

## **General Disclaimer**

### **One or more of the Following Statements may affect this Document**

- This document has been reproduced from the best copy furnished by the organizational source. It is being released in the interest of making available as much information as possible.
- This document may contain data, which exceeds the sheet parameters. It was furnished in this condition by the organizational source and is the best copy available.
- This document may contain tone-on-tone or color graphs, charts and/or pictures, which have been reproduced in black and white.
- This document is paginated as submitted by the original source.
- Portions of this document are not fully legible due to the historical nature of some of the material. However, it is the best reproduction available from the original submission.

**NASA CONTRACTOR  
REPORT**

**NASA CR-150405**

**MINIS - MULTIPURPOSE INTERACTIVE NASA INFORMATION SYSTEM**

**By Computer Sciences Corporation  
8300 Whitesburg Drive, South  
Huntsville, Alabama 35802**

**December 1976**



**Prepared for**

**NASA - George C. Marshall Space Flight Center  
Marshall Space Flight Center, Alabama 35812**

**(NASA-CR-150405) MINIS: MULTIPURPOSE  
INTERACTIVE NASA INFORMATION SYSTEM**

**N79-18808**

**(Computer Sciences Corp.) 315 p HC A14/MF**

**A01**

**CSCL 05B**

**Unclas**

**G3/82**

**14552**

## Table of Contents

<u>Section 1 - General</u> .....	1-1
<u>Section 2 - System Operation</u> .....	2-1
2.1 Data Base Structure .....	2-1
2.1.1 File Definitions .....	2-2
2.1.2 Main Data File .....	2-2
2.1.3 Field Definitions .....	2-3
2.1.4 User Messages .....	2-3
2.1.5 User Dictionary .....	2-3
2.1.6 Index Files .....	2-3
2.2 Data Base Select (DBLOAD) .....	2-8
2.3 Data Base Access Language (DABAL) .....	2-9
2.3.1 Instruction Format .....	2-9
2.3.2 Arithmetic Statements .....	2-11
2.3.3 Set Definitions .....	2-13
2.3.4 Subroutine Calls .....	2-14
2.3.5 FROM ... THRU ... Operator .....	2-15
2.3.6 IF ... THEN ... ELSE ... Statement .....	2-15
2.4 Field Definition Setup (FIELDDEFIN) .....	2-16
2.5 Report Generation (OUTPUT) .....	2-19
2.6 Summation Operator (SUMSET) .....	2-20
2.7 Message Management (MESMAN) .....	2-20
2.8 Data Base Update (CREATE) .....	2-22
2.9 Saved Commands .....	2-29
2.9.1 Saved Dialogue (SAVETEXT) .....	2-29
2.9.2 Saved Argument Lists (NAMELIST) .....	2-30
2.9.3 Saved Formats (FORMAT) .....	2-30
2.9.4 Saved Report Headers (HEADER) .....	2-32
2.10 Command Sequence (INSERT) .....	2-32
2.11 Index Generation (INDEX) .....	2-33
2.12 Special Outputs (WRITE) .....	2-35
2.13 LANSAT Data Access (POINT, AREA) .....	2-35
2.14 Set Listing Device (PRINTER, TERMINAL) .....	2-36

Table of Contents (Cont'd)

<u>Section 3 – System Description</u>	3-1
3.1 General System Operation	3-2
3.2 Set Formation	3-3
3.3 Lexical Analyzer	3-4
3.4 Parser	3-6
3.5 Instruction Synthesizer	3-8
3.6 Execution Stack Interpreter	3-10
3.7 Set Summation	3-11
3.8 Report Generation	3-12
3.9 Labeled Common	3-15
3.10 MINIS System Installation	3-35
<u>Appendix A – Examples on Three Data Bases</u>	A-1
<u>Appendix B – Program Specifications</u>	B-1
<u>Appendix C – Program Flow Charts</u>	C-1

## SECTION 1 - GENERAL

The Multipurpose Interactive NASA Information System (MINIS) is a multipurpose data management system for small to medium size computers. The system is designed to accommodate fixed length record data bases with up to 200 fields (an arbitrary limit) and as many records as available mass storage will permit. The system has been operated on a data base with over 185 fields and on another data base with over 160,000 records.

The MINI System is an interactive modular system. Each major function is invoked by user requests in the system language, DABAL. The Data Base Access Language, DABAL, provides the capabilities to form sets, perform mathematical calculations, define new variables from combinations of data base fields and other variables, sum a field or variable within a set, and to invoke any of the modules of the MINIS.

The modular structure of MINIS is an efficient means of providing the existing features and functions of the system, and it also establishes a base to which additional features may readily be attached. The features currently included in the MINI System include:

- User defined data structure
- User constructed output reports
- Inverted index files
- Data base update capability
- Simple equation oriented query language
- Mathematical manipulation of fields and variables
- Saved inquiry capability
- Saved report format capability
- Data code conversions and scaling

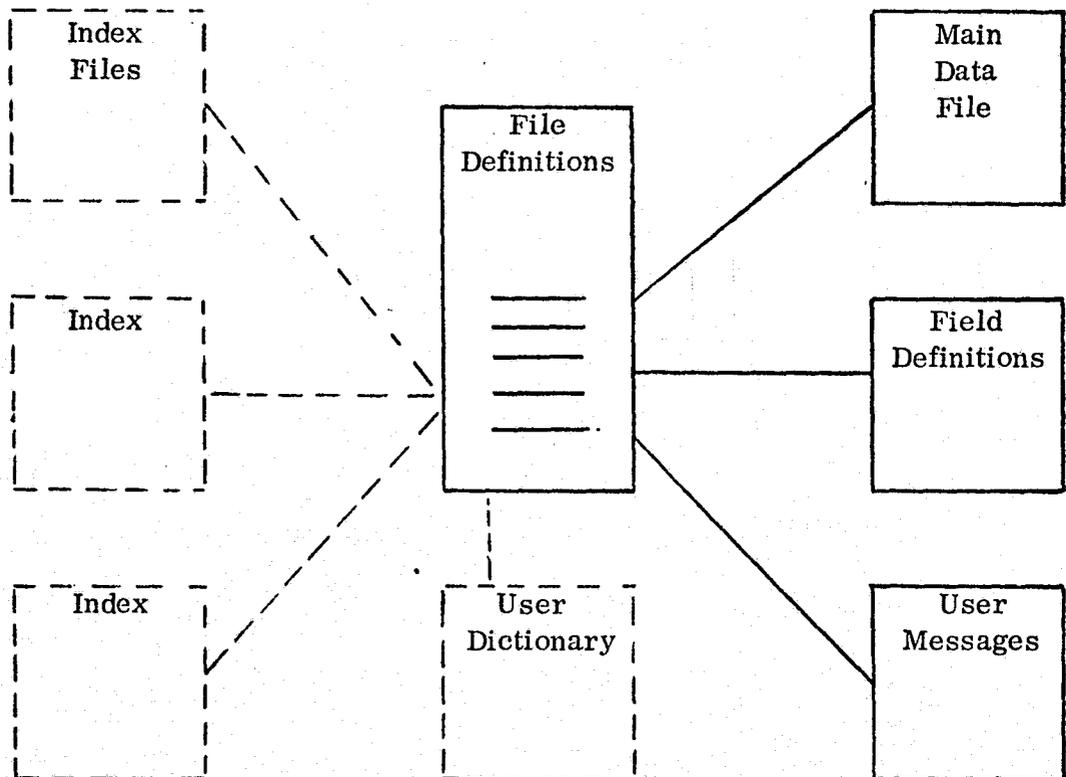
SECTION 2 - SYSTEM OPERATION

The MINI System performs data base searches and set formation through user commands entered via the DATA BASE ACCESS LANGUAGE, DABAL. Adjunct functions, such as data base definition and updating, are performed through the operation of separate modules containing the required dialog for user control. This section describes the operation of each module and the use of DABAL.

2.1 DATA BASE STRUCTURE

For each data base established on the MINI System, there are several files which must be defined and several optional files which can facilitate data base searches. The required structure of these files is defined in this section to provide supporting information to the operation instructions in the following sections.

The block diagram below shows the files required for each defined data base. The "file definitions" file contains summary information for all the other files in the data base. The "main data" file contains all of the actual data and is what the user



Files of a MINI System Data Base



Bits 36 through 47 of each record are unused to have each record end on a word boundary (a multiple of 16).

### 2.1.3 Field Definition File

The field definition file is read into an assigned core area at data base load time and is written when changes are made to the field definitions. The field definitions constitute a bit map for a data base record. The field definition file is the same as the common FIELDS. The FIELDS common description in Section 3 provides a complete explanation of the field definitions.

### 2.1.4 User Messages

The user message file contains all field titles, saved text, headers, formats, and name lists. The user message file is broken into two major segments. An index to the messages, and the messages. The index resides in the first three hundred locations of the file, and consists of byte pointers to the start of each message. Example: Location 1 would probably contain 601 on the NOVA or 901 on the Datacraft. (Word 301 of the file would start at byte 601 on the NOVA.) The messages start at word 301 (byte 601 on NOVA, byte 901 on Datacraft) of the message file. Each message consists of:

Word 1	-	Message number
Word 2	-	Number of bytes in message
Word 3	-	Message text

The messages may be in any sequence within the file as additions and deletions use and provide space in a random fashion.

Message numbers are reserved for special use in blocks. Numbers 1 to 200 are for field titles. Numbers 201 to 299 are reserved for name lists, headers, formats, and saved text. Number 300 contains the length of the message file.

### 2.1.5 User Dictionary

The user dictionary is simply a list of titles associated with saved text, name lists, etc., along with the title type and message number. The user dictionary is used as an extension of the system reserved word list for the current data base

### 2.1.6 Index Files

Index files are designed to speed up the process of set formation by providing a cross-reference between the value of a certain field and the corresponding record numbers. In order to use the index files, two levels of indexing for a

field should exist. The first level file lists the data base record numbers in order of increasing field value. The second level file provides pointers to the first level file of a range of values. Higher level files provide pointers to the previous level file of a larger range of values.

The first level index file is composed of one entry or record corresponding to each data base record. Each entry contains a record number and the field value on that record packed into the minimum number of words necessary to hold the largest pointer and its corresponding field value. The field value is stored in the entry in the same form that it is stored on the data base. The entries are arranged on the file in ascending order of the field values.

Higher level index files are composed of one word entries containing record pointers to the previous level index file. Each entry represents the minimum value of an equal range of values. The range of each entry is calculated by dividing the total field range value by the number of segments requested by the user. The first entry represents the minimum value of that field, therefore requiring one more entry than the requested number of segments. The record number in each entry indicates the record preceding the record of which the corresponding segment value is equal to or greater than the segment value of that entry.

For each index file, a file definition is stored in the common area FILDEF. The index file definitions start at location 31 and end at location 230. This allows a maximum of 19 index file definitions of 10 words each. When an index file is generated, its file definition is stored in the first unused location beyond the index file of the previous level for that field. When an index file is deleted, its file definition is blanked out and the location is then considered unused. Each index file definition is 10 words long and contains the following information: index file name, number of records in the file, number of bytes per record, field number (level 1) or segment size (level 2), and number of bits needed for the record pointer. The file definition is constructed and stored in the common area FILDEF by subroutine WRTDEF.

The index file name is composed of six characters. The first character is a meshed character created from the characters in the data base name. The next three characters are the field number and the last two are the level number and the index file type number. All level 1 index files are type 1. Higher level index files are type 2.

Index Files for Field COUNTY:

<u>Level 1</u>			<u>Level 2 (20 Segments)</u>			<u>Level 3 (10 Segments)</u>		
<u>Record Number</u>	<u>Record Pointer</u>	<u>Value of Field</u>	<u>Record Number</u>	<u>Record Pointer</u>	<u>Corres. Field Value</u>	<u>Record Number</u>	<u>Record Pointer</u>	<u>Corres. Field Value</u>
1	1	11	1	1	11	1	1	11
2	2	11	2	107	19.9	2	2	28.8
.	.	.	3	107	28.8	3	4	46.6
.	.	.	4	107	37.7	4	6	64.4
107	107	11	5	107	46.6	5	8	82.2
108	119	111	6	107	55.5	6	10	100.0
.	.	.	7	107	64.4	7	12	117.8
.	.	.	8	107	73.3	8	14	135.6
124	135	111	9	107	82.2	9	16	153.4
125	108	125	10	107	91.1	10	18	171.2
.	.	.	11	107	100.0	11	20	189.0
.	.	.	12	107	108.9			
135	118	125	13	124	117.8			
136	149	147	14	135	126.7			
.	.	.	15	135	135.6			
.	.	.	16	135	144.5			
162	175	147	17	162	153.4			
163	207	149	18	196	162.3			
.	.	.	19	240	171.2			
.	.	.	20	240	180.1			
196	240	149	21	257	189.0			
197	136	165						
.	.	.						
.	.	.						
209	148	165						
210	176	169						
.	.	.						
.	.	.						
240	206	169						
241	241	187						
.	.	.						
.	.	.						
257	257	187						
258	258	189						
.	.	.						
.	.	.						
300	300	189						

ORIGINAL PAGE IS  
OF POOR QUALITY

### Use of Index Files:

The above table is an example of a system of index files for the field county. In the level 1 file, the actual entry is composed of the record pointer and field value. In the level 2 and level 3 files, the entry is the record pointer. The level 3 record pointers refer to the level 2 record numbers. The level 2 record pointers refer to the level 1 record numbers. The level 1 record pointers refer to data base record numbers.

To use this system of index files for the following set definition: SET = COUNTY .EQ. 165, the number of index levels for COUNTY is first extracted from the field definition of COUNTY. Starting with the highest level file, the minimum field value and segment size are obtained from the file definition, and the record number of the appropriate segment is calculated, which for this example is 9. The ninth record is read from the file and the record pointer in that entry is 16. Using the segment size of the level 2 file, the sixteenth segment value is calculated and found to be less than the value calculated on level 3. The record pointer is incremented and the segment value calculated until the best value is found (18 for this example). The eighteenth record is read and its record pointer is 196. Starting at record 196 of the level 1 file and continuing until the entry value is greater than 165, a record is read and its value checked against 165. If the value is equal to 165, the record pointer is written in the candidate file and the set membership count incremented. The number of records found to have a value equal to 165 is returned to the user as the number of records in the set.

In most cases, only 2 index levels are needed. As you can see in this case, if only 2 levels had existed, the first calculation would have indicated record 18 of the level 2 file and several file accesses would have been eliminated.

The number of segments also affects the efficiency of the use of index files. A level 2 file for COUNTY of 10 segments would be as follows:

<u>Record Number</u>	<u>Record Pointer</u>	<u>Calculated Value</u>
1	1	11.0
2	107	28.8
3	107	46.6
4	107	64.4
5	107	82.2
6	107	100.0
7	124	117.8
8	135	135.6
9	162	153.4
10	240	171.2
11	257	189.0

In this case the calculated segment number would be 9. The record pointer in record 9 is to record 162 on the level 1 file. This will cause more records to be read on the level 1 file in order to find the set members. In this case, a level 2 file of 20 segments is more efficient.

## 2.2 DATA BASE SELECT OR CREATE (DBLOAD)

After establishing the computer connection and executing the MINI System (see log-on procedure for user's computer), the user must select the data base to be accessed. Upon execution, the system will request:

ENTER DATA BASE NAME (6 CHAR)

to which the user may enter an existing data base name, a new data base name to start up a new data file, or an existing data base name followed by a question mark to modify parameters of an existing data base.

If an existing data base name is specified, parameters associated with that base are restored and the system is ready for inquiries on that data base.

If the data base name is followed by a question mark, the number of bytes per data base record is displayed and the user is requested to enter any changes. Then the number of records in the data base is output and the user is permitted to change that number.

If a new data base name is entered, the system informs the user there is no such data base and that a new data base may be created. The name of the new data file is requested and accepted and the system is readied for normal operation. At this point in establishing a new data base, the user must enter the field definition module (See FIELDDEFIN) to define the data structure for each record.

Examples:

<u>Dialogue</u>	<u>Comments</u>
1. Load an existing data base.	
C: ENTER DATA BASE NAME (6 CHAR)	(C: for computer response U: for user input)
U: LANSAT?	Name of an existing data base
C: >	Prompt for user inquiries
2. Change record count of existing data base.	
C: ENTER DATA BASE NAME (6 CHAR)	
U: LANSAT?	Change existing data base
C: NO. OF BYTES PER DATA BASE RECORD = 36 ENTER THE NUMBER OF BYTES IN THE DATA BASE RECORD (BLANK IF NO CHANGE):	(for example)

## Dialogue

## Comments

U: 30 (for example)

C: NO. OF DATA BASE RECORDS = 30  
117314

### 3. Establish a new data base.

C: ENTER DATA BASE NAME (6 CHAR)  
ENTER NO. OF RECORDS ON DATA  
BASE FILE (BLANK IF NO CHANGE):

U: 118419 (for example)

C: (ready for input)

## 2.3 DATA BASE ACCESS LANGUAGE

The DATA Base Access Language (DABAL) provides data base access and calculated quantity capabilities through a sequence of user input statements and equations. DABAL instructions are executed in an immediate mode providing the user with an immediate response to his inquiries.

### 2.3.1 Instruction Format

DABAL instructions are free of any rigid structure. They may appear in columns 1 through 72 on a line with the characters past column 72 being ignored. Leading, imbedded and trailing blanks may appear in an instruction to improve readability.

DABAL's scanner ignores leading blanks and stores the instruction beginning with the first non-blank character. A continuation character (:) may appear at the end of any line so that the instruction may be continued on the next line. A total of 432 characters may appear in a single instruction. Only the characters from the first non-blank to the last character (blank or non-blank) before the continuation character on each line are counted in this total. Any characters past number 432 are ignored.

**Character Set:** The set of characters accepted by DABAL consists of alphabetic (A to Z), numeric (0 to 9), and special characters (=+\*/().,:'% and Blank).

**Constants:** All constants utilized in DABAL can be classified as integer, real, literal or logical.

Integer constants may appear as a string of digits which may be preceded by a + or - sign. The range of integer values is determined by the machine on which the MINI System is implemented. On the Datacraft 6024, integers may range from  $(-2^{23})$  to  $(2^{23}-1)$ . On the NOVA computer, integers range from  $(-2^{15})$  to  $(2^{15}-1)$ .

Real constants appear as strings of digits with a decimal point and/or an exponent. Again, the range of real values is determined by the computer that is used. The following forms may be used to represent real constants:

$\pm 123.4567$     $\pm 12345.$     $\pm .1234$     $\pm 12.34E \pm 15$     $1.E \pm 7$   
 $\pm .123E \pm 45$     $\pm 123E \pm 45$

where plus or minus signs are optional.

Literal constants appear in the form

's'

where s is a string of 0 to 11 characters out of the DABAL character set with the exception of ' and :. If there are less than 11 characters the string is left justified and blank filled. A null string (') results in 11 blanks. All characters past the 11th are ignored, and the trailing ' is required.

Examples:

<u>String</u>	<u>Resulting Form</u>
'A'	A $\emptyset$
'JUST ENOUGH'	J U S T $\emptyset$ E N O U G H
'TOO MANY HERE'	T O O $\emptyset$ M A N Y $\emptyset$ H E
'NO TRAILER	error
'ILLEGAL ' TIC'	error
'CONTINUE:'	error; the : is always taken as a continuation character, and the trailing ' is ignored.

Logical constants may appear in only two forms:

.TRUE. and .FALSE.

The internal values for these constants are -1 and 0 respectively.

Identifiers: Identifiers are alphanumeric strings that name variables, fields, sets and subroutines. Identifiers may be from one to eight characters long, and the first character in an identifier must be a letter. There are certain character strings that are reserved for use in DABAL instructions and must not appear as identifiers. These are:

FROM, THRU, IF, THEN, ELSE, and

Subroutine names (OUTPUT, FORMAT, HEADER, etc.)

Any nonreserved name may be used as a variable name, set name or field name. When an identifier is defined as a field name, it may not be used for any other purpose until deleted. Any symbol used as a variable must not subsequently appear as a set name during the session, nor should a set name later appear as a variable.

Variable names identify quantities that result from execution of an arithmetic statement or from assignment to a constant value. No data type is associated with a variable name, so any name may identify any integer, real, literal or logical quantity.

Field names identify a previously defined portion of a data base record. These names may appear in arithmetic expressions but should never be assigned new values (i. e., they should never appear on the left-hand side of an =.) The only way new fields may be defined is by a call to a special MINIS subroutine.

Set names identify a collection of data base records that meet all of the requirements specified in a set definition statement.

Subroutine names identify previously defined subroutines in the MINIS. These names may not appear in the normal DABAL arithmetic expressions and set definition statements.

Subroutine names presently recognized by the system are:

FIELDFIN	AREA
MESSAGE	POINT
RESET	WRITE
FORMAT	OUTPUT
HEADER	PRINTER
NAMELIST	TERMINAL
SAVETEXT	SUMSET
INDEX	CREATE

### 2.3.2 Arithmetic Statements

Arithmetic statements are used to assign real or integer values to variable names. They are of the form:

$$v = e$$

where v is any legal variable name and e is an arithmetic expression.

An arithmetic expression is a series of constants, variables and/or field names connected by arithmetic operators. Arithmetic expressions may be used to assign a value to a variable, as above, or may become a part of set definition statement as described in Section 2.3.3.

The arithmetic operators follow the normal order of precedence (exponentiation before multiplication or division before addition or subtraction), except when grouped with parenthesis. Parenthetic expressions are always evaluated beginning with the innermost set of parenthesis. At any level of evaluation, operations of the same order of precedence are evaluated from right to left. Thus  $A + B + C$  is equivalent to  $A + (B + C)$ .

The arithmetic operators are listed below along with their meaning and numbers indicating their order of precedence.

**	Exponentiation	1
*	Multiplication	2
/	Division	2
+	Addition	3
-	Subtraction	3

Example:

The arithmetic statement  $A = B * (C + D * (F / (G + H) - I) + K)$  is evaluated in the following sequence:

$$\begin{aligned}
 t_1 &= G + H \\
 t_2 &= F / t_1 \\
 t_3 &= t_2 - I \\
 t_4 &= D * t_3 \\
 t_5 &= C + t_4 + K \\
 t_6 &= B * t_5 \\
 A &= t_6
 \end{aligned}$$

where the  $t_1$  indicates the sequence of evaluation.

Some example arithmetic statements calculating income data for a cell in a census data base are:

$$\begin{aligned}
 INCOM &= INLT5K * 2500 + IN5T10 * 7500 + \\
 &IN1020 * 15000 + IN2030 * 25000
 \end{aligned}$$

PEOPLE = INLT5K + IN5T10 + IN1020 + IN2030

AVGINC = INCOM/PEOPLE

PCTLT10 = (INLT5K + IN5T10)/PEOPLE

### 2.3.3 Set Definition Statements

Set definition statements are used to define constraints that are to be applied to data base records to form a set of member records. Set definition statements are in the form:

$$S = R_1 \ 1_1 \ R_2 \ 1_2 \ R_3 \ \dots$$

where S is any legal set name, the 1's are logical operators (.AND., .OR., .XOR., .ANDNOT., .ORNOT., .NAND., or .NOR.) and the R's are relations defined as follows:

$$R = \{ \text{.NOT.} \} e_1 \ r \ e_2 \quad \{ \} \text{ denotes optional}$$

The r is a relational operator (.EQ., .GE., .GT., .LE., .LT., or .NE.) and the e's are arithmetic expressions like those described for arithmetic statements. .NOT. is an optional unary operator which applies to the result of  $e_1 \ r \ e_2$ .  $R = \text{.NOT.} \ e_1 \ r \ e_2$  means that R is true only when  $e_1 \ r \ e_2$  is false.

Set definition statements are evaluated in the following sequence:

1. Arithmetic expressions
2. Relational expressions
3. .NOT. operations
4. Other logical operations

Again, at any level of evaluation, operations of the same order of precedence are evaluated from right to left. Parentheses nested to any depth may be used to change the order of evaluation.

Examples:

1. SET 1 = HOUSES .GT. 2 \* (5 + 3)

forms the set of records from the data base for which the field HOUSES contains a number greater than 16.

2. SET 2 = HOUSES .GT. 16 .AND. PEOPLE. LT. 30  
forms the set of records for which HOUSES is greater than 16 and PEOPLE is less than 30.
3. SET 3 = .NOT. HOUSES. LE. 16 .AND. PEOPLE. LT. 30  
does the same as Example 2.
4. SET 4 = .NOT. (HOUSES. GT. 16 .AND. PEOPLE. LT. 30)  
forms the complement of SET 2.
5. SET 5 = (.NOT. ( (HOUSES. GT. 16) .AND. (PEOPLE. LT. 30)))  
is equivalent to Example 4.
6. SET 6 = .NOT. A .GE. 10 .OR. B .LT. 5 .ANDNOT. .NOT.  
.C.EQ. 15 .AND. D .NE. 20

is evaluated in the following sequence:

$t_1 = D.NE.20$ $t_2 = C.EQ.15$ $t_3 = .NOT.t_2$ $t_4 = t_3 .AND. t_1$ $t_5 = B.LT.5$	$t_6 = t_5 .ANDNOT. t_4$ $t_7 = A.GE.10$ $t_8 = .NOT. t_7$ $t_9 = t_8 .OR. t_6$ <b>SET 6 = <math>t_9</math></b>
---	---

where the  $t_i$  indicates the levels of evaluation.

### 2.3.4 Subroutine Calls

Subroutine calls can take any of three forms.

SUBNAME ( $a_1, a_2, \dots$ ),  
 $i =$  SUBNAME ( $a_1, a_2, \dots$ ), or  
 SUBNAME

where SUBNAME is any defined subroutine name, the  $a_i$  are arguments, and  $i$  is any legal identifier for a set or variable. Identifier types to the left of the = and in the argument list are defined by the particular subroutine being called.

Examples:

OUTPUT (SET1, ALL)

outputs all fields of each record in the set SET1.

TOTMASS = SUMSET (SETX, MASS)

sums the MASS of each item in set SETX and assigns the total to TOTMASS.

FIELDFIN  
MESSAGE

these two calls invoke separate modules which obtain arguments through additional user dialog.

### 2.3.5 FROM THRU Operator

A shorthand method of defining a range of values exists for set definition statements using the reserved words FROM and THRU. A FROM. . . THRU. . . operation may appear by itself on the right side of the = or it may be used in conjunction with other logical operations.

Examples:

1. SET 1 = HOUSES FROM 10 THRU 20  
is equivalent to the instruction  
SET 1 = HOUSES.GE.10.AND.HOUSES.LE.20
2. SET 2 = HOUSES FROM 10 THRU 20 .AND. PEOPLE.LT.30
3. SET 3 = HOUSES.EQ.10.OR. PEOPLE FROM 20 THRU 30

### 2.3.6 IF Statement

IF statements provide for the logical test of a relational expression to control the execution of an arithmetic statement. The IF statement takes the form:

IF R THEN  $a_1$  ELSE  $a_2$

where R is a relation defined just as those used in the set definition statements. The a's are arithmetic statements like those described earlier.

The "IF" and "THEN" clauses are mandatory while the "ELSE" clause is optional. If the relation R is true, then  $a_1$  is executed. Otherwise, when the "ELSE" clause is present,  $a_2$  is executed.

**Examples:**

1. IF HOUSES.GT.2 THEN X = 1

means that for a data base record the field HOUSES must be greater than 2 in order for X to be assigned a 1. Otherwise, no action is taken.

2. IF HOUSES.GT.2 THEN X = 1 ELSE X = -1

accomplishes the same thing as Example 1 as well as setting X to -1 when HOUSES has a value that is less than or equal to 2.

**2.4 FIELD DEFINITIONS (FIELDFIN)**

The field definitions of a data base may be initialized, modified or displayed through the use of the FIELDFIN module. The field definition module is invoked by the subroutine call:

**FIELDFIN**

The field definition module responds with an acknowledgement: FIELD DEFINITION SETUP. and begins the dialog with: ENTER OPERATION TYPE. (ADD, DELETE, EXIT, LIST, MODIFY, OR REPLACE).

ADD, MODIFY, and REPLACE perform essentially the same function of establishing a field definition. The dialog which follows is the same for all three functions except that blank line inputs are treated as zero for ADD and REPLACE, and as no change for MODIFY. The following dialog example for the ADD operation is applicable for all three.

U: ADD  
C: NEXT AVAILABLE FIELD DEFINITION  
NUMBER IS: xx  
C: OK? (Y OR N)  
U: Y (if field no. xx is acceptable)  
C: ENTER NEW FIELD LABEL.  
U: APPLES (up to 8 character field name)  
C: FIELD ID CONFLICT (if APPLES is already used), or  
C: INPUT FIELD TITLE IF DIFFERENT. (enter a title for printouts)  
U: NO. OF APPLES IN BUSHELS. (example)  
C: ENTER DATA TYPE. (0 to 15 see table in COMMON)

U:	0	(0 is integer type)
C:	ENTER CONVERSION CODE.	(0 to 15 see table)
U:	0	(0 or blank for no conversion)
C:	ENTER FIELD SIZE IN BITS (BYTES IF TEXT):	(see structure)
U:	10	(10 bits to permit 1024 maximum integer)
C:	ENTER FIRST BIT POSITION OF FIELD IN RECORD.	(see structure)
U:	165	(for example) At this point if a split field (see data structure) was specified, the user is requested to input the subfield size and the number of repeats.
C:	ENTER FIELD NO. OF KEY, IF ANY.	(Enter number if some other field must assume a certain value for this field to exist)
U:	5	(for example)
C:	ENTER VALUE OF KEY FOR PRESENCE OF THIS FIELD.	(1-15)
U:	9	(required Key value)
C:	ENTER NORMAL OUTPUT FORMAT FIELD WIDTH OR REPEAT COUNT.	(see FORMAT for format specifications)
U:	1	(1 repeat)
C:	ENTER FORMAT TYPE.	(A-alphanumeric, I-integer, etc.)
U:	I	(Integer format)
C:	ENTER SIGNIFICANT PLACES OR CHARACTERS PER WORD.	
U:	4	(4 digit numbers)
C:	IS THIS FIELD SEQUENTIAL ON THE DATA BASE?	
U:	N	(Y if variable is monotonically increasing as the record number goes from 1 to the maximum)
C:	FIELD ID OPERATION COMPLETE.	

The MODIFY operation is similar to the ADD operation with a few exceptions. When MODIFY is specified, the user is prompted:

C: ENTER FIELD ID TO BE MODIFIED.

U: APPLES (enter a previously defined first name)

C: ENTER NEW FIELD NAME.

U: ORANGES (if no change enter a blank line)

C: ENTER FIELD TITLE IF DIFFERENT.

U:

The balance of the dialog for MODIFY is the same as in the ADD operation. A blank line response to any request is interpreted as no change to the affected parameter.

If the REPLACE operation is specified, the following typical sequence occurs:

C: ENTER FIELD NAME TO BE REPLACED.

U: APPLES (field to be deleted)

C: FIELD ID DELETED.

C: ENTER NEW FIELD LABEL.

U: ORANGES

The remainder of the REPLACE dialog is the same as for the ADD operation.

The DELETE operation removes the specified field definition from the definition table and packs the remaining definitions to remove any blanks. The field numbers associated with all succeeding fields are adjusted to reflect the change. The DELETE dialog is as follows:

C: ENTER FIELD ID TO BE DELETED.

U: APPLES

The LIST command provides a listing of any or all field definitions in a horizontal or vertical format. The horizontal format lists each parameter of the field on a separate line. The vertical format lists the parameters in columns. An example of the vertical format is included in Appendix A. The dialog for the LIST command is as follows:

C: HORIZONTAL OR VERTICAL LIST (H OR V):

U: V (for columnar list, H for long list)

C: ENTER FIELD ID TO BE LISTED. (ALL TO LIST ALL IDS)  
 U: ALL (table will be output as in Appendix A)

The EXIT command is the only correct method of returning to the MINIS. If the system escape and restart is used, all changes input during the current session are lost. The EXIT dialog is as follows:

C: SAVE FIELD DEFINITION CHANGES? (Y OR N)  
 U: N (will exit with no changes) or  
 U: Y  
 C: ADJUST FIRST BIT POSITION FOR SUCCEEDING FIELDS (Y OR N)?  
 C: (DO NOT ADJUST UNLESS YOUR DATA BASE HAS NO OVERLAPPING FIELDS AND FIELD NUMBERS CORRESPOND TO THE ORDER OF THE FIELDS ON THE DATA BASE RECORD.)  
 U: Y (fields will be concatenated with starting bit numbers corrected)  
 or  
 U: N (always use N if there are any "phantom" fields.)  
 C: ENTER THE NUMBER OF BYTES IN THE DATA BASE RECORD (BLANK IF NO CHANGE):  
 U: 36 (For example. Round up to the next word boundary.)

## 2.5 REPORT GENERATION (OUTPUT)

The OUTPUT permits the listing of all data base records of a set in user specified format or the default format specified in the field definitions. Any or all fields as well as any created variables may be output for each record. The user may also specify the use of special headers and previously defined lists of variables or fields to be output.

To invoke the report generator, execute a call of the form:

```
OUTPUT (SETNAM, { HEADR, } { FORMT, } { ALL } .VAR1, VAR2,
. . . VARN)
           { } denotes optional
```

where SETNAM is the name of the set to be output, HEADR and FORMT are the names of predefined header and format specifications, and VAR1, VAR2, etc. are any field or created variable names.

The only required arguments are the name of a defined set and one or more variable names. All existing fields may be output by inserting ALL as the variable name. A predefined list of variables may be output by inserting the name of the list in the OUTPUT argument string.

When an output format is specified, it must be the correct format for the variable list. (See FORMAT Section 2.9.3.) For unspecified formats the available output columns are divided by the number of output variables. If the result is greater than eight spaces for each variable, the field title is used for each column heading. Otherwise the field names are used for headers. The field names are printed horizontally if 6 to 8 columns are available. Otherwise, they are printed vertically. Special user specified headers may be used by including the name of the header in the argument string. (See HEADER Section 2.9.4.)

Several examples of the use of the OUTPUT statement may be found in Appendix A.

## 2.6 SUMMATION OPERATOR (SUMSET)

SUMSET provides the capability of summing the values of a field or created variable for each record of a set. The operation is performed immediately and the result is printed when the calculations are complete. The SUMSET operator is used in equation form. For example:

```
TOTAL = SUMSET (SET1, APPLES)
```

The value, or quantity, of APPLES is summed for each record in the set, SET1. That total is then assigned to the new created symbol, TOTAL.

## 2.7 MESSAGE MANAGEMENT (MESMAN)

The message manager provides a means of adding, deleting, replacing or listing user messages. System messages may also be listed but not altered. The message manager is primarily used by other MINIS routines for message handling (HEADER, FORMAT, etc.), but is made available to the user for direct message manipulation. To call MESMAN from the system, the user must enter: MESSAGE. The following dialog contains examples of each operation:

```
C: MESSAGE MANAGER. INPUT OPERATION
   CODE.
U: ? (if request not understood)
C: ILLEGAL OPERATION. ENTER FIRST
   LETTER OF ONE OF THE FOLLOWING
   OPERATIONS: ADD, REPLACE, DELETE,
   EXIT, PACK, LIST, OR CHANGE.
```

U: ADD (for example)

C: ENTER MESSAGE NUMBER, IF KNOWN.

U: (CR)

C: ENTER MESSAGE TEXT. INDENT WITH LEADING SPACES. TERMINATE WITH A BLANK LINE.

U: THIS IS A SAMPLE MESSAGE. (for example)

U: (CR)

C: MESSAGE MANAGER. INPUT OPERATION CODE.

U: REPLACE

C: ENTER MESSAGE NUMBER IF KNOWN.

U: 97 (for example)

C: ENTER MESSAGE TEXT. . . . etc.

U: (new message followed by blank line)

C: MESSAGE MANAGER. INPUT OPERATION CODE.

U: DELETE

C: ENTER MESSAGE NUMBER, IF KNOWN.

U: 97 (for example)

C: MESSAGE NO. 97 DELETED. (if 97 was legal) or

C: INCORRECT MESSAGE NUMBER. (if 97 did not exist)

C: MESSAGE MANAGER. INPUT OPERATION CODE.

U: EXIT (to quit)

C: > (system prompt) or

U: PACK (to pack user message file)

C: MESSAGE FILE PACK IS COMPLETE.

C: MESSAGE MANAGER. INPUT OPERATION CODE.

U: LIST (to list messages)

C: ENTER MESSAGE NUMBER, IF KNOWN. (enter first message number)

U: 1 (for first user message. 1001 for first system message.)

C: ENTER LAST MESSAGE TO LIST.

U: 5 (list first five user messages)

C: 1 UTM ZONE NUMBER. (first five messages of land use data base)

2 UTM CELL DESIGNATOR.

3 NUMBER OF OCCUPIED HOUSEHOLDS.

4 PEOPLE PER HOUSEHOLD.

5 COUNTY CODE.

C: MESSAGE MANAGER. INPUT OPERATION CODE.

U: CHANGE (change message numbers)

C: ENTER MESSAGE NUMBER, IF KNOWN.

U: 97 (example)

C: ENTER NEW MESSAGE NUMBER.

U: 95

C: CHANGE MESSAGE 97 TO 95 OK? (Y OR N)

U: YES

C: MESSAGE MANAGER. INPUT OPERATION CODE., etc.

## 2.8 DATA BASE UPDATE (CREATE)

CREATE enables the user to create or initialize new data base systems, update pre-existing systems, or modify pre-existing systems. CREATE allows the user to create data base records from a raw data source, which can be on cards, on magnetic tape, or on disc. Any field defined for the data base record can be created from one or more elements (bytes, words, etc.) of the raw data. One data base record can be created from any number of raw data physical records. The data base records created using CREATE can be sorted by 1 to 4 fields and/or merged by 1 to 4 fields (merging will save only 1 record for each value of the requested fields; therefore eliminating duplicate records). An example of the use of CREATE to perform an update of the LANSAT data base is in Appendix A.

In order to use CREATE, the following MINIS files must be accessible:

1. The field definition file -AFELDS
2. The system message file -MESSYS
3. The user message file -AUMES
4. The file definition file -(data base access name)

Other necessary files or data are as follows:

1. Raw data (tape, cards, or disc)
2. The operation card file or deck (on cards or disc - OPCARD)
3. The raw data information file or deck (on cards or disc - INPUTD)
4. The input information record (on disc or entered from the terminal-  
INPUTR)

The input information record contains information that defines the input modes for the other files and the raw data. It also defines the byte length of a physical and logical record along with the bit length and configuration of any key values. The format for this record is as follows:

- Column 1 - I, U, or M (Initialization, update, or modify options)  
2 - L if the raw data tape is begun with a label record, otherwise blank  
8 - Input mode for operation deck and raw data information deck (D for disc, C for cards)  
9 - Raw data input device C, T, or D (cards, tape, or disc)  
10-18 - Raw data file name if any  
19-24 - Number of bytes per raw data physical record  
25-30 - Number of bytes per raw data logical record  
31-32 - Number of bits in key value (if any), maximum 32  
33-38 - Number of raw data records (if blank, processing will continue until an end-of-file is found)  
40-71 - Bit configuration of key value

This record is the first requested information after entering the routine CREATE. The user is asked to choose either default or manual mode. If default is requested, the input information record (INPUTR) is read from disc. Manual mode will request the record to be entered from the terminal. If the number of raw data records was left blank, this value will be then requested from the terminal.

The raw data information file (INPUTD) defines the elements on a raw data logical record that will be needed to create a data base record. The format of each record (describing the corresponding element) is as follows:

- Column 1-2 - Number of bits in the element (bytes if text)
- 3-8 - First bit number of element (byte if text)  
bit origin is 0 - byte origin is 1
- 9-10 - conversion code for the element
  - =0 No conversion
  - =1 ASCII to integer
  - =2 BCD to integer
  - =3 EBCDIC to integer
  - =4 ASCII to BCD
  - =5 Character to logical (T or Y=1, other=0)
- 11-12 - data type code
  - =0 integer or binary
  - =1 ASCII
  - =2 BCD
  - =3 EBCDIC
  - =4 Floating point (NOVA - 16 bit words)
  - =5 Floating point (Data craft - 24 bit words)
  - =6 Floating point (Sigma 5 - 32 bit words)
  - =7 Floating point (IBM 360 - 32 bit words)
  - =8 ASCII Text
  - =9 BCD Text
  - =10 EBCDIC Text

Text is defined as any character data of greater than 2 times the number of bytes per word of the host machine (NOVA =4, Datacraft =6).

The operation card file or deck (OPCARD) contains the necessary operations to create the data base fields from the elements on each raw data record. One or more operations may be requested to define a field from the raw data. Each operation card will define one operation. The format of an operation card is as follows:

result: operand - operation - operand

The operands must be a field, intermediate value (value that is defined by the operation cards), element, or constant. Fields are designated by their field names or by their field number preceded by an F (ex: F10).

Intermediate values are designated by I1, I2, I3... Etc. Elements are designated by E1, E2, E3... Etc. Constants can be integer, real, or characters (Characters are surrounded by apostrophes. Only 2 times the number of bytes per word will be recognized within apostrophes, others will be ignored. Ex: on the NOVA 'CATTE' will cause CATT to be recognized; on the DATACRAFT 'CATTE' will cause all 5 characters to be recognized.) Results must be a field number or name, or an intermediate value. A field number or name in a result field will indicate the final operation to define a field and that field value that is computed will be stored on the data base record. There are 13 possible operations as follows with examples of the use of each:

OPERATION		EXAMPLE
Assignment	=	I1= 5, F100 = E5 + 2.0
Addition	+	F1 = I1 + E3 or CLDCOV= +E2
Subtraction	-	I1= F2 - E4 or CLDCOV= -E2
Multiplication	*	I1=E1 * 5
Division	/	I1=E1 / 5
Exponentiation	**	I1=E1 ** I2 or I1= E1 ** 2
Square Root	.SQ.	I1= .SQ. I2 or I1 = I2 .SQ.
Sum of N values	.S+.	F10= I3 .S+. 5 ( The second operation must be an integer constant. This operation will cause I3, I4, I5, I6, I7 to be added and stored on the data base in the F10 location.)

Sum of the products of corresponding elements of Z arrays .S2. I1 = E1 .S2+ E4 \* 5

(This will cause the 1st element of E1 and E4 to be multiplied and added to the product of the other 4 corresponding elements. The 3rd operand must by an integer constant.)

Percent .% F2 = E1. %. E4 (F2 =  $\frac{E1}{E4} * 100$  round to nearest tenth of a percent.)

Character transfer .CH. F3 = E1 .CH. 5 (Five characters will be transferred to F3 on the data base record. On the NOVA the maximum number of characters in an element is 4, so the 5th character will be transferred from E2.)

Corresponding value table .CV. IZ= 2 .CV. E1  
 Search Table .SH. F10 = E5 .SH. I1 \* 5

The .CV. and .SH. operations essentially set up if statements. To use .SH. their must be at least one .CV. operation. An example of their use is as follows:

I1 = 0  
I2 = 2 .CV. E1  
I3 = 3 .CV. E2  
I4 = 4 .CV. E3  
I5 = 5 .CV. E4  
FZ=E5 .SH. I1 \* 5

This will cause a compare of E5 to the values in the table with 5 entries beginning at I1 and store the corresponding value of a true compare in F2. If E5 does not equal any value on the table, set F2 to the default value (1st value in table), in this case zero. For this case if E5 is equal to 2, F2 is set to E1; if E5 is equal to 3, F2 is set to E2; and so on. If E5 does not equal 2,3,4, or 5, F2 will be set to zero. The second operand must be an intermediate value and the 3rd operand must be an integer constant.

The maximum number of intermediate values and constants defined in the operation file is 50 each.

The raw data can be input from cards, tape, or disc. The maximum physical record sizes are as follows:

cards - 1 card (80 bytes or columns)  
tape - 1200 words  
disc - 1200 words

Logical records can be no greater than 200 words in length. A logical record can be made from several physical records or several logical records can be made from one physical record. Also one physical record can equal one logical record. The logical record can be defined as successive groups of bytes on the physical record (Such as every 40 bytes of a 400 byte physical record defines a logical record) or defined such that each logical record begins when a key value is located and continue for a maximum number of bytes or until the next key value is located (a key value can be a maximum of 32 bits and must begin at the beginning of a byte).

The data base records that are created using CREATE can be sorted and/or merged on up to four field values. The fields for the sort are requested after the work or scratch area device has been entered (tape or disc). The fields are entered one on each line. Blank return will indicate no sort or the last field has been entered. The sorts are done in the order entered if requested, the merge fields are then entered in the same manner as the sort fields. A merge will retain only one unique record with the values of the requested fields and delete others. In an update, only the update records are sorted. If a merge is requested, these records will then be merged with the entire data base. If a merge is not requested, the update records will be added to the end of the data base file.

Before returning to the MINIS, any indexes that might exist are updated to include any new records or modified records.

The following is a simple outline of the execution of CREATE: (All user and computer commands are on the left.)

U: CREATE

C: DEFAULT OR MANUAL OPERATION?

Default reads the input mode information record from disc (INPUTR). Manual requests the record to be entered from the terminal.

C: NUMBER OF RAW DATA RECORDS?

U: 200

The raw data information deck and operation cards deck are read in from card reader or disc. The raw data is read and as each logical record is located the operation cards are executed and the values stored on the data base record. This continues until all physical raw data records have been processed. (200 as requested or an end of file reached.)

C: ENTER THE WORK AREA DEVICE (TAPE OR DISC)?

U: T

C: ENTER THE DEVICE NUMBER:

U: 8

C: ENTER THE FIELDS TO SORT ON (BLANK IF NO SORT). ENTER THE LEAST SIGNIFICANT FIELD FIRST (and so on. Max of 4 fields.)

U: HHMMS

U: PHNUM

U: SAT

**U:** (Blank line)

The created data base records are sorted first on DSL, then HHMMS.

**C:** ENTER THE FIELDS TO MERGE (BLANK IF NO MERGE)

**U:** SAT

**U:** DSL

**U:** HHMMS

**U:** (Blank line)

The created data base records are merged (with the existing records if update) on the field values of SAT, DSL, and HHMMS

The indexes are now updated if necessary and control is returned to MINIS.

## 2.9 SAVED COMMANDS

The MINIS is capable of saving formats, namelists, and entire inquiry sessions for subsequent recall at any point in an inquiry session.

Each saved header, format, etc., is assigned to a user message number and stored in the user message file. The names associated with each saved sequence are saved in a separate file along with the type and number of each message.

The actual input and manipulation of the saved sequences are accomplished via calls to the message manager. Each type of saved information and its correct usage is summarized in the following sections.

### 2.9.1 Saved Text (SAVETEXT)

Any legal statements in the MINIS language, DABAL, may be saved for future recall and execution. A typical use for saved text might be a data base where the same criteria are used to reduce a candidate set size or the same calculations and output statements are repeated at each session. The repetitive portions would be saved for subsequent one statement recall. The following is an example use of SAVETEXT.

```
U:  SAVETEXT
C:  ENTER OPERATION TYPE. (ADD, DELETE, EXIT, LIST,
    MODIFY, OR REPLACE)
U:  A
C:  ENTER SAVETEXT TITLE.
U:  TEXT1
C:  ENTER MESSAGE TEXT. INDENT WITH LEADING SPACES.
    TERMINATE WITH A BLANK LINE.
U:  SET 1 = DSL FROM 20 THRU 50 (EXAMPLE)
    SETOUT = SET1 .AND. CC .LT. 30
    INSERT (LIST)
    (blank line)
C:  ENTER OPERATION TYPE ETC.
```

The DELETE, LIST, MODIFY, and REPLACE commands perform the specified operation in the same manner as the message manager except that messages are referred to by their associated titles instead of numbers.

### 2.9.2 Namelist (NAMELIST)

The NAMELIST provides a shorthand method of referencing frequently used argument strings in output calls. Arguments included in a namelist should appear in the exact sequence that they are to appear in the output statement. All spelling must be correct. Each referenced field or variable must exist. And each field must be separated by commas as in an output call.

The method of creating and modifying namelists is the same as for the SAVETEXT operation. When the assigned name of a namelist is referenced in an output statement, the associated message is substituted for the namelist name.

### 2.9.3 Format Specifications (FORMAT)

The allowable output format specifications of the MINIS are basically those of the FORTRAN programming language. The MINIS can accommodate four basic types of data. Those types and their associated format letter codes are:

A	Alphanumeric or Text
F,E	Real or Floating Point
I	Integer
L	Logical (TRUE/FALSE)

All MINIS output is controlled by format field specifications. The field specification defines the number of print columns, the data type, and the number of significant digits of each variable to be output. Default format specifications which are defined for each data base field by the field definition module are the same as FORTRAN specifications with a few exceptions:

- The floating point field width is entered as a repeat count.
- The field width of an alphanumeric field is the product of the repeat count and the number of characters per word.
- Leading zeros may be printed for integers by setting the number of significant digits in the repeat count.

The general forms for each format specification are nAm, nFm, nIm, and nLm. Where n and m are integer counts.

The "A" format is used for character or text data. Character data is used for up to four times the number of bytes per word of the host machine. Any larger field must be classified as text. The number of characters in the field should

appear as the number of significant digits in the character format specification. For example: A 7 character field should be output as 1A7, a 5 character field as 1A5, etc. The number of significant digits should be a multiple of the number of bytes per word for a text format specification. The product of the repeat count and the significant digits should be greater than or equal to the number of characters in a text field. For example: A 37 character text field on a NOVA (a 2 byte machine) should be output with a 19A2 format. The same field on the Datacraft (a 3 byte machine) would use a 13A3 format.

"F" and "E" formats are both set up in the same way. The field width is entered as the repeat count, n, and the number of significant digits, m, follows the format type. Some format specifications and their sample outputs are: 5F2-12.34, 9E2 -0.67 E-11, 4F0 123., 4F1-1.3. The same format specifications appearing in a format statement would be: F5.2, E9.2, F4.0, F4.1.

Integer format specifications should have the maximum number of significant digits specified (m count), and the repeat count set to 1. For example: 1I4 1234, 1I3 123. Integers requiring less than the allocated number of digits are printed with leading spaces in the extra print columns. If leading zeros are desired, the format specification should have the number of significant digits entered as the repeat count. Examples: 5I1 will output the number 345 as 00345, 6I1 with output 7 as 000007.

The logical format, "L", may be used in one way, 1L1. The output is T or F for true and false, respectively.

Special format specifications may be generated and edited by use of the FORMAT call from the MINIS. Each format statement should have a name (title) associated with it for future reference by an OUTPUT call. The format statements are entered and saved as a message in the same manner as in the SAVETEXT operation.

Any FORTRAN format specification that is legal on the host computer is permitted in a saved format statement. But the type and quantity of output variables must agree with the type and quantity specified in the OUTPUT statement lists.

Default format specifications are ignored when using format statements, except for integers with leading zero printing. To suppress previously specified leading zero printing, the default format must be modified. An example of a saved format is the LANSAT listing format. The format name is FORMER. The header format name is HEDMER. And the variable list name is LSTMER. The saved format associated with FORMER is:

(1X,I1,3I1,'-5I1,2I6,'/',I3,3(' ',I3),1X,5A1,1X,6A1,I3,I6,'/'I4,3I2)

The saved format is invoked by reference in the output statement. In the case of the LANSAT printout, the call is:

**OUTPUT(SET OUT, HEDMER, FORMER, IDSMER)**

which produces the sample MERITS printout found in Appendix A.

#### **2.9.4 Headers (HEADER)**

There are two types of headers available for MINIS output; headers generated from the field titles of the output variable list, and user generated header format statements.

The system generated headers are merely a playback of the field names and, where space is available, field titles input by the user through the field definition module. The default headers provide adequate information for nonrepetitive reports.

Headers are created in the same manner as formats and saved text. To create a header the system call is HEADER. The routine requests the header name and accepts the header format as text. The header format should appear exactly as it would in a FORTRAN format statement with no variables to be output. An example of a header format statement is the header used for LANSAT listing. HEDMER. The format associated with HEDMER is:

```
(31H OBSR/'N-ID   ORB  ROLL/POS 1-4, 10X,'LAT   LONG  CC  MFR ',  
'/POSN MODAYR')
```

A printout with this header is found in Appendix A.

#### **2.10 COMMAND SEQUENCE (INSERT)**

Whole sequences of previously saved inquiries and commands (see SAVETEXT) may be recalled for execution by use of the INSERT command. All statements within the saved text segment must be executable commands in the MINIS. A correct example of the use of this command is:

**INSERT (TEXT1)**

(see example in SAVETEXT)

which will cause the following sequence:

**SET 1 = DSL FROM 20 THRU 20**

**SET NAMED SET1 HAS 250 MEMBERS.**

**SETNAM = SET1 .AND. CC .LT. 30**

**SET NAMED SETNAM HAS 10 MEMBERS**

**INSERT (LIST)**

etc.

## 2.11 INDEX GENERATION

In order for the user to generate, update, or delete index files, the dialogue routine INDEX is called. This routine collects and verifies the necessary information from the user and passes it to the generation routines GENDEX and GENATE. The dialogue routine ensures that the following restrictions are adhered to when generating, updating, or deleting index files:

- Fields stored on the data base records in floating point form cannot be indexed.
- Fields that are stored in sequential order on the data base cannot be indexed. A binary search is used to form sets when this type of field is requested.
- To generate an index file, the requested level must not exist and the previous level must exist.
- To update an index file, the requested level must exist and all higher levels must also be updated to concur with the updated level.
- To delete an index file, the requested level must exist and all higher levels must also be deleted.

To generate a first level index file, the information needed for the file definition is located and the file definition constructed and stored. For each record on the data base, a file entry is written. This file is then sorted in ascending order on the field value using the sort-merge routines.

To generate a higher level index file, the segment size is calculated, the maximum and minimum field values determined, and the file definition is constructed and stored. By scanning the previous level index file, the record pointer is determined for each entry and written on the file.

To update any level index file, the index file is regenerated and the file definition written using the current file definition location. An index file is deleted using the routine FILDEL and its file definition is blanked out in the COMMON area FILDEF.

The following is an example of an index generation dialogue:

U: INDEX  
C: ENTER FIELD NAME  
U: APPLES (example)  
C: APPLES HAS 2 INDEX LEVELS. (example)

LEVEL	NO. OF ENTRIES
1	276
2	12

or

C: FIELD IS NOT INDEXED.  
C: ENTER ACTION (GENERATE, UPDATE, DELETE, NONE)  
U: GENERATE (or UPDATE or DELETE) a  
C: ENTER INDEX LEVEL:  
U: 1 (or 2) b  
C: ENTER WORK AREA (DISC OR TAPE): (if b = 1)  
U: TAPE (or DISC) c  
C: ENTER DEVICE NUMBER: (if c = tape)

or if b = 2

C: ENTER NUMBER OF SEGMENTS:  
U: 12

after action is completed

C: GENERATED INDEX LEVEL X FOR APPLES (if a = generate)  
NO. OF ENTRIES = XXXXXXXX

or UPDATED INDEX ...

or DELETED INDEX ....

C: EXIT OR CONTINUE:

U: EXIT (or CONTINUE)

if "CONTINUE",

C: ENTER FIELD NAME:

etc.

## 2.12 SPECIAL REPORTS (WRITE)

Some calculated values are the same for each record in a set. (Such as SUMSET results.) The WRITE instruction is included in the MINIS to permit one time listing of this type of calculated quantity or constant in an output report. The WRITE statement may be used to output as many values as desired, with or without reference to a user provided format.

Examples:

WRITE(X)

RESULT: X= 10.21

WRITE(X, Y)

RESULT: X= 10.21 Y= 20.42

WRITE(X, Y, FMATXY)

FMATXY=(1X, 'THE VALUE OF X IS ', F5.2, 4X, 'THE VALUE OF Y IS ', F5.2)

RESULT: THE VALUE OF X IS 10.21 THE VALUE OF Y IS 20.42

## 2.13 LANSAT DATA BASE ACCESS (POINT AND AREA)

The LANSAT data base is a large fixed record length data base which is compatible with the MINIS. The primary access to LANSAT data is through a two variable geographic index that the MINIS does not normally accommodate. The POINT and AREA routines perform all index searches for LANSAT data base searches. POINT and AREA can only be entered if the data base name is "LANSAT."

POINT selects all records (photo descriptions) with an area of coverage over the specified geographic coordinates. POINT accepts the defining coordinates, locates the appropriate records and names the set POINTSET. An example dialogue is:

U: POINT

C: ENTER LATITUDE

U: 33N 30 (33 Deg. 30 Min. North Latitude)

C: ENTER LONGITUDE

U: 85 (85 Deg. West Longitude)

C: SET NAMED POINTSET HAS 135 MEMBERS.

U: SETOUT = POINTSET .AND. CC .EQ. 0

C: SET NAMED SETOUT HAS 27 MEMBERS

U: INSERT (LIST)

C: A list of the photos in set SETOUT would appear here. See sample outputs in Appendix A.

AREA locates all photo records with center coordinates within the specified rectangular area. The user enters the "lower" and "upper" latitudes and "smaller" and "larger" longitudes to define the rectangle. The coordinate system origin is at the equator at 0 degrees longitude (Greenwich Meridian). Latitudes are entered as negative displacements for the southern hemisphere and range from -90 at the South Pole to +90 at the North Pole. Longitudes are entered as negative for east longitude and positive for west longitude. They range from -180 at the International Date Line, increasing east to west to 0 at the Greenwich Meridian, to +180 at the Date Line. The located set is named AREASET. An example area search follows:

U: AREA

C: ENTER LOWER LATITUDE:

U: -5 or 5S (define an area at the origin)

C: ENTER UPPER LATITUDE:

U: 7N or 7

C: ENTER SMALLER LONGITUDE:

U: -7 or 7E

C: ENTER LARGER LONGITUDE:

U: 4 45 or 4.75 or 4W 45

C: SET NAMED AREASET HAS 15 MEMBERS.

U: SETOUT = AREASET

C: SET NAMED SETOUT HAS 15 MEMBERS.

etc.

## 2.14 DEFINE LISTING DEVICES (PRINTER, TERMINAL)

The on-site user of MINIS on a computer equipped with a line printer has the option of listing his outputs on the printer. Remote site users do not have this option. To set the listing devices to the printer, enter:

C: >

U: PRINTER

C: >

To restore the listing device to the terminal, substitute **TERMINAL** for **PRINTER** in the above dialogue.

### SECTION 3 - SYSTEM DESCRIPTION

This section describes the operation of the major modules of MINIS. These descriptions are similar to the program description documentations included with each subroutine. The descriptions included with the programs for several of the modules are considered to be of sufficient detail and are not repeated in this section. The descriptions included in this section are the following:

- General System Operation
- Set Formation
- Lexical Analyzer
- Parser
- Instruction Synthesizer
- Execution Stack Interpreter
- Report Generation

Complete COMMON listings are also included in this section. A description of each entry is given for all names COMMON array.

**PRECEDING PAGE BLANK NOT FILMED**

## GENERAL SYSTEM OPERATION

The system performs all of its operations through a series of user input commands followed by appropriate system responses.

Following loading of the system, the data base select (DBLOAD) module is invoked. DBLOAD loads the data definitions of the named data base along with indexed variable information and any special vocabulary. Control is then passed to the main system modules.

The main system module (CONTROL) accepts user commands and calls up the compiler routines (LEXICAL, SYNTHE, PARSE) to generate an internal interpretive code. The code is examined to determine the type of command given and an appropriate execution submodule is invoked. Interactive operation modules are loaded into overlay zones and executed for direct commands such as FIELDFIN, INDEX, CREATE, SAVETEXT, etc. An argument list is composed for subroutines with variable lists such as OUTPUT, WRITE, or SUMSET. The module is loaded and control is transferred to it with a pointer to the argument list. Arithmetic statements are simply identified for subsequent execution as data is accessed during set formation and report generation.

As each module completes its operations program control is returned to the input stage of the CONTROL module.

## SET FORMATION

Set formation is initiated when the user defines a set. To identify the data base records which are members of the set, the routine EXSETD first calculates the set limits or endpoints. If an .AND. operation is used to define the set, the data base records listed in the candidate file are examined for set membership using routine SUBSET. For set definitions involving indexed fields, the index routines. VARINX and SCINDEX are used to determine the candidates for set membership. Otherwise, all data base records are candidates for set membership.

In order to determine if a data base record is a member of the set, the instructions which define the set must be executed using the data on the data base record. To do this, each symbol in the set instructions is first reverse scanned to identify and flag any created variables that may be needed to define the set.

The reverse scanning process is performed by REVSCN. The instructions defining each created variable needed to define the set are flagged and examined for any other created variables. If any are found, its instructions are also flagged and examined for created variables. This process continues until all instructions necessary to define a created variable have been flagged.

In order to determine if a data base record is a member of the set, EXCANS executes the set instructions and all flagged instructions using the routine EXQUAD. In EXQUAD, the operands of each instruction are located by EXOPRN and their values extracted from the data base record by EXTFLD. The necessary operation is then identified and performed and the result stored with that instruction. After the execution of all the required instructions, the final result is returned to EXCANS. A final result of true (-1) indicates that the data base record is a member of the set, and the record number is then stored in the first word of the ICAND array and the bit in the last two words of the array referring to the set is turned on. The array is then written in the candidate file and the set membership count incremented.

After each candidate has been examined for set membership, a message is returned to the user by EXSETD giving the number of data base records found to be members of the set.

## LEXICAL ANALYZER

The purpose of the execution stack interpreter for the MINIS is to examine the instruction types indicated by the quad stack entries and to initiate actions parser. Subroutine LEXCAL controls the three step process of isolating a sentence symbol, classifying the symbol and making an entry in the internal sentence stack. Two special functions of LEXCAL are the detection of unary plus and minus operators and the transfer of format specifications to format routines without further processing.

Symbol isolation is accomplished by subroutine GETSMB. A path through a simple tree structure is traced as successive characters are picked up from the input buffer. If no typing errors have occurred, then the symbol is determined to be one of the following general types of character strings:

- Alphanumeric
- Integer constant
- Real constant
- Special (.AND., .LE., etc.)
- Literal (Hollerith) constant

A flag is returned to LEXCAL indicating the type of character string that is stored in the temporary buffer, an error condition or the end of line indicator.

An alphanumeric string is characterized by a leading alphabetic character or a percent (%) sign followed by either alphabetic, numeric or percent characters.

An integer constant can have an optional plus or minus sign and a string of digits.

A real constant is identified by a leading sign and optional integer string followed by a decimal point and another string of digits that represent the fractional portion of the constant, an exponent may also be included which can consist of a sign, the letter E and a two digit integer which represents a power of ten. Special strings are surrounded by periods and contain only alphabetic characters. Literal constant strings are surrounded by "tic" marks and contain as many as eleven characters, none of which can be another "tic" mark. If any less than eleven characters appear in a

literal constant string, trailing blanks are supplied.

Next, LEXCAL passes control to subroutine CHKSYM to further classify the symbol and make entries in the symbol table when necessary. For alphanumeric and special strings, the symbol is lengthened to eight characters by adding trailing blanks.

The symbol table is searched and, if a match is found, the symbol table pointer is returned to LEXCAL. If not, the reserved word and the field name lists are checked in that order. A match in the reserved word list causes an integer reserved word code to be returned. No symbol table entry is made unless the symbol represents a logical constant. (either .TRUE. or .FALSE.). A match in the field name list causes the field name to be entered in the symbol table and a pointer to be returned to LEXCAL.

Integer, real and Hollerith strings are stored in the symbol table as integer, floating point or Hollerith values respectively and the symbol table pointer is returned to LEXCAL if an identical value does not already exist.

Finally, LEXCAL passes control to subroutine MAKSEN to build the internal form of the sentence to be used by the parser. Symbol table entry pointers are stored as negative numbers, and reserved word codes are stored as positive numbers in an array that allows up to one hundred symbols per sentence.

During lexical analysis the following errors can occur:

- Illegal Symbol - an unexpected or illegal character is encountered while building a symbol.
- Too many characters in symbol - more than eight characters in an alphanumeric string or more than 20 characters in a constant string.
- Symbol table overflow - no more 12 byte symbol table entries will fit in the symbol table common area.
- Too many symbols in sentence - more than 100 symbols in a sentence.

## PARSER

Subroutine PARSE controls the analysis of syntax of the internal sentence formed by the lexical analyzer. The general outline of the parser may be found in Algorithm 3 of the March 1970 Computing Surveys, pages 65 through 82.

The Data Base Access Language (DABAL) is set up as an  $N \times 4$  list structure as required in the algorithm. Each entry in the list structure, or syntax graph, contains a value field and three pointers. The value field contains a symbol code which is determined to be terminal or nonterminal by the first pointer field. This "DEF" pointer field can contain a positive pointer to the first in a list of entries that make up the right side of the production that defined the nonterminal symbol. Otherwise, the DEF field contains a special flag indicating that the symbol is a terminal. The next cell, called the "ALT" field, contains a pointer to the first in a list of alternative definitions for the nonterminal symbol. ALT can also contain a flag which indicates the end of the alternative definitions for the non-terminal symbol.

The "SUC" cell, the last pointer in the entry, points to the succeeding symbol in a definition. If no succeeding symbol exists a special flag is entered in place of a pointer.

PARSE contains a nonrecursive, back tracking algorithm which attempts to match the internal sentence formed by LEXCAL to a path through the syntax graph. Any time that the syntax graph indicates that a variable name or constant is required, subroutine PARVOC is called in order to check the existence of the proper entry in the symbol table. In the event that no path is found for the sentence or that a symbol is used improperly, a "SYNTAX ERROR" message is typed on the users terminal. If no errors occur, PARSE produces an  $N \times 2$  stack which contains all of the symbol codes in the sentence along with a positive integer indicating each symbol's level in the grammar tree.

Subroutine PARVOC checks for the existence of symbol table entries in the following

manner. Two arrays are defined in PARVOC to provide a list of legal symbol types for each terminal symbol code that is possible in the value field. The first array corresponds on a one for one basis with all of the values that are possible. For each value, a count is stored in this array that indicates the number of different symbol types that are stored in the second array for that value. The type of the allowable types listed in this second array for that value. The type of the symbol in question is compared with each of the allowable types listed in this second array to determine if a legal symbol has been used. If the proper code is present in the second array, the symbol may be newly created. A flag is set according to whether or not a match of symbol types was found and control is passed back to the parser.

## INSTRUCTION SYNTHESIZER

The function of the instruction synthesizer is to examine the output from the parser, determine sentence type and build a series of execution quadruples. Subroutine **SYNTHE** controls this process by picking up the sentence type from the second entry in the **N X 2** parse stack, determining where each clause begins and ends and passing each clause to subroutine **SYNTH 1** for further analysis.

Subroutine **SYNTH 1** begins by removing all nonterminal symbols from the **N X 2** stack, eliminating any grouping symbols that appear in the stack and sorting the stack members according to the level in the grammar tree. This exercise resolves all questions of hierarchy as well as reducing the number of symbols to be encountered by the rest of the routine.

Next, **SYNTH 1** begins the process of building the execution quadruple stack by determining what type of clause has been passed to it. If the clause is anything other than a subroutine cell, the scan of the **N X 2** array begins. As each entry is encountered a push down stack of symbol table entries is build, and members of this stack are popped up whenever operators are found. Unary operators cause one entry to be popped, and binary operators bring out two entries. A quadruple is built each time an operator is found and a pointer to that "quad is pushed onto the stack. An exception is the case of a "FROM...THRU..." operation. In this instance, no quadruples are formed until the **THRU** operator and its associated operand are found. Then, quadruples are formed for the equivalent expression stated in terms of relational operators. An example of this would be the case in which "X FROM A THRU B" would be converted to "X.GE.A.AND.X.LE.B". Thus, three quads would be formed when only two operators were found in the original expression.

For subroutine calls the **N X 2** array contains the symbol table entry number for the subroutine name and entry numbers for each of the arguments in the calling list. **SYNTH 1** flags the subroutine name and produces a string of quads with no operation

codes and pairs of arguments until all arguments are included.

Finally, if the clause processed by SYNTH 1 was a variable or set definition, the third field of a selected symbol table entry is supplied a value. For set definitions, a count of the number of sets that have been formed is kept. This count is incremented and the value is stored in the symbol table entry for the set name. For variable definitions, a pointer to the first quad that makes up the definition is entered into the symbol table entry for the variable name.

## EXECUTION STACK INTERPRETER

The purpose of the execution stack interpreter for the OLDMAN system is to examine the instruction types indicated by the quad stack entries and to initiate actions accordingly. Subroutine INTERP controls this examination process and provides the added capability of delayed execution and immediately executable instructions.

INTERP first checks a system flag to determine if the delayed mode is currently in effect. If so, this means that most instructions are to be stored for later execution. The only exceptions are a set of immediately executable subroutines that have special codes in the symbol table. One of these subroutines is "EXECUTE" which resets the immediate mode flag and causes all delayed instructions to be executed.

If the system is not in delayed mode, a check is first made to determine if the next instruction is a "DELAYEX". If so, the flag discussed in the preceding paragraph is set so that the delayed mode will be entered. If not, subroutine EXSTMT is called with starting and ending quad pointers of the instruction sent as arguments.

EXSTMT extracts the instruction type from the first quad. No action is required when a variable definition instruction is encountered so control immediately passes back to INTERP in this instance. Set definition instructions, on the other hand, require extensive processing to arrive at a group of candidates that fit the criteria described in the instruction. The starting and ending pointers are passed on to the set formation module so that the quad table can be searched for all variables referenced by the instruction. After the set has been formed, a message is typed on the user's terminal indicating the number of members included in the set. This count is also stored in the SETCNT common area and control is returned to INTERP.

## SET SUMMATION

The function of the routine SUMSET is to sum the values of a field or created variable over a given set. This routine is immediately executed upon a call to SUMSET by the user. The set number, variable or field to sum, and the variable name under which the result is stored, is obtained from the quad created by the call to sumset. The candidate file is scanned and for each candidate in the set, the corresponding data base record is read into core. The field is then extracted or, in the case of a created variable, the quad executed to determine the value for that record, and the value is added to the sum. After all candidates are considered, the result is printed out. The result is also stored in the quad table and can be used in further calculations in the same manner as a created variable.

## REPORT GENERATION

The report generation module (OUTPUT) is designed to allow output of data base records within a given set. All or only requested fields may be output, along with any created variables that may be associated with the given set. The printout can be directed to the originating terminal device (CRT, TTY, Exccuport, etc.) or optionally to the line printer.

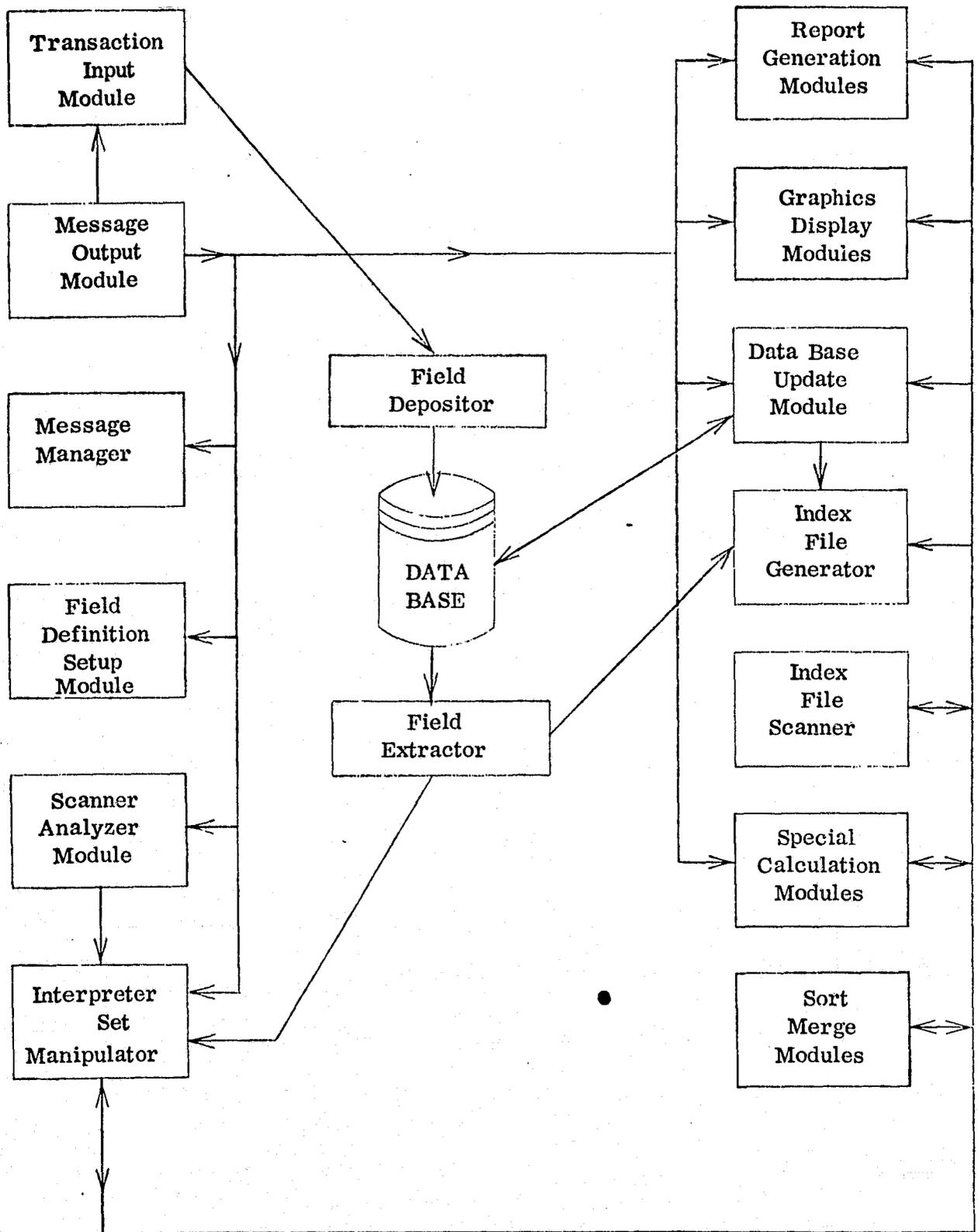
The output is arranged in columnar form, with headings at the top of each page, above each column of each data base record in the set of data and the values for each parameter requested on the same line under the appropriate heading. Spacing between each column of data is determined by the number of parameters requested. If more parameters are requested than there is room for on one page, a second page (or more if necessary) will be printed after all data is printed for the parameters that could be fit on the first page.

The type of heading for each parameter is determined by the amount of spacing available. When less than six spaces are available for a column of data, the parameter name is used vertically in the heading. When six to eight spaces are available, the parameter name is used horizontally. For spacing of greater than eight spaces, the field title is printed horizontally on the least number of lines necessary with a maximum of six lines. For a created variable, the parameter name is always used. Parameter names are always right justified over the columns of data.

