO=MGA;
      VL=JZ;
      TKEY=MGA;
      READ FILE(TEMM) INTO(KEE) KEY (TKEY);
      IF KEE < KEARY(VL) THEN GO TO BD;
      MG=VL+SPLT;
      VP=VL;
      FL=0;
      VL=ADDR(VL);
      GO TO BB;
BD:   MG=VO;
      VP=VO;
      FL=1;
      VO=ADTM(VO);
BE:   MGA=VO;
      TKEY=MGA;
      READ FILE(TEMM) INTO(KEE) KEY (TKEY);
BB:   IF KEE < KEARY(VL) THEN GO TO BF;
      IF FL=0 THEN GO TO BA;
      ADTM(VP)=VL+SPLT;
```

```
      GO TO BI;
BA:   ADDR(VP)=VL+SPLT;
BI:   VP=VL;
      FL=0;
      VL=ADDR(VL);
      IF VL=9999 THEN GO TO BG;
      GO TO BB;
BF:   IF FL=0 THEN GO TO BJ;
      ADTM(VP)=VO;
      GO TO BK;
BJ:   ADDR(VP)=VO;
BK:   VP=VO;
      FL=1;
      VO=ADTM(VO);
      IF VO=9999 THEN GO TO BH;
      GO TO BE;
BG:   ADDR(VP)=VO;
      GO TO BC;
BH:   ADTM(VP)=VL+SPLT;
BL:   VP=VL;
      VL=ADDR(VL);
      IF VL=9999 THEN GO TO BC;
      ADDR(VP)=VL+SPLT;
      GO TO BL;
BC:   JZ=MG;
      PUT EDIT ('ADDRESS OF FIRST WORD ',JZ) (SKIP(5),A,F(8));
      CLOSE FILE(MANE);
      OPEN FILE(MANE) DIRECT UPDATE;
      CLOSE FILE (TEMM);
      OPEN  FILE (TEMM) DIRECT INPUT;
      M=TOP;
      PUT EDIT ('NUMBER SORTED = ',M) (SKIP(2),A,F(7));
                                                   /*
                                         ADD ORT ADDRESSES ETC TO INTERM.
                                         SYNONYM FILE AND REWRITE RECORD
                                                   */
      DO N=1 TO M;
      NKEY=N;
      CKEY=NKEY;
       READ FILE(MANE) INTO (RECD) KEY(CKEY);
      IF N > SPLT THEN GO TO AH;
      READ FILE (TEMM) INTO (KEE) KEY(CKEY);
      IF ADTM(N)=9999 THEN DO; FLDC='99999';
                          GO TO AG;   END;
      JNN=ADTM(N);
      GO TO AK;
AH:   L=N-SPLT;
      TKEY=L;
      KEE=KEARY(L);
      IF ADDR(L)=9999 THEN DO; FLDC='99999';
                          GO TO AG;   END;
      JNN=ADDR(L);
AK:   DO J=1 TO 5;
      DFLD(J) = DJNN(J+2);
      END;
AG:
      DO J=1 TO 25;
      PAKD(J)=PKEY(J);
      END;
      DO J=1 TO 5;
```

```
               NOFL(J)=TEMP(J+13);
               END;
               REWRITE FILE(MANE) FROM (RECD) KEY (CKEY);
               IF N<101 THEN GO TO AI;
              IF N¬=TOP THEN GO TO AJ;
       AI:     READ FILE(MANE) INTO (RECD) KEY(CKEY);
               PUT EDIT (FLDC,FLDD,FLDI,UPAK,(NOFL(J) DO J=1 TO 5),MWD)
                  (SKIP,A(6),2 A(30),SKIP,X(6),A(30),5 A(2),X(5),A(6));
       AJ:     END;
               END PROG;
```

Phase III

```
PROG: PROC OPTIONS (MAIN);
                                                  /*  CHECK CROSS-REFS   */
      DCL CON DEC FIXED (1),
          (KNT,JZ,IL) DEC FIXED (5),
           SPA CHAR (160),
          (OKEY,CKEY,SKEY,SWD) CHAR (8),
           Y(4)  LABEL,
           OREC CHAR (96),
          (WORD,ENGA,SYNO,HOLD,OSYN) CHAR (25),
          (FLDA,MWD) CHAR (6),
           EWD CHAR (6),
               (ADDR,SYAD,SAD) CHAR (5),
             (SYF,DNOS(5),SNOS(5)) CHAR (2),
           ESYN CHAR (25),
           ESBT (25) PACKED CHAR (1) DEFINED ESYN,
           ETMP(25) PACKED CHAR (1) DEFINED ENGA,
          (SYN,TEMP) CHAR (152),
           RSTA CHAR (7),
           REST CHAR (96),
           FLDB CHAR (34),          TTEM CHAR (126),
           MREC CHAP (265),
           STEM(152) PACKED CHAR(1) DEFINED TEMP,
           STYN(25)  PACKED CHAR(1) DEFINED HOLD,
           SRA CHAR (40),           SRB CHAR (80),
           SRC CHAR (120),          SRD CHAR (160),
           TYPA FILE RECORD KEYED ENV(REGIONAL(1) F(40)),
           MANE FILE RECORD KEYED ENV(REGIONAL(1) F(265)),
           OUT FILE RECORD KEYED ENV(REGIONAL(1) F(96)),
           TYPB FILE RECORD KEYED ENV(REGIONAL(1) F(80)),
           TYPD FILE RECORD KEYED ENV(REGIONAL(1) F(160)),
           TYPC FILE RECORD KEYED ENV(REGIONAL(1) F(120));
      GET EDIT (KNT,JZ,SPA) (F(5),F(5),A(70));
                                   /* KNT = NO. SYNS IN FILE(OUT) */
                                   /* JZ  = ADDRES OF FIRST SYN */
      GET EDIT (SPA) (A(160));
      CON=1;
      OPEN FILE(OUT) DIRECT INPUT;
      OPEN FILE(MANE) DIRECT INPUT;
      OPEN FILE(TYPA) DIRECT INPUT;      OPEN FILE(TYPB) DIRECT INPUT;
      OPEN FILE(TYPC) DIRECT INPUT;      OPEN FILE(TYPD) DIRECT INPUT;
      ADDR = SPA;
      SAD  = SPA;
      OKEY=JZ;
      JZ=0;
      CKEY=JZ;
                                                  /*  GET ADDRESS OF 1ST MAIN FILE
                                                      ITEM        */
      READ FILE(TYPA) INTO (SRA) KEY(CKEY);
      GET STRING (SRA)  EDIT (FLDA,FLDB) (A(6),A(34));
      JZ=FLDA*CON;
      JZ=JZ-1;
      CKEY=JZ;
      M=0;
RB:   MM=0;
      IF SAD='99999' THEN GO TO FIB;
                                                  /*  GET NEXT INTERM. FILE RECORD
                                                                                */
      READ FILE(OUT) INTO (OREC) KEY(OKEY);
      GET STRING (OREC) EDIT (SAD,OSYN,SYNO,(SNOS(J) DO J=1 TO 5),WORD,
     EWD) (A(5),2 A(25),5 A(2),A(25),A(6));
```

```
                                                            /*
                                                   SET UP ADDRES OF NEXT INTEM. FILE
                                                   RECORD     */
      IL=SAD*CON;
      OKEY=IL;
      IF OSYN¬=SPA THEN GO TO AB;
      PUT EDIT ('NO SYN FOR  ',EWD,WORD,' PROG1 ERR - REC= ',OREC)
        (SKIP,A,A(6),X(4),A(25),A,SKIP,A(96));
      GO TO RB;
AB:   IF M=1 THEN GO TO AD;
      IF M=4 THEN GO TO AN;
                                                /*  GET NEXT MAIN FILE ITEM */
RA:   IF ADDR='99999' THEN GO TO FIA;
      READ FILE(MANE) INTO (MREC) KEY(CKEY);
      GET STRING (MREC) EDIT (MWD,ADDR,ENGA,RSTA,(DNOS(J) DO J=1 TO 5),
          REST,SYF,SYAD,TTEM)
        (A(6),A(5),A(25),A(7),5 A(2),A(79),A(2),A(5),A(126));
                                                /*  SET UP ADDRESS OF NEXT MAIN FILE
                                                    ITEM      */
      IL=ADDR*CON;
      IL=IL-1;
      CKEY=IL;
      K=1;
      ESYN=SPA;
                                                /*  FIND TONE POSITIONS OF ENGA AND
                                                    SET UP COMPARISON FIELD */
      L=1;
      DO J=1 TO 25;
      IF ETMP(J)¬='''' THEN DO; ESBT(L)=ETMP(J); L=L+1;  END;
      END;
AD:
                                                /*  COMPARE ENGA AND SYNONYM WITHOUT
                                                    TONES    */
      IF SYND > ESYN THEN GO TO RAX;
      IF SYND < ESYN THEN GO TO AM;
                                                /*  COMPARE TONE POSITIONS FO ENGA AND
                                                    SYNONYM    */
      DO J=5 TO 1 BY -1;
      IF SNOS(J) > DNOS(J) THEN GO TO RAX;
      IF SNOS(J) < DNOS(J) THEN GO TO AM;
      END;
      MM=0;
AQ:   IF SYAD > '     ' THEN GO TO AE;
AO:   IF MM = 1 THEN GO TO AP;
      GO TO AL;
RAX:  IF MM = 1 THEN GO TO AO;
      MM=1;
      GO TO RA;
AE:
                                                /*  SET UP ADDRESS OF SYNONYM RECORD
                                                    FOR CURRENT MAIN FILE ITEM
                                                                              */
      IL=SYAD*CON;
      IF IL = 0 THEN GO TO AO;
      IL=IL-1;
      SKEY=IL;
      IL=SYF*CON;
      IL=IL-12;
      IF IL > 4 THEN DO; PUT EDIT ('SYNM TOO LONG',MWD,ENGA)
                         (SKIP,A,A(7),X(2),A(25));
```

```
                                   GO TO RA;  END;
        SYN=SPA;
                                                    /*  GET SYNONYM RECORD FOR CURRENT
                                                        MAIN ITEM    */
        GO TO Y(IL);
Y(1):   READ FILE(TYPA) INTO (SPA)  KEY (SKEY);
        GET STRING (SPA)  EDIT (SWD,SYN)
            (A(9),A(32));
        GO TO AH;
Y(2):   READ FILE(TYPB) INTO (SPB)  KEY (SKEY);
        GET STRING (SPB)  EDIT (SWD,SYN)
            (A(8),A(72));
        GO TO AH;
Y(3):   READ FILE(TYPC) INTO (SPC)  KEY (SKEY);
        GET STRING (SPC)  EDIT (SWD,SYN)
            (A(8),A(112));
        GO TO AH;
Y(4):   READ FILE(TYPD) INTO (SPD)  KEY (SKEY);
        GET STRING (SPD)  EDIT (SWD,SYN)
            (A(8),A(152));
AH:     KA=1;
        IF MM = 1 THEN DO; PUT EDIT (MWD,ENGA,' SHOULD BE SYNS FOR ONE OR
MORE OF ITS SYNONYMS') (SKIP,A(6),X(4),A(25),A);
                             GO TO RA;    END;
        KB=0;
        NC=0;
        TEMP=SYN;
        IL=SYF*CON;
        IL=(IL-12)*40;
AC:     IF NC=0 THEN GO TO AF;
                                                    /*  ISOLATE MAIN FILE SYNONYM
                                                        IGNORING BRACKETS     */
        DO J=KA TO IL;
        IF STEM(J)=')' THEN GO TO AI;
        K=J;
        END;
        GO TO RA;
AI:     NC=0;
        IF STEM(K+2)=',' THEN DO; KA=K+3; GO TO AG; END;
        IF STEM(K+2)=';' THEN DO; KA=K+3; GO TO AG; END;
        IF STEM(K+2)>' ' THEN DO; PUT EDIT ('///EMBEDED ( ) IN SYN FOR  ',
                                  MWD,ENGA)  (SKIP(3),A,A(7),X(2),A(25));
                                  GO TO AK;  END;
        IF STEM(K+3)=' ' THEN GO TO RA;
AK:     KA=K+2;
AG:     IF KA>IL THEN GO TO RA;
AF:     DO J=KA TO IL;
        IF STEM(J)=',' THEN GO TO AA;
        IF STEM(J)=';' THEN GO TO AA;
        IF STEM(J)='(' THEN DO; NC=1; GO TO AA; END;
        IF STEM(J)>' ' THEN GO TO AJ;
        IF STEM(J+1) > ' ' THEN GO TO AJ;
        IF STEM(J+2) > ' ' THEN GO TO AJ;
        KB=3;
        GO TO AA;
AJ:     K=J;
        END;
AA:     N=1;
        KT=K;
        IF (K-KA+1)> 25 THEN DO; PUT EDIT ('***** SYN > 25 CHAR IN ',MWD)
```

