NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Technical Report No. 32-997

A Computer Program for Simplifying Incompletely Specified Sequential Machines Using the Paull and Unger Technique

M. M. Ebersole
P. E. Lecoq

JET PROPULSION LABORATORY
CALIFORNIA INSTITUTE OF TECHNOLOGY
PASADENA, CALIFORNIA

November 15, 1968

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Technical Report No. 32-997

A Computer Program for Simplifying Incompletely Specified Sequential Machines Using the Paull and Unger Technique

M. M. Ebersole
P. E. Lecoq

Approved by:

Joseph L. Sauvain for
R. V. Morris, *Manager*
Guidance and Control Analysis and Integration Section

JET PROPULSION LABORATORY
CALIFORNIA INSTITUTE OF TECHNOLOGY
PASADENA, CALIFORNIA

November 15, 1968

Prepared Under Contract No. NAS 7-100
National Aeronautics & Space Administration

CONTENTS

TABLES

TABLES (Cont'd)

FIGURES

FIGURES (Cont'd)

ABSTRACT

This report presents a description of a computer program mechanized to perform the Paull and Unger process of simplifying incompletely specified sequential machines. An understanding of the process, as given in Ref. 3, is a prerequisite to the use of the techniques presented in this report. This process has specific application in the design of asynchronous digital machines and was used in the design of operational support equipment for the *Mariner* 1966 central computer and sequencer. A typical sequential machine design problem is presented to show where the Paull and Unger process has application. A description of the Paull and Unger process together with a description of the computer algorithms used to develop the program mechanization are presented. Several examples are used to clarify the Paull and Unger process and the computer algorithms. Program flow diagrams, program listings, and a program user operating procedures are included as appendixes.

I. INTRODUCTION

The example of the design of a shift register stage, using only NAND gates, will show how the Paull and Unger process, when applied to a sequential machine design problem, yields a simplified mechanization.

Hand computations of the Paull and Unger process are tedious, and often computational errors result. The majority of the steps in the process are sufficiently well defined so that programming a digital computer to perform the process was feasible. Those steps in the process that require trial and error decisions are left up to the program user, who may enter these decisions into the computer on line via the typewriter.

Running several typical design problems on the computer has shown the merit of the computer program, both in design time savings and computational accuracy.

II. SEQUENTIAL MACHINE DESIGN EXAMPLE: SHIFT REGISTER STAGE

A. Generation of Primitive Flow Table

The following properties characterize the design and generation of a shift register stage (see Ref. 1).

(1) A stage has two inputs; I, which is the output Z of the preceding stage, and S, which is the signal to shift the bit in each stage into the next stage.

(2) When S goes up, the stage maintains the value of the bit stored on the output Z, so that the next stage may be sampling it as an input. When S goes down, Z takes on the value that I had when S was just up (Fig. 1).

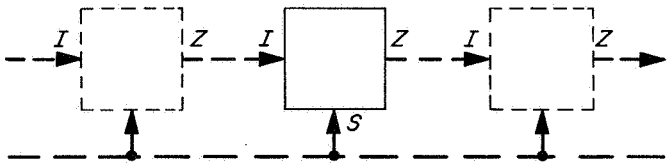


Fig. 1. Shift register stage

From the required properties, a *primitive flow table* (see Ref. 2) may be generated. It must describe the circuit performance for all possible input sequences (Table 1).

Table 1. Primitive flow, shift register stage

S,I 00	01	11	10	Z
①	2	—	3	0
—	②	4	—	0
1	—	—	③	0
—	5	④	—	0
6	⑤	7	—	1
⑥	—	—	8	1
—	5	⑦	—	1
1	—	—	⑧	1

The *circled entries* are stable states of a sequential network that describes the properties of a shift register stage. One stable state is assigned to each combination of input/output variables. *Blank entries* represent transitions within the table that cannot occur; hence, the table is *incompletely specified.*

As an example of how to read the table, consider the case when $S = 0$, $I = 1$, $Z = 0$. The sequential network will be in stable state ②. Suppose a shift pulse is generated and causes S to go to 1. The network will cycle to

the $S = 1, I = 1, Z = 0$ entry in the same row, in this case an uncircled 4, which is an unstable state. The unstable entry is an indication of which stable state the network is next to assume for a particular input variable combination; the network thus assumes stable state ④.

Suppose that the shift pulse S returns to 0, so that $S = 0, I = 1, Z = 0$. The network cycles to the 010 entry, which is unstable 5. Then, as just described, the transfer is to stable state ⑤, and the corresponding output is a 1.

The preceding sequence adheres to the desired properties of the shift register stage. All other input variable sequences will also exhibit the desired shift register properties.

B. Reduction of Number of Primitive Flow Table Rows

If it is possible to reduce the number of rows in the primitive flow table, ultimately the complexity of the network having the properties of a shift register stage will be reduced.

Basically, row reduction is accomplished by merging rows of the primitive flow table so that the desired sequential network properties are preserved.

One method for row reduction (state reduction) has been suggested by M. C. Paull and S. H. Unger (see Ref. 3). The Paull and Unger method consists of the following three steps:

(1) Determination of compatible row pairs

(2) Determination of larger row combinations, called maximal compatibles

(3) Selection of either maximal compatibles or subsets of maximal compatibles, which become single states of a reduced row network .

The three steps will be described in detail on pages 3 through 13. Let it suffice for now to say that the Paull and Unger method, when applied to the shift register problem, yields the reduced-row flow table (Table 2).

Note that rows 1, 2, and 3 of the primitive flow table have been combined to form one row, denoted as 1*. Similarly, rows 5, 6, and 7 have been combined to form one row, namely 3*. Other row assignments are shown in the tabulation to the right of Table 2.

Table 2. Reduced-row flow table for shift register stage

	S, I				
	00	01	11	10	Z
1*	(1*)	(1*)	2*	(1*)	0
2*	—	3*	(2*)	—	0
3*	(3*)	(3*)	(3*)	4*	1
4*	1*	—	—	(4*)	1

1* = 123
2* = 4
3* = 567
4* = 8

C. Shift Register Mechanization

1. Assignment of Code to Flow Table Rows

A binary code is assigned to each row in the reduced flow table. In a 4-row table, two binary variables are required. The assignment is shown in Table 3.

Table 3. Code assignment (excitation matrix)

f_1f_2	S, I 00	01	11	10	Z
1* = 00	(00)	(00)	01	(00)	0
2* = 01	—	11	(01)	—	0
3* = 11	(11)	(11)	(11)	10	1
4* = 10	00	—	—	(10)	1
	F_1F_2				

Note that a transfer from an *unstable* state entry to a *stable* state entry within a column never involves a change of both variables, f_1, f_2. The coding was done in this manner to avoid race conditions which result when two variables try to change at the same time.

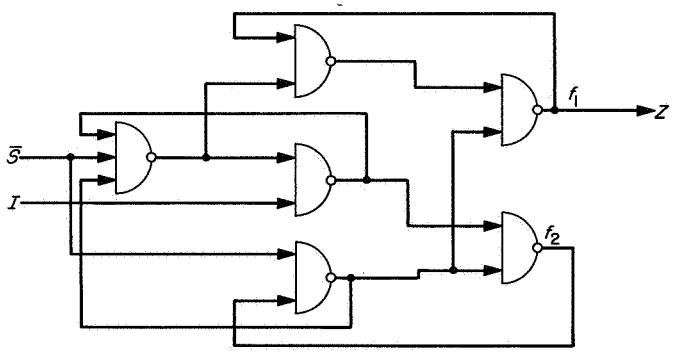


Fig. 2. NAND gate mechanization of shift register element computer mechanization of Paull and Unger Process

2. NAND Gate Mechanization

The resultant NAND gate mechanization is shown in Fig. 2. From the excitation matrix in Table 3, the logical equations for a shift register stage may be written using combinational logic techniques. Note that F_1 and F_2 are the "excitation" applied to the network and that f_1 and f_2 are the network response; hence the term "excitation matrix."

The resultant equations are

$$F_1 = f_1(S + f_2) + \bar{S}f_2; \quad F_2 = I(S + f_2) + \bar{S}f_2; \quad Z = f_1$$

3. Computer Mechanization of Paull and Unger Process

The state reduction portion of the design procedure, introduced in the preceding example, is sufficiently well defined to be programmed for a digital computer. The following sections contain a description of the Paull and Unger process and associated computer algorithms from which the process is mechanized.

III. DETERMINATION OF COMPATIBLE PAIRS

A. Paull and Unger Process

1. Input Table Format

Consider an input table, such as Table 4, which represents the incompletely specified machine to be simplified. The table has the following format.

Table 4. Input (INTAB)

	1	2	3	4	$\longrightarrow N$
1	2,—	3,—	—,—	4,—	
2	3,0	5,—	—,—	—,—	
3	4,—	6,—	3,—	—,0	
↓	5,1	3,—	0,—	1,—	
M					

The columns of the input table represent inputs of the machine to be simplified, and the rows represent machine internal states. Table entries represent the internal state the machine cycles to with the input indicated in the column, and the output of the machine with the particular row/column. Blank entries indicate "don't care," or non-specified entries. For an example of how the table is read, consider Table 4. With input 1 (column 1), and the machine in internal state 2 (row 2), the machine will cycle to the state indicated by the intersection of column 1 and row 2, in this case, state 3. The output associated with input 1 and state 2 is a zero.

2. Compatible Pair Determination from Input Table

Using the method of Paull and Unger (Ref. 3) to determine compatible pairs, a sample implication table (Table 6) is derived from a typical input table (Table 5).

Table 5. Typical input

	I_1	I_2	I_3	I_4
1	2,—	3,0	—,—	4,—
2	3,0	5,0	—,—	—,—
3	4,—	6,—	3,—	—,—
4	5,1	3,0	—,—	1,—
5	—,—	6,0	—,—	—,—
6	—,—	—,1	4,—	2,—

The numbers on the left side of Table 5 identify the rows of the table. On the left side of Table 6 the numbers identify the rows, while those across the bottom designate Table 6 columns. For each *pair of rows* of Table 5, there is one cell in Table 6. The first step is to fill in these cells, one column at a time, starting at the top of each. The location of the top left-hand cell in Table 6, which is specified by *column* number 1 and *row* 2, is referred to as position 12. This number combination is used to signify *row* numbers, only, in Table 5—that is, rows 1 and 2. Reading across Table 5 in row 1 we see a 2, and directly below it in row 2 there is a 3; the row pair number, then, is 23, the number that is inserted in cell 12 of the implication table. Continuing across rows 1 and 2 of the input table, we read a 3 and a 5—which number, also, is inserted in cell 12 of Table 6. With the exceptions described below, all cells of Table 6 are filled in the above manner.