The format used to print each field is found in the corresponding field definition in common (FIELDS). Created variables always use the format F10.2. Spacing between each requested parameter is determined by the field width of the format of each parameter. Extra spaces on each page are divided equally among each field on the page. The maximum column width is 20 spaces and the minimum is one more than the necessary field width.

Upon request from the user to output data, a table is formed listing the parameters to be output and the required field width for that parameter. This table is then used

to build the headers for each page, which are stored in a format array. The data output formats are then built and stored in another format array. After the header is printed at the top of the page, the data for each line is extracted and printed one line at a time. New pages are begun with a header at the top, when more than 33 lines are needed to print the data (60 for the line printer) or if more parameters were requested than could be fitted on one page.



ORIGINAL PAGE IS  
OF POOR QUALITY

## LABELED COMMON

NAME	LENGTH	USAGE
SYMTAB	404	Symbol Table
QUATBL	505	Quad Table
FILDEF	231	File Definitions
DEFNAM	10	Name of FILDEF File
DATARY	1002	Data Array
FIELDS	1200	Field Definitions
IOARRAY	150	I/O Array and Device Info
MESNDX	300	User Message Index
REUSE	10001	Scratch Core Area
COMTAB	100	Communications Table
FILSTK	110	Defines Disc File Stack
OUTBUF	881	I/O Buffering Area
SETCNT	51	Set Information

## SYMBOL TABLE

COMMON/SYMTAB/NSYMS, MXSYM, NSUBS, NISUBS, ISYMS (4, 100)

NSYMS = No. of Symbols in Table  
 MXSYM = Maximum No. of Symbols = 100  
 NSUBS = No. of Defined Subroutines  
 NISUBS = No. of Immediately Executable Subroutines (Listed First in Symbol Table)  
 ISYMS = Entry For Each Symbol

BYTES	INFORMATION
1 - 8	Symbol Name (ASC II) Constants - Value of Constant (Left Justified in Minimum No. of Words)
9	Symbol Type (See Code Table) (Integer)
10 - 12	Referenced Array Pointer (Integer)

## SYMBOL TYPE CODES

CODE	TYPE	REFERENCED ARRAY
0	Subroutine Name	Unused
1	Subroutine Name	Unused
2	Field Name	Field Definition Array (Fields)
3	Set Name	Corresponding Bit in ICAND (2) and ICAND (3) (SETCNT) = Set Number
4	Created Symbol	Last Quad Defining Created Symbol (QUATBL)
5	Subroutine Name	Unused
6	Constant Integer	Unused
7	Constant Real	Unused
8	Constant Logical = 0 False = 1 True	Unused
9	Constant Hollerith	Literal Storage if Needed
10	Special Reserved Words "All" - Indicates Printout of All Fields Defined For the Data Base	Unused
11	Namelist Title	Message no. in User Message File
12	Header Format Title	Message no. in User Message File
13	Data Format Title	Message no. in User Message File
14	Schema Title	Message no. in User Message File
255	Illegal Symbol	

## QUAD TABLE

COMMON/QUATBL/MXQUAD, IQUAD, IDUM1, IDELX, NSET, IQU (5,100)

MXQUAD = Maximum No. of Quads = 100  
 IQUAD = Last Quad Created (Current No. of Quads)  
 IDUM1 = Unused  
 IDELX = First Quad Following a Delay  
 NSET = Number of Defined Sets  
 IQU (5,100) = Quads (5 words)

WORD	USAGE	FORMAT
1	Operation (See Table)	Integer
2	1st Operand Pointer	Integer
3	2nd Operand Pointer	Integer
4-5	Result of Operation	Floating Point

## FIRST QUAD OF EACH STATEMENT

WORD 1 = 73  
 WORD 2 = Statement Definition (See Table)  
 WORD 3 = Execution Flag    -1 Execute  
                                   0 Do Not Execute  
                                   -2 Previously Executed  
 WORD 4-5 = Final Result of Statement (After Execution)

## OPERAND POINTERS

Positive - Point to Symbol Table Entry  
 Negative - Point to Quad Table Entry

## OPERATION CODES

CODE	SYMBOL	DEFINITION	RESULT
1	*	Multiplication	(1st operand * 2nd operand)
2	/	Division	(1st operand/2nd operand)
3	+	Addition	(1st operand + 2nd operand)
4	-	Subtraction	(1st operand-2nd operand)
5	=	Assignment	(2nd operand)
6		Subroutine Call	(1st operand)
7		NOP	Result = -1=True otherwise Result =0=False
8	.OR.	Logical OR	1st Operand + 2nd Operand < 0

QUAD TABLE (continued)

9	.AND.	Logical AND	1st Operand + 2nd Operand + =0
10	.XOR.	Logical XOR	1st Operand + 2nd Operand + 1=0
11	.ANDNOT.	Logical NOT	1st Operand + 2*2nd Operand + 1 = 0
12	.ORNOT.	Logical ORNOT	1st Operand - 2nd Operand - 1<0
13	.NAND.	Logical NAND	1st Operand + 2nd Operand + 1=0
14	.NOR.	Logical NOR	1st Operand + 2nd Operand =
15	.GT.	Relational Operator	1st Operand - 2nd Operand > 0
16	.LT.	Relational Operator	1st Operand - 2nd Operand < 0
17	.EQ.	Relational Operator	1st Operand - 2nd Operand = 0
18	.GE.	Relational Operator	1st Operand - 2nd Operand ≥ 0
19	.LE.	Relational Operator	1st Operand - 2nd Operand ≤ 0
20	.NE.	Relational Operator	1st Operand - 2nd Operand ≠ 0
21	.NOT.	Unary NOT	Result = -1. 0- 1st Operand
22	+	Unary +	Result 1st Operand
23	-	Unary -	Result - 1st Operand
24	**	Exponentiation	Result 1st Operand ** 2nd Operand
73		Beginning Statement Code	

STATEMENT CODE

1  
2  
3  
4  
5  
6  
7  
Code + '40 = THEN  
Code + '60 = ELSE

DEFINITION CODES

Type Statement  
Set Definition  
Unused  
If Statement  
Unused  
Subroutine Call (Assignment Type)  
Subroutine Call (Non-Assignment type)  
(Code + 32)  
(Code + 48)

# FILE DEFINITIONS

## COMMON/FILDEF/IDF (231)

Word	Usage	Format
1-3	Data Base File Name	3A2
4	No. of Records	I
5	No. Of Bytes/Record	I
6-8	Users Message List File Name	3A2
9	No. of Records	I
10	No. of Bytes/Record (NBW)	I
11-13	Field Definition File Name	3A2
14	No. of Records = No. of Fields	I
15	No. of Bytes/Record (18)	I
16-18	Synonym and Vocabulary File Name	3A2
19	No. of record = no. of synoym	I
20	No. of Bytes/Record	I
21-23	User Titles File Name	3A2
24	No. of Records	I
25	No. of Bytes/Record (6*NEW)	I
26-28	Candidate File Name           SCRCH 1 SCRCH 2 or	3A2
29	No. of Records = No. of candidates	I
30	No. of Bytes/Record NBW + 6*NEW	I
31-33	Transaction input file name (for up dates)	3A2
34	No. of Records	I
35	No. of Bytes/Record	I
36-38	System Message File Name (MESSYS)	
39	No. of Records	
40	No. of Bytes/Record (NEW)	
41-N	Index File ID's in 10 word blocks (Total of 19 blocks)	
11-13	Index File Name XFNOLT   X=meshed character from all characters of the data base file name. FNO=Field No. (3 digits) L=Level No. (1 digit) T=Type No. (1digit)	3A2

I4	No. of Records = no. of entries	I
I5	No. of Bytes/Record	I
I6-I7	Level 1    Level 2 Field No.    Minimum Value	I or FLT
I8-I9	Level 1                      Level $\geq$ 2 No. of bits for field    Segment size	I or FLT
I10	No. of Bits for Pointer	I
231	Unused	

All index file ID's are entered in sequence so that a level I file appears before a level 2, level 2 file before a level 3, and so forth. When a file is deleted, all successive files for that field are deleted also.

## DEFNAM

### COMMON/DEFNAM/NAMDEF (10)

NAMDEF	(1-3) - name of FILDEF file	3A2
	(4) - no. of records in file (231)	I
	(5) - no. of bytes/record (1 word/record)	I
	(6-10)- Unused	

The name given by the user at data base select time is the name of the file definition file.

**DATARY**

**COMMON/DATARY/IDM1, IDM2, IARRAY (1000)**

<b>IDM1</b>	<b>Record no. stored in IARRAY</b>
<b>IDM2</b>	<b>Unused</b>
<b>IARRAY</b>	<b>Temporary core storage of one data base record.</b>

# FIELDS

## COMMON/FIELDS/IDFLD (1200)

For each field, 18 bytes hold the field definition. A maximum of 200 fields are allowed.

Bytes	Bits	1	2	3	4	5	6	7	8	
1	0									
2	8									
<b>Word 1</b>	3									
	4	Field Identifier								
	5	(8-Byte Name) (Note 5)								
<b>Word 2</b>	6									
	7									
	8									
								A		
<b>Word 3</b>	9	64 (Note 6) Field # of key, if any I				68 Key value for presence of this field I				
	10									
	11	First bit position of field in record I								
<b>Word 4</b>	12	88 No. of bits in this field I								
		96 Data type (Note 1) I				100 Bits in first sub-field (Note 2) I				
		104 (Note 3) No. of repeats, if any I						110 Field out-		
		112 Put format (A, I, or F) (Note 4) A						118 Format out-		
<b>Word 5</b>	15									
		120 Put field width I				124 Format significant digits I				
		128 Iden. field flag yes=set				129 Code conversions (Note 9)		133 H. flag (Note 10)		134 Levels of indexing I
		136 Message no. of field title (Note 7)						143		
<b>Word 6</b>	18									

Field Definition Bit Map

## FIELDS (Continued)

### Note 1:

#### Field Data Type Codes

<u>Code</u>	<u>Type</u>
0	Integer
1	Real (Floating Point)
2	Message Number
3	UTM Coordinate
4	Boolean (T or F)
5	Signed integer (two's complement form)(- 32768 min. value)
6	Character
7	Text (Note 8)
8	Split Field
9	Both Integer
	Both Real
10	Both Messages
11	1st Message
	2nd Integer
12	
	2nd Real
13	
14	Unused
15	Reserved for 'NO DATA PRESENT'

The first portion of any split field must be an integer identifying code. The code may also be a message or integer number.

### Note 2:

The number of bits in the second portion of a split field is equal to byte 12-byte 13 (b<sub>4</sub>-b<sub>7</sub>). The number of bits in the first portion of the field (number of bits in the subfield) is used to identify the field (Note 7).

### Note 3:

If non-zero, it indicates the number of successive bytes in which the field may appear. Byte size is determined by byte 12. Repetitive fields must be split. The first portion must match byte 18.

### Note 4:

Display format type - Fortran type codes.

If I format is used with output field width (bits 118-123) greater than 1 and significant digits (bits 124-127) equal to 1 (example: 3I1), then data will be printed with leading zeros if any.

D format--for dates--will cause a resultant format of I2, '- ', I2, '- ', ... (example: D6 = I2, '- ', I2, '- ', I2).

## **FIELDS (continued)**

### **Note 5:**

If the field name is longer than eight characters, an identifier is substituted and byte 18 is set to the message number containing the actual field name.

### **Note 6:**

$b_0 - b_3$  is the field number the key or record type identifier.

$b_4 - b_7$  is the value the identifier must be for this field to be present.

If byte 9 is zero, it is assumed that the data base is not keyed or at least that the field is not keyed.

The use of a key is included to permit dissimilar fields of a record to occupy the same physical locations. The use of a key permits linking of two different data structures whose only common elements are record length and the unique identifier field of which, the key may be a sub-field.

### **Note 7:**

A block within a record may be set aside for optionally present fields. All optional (repetitive) fields are split fields, the first portion of which is a key identifying the field. The same key must appear in byte 18 of the field ID for the field to be located. This key (byte 18) may serve double duty as a message number.

### **Note 8:**

If describing a text field, the number of bits is interpreted as the number of bytes or characters in the field. Text data must start in the record at a bit number which is a multiple of 8 (on a byte boundary) from the start of the record.

### **Note 9:**

The data in a field may go through a code conversion prior to use by the data base management routines. When a code conversion is used, (Non-zero) the format information in the field definition applies to the converted data. The data type in the field definition must be consistent with the data in the field. Where the code conversion implies a data type change, the new data type is returned by the field extraction routine (EXTFLD). The following code conversions are assigned:

Code Number

0

Conversion

No Conversion

1	$A_n$ to Integer - where n= no. of bits in field/8 no. of bits must be a multiple of 8 and each character must be numeric.
2	In to An Where n=no. of significant places in the field definition Max=4*NBW.
3	In to An Skewed (used to build ASCII add 32 to I character from 6 bits)
4	Integer to Real
5	Integer to Real and divide by 1000
6	Integer to real and divide by 100
7	Integer to real and divide by 10
8	Integer multiply by $10^6$
9	Integer multiply by $10^5$
10	Integer multiply by $10^3$
11	MERITS Roll Number Conversion: ITEMP = INT/16 IOUT = ITEMP * 100 + INT - ITEMP * 16
12	Date: IDATA(1) = IDATA(1) * 10000 + IDATA(2) * 100 + IDATA(3)
13-15	Unused

Note 10:

H. Flag - Hierarchy flag - indicates that data base is sorted in ascending order on this field and will not be indexed on this field.

## IOARAY

COMMON/IOARAY/IBUF(150)

### IBUF (1-132) - Input/Output Buffers

(133)	INDV	Input device no.
(134)	IJDV	Terminal device no.
(135)	IODV	Output device no.
(136)	LENI	Line width of input device
(137)	LEN2	Line width of terminal
(138)	LEN3	Line width of output device
(139)	NLPP	No. of lines per output page
(140)	IDU(3)	User ID code
(143)	ITIM	Time
(144)	IUNIT	TEK=1, TTY=0
(145)	IMOD	Mode IMMEDIATE = 1 DELAYED = 0
(146)	IBAUD	Baud rate for Tektronix plot routines
(147)	IEXT	Data base extension ID
(148)	NFIL	No. of files in I/O subroutines
(149)	IZDV	Zero device no.
(150)	NBW	No. of bytes/word

**MESNDX**

**COMMON/MESNDX/MINX (300)**

**MINX (1-200) - Pointers to 1st word of field titles**

**MINX (201-300) - Pointers to 1st word of user messages**

**FILE - IDEF (6-10) - User message list 1 word/record**

**For each message:**

<b>Word 1</b>	<b>No. of bytes in message</b>
<b>2</b>	<b>Message no.</b>
<b>3-N</b>	<b>Message</b>

**N=No. of bytes in message/NBW**

**REUSE**

**COMMON/REUSE/IRLEN, IREUSE (10000)**

**IRLEN - Length of reuse area (10000)**

**IREUSE (1-10000) - Reusable (scratch) core area**

## COMMUNICATIONS TABLE

### COMMON/COMTAB/ICON (100)

ICON (1) - Unused

ICON (2) - First quad of subroutine call

ICON (3) - Last quad of subroutine call

ICON (4) - Symbol Table entry for EXECUTE (subroutine)

ICON (5) - Insert Flag

ICON (6-99) - Unused

ICON (100) - Number of candidates in set

## ACTIVE FILE STACK

COMMCN/FILSTK/IPRIOR (10), MFILES (10, 10)

This common retains information on any currently active disc files.

**IPRIOR (1-10)** - Pointers to file blocks (MFILES) in order of greatest priority, such that the pointer to the file currently being used is in IPRIOR (1).

**MFILES (1-3, N)** - File name

(4, N) - No. of records in file

(5, N) - No. of bytes/record

(6, N) - Pointer to entry in file

(7, N) - Logical unit number of file

(8, N) - File status flag

0 = opened for read operation

1 - opened for write operation

3 - error in I/O operation

(9, N) - Buffered write pointer (1- N)

Pointer to buffer entry in common OUTBUF

(10, N) - Record number for start of buffer data.

## OUTBUF

COMMON/OUTBUF/NWOUT, JOUT (880)

NWOUT - Number of words per OUTPUT Buffer.  
= 220 for Datacraft.

JOUT - OUTPUT buffer of segments with 'NWOUT' words per segment

This common will vary for different host machines. The total buffer size is the product of the number of active files and the number of words per buffer.

**SETCNT**

**COMMON/SETCNT/ISET (48), ICAND (3)**

**MAXIMUM NO. OF DEFINED SETS = 48**

**ISET (1-48) = No. of set members for sets 1-48**

**ICAND (1) = Data base record no. of candidate**

**ICAND (2-3) - Bits 1-48 correspond to sets 1-48. For each set that the candidate is a member, the corresponding bit is set.**

**ICAND (1-3) - Is input from the candidate file IDEF(26-30)**

**The no. of candidates in the file is also stored in COMTAB/ICON (100)**

## MINIS SYSTEM INSTALLATION

MINIS systems that now exist on the Datacraft 6024 or NOVA minicomputers can be conveniently installed beginning with either source programs or binary files provided on magnetic tape. The general method of installation is to form relocatable binary files with the FORTRAN compiler or assembler, generate a user library with the host computer's library editor and produce executable modules with predefined relocatable loader commands. MINIS data files necessary for file control, index generation, field definition, user messages, and system messages are also provided on the system tape.

On the Datacraft a system tape is created with files corresponding to each element named in the LOAD command listed in Table 3-1. In addition, a file containing user library routines and a file for MINIS system messages are included. These files can be loaded into a file managed unit as "SRC" files, and all software files can be compiled by the FORTRAN compiler except for IBEXT, IBDEP, F\$DDFU, and BISORT which are assembly language routines.

After all "BIN" files are produced, an overaly load is done with the load statement listed in Table 3-1. A BATCH command file can be generated in a source file to make the device assignments listed in Table 3-1 with all FM assignments set to the desired file managed unit. Then, to execute the MINIS system, the user can first make device assignments by simply entering a BATCH command and typing EXEC MINIS. At this point, the procedures described elsewhere in the manual for defining a database, defining fields, generating indexes and saving user messages can be carried out.

On the NOVA, the MINIS system is designed to run under the Real-time Disk Operating System (RDOS) with files located in two directories. After initializing the two directories, named DP2 and DP3, two tape files can be read in using the CLI commands LOAD MTO:0 and LOAD MTO:1. All source files to be compiled or assembled are named "XXXXXX.OM" where "XXXXXX"

**PRECEDING PAGE BLANK NOT FILMED**

is a six-character routine name and ".OM" is a file extension used to separate MINIS software from any other source files that may exist in the directory. Several command files, identified by the .CM extension, must be executed indirectly to produce all of the save files necessary for MINIS operation. The names and contents of these files are listed in Tables 3-2 through 3-11.

All data files were automatically loaded with the LOAD commands described above. These files reside in directory DP2 and may be identified by listing all ten character file names with no extension. All data files may be examined and modified by the CLI's octal edit routine OEDIT.

Since device assignments are made under program control, all the user needs to do to initiate execution is type MINIS. Other sections of this manual describe starting up a database, saving user messages, and accessing the database.

\$A 4, DUM, 7, DUM, 10, FM2, 11, FM2, 12, FM2, 13, FM2, 5, FM2, 6, LP, 14, FM2

\$A 15, FM2

\$LOAD

\*LOCAL MINIS, DBLOAD, REVSCN, VARINX, VARCHK, EXQUAD, FLDFIN, MESMAN,  
FMATS, WRDAT, INDEX, SUMSET, LEXCAL, PARSE, SYNTH, POINT, NAMLST,  
INSERT, WRITE, INITAL, PROCSS, SRTMER

\$KB

Table 3-1

1/21/77

NOVA 2/10

FILENAME: DP2:OLLIST.CM

000001 DELETE/V OLLIST.LS  
000002 RLDR OLLIST.LS/L MINIS ~  
000003 ECALSUB SUMSET GETCOM QUITTT RESETT, INTERP EXSTMT EXSEDDJ ~  
000004 CDBLOAD ~  
000005 ,LEXCAL GETSMB GETNXT ~  
000006 ,PARSEE PARVOC ~  
000007 ,SYNTHE ~  
000008 ,STEDMP DMPSTK ~  
000009 ,INSERT INSELS ~  
000010 ,VARINX SCINDX ~  
000011 ,VARCHK VARVAL ~  
000012 ,DP3:OBVIDD DP3:STNMMD ~  
000013 ,REVSCN RVSCAN ~  
000014 ,SUBSET EXCANS EXQUAD EXOPRND ~  
000015 ECRREW STNAME IBLANK MESHMH CVREEL LINMES LININN ~  
000016 ,CHKSYM CHKFLD SCANTB CVREAL MAKSEN XFERRR KUMPAR ~  
000017 ,SYNTH1 CHKTYP BUILD ~  
000018 ,RDULIN RDFLIN UNPKER SCIFID DP3:BISORT ~  
000019 ,BNSRCH RFIELD EXFLDD EXTFLDD ~  
000020 MESOUT OLDMAN.LB FORT.LB

Table 3-2

ORIGINAL PAGE IS  
OF POOR QUALITY

1/21/77

NOVA 2/10  
FILENAME: DP2:INDEXX.CM

ORIGINAL PAGE IS  
OF POOR QUALITY

000001 DELETE/V INDEXX.LS  
000002 RLDR INDEXX.LS/L DP3:INDEXX/S ~  
000003 INXXX INDEXX GENDEX GENATE ~  
000004 LWRDEF MESHMH , DP3:SORTIX DP3:BISORT DP3:MERGGG ~  
000005 , UNPKER, ~  
000006 PACKER EXFLDD EXTFLDJ ~  
000007 OLIMAN.LB FORT.LB

Table 3-3

NOVA 2/10

1/21/77

FILENAME: DP2:FLDFIN.CM

000001 RLDR/M/E FLDFNN GETCOM FLDFIN FINFLD MESMAN MESOUT  
OLDMAN.LB FORT.LB

Table 3-4

1/21/77

FILENAME: DP2:OUTPUT.CM

000001 DELETE/V OUTPUT.LS  
000002 RLDR OUTPUT.LS/L DP3:OUTPTT/S OUTPTT OUTPUT EXQUAD EXOPRN MESOUT  
000003 CFMATSS NORMMM"  
000004 ,REVSCN RVSCAN"  
000005 ,WRTOAT EXTFLDJ"

Table 3-5

ORIGINAL PAGE IS  
OF POOR QUALITY

1/21/77

NOVA 2/10  
FILENAME: DP2:WRITEE.CM

000001 RLDR WRITET WRITEE REVSCN RVSCAN EXQUAD EXOPRN EXTFLD  
OLDMAN. LB FORT. LB

Table 3-6

1/21/77

NOVA 2/10

FILENAME: DP2:NAMLST.CM

000001 RLDR NAMLTT NAMLST MESMAN OLIMAN.LB FORT.LB

Table 3-7

**ORIGINAL PAGE IS  
OF POOR QUALITY**

1/21/77

NOVA 2/10

FILENAME: DP3:POINTT.CM

000001 RLDR CALLPT POINTT COORDN LSEARCH BSEARCH INCORD BLOKEM HITESS

Table 3-8

1/21/77

FILENAME: DP3:CREATE.CM

000001 DELETE/V CREATE.LS  
000002 RLDR CREATE.LS/L ^  
000003 CREATT CREATE LPRSTR,INITAL QDFORM DELBLK SCANNN CUREAL ININTT,  
000004 PROCSS RDLREC STELEM C6T088 C8T066 C2T088 FLTPPT COMPUT GETOPR ^  
000005 SUMARY SUMPPO PRCENT CHRTRN SEARCH STOFLO PACDAT, ^  
000006 SORTER SORTIX MERGER FINFLD WRTEFF BISORT FILEND, ^  
000007 MRGERR WRTEFF FNNFLD EXTFLD3 ^  
000008 MESOUT NEWFILWRT PFILWT REDWRT OLDMAN.LB FORT.LB

Table 3-9

ORIGINAL PAGE IS  
OF POOR QUALITY

1/21/77

NOVA 2/10  
FILENAME: DP3:MINDEX.CM

000001 DELETE/V MINDEX.LS  
000002 RLDR MINDEX.LS/L MINDXX MINDEX SORTOT BISRT1  
000003 NEWFILWRT PFILWT REDWRT OLDMAN.LB FORT.LB

Table 3-10

NOVA 2/10

1/21/77

FILENAME: DP3:MERGIT.CM

000001 DELETE/V MERGIT.LS  
000002 RLDL MERGIT.LS/L MERGXX MERGIT BUNDEX ~  
000003 NEWFILWRT PFILWT REDWRT FILEND OLDMAN.LB FORT.LB

Table 3-11

ORIGINAL PAGE IS  
OF POOR QUALITY

**PRECEDING PAGE BLANK NOT FILMED**

**APPENDIX A**  
**EXAMPLES ON THREE DATA BASES**

## EXAMPLES OF POINT AND AREA SEARCHES IN THE LANSAT DATA BASE

U: AREA  
C: ENTER LOWER LATITUDE:  
U: 26  
C: ENTER UPPER LATITUDE:  
U: 28 55  
C: ENTER SMALLER LONGITUDE:  
U: 80  
C: ENTER LARGER LONGITUDE:  
U: 83 30  
C: SET NAMED AREASET HAS 6 MEMBERS

U: POINT  
C: ENTER LATITUDE:  
U: 28 53  
C: ENTER LONGITUDE:  
U: 81 15  
C: SET NAMED POINTSET HAS 2 MEMBERS

AREASET and POINTSET are system supplied set names that may be used just as user named sets. In the examples above, AREASET consists of all photos in an area from 26 degrees to 28 degrees 55 minutes north latitude and 80 degrees to 83 degrees 30 minutes west longitude.

POINTSET consists of all photographs at the point 28 degrees 53 minutes north latitude and 81 degrees 15 minutes west longitude.

# EXAMPLES OF GENERATED REPORTS

LANSAT output using system supplied formats and headers.

ORIGINAL PAGE IS  
OF POOR QUALITY

```

ENTER DATA BASE NAME (6 CHAR)
MERITS
>AREA
ENTER LOWER LATITUDE:
30N
ENTER UPPER LATITUDE:
32N
ENTER SMALLER LONGITUDE:
80W
ENTER LARGER LONGITUDE:
82W
SET NAMED AREASET HAS 4 MEMBERS.
>OUTPUT(AREASET,ALL)
    
```

S	D	H	O	L	LON	O	DATE	C	A	M	M	M	M
A	S	H	R	A		E		L	R	I	2	3	4
T	L	M	B	T		V		D	C	P	P	P	P
N		M	N			I		C	H	O	O	O	O
U		S	U			D		O	V	S	S	S	S
M			M					V	A	N	N	N	N
1	002	16321	27	31N35	96W54	0	7-25-72	10	501	1	13	0	32
1	009	15264	124	31N45	80W28	2	8- 1-72	30	4015	24	54	84	114
1	002	16314	27	33N00	96W28	0	7-25-72	0	1803	158	164	170	176
1	009	15262	124	33N11	80W 2 2	2	8- 1-72	80	4015	23	53	83	113

MFROLL	MFPOSN	R	R	M	R	M	R	R	R	M	M	M	M
		E	B	S	E	S	1	2	3	1	2	3	4
		G	V	S	M	M	C	C	C	C	C	C	C
		I	I	I	O	O	U	U	U	U	U	U	U
		O	D	D	D	D	A	A	A	A	A	A	A
		N			E	E	L	L	L	L	L	L	L
	0	0	1	3	3	1	1	3	2	3	0	0	0
10001	693	1	2	2	1	1	2	2	2	2	2	3	4
10001	5	1	0	2	0	1	0	0	0	2	2	4	2
10001	691	1	2	2	1	1	2	2	2	2	2	2	4

LANSAT output using user supplied name list, headers, and format.

OUTPUT(POINTSET, IDSMER, HEDMER, FORMER)

OBSRV'N-ID	ORB	ROLL/POS 1-4	LAT	LONG	CC	MFR/POSN	MO	DA	YR
1009-15273	124	4015/26, 56, 86, 116	28N53	81W17	20	10001/ 697	8	1	72
2041-15165	570	8004/223, 233, 243, 253	28N53	81W15	20	10003/ 447	3	4	75

The following is an example of a session that might be held using the land use data base.

ENTER DATA BASE NAME (6 CHAR)  
 USER: LNDUSE

>PPLPSM=PPL2HS\*HOUSES\*2.6

>CROWDED=PPLPSM FROM 75 THRU 100

SET NAMED CROWDED HAS 28 MEMBERS

>OUTPUT(CROWDED,CELLID,COUNTY,PPL2HS,HOUSES,PPLPSM)

UTM CELL ATOR.	DESIGN .	COUNTY .	CODE .	PEOPLE PER HOUSEHOLD.	NUMBER OF OCCUPIED H OUSEHOLDS.	PPLPSM
-------------------	-------------	-------------	-----------	--------------------------	---------------------------------------	--------

SER9138		111		3.10	11	88.6600
SER9139		111		3.20	11	85.8000
SER9238		111		3.10	11	88.6600
SER9536		111		3.00	11	85.8000
SER9932		111		3.00	12	93.6000
SER8412		159		3.10	11	88.6600
SER8912		159		3.10	10	80.6000
SER9116		159		2.90	12	90.4800
SER9215		159		2.90	11	82.9400
SER9721		159		2.90	12	90.4800
SER9820		159		2.90	10	75.4000
SER9821		159		2.90	11	82.9400
SER9822		159		2.90	13	98.0200
SER9924		159		2.90	11	82.9400
SER6127		165		3.10	11	88.6600
SER6237		165		3.10	10	80.6000
SER6326		165		3.10	11	88.6600
SER7327		169		3.10	11	88.6600
SER7425		169		3.10	10	80.6000
SER7725		169		3.10	10	80.6000
SFQ3764		175		3.20	12	99.8400
SFQ3565		185		3.10	11	88.6600
SFQ3668		185		3.10	11	88.6600
SFQ4264		185		3.40	9	79.5600
SER6013		189		3.20	11	91.5200

UTM CELL ATOR.	DESIGN .	COUNTY .	CODE .	PEOPLE PER HOUSEHOLD.	NUMBER OF OCCUPIED H OUSEHOLDS.	PPLPSM
-------------------	-------------	-------------	-----------	--------------------------	---------------------------------------	--------

SER6517		189		3.20	10	83.2000
SER7214		189		3.20	10	83.2000
SER7712		189		3.20	12	99.8400

>POINTS=0

>IF GWQUAL .EQ. 1 THEN POINTS=POINTS+10

>IF GWQUAN .EQ. 1 THEN POINTS=POINTS+15

>IF SWQUAN .EQ. 2 THEN POINTS=POINTS+20

>IF SCHOOL .NE. ' ' THEN POINTS =POINTS+5

>IF LANFIL .EQ. 'S' THEN POINTS=POINTS+10 ELSE POINTS=POINTS-10

>IF SETPTIC .EQ. 'S' THEN POINTS=POINTS+15 ELSE POINTS=POINTS-10

>OUTPUT(CROWDED,CELLID,COUNTY,GWQUAL,GWQUAN,SWQUAN,SCHOOL,LANFIL, :

>SEPTIC,POINTS:

>

>)  
 UTM CELL DE COUNTY  
 SIGNATOR. CODE.

ORIGINAL PAGE IS  
 OF POOR QUALITY

G G S S L S  
 W W W C A S  
 Q Q Q H N P  
 U U U O F I T  
 A A A O I L C  
 L N N L L C

POINTS

SER9138	111	2	2	3		M	M	-20.0000
SER9139	111	2	2	3		M	M	-20.0000
SER9238	111	2	2	3		M	M	-20.0000
SER9536	111	3	2	3		M	M	-20.0000
SER9932	111	3	2	3		M	M	-20.0000
SER8412	159	3	2	6		B	B	-20.0000
SER8912	159	3	2	6		B	B	-20.0000
SER9116	159	3	2	3		B	B	-20.0000
SER9215	159	3	2	3		B	B	-20.0000
SER9721	159	3	2	2		B	B	0.0000
SER9820	159	3	2	2		B	B	0.0000
SER9821	159	3	2	2		B	B	0.0000
SER9822	159	3	2	2		M	M	0.0000
SER9924	159	3	2	2		M	M	10.0000
SER6127	165	3	2	3		M	S	5.0000
SER6237	165	3	3	3		M	M	-20.0000
SER6326	165	3	2	3		B	B	-20.0000
SER7327	169	3	2	3		B	B	-20.0000
SER7425	169	3	2	6		B	B	-20.0000
SER7725	169	3	2	3		M	M	-20.0000
SF03764	175	1	2	3		S	M	10.0000
SF03565	185	2	1	3		S	M	15.0000
SF03668	185	2	2	3		S	S	25.0000
SF04264	185	1	2	3		S	M	10.0000
SER6013	189	3	1	6		B	B	-5.0000

UTM CELL DE COUNTY  
 SIGNATOR. CODE.

G G S S L S  
 W W W C A S  
 Q Q Q H N P  
 U U U O F I T  
 A A A O I L C  
 L N N L L C

POINTS

SER6517	189	3	1	6		B	B	-5.0000
SER7214	189	3	2	6		B	B	-20.0000
SER7712	189	3	2	6		B	B	-20.0000

>  
 >SUMPTS=SUMSET(CROWDED,POINTS)

SUMPTS = -265.0000

>SUMHOV=SUMSET(CROWDED,HOUSES)

SUMHOV = 306.0000

>SET189=CROWDED .AND. COUNTY .EQ. 189

SET NAMED SET189 HAS 4 MEMBERS

>OUTPUT(SET189,ALL)

Z	UTM	CEL	H	P	C	D	S	C	C	S	R	B	S	O	L	L	S	D	P	A	S	L	T	R
O	L	DESIG	O	P	O	E	M	O	E	O	I	A	C	W	A	A	L	E	E	A	E	A	O	O
N	NATOR.	U	L	U	V	S	N	N	I	V	S	H	N	N	N	O	P	R	S	P	N	P	D	
E		S	2	N	D	A	G	S	L	E	I	O	E	U	U	P	T	M	H	T	F	S	F	
N		E	H	T	I			U		R	N	O	R	S	S	E	H	E	O	I	I	O	I	
O		S	S	Y	S			S		L				I	2					C	L	L	L	

16	SER6013	11	3.20	189	4	5	4	27	WL	5	K	10	21	21	A	D	T	D	B	B	G	F		
16	SER6517	10	3.20	189	4	5	4	27	TH	5	K	44	21	31	C	S	M	G	B	B	P	P		
16	SER7214	10	3.20	189	4	5	4	27	TH	5	K	10	63	95	C	S	M	G	B	B	P	P		
16	SER7712	12	3.20	189	4	5	4	27	TH	0		10	21	33	C	S	S	G	B	B	P	P		

3	G	S	F	S	C	C	A	E	H	N	G	R	K	E	N	U	T	S
"	W	W	L	W	E	E	I	C	I	A	E	O	A	A	O	R	R	L
Q	Q	Q	D	Q	N	N	R	O	S	T	O	A	R	S	P	B	A	O
H	U	U	I	U	W	S	P	G	I	H	C	D	S	T	T	Z	Z	P
A	A	A	Q	A	A	E	R	E	T	R	A	S	T	I	H	O	O	E
L	N	N	Q	L	T	W	T	O	E	L	R			N	N	N	N	X
3	1	6	F	Q	T	F	Q	Q			QA	D	F	60	13			Q
3	1	6	F	Q	T	F	Q	Q			OL	D	F	65	17			Q
3	2	6	F	Q	T	F	Q	Q			OL	D	T	72	14			Q
3	2	6	F	Q	T	F	Q	Q			OC	D	T	77	12			Q

In this session PPLPSM was first defined as the product of the two fields PPL2HS and HOUSES and the constant 2.6. Any time PPLPSM is referred to in later commands, the number of people per square mile in a cell will be calculated by multiplying the number of people per household times the number of households in a square meter cell times 2.6.

Next a set is formed consisting of all cells that have a population per square mile between 75 and 100. This set, name CROWDED by the user, is then displayed.

The statement POINTS=0 begins a series of tests that will assign a weight factor to each cell in set CROWDED. In these tests points are added or subtracted for presence or absence of selected attributes.

Now, CROWDED is displayed with a different set of fields and the points that each cell accrued.

Following this output, the user has invoked the SUMSET subroutine to get a total of all the POINTS in set CROWDED, a count of the number of households in the same set.

Finally, a subset named SET189 which consists of all cells in CROWDED that also are located in county number 189. This set is output using the ALL designator so that all fields defined for the data base will be displayed.

ORIGINAL PAGE IS  
OF POOR QUALITY

The following is an example of a session that might be held using the CENSUS data base.

```

ENTER DATA BASE NAME (6 CHAR)
USER: CENSUS
>CTY21=COUNTY .EQ. 21
    SET NAMED CTY21      HAS      14 MEMBERS
>POINTS=0
>IF PNONW .LT. 20. THEN POINTS=POINTS+1
>IF %PNONW .LT. 20 \. THEN POINTS=POINTS+2
>IF %PFEMAL .GT. 50. THEN POINTS=POINTS+5 ELSE POINTS=POINTS-5
>IF %F15249 .GT. 40. THEN POINTS=POINTS+3
>IF FAVINC .GT. 10000 THEN POINTS=POINTS+4
>IF NFAVIN .GT. 10000 THEN POINTS=POINTS+6
>IF %UNEMP .LT. 5. THEN POINTS=POINTS+7
>IF AVSCH\%GSCH .GT. 8 THEN POINTS=POINTS+8 ELSE POINTS=POINTS:
>-5
>IF %UNITS .GT. 10. THEN POINTS=POINTS-5
>IF %NOPLUM .GT. 2. THEN POINTS=POINTS-10
>CNGTAX=TAXR73-TAXR74

```

```

>IF CNGTAX .LT. 0 THEN POINTS=POINTS-10 ELSE POINTS=POINTS;10\:\:
>+10
>OUTPUT(CTY21,COUNTY,ENUDIS,PNONW,%PNONW,%PFEMAL,%F15249,FAVINC,
>NFAVIN,%UNEMP,AVGSCH,%UNITS,%NOPLUM,CNGTAX,POINTS)

```

C	E	P	%	%	%	F	N	%	A	%	%
O	N	N	P	P	F	A	F	U	V	U	N
U	U	O	N	F	1	V	A	N	G	N	O
N	D	N	O	E	5	I	V	E	S	I	P
T	I	W	N	M	2	N	I	M	C	T	L
Y	S		W	A	4	C	N	P	H	S	U
21	0004	0	0.0	47.1	8.9	820	0	8.0	9	12.8	9.7
21	0005	166	24.3	51.5	8.1	658	387	0.0	9	12.6	13.5
21	0006	83	9.9	50.4	9.9	959	755	0.0	9	2.0	1.6
21	0007	0	0.0	48.5	4.7	705	0	3.1	9	3.0	17.2
21	0008	0	0.0	52.6	7.3	801	0	2.0	9	0.0	15.3
21	0009	0	0.0	47.7	4.6	619	0	2.4	9	0.0	20.5
21	0010	16	1.1	50.2	11.5	868	0	3.6	9	1.5	7.0
21	0011	0	0.0	60.5	8.6	478	0	0.0	9	6.3	35.0
21	0012	86	12.2	50.0	3.7	769	418	0.0	10	0.0	14.6
21	0013	21	3.0	44.7	12.1	774	0	6.4	8	0.0	34.5
21	0014	111	7.7	50.8	8.4	750	499	.8	9	0.0	25.7
21	0001	0	0.0	50.2	8.2	762	0	2.8	9	1.6	22.6
21	0002	69	5.4	45.6	3.9	782	190	1.0	9	1.8	34.3
21	0003	63	6.1	54.6	.9	591	1080	2.6	8	2.6	39.2



First, set CTY21 is formed consisting of all cells that are in county number 21. Then, a point system similar to the one described in the land use session is defined. Points are added or subtracted according to factors such as non-white population, female population, income, unemployment figures, and education. One extra feature utilized here is the definition of a variable named CNGTAX which will pick up the fields TAXR73 and TAXR74 and calculate the change in tax rate from 1973 to 1974. Tests are then made on this variable instead of on field values.

Later in the session SUMSET is used to calculate total incomes and family counts. Also, a subset named CCTY21 is formed consisting of cells that are in set CTY21 and are also in enumeration district 1.

## EXAMPLES OF DATA BASE FIELD DEFINITIONS

## Field definitions for the LANSAT data base

FIELD NAME	FLD NO.	1ST BIT	NO. OF BITS	OUTPUT FORMAT	DT TY PE	CD CN VN	IN DX FL AG	FIELD TITLE
DSL	1		10	3I 1				DAYS SINCE LAUNCH
HHMMS	2	10	14	5I 1		13		HOURS MINUTES SECONDS
ORHNJM	3	34	14	1I 5				ORBIT NUMBER
LAT	4	96	24	1A 5	6	14		LATITUDE IN MINUTES
LON	5	72	24	1A 6	6	15		LONGITUDE IN MINUTES
OBVID	6		24	1I 1				OBSERVATION ID,
DATE	7		12	1D 6	13	12		DATE OF OBSERVATION,
CLDCOV	8	64	8	1I 2				CLOUD COVER,
ARCHVAL	9	192	14	1I 5		11		ARCHIVAL ROLL NUMBER,
M1POSN	10	207	9	1I 3				MSS 1 POSITION NUMBER
M2POSN	11	144	9	1I 3				MSS 2 POSITION NUMBER,
M3POSN	12	120	9	1I 3				MSS 3 POSITION NUMBER,
M4POSN	13	24	9	1I 3				MSS 4 POSITION NUMBER,
MFROLL	14	129	15	1I 5				MICROFILM ROLL NUMBER (MSS)
MFPOSN	15	156	12	1I 4				MICROFILM POSITION NUMBER,
REGION	16	56	2	1I 1				BLOCK INDICATOR,
RBVID	17	62	2	1I 1				STATION ID-RBV,
MSSID	18	60	2	1I 1				STATION ID - MSS,
RBMODE	19	58	2	1I 1				TRANSMISSION MODE - RBV,
MSMODE	20	56	2	1I 1				TRANSMISSION MODE - MSS,
R1QUAL	21	168	3	1I 1				QUALITY RBV BAND 1,
R2QUAL	22	171	3	1I 1				QUALITY RBV BAND 2
R3QUAL	23	174	3	1I 1				QUALITY RBV BAND 3,
M1QUAL	24	177	3	1I 1				QUALITY MSS BAND 1
M2QUAL	25	180	3	1I 1				QUALITY MSS BAND 2,
M3QUAL	26	183	3	1I 1				QUALITY MSS BAND 3,
M4QUAL	27	186	3	1I 1				QUALITY MSS BAND 4,
M5QUAL	28	189	3	1I 1				QUALITY MSS BAND 5
SATNUM	29	51	3	1I 1				SATELLITE NUMBER,

ORIGINAL PAGE IS  
OF POOR QUALITY

FIELD DEFINITIONS FOR THE LAND USE DATA BASE

FIELD NAME	FLD NO.	1ST BIT OF	NO. OF BITS	OUTPUT FORMAT	DT TY	CD CN	IN DX	ID FL	FIELD TITLE
					PE	VN		AG	
ZONE NO	1	1	5	1I	2				* UTM ZONE NUMBER,
CELL ID	2	6	56	1A	7	6			* UTM CELL DESIGNATOR,
HOUSES	3	62	7	1I	2				NUMBER OF OCCUPIED HOUSEHOLDS,
PPL2HS	4	69	7	F 5.	2		7		PEOPLE PER HOUSEHOLD,
COUNTY	5	76	8	1I	3				COUNTY CODE,
DEVDIS	6	84	4	1I	1				DEVELOPMENT DISTRICT,
SMSA	7	88	4	1I	1				STANDARD METROPOLITAN STATISTICAL AREA,
CONG	8	92	4	1I	1				TENNESSEE CONGRESSIONAL DISTRICT,
CENSUS	9	96	8	1I	3				CENSUS COUNTY DISTRICT,
SOIL	10	104	16	1A	2	6			SOIL ASSOCIATION GROUP,
RIVER	11	120	5	1I	2				RIVER BASIN CODE,
BASIN	12	125	8	1A	1	6			RIVER BASIN CODE, (TOGETHER WITH RIVER)
SCHOOL	13	133	8	1A	1	6			EDUCATIONAL FACILITY,
OWNER	14	141	6	1I	2				OWNERSHIP, -
LANUS1	15	147	10	1I	3				PRIMARY LAND USE OF CELL,
LANUS2	16	157	10	1I	3				SECONDARY LAND USE,
SLOPE	17	167	8	1A	1	6			SOIL SLOPE,
DEPTH	18	175	8	1A	1	6			DEPTH TO BEDROCK,
PERME	19	183	8	1A	1	6			PERMEABILITY,
AASHY	20	191	8	1A	1	6			ENGINEERING CLASS,
SEPTIC	21	199	8	1A	1	6			SEPTIC TANK LIMITS,
LANFIL	22	207	8	1A	1	6			LANDFILL LIMITS,
TOPSOIL	23	215	8	1A	1	6			TOPSOIL SOURCE,
ROADFIL	24	223	8	1A	1	6			ROADFILL SOURCE,
GWQUAL	25	231	2	1I	1				GROUND WATER QUALITY,
GWQUAN	26	233	2	1I	1				GROUND WATER QUANTITY,
SWQUAN	27	235	3	1I	1				SURFACE WATER QUANTITY,
FLD100	28	238	1	1L	1	4			100 YEAR PROBABILITY OF FLOOD,
SWQUAL	29	239	3	1I	1				SURFACE WATER QUALITY PROBLEMS,
CENWAT	30	242	1	1L	1	4			CENTRAL WATER SYSTEM AVAILABLE,
CENSEW	31	243	1	1L	1	4			CENTRAL SEWER SYSTEM AVAILABLE,
AIRPT	32	244	3	1I	1				AIRPORT,
ECONGEO	33	247	6	1I	2				ECONOMIC GEOLOGY,
HISITE	34	253	8	1A	1	6			HISTORIC SITES PRESENT,
NATURL	35	261	8	1A	1	6			NATURAL AREAS PRESENT,
GEOCAR	36	269	16	1A	2	6			GEOLOGICAL CHARACTERISTICS,
ROADS	37	285	8	1A	1	6			ROAD CHARACTERISTICS..
KARST	38	293	1	1L	1	4			KARST FEATURES PRESENT,
EASTIN	39	30	16	1I	2	6	1		EASTING,
NORTHIN	40	46	16	1I	2	6	1		NORTHING,
URBZON	41	294	8	1A	1	6			URBAN AREA ZONE
TRAZON	42	302	8	1A	1	6			TRANSPORTATION AREA ZONE
SLOP=X	43	310	3	1I	1				SLOPE MULTIPLIER

FIELD DEFINITIONS FOR THE CENSUS DATA BASE

FIELD NAME	FLD NO.	1ST BIT OF BTS	NO. OF BITS	OUTPUT FORMAT	DT TY	CD CN	IN DX	ID FL	AG	FIELD TITLE
POP70	1	1	16	11	5					1970 TOTAL POPULATION
SQMI-E	2	17	10	11	4					NUMBER OF SQUARE MILES
POP04	3	27	16	F 5,	1			7		POPULATION PER SQUARE MILE
PAG04	4	43	16	11	5					POPULATION AGE 0 TO 4 YEARS
PAG04	5	59	10	F 5,	1			7		PERCENT POPULATION AGE 0 TO 4 YEARS
PAG514	6	69	16	11	5					POPULATION AGE 5 TO 14 YEARS
PAG514	7	85	10	F 5,	1			7		PERCENT POPULATION AGE 5 TO 14 YEARS
PAG1524	8	95	16	11	5					POPULATION AGE 15 TO 24 YEARS
PAG1524	9	111	10	F 5,	1			7		PERCENT POPULATION AGE 15 TO 24 YEARS
PAG2534	10	121	16	11	5					POPULATION AGE 25 TO 34 YEARS
PAG2534	11	137	10	F 5,	1			7		PERCENT POPULATION AGE 25 TO 34 YEARS
PAG3544	12	147	16	11	5					POPULATION AGE 35 TO 44 YEARS
PAG3544	13	163	10	F 5,	1			7		PERCENT POPULATION AGE 35 TO 44 YEARS
PAG4554	14	173	16	11	5					POPULATION AGE 45 TO 64 YEARS
PAG4554	15	189	10	F 5,	1			7		PERCENT POPULATION 45 TO 64 YEARS
PAG65JP	16	199	16	11	5					POPULATION AGE 65 YEARS AND OVER
PAG65JP	17	215	10	F 5,	1			7		PERCENT POPULATION 65 YEARS AND OVER
PAGALL	18	225	16	11	5					POPULATION ALL AGES
PWHIE	19	241	16	11	5					WHITE POPULATION
PWHIE	20	257	10	F 5,	1			7		PERCENT WHITE POPULATION
PNONW	21	267	16	11	5					NONWHITE POPULATION
PNONW	22	283	10	F 5,	1			7		PERCENT NONWHITE POPULATION
PMAL	23	293	16	11	5					MALE POPULATION
PMAL	24	309	10	F 5,	1			7		PERCENT MALE POPULATION
PFEM	25	319	16	11	5					FEMALE POPULATION
PFEM	26	335	10	F 5,	1			7		PERCENT FEMALE POPULATION
PFAMLYS	27	345	16	11	5					TOTAL NUMBER OF FAMILIES
PFAMLYS	28	361	16	11	5					MEDIAN FAMILY INCOME
PFPOOR	29	377	16	11	5					FAMILIES BELOW POVERTY LEVEL
PFPOOR	30	393	10	F 5,	1			7		PERCENT FAMILIES BELOW POVERTY LEVEL
PF03999	31	403	16	11	5					FAMILIES WITH INCOME \$0 TO 3999
PF03999	32	419	10	F 5,	1			7		PERCENT FAMILIES WITH INCOME \$0 TO 3999
PF46999	33	429	16	11	5					FAMILIES WITH INCOME \$4000 TO 6999
PF46999	34	445	10	F 5,	1			7		PERCENT FAMILIES WITH INCOME \$4000 TO 6999
PF79999	35	455	16	11	5					FAMILIES WITH INCOME \$7000 TO 9999
PF79999	36	471	10	F 5,	1			7		PERCENT FAMILIES WITH INCOME \$7000 TO 9999
PF10149	37	481	16	11	5					FAMILIES WITH INCOME \$10000 TO 14999
PF10149	38	497	10	F 5,	1			7		PERCENT FAMILIES WITH INCOME \$10000 TO 14999
PF15249	39	507	16	11	5					FAMILIES WITH INCOME \$15000 TO 24999
PF15249	40	523	10	F 5,	1			7		PERCENT FAMILIES WITH INCOME \$15000 TO 24999
PF25000	41	533	16	11	5					FAMILIES WITH INCOME \$25000 AND OVER
PF25000	42	549	10	F 5,	1			7		PERCENT FAMILIES WITH INCOME \$25000 AND OVER
PFAMLYS	43	559	16	11	5					TOTAL NUMBER OF NEGRO FAMILIES
PFAMLYS	44	575	16	11	5					MEDIAN NEGRO FAMILY INCOME
PFPOOR	45	591	16	11	5					NEGRO FAMILIES BELOW POVERTY LEVEL
PFPOOR	46	607	10	F 5,	1			7		PERCENT NEGRO FAMILIES BELOW POVERTY LEVEL
PLABOR	47	617	16	11	5					TOTAL LABOR FORCE
PLABOR	48	633	16	11	5					CIVILIAN LABOR FORCE
PEMP	49	649	16	11	5					EMPLOYMENT NUMBER
PEMP	50	665	16	11	5					NUMBER UNEMPLOYED
PEMP	51	681	10	F 5,	1			7		PERCENT UNEMPLOYED
PEMPAGR	52	691	16	11	5					NUMBER EMPLOYED IN AGRICULTURE
PEMPAGR	53	707	10	F 5,	1			7		PERCENT EMPLOYED IN AGRICULTURE
PEMPMAN	54	717	16	11	5					NUMBER EMPLOYED IN MANUFACTURING
PEMPMAN	55	733	10	F 5,	1			7		PERCENT EMPLOYED IN MANUFACTURING
PEMPNMN	56	743	16	11	5					NUMBER EMPLOYED IN NONMANUFACTURING

FIELD NAME	FLD NO.	1ST BIT OF	NO. OF BITS	OUTPUT FORMAT	DT TY	CD CN	IN DX	ID FL
					PE	VN		AG
EMPNUM	57	759	10	F 5, 1			7	
LABOR	58	769	16	11 5				
LABOR	59	785	16	11 5				
EMP	60	801	16	11 5				
EMP	61	817	16	11 5				
UNEMP	62	833	10	F 5, 1			7	
AGFFIM	63	843	16	11 5				
AGFFIM	64	859	10	F 5, 1			7	
MANFCT	65	869	16	11 5				
MANFCT	66	885	10	F 5, 1			7	
CONSTR	67	895	16	11 5				
CONSTR	68	911	10	F 5, 1			7	
TRCMUT	69	921	16	11 5				
TRCMUT	70	937	10	F 5, 1			7	
TRADE	71	947	16	11 5				
TRADE	72	963	10	F 5, 1			7	
MAI-1	73			11 1 15				
MAI-2	74			11 1 15				
FINRE	75	973	16	11 5				
FINRE	76	989	10	F 5, 1			7	
SERVIC	77	999	16	11 5				
SERVIC	78	1015	10	F 5, 1			7	
EDUCAT	79	1025	16	11 5				
EDUCAT	80	1041	10	F 5, 1			7	
PBADM	81	1051	16	11 5				
PBADM	82	1067	10	F 5, 1			7	
PA25JP	83	1077	16	11 5				
AVGSCH	84	1093	5	11 2				
NOYRS	85	1098	16	11 5				
NOYRS	86	1114	10	F 5, 1			7	
YRS17	87	1124	16	11 5				
YRS17	88	1140	10	F 5, 1			7	
YRS8	89	1150	16	11 5				
YRS8	90	1166	10	F 5, 1			7	
YRS911	91	1176	16	11 5				
YRS911	92	1192	10	F 5, 1			7	
YRS12	93	1202	16	11 5				
YRS12	94	1218	10	F 5, 1			7	
YRS1315	95	1228	16	11 5				
YRS1315	96	1244	10	F 5, 1			7	
YRS15	97	1254	16	11 5				
YRS16	98	1270	10	F 5, 1			7	
PA25JP	99	1280	16	11 5				
AVGSCH	100	1296	5	11 2				
NOYRS	101	1301	16	11 5				
NOYRS	102	1317	10	F 5, 1			7	
HOUSES	103	1327	14	11 4				
UNIT1	104	1341	14	11 4				
UNIT1	105	1355	10	F 5, 1			7	
UNITS	106	1365	14	11 4				
UNITS	107	1379	10	F 5, 1			7	
MOBILE	108	1389	14	11 4				
MOBILE	109	1403	10	F 5, 1			7	
OCCUPD	110	1413	14	11 4				
OCCUPD	111	1427	14	11 4				
OCCUPD	112	1441	10	F 5, 1			7	

FIELD TITLE

ORIGINAL PAGE IS OF POOR QUALITY

PERCENT EMPLOYED IN NONMANUFACTURING  
TOTAL NEGRO LABOR FORCE  
NEGRO CIVILIAN LABOR FORCE  
NEGRO EMPLOYMENT NUMBER  
NEGRO NUMBER UNEMPLOYED  
PERCENT NEGRO UNEMPLOYED  
NO. EMPLOYED BY AGRICULTURE, FORESTRY, FISHER  
PERCENT EMPLOYED BY AGRICULTURE, FORESTRY, FI  
NUMBER EMPLOYED BY MANUFACTURING  
PERCENT EMPLOYED BY MANUFACTURING  
NUMBER EMPLOYED BY CONSTRUCTION  
PERCENT EMPLOYED BY CONSTRUCTION  
NUMBER EMPLOYED BY TRANSPORTATION, COMMUNICATI  
PERCENT EMPLOYED BY TRANSPORTATION, COMMUNICA  
NUMBER EMPLOYED BY TRADE  
PERCENT EMPLOYED BY TRADE  
THIS FIELD NUMBER IS AVAILABLE FOR A NEW FIEL  
THIS FIELD IS AVAILABLE FOR A NEW FIELD  
NUMBER EMPLOYED BY FINANCE, INSURANCE, REAL E  
PERCENT EMPLOYED BY FINANCE, INSURANCE, REAL  
NUMBER EMPLOYED BY SERVICE INDUSTRIES  
PERCENT EMPLOYED BY SERVICE INDUSTRIES  
NUMBER EMPLOYED BY EDUCATION  
PERCENT EMPLOYED BY EDUCATION  
NUMBER EMPLOYED BY PUBLIC ADMINISTRATION  
PERCENT EMPLOYED BY PUBLIC ADMINISTRATION  
POPULATION 25 YEARS AND OVER  
MEDIAN SCHOOL YEARS COMPLETED  
NO SCHOOL YEARS COMPLETED  
PERCENT NO SCHOOL YEARS COMPLETED  
1 TO 7 YEARS OF SCHOOL COMPLETED  
PERCENT 1 TO 7 YEARS OF SCHOOL COMPLETED  
8 YEARS OF SCHOOL COMPLETED  
PERCENT 8 YEARS OF SCHOOL COMPLETED  
HIGH SCHOOL, 1 TO 3 YEARS COMPLETED  
PERCENT HIGH SCHOOL, 1 TO 3 YEARS COMPLETED  
HIGH SCHOOL, 4 YEARS COMPLETED  
PERCENT HIGH SCHOOL, 4 YEARS COMPLETED  
COLLEGE, 1 TO 3 YEARS COMPLETED  
PERCENT COLLEGE, 1 TO 3 YEARS COMPLETED  
COLLEGE, 4 YEARS OR MORE COMPLETED  
PERCENT COLLEGE, 4 YEARS OR MORE COMPLETED  
NEGRO POPULATION 25 YEARS AND OVER  
NEGRO MEDIAN SCHOOL YEARS COMPLETED  
NEGRO NO SCHOOL YEARS COMPLETED  
PERCENT NEGRO NO SCHOOL YEARS COMPLETED  
TOTAL HOUSING UNITS  
ONE UNIT STRUCTURES  
PERCENT ONE UNIT STRUCTURES  
TWO UNITS OR MORE  
PERCENT TWO UNITS OR MORE  
MOBILE HOMES  
PERCENT MOBILE HOMES  
TOTAL OCCUPIED UNITS  
OWNER OCCUPIED UNITS  
PERCENT OWNER OCCUPIED UNITS

FIELD NAME	FLD NO.	1ST NO. BIT OF	OUTPUT OF	DT	CD	IN	ID	FIELD TITLE
		BTS	FORMAT	TY	PE	VN	DX FL AG	
RENOC	113	1451 14	11 4					RENTER OCCUPIED UNITS
RENOC	114	1465 10	F 5, 1			7		PERCENT RENTER OCCUPIED UNITS
UNPLUM	115	1475 14	11 4					UNITS LACKING PLUMBING FACILITIES
UNPLUM	116	1489 10	F 5, 1			7		PERCENT UNITS LACKING PLUMBING FACILITIES
OCROWD	117	1499 14	11 4					OVERCROWDED UNITS
OCROWD	118	1513 10	F 5, 1			7		PERCENT OVERCROWDED UNITS
HGVAL	119	1523 16	11 5					MEDIAN HOUSE VALUE
HVRNT	120	1539 10	11 4					MEDIAN CONTRACT RENT
NPLUM	121	1549 14	11 4					NEGRO UNITS LACKING PLUMBING FACILITIES
NPLUM	122	1563 10	F 5, 1			7		PERCENT NEGRO UNITS LACKING PLUMBING FACILITIES
OCROWD	123	1573 14	11 4					NEGRO OVERCROWDED UNITS
OCROWD	124	1587 10	F 5, 1			7		PERCENT NEGRO OVERCROWDED UNITS
HVVAL	125	1597 16	11 5					NEGRO MEDIAN HOUSE VALUE
HVRNT	126	1613 10	11 4					NEGRO MEDIAN CONTRACT RENT
SOILA	127	1623 24	1A 3			6		MAJOR SOIL ASSOCIATIONS CAT, A
SOILA	128	1647 10	F 5, 1			7		PERCENT SOIL CATEGORY A
SOILB	129	1657 24	1A 3			6		MAJOR SOIL ASSOCIATIONS CAT, B
SOILB	130	1681 10	F 5, 1			7		PERCENT SOIL CATEGORY B
SOILC	131	1691 24	1A 3			6		MAJOR SOIL ASSOCIATIONS CAT, C
SOILC	132	1715 10	F 5, 1			7		PERCENT SOIL CATEGORY C
STEEP	133	1725 10	F 5, 1			7		PERCENT OF LAND WITH EXCESSIVE SLOPE
LANDJ1	134	1735 10	11 3					MAJOR LAND USES CAT, 1
LANDU1	135	1745 10	F 5, 1			7		PERCENT LANDUSE CATEGORY 1
LANDJ2	136	1755 10	11 3					MAJOR LAND USE CAT, 2
LANDU2	137	1765 10	F 5, 1			7		PERCENT LAND USE CATEGORY 2
LANDJ3	138	1775 10	11 3					MAJOR LAND USE CAT, 3
LANDU3	139	1785 10	F 5, 1			7		PERCENT LAND USE CATEGORY 3
POP70	140	1795 16	11 5					1970 COUNTY POPULATION
POP60	141	1811 16	11 5					1960 COUNTY POPULATION
CHANGE	142	1827 16	11 6			5		COUNTY POPULATION CHANGE SINCE 1960
CHANGE	143	1843 11	F 6, 1			5 7		PERCENT COUNTY POPULATION CHANGE SINCE 1960
NATCNG	144	1854 16	11 6			5		NATURAL POPULATION CHANGES
MIGRAT	145	1870 16	11 6			5		MIGRATION POPULATION CHANGES
POP73	146	1886 10	11 7			10		ESTIMATED 1973 COUNTY POPULATION
POP80	147	1896 10	11 7			10		ESTIMATED 1980 COUNTY POPULATION
POP90	148	1906 10	11 7			10		ESTIMATED 1990 COUNTY POPULATION
LABOR	149	1916 16	11 6			10		COUNTY TOTAL CIVILIAN WORK FORCE
UNEMP	150	1932 16	11 5					COUNTY UNEMPLOYMENT
UNEMP	151	1948 10	F 5, 1			7		PERCENT COUNTY UNEMPLOYMENT
EMPAN	152	1958 16	11 5					COUNTY MANUFACTURING EMPLOYMENT
EMPAN	153	1974 10	F 5, 1			7		PERCENT COUNTY MANUFACTURING EMPLOYMENT
EMPAN	154	1984 16	11 6			10		COUNTY NONMANUFACTURING EMPLOYMENT
EMPAN	155	2000 10	F 5, 1			7		PERCENT COUNTY NONMANUFACTURING EMPLOYMENT
SELFEMP	156	2010 16	11 5					COUNTY SELF-EMPLOYED
SELFEMP	157	2026 10	F 5, 1			7		PERCENT COUNTY SELF-EMPLOYED
FARM	158	2036 16	11 5					COUNTY FARM EMPLOYMENT
FARM	159	2052 10	F 5, 1			7		PERCENT COUNTY FARM EMPLOYMENT
FARMLND	160	2062 10	F 5, 1			7		PERCENT OF TOTAL LAND IN FARM ACERAGE
FARMS	161	2072 12	11 4					TOTAL NUMBER OF FARMS
AVSIZE	162	2084 12	F 5, 1			7		AVERAGE FARM SIZE IN ACRES
VALPRO	163	2096 10	11 5					TOTAL MARKET VALUE OF AGR., PRO. SOLD/10000
VALACR	164	2106 14	F 5, 2			6		AVERAGE MARKET VALUE PER ACRE
COMMIN	165	2120 15	11 5					COUNTY COMMUTER POPULATION GAIN
COMOUT	166	2135 15	11 5					COUNTY COMMUTER POPULATION LOSS
COMNET	167	2150 16	11 6			5		COUNTY COMMUTER POPULATION NET GAIN OR LOSS
COMNET	168	2166 10	11 3					MAJOR COUNTY COMPOSING NET NO. 1

FIELD NAME	FLD NO.	1ST BIT OF BTS	NO. OF	OUTPUT FORMAT	DT TY	CD CN	IN DX	ID FL	FIELD TITLE
					PE	VN		AG	
CNTY1	169	2176	15		11		5		NUMBER COMMUTING COUNTY 1
TY2	170	2191	10		11		3		MAJOR COUNTY COMPOSING NET NO. 2
CNTY2	171	2201	15		11		5		NUMBER COMMUTING COUNTY 2
CNTY3	172	2216	10		11		3		MAJOR COUNTY COMPOSING NET NO. 3
CNTY3	173	2226	15		11		5		NUMBER COMMUTING COUNTY 3
TY4	174	2241	10		11		3		MAJOR COUNTY COMPOSING NET NO. 4
CNTY4	175	2251	15		11		5		NUMBER COMMUTING COUNTY 4
XR73	176	2266	10	F	5		2	6	1973 COUNTY TAX RATE
SVAL	177	2276	14		11		5		TOTAL ASSESSED COUNTY PROPERTY VALUE/1000000
ACTVAL	178	2290	14		11		5		ESTIMATED ACTUAL COUNTY PROPERTY VALUE/1000000
SCHOOL	179	2304	6		11		2		COUNTY ELEMENTARY AND SECONDARY SCHOOLS
HIGH	180	2310	6		11		2		COUNTY HIGH SCHOOLS
LA	181	2316	16		11		5		COUNTY AVERAGE DAILY ATTENDANCE
EXPJP	182	2332	10		11		4		COUNTY EXPENDITURES PER PUPIL IN A.D.A.
UDIS	183	2342	30		1A	5	6	3	* ENUMERATION DISTRICT
TY	184	2372	14		11		4		CITY
CDIV	185	2386	10		11		3		COUNTY CENSUS DIVISION
COUNTY	186	2396	10		11		3		* COUNTY
XR74	187	2406	10	F	5		2	6	1974 COUNTY TAX RATE

**ORIGINAL PAGE IS  
OF POOR QUALITY**

## HEADER, NAMELIST, AND FORMAT EXAMPLES

The user defined header, namelist, and format for LANSAT may be listed as follows:

>HEADER  
ENTER OPERATION TYPE. (ADD, DELETE, EXIT, LIST, MODIFY, OR REPLACE)

L  
ENTER HEADER TITLE:  
HEDMER

HEDMER 203  
(31H OBSR/'N-ID ORB ROLL/POS 1-4,10K,'LAT' LON: CC MFR '

'/POSN MODAYR')  
ENTER OPERATION TYPE. (ADD, DELETE, EXIT, LIST, MODIFY, OR REPLACE)

E  
>NAMELIST  
ENTER OPERATION TYPE. (ADD, DELETE, EXIT, LIST, MODIFY, OR REPLACE)

L  
ENTER NAMELIST TITLE:  
IDSMER

IDSMER 204  
SAT,DSL,HHMS,ORBNUM,ARCHVAL,11POSN,12POSN,13POSN,14POSN,LAT,LON, CC,MFR  
L,15POSN,DATE

ENTER OPERATION TYPE. (ADD, DELETE, EXIT, LIST, MODIFY, OR REPLACE)

E  
>FORMAT  
ENTER OPERATION TYPE. (ADD, DELETE, EXIT, LIST, MODIFY, OR REPLACE)

L  
ENTER FORMAT TITLE:  
FOR1ER

FOR1ER 202  
(1X,11,311,'-'511,215,'/',13,3(','13),1K,5A1,1K,6A1,13,16,'/'14,312)

ENTER OPERATION TYPE. (ADD, DELETE, EXIT, LIST, MODIFY, OR REPLACE)

E

An example of saved text in LANSAT is:

```
> SAVETEXT
  ENTER OPERATION TYPE. (ADD, DELETE, EXIT, LIST, MODIFY, OR REPLACE)

L
  ENTER SAVETEXT TITLE:
LIST
LIST      205
:
: THE SET TO BE OUTPUT SHOULD BE NAMED 'SETOUT'

OUTPUT(SETOUT, IDSMER, HEDMER, FORMER)
  ENTER OPERATION TYPE. (ADD, DELETE, EXIT, LIST, MODIFY, OR REPLACE)
```

ORIGINAL PAGE IS  
OF POOR QUALITY

Text may appear for explanatory purposes as in the land use data base

```
: THIS IS A ROUTINE THAT SEARCHES THE
: DATA BASE FOR CELLS IN A GIVEN
: COUNTY THAT HAVE A POPULATION PER
: SQUARE MILE WITHIN A GIVEN RANGE.
:
: TYPE THE FOLLOWING
: CNUM= XXX (XXX IS THE COUNTY NUMBER)
: LOW= YYY (YYY IS THE LOW POPULATION BOUND)
: UP=ZZZ (ZZZ IS THE HIGH POPULATION BOUND)
: INSERT(POPSRCH)
```

Note that this text refers to more saved text named POPSRCH which will form the required set and output it as follows:

```
PPLPSM = PPL2HS*HOUSES*2.59
SETXXX = COUNTY .EQ. CNUM .AND. :
        (PPLPSM FROM LOW THRU UP)
OUTPUT(SETXXX, ALL)
```

## EXAMPLES OF CREATE INPUT INFORMATION

Example of OPCARD file for LANSAT data base update.

DSL=E1  
HMMMS=E2  
ORBNUM=E4  
LAT=E12  
LON=E13  
CC=E11  
ARCHVAL=E25  
M1POSN=E27  
M2POSN=E15  
M3POSN=E14  
M4POSN=E3  
M7ROLL=E15  
M7POSN=E17  
REGION=E6  
RBVID=E10  
MSSID=E9  
RMODE=E8  
MSMODE=E7  
R1QUAL=E18  
R2QUAL=E19  
R3QUAL=E20  
M1QUAL=E21  
M2QUAL=E22  
M3QUAL=E23  
M4QUAL=E24  
M5QUAL=E25  
SAT=E5  
SAT23=E30  
I1=E1 + 913  
I2=E1 + 1000  
I3=I2 + 913  
I4=0  
I5=1 .CV. E1  
I6=2 .CV. I1  
I7=5 .CV. I2  
I8=6 .CV. I3  
I9=E5 .SH. I4 \* 5  
I10=I9\*2  
I11=I10-1  
PHNUM=E30+I11



Example of OPCARD file for a CENSUS data base update.

COJNY=E1  
CCDIV=E2  
ENJDIS=E3 ,CH, 5  
CITY=E4  
I1=E5 + E6  
ROP70=I1 + E7  
I2=E8 + E9  
I3= E15 + E16  
PAG04=I2 + I3  
VPAG04=F6 ,V, F5  
I4=E10 ,S+, 5  
I5=E17 ,S+, 5  
PAG514=I4 +I5  
VPAG514=F8 ,V, F5  
RA25UP=E43 ,S+, 16  
I6= E22 + E29  
W6YRS=I6 +E36  
I7=E23 + E30  
YRS17=I7 + E37  
I8=E24 + E31  
YRS8=I8 +E38  
I9= E25 + E32  
YRS911=I9 + E39  
I10=E26 + E33  
YRS12=I10 + E40  
I11=E27 +E34  
YRS1315=I11 +E41  
I12=E28 +E35  
YRS16=I12+E42  
I13=0  
I14=4  
I15=8  
I16=10  
I17=12  
I18=14  
I19=16  
I20=F11 ,S2, I13 \* 7  
I21=F11 ,S+, 7  
AVGSCH=I20 / I21

**APPENDIX B**  
**PROGRAM SPECIFICATIONS**

**PRECEDING PAGE BLANK NOT FILMED**

**ORIGINAL PAGE IS  
OF POOR QUALITY**

Appendix B includes summary documentation for the following programs:

<u>Name</u>	<u>Page</u>	<u>Name</u>	<u>Page</u>	<u>Name</u>	<u>Page</u>
ASCII	B-1	GENDEX	B-48	PROCSS	B-96
BINDEX	B-2	GETNXT	B-49	QDFQRM	B-97
BISORT	B-3	GETOPR	B-50	RDFLIN	B-100
BNSRCH	B-5	GETSMB	B-51	RDULIN	B-101
BSERCH	B-6	HITES	B-52	READFL	B-102
BUILD	B-7	IBDEP	B-53	READLR	B-103
CALSUB	B-8	IBEXT	B-54	REDWRT	B-104
CET08	B-9	IBLANK	B-55	RESET	B-105
CHKFLD	B-10	INCORDE	B-57	RFIELD	B-106
CHKSYM	B-11	INDEX	B-58	REVSCN	B-107
CHKTYP	B-13	ININT	B-60	RVSCAN	B-108
CHRTRN	B-14	INITAL	B-61	SCAN	B-109
COMPUT	B-15	INSERT	B-63	SCANTB	B-112
CONTRL	B-16	INTERP	B-64	SCIFID	B-113
COORD	B-17	INTGER	B-65	SCINDX	B-114
COORDN	B-19	IXTR	B-66	SEARCH	B-115
CREATE	B-20	KSCAN	B-67	SETPRI	B-116
CRREWL	B-22	KUMPAR	B-68	SORTIX	B-117
CVREAL	B-23	LACTUL	B-142	SRTMER	B-118
C6T08	B-24	LEXLAL	B-69	STELM	B-119
C8T06	B-25	LININ	B-70	STNAME	B-121
DBLOAD	B-26	LINMES	B-71	STOFLD	B-122
DELBLK	B-27	LSERCH	B-72	SUBSET	B-123
EXCANS	B-28	MAKSEN	B-72	SUMARY	B-124
EXFLD	B-29	MERGER	B-74	SUMSET	B-125
EXOPRN	B-30	MESH	B-76	SUM2	B-126
EXQUAD	B-31	MESMAN	B-77	SYNTHE	B-127
EXSETD	B-32	MESOUT	B-79	SYNTH1	B-128
EXSTMT	B-34	MINDEX	B-80	TERFNC	B-129
EXTFLD	B-35	MOVOUT	B-82	TERMNL	B-130
EXVARD	B-36	NAMLST	B-84	UNPKER	B-131
FILDEL	B-37	NORM	B-86	VARCHK	B-132
FILEND	B-38	OUTPUT	B-87	VARINX	B-134
FILRED	B-39	PACDAT	B-89	VARVAL	B-135
FILWRT	B-40	PACKER	B-90	WRITE	B-136
FINFLD	B-41	PARSG	B-91	WRITFL	B-102
FLDFIN	B-42	PARVOC	B-92	WRIBUF	B-137
FLTPT	B-43	POINT	B-93	WRIDAT	B-138
FMATS	B-45	PRCENT	B-94	WRTDEG	B-139
GENATE	B-46	PRNTER	B-95	XFER	B-141

# MINIS PROGRAM SPECIFICATION

1.0 PROGRAM NAME- ASCII

1.1 PURPOSE

TO CONVERT INTEGER DIGITS INTO ASCII AND STORE THEM IN THE FIRST BYTE OF EACH WORD OF THE OUTPUT ARRAY.