```
                                        (SKIP,A,A(8));
                                        KT=KA+24;     END;
      HOLD=SPA;
      DO J=KA TO KT;
      STYN(N)=STEM(J);
      N=N+1;
      END;
                                                 /*  COMPARE ENGA'S SYNONYM WITH
                                                     INTERM. FILE SYNONYM'S ENGA
                                                                               */
      IF WORD = HOLD THEN GO TO AP;
      IF KB > 2 THEN GO TO AL;
      KA=K+2;
      IF KA > IL THEN GO TO AL;
      GO TO AC;
                                                 /*  VARIOUS ERROR MESSAGES WHEN
                                                     MATCHES NOT FOUND */
AL:   PUT EDIT ('DIC WORD = ',EWD,WORD,' SHOULD BE SYN IN ',MWD,ENGA)
               (SKIP,A,A(6),X(4),A(25),A,A(6),X(4),A(25));
AP:   M=1;
      GO TO RB;
AM:   PUT EDIT ('SYN FOR ',EWD,WORD,' = ',OSYN,' BUT SYN NOT DIC WORD')
        (SKIP,A,X(4),A(6),X(4),A(25),A,X(5),A(30),A);
      GO TO AP;
FIA:  PUT LIST ('FOLLOWING SYNS NOT IN DIC') PAGE;
      PUT LIST (' ') SKIP(4);
AN:   PUT EDIT (OSYN,' FOR ',EWD,WORD)(SKIP,A(45),A,A(6),X(4),A(25));
      M=4;
      GO TO RB;
FIB:  PUT EDIT ('ALL DIC ITEMS AFTER ',MWD,ENGA,' ARE NOT SYNS')
        (PAGE,A,A(6),X(4),A(25),A);
OVR:  PUT LIST ('END SYN CHECK') SKIP(5);
      END PROG;
```

4.3 *REVERSAL (ENGLISH-ENGA)*

4.3.1 Description of Program

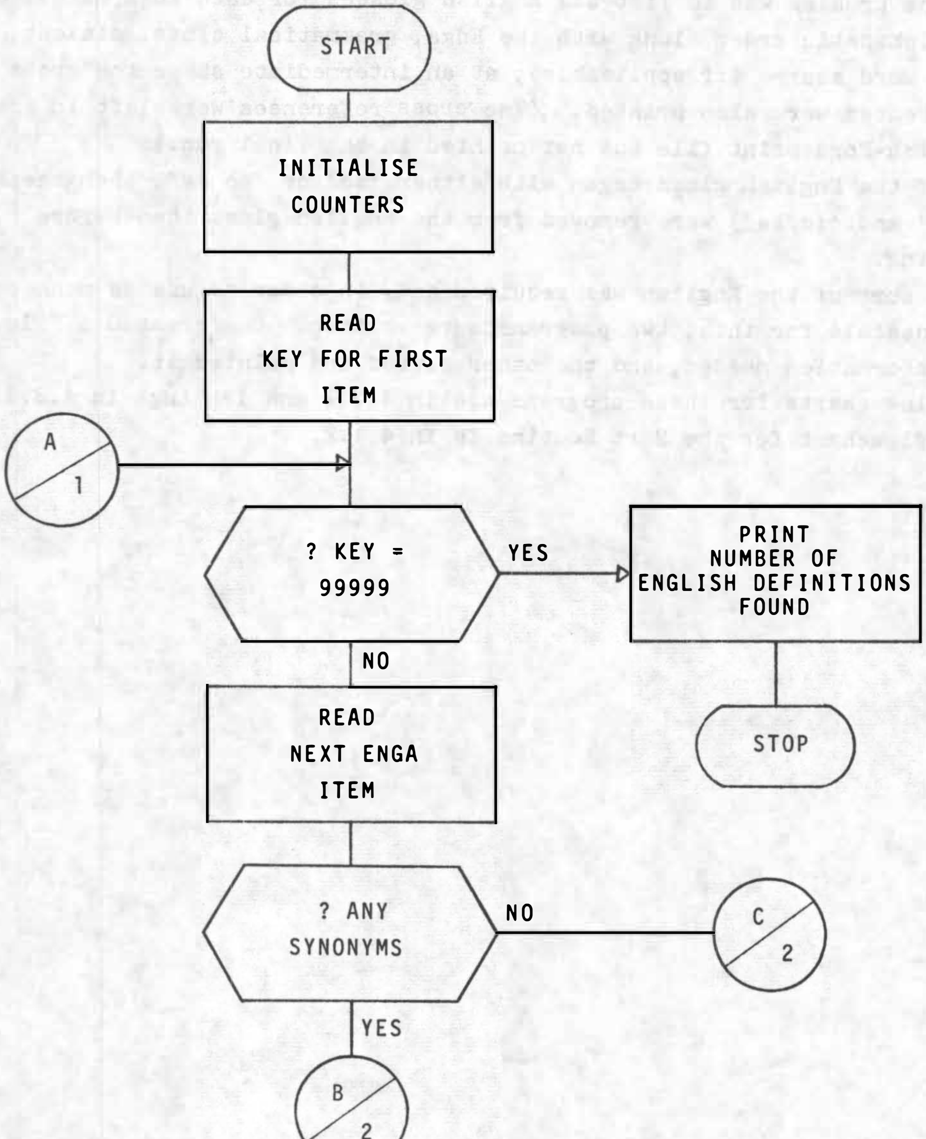
The problem was to list all English glosses for each Enga main item in alphabetic order along with the Enga, grammatical class, dialect, and loan word source (if applicable); at an intermediate stage the cross references were also printed. (The cross references were left in the English-Enga print file but not printed in the final run.)

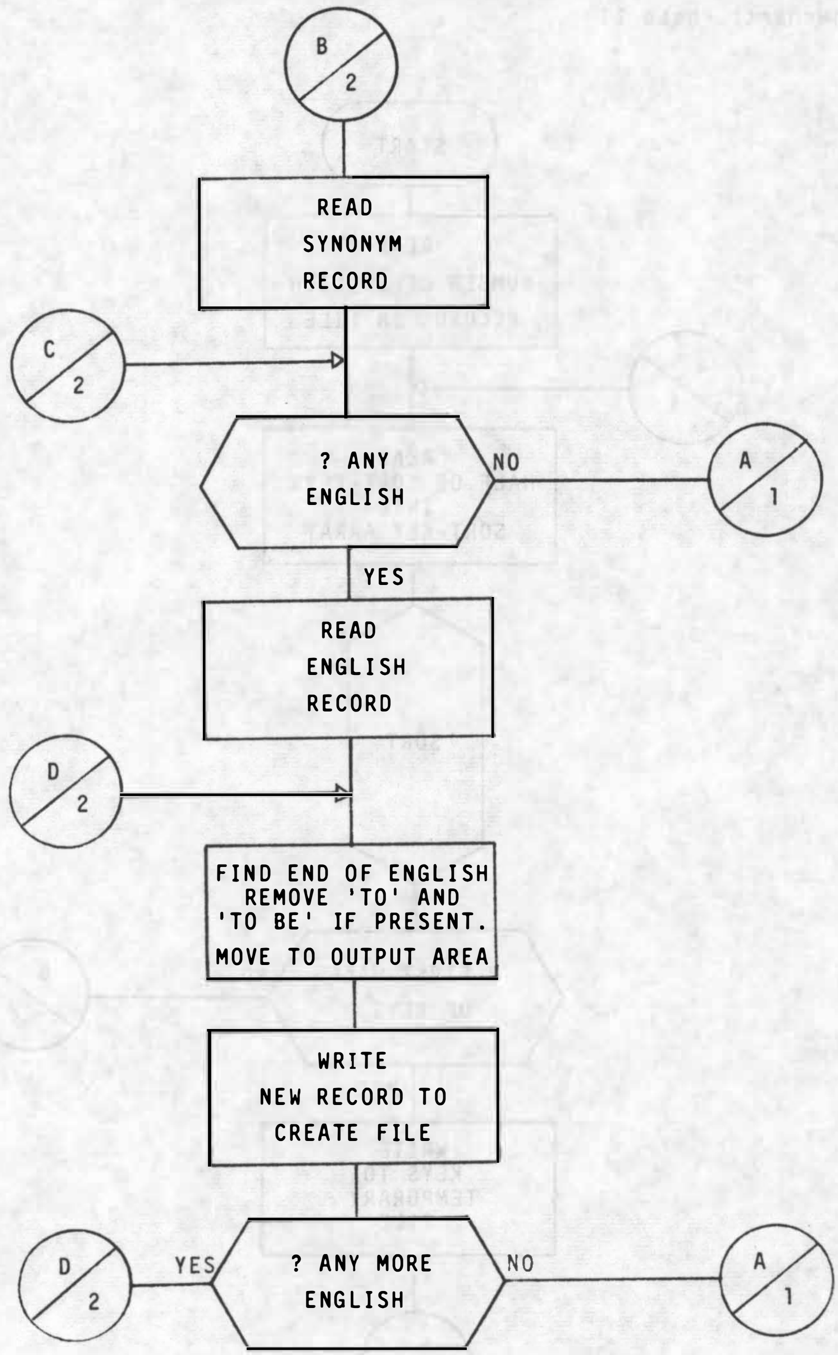
If the English gloss began with either *'to'* or *'to be'*, then these (*'to'* and *'to be'*) were removed from the English gloss item before sorting.

A sort of the English was required and, in order to use as much core as possible for this, two programs were written. One created a file of information needed, and the other sorted and printed it.

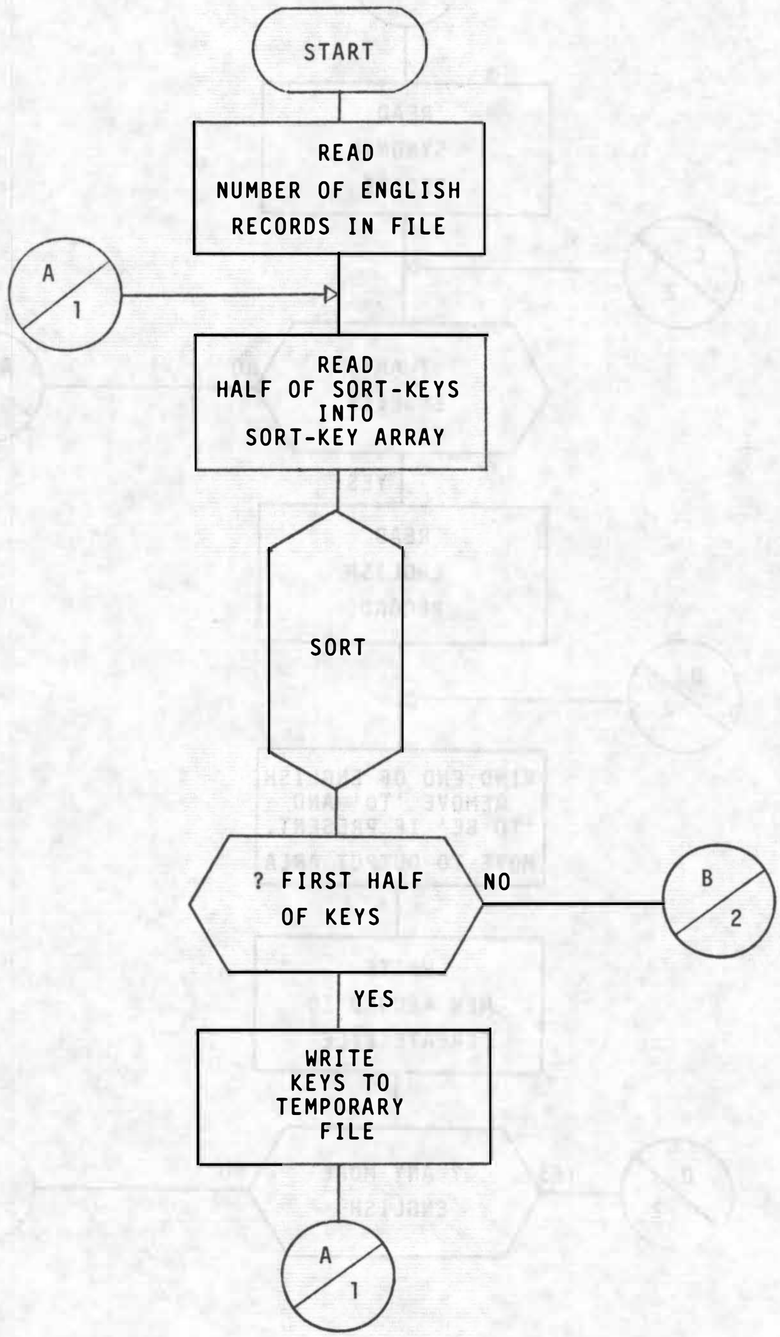
Flow charts for these programs are in 4.3.2 and listings in 4.3.3. The flowchart for the Sort Routine is in 4.1.2.

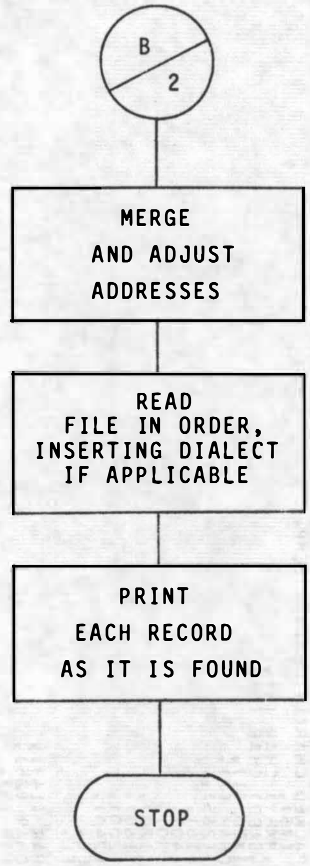
4.3.2 Flowchart: Phase I




B 2
READ SYNONYM RECORD
C 2
? ANY ENGLISH
NO
A 1
YES
READ ENGLISH RECORD
D 2
FIND END OF ENGLISH REMOVE 'TO' AND 'TO BE' IF PRESENT.
MOVE TO OUTPUT AREA
WRITE NEW RECORD TO CREATE FILE
D 2
YES
? ANY MORE ENGLISH
NO
A 1

Flowchart: Phase II




B
2
MERGE
AND ADJUST
ADDRESSES
READ
FILE IN ORDER,
INSERTING DIALECT
IF APPLICABLE
PRINT
EACH RECORD
AS IT IS FOUND
STOP

4.3.3 Listings

Phase I