Table 6. Implication

2	23 35				
3	24 36	34 56			
4	25	X	45 36		
5	36	56	✓	36	
6	X	X	34	X	X
	1	2	3	4	5

Table 7. Final implication

2	X				
3	X	X			
4	X	X	45 36		
5	36	X	✓	36	
6	X	X	34	X	X
	1	2	3	4	5

If identical numbers make up pairs for all input columns of Table 5, checkmarks—instead of numbers—are inserted in the corresponding Table 6 cells; for example, in rows 3 and 5, all row-pairs are made up of either two blanks or two 6's, so a check mark is placed in the 35 cell of Table 6. If a contradictory row-pair output information (0, 1) exists in any input column, an X is placed in the corresponding cell of Table 6; for example, in rows 2 and 6 of Table 5, the output disagreement in column 2 makes the pair incompatible, so an X is inserted in cell 26 of the implication table. The same is true for rows 46, 56, and 24.

After all entries have been made in the implication table cells on the first pass through the table, all cells other than those with X's or checkmarks are inspected, and a final implication table (Table 7) is derived. If the location designation of any X cell is the same number as a row-pair, an X is substituted for that row pair number; for example, in Table 6 an X is inserted in position 13 because this cell contains a 24 entry, and there is an X in the 24 position. Position 56 has an X in Table 6, so positions 23 and 25, which also contain 56 entries, receive X's. Position 12 receives an X because it contains a 23 entry, and 14 because it contains a 25.

The process above is repeated until no new X's are added. At this point the row-pairs corresponding to cells with X entries are *incompatible*; the rows corresponding to *non-X* entries are *compatible.* (Table 7 was shown for clarity; in practice, one implication table can be used for the complete process.)

B. Computer Mechanization

The computer equivalent of the implication table is represented in the computer by two arrays. One array, called ITTAB, stores implied row-pairs. The other array INDEX is used to store pairwise compatibility information. The input table is stored in the computer and given the name INTAB.

1. ITTAB

ITTAB has N columns (the same number of columns as the input table), and the number of ITTAB rows is determined by the number of pairwise combinations of rows of the input table. For example, if the input table has 6 rows ($M = 6$), ITTAB will have 15 rows $[M(M-1)/2]$.

a. ITTAB generation. For the example, in Table 5, ITTAB will be generated per Table 8. A table (LOOK-

Table 8. ITTAB matrix

	Inputs			
	1	2	3	4
1	6	11	0	0
2	7	12	0	0
3	8	0	0	3
4	0	12	0	0
5	0	0	0	7
6	10	15	0	0
7	11	11	0	0
8	0	15	0	0
9	0	0	0	0
10	13	12	0	0
11	0	0	0	0
12	0	0	10	0
13	0	12	0	0
14	0	0	0	1
15	0	0	0	0

UP) is used to assign a number to each pairwise combination (e.g., pair 12 is assigned No. 1; pair 56 is assigned No. 15). In this way, pairs may be processed as individual entities.

b. LOOKUP table. A LOOKUP table is generated with $M-1$ entries. (M = number of rows in INTAB). LOOKUP (I) entries are calculated as follows:

(1) LOOKUP (1) = 0

(2) LOOKUP $(I)_{I=2,\,M-1}$ = LOOKUP $(I-1)+M-I+1$

(3) LOOKUP entries for the example problem are as in Table 9.

Table 9. LOOKUP

1	0
2	5
3	9
4	12
$(M-1)$ 5	14

c. ITTAB entries. Rows of INTAB are processed from pairs 12, 13, $\cdots$16; 23, 24, $\cdots$26; 34, 35, 36; 45, 46; 56 for all inputs (INTAB columns). For each INTAB entry, a pair number is assigned and inserted in ITTAB. Non-specified pair entries result in a zero in ITTAB. For example, if in INTAB, an entry is, say, pair 24, the equivalent number assignment is computed by using LOOKUP: (pair 24) entry = LOOKUP ($K1$) + $K2$ − $K1$ = 7 in ITTAB where $K1 = 2$, $K2 = 4$.

2. INDEX Array

INDEX is used in conjunction with ITTAB to store output-compatible or incompatible information. INDEX is filled during the filling of ITTAB. INDEX will have the same number of rows as ITTAB. A (+1) in INDEX represents output agreement for a particular pair (compatibility), and a (−1) represents disagreement (incompatibility). INDEX is initially set to zero.

3. Iteration of ITTAB

After the information from INTAB has been stored in ITTAB and INDEX, ITTAB must be processed to check for implied output incompatibilities. Prior to checking for implied incompatibilities, ITTAB will be as shown in Table 8, and INDEX as in Table 10 for our example.

Table 10. INDEX

1	+1	12	9	−1	26
2	+1	13	10	+1	34
3	+1	14	11	+1	35
4	+1	15	12	+1	36
5	−1	16	13	+1	45
6	+1	23	14	−1	46
7	−1	24	15	−1	56
8	+1	25			

The ITTAB entries are now numbers or indices which refer to rows of INDEX. If a row/column entry of ITTAB refers to a row in INDEX that has a (−1), the INDEX entry for the ITTAB row being processed is changed to a (−1). For example, the ITTAB entry for row 2 (pair 12) and column 1 (input 1) is 7 (pair 24). The INDEX entry for 7 is a (−1). INDEX entry for row 2 must be changed to a (−1), since incompatibility is implied. All rows of ITTAB are processed in the same fashion. If at least one change in INDEX is made during the processing of all rows of ITTAB, the procedure is repeated for all rows until no changes are made in a complete pass through ITTAB. After the completion of this iterative process, INDEX will have the form of Table 11.

The rows of INDEX corresponding to +1 entries represent the numbers of compatible pairs. INDEX is scanned for +1 entries, and the corresponding compatible pair is printed. The compatible pairs printed are 15, 34, 35, 36, and 45.

Table 11. INDEX

1	−1	12	9	−1	26
2	−1	13	10	+1	34
3	−1	14	11	+1	35
4	+1	15	12	+1	36
5	−1	16	13	+1	45
6	−1	23	14	−1	46
7	−1	24	15	−1	56
8	−1	25			

IV. DETERMINATION OF MAXIMAL COMPATIBLES

A. Paull and Unger Process

Paull and Unger define Maximal Compatible (MC) as a set of input table rows that form a compatible, and which is not included in any larger compatible.

The purpose of finding maximal compatibles is to generate larger subsets from the already determined compatible pairs. For example, if pairs 34, 35, and 45 are compatible pairs, the subset 345 is also compatible.

Paull and Unger suggest several methods of determining maximal compatibles. Consider Table 12, where the X's indicate incompatible pairs. Start with column 1 and work to the right.

Table 12. Final implication

	1	2	3	4	5
2	X				
3					
4	X	X	X		
5		X		X	
6					X

(1) Write down the set of all states except for 1. Then write down a set of states consisting of 1, followed by all states corresponding to the non-X row in column 1. Table 12, at this point, would show 23456 and 1356.

(2) Move to the next column containing X's; say, column j. Examine those sets on the list that contain j. Replace each such set, which also includes at least one state corresponding to rows that are X's in column j, by two sets: the original set without j, and the original set without any of the states that are incompatible with j, as indicated by the X's in the j column. Repeat this process for each column, and at each stage eliminate sets included in other sets on the list.

In our example, the list of sets progresses (as below) after the indicated column is inspected.

(1)	(23456)	(1356)		
(2)	(3456)	(236)	(1356)	
(3)	(456)	(236)	(1356)	
(4)	(46)	(236)	(1356)	
(5)	(46)	(236)	(135)	(136)

The results of Step 5 are the maximal compatibles.

B. Computer Mechanization

1. Maximal Compatible Determination

A table with a variable number of rows, MACOMP, is used in the determination of maximal compatibles. The table has M columns when M is the number of original machine rows. Entries of MACOMP are either 1 or 0. Consider the example of Table 13. The initial entries of MACOMP are

Table 13. MACOMP (initial)

		1	2	3	4	5	6
(23456)	1	0	1	1	1	1	1
(1356)	2	1	0	1	0	1	1

The first row of MACOMP is determined by inserting a 1 as an entry in all locations except for 1, where a zero is inserted.

Row 2 is filled with a 1 in location 1, and 1's in other locations depending on compatibility information stored in the INDEX table. Zeros are filled in the other locations in the row.

A dummy variable, KOL, is used as an index for selecting which rows in MACOMP to process in accordance with Step 2 of the Paull and Unger maximal compatible method. KOL is also used as an index for the INDEX table.

For each value of KOL (each time all rows of MACOMP are processed), it is required to eliminate sets (rows) which are included in larger sets in MACOMP. This is

accomplished by an erase subroutine, which is mechanized as follows:

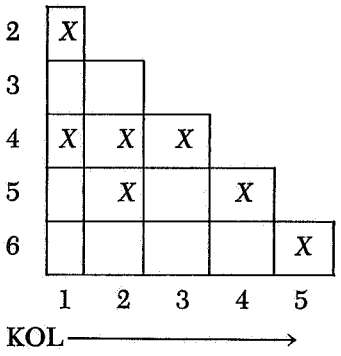
(1) A row is selected to be tested to see if it might be a subset of any other set on the MACOMP list.

(2) All other rows are subtracted, column by column, from the subset candidate.

(3) If a positive quantity results, the row in question is not a subset of the row being subtracted.

(4) If a negative or zero results for all columns, then the candidate row is a subset of the present row being subtracted. The candidate subset is then erased from the list by moving rows below the candidate up one position.

An example will help clarify the method. Given: An implied implication table ("X" information will actually be stored in the computer in INDEX).



			1	2	3	4	5	6	
KOL = 1	(23456)	1	0	1	1	1	1	1	← 1's for all columns except 1
	(1356)	2	1	0	1	0	1	1	← Zeros determined by X's

			1	2	3	4	5	6	
KOL = 2									
	(3456)	1	0	0	1	1	1	1	
	(1356)	2	1	0	1	0	1	1	← (1) Since KOL is not included in this entry, this row remains unchanged.
	(236)	3	0	1	1	0	0	1	

(2) Row 1 is replaced by 2 rows:

(a) One row is the original row, without a 1 in the KOL column.

(b) The other row has a 1 in the KOL column, and 1's in the columns specified by non-X entries in the implication table for KOL = 2.

(c) No erasure is required, since there are no subsets of larger sets.