1.2 IDENTIFICATION

DATE- 8-12-75

REVISION-

AUTHOR- C. J. FARLESS (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 93 LOCATIONS (135 OCTAL)

1.3 DESCRIPTION

AN INTEGER IS CONVERTED TO ASCII AND EACH DIGIT IS STORED IN THE FIRST BYTE OF EACH WORD OF THE GIVEN OUTPUT ARRAY.

1.4 USAGE

CALLING SEQUENCE:

CALL ASCII(INT,IOUT,NCHAR)

INT = INTEGER TO CONVERT

IOUT = OUTPUT ARRAY

NCHAR = NUMBER OF CHARACTERS IN OUTPUT ARRAY

ORIGINAL PAGE IS  
OF POOR QUALITY

## MINIS PROGRAM SPECIFICATION

2.0 PROGRAM NAME- BINDEX

2.1 PURPOSE

TO BUILD THE INDEX TO THE LATLON FILE (LINDEX).

2.2 IDENTIFICATION

DATE- 22 SEPT 1973

REVISION- 31 OCT 73 FORMER MERITS PROGRAM

AUTHOR- GILLIS (205) 453-5230

COMPUTER- DATACRAFT 6024

LANGUAGE- FORTRAN

CORE SIZE- 512 LOCATIONS (1000 OCTAL)

2.3 DESCRIPTION

THE LATLON FILE IS SEARCHED FOR CHANGES IN LATITUDE. THE 'LAT' ELEMENT OF ARRAY 'LINDEX' IS SET TO THE RECORD NUMBER WHERE THE LATITUDE CHANGES. A ZERO PHOTO ID NUMBER SIGNALS THE END OF THE LATLON FILE AND LINDEX(180) IS SET TO THE NUMBER OF THE LAST PHOTO PLUS ONE.

2.4 USAGE

CALLING SEQUENCE

CALL BINDEX

THE INDEX IS BUILT AND WRITTEN ON DISK IN FILE UINDEX, RECORD 0,

## MINIS PROGRAM SPECIFICATION

3.0 PROGRAM NAME- BISORT

3.1 PURPOSE

PERFORM A BINARY SORT OF 'N' ITEMS, OF 'M' WORDS EACH,

3.2 IDENTIFICATION

DATE- 25 MARCH, 1975

REVISION-

AUTHOR- R.L.KEFFER (205)-883-1778

COMPUTER- DATACRAFT 6024

LANGUAGE- BASIC ASSEMBLER

CORE SIZE- 289 LOCATIONS (441 OCTAL)

ORIGINAL PAGE IS  
OF POOR QUALITY

3.3 DESCRIPTION

THE VALUE OF THE FIELD FROM BIT 'KEY1' TO BIT 'KEY2' IS CALLED THE KEY FIELD. THE MINIMUM KEY FIELD IS DETERMINED AND ALL KEYS ARE BIASED BY THAT AMOUNT. A BINARY SORT OF ALL ITEMS IS PERFORMED FOR EACH BIT FROM KEY2 TO KEY1 WITH THE DATA ALTERNATELY SORTED BETWEEN THE EQUAL SIZE BUFFERS, 'BUF1' AND 'BUF2'. WHEN SORTING IS COMPLETE, THE MINIMUM KEY FIELD IS ADDED TO EACH KEY. THE SORTED ARRAY IS RETURNED IN 'BUF1' AND 'BUF2' IS DESTROYED. EXECUTION TIME IS ROUGHLY PROPORTIONAL TO  $N*(K+2)$ , WHERE  $2**K$  IS GREATER THAN THE RANGE OF KEY FIELDS BUT LESS THAN TWICE THE RANGE OF KEY FIELDS. (EG. FOR RANGE OF 513,  $K = 10$ )

3.4 USAGE

CALLING SEQUENCE:

CALL BISORT(KEY1,KEY2,NITEM,NWDS,BUF1,BUF2)

KEY1 = FIRST BIT POSITION OF SORT KEY.

KEY2 = LAST BIT POSITION OF SORT KEY.

0.LE.(K2-K1).LE.22 BITS NUMBERED FROM 0 TO  
(NWDS\*24-1) LEFT TO RIGHT IN EACH ITEM.  
SORT KEYS MAY CROSS WORD BOUNDARIES.

NITEM = NUMBER OF ITEMS TO SORT.

NWDS = NUMBER OF WORDS PER ITEM,

BUF1 = BUFFER CONTAINING ARRAY TO BE SORTED,

BUF2 = SCRATCH BUFFER,

## MINIS PROGRAM SPECIFICATION

4.0 PROGRAM NAME- BNSRCH

4.1 PURPOSE

TO PERFORM A BINARY SEARCH FOR A FIELD VALUE IN AN OLDMAN DATA BASE.

4.2 IDENTIFICATION

DATE- 1 AUGUST 1975

REVISION-

AUTHOR- D.R.SANDERS

COMPUTER- DATACRAFT 6024

LANGUAGE- FORTRAN IV

CORE SIZE- 152 LOCATIONS (230 OCTAL)

ORIGINAL PAGE IS  
OF POOR QUALITY

4.3 DESCRIPTION

BNSRCH COMPARES THE REQUESTED FIELD VALUE TO THOSE IN THE DATA BASE VIA A BINARY SEARCH. THEN, ACCORDING TO WHETHER THE VALUE SOUGHT IS A LOWER OR UPPER LIMIT, THE FIRST OR LAST RECORD, RESPECTIVELY, CONTAINING THE VALUE IS FOUND.

4.4 USAGE

CALLING SEQUENCE:

CALL BNSRCH(IFIELD,IPTR,VALUE)

IFIELD = FIELD NUMBER

IPTR = (INPUT)

-1 FIND LOWER LIMIT

+1 FIND UPPER LIMIT

(OUTPUT)

= POINTER TO D.B. RECORD (-1 IF NOT FOUND)

VALUE = FIELD VALUE SOUGHT

## MINIS PROGRAM SPECIFICATION

5.0 PROGRAM NAME- BSERCH

5.1 PURPOSE

SCAN LATITUDE-LONGITUDE INDEX FOR PHOTOS POSSIBLY WITHIN THE SPECIFIED AREA.

5.2 IDENTIFICATION

DATE- 17 SEPTEMBER, 1975

REVISION- ADAPTED FROM MERITS SUB OF SAME NAME.

AUTHOR- R.L.KEFFER (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 2448 LOCATIONS (370 OCTAL)

5.3 DESCRIPTION

THE INDEX ENTRIES BETWEEN THE SPECIFIED LATITUDE LIMITS ARE EXAMINED FOR LONGITUDES WITHIN THE SPECIFIED RANGE, THE DATA BASE POINTERS AND THE COORDINATES OF ALL CANDIDATES WITHIN THE BAND ARE SAVED IN COMMON 'REUSE'.

5.4 USAGE

CALLING SEQUENCE:

CALL BSERCH(NPOS,LAL,LAH,LOL,LOH,LINDEX)

NPOS = NUMBER OF PHOTOS IN ARRAY.

LAL,LAH,LOL,LOH = LATITUDE, LONGITUDE LIMITS OF AREA,

LINDEX = INDEX OF LATITUDES OF INDEX FILE.

## MINIS PROGRAM SPECIFICATION

6.0 PROGRAM NAME- BUILD

6.1 PURPOSE

BUILD AN EXECUTION STACK IN COMMON AREA QUATBL.

6.2 IDENTIFICATION

DATE- 27 NOVEMBER 1974

REVISION-

AUTHOR- D.R.SANDERS

COMPUTER- DCR 6024

LANGUAGE- FORTRAN IV

CORE SIZE- 122 WORDS (172 OCTAL)

**ORIGINAL PAGE IS  
OF POOR QUALITY**

6.3 DESCRIPTION

THIS ROUTINE CONVERTS HP CODES USED IN THE SYNTAX GRAPH TO OP CODES DEFINED FOR THE QUAD TABLE. IT THEN STORES THIS OP CODE ALONG WITH TWO OPERAND POINTERS IN THE QUATBL COMMON AREA.

6.4 USAGE

CALLING SEQUENCE:

CALL BUILD(IOP,IOD1,IOD2,IER)

IOP = OPERAND FROM SYNTAX GRAPH

IOD1 = OPERAND ONE

IOD2 = OPERAND TWO

IER = ERROR FLAG: 0=GOOD; 1=ERROR

## MINIS PROGRAM SPECIFICATION

7.0 PROGRAM NAME- CALSUR

7.1 PURPOSE

EXECUTE THE SUBROUTINE REFERENCED BY THE QUAD TABLE POINTER.

7.2 IDENTIFICATION

DATE- 24 JANUARY 1975

REVISION-

AUTHOR- D.R.SANDERS

COMPUTER- DCR 6024

LANGUAGE- FORTRAN IV

CORE SIZE- 133 LOCATIONS (205 OCTAL)

7.3 DESCRIPTION

THE FIRST OPERAND IN THE QUAD TABLE AT THE REFERENCED LOCATION PLUS ONE IS USED AS THE INDEX TO A PREDEFINED LIST OF SUBROUTINE CALLS. ALL ARGUMENTS ARE TRANSFERRED TO THE SUBROUTINES VIA COMMON.

7.4 USAGE

CALLING SEQUENCE:

CALL CALSUB(ISYM,IQ1,IQ2)

ISYM = SYMBOL TABLE ENTRY FOR CALLED SUBROUTINE  
IQ1 = ARGUMENTS TO BE TRANSFERRED  
IQ2 TO THE ARGUMENT COMMON AREA.  
IARG(2) AND IARG(3) RESPECTIVELY.

## MINIS PROGRAM SPECIFICATION

8.0 PROGRAM NAME- CET08

8.1 PURPOSE

CONVERT AN EBCDIC STRING TO AN ASCII STRING

8.2 IDENTIFICATION

DATE- 2-11-76

REVISION-

AUTHOR- C.J.FARLESS (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- LOCATIONS (144 OCTAL)

ORIGINAL PAGE IS  
OF POOR QUALITY

8.3 DESCRIPTION

ANY EBCDIC CHARACTER CODE GREATER THAN 249 OR LESS THAN 64 IS CONVERTED TO A QUESTION MARK. CODES 193-249 ARE CONVERTED TO 64-121. THEN 63 IS SUBTRACTED FROM ALL CODES LEAVING CODES 1-58. THE CORRESPONDING HOST MACHINE CHARACTER IS THEN EXTRACTED FROM THE CHARACTER ARRAY AND STORED IN THE STRING.

8.4 USAGE

CALLING SEQUENCE:

CALL CET08(ISTR,NCHAR,IST,IOUT)

ISTR = INPUT CHARACTER STRING,  
NCHAR = NO. OF CHARACTERS TO CONVERT,  
IST = FIRST CHARACTER IN STRING TO CONVERT,  
IOUT = OUTPUT CHARACTER STRING.

## MINIS PROGRAM SPECIFICATION

9.0 PROGRAM NAME- CHKFLD

9.1 PURPOSE

SEARCH THE COMMON AREA, /FIELDS/, FOR A FIELD NAME, AND MAKE A SYMBOL TABLE ENTRY FOR A FIELD THAT IS FOUND.

9.2 IDENTIFICATION

DATE- 29 JANUARY 1975

REVISION-

AUTHOR- D.R. SANDERS

COMPUTER- DCR 6024

LANGUAGE- FORTRAN IV

CORE SIZE- 113 LOCATIONS (161 OCTAL)

9.3 DESCRIPTION

THE FIELD NAME COUNT IS TAKEN FROM CELL 14 OF COMMON AREA /FILDEF/ AND THAT MANY ENTRIES OF /FIELDS/ MAY BE SEARCHED FOR A MATCH WITH THE CHARACTER STRING IN QUESTION. IF A MATCH IS FOUND A FLAG IS SET AND THE FIELD NAME, TYPE CODE FOR A FIELD AND A POINTER TO THE FIELD DEFINITION ARRAY ARE ENTERED IN THE SYMBOL TABLE. IF NO MATCH IS FOUND, CONTROL IS RETURNED TO 'CHKSYM'.

9.4 USAGE

CALLING SEQUENCE:

CALL CHKFLD(ICHARY,NCH,IENTRY,IER)

ICHARY = ARRAY CONTAINING CHARACTER STRING  
NCH = NUMBER OF CHARACTERS IN 'ICHARY'  
IENTRY = NEGATIVE POINTER TO SYMBOL TABLE OR  
ZERO IF NO MATCH FOUND,  
IER = ERROR FLAG

## MINIS PROGRAM SPECIFICATION

10.0 PROGRAM NAME- CHKSYM

### 10.1 PURPOSE

CHECK THE CURRENT SYMBOL PASSED FROM SUBROUTINE LEXCAL TO SEE IF IT IS EITHER IN THE SYMBOL OR RESERVED WORD LIST, IF NOT IN EITHER, IT IS ENTERED IN THE SYMBOL TABLE,

### 10.2 IDENTIFICATION

DATE- 11 NOVEMBER 1974

REVISION-

AUTHOR- D.R.SANDERS (205)-883-1778

COMPUTER- DCR 6024

LANGUAGE- FORTRAN

CORE SIZE- 567 LOCATIONS (1067 OCTAL)

ORIGINAL PAGE IS  
OF POOR QUALITY

### 10.3 DESCRIPTION

SUBROUTINE CHKSYM RECEIVES 'ITYPE' FROM LEXCAL WHICH INDICATES NUMERIC, ALPHABETIC OR SPECIAL CHARACTER STRING, IF NUMERIC, THE STRING IS CONVERTED TO INTEGER OR REAL, AND A CONSTANT ENTRY IS ADDED TO THE SYMBOL TABLE, IF ALPHABETIC OR SPECIAL, THE STRING IS SEARCHED FOR FIRST IN THE SYMBOL TABLE AND THEN IN THE RESERVED WORD LIST. IF IT IS IN THE RESERVED WORD LIST A POSITIVE INTEGER DEFINITION CODE IS STORED IN 'IENTRY'

IF A SYMBOL APPEARS IN OR IS INSERTED INTO THE SYMBOL TABLE, THE NEGATIVE OF THE POINTER TO THAT SYMBOL IS STORED IN 'IENTRY' AND CONTROL IS PASSED BACK TO LEXCAL. SPECIAL STRINGS .TRUE, AND .FALSE, ARE TREATED AS CONSTANTS AND ASSIGNED THE VALUES -1 AND 0 RESPECTIVELY.

### 10.4 USAGE

CALLING SEQUENCE:

CALL CHKSYM(ICHARY,NCH,ITYPE,IENTRY,IRFSWD,IER)

ICHARY= ARRAY CONTAINING CHARACTER STRING

NCH = NUMBER OF CHARACTERS IN 'ICHARY'

ITYPE = TYPE OF CHARACTER STRING (SEE GETSMB)

IENTRY= NEGATIVE POINTER TO SYMBOL TABLE  
OR POSITIVE RESERVED WORD DEFINITION CODE,

IRFSWD= RESERVED WORD LIST

IER = ERROR FLAG (SEE LEXCAL)

## MINIS PROGRAM SPECIFICATION

11.0 PROGRAM NAME- CHK TYP

11.1 PURPOSE

CHECK NEWLY CREATED SYMBOLS FOR SYNTAX IN SYNTHESIS.

11.2 IDENTIFICATION

DATE- 27 NOVEMBER 1974

REVISION-

AUTHOR- D.R.SANDERS

COMPUTER- DCR 6024

LANGUAGE- FORTRAN IV

CORE SIZE- 154 WORDS (232 OCTAL)

11.3 DESCRIPTION

CHK TYP ACCEPTS ONE VAL FIELD FROM THE SYNTAX GRAPH AND ONE SYMBOL TABLE POINTER FROM THE SENTENCE BEING SYNTHESIZED. THE VAL FIELD INDICATES THE TYPE OF VARIABLE OR CONSTANT ALLOWABLE. IF THE SYMBOL TYPE IS GREATER THAN 100 A CHECK IS MADE TO SEE IF THE SYMBOL WAS USED CORRECTLY. IF SO, 100 IS SUBTRACTED FROM THE TYPE FIELD IN THE SYMBOL TABLE. IF NOT, AN ERROR FLAG IS SET.

11.4 USAGE

CALLING SEQUENCE:

CALL CHK TYP (IVAL, ISNSMB, MATCH)

IVAL = VAL FIELD FROM SYNTAX GRAPH

ISNSMB = SYMBOL POINTER FROM SENTENCE

MATCH = FLAG: 0=MATCH, 1=MISMATCH.

ORIGINAL PAGE IS  
OF POOR QUALITY.

## MINIS PROGRAM SPECIFICATION

12.0 PROGRAM NAME- CHRTRN

12.1 PURPOSE

TO TRANSFER CHARACTERS IN THE ELEMENT ARRAY INTO COMMON FOR TRANSFER INTO THE DATA BASE RECORD.

12.2 IDENTIFICATION

DATE- 12-22-75

REVISION-

AUTHOR- C.J.FARLESS (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 185 LOCATIONS (271 OCTAL)

12.3 DESCRIPTION

THE NUMBER OF CHARACTERS REQUESTED ARE TRANSFERRED FROM THE ELEMENT ARRAY INTO REUSE FOR LATER TRANSFER INTO THE DATA BASE RECORD.

12.4 USAGE

CALLING SEQUENCE:

CALL CHRTRN(LOC,NCHAR,NEXT,ITYPE)

LOC = LOCATION OF CHARACTER IN ELEMENT ARRAY.

NCHAR = NUMBER OF CHARACTERS TO TRANSFER,

NEXT = NEXT AVAILABLE BYTE AFTER TRANSFER,

ITYPE = TYPE OF BYTE: 0=8 BIT 6=6 BIT

IF ITPR=6 THEN NCHAR MUST NOT BE GREATER THAN  
NBW\*2/6

## MINIS PROGRAM SPECIFICATION

13.0 PROGRAM NAME- COMPUT

13.1 PURPOSE

THE QUADS NECESSARY TO DEFINE ONE FIELD ARE EXECUTED AND THE RESULT IS STORED ON THE DATA BASE RECORD.

13.2 IDENTIFICATION

DATE- 11-30-75

REVISION-

AUTHOR- C.J. FARLESS (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 483 LOCATIONS (743 OCTAL)

13.3 DESCRIPTION

USING THE QUADS STORED IN IQU(5,500), THE QUADS FOR ONE FIELD ARE EXECUTED. BEFORE EXECUTING THE OPERATION REQUESTED IN IQU(1,1), THE OPERANDS IN WORD 2 AND WORD 3 ARE OBTAINED USING 'GETOPR'. THESE OPERANDS ARE THEN USED TO PERFORM THE OPERATION. THE RESULT OF THE OPERATION IS STORED IN WORDS 4 AND 5. WHEN A 73 IS LOCATED THE FINAL RESULT IS STORED ON THE DATA BASE RECORD USING 'SI0FLD'.

13.4 USAGE

CALLING SEQUENCE:

CALL COMPUT(NQUAD)

NQUAD = 1ST QUAD TO EXECUTE IN A FIELD

ORIGINAL PAGE IS  
OF POOR QUALITY

## MINIS PROGRAM SPECIFICATION

14.0 PROGRAM NAME- CTRL

14.1 PURPOSE

CONTROL THE COMMAND INPUT, SCANNER, ANALYZER AND INTERPRETER PROCESSES IN OLDMAN,

14.2 IDENTIFICATION

DATE- 15 NOVEMBER 1974

REVISION-

AUTHOR- D.R.SANDERS (205)-883-1778

COMPUTER- DCR 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 1332 LOCATIONS (2464 OCTAL)

14.3 DESCRIPTION

THIS ROUTINE READS THE RESERVED WORD LIST INTO THE SCRATCH COMMON AREA IF IT IS NOT ALREADY THERE AND INPUTS THE SYNTAX GRAPH FOR DAPAC. A LOOP BEGINS BY READING AN INSTRUCTION AND THEN CALLS ARE MADE TO THE LEXICAL SCANNER, PARSER, SYNTHESIZER AND INTERPRETER, ANY APPROPRIATE ERROR MESSAGES ARE PRINTED, AND CONTROL IS PASSED BACK TO THE TOP OF THE LOOP.

14.4 USAGE

CALLING SEQUENCE:

CALL CTRL

## MINIS PROGRAM SPECIFICATION

15.0 PROGRAM NAME- COORD

15.1 PURPOSE

TO ACCEPT LATITUDE OR LONGITUDE DEGREES AND MINUTES FROM THE USER TERMINAL.

15.2 IDENTIFICATION

DATE- 5 AUGUST 73

REVISION- 29 AUGUST 1973 FORMER MERITS PROGRAM

AUTHOR- M.E.GILLIS

COMPUTER- DC 6024

LANGUAGE- FORTRAN

CORE SIZE- 266 LOCATIONS (412 OCTAL)

15.3 DESCRIPTION

DEGREES, MINUTES, OBSERVATION ID, AND ROLL/POSITION ARE ACCEPTED IN SEVERAL FORMATS. TYPE OF DATA BEING INPUT IS SELECTED BY THE CALLING PROGRAM BY USE OF ARGUMENT LL (1=LAT, 2=LONG, 4=OBVID, 5=ROLL/POS). LATITUDE IS RETURNED AS DEGREES AND MINUTES WITH DEGREES NEGATIVE IF SOUTH LATITUDE AND WITH A RANGE OF PLUS OR MINUS 90. WEST LONGITUDE IS POSITIVE TO USER, BUT RETURNED AS NEGATIVE FOR ERIS LOGIC. LONGITUDE IS PLUS OR MINUS 180, IF FORMAT CANNOT BE DECIPHERED, LL IS SET TO 3. MINUTES ARE ALWAYS POSITIVE.

NORTH LATITUDE CAN BE SPECIFIED BY + OR N, SOUTH BY S OR -. IF NONE SPECIFIED, NORTH ASSUMED. WEST LONGITUDE IS SPECIFIED BY W OR +, EAST BY E OR -. WEST ASSUMED IF NONE SPECIFIED. DECIMAL FRACTIONS CAN BE USED TO SPECIFY DEGREES. ALLOWED CHARACTERS ARE: NUMBERS, +, -, N, E, S, W, D, M, ,, COMMA, /, AND SPACE.

EXAMPLES:

LATITUDE	33 30	= 33 DEGREES, 30 MINUTES NORTH
	33N30	= 33 DEGREES, 30 MINUTES NORTH
	33 30N	= 33 DEGREES, 30 MINUTES NORTH
	33D30MN	= 33 DEGREES, 30 MINUTES NORTH
	+33D30M	= 33 DEGREES, 30 MINUTES NORTH
	33.5	= 33 DEGREES, 30 MINUTES NORTH
	33.5N	= 33 DEGREES, 30 MINUTES NORTH
	N33,30	= 33 DEGREES, 30 MINUTES NORTH
LONGITUDE	120W45	= 120 DEGREES, 45 MINUTES WEST
	120,75	= 120 DEGREES, 45 MINUTES WEST
	120,75W	= 120 DEGREES, 45 MINUTES WEST

120D45MW= 120 DEGREES, 45 MINUTES WEST  
E30M = 0 DEGREES, 30 MINUTES EAST  
-30M = 0 DEGREES, 30 MINUTES EAST  
30E = 30 DEGREES, 0 MINUTES EAST

#### 15.4 USAGE

##### CALLING SEQUENCE:

CALL COORD(IN,LDEG,LMIN,LL,KPOINT)

LDEG RETURNS DEGREES (PLUS OR MINUS)  
LMIN RETURNS MINUTES (PLUS)  
LL SPECIFIES LATITUDE OR LONGITUDE

LL = 1 FOR LATITUDE  
LL = 2 FOR LONGITUDE  
LL = 4 FOR OBSERVATION ID  
LL = 5 FOR FILM ROLL/POSITION

LL RETURNED AS 3 IF AN ERROR WAS DETECTED DURING SCAN,

KPOINT POINTS TO CHARACTER IN INPUT STRING WHICH DELIMITS  
OBSERVATION ID OR FILM/ ROLL INDICATOR,

OBSERVATION ID (OBVID) OR MICROFILM ROLL/POSITION ARE ENTERED AS  
FOLLOWS:

OBVID FORMAT IS NUMBER-NUMBER  
ROLL FORMAT IS NUMBER/NUMBER

IF AN ILLEGAL CHARACTER IS DETECTED DURING SCAN, AN ERROR RETURN  
NUMBER IS SET IN LL.

THE POSITION OF THE FIRST CHARACTER AFTER AN OBVID OR ROLL NUMBER  
IS RETURNED IN KPOINT SO THAT FURTHER SCAN CAN BE ACCOMPLISHED BY  
OTHER PROGRAMS.

NORTH LATITUDE IS PLUS, WEST LONGITUDE IS ACCEPTED AS PLUS,  
RETURNED AS MINUS.

COORD USES FUNCTION KSCAN FOR CHARACTER SCAN.

## MINIS PROGRAM SPECIFICATION

16.0 PROGRAM NAME- COORDN

16.1 PURPOSE

ACCEPT COORDINATES OF POINT OR AREA AND CONDUCT SEARCH OF INDEX FILES FOR PHOTOS COVERING THE POINT OR AREA.

16.2 IDENTIFICATION

DATE- 17 SEPTEMBER, 1975

REVISION-

AUTHOR- R.L.KEFFER (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 150 LOCATIONS (226 OCTAL)

16.3 DESCRIPTION

THE COORDINATES OF THE POINT OR AREA ARE INPUT THROUGH A DIALOG WITH THE USER. THESE COORDINATES ARE USED AS ENTRIES TO THE INDEX FILES TO DETERMINE A LIST OF CANDIDATES FOR MEMBERSHIP IN THE SET. IF THE DEFINED AREA IS A POINT, THE FILE IS CHECKED FOR ACTUAL MEMBERS THROUGH A GEOGRAPHIC CORRECTION ROUTINE.

16.4 USAGE

CALLING SEQUENCE:

CALL COORDN(KEY,NPHOT,LINDEX)

KEY = USER SET FLAG SET TO 0 OR 1 FOR POINT OR AREA COVERAGE RESPECTIVELY.

NPHOT = NUMBER OF PHOTOS FOUND.

LINDEX= ADDRESS OF INDEX TO LATLON INDEX.

ORIGINAL PAGE IS  
OF POOR QUALITY

## MINIS PROGRAM SPECIFICATION

17.0 PROGRAM NAME- CREATE

17.1 PURPOSE

TO CREATE A DATA BASE FILE AND OTHER NECESSARY AUXILLARY FILES TO BE USED WITH THE DATA BASE MANAGER PROGRAM 'OLDMAN',

17.2 IDENTIFICATION

DATE- 10-20-75

REVISION-

AUTHOR- CATHY FARLESS (205) 883-1778

COMPUTER- DATARAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 221 LOCATIONS (335 OCTAL)

17.3 DESCRIPTION

THE NECESSARY AUXILLARY FILE NAMES AND FILES ALONG WITH THE DATA BASE FILE NAME ARE CREATED IF NOT LOADED WITH OLDMAN. THE FIELD DEFINITION FILE MUST BE CREATED BEFORE EXECUTING CREATE. THE RAW DATA INFORMATION INPUT DECK IS THEN ENTERED THROUGH THE CARD READER AND STORED IN THE ARRAY INPT. THE NUMBER OF CARDS IN THIS DECK WILL DETERMINE THE NUMBER OF ELEMENTS NEEDED TO CREATE A DATA BASE RECORD. THE INPUT MODE INFORMATION IS THEN ASSEMBLED (TYPE OF DEVICE TO INPUT DATA), NUMBER OF BYTES PER PHYSICAL AND PER LOGICAL RECORDS, THE NUMBER OF BYTES IN THE KEY VALUE, AND THE KEY VALUE, IF ANY). THE OPERATION CARDS ARE THEN READ FROM THE CARD READER AND THE NECESSARY EXECUTION QUADS ARE CREATED (QDFORM). THE RAW DATA IS THEN READ AND THE ELEMENTS STORED IN THE ELEMENT ARRAY (READLR AND STELEM) ONE LOGICAL RECORD AT A TIME. USING THIS LOGICAL RECORD AND ITS RESULTANT ELEMENT ARRAY, THE QUADS ARE EXECUTED FOR ALL DEFINED FIELDS (COMPUT) AS EACH FIELD VALUE IS DETERMINED, IT IS PACKED INTO THE DATA BASE RECORD ARRAY (PACDAT AND STOFD). AFTER ALL FIELDS HAVE BEEN COMPUTED, THE DATA BASE RECORD IS WRITTEN. ANOTHER RAW DATA RECORD IS READ AND THE PROCESS CONTINUES UNTIL ALL THE RAW DATA HAS BEEN PROCESSED. THE NUMBER OF DATA BASE RECORDS CREATED ARE STORED IN THE IDFF FILE. IF THE USER INDICATES TO SAVE THE CREATED RECORDS, THE FILE DEFINITION FILE IS WRITTEN AND CLOSED AND THE DATA BASE DATA FILE IS CLOSED. OPTIONS AVAILABLE: UPDATE, SORT, MERGE, MODIFY, AND INITIALIZATION. UPDATE ALLOWS NEW RECORDS TO BE ADDED IN AN ALREADY EXISTING DATA BASE FILE. SORT WILL SORT THE ENTIRE DATA BASE IN UP TO 4 FIELDS (A FIELD TO SORT ON MUST BE 2 WORDS LESS 2 BITS OR LESS IN LENGTH). MERGE WILL MERGE RECORDS WHOSE REQUESTED FIELD VALUES ARE IDENTICAL, SAVING THE MOST CURRENT RECORD, DELETING THE OLD RECORD. MODIFY WILL ALLOW NEW FIELDS TO BE ADDED TO EXISTING RECORDS OR OLD FIELD VALUES TO BE CHANGED. INITIALIZATION WILL ALLOW A NEW DATA BASE TO BE CREATED FROM SCRATCH.

17.4 USAGE

CALLING SEQUENCE:

CALL CREATE

USER: CREATE

ORIGINAL PAGE IS  
OF POOR QUALITY

## MINIS PROGRAM SPECIFICATION

18.0 PROGRAM NAME- CRREWL

18.1 PURPOSE

CREATE THE RESERVED WORD LIST TO BE USED BY SUBROUTINE LEXCAL.

18.2 IDENTIFICATION

DATE- 8 NOVEMBER 1974

REVISION-

AUTHOR- D.R.SANDERS

COMPUTER- DCR 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 213 LOCATIONS (325 OCTAL)

18.3 DESCRIPTION

CRREWL INPUTS EIGHT CHARACTER RESERVED WORDS AND AN INTEGER FROM LOGICAL UNIT 5 USING FORMAT(8A1,I2) AFTER A FOUR DIGIT INTEGER IS READ INDICATING THE NUMBER OF RESERVED WORDS, THE CHARACTERS ARE PACKED INTO A TABLE WITH NINE BYTES PER ENTRY.

18.4 USAGE

CALLING SEQUENCE:

CALL CRREWL(IRESWD,IER)

IER = ERROR FLAG: 0 IF GOOD; 1 IF TOO MANY RES. WORDS

IRESWD= RESERVED WORD LIST

RESERVED WORD LIST IS STORED IN  
COMMON/REUSE/ STARTING WITH THE SECOND WORD,  
A 99 IS STORED AFTER THE LAST ENTRY IN THE NINTH BYTE.

## MINIS PROGRAM SPECIFICATION

19.0 PROGRAM NAME- CVREAL

19.1 PURPOSE

CONVERT A CHARACTER STRING TO A FLOATING POINT NUMBER

19.2 IDENTIFICATION

DATE- 8 NOVEMBER 1974

REVISION-

AUTHOR- D.R.SANDERS (205)-883-1778

COMPUTER- DCR 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 245 LOCATIONS (365 OCTAL)

19.3 DESCRIPTION

'N' CHARACTERS ARE EXAMINED TO BUILD AN UNSIGNED REAL VALUE FROM 'ISTR'. E FORMAT OR REGULAR REAL NOTATION CAN BE CONVERTED. IF AN ERROR IS ENCOUNTERED IN 'ISTR' A FLOATING POINT REPRESENTATION OF THE NUMBER TO THE LEFT OF THE POINT OF INFRACTION IS RETURNED IN 'CVREAL' AND 'IER' IS SET.

19.4 USAGE

CALLING SEQUENCE:

REAL = CVREAL(ISTR,N,I,IER)

REAL = REAL RESULT.

ISTR = ARRAY CONTAINING CHARACTER STRING.

N = NUMBER OF CHARACTERS IN STRING.

I = FIRST CHARACTER POSITION IN STRING.

IER = ERROR FLAG SET AS FOLLOWS:  
0 FOR SUCCESSFUL CONVERSION,  
1 FOR INVALID CHARACTER IN STRING.

ORIGINAL PAGE IS  
OF POOR QUALITY

## MINIS PROGRAM SPECIFICATION

20.0 PROGRAM NAME- C6T08

20.1 PURPOSE

CONVERT A 6 BIT CHARACTER STRING TO AN 8 BIT CHARACTER STRING.

20.2 IDENTIFICATION

DATE- 12-22-75

REVISION-

AUTHOR- C.J. FARLESS (205) 883-1778

COMPUTER- DATAKRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 40 LOCATIONS (50 OCTAL)

20.3 DESCRIPTION

EACH 6 BIT BYTE OF THE INPUT STRING IS CONVERTED TO AN 8 BIT BYTE BY ADDING 32 TO EACH BYTE.

20.4 USAGE

CALLING SEQUENCE:

```
CALL C6T08 (STRING, NCHAR, IFIR, ARPAY)
STRING = 6 BIT CHARACTER INPUT STRING.
NCHAR = NUMBER OF 6 BIT BYTES TO CONVERT.
IFIR = FIRST BIT IN INPUT ARRAY TO CONVERT.
ARRAY = 8 BIT BYTE OUTPUT ARRAY.
```

# MINIS PROGRAM SPECIFICATION

21.0 PROGRAM NAME- C8T06

21.1 PURPOSE

TO CONVERT AN 8 BIT CHARACTER STRING TO A 6 BIT CHARACTER STRING,

21.2 IDENTIFICATION

DATE- 12-22-75

REVISION-

AUTHOR- C.J. FARLESS (205) 883-1778

COMPUTER- DATARAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 50 LOCATIONS (62 OCTAL)

21.3 DESCRIPTION

EACH 8 BIT CHARACTER IS CONVERTED TO A 6 BIT CHARACTER BY SUBTRACTING 32 FROM THE 8 BIT CHARACTER,

21.4 USAGE

CALLING SEQUENCE:

CALL C8T06 (STRING, NCHAR, ISTART, ARRAY)

STRING = INPUT STRING OF 8 BIT CHARACTERS,

NCHAR = NO. OF CHARACTERS TO CONVERT,

ISTART = BYTE NO. TO BEGIN WITH,

ARRAY = OUTPUT 6 BIT CHARACTER STRING,

## MINIS PROGRAM SPECIFICATION

22.0 PROGRAM NAME- DRLMAD

22.1 PURPOSE

SELECT A DATA BASE AND READ ASSOCIATED DEFINITION FILES,

22.2 IDENTIFICATION

DATE- 9 JULY, 1975

REVISION-

AUTHOR- R.L.KEEFFER (205)-883-1778

COMPUTER- DATACRAFT 6024

LANGUAGE- FORTRAN IV

CORE SIZE- 632 LOCATIONS (1170 OCTAL)

22.3 DESCRIPTION

THE DATA BASE NAME IS REQUESTED IF NOT ALREADY KNOWN, THE FILE  
DEFINITION FILE IS READ INTO 'IDEF', THE USER MESSAGE INDEX AND  
THE FIELD DEFINITION FILES ARE READ INTO COMMON AREAS.

22.4 USAGE

CALLING SEQUENCE:

CALL DRLMAD

# MINIS PROGRAM SPECIFICATION

23.0 PROGRAM NAME- DFLBLK

23.1 PURPOSE

DELETE BLANKS IN 80 COLUMNS OF THE OPERATION CARDS AND RETURN THE NUMBER OF COLUMNS OF CHARACTERS LEFT.

23.2 IDENTIFICATION

DATE- 11-20-75

REVISION-

AUTHOR- C.J. FARLESS (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 82 LOCATIONS (122 OCTAL)

23.3 DESCRIPTION

THE OPERATION CARD IS STORED IN IARRAY FOR ERROR PRINTOUT, THEN THE LINE IS STORED IN ILINE AND IS SCANNED, REMOVING BLANK CHARACTERS. A COUNT IS MADE OF THE NUMBER OF NON-BLANK CHARACTERS FOUND AND IS RETURNED AS THE NUMBER OF COLUMNS (NCOL).

23.4 USAGE

CALLING SEQUENCE:

CALL DELBLK(ILINE,NCOL)

ILINE = ARRAY CONTAINING LINE TO SCAN.  
NCOL = NO. OF COLUMNS OF CHARACTERS  
(NON-BLANK) FOUND.

## MINIS PROGRAM SPECIFICATION

24.0 PROGRAM NAME- EXCANS

24.1 PURPOSE

EXECUTE FLAGGED STATEMENTS IN THE QUAD TABLE TO DETERMINE A CANDIDATE RECORD'S SET MEMBERSHIP STATUS.

24.2 IDENTIFICATION

DATE- 2-3-75

REVISION-

AUTHOR- C.J.FARLESS (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 49 LOCATIONS (61 OCTAL)

24.3 DESCRIPTION

IF NOT ALREADY PRESENT, THE CANDIDATE RECORD IS BROUGHT INTO CORE. ALL FLAGGED STATEMENTS WITHIN THE GIVEN LIMITS(IQF TO IQL) ARE EXECUTED WITH THE RESULTS OF EACH QUAD STORED IN THAT QUAD. THE FINAL RESULT OF EACH STATEMENT IS STORED AT THE BEGINNING QUAD OF THAT STATEMENT AND A BIT CORRESPONDING TO THE SET NUMBER IS SET IN THE CANDIDATE ARRAY(ICAND) IF THE RESULT IS TRUE. THE NUMBER OF CANDIDATES IS ALSO INCREMENTED FOR A TRUE RESULT.

24.4 USAGE

CALLING SEQUENCE:

CALL EXCANS(IQF,IQL,NCAND)

IQF = FIRST QUAD TABLE ENTRY TO EXECUTE

IQL = LAST QUAD TABLE ENTRY TO EXECUTE

NCAND = NUMBER OF SET MEMBERS

ICAND-THREE WORD CANDIDATE ARRAY IN COMMON

ICAND(1)-DATA BASE ENTRY

ICAND(2)-(3)-TWO WORD SET MEMBERSHIP ARRAY. BITS 1-N  
CORRESPOND TO SETS 1-N

## MINIS PROGRAM SPECIFICATION

25.0 PROGRAM NAME- EXFLD

25.1 PURPOSE

LOCATE THE SPECIFIED DATA BASE RECORD AND EXTRACT THE NUMBERED FIELD.

25.2 IDENTIFICATION

DATE- 15 OCTOBER 1974

REVISION-

AUTHOR- R.L.KEEFFER (205)-883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN-IV

CORE SIZE- 35 LOCATIONS (43 OCTAL)

25.3 DESCRIPTION

THE NUMBERED RECORD OF THE DATA BASE IS LOADED INTO A COMMON ARRAY. IF NOT ALREADY IN THAT ARRAY, THE RECORD NUMBER IS ALSO SET IN THAT ARRAY FOR FUTURE CHECKS. THE FIELD IS EXTRACTED BY SUBROUTINE EXFLD.

25.4 USAGE

CALLING SEQUENCE:

CALL EXFLD(IDATA, ITYPE, IFLDNO, IARRAY, MSG, IRECNO)

IDATA = DATA DESTINATION ARRAY, (4 WORDS)

ITYPE = DATA TYPE (SEE FIELD DEFINITIONS)

IFLDNO = FIELD DEFINITION NUMBER,

IARRAY = INTEGER ARRAY FOR SOURCE RECORD,

MSG = MESSAGE NUMBER OF FIELD TITLE,

IRECNO = RECORD NO. OF DATA BASE ENTRY CONTAINING FIELD,

## MINIS PROGRAM SPECIFICATION

26.0 PROGRAM NAME- EXOPRN

26.1 PURPOSE

PRESENT DATA IN FLOATING POINT FORMAT FOR QUAD TABLE OPERAND POINTERS.

26.2 IDENTIFICATION

DATE- 7-3-75

REVISION-

AUTHOR- C.J.FARLESS (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 270 LOCATIONS (416 OCTAL)

26.3 DESCRIPTION

IF THE OPERAND POINTER IS NEGATIVE, THE POINTER REFERS TO THE QUAD TABLE AND THE 4TH AND 5TH WORDS OF THE REFERENCED QUAD IS RETURNED AS DATA. IF THE OPERAND POINTER IS POSITIVE, THE POINTER IS AN ENTRY TO THE SYMBOL TABLE. THE SYMBOL TYPE AND ARRAY POINTER ARE EXTRACTED FROM THE SYMBOL TABLE. IF THE TYPE IS NOT A FIELD NAME, SET NAME, CREATED SYMBOL, OR CONSTANT, DATA IS RETURNED AS ZERO. FIELD NAME VARIABLES ARE EXTRACTED VIA SUBROUTINE EXTFLD USING THE ARRAY POINTER AND CONVERTED TO FLOATING POINT IF NECESSARY. OTHERWISE DATA IS RETURNED AS ZERO. FOR SET NAMES, THE ARRAY POINTER REFERENCES A SET NUMBER AND ITS CORRESPONDING BIT IS EXTRACTED FROM THE CURRENT CANDIDATE ARRAY, MULTIPLIED BY -1, AND RETURNED AS DATA. FOR CREATED SYMBOLS, THE ARRAY POINTER REFERENCES A CORRESPONDING QUAD AND WORDS 4 AND 5 ARE RETURNED AS DATA. FOR CONSTANTS, THE ARRAY POINTER REFERENCES THE CORRESPONDING VALUE IN THE SYMBOL TABLE WHICH IS RETURNED AS DATA.

26.4 USAGE

CALLING SEQUENCE

CALL EXOPRN(IOPRN, IDATA)

IOPRN = OPERAND POINTER FROM QUAD TABLE

IDATA = TWO WORD FLOATING POINT RETURN ARRAY

## MINIS PROGRAM SPECIFICATION

27.0 PROGRAM NAME- EXQUAD

27.1 PURPOSE

EXECUTE ALL FLAGGED QUAD TABLE ENTRIES IN A SEGMENT OF THE QUAD TABLE.

27.2 IDENTIFICATION

DATE- 7-9-75

REVISION-

AUTHOR- C.J. FARLESS (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 578 LOCATIONS (1102 OCTAL)

27.3 DESCRIPTION

EACH FLAGGED STATEMENT IN THE QUAD TABLE BETWEEN THE GIVEN LIMITS IS EXECUTED ONE QUAD AT A TIME. THE ONLY QUAD TYPES THAT ARE NOT EXECUTED ARE STATEMENT START CODES(OP CODE 73), SUBROUTINE CALLS(OP CODE 6), AND NO-OPS(OP CODE 7). THE OPERANDS FOR EACH QUAD ARE OBTAINED VIA SUBROUTINE EXOPRN, RESULTS ARE STORED IN THE 4TH AND 5TH WORDS OF EACH QUAD IN THE TABLE, AND THE FINAL RESULT OF A STATEMENT IS STORED IN THE 4TH AND 5TH WORDS OF THE STATEMENT START QUAD(OP CODE 73).

27.4 USAGE

CALLING SEQUENCE:

VAL = EXQUAD(IQF,IQL)

IQF=STARTING QUAD NUMBER TO BE EXECUTED  
IQL=LAST QUAD NUMBER TO BE EXECUTED  
VAL=RESULT OF THE LAST EXECUTED QUAD

## MINIS PROGRAM SPECIFICATION

28.0 PROGRAM NAME- EXSETD

28.1 PURPOSE

EXECUTE A SET DEFINITION STATEMENT OF THE QUAD TABLE.

28.2 IDENTIFICATION

DATE- 3 FEBRUARY 1975

REVISION-

AUTHOR- D.R.SANDERS

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 1001 LOCATIONS (1751 OCTAL)

28.3 DESCRIPTION

EACH VARIABLE IN THE STATEMENT IS EXAMINED THROUGH SUBROUTINE 'REVSCN' TO DETERMINE ALL PREVIOUS STATEMENTS TO BE EXECUTED IN DEFINING THE SET.

THE STATEMENT IS ANALYZED TO ESTABLISH A LIST OF SET LIMITING RELATIONS. ('ADDED' RESTRICTIONS) THIS LIST OF RESTRICTIONS IS SCANNED FOR THE PRESENCE OF AN INDEXED VARIABLE. IF FOUND, THE ASSOCIATED RESTRICTION IS APPLIED TO BUILD A CANDIDATE FILE. EACH ENTRY OF THE CANDIDATE FILE IS A THREE WORD ARRAY. THE FIRST WORD IS THE DATA BASE ENTRY POINTER. THE SECOND AND THIRD WORDS (SET MEMBERSHIP BIT MAP) ARE SET TO ZERO. THE QUAD TABLE ENTRY USED TO ESTABLISH THE CANDIDATE FILE IS SAVED AND SET TO A NOP WITH A TRUE VALUE. EACH CANDIDATE IS THEN CHECKED FOR SET MEMBERSHIP VIA SUBROUTINE 'SUBSET'. THE SAVED QUAD TABLE ENTRY IS RESTORED. THE NUMBER OF SET MEMBERS IS SAVED AND THE ROUTINE IS EXITED.

IF NO INDEXED VARIABLES ARE FOUND, THE ENTIRE DATA BASE IS CHECKED VIA SUBROUTINE 'EXCANS', AND FOUND MEMBERS ARE SAVED WITH THEIR BIT MAPS IN THE CANDIDATE FILE. THE FILE IS CLOSED, INPUT FILE NAME IS SWAPPED AND THE NUMBER OF CANDIDATES IS RETURNED.

28.4 USAGE

CALLING SEQUENCE:

CALL EXSETD(IQF,IQL,NCAND)

IQF = FIRST QUAD TABLE ENTRY OF STATEMENT  
IQL = LAST QUAD TABLE ENTRY OF STATEMENT  
NCAND = NUMBER OF CANDIDATES FOUND.

# MINIS PROGRAM SPECIFICATION

29.0 PROGRAM NAME- EXSTMT

29.1 PURPOSE

EXECUTE A STATEMENT IN THE QUAD TABLE,

29.2 IDENTIFICATION

DATE- 24 JANUARY 1975

REVISION-

AUTHOR- D.R. SANDERS

COMPUTER- DCR 6024

LANGUAGE- FORTRAN IV

CORE SIZE- 215 LOCATIONS (327 OCTAL)

**ORIGINAL PAGE IS  
OF POOR QUALITY**

**PRECEDING PAGE BLANK NOT FILMED**

29.3 DESCRIPTION

THE STATEMENT TYPE IS EXAMINED TO DETERMINE THE APPROPRIATE ACTION TO BE TAKEN. FOR THE FIRST SET CREATION TYPE (1), THE SET IS DEFINED VIA SUBROUTINE 'EXSETD'. FOR A SUBSET CREATION TYPE STATEMENT (2), THE SET IS DEFINED VIA SUBROUTINE 'SURSET'. FOR A VARIABLE CREATION TYPE STATEMENT (3), NO ACTION IS TAKEN. FOR AN ASSIGNMENT TYPE SUBROUTINE CALL STATEMENT TYPE (6), THE DEFINED VARIABLE IS CHECKED FOR SYMBOL TYPE. IF THE SYMBOL TYPE IS A SET NAME, THE REFERENCED SUBROUTINE IS CALLED. OTHERWISE, THE STATEMENT IS IGNORED. IF THE STATEMENT TYPE IS A NON-ASSIGNMENT SUBROUTINE CALL (7), THE REFERENCED SUBROUTINE IS CALLED.

AFTER ANY SUBROUTINES ARE CALLED, THE QUAD STACK IS SCANNED THRU THE CURRENT STATEMENT TO FIND AND RESET ANY FLAGGED STATEMENTS. SET COUNTS OF MEMBERS OF CREATED SETS ARE SAVED IN COMMON/SETCNT/.

29.4 USAGE

CALLING SEQUENCE:

CALL EXSTMT(IQF, IQL)

IQF = FIRST QUAD TABLE ENTRY OF STATEMENT,  
IQL = LAST QUAD TABLE ENTRY OF STATEMENT.

## MINIS PROGRAM SPECIFICATION

30.0 PROGRAM NAME- EXTFLD

30.1 PURPOSE

EXTRACT THE SPECIFIED FIELD FROM THE GIVEN ARRAY.

30.2 IDENTIFICATION

DATE+ 7 JULY, 1975

REVISION-

AUTHOR- R.L.KEEFER (205)-883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN-IV

CORE SIZE- 1235 LOCATIONS (2323 OCTAL)

ORIGINAL PAGE IS  
OF POOR QUALITY

30.3 DESCRIPTION

THE FIELD DEFINITION ARRAY IS REFERENCED TO DETERMINE STARTING BIT POSITIONS, NUMBER OF BITS, DATA TYPE, AND DATA KEY. IF THE FIELD IS KEYED, THE FIELD KEY IS COMPARED WITH THE DATA RECORD KEY. THE DATA TYPE, 'ITYPE', IS SET ACCORDINGLY. THE EXTRACTED FIELD IS TRANSFERRED TO THE OUTPUT DATA ARRAY WORDS ONE AND TWO. IF THERE IS A SUBFIELD, IT IS TRANSFERRED TO WORDS THREE AND FOUR OF THE DATA ARRAY. IN BOTH CASES, THE DATA IS RETURNED RIGHT ADJUSTED IN THE MINIMUM NUMBER OF REQUIRED WORDS, (4MAX) IF THE DATA IS TEXT, THE FIRST BYTE NUMBER IS RETURNED IN IDATA(1). IF THE DATA TYPE IS CHARACTER(6), THE SPECIFIED NO. OF CHARACTERS (4\*NRW MAX) IS RETURNED LEFT JUSTIFIED IN IDATA ARRAY.

30.4 USAGE

CALLING SEQUENCE:

CALL EXTFLD(IDATA,ITYPE,IFLDNO,ARRAY,MSG)

IDATA = DATA DESTINATION ARRAY (4 WORDS)

ITYPE = DATA TYPE (SEE FIELD DEFINITIONS)

IFLDNO = FIELD DEFINITION NUMBER,

ARRAY = INTEGER ARRAY OF SOURCE RECORD,

MSG = MESSAGE NUMBER OF FIELD TITLE.

## MINIS PROGRAM SPECIFICATION

31.0 PROGRAM NAME- EXVARD

31.1 PURPOSE

EXECUTE A VARIABLE DEFINITION STATEMENT OF THE QUAD TABLE

31.2 IDENTIFICATION

DATE- 3-4-75

REVISION-

AUTHOR- C. J. FARLESS (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 202 LOCATIONS (312 OCTAL)

31.3 DESCRIPTION

GIVEN A QUAD TABLE ENTRY FOR A VARIABLE DEFINITION STATEMENT, EACH SUCCESSIVE QUAD IS EXECUTED, STORING THE RESULT IN WORDS 4 AND 5 OF THE QUAD, UNTIL ANOTHER STATEMENT BEGINS (OPERATION CODE=73) OR THE END OF THE QUAD TABLE IS REACHED (OPERATION CODE=0). AFTER COMPLETION OF A STATEMENT, THE RESULT OF THE FINAL OPERATION IS STORED IN THE 4TH AND 5TH WORDS OF THE FIRST QUAD TABLE ENTRY OF THE STATEMENT. IT IS ASSUMED THAT THE REFERENCED DATA BASE ENTRY IS IN CORE. ALL OPERANDS ARE EXTRACTED VIA SUBROUTINE EXOPRN AND ARE ASSUMED TO BE IN FLOATING POINT,

31.4 USAGE

CALLING SEQUENCE:

CALL EXVARD(IQST)

IQST = ENTRY POINTER TO STATEMENT START IN QUAD TABLE

## MINIS PROGRAM SPECIFICATION

32.0 PROGRAM NAME- FILDEL

32.1 PURPOSE

DELETE A FILE FROM THE ULDMAN SYSTEM.

32.2 IDENTIFICATION

DATE- 19 MAY, 1975

REVISION-

AUTHOR- R.L.KEFFER (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 285 LOCATIONS (435 OCTAL)

**ORIGINAL PAGE IS  
OF POOR QUALITY**

32.3 DESCRIPTION

THE NAMED FILE IS CLOSED. IF ACTIVE, THE EXISTENCE OF THE FILE IS CHECKED AND THE FILE IS DELETED VIA A DELETE COMMAND IF IT IS FOUND TO EXIST.

32.4 USAGE

CALLING SEQUENCE:

CALL FILDEL(IDE)

IDE = 5 WORD FILE DEFINITION WITH WORDS  
1-3 CONTAINING THE NAME (3A2)

## MINIS PROGRAM SPECIFICATION

33.0 PROGRAM NAME- FILEND

33.1 PURPOSE

CLOSE THE NAMED DISK FILE, IF OPEN, AND DELETE THE FILE FROM THE ACTIVE FILE LIST.

33.2 IDENTIFICATION

DATE- 11 DECEMBER, 1974

REVISION-

AUTHOR- R.L.KEFFER (205)-883-1778

COMPUTER- DATACRAFT 6024

LANGUAGE- FORTRAN IV

CORE SIZE- 206 LOCATIONS (316 OCTAL)

33.3 DESCRIPTION

THE NAMED FILE IS LOCATED IN THE FILE STACK, CLOSED, AND DELETED FROM THE STACK. THE CORRESPONDING ENTRY IN THE PRIORITY STACK IS DELETED AND ALL FOLLOWING ENTRIES ARE MOVED UP IN THE TABLE.

33.4 USAGE

CALLING SEQUENCE:

CALL FILEND(NAMFIL)

NAMFIL = FILE NAME ARRAY (3A2)

## MINIS PROGRAM SPECIFICATION

34.0 PROGRAM NAME- FILRED

34.1 PURPOSE

READ PORTIONS OR ENTIRE RANDOM DISC FILES.

34.2 IDENTIFICATION

DATE- 12 DECEMBER, 1974

REVISION-

AUTHOR- R.L.KEFFER (205)-883-1778

**ORIGINAL PAGE IS  
OF POOR QUALITY**

COMPUTER- DATACRAFT 6024

LANGUAGE- FORTRAN IV

CORE SIZE- 388 LOCATIONS (604 OCTAL)

34.3 DESCRIPTION

THE NAMED FILE IS ADVANCED TO THE TOP PRIORITY AND OPENED VIA 'SETPRI'. THE REQUESTED NUMBER OF BYTES IS CALCULATED AS WORDS AND THE RECORD NUMBER IS USED AS AN ENTRY POINT TO THE FILE. THE REQUESTED NUMBER OF WORDS ARE TRANSFERRED TO THE GIVEN ARRAY VIA SUBROUTINE 'REDWRT'.

34.4 USAGE

CALLING SEQUENCE:

CALL FILRED(NAMFIL,NUMREC,NBYT,ARRAY,IER)

NAMFIL = FILE NAME ARRAY. (NAME(3A2),NREC,IRECSZ)

NUMREC = RECORD NUMBER TO START READ,

NBYT = NUMBER OF BYTES TO READ,

ARRAY = ARRAY TO BE READ INTO,

IER = ERROR CODE SFT AS FOLLOWS:  
= 0 FOR NO ERRORS,  
= 1 FOR READ ERROR.

## MINIS PROGRAM SPECIFICATION

35.0 PROGRAM NAME- FILWRT

35.1 PURPOSE

WRITE ALL OR PART OF A RANDOM DISC FILE,

35.2 IDENTIFICATION

DATE- 12 DECEMBER, 1974

REVISION-

AUTHOR- R.L.KEEPER (205)-883-1778

COMPUTER- DATACRAFT 6024

LANGUAGE- FORTRAN IV

CORE SIZE- 223 LOCATIONS (337 OCTAL)

35.3 DESCRIPTION

THE NAMED FILE IS ADVANCED TO THE TOP PRIORITY AND OPENED VIA 'SETPRI'. THE REQUESTED NUMBER OF BYTES ARE WRITTEN FROM THE CALLING ARRAY VIA SUBROUTINE 'REDWRT'.

35.4 USAGE

CALLING SEQUENCE:

CALL FILWRT(NAMFIL,NUMREC,NBYT,ARRAY,IER)

NAMFIL = FILE NAME ARRAY. (NAME(3A2),NRECS,IRECSZ)

NUMREC = RECORD NUMBER TO START WRITING AT,

NBYT = NUMBER OF BYTES TO TRANSFER,

ARRAY = ARRAY TO BE WRITTEN FROM,

IER = ERROR CODE SET AS FOLLOWS:  
= 0 FOR NO ERRORS.  
= 1 FOR WRITE ERROR,

## MINIS PROGRAM SPECIFICATION

36.0 PROGRAM NAME- FINFLD

36.1 PURPOSE

INPUT A FIELD NAME AND COMPARE TO FIELD NAMES IN FIELD SPECIFICATION ARRAY,

36.2 IDENTIFICATION

DATE- 23 OCTOBER, 1974

REVISION-

AUTHOR- R.L.KEFFER (205)-883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN-IV

CORE SIZE- 62 LOCATIONS (76 OCTAL)

36.3 DESCRIPTION

A CALLER SPECIFIED MESSAGE IS OUTPUT SPECIFYING THE USE TO BE MADE OF THE FIELD NAME TO BE INPUT, A LINE IS INPUT AND THE FIRST EIGHT CHARACTERS ARE COMPARED TO EACH FIELD ID. IF A MATCH IS PRODUCED, THAT FIELD NUMBER IS RETURNED, OTHERWISE ZERO FOR NO MATCH AND A -1 FOR A BLANK BUFFER,

36.4 USAGE

CALLING SEQUENCE:

CALL FINFLD(MES,IFNO,MXFL,IBUF)

MES = MESSAGE NUMBER TO OUTPUT,

IFNO = FIELD NUMBER OF FOUND FIELD,

MXFL = SET TO TOTAL IDS IN CORE,

IBUF = BUFFER TO READ IN FIELD NAME,

ORIGINAL PAGE IS  
OF POOR QUALITY

## MINIS PROGRAM SPECIFICATION

37.0 PROGRAM NAME- FLDFIN

37.1 PURPOSE

ADD, DELETE, REPLACE, MODIFY, OR LIST FIELD DEFINITIONS.

37.2 IDENTIFICATION

DATE- 3 JULY, 1975

REVISION-

AUTHOR- R.L.KEEPER (205)-883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 2100 LOCATIONS (4064 OCTAL)

37.3 DESCRIPTION

THE USER IS REQUESTED TO INPUT THE DESIRED OPERATION TYPE. THE FIRST LETTER OF THE RESPONSE DETERMINES THE ACTION TAKEN. FOR ADD, REPLACE, AND MODIFY ACTIONS THE USER IS INSTRUCTED TO ENTER EACH SEGMENT OF THE FIELD DEFINITION AFFECTED. FOR THE DELETE ACTION, THE FIELD ID IS CHECKED TO DETERMINE IF IT IS A KEY FIELD. IF SO, A MESSAGE IS GIVEN TO DELETE REFERENCES TO THE KEY FIELD BEFORE IT CAN BE DELETED. OTHERWISE, THE FIELD DEFINITION IS DELETED. CROSS POINTERS ARE ADJUSTED AND THE ASSOCIATED MESSAGE IS DELETED FOR THE LIST ACTION. EACH SEGMENT OF THE FIELD DEFINITION IS LISTED. IF THE ACTION IS EXIT, THE USER IS ASKED IF THE PREVIOUS CHANGES SHOULD BE EFFECTED. IF SO, THE APPROPRIATE ARRAYS ARE WRITTEN ON THE SAVE FILE.

37.4 USAGE

CALLING SEQUENCE:

FLDFIN

# MINIS PROGRAM SPECIFICATION

38.0 PROGRAM NAME- FLTPT

38.1 PURPOSE

CONVERT CERTAIN FLOATING POINTS TO HOST MACHINE FLOATING POINT.

38.2 IDENTIFICATION

DATE- 2-17-76

REVISION-

AUTHOR- C.J. FARLESS (205) 883-1778

**ORIGINAL PAGE IS  
OF POOR QUALITY**

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 435 LOCATIONS (663 OCTAL)

38.3 DESCRIPTION

FLOATING POINT OF THE FOLLOWING FORMATS ARE CONVERTED TO THE HOST MACHINE FLOATING POINT FORMAT.

16 BIT WORDS-NOVA- DATA GENERAL 1 FLOATING POINT WORD=2 16 BIT WORDS BIT 0=SIGN BITS 1-7=EXPONENT+64 (•100) BITS 8-32 (OR 8-15 AND 0-15 OF SECOND WORD)=MANTISSA EXPONENT INDICATES THE POSITION OF THE DECIMAL POINT. AN EXPONENT OF 0 PUTS THE DECIMAL POINT TO THE LEFT OF BIT 8. AN EXPONENT OF 1 WILL PUT THE DECIMAL POINT TO THE LEFT OF BIT 12. AN EXPONENT OF -1 WILL PUT THE DECIMAL POINT TO THE LEFT OF 4 BITS OR BINARY DIGITS LEFT OF BIT 8. NEGATIVE NUMBERS (MANTISSA) ARE INDICATED WHEN BIT 0 IS 1 AND ARE IN SIGN MAGNITUDE FORM.

24 BIT WORDS-DATACRAFT 6024/3 1 FLOATING POINT WORD=2-24 BIT WORDS. WORD1 BIT 0=SIGN BITS 1-23=MANTISSA WORD 2 BIT 16=SIGN OF EXPONENT BITS 17-23=EXPONENT NEGATIVE MANTISSAS AND EXPONENTS ARE EXPRESSED IN TWO'S COMPLEMENT FORM.

32 BIT WORDS-SIGMA 5 1 FLOATING POINT WORD=1 32 BIT WORD BIT 0=SIGN BITS 1-7=EXPONENT+64 BITS 8-31=MANTISSA EXPONENTS ARE IN POWERS OF 4. FOR A NEGATIVE VALUE OF THE MANTISSA THE ENTIRE 32 BITS ARE IN TWO'S COMPLEMENT FORM.

32 BIT WORD-IBM 360 1 FLOATING POINT WORD=1 32 BIT WORD BIT 0=SIGN BITS 1-7=EXPONENT + 64 BITS 8-15=MANTISSA NEGATIVE MANTISSA VALUES ARE IN SIGN MAGNITUDE FORM WITH BIT 0 =1 EXPONENTS ARE IN POWERS OF 4. THE DECIMAL POINT IS TO THE LEFT OF BIT 8 WHEN THE EXPONENT IS 0.

IF NO CONVERSION IS NEEDED THE VALUE IS TRANSFERRED FROM THE INPUT STRING TO THE OUTPUT STRING.

## 38.4 USAGE

### CALLING SEQUENCE:

CALL FLTPT(ISTR,IFIR,IOUT,ITYP)

ISTR = INPUT STRING  
IFIR = FIRST BIT IN INPUT STRING TO CONVERT  
IOUT = OUTPUT STRING  
ITYP = DATA TYPE

## MINIS PROGRAM SPECIFICATION

39.0 PROGRAM NAME- FMATS

39.1 PURPOSE

SETUP OUTPUT FORMAT AND HEADER SPECIFICATIONS FOR OUTPUT REPORTS.

39.2 IDENTIFICATION

DATE- 7-3-75

REVISION-

AUTHOR- CATHY J. FARLESS (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 1387 LOCATIONS (2553 OCTAL)

39.3 DESCRIPTION

FROM A LIST OF ITEMS IN COMMON(REUSE), FORMATS TO PRINT THE HEADERS AND DATA LINES ARE CREATED AND STORED IN THE ARRAYS 'FMAT' AND 'FDMAT'.

39.4 USAGE

CALLING SEQUENCE:

CALL FMATS(ISTART,ISTOP,NCHAR,LCNT)

ISTART = NO. OF FIRST ITEM TO OUTPUT  
ISTOP = NO. OF LAST ITEM TO OUTPUT  
NCHAR = NO. OF CHARACTERS IN HEADER FORMAT  
LCNT = NO. OF LINES IN HEADER

ORIGINAL PAGE IS  
OF POOR QUALITY

## MINS PROGRAM SPECIFICATION

40.0 PROGRAM NAME- GENATE

40.1 PURPOSE

GENERATE THE REQUESTED INDEX FILE AND ITS FILE DEFINITION.

40.2 IDENTIFICATION

DATE- 5-28-75

REVISION-

AUTHOR- C. J. FARLESS (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 707 LOCATIONS (1303 ACTAL)

40.3 DESCRIPTION

IF TYPE EQUALS 1, THE NUMBER OF BITS NEEDED TO STORE THE LARGEST DATA BASE RECORD IS CALCULATED, THE NUMBER OF BITS NEEDED TO STORE THE FIELD IS EXTRACTED, AND THE MINIMUM NO. OF WORDS FOR AN INDEX ENTRY IS CALCULATED. THE FILE DEFINITION IS STORED IN IDEF(WRTDEF). THE INDEX FILE IS CREATED BY PACKING(PACKER) EACH DATA BASE RECORD NUMBER AND THE CORRESPONDING FIELD VALUE AND WRITING THE ENTRY IN THE FILE. AFTER ALL ENTRIES ARE ON FILE, THE FILE IS SORTED BY FIELD VALUE IN ASCENDING ORDER, THE NUMBER OF ENTRIES IN THE FILE WILL BE EQUAL TO THE NO. OF DATA BASE RECORDS. IF TYPE = 2, THE INDEX FILE DEFINITION IS LOCATED IN IDEF, THE MAXIMUM AND MINIMUM VALUES ARE CALCULATED, ALONG WITH SEGMENT SIZE, AND NUMBER OF WORDS FOR EACH ENTRY. THE FILE DEFINITION IS STORED IN IDEF(WRTDEF) AND THE INDEX FILE WRITTEN ON DISC BY STORING THE POINTER TO THE INDEXED FILE FOR EACH SEGMENT'S CORRESPONDING VALUE. THE NUMBER OF ENTRIES IN THE INDEX FILE WILL BE EQUAL TO THE NUMBER OF SEGMENTS PLUS 1, WHICH IS RETURNED IN NSEG. IF THE NUMBER OF RECORDS IN THE FILE IS RETURNED FROM WRTDEF AS 0, NSEG IS SET TO 0, INDICATING THAT WRTDEF COULD NOT STORE THE FILE DEFINITION, OR THAT THE INDEXED FILE COULD NOT BE FOUND.

40.4 USAGE

CALLING SEQUENCE:

CALL GENATE(IFLDNO, LEVEL, ITYPE, NSEG, WDEV)

IFLNO = FIELD NUMBER TO INDEX  
LEVEL = INDEX LEVEL  
ITYPE = TYPE INDEXING  
NSEG = REQUESTED NUMBER OF SEGMENTS  
WDEV = WORK AREA DEVICE NUMBER FOR SORT

= 0 FOR DISC  
= NUMBER ASSIGNED TO MT FOR TAPE.

**ORIGINAL PAGE IS  
OF POOR QUALITY**

## MINIS PROGRAM SPECIFICATION

41.0 PROGRAM NAME- GENDEX

41.1 PURPOSE

GENERATE, UPDATE, AND DELETE INDEX FILES, AND RECREATE INDEX FILES WHEN UPDATING THE DATA BASE,

41.2 IDENTIFICATION

DATE- 5-28-75

REVISION-

AUTHOR- C.J. FARLESS (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 226 LOCATIONS (342 OCTAL)

41.3 DESCRIPTION

IF THE FIELD NUMBER IS ZERO, GENDEX WILL REGENERATE ALL INDEX FILES DESCRIBED IN THE INDEX FILE DEFINITIONS(IDEF 41-N) USING GENATE, AND MAKE ANY NECESSARY CHANGES TO THE FILE DEFINITIONS, IF ITYPE IS ZERO, GENDEX WILL DELETE THE INDEX FILE AND ITS FILE DEFINITIONS USING DELIDX. FOR CALLS IN WHICH IFLDNO, LEVEL, AND ITYPE ARE ALL NONZERO, GENDEX WILL GENERATE THE REQUESTED INDEX FILE USING GENATE. THE NUMBER OF ENTRIES IN THE INDEX FILE WILL BE RETURNED THROUGH THE ARGUMENT NSEG.

41.4 USAGE

CALLING SEQUENCE:

CALL GENDEX(IFLDNO,LEVEL,ITYPE,NSEG,WDEV)

IFLDNO = FIELD NUMBER (=0 WHEN UPDATING DATA BASE)

LEVEL = LEVEL OF INDEX TO GENERATE OR DELETE

ITYPE = TYPE OF INDEXING

= 1 FOR LEVEL ONE INDEXING

= 2 FOR LEVEL 2 INDEXING

= 0 WHEN DELETING INDEX FILE

NSEG = NUMBER OF SEGMENTS OF INDEXED FILE FOR LEVEL 2 INDEXING. THE NUMBER OF ENTRIES IN THE GENERATED FILE IS RETURNED IN NSEG.

WDEV = DEVICE NUMBER OF SORT WORK AREA

= 0 FOR DISC

= NUMBER ASSIGNED TO MT FOR TAPE.

# MINIS PROGRAM SPECIFICATION

42.0 PROGRAM NAME- GETNXT

42.1 PURPOSE

PROVIDE SUBROUTINE GETSMB WITH THE NEXT CHARACTER DURING THE LEXICAL SCANNING PROCESS,

42.2 IDENTIFICATION

DATE- 25 OCTOBER 1974

REVISION-

AUTHOR- D.R.SANDERS

COMPUTER- DCR 6024

LANGUAGE- FORTRAN IV

CORE SIZE- 50 LOCATIONS (62 OCTAL)

42.3 DESCRIPTION

GETNXT FIRST DEPOSITS THE CHARACTER JUST CHECKED INTO THE OUTPUT ARRAY. THEN IT STEPS THE POINTER TO THE INPUT ARRAY BY ONE, COMPARES THE CHARACTER COUNT TO THE MAXIMUM ALLOWABLE IN A SYMBOL, AND EXTRACTS THE CHARACTER FROM THE INPUT ARRAY IF THERE IS NO OVERFLOW, CONTROL IS RETURNED TO GETSMB AT THIS POINT,

42.4 USAGE

CALLING SEQUENCE:

CALL GETNXT(IARY, NSVE, NEXT, ICHARY, ICHR, NLIM)

IARY = INPUT ARRAY

NSVE = POINTER TO START OF SYMBOL IN IARY

NEXT = POINTER TO CURRENT SYMBOL IN IARY

ICHARY = OUTPUT ARRAY

ICHR = CURRENT CHARACTER FROM IARY

NLIM = MAXIMUM CHARACTER COUNT

ORIGINAL PAGE IS  
OF POOR QUALITY

## MINIS PROGRAM SPECIFICATION

43.0 PROGRAM NAME- GETOPR

43.1 PURPOSE

TO LOCATE AND RETURN THE VALUE OF AN OPERAND.

43.2 IDENTIFICATION

DATE- 12-16-75

REVISION-

AUTHOR- C.J.FARLESS (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 166 LOCATIONS (246 OCTAL)

43.3 DESCRIPTION

THE TYPE OF OPERAND IS DETERMINED BY THE INPUT ARGUMENT IVAL (VALUE OF OPERAND IN THE QUAD TABLE). NEGATIVE VALUES INDICATE A QUAD TABLE VALUE OR INTERMEDIATE VALUE. THEIR VALUES ARE EXTRACTED FROM THE 4TH AND 5TH WORDS IN THE DESIGNATED QUAD OF THE QUAD TABLE AND RETURNED IN THE IOP ARGUMENT. ALL OTHER VALUES OF IVAL ARE DIVIDED BY 1000 AND 1 ADDED TO DETERMINE THE TYPE. THE REMAINDER IS STORED IN NUM. IF ITYPE=1 (OR IVAL IS LESS THAN 1000), IT IS AN ELEMENT. THE VALUE OF NUM DETERMINES WHICH ELEMENT IN THE ELEMENT ARRAY IS RETURNED IN IOP. IF THE TYPE IS 2 OR 3 (IVAL IS GREATER THAN 1000 AND LESS THAN 3000), IT IS A CONSTANT. THE VALUE OF NUM DETERMINES WHICH CONSTANT IN THE ICON ARRAY IS RETURNED IN IOP. A FIELD NUMBER IS TYPE 4 (IVAL=3001 TO 3200). THE VALUE OF NUM INDICATES WHERE IN THE FIELD NUMBER ARRAY THAT THE QUAD NUMBER OF THE FIELD VALUE IS STORED. THE VALUE STORED IN THAT QUAD (WORD 4 AND 5) IS RETURNED IN IOP.

43.4 USAGE

CALLING SEQUENCE:

CALL GETOPR(IVAL, IOP)

IVAL = OPERAND VALUE FROM QUAD TABLE

IOP = 2 WORD RETURN ARRAY FOR VALUE OF OPERAND

# MINIS PROGRAM SPECIFICATION

44.0 PROGRAM NAME- GETSMR

44.1 PURPOSE

RETRIEVE A SYMBOL FOR SUBROUTINE LEXCAL AND NOTIFY THAT ROUTINE WHEN AN ILLEGAL SYMBOL OR THE END OF AN INPUT LINE IS ENCOUNTERED,

44.2 IDENTIFICATION

DATE- 25 OCTOBER 1974

REVISION-

AUTHOR- D.R.SANDERS

ORIGINAL PAGE IS  
OF POOR QUALITY

COMPUTER- DCR 6024

LANGOAGE- FORTRAN IV

CORE SIZE- 525 LOCATIONS (1015 OCTAL)

44.3 DESCRIPTION

GETSMB ACCEPTS THE INPUT ARRAY AND A POINTER TO THE FIRST CHARACTER IN THE CURRENT SYMBOL IN THAT ARRAY. A TREE STRUCTURE IS FOLLOWED IN ORDER TO CLASSIFY THE NEXT SYMBOL AS ALPHANUMERIC, INTEGER, REAL OR SPECIAL AND TO SAVE ALL OF THE CHARACTERS IN THE SYMBOL. CONTROL IS RETURNED TO LEXCAL WITH A POINTER TO THE NEXT SYMBOL IN THE INPUT ARRAY.