```
PROG: PROC OPTIONS (MAIN);
                                                  /*  MAKE REVERSAL FILE  */
   DCL
         (IL,KNT,ENAD, TOP) DEC FIXED (5),
        CONT DEC FIXED (2),
        TEMP CHAR (184),
       ENF DEC FIXED (2),
        REST CHAR (126),
        UVA CHAR (8),
        TOB CHAR (3),
        TO_BE CHAR (6),
       ASTR CHAR (3),
       ASTN CHAR (6),
        KEE CHAR (143),
        STYN(143) PACKED CHAR(1) DEFINED KEE,
       STEM(144) PACKED CHAR(1) DEFINED TEMP,
         SPA CHAR (75),
       (EKEY,TKEY,MKEY) CHAR (8),
       (EGAD,ADDE,SADD) CHAR (5),
        CON DEC FIXED (1),
       (EGF,SGF) CHAR (2),
        MREC CHAR (265),
        END CHAR (16),
        ERA CHAR (40),
        ERB CHAR (80),
        ERC CHAR (120),
        ERD CHAR (160),
        ERE CHAR (200),
        ERF CHAR (240),
        Y (6) LABEL,
        Z (4) LABEL,
        1  TOTL,
          2 KEA CHAR (143),
          2 ENGA CHAR (25),
          2 MWD CHAR (5),
          2 ECD CHAR (32),
          2 CNT CHAR (5),
          2 TSYN,
            3 ASYN CHAR (32),
            3 BSYN CHAR (40),
            3 CSYN CHAR (40),
            3 DSYN CHAR (40),
            3 ESYN CHAR (40),
            3 FSYN CHAR (40),
      KEYA FILE RECORD KEYED ENV (REGIONAL(1) F(442)),
      MANE FILE RECORD KEYED ENV (REGIONAL(1) F(265)),
      TYPA FILE RECORD KEYED ENV (REGIONAL(1) F(40)),
      TYPB FILE RECORD KEYED ENV (REGIONAL(1) F(80)),
       TYPD FILE RECORD KEYED ENV (REGIONAL(1) F(160)),
      TYPE FILE RECORD KEYED ENV (REGIONAL(1) F(200)),
      TYPF FILE RECORD KEYED ENV (REGIONAL(1) F(240)),
      TYPC FILE RECORD KEYED ENV (REGIONAL(1) F(120));
   KNT=0;
   ON ERROR BEGIN;  PUT DATA (IL,KNT,J,MWD,ENGA,ENF,EGF,ENAD,EGAD);
                    CLOSE FILE (SYSPRINT);            END;
   OPEN FILE(MANE) DIRECT INPUT;
   OPEN FILE(TYPA) DIRECT INPUT;
   OPEN FILE(TYPB) DIRECT INPUT;
   OPEN FILE(TYPC) DIRECT INPUT;
   OPEN FILE (TYPD) DIRECT INPUT;
```

```
        OPEN FILE (TYPE) DIRECT INPUT;
        OPEN FILE (TYPF) DIRECT INPUT;
        OPEN FILE(KEYA) DIRECT OUTPUT;
                                                          /*  GET ADDRESS OF 1ST MAIN FILE
                                                              ITEM        */
        GET EDIT (TOP,SPA) (F(5),A(75));
        CON=1;
        MKEY=KNT;
        READ FILE (TYPA) INTO (ERA) KEY (MKEY);
        GET STRING (ERA) EDIT (TO_BE,TEMP) (A(6),A(34));
        IL=TO_BE*CON;
        IL=IL-1;
        MKEY=IL;
        ADDE=SPA;
        TOB='TO ';
        TO_BE='TO BE ';
                                                          /*  GET NEXT MAIN FILE ITEM */
REED: IF ADDE = '99999' THEN GO TO FIN;
        READ FILE (MANE) INTO (MREC)  KEY (MKEY);
        GET STRING (MREC) EDIT (MWD,EWD,ADDE,ENGA,TEMP,ECD,EGF,EGAD,
            SGF,SADD,REST)
            (A(5),A(1),A(5),A(25),A(57),A(32),A(2),A(5),A(2),A(5),A(126));
                                                          /*  SET UP ADDRESS OF NEXT MAIN FILE
                                                              ITEM        */
        IL=ADDE*CON;
        IL=IL-1;
        MKEY=IL;
                                                          /*  SET UP ADDRESS OF SYNONYM FOR
                                                              CURRENT MAIN ITEM     */
        ENF=SGF*CON;
        ENAD=SADD*CON;
        IF ENAD > 0 THEN GO TO AR;
BN:     TSYN=SPA;
        CONT=0;
        GO TO AS;
AR:     IL=ENAD-1;
        EKEY=IL;
        IL=ENF-12;
        IF IL > 6 THEN DO; PUT EDIT ('SYN FILES GT 6 = ',IL)
                               (SKIP,A,F(4));  GO TO BN;  END;
                                                          /*  GET SYNONYM RECORD  */
        GO TO Y(IL);
Y(1):   READ FILE(TYPA) INTO (ERA)  KEY (EKEY);
        GET STRING (ERA)  EDIT (OVA,ASYN)
            (A(8),A(32));
        CONT=1;
        GO TO AD;
Y(2):   READ FILE(TYPB) INTO (ERB)  KEY (EKEY);
        GET STRING (ERB)  EDIT (OVA,ASYN,BSYN)
            (A(8),A(32),A(40));
        CONT=2;
        GO TO AD;
Y(3):   READ FILE(TYPC) INTO (ERC)  KEY (EKEY);
        GET STRING (ERC)  EDIT (OVA,ASYN,BSYN,CSYN)
            (A(8),A(32),2A(40));
        CONT=3;
        GO TO AD;
Y(4):   READ FILE(TYPD) INTO (ERD)  KEY (EKEY);
        GET STRING (ERD)  EDIT (OVA,ASYN,BSYN,CSYN,DSYN)
            (A(8),A(32),3A(40));
```

```
       CONT=4;
       GO TO AD;
Y(5):  READ FILE(TYPE) INTO (ERE)  KEY (EKEY);
       GET STRING (ERE)  EDIT (OVA,ASYN,BSYN,CSYN,DSYN,ESYN)
           (A(8),A(32),4A(40));
       CONT=5;
       GO TO AD;
Y(6):  READ FILE(TYPF) INTO (ERF)  KEY (EKEY);
       GET STRING (ERF)  EDIT (OVA,ASYN,BSYN,CSYN,DSYN,ESYN,FSYN)
           (A(8),A(32),5A(40));
       CONT=6;
AD:    CNT=CONT;
AS:
                                                    /*  SET UP ADDRESS FOR ENGLISH OF
                                                        CURRENT MAIN ITEM  */
       ENF=EGF*CON;
       ENAD=EGAD*CON;
       IF ENAD > 0 THEN GO TO AF;
       GO TO REED;
AF:    IL=ENAD-1;
       EKEY=IL;
       IL=ENF-12;
       IF IL > 4 THEN DO; PUT EDIT ('EGL FILES GT 4 = ',IL)
                                 (SKIP,A,F(4));  GO TO REED;  END;
       IF IL < 1 THEN DO; PUT EDIT ('EGL FILES LT 1 = ',IL)
                                 (SKIP,A,F(4));  GO TO REED;  END;
                                                    /*  GET ENGLISH RECORD  */
       GO TO Z(IL);
Z(1):  READ FILE(TYPA) INTO (ERA)  KEY (EKEY);
       GET STRING (ERA)  EDIT (EWD,TEMP)
           (A(16),A(24));
       GO TO AB;
Z(2):  READ FILE(TYPB) INTO (ERB)  KEY (EKEY);
       GET STRING (ERB)  EDIT (EWD,TEMP)
           (A(16),A(64));
       GO TO AB;
Z(3):  READ FILE(TYPC) INTO (ERC)  KEY (EKEY);
       GET STRING (ERC)  EDIT (EWD,TEMP)
           (A(16),A(104));
       GO TO AB;
Z(4):  READ FILE(TYPD) INTO (ERD)  KEY (EKEY);
       GET STRING (ERD)  EDIT (EWD,TEMP)
           (A(16),A(144));
AB:
       KA=1;
       KB=0;
       IL=(ENF-12)*40;
       IF IL > 120 THEN IL=145;
AN:    L=0;
       LX=0;
                                                    /*  LOOP THROUGH EACH CHARACTER OF
                                                        ENGLISH RECORD FOR END OF EACH
                                                        DEFINITION AND IGNORE BRACKETS
                                                                                   */
       DO K=KA TO IL;
       IF LX<1 THEN GO TO AA;
       IF STEM(K) = '(' THEN DO; LX=LX+1; PUT LIST ('IMBD BRCH') SKIP;
                           PUT DATA (KA,IL,K,L,LX);
                           PUT EDIT (MWD,TEMP,(STEM(J) DO J=105 TO 144))
                      (SKIP,X(6),A(5),X(3),A(104), SKIP,X(14),40A(1));
```

```
                        GO TO AL;      END;
      IF STEM(K) = ')' THEN DO; LX=LX-1; GO TO AL; END;
      GO TO AC;
AA:   IF STEM(K)=',' THEN GO TO AO;
      IF STEM(K)=';' THEN GO TO AO;
      IF STEM(K) = '(' THEN DO; LX=1; GO TO AL; END;
AC:   IF STEM(K)>' ' THEN GO TO AL;
      IF STEM(K+1) > ' ' THEN GO TO AL;
      IF STEM(K+2) > ' ' THEN GO TO AL;
      KB=3;
      GO TO AO;
AL:   L=K;
      END;
AO:   IF L=0 THEN DO; PUT EDIT('ENGLISH BEGINS WITH BLANKS IN ',MWD,
                      '    VIZ:    ',TEMP)
                      (SKIP,A,A(6),A,SKIP,X(10),A(104));
                      GO TO REED;      END;
      IF STEM(KA) = ' ' THEN KA=KA+1;
      KT=L;
      IF (L-KA+1) > 143 THEN KT=KA+142;
                                       /*  DETERMINE IF "TO" OR "TO BE"
                                           IS PRESENT          */
      ASTN=SPA;
      ASTR=SPA;
      NN=0;
      ASTR=SUBSTR(TEMP,KA,(KA+2));
      IF ASTR ¬= TOB THEN GO TO AY;
      ASTN=SUBSTR(TEMP,KA,(KA+5));
      IF ASTN = TO_BE THEN GO TO AZ;
      NN=3;
      GO TO AY;
AZ:   NN=6;
AY:   KEE=SPA;
                                       /*  MOVE DEFINITION TO OUTPUT AREA,
                                           REMOVING "TO" OR "TO BE" IF
                                           PRESENT     */
      N=1;
      DO K=(KA+NN) TO KT;
      STYN(N)=STEM(K);
      N=N+1;
      END;
      IF KEE=SPA THEN GO TO REED;
      KNT=KNT+1;
      IF KNT > 7500 THEN DO; PUT LIST ('NO. OF EGLS GT 7500') SKIP;
                             PUT DATA (KNT,J);
                             GO TO REED;     END;
      TKEY=KNT;
      KEA=KEE;
                                        /* WRITE RECORD TO INTERMEDIATE FILE
                                                                        */
      WRITE FILE (KEYA) FROM (TOTL) KEYFROM(TKEY);
                                       /*  CHECK IF ANY MORE ENGLISH IN
                                           THIS RECORD         */
      IF KB>2 THEN GO TO REED;
      KA=L+2;
      IF KA>IL THEN GO TO REED;
      GO TO AN;
FIN:
      PUT EDIT ('NO. OF ENGLISH DEFS = ',KNT) (SKIP(2),A,F(6));
      END PROG;
```

Phase II

```
PROG: PROC OPTIONS (MAIN);                                /*  SORT & PRINT REVERSAL */
     DCL
          (NG,MG,MX,IL,VO,VL,VP) DEC FIXED (5),
             (J,MGA) DEC FIXED (5),
          (L,M,N,NC,NN,JZ) DEC FIXED (5),
            KEARY(3750) CHAR (20),
            ADDR(3750) DEC FIXED (4),
            ADTM(3750) DEC FIXED (4),
         WORD CHAR (20),
          (FL,CON) DEC FIXED (1),
          (TOP,SPLT,ENF) DEC FIXED (5),
           KEE CHAR (20),
           TKEY CHAR (8),
           SPA CHAR (75),
          (BRA,BRB) CHAR (1),
          (ORA,ORB) CHAR (1),
          (EVB,EST) CHAR (1),
           OCD CHAR (2),
           SPN CHAR (22),
           1 TOTL,
             2 KEA CHAR (143),
             2 ENGA CHAR (25),
             2 MWD CHAR (5),
             2 ECD CHAR (32),
             2 CNT CHAR (5),
             2 TSYN,
               3 ASYN CHAR (32),
               3 BSYN CHAR (40),
               3 CSYN CHAR (40),
               3 DSYN CHAR (40),
               3 ESYN CHAR (40),
               3 FSYN CHAR (40),
          TEMM FILE RECORD KEYED ENV (REGIONAL(1) F(20)),
          KEYA FILE RECORD KEYED ENV (REGIONAL(1) F(442));
     CON=1;
     KNT=0;
     KNT1=0;
     BRA='(';
     BRB=')';
     ON ERROR GO TO ERR;
                                                          /*  READ NUMBER OF RECORDS IN
                                                              REVERSAL FILE AND WORD FROM WHICH
                                                              TO BEGIN PRINTING   */
     GET EDIT (TOP,SPLT,WORD,SPA) (2 F(5),A(20),A(50));
     OPEN FILE (KEYA)  DIRECT INPUT;
     MA=1;
     MB=SPLT;
                                                          /*  READ HALF OF SORT KEYES INTO
                                                              SORT ARRAY     */
AA:  DO J=MA TO MB;
     TKEY=J;
     READ FILE (KEYA) INTO (TOTL) KEY (TKEY);
     KNT=KNT+1;
     KEARY(KNT)=KEA;
     IF J < 50 THEN PUT EDIT(KEARY(KNT)) (SKIP,A(20));
     END;
AB:
                                                          /*     SORT     */
     NN=KNT;
```