KOL = 3 (before ERASE)

		1	2	3	4	5	6
(456)	1	0	0	0	1	1	1
(156)	2	1	0	0	0	1	1
(26)	3	0	1	0	0	0	1
(356)	4	0	0	1	0	1	1
(1356)	5	1	0	1	0	1	1
(236)	6	0	1	1	0	0	1

For each row for KOL = 2, 2 rows are generated for KOL = 3

The ERASE subroutine is now used:

Row 2 is a subset of row 5, therefore erase row 2

Row 3 is a subset of row 6, therefore erase row 3

Row 4 is a subset of row 5, therefore erase row 4

After erasure, MACOMP will have 3 entries:

KOL = 3 (after ERASE)

		1	2	3	4	5	6
(456)	1	0	0	0	1	1	1
(1356)	2	1	0	1	0	1	1
(236)	3	0	1	1	0	0	1

KOL = 4 (before ERASE)

		1	2	3	4	5	6
(56)	1	0	0	0	0	1	1
(1356)	2	1	0	1	0	1	1
(236)	3	0	1	1	0	0	1
(46)	4	0	0	0	1	0	1

KOL = 4 (after ERASE)

		1	2	3	4	5	6
(1356)	1	1	0	1	0	1	1
(236)	2	0	1	1	0	0	1
(46)	3	0	0	0	1	0	1

Final MACOMP

		1	2	3	4	5	6
(136)	1	1	0	1	0	0	1
(231)	2	0	1	1	0	0	1
(46)	3	0	0	0	1	0	1
(135)	4	1	0	1	0	1	0

The final MACOMP table is scanned for 1's, and the resultant maximal compatibles printed. In this case they are 136, 236, 46, and 135.

2. Determination of Lower Bound on Number of States in the Minimized Table

The procedure for finding maximal compatibles can also be used to determine a lower bound of the number of states in a minimized table. All that need be done is to change the *X*'s with the non-*X* entries in the implication table (in the computer exchange (+1), and −1 in the INDEX table).

The rows of MACOMP are now referred to as *maximal incompatibles*. The number of elements in the largest maximal incompatible is a *lower bound* on the number of rows in the reduced table. (The maximum number of 1's in any row of MACOMP will be the *upper bound*).

A dummy variable, INC, is used to indicate to the maximal compatible program, whether the maximal compatibles are to be determined, or the lower bound is to be determined. The program presently determines MC's first (INC = −1), and the lower bound second (INC = + or 0).

V. CHECK FOR CLOSURE: PAULL AND UNGER PROCESS

The maximal compatibles, computed in Part II of the program, are candidates for internal states of a new machine with a reduced number of internal states. There remains to check selected candidates for closure.

A. Definition

A grouping of all rows of a flow table into C sets (not necessarily disjoint) will be said to be *closed* if:

(1) Every set implied by C_i is included in one of the C sets (where i = the number of the Cth set).

(2) The output $Z(p, I_k) = Z(t, I_k)$, for all row pairs (p, t) in C_i for every I_k and i.

Condition (2) is satisfied by the compatible pair and maximal compatible portions of the program. A test to determine whether condition (1) is satisfied is demonstrated in Part B of this section.

The mechanics of testing for closure may best be described by an example. Consider an input table and the maximal compatibles generated from input table information:

Table 14. Input

	I_1	I_2	I_3	I_4
1	—	3,1	5,1	2,1
2	5,0	—	—	—
3	6,0	6,1	—	—
4	—	—	2,1	—
5	—	6,0	1,0	4,1
6	3,0	—	2,0	3,1

Table 15. Maximal compatibles

1	1234
2	36
3	56
4	25

B. Closure Procedure: Attempt 1

(1) Choose from the maximal compatible table entries whose union covers all of the internal states of the original machine. In our example, an obvious choice would be entries 1 and 3: 1234 U 56 = 123456.

(2) Assign new numbers to the chosen entries so as to distinguish them from original input table entries:

$1^* = 1234$

$2^* = 56$

(3) The chosen sets must be checked vs the input table for all inputs for satisfaction of condition (1) in the definition of closure.

(a) 1^* with input I_1 implies 56:

$1^*(I_1) \rightarrow 56,$

since 56 = 2^*, condition (1) of the closure definition is met.

(b) 1^* with input I_2 implies 36:

$1^*(I_2) \rightarrow 36,$

since 36 is not included as one of the chosen sets, condition (1) of the closure definition is not satisfied. The chosen sets, therefore, do not satisfy the closure criteria and another selection must be made.

It should be noted that since 1^* and 2^* form the only covering of the original machine states using two entries, at least three entries (rows of the reduced machine) are required.

C. Closure Procedure: Attempt 2

(1) Since states 1 and 4 of the original machine are members of only maximal compatible set 1, it is imperative that set 1, or subsets of set 1, be included in the group of chosen candidates for the reduced machine.

Suppose subsets 12 and 34 of 1234 are chosen. (Subsets of larger compatible sets are also compatibles.) Pair 56 is chosen so that the union of 12, 34, 56 forms a covering of input states of the original machine.

(2) Let $1^* = 12$

$2^* = 34$

$3^* = 56$

(3) Checking each chosen item, 1^*, 2^*, and 3^* vs the input table for all inputs $I_1 \rightarrow I_4$ will appear as in Table 16.

Table 16. Input

	I_1	I_2	I_3	I_4
1^* (1, 2)	—, 5 → 3^*	3, — → 2^*	5, — → 3^*	2, — → 1^*
2^* (3, 4)	6, — → 3^*	6, — → 3^*	—, 2 → 1^*	—, — → —
3^* (5, 6)	—, — → 2^*	6, — → 3^*	1, 2 → 1^*	4, 3 → 2^*

(4) The check shows that all of the chosen items for all inputs imply pairs that are also one of the chosen subsets. Thus, condition (1) of the closure definition is satisfied. Blank entries are "don't cares" and may be given a value of any state desired (e.g., pair 34 implies 6 —). The blank is given a value of 5, such that pair 34 implies 65. (The order of the states is immaterial: 65 = 56.) Five-six is one of the chosen subsets.

(5) The final reduced machine table is determined.

Table 17. Reduced machine table

	I_1	I_2	I_3	I_4
1^*	3^*,0	2^*,1	3^*,1	1^*,1
2^*	3^*,0	3^*,1	1^*,1	—,—
3^*	2^*,0	3^*,0	1^*,0	2^*,1

The original 6-row machine has been reduced to a 3-row machine. Note that the resultant 3-row machine is not the only 3-row machine that can be generated.

D. Computer Mechanization

This part of the program consists of (1) a recognizer routine to accept user inputs from the typewriter, and (2) a closure routine.

1. Recognizer routine

The selection of candidate sets from the maximal compatible list is left to the discretion of the program user. A method for selection of candidate sets (other than enumeration) has been suggested by A. Grasselli and F. Luccio (see Ref. 1); however, programming the suggested method would necessitate a two fold expansion of an already lengthy program. The user, communicating with the computer via the typewriter, has several options:

a. SELECT. The user may select items from the printed list of maximal compatibles by typing the word *SELECT*, followed by those items separated by commas that are to be tested for closure.

In the select mode, the program will recognize the letter S and branch to a routine which places a number corresponding to the number of the item selected in the $M + 1$ column of the MACOMP table, which is generated in Part II of the program. For example, consider the following list of maximal compatibles that have been printed as the output of Part II of the program:

Table 18. Maximal compatible list

1	1	2	3	4
2	3	6		
3	5	6		
4	2	5		

Suppose the user types *SELECT* 1, 3. The resultant MACOMP table will appear in the computer as

Table 19. MACOMP after *SELECT*

1	2	3	4	5	6	$M+1$
1	1	1	1	0	0	1^*
0	0	1	0	0	1	0
0	0	0	0	1	1	2^*
0	1	0	0	1	0	0

b. ADD. The user may add items to the present list of maximal compatibles by typing the word *ADD*, followed by the item to be added. The item will be automatically added to the bottom of the list of maximal compatibles.

For example, consider the list in Table 18, and suppose the user types *ADD* 3, 4; the resultant maximal compatible table will appear in the computer as

Table 20. MACOMP after *ADD*

1	2	3	4	5	6	$M+1$
1	1	1	1	0	0	0
0	0	1	0	0	1	0
0	0	0	0	1	1	0
0	1	0	0	1	0	0
0	0	1	1	0	0	0

c. AND. The word *AND* can be used in place of the word *ADD*, e.g., a typed sentence might be *ADD* 3, 4 *AND* 1, 2 *AND*, etc.

d. REPLACE. The user may replace items on the list of existing maximal compatibles by typing the word *REPLACE*, followed by the number of the item to be replaced, followed by a slash (/), and the replacement item.

Example:

Consider the previous example (Table 16), where it became necessary to separate four-member set 1234 into two sets, 12 and 34.

Table 21. Maximal compatibles (MACOMP) in computer

1	1	1	1	1	0	0
2	0	0	1	0	0	1
3	0	0	0	0	1	1
4	0	1	0	0	1	0

If the user types

ADD 3, 4
REPLACE 1 / 1, 2
SELECT 1, 3, 5,

the resultant table will be

Table 22. Resultant MACOMP after *ADD*, *REPLACE*, and *SELECT* instructions

	1	2	3	4	5	6	$M+1$
1	1	1	0	0	0	0	1*
2	0	0	1	0	0	1	0
3	0	0	0	0	1	1	2*
4	0	1	0	0	1	0	0
5	0	0	1	1	0	0	3*

At this point, the user is ready to test the selected subsets as indicated in the $M + 1$ column of MACOMP.

e. GO. The typing of the word *GO* calls the closure subroutine, which tests the selected subsets for closure.

2. Closure Routine

a. Arrays. The closure routine makes use of four arrays:

(1) INTAB: the original machine input information used in Part I of the program.

(2) MACOMP: the maximal compatible table generated in Part II of the program. The $M + 1$ column of MACOMP will contain an indication of whether or not a set is to be tested for closure.

(3) KTEMP: a temporary storage array.

(4) KEQUIV: after the completion of the closure test, KEQUIV will contain the reduced machine information to be printed.

b. Sequence of operation. The closure routine may best be described with the aid of an example. Consider the input table (INTAB) and maximal compatible (MACOMP) of Tables 23 and 24.

Table 23. Input

	I_1	I_2	I_3	I_4
1	—	3,1	5,1	2,1
2	5,0	—	—	—
3	6,0	6,1	—	—
4	—	—	2,1	—
5	—	6,0	1,0	4,1
6	3,0	—	2,0	3,1

Table 24. MACOMP

	1	2	3	4	5	6	$M+1$
1	1	1	0	0	0	0	1*
2	0	0	1	0	0	1	0
3	0	0	0	0	1	1	2*
4	0	1	0	0	1	0	0
5	0	0	1	1	0	0	3*

(1) The $M + 1$ column of MACOMP is scanned until a positive quantity is found, indicating a MACOMP row to be tested for closure. In our example, a "1" is found opposite the 1st row; therefore, the 1st row entry is to be tested for closure.