44.4 USAGE

CALLING SEQUENCE:

CALL GETSMR(IARY, ICHARY, NCH, ITYPE, NEXT, LIMIT)

IARY = INPUT ARRAY

ICHARY = OUTPUT ARRAY

NCH = NO. OF CHARACTERS IN SYMBOL

ITYPE = TYPE OF SYMBOL

1=ALPHANUMERIC STRING

2=INTEGER CONSTANT

3=REAL CONSTANT

4=SPECIAL CHARACTER STRING (.AND., +, -, \*, ETC.)

5=LITERAL STRING

-1=ERROR

0=NO CHARACTERS REMAINING

NEXT = POINTER TO CURRENT SYMBOL IN IARY

LIMIT = MAXIMUM NUMBER OF CHARACTERS IN IARY

COMMON USED: NONE

## MINIS PROGRAM SPECIFICATION

45.0 PROGRAM NAME- HITES

45.1 PURPOSE

TO DETERMIN IF A SPECIFIED POINT ON EARTH IS ON A PARTICULAR MERITS PHOTO.

45.2 IDENTIFICATION

DATE- 18 SEPTEMBER, 1975

REVISION- ADAPTED FROM A MERITS SUR OF THE SAME NAME.

AUTHOR- R.L.KEEFFER (205) 883-1778

COMPUTER- DATACRAFT 6024/33

LANGUAGE- FORTRAN IV

CORE SIZE- 121 LOCATIONS (171 OCTAL)

45.3 DESCRIPTION

PHOTO COVERAGE, ASSUMED TO BE 100 NAUTICAL MILES, IS COMPENSATED FOR LATITUDE AND BEARING AND COMPARED TO THE SPECIFIED POINT.

45.4 USAGE

CALLING SEQUENCE:

CALL HITES(LACP,LQCP,BEAR,LAS,LQS,IA)

LACP = LATITUDE OF PHOTO

LQCP = LONGITUDE OF PHOTO.

BEAR = BEARING OF SATELLITE.

LAS = LATITUDE OF POINT

LQS = LONGITUDE OF POINT.

IA = FLAG SET TO 1 IF POINT IN PHOTO, ELSE 0.

# MINIS PROGRAM SPECIFICATION

46.0 PROGRAM NAME- IRDEP

46.1 PURPOSE

PROVIDE A FORTRAN ROUTINE TO DEPOSIT ANY N-BIT BYTE, WHERE  
N.LE.24, IN AN INTEGER ARRAY;

46.2 IDENTIFICATION

DATE- 10 SEPTEMBER 1974

REVISION-

AUTHOR- D.R.SANDERS (205)-883-1778

COMPUTER- DCR 6024

LANGUAGE- ASSEMBLER

CORE SIZE- 76 LOCATIONS (114 OCTAL)

**ORIGINAL PAGE IS  
OF POOR QUALITY**

46.3 DESCRIPTION

THE SPECIFIED FIRST BIT POSITION IS DIVIDED BY 24 TO DETERMINE WHICH WORD(S) IN THE TARGET ARRAY IS(ARE) TO BE CHANGED, THE LAST 'NBITS' OF THE SOURCE WORD ARE DEPOSITED IN THE TARGET ARRAY BEGINNING AT BIT POSITION 'IFIR'. IF THE INSERTED BYTE EXTENDS BEYOND THE LAST BIT OF ONE WORD, THE ADDITIONAL BITS GO INTO THE NEXT WORD IN THE TARGET ARRAY.

46.4 USAGE

CALLING SEQUENCE:

CALL IRDEP(IFIR,NBITS,ITARGET,ISOURCE)

IFIR = FIRST BIT IN TARGET FOR BYTE, SEE NOTE1.

NBITS = NO. OF BITS TO BE DEPOSITED.

ITARGET=TARGET ARRAY FOR BYTE,

ISOURCE= SOURCE WORD RIGHT JUSTIFIED,

NOTE1: IFIR MAY BE USED TO DEPOSIT THE I-TH BYTE OF ANY SIZE (UP TO 24) IN THE TARGET ARRAY. EG., IF USING A BYTE SIZE OF 15 AND BYTE NUMBER 11 IS TO BE DEPOSITED, SET ARGUMENT IFIR=BYTESIZE\*(BYTENUM-1) OR IFIR=150 IN THE CALL TO IRDEP.

# MINIS PROGRAM SPECIFICATION

47.0 PROGRAM NAME- IBEXT

47.1 PURPOSE

PROVIDE A FORTRAN CALLABLE FUNCTION TO EXTRACT ANY N-BIT BYTE, WHERE N.LE.24, FROM AN INTEGER ARRAY.

47.2 IDENTIFICATION

DATE- 10 SEPTEMBER 1974

REVISION-

AUTHOR- D.R.SANDERS (205)-883-1778

COMPUTER- DCR 6024

LANGUAGE- ASSEMBLER

CORE SIZE- 60 LOCATIONS (74 OCTAL)

47.3 DESCRIPTION

THE SPECIFIED BYTE MAY BE ENTIRELY WITHIN ONE WORD OR PARTIALLY IN EACH OF TWO ADJACENT WORDS. THE FIRST WORD CONTAINING ANY OF THE BYTE IS LOCATED BY ADDING THE ARRAY ADDRESS TO THE FIRST BIT NUMBER DIVIDED BY 24. THE BYTE IS EXTRACTED FROM THE APPROPRIATE WORD(S) AND RETURNED RIGHT JUSTIFIED WITH THE UNUSED LEADING BITS ZEROED. IF ZERO OR NEGATIVE BITS ARE REQUESTED OR N.BITS.GT.24, AN ENTIRE WORD IS RETURNED.

47.4 USAGE

CALLING SEQUENCE:

IBYTE = IBEXT(IFIR,NBITS,IWORD)

IFIR = FIRST BIT OF SELECTED BYTE,(0-N) SEE NOTE1.

NBITS = NUMBER OF BITS IN SELECTED BYTE,(0-24)

IWORD = WORD ARRAY CONTAINING DESIRED BYTE.

NOTE1: IFIR MAY BE USED TO SELECT THE I-TH BYTE OF ANY SIZE (UP TO 24 BITS) IN THE ARRAY 'IWORD'. EG., IF DESIRED BYTE SIZE IS 17 AND THE BYTE NUMBER IS 13: SET ARGUMENT IFIR=BYTESIZE\*(BYTENUM-1) OR 204 IN THE CALL TO IBEXT.

## MINIS PROGRAM SPECIFICATION

48.0 PROGRAM NAME- IBLANK

48.1 PURPOSE

TEST CHARACTER INPUT BUFFER FOR A BLANK OR ZERO LINE,

48.2 IDENTIFICATION

DATE- 23 OCTOBER, 1974

REVISION-

AUTHOR- R.L.KEFFER (205)-883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN-IV

CORE SIZE- 69 LOCATIONS (105 OCTAL)

ORIGINAL PAGE IS  
POOR QUALITY

48.3 DESCRIPTION

THE FIRST 'N' CHARACTERS OF THE ARRAY ARE CHECKED FOR ZERO OR BLANK CHARACTERS. THE FUNCTION IS SET BY THE TEST.

48.4 USAGE

CALLING SEQUENCE:

IFLAG = IBLANK(IBUF,NCHAR)

IBUF = BUFFER CONTAINING 'NCHAR'  
CONTIGUOUS CHARACTERS.

NCHAR = NUMBER OF CHARACTERS TO CHECK.

IFLAG = 0 FOR A ZERO BUFFER,  
-1 FOR A BLANK BUFFER,  
+1 FOR CHARACTERS IN BUFFER.

# MINIS PROGRAM SPECIFICATION

50.0 PROGRAM NAME- INCORD

50.1 PURPOSE

PRECEDING PAGE BLANK NOT FILMED

READ AND RETURN A COORDINATE,

50.2 IDENTIFICATION

DATE- 17 SEPTEMBER, 1975

REVISION-

AUTHOR- R.L.KEFFER (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 49 LOCATIONS (61 OCTAL)

ORIGINAL PAGE IS  
OF POOR QUALITY.

50.3 DESCRIPTION

A MESSAGE IS OUTPUT PROMPTING THE USER TO ENTER A PARTICULAR COORDINATE. THE RESPONSE IS SCANNED VIA SUBROUTINE 'COORD' FROM THE MERITS SYSTEM, FOR DEGREES AND MINUTES OF THE COORDINATE. THE ENTERED COORDINATE TYPE IS CHECKED AGAINST THE REQUESTED TYPE. IF DIFFERENT, THE USER IS PROMPTED TO RE-ENTER THE COORDINATE.

50.4 USAGE

CALLING SEQUENCE:

CALL INCORD(MESNO,ITYP,MIN)

MESNO = MESSAGE NUMBER OF PROMPT,

ITYP = TYPE OF COORDINATE DESIRED

= 1 FOR LATITUDE,

= 2 FOR LONGITUDE.

MIN = COORDINATE IN MINUTES,

## MINIS PROGRAM SPECIFICATION

51.0 PROGRAM NAME- INDEX

51.1 PURPOSE

TO ALLOW THE USER TO CREATE, UPDATE OR DELETE INDEX FILES.

51.2 IDENTIFICATION

DATE- 5-28-75

REVISION-

AUTHOR- C. J. FARLESS (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 837 LOCATIONS (1505 OCTAL)

51.3 DESCRIPTION

INDEX ALLOWS THE USER TO CREATE, UPDATE, OR DELETE INDEX FILES WITH SUBROUTINE GENDEX. INDEX ASKS FOR THE FIELD NAME THE INDEX REFERENCES, PRINTS OUT A CURRENT STATUS REPORT, ASKS FOR THE REQUESTED ACTION (GENERATE, UPDATE, OR DELETE) AND THE LEVEL AND NO. OF SEGMENTS. FOR EACH ACTION, THE INPUTS ARE CHECKED FOR CORRECT USAGE. A FILE CANNOT BE GENERATED IF IT EXISTS AND CANNOT BE UPDATED OR DELETED IF IT DOES NOT EXIST. IF A FILE IS UPDATED OR DELETED FOR WHICH FILES OF HIGHER LEVELS EXIST, THE HIGHER LEVELS MUST ALSO BE UPDATED OR DELETED. AFTER THE ACTION HAS BEEN PERFORMED, A MESSAGE LISTING THE ACTION TAKEN IS PRINTED OUT. THE USER CAN THEN REQUEST EXIT (TERMINATING INDEX) OR CONTINUE (NEXT FIELD IS ASKED FOR). IF ERRORS ARE MADE WHEN ENTERING AN ACTION, INDEX WILL ASK FOR THE ACTION AGAIN. IF 0 IS ENTERED FOR LEVEL OR NUMBER OF SEGMENTS, OR CARRIAGE RETURN IS ENTERED FOR ANY OTHER REQUEST, INDEX WILL ASK FOR CONTINUE OR EXIT ACTION.

51.4 USAGE

CALLING SEQUENCE:

INDEX (NO ARGUMENTS)

DIALOGUE:

PROGRAM  
ENTER FIELD NAME:  
(FIELD NAME) HAS X INDEX LEVELS,  
LEVEL NO. OF ENTRIES

USER  
(UP TO 8 CHARACTERS)

X            XXXXX  
.  
.  
.  
X            XXXXX

OR:  
FIELD IS NOT INDEXED.

ENTER ACTION (GENERATE, UPDATE, DELETE) 'GENERATE', 'UPDATE', 'DELETE'

ENTER INDEX LEVEL:

1 OR 2

ORIGINAL PAGE IS  
OF POOR QUALITY

IF 2:

ENTER NUMBER OF SEGMENTS:

NUMBER OF SEGMENTS

IF GENERATE OR UPDATE LEVEL 1:

ENTER WORK AREA (DISC OR TAPE): 'DISC' OR 'TAPE'

IF TAPE:

ENTER DEVICE NUMBER:

DEVICE NUMBER ASSIGNED TO MT

AFTER ACTION HAS BEEN COMPLETED:

GENERATED INDEX LEVEL X FOR (FIELD NAME) NO. OF ENTRIES=XXXXXX

UPDATED OR

DELETED

EXIT OR CONTINUE

'EXIT' OR 'CONTINUE'

FOR CONTINUE: PROGRAM RETURNS TO 'ENTER FIELD NAME'

FOR EXIT: EXIT INDEX

## MINIS PROGRAM SPECIFICATION

52.0 PROGRAM NAME- ININT

52.1 PURPOSE

REQUEST AND ACCEPT AN INTEGER FROM THE USER'S TERMINAL.

52.2 IDENTIFICATION

DATE- 23 OCTOBER, 1974

REVISION-

AUTHOR- R.L.KEEFER (205)-883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN-IV

CORE SIZE- 60 LOCATIONS (74 OCTAL)

52.3 DESCRIPTION

THE CALLER SPECIFIED MESSAGE IS OUTPUT REQUESTING THE USER TO INPUT AN INTEGER. THE INPUT IS ACCEPTED AND CONVERTED TO AN INTEGER VIA 'INTEGER'. ERRORS IN THE INPUT STREAM ARE CITED AND THE REQUEST MESSAGE IS REPEATED. A BLANK INPUT BUFFER RETURNS A MINUS ONE. A LIMIT IS IMPOSED ON THE INTEGER RETURNED.

52.4 USAGE

CALLING SEQUENCE:

INT = ININT(MESNO,MAXVAL)

MESNO = MESSAGE NUMBER OF REQUEST.

MAXVAL = MAXIMUM INTEGER VALUE RETURNED.

INT = INTEGER VALUE RESULT.

= -1 IF INPUT BUFFER IS BLANK

# MINIS PROGRAM SPECIFICATION

53.0 PROGRAM NAME- INITIAL

53.1 PURPOSE

INITIALIZE INPUT DATA FOR CREATE,

ORIGINAL PAGE IS  
OF POOR QUALITY.

53.2 IDENTIFICATION

DATE- 2-19-76

REVISION-

AUTHOR- C.J. FARLESS (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 560 LOCATIONS (1060 OCTAL)

53.3 DESCRIPTION

THE USER IS ASKED WHETHER HE WISHES TO EXECUTE ON A DEFAULT MODE, OR MANUAL MODE (ENTERING THE INPUT INFORMATION). DEFAULT WILL READ THE INPUT INFORMATION AS THE FIRST RECORD OF THE INPUT DECK (INPUTD SRC-DISC FILE), MANUAL WILL REQUEST THE USER TO ENTER THE INFO CARD; THE FORMAT OF THE CARD IS AS FOLLOWS:

COL:1 = I, U, OR M (INITIALIZATION, UPDATE, MODIFY)  
COL:2 = L IF LABEL RECORD ON RAW DATA TAPE  
COL:8 = INFO. AND OP-CARD DECKS INPUT DEVICE D-DISC, C-CARD READER  
COL:9 = RAW DATA INPUT DEVICE--C, T, OR D (CARDS, TAPE, DISC)  
COL:10-18 = FILE NAME AND EXTENSION (FOR DISC ENTRY ONLY)  
COL:19-24 = NO. OF BYTES IN THE RAW DATA PHYSICAL RECORD,  
COL:25-30 = NO. OF BYTES IN THE RAW DATA LOGICAL RECORD,  
COL:31-32 = NO. OF BITS IN THE KEY VALUE (MAXIMUM OF 40)  
COL:33-38 = NO. OF RAW DATA RECORDS,  
COL:40-79 = BIT PATTERN OF KEY VALUE,

MERITS DEFAULT INFO. CARD:

=0L DT 2048 16432 111110100000000000000000

IF THE NO. OF RAW DATA RECORDS IS LEFT BLANK, THE USER WILL BE ASKED TO INPUT THEM.

AFTER THE DATA IS DECODED, THE INFO. DECK IS READ IN AND THE OP CARD DECK IS READ IN. THEN THE QUADS ARE FORMED AND CONTROL IS RETURNED TO CREATE.

53.4 USAGE

CALLING SEQUENCE:

CALL INITIAL(IRR)

IRR=ERROR FLAG =1 FOR INPUT ERROR

## MINIS PROGRAM SPECIFICATION

54.0 PROGRAM NAME- INSERT

54.1 PURPOSE

TRIGGER THE INSERTION OF SAVED TEXT.

54.2 IDENTIFICATION

DATE- 2 DECEMBER 1975

REVISION-

AUTHOR- D.R.SANDERS

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE-

54.3 DESCRIPTION

'INSERT' PLACES THE SYMBOL TABLE POINTER FOR THE SAVED TEXT'S SYMBOL NAME IN CELL 5 OF THE COMTAB COMMON AREA.

54.4 USAGE

CALLING SEQUENCE:

INSERT (TEXTNAME)

TEXTNAME = NAME OF SAVED TEXT TO BE INSERTED

ORIGINAL PAGE #  
OF POOR QUALITY

# MINIS PROGRAM SPECIFICATION

55.0 PROGRAM NAME- INTERP

55.1 PURPOSE

INTERPRET COMMANDS THAT APPEAR IN THE QUAD TABLE.

55.2 IDENTIFICATION

DATE- 24 JANUARY 1975

REVISION-

AUTHOR- D.R.SANDERS

COMPUTER- DCR 6024

LANGUAGE- FORTRAN IV

CORE SIZE- 369 LOCATIONS (561 OCTAL)

55.3 DESCRIPTION

A POINTER TO THE FIRST QUAD IN A COMMAND IS SENT AS AN ARGUMENT TO THIS ROUTINE. 'NEXT' IS SET TO THE NEXT ZERO QUAD. IN THE IMMEDIATE MODE THE COMMAND IS AUTOMATICALLY EXECUTED. A 'DELAYEX' COMMAND CAUSES THE IMMEDIATE MODE FLAG TO BE RESET. IN THE DELAYED MODE ONLY IMMEDIATE SUBROUTINE CALLS ARE MADE. ALL OTHER COMMANDS ARE IGNORED UNTIL AN 'EXECUTE' COMMAND IS ENCOUNTERED.

55.4 USAGE

CALLING SEQUENCE:

CALL INTERP(IQST, NEXT)

IQST = POINTER TO STARTING QUAD  
NEXT = NEXT AVAILABLE QUAD.

## MINIS PROGRAM SPECIFICATION

56.0 PROGRAM NAME- INTGER

56.1 PURPOSE

CONVERT A CHARACTER STRING TO AN INTEGER.

56.2 IDENTIFICATION

DATE- 23 OCTOBER, 1974

REVISION-

AUTHOR- R.L.KEEFFER (205)-883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN-IV

CORE SIZE- 103 LOCATIONS (147 OCTAL)

ORIGINAL PAGE IS  
OF POOR QUALITY

56.3 DESCRIPTION

'NDIGIT' CHARACTERS ARE EXAMINED TO BUILD AN INTEGER VALUE FROM 'IARRAY'. BLANKS ARE IGNORED. A MINUS SIGN ANYWHERE IN THE FIELD NEGATES THE RESULT EXCEPT IN CASE OF SUBSEQUENT ERRORS. ANY OTHER NON-NUMERIC CHARACTER CAUSES AN ERROR RETURN.

56.4 USAGE

CALLING SEQUENCE:

INT = INTGER(IARRAY,NDIGIT,IFIRST,IERROR)

INT = INTEGER RESULT.

IARRAY = ARRAY CONTAINING CHARACTER STRING.

NDIGIT = NUMBER OF DIGITS IN STRING.

IFIRST = FIRST CHARACTER POSITION IN STRING.

IERROR = ERROR FLAG SET AS FOLLOWS:  
0 FOR SUCCESSFUL CONVERSION.  
1 FOR INVALID CHARACTER IN STRING.

# MINIS PROGRAM SPECIFICATION

58.0 PROGRAM NAME- KSCAN

58.1 PURPOSE

TO SCAN AN INPUT ARRAY AND RETURN THE PROPER CODE FOR THE CHARACTER FOUND (FOR USE BY SUBROUTINE COORD),

58.2 IDENTIFICATION

DATE- 29 AUGUST 1973

REVISION- FORMER MERITS PROGRAM

AUTHOR- M.E.GILLIS (205) 453-5230

COMPUTER- DATACRAFT 6024

LANGUAGE- FORTRAN

CORE SIZE- 75 LOCATIONS (113 OCTAL)

ORIGINAL PAGE IS  
OF POOR QUALITY

58.3 DESCRIPTION

EACH CHARACTER IS COMPARED AGAINST THE ALLOWED LIST. IF FOUND, THE FUNCTION IS SET TO THE CORRESPONDING CODE. IF NOT FOUND, THE FUNCTION IS SET TO 9 (ABORT)

THE ALLOWED CHARACTERS AND CODES ARE:

T	1
W	1
N	1
-	2
S	2
E	2
.	3
M	3
D	4
SPACE	4
,	4
NUMBERS	5
.	6
/	4
OTHER	9

PRECEDING PAGE BLANK NOT FILMED

58.4 USAGE

N = KSCAN(K)

K = CHARACTERS(EXTRACTED BY IXTR)

N = CODE PER ABOVE TABLE

# MINIS PROGRAM SPECIFICATION

59.0 PROGRAM NAME- KUMPAR

59.1 PURPOSE

COMPARE TWO BYTE STRINGS OF ANY LENGTH BEGINNING AT ANY CHARACTER POSITION.

59.2 IDENTIFICATION

DATE- 23 OCTOBER, 1974

REVISION-

AUTHOR- R.L.KEEFER (205)-883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN-IV

CORE SIZE- 54 LOCATIONS (66 OCTAL)

59.3 DESCRIPTION

'N' BYTES OF THE TWO CHARACTER STRINGS ARE COMPARED ONE AT A TIME VIA 'IXTR'. THE FUNCTION IS SET BY THE COMPARISON.

59.4 USAGE

CALLING SEQUENCE:

IFLAG = KUMPAR(IARY, IB1, JARY, JB1, NBYT)

IARY = FIRST CHARACTER ARRAY.

JARY = SECOND CHARACTER ARRAY.

IB1 = FIRST CHARACTER POSITION IN IARY.

JB1 = FIRST CHARACTER POSITION IN JARY.

NBYT = NUMBER OF BYTES TO COMPARE.

IFLAG = FUNCTION VALUE UPON RETURN SET  
AS FOLLOWS:

0 FOR BOTH STRINGS EQUAL.

+1 FOR IARY ALPHABETICALLY BEYOND JARY.

-1 FOR IARY ALPHABETICALLY BEFORE JARY.

# MINIS PROGRAM SPECIFICATION

60.0 PROGRAM NAME- LEXCAL

60.1 PURPOSE

CONVERT AN INPUT LINE INTO INTERNAL FORM FOR ANALYSIS BY SUBROUTINE PARSE.

60.2 IDENTIFICATION

DATE- 13 NOVEMBER 1974

REVISION-

AUTHOR- D.R. SANDERS (205)-883-1778

COMPUTER- DCR 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 131 LOCATIONS (203 OCTAL)

ORIGINAL PAGE IS  
OF POOR QUALITY

60.3 DESCRIPTION

LEXCAL CALLS GETSMB TO BUILD THE NEXT SYMBOL FROM AN INPUT LINE, CALLS CHKSYM TO INSURE ITS EXISTENCE IN THE SYMBOL TABLE OR RESERVED WORD LIST AND TO PRODUCE AN INTERNAL INTEGER CODE, AND CALLS MAKSEN TO STORE THE CODE IN 'ISEN', THIS PROCESS IS REPEATED FOR EVERY SYMBOL IN THE INPUT LINE, AN ERROR CHECK IS MADE ON RETURNING FROM EACH SUBROUTINE CALLED BY LEXCAL.

60.4 USAGE

CALLING SEQUENCE:

CALL LEXCAL(IARY, ISEN, LAST, IRESWD, IER)

IARY = INPUT ARRAY

ISEN = INTERNAL SENTENCE ARRAY

LAST = LOCATION OF LAST ENTRY IN 'ISEN'

IRESWD = RESERVED WORD LIST

IER = ERROR CODES

1 = ILLEGAL SYMBOL

2 = TOO MANY SYMBOLS IN SENTENCE

3 = TOO MANY CHARACTERS IN SENTENCE

4 = SYMBOL TABLE OVERFLOW

5 = CONSTANT TABLE OVERFLOW

6 = ILLEGAL CONSTANT

7 = CONSTANT TABLE OVERFLOW

## MINIS PROGRAM SPECIFICATION

61.0 PROGRAM NAME- LININ

61.1 PURPOSE

READ A LINE AND RETURN REQUESTED NUMBER OF CHARACTERS,

61.2 IDENTIFICATION

DATE- 23 OCTOBER, 1974

REVISION-

AUTHOR- R.L.KEEFER (205)-883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN-IV

CORE SIZE- 88 LOCATIONS (130 OCTAL)

61.3 DESCRIPTION

A LINE IS READ IN FROM THE NORMAL INPUT DEVICE AND THE FIRST 'N' CHARACTERS ARE TRANSFERRED TO THE CALLER'S ARRAY. IF ZERO CHARACTERS WERE REQUESTED, NO READ TAKES PLACE, THE COMMON BUFFER USED FOR INPUT MAY ALSO BE DESTINATION BUFFER.

61.4 USAGE

CALLING SEQUENCE:

CALL LININ(IBUFR,NCHAR)

IBUFR = DESTINATION ARRAY FOR READ,

NCHAR = NUMBER OF CHARACTERS TO READ,

# MINIS PROGRAM SPECIFICATION

62.0 PROGRAM NAME- LINMES

62.1 PURPOSE

INPUT A LINE (THAT IS TERMINATED BY A NULL CHARACTER) FROM THE USER MESSAGE FILE.

62.2 IDENTIFICATION

DATE- 2 DECEMBER 1975

REVISION-

AUTHOR- D.R. SANDERS

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 98 LOCATIONS (142 OCTAL)

ORIGINAL PAGE IS  
OF POOR QUALITY

62.3 DESCRIPTION

IF 'ISTART' IS ONE, THE BYTE COUNT IS READ FROM THE USER MESSAGE FILE AND SAVED IN A DUMMY ARGUMENT. IN ANY CASE, 'IRECLN' BYTES ARE READ INTO 'IBUFR' AND THE BUFFER IS SCANNED UNTIL A NULL CHARACTER IS FOUND. BLANKS ARE STORED FROM THAT POINT TO THE END OF 'IBUFR', AND THE NEXT AVAILABLE BYTE IN THE USER MESSAGE FILE IS STORED IN 'IBYTE'.

62.4 USAGE

CALLING SEQUENCE:

CALL LINMES (IBUFR, IRECLN, MESNO, IRYTE, NRYTE)

IBUFR = INPUT BUFFER (LARGE ENOUGH TO HOLD 'IRECLN' BYTES)  
IRECLN = MAXIMUM LINE SIZE IN BYTES  
MESNO = USER MESSAGE NUMBER  
IRYTE = BEGINNING BYTE POINTER IN MESSAGE FILE  
(INPUT 1 TO INITIALIZE THE FILE READ)  
(OUTPUTS POINTER TO NEXT AVAILABLE BYTE)  
(OUTPUTS -1 TO INDICATE END OF FILE)  
NRYTE = NUMBER OF BYTES IN MESSAGE  
(NOT TO BE DISTURBED BY CALLING PROGRAM)

## MINIS PROGRAM SPECIFICATION

63.0 PROGRAM NAME- LSERCH

63.1 PURPOSE

SCAN INDEX FILES FOR PHOTOS WITH COORDINATES WITHIN SPECIFIED AREA.

63.2 IDENTIFICATION

DATE- 17 SEPTEMBER, 1975

REVISION- ADAPTED FROM MERITS SUB OF SAME NAME

AUTHOR- R.L.KEEFER (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 90 LOCATIONS (132 OCTAL)

63.3 DESCRIPTION

THE AREA DEFINED BY THE CALLING ARGUMENTS, OR BY THE CALLING POINT PLUS OR MINUS A TOLERANCE, IS USED TO SEARCH THE 'LATLON' INDEX FOR POSSIBLE PHOTO COVERAGE. IF THE SEARCH IS FOR A POINT, THE SEARCH IS PERFORMED FOR A SQUARE AROUND THE POINT.

63.4 USAGE

CALLING SEQUENCE:

CALL LSERCH(LAL,LAH,LML,LNH,NPOS,LINDEX)

LAL,LAH,LML,LNH = LOWER AND UPPER LATITUDE AND SMALLER AND LARGER LONGITUDE,

NPOS = NUMBER OF PHOTOS FOUND IN AREA.

LINDEX = ARRAY CONTAINING INDEX OF LATITUDES,

# MINIS PROGRAM SPECIFICATION

64.0 PROGRAM NAME- MAKSEN

64.1 PURPOSE

ACCUMULATE INTERNAL SENTENCE ARRAY TO BE PASSED TO SUBROUTINE PARSE,

64.2 IDENTIFICATION

DATE- 11 NOVEMBER 1974

REVISION-

AUTHOR- D.R. SANDERS

COMPUTER- DCR 6024

LANGUAGE- FORTRAN IV

CORE SIZE- 31 LOCATIONS (37 OCTAL)

ORIGINAL PAGE IS  
OF POOR QUALITY

64.3 DESCRIPTION

MAKSEN CHECKS THE NUMBER OF SYMBOLS IN THE SENTENCE ALREADY, AND IF THERE IS STILL ROOM IT ADDS THE NEXT SYMBOL TO THE SENTENCE ARRAY,

64.4 USAGE

CALLING SEQUENCE:

CALL MAKSEN(ISEN, LAST, IENTRY, IER)

ISEN = SENTENCE ARRAY

LAST = NUM. OF LAST LOCATION USED IN 'ISEN'

IENTRY = INTERNAL FORM OF NEXT SYMBOL

IER = ERROR FLAG(0 IF OK, 2 IF OVERFLOW)

## MINIS PROGRAM SPECIFICATION

65.0 PROGRAM NAME- MERGER

65.1 PURPOSE

MERGE BLOCKS PREVIOUSLY SORTED BY 'BISORT' TO CREATE ONE SORTED FILE.

65.2 IDENTIFICATION

DATE- 29 APRIL, 1975

REVISION-

AUTHOR- R.L.KEEFFER (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 1358 LOCATIONS (2516 OCTAL)

65.3 DESCRIPTION

THE DISC FILE 'IFILIN' IS ASSUMED TO BE BROKEN INTO 'NSEG' SEGMENTS OF 'IPSEG' ITEMS EACH. THE WORKING BUFFER 'BUF' IS BROKEN INTO 'NSEG' SECTIONS OF 'NITM' ITEMS EACH, ( $NITM = LBUF / ITMSZ / NSEG$ , WHERE 'ITMSZ' =  $IFILIN(5) / NRW$  AND  $NRW = N\theta$ , OF BYTES PER WORD). A POINTER (ICUR(I)) IS MAINTAINED FOR EACH CORE SEGMENT INDICATING THE CURRENT ITEM OF THAT SEGMENT TO BE MERGED WITH THE CURRENT ITEM OF EACH OF THE OTHER 'NSEG' SEGMENTS. AN ITEM NUMBER 'ITEMNO(I)' IS MAINTAINED FOR EACH FILE SEGMENT INDICATING THE FIRST ITEM NUMBER OF THE NEXT 'NITM' ITEMS TO BE READ WHEN THE CORRESPONDING CORE SEGMENT IS EMPTIED.

MERGING IS ACCOMPLISHED VIA A THREADED BUFFER POINTER WHICH MAINTAINS A POINTER TO THE CORE SEGMENT CONTAINING THE NEXT SMALLEST ITEM. THE SMALLEST ITEM (SMALLEST MEANING LOWEST MAGNITUDE SORT KEY) IS MOVED TO AN OUTPUT BUFFER. THE POINTER TO THE SEGMENT PROVIDING THE SMALLEST ITEM IS ADJUSTED TO SELECT THE NEXT ITEM. THAT ITEM IS COMPARED TO THE CURRENT ITEMS OF THE OTHER CORE SEGMENTS, VIA THE THREADED BUFFER POINTERS 'THREAD(I)', TO DETERMINE ITS POSITION IN THE SORTED LIST, WHEN ITS POSITION IS ESTABLISHED, THE THREAD IS SPLICED TO LINK IN THE N-W ITEM.

AS EACH NEW ITEM BECOMES CURRENT, ITS SORT KEY IS EXTRACTED, POSITIONED, AND PUT IN A SORT KEY BUFFER FOR CONVENIENCE IN SUBSEQUENT COMPARISONS.

EACH TIME THE LAST ITEM IN A CORE SEGMENT IS MOVED TO THE OUTPUT BUFFER AN I/O CALL IS MADE TO REFILL THE CORE SEGMENT. THE LAST READ OF A SEGMENT MAY NOT FILL THE CORE BUFFER AVAILABLE TO THAT SEGMENT. IN SUCH CASE, THE UNFILLED PORTION OF THE BUFFER IS ZEROED.

SIMILIARLY, EACH TIME THE OUTPUT BUFFER IS FILLED, IT IS WRITTEN

**MINIS PROGRAM SPECIFICATION  
TO THE SPECIFIED OUTPUT FILE OR DEVICE.**

**65.4 USAGE**

**CALLING SEQUENCE:**

**CALL MERGER(IFILIN,IFILOT,IOUT,K1,K2,NSEG,IPSEG,BUF,LBUF)**

**IFILIN(1-5) = INPUT FILE DEFINITION ARRAY AS FOLLOWS:**  
    **(1-3) = FILE NAME (3A2)**  
    **(4) = NO. OF ITEMS IN FILE,**  
    **(5) = NO. OF 8-BIT BYTES PER ITEM.**  
**IFILOT(1-5) = OUTPUT FILE DEFINITION AS ABOVE,**  
**K1 = FIRST BIT NO. OF SORT KEY,**  
**K2 = LAST BIT NO. OF SORT KEY,**  
**NSEG = NO. OF SEGMENTS (SORTED ARRAYS) IN FILE,**  
**IPSEG = NO. OF ITEMS PER SEGMENT,**  
**BUF = WORKING BUFFER,**  
**LRUF = LENGTH OF WORKING BUFFER**  
**IOUT = OUTPUT DEVICE NO. (0 FOR DISK)**

**ORIGINAL PAGE IS  
OF POOR QUALITY**

## MINIS PROGRAM SPECIFICATION

66.0 PROGRAM NAME- MESH

66.1 PURPOSE

CREATE A MESHED CHARACTER FROM A FILE DEFINITION THAT WILL BE UNIQUE FOR EACH DATA BASE.

66.2 IDENTIFICATION

DATE- 11-26-75

REVISION-

AUTHOR- C.J. FARLESS (205) 883-1778

COMPUTER- DATACRAFT 602473

LANGUAGE- FORTRAN IV

CORE SIZE- 50 LOCATIONS (62 OCTAL)

66.3 DESCRIPTION

THE CHARACTERS IN THE REQUESTED FILE DEFINITION(NAM) ARE MESHED USING THE FOLLOWING ALGORITHM:

$X = (X \text{ .XOR. CHAR} ) \text{ .ROTAT. } 1$

$X = X \text{ .AND. } 1377$

66.4 USAGE

CALLING SEQUENCE:

$X = \text{MESH}(\text{NAM})$

$X =$  INTEGER RETURN OF CHARACTER  
 $\text{NAM} =$  FILE DEFINITION

# MINIS PROGRAM SPECIFICATION

67.0 PROGRAM NAME- MFSMAN

67.1 PURPOSE

PROVIDE CAPABILITY TO ADD, DELETE, REPLACE, LIST, PACK, OR CHANGE NUMBERS OF USER MESSAGES.

67.2 IDENTIFICATION

DATE- 21 NOVEMBER, 1974

REVISION-

AUTHOR- R.L.KEFFER (205)-883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 1667 LOCATIONS (3203 OCTAL)

ORIGINAL PAGE IS  
OF POOR QUALITY

67.3 DESCRIPTION

'MESMAN' MAY BE ENTERED IN A CONVERSATIONAL MODE FOR ANY OF THE MESSAGE MANIPULATION FEATURES, OR UNDER PROGRAM CONTROL TO DELETE OR WRITE ANY ONE MESSAGE. 'ADD' MAY ADD A SPECIFIED MESSAGE OR TAKE THE FIRST AVAILABLE MESSAGE NUMBER. IN REPLACE OR WRITE OPERATIONS THE MESSAGE IS WRITTEN IN PLACE OF THE PRIOR MESSAGE IF ROOM IS AVAILABLE. THE CALLING OPERATION TYPE CODE IS CHECKED FOR A LEGAL NUMBER. IF LEGAL, THE APPROPRIATE OPERATION IS STARTED. (ICP= 1-ADD, 2-REPLACE, 3-DELETE, 4-EXIT, 5-PACK, 6-LIST, 7-CHANGE, 8-WRITE, OTHER-ENTER DIALOG) ILLEGAL OP CODES ENTER THE DIALOG ARGUMENT SETUP SECTION. THE ADD OPERATION, AFTER CHECKING FOR OR ASSIGNING A LEGAL MESSAGE NUMBER, REQUESTS AND ACCEPTS THE NEW MESSAGE INTO SCRATCH COMMON 'REUSE'. EXCESS BLANKS ARE PACKED OUT AND THE MESSAGE, IT'S NUMBER, AND IT'S LENGTH ARE TRANSFERRED TO THE MESSAGE FILE. UPON INPUT OF A BLANK LINE THE MESSAGE FILE LENGTH IS ADJUSTED AND THE DIALOG IS RESUMED.

FOR 'REPLACE', THE OLD MESSAGE IS LOCATED AND SIZED. THE NEW MESSAGE IS INPUT AND MEASURED TO DETERMINE IF IT WILL FIT IN THE OLD MESSAGE AREA. THE OLD MESSAGE IS DELETED AND THE NEW MESSAGE REPLACES IT OR ADDS TO THE END OF THE MESSAGE FILE.

THE 'WRITE' OPERATION IS THE SAME AS THE REPLACE OPERATION EXCEPT THE NEW MESSAGE IS IN THE CALLING STRING. ALSO, IF THE FIRST WORD OF THE NEW MESSAGE IS ZERO, THE OLD MESSAGE IS DELETED WITHOUT REPLACEMENT. THERE IS NO USER INTERACTION FOR THE WRITE OPERATION.

THE 'DELETE' OPERATION CHECKS FOR A LEGAL MESSAGE NUMBER, ZEROS

OUT THE MESSAGE AND THE INDEX ENTRY, AND OUTPUTS AN ACKNOWLEDGMENT MESSAGE.

'EXIT' CLOSSES THE MESSAGE FILE AND EXITS THE ROUTINE.

'PACK' DUBS IN MESSAGE NUMBERS FOR EACH ENTRY OF THE INDEX, READS AND REWRITES THE MESSAGE FILE, SKIPPING ALL ZERO LOCATIONS, AND REGENERATES THE INDEX BY RESCANNING THE MESSAGE FILE.

'LIST' OUTPUTS ALL MESSAGES AND THEIR NUMBERS BETWEEN THE SPECIFIED LIMITS.

'CHANGE' REPLACES THE GIVEN OLD MESSAGE NUMBER WITH THE NEW NUMBER AND MOVES THE INDEX POINTER ACCORDINGLY.

#### 67.4 USAGE

##### CALLING SEQUENCE:

CALL MESMAN(BUFFER, MESNO, NCHAR, IOPTN)

  BUFFER = 80 CHAR MESSAGE BUFFER FOR 'WRITE'.

  MESNO = MESSAGE TO BE AFFECTED.

  NCHAR = NO. CHARACTERS IN BUFFER FOR 'WRITE'.

  IOPTN = USER SELECTED OPERATION SET AS FOLLOWS:

    = 1 FOR ADD

      2 FOR REPLACE

      3 FOR DELETE

      4 FOR EXIT

      5 FOR PACK

      6 FOR LIST

      7 FOR CHANGE

      8 FOR WRITE

    ANY OTHER VALUE TO ENTER DIALOG.

## MINIS PROGRAM SPECIFICATION

68.0 PROGRAM NAME- MESOUT

68.1 PURPOSE

OUTPUT SYSTEM OR USER MESSAGES ON USER'S TERMINAL

68.2 IDENTIFICATION

DATE- 11 NOVEMBER, 1974

REVISION-

AUTHOR- R.L. KEEFER (205)-883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 619 LOCATIONS (1153 OCTAL)

ORIGINAL PAGE IS  
OF POOR QUALITY

68.3 DESCRIPTION

THE CALLING ARGUMENT IS TESTED TO DETERMINE IF THE MESSAGE IS A SYSTEM MESSAGE (ARG > 1000) OR A USER MESSAGE (1-1000). IF THE MESSAGE IS A SYSTEM MESSAGE, THE SYSTEM FILE INDEX IS READ FOR THE MESSAGE START LOCATION IN THE FILE. FOR USER MESSAGES, THE INDEX ENTRY IS IN CORE. THE START OF THE MESSAGE IS CHECKED TO DETERMINE THE MESSAGE LENGTH (FIRST WORD OF MESSAGE) AND THE LEFT HAND MARGIN (NUMBER OF LEADING BLANKS). THE MESSAGE IS ALTERNATELY READ INTO TWO 150 BYTE BUFFERS WITH THE NEXT BUFFER BEING FILLED EACH TIME THE CURRENT LINE EXTENDS BEYOND THE CURRENT BUFFER. EACH LINE TO BE OUTPUT IS SCANNED TO FIND THE NUMBER OF BLANKS AND THE LAST NON-BLANK CHARACTER. THE LINE IS TRANSFERRED TO THE I/O COMMON BUFFER WITH BLANKS INSERTED TO RIGHT-JUSTIFY ALL BUT THE LAST LINE OF EACH MESSAGE.

68.4 USAGE

CALLING SEQUENCE:

CALL MESOUT(MESNUM)

MESNUM = MESSAGE NUMBER TO BE OUTPUT

= 1-1000 FOR USER SUPPLIED MESSAGES

= 1001 AND UP FOR SYSTEM MESSAGES,

## MINIS PROGRAM SPECIFICATION

69.0 PROGRAM NAME- MINDEX

69.1 PURPOSE

TO CREATE THE LATLON FILE AND UINDEX FILE FOR A MERITS DATA BASE CREATE RUN.

69.2 IDENTIFICATION

DATE- 3-9-76

REVISION-

AUTHOR- C.J.FARLESS (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 576 LOCATIONS (1100 OCTAL)

69.3 DESCRIPTION

ASSUMING THAT THE DATA BASE HAS BEEN MERGED AND IS READY FOR USE, LAT AND LON ARE EXTRACTED FROM EACH DATA BASE RECORD MAKING ONE LATLON RECORD FOR EACH DATA BASE RECORD. LAT IS BIASED BY 5400 MINUTES(90 DEGREES) AND ALL VALUES OF LAT LESS THAN 0 OR GREATER THAN 10800 ARE SET TO 5400 MINUTES. LON IS BIASED BY 10800 MINUTES(180 DEGREES) AND ALL VALUES OF LON LESS THAN 0 OR GREATER THAN 21600 ARE SET TO 10800 MINUTES. LAT IS DIVIDED INTO 2 WORDS, LATD(LATITUDE IN DEGREES) AND LATM(LATITUDE MINUTES). THESE VALUES ALONG WITH THE RECORD NUMBER ARE PACKED INTO 48 BITS AS FOLLOWS:

BIT 0 = 0  
BIT 1-8 = LATD  
BIT 9-23 = LON  
BIT 24-29 = LATM  
BIT 30-47 = RECORD NUMBER-1

LAT MUST BE FIELD NUMBER 4 AND LON MUST BE FIELD NUMBER 5. THE RESULTANT FILE IS SORTED ON BITS 0-23 AND STORED ON DISC AS THE LATLON FILE.

THE UINDEX FILE CONTAINS 180 ENTRIES EACH CORRESPONDING TO THE ENTRY NUMBER-1 NUMBER OF DEGREES. THE LATLON FILE IS SEARCHED, EXAMINING LATD FOR THE FIRST RECORD OF EACH DEGREE INCREMENTATION. THIS RECORD NUMBER IS WRITTEN AS THE ENTRY OF THE FILE. IF THERE IS NO LATLON RECORD OF A CERTAIN NUMBER OF DEGREES, THE ENTRY IS SET TO THE RECORD NUMBER OF THE LAST ENTRY.

UINDEX RECORD 0 = 0 DEGREES ENTRY=RECORD NUMBER 0 1 =1 DEGREE

=RECORD OF THE FIRST LATLON RECORD OF 1 DEGREE LATITUDE . . . . .  
. 178 =178 DEGREES RECORD NUMBER OF THE FIRST RECORD WITH 178  
DEGREES LATITUDE, 179 =NUMBER OF LATLON RECORDS+1

69.4 USAGE

CALLING SEQUENCE

CALL MINDEX(IWDEV)

IWDEV=SCRATCH AREA DEVICE NUMBER 0=DISC OTHER =TAPE

**ORIGINAL PAGE IS  
OF POOR QUALITY**

## MINIS PROGRAM SPECIFICATION

70.0 PROGRAM NAME- MOVOUT

70.1 PURPOSE

MOVE THE GIVEN ITEM TO AN OUTPUT BUFFER AND WRITE THE OUTPUT BUFFER IF FULL.

70.2 IDENTIFICATION

DATE- 29 APRIL, 1975

REVISION-

AUTHOR- R.L.KEFFER (205)-883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 192 LOCATIONS (300 OCTAL)

70.3 DESCRIPTION

THE PRESENTED ITEM IS TRANSFERRED TO THE OUTPUT BUFFER. IF THE OUTPUT BUFFER IS FULL, IT IS WRITTEN TO THE SPECIFIED OUTPUT DEVICE(IOUT). IF THE NUMBER OF WORDS PER ITEM, 'NWDS', IS EQUAL TO ZERO, THE CURRENT CONTENTS OF THE BUFFER ARE WRITTEN AND THE FILE IS CLOSED (FOR DISK) OR AN END OF FILE IS WRITTEN (OTHER DEVICES). THE ITEM COUNT (KOUNT) IS INCREMENTED FOR EACH ITEM TRANSFERRED AND RESET TO ONE AFTER EACH BUFFER WRITE. IF THE NUMBER OF WORDS PER ITEM IS NEGATIVE, THE ROUTINE IS INITIALIZED.

70.4 USAGE

CALLING SEQUENCE:

CALL MOVOUT(ITEM,IARGS,BUFF)

ITEM = ARRAY TO BE MOVED TO BUFFER,

BUFFER = OUTPUT BUFFER,

IARGS = WORD ARGUMENT ARRAY SET AS FOLLOWS:

(1-5) = OUTPUT FILE CONTROL BLOCK, WORDS

1-3=FILE NAME(3A2) WORD 4=NO. OF  
ITEMS IN FILE, WORD 5=NO. OF 8-BIT  
BYTES PER ITEM.

(6) = IOUT, OUTPUT DEVICE ASSIGNMENT,  
0 FOR DISK.

(7) = NWDS, NUMBER OF WORDS PER ITEM,  
0 TO WRITE LAST BUFFER AND AN EOF,

-NO. OF WORDS PER ITEM TO INITIALIZE,  
(8) = MXITM, MAXIMUM NUMBER OF ITEMS TO

STORE IN 'BUFFER', SET BY USER  
PRIOR TO FIRST CAL. SET BY MOVOUT  
TO TOTAL NO. OF ITEMS MOVED  
AFTER AN EOF CALL.

ORIGINAL PAGE IS  
OF POOR QUALITY

## MINIS PROGRAM SPECIFICATION

71.0 PROGRAM NAME- NAMLIST

71.1 PURPOSE

TO CREATE AND MANAGE NAMLIST, HEADER, FORMAT, AND SAVETEXT FILES,

71.2 IDENTIFICATION

DATE- 11-15-75

REVISION-

AUTHOR- C.J. FARLESS (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE-

71.3 DESCRIPTION

THIS PROGRAM MAKES THE NECESSARY ASSIGNMENTS TO USE THE MESSAGE MANAGER TO CREATE AND MAINTAIN TITLE FILES. A FILE CONTAINING THE TITLES, THE CORRESPONDING SYMBOL TYPE, AND THE ASSOCIATED MESSAGE NUMBER IS ALSO MAINTAINED. THIS FILE CONTAINS, ALONG WITH THE NAMLIST TITLES, THE SAVETEXT TITLES AND THE FORMAT TITLES (FOR BOTH HEADERS AND DATA FORMATS). EACH RECORD OF THE FILE HAS 6 WORDS IN IT. WORDS 1 THRU 4 CONTAIN AN 8 CHARACTER (MAXIMUM) TITLE, WORD 5 CONTAINS THE SYMBOL TYPE. WORD 6 CONTAINS THE MESSAGE NUMBER (201-300). A TITLE CAN BE ADDED TO THE FILE, REPLACED, BE DELETED FROM THE FILE, OR MODIFIED. THESE FUNCTIONS ARE PERFORMED BY THE MESSAGE MANAGER (MESMAN).

71.4 USAGE

CALLING-SEQUENCE:

(USER OF OLDMAN) NAMLIST  
HEADER  
FORMAT  
SAVETEXT

REQUIRED FILES:

FILE FOR TITLES:  
FILE DEFINITION:

XTITLE (X IS THE MESHED CHARACTER)  
IDFF(21)-IDFF(25)  
IDFF(21-23)=XTITLE  
IDFF(24)=NO. OF RECORDS

FILE FORMAT:

IDEF(25)=6\*NBW  
1 RECORD--6 WORDS  
WORDS(1-4)--NAMELIST,SAVETEXT,OR  
FORMAT TITLE  
WORD 5--SYMBOL TYPE  
11=NAMELIST  
12=HEADER FORMAT  
13=DATA FORMAT  
14=SCHEMA  
WORD 6--MESSAGE NUMBER

TITLE MESSAGES:

USER MESSAGE FILE(IDEF(6-10))

ORIGINAL PAGE IS  
OF POOR QUALITY

## MINIS PROGRAM SPECIFICATION

72.0 PROGRAM NAME- NORM

72.1 PURPOSE

RIGHT JUSTIFY TEXT, DELETING TRAILING BLANKS,

72.2 IDENTIFICATION

DATE- 7-9-75

REVISION-

AUTHOR- C. J. FARLESS (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 75 LOCATIONS (113 OCTAL)

**ORIGINAL PAGE IS  
OF POOR QUALITY**

72.3 DESCRIPTION

FOR THE REQUESTED NUMBER OF CHARACTERS, TEXT IS RIGHT JUSTIFIED,  
DELETING ANY TRAILING BLANKS,

72.4 USAGE

CALLING SEQUENCE:

CALL NORM(IARRAY,NBYTES)

IARRAY = FIRST WORD OF ARRAY TO RIGHT JUSTIFY

NBYTES = NO. OF CHARACTERS IN THE ARRAY TO RIGHT JUSTIFY

# MINIS PROGRAM SPECIFICATION

73.0 PROGRAM NAME- OUTPUT

73.1 PURPOSE

OUTPUT REQUESTED FIELDS AND VARIABLES OF RECORDS WITHIN A GIVEN SET.

73.2 IDENTIFICATION

DATE- 7-3-75

REVISION-

AUTHOR- CATHY J. FARLESS (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 1538 LOCATIONS (3002 OCTAL)

ORIGINAL PAGE IS  
OF POOR QUALITY

73.3 DESCRIPTION

FOR A GIVEN SET, FIELDS AND VARIABLES ARE OUTPUT IN COLUMN FORM WITH A HEADER ABOVE EACH COLUMN. THE HEADER IS OBTAINED FROM THE FIELD TITLES UNLESS THE FIELD WIDTH IS DETERMINED TO BE 6 CHARACTERS OR LESS, IN WHICH CASE THE FIELD NAME IS USED. FOR VARIABLES, THE SYMBOL NAME IS USED. THE FIELD OUTPUT FORMAT STORED IN CORE IS USED FOR OUTPUT OF FIELD DATA AND THE FORMAT F10.2 IS USED FOR VARIABLES.

OPTIONAL USER SPECIFIED REPORT HEADERS, LINE FORMATS, AND NAMELISTS MAY BE INVOKED BY NAMING SAME IN THE OUTPUT CALL STATEMENT. HEADERS AND FORMATS MUST BE FORTRAN FORMAT SPECIFICATIONS COMPATIBLE WITH THE HOST MACHINE FORTRAN. VARIABLE LISTS MUST HAVE THE SAME DATA TYPES IN THE SAME SEQUENCE AS SPECIFIED IN THE FORMAT STATEMENT WHEN BOTH ARE REFERENCED IN THE SAME CALL TO OUTPUT.

73.4 USAGE

CALLING SEQUENCE:

```
OUTPUT(SETNAME, ARG1, ARG2, ..., ARGN)           OR  
OUTPUT(SETNAME, ALL)                             OR  
OUTPUT(SETNAME, HEDNAME, LISTNAM, FMTNAM)        OR  
OUTPUT(SETNAME, LISTNAM)                         OR  
OUTPUT(SETNAME, HEDNAME, FMTNAM, ARG1, ARG2, ..., ARGN) OR ETC.
```

```
SETNAME = NAME OF SET TO OUTPUT  
ARG1    = FIELD OR VARIABLE TO OUTPUT  
ARG2    = " " " " " "
```

C-3

C-3

ARGN = THE ARGUMENT 'ALL' WILL CAUSE ALL FIELDS IN THE  
DATA BASE TO BE OUTPUT  
FMATNAM = NAME OF PREDEFINED FORMAT SPECIFICATION,  
HEDNAME = NAME OF PREDEFINED HEADER SPECIFICATION,  
LISTNAM = NAME OF PREDEFINED VARIABLE LIST.

## MINIS PROGRAM SPECIFICATION

74.0 PROGRAM NAME- PACDAT

74.1 PURPOSE

PACK A GIVEN NUMBER OF BITS IN THE DATA BASE RECORD AT THE REQUESTED BIT NUMBER.

74.2 IDENTIFICATION

DATE- 11-20-75

REVISION-

AUTHOR- C.J. FARLESS (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 79 LOCATIONS (117 OCTAL)

ORIGINAL FILE IS  
OF POOR QUALITY

74.3 DESCRIPTION

THE STARTING AND ENDING BIT NUMBERS ARE CALCULATED. IF THE DATA IS LEFT JUSTIFIED (IJUST=1), THE STARTING BIT NUMBER IS BIT ZERO. THE DATA IS THEN EXTRACTED FROM THE DATA VALUE ARRAY ONE BIT AT A TIME; THE NEXT BIT NUMBER AFTER THE LAST BIT USED IS RETURNED IN IBIT.

74.4 USAGE

CALLING SEQUENCE:

CALL PACDAT(IVAL,NBITS,IARRAY,IBIT,IJUST)

IVAL =DATA VALUE ARRAY

NBITS =NO. OF BITS TO PACK

IARRAY=DATA BASE RECORD ARRAY

IBIT =FIRST BIT NUMBER

IJUST =0 RIGHT JUSTIFIED IN 1ST WORD

=1 LEFT JUSTIFIED IN 1ST WORD

## MINIS PROGRAM SPECIFICATION

75.0 PROGRAM NAME- PACKER

75.1 PURPOSE

PACK THE FIELD VALUE AND DATA BASE RECORD POINTER INTO ONE ENTRY OF A FIRST LEVEL INDEX FILE,

75.2 IDENTIFICATION

DATE- 11 MARCH 1975

REVISION-

AUTHOR- D.R.SANDERS

COMPUTER- DCR 6024

LANGUAGE- FORTRAN IV

CORE SIZE- SIZE... 154 LOCATIONS (232 OCTAL)

75.3 DESCRIPTION

THE FIELD VALUE FOR 'IFLDNO' IN DATA BASE RECORD NUMBER 'IENTRY' IS STORED RIGHT JUSTIFIED IN THE FIRST 'NRFLD' BITS OF 'IARRAY', 'IENTRY' IS STORED RIGHT JUSTIFIED IN THE NEXT 'NBENT' BITS OF 'IARRAY'.

75.4 USAGE

CALLING SEQUENCE:

CALL PACKER(IARRAY,NWORDS,IENTRY,NBENT,IFLDNO,NRFLD)

IARRAY = DESTINATION ARRAY (1 BY 'NWORDS')  
NWORDS = NUMBER OF WORDS IN 'IARRAY'  
IENTRY = DATA BASE ENTRY OF SOURCE RECORD  
NBENT = NUMBER OF BITS IN ENTRY POINTER AREA  
IFLDNO = FIELD NUMBER OF SOURCE FIELD  
NRFLD = NUMBER OF BITS IN SOURCE FIELD

## MINIS PROGRAM SPECIFICATION

76.0 PROGRAM NAME- PARSE

76.1 PURPOSE

PROVIDE A GENERALIZED FORTRAN SUBROUTINE WHICH FOLLOWS COHEN AND GOTTLIEB'S NONRECURSIVE PARSING WITH BACKTRACK METHOD (ALGORITHM 3, COMP. SURVEYS, VOL. 2 NO. 1, MARCH 1970, PP. 65- 82).

76.2 IDENTIFICATION

DATE- 8 OCTOBER 1974

REVISION-

AUTHOR- D.R. SANDERS (205)-883-1778

COMPUTER- DCR 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 546 LOCATIONS (1042 OCTAL)

ORIGINAL PAGE IS  
OF POOR QUALITY

76.3 DESCRIPTION

PARSE ACCEPTS A SENTENCE IN THE FORM OF A STRING OF INTEGER VALUES WHICH ARE PRODUCED BY THE LEXICAL SCANNER (SEE LEXCAL DOCUMENTATION). THE ALGORITHM MENTIONED ABOVE IS USED IN CONJUNCTION WITH A SYNTAX GRAPH LIKE THOSE DISCUSSED IN THE SAME PAPER TO CONSTRUCT A TREE APPROPRIATE FOR THE INPUT SENTENCE. IF A SYNTAX ERROR IS DETECTED AN ERROR FLAG IS SET AND CONTROL IS RETURNED TO THE CALLING PROGRAM.

76.4 USAGE

CALLING SEQUENCE:

CALL PARSE(ISEN, LAST, ISTACK, IS, ISGR, IER)

ISEN = (N BY 1) ARRAY CONTAINING INTERNAL SENTENCE FORM

LAST = NUMBER OF SYMBOLS IN SENTENCE

ISTACK = (N BY 2) ARRAY CONTAINING TREE RESULTING FROM PARSE

IS = NUMBER OF ELEMENTS IN 'ISTACK'

ISGR = (N BY 4) ARRAY CONTAINING SYNTAX GRAPH

IER = ERROR FLAG: 7 FOR ERROR, 0 FOR NORMAL

## MINIS PROGRAM SPECIFICATION

77.0 PROGRAM NAME- PARVOC

77.1 PURPOSE

CHECK THE SYMBOL TABLE FOR THE EXISTENCE OF VARIABLES OR CONSTANTS DURING PARSE.

77.2 IDENTIFICATION

DATE- 10 OCTOBER 1974

REVISION-

AUTHOR- D.R.SANDERS

COMPUTER- DCR 6324

LANGUAGE- FORTRAN IV

CORE SIZE- 183 LOCATIONS (267 OCTAL)

77.3 DESCRIPTION

PARVOC ACCEPTS ONE VAL FIELD FROM THE SYNTAX GRAPH AND ONE SYMBOL TABLE POINTER FROM THE SENTENCE BEING PARSED. THE VAL FIELD CONTAINS A NEGATIVE VALUE INDICATING WHETHER A SET NAME OR A REAL NAME AND/OR INTEGER VARIABLE AND/OR CONSTANT IS ALLOWABLE AS A TERMINAL SYMBOL. PARVOC RETURNS A FLAG WHICH INDICATES A MATCH/MISMATCH OF THE SYMBOL TYPE WITH THAT IN THE SYMBOL TABLE.

77.4 USAGE

CALLING SEQUENCE:

CALL PARVOC(IVAL, ISNSMB, MATCH)

IVAL = VAL FIELD FROM SYNTAX GRAPH (NEGATIVE)

ISNSMB = SYMBOL POINTER FROM SENTENCE (NEGATIVE)

MATCH = FLAG: +=MATCH, -=MISMATCH

NOTE:

NEW VARIABLE, SET, SUBROUTINE AND FIELD NAMES ARE ASSIGNED TYPES 100 GREATER THAN THEIR NORMAL TYPES SO THE SYNTHESIZER CAN DETECT AND CHECK THEM FOR CORRECTNESS.

# MINIS PROGRAM SPECIFICATION

78.0 PROGRAM NAME- POINT

78.1 PURPOSE

FIND ALL MERITS PHOTOS ASSOCIATED WITH A POINT OR AREA.

78.2 IDENTIFICATION

DATE- 17 SEPTEMBER, 1975

REVISION-

AUTHOR- R.L.KEEFER (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- LOCATIONS (730 OCTAL)

78.3 DESCRIPTION

THE COORDINATES OF THE POINT OR AREA ARE ACCEPTED FROM THE USER AND USED TO INTERROGATE THE LATLON INDEX OF THE MERITS DATA BASE. IF THE QUERY IS FOR A POINT, THE FOUND LIST OF CANDIDATES IS CHECKED FOR ACTUAL COVERAGE OF THE POINT AND THE POINTERS TO THE PHOTOS COVERING THE POINT ARE SAVED IN THE NCAND FILE. IF THE QUERY IS FOR AN AREA, ALL FOUND CANDIDATES ARE SAVED IN THE NCAND FILE.

78.4 USAGE

CALLING SEQUENCE:

POINT

ORIGINAL PAGE IS  
OF POOR QUALITY

# MINIS PROGRAM SPECIFICATION

79.0 PROGRAM NAME- PRCENT

79.1 PURPOSE

TO COMPUTE A PERCENTAGE VALUE GIVEN A NUMERATOR AND DENOMINATOR,

79.2 IDENTIFICATION

DATE- 12-22-75

REVISION-

AUTHOR- C.J. FARLESS (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 70 LOCATIONS (106 OCTAL)

79.3 DESCRIPTION

GIVEN THE INPUTS, NUMERATOR AND DENOMINATOR, THE FOLLOWING IS USED TO CALCULATE THE PERCENT VALUE:

$$\text{PCENT} = \text{XNUM} / \text{DEN} * 1000, 0.$$

79.4 USAGE

CALLING SEQUENCE:

$$X = \text{PCENT}(\text{XNUM}, \text{DEN})$$

X = RETURNED VALUE OF PERCENT

XNUM = NUMERATOR

DEN = DENOMINATOR

# MINIS PROGRAM SPECIFICATION

80.0 PROGRAM NAME- PRINTER

80.1 PURPOSE

SET OUTPUT LISTING DEVICE TO THE PRINTER.

80.2 IDENTIFICATION

DATE- 26 FEBRUARY, 1975

REVISION-

AUTHOR- R.L.KEEFFER (205)-883-1778

COMPUTER- DATACRAFT 6024

LANGUAGE- FORTRAN

CORE SIZE- 10 LOCATIONS (12 OCTAL)

80.3 DESCRIPTION

THE OUTPUT LISTING DEVICE SPECIFIED IN COMMON IOARRAY AT IBUF(135) IS SET TO THE PRINTER (DEVICE NUMBER 6), THE NO. OF LINES PER PAGE(NLPP) IS SET TO 60(IBUF(139)).

80.4 USAGE

CALLING SEQUENCE:

PRINTER

ORIGINAL PAGE IS  
OF POOR QUALITY

## MINIS PROGRAM SPECIFICATION

81.0 PROGRAM NAME- PROCESS

81.1 PURPOSE

TO PROCESS THE RAW DATA RECORDS INTO DATA BASE RECORDS.

81.2 IDENTIFICATION

DATE- 2-19-76

REVISION-

AUTHOR- C.J. FARLESS (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 150 LOCATIONS (226 OCTAL)

81.3 DESCRIPTION

EACH RAW DATA RECORD IS READ IN AND STORED IN THE ELEMENT ARRAY, THEN THE QUADS ARE EXECUTED FOR EACH DEFINED FIELD TO CREATE THE DATA BASE RECORD. AFTER THIS IS COMPLETE, THE CONTROL IS RETURNED TO CREATE.

81.4 USAGE

CALLING SEQUENCE: CALL PROCESS(IRR, IREC)

IRR=ERROR FLAG =1=ERROR  
IREC=NO. OF RECORDS CREATED

# MINIS PROGRAM SPECIFICATION

82.0 PROGRAM NAME- QDFORM

82.1 PURPOSE

TO CREATE THE NECESSARY QUADS FOR AN OPERATION CARD OR RECORD.

82.2 IDENTIFICATION

DATE- 10-15-75

REVISION-

AUTHOR- C.J. FARLESS (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- LOCATIONS (2762 ACTAL)

82.3 DESCRIPTION

TO CREATE THE NECESSARY QUADS FOR AN OPERATION CARD, FIRST THE ARRAY CONTAINING THE OPERATION(ILINE) IN ASCII FORM IS SCANNED BY THE ROUTINE 'DELBLK' TO REMOVE ALL BLANKS AND RETURN THE NUMBER OF CHARACTERS IN THE OPERATION. THESE COLUMNS ARE THEN SEARCHED FOR THE EQUALS SYMBOL(=), SAVING THE COLUMN NUMBER BEFORE AND AFTER IT. THE INTERPRETING PROCESS BEGINS WITH THE OPERAND FIELD (STARTING IN THE COLUMN AFTER THE EQUALS SYMBOL), THE FIRST SYMBOL IS SCANNED (SCAN) AND ITS SYMBOL TYPE RETURNED AND THE REQUIRED CODES STORED IN THE QUAD. THIS PROCESS CONTINUES UNTIL ALL SYMBOLS IN THE OPERAND FIELD AND RESULT FIELD HAVE BEEN IDENTIFIED AND THE CORRESPONDING CODES STORED IN THE QUADS, THE FOLLOWING FORMS ARE ACCEPTABLE:

RESULT FIELD      OPERAND FIELD

INTERIM      = OPERAND

FIELD NO. OR

FIELD NAME

III      = OPERAND (+,-,\*,/,\*\*) OPERAND

II      = OPERAND .SQ,

III      = (SQ,+,-) OPERAND

III      = OPERAND ,S+, N

III      = OPERAND ,S2, OPERAND \* N

III      = OPERAND ,J, OPERAND

III      = OPERAND ,CH, N

III      = OPERAND1 ,CV, OPERAND2

**ORIGINAL PAGE IS  
OF POOR QUALITY.**

OPERAND MUST BE 1ST OF N  
ELEMENTS, FIELDS, OR  
INTERMEDIATE VALUES,

N=N# OF BYTES TO TRANS-  
FER. MAX N=132. OPERAND  
CANNOT BE A FIELD NO.  
CORRESPONDING VALUES.  
OPERAND1 CORRESPONDS TO  
OPERAND 2. RESULT MUST BE AN  
AN INTERMEDIATE VALUE.

III = OPERAND 1, SH, OPERAND 2 \* N SEARCH TABLE FOR CORRESPONDING VALUE FOR OPERAND1 TO GET VALUE OF RESULT, OPERAND2 INDICATES THE FIRST OF N ELEMENTS IN THE TABLE, OPERAND2 MUST BE AN INTERMEDIATE VALUE.

OPERANDS=ELEMENT NUMBER, FIELD NUMBER, INTERMEDIATE VALUE, INTEGER CONSTANT, REAL CONSTANT, OR CHARACTER(EX.: 'A'), UNLESS OTHERWISE STATED. THE QUADS FORMED FOR EACH OPERATION ARE ARRANGED AS FOLLOWS:

WORD 1 =OPERATION CODE  
 WORD 2-3 =OPERANDS  
     QUAD NUMBER = NEGATIVE NUMBER FROM -1 TO -1000  
                   WHEN AN INTERMEDIATE VALUE IS REFERRED TO AS THE FIRST ELEMENT OF AN ARRAY OR TABLE, THIS WILL BE EXPRESSED AS THE INTERMEDIATE VALUE NO. + 4000,  
     ELEMENT NUMBER = POSITIVE NUMBER FROM 1 TO 500  
     INTEGER CONSTANT = THE LOCATION OF THE VALUE IN  
                       ICON + 1000(1001 TO 1100)  
     REAL CONSTANT = THE LOCATION OF THE VALUE  
                       IN ICON + 2000(2001 TO 2100)  
     FIELD NUMBER = FIELD NUMBER + 3000 (3001 TO 3200)  
     CHARACTERS = THE LOCATION OF THE CHARACTER IN ICON  
                   +2200(2201 TO 2300)  
 WORD4-5 =RESULT OF THE OPERATION

ALL RESULTS WILL BE STORED IN WORDS 4 AND 5 IN FLOATING POINT EXCEPT IN THE CASE OF TEXT AND CHAR. TRANSFER, TEXT TRANSFER RESULTS WILL BE STORED IN THE TRANSFER ARRAY IN REUSE.

WHEN A FIELD NUMBER OR FIELD NAME IS FOUND IN THE RESULT FIELD, TWO QUADS WILL BE FORMED. THE FIRST QUAD WILL CONTAIN THE OPERATIONS DEFINED ON THE OPERATION CARD. THE SECOND QUAD WILL BE ARRANGED AS FOLLOWS:

WORD 1 =73  
 WORD 2 =FIELD NUMBER  
 WORD 3 =PREVIOUS QUAD NUMBER(NEGATIVE)  
 WORD 4-5 UNUSED

AT THIS POINT THE FIELD IS CONSIDERED DEFINED AND WHEN THE QUADS ARE EXECUTED THE 73 QUAD WILL CAUSE THE FIELD TO BE STORED ON THE DATA BASE RECORD. ANY OPERATION CARDS FOLLOWING WILL REFER TO THE NEXT FIELD.

TWO QUADS ARE FORMED FOR THE S2 AND SH OPERATIONS. THE FIRST WILL CONTAIN THE TWO ARRAY POINTERS, AND THE SECOND WILL CONTAIN THE NUMBER OF ELEMENTS TO SUM.

EXAMPLE\*  
 OPERATION CARD: F10=E4 S2 E20 \* 10

## MINIS PROGRAM SPECIFICATION

### RESULTING QUADS:

WORD 1	WORD 2	WORD 3	WORD 4	WORD 5
R NR 12	4	20	0	0
R NR 12	1001	0	0	0

CONSTANT TABLE LOCATION 1 = 10

### OPERATION CODES:

0 = ASSIGNMENT

1 = +

2 = -

3 = \*

4 = /

5 = \*\*

6 = .SQ. (SQUARE ROOT)

7 = .S+. (SUM OF N VALUES)

8 = .SP. (SUM OF PRODUCT OF TWO ARRAYS OF N VALUES)

9 = .V. (PERCENT)

10 = .CH. (CHARACTER TRANSFER)

11 = .CV. (STORE AS CORRESPONDING VALUE TABLE)

12 = .SH. (SEARCH TABLE FOR CORRESPONDING VALUE)

ERROR MESSAGES (THE CARD TO WHICH THE ERROR REFERS TO IS PRINTED ABOVE THE MESSAGE):

\*\*\*\*SYNTAX ERROR\*\*\*\*

THE OPERATION CARD IS NOT IN THE ACCEPTABLE FORM.

\*\*\*\*MORE THAN 500 ELEMENTS\*\*\*\*

ATTEMPTED TO DEFINE OR REFERENCE AN ELEMENT NUMBER GREATER THAN 500.

\*\*\*\*CONSTANT TABLE OVERFLOW\*\*\*\*

MORE THAN 100 CONSTANTS

\*\*\*\*MORE THAN 300 INTERIM EQ'NS\*\*\*\*

ATTEMPTED TO DEFINE OR REFERENCE MORE THAN 300 INTERMEDIATE EQUATIONS.

\*\*\*\*FIELD NUMBER GREATER THAN 200\*\*\*\*

ATTEMPTED TO DEFINE OR REFERENCE A FIELD NUMBER GREATER THAN 200.

## 82.4 USAGE

### CALLING SEQUENCE:

CALL QDFORM(ILINE, NQUAD)

ILINE = OPERATION CARD OR RECORD  
NQUAD = CURRENT QUAD NUMBER

ORIGINAL PAGE IS  
OF POOR QUALITY

## MINIS PROGRAM SPECIFICATION

83.0 PROGRAM NAME- R0FLIN

83.1 PURPOSE

READ THE DATA BASE POINTER AND FIELD VALUE FROM A FIRST LEVEL INDEX FILE RECORD.

83.2 IDENTIFICATION

DATE- 19 MARCH 1975

REVISION-

AUTHOR- D.R.SANDERS

COMPUTER- DCR 6024

LANGUAGE- FORTRAN IV

CORE SIZE- 74 LOCATIONS (112 OCTAL)

83.3 DESCRIPTION

FILE NAME, WORDS PER ENTRY, FIELD SIZE AND POINTER SIZE ARE TAKEN FROM THE ENTRY FOR THIS FILE IN /FILDEF/. THE FIRST LEVEL INDEX ENTRY IS READ AND UNPACKED TO FORM A FLOATING POINT VALUE AND A DATA BASE POINTER.

83.4 USAGE

CALLING SEQUENCE:

CALL R0FLIN(IRKPTR,IFPT,FLDVAL,IDBP)

IRKPTR = POINTER TO INDEX FILE DEF. BLOCK  
IFPT = INDEX FILE ENTRY NUMBER  
FLDVAL = FIELD VALUE  
IDBP = DATA BASE POINTER

## MINIS PROGRAM SPECIFICATION

84.0 PROGRAM NAME- RDULIN

84.1 PURPOSE

READ TWO ENTRIES IN AN UPPER LEVEL INDEX FILE,

84.2 IDENTIFICATION

DATE- 19 MARCH 1975

REVISION-

AUTHOR- D.R.SANDERS

COMPUTER- DCR 6024

LANGUAGE- FORTRAN IV

CORE SIZE- 81 LOCATIONS (121 OCTAL)

84.3 DESCRIPTION

FILE NAME AND WORDS PER ENTRY ARE TAKEN FROM THE ENTRY FOR THIS FILE IN /FILDEF/. A CALL TO THE FILE READ ROUTINE IS SET UP USING THIS INFORMATION ALONG WITH THE RECORD NUMBERS PROVIDED IN THE ARGUMENT LIST. THE CONTENTS OF THE TWO INDEX RECORDS REPLACE THEIR CORRESPONDING RECORD NUMBERS IN THE ARGUMENT LIST.

