AD NUMBER

AD838725

NEW LIMITATION CHANGE

TO

Approved for public release, distribution
unlimited

FROM

Distribution authorized to U.S. Gov't.
agencies and their contractors; Critical
Technology; JUN 1958. Other requests shall
be referred to Commanding Officer,
Edgewood Arsenal, Attn: SMUEA-TSTI-T,
Edgewood Arsenal, MD 21010.

AUTHORITY

USAEA notice, 11 Dec 1968

AD 688725

CIDS No. 5

COMPUTER PROGRAMMING FOR AN EXPERIMENTAL CHEMICAL INFORMATION AND DATA SYSTEM

Status Report

by

David Lefkovitz

Ruth V. Powers

Helen Hill

June 1968

307

Distribution Statement

This document is subject to special export controls and each transmittal to a foreign government or foreign national may be made only by prior approval of the Commanding Officer, Edgewood Arsenal, ATTN: SMUEA-TSTI-T, Edgewood Arsenal, Maryland 21010

Disclaimer

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Disposition

Destroy this report when no longer needed. Do not return it to the originator.

BLANK PAGE

CIDS No. 5

COMPUTER PROGRAMMING FOR AN EXPERIMENTAl
CHEMICAL INFORMATION AND DATA SYSTEM

Status Report

by

David Lefkovitz

Ruth V. Powers

Helen N. Hill

June 1968

Distribution Statement

DEPARTMENT OF THE ARMY
EDGEWOOD ARSENAL
Technical Support Directorate
Technical Data Coordination Office
Edgewood Arsenal, Maryland 21010

Contract DA-18-035-AMC-288 (A)

Task 2P062101A72702

UNIVERSITY OF PENNSYLVANIA

Philadelphia, Pennsylvania 19104

FOREWORD

The work described in this report was authorized under Task 2P062101A72702, Army Chemical Information and Data Systems (U). The work was started in July 1964 and is continuing. The information contained in this report represents part of the work accomplished during the calendar years 1966 and 1967.

The information in this document has not been cleared for release to the general public.

Acknowledgment

This document presents descriptions of a series of computer programs that collectively represent the present state of development of a comprehensive chemical information storage and retrieval system. The development represents the combined contributions of individual analysts, programmers and administrators both at the University of Pennsylvania and at Edgewood Arsenal.

The authors of this report, therefore, wish to acknowledge the able and effective project administration and technical guidance of the late Mr. James P. Mitchell, Edgewood Arsenal, and of Dr. Clarence T. Van Meter, the principal investigator of Project CIDS at the University of Pennsylvania.

The authors also wish to acknowledge the assistance of Dr. Eric N. Goldschmidt in the various chemical phase of the work and the contributions of the following analysts and programmers: Peter J. Brown, James W. Gerber, Richard J. Haber, Jeffrey Kulick, John D. Leggett, Morris Plotkin, Bonnie Sherr, and Paul R. Weinberg of the University of Pennsylvania, and of Messrs. Robert E. Amos, Robert M. Black, Robert T. Brown, Robert J. Hartman, Joel K. Kaplan, Eugene C. Logue and Capt. Walter M. Bryant of the Edgewood Arsenal.

The authors are also grateful for the generous assistance of Mary Jane Potter in performing all of the art work and of Col. Frank M. Steadman in various editorial capacities.

Reproduction

Reproduction of this document in whole or in part is prohibited except with permission of CO, Edgewood Arsenal, ATTN: SMUEA-TSTD, Edgewood Arsenal, Maryland 21010; however, Defense Documentation Center is authorized to reproduce the document for US Government purposes.

DIGEST

In April 1965, a list-structured, real time chemical information system was demonstrated on a miniature file of 1800 compounds. It was known at that time that (1) a considerably more comprehensive set of chemical screens (or keys) would be required to effectively partition a large scale chemical file for a list-structured information retrieval system, and (2) some modification in the list-structuring techniques would be required in order to efficiently process such large files. CIDS No. 3 Comprehensive Summary Report, A Proposed Chemical Information and Data System, December 1965, reported on the software implementation of this demonstration system.

The above stated requirements have been further developed and both a comprehensive chemical screen system and a more efficient real time and batched automated processor for large scale files can now be reported. The former is described in CIDS No. 4 An Experimental Chemical Information and Data System, Status Report, and the latter is documented in this CIDS No. 5 Report.

The basic difference in file organization concept between the system described in CIDS No. 3 and that described in this report is that the former was a variant of threaded list structures called the Cellular Multilist, whereas the present system uses inverted lists. The principal reason for making this change is that the exceedingly long list lengths* that are produced by the screening system of CIDS No. 4, when used to process queries that contain Boolean combinations (conjunctions, disjunctions and negations), must in the foreseeable future be stored as inverted lists on mass random access memories.

The system has been subjected to user oriented tests comprising 273 questions on a file of 290,000 compounds. The list lengths for this file ranged from 1 to 343,473. Results of these tests, including statistics on the cost of assigning the screens, generating the list-structured search files and searching the files with the batched processing system, are presented in the document entitled Report to the AMC User Advisory Group on the Initial Test of an Experimental CIDS, 2 October 1967.

In addition to a considerably more comprehensive chemical screen assignment and more efficient list search implementation, this report contains the complete documentation of the CHEMTYPE system, which converts structural formulas that have been typed on a chemical typewriter to connection tables, and of the CIDS isomer sort registry system. Thus, the system described in this report processes chemical structures plus auxiliary data in the following major steps: (1) Edited hard copy of the structures and data are typed on a chemical typewriter; (2) the CHEMTYPE system generates connection tables and formats the non-structural data; (3) the structure records are registered via the isomer sort registry system; (2a, 3a) alternatively, the system can accept connection tables from the CAS registry system; (4) structural screens are automatically assigned; (5) the list-structured search file is generated for either the real time or batched system; (6) the files are searched by the real time, on-line

---

* A list is created for every screen, which list has all compounds that contain or are described by the given screen.

system or by the batched system. The output of the search system is the structural formula plus all of the associated data. At present the formula is printed on a Dura Mach chemical typewriter or a Data Products line printer. Shortly, display on a cathode ray tube will be implemented.

TABLE OF CONTENTS

TABLE OF CONTENTS continued

TABLE OF CONTENTS continued

APPENDIXES

LIST OF TABLES

LIST OF FIGURES

COMPUTER PROGRAMMING FOR AN

EXPERIMENTAL CHEMICAL INFORMATION AND DATA SYSTEM

1. INTRODUCTION

This report describes all programs written to date for the experimental
U. S. Army Chemical Information and Data System. They are interim programs
in the sense that they will be augmented and refined, in accord with present
plans and future experience, to meet more fully the requirements of an
operational system. Disclosure at this interim stage is designed primarily
to acquaint computer systems analysts with (a) the basic design principles
of the system, and (b) sufficient programming and logic detail to indicate
how these principles have been implemented on the IBM 7040 computer.

The descriptions are presented at three levels. The Introduction pre-
sents an overview of the entire system, including the relationships among
the major subsystems, the generation and flow of data within and through
the system, and the structure and content of the principal hardcore data
record. The subsequent two sections of the report are organized according
to the two major systems, which are the file generation and the search
systems. The latter is composed of a batched search system and a real time
search system. Within these respective sections, the total system is
described and then the individual programs are functionally and operationally
described.

The functional program descriptions relate only to how a given program
functions within its own executive environment and not its relation to the
system as a whole. That is, it is a detailed description of the specific
task that a given program is to perform and is intended for an analyst who
might wish to see the design outline of each individual program below the
level of a total system functional description. Therefore, such a descrip-
tion will include a brief statement of program function called the <u>Abstract</u>
and a somewhat more detailed program description which includes, where
appropriate, a block diagram, buffers, lists, record format, input and out-
put arguments and related sub or main programs.

The operational descriptions similarly relate to individual programs
and include, where required, program operating instructions such as tape
requirements, interpretation of error messages and restart instructions.
No microflow charts or listings are provided in this documentation. All
programs are written in the MAP language for the IBM 7040.

As an aid to the understanding of the relationship between the pro-
grams described in this report, a system flow chart of the complete computer
system is presented in Appendix A. The code names of the programs required
to perform each stage of processing are included. In Appendix B, abstracts
for each program described in this report are presented in alphabetical
order by code name. Important data formats appear in Appendix D as well
as in appropriate places in the text. Unless otherwise specified, all data
elements are stored in binary.

This report does not contain a system functional description in the
instructional sense for potential users of the system. Four existing reports

13

and CIDS No. 6 will collectively provide a user-oriented system functional description. The four existing reports are those identified by the numbers (1), (4), (5), and (6) in the literature citations on page 233.

The CIDS No. 4 report (1) describes the system of structural keys that are automatically assigned to the compounds and which serves as the algorithmic basis of the screen assignment programs described in Section 2.4. The Guide to the CIDS Retrieval Language (4) specifies the mode of querying the system both in the batched and real time systems. The ACT II and III Chemical Typing Conventions (5) and (6) specify the rules for editing and drawing structures for system input and the rules for typing them along with other related data such as the molecular formula and nomenclature.

Another document, the Report of the AMC User Advisory Group on the Initial Test of an Experimental CIDS (2), is of correlative interest. It presents the detailed results of a large scale experiment in which 180 structural questions were submitted to this system at a time when the file size was 290,000 compounds.

Consonant with the purpose of producing an experimental system, the programs described in this report continue to be modified and improved. Those most subject to change are the screen assignment programs since they relate directly to the experiments, and since the over-all performance of the system is most sensitive to the quality and balance of these screens. Other parts of the system, such as the list structuring programs have performed well and are more stable, although it is planned to increase their efficiency somewhat in order to better accommodate massive files. A few additional programs, which were initially recognized as necessary for a maximally effective system but whose development has been intentionally deferred until results of large scale experimentation were available, will be incorporated.

Fig. 1 presents the three systems that comprise the U. S. Army CIDS and defines their interrelationship. These systems are labelled: (A) File Construction (B) Batched Search, and (C) Real Time Search.

In System A the structural formula of a compound must be represented as a connection table before it can be screened and added to the file. The CIDS file construction programs accept this data from two sources, the Chemical Abstracts Service registry system and the University of Pennsylvania CHEMTYPE system.* The connection tables from either of these sources are formatted into a record along with other data, and the structural screens** are automatically assigned. Then, based upon the assigned keys (screens), an inverted list is generated in which all compound record addresses having a given key are listed in sequence. The outputs of this file generation program are the <u>inverted key index</u> and the <u>search file</u>. The program is capable of producing such a list-structured file for the batch processing system (B) and the real time system (C). In the batch system,

---

*The CHEMTYPE system was developed at the University of Pennsylvania under contract NSF C-467. Input, output and chemical verification programs required to process CIDS compounds were produced under the University of Pennsylvania Project CIDS, Contract DA-18-035-AMC-288 (A).

**See CIDS No. 4, Section 3, for a description of these screens.
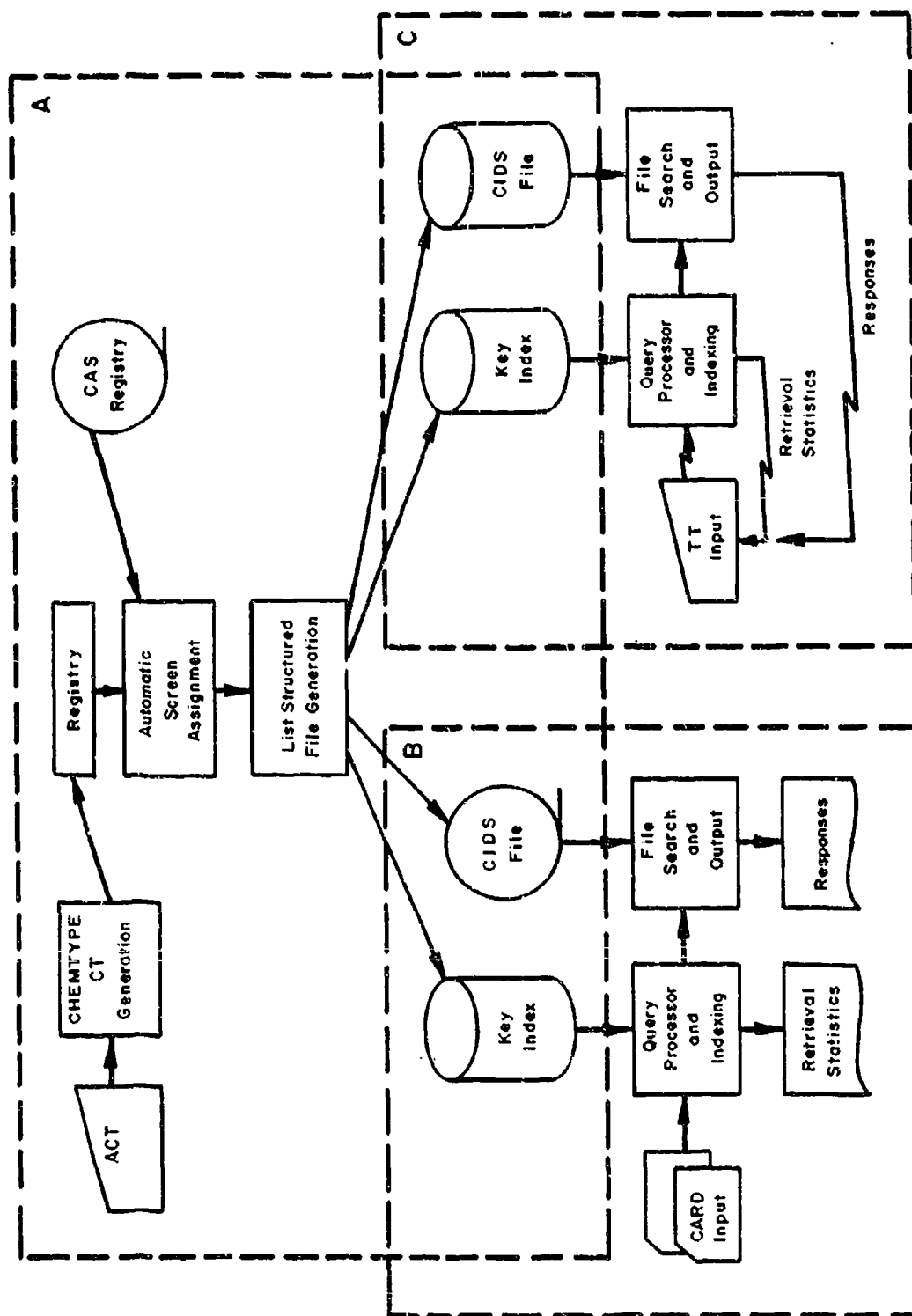
Figure I The U.S. Army Chemical Information and Data System

the index is stored on magneti disk for more efficient processing of the
inverted lists and the file is stored serially on magnetic tape; in the
real time system, both the index and the file are stored on magnetic
disk.

The batched pro essing system, represented in Fig. 1 as (dashed) block
B, accepts queries from punched cards in batches of up to 2000 queries and
processes them in two stages. The query formats are described in Reference
4. The first processing stage performs list intersections, merges, or
deletions as required by the logical expression of keys in the query. The
number of compound record addresses that respond to this stage are printed
on the line printer as a retrieval statistic.

The second stage of the process utilizes the address records produced
by the first stage and performs all necessary subsequent processing on the
accessed records. This may include molecular formula qualification and
structural atom-by-atom search The responses from stage two are sorted
by query ID number and printed on the high speed line printer.

In the real time system, shown in (dashed) block C, both the index
and file are on random access disk. The organization of this system is
similar in its two stage operation to the batched system, differing in the
following respects: (1) The inputs and outputs are via teletype and
teletype/Dura Mach chemical typewriter, respectively. These are connected
on-line to the system via data sets. (2) The queries are processed one
at a time as soon as they are received. (3) The retrieval statistics
are returned immediately to the on-line typewriter as soon as they are
computed in stage one. (4) The compound records are retrieved randomly
from the disk and returned immediately to the teletype, where they are
punched on paper tape and can be printed, with structural formula, on the
Dura Mach Chemical typewriter.

1.1 THE CIDS RECORD AND DATA STRUCTURES

The data fields of the CIDS records are listed below:

Registry Number

Additional Compound Identification Numbers

Molecular Formula

Connection Table and Abnormality Table

* Structural Formula Image

Structural Keys

Reference Block:

*   Nomenclature

*   Descriptors

*   Security Indicator

*   Stereo Indicator

The fields marked by an asterisk appear in the records processed by the CHEM-TYPE system, but not in the CAS record. Separate tape files of the CAS system contain nomenclature and bibliographic references.

Three basic kinds of data or information structures are represented in this record. These are: (1) standard alphanumeric, (2) graphs, (3) pictorial displays. The graphs are represented by a connection table which cites the nodes and branches of the graph along with their values, such as C,O,S, etc. and single, double, triple bonds. Abnormalities related to specific nodes of the graph, such as charge, mass, valence, are cited in a correlated table called the abnormality table. The format of these tables is described in Sections 2.1.2.2 and 3.1.6.2.

The pictorial display data contains the compound structural formula and is represented in the record as a table which contains every typed symbol (from the chemical typewriter) of the structure along with a number which gives its relative location within a display matrix. In the future, it is intended to replace this memory consuming representation with a more concise one which stores only node coordinates, whereby all bond symbols can be algorithmically reconstructed via Cartesian geometry plus a few heuristics. This part of the record is called the structural formula image.

The remainder of the record is standard alphanumeric data, although the nomenclature requires an extended symbol set because it contains Greek letters, upper and lower case letters, plus other special characters.

The representation of molecular formula, structural formula and nomenclature, therefore, requires a considerably expanded printer and display font capability, and both the input devices (Dura Mach and Mergenthaler chemical typewriters) and the output devices (Line printer, Dura Mach and CRT) are designed to meet these specifications, although none of them has compatible code sets. The character sets and binary code assignments of the Data Products line printer and the Dura Mach and Mergenthaler chemical typewriters are presented in Appendix F.

2. FILE CONSTRUCTION

This section describes in more detail the preparation of data for use by the CIDS Retrieval System. Dashed block A in Figure 1 gives a simplified view of the processing required for data from each of the two accepted sources, the CAS Registry System and the University of Pennsylvania CHEMTYPE system.

Each of the following subsections describes one of the major phases of processing in the construction of the Search File. These are (1) CAS Conversion, (2) Chemical Typewriter Input, (3) Registration, (4) Key Assignment, and (5) List-Structured File Construction.

Figure 2 gives an overall view of the processing of data received from the CAS Registry System. The major processing phases required are those numbered (1), (4) and (5) above. The process blocks in the flow chart are numbered in this same way, and contain the code names of the programs required to perform the processing.

Figure 3 gives an overall view of the processing required for data entered through a chemical typewriter. Process blocks numbered (2), (3), (4) and (5) refer to the major processing phases listed above. The blocks include the code names of the programs required in each phase.

2.1 C.A.S. CONVERSION

This section describes the programs required to prepare data received from the CAS Registry System for use by CIDS. The first two programs, CASFMT and CONVRT, translate data from the CAS Structure Master File to the CIDS format. The next two programs described, ADDMF and MOLEF, perform a conversion of molecular formula data found in the CAS Bibliography File and adds this to the corresponding compound record. The output of this phase of processing is a compound tape in the CIDS record format suitable for input to the Key Assignment System.

18
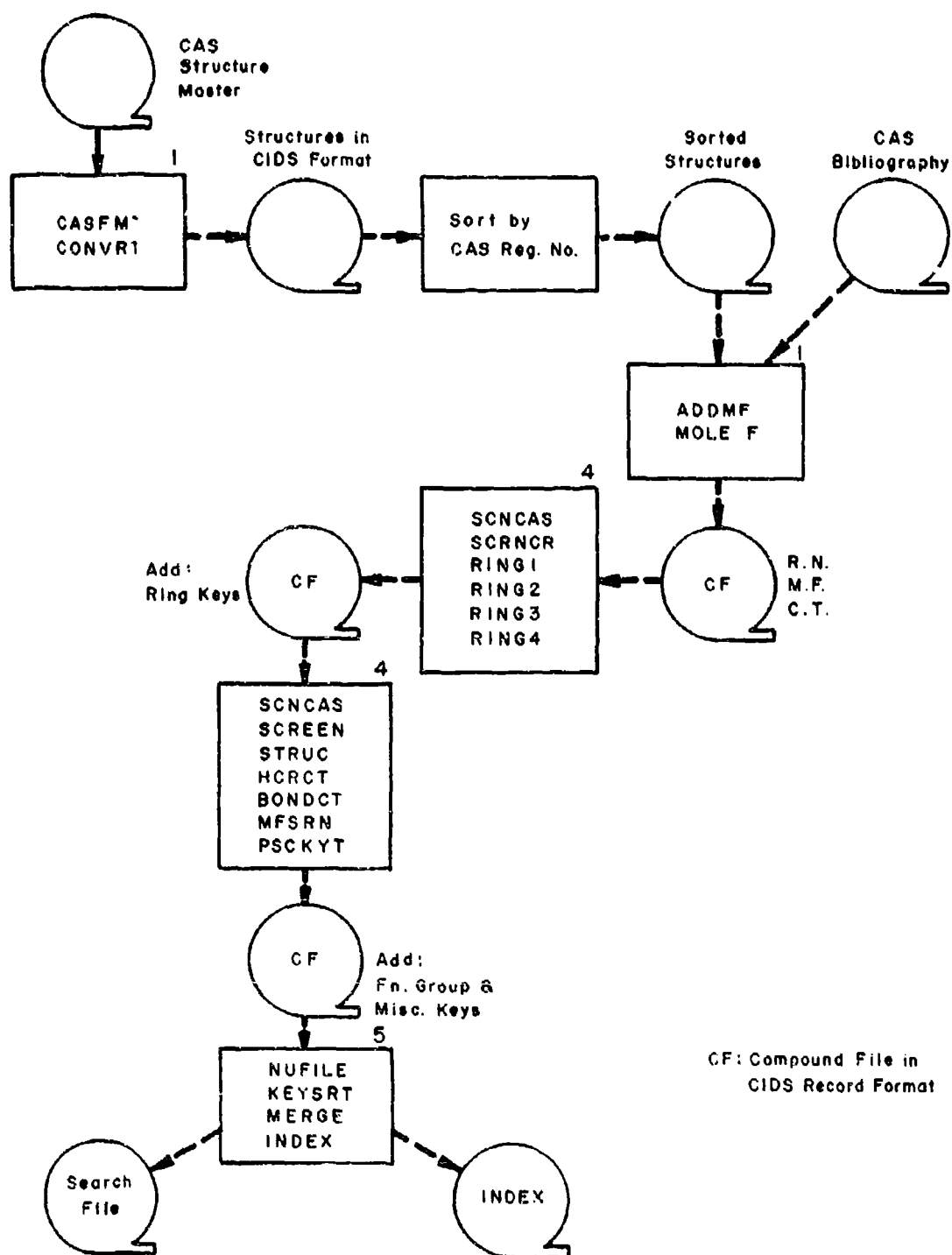
Figure 2. File Construction From CAS Data

19
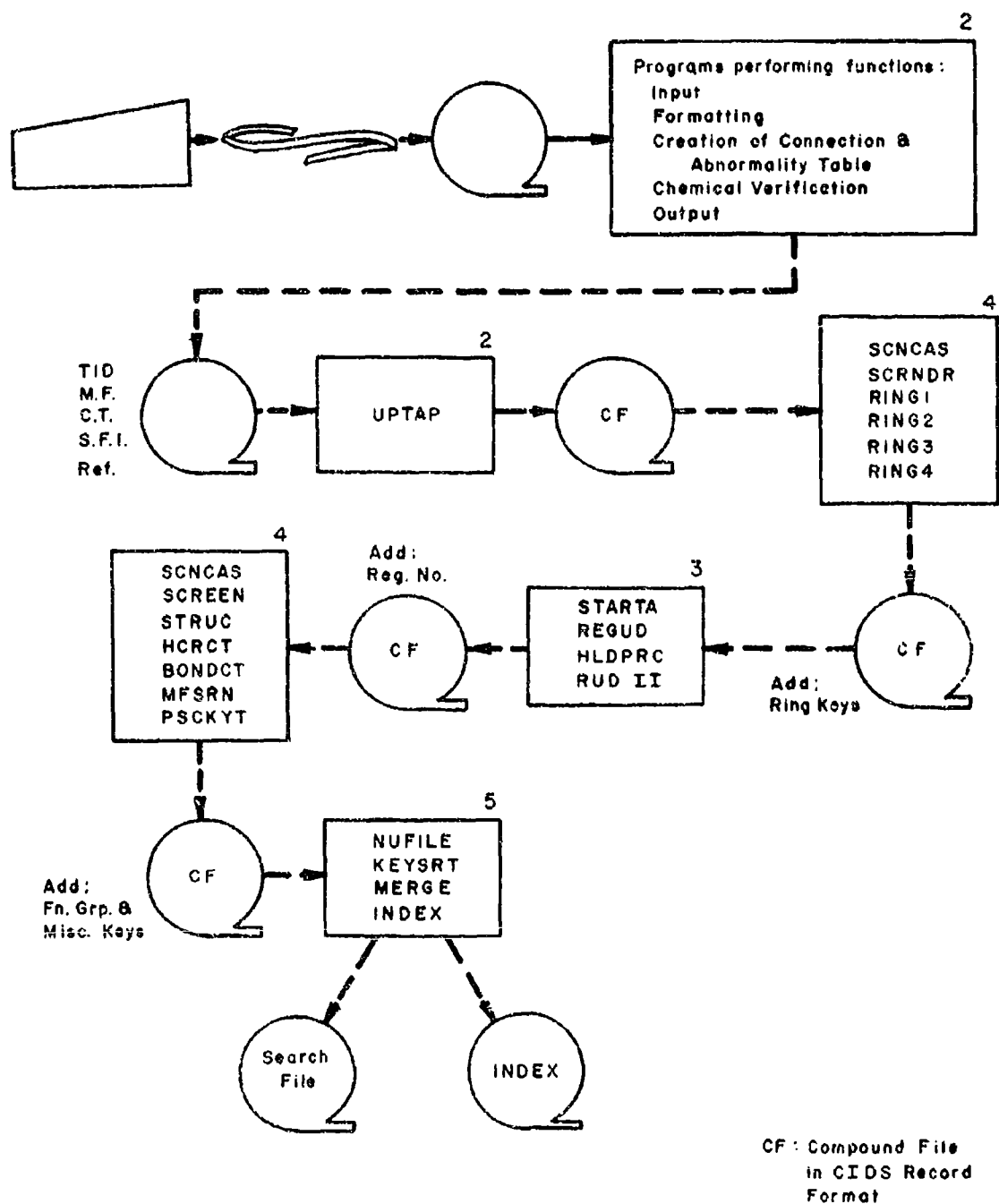
Figure 3. File Construction From Chemical Typewriter Input

2.1.1 CAS Structure Conversion

Code Name: CASFMT

Programmer: James Gerber

Abstract: CASFMT reads the CAS Structure Master File and translates the information to the CIDS format. The output of CASFMT is a tape containing the registry number, connection table, and abnormality table (if present) for each CAS compound converted.

2.1.1.1 Program Description

CASMFT accepts CAS Structure Master tapes as input. The format of these tapes can be found in Appendix C. These tapes contain a series of compound descriptions made up of one to four records of types F1, F2, F3 and F4. The program first reads a type F1 record and makes a table of "From-Attachments." Then a type F2 record is read to obtain a list of element symbols. An F3 record is read to obtain a table of bond types. The registry number and subsidiary information are read from an F4 record and converted for use as textual descriptions. After an F4 record has been read, a CIDS connection table is produced by program CONVRT.

The CAS File is ordered to take advantage of the fact that many compounds have identical first records, first two records or even first three records. These identical records are not repeated, so that after reading and converting a type F4 record (which must be the last record of every description), the next record read (beginning a new description) need not necessarily be of type F1. The new description may begin with a type F2, F3, or F4 record, indicating that the beginning records which have been omitted are identical to the corresponding type records for the previous compound.

After a complete compound description has been read, CASFMT produces X, B and E tables as input to program CONVRT (Section 2.1.2). It reads and reformats the modification list (abnormality cable), putting the charge, mass, and valence information into a CIDS abnormality table and the single-atom-addend information into the CIDS connection table. Program CONVRT then converts the connection table into CIDS format and renumbers the abnormality table to agree with the connection table which has been renumbered by CONVRT.

CASFMT will produce CIDS output tapes which are used as input to programs ADDMF and MOLEF which add molecular formula information and reformat the records.

A macro flow chart of the program is presented in Figure 4.

2.1.1.2 Program Structure

CASFMT is a main program which calls subroutine CONVRT. The input consists of the tapes holding the CAS Structure Master File. The output consists of a tape of compound descriptions in the following format:
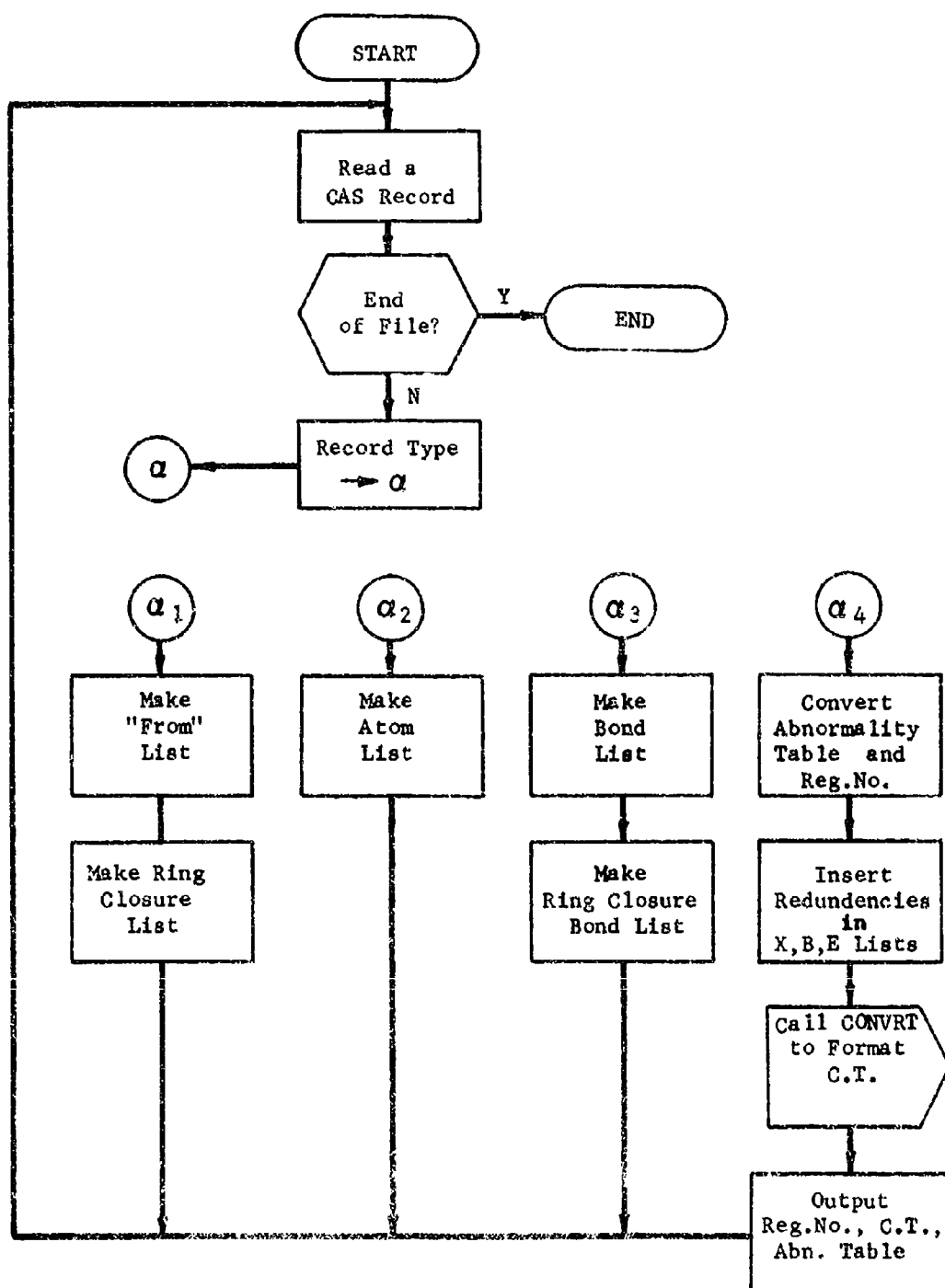
Figure 4. Macro Flow Chart - CASFMT

Word	Contents
0	Bits 0-17: Number of rings in structure (Binary) Bits 18-35: Number of words in connection table (Binary)
1-2	CAS registry number (9 characters, right-justified)
3-m	Connection table (CIDS format) (See Sec. 2.1.2.2)
m+1-n	Abnormality table (if any-last word is 0) (See Section 2.2.11.2)

Some compounds are allowed in the CAS system that cannot be converted to CIDS format. These compounds are rejected and the message

<center>REGISTRY NO. .... DELETED</center>

is printed on the line printer. No action is required. If Sense Switch 6 is in, the connection table will be printed before the deletion message.

2.1.1.3 Operating Instructions

When running this program, tapes must be mounted as follows:

```
CAS Structure Master          -- S.SU05
Checkpoint tape (no ring)      -- S.SU25
Output tapes as specified on $FILE card
```

At the end of an input reel, the computer will halt after typing a message. If the reel is not the last, press SS5 in and then start. This will cause a checkpoint to be taken as a safety measure. Then restart with the new reel mounted using the checkpoint code typed out. If the reel is the last, leave SS5 out and press start. This will cause the output files to be closed (all remaining output is written and a file mark is written) and the program will exit. Output reel switching is automatic. When starting from a checkpoint, the same output reels that were mounted when the checkpoint was taken should be mounted.

The following sense switch settings alter the program:

SS6: in- connection tables printed on line printer
      out- no connection table printing

SS5: in- take checkpoint and terminate job
      out- no action

When a check point is taken, all reel repositioning information is retained. Restart can be achieved either by a $RESTART card or by an operator interrupt with the restart code entered into the console keys.

2.1.2 Structure Conversion and Compression

Code Name: CONVRT

Programmer: John D. Leggett

Abstract: The purpose of the program is to convert a structure to a format suitable for storage and searching. The structure is compressed to facilitate the atom-by-atom search. To accomplish this compression, carbon atoms with exactly two direct attachments are removed, and the path lengths and bonding are indicated. The program will also format structures which are query fragments, in which case the resulting connection table has redundancy removed and the atoms are ordered to speed searching. In addition, the various types of free, or hanging, bonds are formatted.

2.1.2.1 Program Description

The first process is the addition of artificial atoms at the ends of any hanging bonds in a query fragment. For a hanging bond, a carbon with "don't care" number of connections is added. For dashed bonds (which are entered as type 5 bonds and indicate attachment to C or H), the bond is deleted and a special indicator is placed on the atom.

The next step is to compress by removal of carbon atoms with exactly two attachments. The list of connections is examined to locate these atoms which are then removed. Before removal of any atoms, each connection is considered to be of path length 1. As an atom is removed, the atoms to which it is connected are then indicated as being attached to each other by a path length which is the sum of the two path lengths incident on the removed atom. The bonds corresponding to these paths are concatenated and placed with the atoms to which the removed atom is attached. When all the carbons with two connections have been removed, the program renumbers the atoms to form a compact set of atom numbers, and formats the connection table.

If the structure is a query fragment, CONVRT then orders the atoms in the connection table on the basis of element kind and atom connection complexity, such that the most unusual or significant atom appears first. This technique speeds atom-by-atom search, as the query fails an irrelevant compound more quickly. Redundant entries are then removed from the connection table.

A macro flow chart of CONVRT is presented in Figure 5.

2.1.2.2 Program Structure

Program CONVRT is a subroutine which is utilized in many phases of the CIDS system. The input consists of the connection table in the format described below, the abnormality table (if any), an indication of whether the structure is a query fragment, the location in core where the final connection table is to be placed, and the number of atoms in the structure.
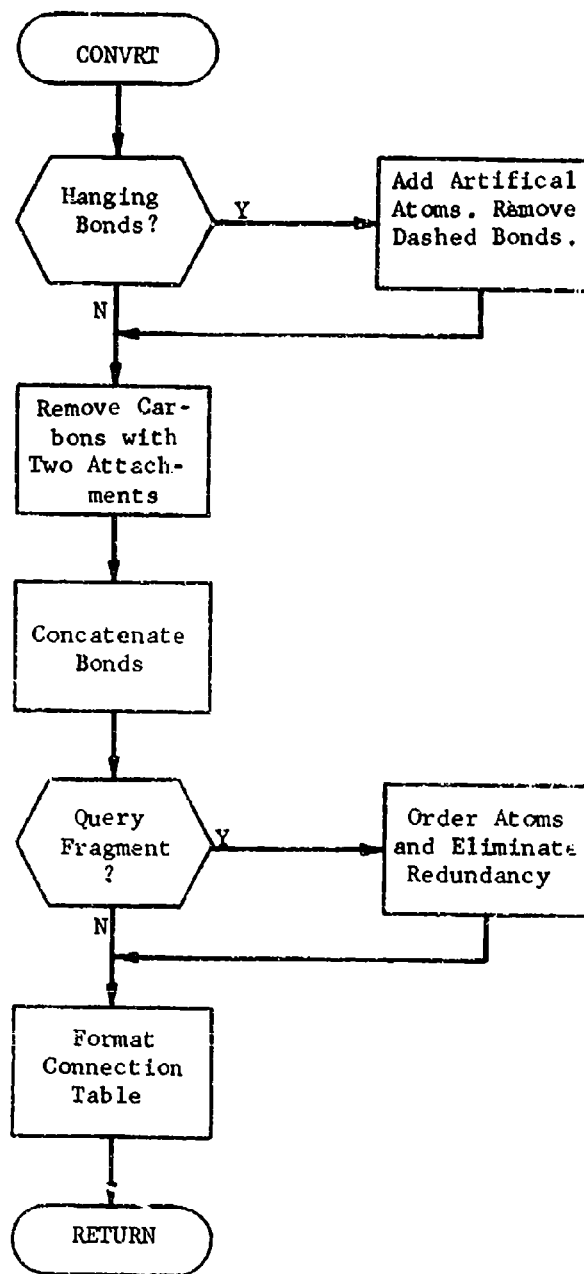
Figure 5.   Macro Flow Chart - CONVRT

25

The input connection table consists of three lists: X, B, and E, in which each atom and its connections are described in eight-word blocks. The first eight words of each array are allocated to atom 1, the next 8 words to atom 2, etc. Each eight word block in the X list contains the atom numbers for up to eight connections from that atom, right-adjusted in consecutive words. The corresponding words in the B list contain the bond type of the connection, right-adjusted. In the E list, the first word of each group of eight contains the element kind for that atom, right-adjusted in BCD. In addition, bit 17 is set to 1 for each word of the E list corresponding to an entry in the X list. If the connection is a ring connection, the corresponding E word is set minus. In the example below, the X, B, E representation is shown in octal, with leading zeros omitted.

$$H_1 -- C_3 \equiv N_2$$

X		B		E		
3		1		1006042	#1	
0		0		0		
0		0		0		
0		0		0		
0		0		0		
0		0		0		
0		0		0		
0		0		0		
3		3		1006045	#2	
0		0		0		
0		0		0		
0		0		0		
0		0		0		
0		0		0		
0		0		0		
0		0		0		
1		1		1006023	#3	
2		3		1000000		
0		0		0		
0		0		0		
0		0		0		
0		0		0		
0		0		0		
0		0		0		

26

If the connection is a ring connection, the corresponding E word is set minus.

The output consists of the connection table (C.T.) stored in a block of consecutive 7040 words. The connection table is divided into three parts: the connection segment, the bond index, and the bond segment. In addition, the first word of the C.T. is an index to the three parts. The address (bits 21-35) of the index word contains the relative location of the bond index segment; the decrement (bits 3-17) contains the relative location of the bond segment. This is illustrated below:

```
                00000x00000y         Index Word
     1                           )
     2                           |
     .                           }  Connection Segment
     .                           |
     y-1                         )
     y                           )
     .                           |
     .                           }  Bond Index
     x-1                         )
     x                           )
     .                           |
     .                           }  Bond Table
     .                           |
     .                           )
```
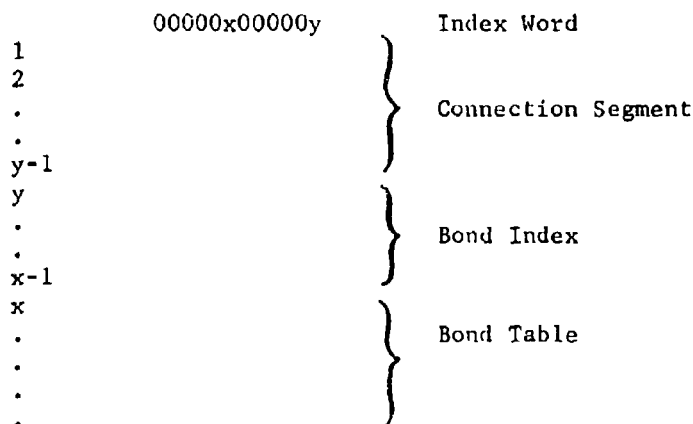
Connection Segment:--In the connection segment, carbon atoms with exactly two attachments are not explicitly stored. The presence of these atoms is indicated in the C.T. as follows:

$$^1 N - C = C - C - ^KC < ^{CH_3}_{CH_3}$$
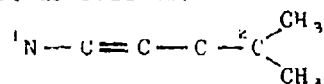
Atom number 1 is connected to atom number 2 by a path of length 4. Likewise for a file compound the redundant connection is indicated; atom 2 is connected to atom 1 by a path of length 4. Each atom present in the C.T. is stored as follows:

1st word:

Bits	Contents
s	0
1	= 1 if atom is in a ring = 0 otherwise
2-5	No. of connections to this atom
6	= 1 if 1st connection is part of a ring = 0 otherwise
7-11	Path length to 1st connection

12-17	Atom no. of 1st connection
18-29	Element kind in BCD, right-justified
30-35	Node type (see below)

2nd word: (if necessary)

s	1
1-11	0
12	= 1 if 3rd connection is part of a ring = 0 otherwise
13-17	Path length to 3rd connection
18-23	Atom no. of 3rd connection
24	= 1 if 2nd connection is part of a ring = 0 otherwise
25-29	Path length to 2nd connection
30-35	Atom no. of 2nd connection

3rd, 4th words (if necessary)

Same format as 2nd word for the remaining connections.

An atom is node type 1 if it is not carbon, node type 2 if it is a carbon with more than two attachments, and node type 4 if it is a carbon with one connection (i.e. a branch end). If a compound contains only carbons with 2 connections (ex. benzene), one atom is chosen as node type 3, and the rest are compressed.

Bond Index: The bond index serves the purpose of locating entries in the bond table corresponding to each atom in the connection segment. Each entry in the bond index requires 6 bits. The rightmost 6 bits of the first bond index word gives the location, relative to the head of the bond table, of the start of the bond entries of the second atom (the entries for the first atom are to begin with the first word of the bond table). The format is:

Word 1:

Bits	Relative Location of Bonds for
30-35	Atom 2
24-29	Atom 3
18-23	Atom 4
12-17	Atom 5
6-11	Atom 6
s-5	Atom 7

Word 2 (if necessary)

30-35	Atom 8
24-29	Atom 9
.	.
.	.

The table continues for as many words as necessary to provide an entry for each atom. The last entry gives the relative location of the word following the last word of the bond table.

Bond Table: The bond table consists of a number of groups (one group for each atom) of bond entries. The location of the beginning of each group is specified by the bond index table. Each word of a given group represents the bonds in a path from the given atom to another atom, in the form of a string of three-bit digits, each of which represents the bond type of one segment of the path. The rightmost six bits of each word contains the number of the atom to which the string is connected. For a path of length greater than 10, the bond string is continued in the next word where bits 30-35 are set zero. The compound below:

$$^1N \equiv C - C =^2 C \begin{array}{c} ^3CH_3 \\ CH_3 \end{array}$$
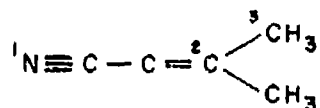
has the bond table:

000000031202	#1
000000021301	# 2
000000000103	
000000000104	
000000000102	#3
000000000102	4

A bond of type 4 indicates a non-fixed bond, i.e., a "resonant" bond.

29

As an example, the octal representation of the connection table as formatted by CONVRT for the following compound is shown below:

000016000014		Index Word
234601602302	#1	
- 1024601		
030101602302	#2	
- 1040103		
010102604601	#3	Connection Segment
020102604601	#4	
- 305		
030304604501	#5	
- 1070106		
010105604601	#6	
010105604601	#7	
151411070603		Bond Index
16		
44444401	#1	
44444401		
102		
101	#2	
203		
104		
202	#3	Bond Table
102	#4	
12105		
12104	#5	
206		
207		
205	#6	
205	#7	

30

2.1.3 Addition of Molecular Formula

Code Name: ADDMF

Programmer: Ruth V. Powers

Abstract: ADDMF reads a tape of compound connection tables which have been translated from CAS to CIDS format and are ordered by CAS Registry Numbers. Molecular formula data from the CAS Bibliography File is added to this tape and the compound records are rewritten in CIDS record format.

2.1.3.1 Program Description

ADDMF reads a tape containing compound connection table (C.T.) records which have been converted to CIDS connection table format from the CAS Structure Master File. ADDMF calls subroutine MOLEF (Section 2.1.4) to read molecular formula data from the CAS Bibliography File.

As each compound C.T. record is read, the CAS Registry Number (R.N.) is given to subroutine MOLEF which reads the bibliography tape until the molecular formula record for that compound has been found or until a larger R.N. is read, indicating that the desired record is not on the tape. When the desired molecular formula record is found a Hill and addend formula is formed in the CIDS format by MOLEF. Compounds for which no M.F. can be found are rejected with error messages.

ADDMF stores the R.N., C.T., and newly formatted M.F. in the CIDS compound record format. Also stored at this time is a count of the total number of rings in the compound which was obtained from the output of CASFMT. This is stored as the first key in the Key block.

A macro flow chart of the program is presented in Figure 6.

2.1.3.2 Program Structure

ADDMF is a main program which calls subroutine MOLEF. It prepares data for input to the Screen Assignment System.

The input to ADDMF consists of two tape files. One is the output of CASFMT (Section 2.1.1) which contains C.T. records of the following format:

Word	Contents
1	D = No. Rings A = No. Words in C.T.
2,3	R.N.
4	C.T.
	.
	.
	.
	.

31
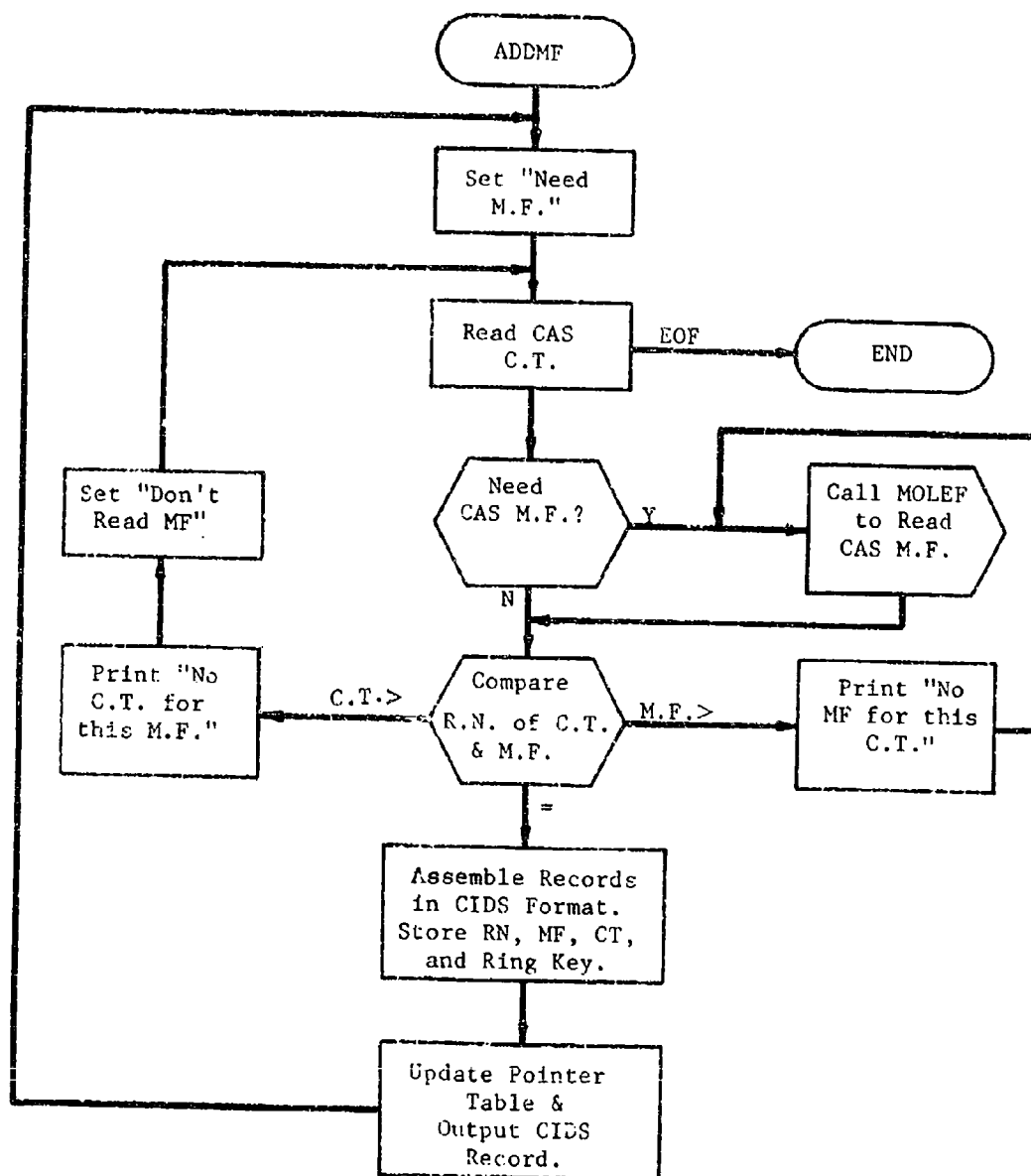
Figure 6. Macro Flow Chart - ADDMF

32

Word	Contents

| m | Abnormalities (if any) |
| | (If present, the last word is zero) |

In addition, the CAS Bibliography file is an input which is read by sub-
routine MOLEF.

The output of ADDMF is an IOBS type 2 tape. Each compound record is a
logical record. These are grouped into physical records of 1000 words or less.
The CIDS record format follows:

Word	Bits	Contents
1	(3 17)	2's C (# words preceding Addit. Reg. No.)
	(21-35)	2's C (# words in logical record)
2	(3-17)	2's C (# words preceding Abnormality Table)
	(21-35)	2's C (# words preceding C.T.)
3	(3-17)	2's C (# words preceding References)
	(21-35)	2's C (# words preceding S.F.I.)
4	(3-17)	2's C (# words preceding Keys)
	(21-35)	2's C (# words preceding Qualifiers)
5,6		Primary Registry Number (BCD)
7		Mol Form
.		
m		Additional Registry Number
.		
n		Structure (C.T.)
.		
o		Abnormality Table (if any)
.		
p		Structural Formula Image (if any)
.		
q		Reference (if any)
		Qualifiers (if any)
.		
s		Keys (2 words per key)

Note that several of the data blocks will be empty (or have zero word length).
The pointers to these blocks will point to the location where the data would be
stored if present.

If the M.F. record for a particular R.N. cannot be found, the following error message is typed: "NO MF FOR" where the CAS R.N. is given.

2.1.3.3   Operator Instructions

The C.T. input tapes (output of CASFMT) are loaded sequentially on S.SU06. When the end of each C.T. input tape is read, a message is printed requesting that sense switch 3 be pressed in if this is the last input tape. The loading of the CAS Bibliography tapes and other sense switch settings are described in Section 2.1.4.3.

2.1.4 Molecular Formula Extraction Program

Code Name: MOLEF

Programmer: Paul R. Weinberg

Abstract: Subroutine MOLEF consists of a package of programs that locate and extract the file record corresponding to a given registry number from the CAS Bibliography tapes. Summation and addend molecular formulas are computed and returned in a format appropriate for the CIDS file.

2.1.4.1 Program Description

Subroutines within MOLEF allow addressing characters within the CAS tapes and positioning on a character basis. A facility is also included to collect characters. These subroutines are used to find and extract the CAS record for a given registry number. Routine HILLFM is then used to compute the summation formula and addend formula. The subroutines within MOLEF are:

(1) HILLFM

Purpose: Form the Hill and addend molecular formulas from a pre-positioned CAS tape.

Input: Index position in POINT of start of formula in buffer.

Output: Molecular formula in CIDS format in OUFMLA block. The accumulator contains zero at exit if formula is unacceptable.

(2) COLEC2, COLEC3, COLEC4

Purpose: Get 2, 3, or 4 characters respectively from the current CAS record. Characters are returned right-justified in the accumulator. (Zeros fill unused positions.)

(3) CONVRT

Purpose: Convert a character number to index register codes.

Input: Character number in CHAR

Output: Index register codes (to access the character) stored in INDEX. These codes point to the word number and the character number within the word of the designated character.

(4) FORWRD

Purpose: Positions forward in the tape buffer a given number of characters. Repositions physical tape if necessary.

Input: Number of characters to be skipped.

Output: Length of the current block in CUR. Current register setting of the current block in POINT. (Stored as in INDEX.)

(5) REVERT

Purpose: Inverse of CONVRT
Input: From INDEX
Output: To CHAR

(6) LOCATE

Purpose: Finds a registry number on the tape.
Input: Registry number in REG and REG + 1.
Output: Positioned tape and location of first character fol-
          lowing the registry number in POINT.

Figure 7 illustrates the internal organization of the molecular formula
data for each fragment or addend. The Hill molecular formula for the compound
is formed from these tables using the rule:

(1) The coefficient for each fragment is multiplied by the smallest
     number that will make all fractional coefficients integers.
     The result becomes a new coefficient.

(2) The number of hydrogens to be subtracted from each fragment is
     calculated by multiplying the entries in SUBTR by the corres-
     ponding coefficient.

(3) The summation formula is calculated by multiplying the weights
     in WEIGHT by the appropriate coefficient and summing for each
     element over all fragments.

(4) The number of hydrogens in the summation formula is adjusted
     by subtracting the entries in SUBTR.

A macro flow chart for the program is presented in Figure 8.

2.1.4.2 Program Structure

MOLEF is a subroutine which is called with a standard MAP CALL statement.
The input to the program is a CAS Registry Number in characters stored in REG
and REG + 1. Before calling MOLEF the first time, the calling program calls
subroutine INITL to initialize MOLEF (open files, set up error recovery pro-
cedures, etc.).

The output consists of the Hill and addend molecular formula for the re-
quested compound. The format of this block is described in Section 2.2.4.

The contents of the accumulator at exit indicate the following conditions:

AC = 0 means that the registry number is not in the tape. The
        current registry number is stored in REGFD and REGFD+1.

AC = 1 means that the number has been found but the file has been
        rejected for some other reason.

AC = 2 means that there are no errors
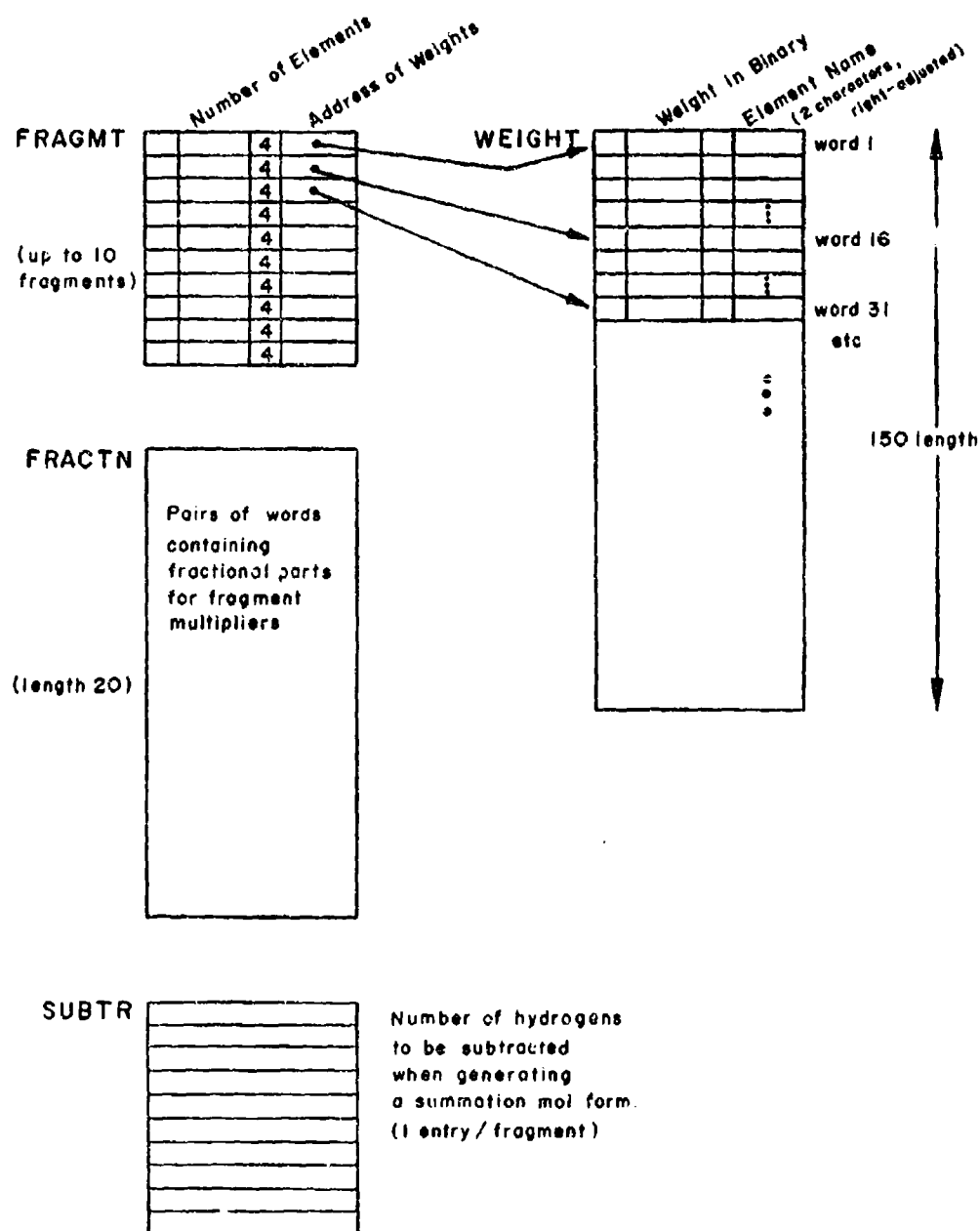
Figure 7. Internal Organization of Molecular Formula Data

Produce
Summation
Formula

MOLEF

Enter with
desired registry
number in
REG and REG+1
in BCD

Is Tape
Positioned
at a Registry
No. ?

N   Y

Is it the
Desired
Registry
No. ?

N   Y

FEND

Call LOCATE
to Reposition
to the Desired
Reg. No.

Number Not
on Tape

Merge
Fragment
Tables

AC = 0
?

Y

RETURN

N

Put Output
in CIDS
Format

Find Formula
in Compound
Record.

RETURN

Call
FORWRD
to Position
to Formula

Error if no
Formula Given
or Record
Deleted

For each Fragment: Store Elements
and Weights in Fragment Tables. Also
Fractional Coefficients and No.
of Hydrogens to be Subtracted

FEND

Y

Last
Fragment
Completed ?

N

Figure 8.   Macro Flow Chart - MOLEF

38

Users of the program should note that FORM10, FORM20, FORM30 and FORM40 are the names of the decks containing MOLEF and related subroutines. The user is warned that MOLEF alters the system control blocks for utilities S.SU05 and S.SU07 and restores them at exit. If the program fails and does not exit normally, the operating system must be reloaded.

The following error messages may be printed when the described conditions occur:

(1) END OF FILE READ BY MOLEF
MOLEF has read a file mark and has returned assuming the registry number does not appear on the tape.

(2) RECORD DELETED FOR REGISTRY NUMBER XXXXXXXXX
The file has been located but the record has been deleted by CAS. MOLEF assumes the registry number does not appear on the tape.

(3) NO FORMULA GIVEN BY CAS FOR NUMBER XXXXXXXXX
MOLEF assumes the registry number does not appear on the tape.

(4) SUMMATION FORMULA FOR REGISTRY NUMBER XXXXXXXXX
Deleted due to missing fraction coefficent XXXXXX
Syntactical error on the part of CAS. Return with 1 in the AC.

(5) STORAGE ALLOCATION PROBLEM FOR REG. XXXXXXXXX DELETING SUMMATION FORMULA
Not enough buffer space has been assigned to compute the summation formula. Return with 1 in the AC.

(6) TOO MANY FRAGMENTS IN REG NUMBER XXXXXXXXX DELETING SUMMATION FORMULA
More than 10 fragments are not allowed. Return with 1 in AC.

(7) FORMULA TOO COMPLEX FOR REG. NUMBER XXXXXXXXX DELETING SUMMATION FORMULA
Not enough buffer space. Return with 1 in AC.

(8) CONTROL WORD OVERFLOW
More than 18 elements have been found. Not enough room in format.

(9) END OF BUFFER REACHED AT NUMBER XXXXXXXXX
SUBROUTINE INDEX2 SKIPPING TO NEXT TAPE RECORD
Program error. Return with 0 in AC.

(10) UNABLE TO FIND REGISTRY XXXXXXXXX
TAPE POSITION TO XXXXXXXXX
A number higher than the desired registry number has been located. MOLEF returns with the number found in REGFD and REGFD+1. AC set to 0.

39

2.1.4.3 Operating Instructions

MOLEF expects to find the first CAS Bibliography tape on unit S.SU05.
The remainder of the tapes should be mounted alternately on S.SU07 and S.SU05
to allow reel switching to take place. Input is doubly buffered and reel
switching is automatic.

If sense switch 6 is pressed in, a listing of the output blocks will be
produced.

2.2 CHEMICAL TYPEWRITER INPUT

The purpose of the Chemical Typewriter Input Programs (CHEMTYPE) is to provide a means of introducing chemical structures into registry files which can then be used directly with structural key assignment to generate search files.

The overall process consists of the following steps:

(1) Information about each compound to be entered is written by a chemist in a standard form.

(2) A typist, using either a Dura Mach or Mergenthaler Chemical Typewriter, transcribes the information simultaneously to typed copy and to punched paper tape.

(3) The paper tape image is transcribed by a computer to magnet' tape.

(4) The CHEMTYPE programs exhaustively analyze the magnet'c tape images and produce files which are suitable as input to a chemical registry system.

The entire process of entering chemical data requires that care be taken by the chemist and the typist to ensure accuracy of input. However, a primary objective of the CHEMTYPE programs was to permit the chemist and the typist the widest possible latitude both in entering data and in making corrections that are consistent with the unambiguous interpretation of the paper tape input stream. Thus, the CHEMTYPE programs recover a variety of retrievable errors and permit wide variability of input formats. They signal errors only when the input contains an error they cannot correct.

The chemist provides the original input data for the typist. It is his job to present the structures to the typist in such a way that she may type them with no knowledge of chemistry. The rules for the chemist have been previously stated in ACT II Typing Conventions and ACT III Typing Conventions, and are consistent with standard structuring conventions. These documents describe the rules to be applied to the Mergenthaler and Dura Mach typewriters respectively. The rules for the chemist are identical in both cases.

Figure 9 is an example of the copy which the chemist produces.

The typist must type so that all pertinent information is recorded on digital paper tape. The typewriter input is such that control characters are punched which later are used to determine the location of each typed character in a two dimensional matrix. The typist, therefore, has a certain amount of freedom to move randomly within the record since the physical location of characters is not determined by the strict order in which they appear on paper tape. The specific rules for the typist to follow have been previously stated in ACT II Typing Conventions[5] and ACT III Typing Conventions[6].

The typist is allowed a certain leeway in correcting errors. Procedures are given for correcting a specific error and for deleting a record which is partially typed and starting over again.

$C_4H_{10}AsCl_3N_4S_2$ $C_4H_8AsClN\ S\ \cdot 2ClH$
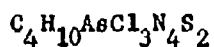
$TL90$

[Arsine, (2-chlorovinyl)bis(guanylmercapto)-] [1083]

The dihydrochloride: ~~ClCH-CHAs—SC(=NH)NH₂·HCl₂~~

Ethylenedithioarsonit, 2-chloro-, diguanyl,
    dihydrochloride
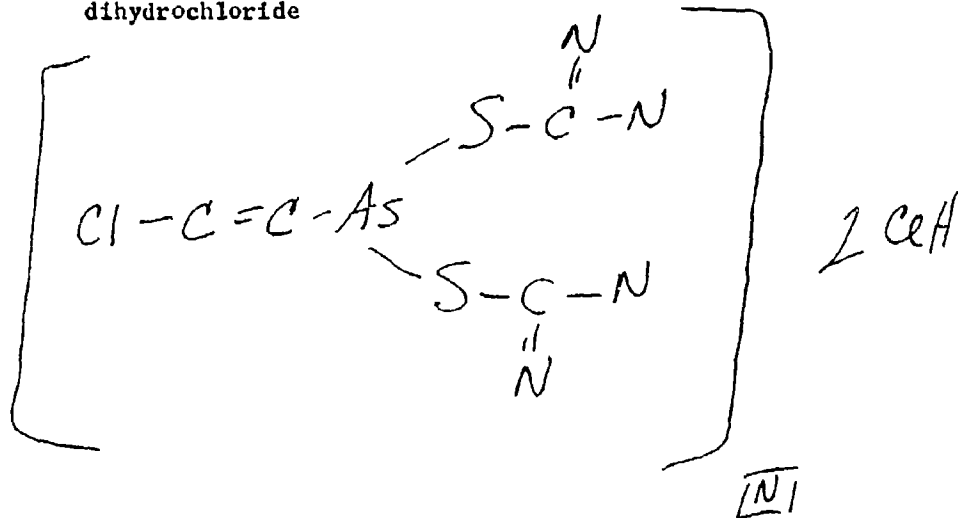
Figure 9. Copy Produced by Chemist

Figure 10 is the hard copy output produced by the typist from the input copy shown in Figure 9. It must be noted that the nomenclature in both Figure 9 and Figure 10 is inaccurate. It is given to the typist as received without editing since the nomenclature as presented may have served as an index term on many previous occasions.

The Programmer has designed the system to recognize all possible inputs from completely meaningless information to good chemical records. A great deal of error checking has been introduced into the system so that only correct chemical records are entered into the file. The only real limitation on this has been the impossibility of checking the accuracy of the nomenclature entered for each compound, or the accuracy of the local control number (TID), stereo or classification information. The only way incorrect data of this sort can be kept out of the file is to visually check all typed copy and note the TID of incorrect records. These records may then be deleted by using the TID's as input to a subroutine which prevents these compounds from being entered into the file during processing.

The accuracy of the molecular formula may be checked within limits by the hydrogen parity rule and by comparison with the typed structure. Any discrepancy between the molecular formula and the typed structure is assumed due to an error in one or the other.

The two kinds of typewriters produce almost identical copy with the exception that the Dura Mach has a more restricted character set than the Mergenthaler

42

and several of the required symbols must be simulated. There are, however, enough differences in the manner in which the two typewriters accomplish this that separate input programs had to be written specially tailored for the specific typewriter.

>
T00937   (U)   C4H10AsCl3N4S2
                                                             :C4H8AsClN4S2•ClH
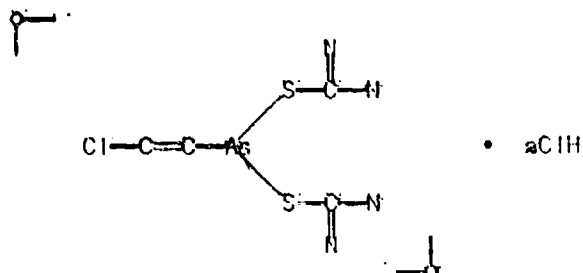
STEREO N
(Arsine, (a-chlorovinyl)bis(guanymercapto)-)
The dihydrochloride:
Ethylenedithioarsonit, 2-chloro-, diguanyl,
dihydrochloride
=TL90
=1083
**

Figure 10. Copy Produced by Chemical Typist

THE DURA MACH

Since the Dura Mach does not record any coordinate information, the analysis of structures entered through the Dura Mach must depend on line and space control punches. As a result the typist may not move the platen by hand, as this does not register on the paper tape. Similarly, tab and margin use is restricted as specified in the typing conventions.

The limited symbol set on the Dura Mach requires the synthesis of certain symbols. For example, the triple bond is indicated by a single bond overtyped by an asterisk. This is true for all the triple bonds. The Dura Mach characters do not include a parity bit and no error checking is done directly by the hardware.

THE MERGENTHALER

The Mergenthaler typewriter produces coordinates as a result of a backspace, line advance, carriage return, tab or white ribbon or moving the platen by hand. These decode into the x and y coordinate for the first character which was typed after the last coordinates were produced. As a result, the typist may move freely within a record, moving the platen by hand at will. There are certain classes of characters whose coordinates must not only be decoded, but to which a correction must

43

be applied before the character may be placed into a 2 dimensional matrix. These are characters which print above or below the line and whose coordinates are given as though they were typed directly on the line (see Section 4.1.1).