(2) The contents of row 1 of MACOMP serve as indices for rows of the input table. The contents of MACOMP (1) point to rows 1 and 2 of the input table.

(3) The contents of the input table for a particular I_n serve as pointers back to the MACOMP table. A temporary array, KTEMP, is used to store selected input table information. In the example, the pair in INTAB corresponding to column I_1, and rows 1 and 2 is (—, 5). Since blanks are ignored, the information stored in KTEMP is

	1	2	3	4	5	6
KTEMP	0	0	0	0	1	0

(4) By subtracting KTEMP bit by bit from all MACOMP rows which have a positive quantity in the $M + 1$ column, it is determined whether the KTEMP set is a subset of one of the MACOMP sets; if so, the condition for closure is satisfied. In our example, KTEMP is a subset of MACOMP (3).

Table 25. Intersection of MACOMP and KTEMP

	1	2	3	4	5	6
KTEMP	0	0	0	0	1	0
MACOMP (1)	1	1	0	0	0	0
MACOMP (3)	0	0	0	0	1	1
MACOMP (5)	0	0	1	1	0	0

(5) An entry may now be made in the KEQUIV table. The KEQUIV table is a representation of the new machine. The entry will be the number in the $M + 1$ column where the closure condition was met.

Table 26. Entry in KEQUIV

	I_1	I_2	I_3	I_4
1*	2*, 0---------->			

The output information stored in INTAB is transferred to KEQUIV after the closure condition is satisfied.

(6) If the test for closure fails, the program is returned to the hands of the user so that he may select or add other sets to be tested for closure.

(7) Steps 1 through 5 are repeated for all I_n and all selected MACOMP sets. If Step 4 satisfies the condition for closure for all selected sets for all inputs, then KEQUIV will be printed as the new reduced machine. The final resultant KEQUIV will be

Table 27. Resulting reduced-row machine representation in KEQUIV

	I_1	I_2	I_3	I_4
1*	2*,0	3*,1	2*,1	1*,1
2*	3*,0	2*,0	1*,0	3*,1
3*	2*,0	2*,1	1*,1	—,—

APPENDIX A

Operating Procedure

I. LOADING THE PROGRAM

The program consists of six parts, PAU1 through PAU6. The six links are loaded on the disk and called by an XEQSPAU1 MONITOR control card. PAU2 through PAU4 are automatically called in order. PAU4, the communicator, then calls PAU5 or PAU6 as required. PAU5 and PAU6 each call back PAU4 at their completion. When the operator desires to terminate the program, PAU4 returns control to MONITOR.

The listing of Appendix C contains the source deck and required MONITOR control cards to load the six links onto the disk of a 1620/1622 II computer, controlled by MONITOR II D. The JOB card may have to be changed to comply with the operating procedures at different installations.

If the program is to be used on other computers with FORTRAN compilers, changes to the body of the program may be needed to transfer the data from one link to another. These changes are left up to the user.

II. PROGRAM INPUT/OUTPUT

The program is divided into three operational phases: Phase I, II, and III. The inputs required for and outputs generated by each phase (Fig. A-1) are as follows:

A. Phase I

1. Input: Data Card Preparation

The Input consists of the information contained in the flow table that represents the sequential machine to be simplified. The format of the input data follows closely the standard format of a sequential flow table.

The first card must specify the limits of the machine to be simplified. The first field contains M, the number of states or rows. The second field contains N, the number of inputs or columns (Table A-2). Throughout the program, all input and output fields will be of four digits, so M must be in column 4, while N is in column 8. The maximum M for this program is 20; maximum N is 10. For the flow table itself, output and next-state information is represented on subsequent cards by a zero or number. A non-specified entry is represented by a minus one (−1). The information for each internal state is punched on one card.

For an example of the required input data format, consider the machine to be simplified, defined by Table A-1. Seven data cards are required; one to define the dimensions and one for each machine state as shown in Table A-2. For each input, the next-state information is followed by the corresponding output.

2. Input Data Format

Table A-1. Typical sequential machine flow

	I_1	I_2	I_3	I_4
1	4,—	—,—	3,0	2,—
2	—,—	—,—	5,0	3,0
3	—,1	3,1	6,—	4,—
4	1,—	—,—	3,0	5,1
5	—,—	—,—	6,1	—,0
6	2,1	4,1	—,1	—,0

3. Output Data

The computer output of Phase I is:

(1) A printed or typed list of compatible pairs.

(2) A printed or typed list of maximal compatibles.

(3) A number indicating the least number of states possible in a reduced machine.

For a typical printout, see Fig. A-2. At the completion of Phase I, the typewriter will type "BEGIN PHASE II."

B. Phase II

The primary purpose of Phase II is to allow for communication between the program user and the computer so that the user may select items from the list of maximal compatibles to be checked for closure.

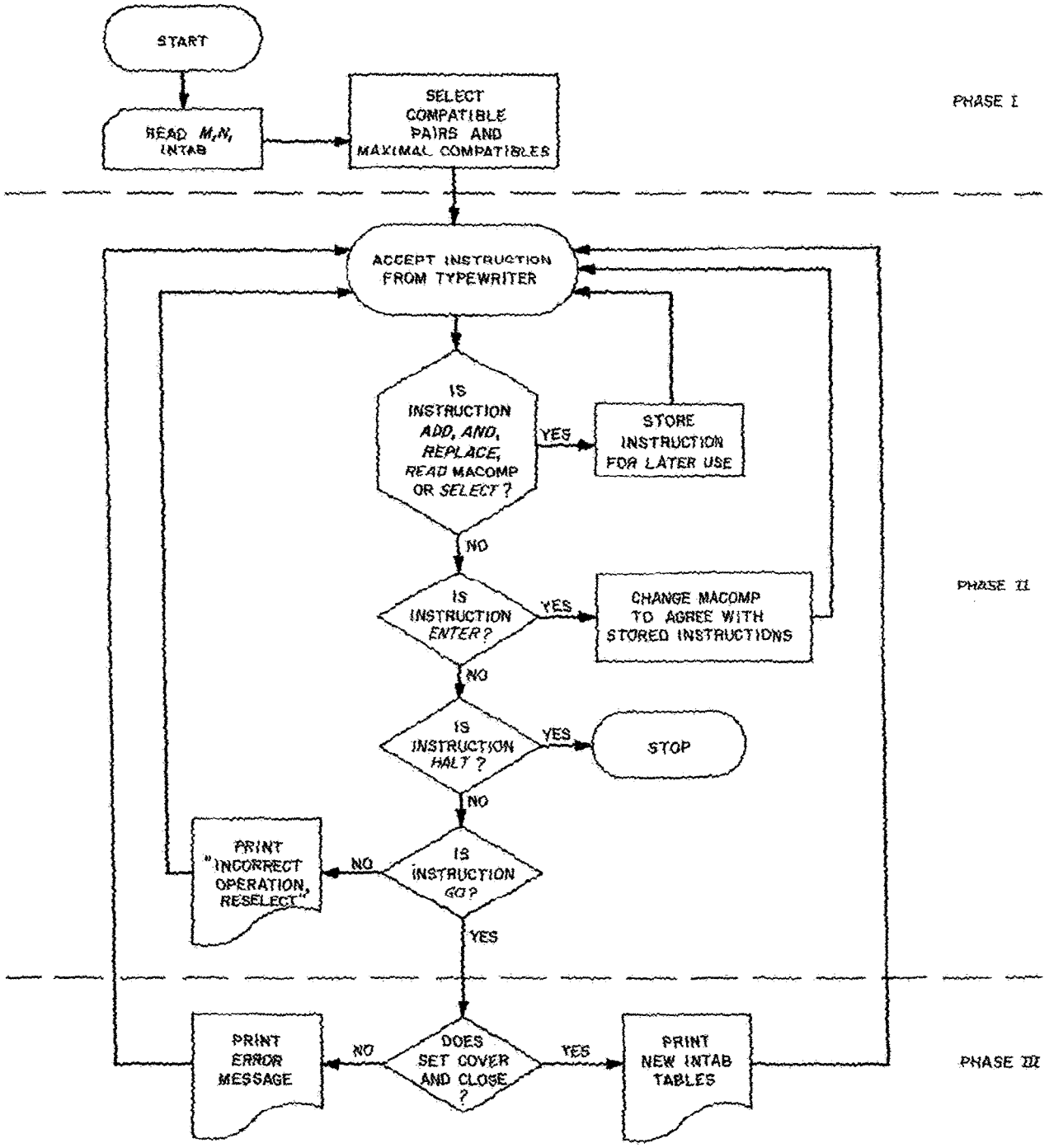


Fig. A-1. Gross flow chart

Table A-2. Input data for flow chart of Fig. A-1 (when using XEQSPAU1)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	
M, N				6				4																											Card 1
Row 1				4			—	1			—	1			—	1				3				0				2			—	1			Card 2
Row 2			—	1			—	1			—	1			—	1				5				0				3				0			Card 3
Row 3			—	1				1				3				1				6			—	1				4			—	1			Card 4
Row 4				1			—	1			—	1			—	1				3				0				5				1			Card 5
Row 5			—	1			—	1			—	1			—	1				6			—	1			—	1				0			Card 6
Row 6				2				1				4				1			—	1			—	1			—	1				0			Card 7
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	

Table A-3. Instruction options

User typewritten instruction	Remarks
‡*ADD* S1, S2, S3, S4 *R/S* ‡*AND* S1, S2, S3 · · · *R/S* Example: *ADD* 1, 3, 5, *R/S*	Add an entry to the list of maximal compatibles.
‡*REPLACE R*/S1, S2, S3, S4 *R/S*	Replace row *R* of the maximal compatible list with the entry S1, S2, S3, S4.
Example: *REPLACE* 3/1, 6, 2 *R/S*	Replace row 3 with the entry 1, 2, 6.
‡*SELECT* S1, S3, S6, S2 *R/S*	Select maximal compatible list rows S1, S2, S3, S6 to be checked for closure.
‡Example: *SELECT* 1, 5, 2, *R/S*	Sets 1, 2 & 5 are to be checked for cover and closure.
READ MACOMP	Read new MACOMP entries from cards, until terminator card (3123 in first field) is encountered.
ENTER R/S	Decode the preceding instructions and make the called for changes to maximal compatible list table.
GO R/S	Call the closure check routine. A set of rows from the maximal compatible list must have been selected prior to this instruction.
HALT R/S	Stop computation, return control to monitor.
If any other code word is typed in, this error message will be typed "INCORRECT OPERATION RESELECT."	

NOTE:

After the code word, any spacing combination may be used. Entries need not be in numerical order. Commas must separate all entries.