84.4 USAGE

CALLING SEQUENCE:

CALL RDULIN(IBKPTR,IPTA,IPTB)

IBKPTR = POINTER TO INDEX FILE DEF, BLOCK

IPTA = FIRST RECORD POINTER

(RETURNED AS CONTENTS OF THIS RECORD)

IPTB = SECOND RECORD POINTER (SAME AS ABOVE)

## MINIS PROGRAM SPECIFICATION

85.0 PROGRAM NAME- READFL, WRITFL

85.1 PURPOSE

WRITE OR READ AN ENTIRE FILE TO OR FROM THE DISC,

85.2 IDENTIFICATION

DATE- 11 DECEMBER, 1974

REVISION-

AUTHOR- R.L.KEFFER (205)-883-1778

COMPUTER- DATACRAFT 6024

LANGUAGE- FORTRAN IV

CORE SIZE- 112 LOCATIONS (160 OCTAL)

85.3 DESCRIPTION

THE NAMED FILE IS CLOSED, IF OPEN, TO SAVE ANY RECENT CHANGES BEFORE READING THE FILE. THE FILE IS READ OR WRITTEN ACCORDING TO THE ERROR FLAG. (IF IER=-9, READ) THE FILE IS CLOSED AND DELETED FROM THE ACTIVE FILE LIST. ACTUAL I/O IS PERFORMED VIA 'READWRI', 'READFL' SETS THE ERROR FLAG TO -9 AND CALLS 'WRITFL', WRITE SETS THE NUMBER OF RECORDS AND THEIR SIZE IN THE FOURTH AND FIFTH WORDS OF THE NAME ARRAY.

85.4 USAGE

CALLING SEQUENCE:

CALL WRITFL(NAMFIL,NUMREC,NBYT,ARRAY,IER)

CALL READFL(NAMFIL,NUMREC,NBYT,ARRAY,IER)

NAMFIL = FILE NAME ARRAY, (NAME(3A2),NREC,1S1Z)

NUMREC = NUMBER OF RECORDS IN FILE,

NBYT = NUMBER OF BYTES PER RECORD,

ARRAY = ARRAY TO BE READ/WRITTEN,

IER = ERROR CODE SET UPON RETURN:  
= 0 FOR NO ERRORS,  
= 1 FOR NO SUCH FILE IN READ,  
= 2 FOR ARGUMENT ERROR,

# MINIS PROGRAM SPECIFICATION

86.0 PROGRAM NAME- READLR

86.1 PURPOSE

READ IN THE NECESSARY NUMBER OF PHYSICAL RECORDS TO MAKE A LOGICAL RECORD, FIND KEY VALUE FOR START OF LOGICAL RECORD IF NECESSARY AND RETURN 1 LOGICAL RECORD STORED IN THE ELEMENT ARRAY.

86.2 IDENTIFICATION

DATE- 11-20-75

REVISION-

AUTHOR- C.J. FARLESS (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 384 LOCATIONS (600 OCTAL)

86.3 DESCRIPTION

THE NECESSARY READ FORMAT IS USED TO READ ONE RECORD, IF THE INPUT IS KEYED, THE RECORD IS EXAMINED FOR THE KEY VALUE, THE RECORD IS THEN STORED IN THE INPUT BUFFER, THIS IS REPEATED UNTIL ONE LOGICAL RECORD IS IN THE INPUT BUFFER, THIS LOGICAL RECORD IS THEN SENT TO 'STELEM' TO PRODUCE THE ELEMENT ARRAY, AFTER WHICH CONTROL IS RETURNED TO 'CREATE', THE MAXIMUM PHYSICAL RECORD SIZE IS 1000 WORDS.

86.4 USAGE

CALLING SEQUENCE:

CALL READLR(IREC,IEND)

IREC=PHYSICAL RECORD NUMBER(FOR THE FIRST CALL TO  
READLR, SET IREC TO 0)  
IEND=END OF FILE LOCATED=1, OTHERWISE=0

ORIGINAL PAGE IS  
OF POOR QUALITY

## MINIS PROGRAM SPECIFICATION

87.0 PROGRAM NAME- REDWRT

87.1 PURPOSE

READ OR WRITE RANDOM DISC FILES.

87.2 IDENTIFICATION

DATE- 12 DECEMBER, 1974

REVISION-

AUTHOR- R.L.KEEFER (205)-883-1778

COMPUTER- DATACRAFT 6024

LANGUAGE- FORTRAN IV

CORE SIZE- 108 LOCATIONS (154 OCTAL)

87.3 DESCRIPTION

THE LOGICAL UNIT IS DETERMINED BY THE STACK ENTRY, A READ OR WRITE IS PERFORMED ACCORDING TO THE OP CODE.

CALLING PROGRAMS TREAT DISK FILES AS TWO DIMENSIONAL ARRAYS WITH THE FIRST SUBSCRIPT EQUATED TO A RECORD NUMBER WHOSE SMALLEST VALUE IS ONE. THE ACTUAL FILE IS WRITTEN ON OR READ FROM WITH THE USER'S RECORD NUMBER BIASED BY ONE. I.E. USER'S RECORD NUMBER TEN BECOMES FILE RECORD NUMBER NINE. THIS BIASING IS USED TO PERMIT THE USER TO CHANGE THE DIMENSIONS OF HIS DISC ARRAYS.

87.4 USAGE

CALLING SEQUENCE:

CALL REDWRT(ISTK,NWDS,IARY,IOP,NREC)

ISTK = ENTRY TO ACTIVE FILE STACK.

NWDS = NUMBER OF WORDS TO TRANSFER,

IARY = SOURCE/DESTINATION ARRAY.

IOP = READ/WRITE FLAG (1=WRITE)

NREC = STARTING RECORD POINTER,

## MINIS PROGRAM SPECIFICATION

88.0 PROGRAM NAME- RFSET

88.1 PURPOSE

RESET SYSTEM PARAMETERS TO ERASE ALL KNOWLEDGE  
OF PREVIOUS USER DIALOG.

88.2 IDENTIFICATION

DATE- 18 SEPTEMBER 1975

REVISION-

AUTHOR- R.L.KEEFER (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 113 LOCATIONS (161 OCTAL)

88.3 DESCRIPTION

THE QUAD TABLE IS CLEARED AND THE LAST USED QUAD  
POINTER IS SET TO ZERO. ALL CREATED SYMBOLS (NON SYSTEM SYMBOLS)  
ARE REMOVED FROM THE SYMBOL TABLE AND THE SYMBOL COUNT IS  
RESET. THE SET AND SET MEMBERSHIP COUNT ARRAY IS ZEROED, THE  
EXECUTION MODE IS SET TO IMMEDIATE.

88.4 USAGE

CALLING SEQUENCE:

CALL RESET

**ORIGINAL PAGE IS  
OF POOR QUALITY.**

## MINIS PROGRAM SPECIFICATION

89.0 PROGRAM NAME- RFIELD

89.1 PURPOSE

INPUT A SPECIFIC FIELD FROM A SPECIFIC RECORD AND CONVERT THE VALUE TO REAL.

89.2 IDENTIFICATION

DATE- 1 AUGUST 1975

REVISION-

AUTHOR- D.R.SANDERS

COMPUTER- DATACRAFT 6024

LANGUAGE- FORTRAN IV

CORE SIZE- 108 LOCATIONS (154 OCTAL)

89.3 DESCRIPTION

RFIELD CALLS 'EXFLD' TO EXTRACT THE INTEGER FIELD FROM THE D.B. RECORD, AND THEN THE FIELD DEFINITION INFORMATION IS USED TO CALCULATE THE NUMBER OF BITS IN THE CONVERSION LOOP TO ARRIVE AT A FLOATING POINT FIELD VALUE.

89.4 USAGE

CALLING SEQUENCE:

VALUE = RFIELD(IREC,IFLDNO)

VALUE = CALCULATED VALUE

IREC = D.B. RECORD NUMBER

IFLDNO = FIELD NUMBER

## MINIS PROGRAM SPECIFICATION

90.0 PROGRAM NAME- REVSCN

90.1 PURPOSE

TO SCAN THE QUAD TABLE CALLING 'RVSCAN' FOR EACH STATEMENT ASSIGNING A VALUE TO THE REQUESTED SYMBOL.

90.2 IDENTIFICATION

DATE- 3-24-75

REVISION-

AUTHOR- C.J.FARLESS (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 237 LOCATIONS (355 OCTAL)

90.3 DESCRIPTION

THIS ROUTINE SCANS THE QUAD TABLE FROM THE LAST DELAYX TO THE CURRENT QUAD CALLING 'RVSCAN' FOR ANY STATEMENT WHICH ASSIGNS A VALUE TO THE REQUESTED SYMBOL ISYMS(ISYM). IF AN IF STATEMENT IS INCLUDED REFERRING TO THE REQUESTED SYMBOL, THE THEN AND ELSE STATEMENTS ARE ALSO SCANNED BY 'RVSCAN'. THIS ROUTINE ALLOWS A VARIABLE TO BE DEFINED BY OPERATIONS ON ITS CURRENT VALUE.

90.4 USAGE

CALLING SEQUENCE

CALL REVSCN(ISYM)

ISYM = SYMBOL TABLE ENTRY OF SYMBOL TO EXAMINE

ORIGINAL PAGE IS  
OF POOR QUALITY

## MINIS PROGRAM SPECIFICATION

91.0 PROGRAM NAME- SUBROUTINE RVSCAN

91.1 PURPOSE

TO SCAN THE QUAD TABLE IN REVERSE TO DETERMINE THE ROOTS OF A CREATED VARIABLE.

91.2 IDENTIFICATION

DATE- 4-18-75

REVISION-

AUTHOR- C. J. FARLESS (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 650 LOCATIONS (1212 OCTAL)

91.3 DESCRIPTION

THE REFERENCED VARIABLE IS CHECKED (BY ITS SYMBOL TYPE) TO DETERMINE IF IT IS A CREATED VARIABLE, IF NOT, NO ACTION IS TAKEN. IF IT IS A CREATED VARIABLE, THE ASSOCIATED STATEMENT IN THE QUAD TABLE IS FLAGGED (WORD 3 OF THE FIRST QUAD OF THAT STATEMENT FLAGGED AS -1) AS REQUIRING SUBSEQUENT EXECUTION, THAT STATEMENT IS THEN SCANNED FOR THE EXISTENCE OF ANY OTHER CREATED SYMBOLS, WHICH ARE TREATED IN THE SAME MANNER. THE RESULT IS TO HAVE EACH STATEMENT WHICH NEEDS TO BE EXECUTED TO DEFINE THE CALLING VARIABLE, FLAGGED FOR LATER EXECUTION.

91.4 USAGE

CALLING SEQUENCE

CALL RVSCAN(ISYM)

ISYM = SYMBOL TABLE ENTRY OF SYMBOL TO EXAMINE

# MINIS PROGRAM SPECIFICATION

92.0 PROGRAM NAME- SCAN

92.1 PURPOSE

SCAN AN OPERATION CARD FROM REQUESTED COLUMN, FINDING AND IDENTIFYING A SYMBOL.

92.2 IDENTIFICATION

DATE- 11-20-75

REVISION-

AUTHOR- C.J. FARLESS (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 961 LOCATIONS (1701 OCTAL)

**ORIGINAL PAGE IS  
OF POOR QUALITY**

92.3 DESCRIPTION

THE CHARACTER IN THE REQUESTED COLUMN IS IDENTIFIED. THE FOLLOWING OCCURS DEPENDING ON THE FIRST CHARACTER:

FIRST CHARACTER	RESULTING ACTION
E	THE NUMERIC CHARACTERS FOLLOWING THE E ARE CONVERTED TO AN INTEGER AND STORED IN VALUE AS THE ELEMENT NUMBER AND ICODE=0, IF NO NUMERIC CHARACTERS FOLLOW, ICODE=6.
F	THE NUMERIC CHARACTERS FOLLOWING THE F ARE CONVERTED TO AN INTEGER AND STORED IN VALUE AS A FIELD NUMBER AND ICODE=2, IF NO NUMERIC CHARACTERS FOLLOW, ICODE=6.
I	THE NUMERIC CHARACTERS FOLLOWING THE I ARE CONVERTED TO AN INTEGER AND STORED IN VALUE AS AN INTERMEDIATE VALUE NUMBER AND ICODE=1, IF NO NUMERIC CHARACTERS FOLLOW, ICODE=6.
+	ICODE=5, VALUE=1,
.(DECIMAL POINT)	THE DECIMAL POINT AND THE FOLLOWING NUMERIC CHARACTERS ARE CONVERTED TO A FLOATING POINT NUMBER AND STORED IN VALUE, ICODE=4, IF ICODE IS NOT FOLLOWED BY NUMERIC CHARACTERS, SPECIAL OPERATIONS ARE CHECKED: .SQ. = SQUARE ROOT(ICODE=5,VALUE=6) .S+ = SUMARY(ICODE=5,VALUE=7) .S2 = PRODUCT SUMARY(ICODE=5,VALUE=8) .V. = PERCENT(ICODE=5,VALUE=9) .CH. = CHARACTER TRANSFER(ICODE=5,VALUE=10)

,CV, = TABLE ENTRY(ICODE=5,VALUE=11)  
,SH, = SEARCH TABLE(ICODE=5,VALUE=12)  
OTHERWISE, ICODE=6, VALUE=0.

ICODE=5, AND VALUE=2,

ICODE=5, VALUE=3, IF THE NEXT CHARACTER IS  
ALSO AN \*, THE VALUE=5,

ICODE=5, VALUE=4,

0 THRU 9

EACH DIGIT IS STORED UNTIL A NON-NUMERIC  
CHARACTER IS FOUND, IF THE NUMERIC DIGIT  
CONTAINS A DECIMAL POINT, THE NUMBER IS  
CONVERTED TO FLOATING POINT AND STORED IN  
THE CONSTANT TABLE WITH ICODE=4 AND THE  
LOCATION IN THE CONSTANT TABLE IN VALUE. IF  
NO DECIMAL POINT IS FOUND, THE NUMERIC  
CHARACTERS ARE CONVERTED TO INTEGER AND  
STORED IN THE CONSTANT TABLE WITH ICODE=3  
AND THE CONSTANT TABLE LOCATION STORED IN  
VALUE.

(APOSTROPHE)

STORE THE CHARACTERS BEFORE THE NEXT  
APOSTROPHE IN ICAN ICODE=7,VALUE=LOCATION  
IN CONSTANT TABLE. IF NO 2ND APOSTROPHE IS  
FOUND, ICODE=6,VALUE=0. MAXIMUM OF NBW\*2  
CHARS. WILL BE STORED. ANY MORE CHARS.  
WITHIN THE APOSTROPHES WILL BE IGNORED.

WHEN THE ICODE AND VALUE PARAMETERS HAVE BEEN DETERMINED, SCAN  
RETURNS CONTROL TO EXTPR.

#### 92.4 USAGE

##### CALLING SEQUENCE:

CALL SCAN(NCHAR,ICODE,VALUE,ILINE)

NCHAR = COLUMN NUMBER TO BEGIN SCAN.

ICODE = SYMBOL CODE.

VALUE = VALUE OF SYMBOL.

ILINE = OPERATION CARD ARRAY.

ICODE	VALUE
0=ELEMENT	ELEMENT NUMBER
1=INTERMEDIATE VALUE	INTERMEDIATE VALUE NUMBER
2=FIELD NUMBER	FIELD NUMBER
3=INTEGER CONSTANT	LOCATION OF CONSTANT IN CONSTANT TABLE
4=REAL CONSTANT	LOCATION OF CONSTANT IN CONSTANT TABLE
5=OPERATION	OPERATION CODE
6=OTHER	0

7=CHARACTER

LOCATION OF CHARACTER STRING LEFT JUSTIFIED  
IN CONSTANT TABLE

OPERATION CODES

+ = 1	.SQ, = 6	.CV, = 11
- = 2	.S+, = 7	.SH, = 12
* = 3	.S2, = 8	
/ = 4	.J, = 9	
** = 5	.CH, = 10	

ORIGINAL PAGE IS  
OF POOR QUALITY

# MINIS PROGRAM SPECIFICATION

93.0 PROGRAM NAME- SCANTB

## 93.1 PURPOSE

SCAN OLDMAN SYMBOL TABLE CHECKING A PARTICULAR BYTE IN EACH ENTRY FOR A DEFINED VALUE.

## 93.2 IDENTIFICATION

DATE- 11 NOVEMBER 1974

REVISION-

AUTHOR- D.R. SANDERS

COMPUTER- DCR 6024

LANGUAGE- FORTRAN IV

CORE SIZE- 66 LOCATIONS (102 OCTAL)

## 93.3 DESCRIPTION

SCANTB SCANS THE SYMBOL TABLE WHICH CONTAINS 12 BYTES PER ENTRY CHECKING 'IBYTE' FOR 'LOW', I.E. VALUE('IBYTE'), I.E. 'IUP' STARTING WITH ENTRY NUMBER 'IENTRY'. THE ENTRY AT WHICH A MATCH IS FIRST FOUND IS STORED IN 'IENTRY'.

## 93.4 USAGE

CALLING SEQUENCE:

CALL SCANTB(IBYTE,LOW,IUP,IENTRY)

IBYTE = BYTE NUM. (1 TO 12) TO BE TESTED.

LOW = LOWER LIMIT FOR VALUE OF 'IBYTE'

IUP = UPPER LIMIT

IENTRY= STARTING ENTRY  
RETURNS WITH ENTRY NUM. OF FIRST MATCH  
OR -1 IF NO MATCH.

## MINIS PROGRAM SPECIFICATION

94.0 PROGRAM NAME- SCIFID

94.1 PURPOSE

SCAN THE INDEX FILE ID BLOCKS IN /FILDEF/ FOR A GIVEN INDEX LEVEL FOR A GIVEN FIELD.

94.2 IDENTIFICATION

DATE- 17 MARCH 1975

REVISION-

AUTHOR- D.R. SANDERS

COMPUTER- DCP 6024

LANGUAGE- FORTRAN IV

CORE SIZE- 92 LOCATIONS (134 OCTAL)

94.3 DESCRIPTION

BEGINNING WITH THE 41-<sup>ST</sup> CELL OF /FILDEF/, EACH TEN WORD INDEX FILE ID BLOCK IS TESTED IN SEQUENCE UNTIL MATCHES ARE MADE ON FIELD NUMBER AND INDEX LEVEL. IF NO MATCH IS FOUND, A POINTER TO THE ID BLOCK WITH THE HIGHEST INDEX LEVEL FOR THE FIELD NUMBER IS RETURNED.

94.4 USAGE

CALLING SEQUENCE:

CALL SCIFID(IFLDNO,LEVEL,IBLOCK)

IFLDNO = FIELD NUMBER

LEVEL = CALLED WITH THE DESIRED INDEX LEVEL  
RETURNED WITH MINIMUM OF (LEVEL,HIGHEST FOUND)

IBLOCK = NUMBER OF INDEX FILE ID BLOCK  
0 IF FIELD NUMBER NOT FOUND

ORIGINAL PAGE IS  
OF POOR QUALITY

## MINIS PROGRAM SPECIFICATION

95.0 PROGRAM NAME- SCINDEX

95.1 PURPOSE

EMPLOY AN INDEXING TECHNIQUE TO COMPILE A CANDIDATE FILE OF DATA BASE ENTRY POINTERS FOR A RANGE OF VALUES FOR A FIELD,

95.2 IDENTIFICATION

DATE- 14 MARCH 1975

REVISION-

AUTHOR- D.R. SANDERS

COMPUTER- DCR 6024

LANGOAGE- FORTRAN IV

CORE SIZE- 654 LOCATIONS (1216 OCTAL)

95.3 DESCRIPTION

UPPER LEVEL INDEX FILE POINTERS ARE CALCULATED USING THE MINIMUM VALUE AND SEGMENT SIZE FIELDS OF THE INDEX FILE DEFINITION, THE PROCESS OF COMPARING MAXIMUM AND MINIMUM FIELD VALUES WITH CALCULATED VALUES TO ARRIVE AT TWO POINTERS TO THE INDEX FILE FOR THE NEXT LOWER LEVEL IS REPEATED UNTIL THE FIRST LEVEL INDEX FILE IS REACHED, THEN THE DATA BASE POINTERS EXTRACTED FROM THE FIRST LEVEL INDEX FILE ARE STORED IN THE CANDIDATE FILE AND SORTED IN ASCENDING ORDER,

95.4 USAGE

CALLING SEQUENCE:

CALL SCINDEX(IFLDNO,LEVEL,VALMIN,VALMAX,NCAND)

IFLDNO = FIELD NUMBER

LEVEL = BEGINNING INDEX LEVEL (DEFAULTS TO HIGHEST DEFINED)

VALMIN = MINIMUM FIELD VALUE

VALMAX = MAXIMUM FIELD VALUE

NCAND = NUMBER OF CANDIDATES IN HIT FILE

# MINIS PROGRAM SPECIFICATION

96.0 PROGRAM NAME- SEARCH

96.1 PURPOSE

SEARCH CONVERSION TABLE FOR VALUE,

96.2 IDENTIFICATION

DATE- 2-17-76

REVISION-

AUTHOR- C. J. FARLESS (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 151 LOCATIONS (227 OCTAL)

ORIGINAL PAGE IS  
OF POOR QUALITY

96.3 DESCRIPTION

EACH OF N TABLE ENTRIES ARE COMPARED TO THE VALUE INDICATED AT IQU(2,NQUAD) UNTIL AN EQUIVALENT VALUE IS FOUND, THE CORRESPONDING VALUE IS RETURNED FOR THE CONVERSION. IF NO EQUIVALENT VALUE IS LOCATED THE VALUE OF THE FIRST TABLE ENTRY IS RETURNED (FIRST TABLE ENTRY IS STORED IN IQU(3,NQUAD)), EXAMPLE:

```
I11=0
I12='A' .CV. 1
I13='B' .CV. 2
I14='D' .CV. 3
I15='E' .CV. 4
FLD4=E12 .SH. I11*5
IF E12 IS AN 'A' FLD4=1
          'B' FLD4=2
          'D' FLD4=3
          'E' FLD4=4
          OTHER FLD4=0
```

96.4 USAGE

CALLING SEQUENCE:

```
VAL = SEARCH(LQC1,LQC2,N)
```

```
VAL = RETURNED CORRESPONDING VALUE
LQC1 = QUAD TABLE ENTRY FOR OPERAND1
LQC2 = QUAD TABLE ENTRY FOR OPERAND2
N     = NUMBER OF VALUES IN TABLE
```

## MINIS PROGRAM SPECIFICATION

97.0 PROGRAM NAME- SETPRI

97.1 PURPOSE

LOCATE GIVEN FILE NAME IN AN ACTIVE FILE TABLE AND SET PRIORITY LEVELS.

97.2 IDENTIFICATION

DATE- 11 DECEMBER, 1974

REVISION-

AUTHOR- P.L.KEFFER (205)-883-1778

COMPUTER- DATACRAFT 6024

LANGUAGE- FORTRAN IV

CORE SIZE- 514 LOCATIONS (1002 OCTAL)

97.3 DESCRIPTION

THE CALLING FILE NAME IS COMPARED AGAINST EACH ACTIVE FILE NAME. IF LOCATED, THAT FILE IS PROMOTED TO FIRST PRIORITY AND ANY HIGHER PRIORITY FILES ARE 'DEMOTED' TO MAKE ROOM. IF THE FILE IS NOT IN THE ACTIVE STACK, AN ENTRY IS MADE IN THE STACK. (THE LOWEST PRIORITY FILE IS CLOSED, IF NECESSARY) AND PRIORITIES ARE ADJUSTED AS ABOVE. THE FILE IS OPENED, IF IT IS NEW TO THE STACK, AND THE ENTRY TO THE FILE STACK IS RETURNED.

97.4 USAGE

CALLING SEQUENCE:

CALL SETPRI(IDFILE,ISTACK,IER)

IDFILE = 5 WORD FILE DEFINITION ARRAY SET AS FOLLOWS:  
IDFILE(1-3) CONTAINS THE 6 CHARACTER FILE NAME.(3A2)  
IDFILE(4) CONTAINS THE NUMBER OF RECORDS IN FILE.  
IDFILE(5) CONTAINS THE LENGTH OF A RECORD IN BYTES,

ISTACK = ENTRY TO FILE STACK CONTAINING THIS ID.  
IER = FILE STATUS ERROR FLAG SET AS FOLLOWS:  
= 0 IF FILE PREVIOUSLY EXISTED.  
= 1 IF FILE IS A NEW FILE.

# MINIS PROGRAM SPECIFICATION

98.0 PROGRAM NAME- SORTIX

98.1 PURPOSE

SORT LARGE INDEX FILES.

98.2 IDENTIFICATION

DATE- 16 MAY, 1975

REVISION-

AUTHOR- R.L.KEEGER (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 279 LOCATIONS (427 OCTAL)

98.3 DESCRIPTION

THE AVAILABLE CORE, SPECIFIED IN THE FIRST WORD OF COMMON 'REUSE', IS USED AS A SCRATCH AREA FOR A BINARY SORT ROUTINE. THE NAMED FILE IS BROKEN INTO NSEG SEGMENTS, READ IN CORE, SORTED, AND WRITTEN BACK TO THE DISC FILE,  $NSEG=1+ITMTOT*ITMLN*2/LENBUF$  WHERE ITMTOT IS THE NUMBER OF ITEMS IN THE FILE, ITMLN IS THE NUMBER OF WORDS PER ITEM, AND LENBUF IS THE LENGTH OF THE AVAILABLE SCRATCH BUFFER. THE SORTED BLOCKS ARE MERGED TOGETHER VIA 'MERGER' TO THE SPECIFIED OUTPUT DEVICE OR FILE, 'IOT'. THE COMMON 'REUSE' IS DIVIDED INTO NSEG SECTIONS FOR INPUT BUFFERING. AFTER A SUCCESSFUL MERGE, THE INPUT FILE IS DELETED. IF THE SORT KEY IS TOO LONG (.GT. 2 WORDS LESS 2 BITS), NO SORTING IS PERFORMED AND CONTROL IS RETURNED TO THE CALLING PROGRAM.

98.4 USAGE

CALLING SEQUENCE:

CALL SORTIX(INPUT, IOUTF, IOT, KEY1, KEY2)

INPUT = INPUT FILE NAME AND SIZE DATA,  
IOUTF = OUTPUT FILE NAME AND SIZE DATA,  
IOT = OUTPUT DEVICE NUMBER, (0 FOR DISC)  
KEY1 = FIRST BIT OF SORT KEY1, (0-N)  
KEY2 = LAST BIT OF SORT KEY, (GREATER OR EQUAL TO KEY1)

ORIGINAL PAGE IS  
OF POOR QUALITY

## MINIS PROGRAM SPECIFICATION

99.0 PROGRAM NAME- SRTMER

99.1 PURPOSE

SORT DATA BASE FILE ON UP TO 4 FIELDS AND/OR MERGE DATA BASE FILE  
RETAINING THE NEWEST RECORDS,

99.2 IDENTIFICATION

DATE- 2-17-76

REVISION-

AUTHOR- C.J.FARLESS (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 1805 LOCATIONS (3415 OCTAL)

99.3 DESCRIPTION

IF SORTS ARE REQUESTED, SORTS ARE PERFORMED IN THE ORDER THE  
FIELDS WERE REQUESTED, BY SORTING ON EACH FIELD USING 'SORTIX',  
AFTER SORTING HAS BEEN COMPLETED, THE SORTED FILE IS WRITTEN ON  
THE DATA BASE FILE. IF A MERGE IS REQUESTED, THE MERGE IS DONE BY  
COMPARING THE VALUE OF THE FIELD ON EACH RECORD TO ITS SUCCESSIVE  
RECORDS. IF EQUIVALENT RECORDS ARE FOUND, THE LAST ONE (OR MOST  
RECENT) IS RETAINED AND THE DUPLICATES DELETED THIS FILE IS ALSO  
WRITTEN ON THE DATA BASE FILE, UPDATING THE NO. OF DATA BASE  
RECORDS IN IDEF(4). IN THE CASE OF AN UPDATE RUN, THE DATA IS NOT  
WRITTEN ON THE DATA BASE FILE UNTIL MERGED WITH THE DATA BASE  
FILE; IF NO MERGE IS REQUESTED, THE UPDATE RECORDS ARE ADDED TO  
THE END OF THE DATA BASE RECORDS.

99.4 USAGE

CALLING SEQUENCE:

CALL SRTMER(NREC, IWDEV)

IWDEV = DEVICE NO. OF SCRATCH FILE 0=DISC  
OTHER=TAPE DEVICE NO.

NREC = NO. OF RECORDS IN DATA BASE FILE, -OUTPUT  
= NO. OF RECORDS CREATED, -INPUT

MINIS PROGRAM SPECIFICATION

ORIGINAL PAGE IS  
OF POOR QUALITY

100.0 PROGRAM NAME- STELEM

100.1 PURPOSE

STORE ONE LOGICAL RECORD OF INPUT DATA IN THE ELEMENT ARRAY  
ACCORDING TO THE INFORMATION IN THE INPUT DECK (STORED IN COMMON),

100.2 IDENTIFICATION

DATE- 12-2-75

REVISION-

AUTHOR- C.J. FARLESS (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 813 LOCATIONS (1455 OCTAL)

100.3 DESCRIPTION

BEFORE ENTERING THIS ROUTINE, THE INPUT DECK HAS BEEN STORED IN  
THE COMMON REUSE WITH THE INFORMATION FOR EACH ELEMENT STORED IN  
NUMERICAL ORDER, ALSO, ONE LOGICAL RECORD HAS BEEN STORED IN THE  
COMMON AREA REUSE, THE FOLLOWING INFORMATION IS STORED FOR EACH  
ELEMENT IN INPT:

INPT(4,500)

INPT(1,1)=NBITS=NO. OF BITS OR BYTES (IF TEXT) IN THE ELEMENT

INPT(2,1)=IFIR=FIRST BIT OR BYTE NO. OF THE ELEMENT,

(FIRST BIT IN ARRAY = 0)

(FIRST BYTE IN ARRAY = 1)

INPT(3,1)=ICODE=CONVERSION CODE FOR THE ELEMENT

ICODE=0 NO CONVERSION

=1 ASCII TO INTEGER

=2 BCD TO INTEGER

=3 EBCDIC TO INTEGER

=4 ASCII(8BIT) TO BCD(6BIT)

=5 CHARACTER TO LOGICAL (T OR Y =1, OTHERS=0)

ORIGINAL PAGE IS  
OF POOR QUALITY

INPT(4,1)=ITYP =DATA TYPE CODE

ITYP =0= INTEGER OR BINARY

=1= ASCII(8 BIT CHARACTER CODE)

=2= BCD (6 BIT CHARACTER CODE)

=3= EBCDIC (8 BIT CHARACTER CODE)

=4= FLOATING POINT (NOVA=16 BIT WORDS)

=5= FLOATING POINT (DATACRAFT=24 BIT WORDS)

=6= FLOATING POINT (SIGMA 5=32 BIT WORDS)

=7= FLOATING POINT (IBM 360=32 BIT WORDS)

=8= TEXT ASCII (NBITS AND IFIR SHOULD BE IN BYTES)

=9= TEXT BCD           #           #           #           #

=10=TEXT EBCDIC       #           #           #           #

FOR EACH ELEMENT, THE DATA IS EXTRACTED FROM REUSE, THE REQUIRED

CONVERSION IS MADE, AND THE DATA STORED IN THE ELEMENT ARRAY (LEMT). TWO LOCATIONS ARE AVAILABLE FOR EACH ELEMENT ALLOWING FOR FLOATING POINT STORAGE AND TWO WORDS OF CHARACTER DATA (4 BYTES/16 BIT WORDS; 6 BYTES/24 BIT WORDS). ALL DATA IS LEFT JUSTIFIED IN THE TWO WORDS. ALL DATA NOT IDENTIFIED OR CONVERTED TO CHARACTER OR TEXT WILL BE STORED IN THE ELEMENT ARRAY IN FLOATING POINT. CHARACTER DATA WILL BE EITHER ASCII OR BCD. TEXT WILL CREATE ENOUGH ELEMENTS TO STORE  $NRW * 2$  CHARACTERS PER ELEMENT, ALL TEXT WILL BE STORED IN ASCII.

LEGAL CONVERSIONS:

DATA TYPE	CONVERSION --CODE
BINARY	NO CONV, --0
ASCII	NO CONV, --0
	ASCII TO INTEGER--1
	ASCII TO BCD --4
	CHARACTER TO LOGICAL --5
BCD	NO CONV, -- 0
	BCD TO INTEGER -- 2
	CHARACTER TO LOGICAL -- 5
EBCDIC	NO CONV, -- 0 (WILL BE STORED IN ASCII)
	EBCDIC TO INTEGER -- 3
	EBCDIC TO BCD -- 4
	CHARACTER TO LOGICAL -- 5
ALL FLOATING POINT	NO CONV, -- 0
ALL TEXT	NO CONV, -- 0

100.4 USAGE

CALLING SEQUENCE:

CALL STELEM(NELMS)

NELMS = NUMBER OF ELEMENTS IN ARRAY

# MINIS PROGRAM SPECIFICATION

101.0 PROGRAM NAME- STNAME

101.1 PURPOSE

ALLOW INDEPENDENT ENTRY OF SYMBOLS INTO THE SYMBOL TABLE

101.2 IDENTIFICATION

DATE- 18 SEPTEMBER 1975

REVISION-

AUTHOR- D.R. SANDERS (205)-883-1778

COMPUTER- DCR 6024

LANGUAGE- FORTRAN IV

CORE SIZE- 230 LOCATIONS (346 OCTAL)

ORIGINAL PAGE IS  
OF POOR QUALITY

101.3 DESCRIPTION

THE NUMBER OF BYTES POSSIBLE IN 'NAME' IS CHOSEN ACCORDING TO THE TYPE OF SYMBOL. THE SYMBOL TABLE IS SCANNED FOR A MATCH ON THIS NUMBER OF BYTES AND THE CORRECT SYMBOL TYPE. IF A MATCH IS FOUND, THE REFERENCED ARRAY FIELD IS RETURNED IN 'NREF' AND THE SYMBOL TABLE ENTRY NUMBER IS RETURNED IN 'IENTRY'. IF NO MATCH IS FOUND, THE CORRECT NUMBER OF BYTES IN 'NAME' ARE ENTERED INTO THE SYMBOL TABLE, THE ENTRY POINTER IS STORED IN 'IENTRY' AND A MINUS ONE IS RETURNED IN 'NREF' TO INDICATE THAT A NEW SYMBOL WAS ENTERED.

101.4 USAGE

CALLING SEQUENCE:

CALL STNAME(NAME,NREF,ITYPE,IENTRY)

NAME = SYMBOL NAME (UP TO 11 CHARACTERS)

NREF = REFERENCED ARRAY FIELD

-1 IF NEW SYMBOL CREATED

ITYPE = SYMBOL TYPE

IENTRY = SYMBOL TABLE ENTRY POINTER

## MINIS PROGRAM SPECIFICATION

102.0 PROGRAM NAME- ST0FLD

102.1 PURPOSE

STORE A COMPUTED FIELD VALUE ON THE DATA BASE RECORD.

102.2 IDENTIFICATION

DATE- 11-20-75

REVISION-

AUTHOR- C.J. FARLESS (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 127 LOCATIONS (177 OCTAL)

102.3 DESCRIPTION

THE NUMBER OF BITS (OR BYTES FOR TEXT), THE FIRST BIT (OR BYTE) NUMBER, AND THE DATA TYPE ARE EXTRACTED FROM THE CORRESPONDING FIELD DEFINITIONS. EXCEPT IN THE CASE OF DATA TYPE EQUAL TO REAL, CHARACTER, OR TEXT, THE DATA IS CONVERTED TO INTEGER AND STORED IN THE FIRST WORD OF THE FIELD VALUE ARRAY. THEN THE DATA IS PACKED INTO THE DATA BASE RECORD ARRAY (IN REUSE) USING PACDAT. WHEN ENTERING PACDAT INTEGER DATA IS STORED RIGHT JUSTIFIED IN THE FIRST WORD OF THE FIELD VALUE ARRAY. FLOATING POINT IS STORED IN THE FIRST 2 WORDS OF THE FIELD VALUE ARRAY. CHARACTER AND TEXT ARE LEFT JUSTIFIED IN THE LEAST NUMBER OF BYTES NEEDED BEGINNING IN THE FIRST WORD OF THE FIELD VALUE ARRAY. THE HIGHEST BIT NUMBER AVAILABLE IS RETURNED.

102.4 USAGE

CALLING SEQUENCE:

CALL ST0FLD(IFLD,IVAL,IEND,IMOD)

IFLD =FIELD NUMBER

IVAL =FIELD VALUE ARRAY

IEND =THE NEXT BIT NUMBER AFTER THE HIGHEST  
ONE USED(SET TO 0 WHEN STARTING A  
NEW RECORD).

IMOD =RECORD NO. TO MODIFY

# MINIS PROGRAM SPECIFICATION

103.0 PROGRAM NAME- SUBSET

103.1 PURPOSE

DEFINE A SUBSET FROM THE SET CANDIDATE FILE.

103.2 IDENTIFICATION

DATE- 2-3-75

REVISION-

AUTHOR- C.J. FARLESS (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 97 LOCATIONS (141 OCTAL)

ORIGINAL PAGE IS  
OF POOR QUALITY

103.3 DESCRIPTION

THE REFERENCED STATEMENT IS EXECUTED FOR EACH MEMBER OF THE CANDIDATE INPUT FILE VIA SUBROUTINE 'EXCANS'. EACH THREE WORD CANDIDATE ARRAY IS TRANSFERRED TO THE OUTPUT CANDIDATE FILE UNLESS WORDS 2 AND 3 ARE BOTH ZERO (THE CORRESPONDING BITS OF WORDS 2 AND 3 ARE SET IF A CANDIDATE IS A MEMBER OF THE CORRESPONDING SET). AFTER ALL THE CANDIDATES ARE EXAMINED, BOTH FILES ARE CLOSED, AND THE NUMBER OF CANDIDATES IN THE FILE IS UPDATED IN COMMON(/COMTAB/ICON(100)).

103.4 USAGE

CALLING SEQUENCE:

CALL SUBSET(IQF,IQL,NCAND)

IQF = FIRST QUAD TABLE ENTRY OF THE STATEMENT  
IQL = LAST QUAD TABLE ENTRY OF THE STATEMENT  
NCAND = NUMBER OF MEMBERS IN THE NEW SET(OUTPUT),

## MINIS PROGRAM SPECIFICATION

104.0 PROGRAM NAME- SUMMARY

104.1 PURPOSE

TO CALCULATE THE SUM OF N ELEMENTS IN AN ARRAY.

104.2 IDENTIFICATION

DATE- 12-22-75

REVISION-

AUTHOR- C.J.FARLESS (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 42 LOCATIONS (52 OCTAL)

104.3 DESCRIPTION

N ELEMENTS OF THE REQUESTED ARRAY ARE SUMMED AND THE RESULT RETURNED.

104.4 USAGE

CALLING SEQUENCE:

X = SUMMARY(LOC,NUM)

LOC = QUAD OPERAND VALUE OF 1ST OF N ELEMENTS IN THE ARRAY.

NUM = NUMBER OF ELEMENTS.

X = RESULT OF SUM.

# MINI PROGRAM SPECIFICATION

105.0 PROGRAM NAME- SUMSET

105.1 PURPOSE

SUM A VARIABLE OR FIELD OVER A GIVEN SET,

105.2 IDENTIFICATION

DATE- 4-15-75

REVISION-

AUTHOR- C. J. FARLESS (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 468 LOCATIONS (724 OCTAL)

105.3 DESCRIPTION

FOR A GIVEN SET THE REQUESTED VARIABLE OR FIELD IS SUMMED AND THE RESULT ADDED TO THE REQUESTED VARIABLE NAME. THE FIRST VARIABLE IN THE ASSIGNMENT QUAD OF THE STATEMENT IS TAKEN AS THE VARIABLE TO WHICH TO ADD THE RESULT, THE RESULT IS IN FLOATING POINT.

105.4 USAGE

CALLING SEQUENCE:

SUMNAME = SUMSET(SETNAME, SUMMEE)

SETNAME = NAME OF SET TO SUM OVER

SUMMEE = VARIABLE OR FIELD TO SUM

SUMNAME = NAME OF VARIABLE TO STORE SUM IN

NOTE: SUMMEE AND SETNAME CAN BE IN ANY ORDER IN THE ARGUMENT LIST.

ORIGINAL PAGE IS  
OF POOR QUALITY

## MINIS PROGRAM SPECIFICATION

106.0 PROGRAM NAME- SUM2

106.1 PURPOSE

COMPUTE THE SUM OF THE PRODUCT OF TWO ARRAYS.

106.2 IDENTIFICATION

DATE- 12-22-75

REVISION-

AUTHOR- C.J. FARLESS (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 53 LOCATIONS (65 OCTAL)

106.3 DESCRIPTION

THE CORRESPONDING MEMBERS OF 2 ARRAYS ARE MULTIPLIED TOGETHER AND THE PRODUCTS SUMMED AND RETURNED AS THE RESULT.

106.4 USAGE

CALLING SEQUENCE:

X = SUM2(LOC1,LOC2,NUM)

X = RESULTANT VALUE

LOC1 AND LOC2=OPERAND VALUE FROM QUAD TABLE WHICH  
CORRESPONDS TO THE 1ST MEMBER OF EACH ARRAY.

NUM = NUMBER OF ELEMENTS IN EACH ARRAY

# MINIS PROGRAM SPECIFICATION

107.0 PROGRAM NAME- SYNTHE

107.1 PURPOSE

BREAK AN INTERNAL SENTENCE INTO SURCLAUSES FOR FORMATION OF EXECUTION TRIPLES.

107.2 IDENTIFICATION

DATE- 4 MARCH 1975

REVISION-

AUTHOR- D.R.SANDERS

COMPUTER- DCR 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 362 LOCATIONS (552 OCTAL)

107.3 DESCRIPTION

THE SENTENCE TYPE, DETERMINED BY 'ISTACK(2,2)' IS STORED IN THE QUAD STACK. THEN DIFFERENT ACTIONS ARE TAKEN ACCORDING TO THE SENTENCE TYPE. FOR SET OR VARIABLE DEFINITION, CONTROL IS PASSED DIRECTLY TO 'SYNTH1'. FOR 'IF' STATEMENTS CONTROL PASSES TO 'SYNTH1' FOR EACH IF, THEN AND ELSE CLAUSE. FOR SUBROUTINE CALLS, CONTROL IS PASSED TO 'SYNTH1' ONLY WHEN AN ARITHMETIC EXPRESSION APPEARS IN THE ARGUMENT LIST, OTHERWISE, THE QUADS ARE FORMED IN 'SYNTHF'.

107.4 USAGE

CALLING SEQUENCE:

CALL SYNTHE(ISEN, LAST, ISTACK, IS, ISGR, IER)

ISEN = INTERNAL SENTENCE FORM  
LAST = NUM. OF ENTRIES IN 'ISEN'  
ISTACK = OUTPUT OF PARSE  
IS = NUM. OF ENTRIES IN 'ISTACK'  
ISGR = SYNTAX GRAPH  
IER = ERROR FLAG: 0=GOOD; 8=ERROR

ORIGINAL PAGE IS  
OF POOR QUALITY

## MINIS PROGRAM SPECIFICATION

106.0 PROGRAM NAME- SYNTH1

106.1 PURPOSE

CONVERT THE INTERNAL SENTENCE AND PARSE OUTPUT STACK INTO AN EXECUTION STACK.

106.2 IDENTIFICATION

DATE- 27 NOVEMBER 1974

REVISION-

AUTHOR- D.R.SANDERS

COMPUTER- DCR 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 846 LOCATIONS (1516 OCTAL)

106.3 DESCRIPTION

'ISEN' FROM LEXCAL AND 'ISTACK' FROM PARSE ARE USED IN THIS ROUTINE ALONG WITH THE SYNTAX GRAPH TO FORM THE EXECUTION TRIPLES SAVED IN 'QUATRL'. 'ISTACK' IS SORTED IN DESCENDING ORDER USING THE FIRST WORD IN THE TWO-WORD ENTRIES AS A KEY. 1=SET DEFINITION 2=SUBSET DEFINITION 3=VARIABLE DEFINITION 4=IF STATEMENT 5=UNUSED 6=SUBROUTINE CALL (ASSIGNMENT TYPE) 7=SUBROUTINE CALL (EXECUTION TYPE)

106.4 USAGE

CALLING SEQUENCE:

CALL SYNTH1(ISEN, ISNBGN, LAST, ISNSAV, ISTACK, ISTBGN, IS, ISGR, IER)

ISEN = INTERNAL SENTENCE FORM  
ISNBGN= BEGINNING OF SYMBOLS IN 'ISEN'  
LAST = LAST ENTRY IN 'ISEN' TO BE SCANNED  
ISNSAV= SENTENCE TYPE  
ISTACK= OUTPUT OF PARSE  
ISTBGN= BEGINNING ENTRY IN 'ISTACK'  
IS = LAST ENTRY IN 'ISTACK'  
ISGR = SYNTAX GRAPH  
IER = ERROR FLAG: 0=GOOD; 8=ERROR.

## MINIS PROGRAM SPECIFICATION

109.0 PROGRAM NAME- TERFNC

109.1 PURPOSE

ACCESS ANY OF 5 TERMINAL FUNCTIONS PROVIDED BY TSOS,

109.2 IDENTIFICATION

DATE- 29 SEPTEMBER 1975

REVISION-

AUTHOR- D.R.SANDERS

COMPUTER- DCR 6024/3

LANGUAGE- ASSEMBLER

CORE SIZE- 13 WORDS (15 OCTAL)

ORIGINAL PAGE IS  
OF POOR QUALITY

109.3 DESCRIPTION

AN INTEGER FUNCTION CODE FROM 1 TO 5 IS PASSED TO S,TF IN TSOS AND THE CORRESPONDING TERMINAL FUNCTION IS EXECUTED,

109.4 USAGE

CALLING SEQUENCE:

```
CALL TERFNC(IFNC,IRES)
IFNC = FUNCTION CODE
      1=RESET SWITCHES
      2=RESET LINE COUNT
      3=LOAD SWITCHES
      4=LOAD LINE COUNT
      5=LOAD TERMINAL TYPE
IRES = RESULT OF FUNCTION EXECUTION
```

## MINIS PROGRAM SPECIFICATION

110.0 PROGRAM NAME- TERMNL

110.1 PURPOSE

SET THE OUTPUT LISTING DEVICE TO THE TERMINAL.

110.2 IDENTIFICATION

DATE- 26 FEBRUARY, 1975

REVISION-

AUTHOR- R.L.KEFFER (205)-883-1778

COMPUTER- DATACRAFT 6024

LANGUAGE- FORTRAN IV

CORE SIZE- DECIMAL 12 OCTAL

110.3 DESCRIPTION

THE OUTPUT LIST DEVICE, SPECIFIED IN COMMON 'IOARAY' AT IBUF(135) IS SET TO THE TERMINAL (1). THE NO. OF LINES PER PAGE IS SET TO 33(IBUF(139)).

110.4 USAGE

CALLING SEQUENCE:

CALL TERMNL

REQUIRED COMMONS

COMMON /IOARAY/ IBUF(134),LST,II(15)

# MINIS PROGRAM SPECIFICATION

111.0 PROGRAM NAME- UNPKER

111.1 PURPOSE

UNPACK THE FIELD VALUE AND DATA BASE RECORD POINTER FROM AN ENTRY  
OF A FIRST LEVEL INDEX FILE.

111.2 IDENTIFICATION

DATE- 19 MARCH 1975

REVISION-

AUTHOR- D.R.SANDERS

COMPUTER- DCR 6024

LANGUAGE- FORTRAN IV

CORE SIZE- 182 LOCATIONS (266 OCTAL)

ORIGINAL PAGE IS  
OF POOR QUALITY

111.3 DESCRIPTION

'IENTRY' IS PICKED UP FROM THE RIGHT-MOST 'NBENT' BITS OF THE  
SOURCE ARRAY. 'FLDVAL' IS PICKED UP AS AN INTEGER FROM THE SOURCE  
ARRAY'S NEXT 'NBFLD' BITS FROM RIGHT TO LEFT AND CONVERTED TO  
FLOATING POINT.

111.4 USAGE

CALLING SEQUENCE:

CALL UNPKER(IARRAY,NWORDS,IENTRY,NBENT,FLDVAL,NBFLD)

IARRAY = SOURCE ARRAY (1 BY 'NWORDS')

NWORDS = NUMBER OF WORDS IN 'IARRAY'

IENTRY = DATA BASE ENTRY OF SOURCE RECORD

NBENT = NUMBER OF BITS IN ENTRY POINTER AREA

FLDVAL = FIELD VALUE IN SOURCE RECORD

NBFLD = NUMBER OF BITS IN FIELD

## MINIS PROGRAM SPECIFICATION

112.0 PROGRAM NAME- VARCHK

112.1 PURPOSE

EXAMINE POSSIBLE QUAD AND FIELD LISTS TO PRODUCE A FIELD FOR WHICH AN INDEX FILE SCAN IS TO BE MADE.

112.2 IDENTIFICATION

DATE- 31 MARCH 1975

REVISION-

AUTHOR- D.R. SANDERS

COMPUTER- DCR 6024

LANGUAGE- FORTRAN IV

CORE SIZE- 523 LOCATIONS (1013 OCTAL)

112.3 DESCRIPTION

FIRST, THE POSSIBLE FIELD LIST IS CULLED OF ALL NON-INDEXED ENTRIES. IF ANY ENTRIES REMAIN, THEY ARE TESTED AGAINST A HIERARCHY OF CRITERIA IN ORDER TO ARRIVE AT A FIELD AND ITS ASSOCIATED BOUNDARY VALUES TO BE USED AS INPUT TO 'VARINX'. THE MOST DESIRABLE CRITERION IS THAT THE VARIABLE APPEARS IN THE POSSIBLE FIELD LIST TWICE- ONCE WITH AN UPPER BOUND AND ONCE WITH A LOWER BOUND. THE NEXT LOWER IN HIERARCHY IS THAT A FIELD APPEAR WITH ONE BOUNDARY, AND THE MAX OR MIN FIELD VALUE IS IMPLIED FOR THE SECOND BOUNDARY. (EXAMPLE: FIELD,LF,20 IMPLIES 'FIELD' IS BETWEEN THE MINIMUM FIELD VALUE AND 20.) THE THIRD POSITION IN HIERARCHY COVERS THE CASE OF A FIELD USED WITH AN ',EQ,' RELATION. THIS CASE TRANSLATES INTO AN UPPER BOUND AND A LOWER BOUND WHICH ARE THE SAME.

',GT,' AND ',LT,' RELATIONS ARE CONVERTED INTO ',GE,' AND ',LE,' RELATIONS BY USE OF THE MINIMUM VALUE AND SEGMENT SIZE IN THE SECOND LEVEL INDEX DEFINITION BLOCK.

112.4 USAGE

CALLING SEQUENCE:

CALL VARCHK(IPFLD,IPQUAD,IN,NFLD,V1,V2,ISEL)

IPFLD = POSSIBLE FIELD LIST

IPQUAD = POSSIBLE QUAD LIST

IN = NUMBER OF MEMBERS IN 'IPFLD' AND 'IPQUAD'

NFLD = FIELD NUMBER ON WHICH TO INDEX

0 IF NAME FOUND  
V1 = LOWER ROUND FOR VALUE OF THE FIELD  
V2 = UPPER ROUND FOR VALUE OF THE FIELD  
ISEL = POINTERS TO SELECTED QUADS IN 'IPQUAD' (2 WORDS)

ORIGINAL PAGE IS  
OF POOR QUALITY

ORIGINAL PAGE IS  
OF POOR QUALITY

## MINIS PROGRAM SPECIFICATION

113.0 PROGRAM NAME- VARINX

113.1 PURPOSE

BUILD A CANDIDATE FILE USING AN INDEXED VARIABLE.

113.2 IDENTIFICATION

DATE- 31 MARCH 1975

REVISION-

AUTHOR- D.R.SANDERS

COMPUTER- DCR 6024

LANGUAGE- FORTRAN IV

CORE SIZE- 378 LOCATIONS (572 OCTAL)

113.3 DESCRIPTION

'SCINDX' IS CALLED TO FORM A LIST OF CANDIDATES IN /REUSE/. FILE NAMES ARE SWAPPED AND THE CANDIDATE LIST IS MERGED WITH THE EXISTING CANDIDATE FILE. THE NEW ENTRIES CONTAIN ZERO IN THE SECOND AND THIRD WORDS. CONTROL IS RETURNED TO 'EXSETD'.

113.4 USAGE

CALLING SEQUENCE:

CALL VARINX(NFIELD,BLOWER,BUPPER,NCAND,ISETNO)

NFIELD = FIELD NUMBER

BUPPER = UPPER BOUNDARY VALUE

BLOWER = LOWER BOUNDARY VALUE

NCAND = NUMBER OF CANDIDATES (OUTPUT)

= CANDIDATE FIELD VALUE (INPUT)

ISETNO = SET NUMBER (INPUT)

# MINIS PROGRAM SPECIFICATION

114.0 PROGRAM NAME- VARVAL

114.1 PURPOSE

BUILD A BOUNDARY VALUE FOR 'VARCHK' AND INDICATE WHETHER THE VALUE IS A MAXIMUM OR MINIMUM.

114.2 IDENTIFICATION

DATE- 2 APRIL 1975

REVISION-

AUTHOR- D.R. SANDERS

COMPUTER- DCR 6024

LANGUAGE- FORTRAN IV

CORE SIZE- 164 LOCATIONS (244 OCTAL)

114.3 DESCRIPTION

THE POINTER TO THE QUAD TABLE ENTRY IS USED TO EXTRACT THE RELATIONAL OPERATOR FROM THE QUAD AND THE BOUNDARY VALUE THAT IS LOCATED IN THE SYMBOL TABLE. A FLAG IS SET INDICATING MAXIMUM OR MINIMUM BOUNDARY AND CONTROL IS PASSED BACK TO 'VARCHK'.

114.4 USAGE

CALLING SEQUENCE:

CALL VARVAL(IPFL, IQD, VALUE, MINMAX)

IFLD = FIELD NUMBER  
IQD = QUAD TABLE POINTER  
VALUE = BOUNDARY VALUE  
MINMAX = -1 FOR MINIMUM  
          +1 FOR MAXIMUM  
          0 FOR BOTH

ORIGINAL PAGE IS  
OF POOR QUALITY

## MINIS PROGRAM SPECIFICATION

115.0 PROGRAM NAME- WRITE

115.1 PURPOSE

TO OUTPUT A CREATED SYMBOL VALUE OR VALUES WHICH ARE NOT VARIABLE OVER A SET.

115.2 IDENTIFICATION

DATE- 12-16-75

REVISION-

AUTHOR- C.J. FARLESS (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 1133 LOCATIONS (2155 OCTAL)

115.3 DESCRIPTION

WRITE OUTPUTS THE VALUE OF A CREATED SYMBOL WHICH IS NOT RELATED TO A PARTICULAR SET OR IS A VALUE CALCULATED FROM A SUM OF A PARAMETER OVER A SET. OPTIONAL FORMATING CAN BE UTILIZED BY ADDING A FORMAT TITLE TO THE ARGUMENT LIST, THEREFORE ALLOWING EXPLANATORY MATERIAL TO BE PRINTED ALONG WITH THE VALUE, IN LIEU OF A REQUESTED FORMAT THE VALUE IS PRINTED OUT WITH ITS SYMBOL NAME. MULTIPLE PARAMETERS ARE OUTPUT ON THE SAME LINE UNLESS MORE SPACE IS REQUIRED, EXAMPLES:

WRITE(X)  
RESULT: X= 10,21

WRITE(X,Y)  
RESULT: X= 10,21 Y= 20,42

WRITE(X,Y,FMATXY)  
FMATXY=(1X,'THE VALUE OF X IS ',F5,2,4X,'THE VALUE OF Y IS ',F5,2)  
RESULT: THE VALUE OF X IS 10,21 THE VALUE OF Y IS 20,42

115.4 USAGE

CALLING SEQUENCE:

WRITE(ARG1,ARG2,...)

ARG1,ARG2,...=THE CREATED SYMBOL NAME OR FORMAT TITLE TO PRINT,

## MINIS PROGRAM SPECIFICATION

116.0 PROGRAM NAME- WRTRUF

116.1 PURPOSE

TO BUFFER THE MERGE OUTPUT INTO BLOCKS TO INCREASE FILE WRITE AND TAPE WRITE SPEED.

116.2 IDENTIFICATION

DATE- 3-15-76

REVISION-

AUTHOR- C.J.FARLESS (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CODE SIZE- 609 LOCATIONS (1141 OCTAL)

116.3 DESCRIPTION

116.4 USAGE

CALLING SEQUENCE:

CALL WRTRUF(NAM1, ISTART, ISTOP, NAM2, NREC, IWDEV, NBUF, IEND, MER)

NAM1 = FILE DEFINITION OF INPUT FILE.  
ISTART = FIRST RECORD OF INPUT.  
ISTOP = LAST RECORD OF INPUT.  
NAM2 = FILE DEFINITION OF OUTPUT FILE.  
NREC = NEXT RECORD NUMBER OF OUTPUT TO WRITE.  
IWDEV = OUTPUT FILE DEVICE NO. 0=DISC, OTHER=TAPE.  
IEND = END FLAG. WRITE ALL BUFFER IF SET.  
MER = MERITS FLAG--BIT NO OF FIELD FOR SEQUENCE NO.

## MINIS PROGRAM SPECIFICATION

117.0 PROGRAM NAME- WRTDAT

117.1 PURPOSE

STORE DATA VALUES INTO A DATA ARRAY AND OUTPUT HEADERS AND DATA.

117.2 IDENTIFICATION

DATE- 7-8-75

REVISION-

AUTHOR- C. J. FARLESS (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 843 LOCATIONS (1513 OCTAL)

117.3 DESCRIPTION

FOR THE RECORDS IN A GIVEN SET(NSFT), THE REQUIRED FIELDS ARE EXTRACTED AND VARIABLES CALCULATED. THE RESULTANT VALUES ARE STORED IN THE OUTPUT ARRAY. THE HEADERS ARE WRITTEN FROM THE FORMAT IN ARRAY 'FMAT' AND DATA IS OUTPUT FROM THE FORMAT IN ARRAY 'FDMAT'.

117.4 USAGE

CALLING SEQUENCE:

CALL WRTDAT(ISTART,ISTOP,LCNT,NSET,LCNTHD)

ISTART=NO. OF FIRST ITEM TO OUTPUT

ISTOP =NO. OF LAST ITEM TO OUTPUT

LCNT =NO. OF LINES PRINTED

NSET =SET NUMBER

LCNTHD=NO. OF LINES IN HEADER

# MINIS PROGRAM SPECIFICATION

118.0 PROGRAM NAME- WRIDEF

118.1 PURPOSE

STORE FIELD DEFINITIONS IN IDEF,

118.2 IDENTIFICATION

DATE- 4-22-75

REVISION-

AUTHOR- C. J. FARLESS (205) 883-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 364 LOCATIONS (554 OCTAL)

118.3 DESCRIPTION

USING THE SUBROUTINE ARGUMENTS, THE 10 WORD FILE DEFINITION BLOCK IS CONSTRUCTED. IF A DEFINITION ALREADY EXISTS FOR THE INDEX FILE, THE NEW DEFINITION IS STORED IN THE OLD LOCATION, IF ONE DOES NOT EXIST, THE DEFINITION IS STORED IN THE FIRST AVAILABLE LOCATION AFTER ITS INDEXED FILE. IN THE CASE OF A LEVEL 1 INDEX FILE, THIS WOULD BE THE FIRST AVAILABLE LOCATION IN THE INDEX FILE DEFINITIONS. IF THERE ARE NO AVAILABLE LOCATIONS, THE ROUTINE SETS THE NUMBER OF RECORDS IN THE FILE TO 0 AND RETURNS TO THE CALLING PROGRAM.

118.4 USAGE

CALLING SEQUENCE:

CALL WRIDEF(IFLNO,LEVEL,ITYPE,NREC,NBYTE,XMIN,SIZE,NBPTR,FINAM)

IFLNO = FIELD NUMBER

LEVEL = INDEX LEVEL

ITYPE = TYPE INDEXING

NREC = NUMBER OF RECORDS IN FILE (NUMBER OF ENTRIES)

NBYTE = NUMBER OF BYTES PER RECORD

XMIN = FOR LEVEL 1=FIELD NUMBER  
FOR LEVEL 2=MINIMUM VALUE

SIZE = FOR LEVEL 1=NUMBER OF BITS IN FIELD  
FOR LEVEL 2=SEGMENT SIZE

ORIGINAL PAGE IS  
OF POOR QUALITY

NBPTR = NUMBER OF BITS NEEDED FOR POINTER

FINAM = TEN WORD FILE DEFINITION BLOCK

WORD	CONTENTS
1-3	FILE NAME XFLDLI X=MESHED CHARACTER OF ALL LETTERS IS DATA BASE NAME FLD-3 DIGIT FIELD NUMBER L-1 DIGIT LEVEL NUMBER T-1 DIGIT TYPE NUMBER
4	NUMBER OF RECORDS IN FILE
5	NUMBER OF BYTES/RECORD
6-7	LEVEL 1 FIELD NUMBER(WORD 6) LEVEL 2 MINIMUM VALUE
8-9	LEVEL 1 NUMBER OF BITS IN FIELD(WORD 8) LEVEL 2 SEGMENT SIZE
10	NUMBER OF BITS IN POINTER

# MINIS PROGRAM SPECIFICATION

119.0 PROGRAM NAME- XFER

119.1 PURPOSE

TRANSFER A CONTIGUOUS STRING OF CHARACTERS FROM ONE ARRAY TO ANOTHER

119.2 IDENTIFICATION

DATE- 8 NOVEMBER 1974

REVISION-

AUTHOR- D.R.SANDERS (205)-883-1778

COMPUTER- DCR 6024

LANGUAGE- FORTRAN IV

CORE SIZE- 42 WORDS (52 OCTAL)

119.3 DESCRIPTION

'N' CONTIGUOUS BYTES ARE TRANSFERRED FROM 'ISRC' BEGINNING WITH THE 'I' TH CHARACTER INTO 'IDES' BEGINNING AT THE 'J' TH CHARACTER.

119.4 USAGE

CALLING SEQUENCE:

CALL XFER(ISRC,I,IDES,J,N)

ISRC = SOURCE ARRAY

I = STARTING CHARACTER IN ISRC

IDES = DESTINATION ARRAY

J = STARTING CHARACTER IN IDES

N = NUMBER OF CHARACTERS TO TRANSFER

ORIGINAL PAGE IS  
OF POOR QUALITY

## MINIS PROGRAM SPECIFICATION

120.0 PROGRAM NAME- LACTUL(NPDS,LAT,LON)

120.1 PURPOSE

DETERMINE IF THE PICTURE COORDINATES IN AN ARRAY ARE IN A SPECIFIED AREA.

120.2 IDENTIFICATION

DATE- 17 SEPTEMBER, 1975

REVISION- ADAPTED FROM A MERITS SUB OF THE SAME NAME.

AUTHOR- R.L.KEFFER (205) 683-1778

COMPUTER- DATACRAFT 6024/3

LANGUAGE- FORTRAN IV

CORE SIZE- 210 LOCATIONS (322 OCTAL)

120.3 DESCRIPTION

EACH PHOTO IN COMMON REUSE IS CHECKED TO SEE IF IT COVERS THE SPECIFIED POINT (LAT,LON). IF NOT, THE ENTRY IS DELETED FROM REUSE AND THE LIST MOVED UP 1 NOTCH. SUBROUTINE HITES DOES THE ACTUAL COORDINATE CHECKING. THE PICTURE POSITION IS NORMALIZED TO NORTH LATITUDE, AND 90 WEST LONGITUDE TO SIMPLIFY HITES TASK.

120.4 USAGE

CALLING SEQUENCE:

CALL LACTUL(NPDS,LAT,LON)

NPDS = NUMBER OF POSSIBLE PHOTOS. SET TO ACTUAL AT END,

LAT = LATITUDE OF POINT TO BE CHECKED,

LON = LONGITUDE OF POINT TO BE CHECKED.