Since each Mergenthaler character includes a parity bit and the machine has fairly comprehensive error detection hardware which causes the keyboard to lock on the detection of a parity error, it results in fewer mispunches reaching the computer. When parity errors are detected by the computer it is almost certain to be either because the typist made a correction incorrectly, or there was an error due to the paper tpe reader when the paper tape was transferred to magnetic tape.

In addition to the above considerations, certain syntax requirements have been placed on each of the various fields of typed information. These are described fully under the program descriptions associated with manipulating this information.

Figure 11 describes the major functions performed by the CHEMTYPE System.

The relation of each program to the total CHEMTYPE System is pictured in Figure 12 and is described in the following paragraphs.

(1) Input of Chemical Typewriter Information

    (a) INPUTD-reads Dura Mach records from magnetic tape and translates all of the typed information for a chemical compound into Mergenthaler Code and places it in a 2-dimensional matrix.

    (b) TAPWRM- does the same for Mergenthaler records.

(2) Formatting

    (a) ORGNZR- processes a single chemical record formatting the following fields:

        (1) Temporary Identification (TID) (Local Control Number)

        (2) Security classification

        (3) Stereo information

        (4) Structural formula image (SFI). If bracketed information is present, ORGNZR calls on REGRUP to reorder the SFI so that all characters within a single set of brackets will appear compactly. If any atoms appear outside of the brackets, REGRUP in turn calls on XCESS to format 'his.

        (5) Molecular formula (by calling MOLFRM)

        (6) Nomenclature and reference fields (by calling MONIKR. which in turn calls PUNCH to punch cards for TOXINFO file).
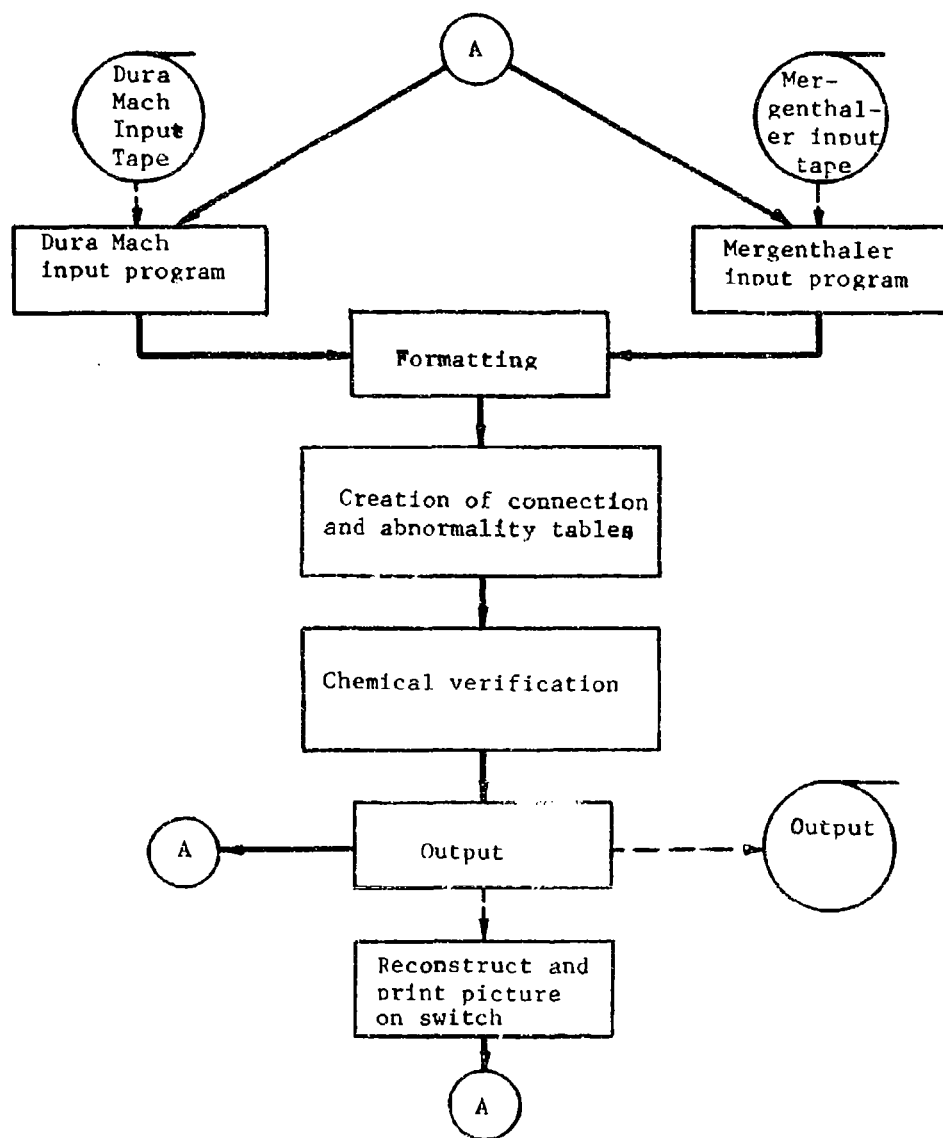
Figure 11.   CHEMTYPE System Process Chart

INPUT:
Read one chemical record from magnetic tape and construct matrix

DECK NAMES:
INPUID - treats Dura Mach Data
TAPWRM - treats Mergenthaler Data
NOTE: only one of the above is loaded into core at a time

FORMATTING:
Formats fields; constructs pointers and lists for later use

DECK NAMES:
ORGNZR - main program
MOLFRM - formats molecular formula
MONIKR - formats nomenclature and reference field
REGRUP - reorders SCRUB
EXCESS - formats elements not in Connection Table
APOLGY - writes error messages

CREATION OF CONNECTION AND ABNORMALITY TABLE
The Abnormality Table and Connection Table are written in . internal code.

DECK NAMES:
SETUP - selects and classifies linear string
CLEANM - reduces linear string to a point, writes internal Connection Table for Ph and carbon chains; writes Abnormality Table
MAKECT - formats Connection Table

CHEMICAL VERIFICATION:
Verifies validity of Structural Formula molecular formula, and Abnormality Table. Rejects incorrect records.

DECK NAMES:
PHASE 5 - calls VERIFY. Transfers to NFCF if correct otherwise rejects record
VERIFY - Verification program

OUTPUT:
Creates output record on mag. tape or punched cards.

DECK NAMES:
NFCF - transforms Connection Table to a format acceptable by CONVRT
DECKA - produces line printer listing of Connection Table on switch
CONVRT - compresses Connection Table and Abnormality Table
TICKER - writes output tapes
PIX, DURA, LINPIX - reconstruct structural formula image

END:
Transfers back to INPUT

DECK NAMES:
AEND - normal end
REJECT - error messages printed

Figure 12. Interrelation of Programs in the CHEMTYPE System

(3) Connection Table Generation and Verification

    (a) SETUP - uses SFI to find a capital letter in the matrix and calls on CLEANM to remove all characters but atoms and bonds from the matrix. It creates the Abnormality Table and expands any instances of Ph, $(CH)_n$, or $(C)_n$.

    (b) MAKECT creates the Connection Table.

    (c) PHASE5 calls on VERIFY to verify the validity of the Chemical Structure, the Abnormality Table and the Molecular Formula.

(4) Output

    (a) NFCF expands the Connection Table to a form acceptable as input to CONVRT. NFCF calls on DECKA to print the Connection Table when a switch is set. NFCF calls on CONVRT to transform the Connection Table to compacted format and on TICKER to create an output tape containing all the formatted information.

    (b) PIX, DURPIX and LINPIX may be called on to reconstruct the structural formula image for output on punched paper tape from the teletype which can then be printed on the Dura Mach, for output directly on a Chemical Line Printer, or for simulated output on a 1403 line printer. These programs are described in Section 3.3, since they are used to output responses to queries.

Error messages are printed as each program encounters error conditions in the typed record. These are discussed in greater depth in Section 5 of the CHEMTYPE System[3]. A listing of the possible errors appears in Appendix E.

The typewriter translation programs require the entire available core of about 23,000 locations and must use blocks of storage for more than a single purpose. The entire CHEMTYPE system of programs is in core at all times.

The system input is a magnetic tape containing Mergenthaler or Dura Mach code formatted as follows in fixed 300 word records. Each paper tape constitutes a file which begins with a six character tape number whose first character denotes the source of the typing: E=Edgewood Arsenal, F=Frankford Arsenal, U=University of Pennsylvania. The last paper tape on a magnetic tape is followed by a 300 word record of all 7's.

Mergenthaler or Dura Mach characters on magnetic tape are packed as follows during transcription of the paper tape onto magnetic tape.

```
 0◄─►3 4 ◄────────►11 12◄►15 16 ◄─────► 23 24◄►27 28 ◄──────► 35
┌─────┬──────────────┬───────┬──────────────┬───────┬──────────────┐
│  0  │              │   0   │              │   0   │              │
└─────┴──────────────┴───────┴──────────────┴───────┴──────────────┘
              ►Typewriter Character ◄
```
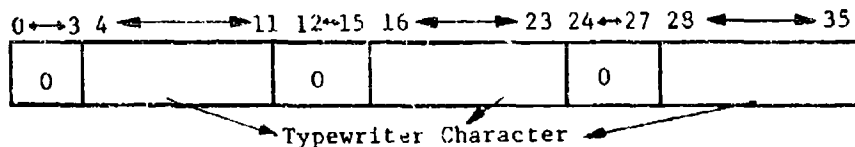
Data card input consists of a card giving the number of records on the input
tape to be skipped, and cards giving the TID's of typed records to be deleted
during processing. These must be followed by one blank card. The format of
the TID's to be deleted is as follows:
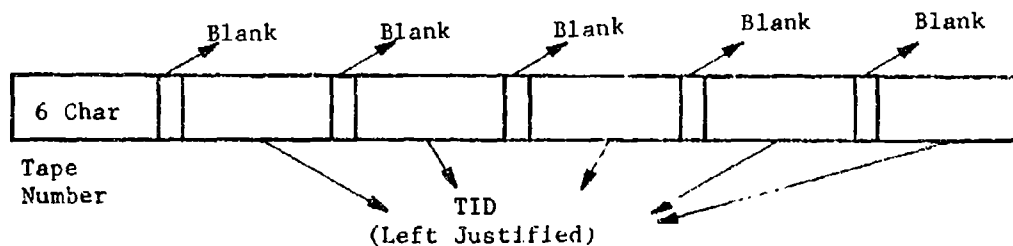
The output tape which is created by this system, consists of variable
length records, each record consisting of a single chemical record. The
format is presented in Figure 13.

	S↔2 3 ◄──────► 17	18↔20 21 ◄────► 29	30 ◄──► 35	
WORD 0	No. of words in block not including Word 0	010		F (BCD)
WORD 1	2's complement of first location of MOLTAB		2's complement of 1st word of connection table	}
WORD 2	2's complement of first location of abnormality table (0 if not present)		2's complement of first location of nomenclature	} Locations to WORD 0
WORD 3	2's complement of first location of SCRUB list		Number of RINGS	}

WORD 4	REGISTRY NUMBER (first 6 characters)
WORD 5	REGISTRY NUMBER (second 6 characters)
WORD 6	CLSTER (Classification and Stereo)
	MOLTAB BLOCK
	CONNECTION TABLE BLOCK
	ABNORMALITY TABLE BLOCK (if present)
	NOMENCLATURE & REFERENCE BLOCK

	S↔2	3↔5	◄──────► 17	◄──► 25	26 ◄──► 34	35	
SFI Header	-	+		DELY	DELX		↖ Set if underline table exists

Charges outside of brackets (total)   SFI BLOCK

Figure 13. CHEMTYPE Output Format For One Chemical Record

2.2.1   Mergenthaler Input Program

Code Name: TAPWRM

Programmer: Helen Hill

Abstract: TAPWRM reads typewriter characters in Mergenthaler Code from
a magnetic tape. It interprets these codes and constructs a 2-dimensional array containing an image of the typed chemical record.

2.2.1.1   Program Description

A macro flow chart describing this program is presented in Fig. 15.
TAPWRM reads input cards giving the number of magnetic tape records to be
skipped before processing begins and the TID's of compounds to be deleted by
the program. It then begins reading the Mergenthaler character stream from
magnetic tape.

Parity is checked on every typewriter character and the program looks for
one of the following at the beginning of a paper tape: (a) a case character,
(b) a lozenge, (c) an E, F, or U indicating paper tape number follows. Each
case character encountered is identified and the current case is stored. When
an E, F, or U is found, the tape number which follows is formatted and printed
on the line printer. (The tape number is not necessarily present). The presence of a lozenge ($\diamond$) indicates the beginning of a chemical record.

The remainder of the chemical record is then read, one character at a time,
and the characters are stored in the proper locations in the matrix until a
box, or ** are encountered. The presence of a box indicates that the typist
will begin the record anew and the record to this point should be ignored.
The ** indicates that the chemical record has been completed. When the ** is
reached, the program continues reading in characters until the next lozenge
is found, since the typist may correct the previous record at any time before
typing the next lozenge. The program makes corrections on the presence of a
white ribbon punch which indicates that any characters found before the occurrence of a black ribbon punch are to be erased. Any characters which are
found to be typed into a location that already contains a character and which
are not legitimate overtypes (special characters not included in the character
set) are considered to be corrections and replace the character already in
the matrix. Legitimate overtypes are:

///	is typed as	⅄	or	⅋
\\\	is typed as	⃰	or	⃰
∣∣∣	is typed as	⸸	or	⸸

50

These are replaced in the matrix by the correct triple bond. When bond cross-

ings, such as ─┼─ or ╲╱ are encountered, one bond must be erased resulting

in ─│─ or ╲╱ in the matrix. In the case of underlines, (which print in

same matrix location as character underlined), the character is made minus and
the underline dropped. If any bond is typed into a location containing a brack-
et corner, it does not replace the bracket corner. Since typists have been
found typing bonds in the same location as the bracket corners, the program was
altered to prevent this fact from erasing the corner itself.

The actual input into the matrix is accomplished by reading characters
and storing them in a work area until coordinate punches are encountered which
give the coordinates for the first character which now occupies the work area.
Each set of coordinates appears as a series of six paper tape characters. The
first of these signals that coordinates are coming, the next two decode into
the y coordinate, the next two into the x coordinate, and the last is blank
tape. Using the decoded coordinates, the characters are placed in the appro-
priate location in the matrix after proper analysis has been made. Characters
which are typed while the carriage is in one location but which actually print
below the line one or two spaces, characters for which the carriage does not ad-
vance, and characters which are double symbols typed in a single location are
classes of characters which are recognized and the proper corrections made.
These classes are as follows:

(1) Characters whose Y coordinate is given for the line on which
     the carriage rests, but which p int one Y coordinate below
     this. These are:

              sub case   ╱ ╲ ╱ ╲

              upper case ╲ ╱

(2) Characters which print two Y coordinates below the Y coordinates
     registered for carriage location:

              sub case   ═

              sub case   ─

              carbon dot   •

(3) "Non-escaping" characters for which the carriage does not move
     after typing, but instead, stays in the same position for the
     typing of the next character. There are three special groups
     of these characters which in addition to being "non-escaping"
     have one of the following characteristics:

         (a) The character types one y coordinate below the coordin-
              ate registered for the position of the carriage.

                   sub case   ‖ , |

                   upper case | , ‖

         (b) The character types one y coordinate higher than the
              carriage position.

                   lower case   ••

51

The virgule is one non-escaping character which because of its special use must be mentioned separately. The non-escaping characteristic requires that single digit fractions be typed in the following manner: virgule, numerator, denominator: and in the case of a two-digit numerator as follows: first digit of numerator, virgule, second digit of numerator, denominator.

(4) Special double characters for which the registered y coordinate is correct for the upper half of the character but for which the lower half prints one y coordinate before the registered coordinate:

sub case ↑ ⇑ (used in typing benzene ring, etc.)

The only analysis of fields in the matrix that TAPWRM does is to locate the word STEREO which tells the program that the end of the structure has been found and to locate the ** which indicate the end of the chemical record. Since the word STEREO may be forgotten and typed in later, if the program does not find STEREO as it goes along, a rescan for the word STEREO is done when the ** are found.

2.2.1.2 Program Structure

TAPWRM utilizes the following subroutines:

(1) INPUT- calls for the next 300 word logical record to be read from magnetic tape into a buffer and looks for the terminating record of all 7's. When this is found, it sets a bit and returns to the program to finish processing the last record before exiting.

(2) NEXT- unpacks one 7040 word, placing the next eight bit character into the accumulator after stripping off the parity bit. When the contents of a 300 word buffer have been exhausted, it calls INPUT to get the next record. NEXT checks for parity errors on each input character. If a parity error is found on input, a bit is set, a parity error counter is incremented, and the record is deleted. A count of the total parity error encountered between two correct compounds is printed out when compounds are entered into file.

(3) CALCXY- calls NEXCOR to get each next coordinate word. It unpacks the coordinates (next 5 input words) which come in as eight bit Mergenthaler characters and calculates the x and y coordinates. If the last character in coordinates is not a zero (or blank punch) CALCXY rejects the record. If a Code delete is found in the coordinates, CALCXY assumes that a parity error was detected by the typewriter and that the typist retyped the last typed block. This block is erased and the program returns to processing. A parit error in coordinate input results in an error message and rejection of the record.

(4) NEXCOR- calls NEXT to get the next input word of coordinate input, checks for the presence of either a code delete or parity error, and strips off the low order bit of the coordinate word.

52

(5) YCOORD- recalculates input y coordinate, modulo 132, correcting on the basis of the y coordinate of the lozenge as equal to 1.

(6) CASES- determines whether an input character is a case character, and if so, stores the current case.

(7) STORE- stores the character found in the accumulator in the next space in the work area.

(8) TERMIN- handles a normal end of tape situation.

(9) EOFEXI- is entered when end of paper tape image is encountered (indicated by presence of file mark). It writes the following on the line printer:

    (a) total number of records processed (count of lozenges encountered).

    (b) total number of records deleted by typist.

    (c) total number of records deleted by program.

    (d) total number of records entered into the file.

(10) ERASE1- erases a record which is rejected before the system exits from TAPWRM. All necessary locations are reinitialized.

(11) ERASE- erases records which are rejected after the system exits from TAPWRM. All necessary locations are reinitialized.

TAPWRM takes as input Mergenthaler characters read from magnetic tape and packed as follows:
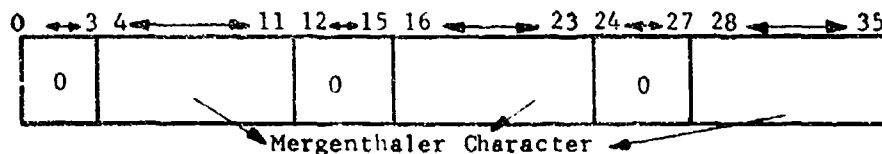
Physical records are 300 7040 words long (900 characters). If a paper tape ends before the end of a physical record, the record is filled out with zeroes. Each paper tape record is followed on magnetic tape by a file mark. The final paper tape on a magnetic tape appears as follows:

(1) paper tape characters

(2) physical record filled with zeroes after end of paper tape

(3) file mark

(4) a physical record containing all 7's

(5) a second file mark

All Mergenthaler characters appearing on the paper tape appear also on the magnetic tape. Each paper tape will have a tape number of up to six characters which will be found at the beginning of the tape image. The first character is :

(1) E for Edgewood tapes

(2) F for Frankford tapes

(3) U for University tapes

Each Mergenthaler character is eight bits and may be a typed character, a case character (upper, lower, or sub-case), a space, a control character white ribbon, black ribbon, code delete, blank tape, or part of six character coordinate punch as illustrated in Figure 14.
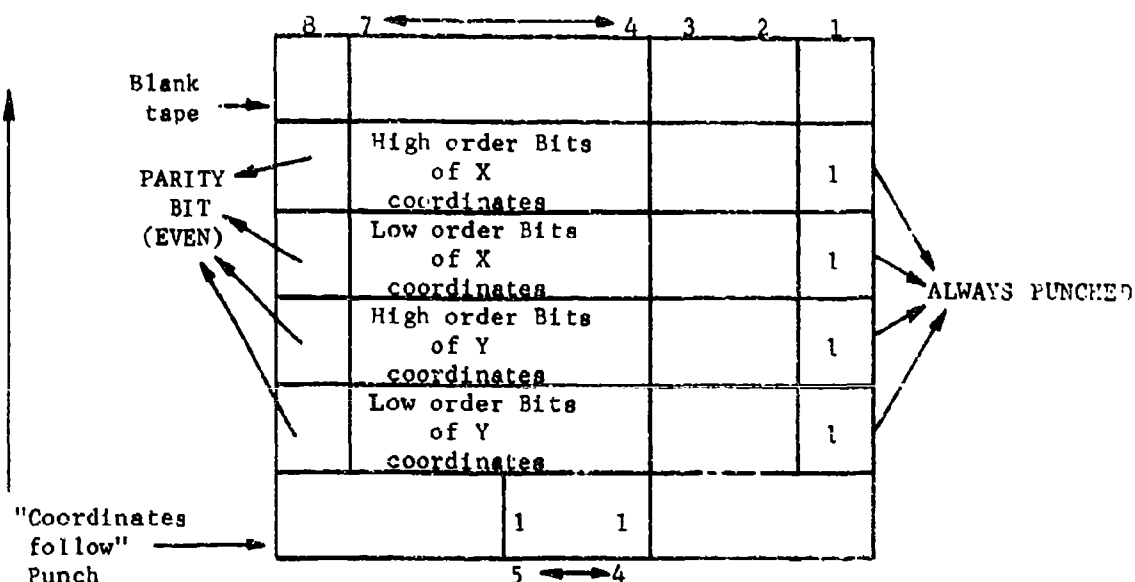
Figure 14. MERGENTHALER COORDINATE PUNCH CODE (BINARY)

TAPWRM also requires input cards which specify the following:

(1) record number on magnetic tape on which processing is to begin.

(2) registry numbers of compounds to be deleted on processing and paper tape number on which they were typed.

(3) a blank card.

Outputs from TAPWRM are as follows:

(1) MATRIX--a 10000 location matrix containing on complete chemical record in Mergenthaler code with an added case bit. Each location containing a character is formatted as follows:

```
     0 1 ◄──────────────────────────► 26 27 28 29 ◄────► 35
    ┌──┬──────────────────────────────┬─────────┬──────────────┐
    │  │                              │ Case    │ Input char-  │
    │  │                              │ bits    │ acter with-  │
    │  │                              │         │ ou  parity   │
    │  │                              │         │ bi           │
    └──┴──────────────────────────────┴─────────┴──────────────┘
```

1 for underlined
character

SUB   = 01
LOWER = 10
UPPER = 11
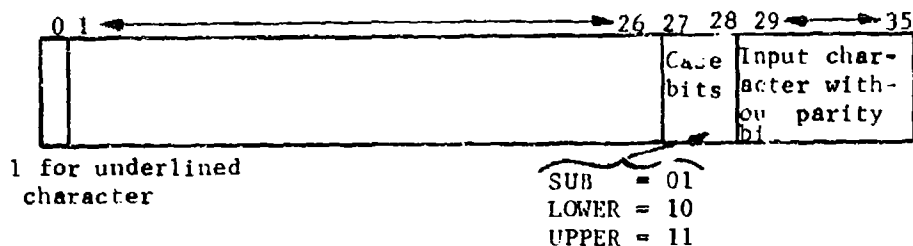
(2) TAPE Number--printed on line printer at beginning of each paper tape which was numbered.

(3) REGTAB -- a table of two word TID's in BCD of compounds to be deleted by ORGNZR.

(4) TAPTAB--table of paper tape numbers associated with the registry numbers in REGTAB. Five registry numbers are associated with each paper tape number as follows:

TAPTAB   REGTAB
(BCD)   (BCD)

TAPE 1   REGISTRY NUMBER 1

TAPE 2   "   "   2

                                      "   "   3

                                      "   "   4

                                      "   "   5

                                    REGISTRY NUMBER 6

                                        etc.
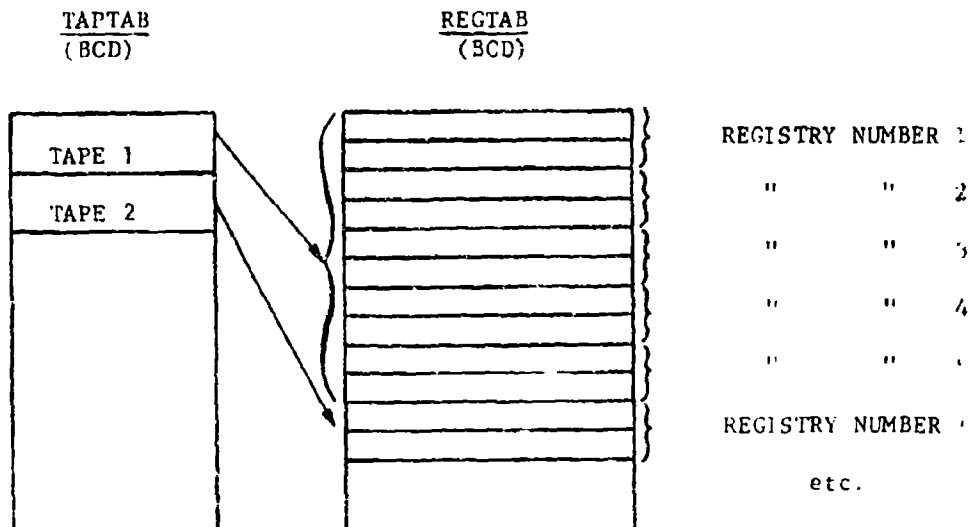
There may be more than one entry for a single paper tape.

(5) TAPXR--pointer to last entry in TAPTAB.

(6) WRTREG--indicator made minus if registry number is to be printed
     out when it has been formatted by ORGNZR (usually associated
     with error message).

(7) DELX--horizontal size of the matrix.

(8) STELOC--two's complement of location with matrix of the S in STEREO.

(9) MTXLOC--location used to hold current matrix location (two's com-
     plement pointer) during processing.

(10) WIPOUT--made minus if record is to be deleted after registry num-
     ber has been formatted and printed. (due to error in input).
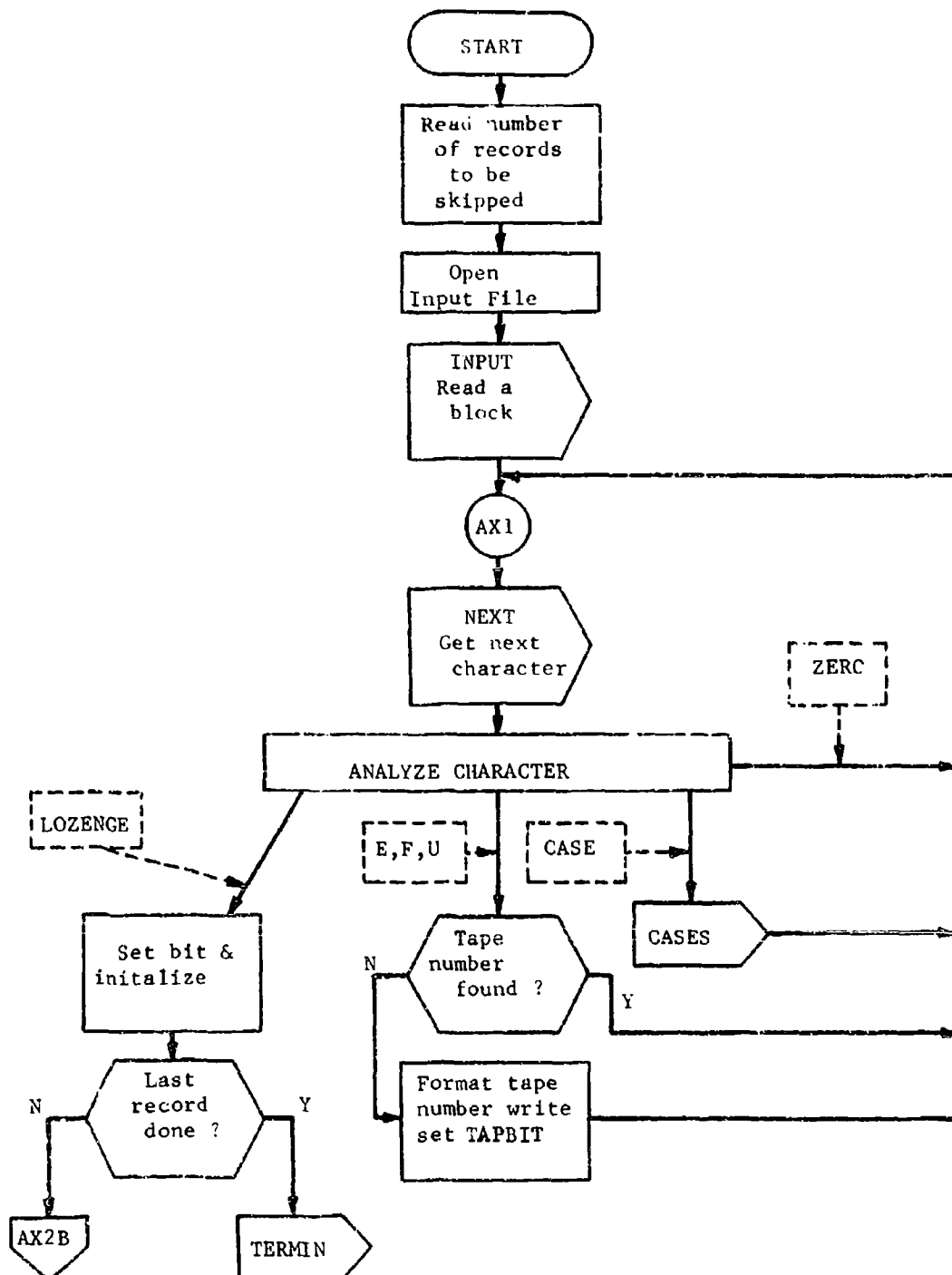
Figure 15. Macro Flow Chart - TAPWRM
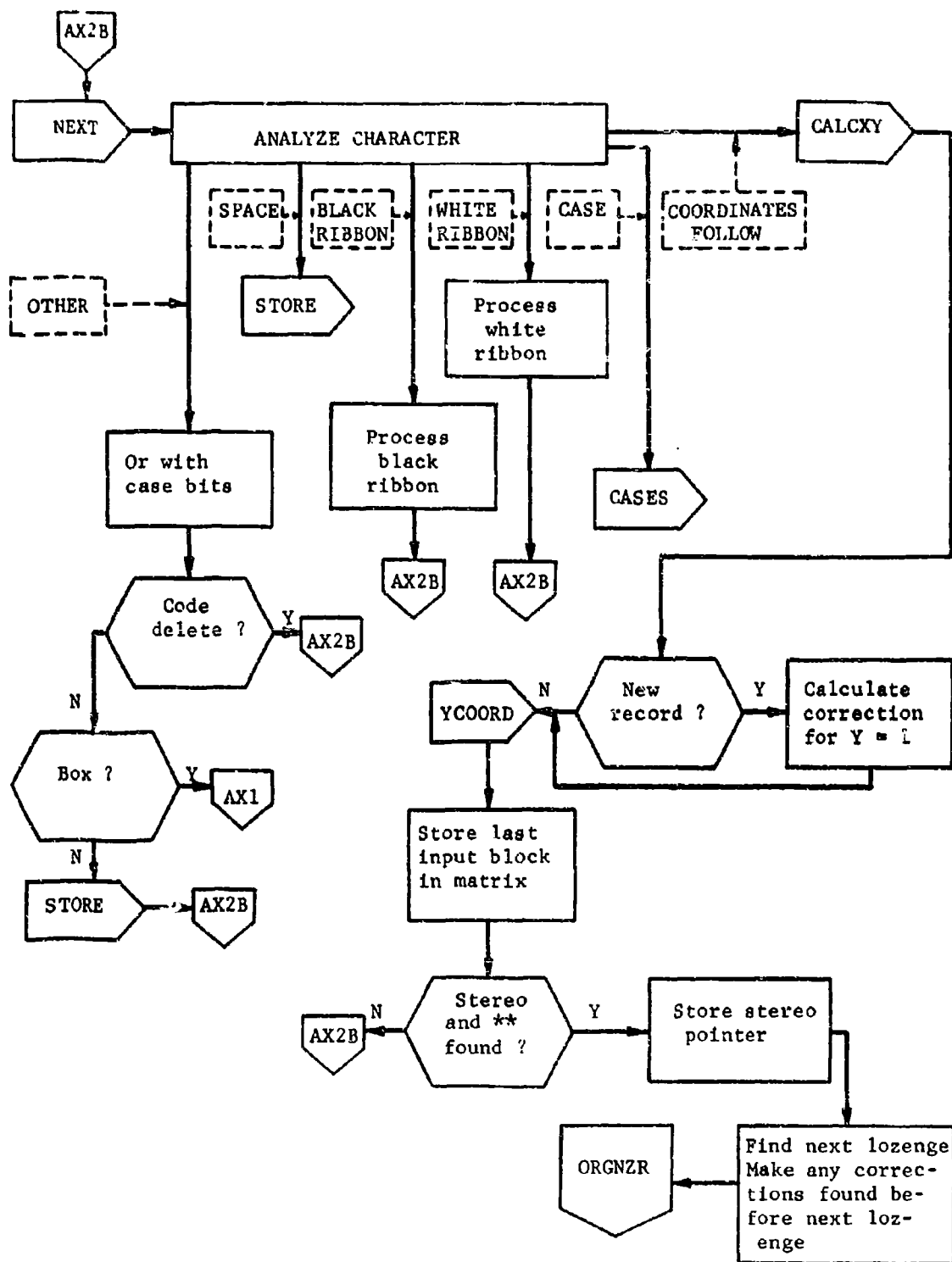
Figure 15. Chart - TAPWRM (continued)

2.2.2   Dura Mach Input Program

Code Name: INPUTD

Programmer: Bruce Hack

Abstract: This program accepts magnetic tape images of the paper tape chemical records typed by the Dura Mach chemical typewriter and reconstructs the chemical record in a 2-dimensional array called MATRIX.

2.2.2.1   Program Description

A macro flow chart describing this program is presented in Figure 9.

Since it is sometimes desired to skip a certain number of physical records on the magnetic tape before beginning to process the input data, the program first reads from a card the number of physical records to skip. The size of the record is specified on the $FILE card.

Certain chemical records may have been seen on the hard copy to be in error before computer processing began. These records may be deleted by inserting cards following the card containing the number of physical records to be skipped.

The tape number is formatted and printed at the beginning of each paper tape. A data block is printed containing the number of records entered in the file and the number rejected at the end of each paper tape.

The program finds the beginning of the chemical record to be processed which is indicated by a special Dura Mach code, a leading wedge (>). Each typed character is then entered into a 2-dimensional array called MATRIX. A cartesian coordinate system is used, so that each typed character occupies a particular (x,y) coordinate. Two coordinate registers are maintained which are incremented or decremented as a result of control character punches such as index, space, reverse index, etc. The occurence of a typed character increments the x-register by 1. Characters which are found to belong in an already occupied matrix location are allowed to replace the character which is already there with the exception of the legitimate overtypes described in Section 2.2.1.1. The end of a record is signalled by a double asterisk (**) in the input stream. The MATRIX is then converted to an internal code (Mergenthaler code with added case bit) so that the input from several devices looks the same to the programs which follow. Field pointers are then set up and the program exits.

2.2.2.2   Program Structure

Input to INPUTD consists of cards as described in Section 2.2.1.2 and magnetic tape containing 8 bit Dura Mach characters formatted as described in Section 2.2.1.2.

Output from INPUTD is the same as that described in Section 2.2.1.2.

59
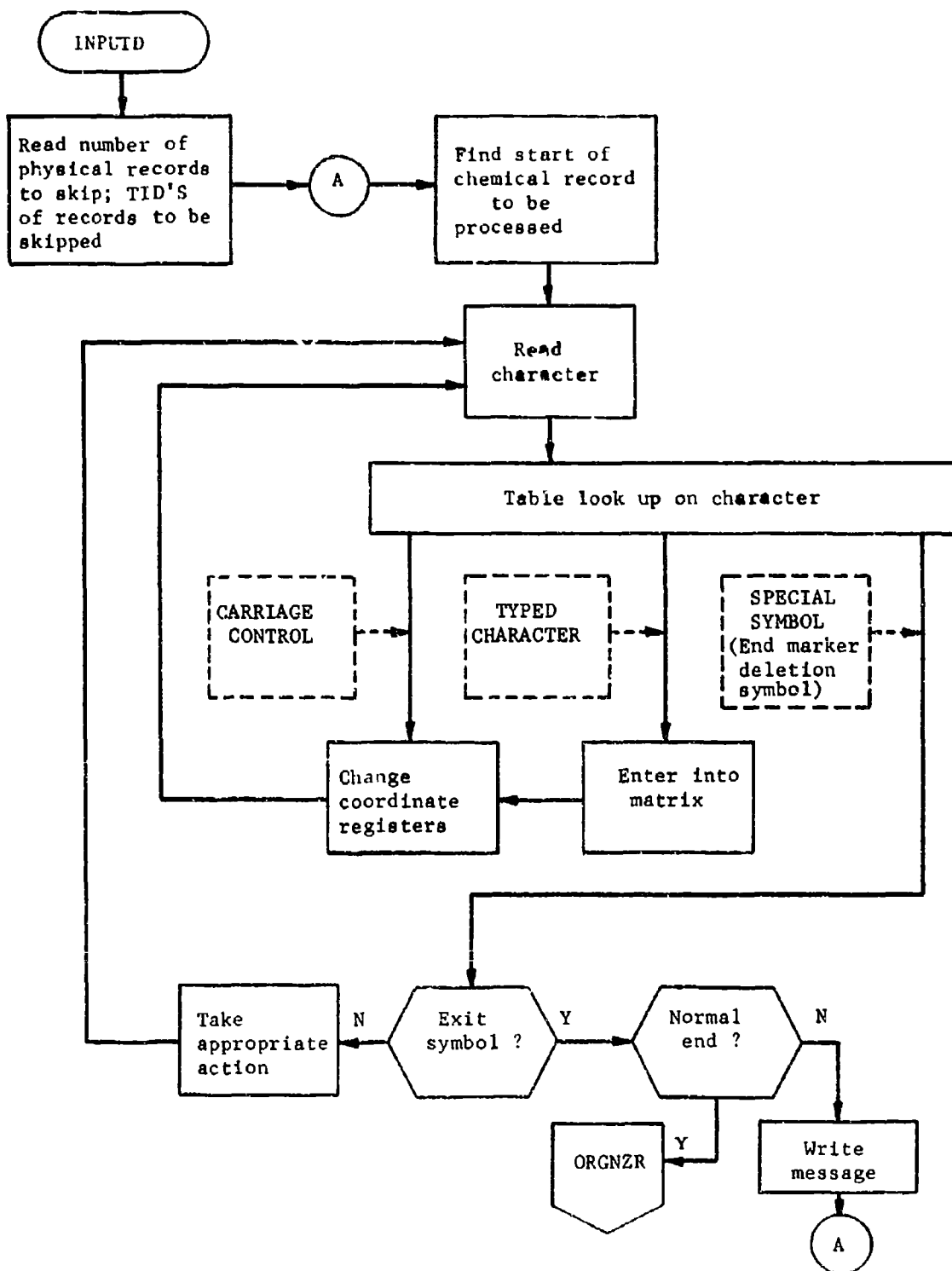
Figure 16. Macro Flow Chart - INPUTD

60

<u>Code Name</u>:   ᴼRGNZR

<u>Programmer</u>:   Helen Hill

   <u>Abstract</u>:   ORGNZR takes the reconstructed matrix (in Mergenthaler code with a case bit added) and recognizes each field in the chemical record. It formats the temporary identification number, the security classification. the molecular formula(both Hill and addend molform if the latter is present), the structural formula image, the stereo information and the nomenclature.

2.2.3.1 Program Description

   A macro flow chart describing this program is presented in Fig. 1.

   The program takes as input the reconstructed matrix containing one chemical record stored in Mergenthaler code with a case bit added, and proceeds as follows:

   (1)   The TID (the first number or letter in the matrix) is found and formatted. The TID must meet the following requirements:

      (a) The number may not exceed 12 digits.

      (b) The number may contain only upper or lower case numbers, upper or lower case letters, commas, periods, and dashes.

      (c) The end of the TID must be signalled by a space and no spaces are allowed within the TID. The TID is stored in BCD in two words starting with REGNO. If WIPOUT has been set by the input program as a result of an error condition, the TID is printed and control is returned to the input program to re-initialize and get the next record.

   (2)   ORGNZR then expects the security classification to follow the TID. The presence of the security classification is signalled by a left parenthesis and the end of the security classification by a right parenthesis or by blanks. The program accepts one of the following in the classification field: (C), (U), (c), (u), () and allows for the addition of other codes in the future. If the first character following the TID is not a left parenthesis, it is assumed that the classification is missing; a special code for a missing classification is assigned (Sec. 2.2.3.2), and the program continues to (3).

   (3)   The program expects the Hill molecular formula to follow the security classification and to begin with either an upper case letter or a number. ORGNZR then calls on MOLFRM to format the molecular formula. If the first character after the classification is not a number or capital letter or a left parenthesis, the record is rejected.

(4) The program then locates the first character following the Hill molecular formula. If this is an upper left bracket, it checks to see whether a single bond is present immediately below it. If the bond is present, the bracket signals the beginning of the structure of a bracketed compound and the program proceeds to (5). If no bond is present beneath an upper left bracket, or if the first character following the molecular formula is a colon, the addend molecular formula is present and ORGNZR calls MOLFRM to format this also.

(5) If the first character following the molform is not a colon or an upper left bracket, it is assumed that this is the beginning of the structural formula. In this case, a list is constructed containing each structural character and its relative matrix location. This is the structural formula image (SFI) which is stored for future reconstruction of the structure for output. If brackets are found to have been typed around any portion of the structure, they are erased except for the upper left and lower right corners whose coordinates will be used to reconstruct the brackets for structural output in PIX and DURPIX. The x coordinate of all lower right corner brackets is stored in table BXBRAK to be used by REGRUP in reordering the SCRUB list. Any multiplier found to the right of this bracket corner is stored in table MULTAB to be used by VERIFY. If a dot is found not connected to a bond, it is assumed that this indicates a monovalent salt and its x coordinate is stored in the bracket table. The last entry in this table is always the maximum x coordinate in the matrix.

(6) It is assumed that the structure must end before the Stereo field is reached (the location of this is provided by TAPWRM or INPUT). When the STEREO field is reached, the information stored here is formatted.

(7) The nomenclature is assumed to start one or more lines under the S in STEREO or one or two spaces to the right of this. When the nomenclature is located, ORGNZR calls MONIKR to format this field whose end is signalled by the double asterisk at the end of the chemical record.

(8) During the formatting by ORGNZR, the following error conditions are considered cause for rejection of the record:

 (a) The TID contains an inadmissable character or more than 12 characters.

 (b) The classification contains an inadmissable character.

 (c) The molecular formula begins with a lower case letter.

 (d) Molecular formula syntax errors were found in MOLFRM.

 (e) An unidentified character was found in matrix.

62

(f) The structural formula extends beyond the STEREO field.

(g) The structure extends into the molecular formula line.

(h) Too many characters were found for the SCRUB list.

(i) The addend molform is missing. At present, it is assumed that if a bracketed structure is present and there is no charge in the structure, an addend molform must be present.

(j) Stereo information does not match with admissible codes.

(9) If brackets were found during the formation of the SFI or if a dot was found not connected to any other character(indicating a monovalent salt), REGRUP is called to reorder the SFI so that all characters within a set of brackets appear compactly in the list. If, in addition, it is found that there are atoms outside the brackets, REGRUP in turn calls EXCESS to format this information which will later be used by the chemical verification program.

2.2.3.2 Program Structure

ORGNZR requires 4150 core locations and includes the following Tables which are also used by other programs.

(1) AXACT2 - Mergenthaler input codes without a parity bit but with a case bit added.

(2) INT - Internal Code (modified Dura Mach) counterparts of table AXACT2.

(3) AXBCD1 - BCD numbers

(4) AXBCD - BCD letters

It contains the definitions of the following large blocks:

SCRUB list - 700 locations

B list	1201	
Ex list	100	used by CONVRT and shared by REGRUP
Y	400	
G	400	

It makes use of a subroutine, DELETE, which can delete from processing any compounds whose TID's are entered on data cards and stored in REGTAB and TAPTAB (supplied by the input programs).

Input to ORGNZR consists of:

(1) DELX - width of the matrix.

(2) MATRIX - the reconstructed chemical record.

(3) STELOC - the location of the STEREO field in the matrix.

(4) REGTAB and TAPTAB - tables giving paper tape number and TID's of compounds to be deleted.

Output from ORGNZR is as follows:

(1) SCRUB - list of all structural characters and their relative matrix location formatted as follows:

3 ⟷ 17	20			30 ⟷ 35
Relative matrix loc.				Modified dura code

Case bit
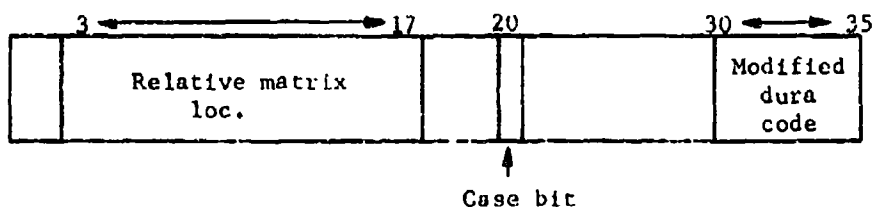
The relative matrix location is given relative to the start of the 1st location, one y coordinate before the first Structural image y coordinate.

(2) REGNC - the formatted TID, up to 12 BCD characters left justified.

(3) CLSTER - contains stereo and security classification information presently in use, and which can be greatly expanded in the future.
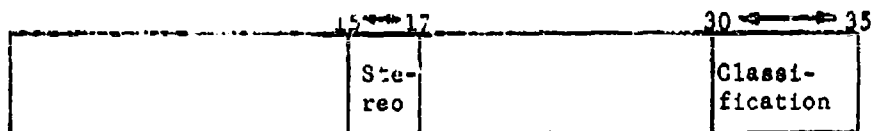
	15 ⟷ 17		30 ⟷ 35
	Ste-reo		Classi-fication

1 = Non-Stereo   1 = unclassified

2 = Stereo   2 = blank

3 = Stereo iridentified 3 = classified

0 = not present.

(4) DELY — actual y coordinate size of the matrix required by the structure.

(5) ASCRUB - pointer to the end of scrub lis

(6) BXBRAK - table of x coordinates of right hand brackets in structure

18 ←	→ 35
	x coordinate
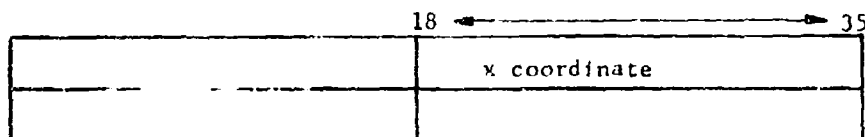
last x coord is = DELX

(7) GRP2 - pointer to last entry in SCRUB that lies within brackets if they are present, and last entry in SCRUB if brackets are not present.

(8) AXTMOD - absolute address of first location of the SFI in the MATRIX.

(9) MULTAB - table of multipliers for each portion of a structure that is within a set of brackets.

MULTAB

18 21 22 ←	→ 35
		multiplier
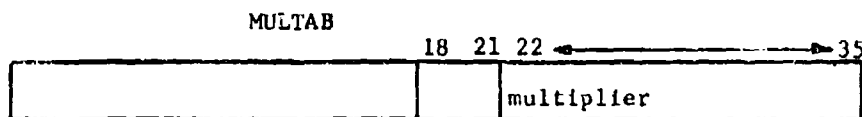
last multiplier is = to 1

(10) UNDTAB - A table of underlines (10 locations) which contains the SCRUB list entries of underlined characters to be used in reconstructing the matrix for print out.

(11) MOLTAB - formatted molecular formula as described in Section 2.2.4.2.

(13) NOMTAB - formatted nomenclature and reference fields as described in Section 2.2.5.2.

(14) MATRIX - the Structural formula in the matrix is stored in modified Dura Mach code as follows on output from ORGNZR.

20	30 ← → 35
	Dura code

Case bit
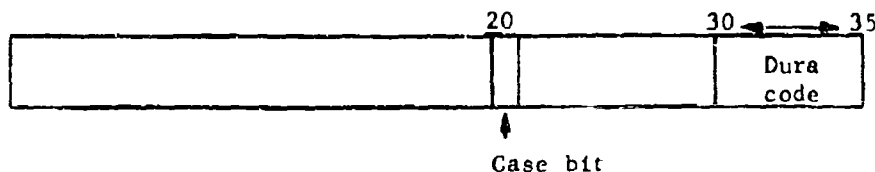
The rest of the record in the matrix remains in Mergenthaler Code.

The program produces diagnostic P-dumps when sense switch 5 is set. When
sense switch 6 is set, the formatted information is printed at the end of ORGNZR.
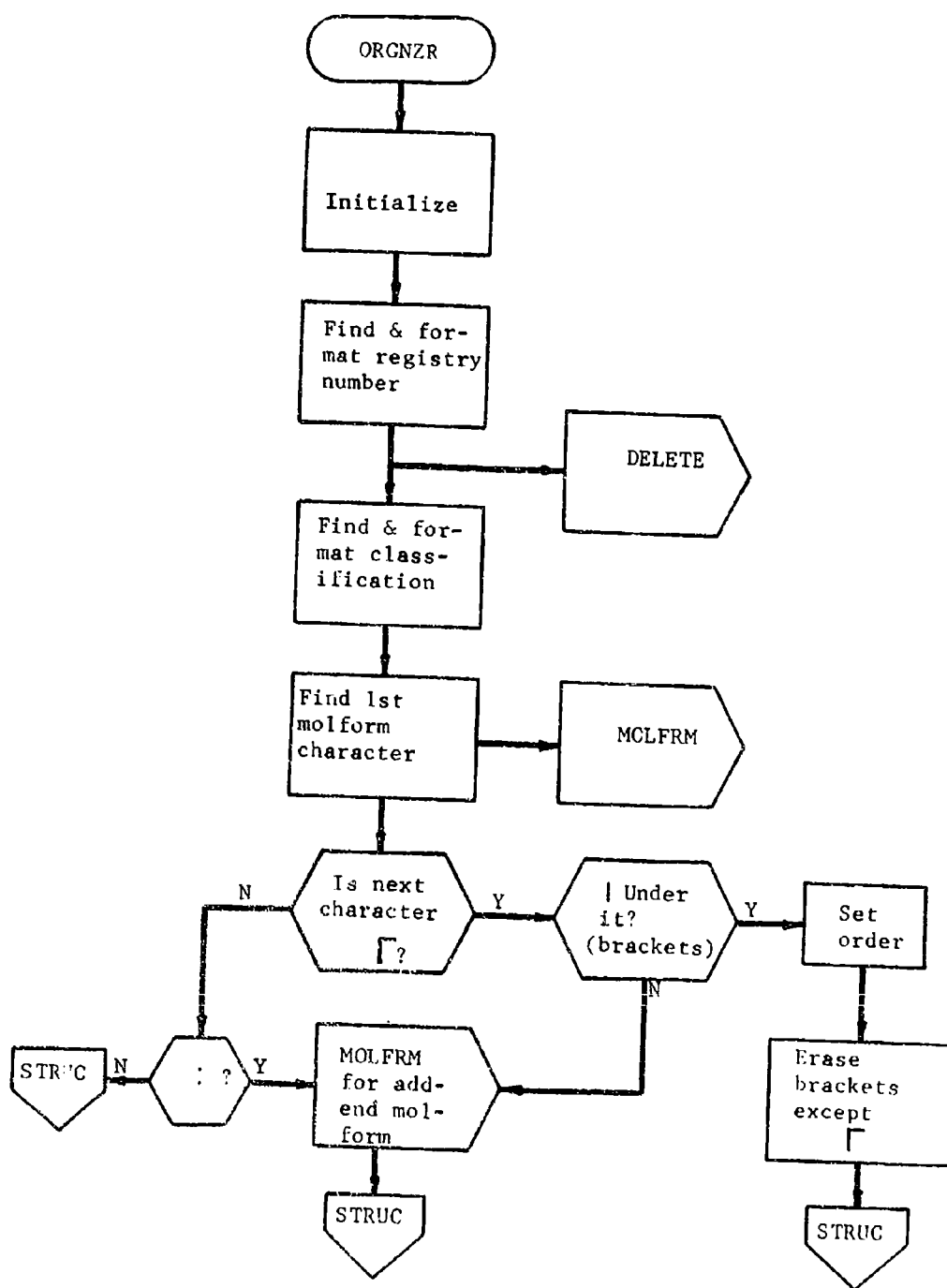
Figure 17. Macro Flow Chart — ORGNZR

67
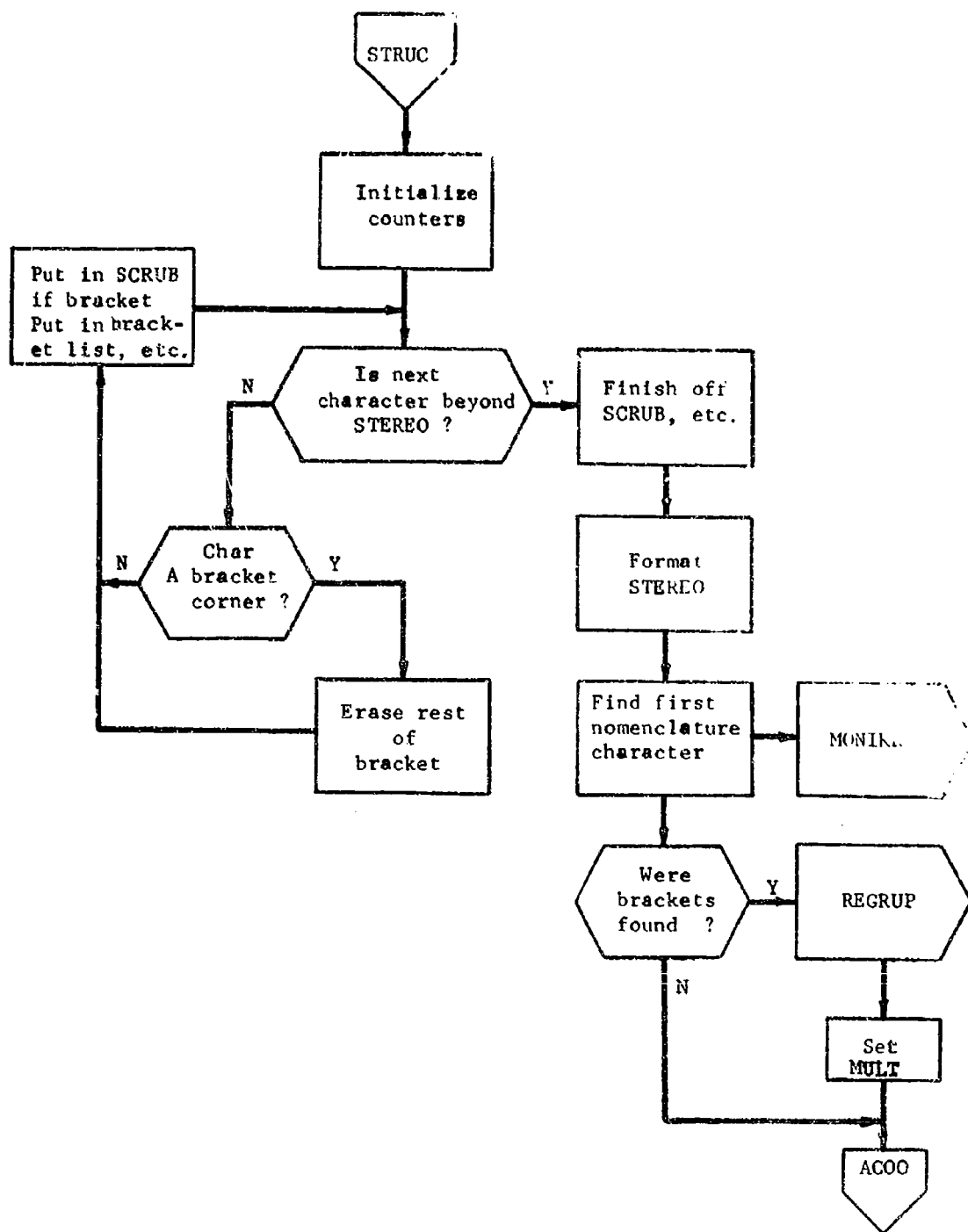
Figure 17. Macro Flow Chart — ORGNZR (continued)

68

2.2.4 Molecular Formula Format Program

Code Name: MOLFRM

Programmer: Helen Hill

Abstract: This program formats the molecular formulas.

2.2.4.1 Program Description

A macro flow chart describing this program is presented in Figure 19.

MOLFRM is provided with a pointer to the first molecular formula character in the matrix by ORGNZR. It then produces the formatted molecular formula.

The following are definitions* of syntactical terms:

<PREFIX>::=<NUMBER>|<EMPTY>

<SYMBOL>::=<CAPITAL LETTER>|<CAPITAL LETTER> <SMALL LETTER>

<SEPARATOR>::=<DOT>|<BLANK> <DOT>|<BLANK> DOT <BLANK>|<DOT> BLANK

<TERMINATOR>::=<BLANK> <BLANK>

<SUBSCRIPT>::=<NUMBER>

<COMPOUND SYMBOL>::=<SYMBOL>|<SYMBOL> <SUBSCRIPT>

<FORMULA PART>::=<COMPOUND SYMBOL>|<COMPOUND SYMBOL> COMPOUND SYMBOL>

The program requires the following syntax for the Hill Molecular formula of the simple, non-hydrated compound:

<HILL MOLFORM>::=<FORMULA PART> <TERMINATOR>

The hydrated Hill molecular formula must have the following syntax which allows a multiplier for either or both the Hill parent and the water.

<HYDRATED HILL MOLFORM>::=<PREFIX> <FORMULA PART> <SEPARATOR> <PREFIX>

<FORMULA PART> <TERMINATOR>

The addend molecular formula which must be present in addition to the Hill Molecular formula for addends is either preceded by a colon (:) or is surrounded by an upper left and upper right bracket. The addend molecular formula requires the use of the following expanded definitions:

<TERMINATOR>::=<BLANK> <BLANK>|<UPPER RIGHT CORNER> <BLANK> <BLANK>

<SIGNAL>::= |<UPPER LEFT CORNER>

<SEPARATOR>::=<DOT>|<BLANK> <DOT>|<BLANK> <DOT> BLANK>|<DOT>

<BLANK>|<DOT> <BLANK>|<UPPER RIGHT CORNER> DOT |

<UPPER RIGHT CORNER> <DOT> <BLANK>

* Backus-Naur Form.

<ADDEND PART>::=<PREFIX> <FORMULA PART> <SEPARATOR>

<FINAL ADDEND PART>::=<PREFIX> <FORMULA PART> <TERMINATOR>

The syntax required for the addend molecular formula is as follows:

<ADDEND MOLFORM>::= <ADDEND PART> <FINAL PART>|<ADDEND PART>

                     <ADDEND PART> <FINAL PART>|<ADDEND PART>

                     <ADDEND PART> <ADDEND PART> <FINAL PART>

The addend molecular formula syntax allows for handling compounds consisting of up to four addend parts. In the case of addend molecular formulas, water of hydration is treated as an addend part. Provision has been made, however, not to reject a compound in which the typist has placed the dot and the water outside the upper right corner bracket.

This program assumes that all polymers admitted to the system are condensation polymers and will ap_r with the sum of their atom counts in the Hill molecular formula. Condensation polymers of indefinite size are assumed to be present in a $(...)_n$ form and a bit is set by ORGNZR which instructs MOLFRM to ignore the n.

2.2.4.2 Program Structure

MOLFRM is a subroutine requiring 448 core locations. It uses tables in ORGNZR when recognizing Mergenthaler characters and converting them to BCD. Its output, MOLTAB is a 25 location table.

The input to MOLFRM is:

      (a)   Pointer to first molform character in MATRIX

      (b)   MATRIX

The output from MOLFRM is:

      MOLTAB - formatted molecular formula is shown in Figure18.

WORD 0

0	1-2	3 ⟵——————————⟶ 17	18 ⟵————⟶ 26	27 ⟵————⟶ 35
		Total no. words in molform block including header	Multiplier of Hill parent if a hydrate	Multiplier of water if a hydrate

Set to 1 if addend molform exists

4 Bits	4 Bits	

numerator   denominator

1 if multiplier is a fraction; 0 if an integer.

If it is not a fraction, multiplier fills 8 Bits, right justified.

Set if indefinite polymer

WORD 1

1 ⟷ 3	4 ⟵——⟶ 10	11 ⟵——⟶ 17	18 ⟵———⟶ 26	27 ⟵——⟶ 34	35
	Number of oxygen atoms	Number of nitrogen atoms	Number of hydrogen atoms	Number of carbon atoms	

No. words in mol. formula (Hill)

WORD 2-M (Hill)

S

0 ⟵————⟶ 11	12 ⟵⟶ 17 — 18 ⟵———————⟶ 29	30 ⟵—⟶ 35
Element kind (BCD)	Number of atoms of element	Element kind (BCD)	Number of atoms of element

WORD M+1

0 ⟵——⟶ 8	9 ⟵———⟶ 17	18 ⟵———⟶ 26	27 ⟵————⟶ 35
Multiplier of first addend	Multiplier of second addend	Multiplier of third addend	Multiplier of fourth addend

FORMAT OF MULTIPLIER SAME AS WORD 0 MULTIPLIER

WORD M+2   SAME AS WORD 1 -- but for addend molform

WORD M+3   SAME AS WORD 2—M -- but for addend molform

Figure 18.   Formatted Molecular Formula

In the preceding, the following rules hold:

1. If the element type is a single letter element, the first character in words 2-M is a BCD blank.

2. If no addend molform exists, the block ends with word M. If there is more than one addend there is a separate block for each addend containing words M+2 to M+n for that addend.

3. If the only addend is water, no addend form appears. If there is an addend in addition to water, the water appears in the Hill form as above and in the addend form. When the only addend is water, the addend bit is not set in the Header word.
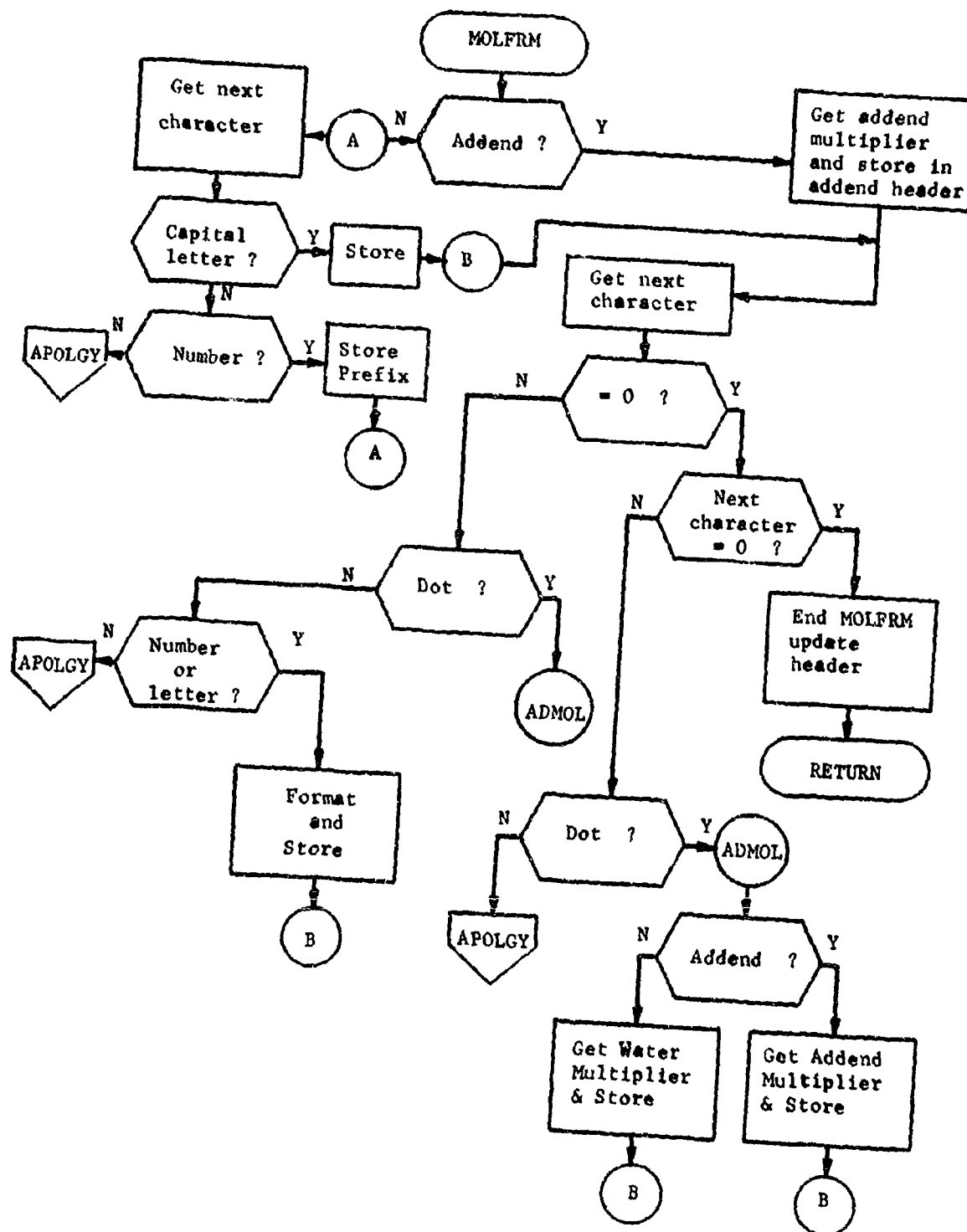
Figure 19.   Macro Flow Chart   -   MOLFRM

73

2.2.5 Nomenclature And Reference Field Formatting Program

Code Name: MONIKR

Programmer: Helen Hill

Abstract: MONIKR formats the nomenclature and any other information typed with it.

2.2.5.1 Program Description

A macro flow chart describing this program is presented in Figure 20.

MONIKR takes a pointer to the first word in the nomenclature and formats all the information it finds until a double asterisk (**) is encountered. The presence of a double bond or triple bond in this field signals the start of the reference field. MONIKR allows for the presence of underlines, superscripts, and any character which can be typed on a chemical typewriter with the exception of certain bonds. It takes the characters which are in modified Mergenthaler code in the MATRIX and translates them to the following for storage.

```
 ___ ___ ___ _____
|   |   |   | 6 Bit   |      9 Bit character
|___|___|___| Dura Code|
   ↑   ↑   ↑
Superscript Underline  Dura case bit
```
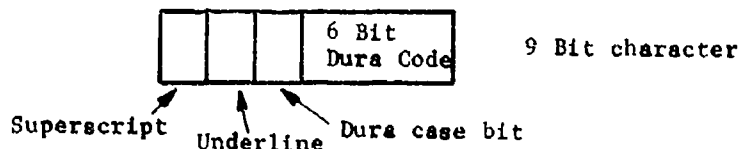
Each line of nomenclature is taken to be all the material in the same coordinate and followed by at least three blanks in the matrix and is delimited from the other lines in the output table. The same is done with the reference material.

MONIKR allows up to 400 characters in these fields and rejects the compound if there are more. It also checks to make certain it has not exceeded the matrix area while getting characters to store.

MONIKR calls PUNCH if it is desired to punch out certain descriptor information on cards.

2.2.5.2 Program Structure

This program is a subroutine which requires 423 core locations. It uses tables in ORGNZR to recognize Mergenthaler characters and translate them to modified Dura Mach.

Input to MONIKR consists of:

    (1) Pointer to first character in field.

    (2) MATRIX - described in Section 2.2.1.2.

74

Output from MONIKR consists of NOMTAB, a table of up to 100 locations containing the formatted nomenclature and reference field. Each nomenclature entry (one line of nomenclature) is delimited by a 777_8 character from the next. The last nomenclature word is filled out with zeros. References are separated from each other by a 777_8 character and the beginning of the reference field is preceded by a triple bond character in the output block. The last word in the reference field is also filled with zeros.