‡Up to 10 of the user instructions, each followed by depressing the *R/S* key, may be typed in at a time. The number of instructions is indicated by the index typed following *PROCEED* (see Fig. A-3). These instructions must be decoded and entered into the computer (by typing the word *ENTER*) before calling the closure check.

1. Input/Output Communication

(1) At the beginning of Phase II the typewriter will type "SELECT SETS or ADDITIONS" followed by a complete list of maximal compatibles with format as in Fig. A-3. The first column in this output list denotes which rows, if any, have been selected to be checked for closure. The * is followed by the list of states that make up the maximal compatibles.

(2) "*PROCEED* 1" is then typed. This signifies that the computer is ready to receive typed instructions.

```
                BEGIN PHASE I

INTAB

        1         2         3         4
  4   -1   -1   -1    3    0    2   -1
 -1   -1   -1   -1    5    0    3    0
 -1    1    3    1    6   -1    4   -1
  1   -1   -1   -1    3    0    5    1
 -1   -1   -1   -1    6    1   -1    0
  2    1    4    1   -1    1   -1    0

COMPATIBLE PAIRS
  3 ,   5
  5 ,   6
MAXIMAL COMPATIBLES

  1 /   1
  2 /   2
  3 /   4
  4 /   5   6
  5 /   3   5
MINIMUM SIZE
  5


                BEGIN PHASE II
MAXIMAL COMPATIBLES
  1 /   0 *   1
  2 /   0 *   2
  3 /   0 *   4
  4 /   0 *   5   6
  5 /   0 *   3   5
              INPUT   1 ,SELECT  1, 2, 3, 4, 5
              INPUT   2 ,ENTER
MAXIMAL COMPATIBLES
  1 /   1 *   1
  2 /   2 *   2
  3 /   3 *   4
  4 /   4 *   5   6
  5 /   5 *   3   5
              INPUT   1 ,GO


              PHASE III

NEW INTAB

        1    2    3    4
1 *     3   -1    5    2
2 *    -1   -1    4    5
0 *     0    0    5    0
3 *     1   -1    5    4
0 *     0    0    0    5
4 *     2    3    4   -1
5 *    -1    5    4    3

NEW TAB OUTPUT

        1    2    3    4
1 *    -1   -1    0   -1
2 *    -1   -1    0    0
3 *    -1   -1    0    1
4 *     1    1    1    0
5 *     1    1    1    0
```

Fig. A-2. Typical problem printout

(3) The user may type instructions from the list of options of Table A-3. The instruction code word must be spelled correctly, with no spaces either in front of or within the word. The numbers and punctuation may be spaced in any convenient manner.

Each time an *ADD*, *AND*, *SELECT*, *REPLACE*, or *READ MACOMP* instruction is typed, the computer will acknowledge acceptance by typing "*PROCEED N*" where the index *N* is a count of

```
‡‡XEQSPAU1
EXECUTION


                BEGIN PHASE I


                BEGIN PHASE II
SELECT SETS OR ADDITIONS
MAXIMAL COMPATIBLES
  1 /   0 *   1
  2 /   0 *   2
  3 /   0 *   4
  4 /   0 *   5   6
  5 /   0 *   3   5
PROCEED 1
SELECT 1, 2, 3, 4, 5 R/S
PROCEED 2
ENTER R/S


MAXIMAL COMPATIBLES
  1 /   1 *   1
  2 /   2 *   2
  3 /   3 *   4
  4 /   4 *   5   6
  5 /   5 *   3   5
PROCEED 1
GO R/S


                PHASE III

                BEGIN NEW PROBLEM, PHASE II


MAXIMAL COMPATIBLES
  1 /   1 *   1
  2 /   2 *   2
  3 /   3 *   4
  4 /   4 *   5   6
  5 /   5 *   3   5
PROCEED 1
ADD 1 R/S
PROCEED 2
HALT R/S
STOP
END OF JOB
```

Fig. A-3. Typical problem typewriter input/output

how many instructions have been typed. The five above listed instructions are stored on the disk to be decoded later.

(4) Use of the *ENTER* instruction. After the desired additions or selections are made, type the word *ENTER*. The desired modifications to or selections of maximal compatible entries will then be made.

A printout of a new maximal compatible list will result, indicating the selections or additions made.

C. Phase III

The typed instruction *GO* starts Phase III. The selected sets are checked for cover and closure. If the sets do not cover or close, an appropriate error message is typed and control is returned to Phase II.

At this time, a different set of maximal compatibles may be selected to try for an alternate solution, or the instruction *HALT* may be typed, which terminates the program. Control may then be returned to monitor by depressing the computer START switch.

If an error has been made in typing, it may be corrected by using the CORR key on a typewriter equipped with the SELECTRIC feature. If the typewriter is not a SELECTRIC, or if you are unsure how to use it, the entry may be deleted in the following manner: Set sense switch 2 to ON, depress R/S key on typewriter. When "PROCEED N" is typed, set sense switch 2 to OFF and enter correct information.

The primary output of Phase II are two tables making up the flow table of the simplified sequential machine. NEW INTAB lists the next state information and NEW INTAB OUTPUT lists the new machine's output information.

The numbers followed by an asterisk (e.g., 1*, 6*, etc.) are the internal states of the simplified machine and are made up of the original machine states specified in the maximal compatible list row of the same asterisk number. As an example, if the maximal compatible list row 3* contained the numbers 1, 3, 5, then the NEW INTAB row 3* would represent the combination of states 1, 3, and 5 of the original machine.

1. Reduced Machine Redundancies

In some cases a reduced machine internal state for a given input may have a choice of which internal state to cycle to next. The output of Phase III will print all choices. As an example, consider the following reduced machine output:

New INTAB

	1	2	3	4
1*	3	—1	5	2
2*	—1	—1	4	5
0*	0	0	5	0
3*	1	—1	5	4
0*	0	0	0	5
4*	2	3	4	—1
5*	—1	5	4	3

From the example above, internal state 2* with input 3 may cycle to either state 4* or 5*. Internal state 3* with input 4 may cycle to either state 4* or 5*.

2. Off Line Analysis

If a solution is not readily obtained, the program may be halted before successful completion of Phase III. The maximal compatible list may then be studied off line. At a later time, the program may be entered at the beginning of Phase II to try any different combinations which may have been found off line. This may be accomplished by setting SENSE SWITCH 1 to ON and calling the PAU4 with an XEQSPAU4 MONITOR control card. The communicator will then read values for M and N from a data card. (The first card from the Phase I input data deck may be used here.) The operator should then turn SENSE SWITCH 1 to OFF and generate either by typewriter or from cards a maximal compatible table that contains all the desired rows. The *READ MACOMP instruction is quite useful here, since it will cause as many* cards as are necessary to be read, each representing a maximal compatible. The data punched on these cards are simply as many four digit fields as are required to represent the contents of the desired maximal compatible row. The computer will continue to read in cards until a terminating card is encountered with the digits 3123 in the first field. The proper sets may then be selected. When *ENTER* is typed in, the MACOMP data is read and the desired sets are selected; *GO* may then be typed to call Phase III. At this time, the computer will read the INTAB data. (The 2nd through 9th cards of the Phase I input data deck may be used here. See Table A-2 for an example of the proper data format.)

III. ERROR MESSAGES

Phase I	
Printed Message	*Remarks*
Goof Stop.	A conditional transfer was made on improper data—either bad input data or program error.
There are no compatible pairs.	It is impossible to minimize flow table further. Program terminated.

Phase II	
Typed Message	*Remarks*
You have entered an incorrect operation, reselect.	An improperly spelled instruction has been typed in. Type desired instruction again—correctly.
Iteration not included, reselect.	An improper instruction has been typed in, same as above. Type desired instruction again—correctly.
Error deleted.	The previous instruction has been properly deleted using sense switch 2. Set sense switch 2 to OFF and retype instruction—correctly.

Phase III	
Printed Message	*Remarks*
Sets do not cover, reselect.	A set has been selected that does not include all states of the original machine. Select a new set that does.
Not closed, star No. $jI = i$.	Selected set does not close: j = the star number of the first MACOMP row to fail to check out; i = the column of (INTAB) that failed to check out. Select another set or modify MACOMP.

APPENDIX B

Program Flow Diagrams

Figures B-1 through B-18 are the flow diagrams for this program.