```
      NG=.5*(SQRT(5*NN));
      NC=1;
      MG=0;
CC:   IF NC>NN THEN GO TO CA;
      N=NC;
      MX=NC;
      JZ=NC;
      KEE=KEARY(NC);
      IL=NC;
      NC=NC+NG;
      IF NC>NN THEN DO;  NC=NN+1;    END;
      ADDR(IL)=NC;
AT:   N=N+1;
      IF N>NC THEN GO TO AX;
      IF N=NC THEN GO TO AX;
      M=IL;
      IF KEARY(N)>KEE THEN GO TO AU;
      IF KEARY(N)=KEE THEN GO TO AU;
      ADDR(N)=JZ;
      KEE=KEARY(N);
      JZ=N;
      GO TO AT;
AU:   IF KEARY(N)>KEARY(M) THEN GO TO AV;
      IF KEARY(N)=KEARY(M) THEN GO TO AV;
      M=M+1;
      GO TO AU;
AV:   IF ADDR(M)¬=NC THEN GO TO AW;
      ADDR(M)=N;
      ADDR(N)=NC;
      MX=N;
      GO TO AT;
AW:   L=M;
      M=ADDR(M);
      IF KEARY(N)>KEARY(M) THEN GO TO AV;
      IF KEARY(N)=KEARY(M) THEN GO TO AV;
      ADDR(N)=M;
      ADDR(L)=N;
      GO TO AT;
AX:   ADDR(MX)=9999;
      IF MG¬=0 THEN GO TO CB;
      MG=JZ;
      GO TO CC;
                                                /*   MERGE    */
CB:   VO=MG;
      VL=JZ;
      IF KEARY(VO)<KEARY(VL) THEN GO TO CD;
      IF KEARY(VO)=KEARY(VL) THEN GO TO CD;
      MG=VL;
      VP=VL;
      VL=ADDR(VL);
      GO TO CE;
CD:   MG=VO;
      VP=VO;
      VO=ADDR(VO);
CE:   IF KEARY(VO)<KEARY(VL) THEN GO TO CF;
      IF KEARY(VO)=KEARY(VL) THEN GO TO CF;
      ADDR(VP)=VL;
      VP=VL;
      VL=ADDR(VL);
      IF VL=9999 THEN GO TO CG;
```

```
      GO TO CE;
CF:   ADDR(VP)=VO;
      VP=VO;
      VO=ADDR(VO);
      IF VO=9999 THEN GO TO CH;
      GO TO CE;
CG:   ADDR(VP)=VO;
      GO TO CC;
CH:   ADDR(VP)=VL;
      GO TO CC;
CA:   JZ=MG;
      PUT EDIT ('ADDRESS OF FIRST WORD ',JZ,'NUM = ',KNT)
           (SKIP(5),A,F(8),A,F(8));
       IF  MB ¬=SPLT THEN GO TO AE;
                                                       /*
                                 FILE 1ST HALF OF SORK KEYS
                                                        */
      OPEN FILE(TEMM) DIRECT OUTPUT;
      MGA=JZ;
      KNT1=KNT;
      DO N=1 TO KNT;
      KEE=KEARY(N);
      TKEY=N;
      WRITE FILE(TEMM) FROM (KEE) KEYFROM (TKEY);
      ADTM(N)=ADDR(N);
      END;
      CLOSE FILE (TEMM);
      KNT=0;
      MA=SPLT+1;
      MB=TOP;
      GO TO AA;
ERR:  VO = KNT+SPLT;
      PUT DATA (KNT,MA,MB,VO,SPLT,TOP) SKIP;
      GO TO AB;
AE:
                                                       /*
                                 MERGE ADDRESSES OF BOTH HALVES
                                                        */
      OPEN FILE(TEMM) DIRECT INPUT;
      VO=MGA;
      VL=JZ;
      TKEY=MGA;
      READ FILE(TEMM) INTO(KEE) KEY (TKEY);
      IF KEE < KEARY(VL) THEN GO TO BD;
      IF KEE = KEARY(VL) THEN GO TO BD;
      MG=VL+KNT1;
      VP=VL;
      FL=0;
      VL=ADDR(VL);
      GO TO BB;
BD:   MG=VO;
      VP=VO;
      FL=1;
      VO=ADTM(VO);
BE:   MGA=VO;
      TKEY=MGA;
      READ FILE(TEMM) INTO(KEE) KEY (TKEY);
BB:   IF KEE < KEARY(VL) THEN GO TO BF;
      IF KEE = KEARY(VL) THEN GO TO BF;
       IF FL=0 THEN GO TO BA;
```

```
      ADTM(VP)=VL+KNT1;
      GO TO BI;
BA:   ADDR(VP)=VL+KNT1;
BI:   VP=VL;
      FL=0;
      VL=ADDR(VL);
      IF VL=9999 THEN GO TO BG;
      GO TO BB;
BF:   IF FL=0 THEN GO TO BJ;
      ADTM(VP)=VO;
      GO TO BK;
BJ:   ADDR(VP)=VO;
BK:   VP=VO;
      FL=1;
      VO=ADTM(VO);
      IF VO=9999 THEN GO TO BH;
      GO TO BE;
BG:   ADDR(VP)=VO;
      GO TO BC;
BH:   ADTM(VP)=VL+KNT1;
BL:   VP=VL;
      VL=ADDR(VL);
      IF VL=9999 THEN GO TO BC;
      ADDR(VP)=VL+KNT1;
      GO TO BL;
BC:   JZ=MG;
      PUT EDIT ('ADDRESS OF FIRST WORD ',JZ) (SKIP(5),A,F(8));
      CLOSE FILE(KEYA);
      OPEN FILE (KEYA) DIRECT INPUT;
      M=KNT+KNT1;
      TKEY=JZ;
      L=JZ;
      PUT EDIT ('NUMBER SORTED = ',M) (SKIP(2),A,F(7));
      PUT LIST (' ') PAGE;
                                                                  /*
                                              ADD *  (   )  AND PRINT
                                                                  */
      DO N=1 TO M;
      READ FILE(KEYA) INTO(TOTL) KEY(TKEY);
      GET STRING (ECD) EDIT (KEE,OCD,EVB,SPN) (A(7),A(2),A(1),A(22));
      IF EVB=' ' THEN EST=' ';
                 ELSE EST='*';
      IF OCD='  ' THEN DO; ORA=SPA;
                           ORB=SPA;
                           GO TO AC;  END;
      ORA=BRA;
      ORB=BRB;
AC:
      ENF= CNT*CON;
      IF KEA < WORD THEN GO TO AQ;
      PUT EDIT (KEA,EST,ENGA,ORA,OCD,ORB,ECD,MWD)
     (SKIP(2),A(92),SKIP,X(20),A(1),A(28),A(1),A(2),A(1),X(3),A(32),
          X(15),A(5));
AQ:
      IF L > KNT1 THEN GO TO AH;
       MX =ADTM(L);
      GO TO AJ;
AH:   MX=ADDR(L-KNT1);
AJ:   TKEY=MX;
      L=MX;
      END;
      END PROG;
```

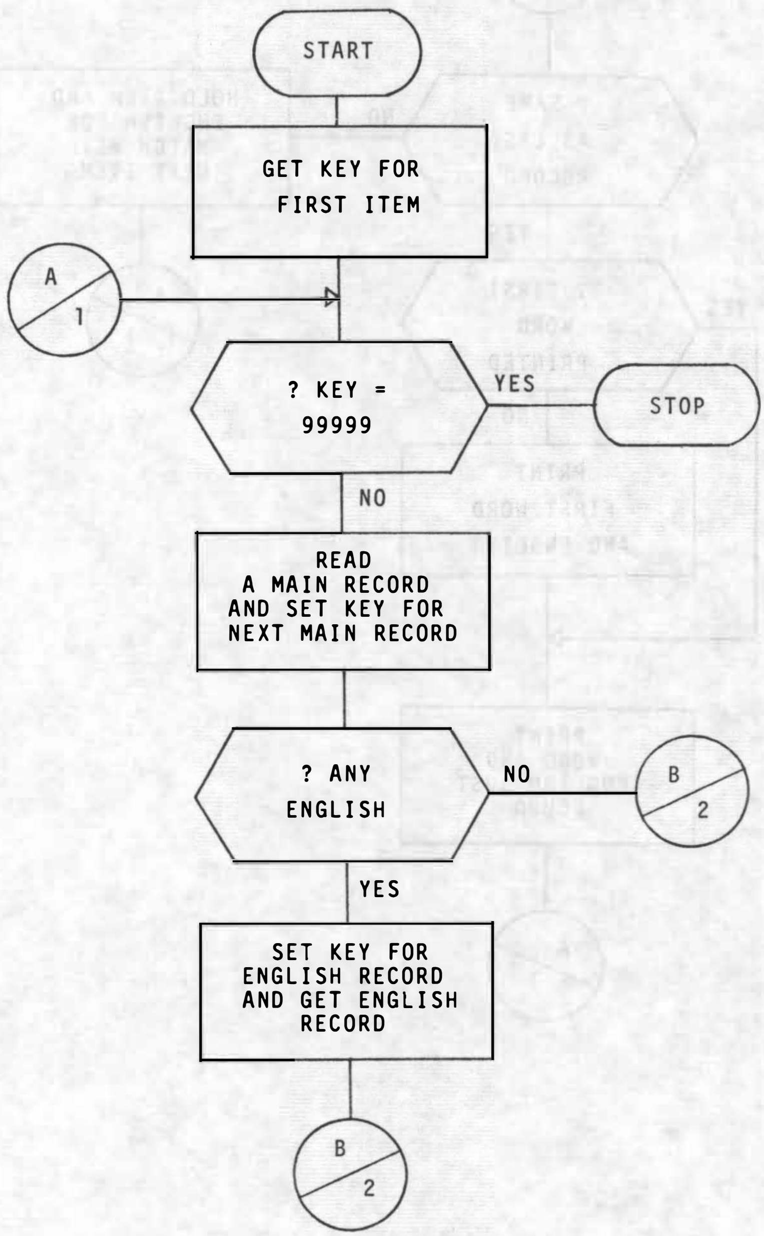
4.4 *LIKE WORDS*

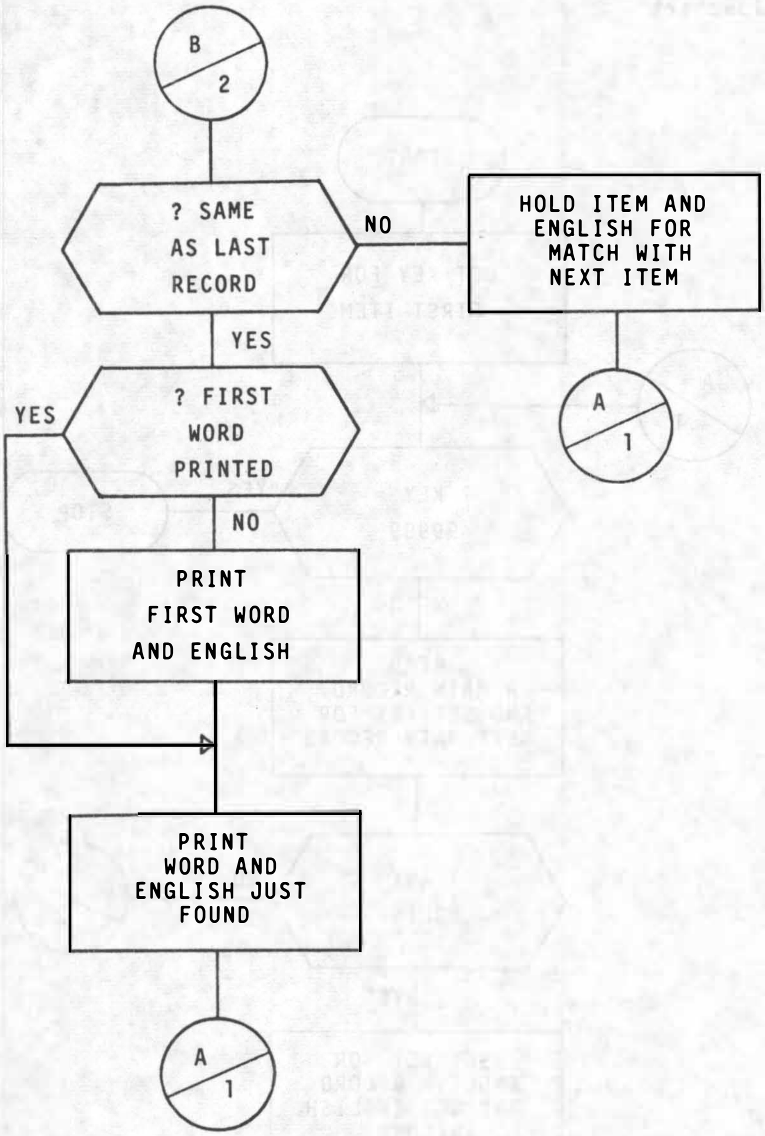
4.4.1 Description of Program

This program produced a list of the items which had the same orthography, ignoring tone positions, plus their English glosses. This program did not require the use of the sort routine because the main file was accessed in sort order. However, PL1 was required to remove tones before comparing items.

The flow chart for this program is in 4.4.2 and the listing in 4.4.3.

4.4.2 Flowchart




B
2
? SAME AS LAST RECORD
NO
HOLD ITEM AND ENGLISH FOR MATCH WITH NEXT ITEM
A
1
YES
? FIRST WORD PRINTED
YES
NO
PRINT FIRST WORD AND ENGLISH
PRINT WORD AND ENGLISH JUST FOUND
A
1

4.4.3 Listing