3 ⟵————————➤17		21⟵————————➤ 35	
2's comp. of number nomenclature words		2's comp. of total number of words in block	Header of nomenclature block
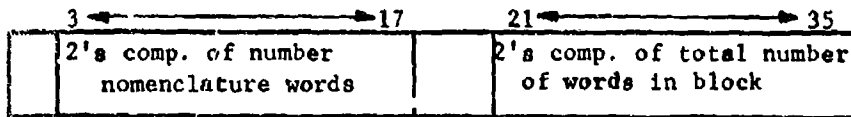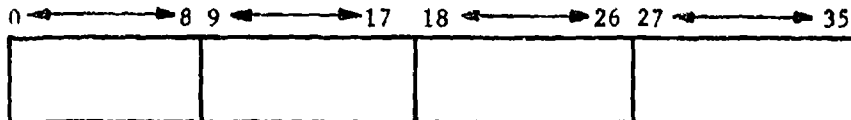
0 ⟵———➤8 9 ⟵———➤17	18 ⟵———➤26	27 ⟵———➤ 35	
			4 characters per 7040 word
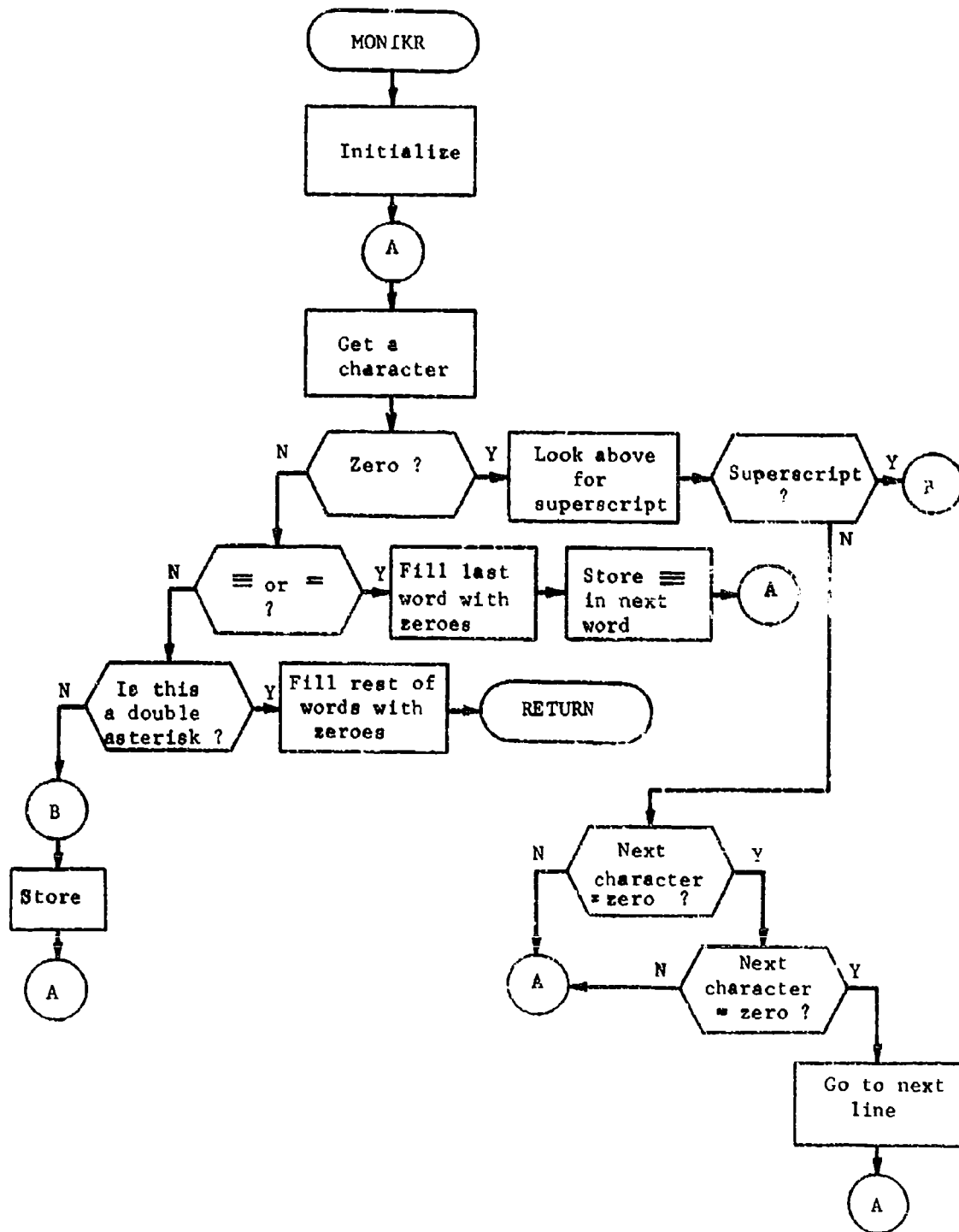
Figure 20. Macro Flow Chart -   MONIKR

76

2.2.6   Descriptor Punch Program

Code Name: PUNCH

Programmer: R. Chao

Abstract: This program finds the EA, T, and TL descriptors, if there are any. It then gets the corresponding TID number of the compound and punches it on a card followed by the EA, T, or TL descriptor number.

2.2.6.1   Program Description

A macro flow chart describing this program is presented in Figure 21.

PUNCH gets the matrix and the first location of the reference field as input from the main program and searches for EA, T, and TL descriptors. If one of the descriptors listed above is found, a card is punched containing the TID compound number and the descriptor.

2.2.6.2   Program Structure

PUNCH is a subroutine called from MONIKR which takes as input the MATRIX and RFIELD which points to the reference field. Its output is cards with the following format:

Card Column	1-12	12 character TID # (left justified)
	13	Blank
	14, 15	Codes EA, T, TL
	16-20	Number (right justified)
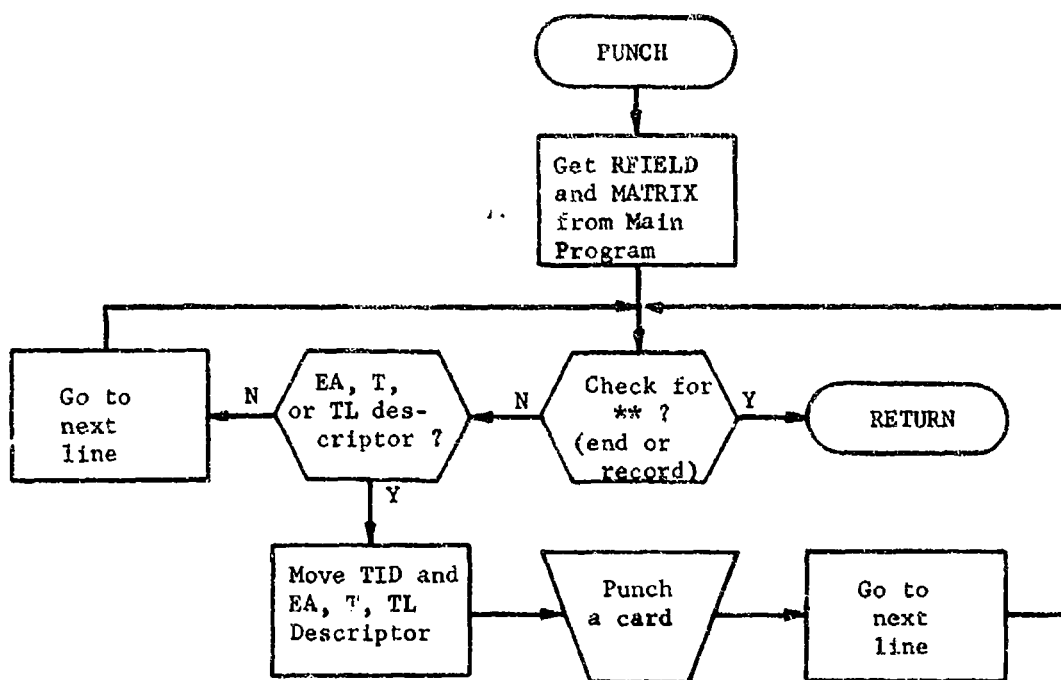
Figure 21. Macro Flow Chart — PUNCH

2.2.7   SFI Reordering Program

Code Name: REGRUP

Programmer: Helen Hill

Abstract: Program reorders the SFI when brackets or a monovalent salt are present so that all characters within a given set of x coordinates appear compactly in the SFI.

2.2.7.1   Program Description

A macro flow chart describing this program is presented in Figure 22.

REGRUP takes the list of x coordinates of right lower corner brackets and reorders them to make certain the x coordinates are in an ascending order, checking for the following error conditions:

(1)   2 identical x coordinates

(2)   An empty bracket list

These errors result in the rejection of a chemical record.   It then takes each entry in the scrub list, computes the x coordinate from the relative matrix location, and stores this entry in the appropriate list.   The following errors result in rejection of a compound:

(1)   Too many characters for a given list

(2)   Empty or incorrect bracket list

REGRUP stores a pointer to the last entry in each list (which corresponds to one set of brackets, the final set being assumed to be everything to the right of the last explicit bracket) in MULTAB which will be used by VERIFY.   It then replaces the reordered lists in order in the SFI and stores a pointer to the first character beyond explicit brackets (if any).   REGRUP counts up all plus and minus charges outside of explicit brackets and stores the totals in the Header word of the SFI.   If characters appear outside the explicit brackets, REGRUP calls EXCESS to format them.

2.2.7.2   Program Structure

REGRUP is a subroutine which requires 303 core locations.   It uses areas defined in ORGNZR (13^0 locations) to reorder the various lists and it uses the SCRUB list from ORGNZR (701 locations).

It is made up of 5 Macros:

(a)   ADDUP and ORDER - sort the brackets.

(b)   LIST - puts an entry in proper list and increments right pointers.

(c) REFORM - transmits reordered lists to SCRUB list after reordering.

(d) STOMLT- stores necessary information in MULTAB.

REGRUP takes as input the following:

SCRUB - SFI characters and their Relative Matrix location.

BXBRAK - list of x coordinates of lower right corner brackets.

MULTAB - contains multiplier associated with each set of brackets.

ASCRUB - pointer to end of SCRUB.

Output from REGRUP is as follows:

(a) Reordered SCRUB

(b) MULTAB - with pointers to last character in SCRUB using a given multiplier
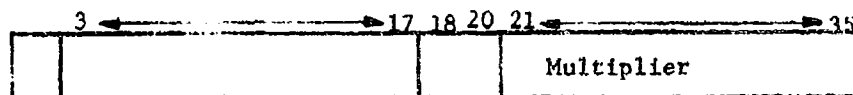
2's comp. pointer
to last entry in
SCRUB list for this
Multiplier (and this
set of brackets)
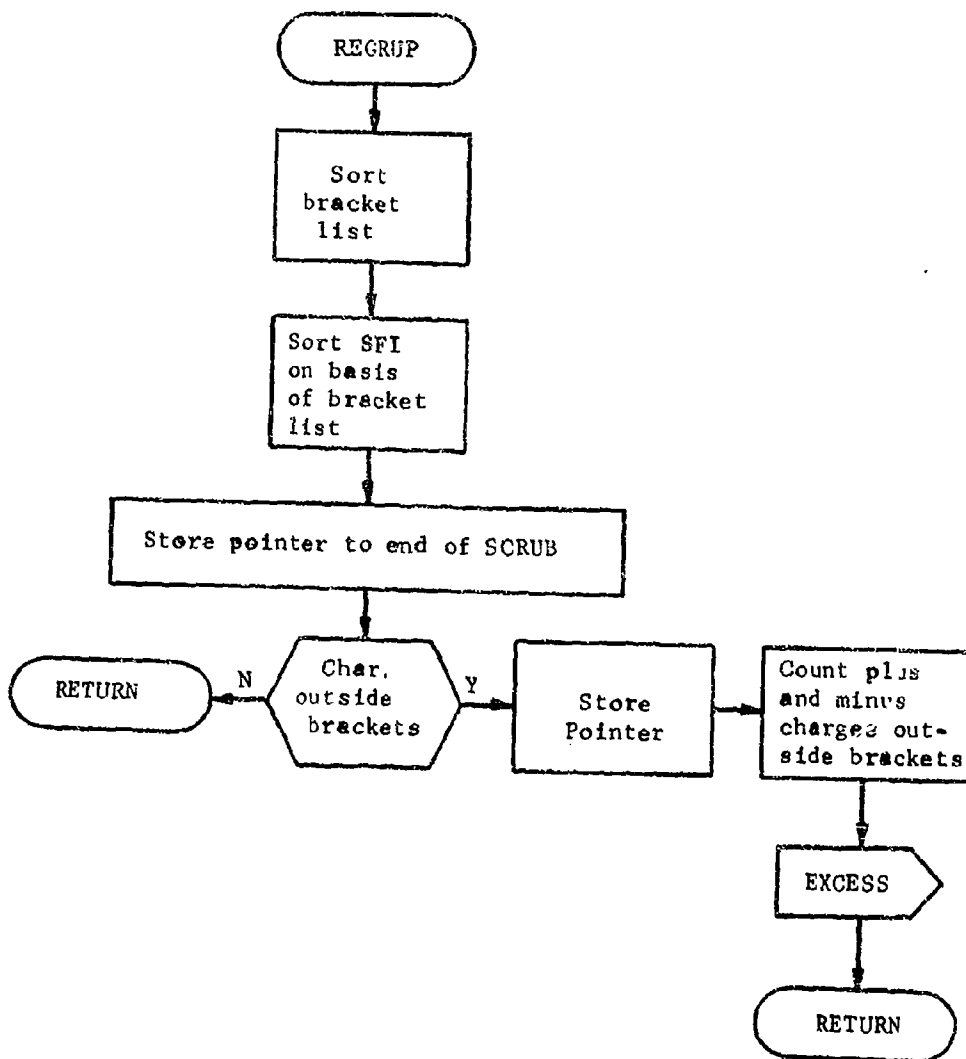
Figure 22. Macro Flow Chart - REGRUP

2.2.8 Structure Of Non-Bracketed Information

Code Name: EXCESS

Programmer: Helen Hill

Abstract: EXCESS formats all structural characters appearing outside of brackets when brackets appear in the structure.

2.2.8.1 Program Description

A macro flow chart describing this program is presented in Fig. 23.

EXCESS formats information found outside brackets in structures such as (....) H_2SO_4 since this portion of the compound does not appear in the connection table and is needed for chemical verification. EXCESS uses the pointer to the first word outside brackets in the SCRUB list to locate that character in the matrix using the relative matrix location in the SCRUB list for that character. It then sets a bit (ION) to be used by VERIFY and looks for the first addend dot in the matrix outside of brackets. It formats all characters on that y coordinate from that x coordinate to the end of the line. EXCESS totals plus and minus charges outside brackets, using prefix multipliers if any. EXCESS requires the following syntax in the information it formats.

<SYMBOL>::=<CAP>|<CAP> <SMALL LETTER>

<SUBSCRIPT>::= <NUMBER>

<COMPOUND SYMBOL>::=<SYMBOL>|<SYMBOL> <SUBSCRIPT>

<FORMULA PART>::=<COMP. SYMBOL>|<FORMULA PART> <COMP. SYMBOL>

<EXCESS FORMULA PART>::=<SIGNAL> <PREFIX> <FORMULA PART> <COMP. SEPARATOR>

<PREFIX>::=<NUMBER> | ∅

<SIMPLE SEPARATOR>::=<BLANK> | ∅

<COMP. SEPARATOR>::=<SEPARATOR>|<SEPARATOR> <SEPARATOR> | + <SEPARATOR> |
     , <SEPARATOR>

<SIGNAL>::= . | .<BLANK>

Everything that is found having the same y coordinate as the first addend dot outside the brackets is formatted provided it satisfies the following definition:

<FORMAT>::= EXCESS FORMULA PART> | <FORMAT> <EXCESS FORMULA PART>

The prefix in this case is used to multiply each compound symbol until the next prefix is found.

When water is found in the string it is ignored. EXCESS allows any combination such as (.H_2SO_4.2Cl¯.H_2O) as long as the elements are all typed on the same line.

2.2.8.2 Program Structure

EXCESS is a subroutine which requires 335 core locations and utilizes tables AXACT2 and INT in ORGNZR. It contains a subroutine, SEARCH, which takes a given SCRUB list character and translates it to Mergenthaler code to allow range tests to be done for syntax analysis.

EXCESS takes as input the following output from ORGNZR described in section 2.2.3.2.

SCRUB
MULTAB
DELX
GRP2   See ORGNZR Output, Section 2.2.3.2.
MATRIX
SFILOC

Output from EXCESS is the following:

XTAB - a table formatted as follows.

if a single letter
element, the second
character is a BCD blank
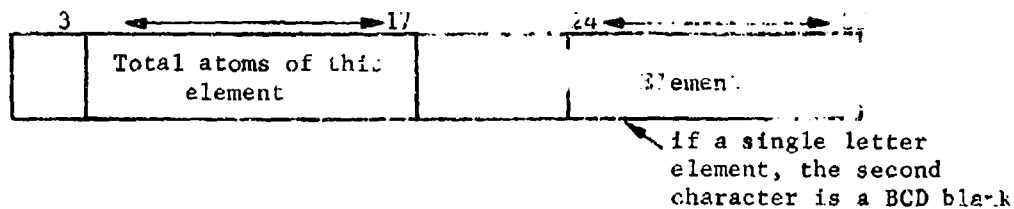
PL - total plus charges outside brackets

MIN - total minus charges outside brackets

ION - set to signal that there is something in XTAB for VERIFY to use.
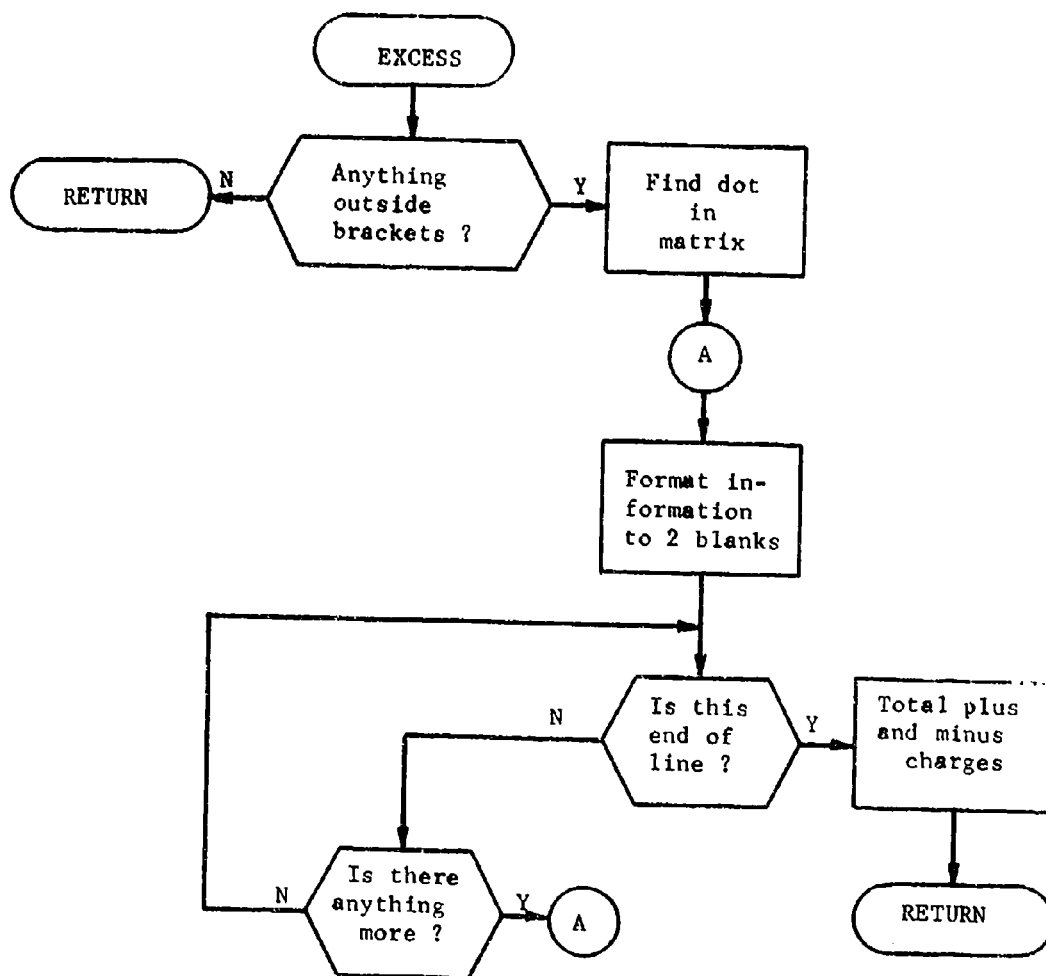
Figure 23. Macro Flow Chart - EXCESS

84

2.2.9　Error Message Program

Code Name:　APOLGY

Programmer:　Helen Hill

Abstract:　APOLGY writes error messages.

2.2.9.1 Program Description

APOLGY is transferred to from ORGNZR, EXCESS, REGRUP, MOLFRM and MONIKR to write error messages using Fortran read-write routines.

```
  ╭─────────────╮
  │   APOLGY    │
  ╰──────┬──────╯
         │
  ┌──────▼──────┐
  │ Print       │
  │ appropriate │
  │ message     │
  └──────┬──────┘
         │
  ┌──────▼──────┐
  │   INPUT     │
  │    or       │
  │   TAPWRM    │
  └───────────┘
```
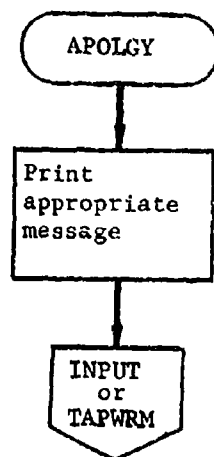
2.2.9.2　Program Structure

APOLGY is a Subroutine.

2.2.10 Linear String Classification

Code Name: SETUP

Programmer: Bruce Hack

Abstract: This program finds a capital letter in the SCRUB list and then scans to the left and right of this letter in the MATRIX assigning a type code to the linear string. It then transfers to CLEANM for processing.

2.2.10.1 Program Description

A macro flow chart describing this program is presented in Figure 24.

SETUP scans the SCRUB list until a capital letter is found and then locates the left bound of the linear string on the basis of the following definition:

Definition: A _linear string_ is a set of symbols on a horizontal line bounded on the left and right by bonds or blanks and containing at least one capital letter.

A left to right scan is made of the linear string placing each symbol in one of the following classifications:

(a) C

(b) P

(c) H

(d) Other capital

(e) Small letter

(f) (

(g))

(h) number

(i) Illegal symbol

The concatenation of these classifications is the code for the string.

e.g. $-(CH_2)_6-$ has the code: fachgh

An illegal symbol immediately causes rejection of the record. Control is given to CLEANM upon concatenation.

2.2.10.2   Program Structure

SETUP is a main program which receives control from ORGNZR and gives control to CLEANM when through.

Input to SETUP are the SCRUB list and the MATRIX. These are described in Section 2.2.3.2.

Output from SETUP is LINSTG, the code classification of a node.
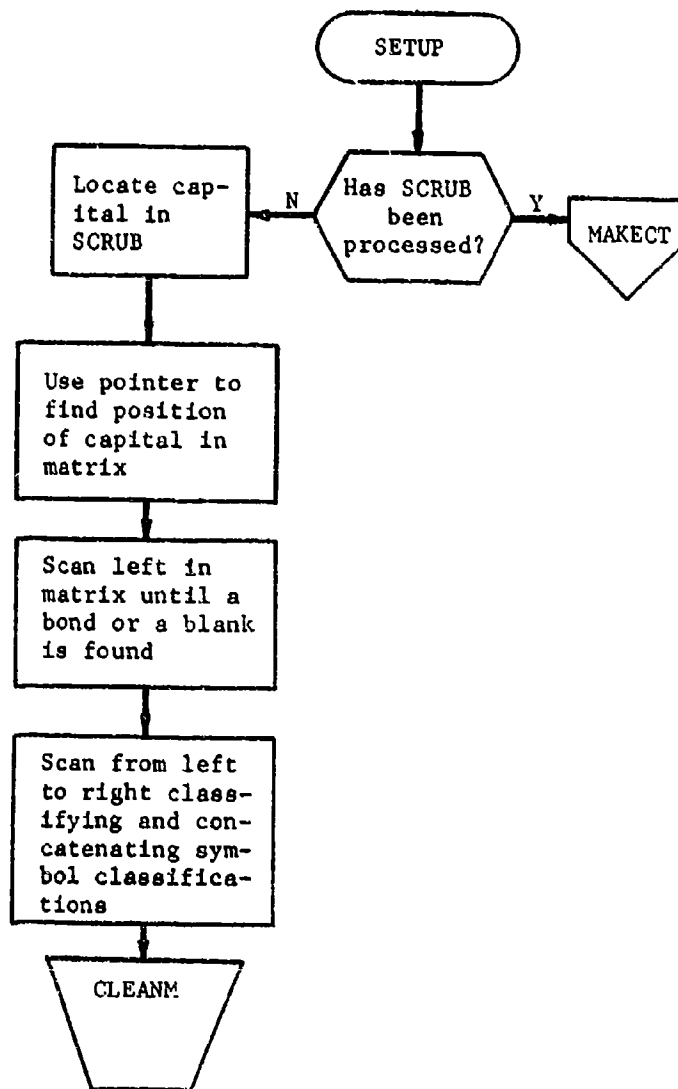
Figure 24. Macro Flow Chart — SETUP

83

2.2.11 Reduction Of The Matrix To Points And Lines.

Code Name: CLEANM

Programmer: Bruce Hack

Abstract: CLEANM is given a pointer to a specific node by SETUP. It then "cleans" the eight locations around that node in the matrix for use in MAKECT. All charge signs and mass numbers are removed, double letter elements are replaced by a one word symbol, and special cases such as Ph and $-(C)_n-$ are treated. An abnormality table of abnormal masses, charges and valences is created. A connection table number is assigned to each atom and the word in SCRUB corresponding to a node which has been processed by CLEANM is made minus. Control is returned to SETUP after operation on the given node is complete.

2.2.11.1 Program Description

A macro flow chart describing this program is presented in Figure 25.

A code for the node to be operated on is provided by SETUP, classifying the node in one of the following classes:

(a)   Single letter atom (e.g., C)

(b)   Double letter atom (e.g., Cl)

(c)   H

(d)   Ph (representing a phenyl or ( ) group)

(e) $-(C)_n-$

(f) $-(CH_2)_n-$

In the typed structure, it is assumed that any symbol in any of the 8 positions surrounding an atom belongs to that atom.

Example

The bond shown is assumed to belong to the N. This positioning of a bond is incorrect and the computer will reject the record.
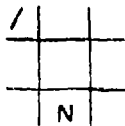
The bond shown here is assumed not to belong to the N.

NOTE: By "belong to" we mean connected to in the case of
      a bond or associated with in the case of a charge sign,
      mass number, etc.

The following actions are taken by CLEANM:

(a) The sign of the SCRUB word pointing to this atom is made
    minus for use in MAKECT and the next available connection
    table number is then assigned to this atom. This number
    is placed in the MATRIX word containing the atom. A search
    is then made around the atom, counting the connecting bonds
    for comparison with the normal valence for that atom. If
    the valence is found to be larger than the normal valence,
    an entry is r de in the abnormality table. The upper left
    is checked for a mass number. If one is found, an entry
    is made in the abnormality table. The number is then removed
    from the matrix and replaced by a bond if necessary.

e.g.

before

after

Note: The "1" need not be erased
      since it will not interfere
      with future processing.

Similarly the upper right is checked for a charge and an entry is
made in the abnormality table if one is found.

(b) The lower case letter of a 2 character element is placed in
bits 24-29 of the MATRIX entry for the upper case letter and
the lower case letter is replaced in the matrix by a bond
connection if necessary. The following example describes
this process.

C | L | —   MATRIX before (b)

LC | — | —   MATRIX after (b)

control is then given to case (a).

(c) The presence of a hydrogen atom with a single connection is
ignored. A hydrogen atom with two connections is treated as
a single letter atom and control is given to case (a). The
presence of a hydrogen in the connection table will later
cause rejection of the record during chemical verification.

(d) The phenyl group (Ph) is replaced by a single C atom in the
MATRIX and the internal connections of the expanded phenyl
group are placed directly in the connection table. The sign
of the SCRUB list word containing the "P" is made minus and
the first and last connection table numbers of the atoms in-
cluded in the expansion are entered into the C word in the
MATRIX.

P | h | —

before

after

```
┌──┬──┬──┬──┐
│  │C │──│──│
├──┼──┼──┼──┤
│  │  │  │  │
└──┴──┴──┴──┘
```

(e) Carbon chains are replaced by a special code in the MATRIX. The SCRUB list words are then handled as in case (d).

before

```
┌──┬──┬──┬──┬──┬──┐
│──│( │C │) │3 │──│
├──┼──┼──┼──┼──┼──┤
│  │  │  │  │  │  │
└──┴──┴──┴──┴──┴──┘
```

after

```
┌──┬──┬──┬──┬──┬──┐
│──│──│W │──│──│──│
├──┼──┼──┼──┼──┼──┤
│  │  │  │  │  │  │
└──┴──┴──┴──┴──┴──┘
```

CLEANM rejects chemical records for the following reasons:

(1) More than 19 abnormalities present in compound.

(2) Inadmissible string found in structural formula

(3) Illegal symbol found around an atom.

(4) Hydrogen found in the wrong place.

2.2.11.2 Program Structure

CLEANM is a main program which is transferred to by SETUP and which transfers back to SETUP when processing is complete.

Input to CLEANM consists of the

(a) SCRUB - list which contains all structural formula information.

(b) MATRIX - the coded two-dimensional picture of the structural formula.

(c) DELX - the width of the MATRIX.

92

Output from CLEANM is the following:

(a) Partial CONNECTION TABLE (i.e., expanded Ph and $-(C)_n-$)

(b) ABNORMALITY TABLE - this table is a series of words, where each word gives information about one atom which has abnormal mass or valence or has a charge on it.
i.e.

Bits	Contents
(S,1,2)	Type of abnormality 101=charge 110=mass 111=valence
(3-17)	Atom number
(18-35)	Value of abnormal mass, abnormal valence, or signed charge. The sign of a signed charge is indicated by bit 18.

A word of zeros follows the last abnormality word.

(c) The MATRIX "cleaned" for processing by MAKECT and containing only nodes and bonds.

CLEANM → Table look on node code

```
i = 1 for single letter atom
i = 2 "  double      "     "
i = 3 "  H
i = 4 "  Ph
i = 5 "  - (C)_n -
i = 6 "  - (CH)_n -
```

α_i

β_4 β_5 → Make entry in abnormality table → Erase abnormal character and complete connection to atom

α_1 → Make sign of SCRUB word minus

β_1 β_2 β_3 → Add to i counter

Assign C.T. number

Write AT ← Y ← Greater than normal valence ? ← Y ← Finished ? → N → Search for non-blank

N → SETUP

```
j = 1 for single bond
j = 2 "  double bond
j = 3 "  triple bond
j = 4 "  number
j = 5 "  charge
```

Table lookup on non-blank

β_j

Figure 25. Macro Flow Chart - CLEANM

Figure 25. Macro Flow Chart – CLEANM (continued)

95

2.2.12   Generation Of The Connection Table

Code Name: MAKECT

Programmer: Bruce Hack

John Powers

Abstract: This program assumes a MATRIX of nodes and connecting lines. A list is generated for each node indicating the type of node (element type), all associated points, and the connecting line types (bonds). This program also indicates if the atom is to multiplied in order to be correctly compared with the molecular formula during chemical verification.

2.2.12.1   Program Description

A macro flow chart describing this program is provided in Figure 26.

The SCRUB list contains each atom, bond, or other typed symbol in the structural formula and its relative MATRIX location. All atoms have been indicated by a previous program (CLEANM) (i.e., capital letters in the SCRUB list are now minus). The MATRIX now contains only atoms and bonds. The program proceeds by first finding a minus sign in SCRUB and then calculating the position of this in core(i.e., the absolute MATRIX location).

Search is then made in the eight locations around the atom until a bond is found. The bond type is noted and the bond is followed until a change in the symbol type occurs. This new character is classified as one of the following cases and the appropriate action is taken:

(a)   Another atom - place the proper entry in the Connection Table.

(b)   A blank - check to see if it is a bond corner. If so, continue path; if not assume that an unknown attachment has been found and place an entry in the Connection Table.

(c)   A reduced carbon chain - make a Connection Table entry for a carbon at this location.

(d)   A hydrogen atom - if it has been assigned a Connection Table number it is entered into the Connection Table. Otherwise, it is ignored.

A check is made to see if the search around the initial atom is complete, and that all bonds have been followed. If so, the next atom is located in the SCRUB list and the process continues. If not, the next bond is found, and the bond is followed as above. In the case of a carbon chain a check is made to see that the bond attachments are unambiguous. At the completion, all multiplier pointers are inserted in the Connection Table.

2.2.12.2   Program Structure

MAKECT is a main program that takes as input the following:

SCRUB - list of all typed characters in the structural formula and their relative MATRIX locations.

MATRIX - now containing only nodes and bonds.

AXTMOD - the absolute address of the first character of the SFI in the MATRIX is stored in the address of this location by ORGNZR.

MAKECT rejects chemical records for the following reasons:

(a)   Error in typing structural formula

(b)   Illegal symbol in structural formula

(c)   Bond in wrong place

(d)   Typed symbols are too close for unambiguous analysis

(e)   Non-straight attachments to carbon chains

Output from MAKECT is CT- the internal Connection Table whose entries are formatted as follows:

3 ←————————————→ 17	18-20	21-23	24 ←→ 29	30 ←→ 35
Number of atom bonded to	Mul-ti-pli-er	Bond type	2nd letter of atom	1st letter of atom

The multiplier points to an entry in the list of bracket multipliers (MULTAB, described in Section 2.2.7.2) where applicable. Each atom has 8 such entries only the first of which contains the atom name. The second and third contain the relative matrix location of this atom as follows:

3 ←————————→ 17	18-20	21-23	24 ←→ 27	28 ←————→ 35
SAME AS ABOVE			0000	

Bond type

Word 2 for a given atom: 3 Low order digits of relative matrix location

3 ←————————→ 17	18-20	21-23	24 ←————→ 29	30 ←→ 35
SAME AS ABOVE			000000	

Bond type

Word 3 for a given atom: 2 high order digits of relative matrix location.

ex. relative matrix location 14321 is represented

	28 - 35
	321

	30 - 35
	14
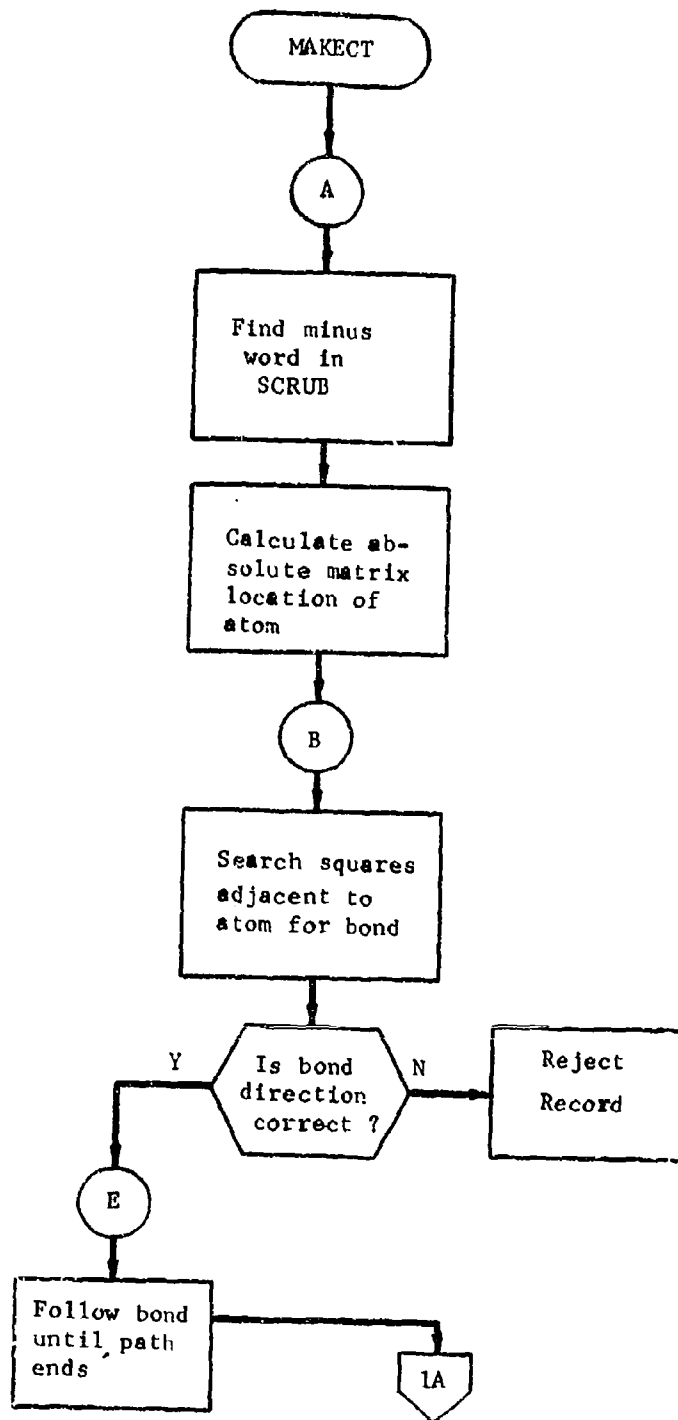
Figure 26. Macro Flow Chart — MAKECT
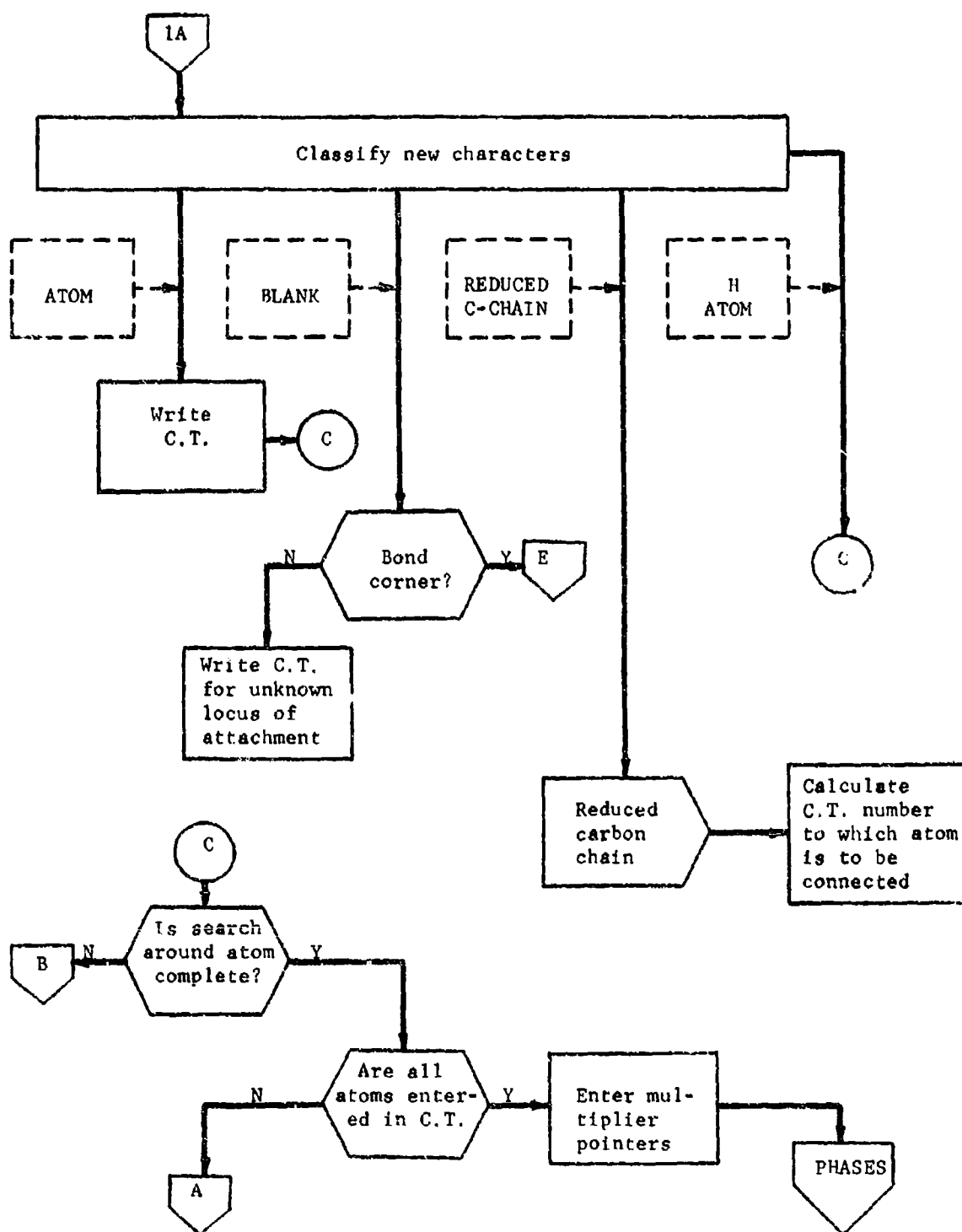
Figure 26. Macro Flow Chart — MAKECT (continued)

2.2.13   Calling Program For Chemical Verification

        Code Name: PHASE5

        Programmer: Bruce Hack

        Abstract: This is the call program for VERIFY.
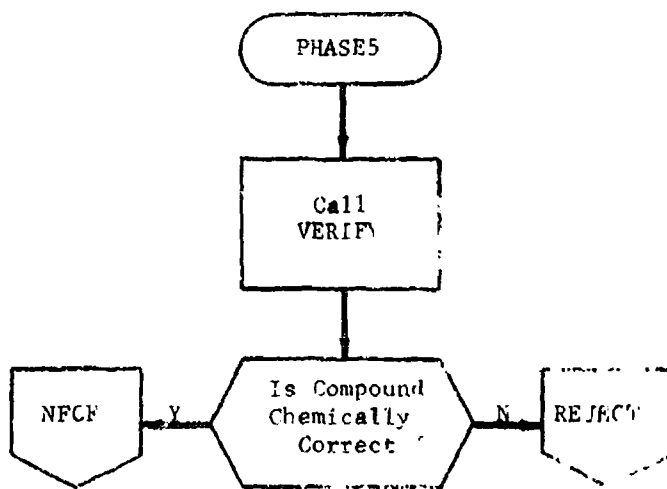
2.2.13.1 Program Description

    If compound is found to be correct by verification, this program transfers
to NFCF. Otherwise, an error exit is taken and control is transferred to REJECT.

2.2.13.2 Program Structure

    PHASE5 is a program that serves as a switch.

2.2.14 Chemical Verification

Code Name: VERIFY

Programmer: Helen Hill

Abstract: VERIFY checks the chemical consistency of the structural formula, molecular formula, and connection table, and verifies the valence of each element in the connection table and in the abnormality table.

2.2.14.1   Program Description

A macro flow chart describing this program is presented in Figure 27.

VERIFY utilizes a table (Table A) as a table containing all elements and the acceptable valences for each element. This table indicates whether an element is in odd or even group of the periodic table. A portion of the table is used as a counter for storing element totals during processing. VERIFY proceeds as follows:

VERIFY first takes each atom in connection table and totals all the explicit bonds. It then looks in the abnormality table to determine if an abnormality exists for this atom. If a valence abnormality is present, VERIFY uses the abnormal valence as the valence for this atom and checks to see if it is a legitimate valence for this element. If no valence abnormality is present, the minimum valence for this element (which is found in bits 15-23 of Table A) is used.

If a charge abnormality exists, the charge is subtracted from the hydrogen count. The hydrogen count for a molecular structure is equal to (all explicit hydrogens plus (valence of each atom minus total explicit attachments to this atom minus total charges in abnormality table for this atom minus total unknown attachments from this atom)).

The program translates the atom type from Dura Code to BCD. It then looks for this atom type in Table A and adds one to the tabulation in Table A of the total number of occurrences of this element in the connection table. Hydrogens are accumulated in HCTR rather then in Table A. If any hydrogens are found in Table A, it indicates the illegal presence of hydrogen in the connection table and the Chemical record is rejected. If there is a multiplier for the atom being processed, it is used to increase the atom count of this element by the actual number of occurrences.

When all entries in the Connection Table have been processed, the program checks to see if ION is set, indicating the presence of atoms not included in CT. If ION is set, the program then uses subroutine IONIC to add these atoms which are formatted in XTAB, to the totals in Table A.

Next VERIFY totals the number of atoms of each element present in the Hill molform and places them in Table A. C, H, N and O are totaled in MFC, MFH, MFN, and MFO. Multipliers of the Hill molform found in case of a Hydrate are

102

applied in totaling the Hill parent, but the water is ignored. If the Hill
parent multiplier is a fraction it results in rejection of the compound.

The program then compares the Hill molform totals with the CT plus XTAB
totals and adds up the elements in the odd group to be used in the H parity count.

If ADDEND is found to be set, VERIFY totals the addend molform atoms using
any multipliers present and compares the total count for each element in the
addend molform with the totals in the Hill molform.

VERIFY next totals the minus and plus charges found attached to elements
in the Connection Table, and compares the totals to see that plus charges equal
minus charges.

Finally VERIFY performs the Hydrogen Parity test on the Hill molform.

Error conditions which result in rejection of the compound are the following:

(1) An illegal element was found in a molecular formula.

(2) An illegal element was found in Connection Table.

(3) An element in the Connection Table has high valence which is
     incorrect for this element.

(4) The molecular formula contains a fraction.

(5) Addends are present but the first multiplier is zero.

(6) Hydrogen is present in the Connection Table.

(7) The assumed hydrogen count differs from the molform hydrogen
     count.

(8) There was a hydrogen parity check error.

(9) Connection Table C,H,N, or O count different from the molform
     C,H,N, or O count.

(10) An illegal valence was found in the Connection Table.

(11) The C,H,N, or O count in Hill molform differs from that in the
      addend molform.

(12) Totals for elements other than C,H,N, or O are not the same in
      the Hill and addend molforms.

(13) An illegal element was found outside of brackets.

(14) Connection Table total for C,H,N, or O is not equal to the Hill
      molform total for the same element .

(15) The multiplier of the Hill molform in a hydrate is a fraction.

103

(16) Plus and minus charges do not balance.

2.2.14.2   Program Structure

VERIFY is a subroutine which occupies 983 core locations. Table A of elements and valences is 102 locations long. In addition there is a table which relates modified Dura Mach code to BCD.

VERIFY uses the following input:

CONNECTION TABLE   described in Section 2.2.12.2.

ABNORMALITY TABLE   described in Section 2.2.11.2.

PL ------------ total plus charges outside of CT

MIN ---------- --- total minus charges outside of CT

XPOINT -------- pointer to end of XTAB

ION ----------- set if atoms exist outside of CT

MULT ---------- pointer set if multipliers exist

MULTAB -------- table of multipliers from REGRUP described in Section
                2.2.7.2

XTAB ---------- table of atoms outside CT from EXCESS described in
               Section 2.2.8.2.

MOLTAB -------- formatted molform from MOLFRM described in Section
               2.2.4.2.

On output, VERIFY sets bits in the accumulator to indicate the type of error found if the compound was rejected.

104
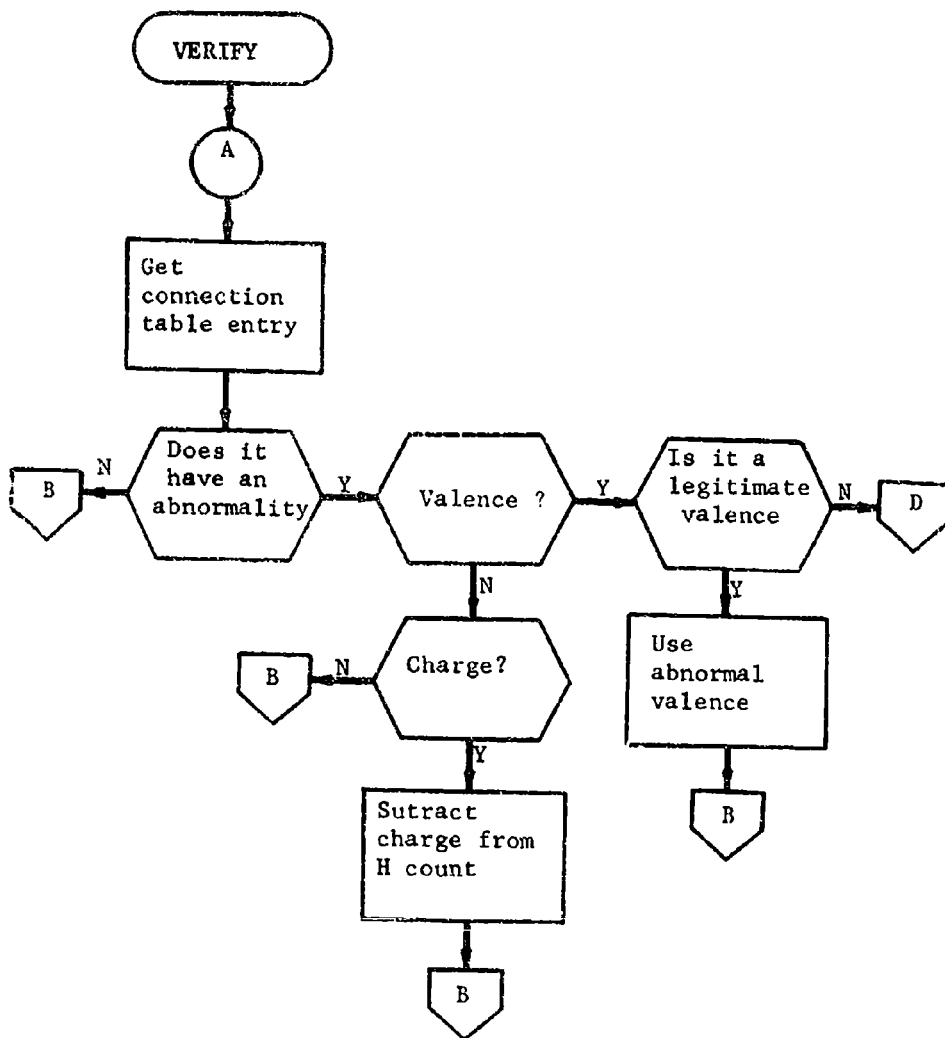
Figure 27. Macro Flow Chart - VERIFY
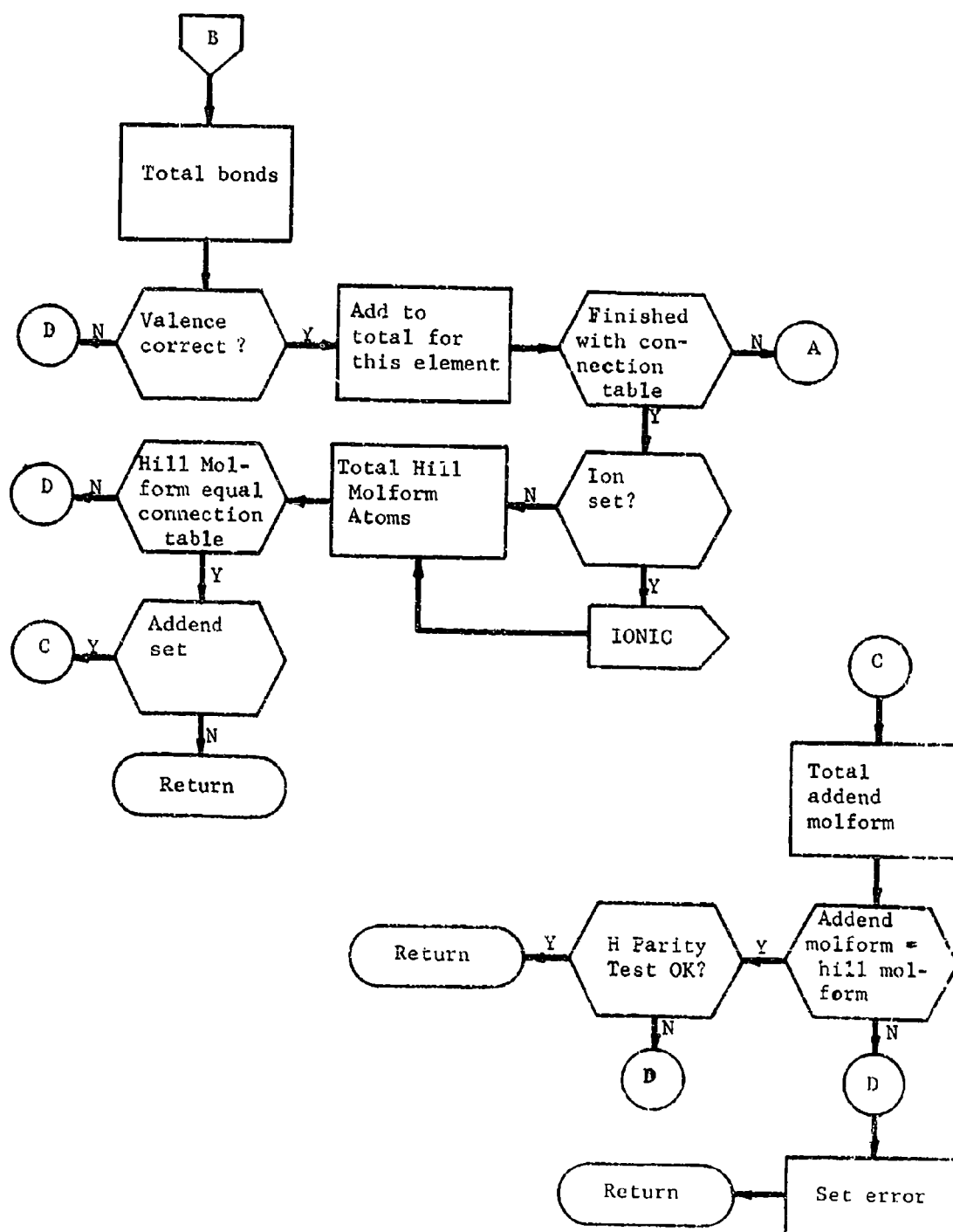
Figure 27. Macro Flow Chart – VERIFY (continued)

106

2.2.15   Expansion Of The Connection Table

Code Name: NFCF

Programmer: Bruce Hack

Nick Homer

Abstract: This program expands the connection table from the internal format to the format acceptable by program CONVRT and will print the connection table and abnormality table if a switch is set.

2.2.15.1 Program Description

A macro flow chart describing this program is presented in Figure 28.

The connection table list is broken down into three lists:

(1) The E list - this contains the atom name.

(2) The B list - this contains the bond type.

(3) The X list - this contains the number of the connected atoms.

The format of these lists is described in Section 2.1.2.2.

If a switch is set the lists are printed by the line printer under appropriate headings and, the abnormality table is decoded and printed. The program then calls CONVRT and TICKER.

2.2.15.2   Program Structure

NFCF is a main program which takes the connection table described in Section 2.2.12.2 as input and provides as output three expanded lists:

X - The connection list

B - The bond list

E - The atom name list

NFCF rejects chemical records for the following reasons:

(a) Empty connection table

(b) Incorrect element symbol in bits 24-35 of CT

NFCF calls DECK A, a subroutine, which prints Connection Table titles when a switch is set.

107
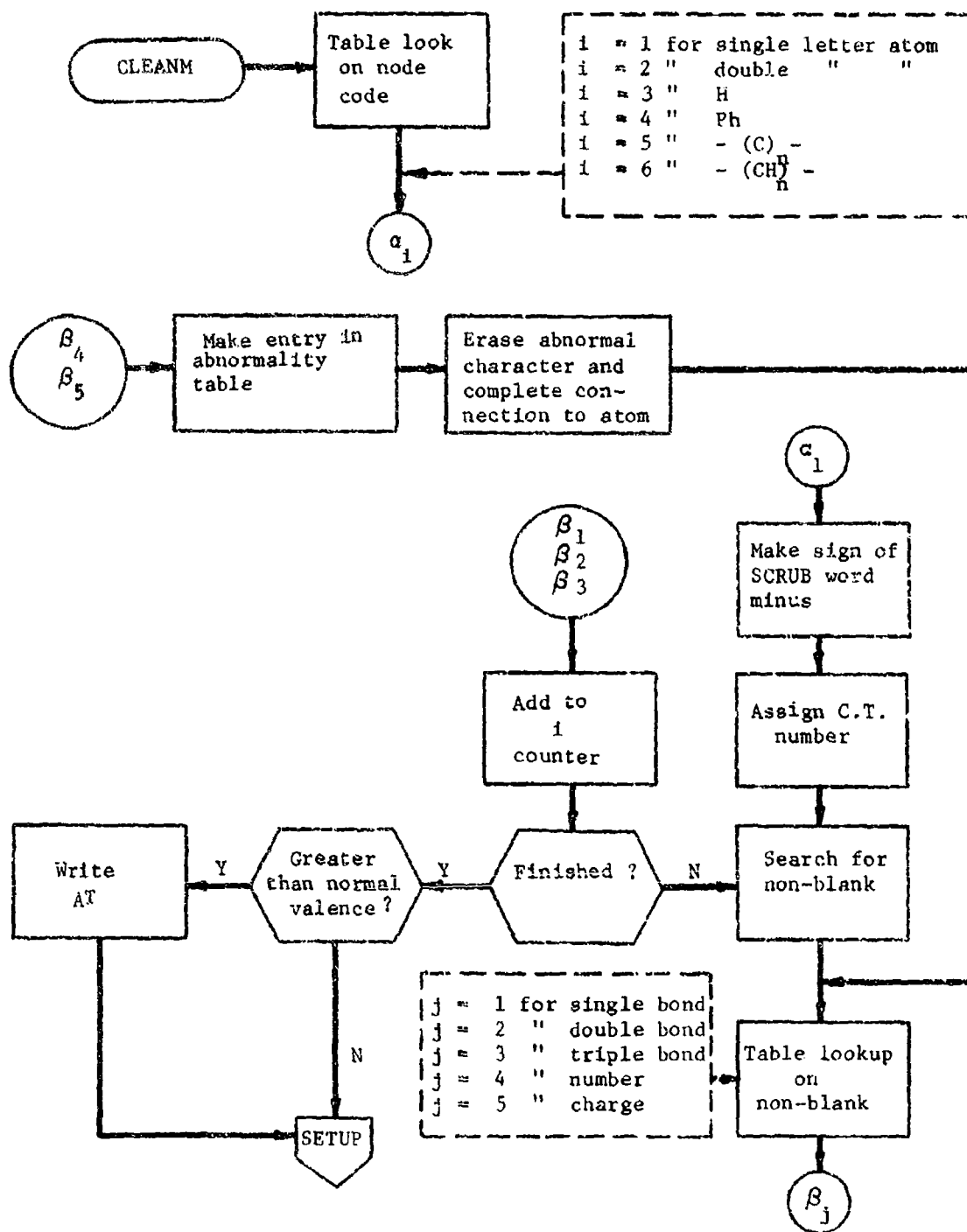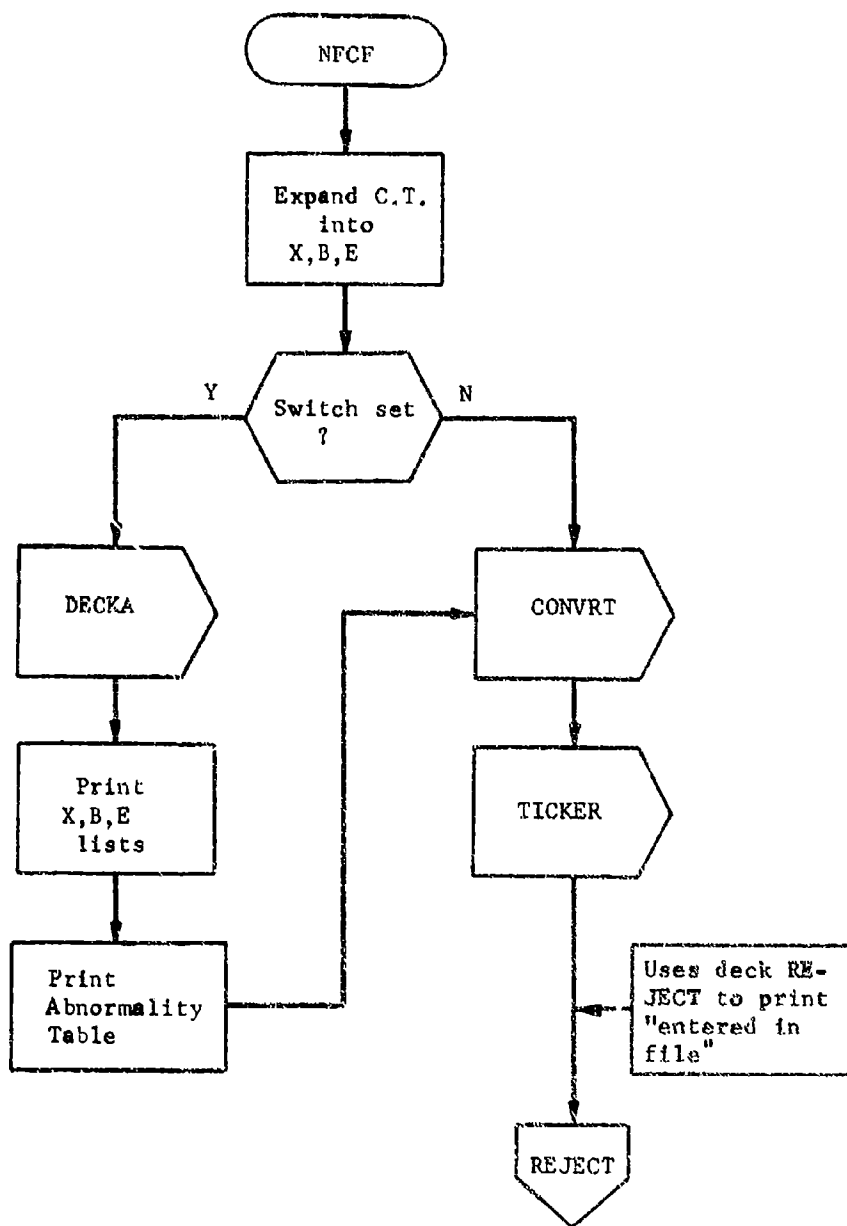
Figure 28. Macro Flow Chart - NFCF

2.2.16 Output Of Chemical Record

Code Name: TICKER

Programmer: Helen Hill

Abstract: TICKER writes an output tape containing the TID, classification and stereo information, molform, nomenclature and references, structural formula image, connection table, and abnormality table.

2.2.16.1   Program Description

A macro flow chart describing this program is presented in Figure 29.

TICKER uses a portion of the MATRIX into which to transmit all the information required for the output and writes this block onto magnetic tape, writing one record per chemical compound. It uses information provided by MOLFRM, VERIFY, and CONVRT to calculate the number of rings in the compound and stores this in the output record. It then writes the total number of parity errors encountered since the last compound if the input was from a Mergenthaler typewriter. If switch 2 is set, TICKER calls PIX or DURPIX to produce pictures.

2.2.16.2   Program Structure

TICKER is a subroutine which requires 142 core locations and utilizes the MATRIX area as an output buffer.

It requires the following input:

MOLTAB -   the molecular formula described in Section 2.2.4.2

NOMTAB -   the nomenclature described in Section 2.2.5.2

REGNO -   the registry number

CLSTER -   formatted classification and stereo information described in Section 2.2.3.2

CONTOT -   the number words in the connection table

UNDTAB -   the underline table described in Section 2.2.3.2

SCRUB -   the SFI described in Section 2.2.3.2

CELLB -   connection table described in Section 2.1.2

ATIR -   the number of words in the abnormality table

ASCRUB -   a pointer to end of the scrub list

ADTOT -   number of addend fragments in the compound

DELX -   x size of matrix

DELY - y size of matrix

RINGS - used to calculate total rings

The output from TICKER is the formatted chemical record described in Figure 13. This is on magnetic tape, one chemical record per physical record.
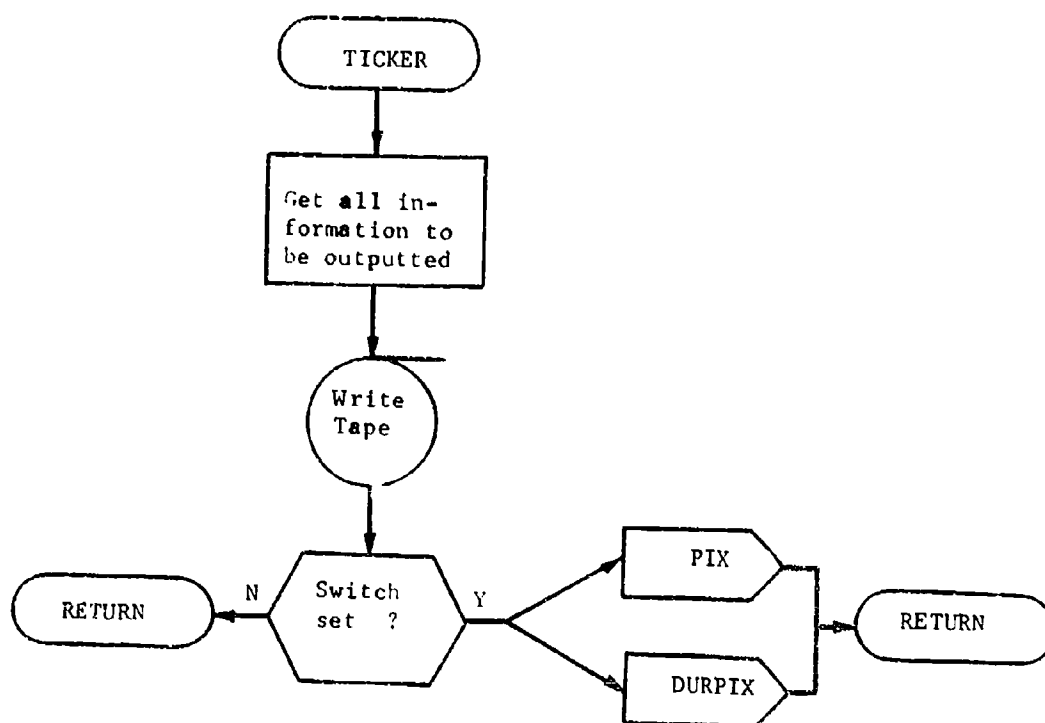
Figure 29. Macro Flow Chart - TICKER

116

2.2.17 Rejection Of Incorrect Records

Code Name: REJECT

Programmer: John Powers

Bruce Hack

Abstract: This program is transferred to from various portions of the system. A message is printed out and the program transfers to AEND.

2.2.17.1 Program Description

The following macro flow chart describes this program.
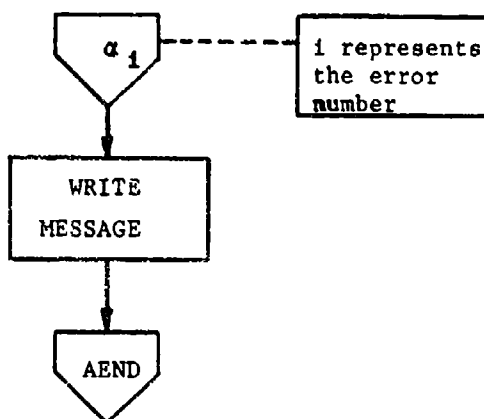
NOTE: AEND transfers the program from the generalized processing portion of the system to one of the input programs for the processing of the next chemical record.

2.2.17.2 Program Structure

REJECT is a subroutine.

112

2.2.18   CHEMTYPE To CIDS Format Conversion

Code Name: UPTAP

Programmer: Ed Hebel

Abstract: This program reformats the output of the CHEMTYPE system into CIDS format and merges into the record descriptors which were introduced through punched cards.

2.2.18.1   Program Description

UPTAP passes through the following phases: sort, format, card read, print TID's (temporary identification), and sort.

The first phase of UPTAP sorts on the two word field TID for each logical record created by the CHEMTYPE programs. At the start of the program a check is made to see if any record read exceeds 1,000 words. Should a record exceed 1000 words, it is skipped and its TID is printed.

The next phase inserts one to four words from the Molform Table at the beginning of each record. Word five is the base address for all relative addresses to be calculated. The actual word count of the entire CIDS record with the exception of the four MF words, and the two's complement relative addresses in the record for Additional Registry Numbers, Abnormality Table, Compound Connection Table, Reference Block, Structural Formula Image, Keys, and Qualifiers are calculated and placed in words 5,6,7,8. (See Section 2.1.3.2).

Within the Reference Block, a header and a table of contents is created. The decrement of the header holds the count for the table of content words. The address portion contains the number of words in the Reference Block including the header. The decrement in each table of contents word contains the identification number for the type of information. In this portion of the Reference Block, Bits 18-20 indicate the type of data: BCD, BINARY, Modified DURA, or Compressed Modified Dura. The address portion of each word has the address relative to the header of the Reference Block. The table of contents shown in Section 2.1.3.2 allows the Reference Block to be expanded without difficulty in the future.

The card routine reads a card and compares its TID against the TID of the current record. If the record TID proves to be larger than the card TID an error message is generated stating that fact. The next card is read and the same test applied.

If more than one card is present with the same TID a check is made to ascertain that the cards are in contiguous sequence. If they are not, an error message is generated. Whenever a card is encountered with a new TID, the descriptor on the card for the previous TID will be transferred into the current record. Prior to transfer, a check is made to see whether the new TID on card is greater than the TID of the descriptor being stored. Should the new TID prove to be less, an error message results.

The TID of all records processed are saved on an output tape called 'TIDNR'.

115

After formatting the CIDS records, UPTAP prints the number of input and output records processed.

In the final phase of the program a sort is performed on two fields: 1) the four Molform Table words, and, 2) the two word TTD. The sorted records are written on the final output tape for input to the registry system.