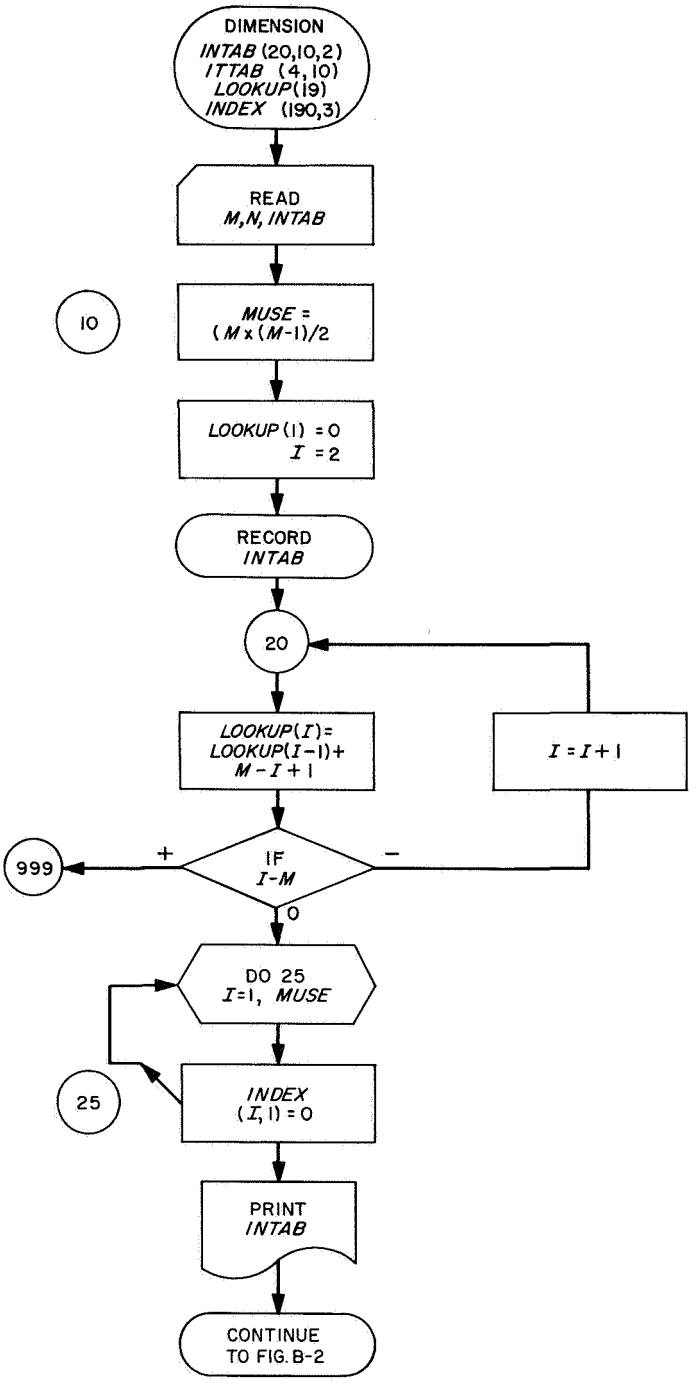


Fig. B-1. PAU1, Part 1

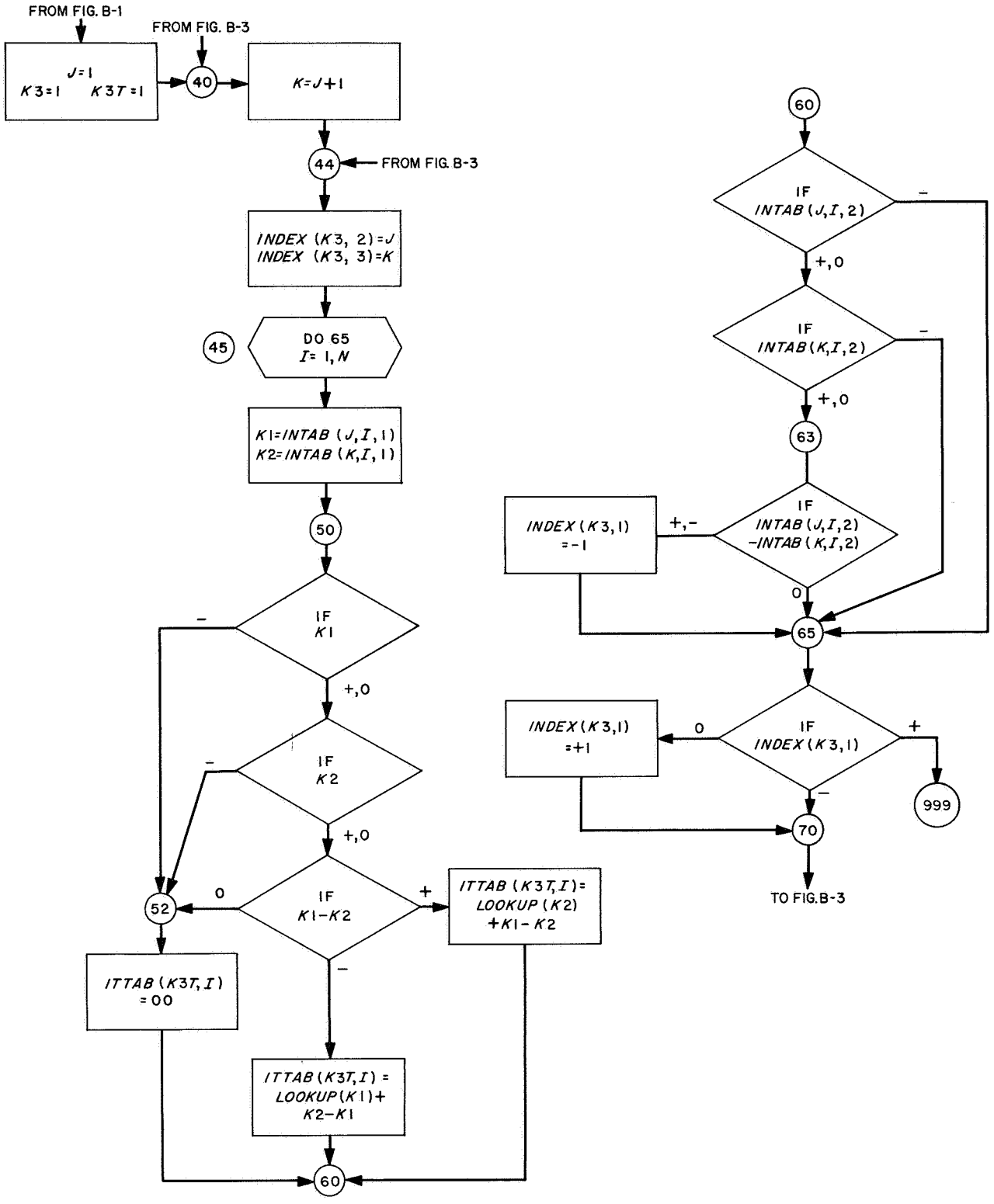


Fig. B-2. PAU1, Part 2

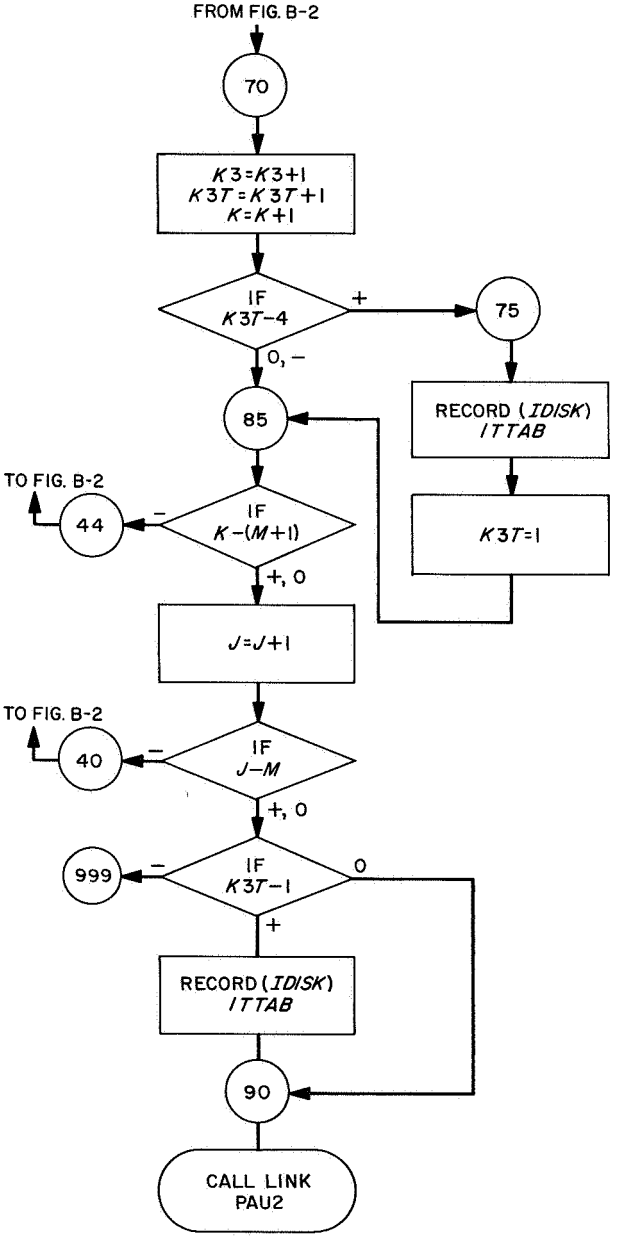


Fig. B-3. PAU1, Part 3

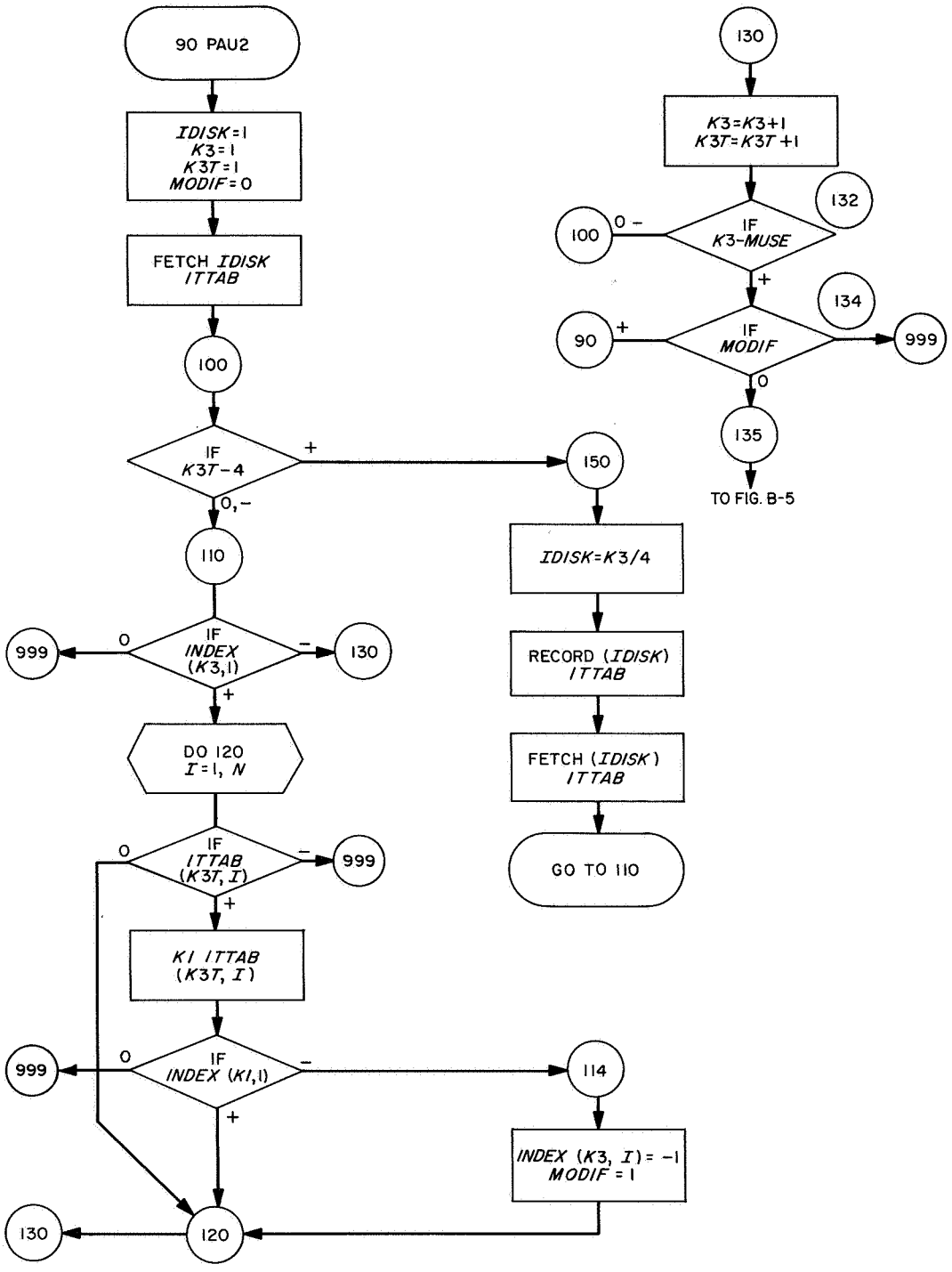