```
PROG: PROC OPTIONS (MAIN);                                  /*  LIST LIKE WORDS    */
     DCL
          (CON,IND) DEC FIXED (1),
          (JZ,IL) DEC FIXED (5),
          (LAST,KEE,ENGA) CHAR (25),
           LENA CHAR (25),
           CKEY CHAR (8),
           SRA CHAR (40),
           SRB CHAR (80),
           SRC CHAR (120),
           SRD CHAR (160),
           EWD CHAR (16),
           ENGL CHAR (64),
           DUPL CHAR (80),
           EKEY CHAR (8),
          (TOBG,COMT) CHAR (30),
           EGF CHAR (2),
           EGAD CHAR (5),
           ITEM CHAR (133),
           LWD CHAR (6),
           LENG CHAR (64),
           AIGN CHAR (80),
           Y(4) LABEL,
            LY DEC FIXED (2),
           ADDR CHAR (5),
           MWD CHAR (6),
           REST CHAR (89),
           MREC CHAR (265),
           WORD (25) PACKED CHAR (1) DEFINED ENGA,
           PKEY (25) PACKED CHAR (1) DEFINED KEE,
           TYPA FILE RECORD KEYED ENV(REGIONAL(1) F(40)),
           TYPB FILE RECORD KEYED ENV(REGIONAL(1) F(80)),
           TYPC FILE RECORD KEYED ENV(REGIONAL(1) F(120)),
           TYPD FILE RECORD KEYED ENV(REGIONAL(1) F(160)),
           MANE FILE RECORD KEYED ENV(REGIONAL(1) F(265));
     CON = 1;
     LAST='                         ';
     COMT='NO ENGLISH                    ';
     TOBG='SORRY - TOO BIG               ';
     IND=0;
     ADDR='     ';
     OPEN FILE(TYPA) DIRECT INPUT;
     OPEN FILE(TYPB) DIRECT INPUT;
     OPEN FILE(TYPC) DIRECT INPUT;
     OPEN FILE(TYPD) DIRECT INPUT;
     OPEN FILE(MANE) DIRECT INPUT;
     ON ERROR GO TO XYZ;
     JZ =0;
     CKEY=JZ;
                                                            /*  GET ADDRESS OF 1ST MAIN FILE
                                                                ITEM       */
     READ FILE (TYPA) INTO (SRA)  KEY(CKEY);
     GET STRING (SRA) EDIT(MWD,REST) (A(6),A(34));
     IL=MWD*CON;
     IL=IL-1;
     CKEY=IL;
                                                            /*  GET NEXT MAIN FILE ITEM */
REED: IF ADDR='99999' THEN GO TO FIN;
     READ FILE (MANE) INTO (MREC) KEY(CKEY);
```

```
         GET STRING(MREC) EDIT(MWD,ADDR,ENGA,REST,EGF,EGAD,TTEM)
              (A(6),A(5),A(25),A(89),A(2),A(5),A(133));
                                                            /*  SET UP ADDRESS OF NEXT MAIN FILE
                                                                ITEM     */
         IL=ADDR*CON;
         IL=IL-1;
         CKEY=IL;
         IF EGAD < '    1' THEN GO TO AD;
                                                            /*  SET UP ADDRESS OF ENGLISH RECORD
                                                                FOR CURRENT MAIN ITEM   */
         IL=EGAD*CON;
         IL=IL-1;
         EKEY=IL;
         LY=EGF*CON;
         IL=LY-12;
         IF IL > 4 THEN DO; PUT LIST ('ENGL .GT. 160') SKIP;
                            ENGL=TOBG; GO TO AE;   END;
                                                            /*  GET ENGLISH RECORD  */
         GO TO Y(IL);
   Y(1): READ FILE (TYPA) INTO (SRA) KEY (EKEY);
         GET STRING (SRA) EDIT (EWD,ENGL)
              (A(16),A(24));
         GO TO AE;
   Y(2): READ FILE (TYPB) INTO (SRB) KEY (EKEY);
         GET STRING (SRB) EDIT (EWD,ENGL)
              (A(16),A(64));
         GO TO AE;
   Y(3): READ FILE (TYPC) INTO (SRC) KEY (EKEY);
         GET STRING (SRC) EDIT (EWD,ENGL,DUPL)
              (A(16),A(64),A(40));
         GO TO AE;
   Y(4): READ FILE (TYPD) INTO (SRD) KEY (EKEY);
         GET STRING (SRD) EDIT (EWD,ENGL,DUPL)
              (A(16),A(64),A(80));
   AE:
         KEE ='                           ';
                                                            /* REMOVE TONES FROM ENGA
                                                                                       */
         K=1;
         DO J=1 TO 25;
         IF WORD(J)¬= '''' THEN DO; PKEY(K)=WORD(J); K=K+1;   END;
         END;
                                                            /*  CHECK IF EQUAL TO PREVIOUS ENGA
                                                                                       */
         IF KEE=LAST THEN GO TO AA;
         IND=0;
                                                            /*  HOLD RECORDS FOR COMPARISON WITH
                                                                NEXT ENGA   */
   AC:   LAST = KEE;
         LWD=MWD;
         LENA=ENGA;
         LENG=ENGL;
         LL=IL;
         AIGN=DUPL;
         GO TO REED;
   AA:   IF IND=1 THEN GO TO AB;
                                                            /*  WRITE FIRST OF LIKE WORDS
                                                                                       */
         PUT LIST (' ') SKIP;
         PUT EDIT (LWD,LENA,LENG) (SKIP,A(8),A(27),A(64));
```

```
         IF LL > 2 THEN PUT EDIT (AIGN) (SKIP,X(35),A(80));
         IND=1;
                                                                            /*
                                                      WRITE IF NOT FIRST OF LIKE WORDS
                                                                             */
AB:      PUT EDIT (MWD,ENGA,ENGL) (SKIP,A(8),A(27),A(64));
         IF IL > 2 THEN PUT EDIT (DUPL) (SKIP,X(35),A(80));
         GO TO AC;
AD:      ENGL=COMT;
         GO TO AE;
XYZ:                        PUT DATA (MWD,ENGA,EGAD,EGF) SKIP;
                            CLOSE FILE (SYSPRINT);
FIN: PUT LIST ('END OF JOB') PAGE;
         END PROG;
```

4.5 *TONE PATTERNS*

4.5.1 Description of Program

The purpose of this program was to produce sorted lists of Enga words with like tone patterns. Fifteen groups of pattern types were printed, and these are shown in Table 5.1.

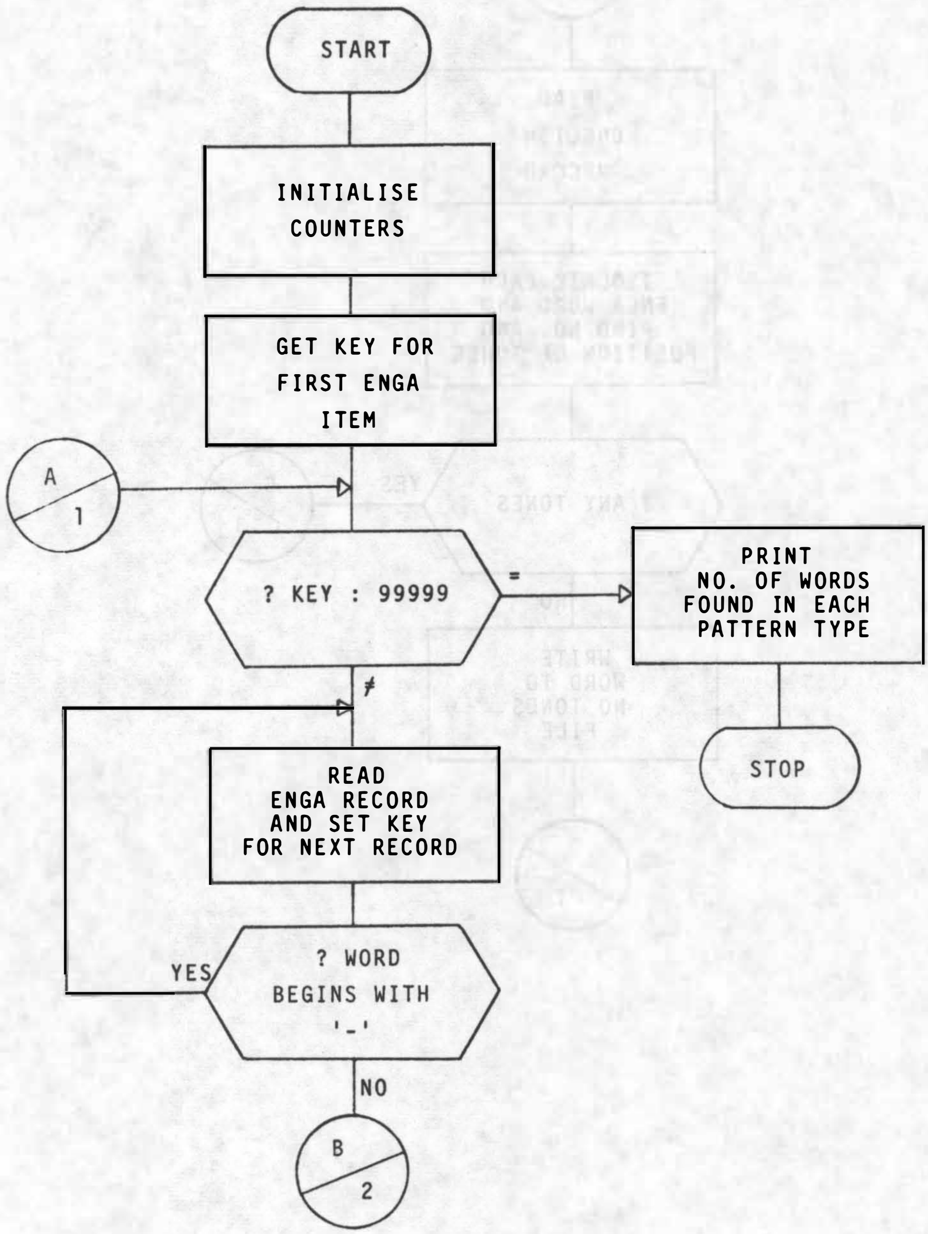
Table 5.1 Tone Pattern Groupings

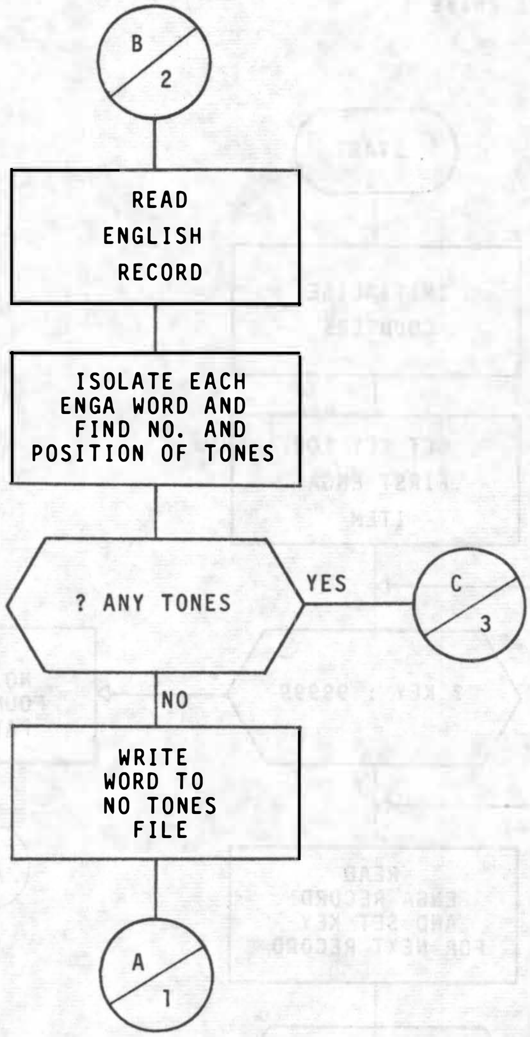
No. of Tones	*Syllable Position*
one	last
one	first
one	second
one	second last
one	third
one	fourth
one	fifth
two	last two
two	first and second
two	second and third
two	third and fourth
two	first and one other
more than two	any position
none	-

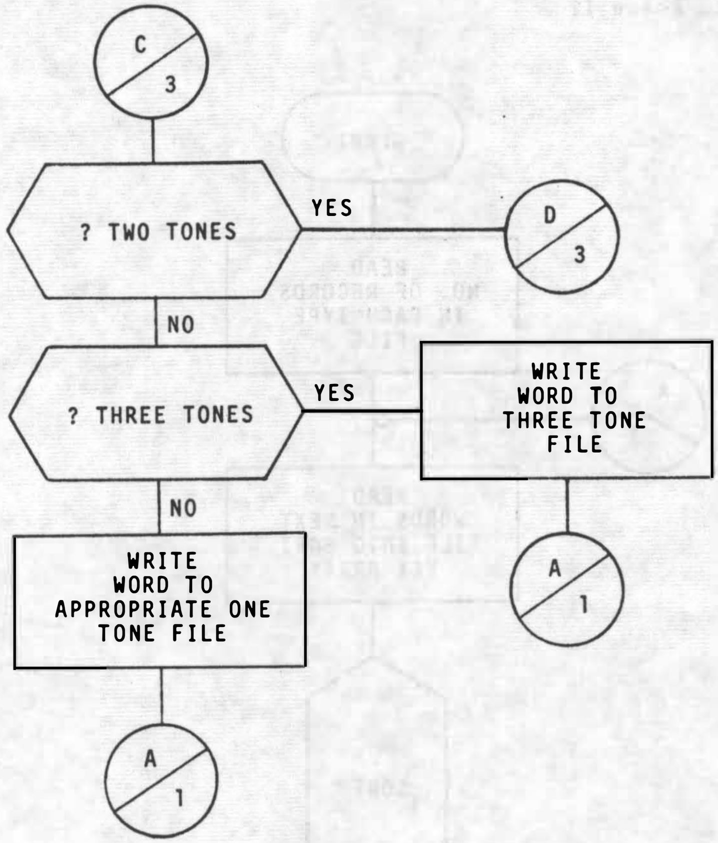
All words in each Enga item were included except for (1) those beginning with a hyphen (particles cf. item 1.10, sub-group 007), and (2) those where the last word of a phrase was a subcategory of verb phrases, since these had been included as single word items.

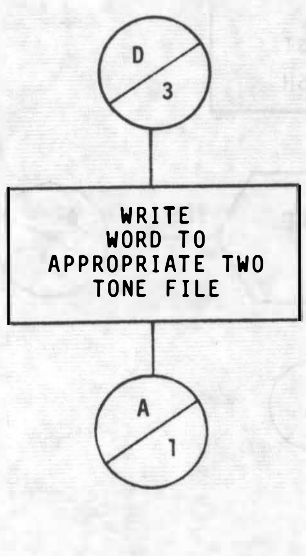
The flow chart is in 4.5.2 and the listing in 4.5.3. The flow chart for the Sort Routine is in 4.1.2.

4.5.2 Flowchart: Phase I

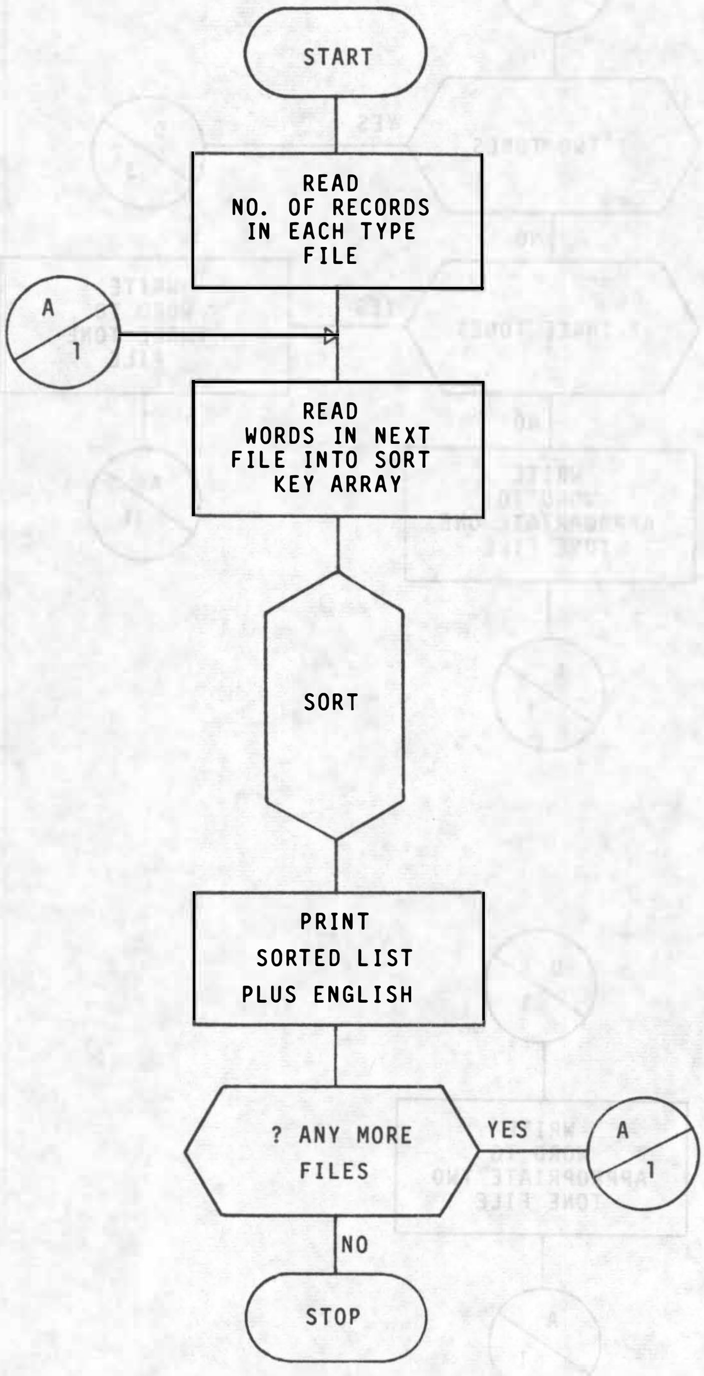



B
2
READ
ENGLISH
RECORD
ISOLATE EACH
ENGA WORD AND
FIND NO. AND
POSITION OF TONES
? ANY TONES
YES
C
3
NO
WRITE
WORD TO
NO TONES
FILE
A
1


C 3
? TWO TONES
YES
D 3
NO
? THREE TONES
YES
WRITE WORD TO THREE TONE FILE
A 1
NO
WRITE WORD TO APPROPRIATE ONE TONE FILE
A 1


D 3
WRITE WORD TO APPROPRIATE TWO TONE FILE
A 1

Flowchart: Phase II



4.5.3 Listings

Phase I