REFERENCE BLOCK FORMAT

WORD	S 2 3 17	18 20	21 35
		HEADER	
0	No. of words in table of contents		No. of words in reference block (including Word 0)
1	CLSTER ** 1	1*	+ RA to Header of Ref Blk
2	Nomenclature ** 2	3*	+ RA to Header of Ref Blk
3	EA No.(type 01) ** 3	0*	+ RA to Header of Ref Blk
4	CLSTER		
5	Nomenclature		
.			
.			
x	EA Number (S)		

NOTE:

* Type of Data

   0 BCD
   1 Binary
   2 Modified Dura
   3 Compressed Modified Dura

** If Decrement is zero
   no data is stored

2. 18.2 Program Structure

UPTAP is a main program.
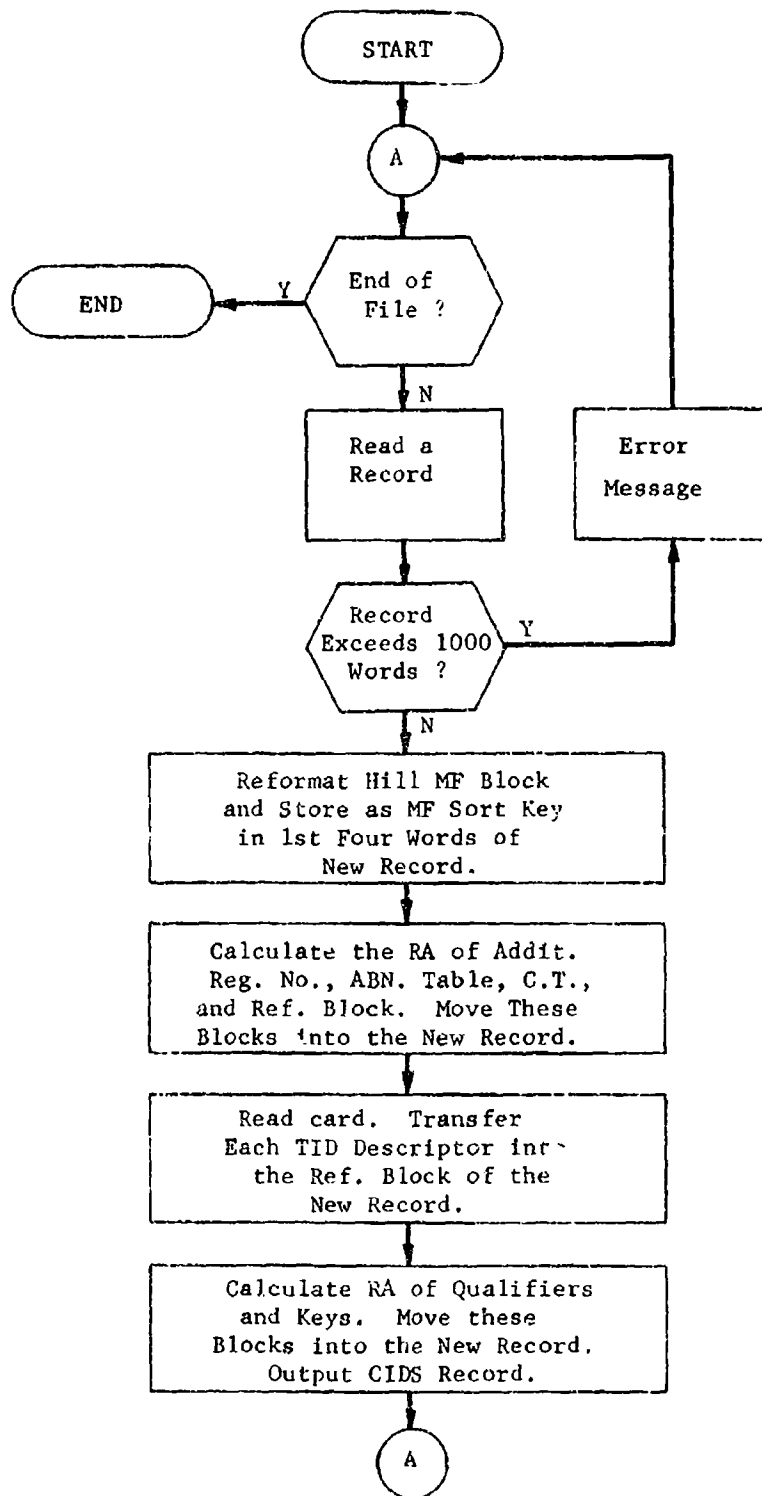
Figure 30. Macro Flow Chart - UPTAP

2.3 REGISTRATION

The CIDS Registry System examines compound records which are candidates for the file in order to determine which compounds are duplicates of those already in the file. In order to locate duplicates, a Master Registry File is maintained in molecular formula sequence. Potential new compounds are sorted in this same order and compared against the Master Registry File. An atom-by-atom search is performed to compare the structure of each potential registrant with each of its isomers (compounds with identical molecular formulas) in the file. If a connection table match is found, a further test is made to see if the complete records are exact duplicates.

Figure 31 shows the general flow of the registration process. The four principal programs STARTA, HLDPRC, REGUD, and RUD II are described in the following sections. The principal files involved are the Master Registry file, the Print file, a d the Structure file. All utilize the CIDS record format as described in Section 2.1.3.2, but the blocks of data actually stored differs between files.

The Master Registry file contains:

> Registry number
>
> Additional compound identification numbers
>
> Molecular formula
>
> Connection table and abnormality table
>
> Reference block

The items in the reference block are listed in Section 1.1.

The Print file is an auxiliary file to the search system. It is maintained separately because the data it contains is not searched, but merely accessed for printing after the answers to a query have been determined. Thus the data it contains does not have to be maintained on the same high speed storage devices as the remainder of the search file. An additional reason for keeping this data separate is that it is likely to be updated, while the rest of the search data remains static. This separation considerably shortens the update process. The items contained in each record on the Print file are:

> Registry number
>
> Additional compound identification numbers
>
> Molecular formula
>
> Structural formula image
>
> Reference block

The Structure file is the input to the key assignment programs described in Section 2.4 and illustrated in Figure 3. The data blocks contained in this
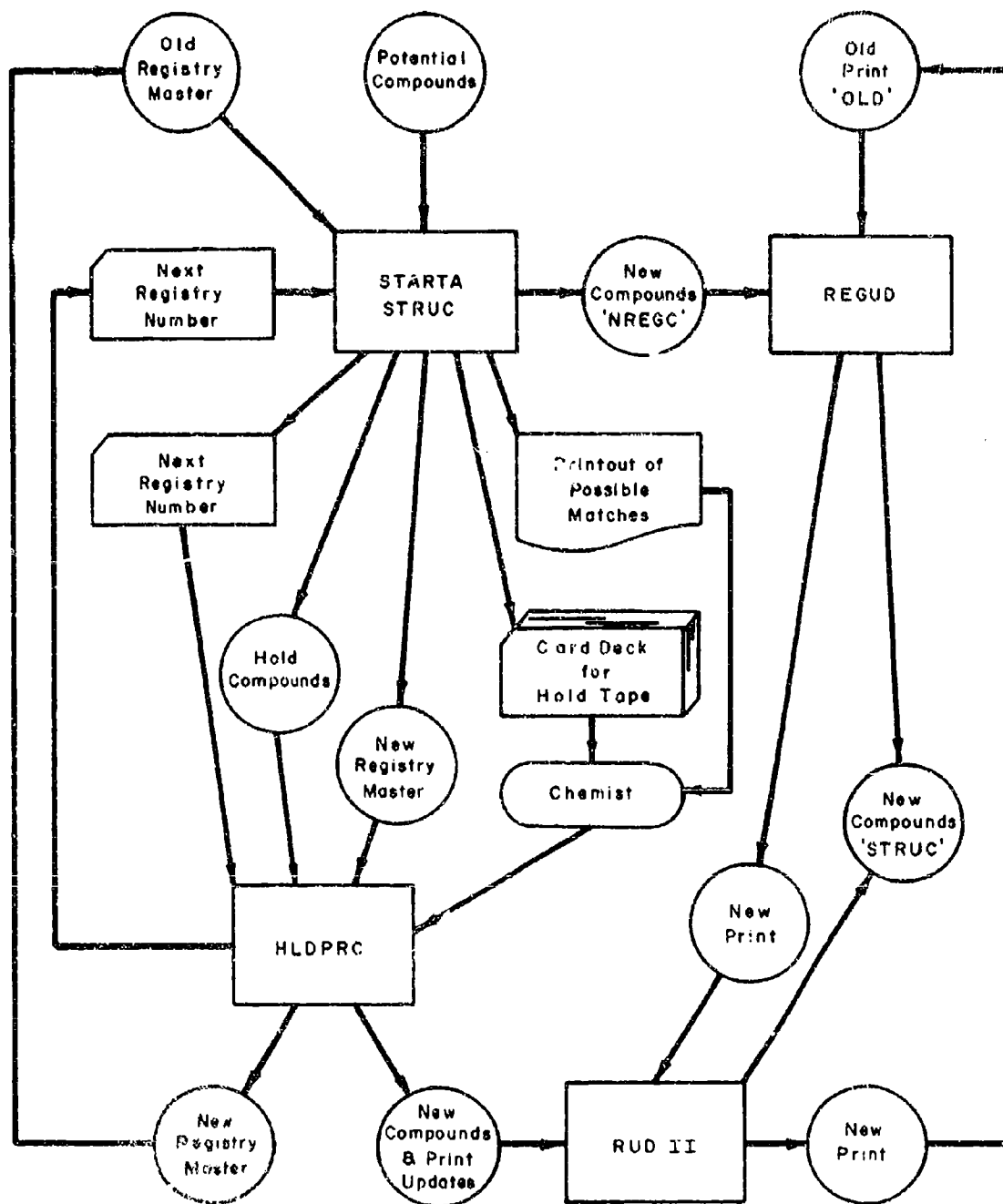
Figure 31.   Registry System

117

file are:

Registry number

Molecular formula

Connection table and abnormality table

Structural Keys,

The structural keys will be removed from the compound records before they are entered into the search file.

Potential registrants to the file are processed in one of the following ways by program STARTA:

(1) Those which have no connection table matches in the file are immediately registered.

(2) Those whose complete records exactly match some compound record on the Master Registry File are discarded.

(3) Those whose connection table matches one or more file compound connection tables, but do not have a complete record match are saved for further inspection by a chemist.

Those compounds which have initially been determined to be unique by program STARTA, can be immediately prepared for entry into the file by program REGUD which adds Print information from these records to the Print tape and outputs a structure tape for input to the functional group key assignment programs.

Those compounds in category (3) above are written on a Hold tape by STARTA and must be examined by a chemist before further action can be taken. He must determine for each compound match whether the two are different compounds (in which case they must be stereoisomers), or whether they are the same compound, but have some different data in the compound record. This difference may occur because of an error in one of the two records, or one may contain additional data which the other does not.

Program HLDPRC processes those compounds on the Hold tape based on the decisions of the chemist. These compounds are processed in one of the following ways:

(1) Those that are different (stereoisomers) of all their isomers in the file are registered as new compounds.

(2) Those that are the same compound are

(a) Ignored if the data already in the file is more complete and more correct than that on the Hold tape.

(b) Selected parts of the data record are used to update the file record if the data on the Hold tape is more complete or more correct.

118

Code Name: STARTA

Programmer: Donald Headley

Abstract: STARTA determines which of a group of potential new compounds
are different from those already registered in the master file. These com-
pounds are registered, positive matches are discarded, and questionable matches
are printed for further examination by a chemist.

2.3.1.1 Program Description

STARTA reads a tape of potential registrants which was produced by program
UPTAP (Section 2.2.17). These are sorted according to the 4-word molecular
formula sort key which precedes the normal CIDS record This tape is then com-
pared against the Master Registry File which is also in MF sequence. Program
STRUC (Section 2.4.8) is called to determine if two connection tables match when-
ever a molecular formula match is found.

If the connection table of the candidate record does not match any records
on the Registry Master File, a unique registry number is assigned to the com-
pound, and it is written on the new Registry Master and on the new compound file
NREGC. This tape will then be processed by program REGUD.

If the connection table of the candidate compound matches that of one or
more file compounds, a further test is made to determine if the entire record
is the same. This means that the temporary identification number (TID) of the
candidate record must match one of the additional registry numbers of the file
compound. As a further check, the stereo indicator and nomenclature must ex-
actly match. If these data fields are all identical, the candidate compound
is considered a duplicate and is discarded.

If there was a connection table match but one of the other fields failed
to match, then that data from the matching compound records must be printed
for examination by a chemist. The compound record for a potential registrant
which falls in this category is stored on the Hold tape for later processing
by program HLDPRC (Section 2.3.2). A card is punched containing the TID of
each compound on the Hold tape. After a decision is made by a chemist, an ac-
tion code must be punched on each of these cards indicating the type of process-
ing to be performed on each.

2.3.1.2 Program Structure

STARTA is a main program which requires subroutine STRUC. The inputs for
the program are:

(1) 'INPUT1': Potential new compounds from program UPTAP.

(2) 'MAST2A': The old Registry Master File. For the initial
     run, a parameter card specifies that the file is not present.

119

(3) A card with the next registry number to be assigned in columns 25-36.

The outputs of the program are:

(1) 'MAST2B': The new Registry Master File

(2) 'HOLDTP': Hold Tape File. The potential registrants from 'INPUT1' that matched a record(s) in the Registry Master File

(3) 'NEWCMP': The new registered compounds.

(4) A listing on the printer of the records on the Hold tape with the matching records from the Registry Master.

(5) A punched card for each record on the Hold tape containing TID number and a card sequence number.

(6) A card with the next registry number to be assigned in columns 25-36.

All files have IOBS type 2 format, with a maximum block size of 1000 words.

2.3.1.3 Operating Instructions

For execution, tapes must be mounted as follows:

'INPUT1'	S.SU07 (B6)
'MAST2A'	S.SU04 (C4)
'MAST2B'	S.SU05 (B5)
'HOLDTP'	S.SU B4)
'NEWCMP'	S.S. JO (B3)
Scratch	S.SU02 (C2)
"	S.SU01 (C3)

For multi-reel operation, a message will print at the end of each input tape. Sense switch 2 must be set to signal the last input reel for file 'INPUT1'. Sense switch 3 must be set to signal the last reel for file 'MAST2A'.

120

2.3.2 <u>Hold Tape Processor</u>

    <u>Code Name</u>: HLDPRC

    <u>Programmer</u>: David Sherr

    <u>Abstract</u>: HLDPRC processes the Hold tape produced by program STARTA according to an action code punched on each card of the TID card deck produced by STARTA. The action codes are the results of a chemist's decision to register, ignore, or update the compound record for each compound on the Hold tape.

2.3.2.1 Program Description

    As each compound record is read from the Hold tape, the next card in the card input is checked to see if it contains the same TID. If it does, the compound record is processed according to the action code punched on the card. The present allowable action codes are:

        (1) Ignore record

        (2) Register as new compound

        (3) Replace nomenclature.

Action code 3 is presently the only type of update available. In this case the registry number of the compound whose nomenclature is being replaced is also punched on the TID cards. Other types of updates are in the process of being implemented.

    Compounds that are to be registered are assigned the next available registry number and the record is written in sequence on the new Registry Master and on 'NREGC', the tape of new compounds to be input to RUD II. Updates are also written on 'NREGC' after the compound record on the new Registry Master is updated.

    A macro flow chart of the program is presented in Figure 32.

2.3.2.2 Program Structure

    HLDPRC is a main program which requires as input:

        (1) 'HOLD': The Hold tape

        (2) 'OLMAS': The old Registry Master File

        (3) A card with the next registry number to be assigned in columns 25-36

        (4) The TID card deck from STARTA with action codes punched in columns 13-18 right-justified.

'21

Figure 32. Macro Flow Chart - HLDPRC

122
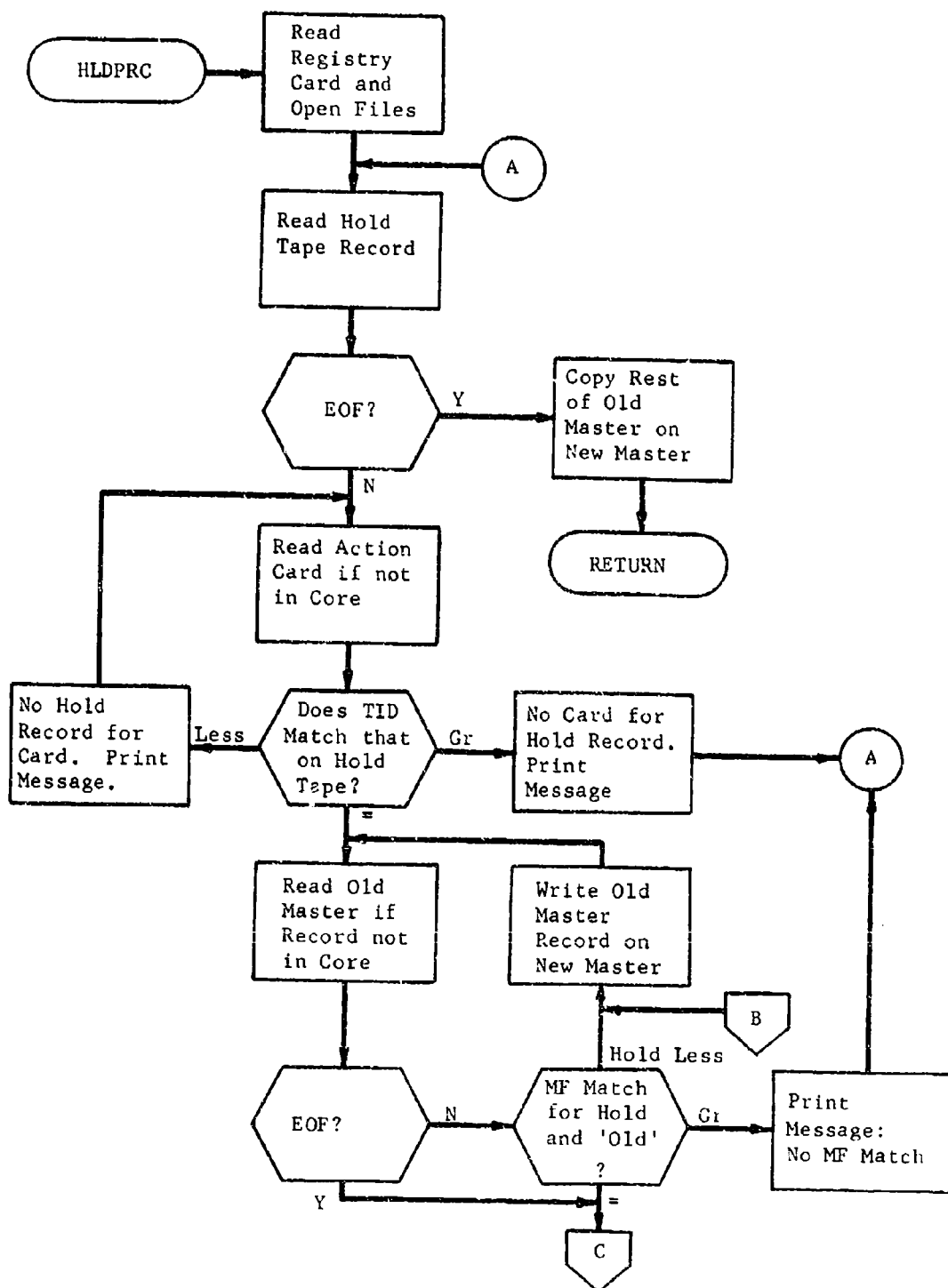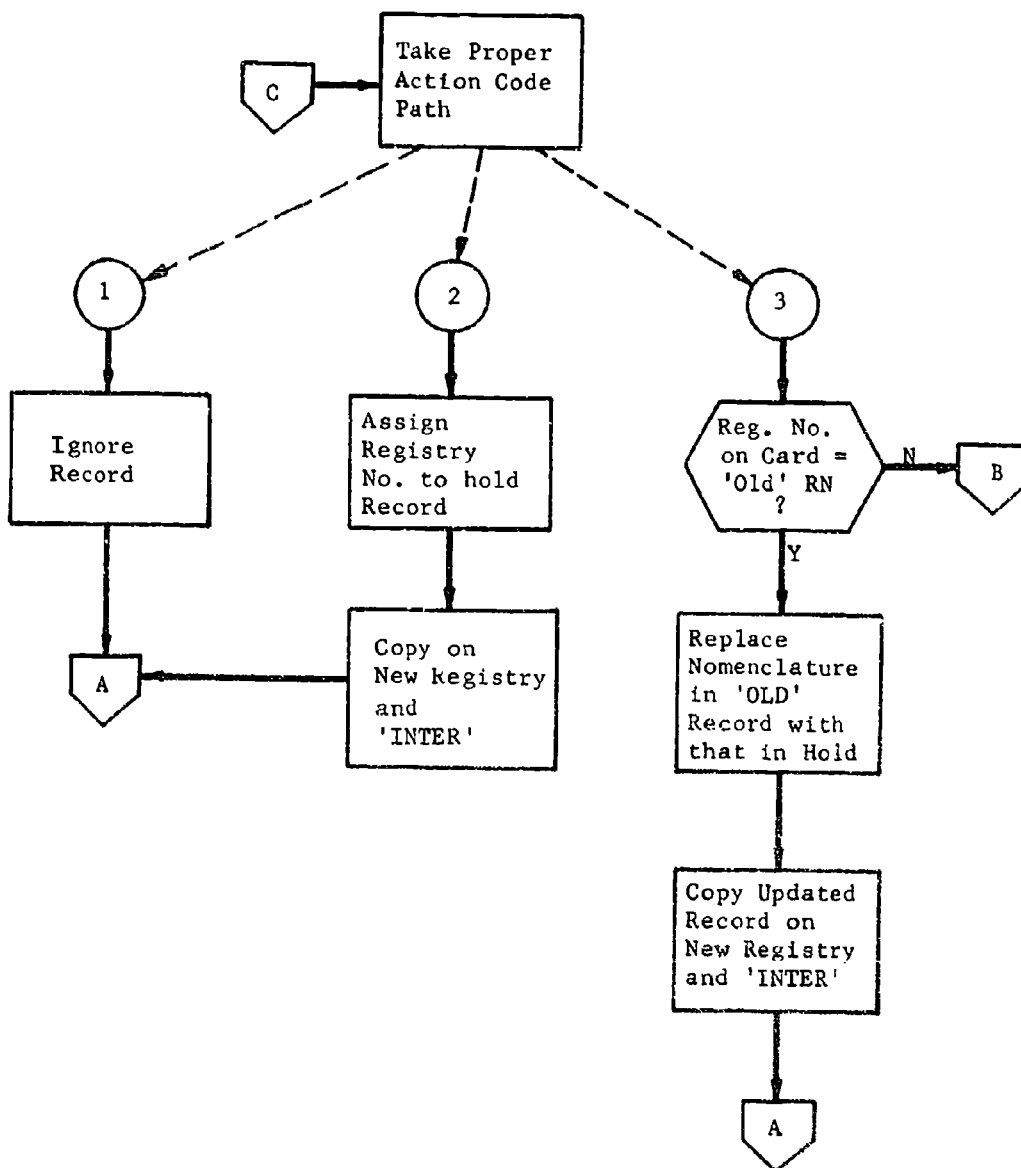
Figure 32. Macro Flow Chart - HLDPRC (Continued)

123

The outputs produced are:

(1) 'NWMAS': The new Registry Master File

(2) 'INTER': An intermediate tape for RUD II containing new compounds and updates.

(3) A card with the next registry number to be assigned in columns 25-36.

All files contain IOBS type 2 records in CIDS record format.

2.3.2.3 Operating Instructions

For execution, tapes must be mounted as follows:

'HOLD'	S.SU05
'OLMAS'	S.SU04
'NWMAS'	S.SU07
'INTER'	S.SU06

When the last reel of the 'OLMAS' file is mounted, sense switch 5 must be set. Similarly, sense switch 4 must be set when the last 'HOLD' reel is mounted.

The first input card must have the next registry number punched in columns 25-36 and must be followed by the action cards (output from STARTA). An end of data card with END punched in columns 1-3 must immediately follow these cards. The action cards are sequenced and must remain in that order.

124

2.3.3 Registry Print Tape Update

Code Name: REGUD

Programmer: David Sherr

Abstract: REGUD updates the Print tape by adding new records for a group of newly registered compounds.

2.3.3.1 Program Description

REGUD reads a tape of newly registered compounds. These records are then sorted by registry number in order to add new records to the Print tape which is in registry number sequence. The program reads the last old Print tape and checks that the last registry number is smaller than the first of the new records. If not, an error is indicated. A new Print tape is written with the new Print records added to the tape.

2.3.3.2 Program Structure

REGUD is a main program which requires as input:

> 'OLD': Old Print tape
> 'NREGC': Tape of new registered compounds
> Data card containing number of reels in the 'OLD' file.

The outputs produced are:

> 'PRNT': New Print tape
> 'STRUC': Structure tape
> Data card containing number of reels in the 'PRNT' file

All tapes are written in IOBS type 2 records and the logical records are in the CIDS record format described in Section 2.1.2.2.

The input tape 'NREGC' was produced by program STARTA (Section 2.3.1). The output tape 'STRUC' is the input to the key assignment programs described in Section 2.4.

2.3.3.3 Operating Instructions

The last reel of the 'OLD' file must be mounted on S.SU04 with SS5 set. An input card with the number of reels in the 'OLD' file must follow the $ENTRY card. The first 'NREGC' file tape must be mounted on S.SU05. Typewriter messages call for successive reels. When the last 'NREGC' tape is mounted, SS 4 must be set. When the program ends, it prints record counts for each file and punches a card with the number of reels in the 'PRNT' file.

2.3.4   Registry Print Tape Update II

Code Name: RUD II

Programmer: David Sherr

Abstract: RUD II updates the Print tape by adding new records for a group of newly registered compounds and updating records corresponding to previously registered compounds.

2.3.4.1   Program Description

RUD II reads a tape produced by program HLDPRC (Section 2.3.2) which contains newly registered compounds and Print data for addition to or replacement of Print records for previously registered compounds. These records are sorted by registry number for merger with the Print tape.

RUD II must pass the entire old Print tape in order to update or replace records which have been changed. When the end of the file is read, records corresponding to newly registered compounds (which automatically have larger registry numbers) are added to the file.

2.3.4.2   Program Structure

RUD II requires the same inputs as REGUD (Section 2.3.2), except that file 'NREGC', which is in this case produced by program HLDPRC (Section 2.3.1), contains both newly registered compounds and updates for previously registered compounds. The outputs produced are the same as those by REGUD.

2.3.4.3   Operating Instructions

The operating procedure is the same as for program REGUD except that all reels of the old Print tape must be read and updated.

2.4 KEY ASSIGNMENT

The Key Assignment System is broken into two subsystems, (1) the ring key assignment system and (2) the specific fragment and miscellaneous key assignment system. The programs are broken up into two groups because of the large core requirements of each.

The ring key assignment programs analyze the ring systems of a structure from its connection table and automatically assign the appropriate CIDS generic ring keys as described in CIDS No. 4. In addition, if the compound connection tables being processed do not have ring atoms and ring bonds explicitly marked, then these programs also perform this function. For this class of data, it is necessary to perform ring analysis before assigning specific fragment and acyclic keys. Otherwise, the order of processing is unimportant. The programs which make up the ring key assignment system are SCNCAS, SCRNCR, SCRNDR, RING1, RING2, RING3, and RING4. SCRNCR and SCRNDR are two versions of the same program, the first being used for data which previously had ring bonds and ring atoms explicitly marked and the second for data which does not yet contain these indicators.

The remainder of the key assignment programs are grouped together. The programs comprising this system are SCNCAS, SCREEN, STRUC, HCRCT, BONDCT, MFSRN, and PSCKYT. The executive program SCNCAS is the same for both systems. SCREEN is the sub-executive program which serves a similar function as SCRNCR and SCRNDR in the ring key assignment system. Each of the other subroutines assigns some particular type of key to a compound. Program STRUC is called to perform an atom-by-atom search to determine if a particular functional group or hydrocarbon radical fragment is present in a compound. Program HCRCT assigns a key when a compound is found to contain a nonspecific hydrocarbon radical as described in Section 2.4.9. Program BONDCT assigns the CIDS acyclic nuclei keys described in CIDS No. 4. Program MFSRN assigns molecular formula keys. Program PSCKYT assigns non-specific phosphorus functional group keys.

2.4.1 Key Assignment Executive

Code Name: SCNCAS

Programmer: Ruth V. Powers

Abstract: SCNCAS is the executive for the Key Assignment programs. For each compound on the input tape, the sub-executive program is called which in turn calls the appropriate screening subroutines. SCNCAS writes the compound record on tape in the same format with the newly assigned keys added to the record. The program has been written so as to provide flexibility in restarting the screening.

2.4.1.1 Program Description

SCNCAS first calls a subroutine which reads the fragment screens (output of SLOAD) if it is used with the fragment screening programs.

A data card is then read which provides the parameters for restarting the program. The first number of the card gives the registry number of the last compound which has already been processed from the input file. SCNCAS skips to this compound on the tape and begins processing with the following compound. If this first number is zero, processing begins with the first compound on the tape.

The second number on the data card gives the registry number of the last compound processed on the previous output tape if it is desired to add to it. This number is zero if a new output tape is to be started.

As each compound record is read from the input tape, pointers are set to the locations of the molecular formula and connection tables. One of several possible sub-executive programs is then called, depending on the type of key assignment to be done. When control is returned to SCNCAS, a test is made to see if the compound was successfully screened. If so, the keys are added to the record and it is rewritten on the output tape. If any type of error is encountered, or if the size of the compound exceeds some program limitation, it is written on a Reject Tape.

Processing is halted and all files closed when either an end-of-file is encountered on the input tape or sense switch 5 is pressed in at the console.

A macro flow chart of the program is presented in Figure 33.

2.4.1.2 Program Structure

SCNCAS is the main program of the Key Assignment system. It calls one of the subexecutive programs SCREEN, SCRNCR, or SCRNDR, depending on the phase of Key Assignment to be accomplished.

128
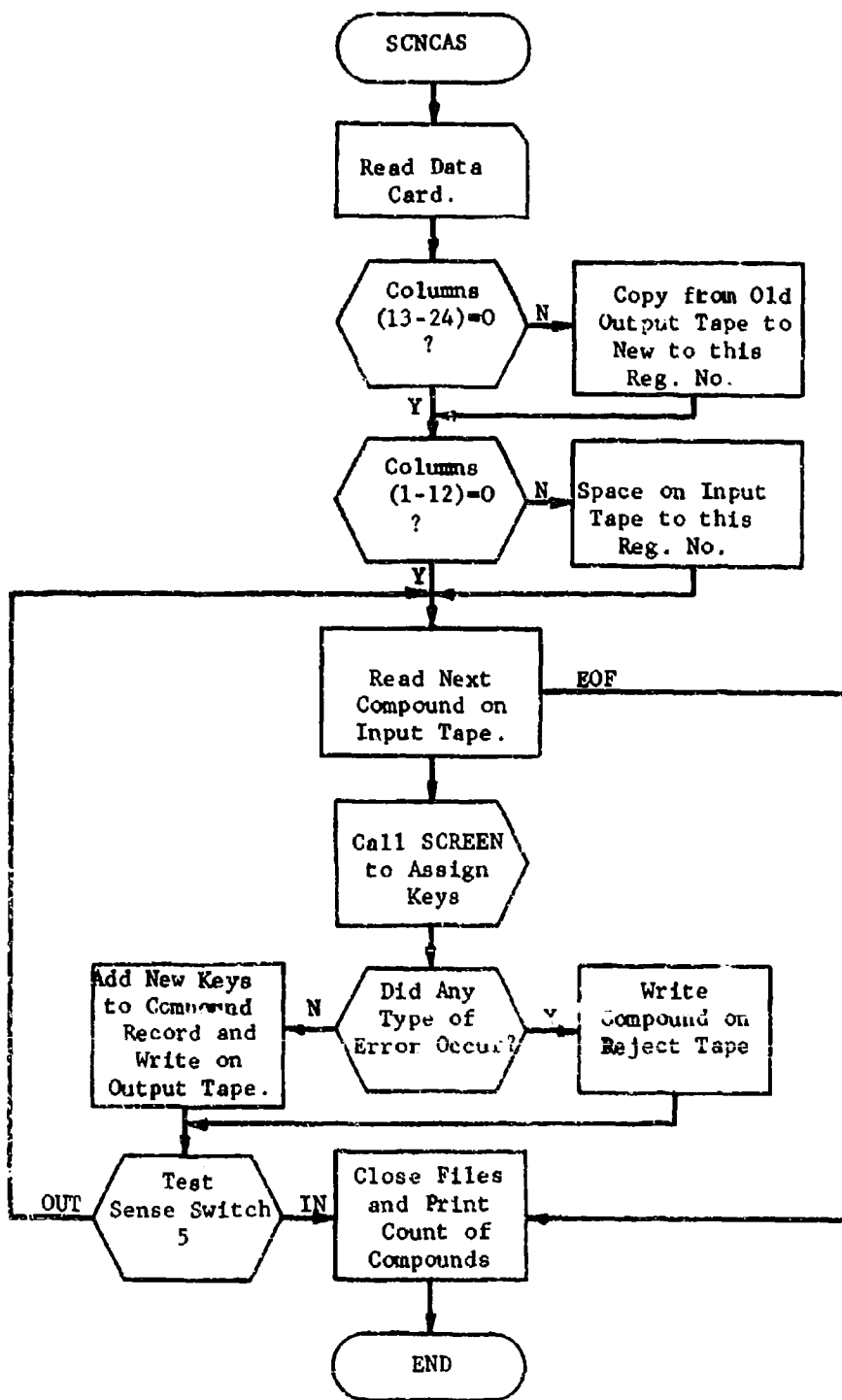
Figure 33. Macro Flow Chart - SCNCAS

129

The input to SCNCAS is the tape of compounds in the CIDS record format. This format is described in Section 2.1.3. In addition, for the assignment of functional group and hydrocarbon radical keys, the tape of screen fragments (output of SLOAD) must be loaded into core. The output of SCNCAS is a compound tape in the same format as the input tape, with any newly assigned keys added to the record.

SCNCAS causes the following messages to be typed for compounds that have been rejected:

        (1)   NO C.T. XXXXXXXXXXXX.
              Compound record contains no connection table because of some
              error in entering the compound in the file.

        (2)   XXXXXXXXXXXX REJECTED. BOND TABLE TOO LONG.
              Number of bonds in the compound exceeds a program limitation.

Further error messages are described in subroutine descriptions.

2.4.1.3 Operator Instructions

Tapes for the key assignment programs are mounted as follows:

Compound Input Tape	S.SU06
Fragment Screens (if needed)	S.SU05
Previous Output Tape (if needed)	S.SU04
Output Tape	S.SU07
Reject Tape	S.SU10

Sense switch 2 is pressed in to print the keys that have been assigned. (Otherwise only the registry numbers of the compounds processed are printed.) Pressing sense switch 5 in causes the program to halt processing and close all files. See subroutines for other switch settings.

130

<u>Key Assignment Sub-Executive</u>

     <u>Code Names</u>: SCREEN, SCRNCR, SCRNDR

     <u>Programmer</u>: Ruth V. Powers

     <u>Abstract</u>: This program serves as part of the Key Assignment Executive.
It is a subroutine of program SCNCAS and acts as an intermediary between it and
the various key assignment subroutines. SCREEN, the version used·when hydrocarbon
radical and functional group fragment keys are being assigned, selects the par-
ticular screen fragments which must be applied to the compound being screened.

2.4.2.1 Program Description

     Three versions exist for the Screen Assignment Sub-Executive program for
use with different data and different types of key assignment. All three are
called by program SCNCAS and serve the functions of initializing counts, print-
ing the registry number of the compound being processed, and transmitting to the
screening subroutines the address of the compound connection table and its length.

     Programs SCRNCR and SCRNDR are employed for the assignment of the generic
cyclic nuclei keys. They in turn call the ring analysis subroutines to assign
these keys. The only difference between these two versions is that in SCRNCR
a location CASWCH is set minus, and in SCRNDR it is set plus. SCRNDR is used
when the connection table data being processed does not yet have the ring bonds,
ring atoms, and resonant bonds marked. In this case, the ring analysis programs
perform this function. SCRNCR is used when CAS data is being processed, in
which case these marks have already been recorded. A flow chart for these two
programs are presented in Figure 34 .

     Program SCREEN is used when hydrocarbon radical and functional group frag-
ment keys are being assigned. It serves the function of selection of the par-
ticular screen fragments which must be applied to the compound being screened.
The program selects the fragments and points to each of them in turn while
calling the atom-by-atom search program (STRUC) to decide whether the fragment
is present in the compound.

     If the result of an atom-by-atom search is affirmative, STRUC is called
again to determine if the fragment is present in another part of the compound.
When STRUC locates a fragment in the file connection table (C.T.), the atoms
which correspond to those in the fragment are "erased" from the core buffer so
that they are not available as possible choices on the next attempt to locate
the fragment in the compound C.T. In this way the total number of occurrences
of the fragment can be determined. This "erasure" is carried through the en-
tire assignment of functional group keys. Since the order in which the frag-
ments will be tested for assignment is from largest to smallest, this means
that if one functional group appears within a larger functional group, only
the key corresponding to the larger one will be assigned.

     The selection of fragments which are potential keys for a particular com-
pound is based on the molecular formula of the compound. The first level of
discrimination is based on the kind of elements present in the compound. This
information is used to select the fragment groups which contain no elements
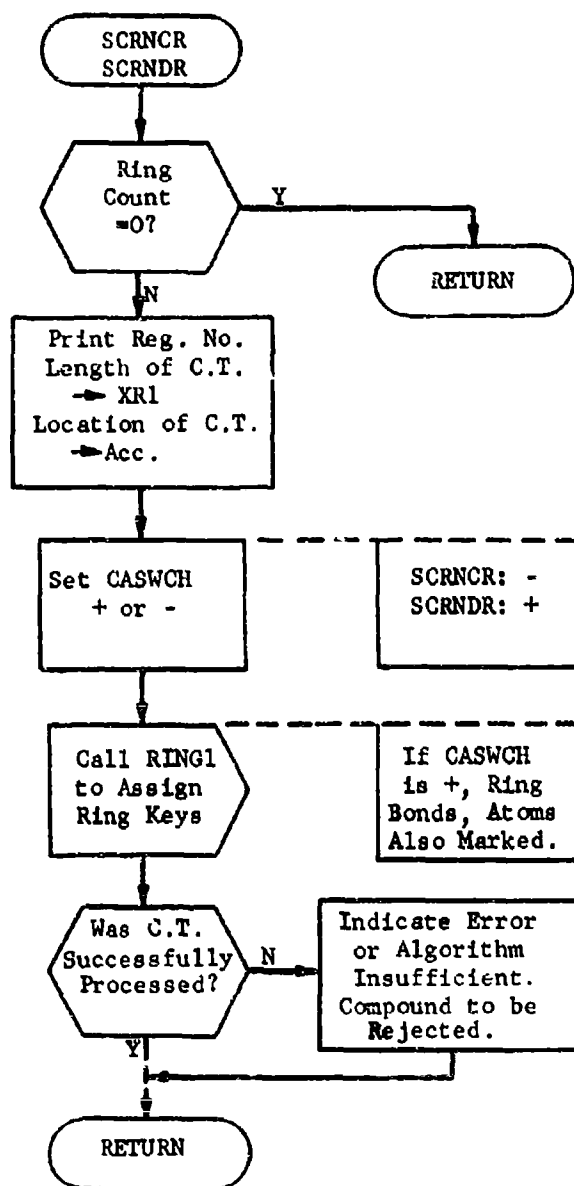
Figure 34. Macro Flow Chart - SCRNCR, SCRNDR

other than those present in the file compound. The CROSS table produced by program SLOAD (Section 2.4.3) is consulted to find those fragment groups which contain acceptable elements for a particular compound.

After a group has been selected, each fragment in the group is submitted to a molecular formula test which requires that the number of atoms of each element be less than or equal to the number of the corresponding element in the file compound. If this test is passed, an atom-by-atom search is then performed. If the fragment passes this search, it is assigned as a key to the compound, and the key number which represents it is stored in the key section of the compound record.

When the last screen of the candidate group is tested, SCREEN returns to its scan of the CROSS array to find the next candidate group. When the scan of CROSS has been completed, all appropriate fragment keys have been assigned.

SCREEN also calls program HCRCT (Section 2.4.9) to assign nonspecific hydrocarbon radical keys, program BONDCT (Section 2.4.10) to assign acyclic keys, program MFSRN (Section 2.4.11) to assign keys based on the Hill molecular formula, and program PSCKYT (Section 2.4.12) to assign nonspecific phosphorus functional group keys. Control is then returned to SCNCAS.

Figure 35 presents a macro flow chart of SCREEN.

2.4.2.2 Program Structure

This program is a subroutine of program SCNCAS in the Key Assignment system. It requires the following input data:

RINGCT--contains total ring count of compound

DOCNO--2 word array containing the registry number in BCD

DOCAD--decrement contains address of compound C.T.

COUNTS--contains number of words in the compound C.T.

The program prints the registry number of all compounds processed. In addition, whenever sense switch 2 is pressed in, the keys which were assigned to the compound are printed.

Figure 35. Macro Flow Chart - SCREEN

134
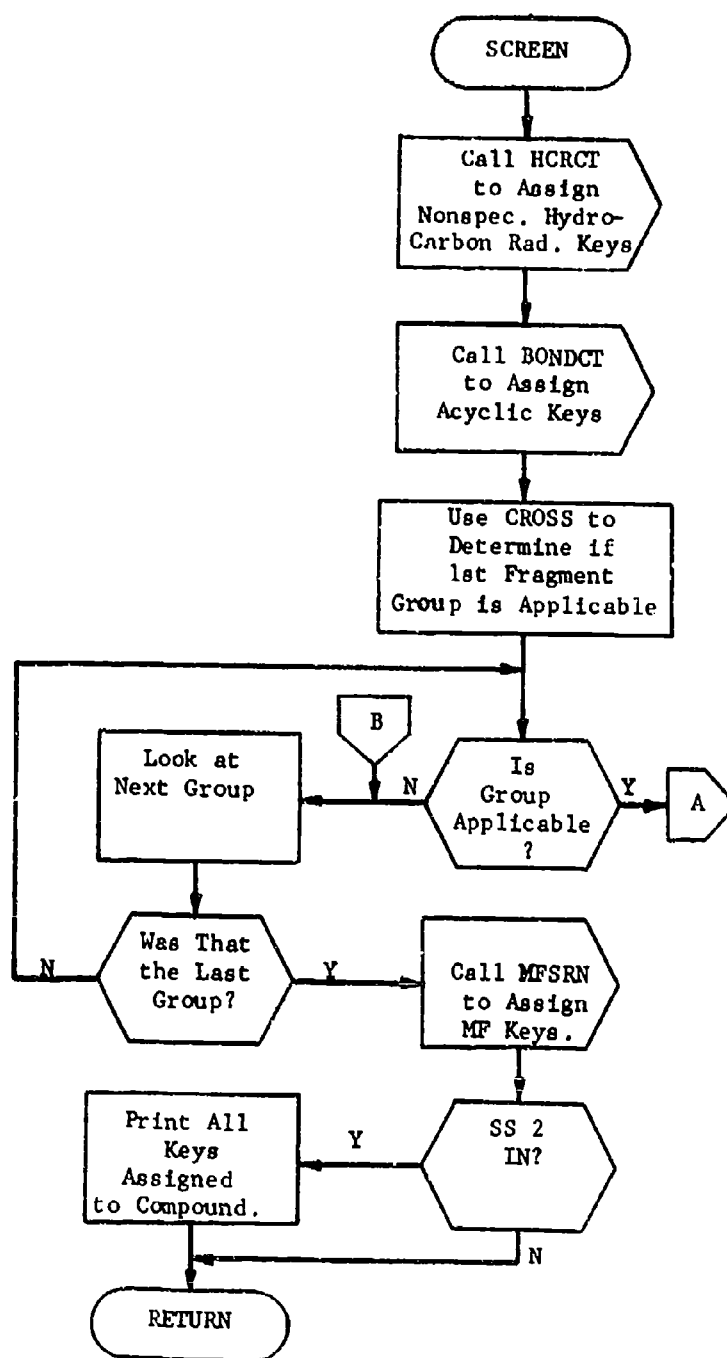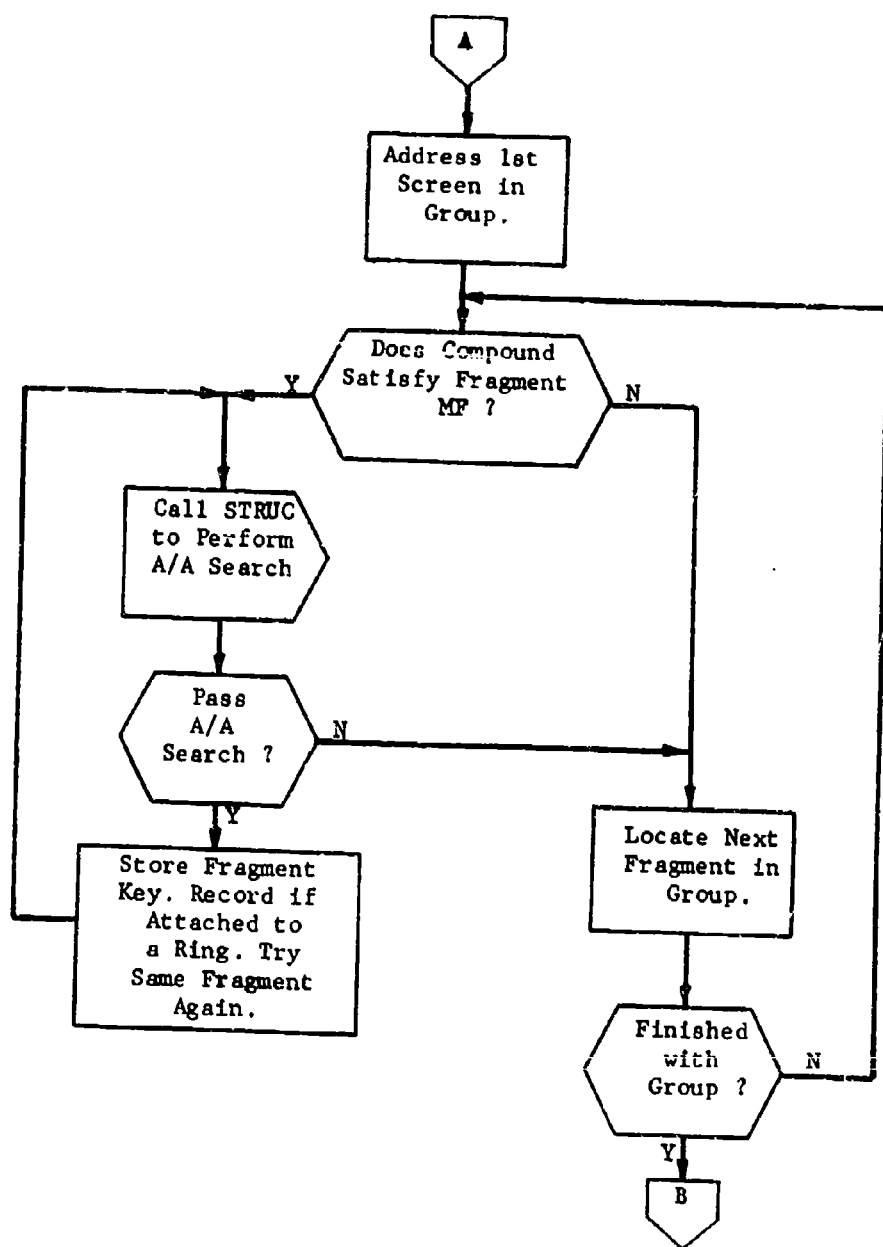
Figure 35. Macro Flow Chart - SCREEN (Continued)

135

2.4.3 Loading of Structural Fragment Screens

Code Name: SLOAD

Programmer: Ruth V. Powers

Abstract: Program SLOAD prepares structural fragment data for use by the screen assignment program. Connection table data defining the specific functional groups and hydrocarbon radicals presented in CIDS No. 4 are read from cards. The fragments are converted to the proper format and stored in groups based on the presence or absence of certain important elements. The fragment groups are written on tape together with an index to the location to each group.

2.4.3.1 Program Description

The cards containing fragment screens are divided into groups on the basis of the presence of certain important elements before they are loaded into the computer. Preceding each group is a title card which names the group. The name is composed of the combination of the following elements which are present in the fragments in the group: C, N, O, P, S, and X, where X is one of the halogens (F, Cl, Br, or I). Examples of titles are: CNO, CX, N. In addition, there will be one group composed of fragments which contain none of these elements. The groups are ordered so that those with the longest titles appear first. Within each group the fragments are ordered by size (number of atoms), with the larger fragments appearing first.

As the data is read from cards the molecular formula, connection table, and abnormalities associated with each fragment are converted to the CIDS internal formats and stored in an 8000 word buffer. Program CONVRT (Section 2.1.2) is called to format the connection table.

When another title card is encountered in the input data, the end of a group has been reached. A word of zeros followed by a word of binary ones is stored after the last fragment of the group in the output buffer. The next group is stored immediately following the word of ones in the buffer.

Whenever the title card of a group is read from the input file, the current buffer address is stored as the location of the beginning of that group. This information is stored as a two word entry in the index table (CROSS) in the format:

Word	Contents
1	Title word of group
2	Location of first screen of group relative to beginning of buffer

The title word indicates by bits in certain positions the contents of the block. The first six bit positions (S,1-5) indicates the presence or absence of C, N, O, P, S, and X in that order.

Screen groups can be separated into larger blocks (corresponding to different fragment types) to interrupt the erasure process and to cause the connection table atoms to be replaced which have been erased because of fragments

136
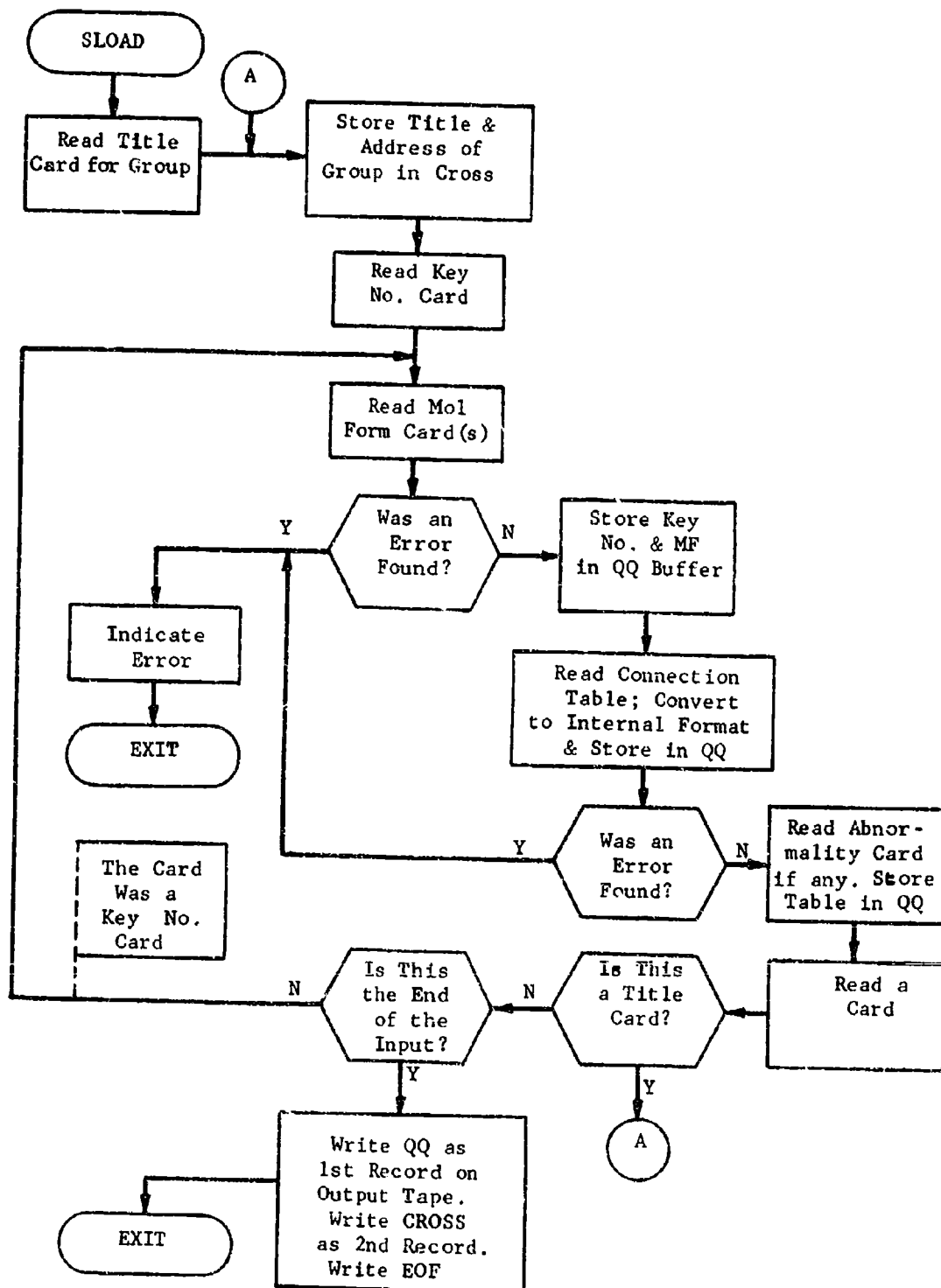
Figure 36. Macro Flow Chart - SLOAD

137

being assigned to it. New values can be given to ERASE at this time to alter the atoms to be erased. It can take on the following values:

> 0 - No erasure
>
> 1 - Erase non-carbons only
>
> 2 - Erase carbons only

This can be accomplished by placing a card with 'REPLAC' in columns 1 to 6 and '00X000' in 7 to 12 where X is the value to be assigned to ERASE. These cards are inserted before group title cards, and cause a two word entry in the CROSS array of the form:

> 777777777777
>
> 00000XYYYYYY

where X is the new value for ERASE and YYYYYY is the address of the previous screen group.

2.4.3.2 Program Structure

The data cards for the CIDS screens are ordered as follows:

> REPLAC002000
>
> C   (Title card)
>
> (Hydrocarbon Radicals)
>
> .
> .
>
> REPLAC001000
>
> CNOP   (Title card)
>
> (Functional Groups Containing C,N,O,P)
>
> CNOS
>
> (Functional Groups Containing C,N,O,S)
>
> .
> .
> .
>
> (Rest of Functional Groups)
>
> .
> .
>
> 000000   (End Card)

The functional groups must be assigned last so that the connection table retains the proper erasure for processing by PSCKYT and NONSPC for the assignment of non-specific functional groups.

The input for each structural fragment consists of data cards containing

138

the following information:

Key Number

Molecular Formula

Connection Table

Abnormalities (if any)

The key number is a 6 character number punched in the first 6 columns of the card. The formats for the mol form and C.T. cards can be found in CIDS No. 3 (pages 164 and 165) with the exception that column 24 of the first mol form card now indicates whether any abnormalities are present. Each abnormality has the form "XY=Z". Where X is the abnormality type, Y is the atom number, and Z is the value of the abnormality. The abnormality types are V (Valence), C (Charge), M (Mass). Examples: V1=5.C1=+1.M5=14.

The data for the next fragment follows immediately except when control cards are needed to separate groups or blocks. See discussion in Section 2.4.3.1.

The output of SLOAD is a tape on which the first physical record contains the fragment data. The format of the data associated with each fragment is stored as follows:

Word	Contents
1	D=No. of words in fragment record A=No. of words preceding structure (m)
2	D=No. of words preceding abnormality table (=0 if no abnormalities) A=No. of words in structure
3	Molecular formula
.	.
.	.
m	Key number
m+1	Connection table
.	.
.	.
.	.
n	Abnormality table (if needed) (a zero word follows last entry)

The internal format of the mol form is the same as that for a query mol form as described in Section 3.2.2.2. The internal format of the C.T. is given in Section 2.1.2.2.

The index is written as the second record on the output tape. The decrement of the first word of this record contains the complement of the number of words in the index. This word is to be read into a location CROSCT, immediately preceding CROSS when the tape is read for screen assignment.

When the program is run, an output tape must be loaded on S.SUO5. The data cards are loaded after the program and must be terminated by a card containing '000000' in the first six columns.

139

2.4.4   Ring Analysis Executive

Code Name: RING1

Programmer: Jeffrey H. Kulick

Abstract: The general function of RING1 is to find the smallest set of smallest cycles in a compound patterned after the rules of the Ring Index. These cycles are determined and the generic cyclic nuclei keys are assigned to the compound. The keys themselves are discussed in CIDS No. 4, and the algorithm for assigning them is discussed below.

2.4.4.1   Program Description

In the process of finding the smallest set of smallest rings (hereafter called SSSR), there are two bas processes, "compression", and "cycle finding". Compression involves a number of passes over the compound record. On each pass a rule is applied until it cannot be applied any further. Control then passes to a succeeding rule, which is again applied as many times as possible.

Compression-rule I is called "side-chain peeling". This involves the removal of any node with only one connection and its redundant entry. Three passes over a typical compound are shown below:

| Pass 0 | Pass 1 | Pass 2 | Pass 3 |
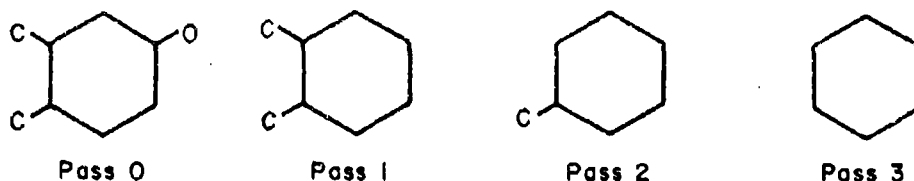
Compression-rule II is called "compression over two-nodes". This means simply that any node with exactly two connections is removed from the table and the two attached nodes are shown as having a longer chain connecting them. This is in addition to the standard CIDS compression over all carbon two-nodes. Typical final compression compounds look as follows:

INITIAL   FINAL

140
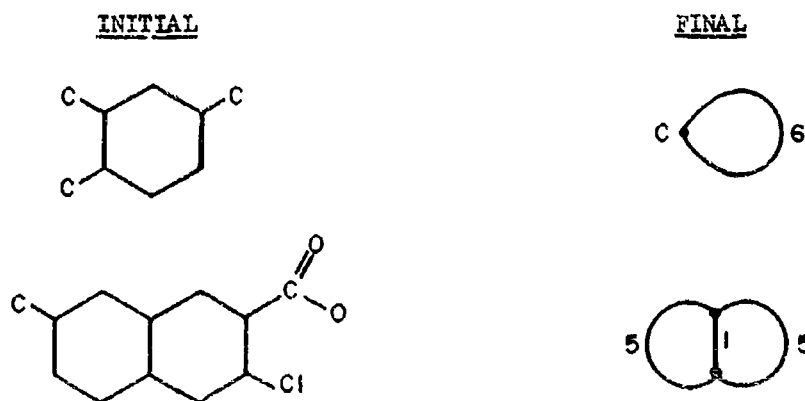
The second process that RING1 performs is to select the SSSR. The following rule is used:

(1) Find a path between two nodes.

(2) Is there a better path between these two nodes?

NO - Go to (1).

YES - Record in cycle storage, the cycle obtained by concatenating the path obtained in (1), and the **best** path obtained in (2). Then erase from the connection table the path obtained in (1) above. Go to (1).

This rule is repeated until all cycles of the structure are found or steps (1) and (2) fail to find a cycle in the compound record. If an insufficient number of cycles are found, the compound is rejected with the notation "ring algorithm insufficient." Otherwise processing proceeds with key assignment. Two examples of the selection of the SSSR is given below:

Consider the compound:
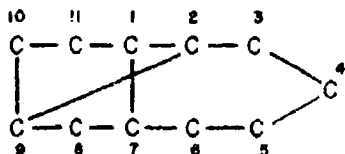
Ring Selection Process:

(1) Select path: 7-8-9

(2) Best alternate path: None

(3) Select path: 9-10-11-1

(4) Best alternate path: 9-2-1

* (5) Record ring: C_5

(6) Remove path: 9-10-11-1

(7) Select path: 2-3-4-5-6-7

(8) Best alternate path: 2-1-7

* (9) Record ring: C_7

(10) Remove path: 2-3-4-5-6-7.
The compound is now

141
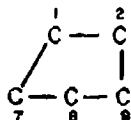
* (11) Record ring: C_5

The program would select rings C_5, C_5 and C_7 as the rings of the above compound.

Consider the compound:
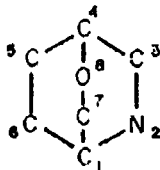
Ring Selection Process:

      (1) Select path: 1-2-3-4

      (2) Best alternate path: 1-6-5-4

* (3) Record ring: C_5N

      (4) Remove path: 1-2-3-4
          The compound is now:
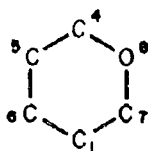
* (5) Record ring: C_5O

The program would select rings C_5N and C_5O for the above compound.

For these above processes, the major sub-programs used are RING2, which performs the alternate path search and RING3 and RING4, which perform the compression. In the alternate path search, one path is considered "better" than another if:

      (1)   It is shorter (contains fewer atoms), or

      (2)   The number of atoms is the same, but the "better" path

          (a) Contains more carbon atoms or

          (b) Contains the same number of carbon atoms but contains more atoms of the highest ranked heteroatom which appears in unequal amounts in the two paths. The precedence of heteroatoms is defined in the order: I, O, S, Se, Te, N, P, As, Sb, Bi, Si, Ge, Sn, Pb, Hg, B, all others.

For example, if a choice is to be made between a C_2OS path and a C_2OP path, the C_2OS path would be considered the "better" path.

The assignment of keys is now performed. First, ring storage is examined to obtain the ring molecular formulas recorded during the ring selection process. For each ring in ring storage, a key is generated, indicating the atom types composing the rings.

Next, the rings in ring storage are sorted according to size, the smallest to the largest. At this point, each ring is assigned a number of the

142

form 2^n (i.e. 1, 2, 4, 8, 16...).

The identification of the nuclei follows. This is done by partitioning the rings into equivalence classes. The classes are defined such that: Ring 1 is a member of the same equivalence class (nucleus) as Ring 2, if and only if there exists at least one atom A, such that A is in both Ring 1 and Ring 2. This is accomplished by intersecting the atom numbers of each ring with the atom numbers of all other rings. If they have an atom in common, they are noted as being in the same nucleus. When two rings are merged to form a nucleus, a notation is made as to which rings and which atoms, are in that nucleus. This process is continued until all intersections yield no further nuclei.

For each nucleus, the list of rings in that nucleus is used to obtain both the key for the number of rings in the nucleus and the redundant numerical ring population key (by going back to ring storage for each ring number and obtaining the size of the ring).

At the same time, for each nucleus, the atoms compositing it are obtained from the list made during the intersection of the rings. The original structure is searched to obtain the element type for each atom, and the skeleton molecular formula key is assigned.

The above processing is performed for each of the nuclei, keeping a count of the number of nuclei. From this, "the number of nuclei" key is generated. These counts are maintained for each addend separately.

The number of double bonds in each nucleus is now computed. In a large part of the data processed by the CIDS programs, resonant bonds have been marked as type 4 bonds, and these must be processed separately from the standard notation for double bonds (i.e. type 2). If no bonds in a structure have been marked resonant, then the number of double bonds in the cyclic part are merely counted. If a cyclic substructure of a structure was previously marked resonant (denoted by type 4 bonds), then the total number of double bonds in the resonant substructure must be computed before determining the number of double bonds in the non-resonant cyclic part. In the resonant part:

$$DB = \frac{RB - RC + 1}{2}$$

where DB is the number of double bonds in the resonant part, RB is the number of resonant bonds, and RC is the number of ring closures.

Consider the structur ·

143
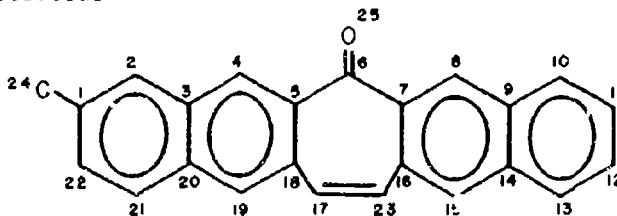
This structure is processed as follows·

(1) Start at node 6.
(2) It is connected to one atom not in the nucleus and so we now have 1 direct attachment.
(3) It is connected to two atoms in the nucleus, 5 and 7, which have not been processed. These are put in the "To Process" list giving (5, 7) and the "Processed" list becomes (6). The number of double bonds in the 6-5 and 6-7 connections are saved.
(4) Now process atom 5.
(5) It has a resonant connection so we put it into the Resonant list giving (4).
(6) It has another resonant connection, so we also put 18 in the Resonant list (4, 18).
(7) The total number of 4-bonds encountered is now (2).
(8) 5 is put into the "Processed" list, giving (6, 5) and the "To Process" list is (7).
(9) Next, process resonant atom 4.
(10) 5 has been processed so look at 3.
(11) 3 is not processed so add it to the "Resonant" list which is now (18, 3).
(12) Put 4 in the "Processed" list giving (6,5,4).
(13) The "Resonant" list is not empty so process 18.
(14) 18 is connected to 17 so add 17 to the "To Process" list giving (7, 17). Also add the number of double bonds to the D bond count.
(15) 18 is also connected to 19 so add 19 to the "Resonant" list (3, 19).
(16) Mark as processed 18, and the "Processed" list is now (6, 5, 4, 18).

The process is continued as follows:

Atom Being Processed	Processed	Resonant	To Process	4-bonds	D-bonds	Ring Closure
3	(4,5,6,18)	(19)	(7, 17)	(4)	(0)	(0)
19	(3,4,5,6,18)	(20, 2)	(7, 17)	(6)	(0)	(0)

Now 19 is processed, but 19 points to 18, which is processed. Therefore this is a ring closure. Add 1 to the ring closure count and continue.

Atom Being Processed	Processed	Resonant	To Process	4-bonds	D-bonds	Ring Closure
20	(3,4,5,6,18, 19)	(2)	(7, 17)	(7)	(0)	(1)
2	(3,4,5,6,18, 19,20)	(21)	(7, 17)	(8)	(0)	(1)
21	(2-6,18,19, 20)	(1)	(7, 17)	(9)	(0)	(1)
1	(2-6,18,19, 20,21)	(22)	(7, 17)	(10)	(0)	(1)

Again a ring closure has been found. Atom 1 connects to 2 which is processed, and to 22 which is in a table. In addition it connects to 24 which is not in this nucleus. Therefore the number of direct attachments is increased to 2.

Atom Being Processed	Processed	Resonant	To Process	4-bonds	D-bonds	Ring Closure
22	(1-6,18,19,20,21)	(0)	(7, 17)	(11)	(0)	(2)

The "Resonant" table has now been emptied. Before going back to the "To Process" table perform the computation:

$$(RB - RC + 1) / 2 = DB$$

$$(11 - 2 + 1) / 2 = 5 \text{ double bonds.}$$

7	(1-6, 18-22)	(0)	(17)	(0)	(5)	(0)
8	(1-7, 18-22)	(8, 16)	(17)	(2)	(5)	(0)
16	(1-8, 18-22)	(9)	(17)	(3)	(5)	(0)
9	(1-8,16,18-22)	(15)	(17, 23)	(4)	(5)	(0)
15	(1-9,16,18-22)	(14,10)	(17, 23)	(6)	(5)	(0)
14	(1-9,15-16,18-22)	(10)	(17, 23)	(7)	(5)	(1)
10	(1-9,14-16,18-22)	(13)	(17, 23)	(8)	(5)	(1)
13	(1-10,14-16,18-22)	(11)	(17, 23)	(9)	(5)	(1)
11	(1-10,13-16,18-22)	(12)	(17, 23)	(10)	(5)	(1)
12	(1-11,13-16,18-22)	(0)	(17, 23)	(11)	(5)	(2)

No more resonant bonds, therefore compute

$$(11 - 2 + 1) / 2 = 5 \text{ double bonds.}$$

17	(1-16,18-22)	(0)	(23)	(0)	(10)	(0)
23	(1-22)	(0)	(0)	(0)	(11)	(0)

Totals: 11 double bonds, 2 direct attachments.

This is the procedure determining the number of double bonds in a structure. The following basic assumptions have been made:

(1) There are no resonant spiro structures
(2) The number of double bonds in a resonant structure is $(RB - RC + 1) / 2$ as stated above.

At this point, the key for the total number of direct attachments to all nuclei is assigned.

During this computation, a note has been made of the number of bonds of each type appearing in each path; i.e. for the path between 16-17, one double bond was noted. Only those structures not marked resonant already (non-CAS data) will be marked resonant by the next process. To mark the resonant structure, ring storage is again consulted. The process is illustrated

by means of a typical example. Consider the compound:
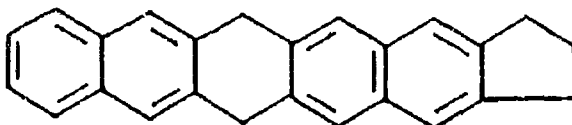
First, all rings with an odd number of atoms are removed. It has been assumed that any structure with an odd number of nodes <u>cannot</u> be resonant.
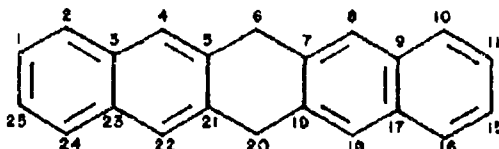
Next try to find a "handle". A handle is defined as a ring which meets the requirement of being resonant as defined by the rule that the number of double bonds be exactly one-half the number of nodes. Suppose that ring (3,4,5,21, 22,23) is chosen. It is determined that <u>by itself</u> it is not resonant (6 nodes - 2 double bonds). Now pick (1,2,3,23,24,25). This satisfies the criterion for resonance, and therefore is considered to be the first handle.