Fig. B-4. PAU2, Part 1

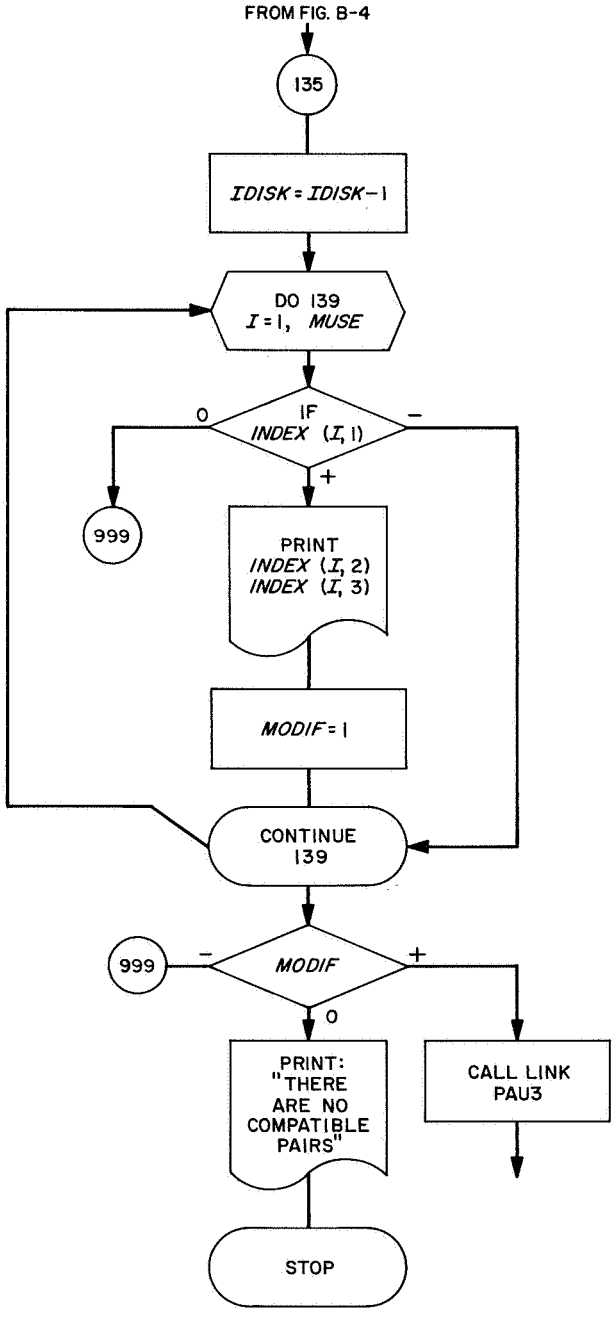


Fig. B-5. PAU2, Part 2

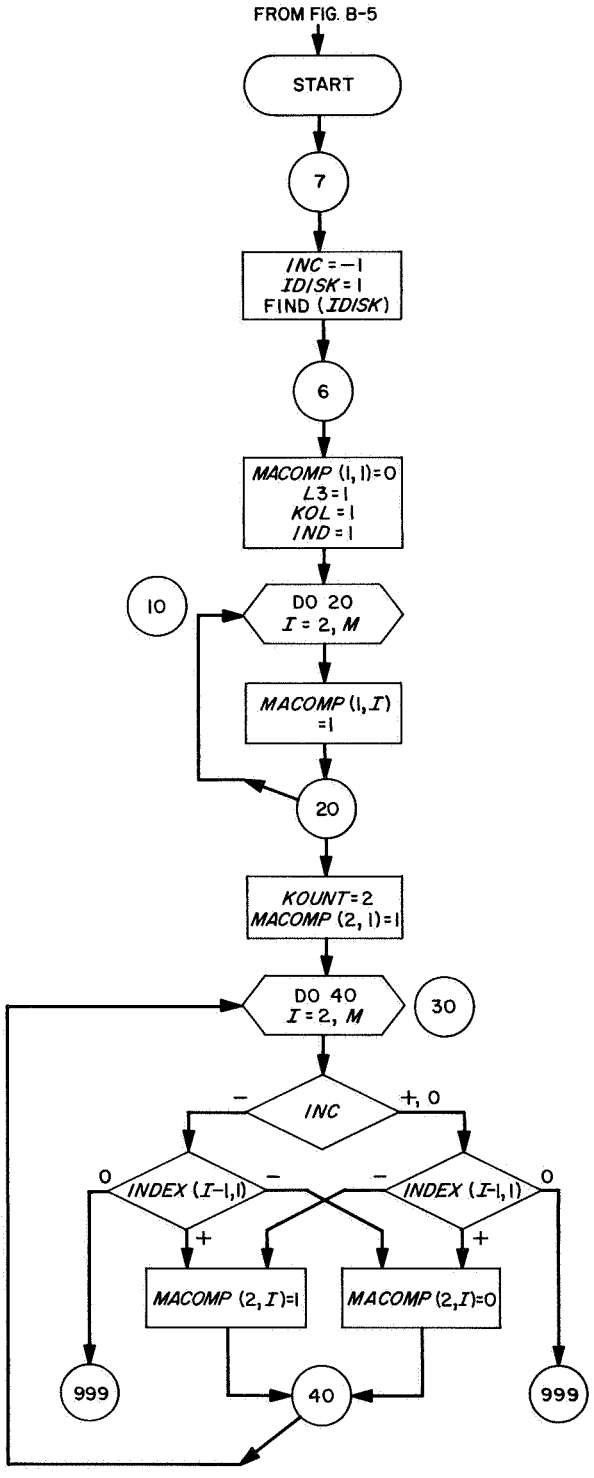


Fig. B-6. PAU3, Part 1

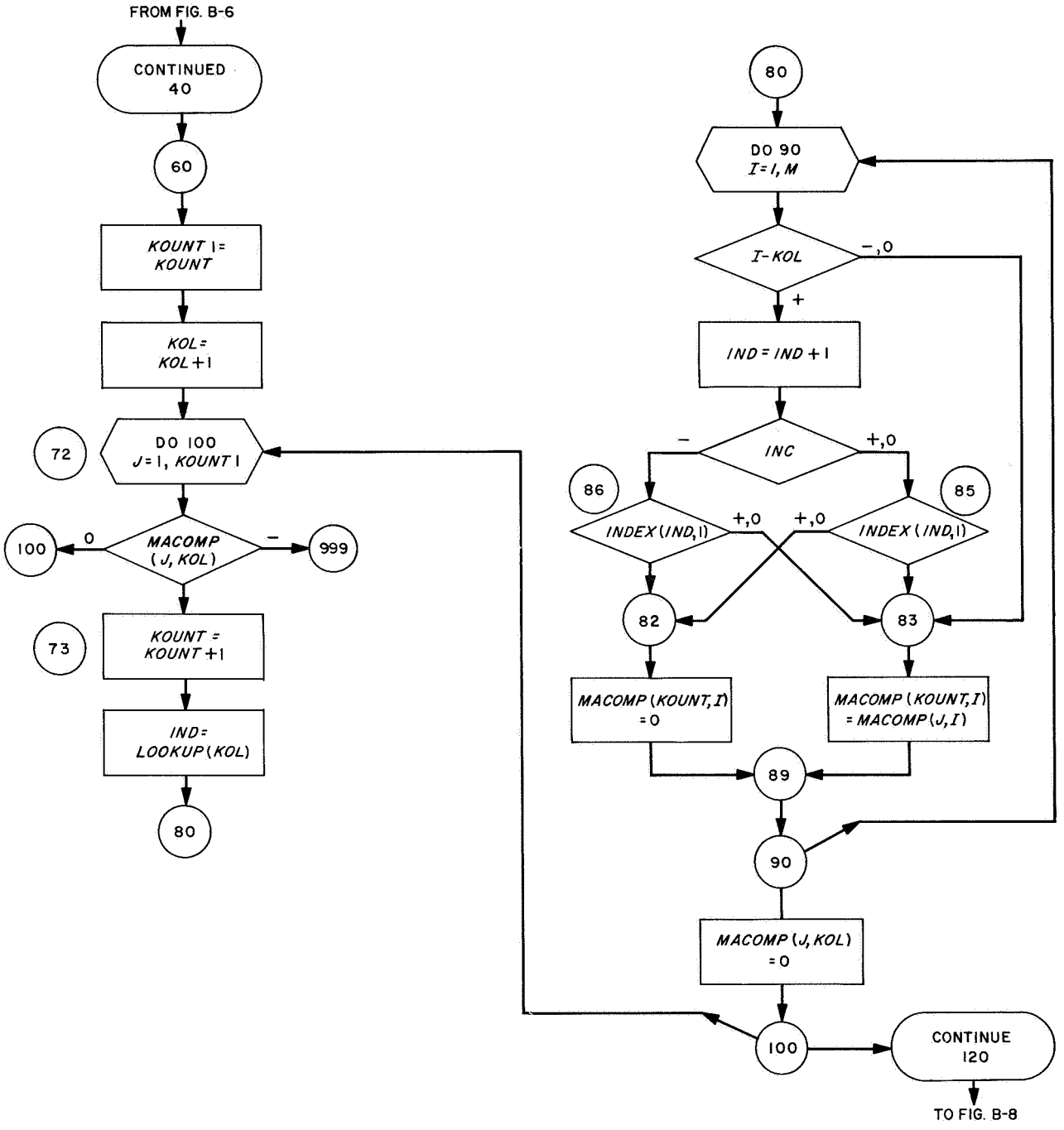