```
PROG: PROC OPTIONS (MAIN);
                                                   /* TONE PATTERNS  (1)  */
     DCL MREC CHAR (265),
            TMN CHAR (9),
         LEST CHAR (133),
         EWD CHAR (16),
         ERA CHAR (40),
         ERB CHAR (80),
         ERC CHAR (120),
         ERD CHAR (160),
         EGF CHAR (2),
         EGAD  CHAR (5),
         NO(15) BINARY FIXED (15),
         Z(4) LABEL,
        (JV,JT) DEC FIXED (2),
         1 FYL,
            2 SIZE CHAR (5),
            2 WORD CHAR (16),
            2 ENGA CHAR (25),
            2 REST CHAR (64),
            2 LAST CHAR (80),
         MWD CHAR (6),
        (EKEY,TKEY) CHAR (8),
         CON DEC FIXED (1),
         INTM CHAR (25),
         DAM CHAR (2),
         TOAN CHAR (55),
         (IFST(8),IFIN(8)) DEC FIXED (2),
         IL DEC FIXED (5),
         DNGA(25) PACKED CHAR (1) DEFINED ENGA,
         ARR(4) DEC FIXED (2),
       (SENG,SPA) CHAR (25),
         SNGA (25) CHAR (1) DEFINED SENG,
         ADDE CHAR (5),
         TONE FILE RECORD KEYED ENV(REGIONAL(1) F(190)),
         TTWO FILE RECORD KEYED ENV(REGIONAL(1) F(190)),
         TTHR FILE RECORD KEYED ENV(REGIONAL(1) F(190)),
         TFOR FILE RECORD KEYED ENV(REGIONAL(1) F(190)),
         TFIV FILE RECORD KEYED ENV(REGIONAL(1) F(190)),
         TSIX FILE RECORD KEYED ENV(REGIONAL(1) F(190)),
         TSVN FILE RECORD KEYED ENV(REGIONAL(1) F(190)),
         TEIT FILE RECORD KEYED ENV(REGIONAL(1) F(190)),
         TNIN FILE RECORD KEYED ENV(REGIONAL(1) F(190)),
         TTEN FILE RECORD KEYED ENV(REGIONAL(1) F(190)),
         TELV FILE RECORD KEYED ENV(REGIONAL(1) F(190)),
         TTWL FILE RECORD KEYED ENV(REGIONAL(1) F(190)),
         TTHT FILE RECORD KEYED ENV(REGIONAL(1) F(190)),
         TFOT FILE RECORD KEYED ENV(REGIONAL(1) F(190)),
         TFIF FILE RECORD KEYED ENV(REGIONAL(1) F(190)),
         TYPA FILE RECORD KEYED ENV(REGIONAL(1) F(40)),
         TYPB FILE RECORD KEYED ENV(REGIONAL(1) F(80)),
         TYPC FILE RECORD KEYED ENV(REGIONAL(1) F(120)),
         TYPD FILE RECORD KEYED ENV(REGIONAL(1) F(160)),
         MANE FILE RECORD KEYED ENV(REGIONAL(1) F(265));
     NO=0;
     ON ERROR GO TO ERR;
     CON=1;
     EKEY=IL;
     OPEN FILE(TONE) DIRECT OUTPUT;  OPEN FILE(TTWO) DIRECT OUTPUT;
     OPEN FILE(TTHR) DIRECT OUTPUT;  OPEN FILE(TFOR) DIRECT OUTPUT;
```

```
      OPEN FILE(TFIV) DIRECT OUTPUT;  OPEN FILE(TSIX) DIRECT OUTPUT;
      OPEN FILE(TSVN) DIRECT OUTPUT;  OPEN FILE(TFIT) DIRECT OUTPUT;
      OPEN FILE(TNIN) DIRECT OUTPUT;  OPEN FILE(TTEN) DIRECT OUTPUT;
      OPEN FILE(TELV) DIRECT OUTPUT;  OPEN FILE(TTWL) DIRECT OUTPUT;
      OPEN FILE(TTHT) DIRECT OUTPUT;  OPEN FILE(TFCT) DIRECT OUTPUT;
      OPEN FILE(TFIF) DIRECT OUTPUT;
      OPEN FILE (MANF) DIRECT INPUT;
      OPEN FILE (TYPA) DIRECT INPUT;
      OPEN FILE (TYPB) DIRECT INPUT;
      OPEN FILE (TYPC) DIRECT INPUT;
      OPEN FILE (TYPD) DIRECT INPUT;
                                                  /*  GET ADDRESS OF 1ST MAIN FILE
                                                      ITEM        */
      READ FILE (TYPA) INTO (ERA) KEY (EKEY);
      GET STRING (ERA) EDIT (MWD,LEST) (A(6),A(34));
      IL=MWD*CON;
      IL=IL-1;
      EKEY=IL;
      GET EDIT (ADDE,SPA,TOAN) (A(5),A(25),A(50));
                                                  /*  GET NEXT MAIN FILE ITEM */
REED:
       IF ADDE = '99999' THEN GO TO NXT;
      READ FILE (MANF) INTO (MREC) KEY (EKEY);
      GET STRING (MREC) EDIT (MWD,ADDE,ENGA,SENG, TOAN,DAM,
         INTM,EGF,EGAD,LEST)
           (A(6),A(5),A(25),A(7),A(55),A(2),A(25),A(2),A(5),A(133));
                                                  /*  SET UP ADDRESS OF NEXT MAIN FILE
                                                      ITEM        */
      IL=ADDE*CON;
      IL=IL-1;
      EKEY=IL;

                                                  /*  IGNORE IF ENGA BEGINS WITH -
                                                                              */
      IF ONGA(1) = '-' THEN GO TO REED;
      IF ENGA = SPA THEN DO; PUT LIST ('NO ENGA FOR') SKIP;
                       PUT EDIT (MWD,ADDE,ENGA,SENG)
                                      (A(6),A(5),A(25),A(7));
                       GO TO REED;
                       END;
                                                  /*  SET UP ADDRESS OF ENGLISH FOR
                                                      CURRENT MAIN ITEM  */
      IL=EGAD*CON;
      IF IL < 1 THEN DO; REST=SPA;
                         GO TO AE;    END;
      IL=IL-1;
      TKEY=IL;
      IL=EGF*CON;
      IL=IL-12;

                                                  /*  GET ENGLISH RECORD  */
      GO TO Z(IL);
Z(1):  READ FILE(TYPA) INTO (ERA)  KEY (TKEY);
      GET STRING (ERA)  EDIT (EWD,REST)
           (A(16),A(24));
      LAST=SPA;
      GO TO AE;
Z(2):  READ FILE(TYPB) INTO (ERB)  KEY (TKEY);
      GET STRING (ERB)  EDIT (EWD,REST)
           (A(16),A(64));
      LAST=SPA;
      GO TO AE;
```