The next step is to find another ring connected to this ring. This is done similarly to the nucleus search. Each of the other rings in ring storage is intersected with the handle. It is found that (3,4,5,21,22,23) is connected to the handle. When (1,2,3,4,5,21,22,23,24,25) is tested for resonance, we find that this whole structure is resonant, and erase from storage the individual rings. This new structure is now the handle. Again, a search is made for a ring that is concatenated with the new structure, and (5,6,7,19,20,21) is found. It is found that (1,2,3,4,5,6,7,19,20,21,22,23,24,25) is <u>not</u> resonant. Not only is ring (5,6,7,19,20,21) rejected as being part of this resonant structure, but it <u>can never</u> be a member of any resonant structure and so it is erased from ring storage. No other rings can be found in common with the first resonant structure, so the program proceeds to mark the bonds of these rings as being resonant.

The same procedure is followed for the second part of the structure. It is found that the second part is also resonant and it is marked accordingly. Finally, the "marked-up" structure is copied into the output area.

2.4.4.2 Program Structure

RING1 is a subroutine of the Key Assignment System. It is called by a "TSL RING1". It in turn calls subroutines RING2, RING3, and RING4. The program requires the following input:

(1) CIDS formatted connection table

(2) Flag word CASWCH set plus for CAS data, otherwise minus

(3) Index Register 1 contains size of connection table

(4) Address of connection table right-adjusted in accumulator

146

The following output is produced by the program:

(1) CIDS formatted connection table. (If non-CAS data, ring bonds, ring atoms, and resonant bonds have been marked. The connection table for CAS data is left unchanged.)

(2) The keys assigned are stored, 2 words per key, starting at location, SCKY+1.

(3) BADAT = 0 if no error was detected in the connection table. BADAT≠0 if bad compound record.

In addition, if sense switch 2 is pressed in, the keys which were assigned are printed. If sense switch 3 is pressed in, an octal dump of the marked-up connection table is printed.

The following diagnostics are printed to signal conditions that cannot be handled by the program:

(1) "Over 3 non-specific types in ring". This means that a ring or nucleus contains more than three different element kinds which are other than C, N, O, P, S.

(2) "Too many rings per nuclei" This means a nucleus contains more than 17 rings.

(3) "Ring algorithm insufficient" This message appears when structures are too complex to be analyzed by the present ring algorithm.

(4) Error E1 - bad connection table.

A macro flow chart of the program is presented in Figure 37.

RING 1

Call RING4 — Assign Numbers to Compressed Carbons

Call RING3 to Perform Compression

Find a Candidate Path

Call RING2 to Find Alternate Path

Is Alternate Path Better?

Erase Candidate Path. Decrease Count of Rings to be Found

Any More Paths?

Enter Ring in Ring Storage

Indicate Ring Algorithm Insufficient

Assign Ring Mol Form Keys

More Rings to be Found

RETURN

Sort Ring Storage to Assign a No. to Each Ring

1

Figure 37. Macro Flow Chart - RING1
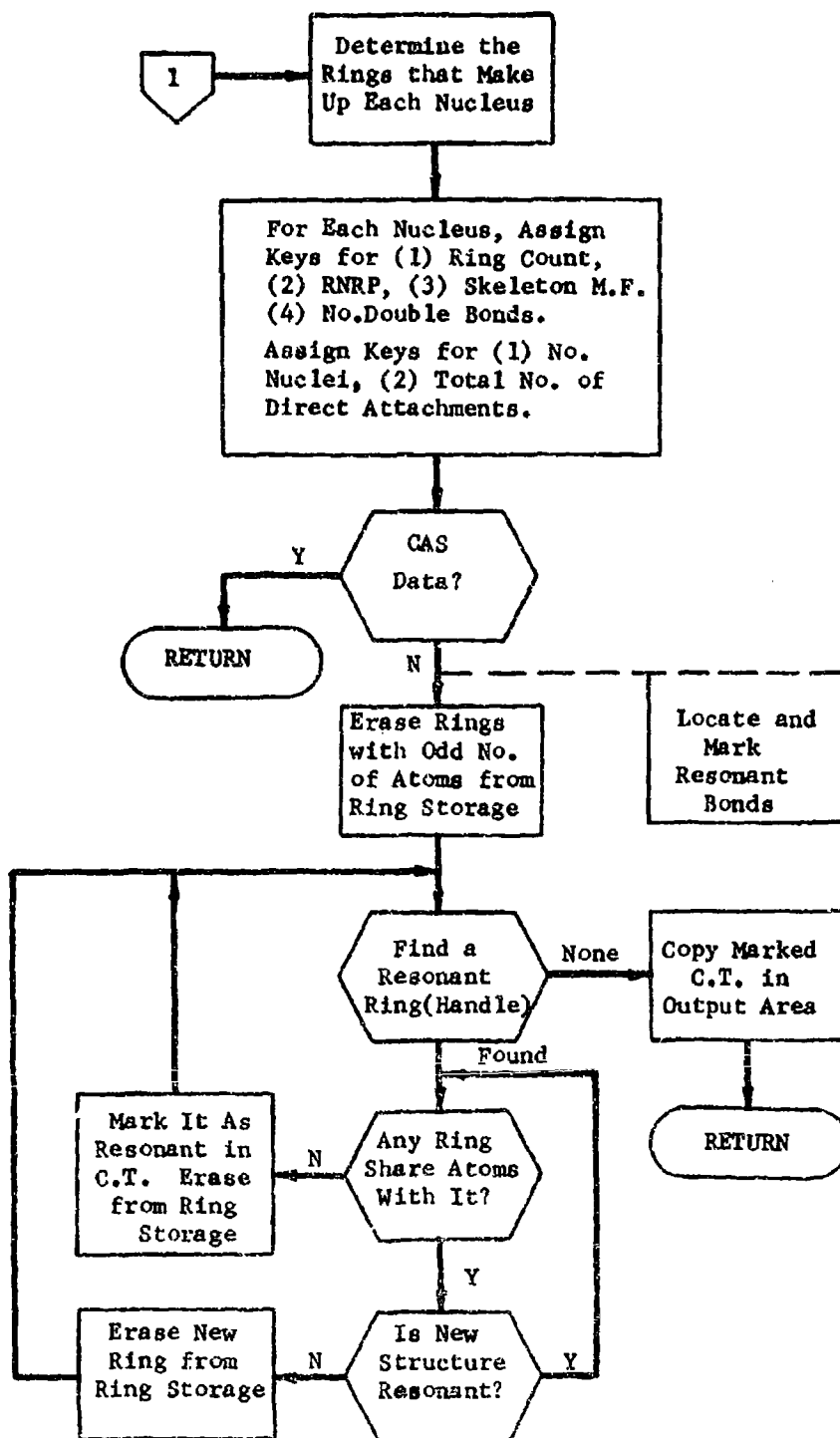
Figure 37. Macro Flow Chart - RING1 (Continued)

2.4.5 Alternate Path Search

Code Name: MUSTRP or RING2

Programmer: Helen Hill

Abstract: MUSTRP is given a connection table path between two nodes in a ring and searches for any alternate paths between these two nodes. If more than one alternate path is found, the "best" of these is chosen.

2.4.5.1 Program Description

MUSTRP examines a connection table which has been compressed as described in Section 2.4.4.1. The program takes a pointer to a given path between two nodes in a ring and looks for the "best" alternate path between those two nodes. The criterion by which one path is considered to be "better" than another path is described in Section 2.4.4.1.

On being called by the main ring program RING1, MUSTRP checks that the given pointer indicates a path of length 2 or greater and also compares the two terminal nodes. If these are found to be the same node, control is returned to the main program with an indication that a self-ring was found. This occurs when a ring contains at most one atom which has more than two connections to it.

The program follows each path from one of the nodes until the path length exceeds the given path length without having passed through the second node. Given the compound
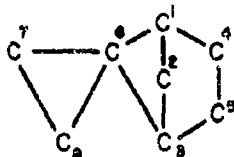
and given the path 1-2-3, the program begins with atom 3 and examines each path from this atom. Path 3-5-4-1 is found to be too long to qualify as "better" than the given path 1-2-3, but it is an alternate path, so this is noted as the present best alternate path. Path 3-6 is a possible candidate, so it is stored in a linked list. The program then examines other paths leading from atom 6 to determine whether the 3-6 path is part of a better path between 3 and 1. Path 3-6-7-8-6 is considered and found to be too long to qualify. Path 3-6-8-7-6 is also too long. The only remaining alternative is 3-6-1, and this now turns out to be a better path than the present best alternate path 3-5-4-1. It therefore replaces it as the best alternate path. This is returned to the main program with a bit set indicating that an alternate path was found. If no alternate path is found, the given path is returned without the bit being set.

A flow chart of the program is presented in Figure 38.

Figure 38.   Macro Flow Chart - RING2

151
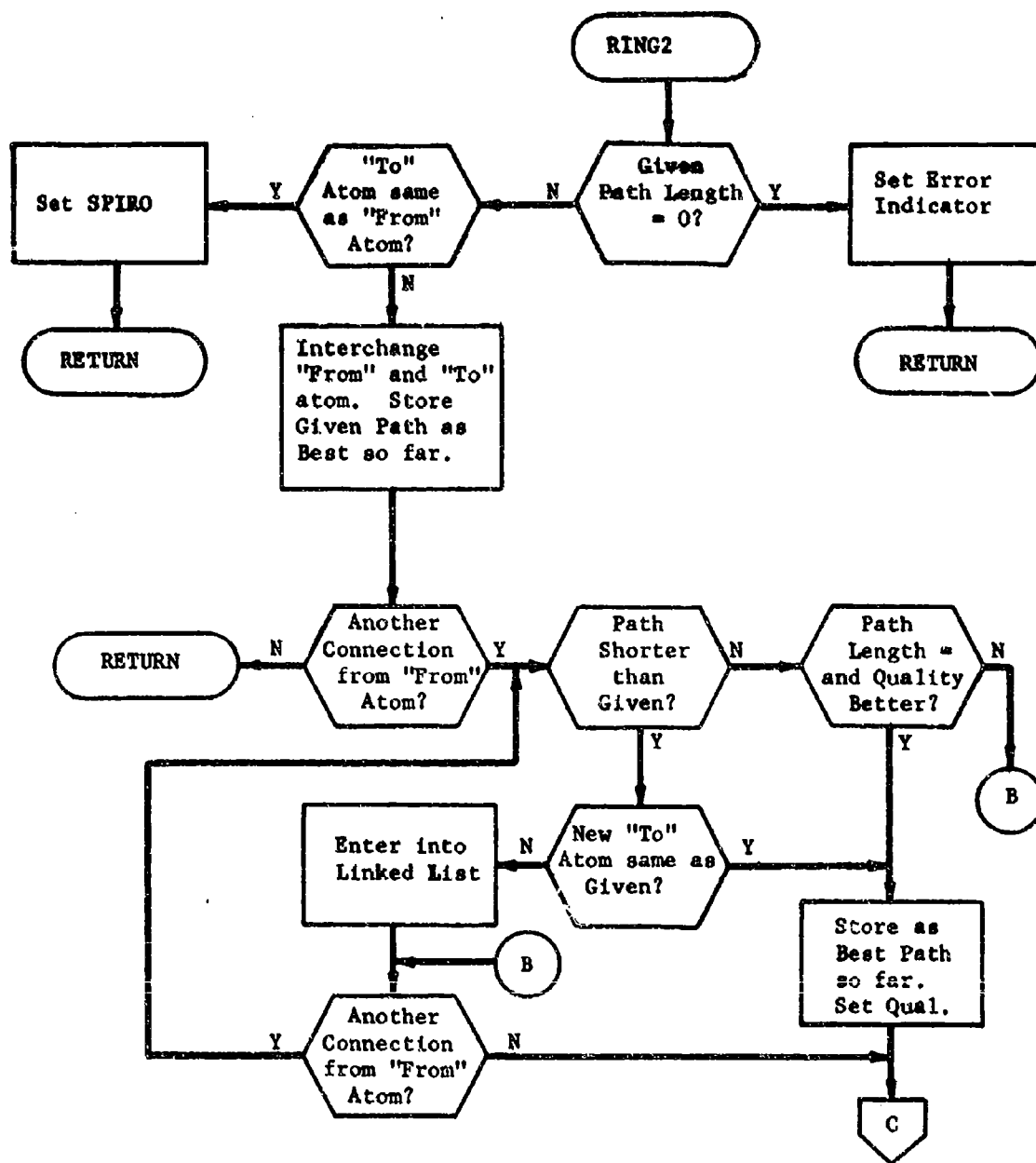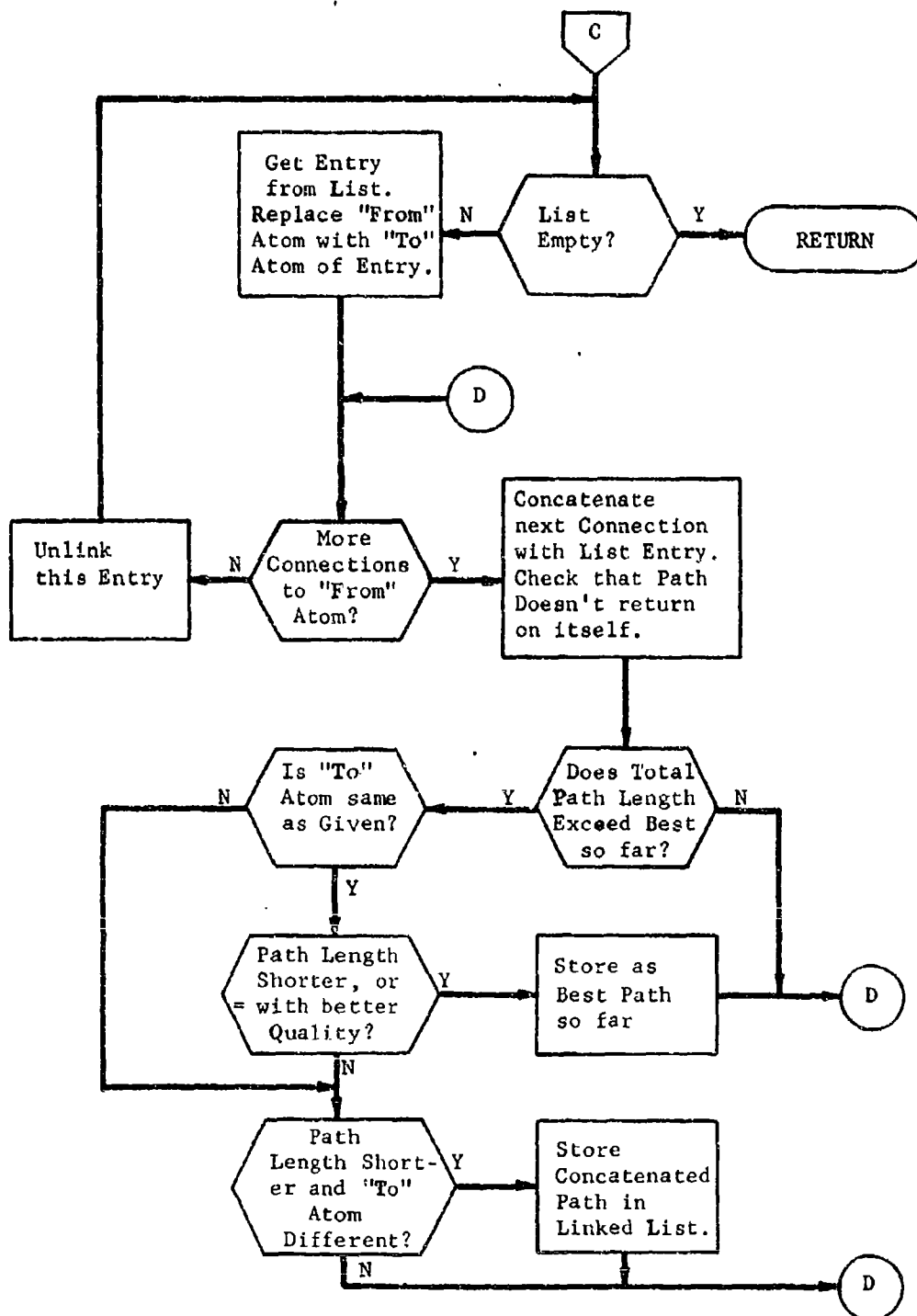
Figure 38. Macro Flow Chart - RING2 (Continued)

152

2.4.5.2 Program Structure

RING2 is a subroutine of the Ring Analysis portion of the Key Assignment System. The input consists of a pointer to a connection table entry. In addition, the X, E, C and ECOUNT tables are in core.

The output consits of a 7 word block describing the best alternate path found by the program. The format of the block is

Word	Contents
1	Path length
2-3	One of 72 bits is set for each atom number in the path
4-7	Stores the count of atoms of each element type in the path

In addition, the following indicators are set by the program:

BADAT - Set to indicate an error was detected
SPIRO - Set if a self-ring was found
QUAL - Set if an alternate path was found.

2.4.6 Ring Compression

Code Name: COMPR or RING3

Programmer: John D. Leggett

Abstract: The purpose of program COMPR is to remove all atoms in the connection table which have exactly two attachments and to remove side chains from the structure, in order that the ring descriptors may be found. The program also contains a subroutine which removes a prescribed path from the structure.

2.4.6.1 Program Description

The compression portion of the program operates similarly to program CONVRT, with two exceptions: (1) all atoms with two connections are removed, and (2) the bond table is not treated. As each atom is removed, the internal node numbers (or numbers representing previously removed atoms are combined to form the description of the path between the two atoms to which the removed atom is connected. Likewise the element description of the paths are combined to give the specification of the new path. Upon completion of the first compression, the side chains are removed, and the structure is compressed again.

When a ring has been found in the structure by program RING1, the path removal subroutine (SSC) is called. The connection table is compressed again and side chains are removed after a path has been deleted. If a self-ring is found at any stage of the compression, control is immediately returned to the main ring program, since this ring must be one of the desired cycles.

A flow chart of the program is presented in Figure 39 .

2.4.6.2 Program Structure

RING3 is a subroutine of program RING1. The input to the program is the connection table produced by RING4. The output consists of the compressed connection table and/or a connection table with some path removed, and/or the location of any self-ring.

If an error is found in the connection table, control is returned to the main ring program with an error bit set.

154
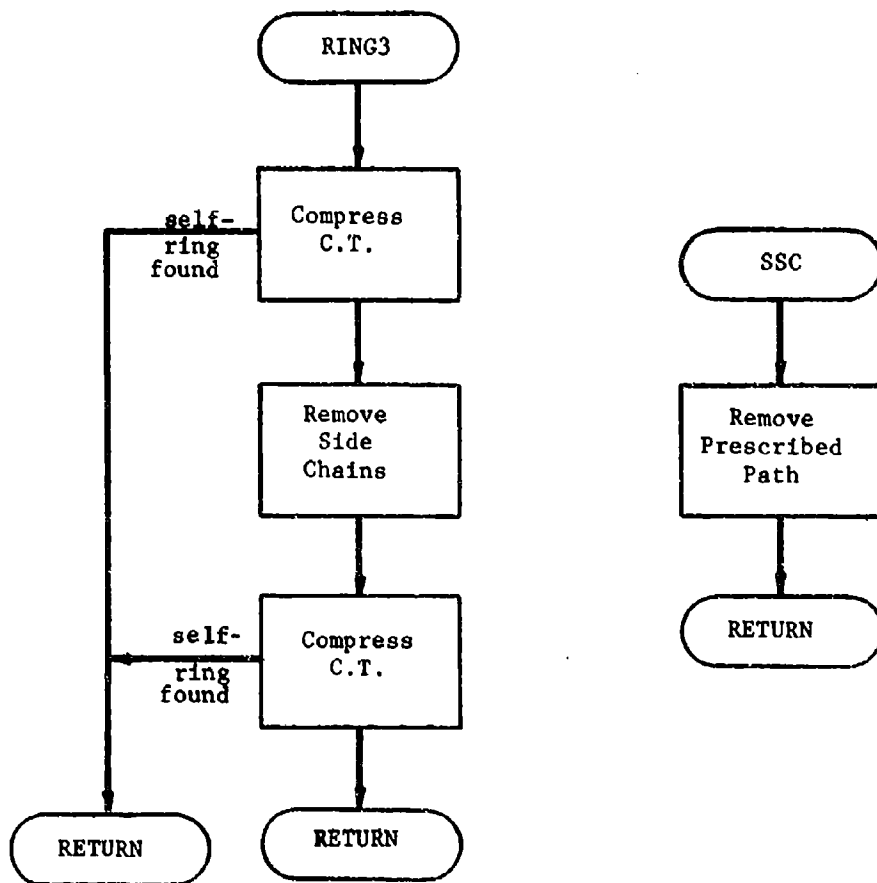
Figure 39.   Macro Flow Chart - RING3

2.4.7 Connection Table Expansion

Code Name: TABLE or RING4

Programmer: John D. Leggett

Abstract: The purpose of the program is to expand a connection table given in the compressed format (see program CONVRT) to a form suitable for application of the ring analysis programs.

2.4.7.1 Program Description

The program takes the connection table in the form in which it is output from CONVRT (Section 2.1.2) and expands it into X and E tables. This form is the same as the stage just prior to formating in CONVRT.

The next step is the generation of path descriptors. The first such descriptor is the atom numbers of the nodes present in a given path, including the "from atom" but not the "to atom". For any atoms not explicitly specified in the connection table (i.e. if a path is of length greater than one) artificial atom numbers are added. The second descriptor is the element kind of each atom in the given path.

A macro flow chart of the program is presented in Figure 40.

2.4.7.2 Program Structure

RING4 is a subroutine which is called by RING1. It is given a compressed connection table (Section 2.1.2.2) as input. The output consists of X and E tables as described in Section 2.1.2.2 and the generation of path descriptors for use by the other ring analysis programs.

If an error is detected in the connection table, control is returned to the main ring program with an error bit set.
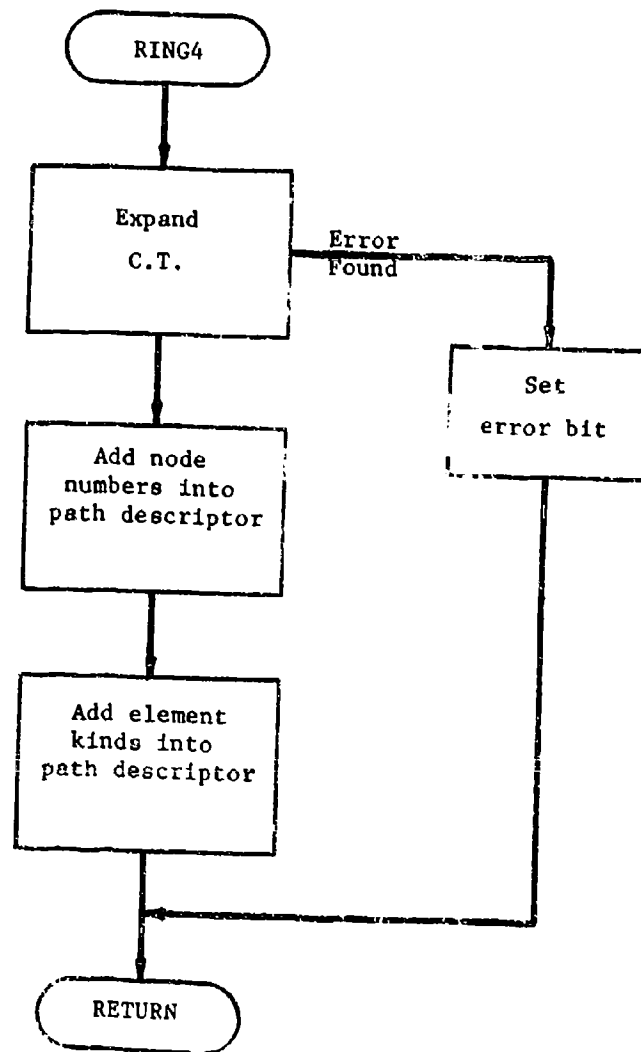
Figure 40.   Macro Flow Chart - RING4

2.4.8 Atom-by-Atom Search

Code Name: STRUC

Programmer: John D. Leggett

Abstract: The purpose of the atom-by-atom search program is to determine if a one-to-one correspondence exists between the nodes (atoms) and connections in a given query structure, and some set of nodes and connections in a given file compound structure. The element kinds of the nodes in each are compared as well as the connections between the atoms, and any cyclic properties. The output of the program is a yes or no answer for a given query fragment and compound structure. Provision is made to erase portions of the file compound in core during processing to avoid fragment overlap during repetitive search.

2.4.8.1 Program Description

In order that a one-to-one correspondence exists between the query nodes (atoms) and branches (bonds), and some set of nodes and branches in the file compound, it is necessary and sufficient that every possible path through the query structure have a corresponding path through the file compound. It is not necessary to examine every path of the file compound, provided that every node and branch in the query is contained in the set of paths that we chose to examine.

It is required that each pair of atoms matched have the same element kind, except for two special codes used in queries, which may match any of a set of element types in a file compound. These are (1) EE which represents any element except hydrogen, and (2) EL which represents any element except carbon and hydrogen. In addition, any abnormality in the query must be satisfied. It is also required that each pair of atoms matched have the same number of direct attachments except where this restriction is specifically relaxed by a special character in the query.

In the connection table, special indicators mark atoms which are members of a ring and paths which are part of a ring. The atom-by-atom search requires that matched atoms must either both be in a ring or neither be in a ring. Likewise, paths must be matched as both either ring or non-ring paths.

The element of the starting query atom is matched against successive file atoms until a match is found. The trace then begins by placing one of the connections in the hypothesis list, and the remainder in a choice list. A similar step is taken in the file compound, and the pair of atoms in the hypothesis list is matched. If the test is successful, the stepping continues until (a) the end of a path is reached, (b) a cycle is closed, or (c) a mismatch is found. If (a) or (b) occurs the program searches back up the choice list for the last available (untraced) path in the query and file compound, and the trace begins again. When a mismatch is found, the trace backs up to the last available alternate path in the file compound and starts again using the same query path.

When the alternate paths have been exhausted, a new file compound start-ing atom must be selected. If the list of possible starting atoms is exhausted, the search is said to fail.

If every path in the query is successfully matched against a path in the file compound, the search is said to pass. In this case any appropriate erasure of the file compound takes place. Thus if the program is called again to determine if the query fragment is present in another part of the file compound, the set of atoms which were previously matched with the query fragment are no longer available choices. Thus the program must look further in the connection table to locate some other set of atoms which satisfies all the requirements of the query.

2.4.8.2 Program Structure

STRUC is a subroutine which serves important functions in several different parts of the CIDS system. In the Key Assignment System, STRUC is called to determine if a given screen fragment is satisfied by a given file compound in order to decide if the corresponding key should be assigned to the compound. In the Retrieval System, STRUC is called to decide whether a file compound should be retrieved by a query by determining whether the compound satisfies the query structure. In the Registration System, STRUC determines if a possible registrant has the same connection table as a registered compound which has the same molecular formula.

The program requires as input the core locations of the connection tables for the query fragment and the file compound, and the length of the latter. In addition, the locations(s) of the abnormality table(s) if any are required. Location ERASE is used to indicate the type of erasure (if any) to be performed.

If a match was found between the query and the file compound, STRUC returns to the calling program with the accumulator non-zero. If the search failed, the accumulator is set to zero before returning control.

A macro flow chart of the program is presented in Figure 41.
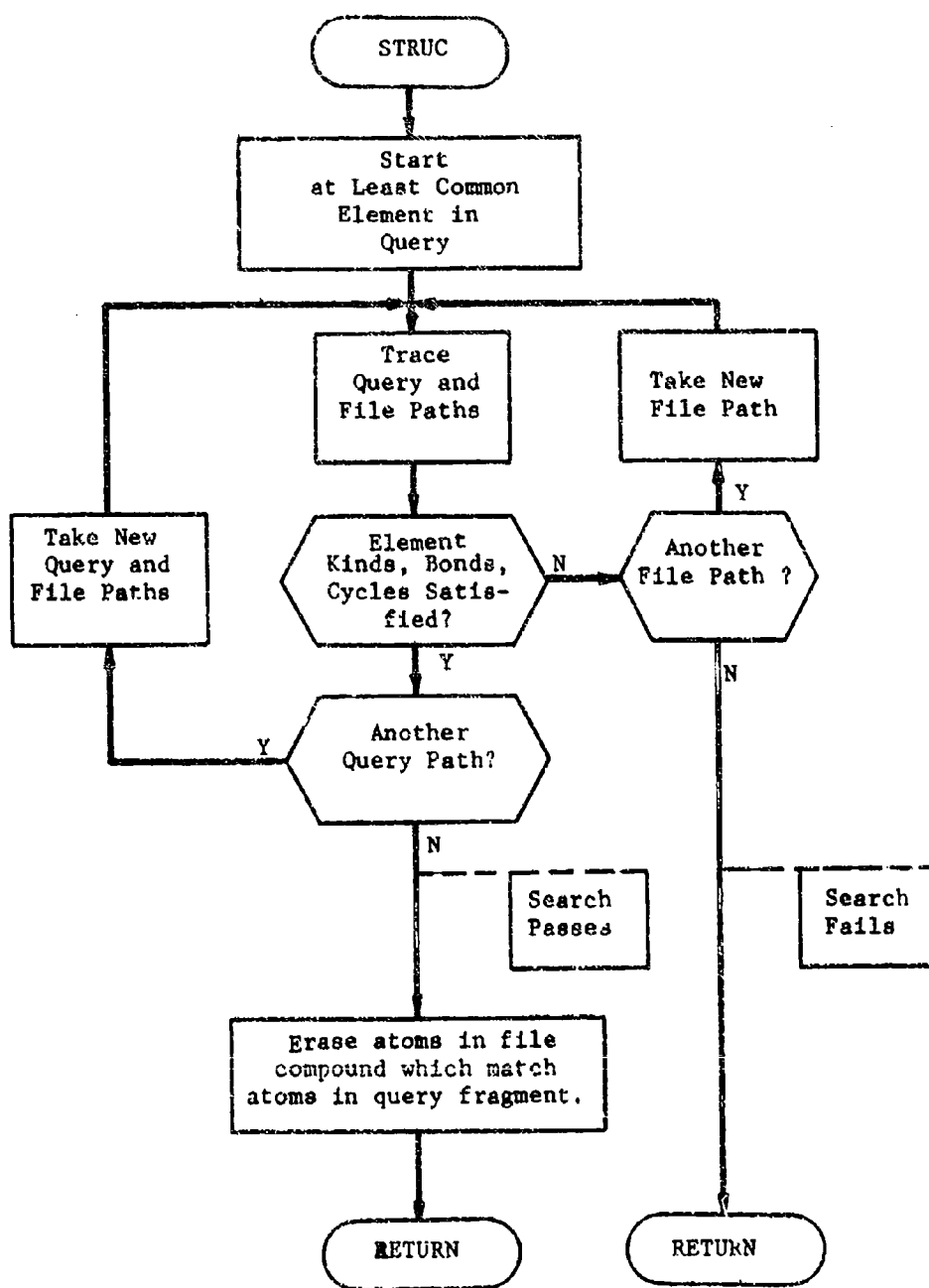
Figure 41.   Macro Flow Chart - STRUC

2.4.9 Nonspecific Hydrocarbon Radical Key Assignment

Code Name: HCRCT

Programmer: Jeffrey H. Kulick

Abstract: Program HCRCT has been developed to assign to a structure a sub-class of the set of aliphatic hydrocarbon radical keys.

2.4.9.1 Program Description

HCRCT assigns to a compound the following hydrocarbon radical keys:

(1) A single key, 3-A-1-28, is assigned to compounds which contain a __straight__ single-bonded (i.e. saturated) carbon chain which is attached to __only one__ non-carbon atom and which contains more than 19 carbons.

(2) The key $(C)_n$ — El is assigned to any single-bonded carbon chain (branched or unbranched) which contains at least 5 carbon atoms and is attached to one and only one non-carbon atom (El). A different key is assigned for each value of n.

Program HCRCT first creates two dictionaries. Both have one entry per atom. The first dictionary contains the following information for each atom:

(1) Ring or non-ring atom
(2) Number of attachments to the atom.

second dictionary contains the following information for each atom:

(1) Carbon or non-carbon
(2) A number which identifies the hydrocarbon radical to which this atom has been assigned
(3) Location of this atom in the connection table.

The program first locates a "candidate atom" (a carbon atom with only one attachment that has __not__ previously been assigned to a hydrocarbon radical). If the attached atom is carbon, then each of its connections are searched in turn to see if the structure qualifies for being a monovalent hydrocarbon radical (HCR). A count of the number of atoms is maintained as this search progresses. Additionally, as each atom is searched in turn, the second entry of Dictionary 2 is filled in, indicating membership in a candidate HCR. This is to prevent double assignment for atoms 1 and 2 in the structure:

$$C_1$$
$$|$$
$$N - C - C - C - C_2$$
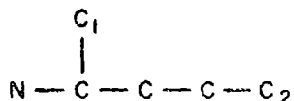
The search along any particular path is continued until a non-carbon is encountered. The number of non-carbons attached to any candidate HCR is saved, and a key is assigned only if this count equals one. A candidate HCR is also rejected if any double or triple bonds appear within it. A further restriction is that none of the carbon atoms within the HCR may be a member of a ring, but the non-carbon to which it is attached may or may not be.

A flow chart of the program is presented in Figure 42.

2.4.9.2 Program Structure

HCRCT is a subroutine of the Key Assignment System. It requires as input a compressed connection table as described in Section 2.1.2.2. The accumulator must contain the core address of the connection table, and index register 1 must contain the length of the connection table. The program stores any keys that are assigned to the compound in an array for later addition to the compound record.
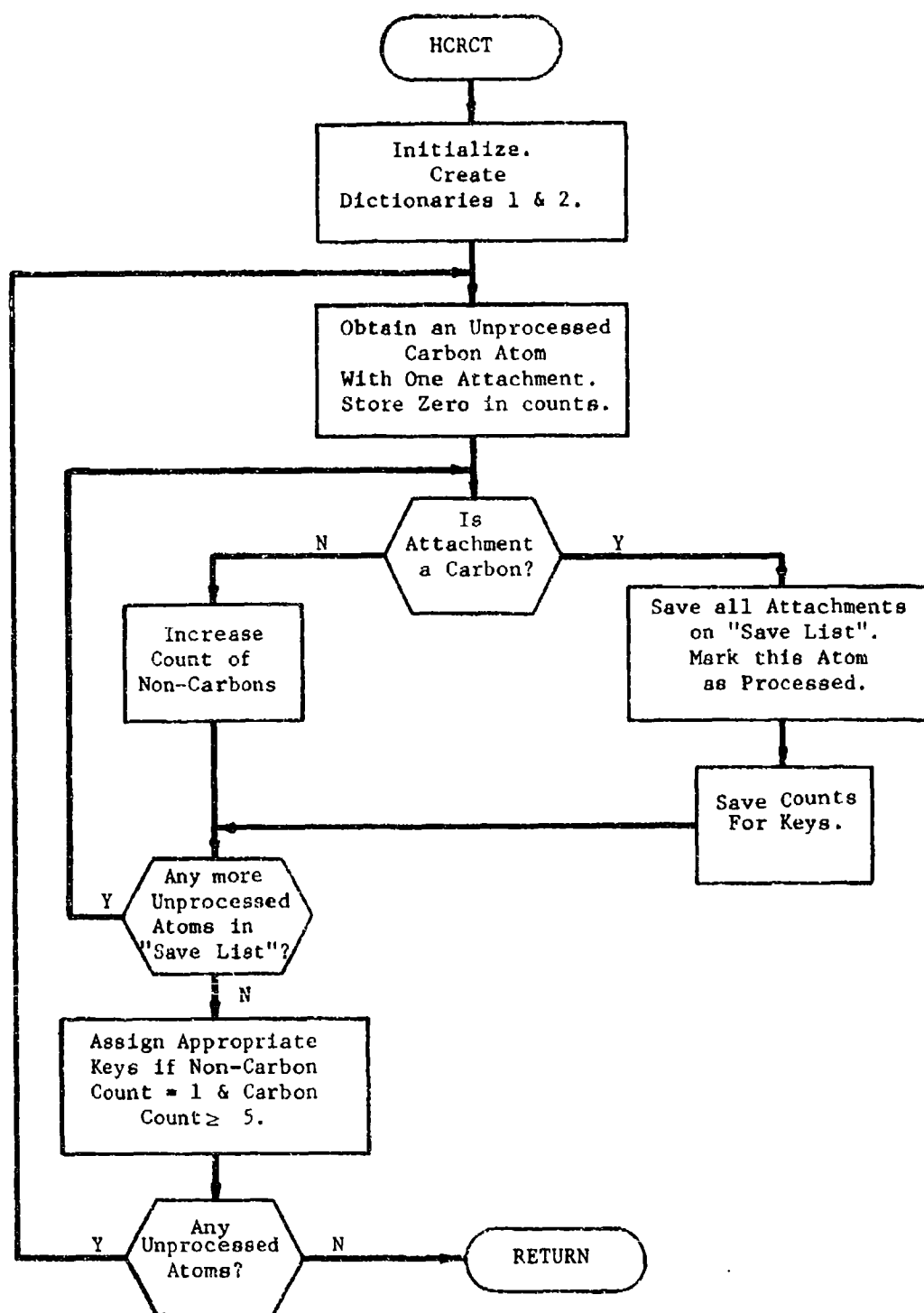
Figure 42. Macro Flow Chart - HCRCT

163

2.4.10   Bond Count

Code Name: BONDCT

Programmer: Jeffrey H. Kulick

Abstract: Program BONDCT was developed to assign a specific limited sub-class of the aliphatic keys. It assigns the acyclic nucleus keys and two types of hydrocarbon radical keys.

2.4.10.1   Program Description

BONDCT assigns the following keys described in CIDS No. 4 as acyclic nuclei keys to acyclic compounds or acyclic addends of cyclic compounds only:

(1)   Number of double bonds between carbon atoms

(2)   Number of triple bonds between carbon atoms

(3)   Number of carbons with three carbon attached regardless of internal bonding or additional non-carbon connections

$$
\begin{array}{cc}
C-\overset{\displaystyle |}{\underset{\displaystyle |}{C}}-C & \qquad C=\overset{}{\underset{\displaystyle |}{C}}-C \\
C & \qquad C
\end{array}
$$
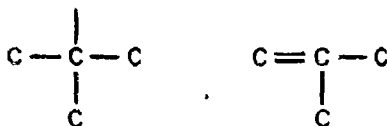
Number of carbons with four carbons attached regardless of additional connections

$$
C-\overset{\displaystyle C}{\underset{\displaystyle C}{\overset{\displaystyle |}{\underset{\displaystyle |}{C}}}}-C
$$
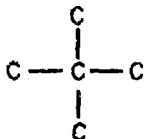
Note that keys are assigned for each of the above four structural features even if some of the resulting counts are zero. Separate counts are made for each acyclic parent or addend, and separate keys are assigned for each, except that at most one zero key is assigned to a compound for any of the four above keys.

In addition, the following hydrocarbon radical keys are assigned to any compounds in which they occur:

(1)   El — (C)$_n$— El   For any aliphatic unbranched single-bonded carbon chains attached to two non-carbons (El), a key is assigned which gives the length of the chain. Each El can be any element other than carbon (or hydrogen) and may or may not be a member of a ring. They cannot, of course, be in the same nucleus.

(2)   El — C ≡ C — El   This key is listed as 3-B-2-1 in CIDS No. 4.

164

Program BONDCT first creates 2 dictionaries. One has an entry for each atom.
This dictionary contains information as to whether the atom is carbon or not,
whether it is in a ring or not, and which addend it is in.

The second dictionary contains one entry for each entry in the bond table,
i.e., for every bond in the CIDS compressed connection table as described in
Section 2.1.2.2. This dictionary tells whether a particular bond is a ring bond
(a bond in a cycle).

BONDCT starts at the last atom, and processes each atom from last to first.
The basic processing for each atom is as follows:

(1) Find an atom

(2) For each bond chain (compressed carbon chain) connected to
that atom:

(a) Count the double and triple bonds between carbon
atoms.

(b) If the "From" and "To" atoms are non-carbon and
all bonds are single bonds, count the bonds and
assign key for $El - (C)_n - El$.

(3) If the "From" atom is carbon, check the number of carbon
connections for possible increase in the count of

$$-\overset{|}{\underset{|}{C}}- \quad \text{or} \quad -\overset{|}{C}- \quad \text{configurations.}$$
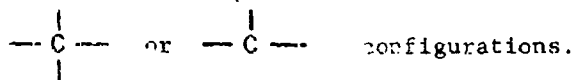
When all atoms have been processed, all sums are divided in half (because of
redundant entries) and keys are assigned.

A flow chart of the program is presented in Figure 43.

2.4.10.2 Program Structure

BONDCT is a subroutine of the Key Assignment System. A call to subroutine
HCRCT must precede a call to BONDCT because that program sets up certain arrays
which are used by BONDCT. The input requirements and the output are the same as
for program HCRCT (Section 2.4.9).
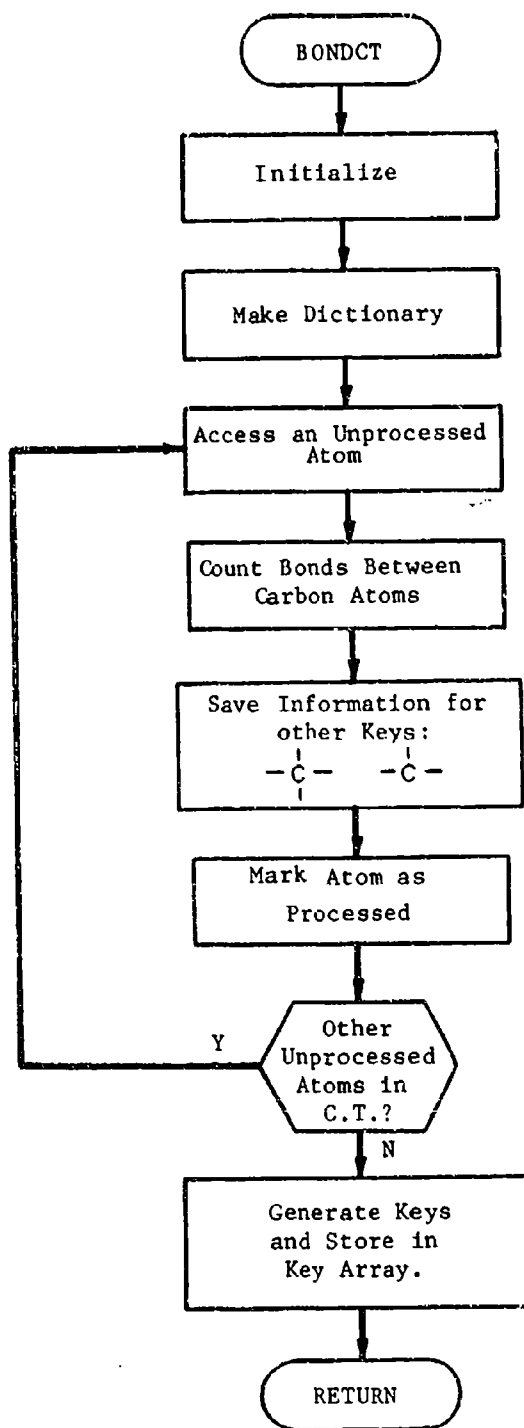
Figure 43. Macro Flow Chart - BONDCT

Molecular Formula Key Assignment

    Code Name: MFSRN

    Programmer: Ruth V. Powers

    Abstract: Molecular Formula keys are assigned to a compound based on the Hill molecular formula. One key is assigned to identify each element present.

2.4.11.1 Program Description

    A key is assigned for each element appearing in the Hill molecular formula. The key specifies the number of atoms present for the elements: C, H, N, O, P, S, F, Cl, Br, I, Si and B. For all other elements the key specifies only the presence of the element, regardless of the number of atoms. In addition, a metal key is assigned to the compound if it contains elements other than the twelve special listed above and As, Sb, Se, Te.

    A flow chart of the program is presented in Figure 44.

2.4.11.2 Program Structure

    MFSRN is a subroutine in the Key Assignment System. It is called by a 'TSL MFSRN'. Location COMPND must be defined in the calling program as an entry point. The location must contain 2 in the tag portion, and the address of the first word of the Hill mol form (the second word of the mol form block) in the address portion.

    The keys assigned to the compound are stored in the SCKY array, which is defined as an entry point in the calling program. The new keys are added to those already stored, and location SCKY is updated to equal the number of words in the array. The second word of each mol form key is zero. The first word has the format:

(S,1-5)	(6-17)	(18-35)
101000	El	Count

"El" is the abbreviation of the element kind in BCD. For a single letter abbreviation, such as "C", a BCD blank precedes the letter. "Count" is the number of atoms of that element kind if it is one of the twelve special elements listed above. Otherwise it is zero. The abbreviation "M" takes the place of the element kind for metal keys.

    If sense switch 2 is pressed in, the keys are printed on the system output tape.
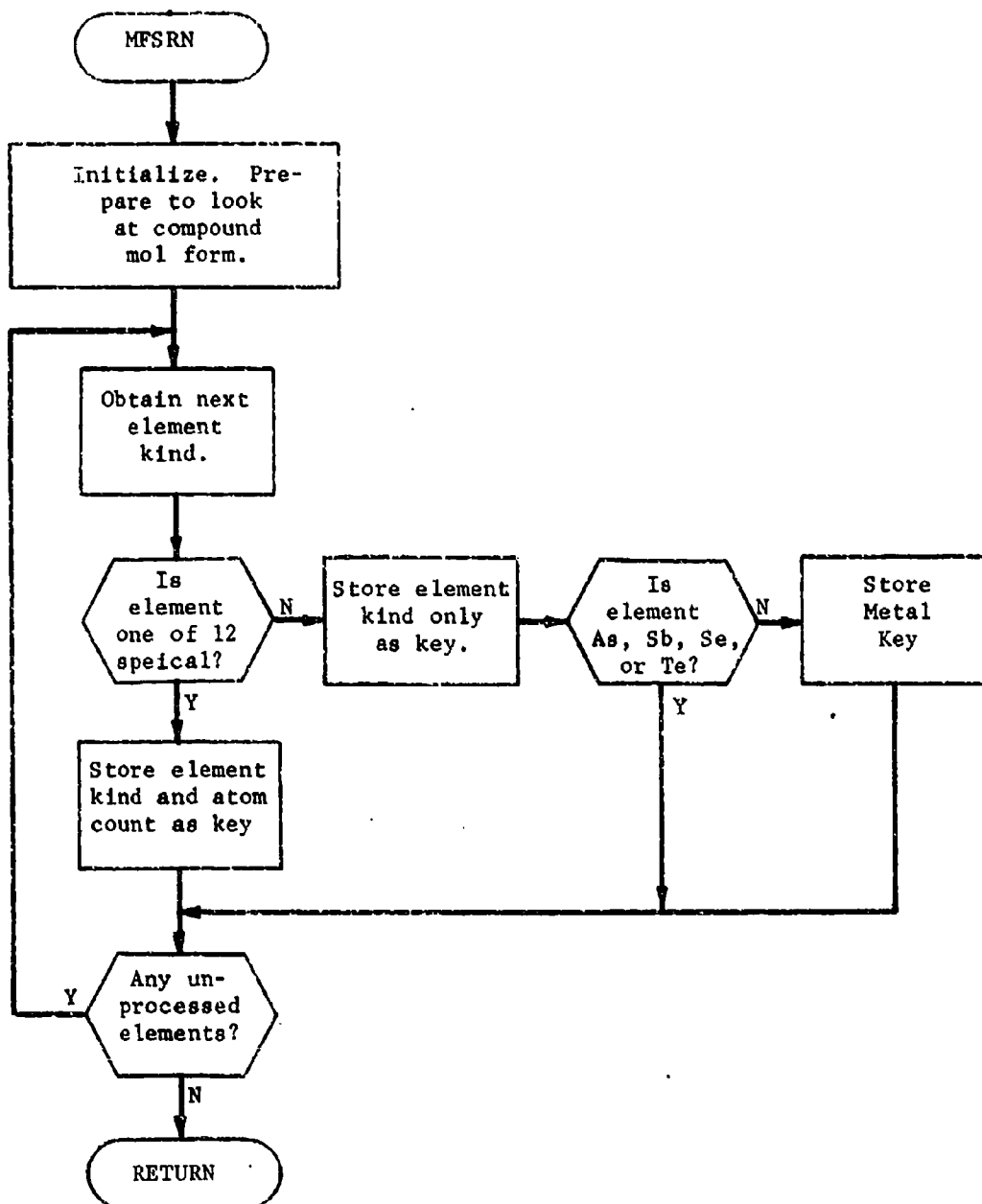
Figure 44. Macro Flow Chart - MFSRN

2.4.12   Nonspecific Phosphorus Functional Group

Code Name: PSCKYT

Programmer: Ed Hebel

Abstract: Subroutine PSCKYT assigns keys to compounds which contain certain types of phosphorus functional groups which were not among those selected as Specific Functional Group keys (listed in Table XIV in CIDS No. 4).

2.4.12.1   Program Description

Subroutine PSCKYT examines a compound connection table (C.T.) in core for the presence of nonspecific phosphorus functional groups. Previous to this, the C.T. was searched for the presence of Specific Functional Group fragments, and whenever one of these was located, the non-carbon atoms which corresponded to those in the fragment were zeroed out in the C.T. Thus, it is known that none of the phosphorus atoms remaining in the C.T. were present in any of the specific fragments.

PSCKYT first examines the C.T. for phosphorus atoms. When one is located, its connections are examined, and keys are assigned based on the combination of the following elements which are attached to the phosphorus atom: N, O, S, X (any of the halogens: F, Cl, Br, I), and the CN group. The various keys which may be assigned are listed in Table XVI in CIDS No. 4. A macro flow chart of the program is presented in Figure 45.

2.4.12.2   Program Structure

Subroutine PSCKYT is called with the core address of the compound connection table in the address portion of the accumulator. If any keys are assigned, the corresponding two-word codes are stored in the SCKY array and the count of keys is increased.
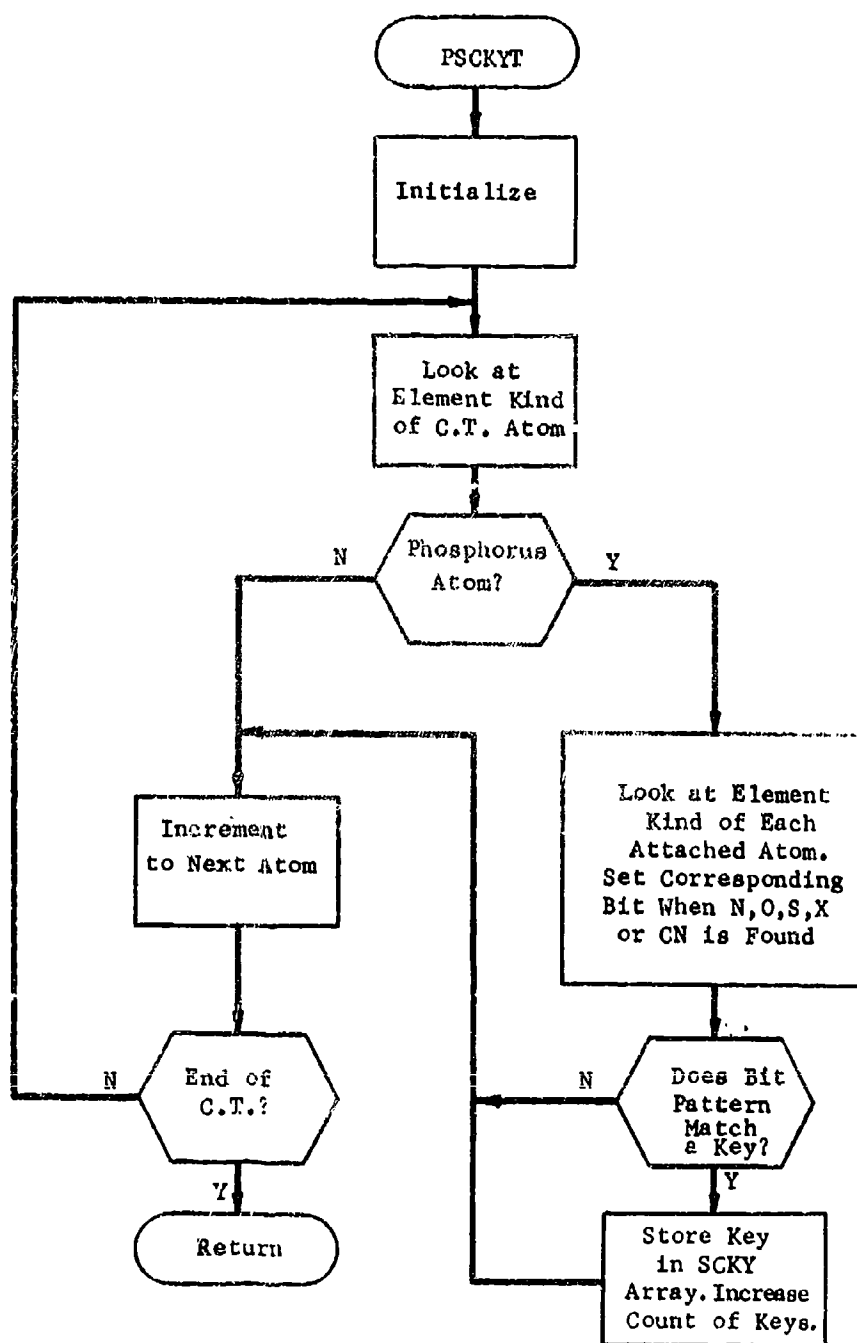
Figure 45.   Macro Flow Chart - PSCKYT

2.5 LIST-STRUCTURED FILE GENERATION

Programs NUFILE, KEYSRT, MERGE, and INDEX together create or update the search file, and form the inverted key index. The tape files produced are the input to the CIDS Retrieval System. The *four* programs are described in this section and their interrelationship is shown in the following schematic:
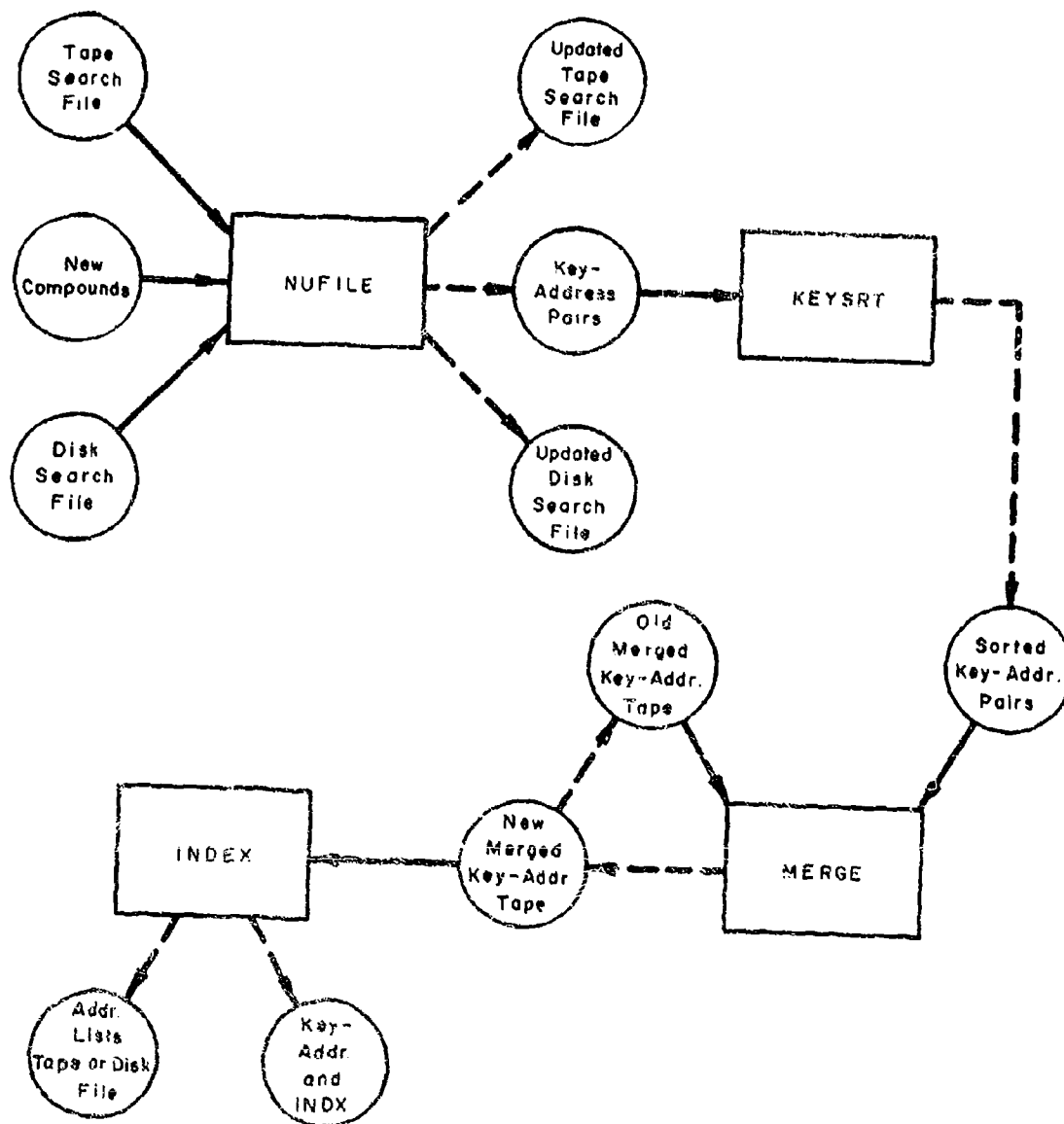
2.5.1   Search File Creation or Update

Code Name: NUFILE

Programmer: Peter J. Brown

Abstract: NUFILE creates a search file of compounds by simply assigning each compound to an area in the file as it is input to NUFILE. An existing file may be updated by the same process. NUFILE simultaneously creates a tape of key and file address pairs which will be used by programs MERGE and INDEX to create an index to the complete compound file.

2.5.1.1   Program Description

NUFILE may either create the initial file of compounds or update an existing search file. Either function may be selected by placing an appropriate data card at the end of the deck (described in Section 2.5.1.2)

NUFILE simultaneously creates or updates both a tape search file and a disk search file. NUFILE may also create or update only a tape search file. Either function may be selected by means of a sense switch (section 2.5.1.2). The disk search file is actually produced on tape in a format which will permit an easy transference to a disk unit and differs from the tape search file only in the blocking of the data. This difference causes the same compound to have different file addresses in the two files. The key-address tape, from which the index is formed, must therefore have two addresses for each key on the list.

A macro flow chart of this program is presented in Figure 46.

2.5.1.2   Program Structure

NUFILE is a main program which requires as input (1)the new compound tape(s) and (2)the last tape of the existing search file.

The new compound file is the output of the final screening program. The information on this tape is blocked in IOBS Type 2 records. The last compound is followed by a special sentinel record, the first word of which is zero, to signal the end of the input data.

The last tape of the existing search file is required only if NUFILE is being used to update an existing file. The information on this tape, up to and including the last compound record, is copied onto the output tape. Following this record and an end-of-file mark, there is a special record containing the count of compounds presently in the file, and the next file address to be assigned to a compound. After this tape is copied, it is rewound and unloaded, and this tape drive becomes the alternate unit for the search-file output tape.

Two data cards must be placed at the end of the deck (in the following order):

> (1)   A card containing the maximum number of blocks to h
> placed on each tape of the Tape Search File. If the

```
                    ┌─────────────┐
                    │   START     │
                    └──────┬──────┘
                           │
    ┌──────────────────────┤
    │                      │
    │              ┌───────▼───────┐
    │              │ Input a New   │
    │              │  Compound     │
    │              └───────┬───────┘
    │                      │
    │                 ╱────▼────╲
    │                ╱ Is it the ╲        Y
    │               ⟨  Sentinel   ⟩──────────────┐
    │                ╲ Compound? ╱                │
    │                 ╲────┬────╱            ┌────▼────┐
    │                      │ N               │   END   │
    │                      │                 └─────────┘
    │           ┌──────────▼──────────┐
    │           │ Place Compound on Tape File. │
    │           │ Place each Key Followed by   │
    │           │ the Compound's Tape Address, │
    │           │  on the Key-Address Tape.    │
    │           └──────────┬──────────┘
    │                      │
    │                 ╱────▼────╲
    │              N ╱ Is Sense  ╲
    ├───────────────⟨  Switch 5   ⟩
    │                ╲    In?    ╱
    │                 ╲────┬────╱
    │                      │ Y
    │           ┌──────────▼──────────┐
    │           │ Place Compound on Disk File. │
    │           │ Place Compound's Disk Ad-    │
    └───────────│  dress After Each Key,       │
                │   on Key-Address Tape.       │
                └─────────────────────┘
```
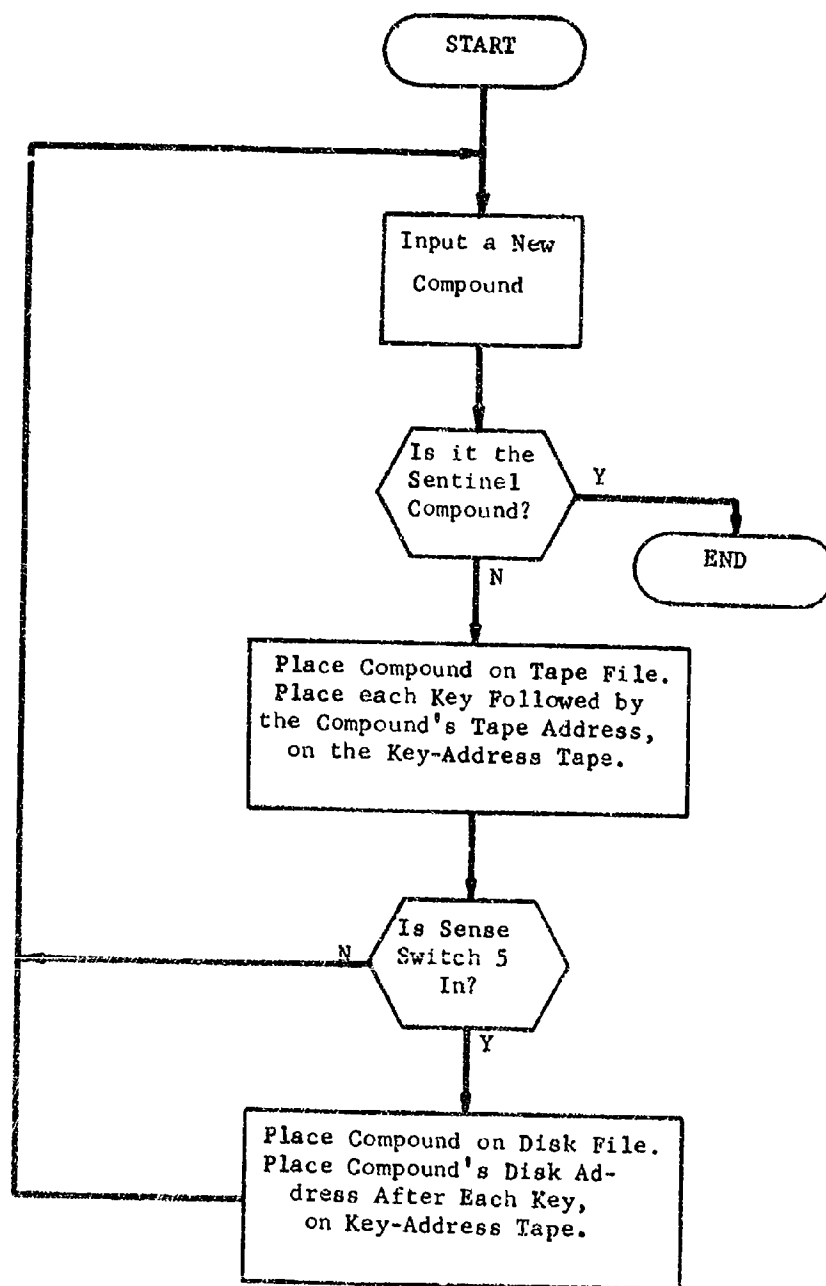
Figure 46.   Macro Flow Chart - NUFILE

tape is filled before this number of blocks is reached, tape switching automatically occurs. This is a safe-guard so that the tape may be copied on a slightly shorter tape and still contain the same number of blocks as the original. For 2400 foot tapes, 2400 blocks leaves about a quarter of an inch of turns unused. This number is punched in columns 1 through 6; leading zeros must be punched.

(2) A card with either the word NUFILE or UPDATE punched in columns 1 through 6: This determines whether a new file is to be created or an existing file updated.

The output consists of (1) the Tape Search File, (2) the Disk Search File, and (3) the Key-Address tape.

The Tape Search File is composed of a series of tapes with variable length block size, up to a maximum of 1000 words. A compound record cannot be split between two physical records. The information on each tape is followed by an end-of-file mark and a small (10 word) dummy block. At the end of the last tape of the file, following two consecutive end-of-file marks, is a special record 10 words long. The first word contains the next file address to be assigned to a compound in the format:

Bits	Contents
(S,1-5)	Tape Number (1 - `
(6-18)	Record Number (0
(19-35)	Relative address (0 -)

The second word contains (right-adjusted) the current number of compounds in the file

The Disk Search File contains the same items as the Tape Search File and in the same order. The block size is 465 words. Compounds may be split between two physical records, but never more than two. The end-of-tape and end-of-file sentinels are the same as for the Tape Search File, except that the 10 word special record at the end of the data file contains the following information:

Word	Bits	Contents
1	(S,1-17)	Record (track) Number (1 -)
	(18-35)	Relative address (0-464)
2		Number of words left unfilled in last data record written on the tape (right-adjusted).

The Key-Address Tape is an IOBS tape of Type 1 records with LRL=4, RCT=250, block size=1000. It contains a four word logical record for each key occurrence on the compound input tape in the format:

Word	Contents
1	Key (high order 36 bits)
2	Key (low order 36 bits)
3	Address in Disk Search File of compound containing this key
4	Address in Tape Search File of compound containing this key.

This tape is used as the input to KEYSRT.

2.5.1.3 Operator Instructions

The new compound tape(s) are loaded alternately on S.SU04(C4) and
S.SU06(C5). Output tapes for the Tape Search File are loaded alternately on
S.SU10(B1) and S.SU07(B6). Output tapes for the Disk Search File are loaded
alternately on S.SU19(C6) and S.SU02(C3). For Update runs the last tape for
the previous Tape File is loaded on the first output unit for the Tape File.
The first output tape is then loaded on the alternate unit. If a Disk File
is to be updated the last tape of the previous Disk File must be loaded on the
first Disk File output unit. The Key·Address tape(s) are loaded on S.SU05 (B5).

2.5.2   Key-Address Sort

Code Name: KEYSRT

Programmer: Peter J. Brown

Abstract: KEYSRT sorts the Key-Address tape which is output from program NUFILE. The key-address pairs are sorted in ascending order according to key number. It maintains the ascending order of addresses as they are produced by NUFILE.

2.5.2.1   Program Description

KEYSRT is an IBSRT program. It is a logical sort wherein the sign bit of a word is considered the high order bit of that word. The option EQUALS insures that in the case of a tie (i.e., two keys being the same), they are put on the output tape in the same order as they appeared on the input tape. This keeps the addresses corresponding to these keys in ascending order.

2.5.2.2   Program Structure

KEYSRT is a main program which requires as an input, the Key-Address tape which is output from NUFILE. The output produced is the Sorted Key-Address tape which is in the same format as the input tape. This tape becomes the input to program MERGE.

2.5.2.3   Operator Instructions

The input tape must be mounted on unit B5 (S.SU05), and the output tape is mounted on unit C4 (S.SU04). This is an order four sort. This requires, in addition to the input and output units, eight scratch units, four on each channel. If there are not enough units available, there are two alternatives

  (1) If short by one unit, the input unit can be used as a
       scratch. Push START when IBSRT requests another unit.
       Then, when the information on the input tapes have been
       read, IBSRT will request that a scratch be mounted and
       readied on that unit. The output unit can be utilized
       similarly -- before the last phase of the sort, IBSRT
       will indicate the unit on which the final sorted infor-
       mation will be placed.

  (2) The other alternative is to select, on-line, a lower
       order sort. This is standard operating procedure.

IBSRT will request that the number of input reels be "keyed" in.

2.5.3 Key-Address Merge

Code Name: MERGE

Programmer: Peter J. Brown

Abstract: MERGE combines the Sorted Key-Address tape (see NUFILE and KEYSRT) with the Old Merged Key-Address tape containing all the keys in the file (prior to the present run) and the addresses of their occurrences. This combination results in a New Merged Key-Address tape, which is an inverted file (by keys) of all the compounds in the file. This tape is the input to program INDEX, which creates a three level key-to-compound Locater Table.

2.5.3.1 Program Description

MERGE groups together all the Search File addresses paired with the same key on the Sorted Key-Address tape. These addresses are written on the New Merged Key-Address tape in one list, with the key present only at the head of the list. This is illustrated in the following example:

Sorted Key-Address Tape	New Merged Key-Address Tape
Key 1	Key 1
Address I	Address I
Key 1	Address II
Address II	Key 2
Key 2	.
.	.
.	.

During update runs, MERGE combines the Old Merged Key-Address tape with the Sorted Key-Address tape to produce a New Merged Key-Address tape.

2.5.3.2 Program Structure

MERGE is a main program which require., as input (1) the Sorted Key-Address tape and (2) the Old Merged Key-Address tape. The format of the Sorted Key-Address tape is described in Section 2.5.1.2.