**APPENDIX C**  
**PROGRAM FLOW CHARTS**

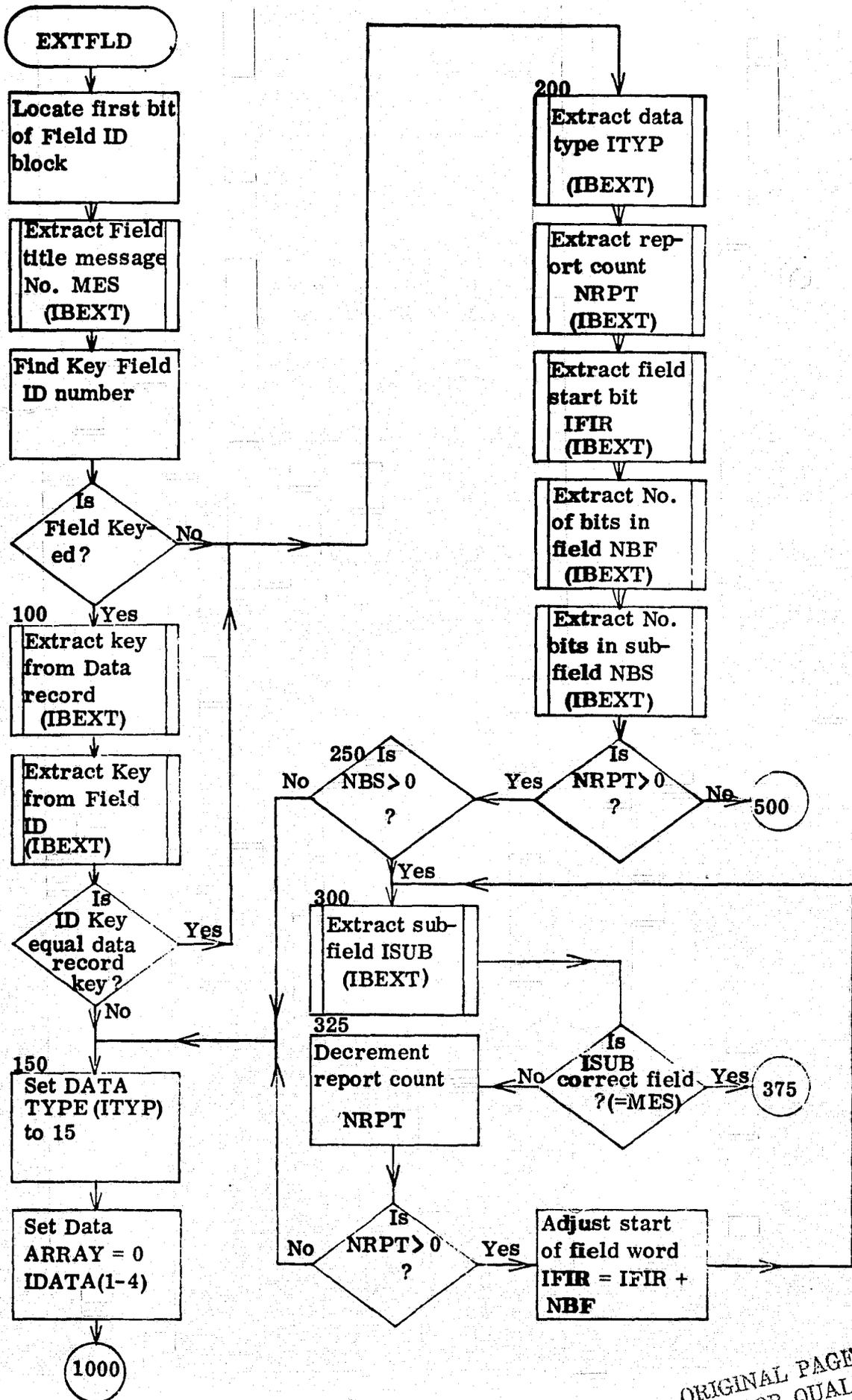
**ORIGINAL PAGE IS  
OF POOR QUALITY**

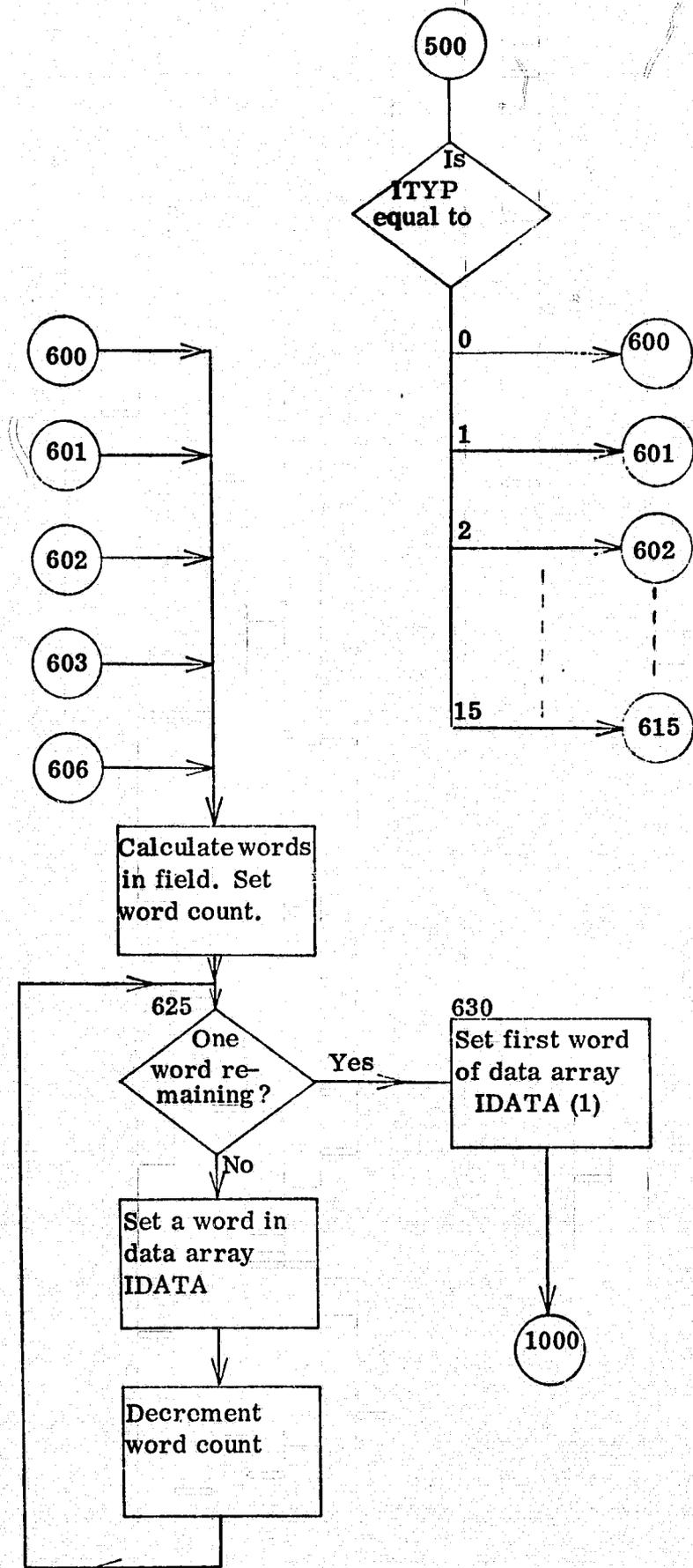
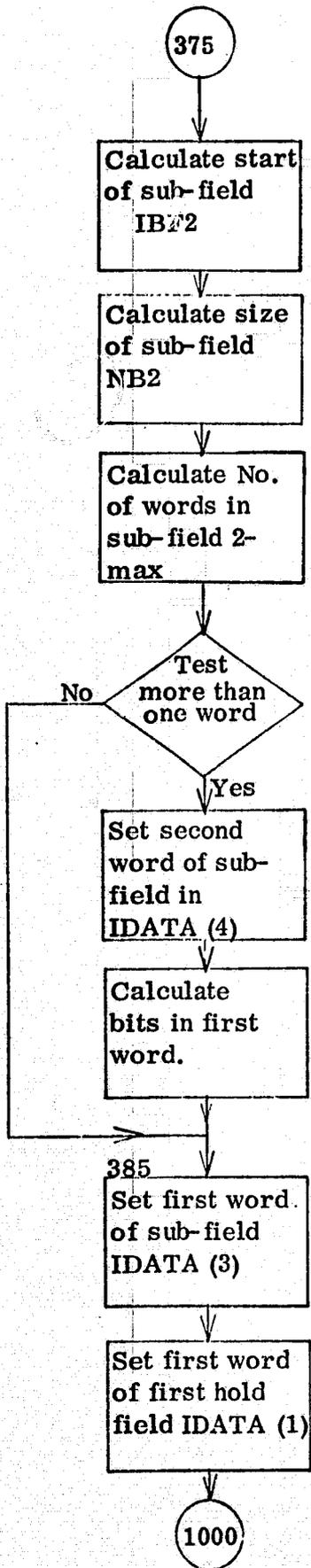
**PRECEDING PAGE BLANK NOT FILMED**

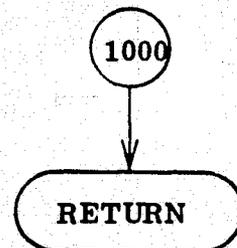
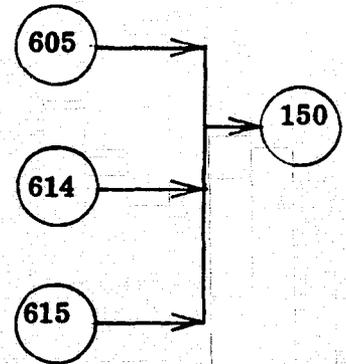
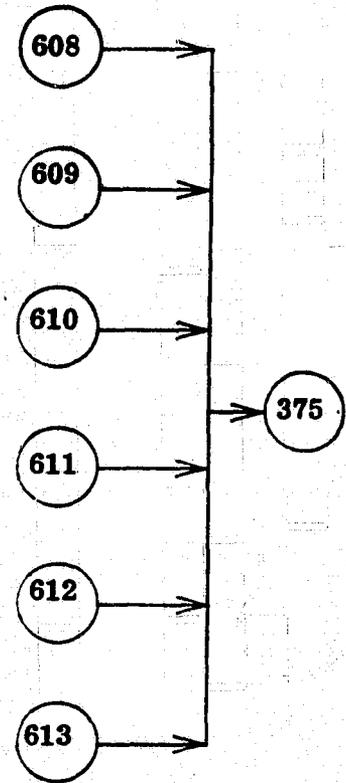
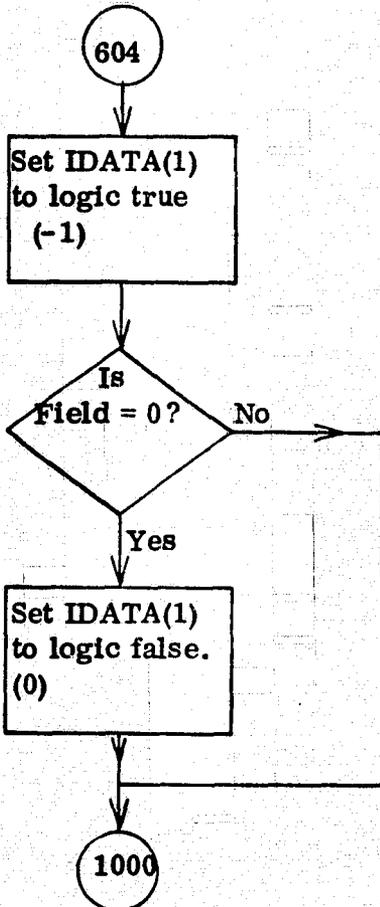
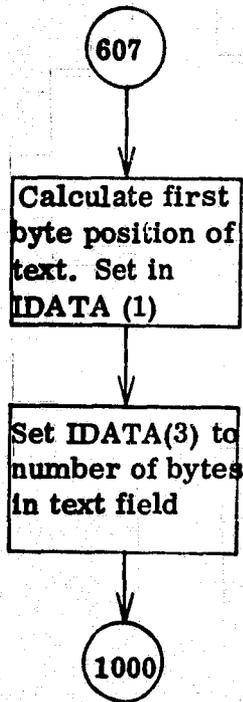
Appendix C includes program flowcharts for the following programs:

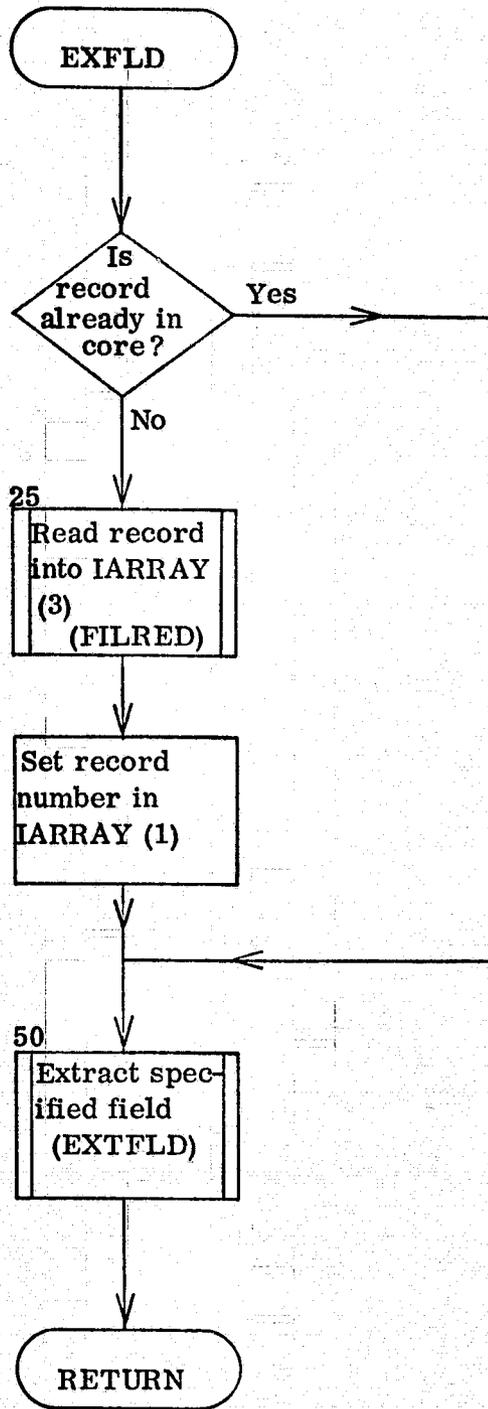
<u>Name</u>	<u>Page</u>	<u>Name</u>	<u>Page</u>
EXTFLD	C-1	FORMAT	C-33
EXFLD	C-4	BISORT	C-38
ININT	C-5	MOVOUT	C-41
IBLANK	C-6	GENDEX	C-42
KUMPAR	C-7	INDEX	C-44
INTGER	C-8	NORM	C-47
LININ	C-9	OUTPUT	C-48
FINFLD	C-10	EXCANS	C-52
FLDFIN	C-11	EXOPRN	C-53
MESOUT	C-20	EXQUAD	C-55
MESMAN	C-21	EXVARD	C-58
SETPRI	C-28	SUBSET	C-59
WRITFL	C-29	REVSCN	C-60
SORTX	C-30	RVSCAN	C-61
MERGER	C-31	SUMSET	C-62
INFMAT	C-32		

ORIGINAL PAGE IS  
OF POOR QUALITY

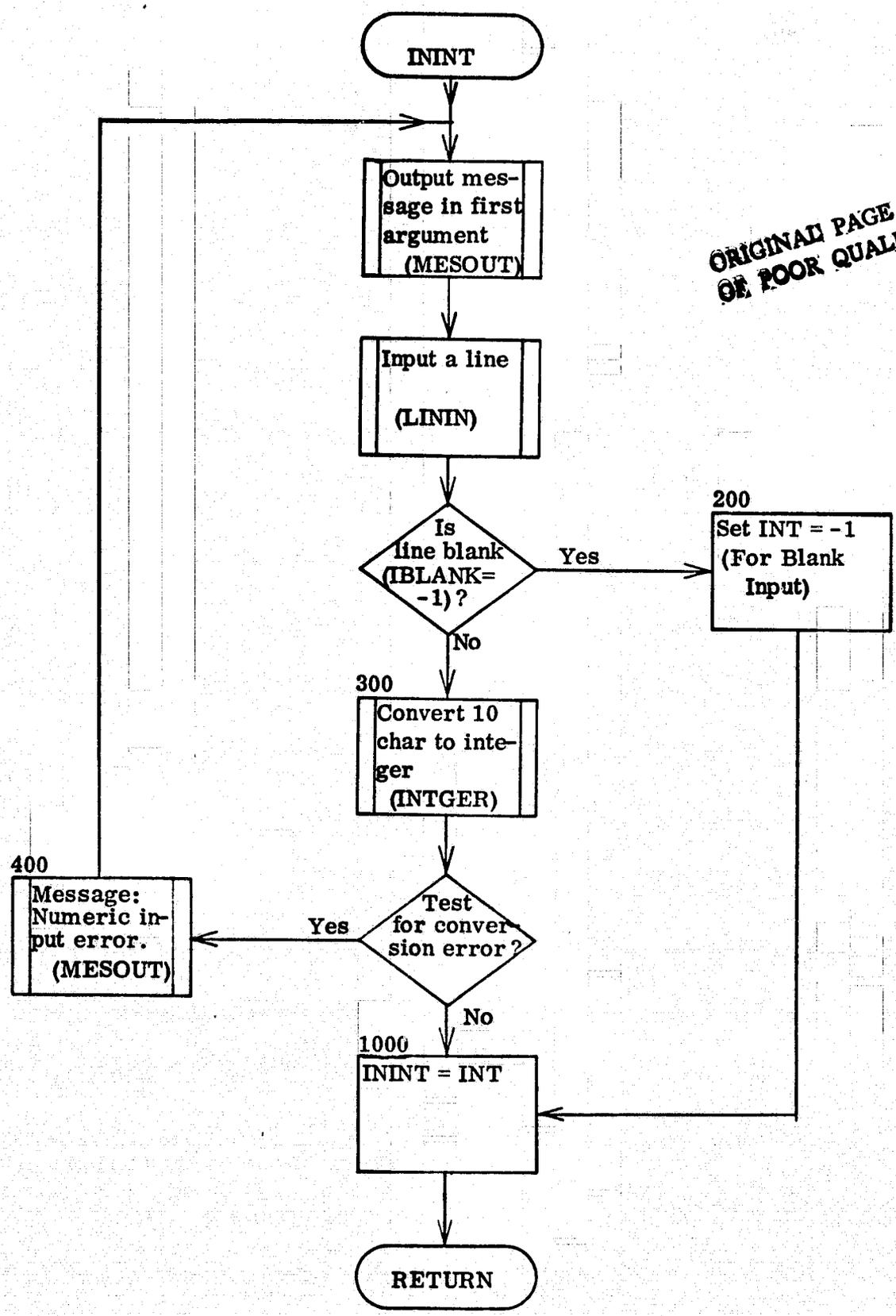


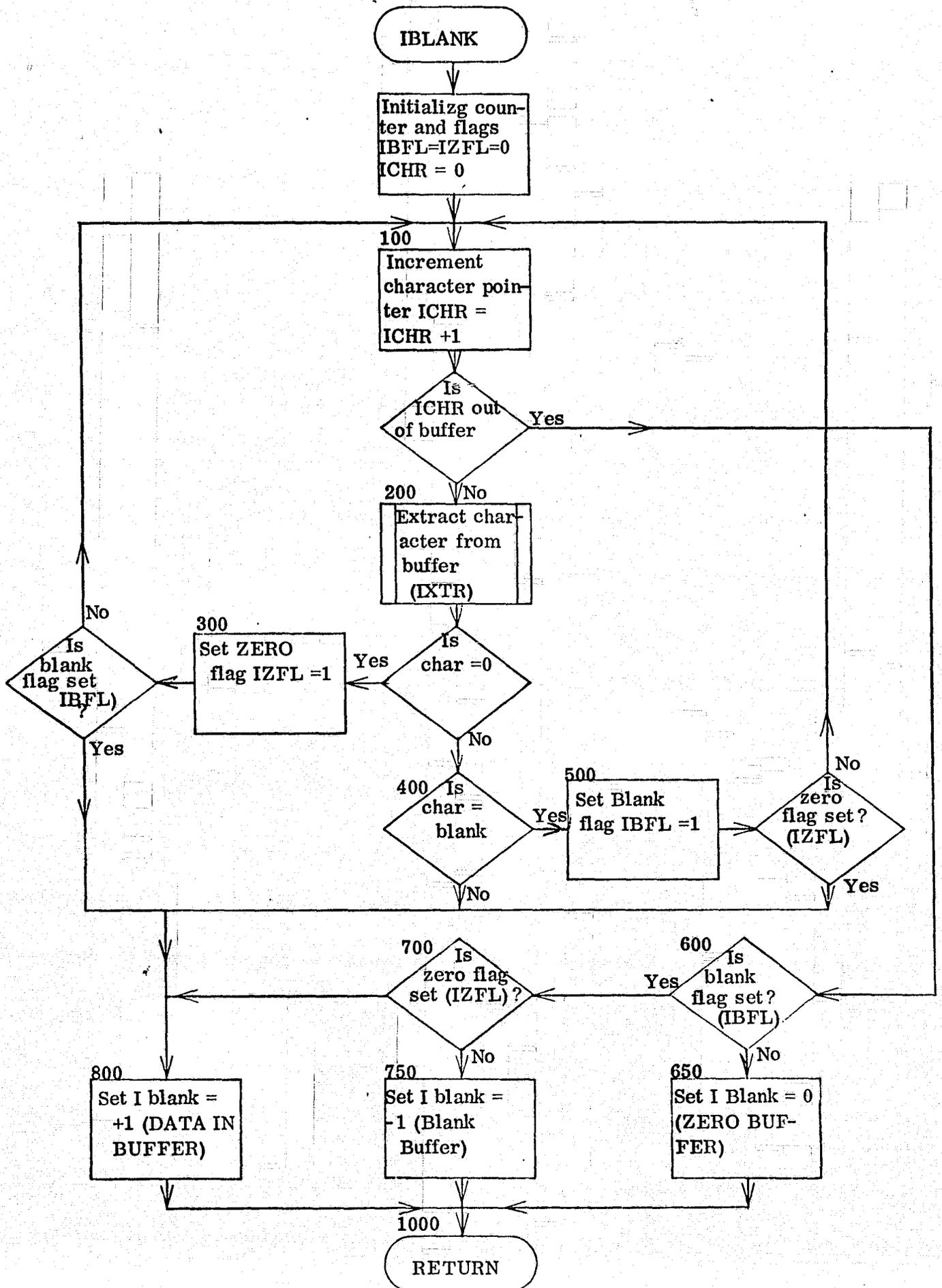


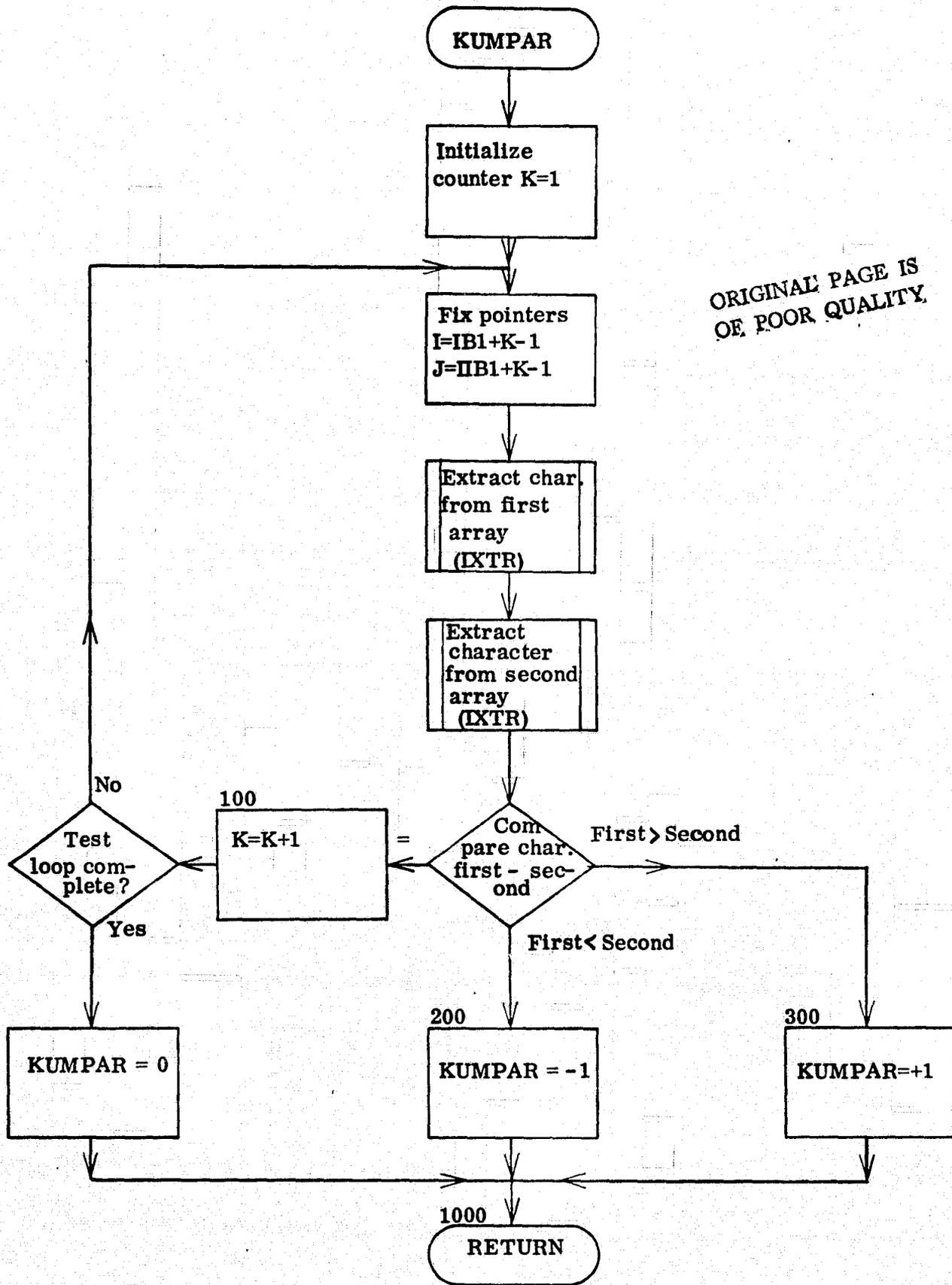




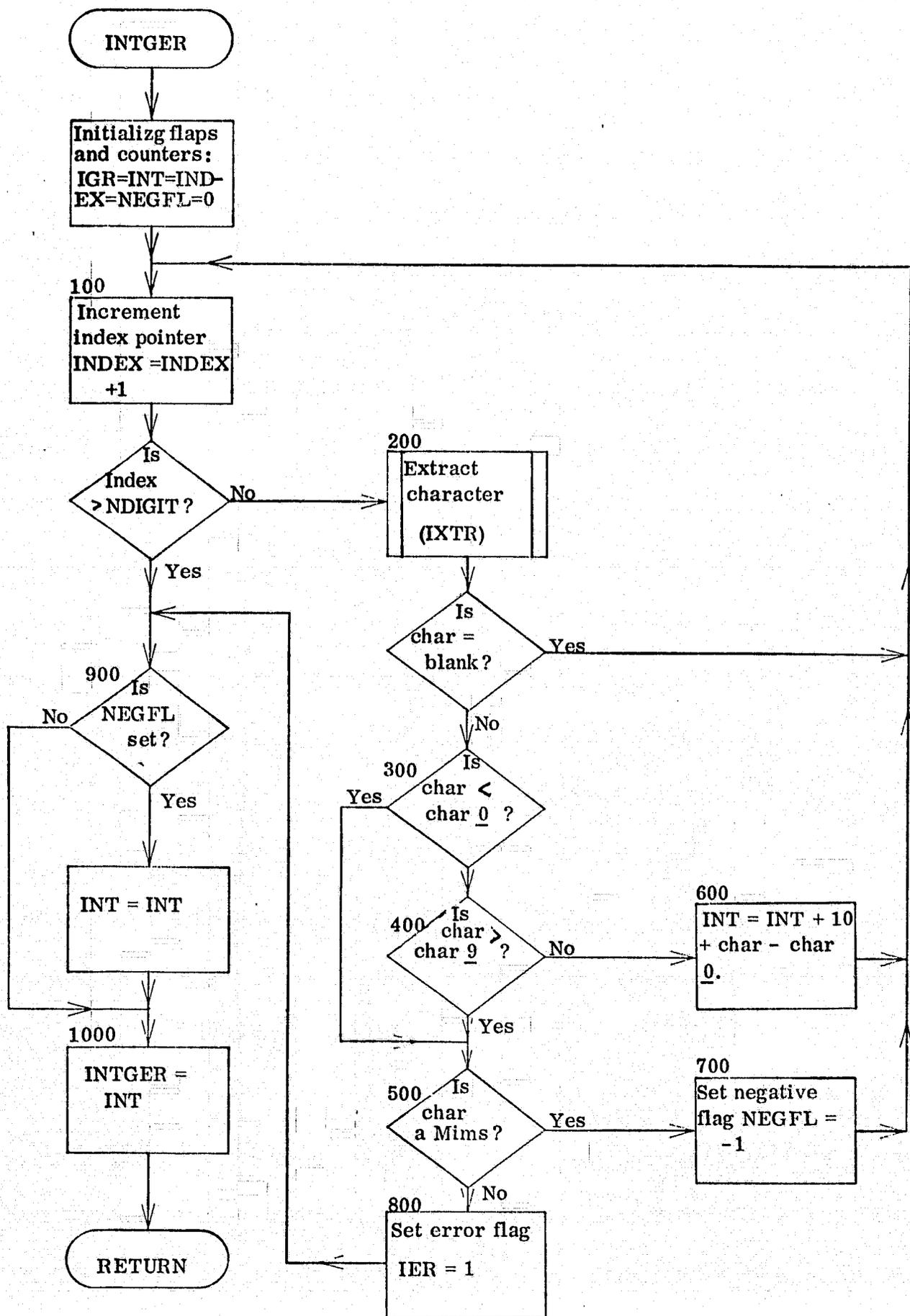
ORIGINAL PAGE IS  
OF POOR QUALITY



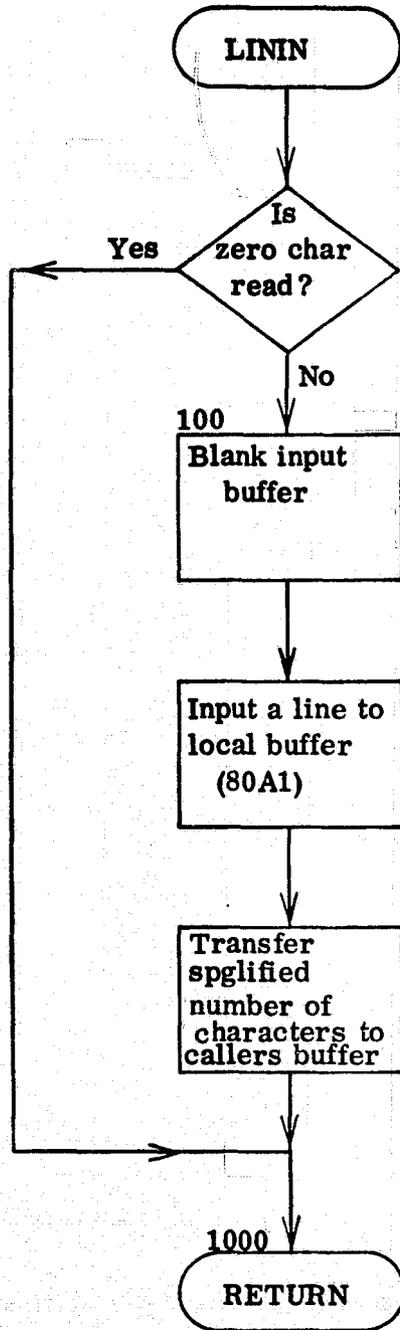


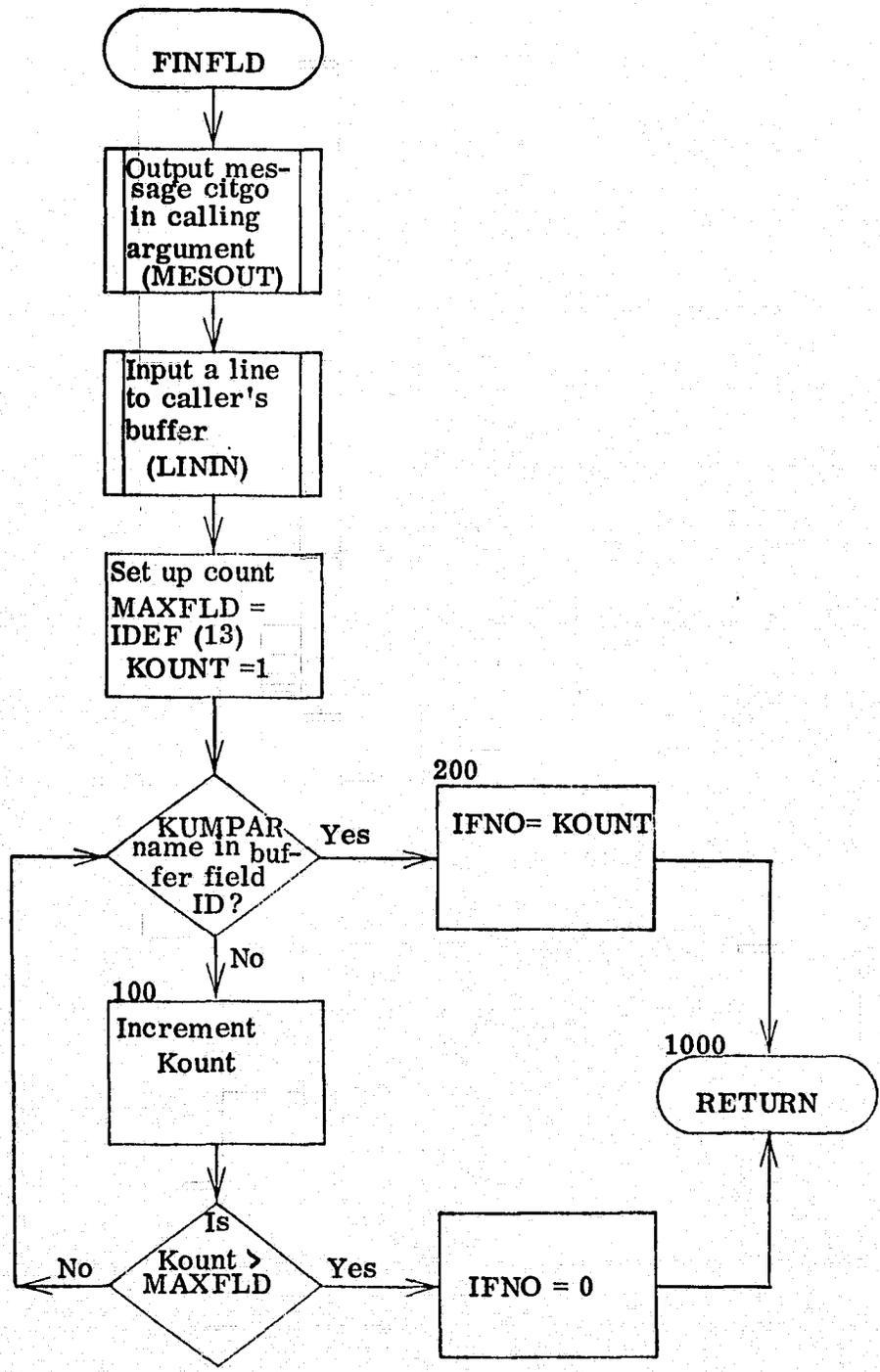


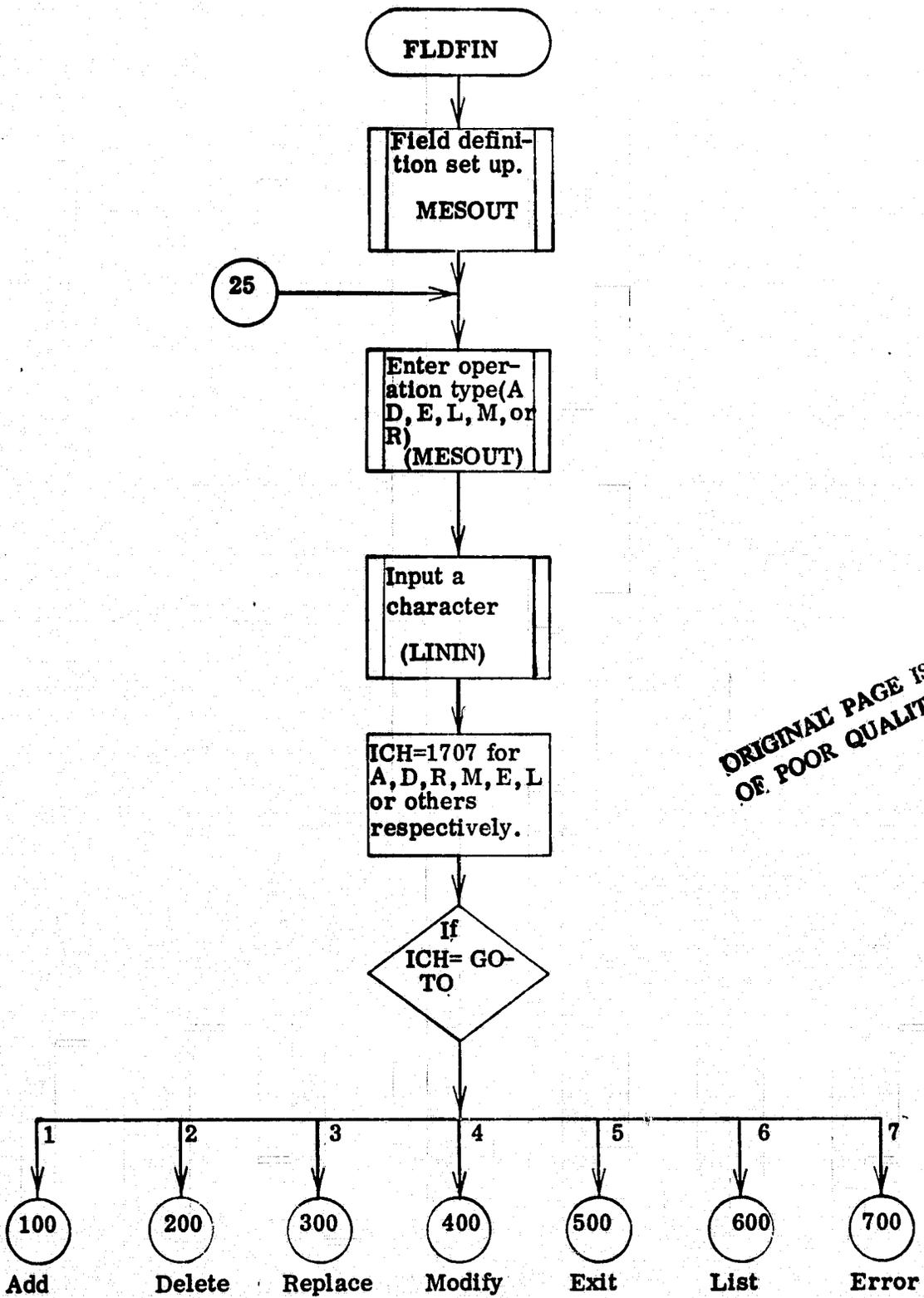
ORIGINAL PAGE IS  
OF POOR QUALITY.



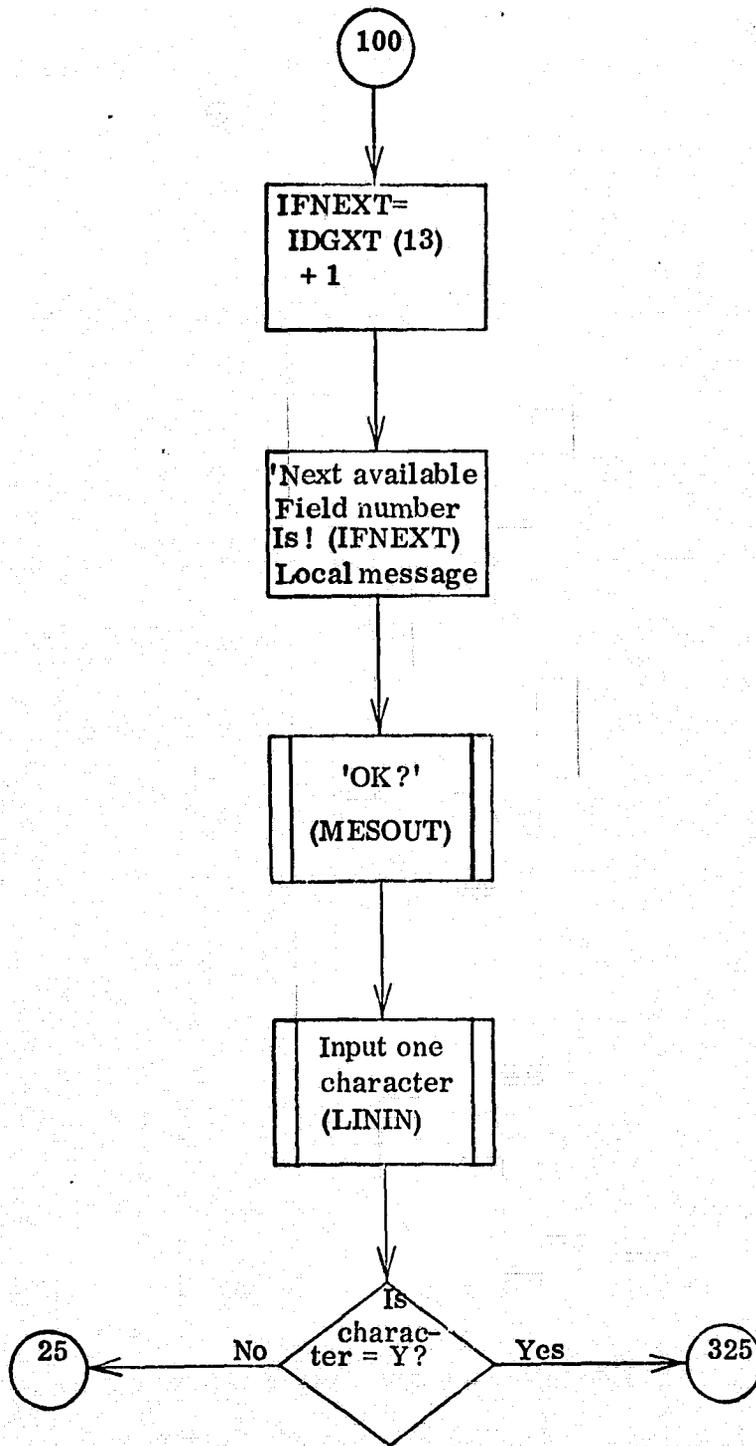
**ORIGINAL PAGE IS  
OF POOR QUALITY**

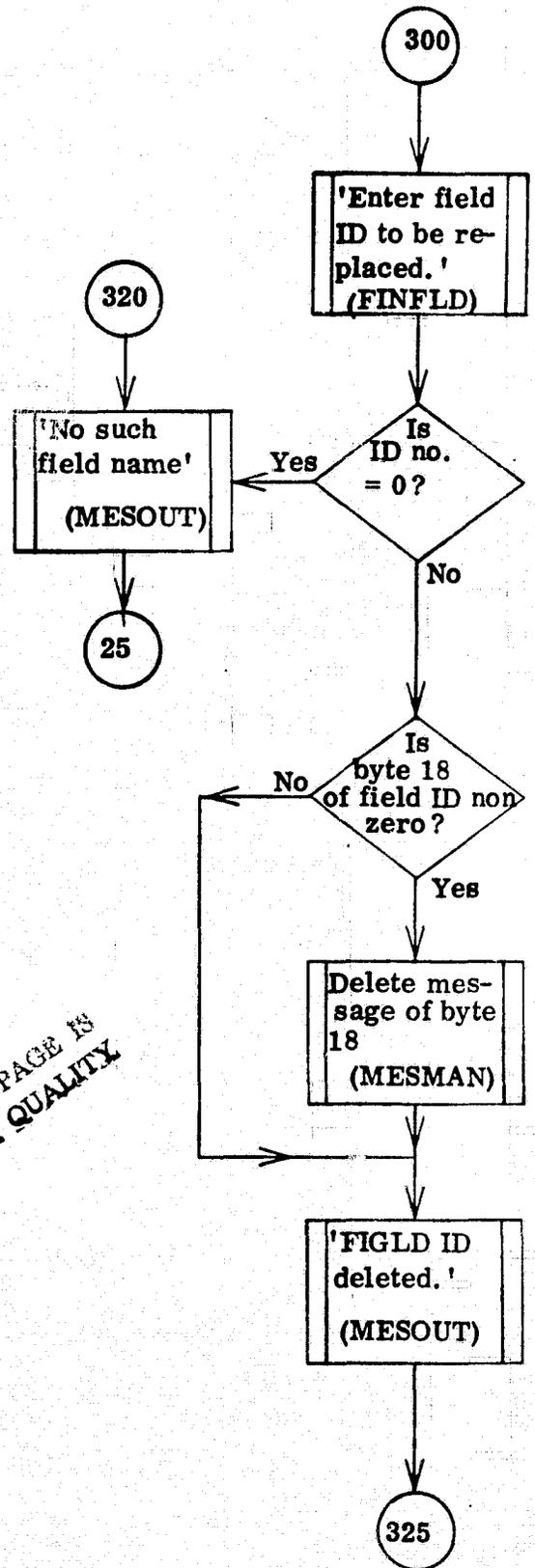
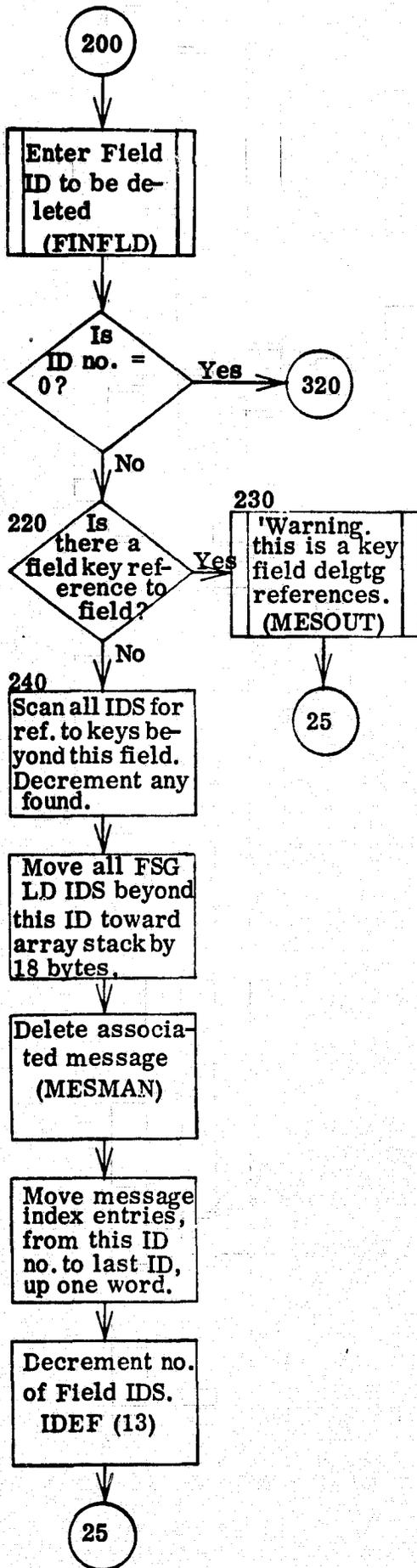




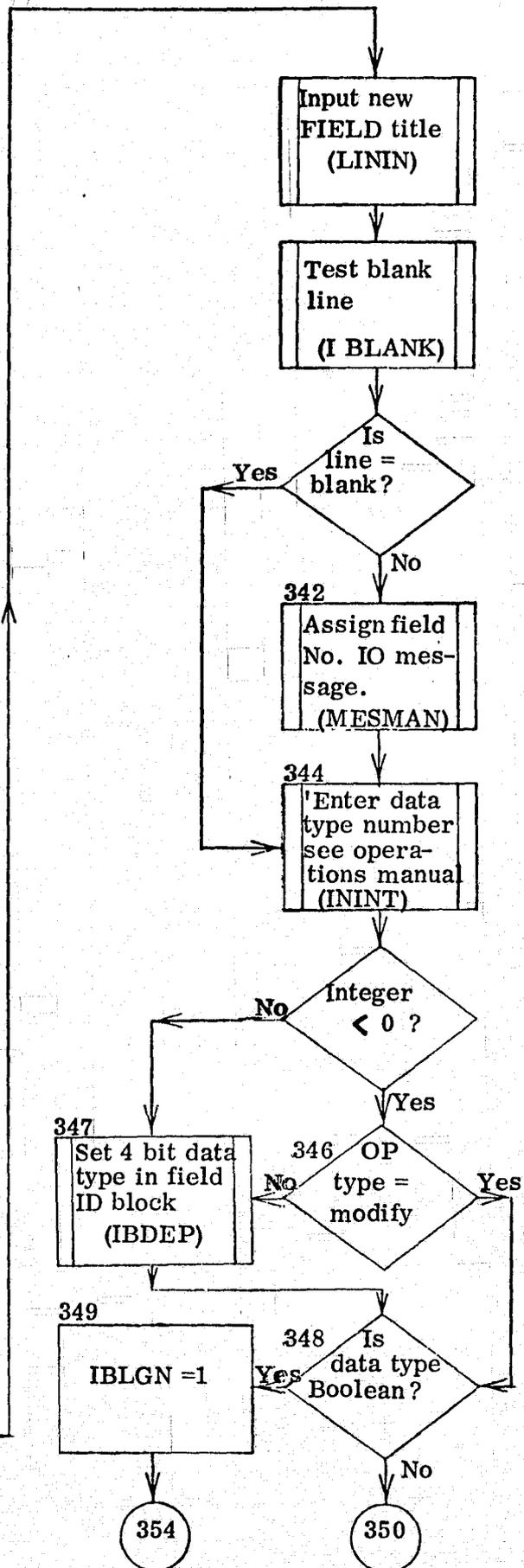
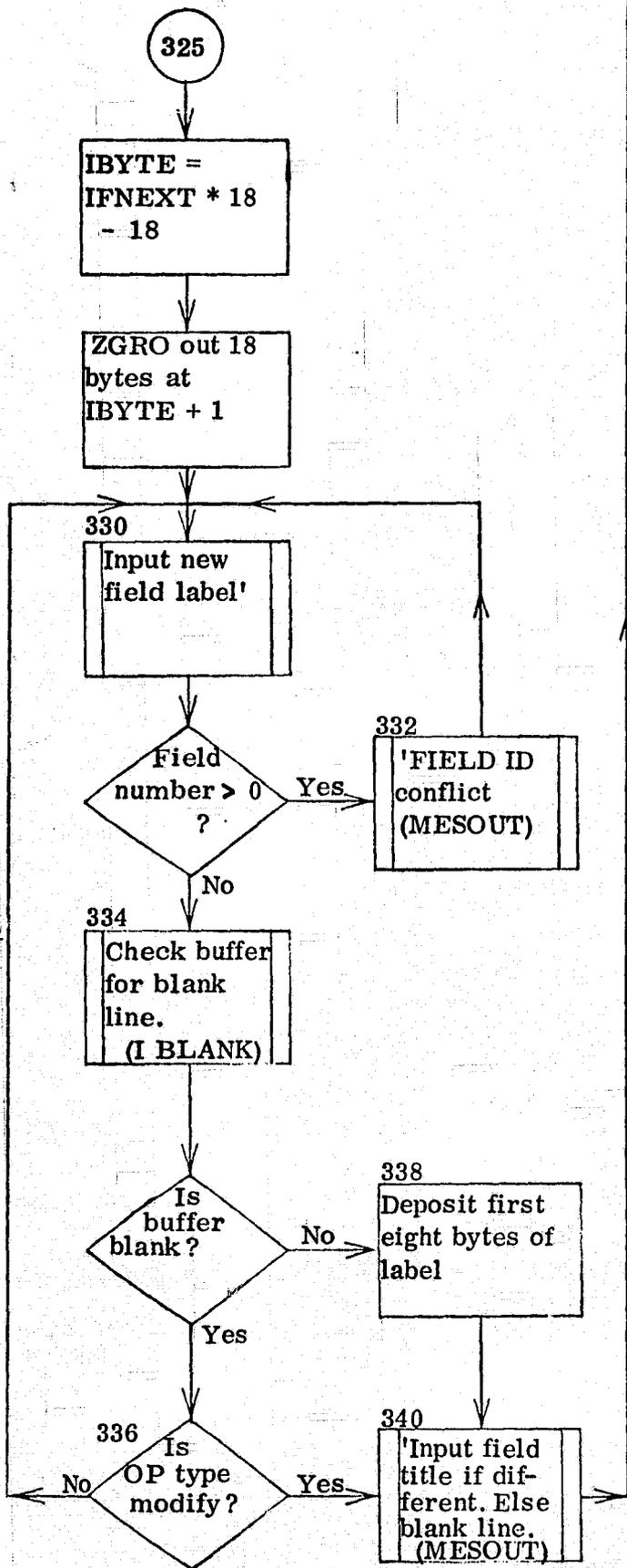


ORIGINAL PAGE IS  
OF POOR QUALITY

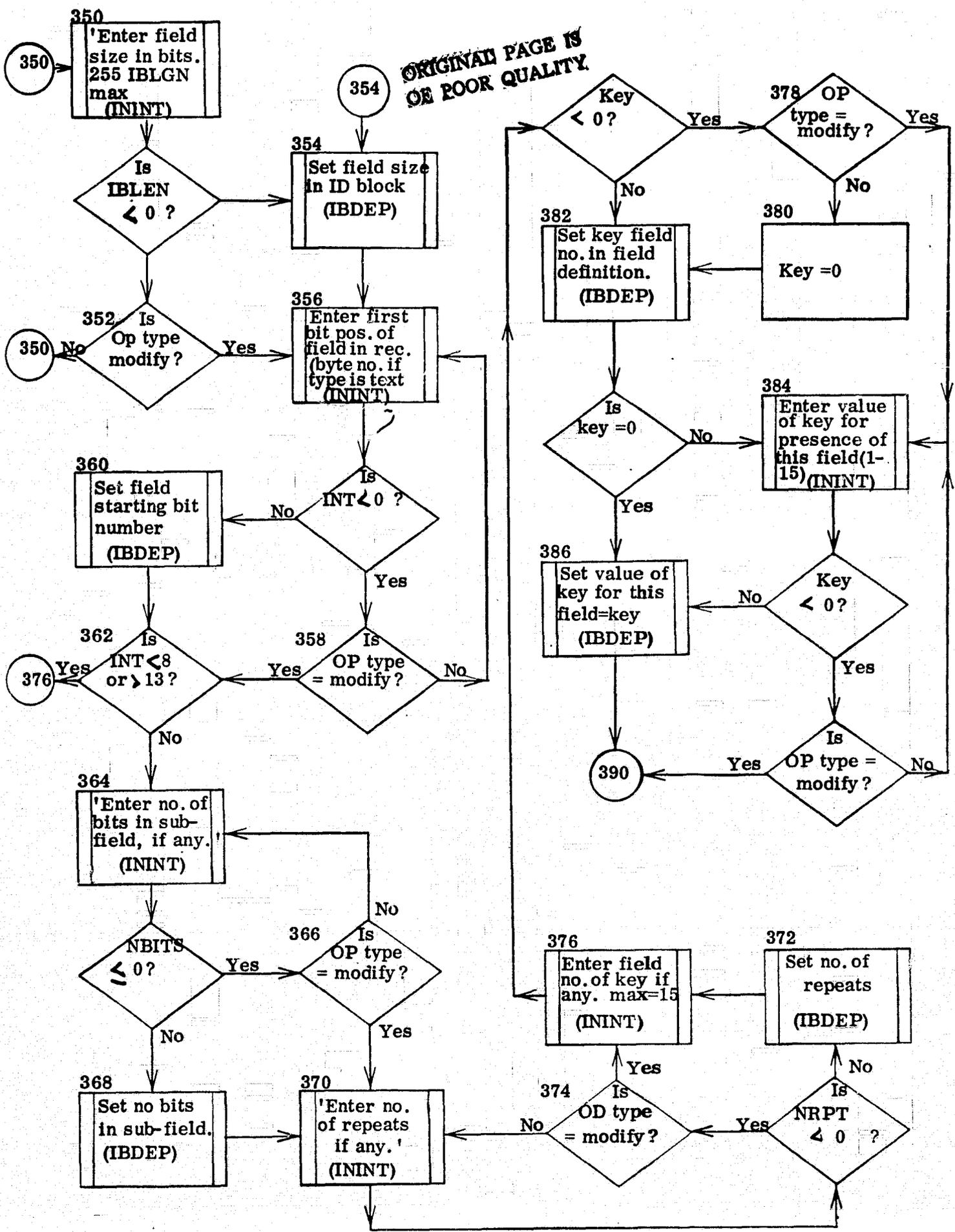


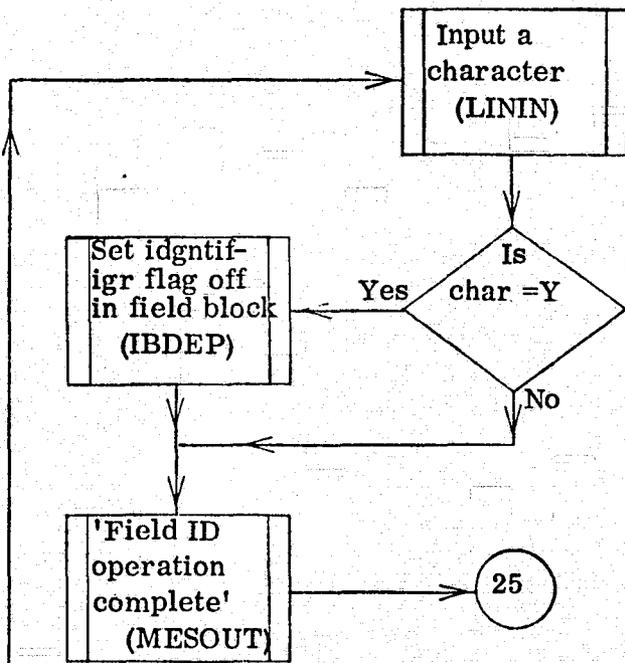
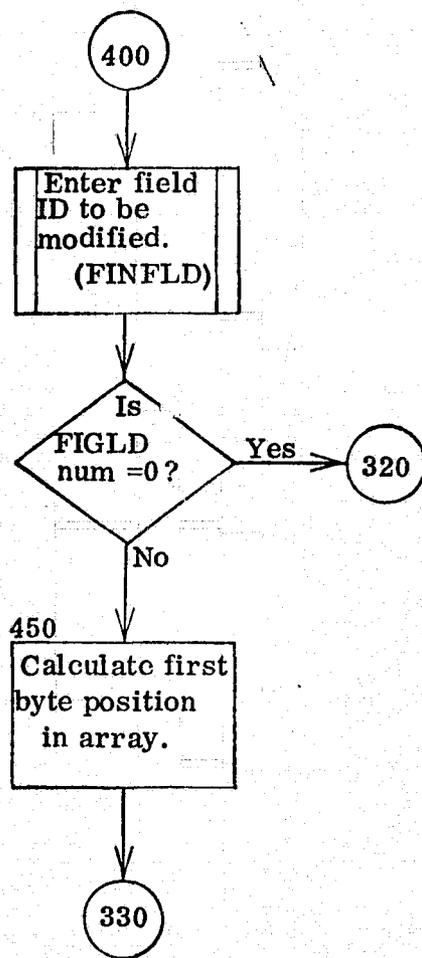
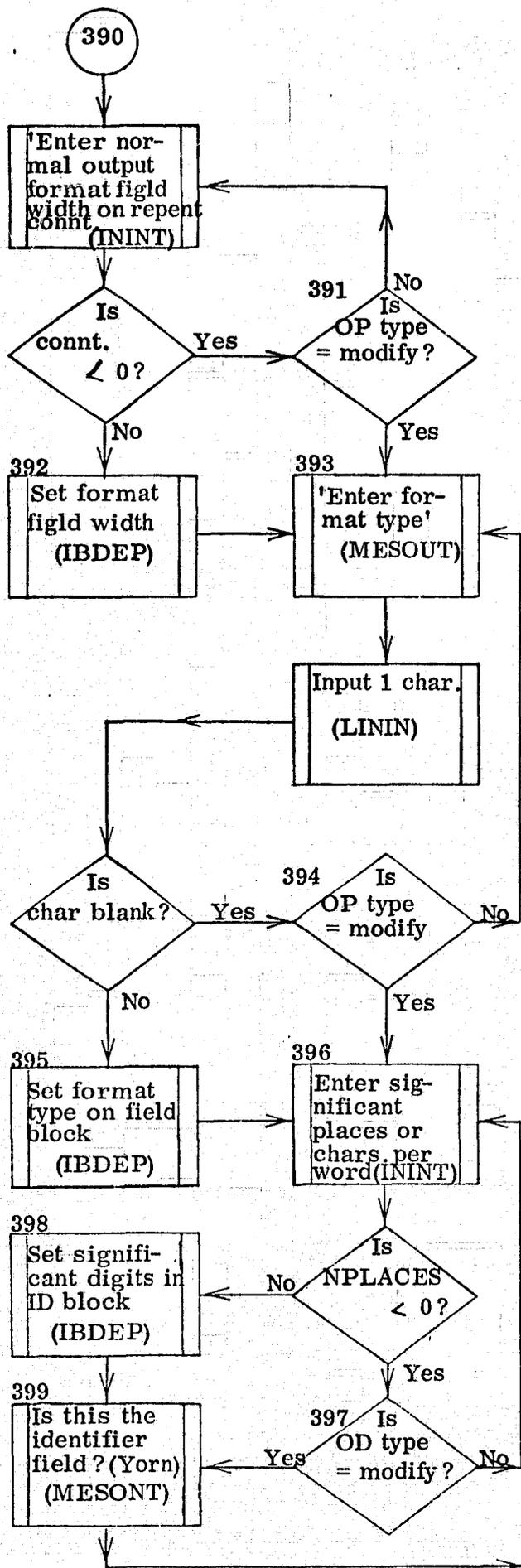


ORIGINAL PAGE IS OF POOR QUALITY

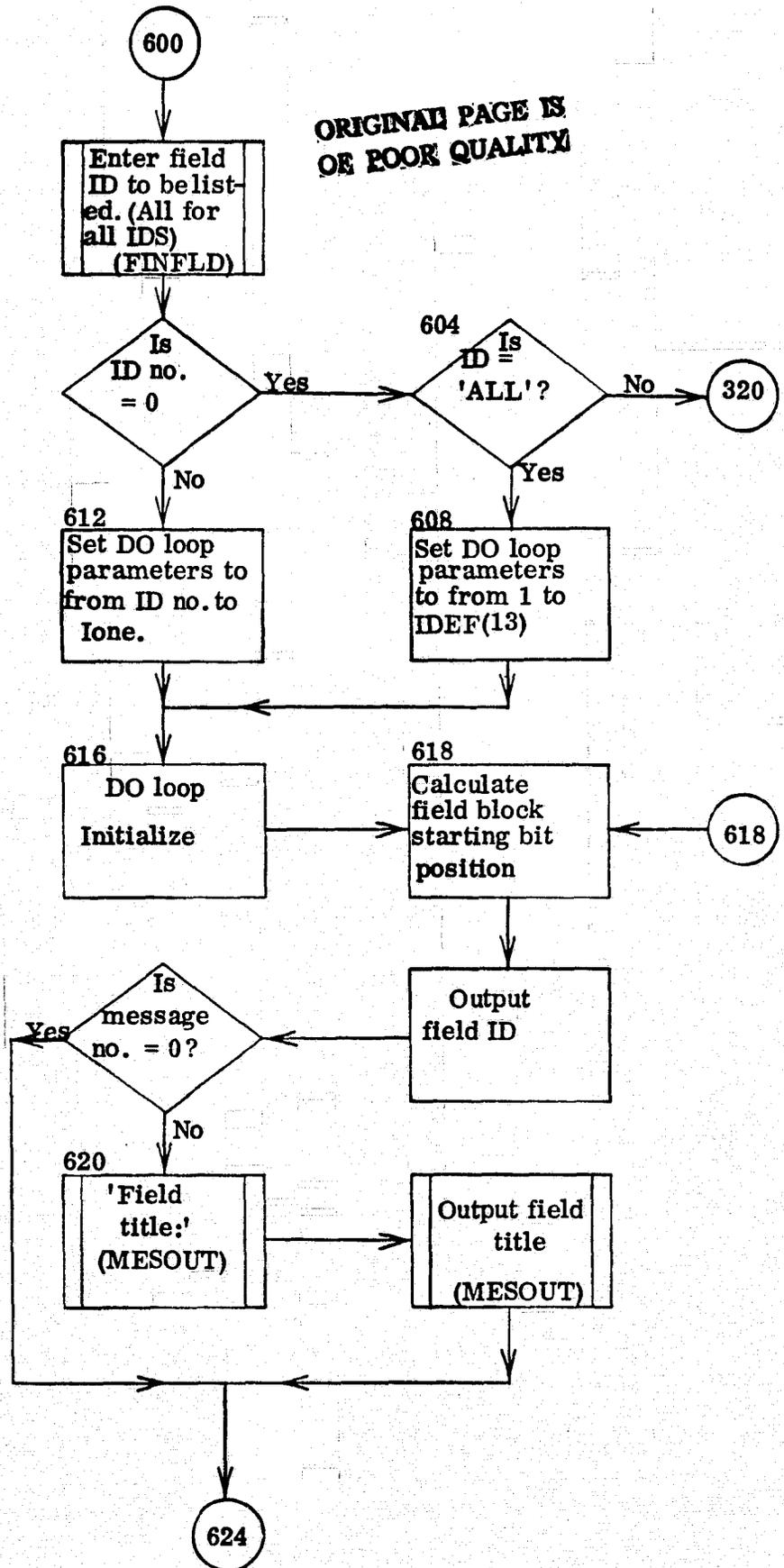
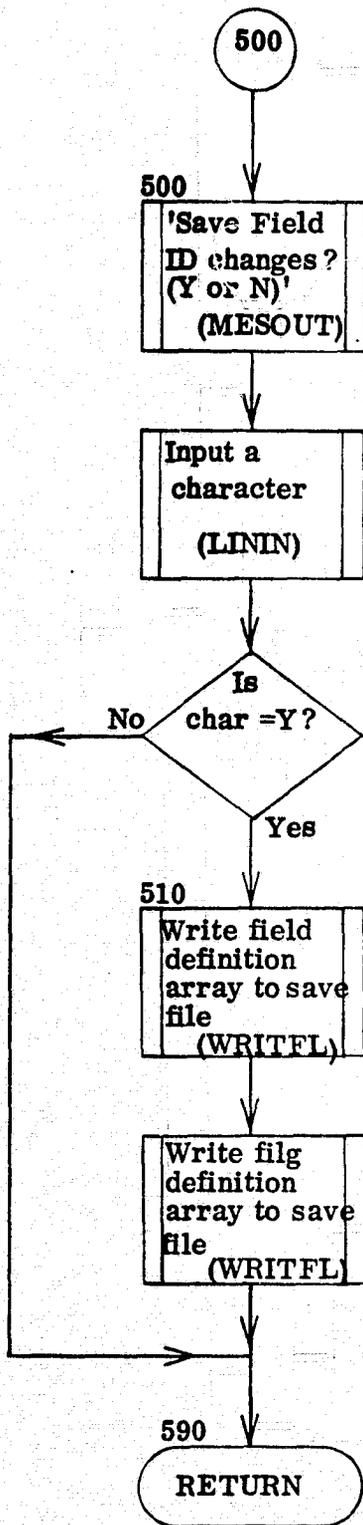


ORIGINAL PAGE IS OF POOR QUALITY.

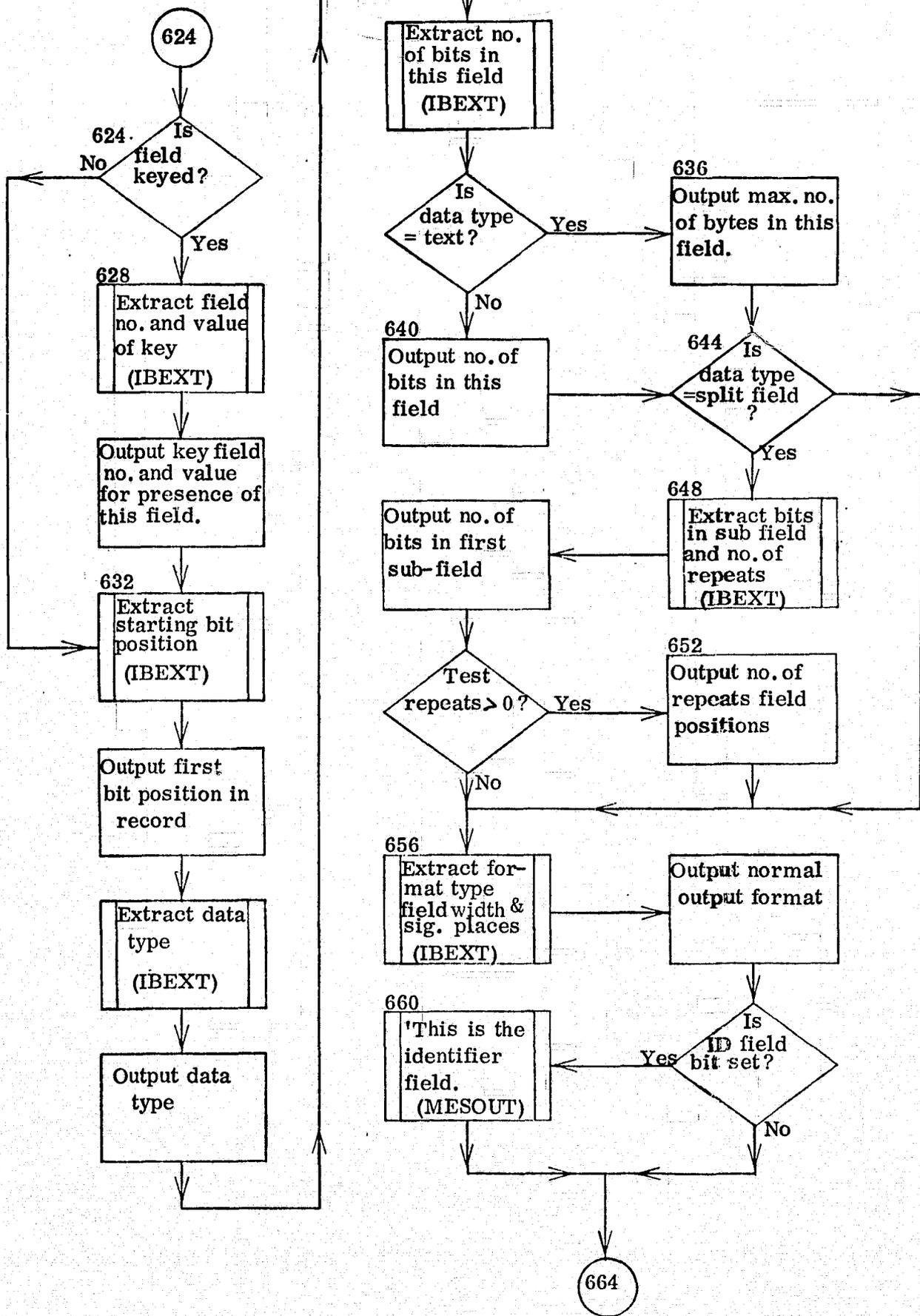




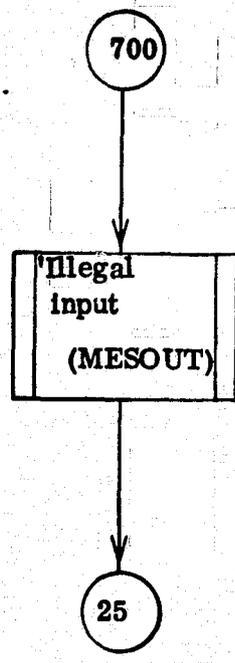
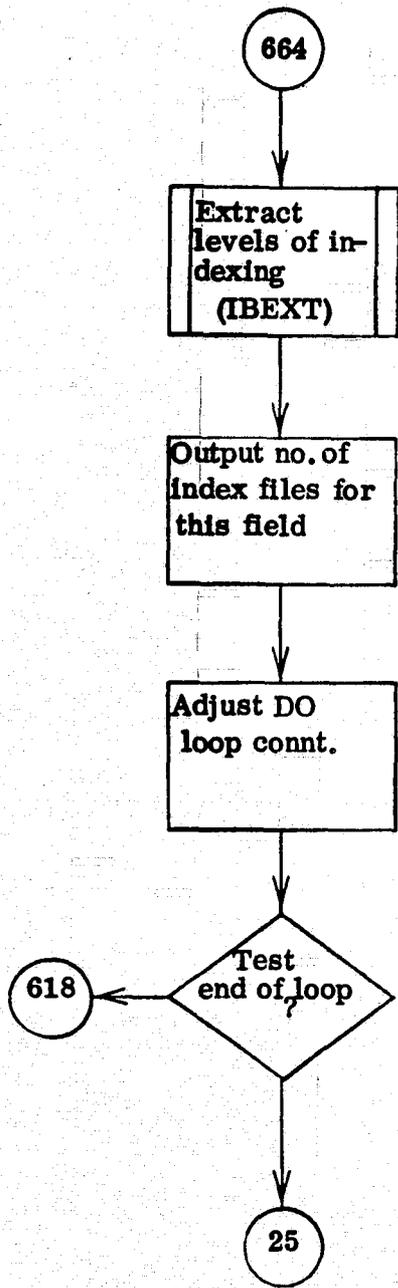
EXIT

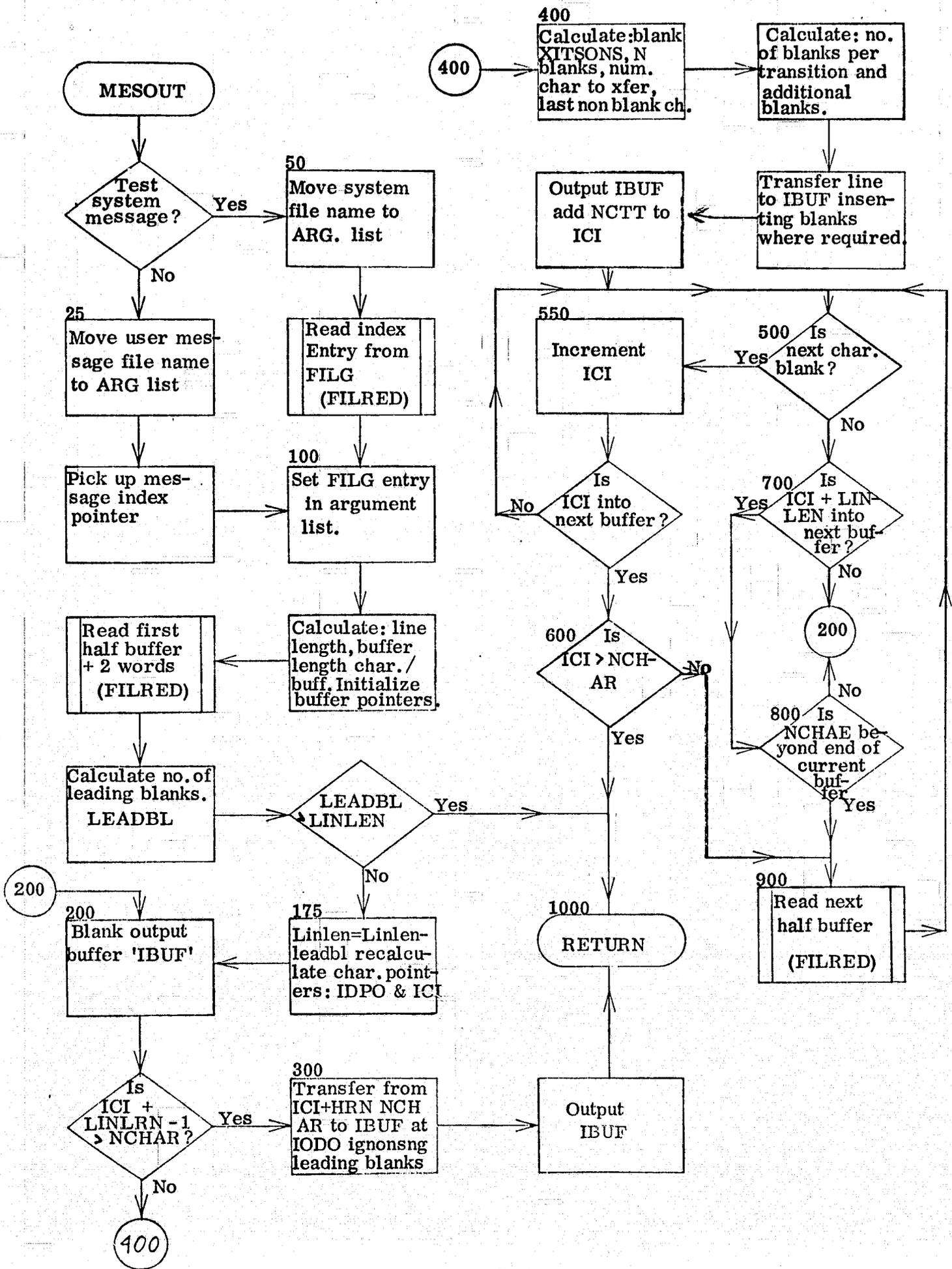


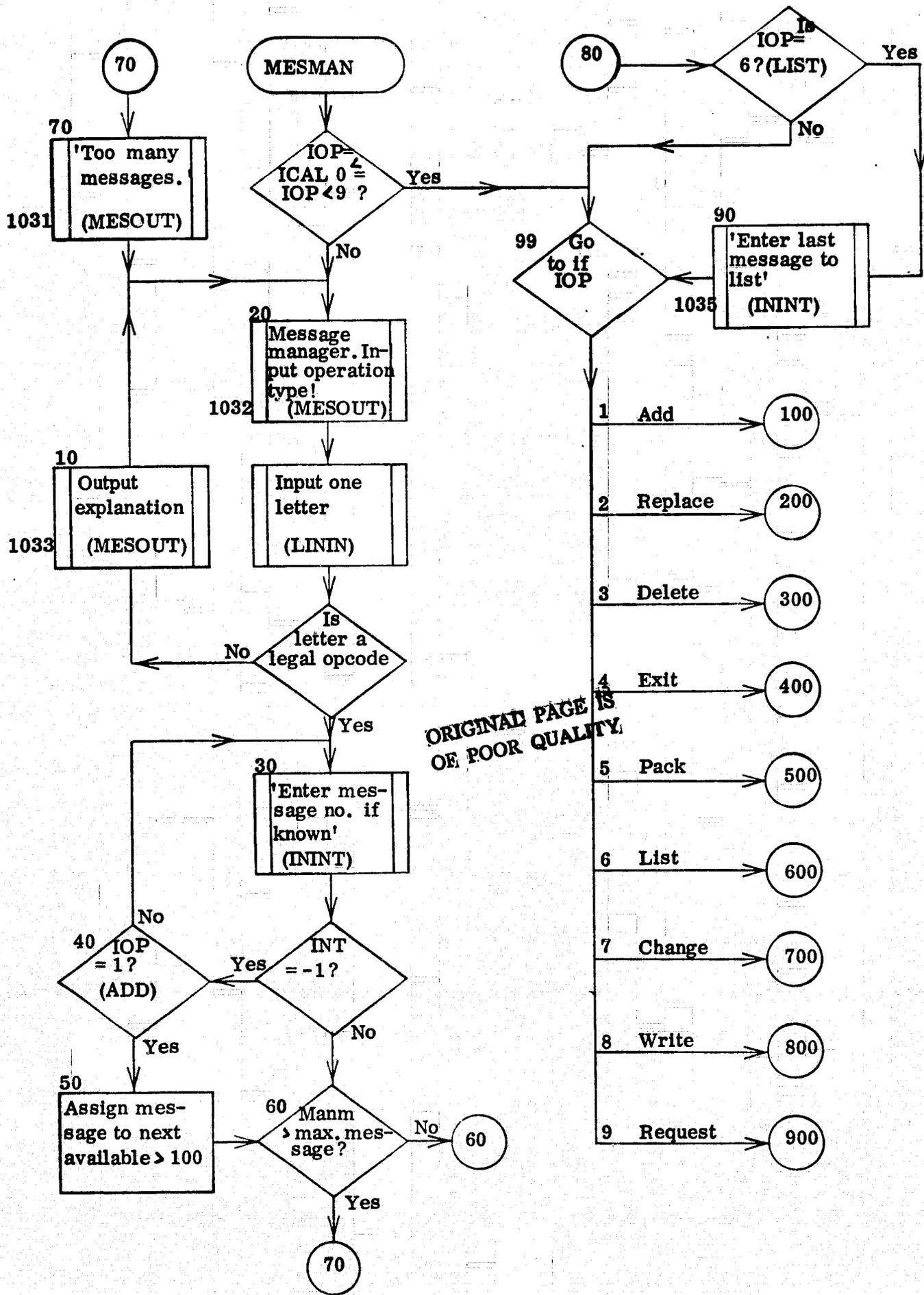
ORIGINAL PAGE IS OF POOR QUALITY



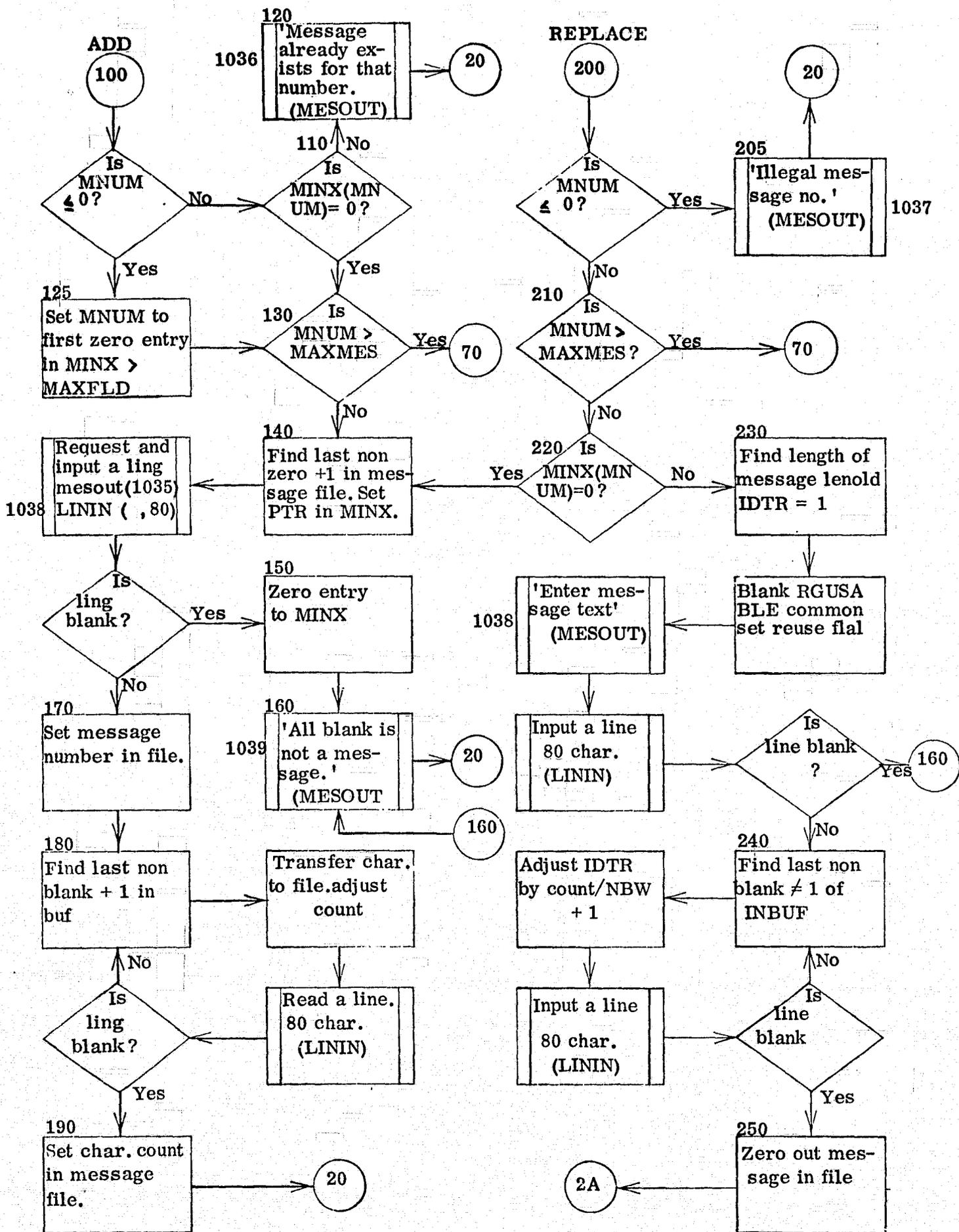
ORIGINAL PAGE IS  
OF POOR QUALITY

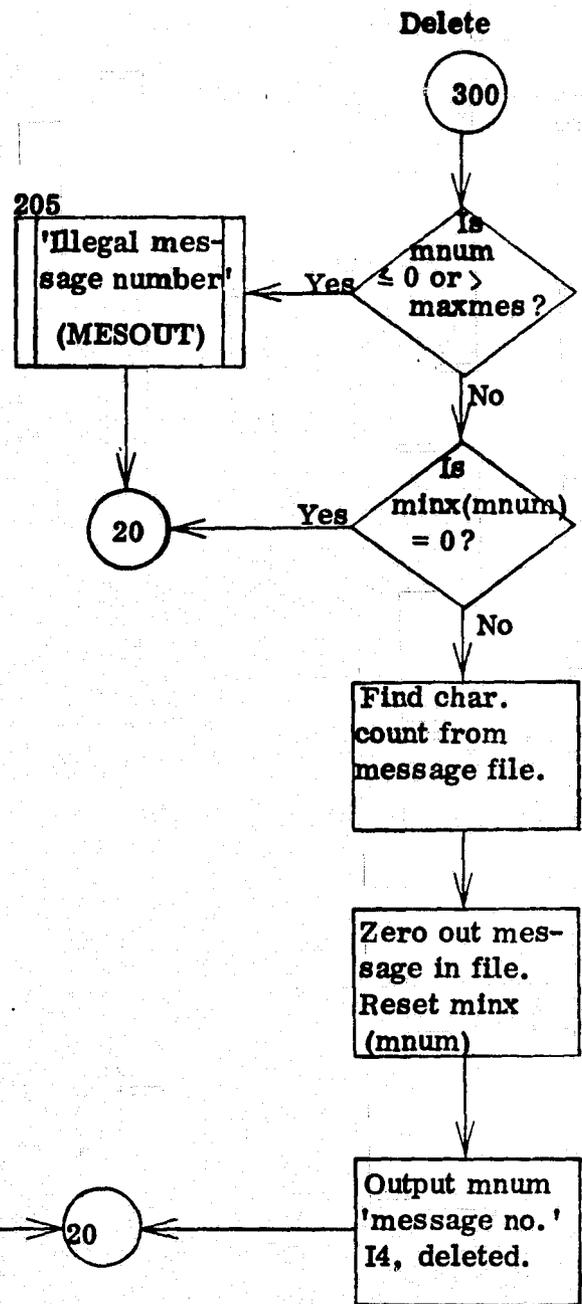
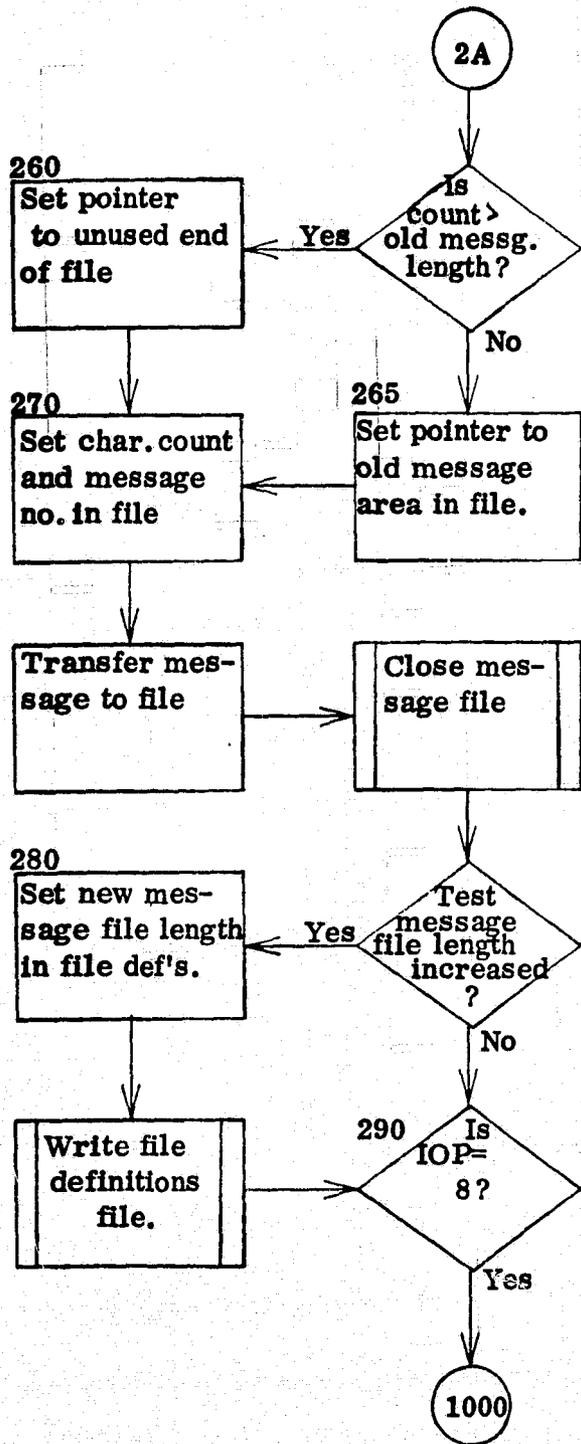