```
Z(3):  READ FILE(TYPC) INTO (ERC)   KEY (TKEY);
     GET STRING (ERC)  EDIT (EWD,REST,LAST)
            (A(16),A(64),A(40));
     GO TO AE;
Z(4):  READ FILE(TYPD) INTO (ERD)   KEY (TKEY);
     GET STRING (ERD)  EDIT (EWD,REST,LAST)
            (A(16),A(64),A(80));
AE:
                     /* FIND LAST CHAR IN EACH WORD  */
     IFIN=0;
     K=1;
     DO J=1 TO 25;
     IF DNGA(J) ¬= ' ' THEN GO TO AA;
     IF DNGA(J-1)  = ' ' THEN GO TO AR;
     IFIN(K)=J-1;
     K=K+1;
     GO TO AR;
AA:  IF J = 25 THEN DO; IFIN(K) =J;  END;
AR:  END;
                     /* FIND FIRST CHAR IN EACH WORD  */
     IFST=0;
     IFST(1)=1;
     K=2;
     DO J=2 TO 25;
     IF DNGA(J)  = ' ' THEN GO TO AS;
     IF DNGA(J-1) ¬= ' ' THEN GO TO AS;
     IFST(K)=J;
     K=K+1;
AS:  END;
                     /*   FIND NUMBER OF WORDS   */
     DO J=1 TO 8;
     IF IFIN(J) > 0 THEN GO TO AY;
     IF IFST(J) = 0 THEN DO; JB=J-1;  GO TO AI;  END;
AZ:  PUT EDIT ('ERR IN WORD POSTIIONS',(IFST(K),IFIN(K) DO K=1 TO 8),
        ENGA)   (SKIP,A,16 F(4),X(5),A(25));
     GO TO REED;
AY:  IF IFST(J) = 0 THEN GO TO AZ;
     END;
                     /*   FIND TONE PATTERNS BY FINDING VOWELS AND SETTING UP
                                   THEIR POSITIONS IN AN ARRAY AND
                                   HOLD A COUNT      */
AI:  DO I=1 TO JB;
     JA=IFIN(I)+1-IFST(I);
     JV=IFST(I);
     SENG=SUBSTR(ENGA,JV,JA);
     ARR=0;
     K=1;
     JV=0;
     DO J=1 TO JA;
     IF SNGA(J)='A' THEN GO TO AC;
     IF SNGA(J)='E' THEN GO TO AC;
     IF SNGA(J)='I' THEN GO TO AC;
     IF SNGA(J)='O' THEN GO TO AC;
     IF SNGA(J)='U' THEN GO TO AC;
     GO TO AD;
AC:  JV=JV+1;
     IF SNGA(J+1) = '''' THEN DO; ARR(K)=JV;  K=K+1;  END;
AD:  END;
     IF JV=0  THEN DO;  PUT EDIT ('NO VOWELS IN ',MWD,ENGA,TOAN)
                        (SKIP,A,A(6),A(28),A(55));
```

```
                        GO TO REED;  END;
     WORD=SENG;
     JT=JA-1;
     SIZE=JT;
                                                                     /*
                                                  NO TONES     */
     IF  K=1  THEN DO; NO(15)=NO(15)+1;  IL=NO(15);  TKEY=IL;
                       WRITE FILE (TFIF) FROM (FYL) KEYFROM (TKEY);
                            GO TO RED;    END;
     IF K=3 THEN GO TO AF;
                                                                     /*
                                                  MORE THAN TWO TONES  */
     IF K=4 THEN DO; NO(14)=NO(14)+1; IL=NO(14); TKEY=IL;
                     WRITE FILE (TFOT) FROM (FYL) KEYFROM (TKEY);
                          GO TO RED;    END;
                                                                 /*  */
                     /*  ONE TONE ONLY  */
     IF ARR(1) =JV   THEN DO; NO(1)=NO(1)+1; IL=NO(1); TKEY=IL;
                     WRITE FILE (TONE) FROM (FYL) KEYFROM (TKEY);
                          GO TO RED;    END;
     IF ARR(1) =1    THEN DO; NO(2)=NO(2)+1; IL=NO(2); TKEY=IL;
                     WRITE FILE (TTWO) FROM (FYL) KEYFROM (TKEY);
                          GO TO RED;    END;
     IF ARR(1) =2    THEN DO; NO(3)=NO(3)+1; IL=NO(3); TKEY=IL;
                     WRITE FILE (TTHR) FROM (FYL) KEYFROM (TKEY);
                          GO TO RED;    END;
     IF ARR(1) =JV-1 THEN DO; NO(4)=NO(4)+1; IL=NO(4); TKEY=IL;
                     WRITE FILE (TFOR) FROM (FYL) KEYFROM (TKEY);
                          GO TO RED;    END;
     IF ARR(1) =3    THEN DO; NO(5)=NO(5)+1; IL=NO(5); TKEY=IL;
                     WRITE FILE (TFIV) FROM (FYL) KEYFROM (TKEY);
                          GO TO RED;    END;
     IF ARR(1) =4    THEN DO; NO(6)=NO(6)+1; IL=NO(6); TKEY=IL;
                     WRITE FILE (TSIX) FROM (FYL) KEYFROM (TKEY);
                          GO TO RED;    END;
     IF ARR(1) =5    THEN DO; NO(7)=NO(7)+1; IL=NO(7); TKEY=IL;
                     WRITE FILE (TSVN) FROM (FYL) KEYFROM (TKEY);
                          GO TO RED;    END;
     PUT EDIT ('SHD HAVE FILED BY NOW ',MWD,ENGA,TOAN)
                          (SKIP,A,A(6),A(28),A(55));
     GO TO REED;
                                                                 /*  */
                     /*  TWO  TONES  */
AF:  IF ARR(2)¬=JV   THEN GO TO AP;
     IF ARR(1)=JV-1  THEN DO; NO(8)=NO(8)+1; IL=NO(8); TKEY=IL;
                     WRITE FILE (TEIT) FROM (FYL) KEYFROM (TKEY);
                          GO TO RED;    END;
     IF ARR(1) = 1 THEN GO TO AO;
     GO TO AQ;
AP:  IF ARR(1)¬= 1 THEN GO TO AM;
     IF ARR(2) = 2 THEN DO; NO(9)=NO(9)+1; IL=NO(9); TKEY=IL;
                     WRITE FILE (TNIN) FROM (FYL) KEYFROM (TKEY);
                          GO TO RED;    END;
     GO TO AO;
AM:  IF ARR(1)¬= 2 THEN GO TO AN;
     IF ARR(2) = 3   THEN DO; NO(10)=NO(10)+1; IL=NO(10); TKEY=IL;
                     WRITE FILE (TTEN) FROM (FYL) KEYFROM (TKEY);
                          GO TO RED;    END;
AN:  IF ARR(1)¬= 3 THEN GO TO AO;
     IF ARR(2) = 4   THEN DO; NO(11)=NO(11)+1; IL=NO(11); TKEY=IL;
```

```
                        WRITE FILE (TELV) FROM (FYL) KEYFROM (TKEY);
                              GO TO RED;     END;
     GO TO AQ;
AO:   NO(12)=NO(12)+1; IL=NO(12); TKEY=IL;
                        WRITE FILE (TTWL) FROM (FYL) KEYFROM (TKEY);
     GO TO RED;
AQ:  NO(13)=NO(13)+1;
     IL=NO(13);
     TKEY=IL;
                        WRITE FILE (TTHT) FROM (FYL) KEYFROM (TKEY);
                                                                   /*
                                                   CHECK IF VERB ENT TO BE OMITTED
                                                                   */
RED: IF JB = 1 THEN GO TO REED;
      IF I ¬= JB-1 THEN GO TO AJ;
      IF DAM ¬= '  ' THEN GO TO REED;
AJ:  END;
     GO TO REED;
ERR: PUT DATA (MWD,ENGA,REST) SKIP;
     PUT EDIT (NO(J),FYL) (SKIP,F(5),A(5),A(16),A(25),A(64),SKIP,X(20),
         A(40));
NXT:
                                                /*  PRINT COUNT FO NUMBER IN EACH
                                                    PATTERN TYPE   */
     PUT EDIT((NO(J) DO J=1 TO 15)) (SKIP,15F(7));
FIN: END PROG;
```

```
PROG: PROC OPTIONS (MAIN);
                                                      /*  TONE PATTERNS (2)  */
    DCL
        KEE CHAR (21),
        CKEE CHAR (16) DEFINED KEE POSITION (6),
        ADDR(1555) DEC FIXED (5),
        KEARY(1555) CHAR (21),
        (NG,MG,MX,    VO,VL,VP) DEC FIXED (5),
        (NN,JZ,NC,N,M,L) DEC FIXED (5),
        SPA CHAR (20),
        TKEY CHAR (8),
        NO(15) BINARY FIXED (15),
        Y (15) LABEL,
        1 FYL,
          2 SIZE CHAR (5),
          2 WORD CHAR (16),
          2 ENGA CHAR (25),
          2 REST CHAR (64),
          2 LAST CHAR (80),
        IL DEC FIXED (5),
        TEMP FILE RECORD KEYED ENV(REGIONAL(1) F(190)),
        TONE FILE RECORD KEYED ENV(REGIONAL(1) F(190)),
        TTWO FILE RECORD KEYED ENV(REGIONAL(1) F(190)),
        TTHR FILE RECORD KEYED ENV(REGIONAL(1) F(190)),
        TFOR FILE RECORD KEYED ENV(REGIONAL(1) F(190)),
        TFIV FILE RECORD KEYED ENV(REGIONAL(1) F(190)),
        TSIX FILE RECORD KEYED ENV(REGIONAL(1) F(190)),
        TSVN FILE RECORD KEYED ENV(REGIONAL(1) F(190)),
        TEIT FILE RECORD KEYED ENV(REGIONAL(1) F(190)),
        TNIN FILE RECORD KEYED ENV(REGIONAL(1) F(190)),
        TTEN FILE RECORD KEYED ENV(REGIONAL(1) F(190)),
        TELV FILE RECORD KEYED ENV(REGIONAL(1) F(190)),
        TTWL FILE RECORD KEYED ENV(REGIONAL(1) F(190)),
        TTHT FILE RECORD KEYED ENV(REGIONAL(1) F(190)),
        TFOT FILE RECORD KEYED ENV(REGIONAL(1) F(190)),
        TFIF FILE RECORD KEYED ENV(REGIONAL(1) F(190));
                                                              /*
                                                              READ NUMBER OF RECORDS IN EACH
                                                              TONE PATTERN TYPE   */
    GET EDIT ((NO(J) DO J=1 TO 15),SPA)  (15 F(5),A(5));
    PUT LIST (' ') PAGE;
                                                      /*  LOOP FOR EACH PATTERN TYPE.
                                                          EACH Y PARAGRAPH BELOW READS IN
                                                          RECORDS FOR A PATTERN TYPE AND
                                                          SETS UP THE SORT KEY ARRAY
                                                                                    */
    DO J=1 TO 15;
    IF NO(J)=0 THEN GO TO AG;
    OPEN FILE (TEMP) DIRECT OUTPUT;
    NN=NO(J);
    GO TO Y(J);
Y(1):  CLOSE FILE(TONE);
    OPEN FILE (TONE) DIRECT INPUT;
    PUT LIST ('TONE ON LAST') PAGE;
    DO L=1 TO NN;
    TKEY=L;
    READ FILE (TONE) INTO (FYL) KEY (TKEY);
    KEE=SIZE;
    CKEE=WORD;
    KEARY(L)=KEE;
```

```
      WRITE FILE (TEMP) FROM (FYL) KEYFROM (TKEY);
      END;
      GO TO AH;
Y(2):  CLOSE FILE(TTWO);
      OPEN FILE (TTWO) DIRECT INPUT;
      PUT LIST ('TONE ON 1ST') PAGE;
      DO L=1 TO NN;
      TKEY=L;
      READ FILE (TTWO) INTO (FYL) KEY (TKEY);
      KEE=SIZE;
      CKEE=WORD;
      KEARY(L)=KEE;
      WRITE FILE (TEMP) FROM (FYL) KEYFROM (TKEY);
      END;
      GO TO AH;
Y(3):  CLOSE FILE(TTHR);
      OPEN FILE (TTHR) DIRECT INPUT;
      PUT LIST ('TONE ON 2ND') PAGE;
      DO L=1 TO NN;
      TKEY=L;
      READ FILE (TTHR) INTO (FYL) KEY (TKEY);
      KEE=SIZE;
      CKEE=WORD;
      KEARY(L)=KEE;
      WRITE FILE (TEMP) FROM (FYL) KEYFROM (TKEY);
      END;
      GO TO AH;
Y(4):  CLOSE FILE(TFOR);
      OPEN FILE (TFOR) DIRECT INPUT;
      PUT LIST ('TONE ON 2ND LAST') PAGE;
      DO L=1 TO NN;
      TKEY=L;
      READ FILE (TFOR) INTO (FYL) KEY (TKEY);
      KEE=SIZE;
      CKEE=WORD;
      KEARY(L)=KEE;
      WRITE FILE (TEMP) FROM (FYL) KEYFROM (TKEY);
      END;
      GO TO AH;
Y(5):  CLOSE FILE(TFIV);
      OPEN FILE (TFIV) DIRECT INPUT;
      PUT LIST ('TONE ON 3RD') PAGE;
      DO L=1 TO NN;
      TKEY=L;
      READ FILE (TFIV) INTO (FYL) KEY (TKEY);
      KEE=SIZE;
      CKEE=WORD;
      KEARY(L)=KEE;
      WRITE FILE (TEMP) FROM (FYL) KEYFROM (TKEY);
      END;
      GO TO AH;
Y(6):  CLOSE FILE(TSIX);
      OPEN FILE (TSIX) DIRECT INPUT;
      PUT LIST ('TONE ON 4TH') PAGE;
      DO L=1 TO NN;
      TKEY=L;
      READ FILE (TSIX) INTO (FYL) KEY (TKEY);
      KEE=SIZE;
      CKEE=WORD;
      KEARY(L)=KEE;
```