The Old Merged Key-Address tape is present for Update runs only. No tape is needed when creating a new Index. This tape, which is the output of program MERGE, is blocked in variable length records, maximum block-size 1000 words. A logical record consists of a key and its occurrences, and thus it varies in size. At the end of each tape is an end-of-file mark followed by a dummy block. Two consecutive end-of-file marks terminate the last tape. The following is the logical record format:

Word	Contents
1	Key (1st half)
2	Key (2nd half)
3	Disk File Address of 1st compound containing this key
4	Tape File Address " " " " " "
.	.
.	.
.	.
2n-1	Disk File Address of last compound containing this key
2n	Tape File Address " " " " "
2n+1	Word of zeros (end of list sentinel).

The program produces as output the New Merged Key-Address tape. This tape has the same format as the Old Merged Key-Address tape.

2.5.3.3.   Operator Instructions

When running MERGE, the Sorted Key-Address tape(s) should be loaded alternately on units S.SU04 (C4) and S.SU06 (C5). For an Update run, the Old Merged Key-Address tape(s) should be loaded alternately on S.SU05 (B5) and S.SU07 (B6). The New Merged Key-Address tape(s) should be loaded alternately on S.SU10 (B1) and S.SU19 (C6).

The on-line typewriter will request that the number of reels be keyed in. The operator must key in the number of Old Merged Key-Address tape reels. For a 'nufile' run, for which there is no Old Merged Key-Address tape, key in zero reels.

2.5.4 Index creation

Code Name: INDEX

Programmer: Peter J. Brown

Abstract: The key-to-compound locater table, used by the CIDS Retrieval System, is created by program INDEX from the inverted key list produced by program MERGE.

2.5.4.1 Program Description

INDEX converts the inverted key list (output from program MERGE) into a three level key-to-compound locater table. The inverted key list, as it appears on the Merged Key-Address tape, contains each key in the system, followed by the Tape Search File and Disk Search File addresses of all corresponding compound records. From this, a Locater Table, or Index, may be created for either the Tape File or the Disk File. Either option may be selected by placing an appropriate parameter card at the end of the program deck.

INDEX places each address list from the inverted key list onto a new file (called the List-of-Addresses), which differs from the original in two ways: The "header" key of each list is removed, and each list is reduced to addresses of compounds in only one of the two search files. Each header key, together with the location on the List-of-Addresses of the list corresponding to this key, is placed on a second new file called the Key-Address List. Both of thse files are produced on tape, but are blocked in 465 word records to permit easy transference to a disk unit. The last key in each record in placed on a third file, called INDX, along with its track (or record) number. This file is small and is loaded into core at search time.

Figure 47 illustrates the construction of a three-level Tape File Index from the Merged Key-Address Tape. In the diagram, D represents the Disk File address and T represents the Tape File address. A_j represents the address of the List-of-Addresses level of the Index for those addresses corresponding to Key_j.

2.5.4.2 Program Structure

INDEX is a main program which requires as input the Merged Key-Address tape, which is output from program MERGE. Its format is described in Section 2.5.3.2. The program produces as output (1) the List-of-Addresses tape and (2) the Key-Address List. The latter tape also contains the INDX level of the Index.

The <u>List-of-Addresses Tape</u> is blocked in 1000 word records. Each list is followed by a word of zeros. Each tape is terminated by an end-of-file mark and a dummy block (10 words). The last tape, however, is terminated by two consecutive end-of-file marks. The addresses within these lists are in ascending order. If the index to the Disk Search File was selected, the

179

Merged Key-Address Tape:

| Key₁ | D | T | D | T | O | Key₂ | D | T | D | T | D | T | O | Key₃ | D | T | D | T | D | T | O | • • • |

Tape File Index:

(1) List-of-Addresses

| T | T | O | T | T | O | T | T | O | • • • |

A_1 A_2 A_3

A_i is the record (track) number and relative address of the list of addresses for key i

(2) Key-Address List

465 words

| Key₁ | A₁ | Key₂ | A₂ | Key₃ | A₃ | • • • • | Key₁₅₅ | A₁₅₅ | • • • • | Sentinel Key | Sentinel Address |

(3) INDX

| Key₁₅₅ | 1 | Key₃₁₀ | 2 | • • • • • | Sentinel Key | Last Track |

Last key on track 1 of the Key-Address List
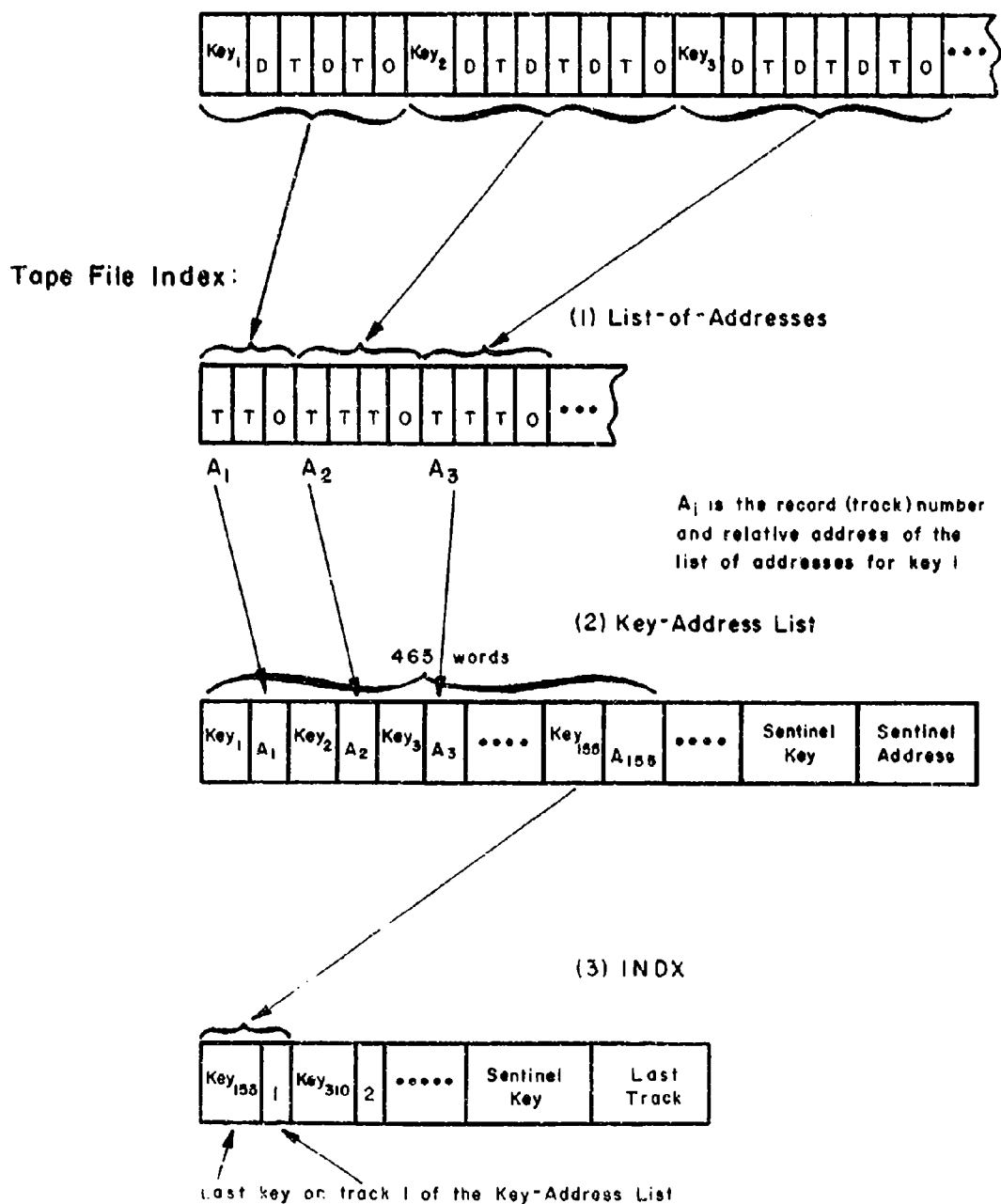
Figure 47.   Construction of Three-Level Index

180

format of each address is:

        Bits (s,1-17):   Track No. (1-)
             (18-35):   Relative Address (0-464)

    If the index to the Tape Search File was selected, the format of each
address is:

        Bits (s,1-5):   Tape No. (1-)
             (6-18):   Record No (0-)
             (19-35):   Relative Address   (0-)

    The <u>Key-Address List</u> contains each key (2 words), coupled with the
address of its corresponding list on the List-of-Addresses tape. The format
of this address is:

        Bits (s,1-17):   Track No. (0-)
             (18-35):   Relative Address   (0-464)

    Since each logical record is three words, there can be as many as 155 keys
on a track. The last key on this file is a special sentinel key (all 1 bits)

    Each Key-Address List tape is terminated by an end-of-file mark and a
dummy block except that the last tape is terminated by two consecutive
end-of-file marks. The third file, INDX, is then placed on this tape, directi
following the two end-of-file marks. This file is always small enough to plac·
on tape in one block, because 1000 words would accomodate a file containing
over 50,000 different keys. This block is then followed by an end-of-file
mark.

    The last key to appear on INDX will naturally be the sentinel key. Since
the last track on the Key-Address List may not be completely filled, the
address corresponding to the sentinel key will not necessarily imply word 4r·
on that track. The logical record format of INDX is·

        Word 1:   Key (1st half)

        Word 2:   Key (2nd half)

        Word 3:   Track (1-), contained in bits (s,1-17)

    Program INDEX also provides a printer listing consisting of each ke
present in the Search File and its number of occurrences in the file (the
number of times the key was assigned). The relative address of the list of
Search File addresses corresponding to each key is also provided. This ke·
listing is a helpful tool in analyzing the usefulness of the keys in the
system and to some extent the nature of the compounds in the file. It pro
vides to the querist an upper limit on the number of responses he can expect
from a query. It can also be an aid in planning search strategy, in re-
ducing the number of keys that must be utilized for optimum retrieval fr
a query

2.5.4.3 Operator Instructions

The Merged Key-Address tape(s) are loaded alternately on S.SU10 (B1) and S.SU19 (C6). The List-of-Addresses tape(s) are loaded alternately on S.SU04 (C4) and S.SU06 (C5). The Key-Address List tape is loaded on S.SU05 (B5).

A card with the word TAPE or DISK punched in columns 1-4 must be placed at the end of the program deck. This card determines whether an index to the Tape Search File or the Disk Search File is to be created.

The typewriter console will request that the number of input reels be keyed in.

3. FILE SEARCH AND RETRIEVAL

This section describes the programs which accomplish the actual file search and retrieval portion of the CIDS system. This process includes three separate and distinct actions, the input and preprocessing of the query, the file search and the output of retrieved records. There are two systems involved, the batch search system which is described by Figure 48 and the on-line search system which is described by Figure 49. These are similar in most respects and are described more fully in Section 1.

3.1 QUERY PREPROCESSING

The following programs read in the queries, do all the necessary translation from external query formats to internal formats, and intersect the lists of addresses for keys as specified in the query. They provide the accession list of actual records to be searched if the query specifies additional requirements that entail a molecular formula search or an atom-by-atom search of the connection table.
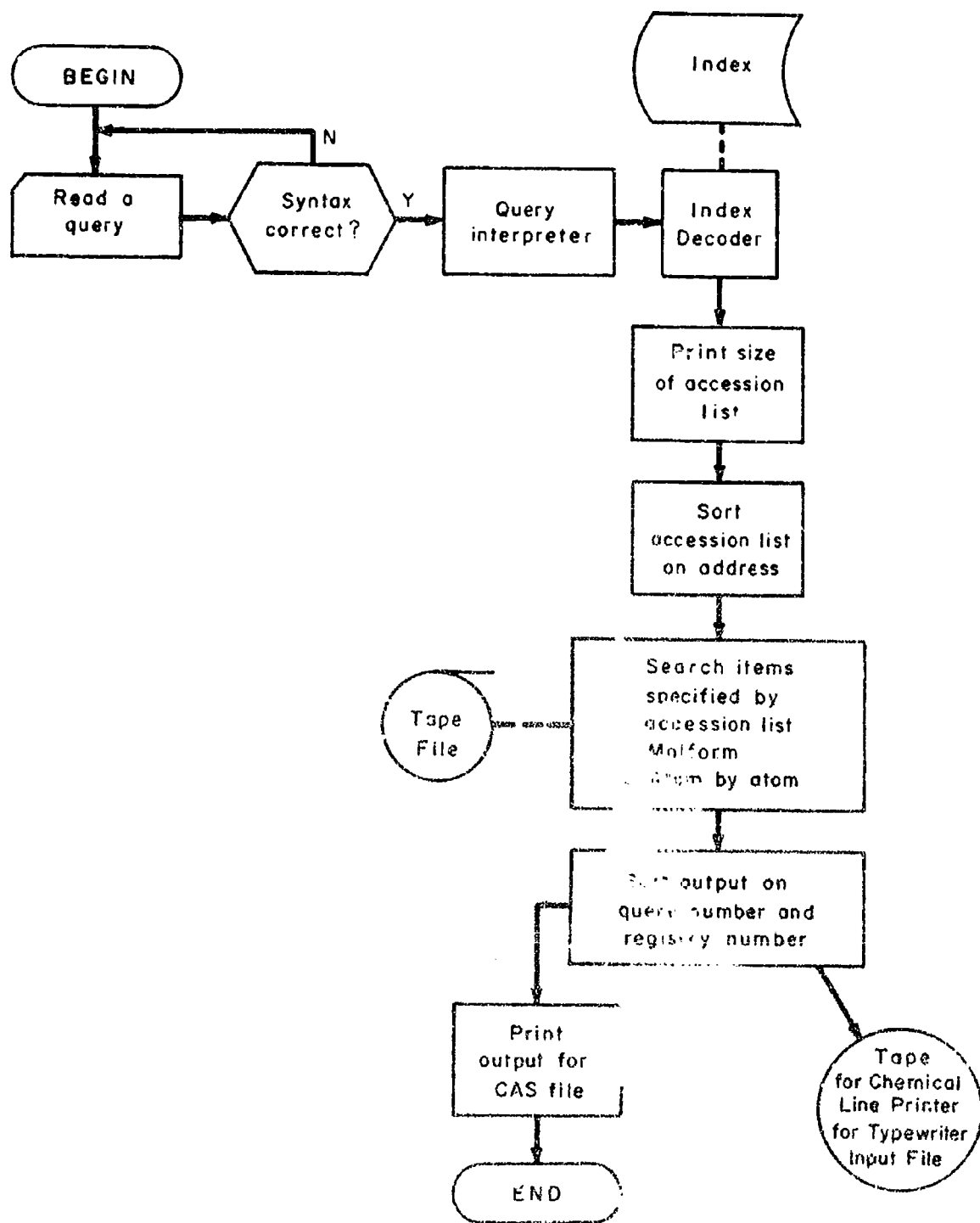
Figure. 48   Batch Search System
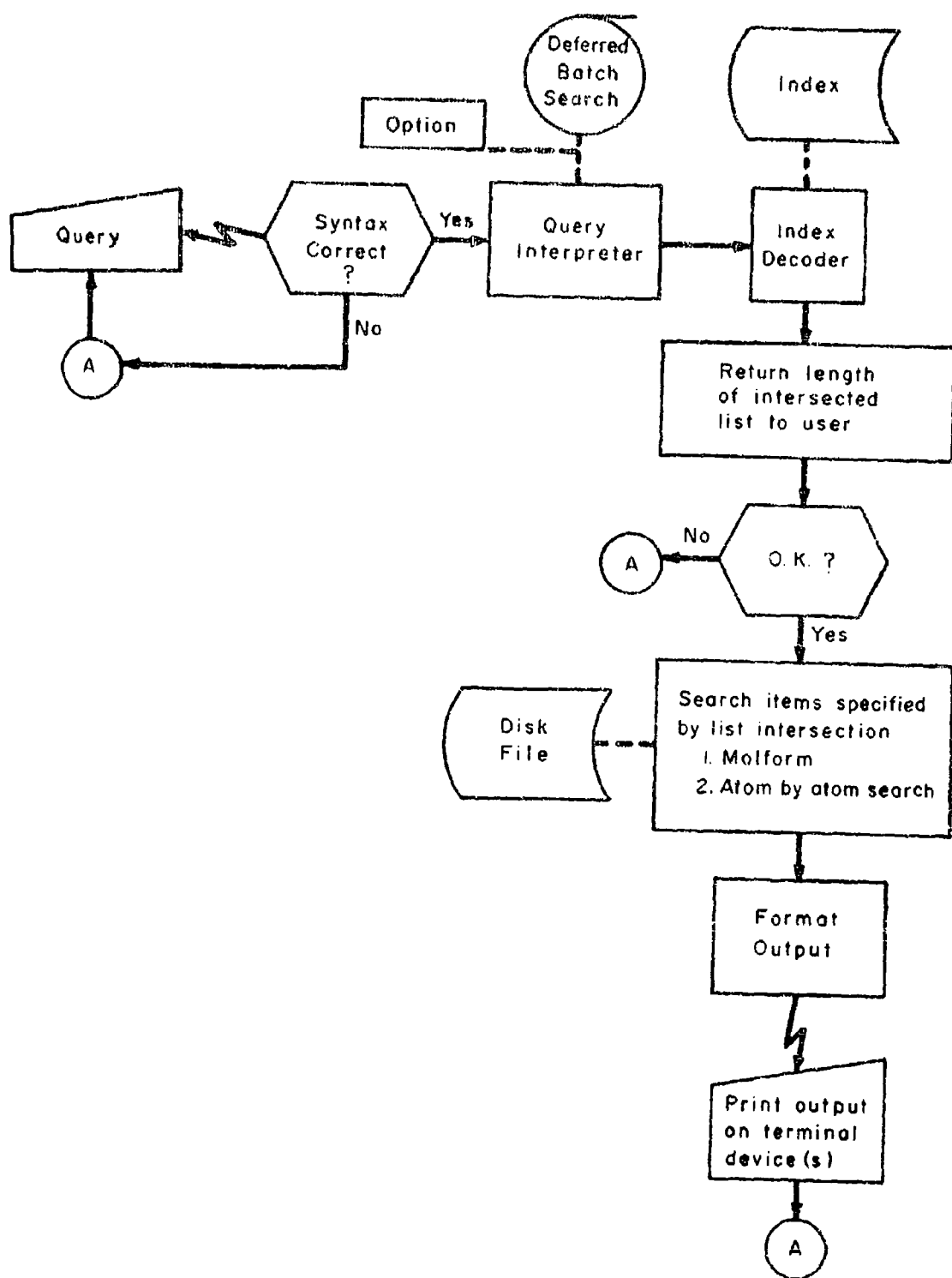
Figure 49. Real Time Search System

3.1.1 Query Input Executive

Code Name: INPUT

Programmer: Richard Haber

Abstract: Program INPUT is used to keep track of the number of queries correctly entered in the system. It also stores the disk address of each query in the query disk-core table.

3.1.1.1 Program Description

A macro flow chart describing this program is presented in Figure 50.

Program INPUT is first used to open the output file on which the accession list is written by program KIAD. It then gives control to program EXEC30 which is used to preprocess a query.

When control is returned from EXEC30, INPUT checks to determine whether the query has been accepted or not. If the query has been accepted, a word containing the number of queries is incremented by one, and the disk address of the query is stored in the query disk-core table. INPUT then determines if more queries can be preprocessed.

If more queries can be preprocessed or if the previous query has been thrown out, INPUT again gives control to EXEC30 to preprocess another query. When no further queries can be preprocessed, control is given to program READ which closes the output file on which the accession list is written.

The number of queries which may be preprocessed depends upon whether the system is operating in batch or real-time mode. At present, only one query at a time may be preprocessed (and then processed) when the real-time system is used. Up to 2000 queries may be preprocessed at a time when the system is operating in batch mode.

3.1.1.2 Program Structure

INPUT is initially given control by program MONITO. It normally calls EXEC30 which, in turn, calls READ. Control is returned to INPUT from READ via EXEC30 unless a $ (signalling the end of queries) is encountered in column one of any input line (either a line of teletype or a punched card). In this case READ retains control and INPUT is not reentered.
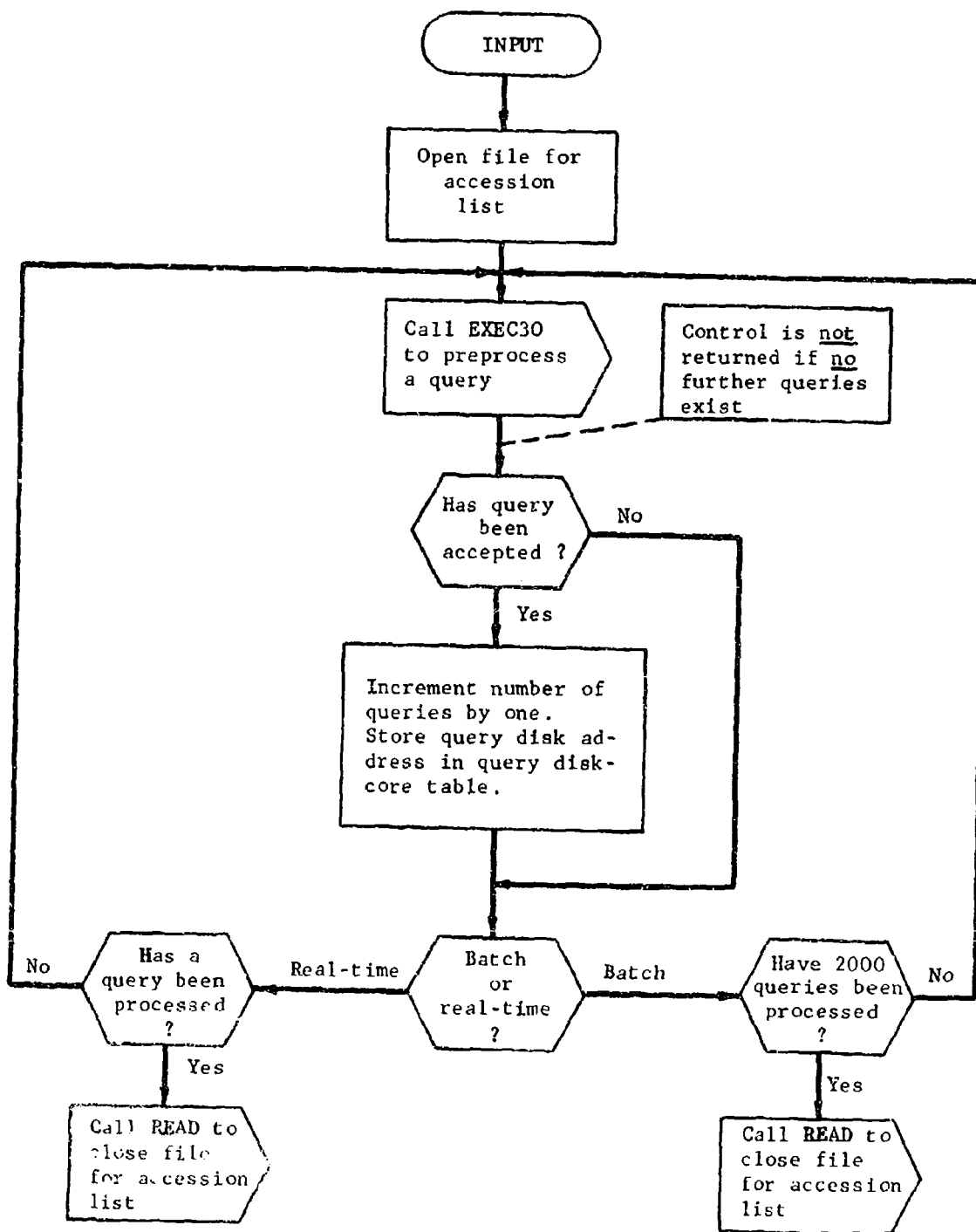
Figure 50. Macro Flow Chart - INPUT

187

3.1.2 Query Preprocessor

Code Name: EXEC3C

Programmer: Paul Weinberg

Abstract: The Query Preprocessor is a set of programs that scans the source text of a query presented by the user and translates it into the internal coding of the retrieval system. Queries are checked for syntactical errors and edited. In this role, the Query Preprocessor communicates with the other programs of the CIDS system to allow the system to adjust to particular user requirements.

3 1.2.1 Program Description

A macro flow chart describing this program is presented in Figure 51.

The entire preprocessor is arranged in one deck, EXEC30. Chains of core and disk storage are located in deck BUFFRS. There are three parts to the preprocessor.

To utilize storage in an efficient manner, a set of subroutines manipulates the lists of core and disk storage used by the preprocessor and real-time monitor. The technique involves chains of buffer control words, each control word representing a block of storage. The subroutines which perform these functions are:

> (1) POPTOP (CHAIN, HOL) removes the "top" buffer from the chain named CHAIN and presents the address of the control word for that buffer in HOL. If the chain is empty, 0 is returned to HOL.

> (2) ADDBUF (CHAIN, EOL) adds the buffer whose control word is indi cated by the address part of EØL to the chain named CHAIN

> (3) MVECHN (CHAIN, HØL) adds the entire list of buffers starting with the control word indicated in HØL to the end of the chain named CHAIN.

A number of macros is available to generate chain structures. A chain should be formed for each different size of buffers used. Chains are used to control disk as well as core storage pools.

A second set of subroutines (SCAN) is used to scan the query text, extract ing _symbols_ (strings of non-blank characters or a delimeter character) in order.

A third set of subroutines is included to interpret the query statements as they are scanned, and set up the block of information that will represent the query internally for the retrieval system.

The basic mechanism of the preprocessor is to scan the input text until a symbol has been collected. A table (known as the dispatcher or scan table) is then consulted to transfer to a routine which interprets the symbol. The

```
                                    ┌──────────────┐
                                    │   EXEC30     │
                                    └──────┬───────┘
                                           │
                                    ┌──────▼───────┐
                                    │ Assume init- │
                                    │ ial LOCATION │
                                    │ is 'start of │
                                    │ query'       │
              ┌ ─ ─ ─ ─ ─ ┐        └──────┬───────┘
                No more     │              │
              │ symbols              ┌─────▼──────┐
               ─ ─ ─ ─ ─ ─ ┘        │ Call 'SCAN'│◄──────┐
  ┌──────────────┐     │            │ to collect │       │
  │ Error-query  │◄────────────────┤ next symbol│       │
  │ terminated   │                  └─────┬──────┘       │
  │ abruptly     │                        │              │
  └──────┬───────┘                 ┌──────▼───────────┐  │
         │                         │ Check scan table │  │
  ┌──────▼───────┐                 │ to deter-        │  │
  │ Set error    │                 │ mine which routine│ │
  │ flags        │                 │ whould be entered │ │
  └──────┬───────┘                 │ to process the    │ │
         │                         │ symbol in the     │ │
  ┌──────▼───────┐                 │ current con-      │ │
  │   RETURN     │                 │ text (LOCATION)   │ │
  └──────────────┘                 └──────┬───────────┘  │
                                          │              │
                                   ┌──────▼───────────┐  │
                                   │ Execute the      │  │
                                   │ routine indi-    │  │
                                   │ cated. (The      │  │
                                   │ routine may      │  │
                                   │ call SCAN5 to get│  │
                                   │ further symbols).│  │
  ┌ ─ ─ ─ ─ ─ ─ ─ ┐               └──────┬───────────┘  │
    (Error if state-                     │              │
  │ ments specified │             ┌──────▼───────────┐  │
    in INPUT                      │ Change 'CURRENT  │  │
  │ statement       │             │ LOCATION' as     │  │
    not processed)                │ indicated in the │  │
  └ ─ ─ ─┬─ ─ ─ ─ ─ ┘             │ scan table (i.e.,│  │
         │                        │ Update assum-    │  │
         │                        │ ed context)      │  │
         │                        └──────┬───────────┘  │
         │                               │              │
  ┌──────▼───────┐        Y     ╱────────▼────────╲   N │
  │ Assemble the │◄────────────┤    'END'          ├────┘
  │ query from   │              ╲  statement ?    ╱
  │ the accumu-  │               ╲───────────────╱
  │ lated parts  │
  └──────┬───────┘
         │
  ┌──────▼───────┐              ┌──────────────┐
  │ Call KIAD to │              │ Find disk    │
  │ create and   ├─────────────►│ space (Sub-  │     ┌──────────┐
  │ merge accession             │ routine POP- ├────►│  RETURN  │
  │ list for the │              │ TOP) and     │     └──────────┘
  │ query        │              │ write the    │
  └──────────────┘              │ query on the │
                                │ disk for lat-│
                                │ er processing│
                                └──────────────┘
```

Figure 51.   Macro Flow Chart - EXEC30

189

scan table (Table I) lists different transfer points depending on the part
of the query in which the scanned symbol has been detected. Consulting this
table, for example, the symbol STRUCTURE causes the routine IS to be entered
if it is encountered while scanning an INPUT string but will cause the rou-
tine D.DFST to be entered if it is encountered while scanning a DEFINE state-
ment.

Each routine may call on SCAN to remove successive symbols from the input
text and process them. At its conclusion the routine returns to the main
scanning mechanism with a new assumed location for the input scan.

When an END statement has been encountered, the preprocessor assembles
the sect'ons of the query that have been collected from individual state-
ments. Routine KIAD is called to compute an accession list and the query
is formatted for searching. The query is then written onto a disk buffer
and control is returned to the search programs which will read the query back
in when searching starts.

The input accepted by the preprocessor is described in a separate doc-
ument describing the query language. The internal format for queries is out-
lined in Table II.

3.1.2.2 Program Organization

EXEC30 is a subroutine which is called by a standard CALL statement.
Its entry name is D.STRT.

Input to EXEC30 is the BCD query text stored in a buffer whose location
is given in D.INBF.

The output is the query in internal format stored on disk.
disk buffers holding output has the name OUCHAN.

The Query Processor generates the following messages:

1. ILLEGAL QUERY CXXXXX.

The first symbol in the query cannot be used as a query identi-
fier. 'C' identifies the console. XXXXX is the bad symbol. The
query is skipped.

2. UNABLE TO LOCATE QUERY

The query preprocessor cannot find a syntactically correct query
in the input text presented.

3. COMMENTS FOR QUERY CXXXXX.

A query has been found and comments regarding it follow. 'C'
identifies the console. XXXXX is the query name.

4. ASSUMED PRIORITY OF 0.

A priority other than 0, 1, ., or 3 has been specified

TABLE I — CONDENSED SCAN TABLE FOR QUERIES

CURRENT LOCATION OF INPUT SCAN	SYMBOL ENCOUNTERED	TRANSFER TO ROUTINE	PURPOSE OF ROUTINE	NEW LOCATION OF INPUT SCAN AFTER ROUTINE EXIT
START OF QUERY	REALTIME PRIORITY INPUT OUTPUT DEFINE STRUCTURE KEYS QUALIFIERS	D.NONM	IGNORE QUERY SINCE NO NAME HAS BEEN GIVEN	POSITIONED TO END OF QUERY. (ENTIRE QUERY SKIPPED).
	ANY OTHER SYMBOL	D.NAME	RECORD SYMBOL AS NAME GIVEN TO THE QUERY	HEADER DECLARATIONS I
HEADER DECLARATIONS I	REALTIME PRIORITY CHEMICAL BATCH TELETYPE EXACT SRCH NOSRCH	D.REAL D.PRIO D.CHEM D.BTCH D.TELE D.EXAC D.SRCH D.NOSR	SET OPTION FLAGS " " " " " " "	HEADER DECL. I " " " " " " "
	$	D.MAIN	FINALIZE OPTION FLAGS	HEADER DECL. II
	ANY OTHER SYMBOL	D.WAIT	PASS SYMBOL ON TO NEXT TEST	HEADER DECL. II
HEADER DECLARATIONS II	INPUT	D.INPT	ACCEPT USER SPECIFIED OUTPUT OPTIONS	INPUT STRING
	OUTPUT	D.OUPT	ACCEPT USER SPECIFIED OUTPUT OPTIONS	OUTPUT STRING
	ANY OTHER SYMBOL.	D.BODD	PASS SYMBOL ON TO NEXT TEST	START OF A STATEMENT
INPUT STRING	KEYS STRUCTURE FORMULA QUALIFIERS REGISTRY	IK IS IF IQ IR	FLAGS INDICATING STATEMENTS TO FOLLOW	INPUT STRING
	$	D.MAIN	TERMINATE STATEMENT	HEADER DECL. II
	ANY OTHER SYMBOL.	D.WAIT	PASS SYMBOL ON TO NEXT TEST	HEADER DECL. II
OUTPUT STRING	KEYS STRUCTURE FORMULA TIME	OK OS OF OTI	SET FLAGS INDICATING TYPES OF OUTPUT DESIRED	OUTPUT STRING
	$	D.BODY	TERMINATE STATEMENT	START OF A STATEMENT
	ANY OTHER SYMBOL.	D.WAIT	PASS SYMBOL ON TO NEXT TEST	HEADER DECL. II
START OF A STATEMENT	STRUCTURE	D.INST	INTERPRET STRUCTURE LOGICAL STATEMENT	START OF A STATEMENT
	KEYS	D.INKY	INTERPRET KEYS LOGICAL STATEMENT	START OF A STATEMENT
	QUALIFIERS	D.INQU	INTERPRET QUALIFIERS LOGICAL STATEMENT	START OF A STATEMENT
	FORMULA	D.INFM	INTERPRET FORMULA STATEMENT	START OF A STATEMENT
	DEFINE	D.PRDS	DEFINE SYMBOL	DEFINE STATEMENT
	END	D.TERM	CREATE AGGREGATION, TEST QUERY, ADD QUERY TO QUEUE	
	ANY OTHER SYMBOL.	D.WAIT	PASS SYMBOL ON TO NEXT TEST	START OF A STATEMENT
DEFINE STATEMENT	KEYS	D.DEKY	DEFINE SYMBOLS AND INTERPRET KEYS LOGICAL STATEMENT	DEFINE STATEMENT
	STRUCTURE	D.DEST	DEFINE SYMBOLS AND INTERPRET STRUCTURE LOGICAL STATEMENT	DEFINE STATEMENT
	ANY OTHER SYMBOL	D.WAIT	PASS SYMBOL ON	DEFINE STATEMENT

NOTE: ...

TABLE II   INTERNAL STORAGE FOR QUERIES (ON DISK TRACKS)

WORD NUMBER	S 2	3 17	18 20	21 35

1	MZE if only track
	
	PON if last track in query
	
	PTW if first track
	
	PZE otherwise
		OUTPUT ELEMENTS
		bit element
		3 Keys
		4 structure
		5 formula
		6 qualifiers
		7 tics
		8 statistics
		9 time
		
		INPUT INFORMATION
		bit
		10 1 if batch
		      0 if real time
		11 1 if chem TT
		      0 if TTY
		12 1 if structure is exact match
		13 PRIORITY
		14 "
		17 For No Atom By Atom Search
			# sequence
			. 1, 2...
			
			(note: first word repeated on each track)
				Disk positioning address of preceding track (77777 if none)

2	Console id (6 bit binary)	Query name (5 character alphameric)		
TABLE OF CONTENTS				
3	Track seq. number of end of keys	Word number within track of end of keys	Track seq. num. of start of keys	Word number within track of start of keys (0 if no keys are found or given)
4	Same as above for formula			
5	Same as above for structure			
6 thru 23	Track sequence number	Word number within track of start of nth structure	Track sequence number	Word number within track of start of (n + 1)st structure
24	Track sequence number	Word number within track of end of qualifiers	Track sequence number	Word number within track of start of qualifiers
25				Control word address of compound accession list
26			Track seq. num. of end of query	Word number of end of query
TABLE OF KEY DEFINITIONS				
27	Start flag 77 for each externally de-fined key 75 for each internally de-fined key	NAME GIVEN by user to key (5 character maximum, otherwise left justified)		Repeated for each key definition.
	Alphameric or internal coding for the key (may extend for many words)			
	Following keys . . .			
KEY MINTERM BIT INDEX				
	74 Flag word			
	Bit position (6 bits)	Key # (6 bits) # of occurrences	Bit position (6 bits)	Key # (6 bits) # of occurrences
		(repeated as many times as necessary)		
KEY MINTERMS				
	76 Flag word			
	Pairs of minterms of up to 36 literals. Bit 1 in first word of each pair indicates the presence of a literal. Bit 1 in second word is used to indicate negation. (Bit order determined by order of key definitions above).			

TABLE 11 INTERNAL STORAGE FOR QUERIES (continued)

WORD NUMBER	S	2	3		17	18		20	21		35

TABLE OF STRUCTURE DEFINITIONS

Start flag

77 for each structure

Name given to structure.
(5 character maximum)

Repeated for each
structure definition.

Connection table
(may extend for many words)

. .

. .
. .

Following structure definitions

.
.
.

Minterm Bit Index

74 Flag word

Bit position	Structure number	# of occurrences	Bit position	Structure number	# of occurrences
(6 bits)	(6 bits)	(6 bits)	(6 bits)	(6 bits)	(6 bits)

(repeated as many times as necessary)

STRUCTURE MINTERMS

76 Flag word

Pairs of words describing minterms (as with keys).

.
.
.

MOLECULAR FORMULA

Pze for Hill / MZE for Addend	length 'a'	bit 19 =1 iff restricted bit 20=1 iff last formula	length 'b'

repeat once
for each
addend

Formula Limit Words

Same format as described in CIDS No. 3

Constraint words

END OF QUERY

FORMAT OF CONSTRAINT WORDS

1	a	b	c	c	d

a. element to left of '=' sign
b. multiplier to a.
c. element to right of '=' sign
d. constant (right hand side multiplier to c.)

1

5. ILLEGAL STRING SENTINEL XXXXXX

    In looking for a statement head, the symbol XXXXXX has been en-
countered and is not a legal statement name. The symbol is ignored.

6. REPETITION TABLE OVERFLOW SCANNING $\begin{Bmatrix} \text{KEYS} \\ \text{STRUCTURE} \end{Bmatrix}$

    Too many (more than 36) defined keys or structures appear in a
logical statement.

7. UNDEFINED SYMBOL XXXXX DETECTED DURING SCAN OF LOGICAL STATEMENT

    $\begin{Bmatrix} \text{KEYS} \\ \text{STRUCTURE} \end{Bmatrix}$

    A symbol which has not been defined in a DEFINE statement has
been used in a logical statement. The symbol is ignored.

8. OVERFLOW OF MINTERM TABLE FOR $\begin{Bmatrix} \text{KEYS} \\ \text{STRUCTURE} \end{Bmatrix}$

    More than 463 'OR' connectives appear in a logical statement.
The statement is ignored.

9. SYNTAX ERROR IN KEY XXXXXX

    General purpose error message in program that converts CIDS keys
(appearing in DEFINE statements) to the internal format of the search
system.

10. SYNTAX ERROR-IGNORE XXXXXX

    Symbol XXXXXX appears out of place and has been ignored.

11. OVERFLOW OF DEFINITION AREA

    A slash is missing in a DEFINE statement.

12. SKIPPING FORMULA

    Used to indicate a syntax error in a FORMULA statement that has
caused it to be skipped.

13. SKIPPING TO END OF QUERY

    A catastrophic error in the specification of a query, typically
the omission of a logical expression for keys, has caused the entire
query to be skipped.

14. NO DEFINED $\begin{Bmatrix} \text{KEYS} \\ \text{STRUCTURE} \end{Bmatrix}$ SKIPPING LOGICAL EXPRESSION

3.1.3 Query Reader

Code Name: RFAD

Programmer: Richard Haber

Abstract: Program READ is used to read queries from an input device. It can be used to read either punched cards or punched paper tape produced by a teletype.

3.1.3.1 Program Description

A macro flow chart describing this program is presented in Figure 52

Program READ is called by program EXEC30 to read a query from an input device and place it in a buffer for processing by EXEC30. It can read either punched cards or punched paper tape, and will read an entire query provided the query is less than 465 words long.

When called by EXEC30, READ begins reading a query one line at a time. If punched paper tape is being read, a line is considered to be one line on the teletype which produced the tape. If punched cards are being used, one line is equivalent to one card.

The query is read, one line at a time, until one of the following three conditions occur:

(a) END occurs in columns 1-3 of a line. This signifies that the entire query has been read. The query is now in a buffer and control is returned to the calling program.

(b) 465 words of the query have been read. Since a single buffer contains room for only 465 words, control is returned to EXEC30 which prepares another buffer for the remainder of the query. READ is then called again by EXEC30 to continue reading the query.

(c) A $ occurs in column 1 of a line. This signals the end of all of the queries. In this case, a dummy record is written to end the accession list produced by program KIAD. The file containing the accession list is closed and is now ready to be sorted. The query preprocessing has been finished and the retrieval system is ready to begin searching the file.

3.1.3.2 Program Structure

RFAD is a subroutine called by EXEC30 as described above. It may also be transferred to by program INPUT when that program has determined that no further queries may be read. In the latter case, the accession list is terminated and the file closed as described above in case (c), and the retrieval system is ready to search the file.
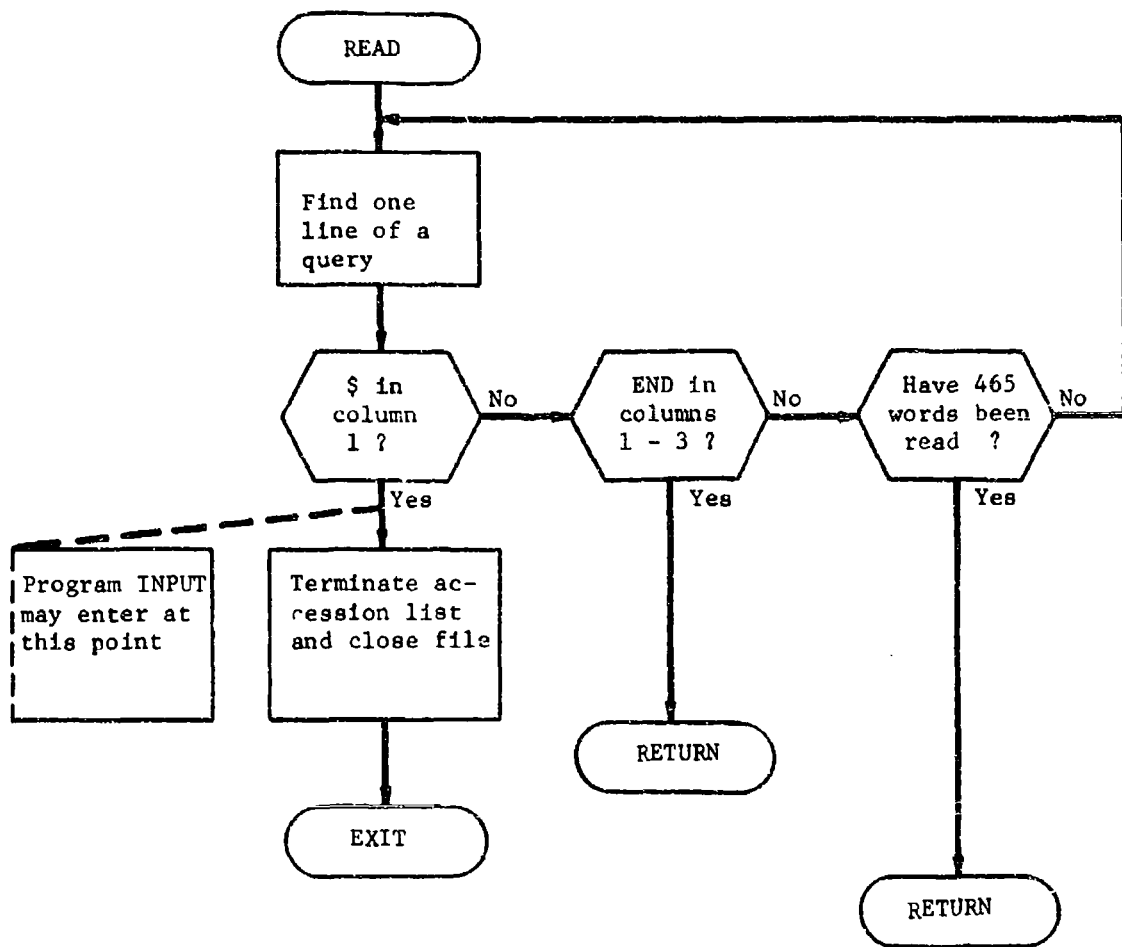
Figure 52. Macro Flow Chart - READ

Any number of END statements may be placed at the end (or the beginning) of a query. However, an error message will be given if the $ statement does not directly follow an END statement. The last query will not be preprocessed correctly in this case, but no previous queries will be affected.

3.1.4 Key-Expression to Accession List Processor

Code Name: KIAD

Programmer: James W. Gerber

Abstract: This program accepts the boolean key expression as input and produces the list of all compound record addresses that pass the key expression (the accession list.)

3.1.4.1 Program Description

A macro flow chart describing this program is presented in Figure 53.

KIAD (Key-Index Address Decoder) is a subroutine which produces the accession list for a query. When it is called the accumulator contains the following information:

  bits 21-35: address of first word of key definitions
  bits 3-17: address of last word of minterms

KIAD produces pairs of words on the output unit (defined in file definition card FILE1) with the following format:

  Word 1: Query number (internal) from external location INDEX
  Word 2: Tape or disk address

The file is opened and closed by the search executive.

KIAD uses the index produced by program INDEX. This index is in three parts. The first part, the list of addresses, is loaded onto S.SU15 (disk) and the second, the Key-Address list, is loaded onto S.SU14. The third part of the index must be loaded into core memory at location INDX. This table is 1000 words or less. Table INDX is external to KIAD.

There are two versions of program KIAD at present. The only difference between them lies in the handling of the output file definition. The tape system version has the file defined with a FILE pseudo-op card in KIAD. In the Pseudo-realtime system (disk system) the file definition is contained in a $FILE card in the main link and is thus external to KIAD.

KIAD makes use of the routines: ADDBUF, POPTOP, MVECHN. These routines allow dynamic disk-buffer storage allocation and are used to produce intermediate storage of partial results. The name of the free storage chain is (external) DISKBF. This chain should contain enough buffers to accept the longest list produced by KIAD. This list length is the longer of:

  (1) The length of the longest of the accession lists produced by the first literal in each minterm

  (2) The total length of the accession list on exit from KIAD.

The input format is described in Table II.
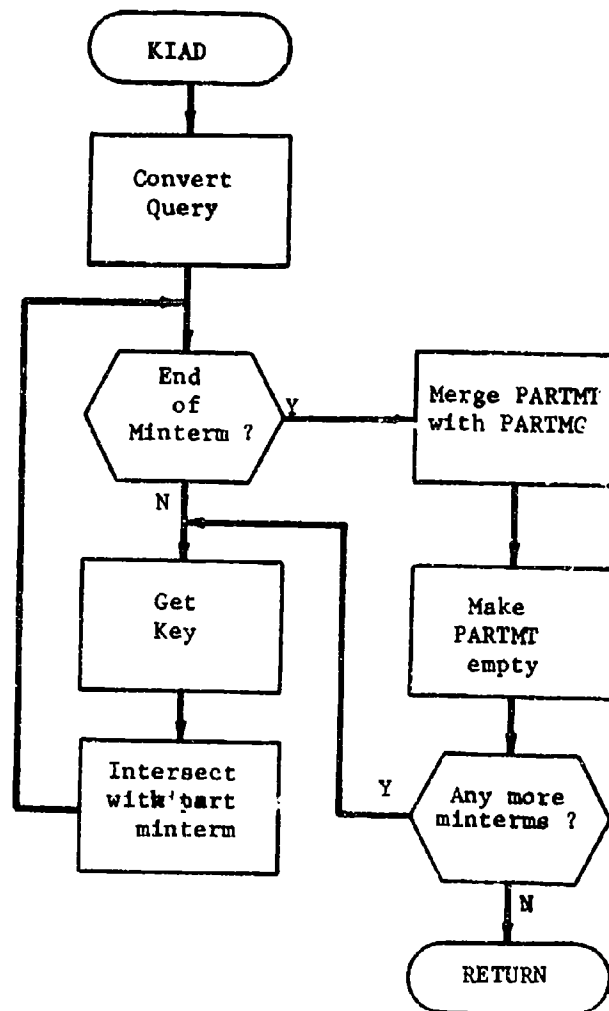
Figure 53.   Macro Flow Chart - KIAD

KIAD first translates the Key Definition table as follows:

1. The word in core following the last minterm is replaced by 0
   (This word will be restored before KIAD returns to its caller)

2. The list of key definitions is transmitted to KEYLST

3. The Key Index is unpacked from 2 to a word representation to
   1 to a word and the result is transmitted to KEYIND

4. Pointer is set up which points to the minterms in location
   MINTRM.

KIAD then proceeds to process the query. Each minterm is handled
separately and the result is merged with the list from the previous minterm
(if any) by routine MERGE. KIAD intersects lists within each minterm by
calling routine INTSEC. INTSEC examines location LITSGN to determine the sign
of the current literal. LITSGN will be plus if the literal is nonnegated.

The result of processing a minterm is left in disk buffer PARTMT. During
intersection, the new result is in buffer chain NWPTMT. MERGE accepts the
results of the minterm in PARTMT and the list in PARTMG and produces a list
containing the union of the addresses of both lists in PARTMG. It uses NWPMG
during merging to store the intermediate result. INTSEC accepts the list in
PARTMT and intersects it with the list for the key whose location is in
location LISTAD. INTSEC expects to find the number of occurrences for the key
in location NUMOCC. This will be set by GETKEY, the routine that finds a
key in the index and sets up a pointer to it in LISTAD. GETKEY will print a
message if the key is not in the index (i.e. it has not been assigned to any
compound on file. The processing of the query will continue as if the list
for that key were present but had no items in it. If the key appears non-
negated in a minterm, that minterm will produce no items for the accession
list. If the key appears negated, it will have no effect on the items in the
accession list.

No minterm may consist only of negated literals. If such a minterm is
encountered by KIAD, it will be treated as if its first key (leftmost bit in
minterm word) were nonnegated. Thus, the following query:

> NOT K1 and NOT K2 and NOT K3

Would be translated as:

> K1 and NOT K2 and NOT K3

The order of negated and nonnegated literals in the query is unimportant.

KIAD treats the multiple occurrence of a key to mean:

> "n or more of ...key"

> thus, NOT 5K5 means:

> NOT 5 or more of K5 (i.e. none, one, two, three, or four of this key).

The only limitation on the length of the query is that only 36 key definitions·
are allowed and that the minterm list be on one disk track (i.e. that it all be
in core memory at once when KIAD is called). This limits the number of minterms
to a maximum of approximately 175-200. There is no limitation to the length of
address lists that KIAD can handle except for the number of buffers available
in the disk buffer chain DISKBF at the time KIAD is entered. KIAD returns all
buffers used to the free chain as soon as they are not needed by the program.
On exit, KIAD has returned all buffers used during its processing.

3.1.4.2   Program Structure

KIAD is a subroutine whose input is the key expression part of the query
(see query internal formats Table II).

The output is the accession list.

3.1.5 Key Packing Program

Code Name: PACKEL

Programmer: James W. Gerber

Abstract: This program translates the individual key names from query language format to internal format.

3.1.5.1 Program Description

A macro flow chart describing this program is presented in Figure 54.

PACKEL is a subroutine. When it is called, the accumulator contains the location of the first word of the key definition as scanned by the executive and the number of words in the key definition. PACKEL returns the key in the accumulator. If a syntactical error is found in the key definition, the accumulator is set to zero.

The possible key formats for input to PACKEL are as follows (It is to be understood that the key in core has been scanned by the standard CIDS input manner):

FRAGMENT KEYS: FRAG aaaaaa bbbbbb or FRAG aaaaaa

aaaaaa is the first part of the key and bbbbbb is the second part of the key and may be omitted if zero.

RING MOLFORM KEYS: RINGMF at count at count ...
SKELETON MOLFORM KEYS·   SKELMF at count at count ...

at is the element kind and is either one or two characters.
count is the number of that element and must be explicitly mentioned even if one. At least one space must follow the element kind and there must be at least one blank between each element kind which follows.

There may be only three elements mentioned which are not one of these:

   C, N, O, S, P.

The total count of all elements not specifically mentioned in RINGMF's and SKELMF's as specified hetero elements should be included in the count under the element kind UH (unspecified hetero). UH counts as one of the three elements mentioned above. There is no restriction on the order of elements in the input, except that C, N, O, S, P must precede other elements. PACKEL reorders the elements to correspond with the assigned order.

REDUNDANT NUMERIC RING POPULATION KEYS: RNRP n_1 or RNRP n_1, n_2, n_3,...n_m
where m is less than or equal to 17. Each of the n's may be any number greater than or equal to 3. If a number greater than 17_8 (15_{10}) is mentioned, it will be translated into the number 1 which is the code used for rings with more than 15 atoms. The order of the n's should be increasing (i.e. n_k is less than or equal to n_{k+1}).
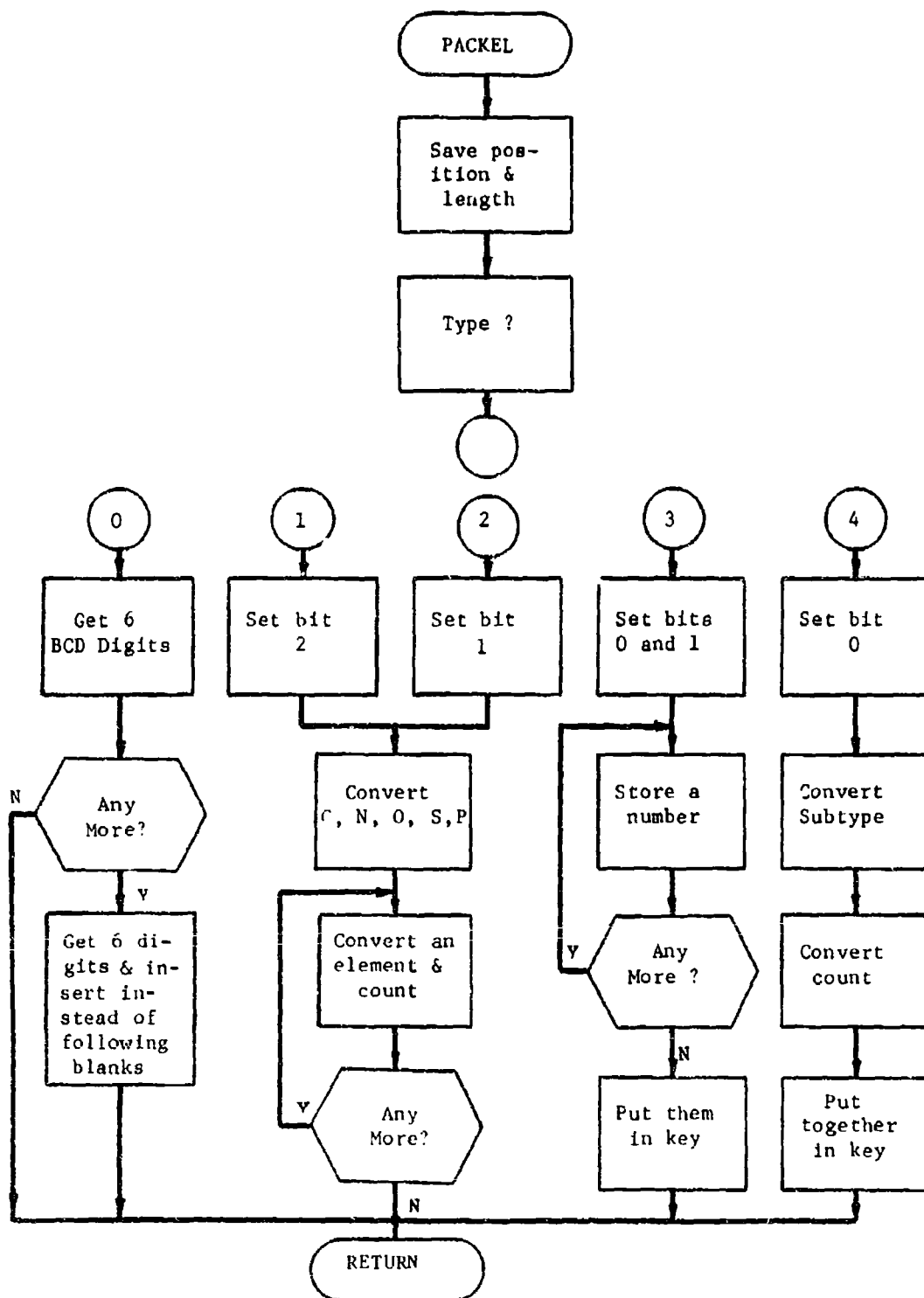
Figure 54. Macro Flow Chart - PACKEL
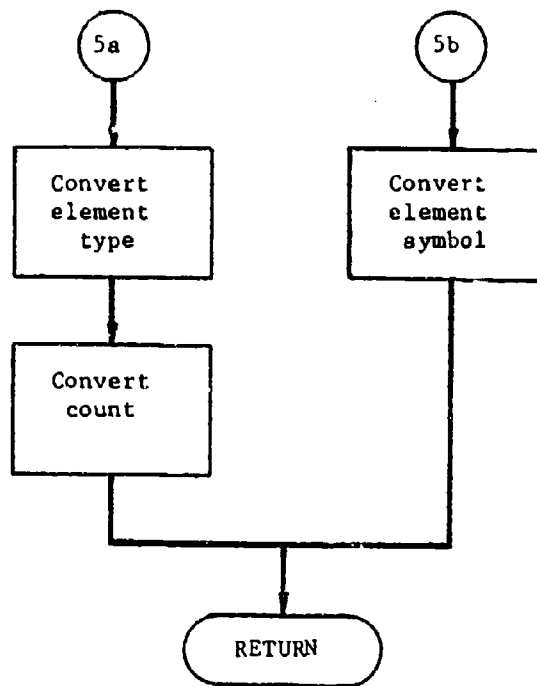
Figure 54.   Macro Flow Chart - PACKEL (continued)

atoms. The order of the n's should be increasing (i.e., n_k is less than or equal to n_{k+1}).

COUNT KEYS: NUMBER s count

s is the subtype. It must be explicitly mentioned, even if 0. It is one digit except for subtype 10. count is the number of this particular feature and should be typed with leading zeros omitted. A count of 0 should be explicitly mentioned.

MOLFORM KEYS: MOLFRM el count

el is the element kind and is one or two characters.
count is the number of that element and is typed with leading zeros omitted. A count of 0 is used only for those elements which do not have molform keys assigned for specific counts but have keys assigned indicating only the presence of the element.

NONSPECIFIC KEYS: NONSPC el

el is the element kind and is one or two characters.

EXAMPLES:

    FRAG 01I001   FRAG 01I001   FRAG 01A015 000001

    RINGMF C 4 N 1 HG 1

    SKELMF C 22 N 3 HG 2

    RNRP 6   RNRP 5,6,6,6

    RNRP 6,20   RNRP 6,69   RNRP 6,999999 (all these will be*
    translated the same as RNRP 6,1 would have been translated)

    NUMBER 4 0   NUMBER 10 0   NUMBER 0 5

    MOLFRM C 12   MOLFRM H 70   MOLFRM BE 0

    NONSPC TE

3.1.5.2 Program Structure

PACKEL is a subroutine whose input is the key in query language as scanned by the input scanner and whose output is the key in internal format.

---

*note that replacing a number greater than 15_{10} with 1 in the input will not cause any change in the translation scheme but it is not recommended since it is not as mnemonic.

3.1.6 Connection Table Processor

Code Name: MOLE

Programmer: Peter J. Brown

Abstract: Program MOLE processes chemical structural data in the form of manually generated connection tables. The data is converted into an intermediate format which is then compressed into the CIDS internal connection table by program CONVRT.

3.1.6.1 Program Description

The structural information is edited into three separate lists which serve directly as input to CONVRT. The format of these lists and of the connection table produced by CONVRT is fully discussed in the documentation of that program. MOLE does little error checking, and if it does encounter a symbol which it does not recognize, or which is out of place, control is returned to the calling program with minus zero stored in location MOLE.

A molecular formula is created from the structural information and this is output along with the connection table. Abnormalities, such as charge, unusual valence, and mass, are indicated in the input structure, and these are reformatted into a coded abnormality table. The actual format of this is discussed below.

3.1.6.2 Program Structure

Program MOLE is a subroutine which is called by EXEC30 to convert a users structural connection table to the internal format. It is a subroutine of the Retrieval System.

The structural information which is the input to MOLE is provided in BCD as it was punched on cards after being pre-processed by program SCAN. (Section 3.1.2.1) An asterisk may appear between the bond and the connection to indicate that the bond is in a ring. Abnormalities are set off by parentheses. All blank characters are ignored. The following example will illustrate this:
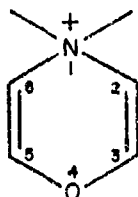
1N1C-10-1*2-1*2-1*6.2C1*1-2*3.3C2*2-1*4.401*3-1*5.5C1*4-2*6 6C2*5-1*1.
(V1=5.C1=+1,)

206

MOLE must be given the core location of this data and the length of the array. These are to be provided by the calling program in the address and decrement portions of the accumulator, respectively.

MOLE generates the following block of data containing the molecular formula, connection table, and abnormality table:

Word	Contents
1	A=No. of words preceding the C.T.(X+2)
	D=Total number of words (X+Y+Z+2)
2	A=No. of words in the C.T. (Y)
	D=No. of words preceding the Abnormality Table
	(X+Y+2 or 0 if no abnormalities)
3	Molecular Formula
.	
.	(X words)
X+2	Connection Table
	(Y words)
X+Y+2	Abnormality Table
	(Z words)

The location of this block is stored in the accumulator, bits (21-35), with the length in bits (3-17), when control is returned to the calling program.

The molecular formula is stored in the same format as the Hill formula for a file compound.

The Abnormality Table will consist of a series of words, where each word contains information about one atom which has an "abnormality", either charge, mass, valence, or attachments. The format of these words is:

Bits	Contents
(S,1,2)	Type of Abnormality:
	101=charge
	110=mass
	111=valence
	100=attachments
(3-17)	Atom number

Bits	Contents
(18)	1 if negative abnormality value (e.g., a negative charge). Otherwise 0.
(21-35)	Value of abnormality: mass, valence, signed charge, or number of attachments.

A word of zeros follows the last abnormality word. This is included in the length of the output buffer.

3.1.7 Molecular Formula Translator

Code Name: MOPACK

Programmer: James W. Gerber

Abstract: This program translates the query molecular formula to internal query format. It checks the query molecular formula for syntactical errors and indicates these to the calling program.

3.1.7.1 Program Description

A macro flow chart describing this program is presented in Figure 55.

Mopack translates the molecular formula from external format to internal format. The external format is contained in EXEC30 described in Table II and the internal format is contained in MOLFRM described in Section 3.2.2.2.

3.1.7.2 Program Structure

Mopack is a subroutine which takes as input the molecular formula as scanned by the query scanning program.

The input to MOPACK is accomplished by the routine external to MOPACK called SCAN69. This routine is called to obtain a scanned word. Each time SCAN69 is called it must return with the next scanned word in the accumulator. MOPACK will return the following output:

(a) Accumulator sign positive and molform in BUFF if molform is found to be correct.

(b) Accumulator sign negative if a syntactical error is found. In this case MOPACK will print the message SYNTAX ERROR before returning.

MOPACK uses external routines SCAN69 and BCDBIN as well as JOBOUL. Entry points for MOPACK are MOPACK and BUFF.
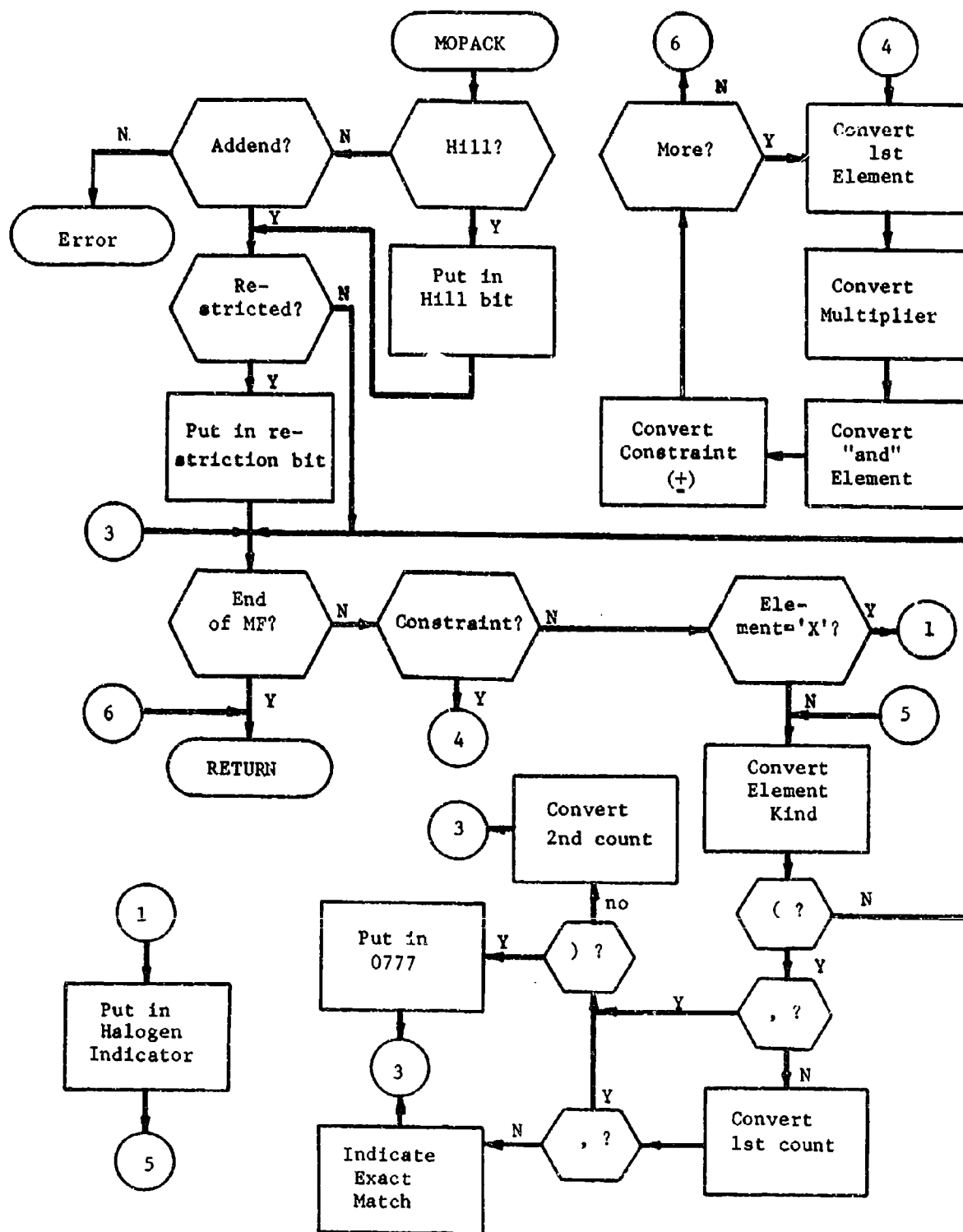
Figure 55. Macro Flow Chart - MOPACK

210

3.2 FILE SEARCH

The programs described in this section perform the actual search of thos
compound records which respond to the key requirements of a query. These pro-
grams determine if the molecular formula or structural requirements which ha···
·een specified by the query are satisfied.

3.2.1 Search Executives

Code Name: TAPE, TXINFO, DISKTT

Programmer: Richard Haber

Abstract: These programs are used to retrieve compounds from a file stored either on tape or on disk. With the aid of the molecular formula search and structure search programs, the executives determine which of the selected compounds actually satisfy the requirements of various queries. Output programs are then called to print the resulting compounds.

3.2.1.1 Program Descriptions

A macro flow chart describing this program is presented in Figure 56.

Three slightly different search executive programs exist:

(a) Program TAPE is used to search a file of compounds stored on a series of magnetic tapes. The output, consisting of query numbers and compound registry numbers, is sorted by query number and printed on a line printer.

(b) Program TXINFO is used to search a file of compounds stored on a disk unit. The output, consisting of query numbers, compound registry numbers, molecular formulas, and structural diagrams, is punched on a paper tape. This tape can then be read by a Dura Mach Chemical typewriter and the results printed.

(c) Program DISKTT is identical to TXINFO except that the output consists of query number, registry numbers and molforms and is printed directly on a teletype. Structural diagrams are not part of this output.

Each of the executives begins by obtaining (from a tape) the query disk-to-core table giving the location of each query stored on the disk.

The executive then obtains (also from a tape) the merged accession list. This is a list of pairs of words containing query numbers and compound addresses. This list has been sorted by compound address, and is used by the executive to determine which compounds must be further processed as possible retrievals to the queries.