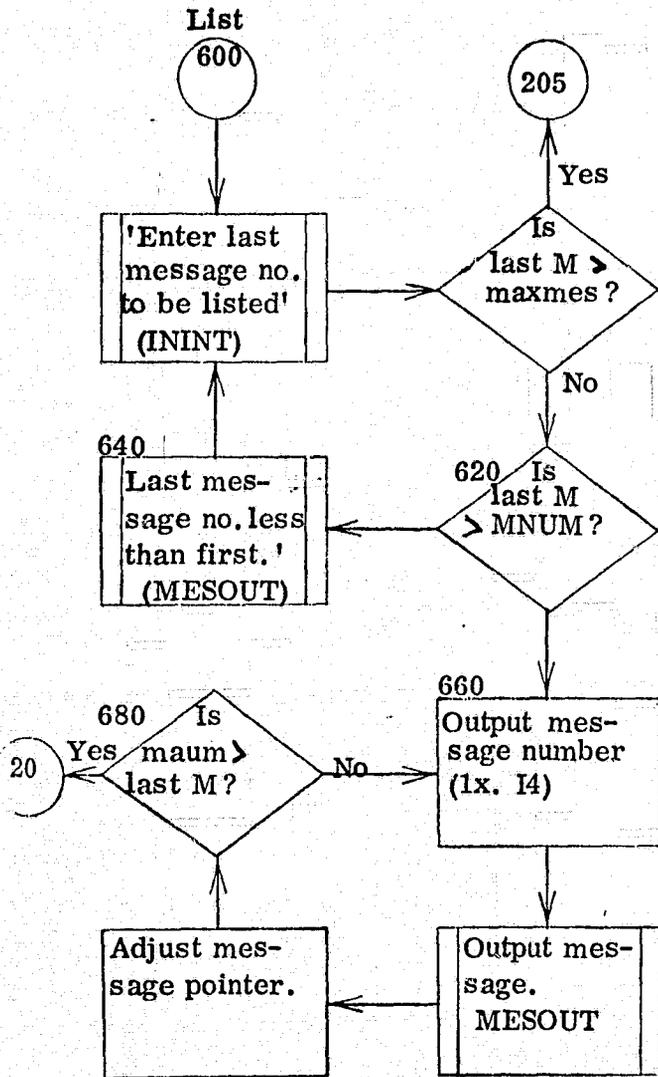
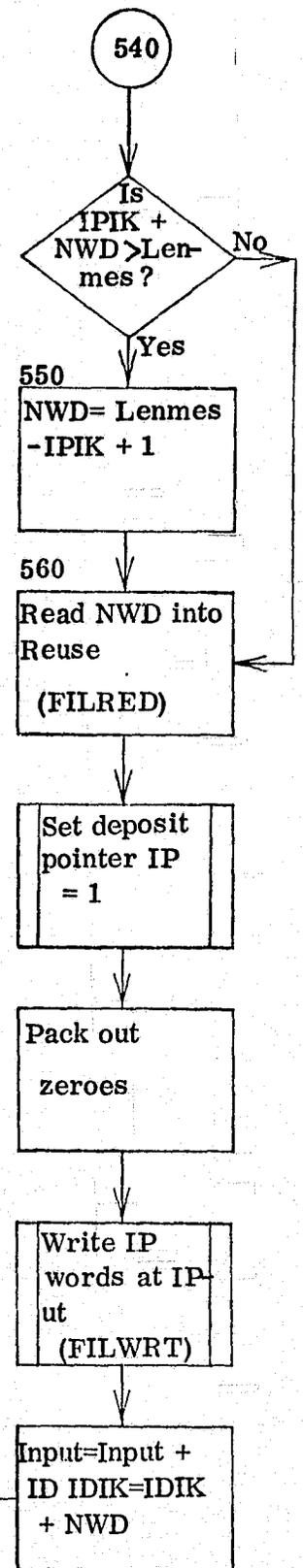
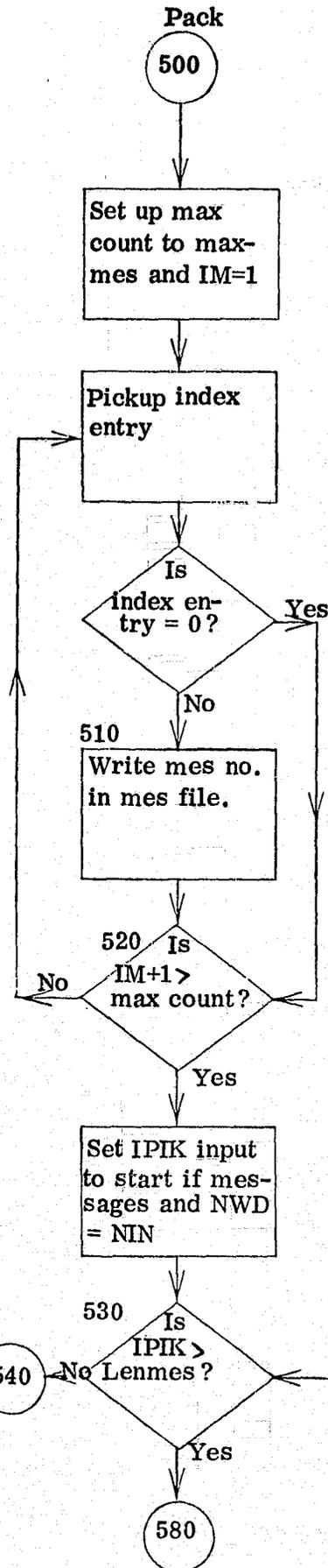
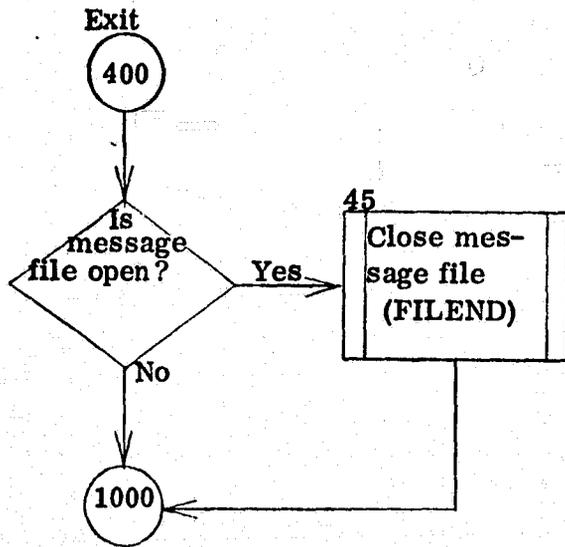


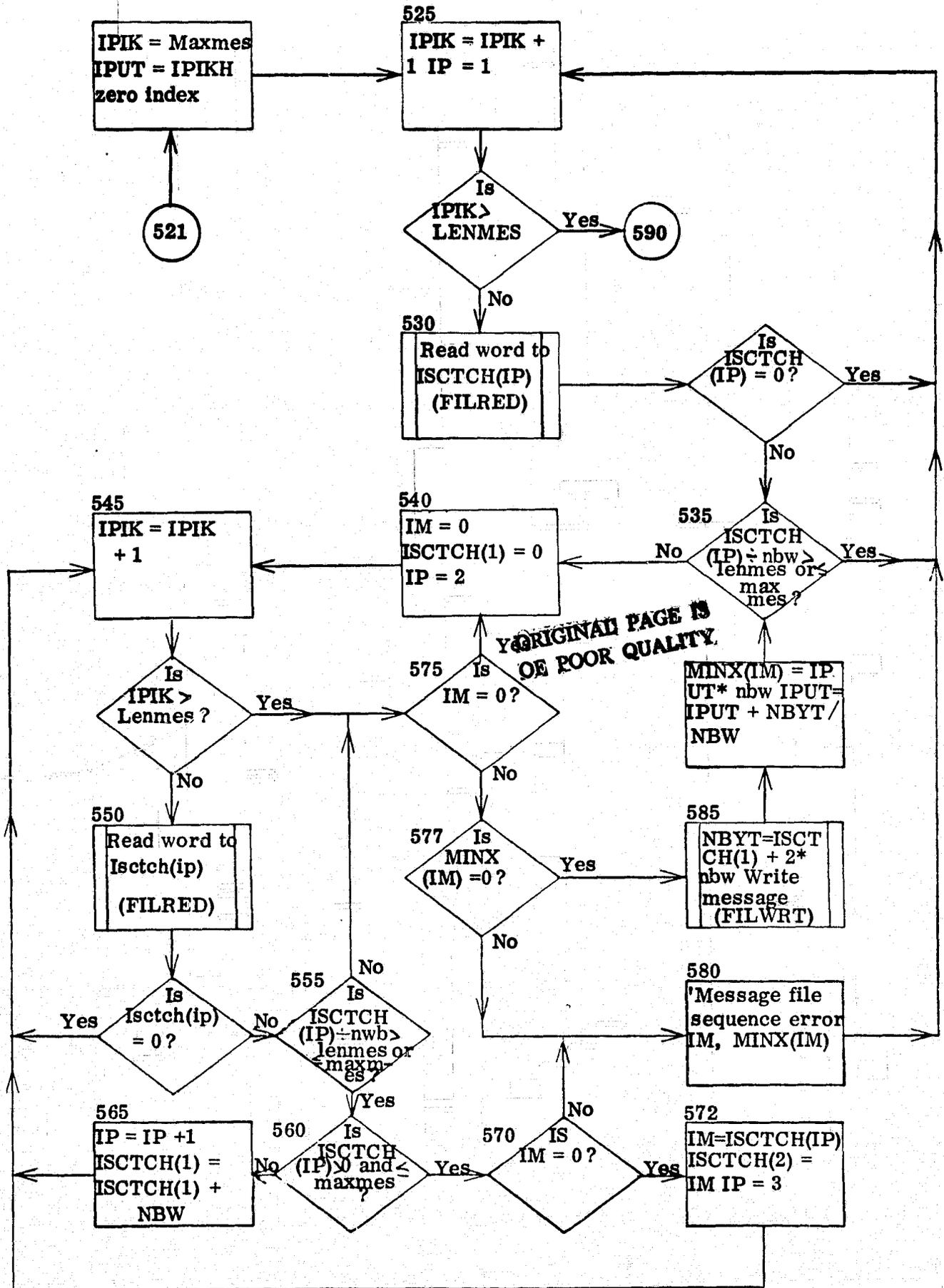


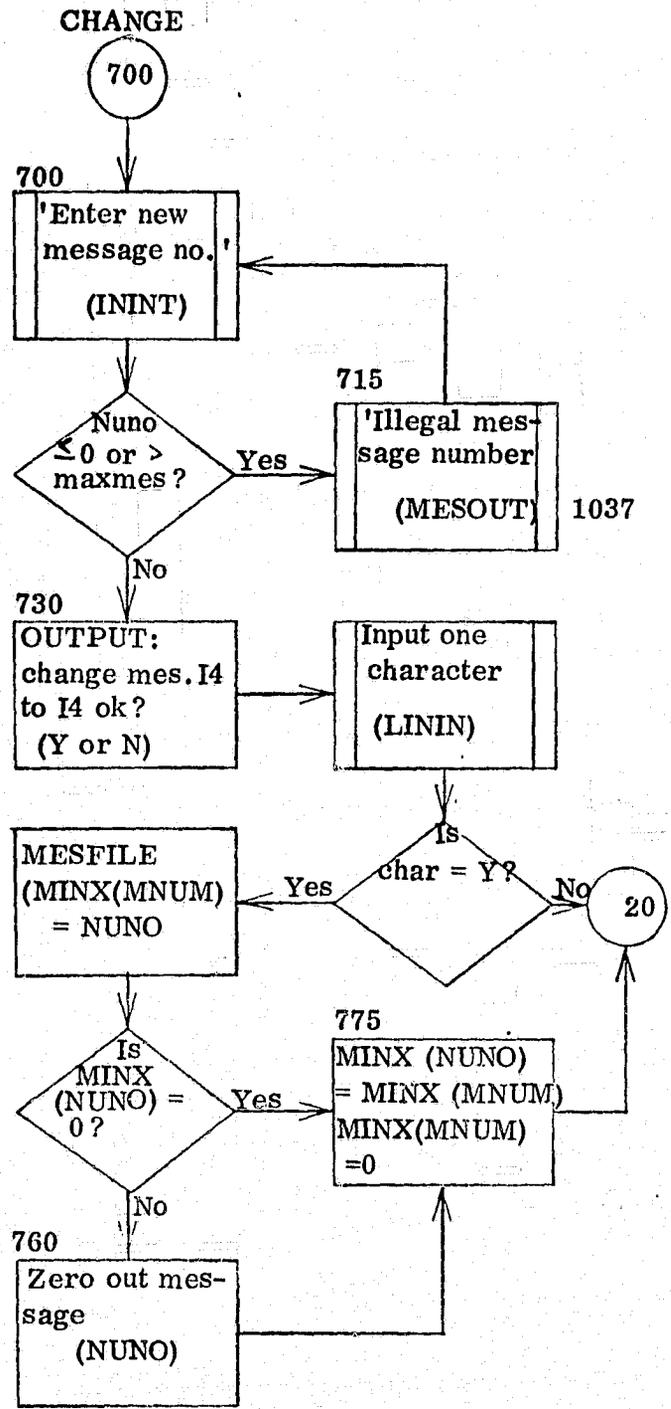
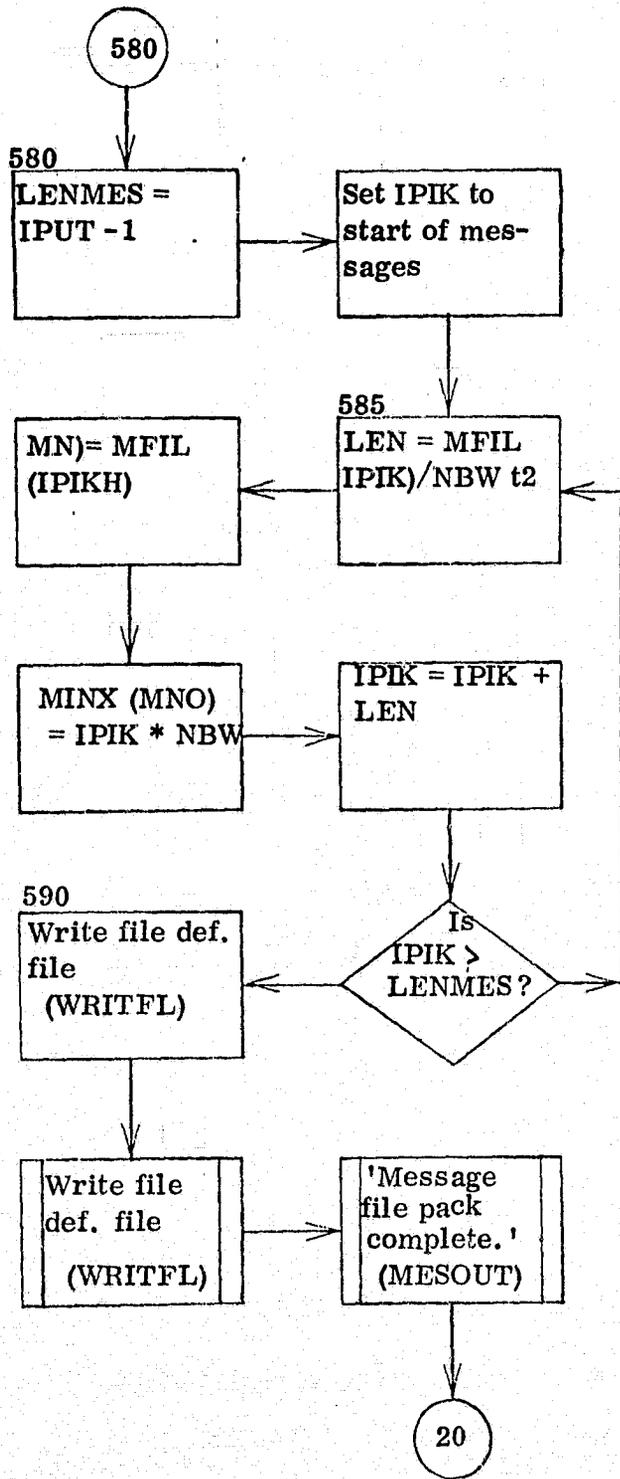
ORIGINAL PAGE IS OF POOR QUALITY

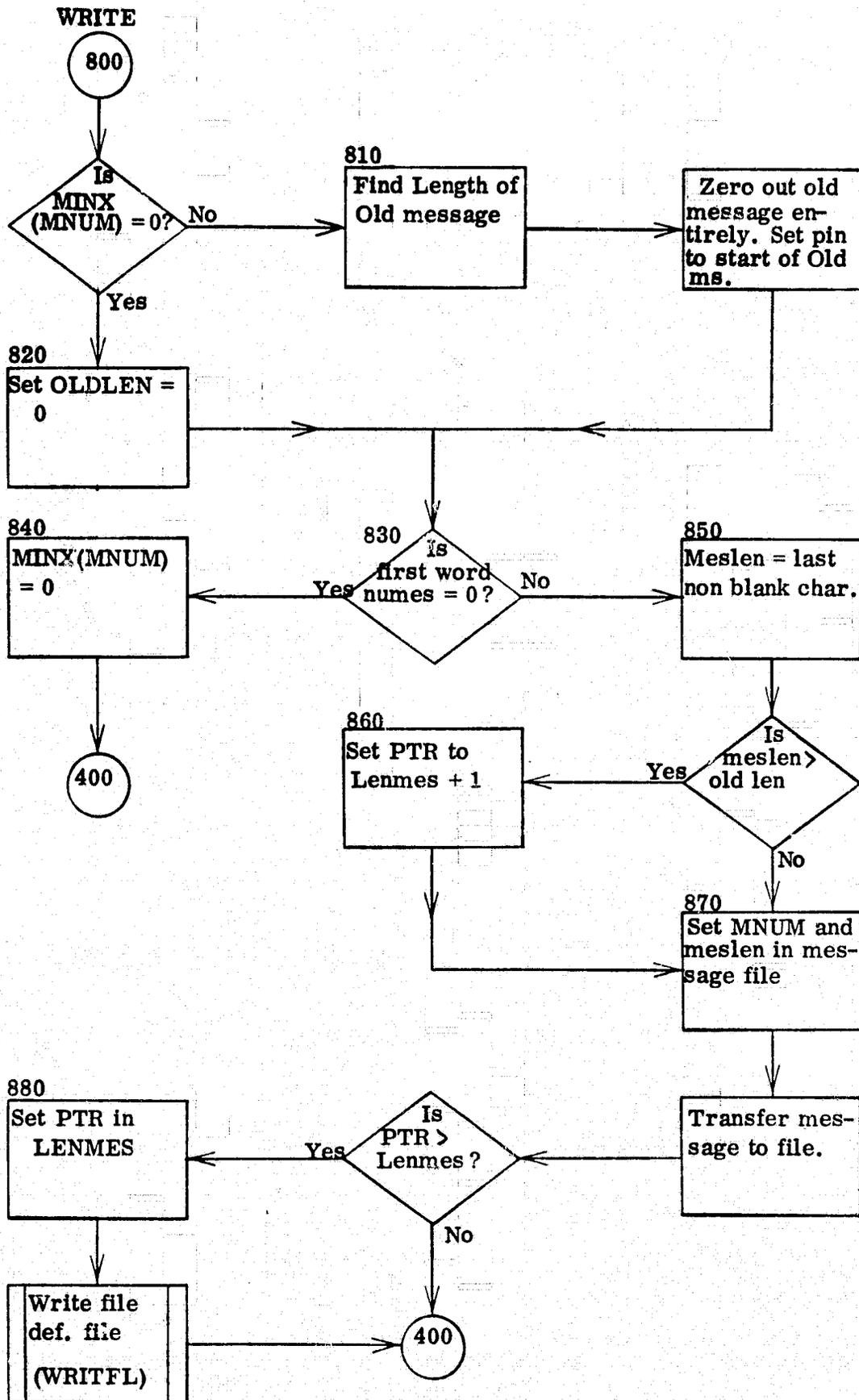


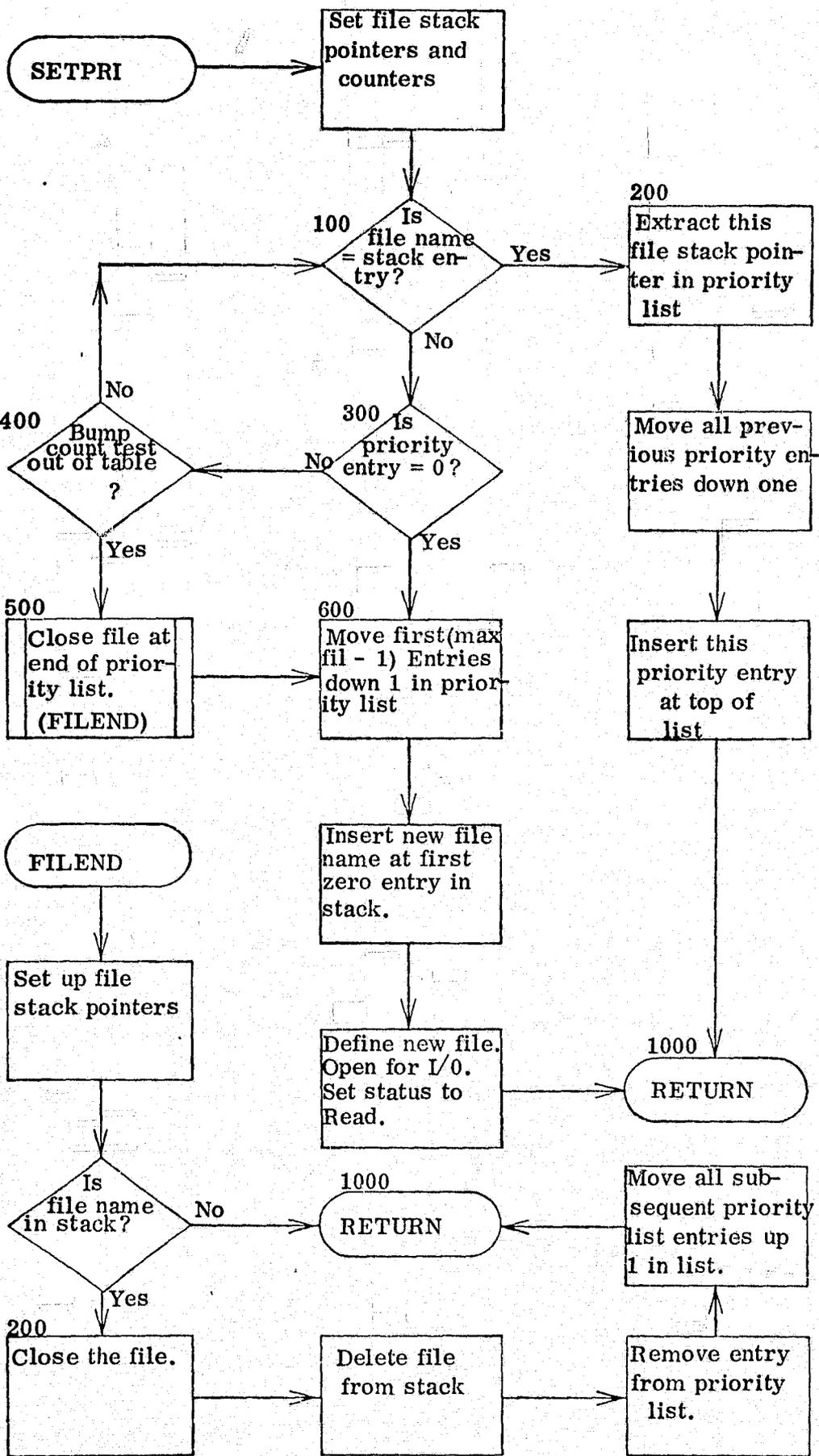


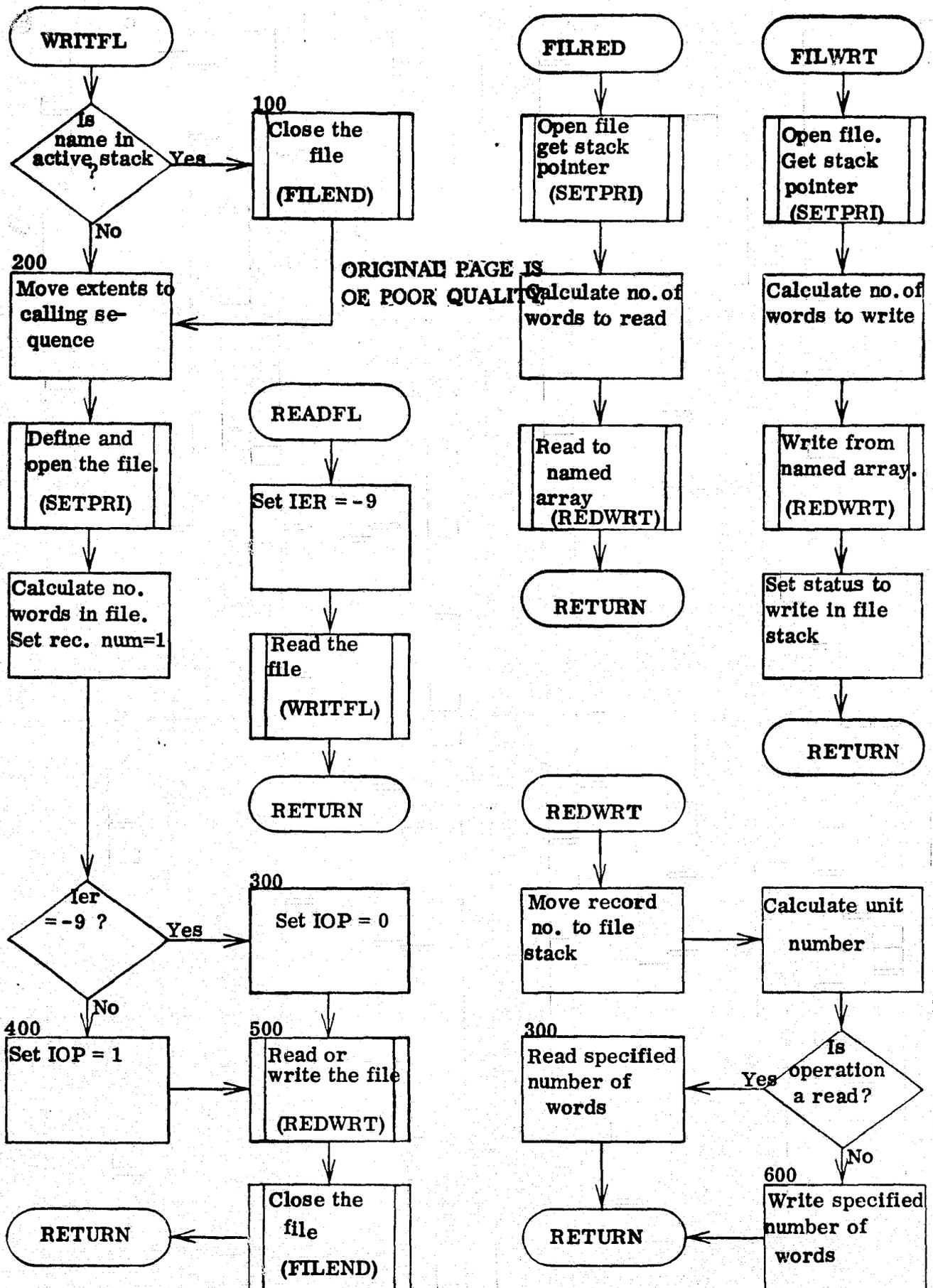


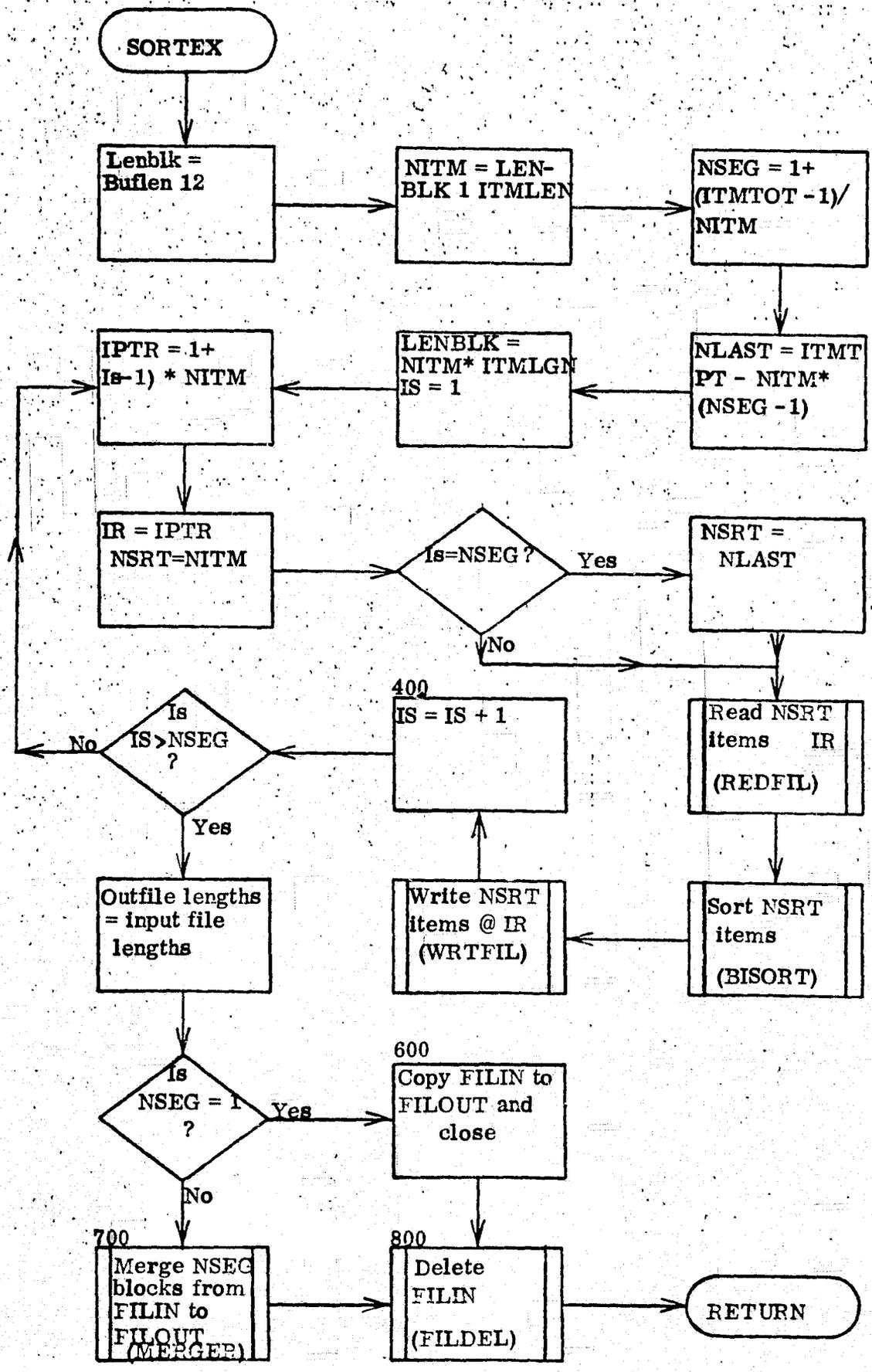


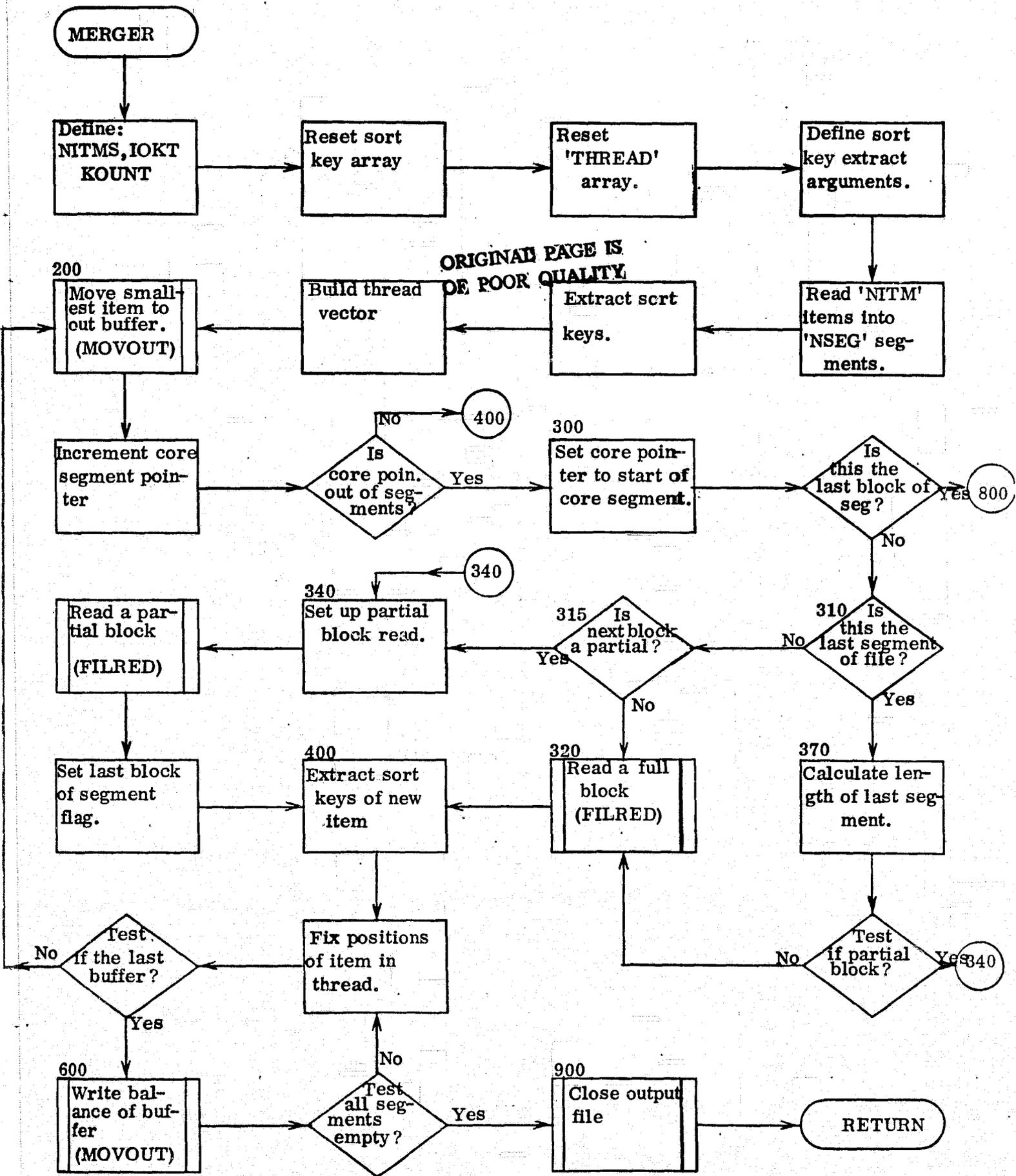


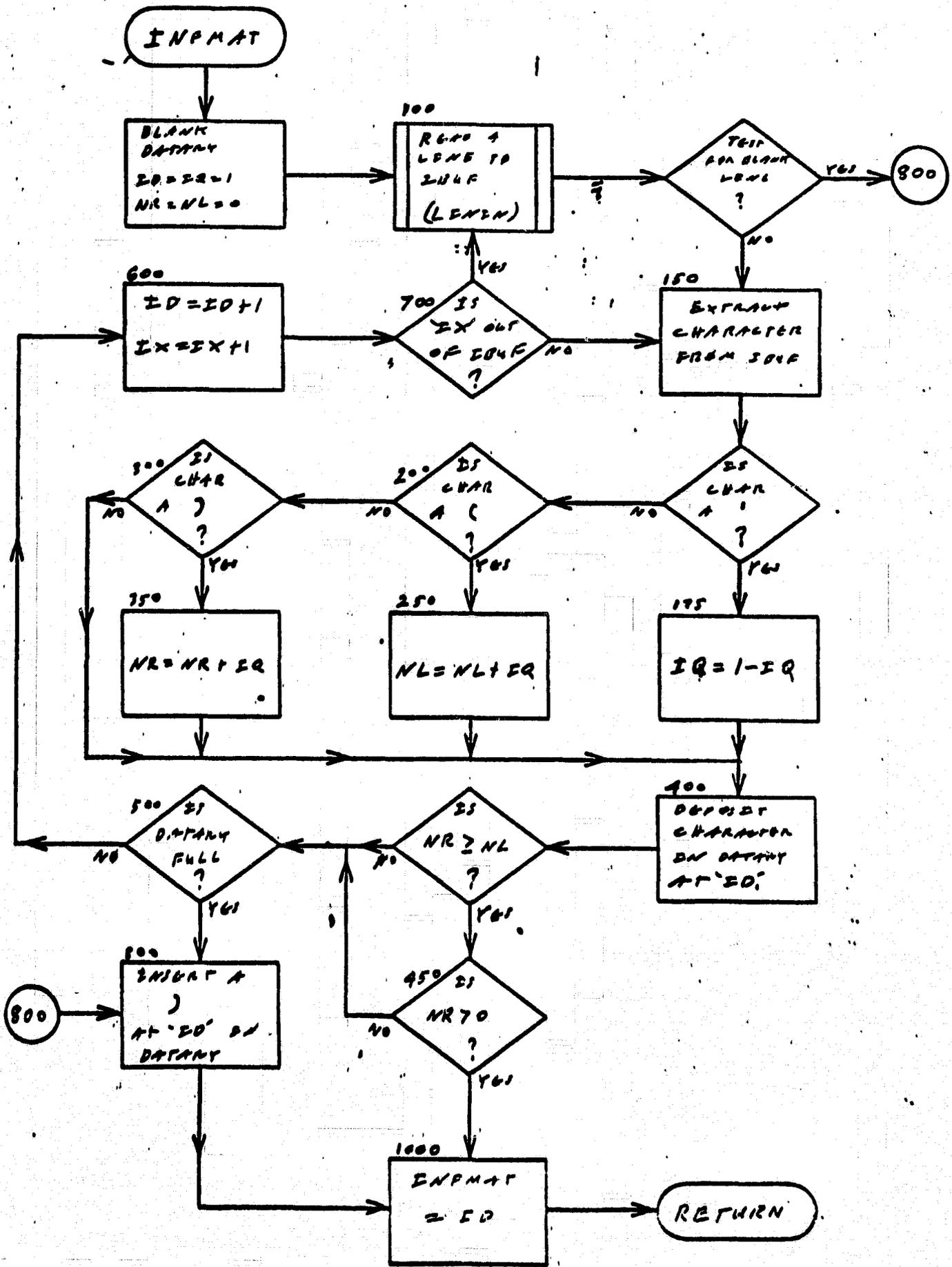




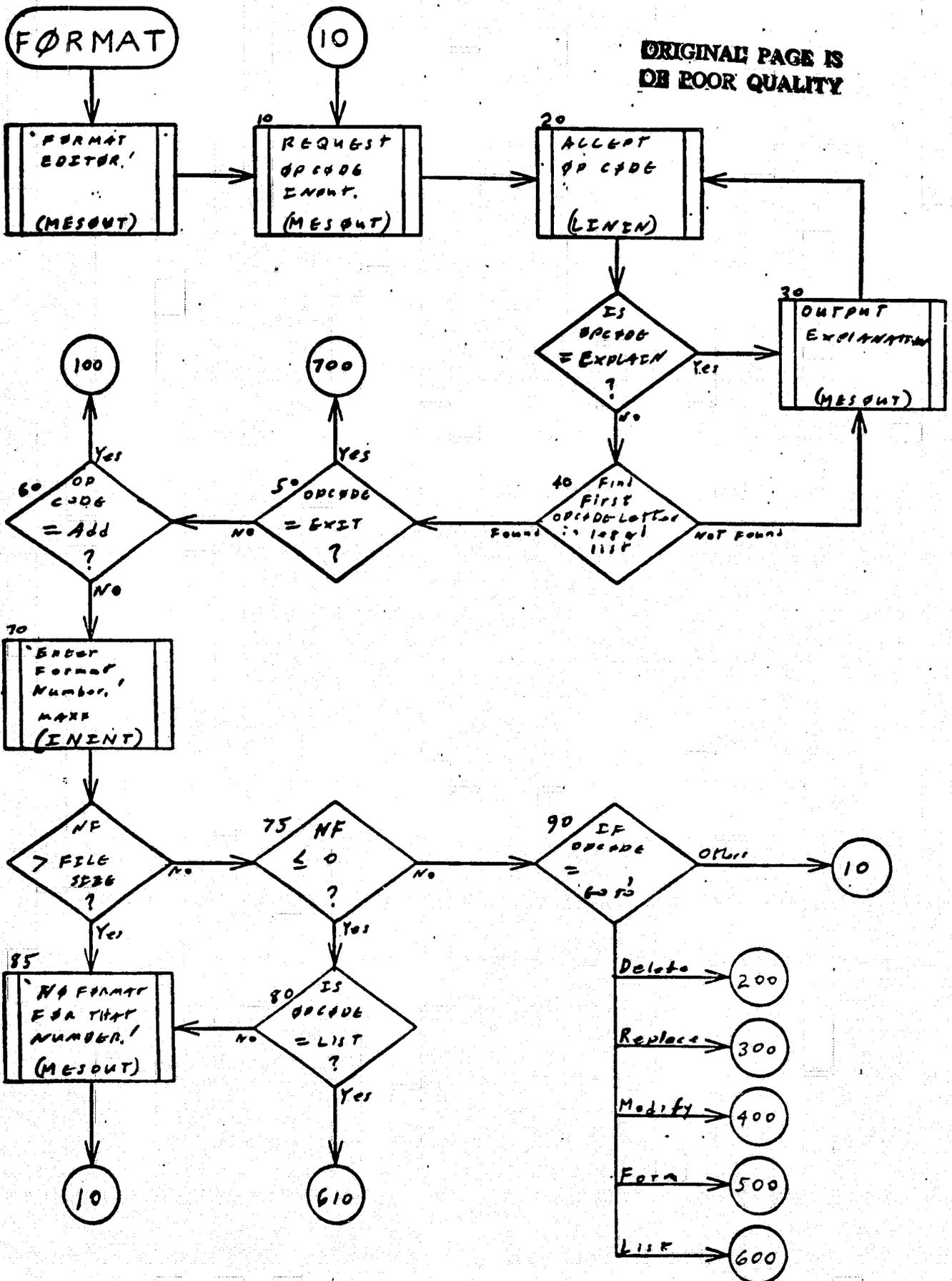


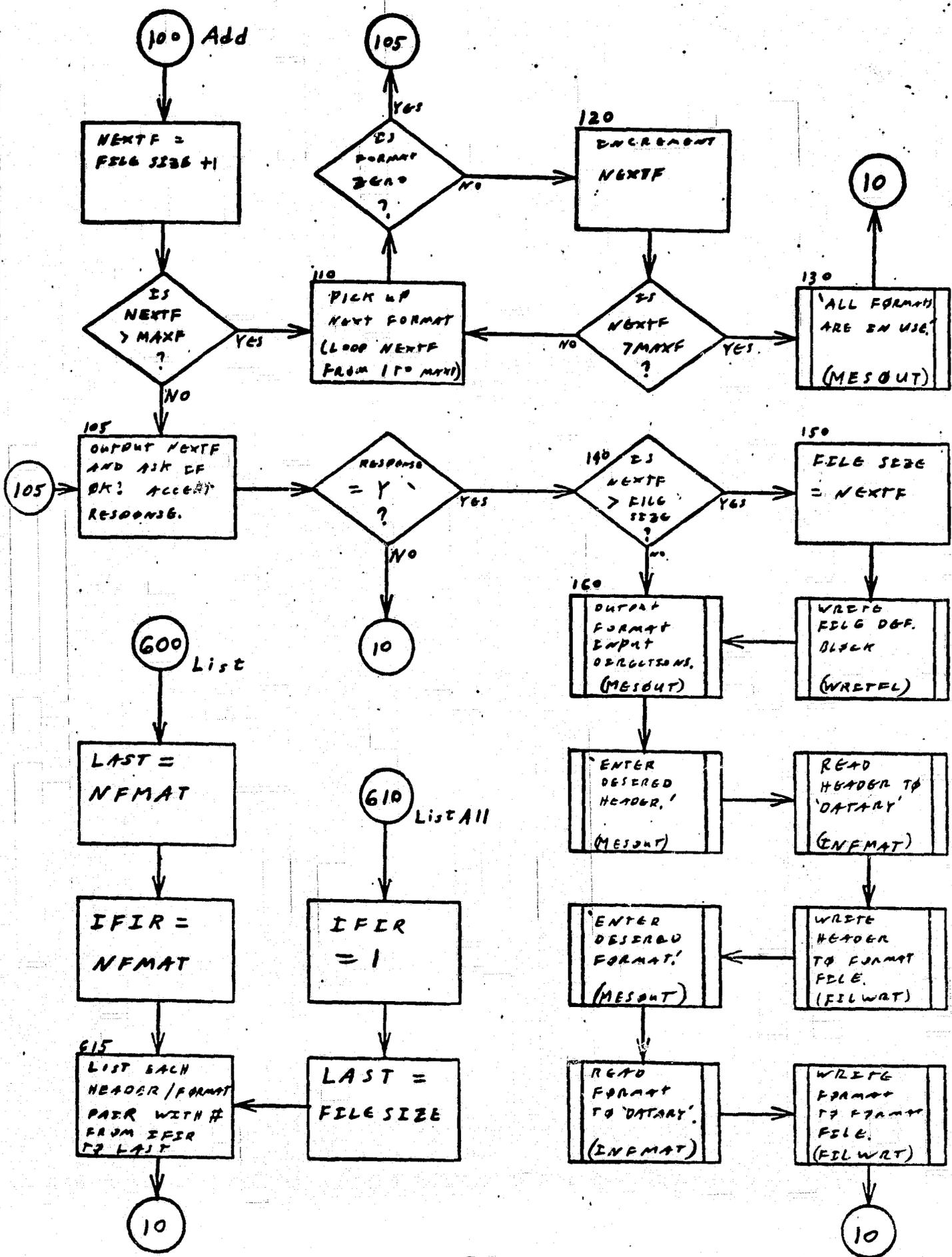


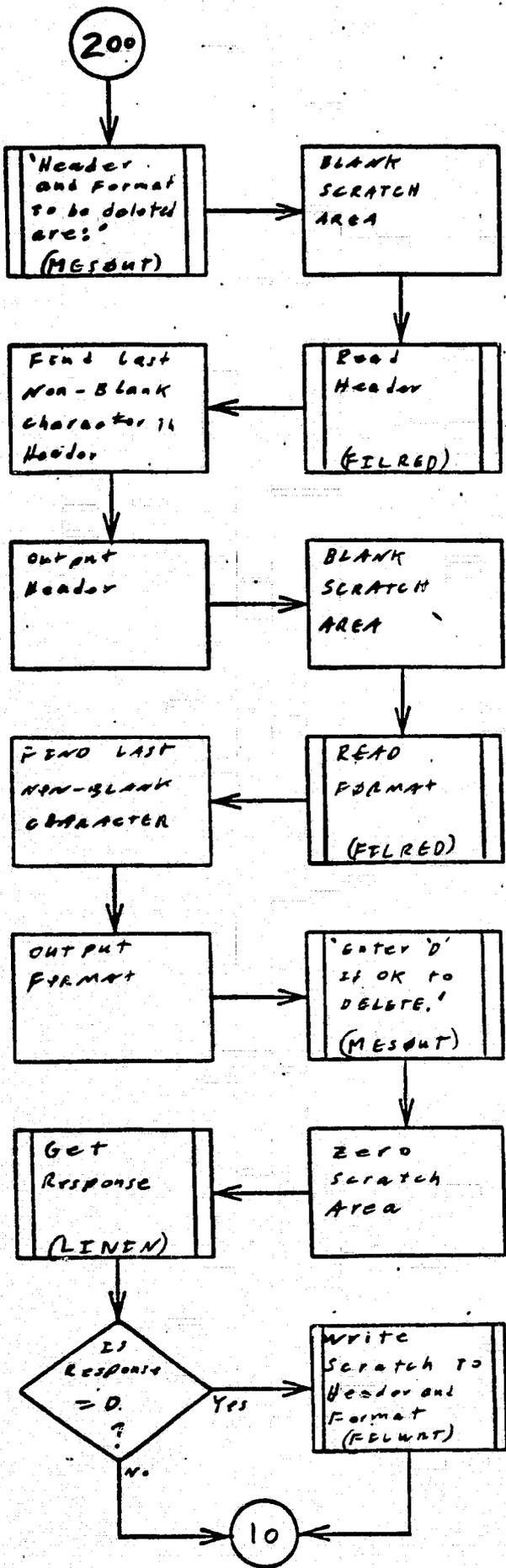




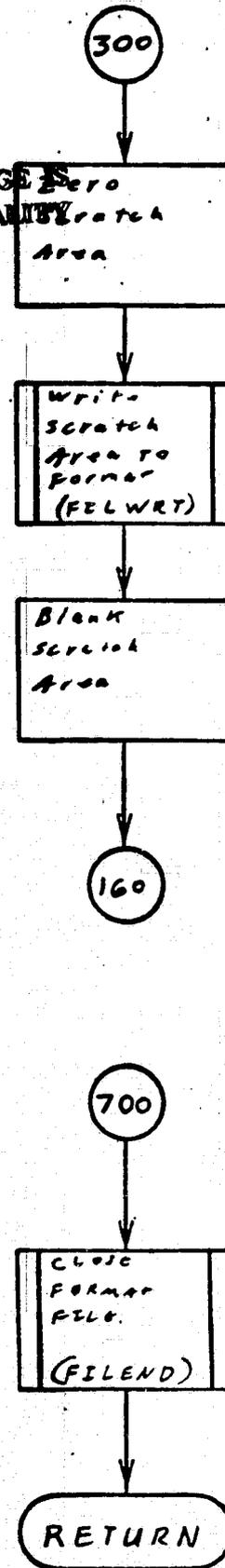
ORIGINAL PAGE IS OF POOR QUALITY





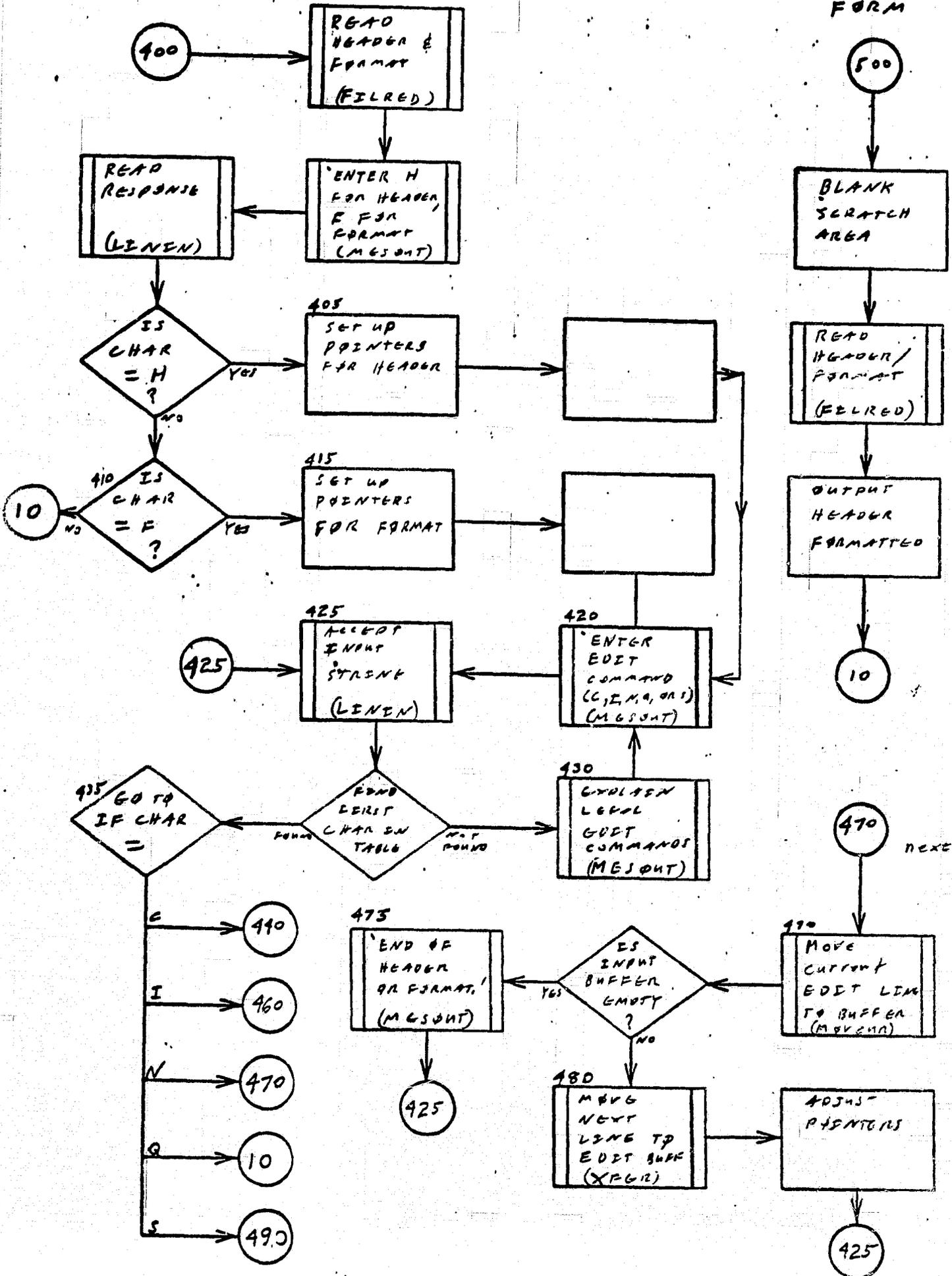


ORIGINAL PAGE IS OF POOR QUALITY

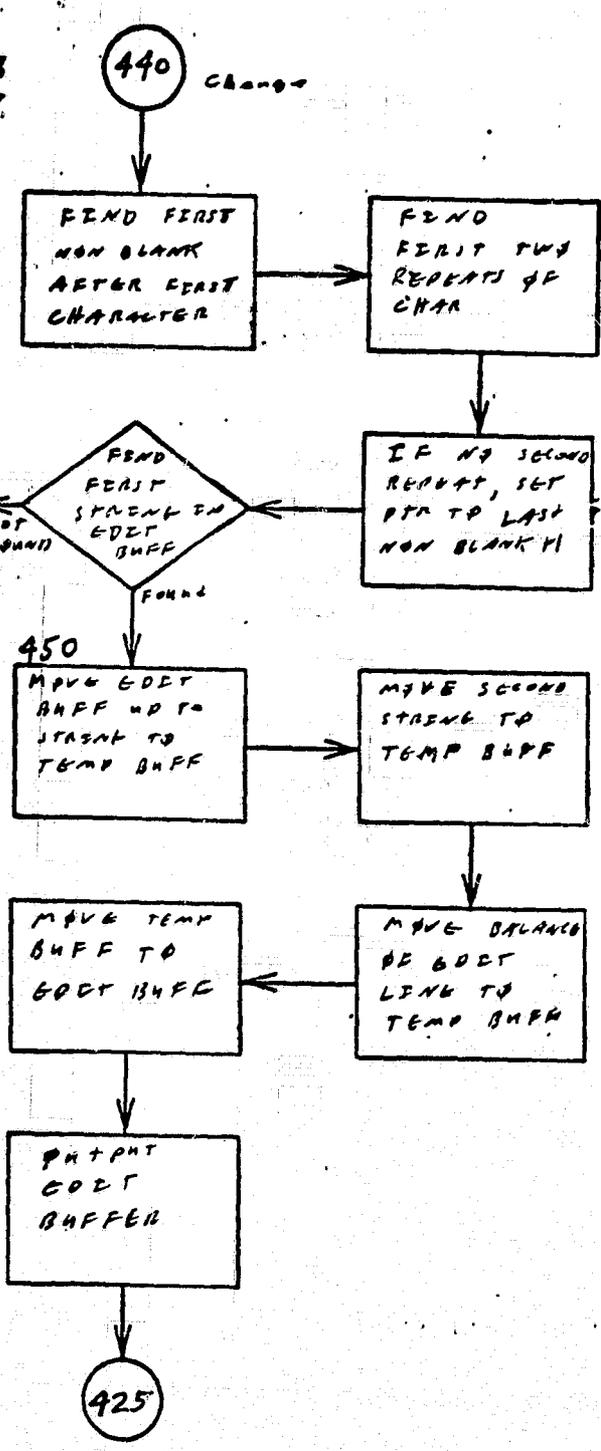
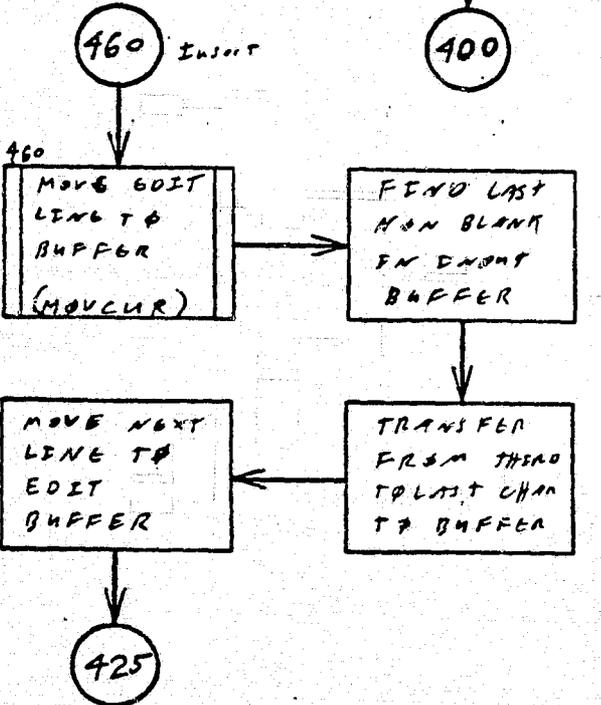
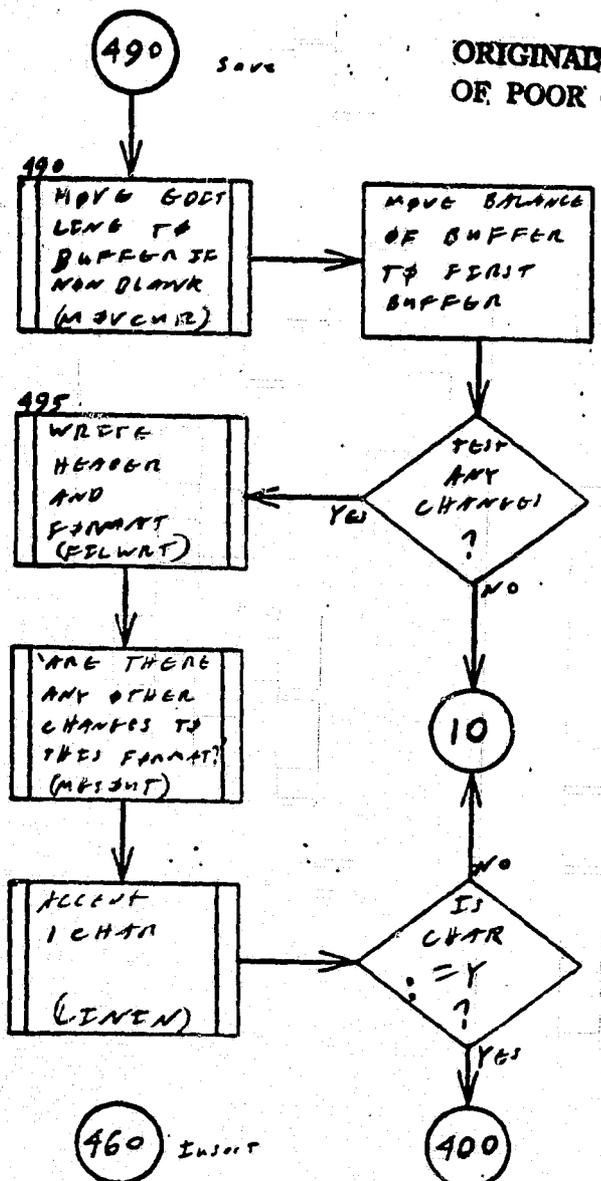