```
      WRITE FILE (TEMP) FROM (FYL) KEYFROM (TKEY);
      END;
      GO TO AH;
Y(7):  CLOSE FILE(TSVN);
      OPEN FILE (TSVN) DIRECT INPUT;
      PUT LIST ('TONE ON 5TH') PAGE;
      DO L=1 TO NN;
      TKEY=L;
      READ FILE (TSVN) INTO (FYL) KEY (TKEY);
      KEE=SIZE;
      CKEE=WORD;
      KEARY(L)=KEE;
      WRITE FILE (TEMP) FROM (FYL) KEYFROM (TKEY);
      END;
      GO TO AH;
Y(8):  CLOSE FILE(TEIT);
      OPEN FILE (TEIT) DIRECT INPUT;
      PUT LIST ('TWO TONES ON LAST TWO') PAGE;
      DO L=1 TO NN;
      TKEY=L;
      READ FILE (TEIT) INTO (FYL) KEY (TKEY);
      KEE=SIZE;
      CKEE=WORD;
      KEARY(L)=KEE;
      WRITE FILE (TEMP) FROM (FYL) KEYFROM (TKEY);
      END;
      GO TO AH;
Y(9):  CLOSE FILE(TNIN);
      OPEN FILE (TNIN) DIRECT INPUT;
      PUT LIST ('TWO TONES ON 1ST AND 2ND') PAGE;
      DO L=1 TO NN;
      TKEY=L;
      READ FILE (TNIN) INTO (FYL) KEY (TKEY);
      KEE=SIZE;
      CKEE=WORD;
      KEARY(L)=KEE;
      WRITE FILE (TEMP) FROM (FYL) KEYFROM (TKEY);
      END;
      GO TO AH;
Y(10): CLOSE FILE(TTEN);
      OPEN FILE (TTEN) DIRECT INPUT;
      PUT LIST ('TWO TONES ON 2ND AND 3RD') PAGE;
      DO L=1 TO NN;
      TKEY=L;
      READ FILE (TTEN) INTO (FYL) KEY (TKEY);
      KEE=SIZE;
      CKEE=WORD;
      KEARY(L)=KEE;
      WRITE FILE (TEMP) FROM (FYL) KEYFROM (TKEY);
      END;
      GO TO AH;
Y(11): CLOSE FILE(TELV);
      OPEN FILE (TELV) DIRECT INPUT;
      PUT LIST ('TWO TONES ON 3RD AND 4TH') PAGE;
      DO L=1 TO NN;
      TKEY=L;
      READ FILE (TELV) INTO (FYL) KEY (TKEY);
      KEE=SIZE;
      CKEE=WORD;
      KEARY(L)=KEE;
```

```
      WRITE FILE (TEMP) FROM (FYL) KEYFROM (TKEY);
      END;
      GO TO AH;
Y(12): CLOSE FILE(TTWL);
      OPEN FILE (TTWL) DIRECT INPUT;
      PUT LIST ('TONES ON 1ST AND 1 OTHER') PAGE;
      DO L=1 TO NN;
      TKEY=L;
      READ FILE (TTWL) INTO (FYL) KEY (TKEY);
      KEE=SIZE;
      CKEE=WORD;
      KEARY(L)=KEE;
      WRITE FILE (TEMP) FROM (FYL) KEYFROM (TKEY);
      END;
      GO TO AH;
Y(13): CLOSE FILE(TTHT);
      OPEN FILE (TTHT) DIRECT INPUT;
      PUT LIST ('REMAINDER WITH TWO TONES') PAGE;
      DO L=1 TO NN;
      TKEY=L;
      READ FILE (TTHT) INTO (FYL) KEY (TKEY);
      KEE=SIZE;
      CKEE=WORD;
      KEARY(L)=KEE;
      WRITE FILE (TEMP) FROM (FYL) KEYFROM (TKEY);
      END;
      GO TO AH;
Y(14): CLOSE FILE(TFOT);
      OPEN FILE (TFOT) DIRECT INPUT;
      PUT LIST ('MORE THAN TWO TONES') PAGE;
      DO L=1 TO NN;
      TKEY=L;
      READ FILE (TFOT) INTO (FYL) KEY (TKEY);
      KEE=SIZE;
      CKEE=WORD;
      KEARY(L)=KEE;
      WRITE FILE (TEMP) FROM (FYL) KEYFROM (TKEY);
      END;
      GO TO AH;
Y(15): CLOSE FILE(TFIF);
      OPEN FILE (TFIF) DIRECT INPUT;
      PUT LIST ('WORDS WITHOUT TONES') PAGE;
      DO L=1 TO NN;
      TKEY=L;
      READ FILE (TFIF) INTO (FYL) KEY (TKEY);
      KEE=SIZE;
      CKEE=WORD;
      KEARY(L)=KEE;
      WRITE FILE (TEMP) FROM (FYL) KEYFROM (TKEY);
      END;
                                                        /*    SORT     */
AH:   IF NN <16 THEN DO; NG=NN; GO TO AK;  END;
      NG=.5*(SQRT(5*NN));
AK:   NC=1;
      MG=0;
CC:   IF NC>NN THEN GO TO CA;
      N=NC;
      MX=NC;
      JZ=NC;
      KEE=KEARY(NC);
```

```
       IL=NC;
       NC=NC+NG;
       IF NC>NN THEN DO;  NC=NN+1;   END;
       ADDR(IL)=NC;
AT:    N=N+1;
       IF N>NC THEN GO TO AX;
       IF N=NC THEN GO TO AX;
       M=IL;
       IF KEARY(N)>KEE THEN GO TO AU;
       IF KEARY(N)=KEE THEN GO TO AU;
       ADDR(N)=JZ;
       KEE=KEARY(N);
       JZ=N;
       GO TO AT;
AU:    IF KEARY(N)>KEARY(M) THEN GO TO AV;
       IF KEARY(N)=KEARY(M) THEN GO TO AV;
       M=M+1;
       GO TO AU;
AV:    IF ADDR(M)¬=NC THEN GO TO AW;
       ADDR(M)=N;
       ADDR(N)=NC;
       MX=N;
       GO TO AT;
AW:    L=M;
       M=ADDR(M);
       IF KEARY(N)>KEARY(M) THEN GO TO AV;
       IF KEARY(N)=KEARY(M) THEN GO TO AV;
       ADDR(N)=M;
       ADDR(L)=N;
       GO TO AT;
AX:    ADDR(MX)=9999;
       IF MG¬=0 THEN GO TO CB;
       MG=JZ;
       GO TO CC;
                                                  /*   MERGE    */
CB:    VO=MG;
       VL=JZ;
       IF KEARY(VO)<KEARY(VL) THEN GO TO CD;
       IF KEARY(VO)=KEARY(VL) THEN GO TO CD;
       MG=VL;
       VP=VL;
       VL=ADDR(VL);
       GO TO CE;
CD:    MG=VO;
       VP=VO;
       VO=ADDR(VO);
CE:    IF KEARY(VO)<KEARY(VL) THEN GO TO CF;
       IF KEARY(VO)=KEARY(VL) THEN GO TO CF;
       ADDR(VP)=VL;
       VP=VL;
       VL=ADDR(VL);
       IF VL=9999 THEN GO TO CG;
       GO TO CE;
CF:    ADDR(VP)=VO;
       VP=VO;
       VO=ADDR(VO);
       IF VO=9999 THEN GO TO CH;
       GO TO CE;
CG:    ADDR(VP)=VO;
       GO TO CC;
```

```
CH:   ADDR(VP)=VL;
      GO TO CC;
CA:   TKEY=MG;
      L=MG;
      OPEN FILE (TEMP) DIRECT INPUT;
                                                        /*
                                          PRINT EACH PATTERN TYPE IN
                                          SORTED ORDER     */
      DO N=1 TO NN;
      READ FILE (TEMP) INTO (FYL)  KEY (TKEY);
      PUT EDIT (WORD,ENGA,REST) (SKIP,A(17),A(26),X(3),A(64));
      IF LAST=SPA THEN GO TO CI;
      PUT EDIT (LAST) (SKIP,X(46),A(80));
CI:   TKEY=ADDR(L);
      L=ADDR(L);
      IF L=9999 THEN GO TO AA;
      END;
AA:   CLOSE FILE (TEMP);
AG:
      END;
      END PROG;
```


Rose, M.L. "Pl/1 Retrieval Techniques". In Lang, A., Mather, K.E.W. and Rose, M.L. editors, *Information storage and retrieval: A dictionary project.*
D-8:77-147. Pacific Linguistics, The Australian National University, 1972. DOI:*10.15144/PL-D8.77*

APPENDIX A

COMPUTATIONAL TERMS

ACCESS TIME The time taken to retrieve data from a storage device, measured from the instant of executing an instruction to call for the data to the moment when the data is stored in the specified location within the computor.

ADDRESS The identification of the position of a location in store.

BACK-UP STORAGE A duplicate or secondary data storage, to be used should the data on the disk happen to be erased.

CODING The writing of instructions or data for a computer.

CODING SHEETS A form on which data or computer instructions are written before being transferred to an input medium, e.g. punched cards (cf. 1.7.1).

CONTROL CARDS A punched card containing data required for control purposes, e.g. totals.

CORE STORE A type of memory composed of magnetic cores; in this case, the main computer memory.

CREATION The initial data collection and organisation of this raw data into a file (e.g. disk file).

DIRECT ACCESS A storage device in which the access time to retrieve items of data is constant, relatively short, and independent of the location previously addressed. For this project, a disk.

DISK A storage device consisting of a number of flat circular plates each coated on both surfaces with some magnetizable material.

(back matter) Lang, A., Mather, K.E.W. and Rose, M.L. editors. *Information storage and retrieval: A dictionary project.*
D-8, viii + 158 pages. Pacific Linguistics, The Australian National University, 1972. DOI:*10.15144/PL-D8.back*

DISK FILE A file stored on a magnetic disk.

EDIT To check data for errors.

FEEDBACK The use of information produced at one stage in a series of operations as input to a preceding stage.

FIELD A subdivision of a record or collection of data items which contains a unit of information. In this project, each item (1.3.1-1.3.3) was assigned a particular field (2.2.5).

FILE An organised collection of data units or items, in this project kept on disk.

FILE DESIGN The organisation of a collection of raw data items into receptacles for holding information (the data) and indexed to help storage and retrieval of data items.

FLOWCHART The diagrammatic representation of a sequence of events, usually drawn with conventional symbols representing different types of events and their interconnection. Flowcharts are used to show diagrammatically the logical relation between successive steps in a computer program.

FORMAT The predetermined arrangement of data, e.g. the layout of the coding sheet, the arrangement of data in a record.

FORTRAN IV/G A programming language for scientific and mathematical use.

LISTING A printout of the program statements.

MASTER FILE A file of reference data which is used to provide data on a routine basis; a current fully updated file to which change records of new data items are applied. In this project, the master file was kept in disk storage.

MERGE An operation performed on two, or more, ordered sets of records to create a single set or file.

OVERWRITE To place information in a location and destroy the information previously contained there.

PL/1 A programming language which allows efficient handling of large volumes of data and character arrangement.

POINTER The identification of the position of a location in store.

PRINTOUT A general term for the output from a printer; printed pages produced by a printer.

READ To obtain data from one form of store (e.g. punched cards) and transfer it to another (e.g. disk or core store).

RECORD A set of related fields treated as a unit. The set of fields related to any one event. In this project, all data fields accompanying one Enga item.

RETRIEVAL The extraction of data from a file or files by searching for specific items contained in records stored on the file.

RUN The performance of one program or routine.

SEQUENTIAL DISK FILES Files, stored on a disk, which can be accessed only in a one after the other order starting with the first record on the file.

SORT To arrange items of information into groups according to some particular criteria, in this project, in alphabetical order.

SORT ORDER The criteria used to arrange items into ordered groups, in this project, alphabetical order.

STORAGE The process of allocating specific areas of a device capable of receiving information and retaining it over a period of time (in this project a disk), and that allows the information to be retrieved and used when required.

THREADED LIST SORT A technique for sorting data where an address or pointer to the next record in order is set up for each record. These addresses then enable ordered movement through the data, but the data itself is never rearranged.

UPDATING To correct, add or delete records and thus ensure that the file reflects the latest information.

BIBLIOGRAPHY

CASAGRANDE, Joseph B. and Kenneth L. Hale

1967 "Semantic relationships in Papago folk-definitions". In Hymes and Bittle (eds.): 165-93.

DRAPER, Sheila

n.d. *English-Kyaka Dictionary.* Mimeo.

GLEASON, H.A.

1955 "Review of Mager, Gedaged-English Dictionary". *Language* 31:163-5.

HYMES, Dell and W.E. Bittle (eds.)

1967 *Studies in Southwestern Ethnolinguistics: Meaning and History in the Languages of the American Southwest.* The Hague: Mouton and Co.

LANG, Adrianne

1971 Nouns and Classificatory Verbs in Enga (New Guinea): A Semantic Study. PhD thesis, The Australian National University.

forthcoming: *Enga Dictionary with English Index. Pacific Linguistics, C20.*

MATHIOT, Madeline

1967 "The Place of the Dictionary in Linguistic Description", *Language* 43:703-24.