The accession list is passed through twice. On the first pass, each query number is checked to determine whether the query has been read into core from its storage location on disk. Queries which have not previously been entered into core are entered at this time until the area set aside for them is filled. When a query is entered into core, its core location is stored in the query disk-to-core table.
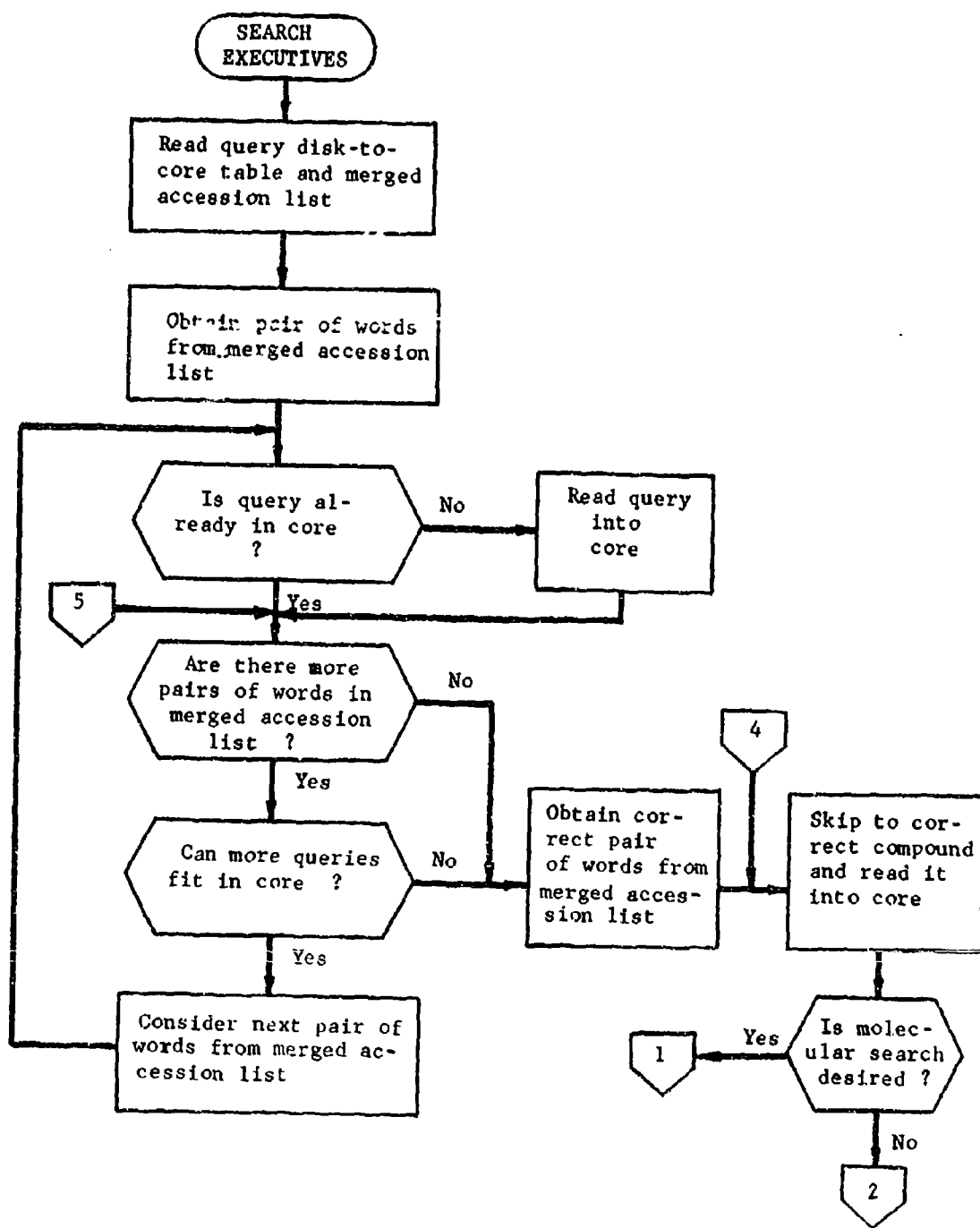
Figure 56. Macro Flow Chart - SEARCH EXECUTIVES
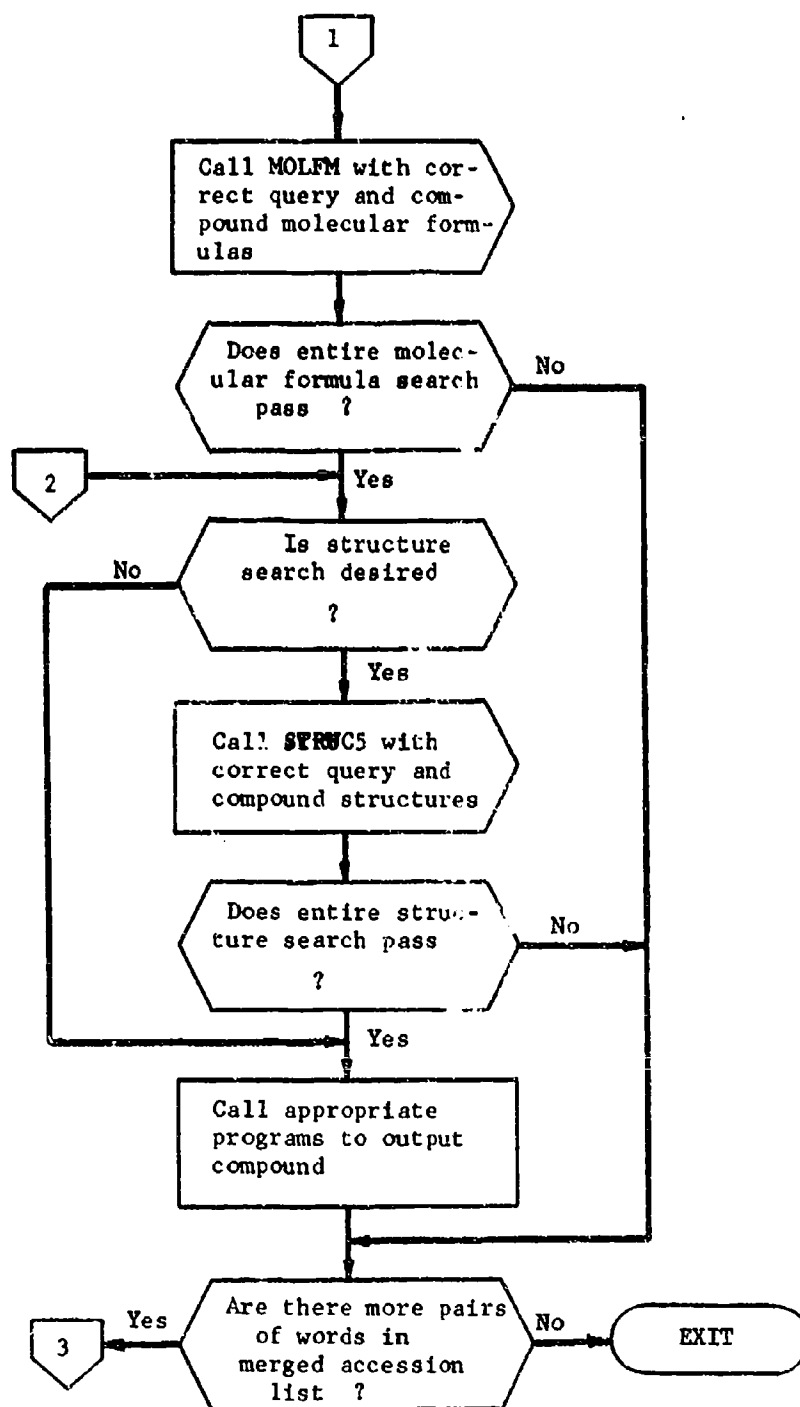
Figure 56. Macro Flow Chart - SEARCH EXECUTIVES
(continued)

214
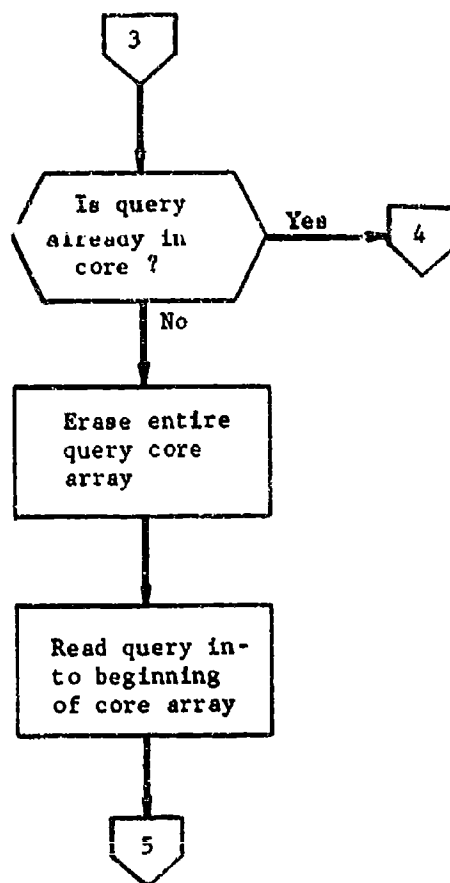
Figure 56.   Macro Flow Chart - SEARCH EXECUTIVES
(continued)

When the query array has been filled, cr when the accession list has been passed through to completion, the executive begins to pass through the accession list again. From each pair of words, a compound address and a query number are obtained. The compound address is used to position the storage device containing the file to the record containing the compound of interest. This record is read into core to be searched on the basis of the further specific requirements of the query. The compound addresses are in ascending order on the accession list in order to insure that the file tapes may be read without having to back up.

The query number is used to obtain the correct word of the query disk-to-core table. From this word the location of the query in core is found. The compound just obtained is tested to determine whether it satisfies the requirements of the query as described below. While this searching proceeds, the next pair of words from the accession list is obtained in order to locate and read the next compound to be tested.

The query is first checked to determine whether a molecular formula search is desired. If it is, the locations of the correct query and compound molecular formulas are obtained and control is given to program MOLFM to perform the search. Either the entire molecular formula (Hill molform) or addend molecular formulas, or both, may be searched. The addend molecular formulas do not have to be in the same order in the query and compound in order to pass. Only one query molform and one compound molform are given to MOLFM at a time. Thus, if addend molecular formula searches are desired, each query molecular formula will be searched against each compound addend molecular formula in order to determine if a match occurs.

A query may include constraint equations with each molecular formula. These allow the querist to use algebraic expressions in place of numbers in the molform. For example, the querist may require the following molecular formula:

$$C_n H_{2n+2} \qquad \text{where } n=2,3,4,5$$
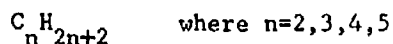
In this case, the constraint equation would be:

$$H=2C+2$$

and the number of carbon atoms would be required to be between 2 and 5.

Any constraint equations associated with a query molform are tested by the executive when MOLFM has indicated that the query and compound molecular formulas match. If the constraint equations are satisfied, the next query molecular formula is considered. If the constraint equations are not satisfied, or MOLFM has indicated that the query and compound molforms do not match, the next compound molform is tested against the current query molecular formula.

If the molecular formula search has passed the compound, or if no molecular formula search is desired, the query is tested to determine if a structure search is required. If it is, the locations of the compound structure and the correct query structure are obtained and control is given to program STRUC (Section 2.4.8) to perform the search.

216

More than one structure may appear in a query. These structures must be combined together in a disjunctive normal form logical expression. More than one occurrence of any structure may be required by the query. In addition, the absence, rather than than the presence of a structure may be desired. All of these conditions are handled in a straightforward manner by the executive, which decides whether or not the structure requirements of the query have been satisfied. If the requirements have not been met, the executive goes on to consider the next pair of words from the accession list.

If no structure search was desired, or if the structure search has been successful, the compound being searched is considered a successful retrieval to the query. In this case the executive either outputs the compound directly, or calls an output program to do so.

(a) Program TAPE writes the query name and compound number directly on tape. These records are later sorted by query number and printed by either program REGPRN or program EAPRN.

(b) Program TXINFØ calls programs MFØU, DURPIX, and DLPADK which format and output the query name, compound number, molecular formulas, and structural diagrams directly on punched paper tape. This tape, punched in DURA code, may then be printed on a Dura Mach chemical typewriter.

(c) Program DISKTT prints the query name and compound number directly on a teletype. DISKTT calls program MFØU which then prints the compound molecular formula on the teletype. A punched paper tape record of what has been printed may also be obtained.

After a compound has been output, the executive goes on to consider the next pair of words from the accession list. The accession list is passed through in the manner described above until either the entire list has been passed through, or a query which has not yet been entered into core is encountered. In the latter case, the executive again goes through the two part process of reading queries and then searching them against the compound file starting from where it had left off before.

When the entire accession list has been traversed and the search has been completed the executive returns control to the operating system.

3.2.1.2 Program Structure

The executive programs accept as input a file of compounds, a set of queries, a query disk-to-core table, and a merged accession list. The query disk-to-core table is used to give the locations of the query both on disk and, when applicable, in core. This table is read from a utility (either tape or disk) which was created by program READ during the query preprocessing.

The accession list is used by the executive to determine which compounds to search against each of the queries. It is read from a utility (either tape or disk) which was created by program KIAD and sorted by query number during the query preprocessing. For purposes of simplicity, it was implied in the

program description above that the entire accession list is read into core at one time. This can not be the case since the accession list may be quite long. Up to 920 words of the accession list may be in core at once. More is read as needed. The program, however, still functions in the manner described above.

The queries are stored on a disk utility by program EXEC30. Each query starts on a new track. The internal format of a query is shown in Table II.

The compound file is contained either on a series of tapes or on a disk utility. Due to its limited size, only approximately 40,000 compounds may be stored on the disk. These compounds are loaded onto the disk by a utility program prior to the running of the search system.

The compound file may be unlimited in size when it is contained on a series of tapes. The executive programs automatically switch from one tape to another. Mounting messages are printed on the on-line typewriter giving the operator the needed information as to which tapes to mount on which tape drives. The executive requires the mounting of only those tapes which are actually to be searched. Thus, in a particular run of the retrieval system, some of the file tapes may not even have to be mounted.

Program TAPE will print an error message for any of the following three conditions:

(1) A tape address is encountered which is smaller than a previous address. This would cause the tape being searched to be rewound.

(2) A tape address is encountered which is larger than the largest possible tape address for the file.

(3) An error occurred while skipping to the next compound to be searched.

Program TXINFØ and DISKTT will print an error message for either of the following two conditions:

(1) The occurrence of an error while skipping to the next compound to be searched.

(2) The occurrence of an error while reading into core a compound to be searched.

When any of the above errors occur, the search run is terminated after the error message has been printed.

3.2.2   Molecular Formula Search

Code Name: MOLFM

Programmer: Richard Haber

Abstract: Program MOLFM is used to determine whether the molecular formula of a particular file compound satisfies the requirements specified in a particular query.

3.2.2.1   Program Description

A macro flow chart describing this program is presented in Figure 57.

Program MOLFM is used to determine whether the molecular formula of a particular file compound satisfies certain requirements specified in a query. MOLFM is called by the search executive when a molecular formula search is asked for in the query.

Both the compound and the query may contain addends. Thus they may each contain more than one molecular formula. MOLFM is used to test one specific compound molecular formula against the requirements of one specific query molecular formula. The detailed formats of the molecular formulas are shown in Section 3.2.2.2.

When MOLFM receives control, it first checks to make certain that the query actually contains a molecular formula. If no molecular formula is present in the query, control is returned to the executive program and the search is considered to have failed.

If a molecular formula is present, the actual molecular formula search is then performed. All information specified in the query must be contained in the compound for the compound to be considered as satisfying the query requirements.

As shown in Figure 18 the number of carbon, hydrogen, nitrogen, and oxygen atoms are stored in the first word of the molecular formula of the compound. All other elements appear alphabetically in the following words. In the query the elements are arranged such that carbon is first, hydrogen is second, and the remaining elements are in alphabetical order.

Various types of searches are possible for each element in the query:

   a) Exact search. A test is made to determine whether the compound contains exactly the same number of atoms of the particular element as appear in the query. Any number of atoms between 0 and 63 may be requested for any element in an exact search. For oxygen, and nitrogen any number of atoms up to 127 may be requested, while the upper limit for carbon is 255 and for hydrogen is 511.

   b) Range search. The compound is checked for the presence of the element in question. When the element has been found, its atom count is compared with the limits appearing
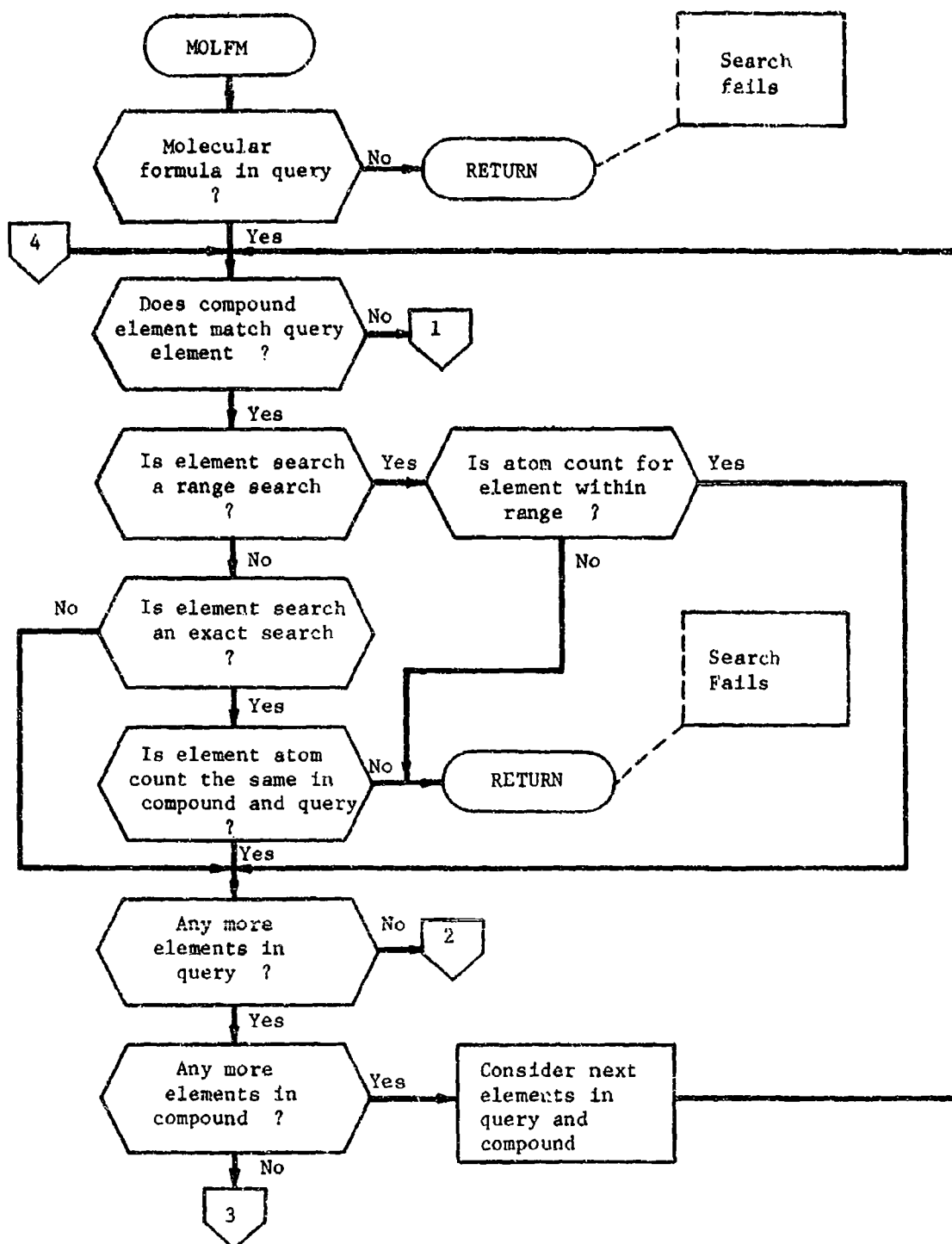
Figure 57. Macro Flow Chart - MOLFM

220

Figure 57.   Macro Flow Chart - MOLFM
(continued)

221
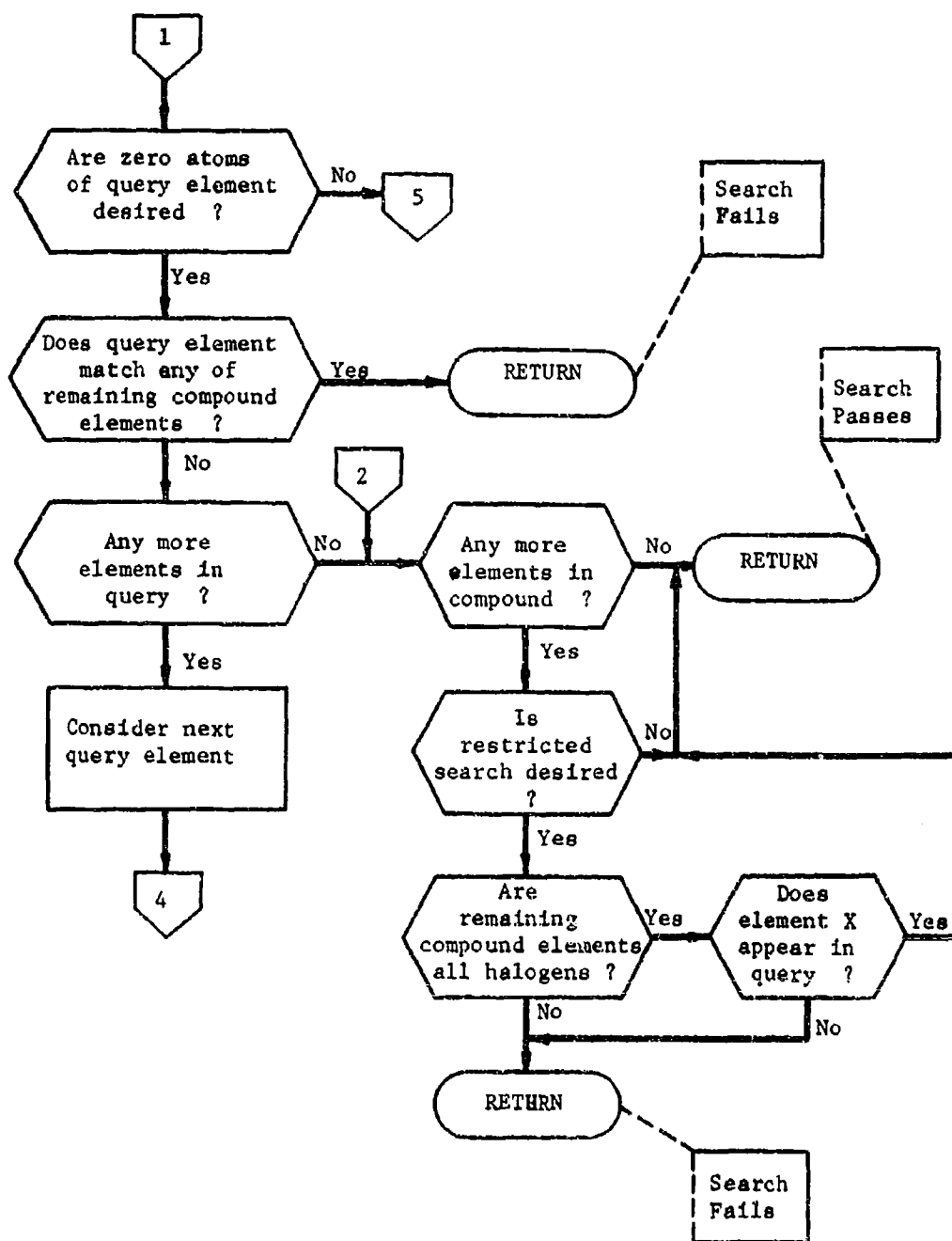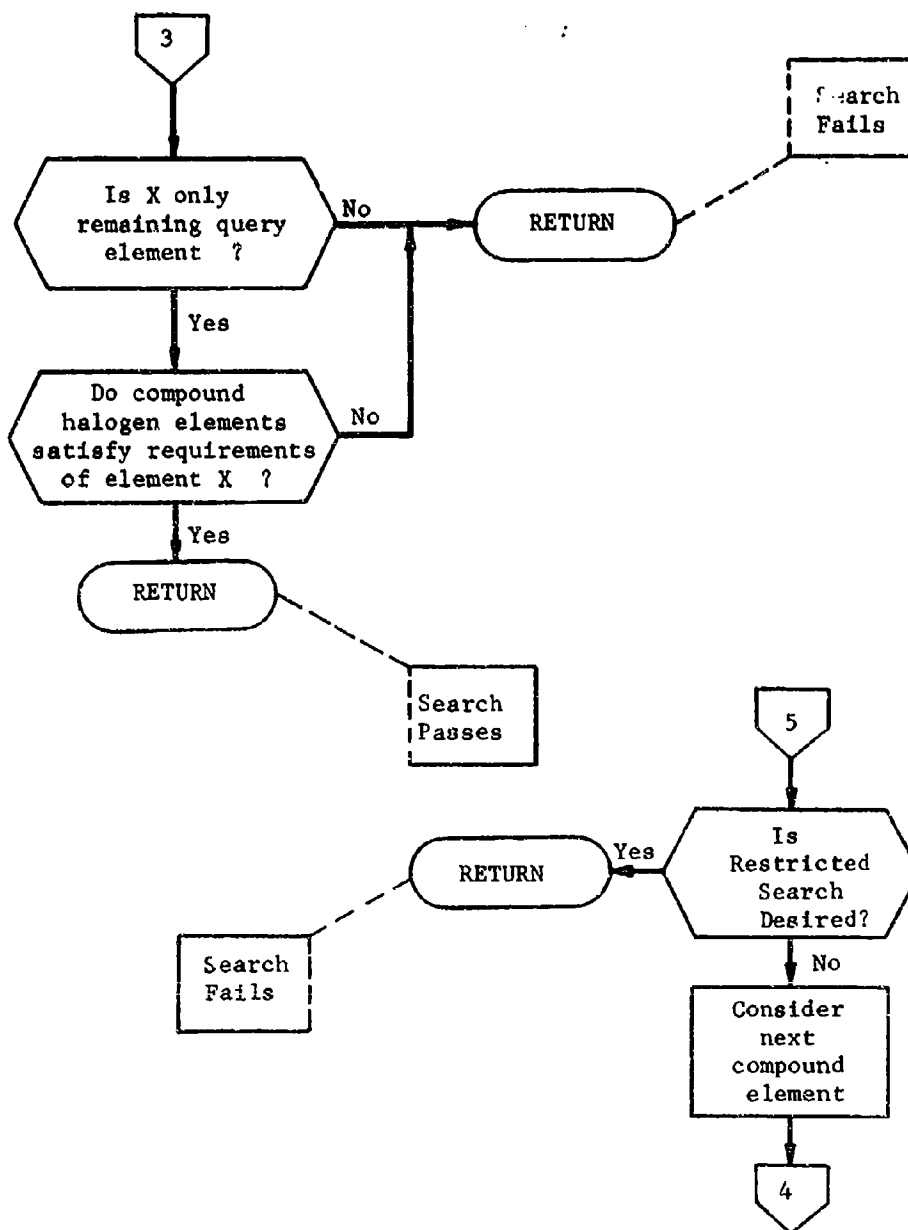
Figure 57. Macro Flow Chart - MOLFM
(continued)

in the query. For the compound to be considered
further, the atom count of the element in the compound must
be greater than or equal to the lower limit and less than or
equal to the upper limit. The same restrictions on the max-
imum number of atoms requested for each element apply here
also.

   c) A search to check for the presence of the given element
within the compound. In this case the number of atoms of
the element does not matter.

The symbol X is used as an element symbol in queries. It represents the sum
of all halogens (Br, Cl, F, and I) and is treated like any other element symbol
by MOLFM. For example, both C_2F_6 and $C_2Cl_2F_4$ would satisfy a query requiring
six halogens (X_6). $C_2Cl_2F_4$ would also satisfy a query requiring four fluorine
atoms and six halogens (F_4X_6) since the total halogen requirements is independ-
ent of the requirements placed on specific halogen elements.

MOLFM also has the capability of performing a restricted search. For a
compound to pass a restricted search, each of its elements must have passed one
of the three types of searches already mentioned. In other words, in order to
pass, a compound may contain no elements other than those listed in the query.

3.2.2.2   Program Structure

MOLFM is a subroutine which returns with the sign of the accumulator set
to plus if the search has passed and to minus if the search has failed. The
format of the molecular formula in the query is as follows:

Word of Formula	Bits	Contents
First Word	3-17	Number of words in molecular formula
	19	= 1 if restricted search = 0 otherwise
Additional Element Words	0-11	Element (BCD)
	12-20	Maximum number of atoms if range search; Number of atoms if exact search; = 0 otherwise
	21-28	Minimum number of atoms if range search =0 otherwise
	34	= 1 if search of atoms is a range search = 0 otherwise
	35	= 1 if search of atom is an exact match search = 0 otherwise

The format of the molecular formula in the compound is given in Section
2.2.4.

3.3 PRESENTATION OF RESPONSE

The programs in this section provide the output of both the batch and on-line query system.

3.3.1 Registry Number and Descriptor Print Program

Code Name: EAPRN

Programmer: Richard Haber

Abstract: Program EAPRN is used to format and print compound registry numbers and descriptors obtained as retrievals to queries searched against the tape file.

3.3.1.1 Program Description

A macro flow chart describing this program is presented in Figure ,8.

Program EAPRN is used to format and print registry numbers and descriptors of compounds retrieved by the batch search system. This information is contained on a system utility as a result of TAPE (Section 3.2 1.1) and has been sorted by query number.

EAPRN obtains the retrievals (query number, registry number and any associated descriptors) one at a time and prints the query number. One twelve character registry number is then printed on each line. It may be followed by 0 - 10 descriptors, each of which is preceded by the letters EA (standing for Edgewood Arsenal).

The number of retrievals obtained for each query is printed below the list of registry numbers for the query. A new page is used to list the answers of each new query.

3.3.1.2 Program Structure

EAPRN is an autonomous program which accepts 14 - word IOBS type J records from system utility unit 14. The first word of each record is considered to contain a query number, the next two, a compound registry number, and the remainder, any descriptors which may be present

EAPRN is terminated when an end-of file is encountered on the utility
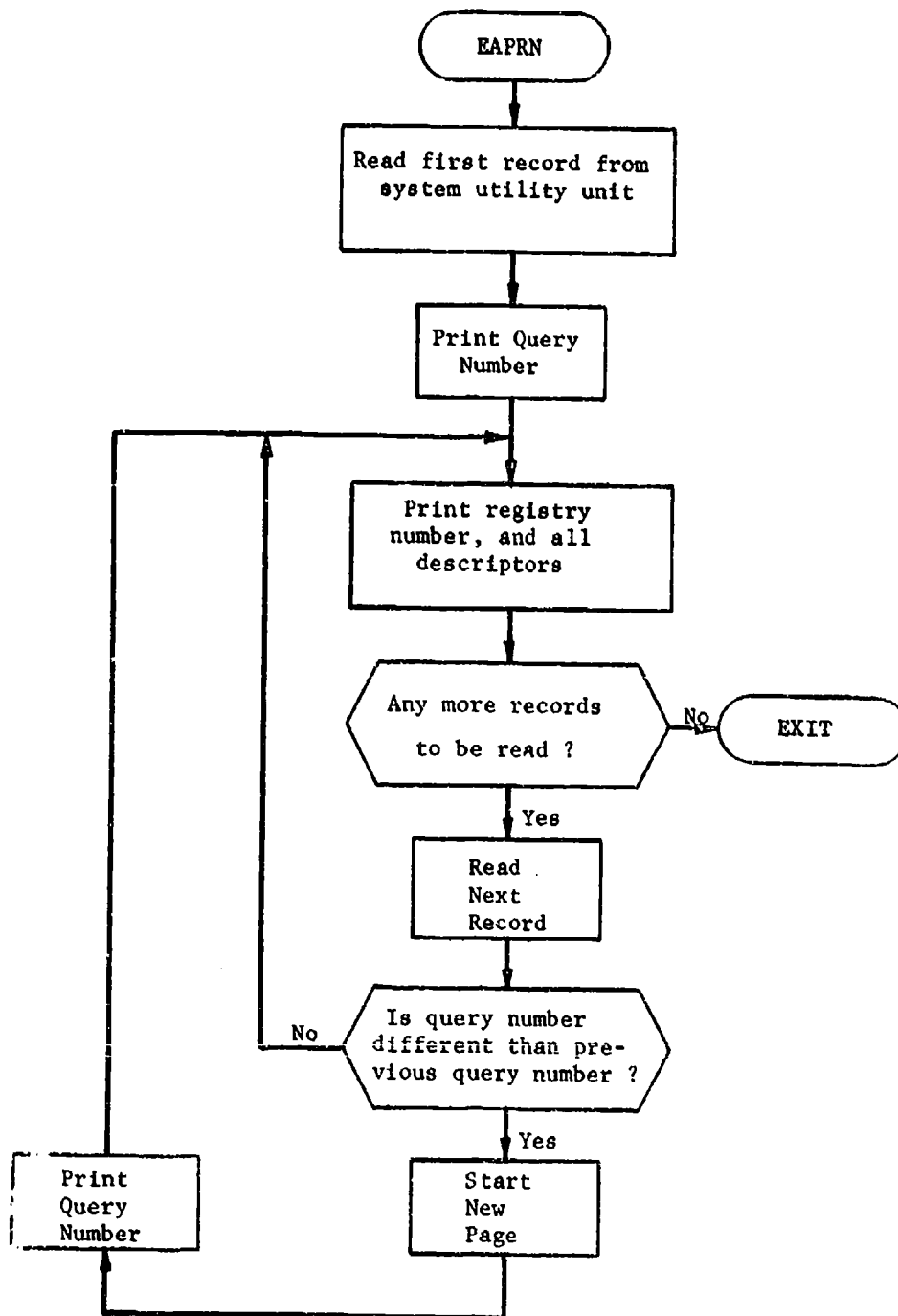
Figure 58.   Macro Flow Chart - EAPRN

3.3.2 Structural Formula Reconstruction For Paper Tape Output

Code Name: DURPIX

Programmer: Helen Hill

Abstract: Reconstructs picture for output from 7040 through PDP-8 to punch Dura Mach paper tape.

3.3.2.1 Program Description

A macro flow chart describing this program is presented in Figure 59.

DURPIX takes the Scrub list and decodes each word one at a time, filling a buffer with the proper Dura Mach characters to punch paper tape efficiently. This tape can then be used to type the picture on the Dura Mach.

3.3.2.2 Program Structure

The program occupies 894 core locations and contains a 127 location table which translates compact Dura Mach characters to actual Dura Mach characters. Subroutine PACK is used to pack Dura Mach characters for output. Transmission of the information to the PDP-8 for the production of paper tape is accomplished by a modified version of Joboul.

DURPIX is a subroutine which takes as input the SCRUB list and outputs a buffer of packed Dura Mach characters.
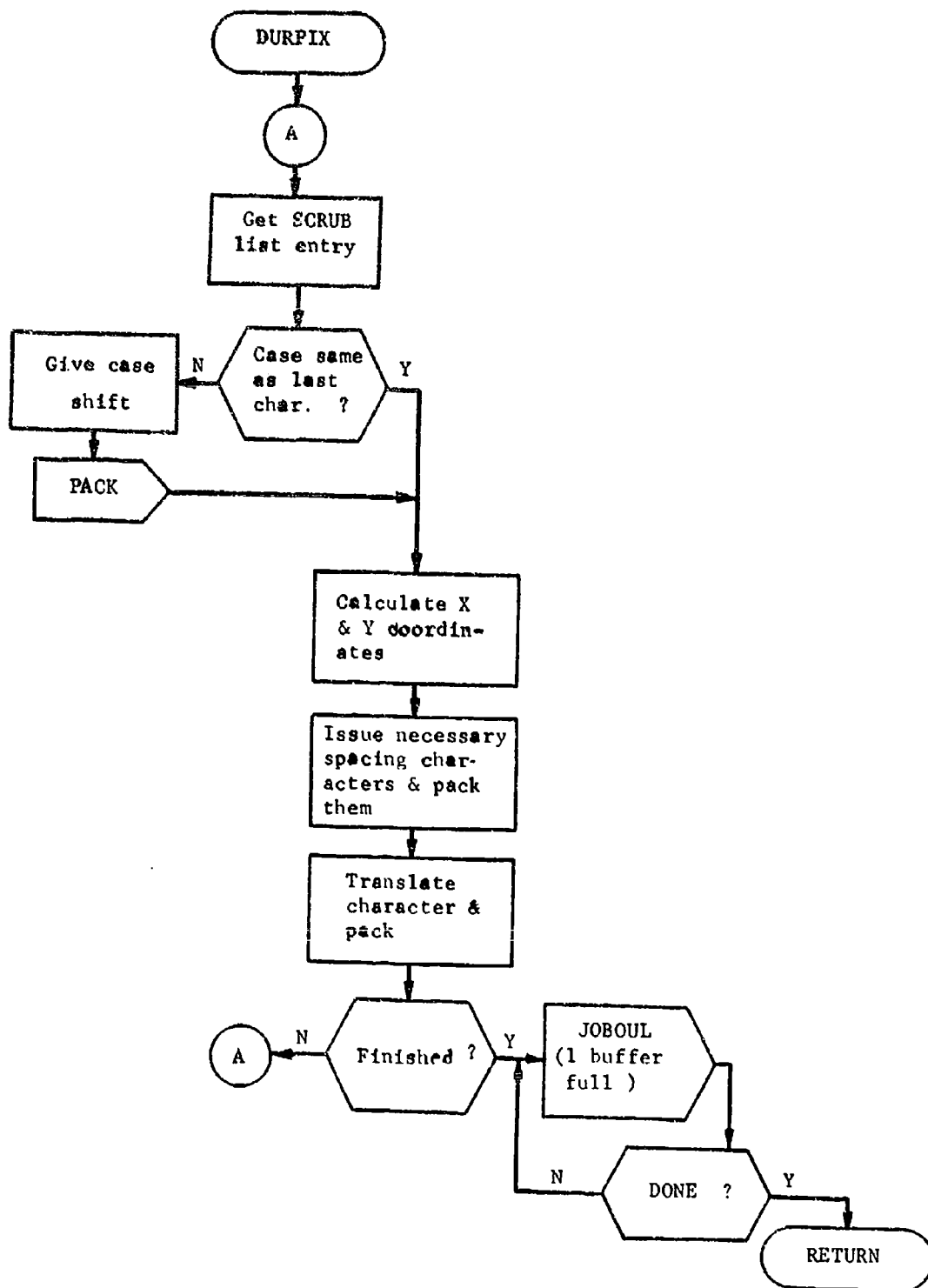
Figure 59. Macro Flow Chart - DURPIX

228

3.3.3   Structural Formula Reconstruction

Code Name: PIX & LINPIX

Programmer: Helen Hill

Abstract: PIX reconstructs the picture from the stored structural formula image and calls TRANSL and WRITE to output on the 1401 printer. LINPIX reconstructs the picture line by line for output on the chemical Line Printer.

3.3.3.1   Program Description

A macro flow chart describing PIX is presented in Figure 25.

PIX and LINPIX take as input the stored structural formula image (SCRUB) containing each character in the structure and its relative location in a matrix, and the x and y size of the matrix. PIX reconstructs the picture in the matrix in compact internal code (six bit code with an added case bit). LINPIX does the same thing one line at a time.
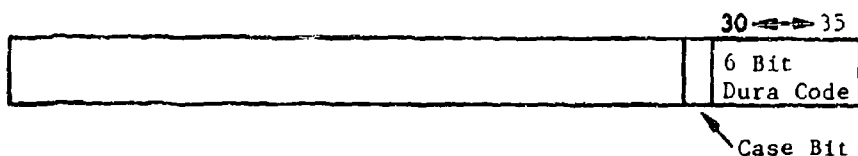
3.3.3.2   Program Structure

PIX is a subroutine which utilizes a 10000 location matrix in which to reconstruct the picture. LINPIX uses a 100 location buffer to reconstruct a single line of the picture .

PIX & LINPIX make use of the following input

    SCRUB - 701 locations

    DELX and DELY

    UNDTAB - table of underlines

```
                    ┌──────────┐
                   (    PIX     )
                    └────┬─────┘
                         │
                         ▼
              ┌─────────────────────┐
              │  Get SCRUB list     │
              │      entry          │
              └──────────┬──────────┘
                         │
                         ▼
              ┌─────────────────────┐
              │  Compact            │
              │  character          │
              │  code               │
              └──────────┬──────────┘
                         │
                         ▼
              ┌─────────────────────┐
              │  Place character    │
              │       in            │
              │     matrix          │
              └──────────┬──────────┘
                         │
                         ▼
              ┌────────────────────┐
              │                     ╲
              │     TRANSL           >
              │                     ╱
              └──────────┬─────────┘
                         │
                         ▼
                   ┌──────────┐
                  (   RETURN   )
                   └──────────┘
```
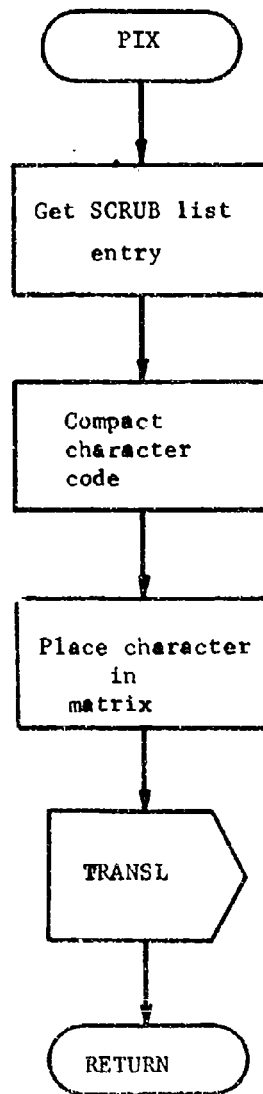
Figure 60. Macro Flow Chart - PIX

230

Code Name: DURADK, MFOU, LEADER

Programmer: James Gerber

Abstract: DURADK contains routines to translate and format information and to punch this on the teletype to produce a type for the Dura Mach typewriter. Leader is produced by routine LEADER in another deck. MFOU will format and print the Molform on the teletype or line printer.

3.3.4.1 Program Description

DURADK consists of the following routines which are called by TXINFO to produce Dura Mach output:

(1) MFDURA will punch the Molform in Dura code to be typed out in readable format. It operates similarly to MFOU. The location of the first word of the Molform is found in the accumulator when the routine is called.

(2) QUERNO will punch out the query number and the header query number. The location of the query number is in the accumulator when QUERNO is called.

(3) REGNO will print the registry number with leading zeros eliminated. The registry number is preceded by the character RN. The registry number is preceded by the Dura "print on" code. The accumulator contains the location of the first word of the two word registry number when REGNO is called.

(4) QNASCI will punch the Dura print off code, the query number in ASCII code. The accumulator should contain the location of the query number when it is called.

DURADK uses routine BINBCD and JOBOUL. If used without the CIDS JOBOUL package, the resulting output on the 1403 line printer will be the garbled version of the Dura output since the output is formatted three characters to a 7040 word. Thus each Dura character will print as two printer characters which will have no relation to the Dura character.

These routines expect the Dura Mach to be at the beginning of a line and always return with the Dura Mach at the beginning of a line. The Dura Mach must be in lower case at both the start and end. Any routines used with this package must therefore return with the Dura Mach in lower case and at the beginning of a line.

3.3.4.2 Program Structure

MFOU consists of two routines:

(1) MFOU is called with the location of the first word of the item Molform in the accumulator. MFOU will format and print the Molform (both hill and addend Molform if present)

using JOBOUL. If the teletype version of JOBOUL is used, this will print out on the on-line(PDP)teletypes.

(2) LEADER will produce about 12" of leader (octal code 200) on the teletype.

MFOU uses routines BINBCD and JOBOUL.

LITERATURE CITED

1. C. T. Van Meter, D. Lefkovitz, and R. V. Powers, An Experimental Chemical Information and Data System, CIDS No. 4, University of Pennsylvania, Philadelphia, Pa., January 1967.

2. Report to The AMC User Advisory Group on The Initial Test of an Experimental CIDS, US Army Materiel Command, US Army Munitions Command, October 2, 1967.

3. B. Hack, H. Hill, D. Lefkovitz, The CHEMTYPE System, Office of Engineering Research, University of Pennsylvania, Philadelphia, Pa., October 22, 1967.

4. P. R. Weinberg, A Guide to the CIDS Retrieval Language, University of Pennsylvania, Philadelphia, Pa., November 1967.

5. ACT II Chemical Typing Conventions, Office of Engineering Research, University of Pennsylvania, Philadelphia, Pa., January 16, 1967.

6. ACT III (Dura) Chemical Typing Conventions, Office of Engineering Research, University of Pennsylvania, Philadelphia, Pa., January 16, 1967.

BLANK PAGE

APPENDIX A

SYSTEM FLOWCHART WITH PROGRAM NAMES

A macro flowchart of the complete CIDS system is presented on the
following pages. The code names of the programs required to perform each
phase of processing are included. The file construction subsystem is
presented first. The initial processing differs for chemical typewriter
input and CAS input, thus these charts are shown separately. A single
chart describes the remainder of the processing which is the same for both
types of input. The search subsystem is divided into two charts - the
batch search system and the real time search system. The designation CF
on a tape symbol means that the compound file is in CIDS record format.

CHEMTYPE: Input of Compounds, Formatting, Creation of Connection Table, Verification

APOLGY	ORGNZR
CLEANM	PHASE5
CONVRT	PUNCH
EXCESS	REGRUP
INPUTD	REJECT
MAKECT	SETUP
MOLFRM	TAPWRM
MONIKR	TICKER
NFCF	VERIFY

Chemical Typewriter Input

TID
M.F.
C.T.
S.F.
Ref

Sort by TID

Creation of CIDS Record

UPTAP

EA Nos.

Add: MF Sort Key EA Descriptors

CF

Sort by MF Sort Key

Ring Key Assignment

SCNCAS
SCRNDR
RING1
RING2
RING3
RING4

CF

Add: Ring Keys

Add: Reg. No.

Registration

STARTA
REGUD
HLDPRC
RUD II
STRUC

CF

Remove: MF Sort Key

A

File Construction (Chemical Typewriter Input)

236

CAS
Structure
Master

CAS Structure
Conversion

CASFMT
CONVRT

Structures in
CIDS Format

Sort by
CAS Reg. No.

Sorted
Structures

CAS
Bibliography

Add Molecular
Formula

ADDMF
MOLEF

Ring Key Assignment

SCNCAS
SCRNCR
RING1
RING2
RING3
RING4

CF

A

Add:
Ring Keys

CF

R.N.
M.F.
C.T.

3
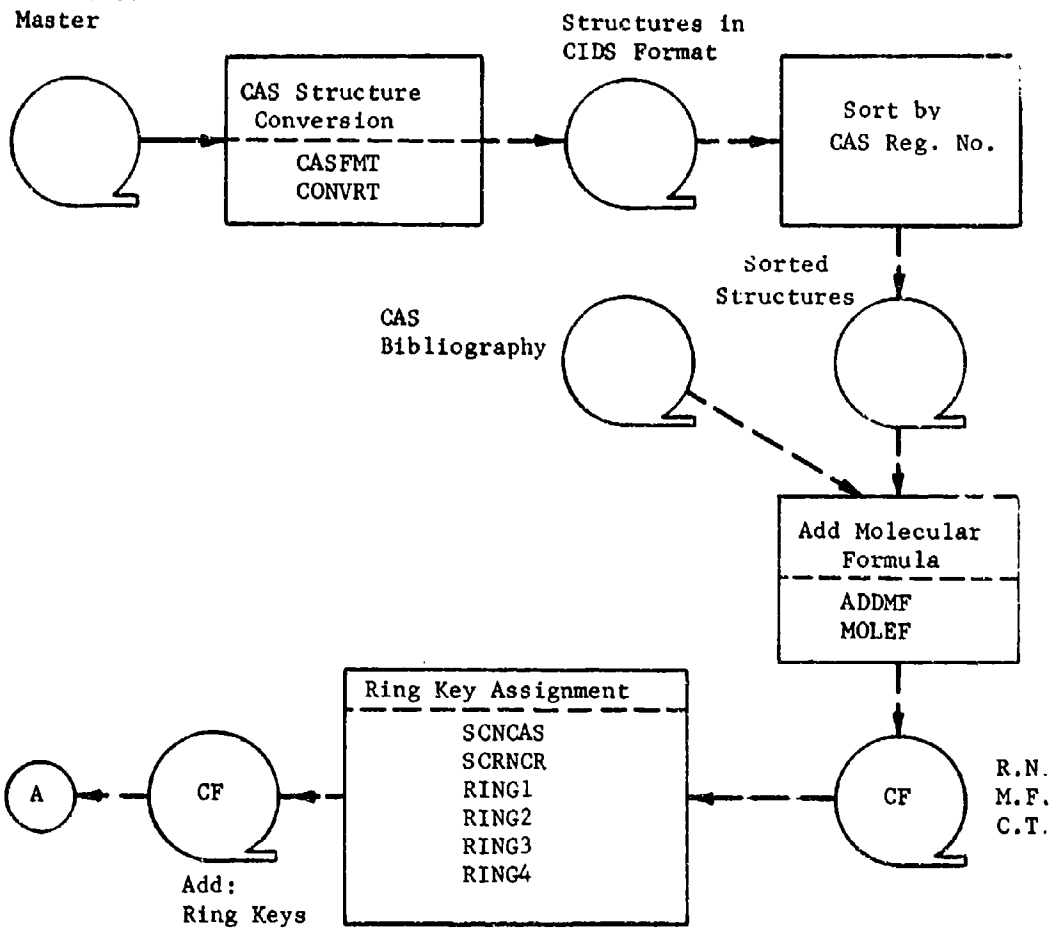
File Construction (CAS Input)

237
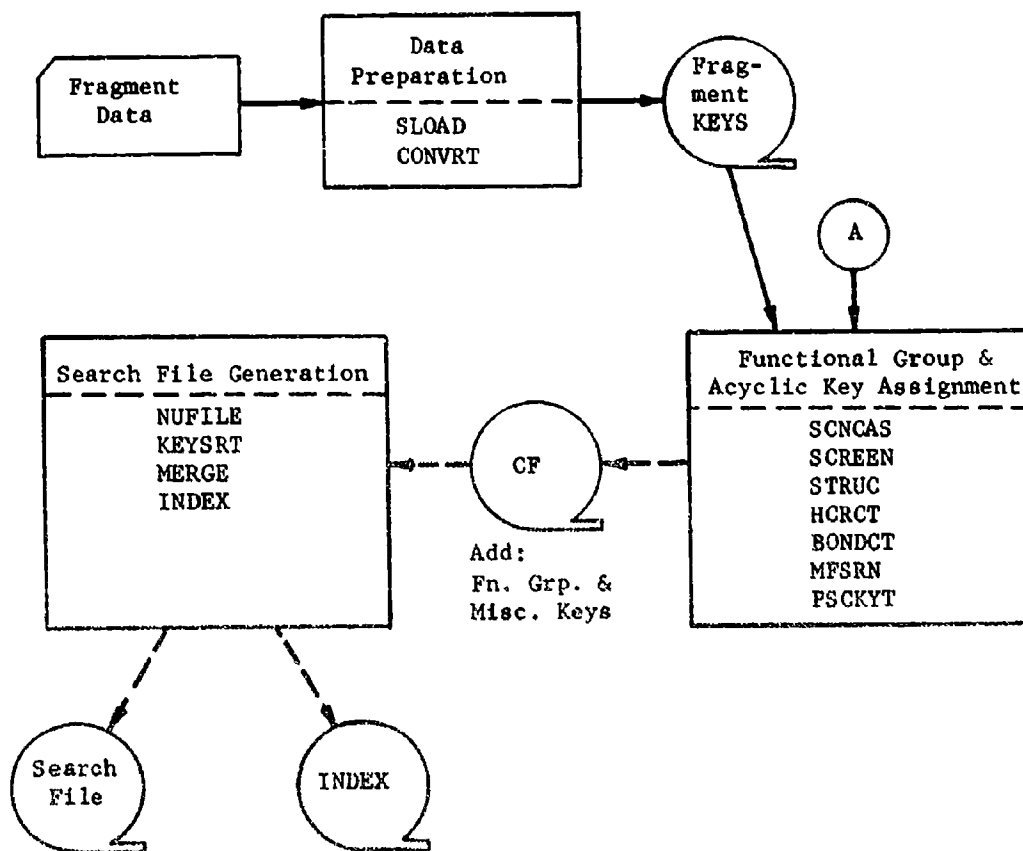
File Construction (All Input) (Continued)

238

Output of Responses

DURADK
DURPIX (or LINPIX)
LEADER

Chemical Line Printer

Search File

File Search

TXINFO (or DISKET)
MOLFM
STRUC

Dura Paper Tape

Search File

Query

Acces-sion List

Query Preproces-sor and Indexing

INPUT
EXEC30
READ
KIAD
PACKEL
MOLE
MOPACK
CONVRT

INDEX

Queries

Real Time Search System

239

Search File

File Search

TAPE
MOLFM
STRUC

Re-
trievals

Sort
by
Query
No.

Output Printed

REGPRN

(or EAPRN)

Queries

Merged
Accession
List

INDEX

Query Processor
and Indexing

INPUT
EXEC30
READ
KIAD
PACKEL
MOLE
MOPACK
CONVRT

Queries

Batch Search System

APPENDIX B

PROGRAM ABSTRACTS

ADDMF - Addition of Molecular Formula (2.1.3)

ADDMF reads a tape of compound connection tables which have been translated from CAS to CIDS format and are ordered by CAS Registry Number. Molecular formula data from the CAS Bibliography File is added to this tape and the compound records are rewritten in CIDS record format.

APOLGY - Error Message Program (2.2.9)

APOLGY is transferred to from ORGNZR, EXCESS, REGRUP, MOLFRM, and MONIKP to write error messages using Fortran read-write routines.

BONDCT - Bond Count (2.4.10)

Program BONDCT assigns a specific limited subclass of the aliphatic keys. It assigns the acyclic nucleus keys and two types of hydrocarbon radical keys.

CASFMT - CAS Structure Conversion (2.1.1)

CASFMT reads the CAS Structure Master File and translates the information to the CIDS format. The output of CASFMT is a tape containing the registry number, connection table, and abnormality table (if present) for each CAS compound converted.

CLEANM - Reduction of the Matrix to Points and Lines (2.2.11)

CLEANM is given a pointer to a specific node by SETUP. It then "cleans" the eight locations around that node in the matrix for use in MAKECT. All charge signs and mass numbers are removed, double letter elements are replaced by a one word symbol, and special cases such as Ph and $-(C)_n-$ are treated. An abnormality table of abnormal masses, charges and valences is created. A connection table number is assigned to each atom and the word in SCRUB corresponding to a node which has been processed by CLEANM is made minus. Control is returned to SETUP after operation on the given node is complete.

COMPR - Ring Compression (2.4.6)

Program COMPR (RING3) removes all atoms in the connection table which have exactly two attachments and removes side chains from the structure, in order that the ring descriptors may be found. The program also contains a subroutine which removes a prescribed path from the structure.

CONVRT - Structure Conversion and Compression (2.1.2)

> This program converts a structure to a format suitable for storage and searching. The structure is compressed to facilitate the atom-by-atom search by removing carbon atoms with exactly two direct attachments, leaving only a description of the bonds in the chain. The program will also format structures which are query fragments, in which case the resulting connection table has redundancy removed and the atoms are ordered to speed searching. In addition, the various types of free, or hanging, bonds are formatted.

DISKTT - See TAPE.

DURADK - Dura Mach Output Package and Teletype Output Package (3.3.4)

> DURADK contains routines to translate and format information and to punch this on the teletype to produce a tape for the Dura Mach type-writer.

DURPIX - Structural Formula Reconstruction for Paper Tape Output (3.3.2)

> Reconstructs picture for output from 7040 through PDP-8 to punch Dura Mach paper tape.

EAPRN - Registry Number and Descriptor Print Program (3.3.1)

> Program EAPRN is used to format and print compound registry numbers and descriptors obtained as retrievals to queries searched against the tape file.

EXCESS - Structure of Non-Bracketed Information (2.2.8)

> EXCESS formats all structural characters appearing outside of brackets in a bracketed structure.

EXEC30 - Query Preprocessor (3.1.2)

> The Query Preprocessor is a set of programs that scans the source text of a query presented by the user and translates it into the internal coding of the retrieval system. Queries are checked for syntactical errors and edited. In this role, the Query Preprocessor communicates with the other programs of the CIDS system to allow system to adjust to particular user requirements.

HCRCT - Nonspecific Hydrocarbon Radical Key Assignment (2.4.9)

> Program HCRCT assigns to a structure a sub-class of the set of aliphatic hydrocarbon radical keys.

HLDPRC - Hold Tape Processor (2.3.2)

HLDPRC is a program of the Registry System which processes compounds on the Hold tape produced by program STARTA, according to an action code punched on each card of the TID card deck produced by STARTA. The action codes are the results of a chemist's decision to register, ignore, or update the compound record for each compound on the Hold tape.

INDEX - Index Creation (2.5.4)

The key-to-compound locater table, used by the CIDS Retrieval System, is created by program INDEX from the inverted key list by program MERGE.

INPUTD - Dura Mach Input Program (2.2.2)

This program accepts magnetic tape images of the paper tape chemical records typed by the Dura Mach chemical typewriter and reconstructs the chemical record in a 2-dimensional array called MATRIX.

INPUT - Query Input Executive (3.1.1)

Program INPUT is used to keep track of the number of queries correctly entered in the system. It also stores the disk address of each query in the query disk-core table.

KEYSRT - Key-Address Sort (2.5.2)

KEYSRT sorts the Key-Address tape which is output from program NUFILE. The key-address pairs are sorted in ascending order according to key number. It maintains the ascending order of addresses as they are produced by NUFILE.

KIAD - Key-Expression to Accession List Processor (3.1.4)

KIAD accepts the boolean key expression as input and produces the list of all compound record addresses that pass the key expression (the accession list.)

LEADER - Dura Mach Output Package and Teletype Output Package (3.3.4)

Paper tape leader is produced by routine LEADER.

LINPIX - Structural Formula Reconstruction (3.3.3)

LINPIX reconstructs the picture from the stored structural formula image line by line for output on the chemical line printer.

243

MAKECT – Generation of the Connection Table (2.2.12)

>MAKECT assumes a MATRIX of nodes and connecting lines. A list is generated for each node indicating the type of node (element type), all associated points, and the connecting line types (bonds). This program also indicates if the atom is to be multiplied in order to be correctly compared with the molecular formula during chemical verification.

MERGE – Key-Address Merge (2.5.3)

>MERGE combines the Sorted Key-Address tape (see NUFILE and KEYSRT) with the Old Merged Key-Address tape containing all the keys in the file (prior to the present run) and the addresses of their occurrences to produce a New Merged Key-Address tape.

MFOU – Dura Mach Output Package and Teletype Output Package (3.3.4)

>MFOU formats and prints the Molform on the teletype or line printer.

MFSRN – Molecular Formula Key Assignment (2.4.11)

>Molecular Formula keys are assigned to a compound based on the Hill molecular formula. One key is assigned to identify each element present.

MOLE – Connection Table Processor (3.1.6)

>Program MOLE processes chemical structural data in the form of manually generated connection tables. The data is converted into an intermediate format which is then compressed into the CIDS internal connection table by program CONVRT.

MOLEF – Molecular Formula Extraction Program (2.1.4)

>Subroutine MOLEF consists of a package of programs that locate and extract the file record corresponding to a given registry number from the CAS Bibliography tapes. Summation and addend molecular formulas are computed and stored in CIDS format.

MOLFM – Molecular Formula Search (3.2.2)

>Program MOLFM is used to determine whether the molecular formula of a particular file compound satisfies the requirements specified in a particular query.

MOLFRM – Molecular Formula Format Program (2.2.4)

>This program formats the molecular formulas in the typed chemical rec...

MONIKR – Nomenclature and Reference Field Formatting Program (2.2.5)

>MONIKR formats the nomenclature and any other information typed with :

MOPACK - Molecular Formula Translator   (3.1.7)

This program translates the query molecular formula to internal query format.   It checks the query molecular formula for syntactical errors and indicates these to the calling program.

MUSTRP - Alternate Path Search (2.4.5)

MUSTRP (RING2) is given a connection table path between two nodes in a ring and searches for any alternate paths between these two nodes. If more than one alternate path is found, the "best" of these is chosen.

NFCF   - Expansion of the Connection Table   (2.2.15)

This program expands the connection table from the internal format to the format acceptable by program CONVRT and will print the connection table and abnormality table if a switch is set.

NUFILE - Search File Creation or Update (2.5.1)

NUFILE creates a search file of compounds by simply assigning each compound to an area in the file as it is input to NUFILE.   An existing file may be updated by the same process.   NUFILE simultaneously creates a tape of key and file address pairs which will be used by programs MERGE and INDEX to create an index to the complete compound file.

ORGNZR - Field Recognizer and Format Program   (2.2.3)

ORGNZR takes the reconstructed matrix (in Mergenthaler code with a case bit added) and recognizes each field in the chemical record. It formats the temporary identification number, the security classification, the molecular formula (both Hill and addend molform if the latter is present), the structural formula image, the stereo information and the nomenclature.

PACKEL - Key Packing Program (3.1.5)

The program translates the individual key names from query language format to internal format.

PHASE5 - Calling Program for Chemical Verification   (2.2.13)

This is the call program for VERIFY.   If a compound is found to be correct by verification, this program transfers to NFCF.   Otherwise, an error exit is taken and control is transferred to REJECT.

PIX   - Structural Formula Reconstruction (3.3.3)

PIX reconstructs the picture from the stored structural formula image and calls TRANSL and WRITE to output on the 1401 printer.

PSCKYT - Nonspecific Phosphorus Functional Group   (2.4.12)

> Subroutine PSCKYT assigns keys to compounds which contain certain types
> of phosphorus functional groups which were not among those selected
> as Specific Functional Group keys.

PUNCH - Descriptor Punch Program (2.2.6)

> This program finds the EA, T, and TL descriptors, if there are any.
> It then gets the corresponding TID number of the compound and punches
> it on a card followed by the EA, T, or Tl descriptor number.

READ - Query Reader   (3.1.3)

> Program READ is used to read queries from an input device. It can be
> used to read either punched cards or punched paper tape produced by a
> teletype.

REGRUP - SFI Reordering Program   (2.2.7)

> Program reorders the SFI when brackets or a monovalent salt are
> present so that all characters within a given set of coordinates appear
> compactly in the SFI.

REGUD - Registry Print Tape Update   (2.3.3)

> REGUD updates the Print tape by adding new records for a group of
> newly registered compounds.

REJECT - Rejection of Incorrect Records   (2.2.17)

> This program is transferred to from various portions of the CHEMTYPE
> system. A message is printed out and the program transfers to AEND.

RING1 - Ring Analysis Executive   (2.4.4)

> The general function of RING1 is to find the smallest set of smallest
> cycles in a compound patterned after the rules of the Ring Index.
> These cycles are determined and the generic cyclic nuclei keys are
> assigned to the compound.

RING2 - See MUSTRP.

RING3 - See COMPR.

RING4 - See TABLE.

RUD II - Registry Print Tape Update II (2.3.4)

> RUD II updates the Print tape by adding new records for a group of
> newly registered compounds and updating records corresponding to
> previously registered compounds.

246

SCNCAS - Key Assignment Executive (2.4.1)

> SCNCAS is the executive for the Key Assignment programs. For each
> compound on the input tape, the sub-executive program is called which in
> turn calls the appropriate screening subroutines. SCNCAS writes the
> compound record on tape in the same format with the newly assigned
> keys added to the record.

SCREEN - Key Assignment Sub-Executive (2.4.2)

> Program SCREEN (other versions SCRNCR, SCRNDR) is a subroutine of
> program SCNCAS and acts as an intermediary between it and the various
> key assignment subroutines. SCREEN, the version used when hydro-
> carbon radical and functional group fragment keys are being assigned,
> selects the particular screen fragments which must be applied to the
> compound being screened.

SCRNCR - See SCREEN.

SCRNDR - See SCREEN.

SETUP   - Linear String Classification (2.2.10)

> This program finds a capital letter in the SCRUB list and then scans
> to the left and right of this letter in the MATRIX assigning a type
> code to the linear string. It then transfers to CLEANM for processing.

SLOAD   - Loading of Structural Fragment Screens (2.4.3)

> SLOAD prepares structural fragment data for use by the screen assignment
> program.

STARTA - Master Registry Program (2.3.1)

> STARTA determines which of a group of potential new compounds are
> different from those already registered in the master file. These
> compounds are registered, positive matches are discarded, and question-
> able matches are printed for further examination by a chemist.

STRUC   - Atom-by-Atom Search (2.4.8)

> The purpose of the atom-by-atom search program is to determine if a
> one-to-one correspondence exists between the nodes (atoms) and con-
> nections in a given query structure, and some set of nodes and connections
> in a given file compound structure.

TABLE   - Connection Table Expansion (2.4.7)

> TABLE (RING4) expands a connection table given in the compressed format
> (see program CONVRT) to a form suitable for application of the ring
> analysis programs.

TAPE   - Search Executives   (3.2.1)

> Programs TAPE,TXINFO and DISKTT are used to retrieve compounds from a
> file stored either on tape or on disk. With the aid of the molecular
> formula search and structure search programs, the executives determine
> which of the selected compounds actually satisfy the requirements of
> various queries. Output programs are then called to print the
> resulting compounds.

TAPWRM - Mergenthaler Input Program   (2.2.1)

> TAPWRM reads typewriter characters in Mergenthaler Code from a magnetic
> tape. It interprets these codes and constructs a 2-dimensional array
> containing an image of the typed chemical record.

TICKER - Output of Chemical Record   (2.2.16)

> TICKER writes an output tape containing the TID, classification and
> stereo information, molform, nomenclature and references, structural
> formula image, connection table, and abnormality table.

TXINFO - See TAPE.

UPTAP - CHEMTYPE to CIDS Format Conversion   (2.2.18)

> UPTAP reformats the output of the CHEMTYPE system into the CIDS
> record format and merges into the record descriptors which were
> introduced through punched cards.

VERIFY - Chemical Verification   (2.2.14)

> VERIFY checks the chemical consistency of the structural formula,
> molecular formula, and connection table, and verifies the valence of
> each element in the connection table and in the abnormality table.

APPENDIX C

CAS FORMATS INPUT TO CIDS

This appendix is a reproduction of selected portions of the Chemical Abstracts Service Registry System Manual, revised February 1966. It describes the formats of the CAS Structure Master File which is the input to CIDS program CASFMT (Section 2.1.1) and the CAS Bibliography File which is the input to CIDS program MOLEF (Section 2.1.4).

Structure File

```
097A010C-1    001001001001002002003003003004006007009010012012013014015016017018020021022023024025+    BL  971 CM

004IF2C   C C C C C O O O O O C C C C C C C C C[+]                                                    RC  104 CM
0053F2C   1 1 1 1 1 2 1 2 1 1 1 1 1 1 2 1 1 1 1+                                                       RC   59 CM
003CF2C   1 1 1 1 1 2 1 2 1 1 1 1 1 1 2 1 1 1 1+  V0025035012S  NS+                                    RC   38 CM
0041F    0018495004301302  V0025035012S  NS+                                                          RC   41 CM

0088F1    001001001001002002003003003004006007009010012012013014015016017018020024025+                RC   88 CM
0061F2C  P P C C O O S O O S P C C C O O O C C C C C C C C C+                                          RC   61 CM
0034F3  1 1 1 1 1 2 1 2 1 1 1 1 1 1 2 1 1 1 1+                                                         RC   34 CM
004IF4000086574703501302  V0025035012S  NS+                                                           RC   41 CM
0076F1    001001001002002002003003004006007009010012012017018020021+                                  RC   76 CM
0053F2C  P P C C O O O O O P C C C C C C O O C C C C +                                                 RC   53 CM
0030F3  1 1 1 1 1 2 1 2 1 1 1 1 1 1 2 1 1 1 1+                                                         RC   30 CM
004IF40000865758027013012  V0025035012S  NS+                                                          RC   73 CM
0073F1    001001001002002002003003004006007009010012012013014015016+                                  RC   73 CM
005IF2C  P P O O O S O O S C C C C C C C C C C C+                                                      RC   51 CM
0029F3  1 1 1 1 1 2 1 2 1 1 1 1 1 1 2 1 1 1 1+                                                         RC   29 CM
0037F400106670202026009 02  V0025003S  NS+                                                            RC   37 CM
0049F1    001001001001002002002003003004006009+                                                       RC   49 CM
0035F2C  S S C C O O O O C C C C +                                                                     RC   35 CM
0021F3  1 1 1 1 1 2 2 1 2 2 1 1 1+                                                                     RC   21 CM
099G0037F40000762000018009 02  V0025003S  NS+                                                          BL  997 CM
0046F1    001001001001002002002003003004012+                                                          RC   46 CM
0033F2C  C C C C C L C L F F F F C I +                                                                 RC   33 CM
0020F3  1 1 1 1 1 1 1 1 1 1 +                                                                          RC   20 CM
0028F400067905004 00002  NS+                                                                          RC   28 CM
0033F2C  C C C F F F F F F C F +                                                                       RC   33 CM
0020F3  1 1 1 1 1 1 1 1 1 1 2 1 +                                                                      RC   20 CM
0028F4001622190020000 2  NS+                                                                          RC   28 CM
0033F2C  C C O C C C C C C C +                                                                         RC   33 CM
```

250

Structure File

The preceding page is a photographic re-production of a tape dump of part of the Structure File. The Structure File contains all the information in the unique/compact connection table generated for a compound by the Registry programs. On the left of the sheet, the notations "BL" and "RC" appear. These notations which are not on the tape, are generated by the program which prints the tape dump. "BL" stands for block length and is followed by a count of the characters in that physical tape record. "RC" stands for record count and is followed by a count of the characters appearing in the associated logical record. For the structure file a logical record may be a "F1", "F2", "F3", or "F4" record (see descriptions). Several logical records may appear within one physical tape record.

Description

1. Block count (IBM standard). This four-position count appears at the beginning of each physical tape record (block) and indicates the number of characters in the block. As per IBM standard, the digits 0-9 in the units position print as +, A, B, C, D, E, F, G, H, and I, respectively.

2. Record count (IBM standard required for variable length, blocked records). This four-position count gives the number of characters in the following logical tape record. The record count is the first field of the logical record.