MODEEY

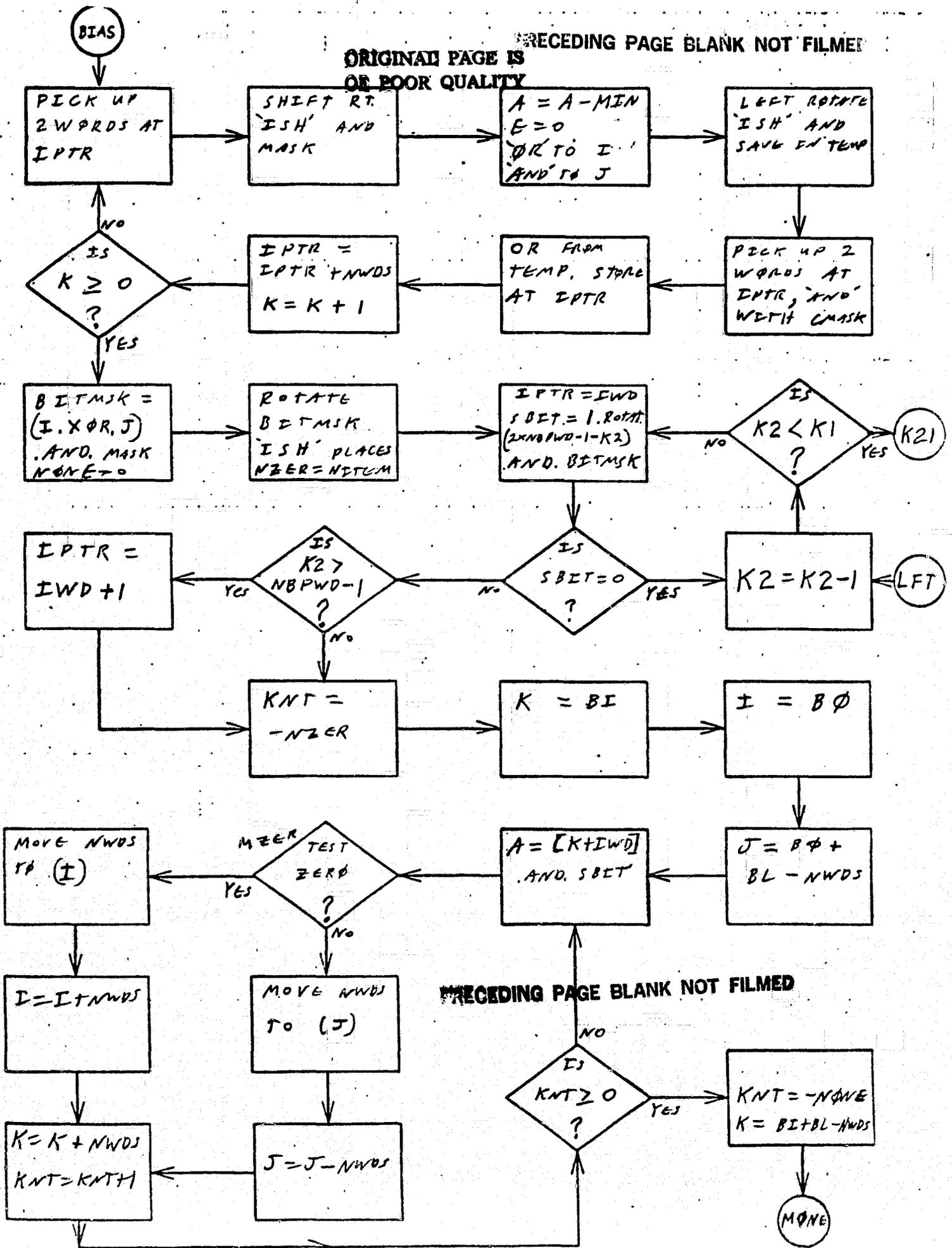
FORM



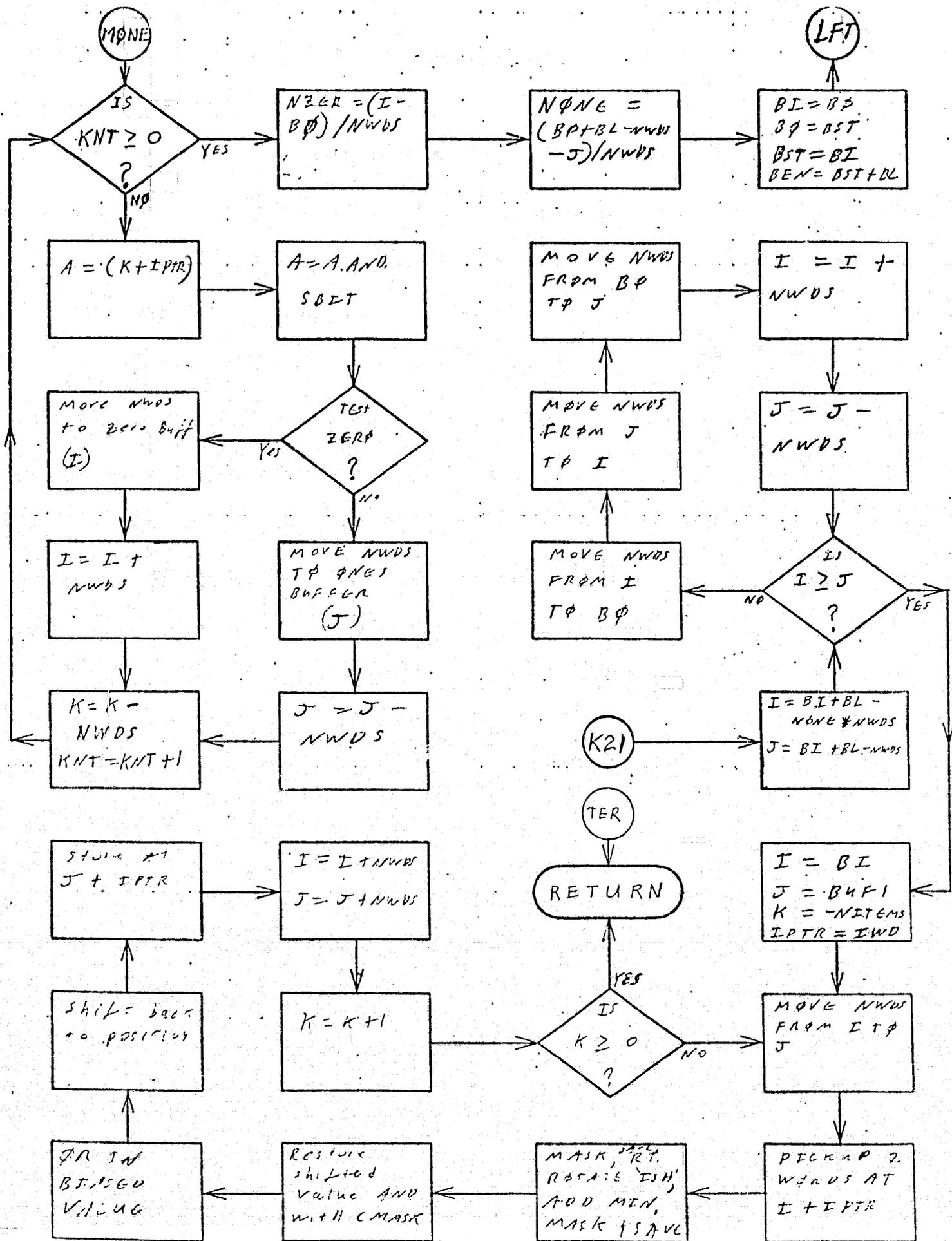
ORIGINAL PAGE IS  
OF POOR QUALITY



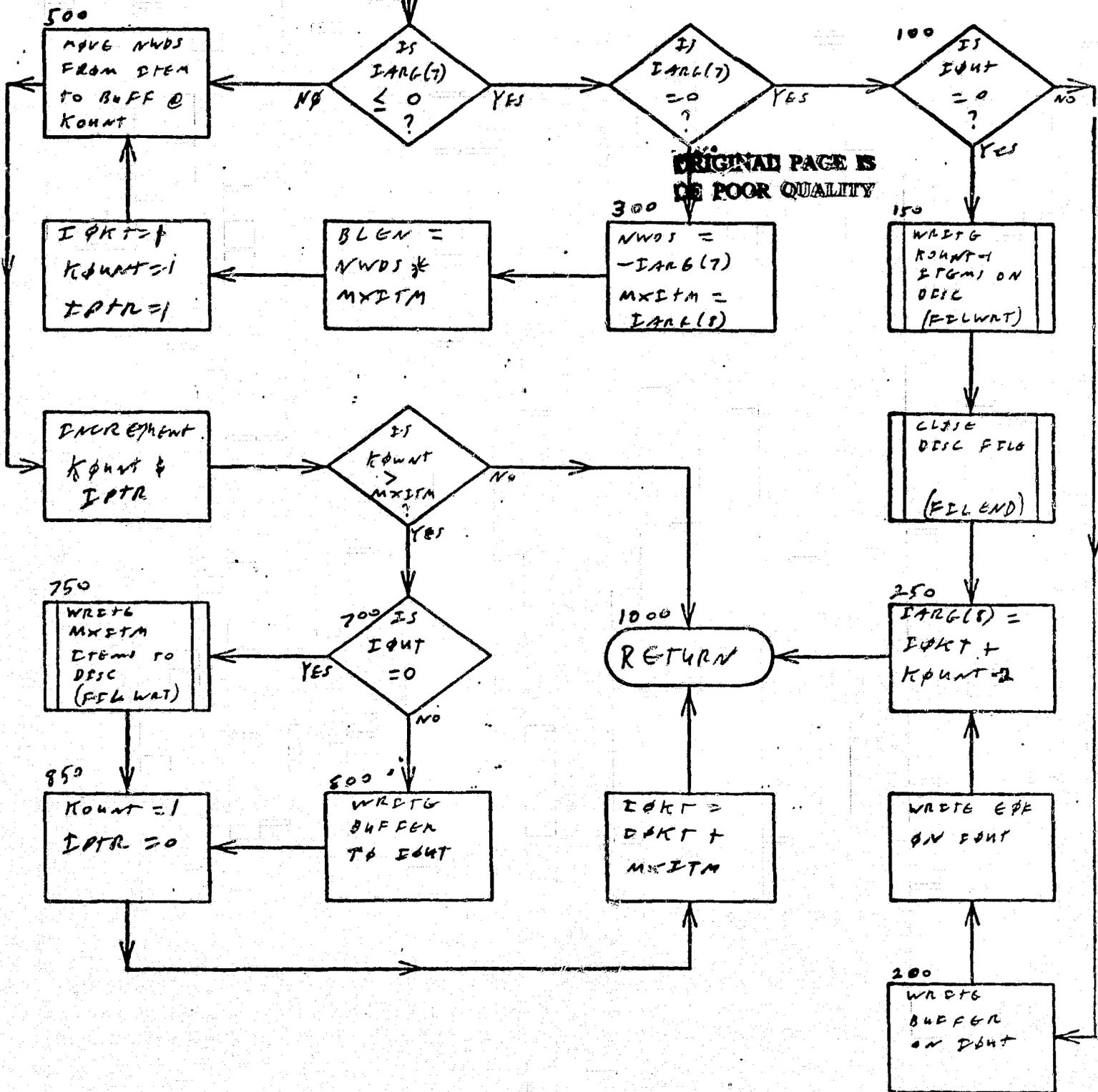
ORIGINAL PAGE IS  
OF POOR QUALITY

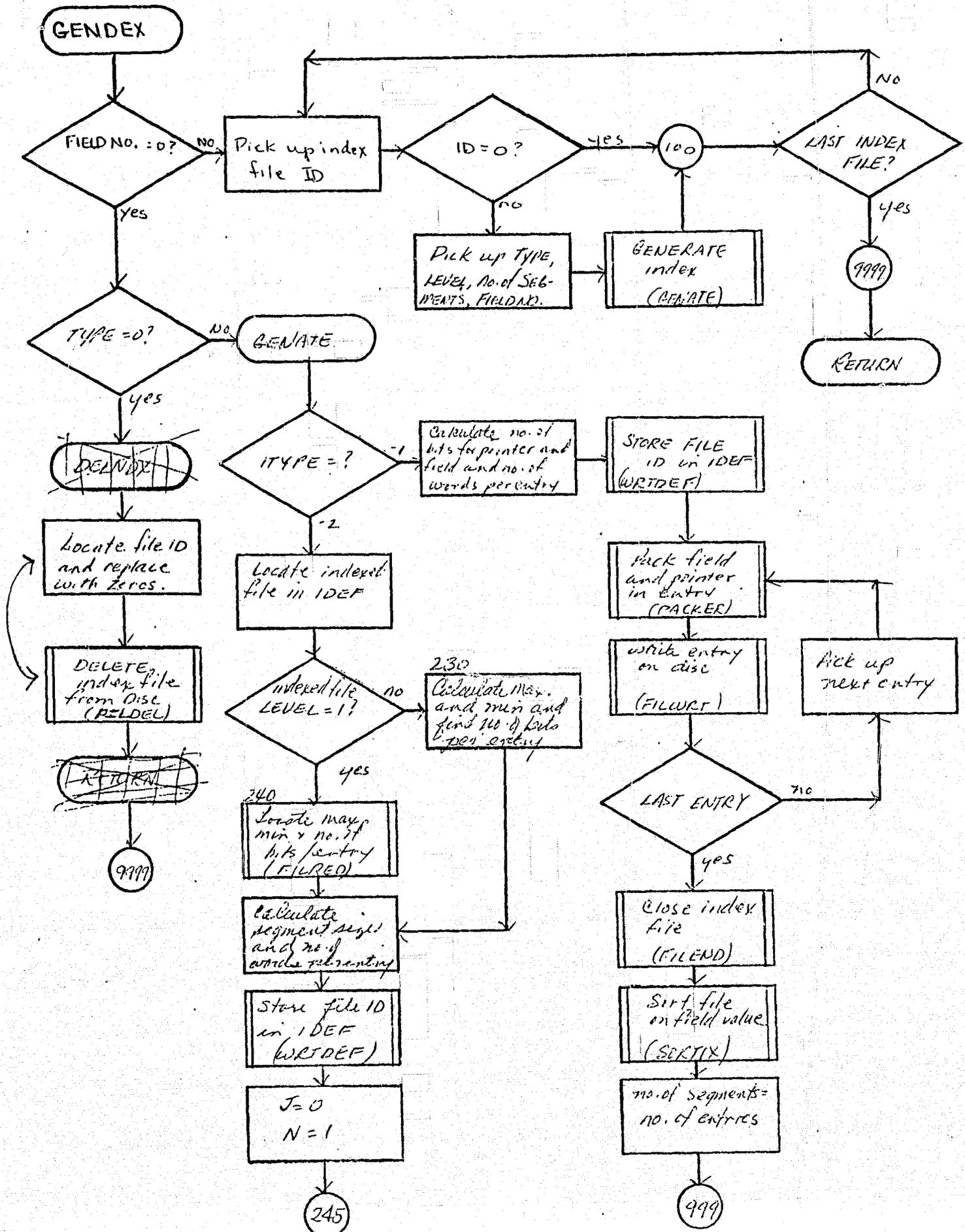


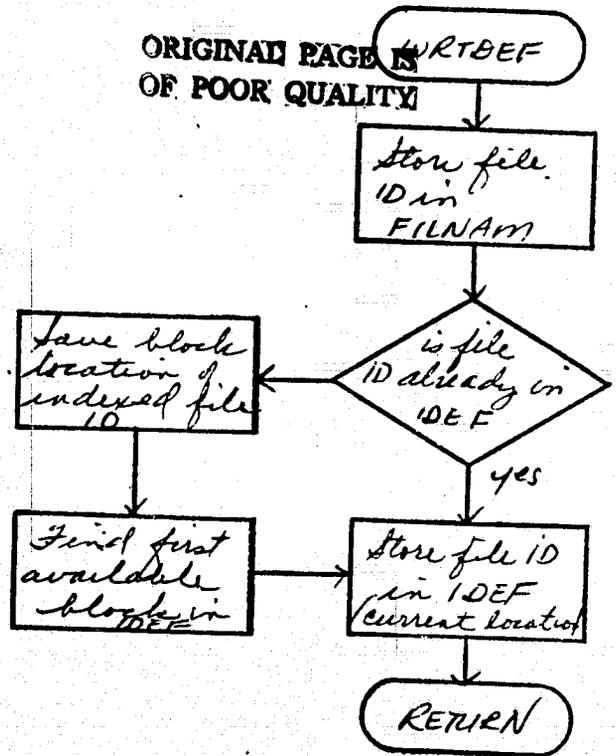
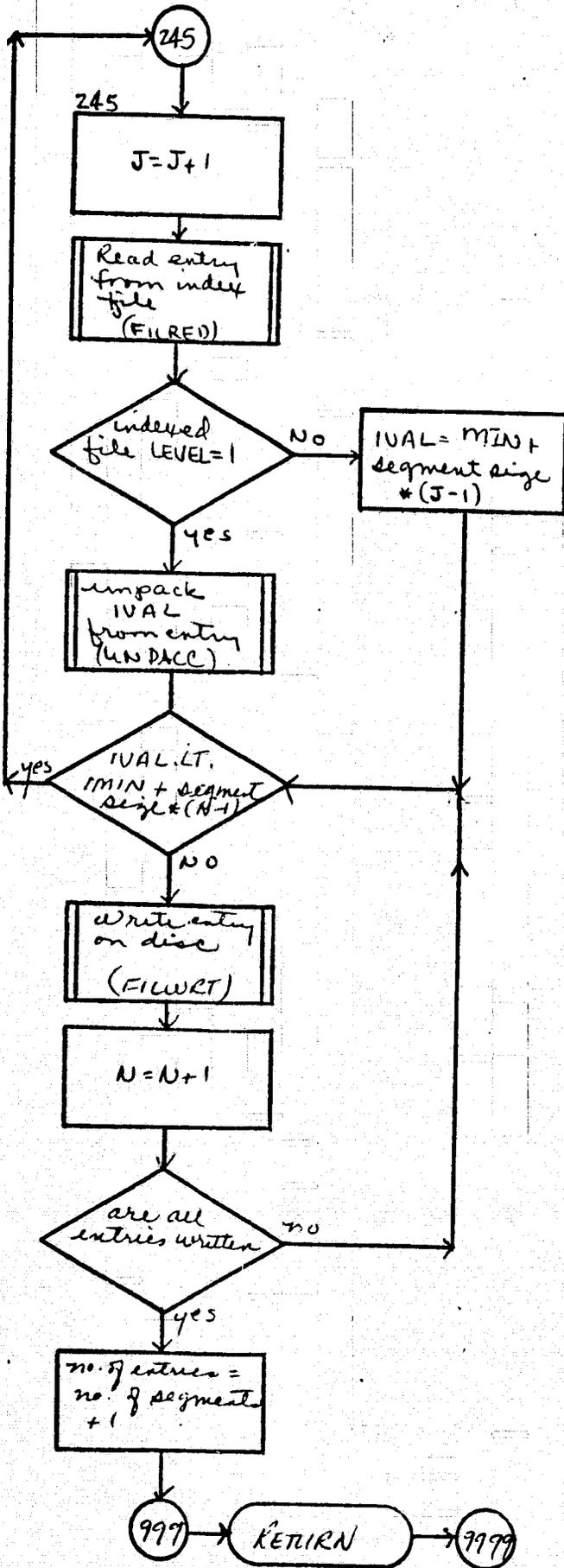
PRECEDING PAGE BLANK NOT FILMED

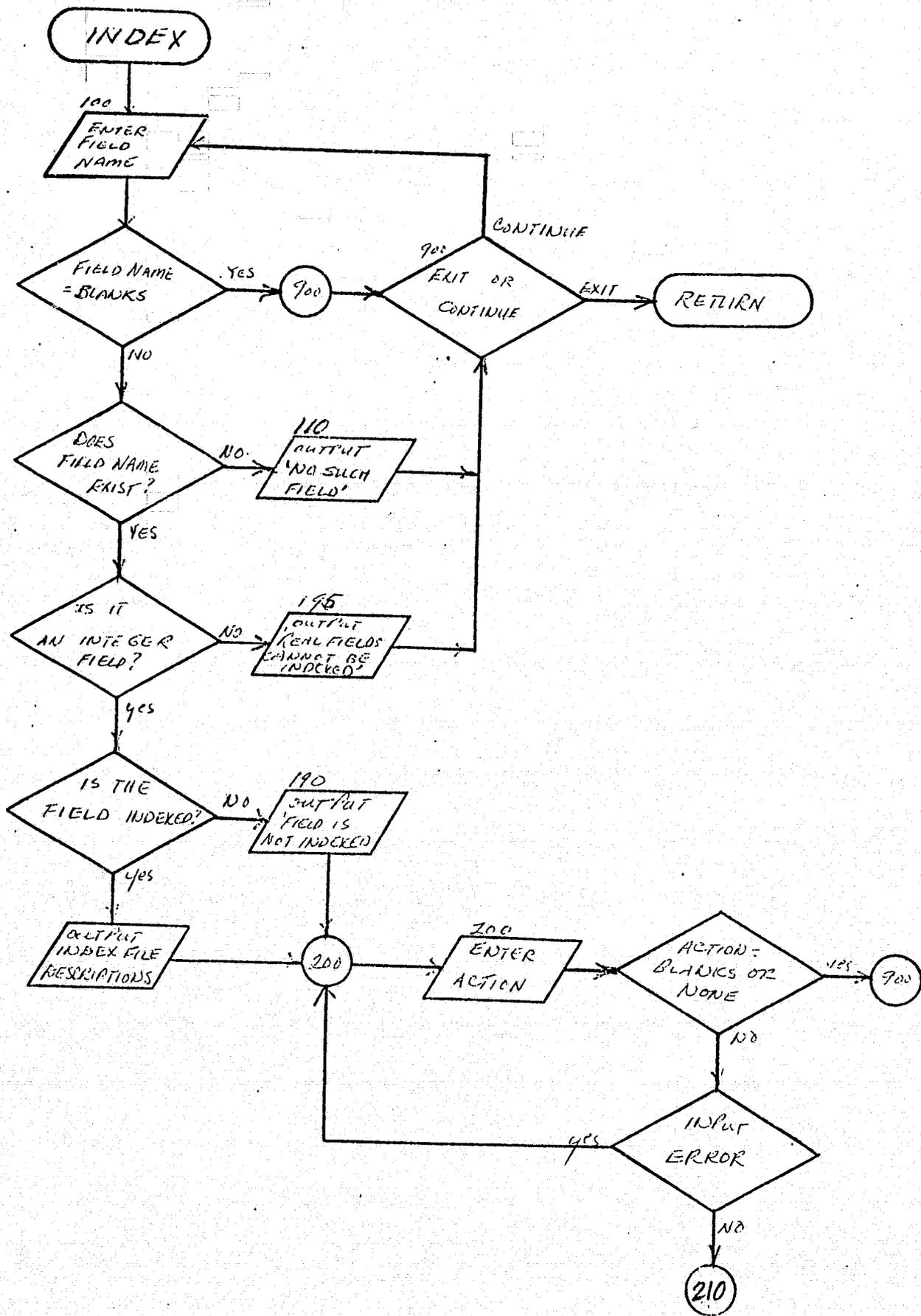


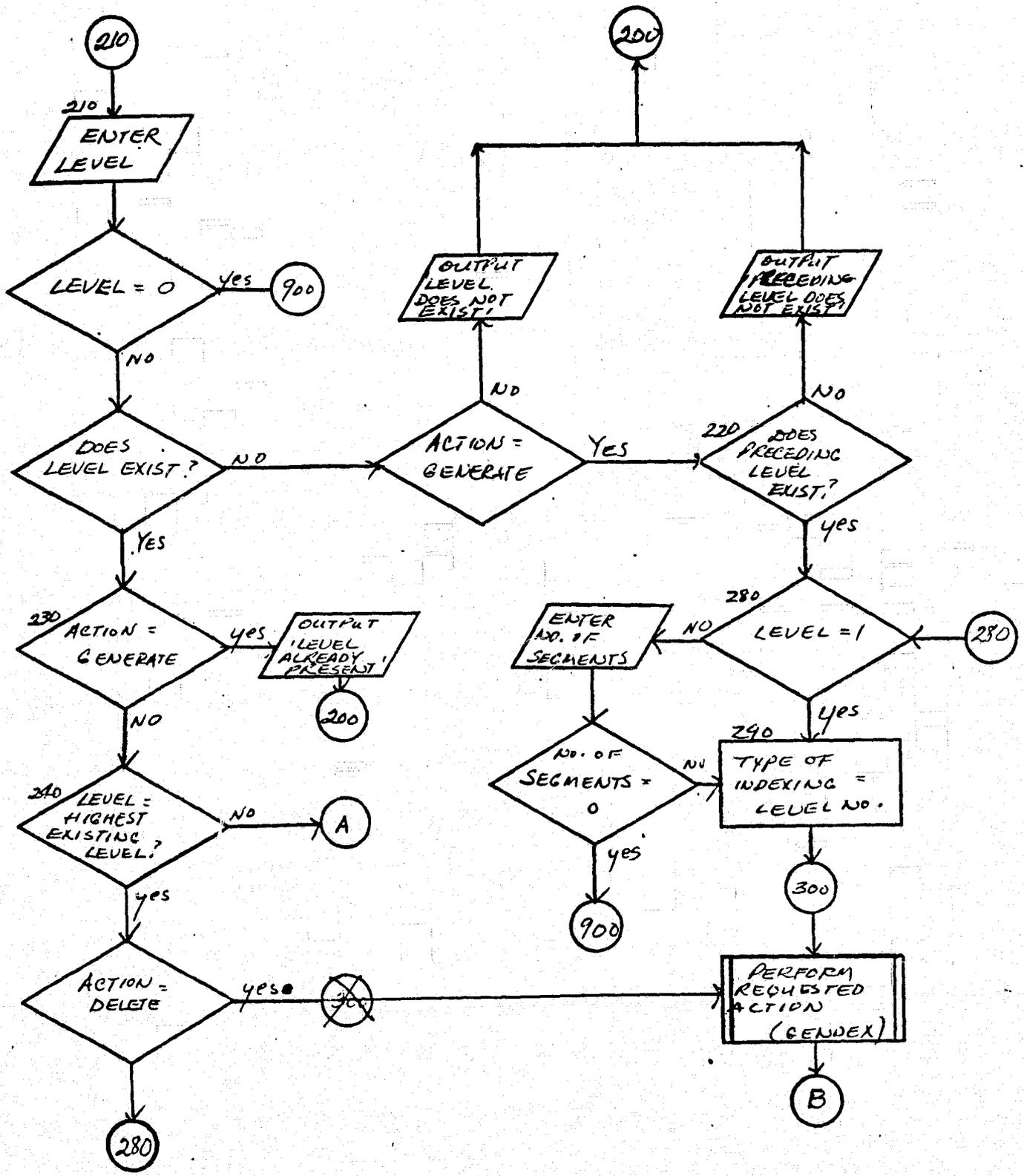
MØVØUT

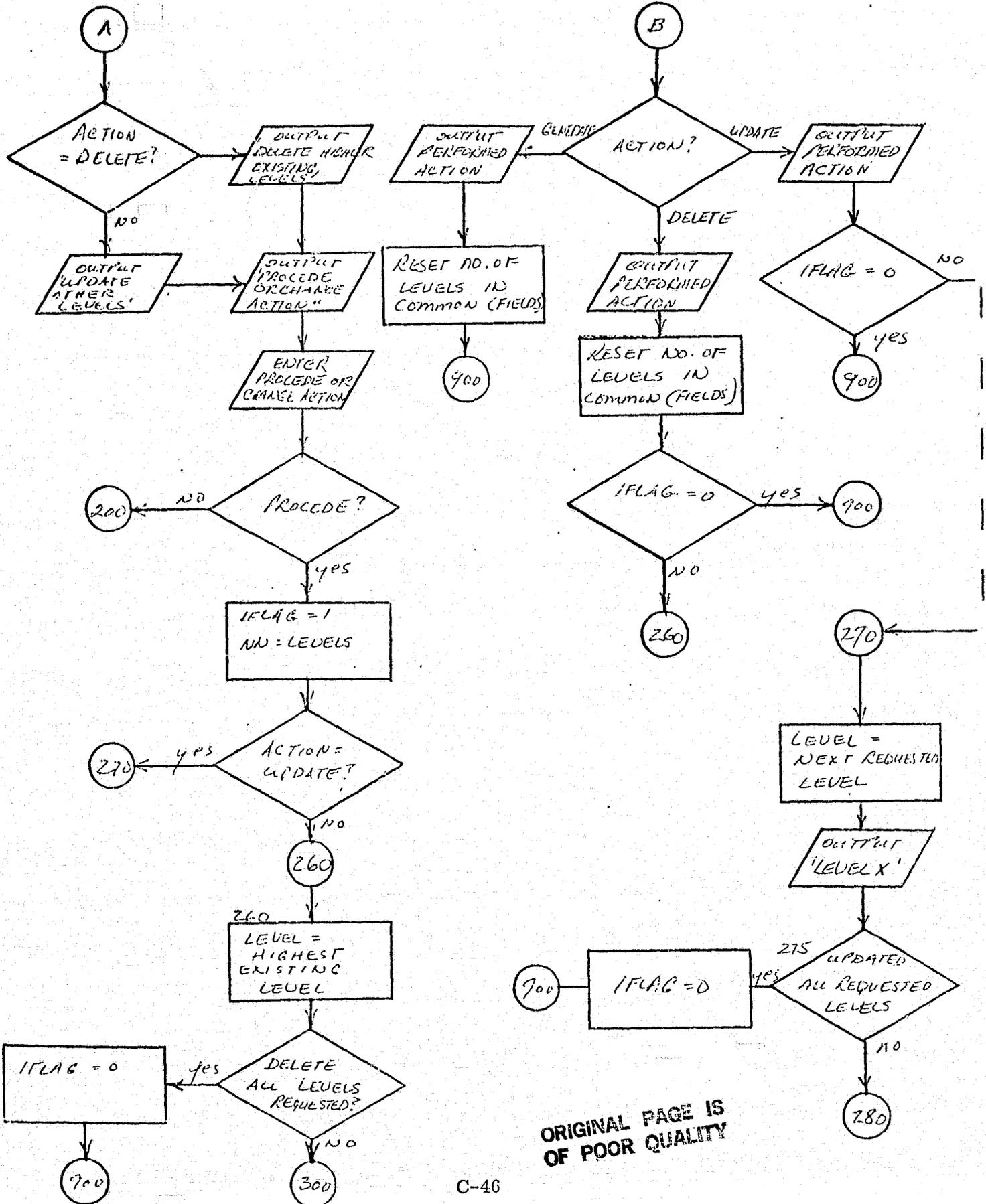






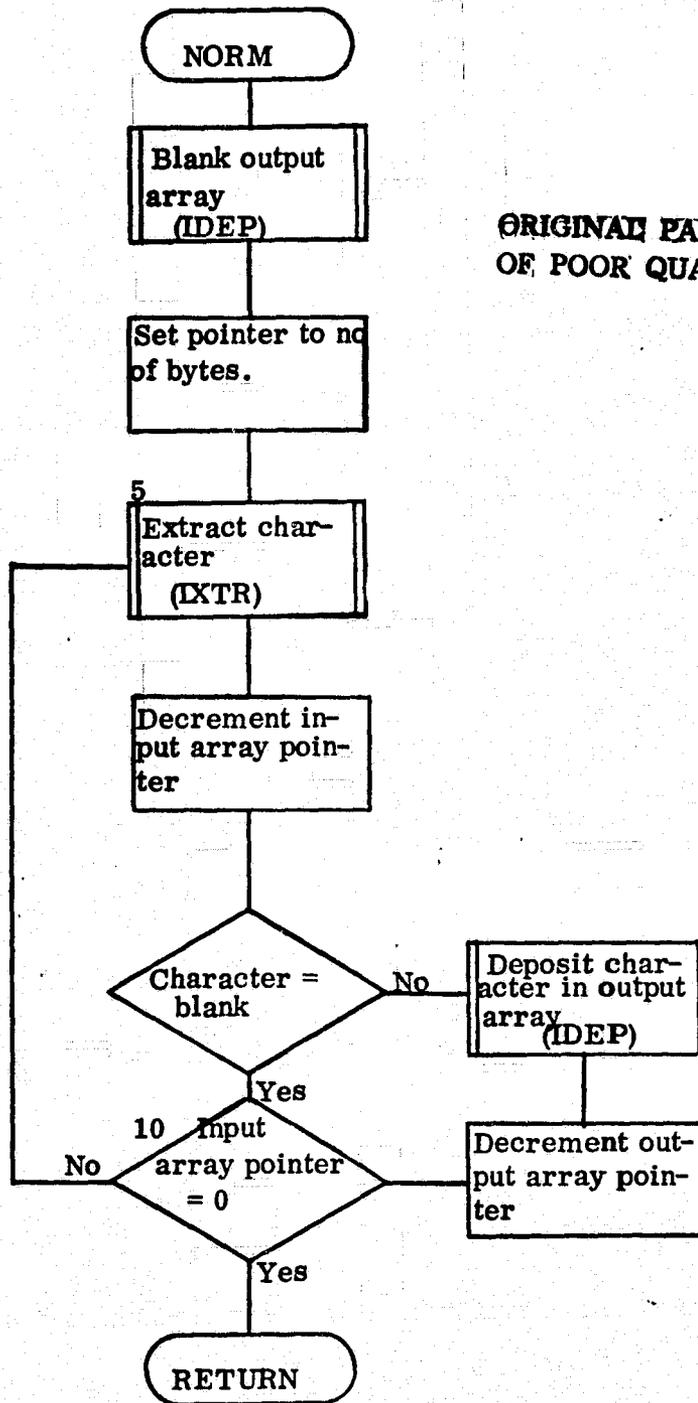


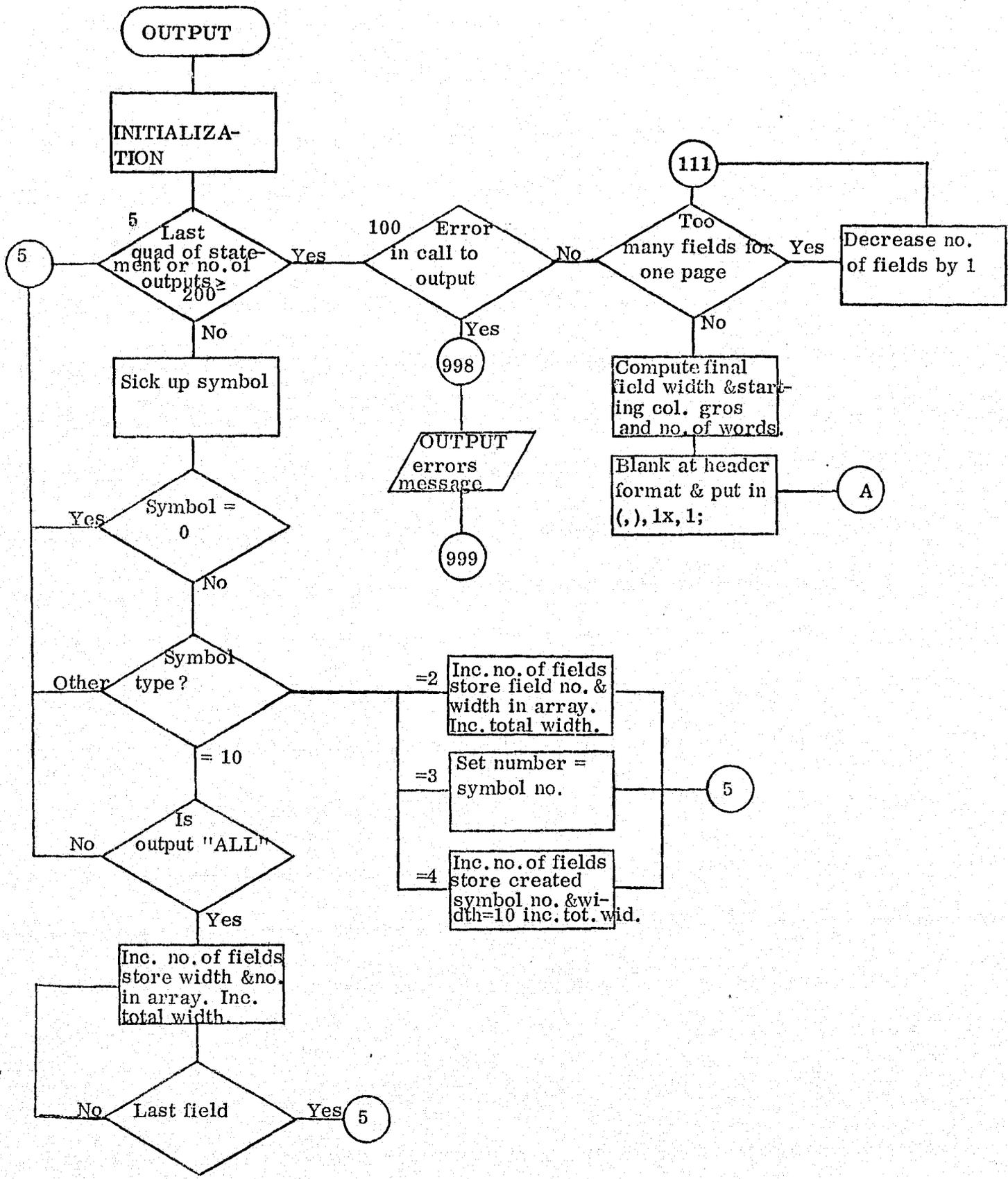


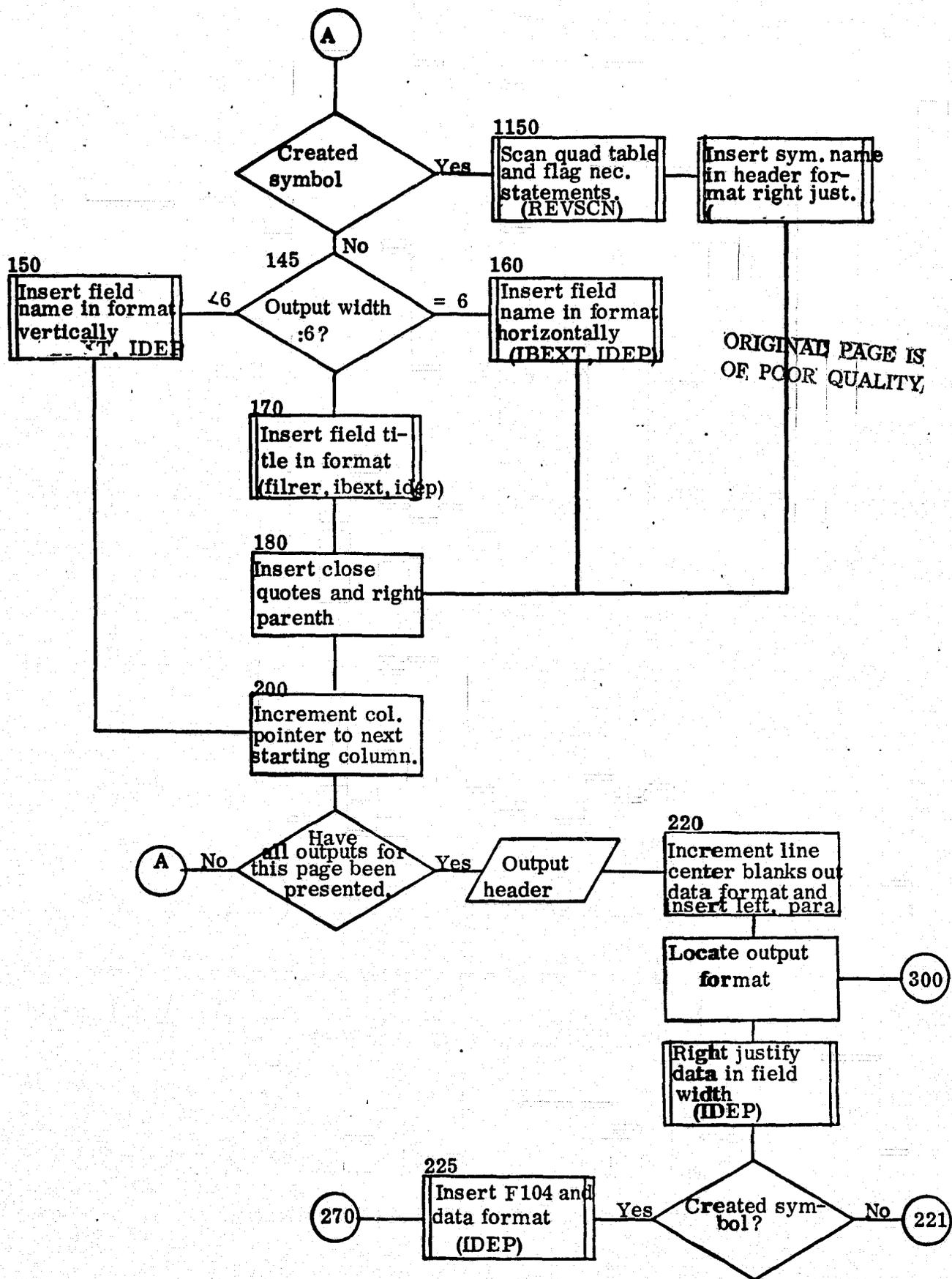


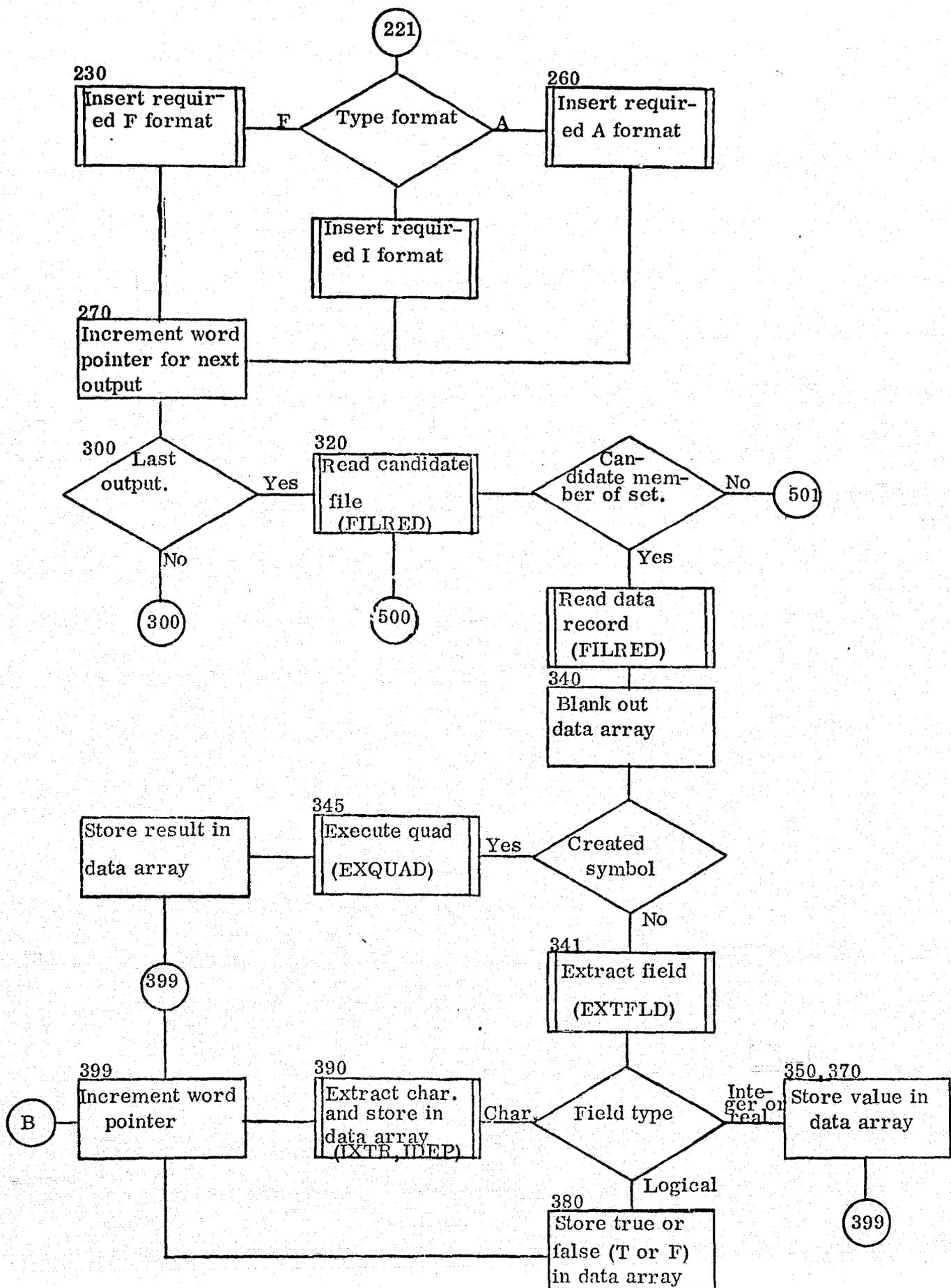
ORIGINAL PAGE IS OF POOR QUALITY

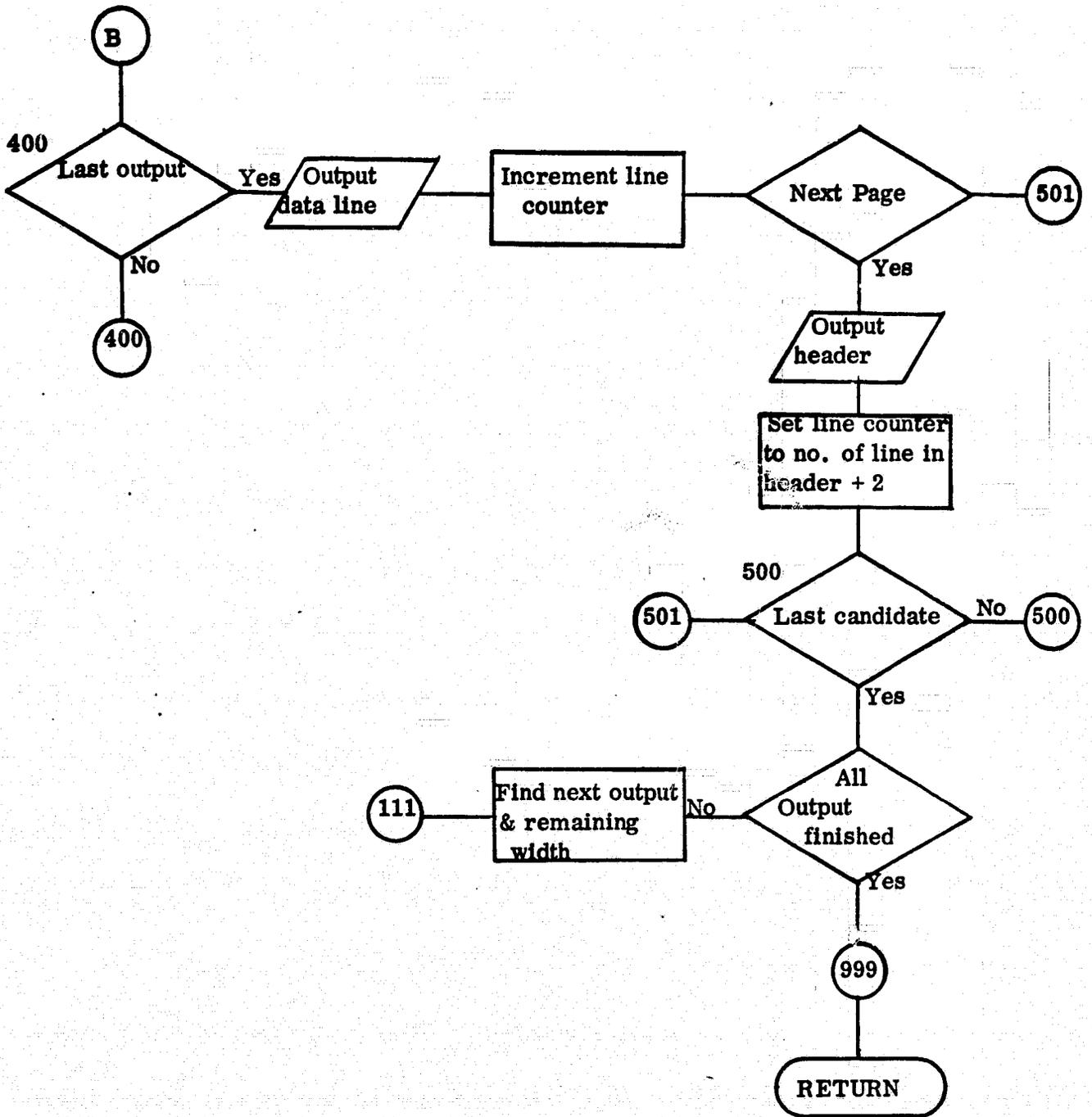
ORIGINAL PAGE IS  
OF POOR QUALITY.

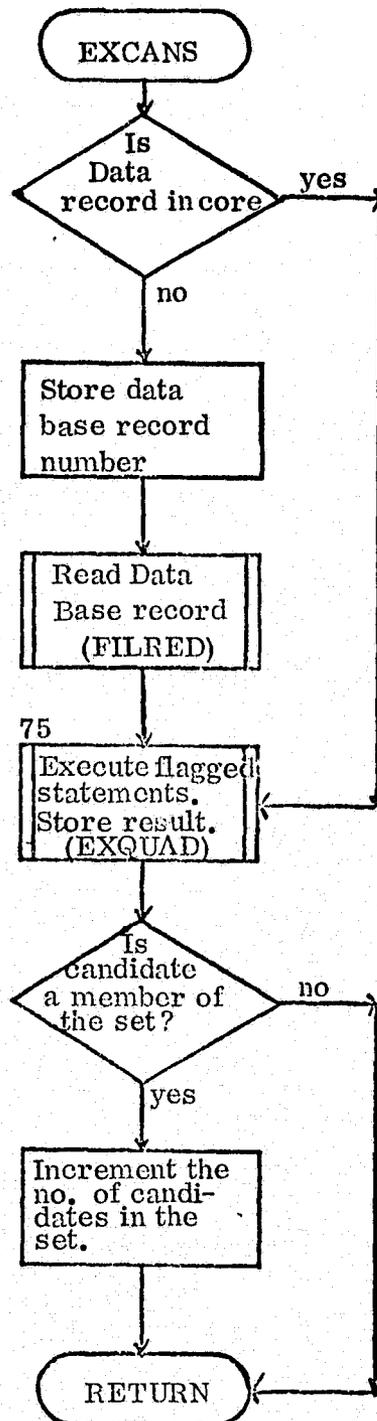






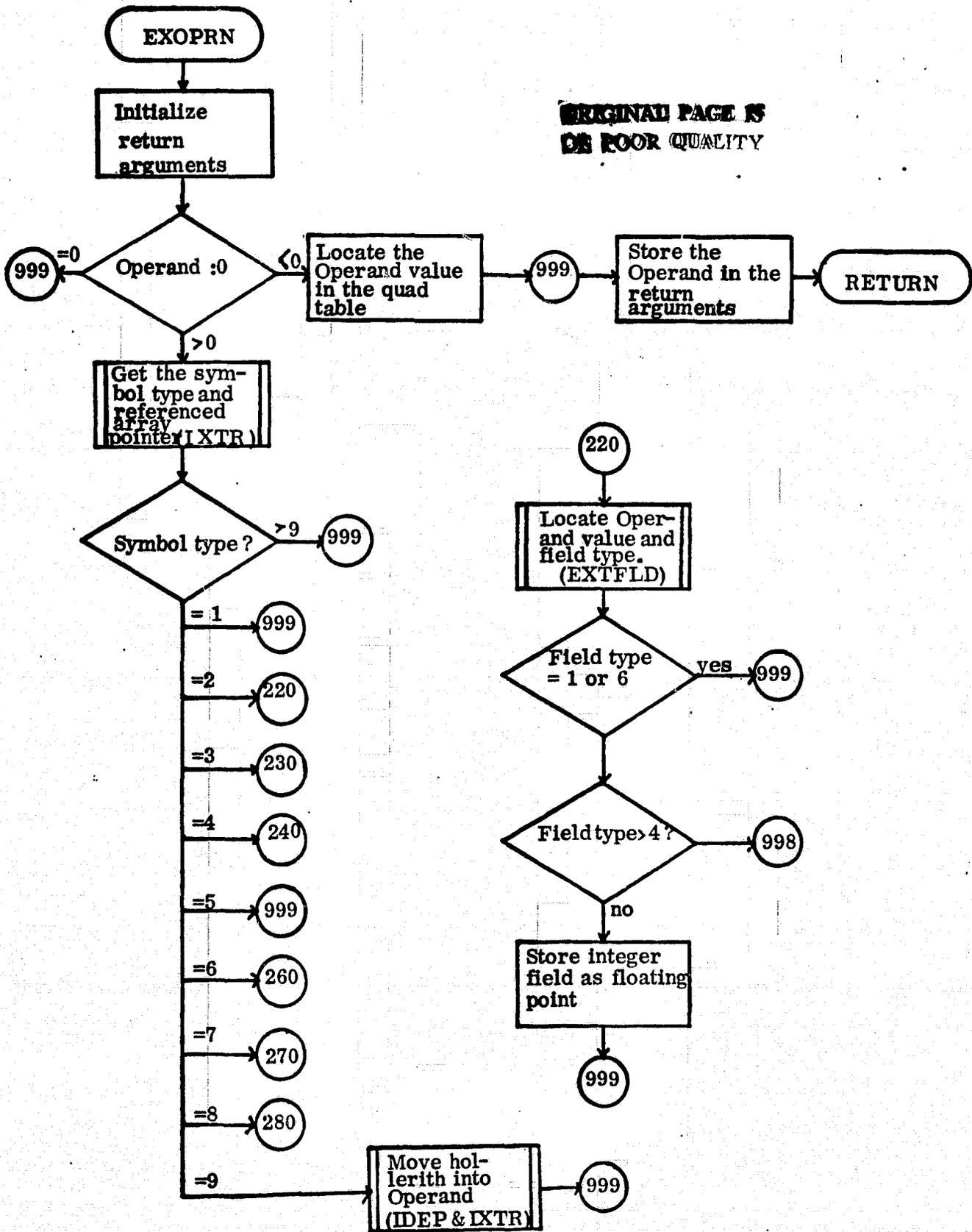


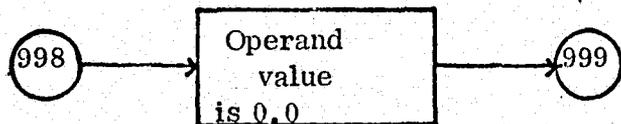
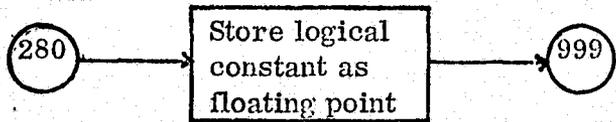
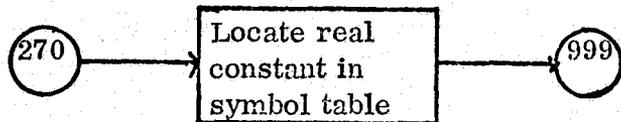
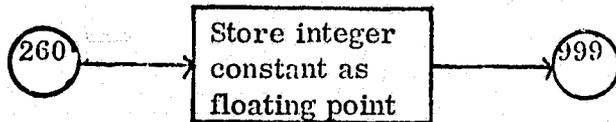
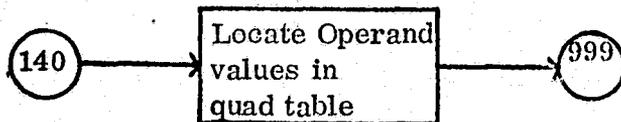
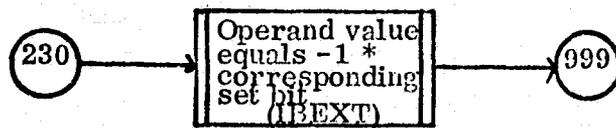


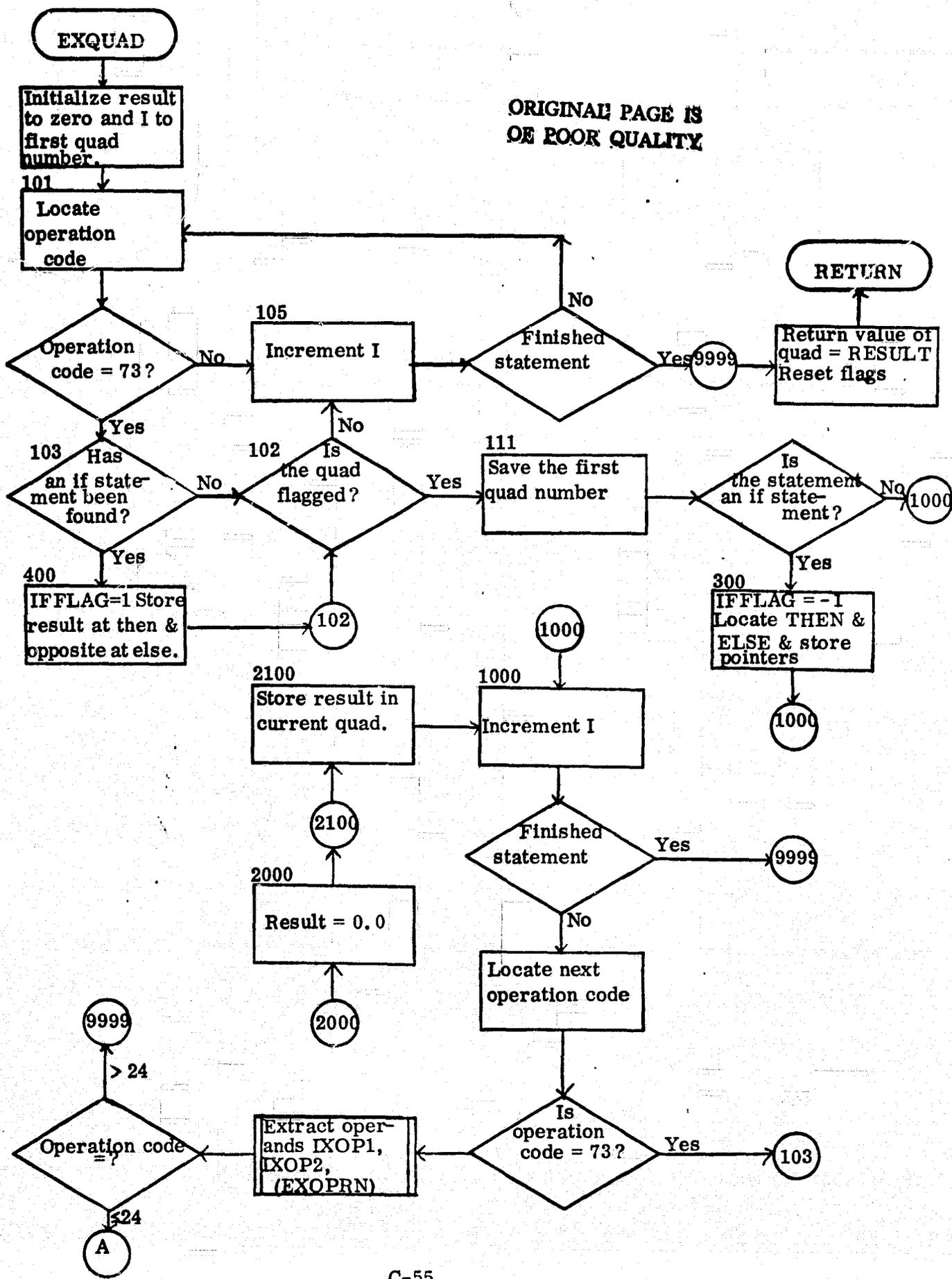


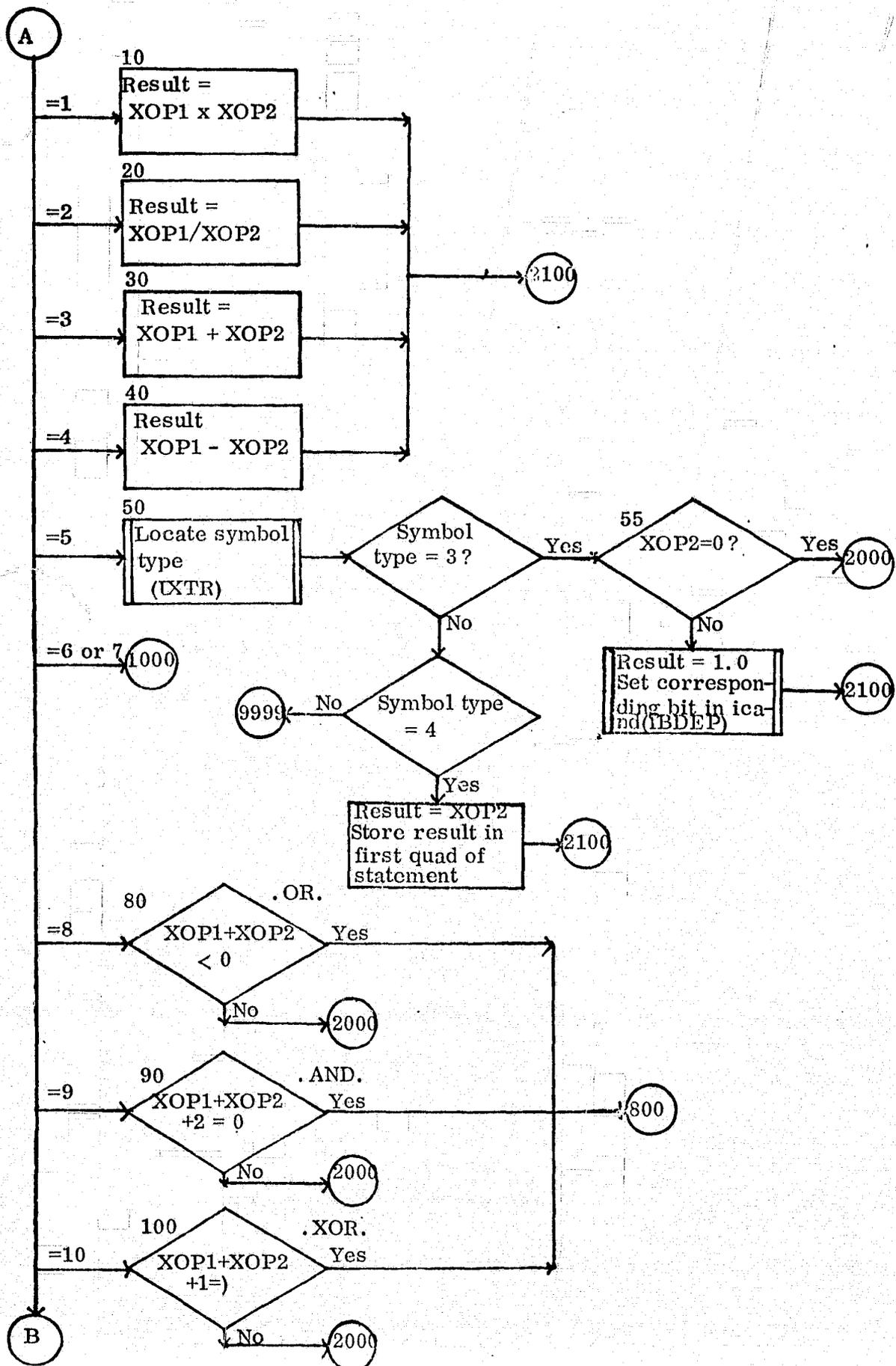
ORIGINAL PAGE IS  
OF POOR QUALITY

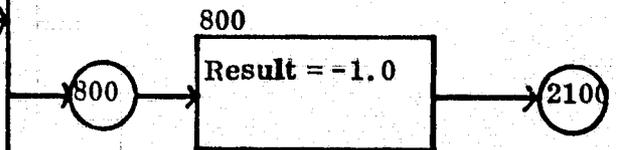
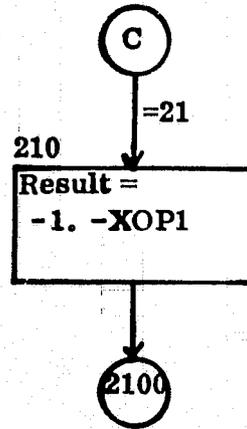
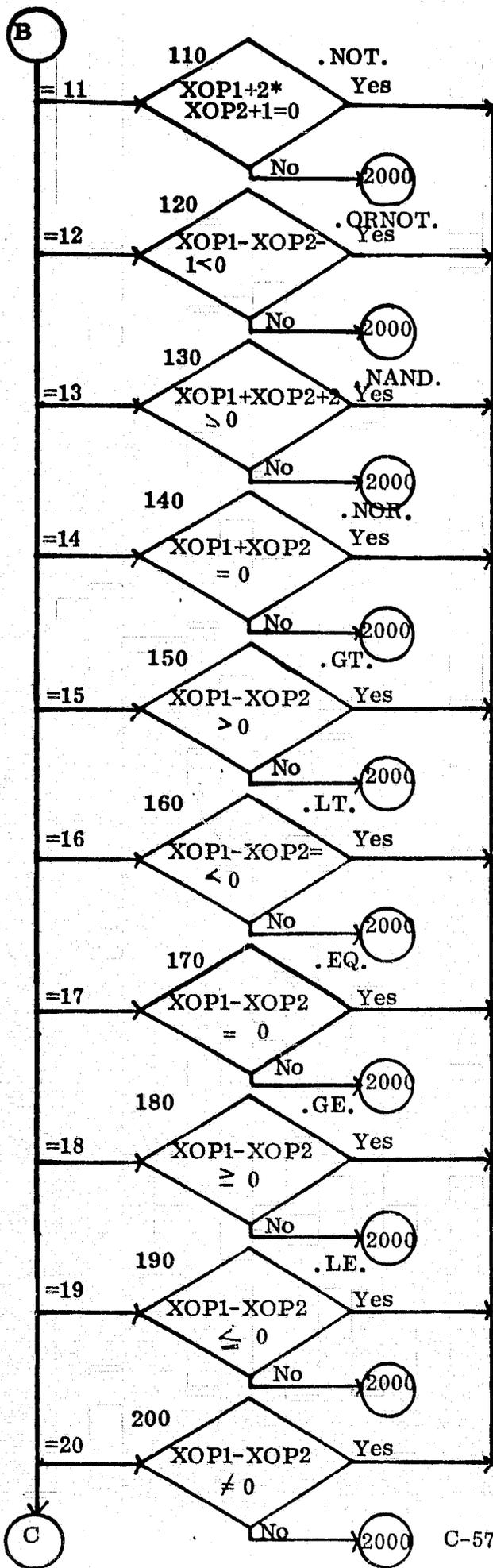
ORIGINAL PAGE IS  
OF POOR QUALITY

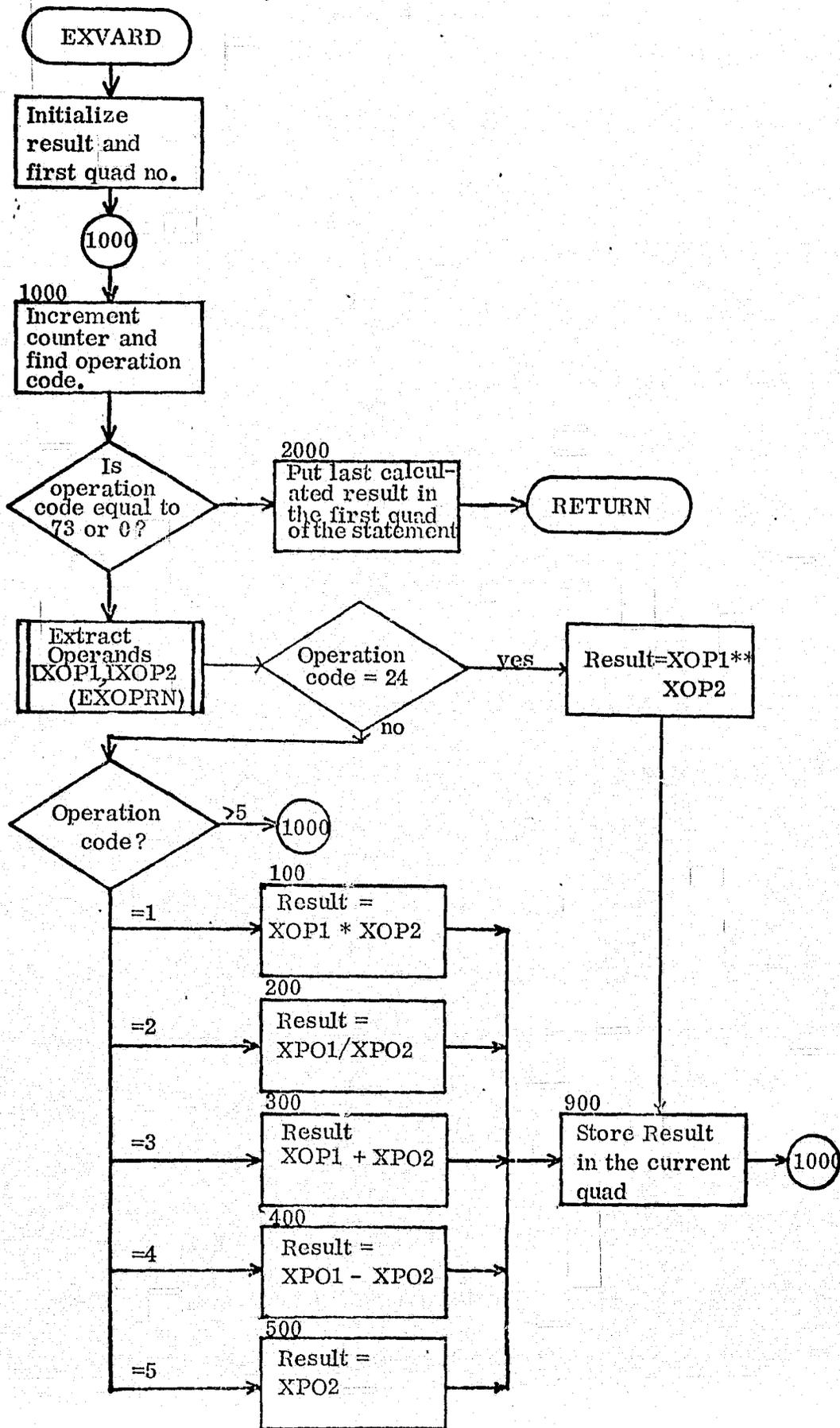


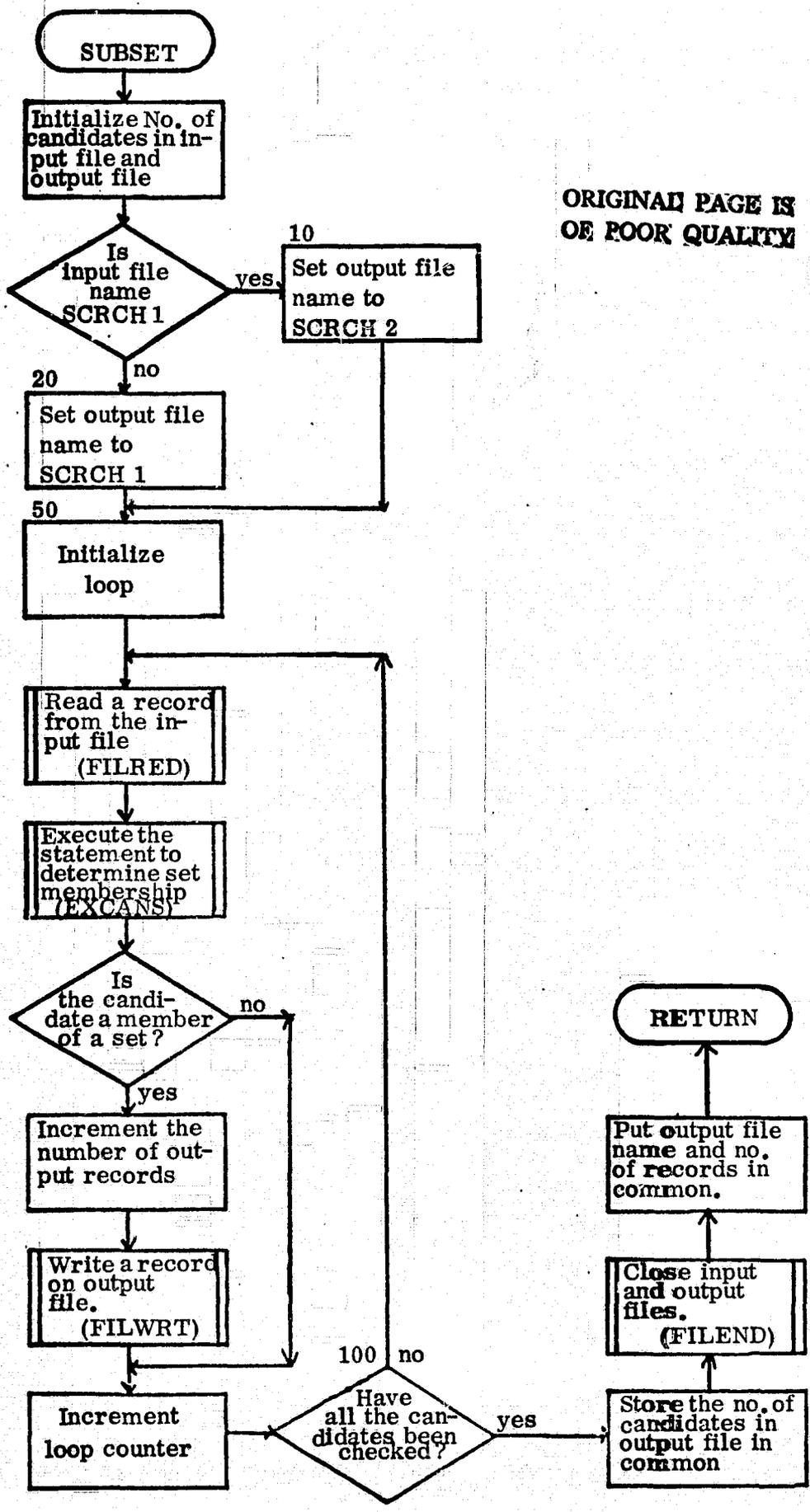












ORIGINAL PAGE IS OF POOR QUALITY

REVSCAN

I = initial quad  
IPL = last operand  
I2 = I  
N1 = ISYMS(4,N)

4  
6  
I1 = I2

5  
I = I + 1

I > IQL

10  
Restore original pointer in symbol table  
ISYMS(4,N) = N1

RETURN

20

10  
Flag if statement  
II = IFS + 1

IQU(1,II) = 0 or 73

8

L = operand

Is L a created symbol

scan for other created symbols (RVSCAN)

II = II + 1

6

scan for created symbols (RVSCAN)

IQU(2,I) = 4

I2 = I

Save IFS at same pointer  
IFS = I

IQU(2,I-1) = N

IQU(1,I-1) = 5

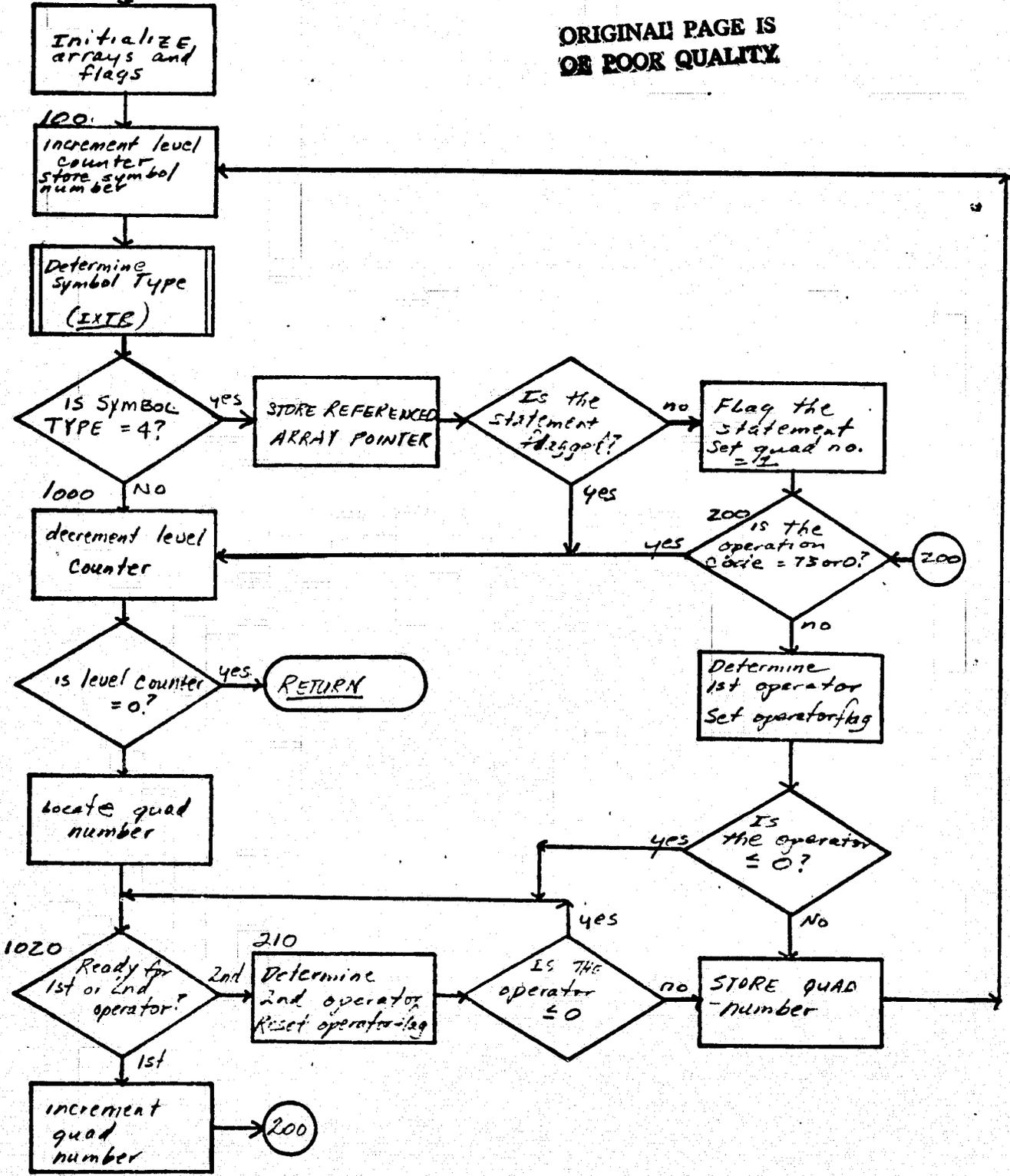
IQU(3,I1) < 6

IQU(2,I1) >= 32

8  
Store I1 in symbol table  
ISYMS(4,N) = I1

RVSCAN

ORIGINAL PAGE IS  
OF POOR QUALITY



SUMSET

INITIALI  
Z = ICON(2) + 1  
IFLAG = 1 NSET = 0  
NFLD = 0 NSUM = 0  
NUAR = 0 L = 3

100  
Pick up argument  
NSYM = EQU(L, I)

L = 2  
IFLAG = 0

Is  
NSYM = 0

600  
NQUAD = I  
I = I + 1

IQU(I, I) = 5

200  
NFLD =  
ISYMS(4, NSYM)

SYMBOL  
TYPE

400  
IFLAG = 0

450  
NANSUM = NSYM  
KAMT = 0  
SUM = KAMT  
I = I + 1

300  
NSET =  
ISYMS(4, NSYM)

NUAR =  
NSYM

601  
IQU(1, I) = 73 or 0

L = 3

L = 3

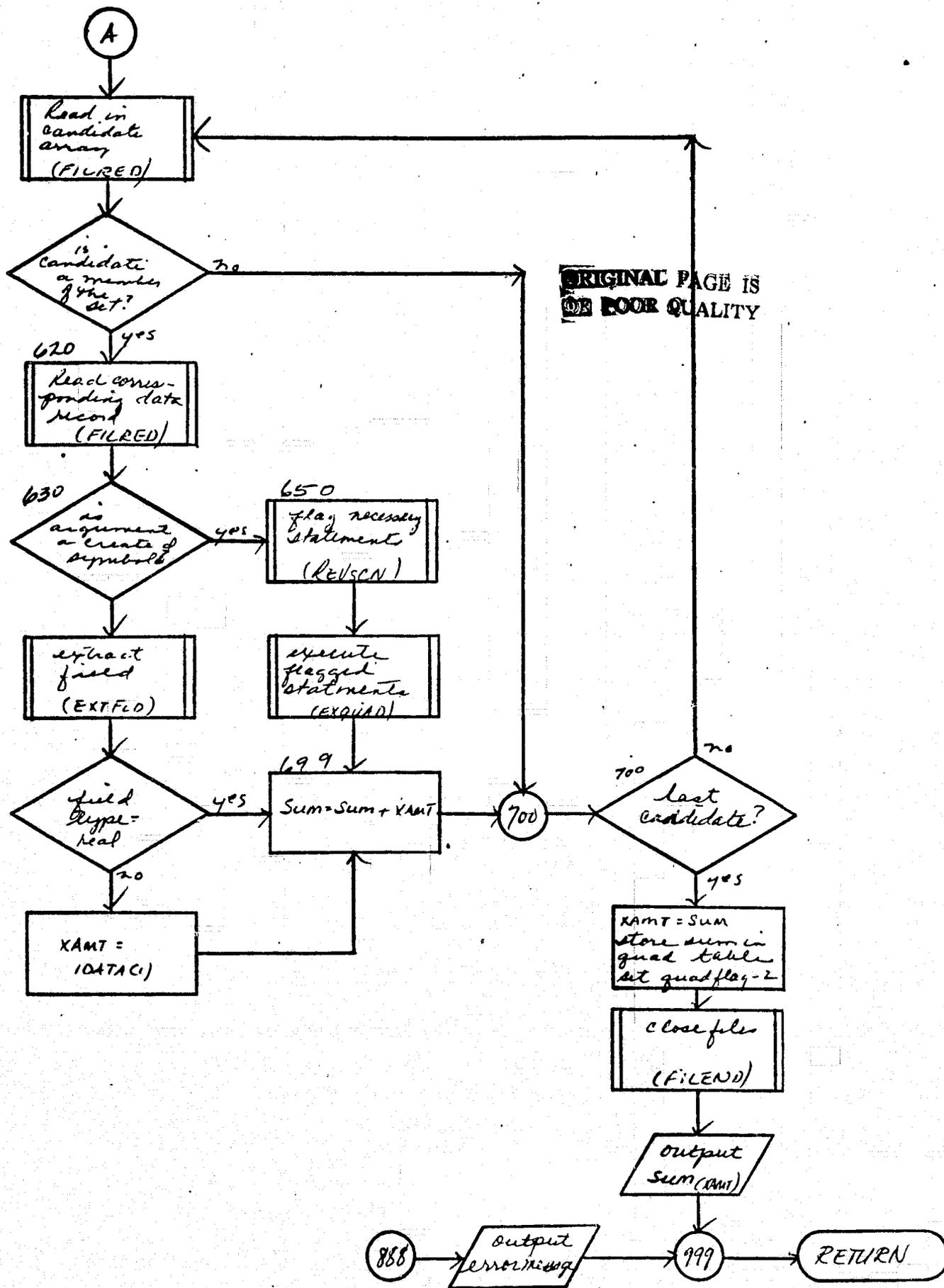
NSET  
or  
NSUM = 0

L = 2  
I = I + 1

NUAR = 0  
and  
NFLD = 0

A

ORIGINAL PAGE IS  
OF POOR QUALITY



ORIGINAL PAGE IS OF POOR QUALITY