3. Record Type. This two-position field defines the contents of the records.

4a. (3 is "F1") From list. The "from list" defines the graph of a compound. (see page 99).

5a. Record Mark. This character marks the end of a logical record.

4b. (3 is "F2"). Element list. This list gives the node values for the previous F1 record. (See page 100).

5b. Record Mark.

4c. (3 is "F3"). Bond list. This list gives the connection values for the last previous F1 record. (See page 101).

5c. Record Mark.

4d. (3 is "F4"). Registry Number. A nine-position field containing the CAS Registry Number assigned to the compound. (See page 88).

5d. Hydrogen Count. A three-position field indicating the number the number of hydrogen atoms in the compound.

6d. Abnormalities Count. This three-position count indicates the length of the abnormalities present.

7d. Text length. A two-position field indicating the length of the textual descriptor.

8d. Abnormalities. A variable length field containing abnormalities and the textual descriptor (See pages ... and ...).

9d. Record Mark.

L. H. Leighner 1/12/66

CHEMICAL ABSTRACTS SERVICE
Format Layout

Format Type (T,C,D, or K) _____ T _____   Format Number __ 14a _____

Record
Length Fl   From List   Ring Closure List

Positions		DATA FIELD	REMARKS
From	To		
1	4	Numeric	Record character count
5	6	Fl	Record identification
7	9		Length from position 8 to start of ring closure list. Blank if no rings.
9	9		Plus zone (+) if discontinuity.
10	12	Numeric	From attachment of second atom.
$3n+6$	$3n+8$	Numeric	From attachment of nth atom
$3n+9$	$3n+9$	Blank or record mark	Blank-ring closures follow
			Record Mark – no ring closures
$3n+10$	$3n+15$	Numeric	First ring closure
$3n+6m+10$	$3n+6m+15$	Numeric	Last ring closure (mth)
$3n+6m+16$	$3n+6m+16$	Record mark	End of record

253

CHEMICAL ABSTRACTS SERVICE
Format Layout

Format Type (T, C, D, or K) _____ T _____   Format Number ____ 14b ____

Record
Length F2 Element List

```
  5    10    15    20    25    30    35    40    45    50
 55    60    65    70    75    80    85    90    95   100
105   110   115   120   125   130   135   140   145   150
```

Positions		DATA FIELD	REMARKS
From	To		
1	4	Numeric	Record character count
5	6	F2	Record identification
7	8	Alphabetic	First atom symbol
9	10	Alphabetic	Second atom symbol
$2n+6$	$2n+7$	Alphabetic	Last atom symbol (nth)
$2n+8$	$2n+8$	Record mark	End of record

259

Format Layout

Format Type (T,C,D, or K) _____ T _____   Format Number _____ 14c _____

Record
Length F3 Bond list   Ring bonds

5	10	15	20	25	30	35	40	45	50
55	60	65	70	75	80	85	90	95	100
105	110	115	120	125	130	135	140	145	150

Positions		DATA FIELD	REMARKS
From	To		
1	4	Numeric	Record character count
5	6	F3	Record identification
7	7	Alphanumeric	First bond type
8	8	Alphanumeric	Second bond type
n+6	n+6	Alphanumeric	Last bond type (nth)
n+7	n+7	Alphabetic or	First ring closure bond type,
			or end of record ( )
n+8	n+8	Alphabetic	Second ring closure bond type
n+m+6	n+m+6	Alphabetic	Last ring closure bond type (mth)
n+m+7	n+m+7	Record mark	End of record
	,	255	

Modification List

CHEMICAL ABSTRACTS SERVICE
Format Layout

Format Type (T,C,D, or K) _____T_____   Format Number ___14d___

Record Length F4 Registry No.	Hyd.Mod. Cnt.Inh.	Txt. Lnh.	Modification List	Textual Descriptor

(ruler: 5 10 15 20 25 30 35 40 45 50)

(ruler: 55 60 65 70 75 80 85 90 95 100)

(ruler: 105 110 115 120 125 130 135 140 145 150)

Positions From	To	DATA FIELD	REMARKS
1	4	Numeric	Record character count
5	6	F4	Record identification
7	15	Numeric	Registry number
16	18	Numeric	Hydrogen count
19	21	Numeric	Length of modification list
22	23	Numeric	Length of textual descriptor
24	24	Alphanumeric	Bit switch for modification list*
25	-	Alphanumeric	Modification list*
-	-		
-	-	Blank	Blank separates modifications and text**
-	-	Alphanumeric	Textual descriptor (up to 50 char.)
-	-	Record mark	End of record
			* See following page
			** If there are no modifications, this
			blank separates the bit switch from
		256	the textual descriptor

Modification List in F4 Record

The modification list is preceded by a bit switch in position 24 which indicates which modifications are present:

1 bit on:	mass citations present
2 bit on:	charge citations present
4 bit on:	fractional coefficients present
AB bit on:	special segments present

Up to five subfields may be present in the modification list. They are: (1) valence, (2) mass, (3) charge, (4) fractional coefficients, and (5) special segments, and if present must appear in that order. These subfields have the following format:

Vaaab...Maaaddd...Caaae...Fgxxxx...Shhhxxxx9jjKLmmmoxxxx....

Where:

V –	valence subfield
aaa –	atom number
b –	valence
M –	mass subfield
ddd –	mass value
C –	charge subfield
e –	charge value
F –	fractional coefficient subfield
g –	fragment number
xxxx –	fractional coefficient (two digit numerator and denominator)
S –	special segment subfield
hhh –	?gment number (ID)
jj –	element symbol
K –	no. of hydrogens attached
L –	actual valence
mmm –	mass citation
o –	charge

K. L. Weisenberger

DOT MOLECULAR FORMULA

The dot molecular formula is composed of summation molecular formulas for each disconnected fragment in the structure.

Associated with each of the molecular formula fragments except the first is a coefficient which defines the number of occurrences of that fragment. The coefficients of the fragments are normalized so that the first fragment has a coefficient of one.

The following are the three types of molform fragments and the rules for ordering within each fragment:

A. Carbon containing fragments: (1) carbon, (2) hydrogen, (3) element symbols in ascending alphabetical order.

B. Non-carbon containing fragments: (1) element symbols in ascending alphabetical order. Preferences are imposed here to change H_2O_4S to H_2SO_4, etc.

C. Single atom fragments: (1) hydrogen, (2) element symbol.

The full dot molecular formula is generated from the fragments by assigning a preference to each fragment based on (1) high carbon count, (2) high hydrogen count, (3) low alphabetical element symbol, (4) high element count. Single atom fragments always appear last.

FORMATS

1. Carbon containing fragments: Carbon and hydrogen have four position
   counts, all hetero atoms have three position counts.

2. Non-carbon containing fragments: All hetero atoms have three position
   counts.

3. Single atom fragments: These fragments have a fixed format.

Hydrogen count   Fractional coefficient

$$H\phi XXELOO1\phi NFFFCCCC$$

Element Symbol   Number of hydrogens to be subtracted
when generating a summation molform.

4. Fractional Coefficients:

$$FFFECCCC$$

Decimal   Fractional
Representation   Representation

5. Unknown Coefficients:

$$99991X01$$

Decimal   Fractional
Representation   Representation
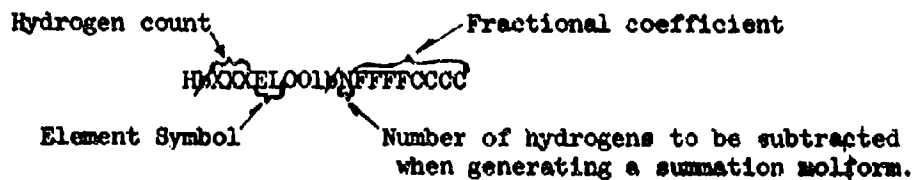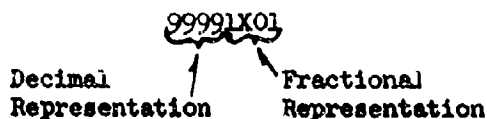
EXAMPLES

1. $C_{19}H_{28}O_2 \cdot xH_2SO_4$

   $C\phi0019\text{-}H\phi\phi0280\phi0\phi2\cdot\phi H\phi\phi0280\phi010\phi0\phi0\phi4\phi\phi99991X01$

2. $C_3H_{12}NO_6P_3\cdot3Na$

   $C\phi\phi\phi03H\phi\phi012N\phi\phi010\phi0\phi06P\phi0\phi3.bH\phi\phi\phi0Na\phi01\phi1\phi30\phi\phi301$

BIBLIOGRAPHY RECORD

CHEMICAL ABSTRACTS SERVICE
Format Layout

Format Type (T,C, or K) ___T-282___   Format Number ___133___

FILEFORM 4,   Blocksize 2000

Positions		DATA FIELD	REMARKS
From	To		
1	4	Record Length	
5	13	Registry Number	
14	14	Manual Registration Indicator	
15	19	Ring Index Number	
20	20	Ring Query Indicator	
21	22	Blank	
23	23	MERCK Indicator	
24	28	Location of CA Added Index Name	
29	33	Location of Molform	
34	38	Location of CA Systematic Index Name	
39	43	Location of Reference	
44	44	Blank	
45	-	Beginning of Data	

260

BIBLIOGRAPHY RECORD

COMPLETE ADDED NAME MATERIAL

Rec Length	Reg No	Ring Mon Reg	Ring Index Query No	Merck Ind	Added Name Ind	Loc. of Added Name	Loc. of Molform	Loc. of Preferred Name	Loc. of Ref.	CA N. Vol. T.	Added Name in UCS	Flag Field
0145	00107321	8	M	04985	Nbb M	00044	00074	00090	00111	bV622 0007	SSQ5R9R6R5QIR3;-	b
4	9	1	5	12 1	5	5	5	5	5 1 3 1 5		1

MOLFORM MATERIAL   *COMPLETE PREFERRED NAME MATERIAL*

VARIABLE LENGTH

CA N. Vol. T.	Preferred Name in UCS	Flag Field
bV63 1100004	4S3R7R5Q8	-----
1 3 1 5		

Molform: C b0012H b00160 b001

VARIABLE LENGTH   1 3 11   5

COMPLETE REFERENCE MATERIAL

Source Code	Date of Entry	Reference	Source Code
b0006 b	65298	C63570100765E04002	b*0007bb
5	11 3	5	11 3 11

VARIABLE LENGTH   1 3 11   5

AT END OF BLOCK, NO MORE REFERENCES ASSOCIATED WITH THIS COMPOUND

References | b | * |

AT END OF BLOCK, "$" INDICATES THAT THE NEXT BLOCK WILL CONTAIN MORE REFERENCES ASSOCIATED WITH THIS COMPOUND

References | b | * | $ |

REGISTRY NUMBER   Proceeded by zeros

MANUAL REGISTRATION INDICATOR   "M"–Compound has been manually registered
  b–Compound has been machine registered

RING INDEX NUMBER   Five digit reference number if compound appears in the ring index. Otherwise, blank

RING QUERY INDICATOR   "N"–No retrieval for ring queries desired
  b –Retrieval wanted

Two blank positions for later expansion

MERCK INDICATOR   "M"–Compound appears in MERCK or USAN; "N"–Compound does not appear in MERCK or USAN
  b–Not known

LOCATION of ADDED NAME   Indicates the number of positions from the beginning of the record to the location of the added CA index name

LOCATION of MOLFORM   Indicates the number of positions from the beginning of the record to the location of the molform

LOCATION of PREFERRED NAM   Indicates the number of positions from the beginning of the record to the location of the preferred CA index name

LOCATION of REFERENCES   Indicates the number of positions from the beginning of the record to the ... of the first reference

BLANK PAGE

APPENDIX D

PRINCIPAL CIDS FORMATS

As an aid to the reader, a collection of the most important formats appearing in this report is presented here. In addition, a few formats are provided which do not appear in the text.

CASFMT OUTPUT FORMAT

Word	Contents
0	Bits 0-17: Number of rings in structure Bits 18-35: Number of words in connection table
1-2	CAS registry number (9 characters, right-justified)
3-m	Connection table (CIDS format)
m+1-n	Abnormality table (if any-last word is 0)

X, B, E LISTS (INPUT TO CONVRT)

The input connection table consists of three lists: X, B, and E, in which each atom and its connections are described in eight-word blocks. The first eight words of each array is allocated to atom 1, the next 8 words to atom 2, etc. Each eight word block in the X list contains the atom numbers for up to eight connections from that atom, right-adjusted in consecutive words. The corresponding words in the B list contain the bond type of the connection, right-adjusted. In the E list, the first word of each group of eight contains the element kind for that atom, right-adjusted in BCD. In addition, bit 17 is set to 1 for each word of the E list corresponding to an entry in the X list. If the connection is a ring connection, the corresponding E word is set minus. In the example below, the X, B, E representation is shown in octal, with leading zeros omitted.

$$K_1 - C_3 \equiv N_2$$

X	B	E	
3	1	1006042	#1
0	0	0	
0	0	0	
0	0	0	
0	0	0	
0	0	0	
0	0	0	
0	0	0	
3	3	1006045	#2
0	0	0	
0	0	0	
0	0	0	
0	0	0	
0	0	0	
0	0	0	
0	0	0	
1	1	1006023	#3
2	3	1000000	
0	0	0	
0	0	0	
0	0	0	
0	0	0	
0	0	0	
0	0	0	
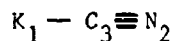
INTERNAL CONNECTION TABLE (OUTPUT OF CONVRT)

The connection table is divided into three parts: the connection segment, the bond index, and the bond segment. In addition, the first word of the C.T. is an index to the three parts. The address (bits 21-35) of the index word contains the relative location of the bond index segment; the decrement (bits 3-17) contains the relative location of the bond segment. This is illustrated below:


```
              00000x00000y              Index Word
    1                              )
    2                              |
    .                              }  Connection Segment
    .                              |
    y-1                            )
    y                             )
    .                             }  Bond Index
    .                             |
    x-1                           )
    x                             )
    .                             |
    .                             }  Bond Table
    .                             |
    .                             )
```
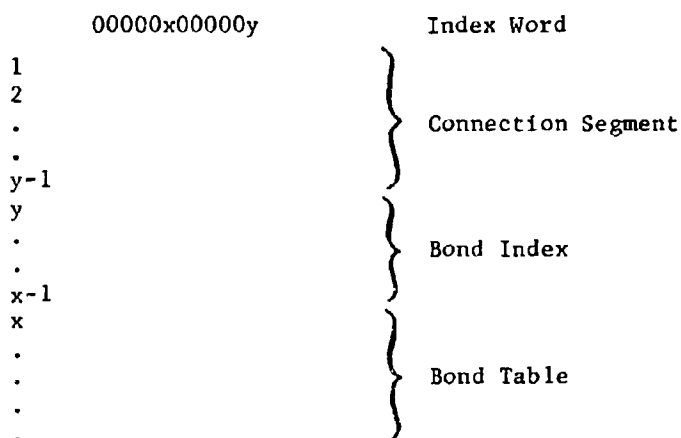
Connection Segment:--In the connection segment, carbon atoms with exactly two attachments are not explicitly stored. Every other atom in the C.T. is stored as follows:

1st word:

Bits	Contents
s	0
1	= 1 if atom is in a ring = 0 otherwise
2-5	No. of connections to this atom
6	= 1 if 1st connection is part of a ring = 0 otherwise
7-11	Path length to 1st connection

12-17	Atom no. of 1st connection
18-29	Element kind in BCD, right-justified
30-35	Node type

2nd word: (if necessary)

s	1
1-11	0
12	$\begin{cases} = 1 \text{ if 3rd connection is part of a ring} \\ = 0 \text{ otherwise} \end{cases}$
13-17	Path length to 3rd connection
18-23	Atom no. of 3rd connection
24	$\begin{cases} = 1 \text{ if 2nd connection is part of a ring} \\ = 0 \text{ otherwise} \end{cases}$
25-29	Path length to 2nd connection
30-35	Atom no. of 2nd connection

3rd, 4th words: (if necessary)

Same format as 2nd word for the remaining connections.

Bond Index: The bond index serves the purpose of location entries in the bond table corresponding to each atom in the connection segment. The format is:

Word 1:

Bits	Relative location of Bonds for
30-35	Atom 2
24-29	Atom 3
18-23	Atom 4
12-17	Atom 5
6-11	Atom 6
s-5	Atom 7

Word 2: (if necessary)

30-35	Atom 8
24-29	Atom 9
.	.
.	.

The table continues for as many words as necessary to provide an entry for each atom. The last entry gives the relative location of the word following the last word of the bond table.

Bond Table: The bond table consists of a number of groups (one group for each atom) of bond entries. The location of the beginning of each group is specified by the bond index table. Each word of a given group represents the bonds in a path from the given atom to another atom, in the form of a string of three-bit digits, each of which represents the bond type of one segment of the path. The rightmost six bits of each word contains the number of the atom to which the string is connected. For a path of length greater than 10, the bond string is continued in the next word where bits 30-35 are set zero.

As an example, the octal representation of the connection table as formatted by CONVRT for the following compound is shown below:

000016000014		Index Word
234601602302	#1	
- 1024601		
030101602302	#2	
- 1040103		
010102604601	#3	Connection Segment
020102604601	#4	
- 305		
030304604501	#5	
- 1070106		
010105604601	#6	
010105604601	#7	
151411070603		Bond Index
16		
44444401	#1	
44444401		
102		
101	#2	
203		
104		
202	#3	Bond Table
102	#4	
12105		
12104	#5	
206		
207		
205	#6	
205	#7	

268

ABNORMALITY TABLE

The abnormality table is a series of words, where each word gives information about one atom which has abnormal mass or valence or has a charge on it.

Bits	Contents
(S,1,2)	Type of abnormality 101=charge 110=mass 111=valence
(3-17)	Atom number
(18-35)	Value of abnormal mass, abnormal valence, or signed charge. The sign of a signed charge is indicated by bit 18.

A word of zeros follows the last abnormality word.

MOLECULAR FORMULA FORMAT

```
     0 1-2  3 ◄──────────────► 17  18 ◄──────► 26  27 ◄──────► 35
    ┌─┬──┬───────────────────────┬──────────────┬──────────────┐
WORD 0       Total no. words in   Multiplier of   Multiplier of
             molform block        Hill parent     water if a
             including header     if a hydrate     hydrate
    └─┴──┴───────────────────────┴──────────────┴──────────────┘
```

Set to 1 if addend
molform exists

```
                                    ┌───────┬───────┬─┐    1 if multiplier
                                    │ 4 Bits│ 4 Bits│ │    is a fraction;
                                    └───────┴───────┴─┘    0 if an integer.
                                        ▲        ▲
                                    numerator  denominator
```

If it is not a fraction, multiplier fills
8 bits, right justified.

Set if indefinite
polymer

```
     │ 1↔3  4 ◄──► 10  11◄──► 17  18 ◄──────► 26  27◄──► 34  35
    ┌─┬───┬──────────┬───────────┬─────────────┬───────────┬──┐
WORD 1     Number of   Number of    Number of     Number of
           oxygen      nitrogen     hydrogen      carbon
           atoms       atoms        atoms         atoms
    └─┴───┴──────────┴───────────┴─────────────┴───────────┴──┘
```

No. words in
mol. formula
(Hill)

```
          0 ◄────────► 11  12 ◄► 17  18 ◄──────────► 29  30 ◄──► 35
         ┌───────────────┬──────────┬──────────────────┬──────────┐
WORD 2-M    Element         Number of   Element kind      Number of
(Hill)      kind            atoms of    (BCD)             atoms of
            (BCD)           element                       element
         └───────────────┴──────────┴──────────────────┴──────────┘
```

```
          0 ◄───► 8  9 ◄──────► 17  18 ◄──────► 26  27 ◄────────► 35
         ┌──────────┬──────────────┬──────────────┬──────────────┐
WORD M+1   Multiplier   Multiplier     Multiplier     Multiplier
           of first     of second      of third       of fourth
           addend       addend         addend         addend
         └──────────┴──────────────┴──────────────┴──────────────┘
```

FORMAT OF MULTIPLIER SAME AS WORD 0 MULTIPLIER

WORD M+2   SAME AS WORD 1 -- but for addend molform

WORD M+3   SAME AS WORD 2—M -- but for addend molform

270

CIDS RECORD FORMAT

The output of ADDMF is an IOBS type 2 tape. Each compound record is a logical record. These are grouped into physical records of 1000 words or less. The CIDS record format follows:

Word	Bits	Contents
1	(3-17)	2's C (# words preceding Addit. Reg. No)
	(21-35)	2's C (# words in logical record)
2	(3-17)	2's C (# words preceding Abnormality Table)
	(21-35)	2's C (# words preceding C.T.)
3	(3-17)	2's C (# words preceding References)
	(21-35)	2's C (# words preceding S.F.I.)
4	(3-17)	2's C (# words preceding Keys)
	(21-35)	2's C (# words preceding Qualifiers)
5,6		Primary Registry Number (BCD)
7		Mol Form
.		
m		Additional Register, Number
.		
n		Structure (C.T.)
.		
o		Abnormality Table (if any)
.		
p		Structural Formula Image (if any)
.		
q		Reference (if any)
.		
r		Qualifiers (if any)
.		
s		Keys (2 words per key)

Note that several of the data blocks will be empty. The pointers to these blocks will point to the location where the data would be stored if present.

MF SORT KEY

The four word MF Sort block is attached at the beginning of each compound record prior to Registry so that the records can be sorted in Hill formula sequence. The order of preference is C, H, followed by the other elements in alphabetical order.

Word 1:

Bits	Contents
S,1-8	No. carbon atoms
9-17	No. hydrogen atoms
18-29	1st element (2 BCD characters)
30-35	No. atoms of 1st element

Word 2:

S,1-11	2nd element (2 BCD characters)
12-17	No. atoms of 2nd element
18-29	3rd element (2 BCD characters)
30-35	No. atoms of 3rd element

Word 3, 4:

Same format at Word 2 for remaining elements. Unused words are set zero.

INPUT TO CHEMTYPE

INPUT to the CHEMTYPE system is a magnetic tape with the chemical typewriter characters packed as follows:
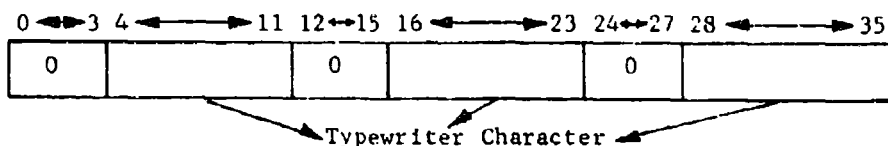
```
0 ◄►3 4 ◄──────►11 12◄►15 16 ◄──────►23 24◄►27 28 ◄──────►35
┌─────┬──────────┬─────┬──────────┬─────┬──────────┐
│  0  │          │  0  │          │  0  │          │
└─────┴──────────┴─────┴──────────┴─────┴──────────┘
```
Typewriter Character

Physical records are 300 7040 words long (900 characters). If a paper tape ends before the end of a physical record, the record is filled out with zeroes. Each paper tape record is followed on magnetic tape by a file mark. The final paper tape on a magnetic tape appears as follows:

(1) paper tape characters

(2) physical record filled with zeroes after end of paper tape

(3) file mark

(4) a physical record containing all 7's

(5) a second file mark

	8	7 ◄──── ─── ── ◄─ '		3	2	1	
Blank tape ►		0 0 0 ^		0 0	0	
PARITY BIT (EVEN)		High order Bits of X coordinates		0 0	1	
		Low order Bits of X coordinates		0 0	1	ALWAYS PUNCHED
		High order Bits of Y coordinates		0 0	1	
		Low order Bits of Y coordinates		0 0	1	
"Coordinates follow" Punch ►	0 0 0	1 1	0 0 0			

5 ◄─► 4

MERGENTHALER COORDINATE PUNCH CODE   (BINARY)

278
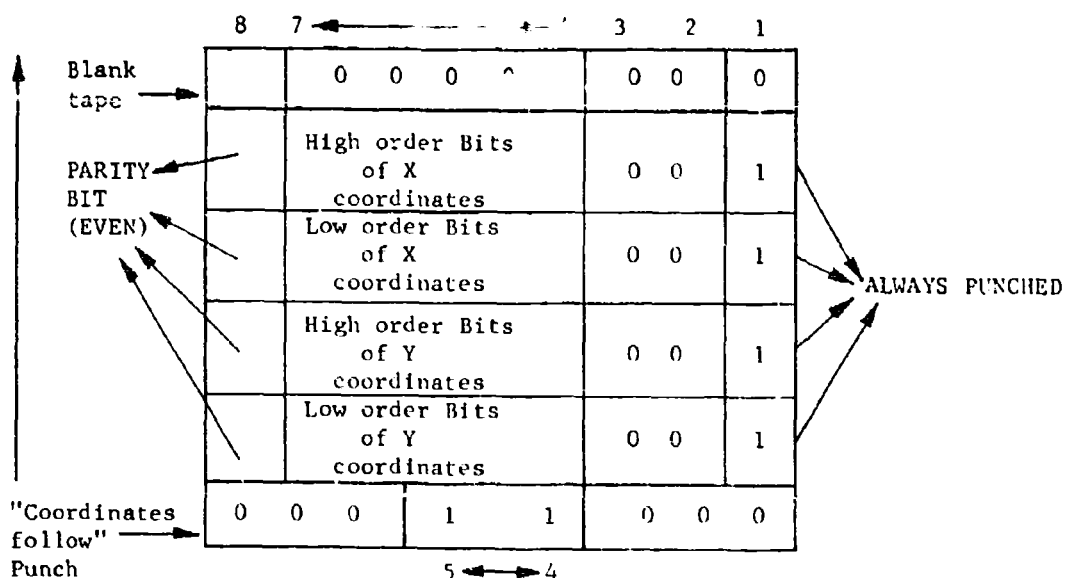
Data card input indicating which compounds on a paper tape are to be
deleted is formatted as follows:
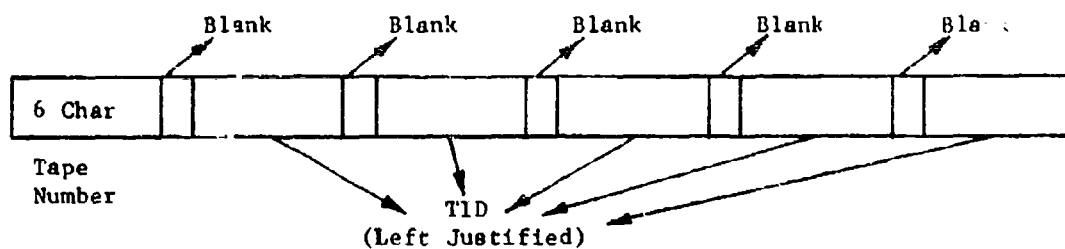
CHEMTYPE INTERNAL FORMATS

MATRIX CONTENTS AT END OF INPUT PROGRAMS:

```
0  1 ◄──────────────────────────► 26 27 28  29 ◄────► 35
┌─┬────────────────────────────────┬──────┬──────────┐
│ │                                │ Case │ Input char-│
│ │                                │ bits │ acter with-│
│ │                                │      │ out parity │
│ │                                │      │ bit        │
└─┴────────────────────────────────┴──────┴──────────┘
 1 for underlined                    SUB   = 01   Mergenthaler
   character                         LOWER = 10       Code
                                     UPPER = 11
```
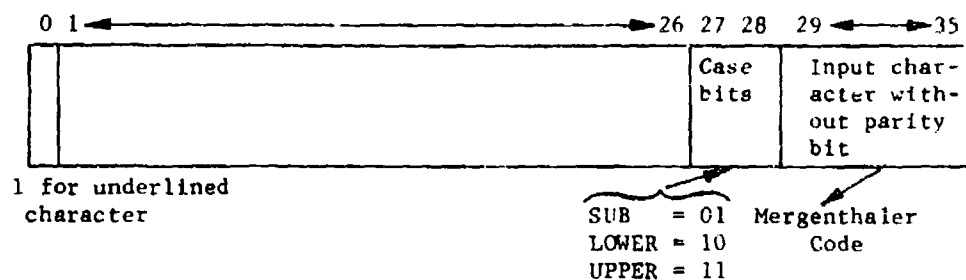

MATRIX CONTENTS AT END OF ORGNZR:

```
                    20           30 ◄────► 35
┌─────────────────┬─┬───────────┬──────────┐
│                 │ │           │   Dura   │
│                 │ │           │   code   │
└─────────────────┴─┴───────────┴──────────┘
                  ▲
               Case bit
```
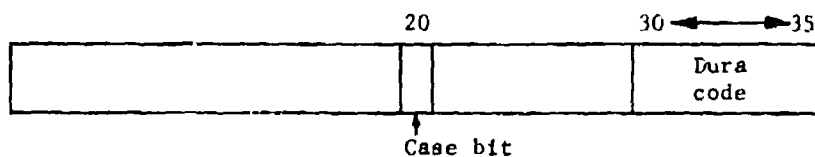

BXBRAK - table of x coordinates of right hand brackets in structure
         last x coord is = DELX

```
                    18 ◄──────────────► 35
┌──────────────────┬───────────────────┐
│                  │   x coordinate     │
├──────────────────┼───────────────────┤
│                  │                    │
└──────────────────┴───────────────────┘
```
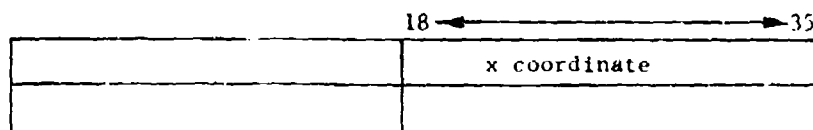
MULTAB AT END OF ORGNZR:

```
                     18↔21 22◄─────────►35
┌─────────────────────┬────┬─────────────────┐
│                     │    │   Multiplier    │
└─────────────────────┴────┴─────────────────┘
```
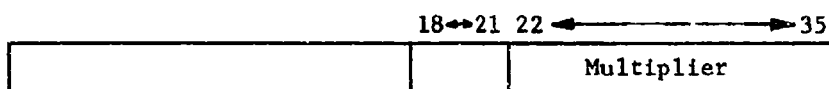
last multiplier is = to 1

MULTAB AT END OF REGRUP:

```
   3◄─────────────►17 18↔20 21◄────────►35
┌─┬───────────────────┬────┬───────────────┐
│ │                   │    │  Multiplier   │
└─┴───────────────────┴────┴───────────────┘
```
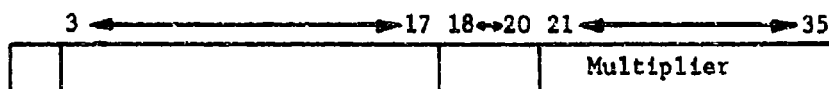
2's comp. pointer to
last entry in SCRUB
list for this Mul-
tiplier (and this set
of brackets)

XTAB - a table formatted as follows:

```
   3◄─────────────►17    24◄──────────►35
┌─┬───────────────────┬────┬───────────────┐
│ │  Total atoms of this │  │   Element    │
│ │      element      │    │               │
└─┴───────────────────┴────┴───────────────┘
```
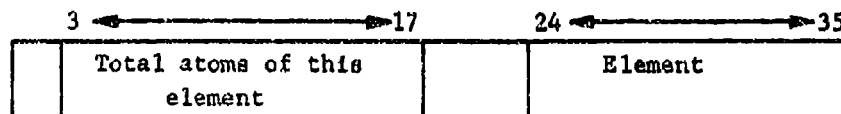
if a single element,
the second character
is a BCD blank

276

CT - tne internal Connection Table whose entries are formatted as follows:

	3 ⟷ 17	18-20	21-23	24 ⟷ 29	30 ⟷ 35
	Number of atom bonded to	Multi-plier	Bond type	2nd letter of atom	1st letter of atom

The multiplier points to an entry in the list of bracket multipliers (MULTAB, described in Section 2.2.7.2) where applicable. Each atom has 8 such entries only the first of which contains the atom name. The second and third contain the relative matrix location of this atom as follows:

	3 ⟷ 17	18-22	21-23	24 ⟷ 27	28 ⟷ 35
	SAME AS ABOVE		Bond type	zeroes	

Word 2 for a given atom 3 Low order digits of relative matrix location

	3 ⟷ 17	18-20	21-23	24 ⟷ 29	30 ⟷ 35
	SAME AS ABOVE		Bond type	zeroes	

Word 3 for a given atom: 2 high order digits of relative matrix location.

ex. relative matrix location 14321 is represented as:

	321

	14

CHEMTYPE OUTPUT RECORD FORMAT

	S↔2 3◄───────►17 18↔20 21◄──────►29 30◄──►35

	S↔2	3 ◄──────────────────► 17	18↔20	21 ◄──────────► 29	30 ◄──►35
WORD 0		No. of words in block not including Word 0	010		F (BCD)
WORD 1		2's complement of first location of MOLTAB		2's complement of 1st word of connection table	
WORD 2		2's complement of first location of abnormality table (0 if not present)		2's complement of first location of nomenclature	
WORD 3		2's complement of first location of SCRUB list		Number of RINGS	

Locations to WORD 0

WORD 4	REGISTRY NUMBER (first 6 characters)
WORD 5	REGISTRY NUMBER (second 6 characters)
WORD 6	CLSTER (Classification and Stereo)
	MOLTAB BLOCK
	CONNECTION TABLE BLOCK
	ABNORMALITY TABLE BLOCK (if present)
	NOMENCLATURE & REFERENCE BLOCK

	S↔2	3↔5	◄──────────►17	◄────►25	26 ◄────────►34	35
SFI Header	-	+		DELY	DELX	/

Set if underline table exists

Charges outside of brackets (total)

SFI BLOCK (SCRUB LIST)

The output tape which is created by this system consists of variable length records, each record consisting of a single chemical record.

278

CHEMTYPE DATA FORMATS

FORMAT OF CLSTER:

		15 ⟷ 17 Stereo		30 ⟷ 35 Classifi-cation

1 = Non-Stereo   1 = unclassified

2 = Stereo   2 = blank

3 = Stereo Unidentified   3 = classified

0 = not present

SCRUB LIST FORMAT:

	3 ⟷ 17 Relative matrix loc.		20		30 ⟷ 35 Modified dura code

Case bit

279
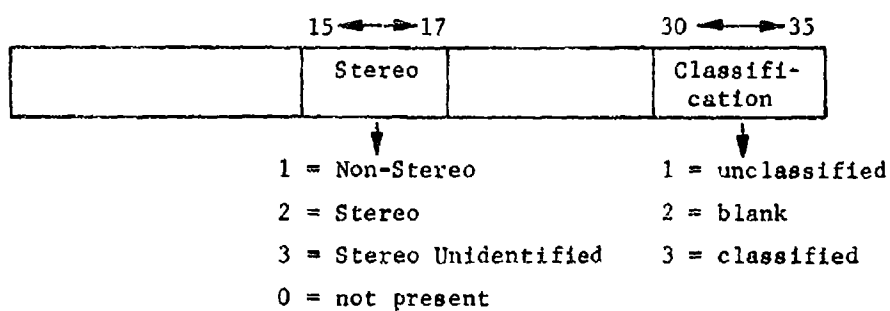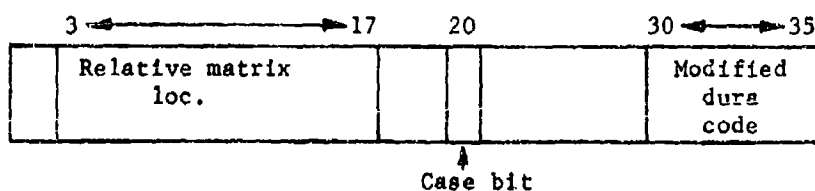
NOMENCLATURE FORMAT:

Each nomenclature entry (one line of nomenclature) is delimited by a 777_8 character from the next. The last nomenclature word is filled out with zeroes. References are separated from each other by a 777_8 character and the beginning of the reference field is preceded by a triple bond character in the output block. The last word in the reference field is also filled with zeroes.

3 ⟵⟶ 17		21 ⟵⟶ 35	
2's comp. of number nomenclature words		2's comp. of total number of words in block	Header of nomenclature block

0 ⟵⟶ 8	9 ⟵⟶ 17	18 ⟵⟶ 26	27 ⟵⟶ 35	
				4 characters per 7040 word

			6 Bit Dura Code	9 Bit character

Superscript   Underline   Dura case bit
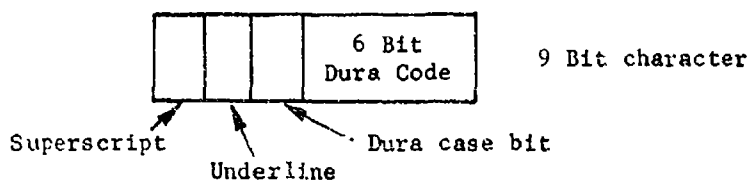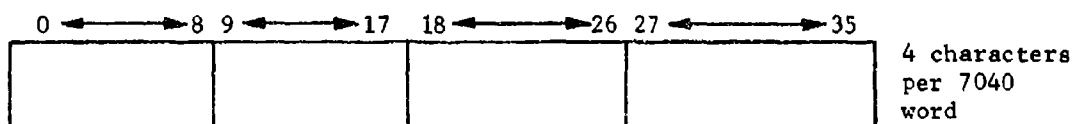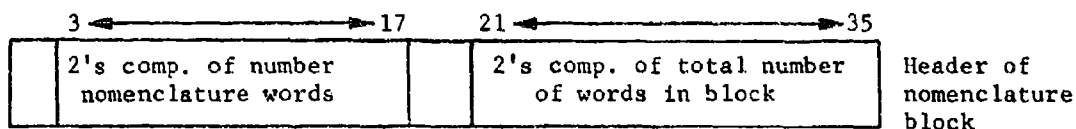
REFERENCE BLOCK FORMAT

(OUTPUT OF UPTAP)

WORD	3 - 17	18 - 20	21 - 35
0	No. of words in table of contents		No. of words in reference block (including Word 0)
1	1**	1*	RA to CLSTER Block
2	2**	3*	RA to Nomenclature Block
3	3**	0*	RA to EA Number Block
4	CLSTER		
5	Nomenclature		
.			
.			
x	EA Number (S)		

NOTE:

* Type of Data

    0 BCD
    1 Binary
    2 Modified Dura
    3 Compressed Modified Dura

** If Decrement is zero
no data is stored

ORDER OF DATA CARDS FOR CIDS

FRAGMENTS AS INPUT TO SLOAD

REPLAC002000

C   (Title card)

(Hydrocarbon Radicals)

    .

    .

REPLAC001000

CNOP   (Title card)

(Functional Groups Containing C,N,O,P)

CNOS

(Functional Groups Containing C,N,O,S)

    .

    .

    .

(Rest of Functional Groups)

    .

    .

000000   (End card)

     The input for each structural fragment consists of data cards containing the following information:

> Key Number
>
> Molecular Formula
>
> Connection Table
>
> Abnormalities (if any)

The key number is a 6 character number punched in the first 6 columns of the card. The formats for the mol form and C.T. cards can be found in CIDS No. (pages 164 and 165) with the exception that column 24 of the first mol form card now indicates whether any abnormalities are present. Each abnormality has the form "XY=Z". Where X is the abnormality type, Y is the atom number, and Z is the value of the abnormality. The abnormality types are V (Valence) C (Charge), M(Mass). Examples: V1=5.C1=+1.M5=14.

     The data for the next fragment follows immediately except when control cards are needed to separate groups or blocks.

OUTPUT OF SLOAD

The output of SLOAD is a tape on which the first physical record contains the fragment data. The format of the data associated with each fragment is stored as follows:

Word	Contents
1	D=No. of words in fragment record A=No. of words preceding structure (m)
2	D=No. of words preceding abnormality table (=0 if no abnormalities) A=No. of words in structure
3	Molecular formula
.	.
.	.
m	Key number
m+1	Connection table
.	.
.	.
.	.
n	Abnormality table (if needed) (a zero word follows last entry)

The index is written as the second record on the output tape. The decrement of the first word of this record contains the complement of the number of words in the index. This word is to be read into a location CROSCT, immediately preceding CROSS when the tape is read for screen assignment.

INTERNAL KEY FORMATS

Each of the CIDS keys occupies two computer words. The first 3
bits of the first word gives the key type, enabling the programs which
process these keys to interpret the information in the remainder of
the two words. The present CIDS keys have the following formats:

TYPE 0:   Structural Fragment Keys

S,1,2	3-35	S,1-34	35
000	BCD Code	0 —————— 0	0 1

Bit 35 $\Big\{$ = 1 if fragment is attached to a ring
Word 2 $\Big\}$ = 0 if fragment is not attached to a ring

Example: Key 1-A-3 is stored internally as the following (in BCD):

	S,1-35					S,1-35					
0	1	A	0	0	3	0	0	0	0	0	0

TYPE 1:   Skeleton Molecular Formula

S,1,2	3	4-10	11-17	18-23	24-29	30-35	S,1-5	6-11	12-17	18-23	24-29	30-35
001	*	C	N	O	S	P	Code 1	Amt 1	Code 2	Amt 2	Code 3	Amt 3

In this key, the number of atoms of C,N,O,S,P are stored in fixed
positions in Word 1. Codes from a table are stored in Word 2 for
other elements occurring in the nucleus, followed by the number of
atoms of that element. Up to three of these "other elements" may
occur and they are stored in alphabetical order.

*Types 1, 2, 3:   Bit 3 of word 1 $\Big\{$ = 0 if 2nd word is unused
 = 1 otherwise

284

Example: Key "SKELMF C12 N1 Sb1" is stored internally as the following binary representation:

S,1,2	3	4-10	11-17	18-35	S,1-5	6-11	12-35
001	1	0001100	0000001	0 —— 0	Code for Sb	000001	0 ——— 0

TYPE 2: Ring Molecular Formula

This key is stored in the same format as type 1 except that bits (S,1,2) of word 1 contain 010.

TYPE 3: Redundant Numerical Ring Population

S,1,2	3	4-7	8-11	12-15 ...				32-35
011	*	R1	R2	R3	R4	R5	R6	R7	R8

S-3	4-7	...						32-35
R9	R10	R11	R12	R13	R14	R15	R16	R17

The count of atoms in each ring of a nucleus is stored in ascending order in consecutive 4-bit blocks in the key. If a ring contains more than 15 atoms, 0001 is stored as the atom count.

Example: Key "RNRP 5,6,10" is stored internally as the following binary representation:

S,1,2	3	4-7	8-11	12-15	16-35	S,1-35
011	0	0101	0110	1010	0 —— 0	0 ——— 0

*Types 1, 2, 3: Bit 3 of word 1 $\begin{cases} = 0 \text{ if 2nd word is unused} \\ = 1 \text{ otherwise} \end{cases}$

TYPE 4:   Counts


```
S,1,2    3-6      7-35              S,1-35
┌─────┬───────┬────────┬────────────────────────┐
│ 100 │ Sub-  │ Count  ││ 0 ───────────── 0      │
│     │ type  │        ││                        │
└─────┴───────┴────────┴────────────────────────┘
```


This format is used to represent a variety of keys which are counts
The subtype field identifies the particular feature that is being
counted.

Subtype	Count of:
0	Rings in nucleus
1	Double bonds in nucleus
2	Nuclei in Molecule
3	Total number of direct attachments to all nuclei
4	Double bonds between C (acyclic)
5	Triple bonds between C (acyclic)

6   C-C-C configurations regardless of bonding (acyclic)
          $\quad\;$ |
          $\quad\;$ C

7   C—C-C configurations (acyclic)
          $\quad\;$ |
          $\quad\;$ C

8         $[C]_n$ — El count of carbons connected by
          $\qquad\qquad$ single bonds, any configuration.

9         $El-[C]_n$ — El count of carbons in unbranched
          $\qquad\qquad\qquad$ chain (single bonds)

10   Total ring count (Summed over all nuclei in
          parent and addends, if any)

Example: Key "Number 2 3" or "Nuclei in Molecule = 3" is stored
internally as the following binary representation:


```
S,1,2     3-6       7-35             S,1-35
┌───────┬────────┬──────────┬──────────────────────────┐
│ 100   │ 0010   │ 0 ── 011 ││ 0 ──────────────── 0     │
└───────┴────────┴──────────┴──────────────────────────┘
```

TYPE 5:   Subtype 0:   Molecular Formula Key

S,1,2	3-5	6-17	18-35	S,1-35
101	000	Element (BCD)	Count or 0	0 ——————— 0

This count of the number of atoms in the Hill formula is gi·en for
elements C,H,N,O,P,S,F,Cl,Br,I,Si,B. For other elements, 0 is stored
instead of the count.

Example: Key "MOLFRM C 10" is stored internally in the following
binary representation:

S,1,2	3-5	6-17	18-35	S,1-35
101	001	110000010011	0 — 01010	0 ——— ——— 0

$\underbrace{\qquad\qquad}$
C in BCD

TYPE 5:   Subtype 1:   Nonspecific Functional Group

S1,2	3-5	6-23	24-35	S,1-35
101	001	0 —— 0	Element (BCD)	0 ——————— 0

Example: Key 'NONSPC AS' is stored internally in the following
binary representation:

S,1,2	3-5	6-17	24-35	S,1-35
101	001	0 —— 0	010001110010	0 ——— 0

$\underbrace{\qquad\qquad}$
As in BCD

LIST-STRUCTURED FILE GENERATION

Programs NUFILE, KEYSRT, MERGE, and INDEX together create or update the search file and form the inverted key index. The formats and a brief description of the final output tapes from this system are listed here for easy reference.

(1) The Tape Search File

The compound records in the Tape Search File are blocked in variable length physical records whose maximum size is 1000 words. A compound record is always entirely contained within one physical record. The information on each tape in the Search File, except the last, is terminated by an end-of-file mark followed by a small (10 word) dummy block. The last tape of the File is terminated by two consecutive end-of-file marks and a special ten word block containing information which NUFILE uses when updating the File. The first word of the special block contains the address which will be assigned to the next compound to be added to the File. The second word contains the total number of compounds in the File.

Word	Bits	Contents
1	(S,1-5)	Tape Number (1-)
	(6-18)	Record Number (0-)
	(19-35)	Relative Address (0-)
2		Number of Compounds in the file (right adjusted)

(2) The Disk Search File

The Compound records in the Disk Search File are blocked in 465 word physical records. Compounds may be split between two physical records, but never more than two. The end-of-tape and end-of-file sentinels are the same as for the Tape Search File, except that the special ten-word record at the end of the File contains the following information:

Word	Bits	Contents
1	(S,1-17)	Record (track) Number (1-)
	(18-35)	Relative Address (0-464)
2		Number of unused words in last data record written on the tape (right adjusted)

(3) **List-of Addresses File**

The address-lists on this file are blocked in variable length records whose maximum size is 1000 words. The data on each tape except the last is terminated by one end-of-file mark followed by a ten-word dummy record. Two consecutive end-of-file marks terminate the last tape in the file. Each address-list (i.e., all the Search File addresses corresponding to a single key) is followed by a word of zeros, and the addresses comprising these lists are in ascending order.

The format of an address in the Index for the Tape Search File is:

Bits	Contents
(S,1-5)	Tape No. (1-)
(6-18)	Record No. (0-)
(19-35)	Relative Address (0-)

The format of an address in the Index for the Disk Search File is:

Bits	Contents
(S,1-17)	Track No. (1-)
(18-35)	Relative Address (0-464)

(4) **The Key-Address List and INDX**

The Key-Address List contains each key (2 words) coupled with the address of its corresponding list on the List-of-Addresses File (1 word). The format of this address is:

Bits	Contents
(S,1-17)	Track Number (0-)
(18-35)	Relative Address (0-464)

This data is blocked in 465 word physical records. Since each logical record is three words, there can be as many as 155 keys per block (or track). The last key on this file is followed by a special sentinel "key" composed of two words of all 1 bits.

Each Key-Address List tape, except the last, is terminated by an end-of-file mark and a dummy block. The Key-Address List data on the last tape in the file is terminated by two consecutive end-of-file marks, directly followed by the third level of the Inverted Key List, INDX. INDX identifies the first key on each track of the Key-Address List in order to provide quick access to the desired key list. INDX is always small enough to place to tape in one block (e.g., 1000 words would accomodate a file containing over 50,000 different keys).

The logical record format of INDX is:

Word	Bits	Contents
1	(S,1-35)	Key (1st half)
2	(S,1-35)	Key (2nd half)
3	(S,1-17)	Track (1-)

The last key to appear on INDX will naturally be the special sentinel.
Since the last track on the Key-Address List may not be completely
filled, the address corresponding to the sentinel key will not neces-
sarily be word 462 on that track as it would normally be for the last
key on a track.

OUTPUT OF MOLE

MOLE generates the following block of data containing the molecular formula, connection table, and abnormality table:

Word	Contents
1	A=No. of words preceding the C.T. (X+2)
	D=Total number of words (X+Y+Z+2)
2	A=No. of words in the C.T. (Y)
	D= No. of words preceding the Abnormality Table (X+Y+2 or 0 if no abnormalities)
3	Molecular Formula
.	(X words)
.	
X+2	Connection Table
	(Y words)
X+Y+2	Abnormality Table
	(Z words)

The molecular formula is stored in the same format as the Hill formula for a file compound. The connection table and abnormality table has the same format as in the file compound except that redundancy has been removed from the C.T.

MOLECULAR FORMULA IN QUERY

Word of Formula	Bits	Contents
First Word	3-17	Number of words in molecular formula
	19	= 1 if restricted search
		= 0 if otherwise
Additional Element Words	0-11	Element (BCD)
	12-20	Maximum number of atoms if range search Number of atoms if exact search = 0 otherwise
	21-28	Minimum number of atoms if range search = 0 otherwise
	34	= 1 if search of atoms is a range search = 0 otherwise
	35	= 1 if search of atoms is an exact match search = 0 otherwise

WORD FORMAT FOR RECONSTRUCTED PICTURE (STRUCTURE OUTPUT)

```
     S ◄──────────────────► 28   29    30 ◄───► 35
    ┌────────────────────────┬────┬──────────────┐
    │ 0 ──────────────── 0   │    │   6 Bit      │
    │                        │    │  Dura Code   │
    └────────────────────────┴────┴──────────────┘
                                 └──► Case Bit
```
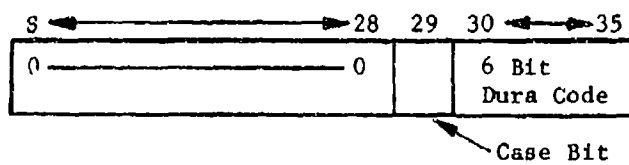
SEARCH SYSTEM OUTPUT

QUERY NUMBER CIDS I

RN A0000514   *CIDS Registry Number*

TN T03603   ⎫   *Identification Numbers*
TN X00000637   ⎬   *in File of Origin*

$C_{15}H_{12}N_2$ *Molecular Formula*

Structure
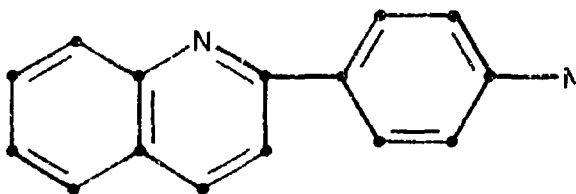
Quinoline, 2-(p-aminophenyl)-   *Nomenclature*

TF7087   ⎫   *Reference Numbers*
EAI698   ⎬

STEREO N CODE U   *Stereo and Security Classification*

DC25440   *Edgewood Arsenal*
                                 Document Code

ERROR DETECTION AND ANALYSIS BY CHEMTYPE

Compounds are rejected during processing by CHEMTYPE as a result of 73 different error conditions. Errors are detected by almost every individual program. The messages and a description of the conditions that cause them are described below.

It is difficult, however, to have an absolutely crystal clear interpretation of the causes of the errors. In many instances, errors may be caused by the paper tape reader during the transfer of the paper tape image to magnetic tape. This may result in bits being dropped or added to a character (or characters) and the program may reject the compound for a reason not ascribed directly to the paper tape reader. In other words, in the case of such errors, a garbled record results and the true reason for the error is not reflected by the program printout.

ERROR MESSAGES

The following list gives the error messages generated by the CHEMTYPE system, the name of the program that detects them, and a description of the error itself.

(1) <u>Number of parity errors since last compound entered.</u> (TAPWRM)

The total parity errors found in Mergenthaler input between two good compounds is printed when each correct compound is entered into the file. This may be due to a typewriter malfunction, an error in the paper tape reader in which a bit is dropped or added, or an error in the procedure followed by the typist in correcting a parity error.

(2) <u>Parity error in coordinate input. No code delete found.</u> (TAPWRM)

Either a paper tape read error occurred resulting in parity error or the typist did not correct a parity error in the proper way.

(3) <u>Low bit not punched in coordinates.</u> (TAPWRM)

Parity error found in the coordinates was a result of the low bit not being punched.

(4) <u>Typist goofed again. Last coordinate word not equal zero</u> (TAPWRM)

If the typist hits a character too quickly after doing a carriage control operation which results in coordinates, the character may land in the middle of the coordinates since each set of coordinates consists of 6 punches.

(5) <u>Coordinate bit 2 or 3 punched.</u> (TAPWRM)

Punch appears in coordinates where it shouldn't as a result
of a hardware (typewriter punch error), a paper tape read
error, or a typing error. The lat...r is caused by the ...pist
responding incorrectly to typewriter parity light.

(6) <u>Unidentified input character.</u> (TAPWRM)

Character does not correspond to any legitimate Mergenthaler
code. May be due to paper tape read error which does not
result in parity error or to a mispunch by the typewriter.

(7) <u>Undefined symbol in record.</u> (INPUTD)

The Dura punched an illegal code.

(8) <u>Overflowed MATRIX erasing brackets.</u> (ORGNZR)

Coordinate error resulting in misplaced characters in MATRIX.

(9) <u>Unidentified character found in nomenclature.</u> (MONIKR)

Input information was in error and may have been mispunched.

(10) <u>Overflowed nomenclature block.</u> (MONIKR)

The nomenclature information was longer than 400 characters.

(11) <u>Unable to identify character found in MATRIX.</u> (ORGNZR)

Input is an unintelligible character.

(12) <u>Brackets contain character other than bond or corner.</u> (ORGNZR)

Wrong character in brackets, due to mispunch.

(13) <u>Input exceeds 10000 MATRIX.</u>(TAPWRM)

Either input record was too large, or the typist typed a
lozenge and then reverse indexed above lozenge and typed a
character. The latter would result in a y coordinate larger
than 100 when the y coordinate is corrected on the basis of the
lozenge coordinate =1. This could also have been caused by
a tape reader error.

(14) <u>Compound too large for MATRIX.</u> (INPUTD)

Compound exceeds dimensions of MATRIX.

     height: 100

     length: 82

(15) <u>Tape reading problem.</u> (TAPWRM)

Physical problem occurred while reading tape resulting in an incomplete word being read during Mergenthaler input.

(16) <u>Magnetic Tape read error.</u> (INPUTD)

Magnetic tape reading problem in Dura Mach input.

(17) <u>First character found in MATRIX not number or letter.</u> (ORGNZR)

The first character of TID must be the first character found in MATRIX and must be a number or letter.

(18) <u>Space not found after 12 registry number characters.</u> (ORGNZR)

TID is too big, or typist did not skip a space.

(19) <u>No match found in table of classification.</u> (ORGNZR)

Classification information is unintelligible.

(20) <u>Classification found to have more than 8 characters.</u> (ORGNZR)

Classification information is unintelligible.

(21) <u>Structure extends past STEREO field.</u> (ORGNZR)

Typist went too far down in record and part of the compound extends past the STEREO line of the input.

(22) <u>No match in table for STEREO information.</u> (ORGNZR)

STEREO information is unintelligible.

(23) <u>Addend molform missing.</u> (MOLFRM)

The structure is bracketed and the addend molform is not present; there is no charge present to indicate that the structure is an ion.

(24) <u>Character in molform wrong.</u> (MOLFRM)

There is a syntax error in molform, or input tape was mispunched.

(25) <u>Character following blank in molform not a dot.</u> (MOLFRM)

There was a syntax error in the molform or input tape was mispunched.

(26) Cannot reorder scrub list. (ORGNZR)

This is an addend but the bracket is missing.

(27) Addend presents problem. Too many characters for list. (REGRUP)

There was either an excessively long addend or coordinates were wrong and other information got into the structure by mistake.

(28) Too many entries for scrub list. (ORGNZR)

Structure consisted of more than 700 characters.

(29) Nomenclature field missing. (ORGNZR)

No nomenclature field was found.

(30) Structural formula too high. (ORGNZR)

Structure extended into molform line.

(31) STEREO field missing. (TAPWRM)

Typist either left out STEREO information, or somehow erased the S as a result of making a correction incorrectly.

(32) STEREO not typed after exit from S.F. field. (INPUTD)

Either program did not find stereo text, typist failed to type STEREO information, or the typewriter mispunched.

(33) Platen reversed above start of record. (INPUTD)

Typing occurred above the leading wedge.

(34) Unknown character outside brackets. (EXCESS)

There was an input error due to a mispunch or a syntax error.

(35) Small letter outside of brackets. (EXCESS)

Formula outside of brackets begins with a small letter due to a syntax error.

(36) Polymer molform subscript not n. (MOLFRM)

Error in input.

(37) Fractional addend multiplier. (MOLFRM)

The multiplier of an addend is a fraction and cannot be handled in verification.

298

(38) <u>Nothing found outside brackets by EXCESS.</u> (EXCESS)

No characters found where they were expected. Input error.

(39) <u>Error in typing structural formula.</u> (CLEANM)

Any violation of the typing conventions for the structural formula.

ex.   C—C—C
            |
            C
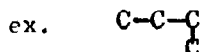
(40) <u>Inadmissible linear string in S.F.</u> (CLEANM)

(a) Unexpanded chemical line notation
     ex.   —CO—

(b) Confusion due to closeness of atoms

     ex.   —ClCl—   or   ClCl
                          C C
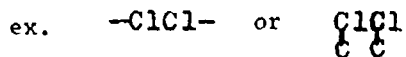
(41) <u>Bond in wrong place.</u> (SETUP)

ex.   —C//

(42) <u>Picture too scrunched up.</u> (MAKECT)

Analysis problem caused by closeness of characters.

It is difficult to determine which bond belongs to upper right carbon.

(43) <u>Illegal symbol around atom.</u> (CLEANM)

An illegal symbol has been detected in one of the eight locations surrounding an atom.

ex.   C
       +

(44) <u>H in wrong place.</u> (CLEANM)

Unexpanded hydrogen connection.

ex.   —CH₃—   or   OH
                    C

299
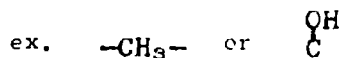
(45) <u>Non-straight attachment to carbon chain.</u>   (MAKECT)

ex. $-(C)_6$

$-(C)_6-$

$(C)_6$

(46) <u>Illegal symbol in structural formula</u> (SETUP)

Symbol in structural formula other than bond, atom, number, charge, bracket corner, or mid-line dot.

(47) <u>Reject symbol typed.</u>   (INPUT)

Typist pressed Ⓢ key to delete record typed thus far for Dura Mach input.

(48) <u>Box found. Record deleted.</u>   (TAPWRM)

Typist pressed □ key to delete record typed thus far for Mergenthaler input.

(49) <u>Format error detected by CONVRT.</u>   (CONVRT)

The compound is unprocessable by the CIDS system.

(50) <u>More than 19 abnormalities.</u>   (CLEANM)

The Abnormality Table (AT) has too many entries.

(51) <u>Bond redundancy error.</u>   (CONVRT)

Atom A is connected to atom B, but atom B is not connected to atom A in the connection table.

(52) <u>Incorrect symbol in bits 24-35 of CT.</u>   (NFCF)

An illegal symbol has been found in the connection table.

(53) <u>Empty connection table.</u>   (NFCF)

There are no atoms listed in the connection table.

(54) <u>Virgule found, fraction not followed by space.</u>   (TAPWRM)

Since the virgule is a non-spacing character, the typist must leave a space after a fraction to allow for spreading the fraction apart in the MATRIX.

(55) <u>Illegal character in MATRIX. Program Error.</u>   (CLEANM)

Character in MATRIX greater than 177.

(56) Registry number does not start with letter. (ORGNZR)

(57) Illegal Character in Registry number. (ORGNZR)

Chemical Verification Errors encountered by VERIFY:

CV(21) Illegal element was found in molecular formula.

CV(22) Illegal element symbol was found in the Connection Table.

CV(23) An element was found in the Connection Table having a
        valance too high to be valid for this element.

CV(24) Molecular formula was found to contain no Carbon.

CV(25) The multiplier of the first addend in the addend molecular
        formula was found to be zero.

CV(26) Hydrogen was found in the Connection Table.

CV(27) The assumed Hydrogen count for the compound was different
        from the Hydrogen count in the molecular formula.

CV(30) Illegal unknown attachment was found.

CV(31) Atom count for C,H,N, or O in Connection Table did not
        equal that in molecular formula.

CV(32) Illegal element symbol was found among elements not
        included in Connection Table.

CV(33) Total element count for non C, H, N, or O not equal to
        that in Connection Table.

CV(41) Illegal valence found in the abnormality table.

CV(42) Total atom count for C, H, N, or O in addend molecular
        formula not equal to that in the Hill molecular formula.

CV(43) Count for non C, H, N, or O elements in addend molecular
        formula is not equal to that in the Hill molecular
        formula.

CV(44) Multiplier for Hill parent of a hydrate is a fraction and
        cannot at present be handled by verification.

CV(45) Total plus charges in the molecule do not equal the total
        minus charges.

BLANK PAGE

APPENDIX F

MERGENTHALER CHEMICAL TYPEWRITERS CODES

OCT. CODE	SUB CASE	LOWER CASE	UPPER CASE
245	‖	ı	‖
254	?	,	—
55	—		\
56	Δ	.	=
60	)	O	o
261	⌠	1	ı
262	°	2	2
63	:	3	3
264	%	4	4
65	+	5	5
66	—	6	6
267	⊤	7	7
270	*	8	8
71	(	9	9
273	⌐		ı
77	•		≈
101	α	a	A
102	β	b	B
303	~	c	C
104	δ	d	D
305	ï	e	E
306	⌡	f	F
107	γ	g	G
110	⌐	h	H
311	/	i	I
312	⌐	j	J
113	∦	k	K
314	λ	l	L
115	μ	m	M
116	~	n	N
317	ω	o	O

OCT. CODE	SUB CASE	LOWER CASE	UPPER CASE
120	⌐	p	P
321	□	q	Q
322	.	r	R
123	◇	s	S
324	i	t	T
125	\	u	U
126	⌐	v	V
327	●	w	W
330	ξ	x	X
131	—	y	Y
132	⌐	z	Z
135	⊖		/

OCT. CODE	CONTROL CHARACTERS
000	Null code
377	Code delete
201	Power on
210	Backspace
11	Tab
12	Line feed
215	Carriage return
216	Upper case
17	Sub case
24	Stop code
30	Coord. follow
232	Lower case
234	Ribbon shift – White
35	Ribbon shift – Black
240	Space

DURA MACH CHEMICAL TYPEWRITER CODES

OCT. CODE	UPPER CASE	LOWER CASE	OCT. CODE	UPPER CASE	LOWER CASE	OCT. CODE	SPECIAL CHARS.
						000	Stop
134	A	a	177	/	!	002	Carriage return
140	B	b	166	//	2		
154	C	c	176	/	3	004	Space
155	D	d	171	‖	4	010	Upper Case
145	E	e	165	ø	5	020	Lower Case
116	F	f	164	ǀ	6		
117	G	g	175	— :	7	040	Backspace
141	H	h	174	=	8	001	Tab
124	I	i	160	\	9	200	Delete
107	J	j	161	⦰	0	003	Nonprint
144	K	k	114	*	,	006	Print restore
151	L	l	126	⁄	.	050	Skip restore
137	M	m	113	:	<		
146	N	n	100	\	+	016	Punch on
131	O	o	111	↖	®	012	Punch off
105	P	p	127	o	-	014	Index
104	Q	q	125	•	>	030	Reverse Index
135	R	r	106	(	)		
121	S	s					
147	T	t					
156	U	u					
136	V	v					
120	W	w					
157	X	x					
101	Y	y					
167	Z	z					

CHEMICAL LINE PRINTER CODES

OCTAL CODE	CHARACTER (Upper Case)	CHARACTER (Lower Case)	OCTAL CODE	CHARACTER (Upper Case)	CHARACTER (Lower Case)
00	√	{	40	SPACE	SPACE
01	A	a	41	!	δ
02	B	b	42	"	ε
03	C	c	43	#	λ
04	D	d	44	$	μ
05	E	e	45	.	%
06	F	f	46	&	ξ
07	G	g	47	≥	'
10	H	h	50	≥	(
11	I	i	51	≡	)
12	J	j	52	*	π
13	K	k	53	+	ρ
14	L	l	54	‖	,
15	M	m	55	-	σ
16	N	n	56	—	.
17	O	o	57	/	ω
20	P	p	60	0	0
21	Q	q	61	1	1
22	R	r	62	2	2
23	S	s	63	3	3
24	T	t	64	4	4
25	U	u	65	5	5
26	V	v	66	6	6
27	W	w	67	7	7
30	X	x	70	8	8
31	Y	y	71	9	9
32	Z	z	72	∫	:
33	Δ	[	73	∞	;
34	\	α	74	≠	<
35	|	]	75	=	∘
36	‖	}	76	®	>
37	∑	γ	77	?	β

305

CONTROL CODES

OCTAL	DEFINITION
15	Shift to upper case
17	File mark
35	Shift to upper case for one character
36	Function code: means a Control Code follows
55	Shift to lower case
72	End of line
75	Shift to lower case for one character

Notes:

1. A function code must precede every control code.

2. The function code defines the next character as a control code except where the function code is followed by an end of line, then the second character is also a control code.

3636 means print ||| or } depending on shift.

LINE SPACE CONTROLS

20	Space 0 lines at six lines per inch
21	Space 1 line at six lines per inch
22	Space 2 lines at six lines per inch
23	Space 3 lines at six lines per inch
24	Space 4 lines at six lines per inch
25	Space 5 lines at six lines per inch
26	Space 6 lines at six lines per inch
27	Space 7 lines at six lines per inch
40	Skip to channel 0 (6 lines per inch)
41	Skip to channel 1 (6 lines per inch)
42	Skip to channel 2 (6 lines per inch)
43	Skip to channel 3 (6 lines per inch)
44	Skip to channel 4 (6 lines per inch)
45	Skip to channel 5 (6 lines per inch)
46	Skip to channel 6 (6 lines per inch)
47	Skip to channel 7 (6 lines per inch)
60	Space 0 lines at twelve lines per inch
61	Space 1 line at twelve lines per inch
62	Space 2 lines at twelve lines per inch
63	Space 3 lines at twelve lines per inch
64	Space 4 lines at twelve lines per inch
65	Space 5 lines at twelve lines per inch
66	Space 6 lines at twelve lines per inch
67	Space 7 lines at twelve lines per inch

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)	2a. REPORT SECURITY CLASSIFICATION
UNIVERSITY OF PENNSYLVANIA Philadelphia, Pennsylvania 19104	UNCLASSIFIED **2b. GROUP** NA

3. REPORT TITLE

COMPUTER PROGRAMMING FOR AN EXPERIMENTAL CHEMICAL INFORMATION AND DATA SYSTEM

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

Status Report, CIDS No. 5 - January 1967-January 1968

5. AUTHOR(S) (First name, middle initial, last name)

Lefkovitz, David, Powers, Ruth V., and Hill, Helen N.

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
June 1968	312	6
8a. CONTRACT OR GRANT NO. DA-18-035-AMC-288 (A) **b. PROJECT NO.** **c.** Task: 2P062101A72702 **d.**	**9a. ORIGINATOR'S REPORT NUMBER(S)** CIDS-5/STATUS REPORT **9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)**

10. DISTRIBUTION STATEMENT This document is subject to special export controls and each transmittal to a foreign national or a foreign government may be made only with prior approval of the Commanding Officer, Edgewood Arsenal, ATTN: SMUEA-TSTI-T, Edgewood Arsenal, Maryland 21010

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY Edgewood Arsenal
Army chemical and information data systems	Technical Support Directorate, Edgewood Arsenal, Maryland 21010 (Stanley Goldberg, Proj. O., Ext 6126)

13. ABSTRACT This report contains computer program documentation of a chemical structure information storage and retrieval system. The primary data source of this system is a drawn structural formula along with auxiliary data, such as a local control number, molecular formula, and nomenclature. The system can also accept magnetic tape input from the CAS registry system. The documentation of the programs is both functional and operational and flow charting is limited to macroflow charts. The documentation is presented at three levels: First, a general description of the total system is given; then a general description is given of each major subsystem; and finally a detailed functional and operational description format is given for each program in the subsystem.

DD FORM 1473 REPLACES DD FORM 1473, 1 JAN 64, WHICH IS OBSOLETE FOR ARMY USE.

311

UNCLASSIFIED
Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Atom-by-atom search						
Batched search system						
Chemical compounds						
Chemical information						
Chemical structure						
Chemical typewriters						
Chemical verification						
CHEMTYPE system						
Computer						
Computer programs						
Connection table generation						
Connection tables						
Consoles						
Data processing						
File construction						
File organization						
Flow charts						
List search implementation						
List-structured file						
Molecular formulas						
Nomenclature						
Nonstructural information						
Queries						
Query format						
Real time retrieval						
Registration of chemical compounds						
Retrieval						
Search techniques						
Software						
Structural diagrams						
Structural formula image						
Structural input						
Structural output						
Structural screens						
Substructure search						
System configuration						