Fig. B-7. PAU3, Part 2

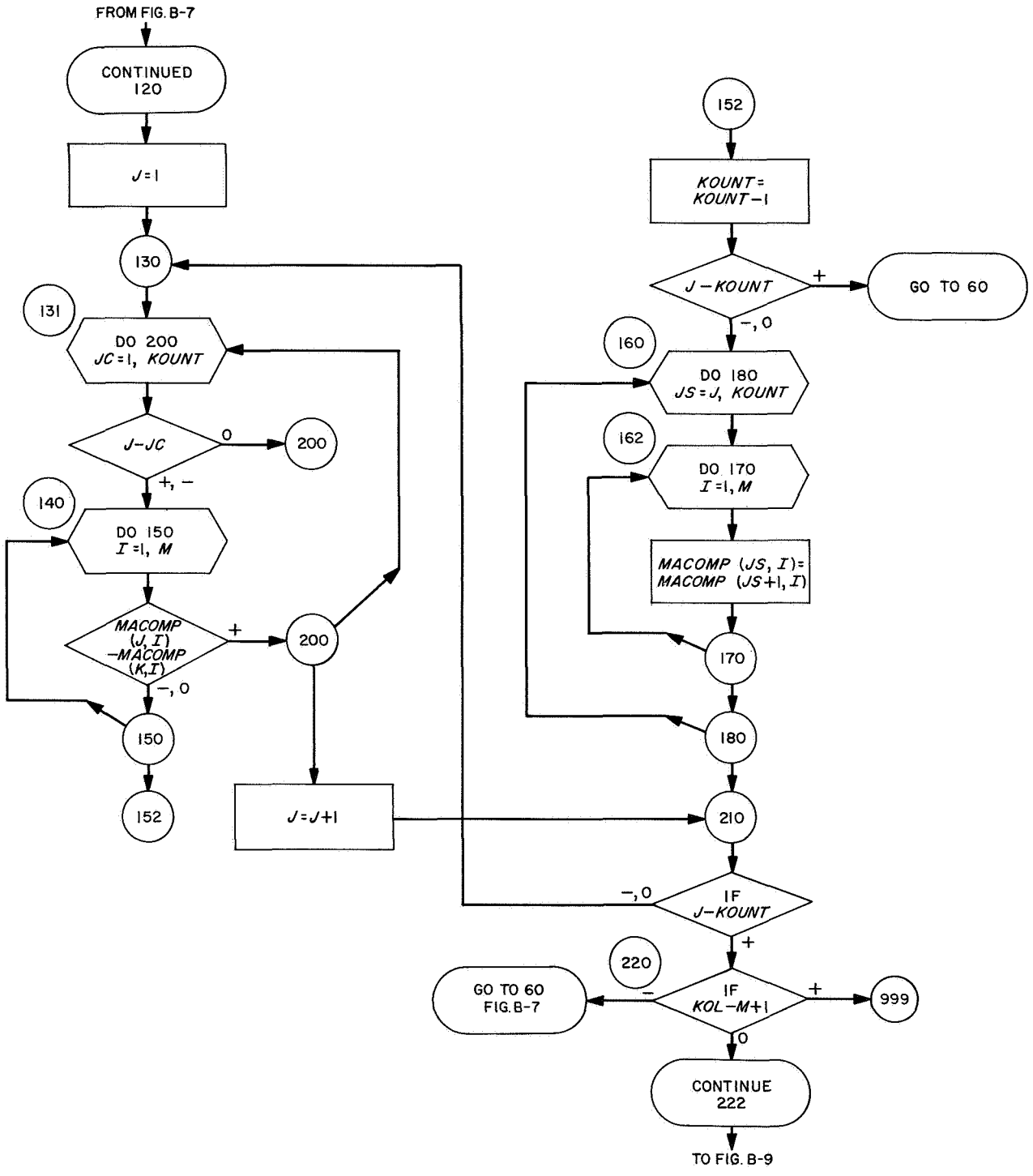


Fig. B-8. PAU3, Part 3

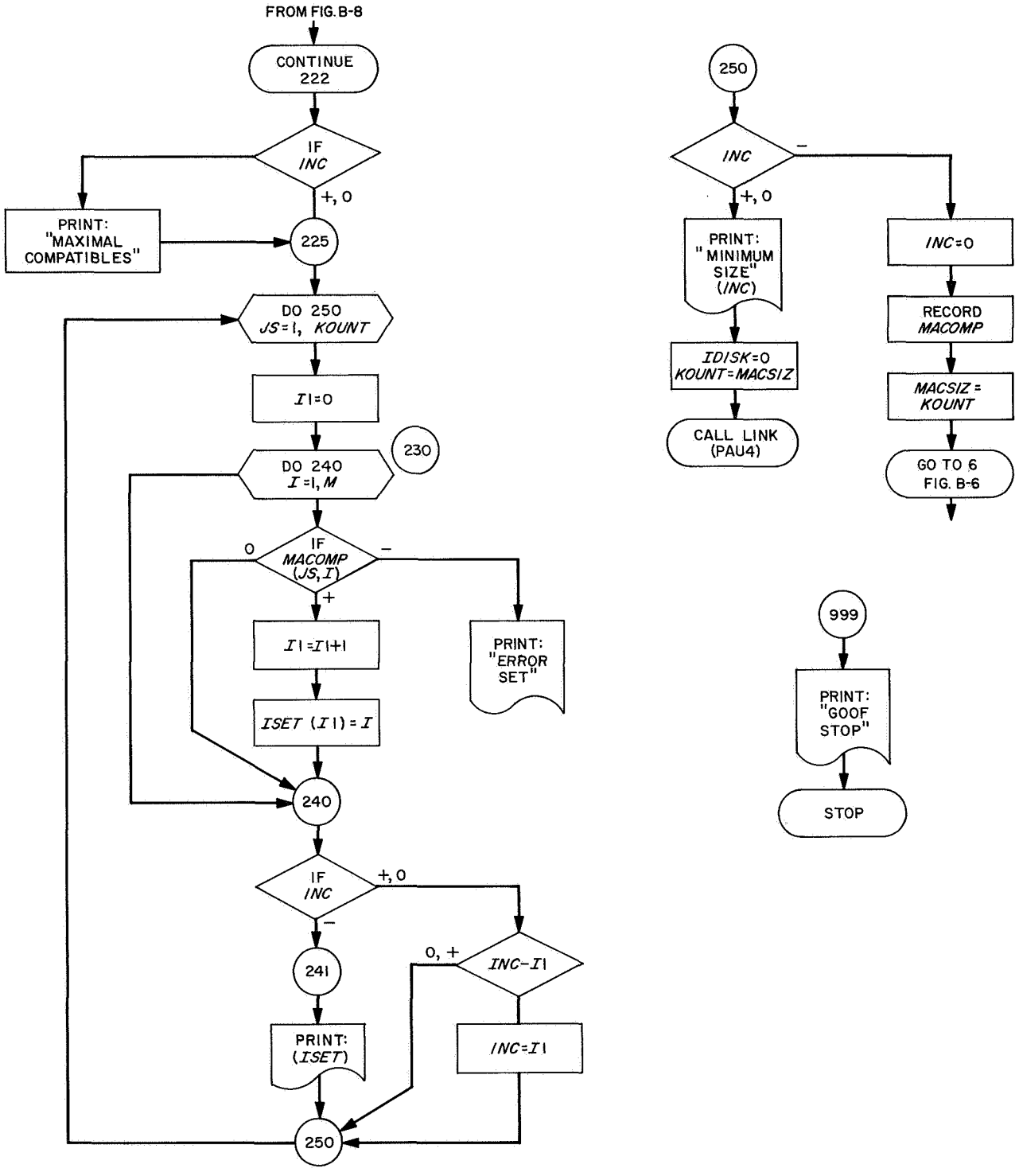


Fig. B-9. PAU3, Part 4

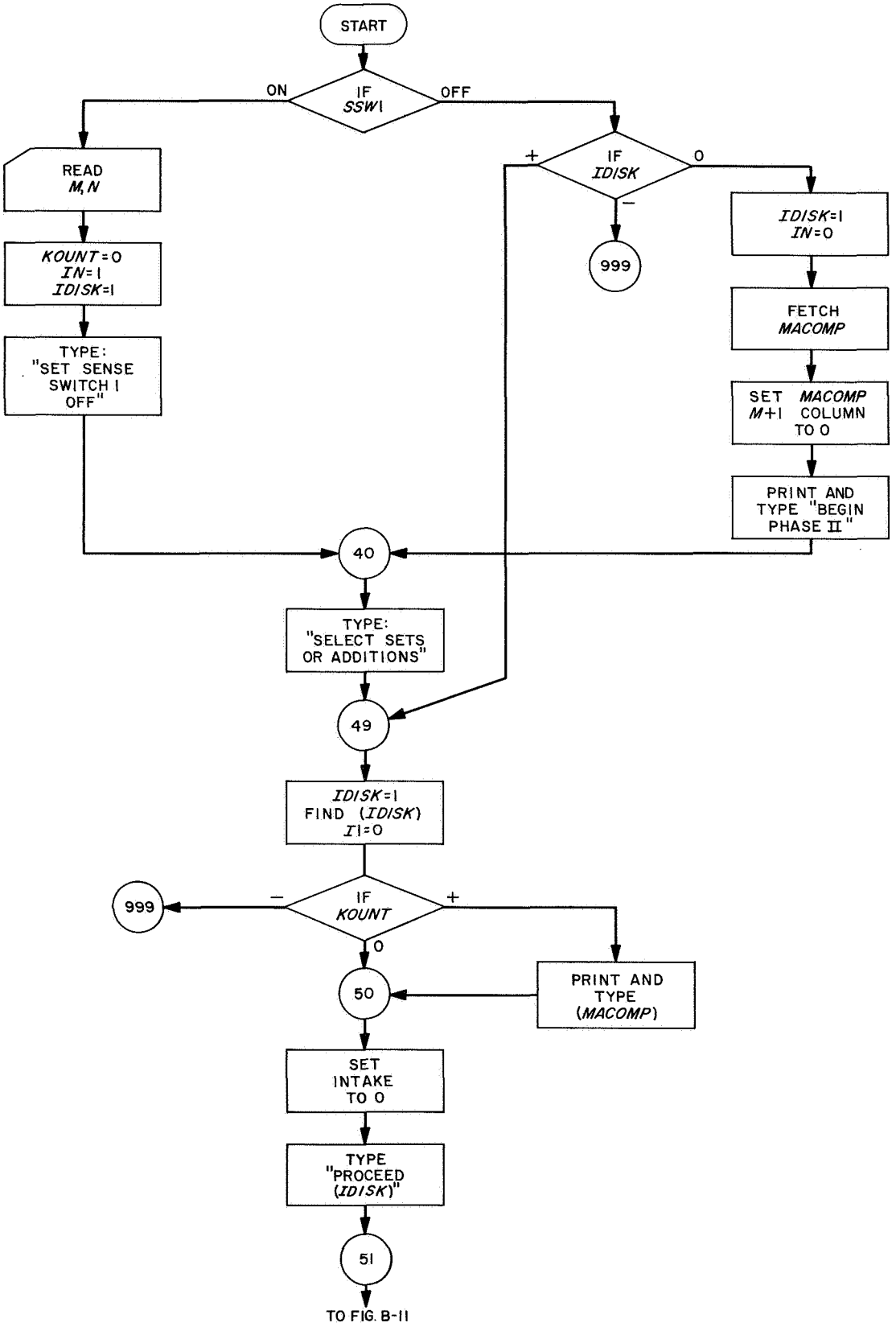


Fig. B-10. PAU4, Part 1

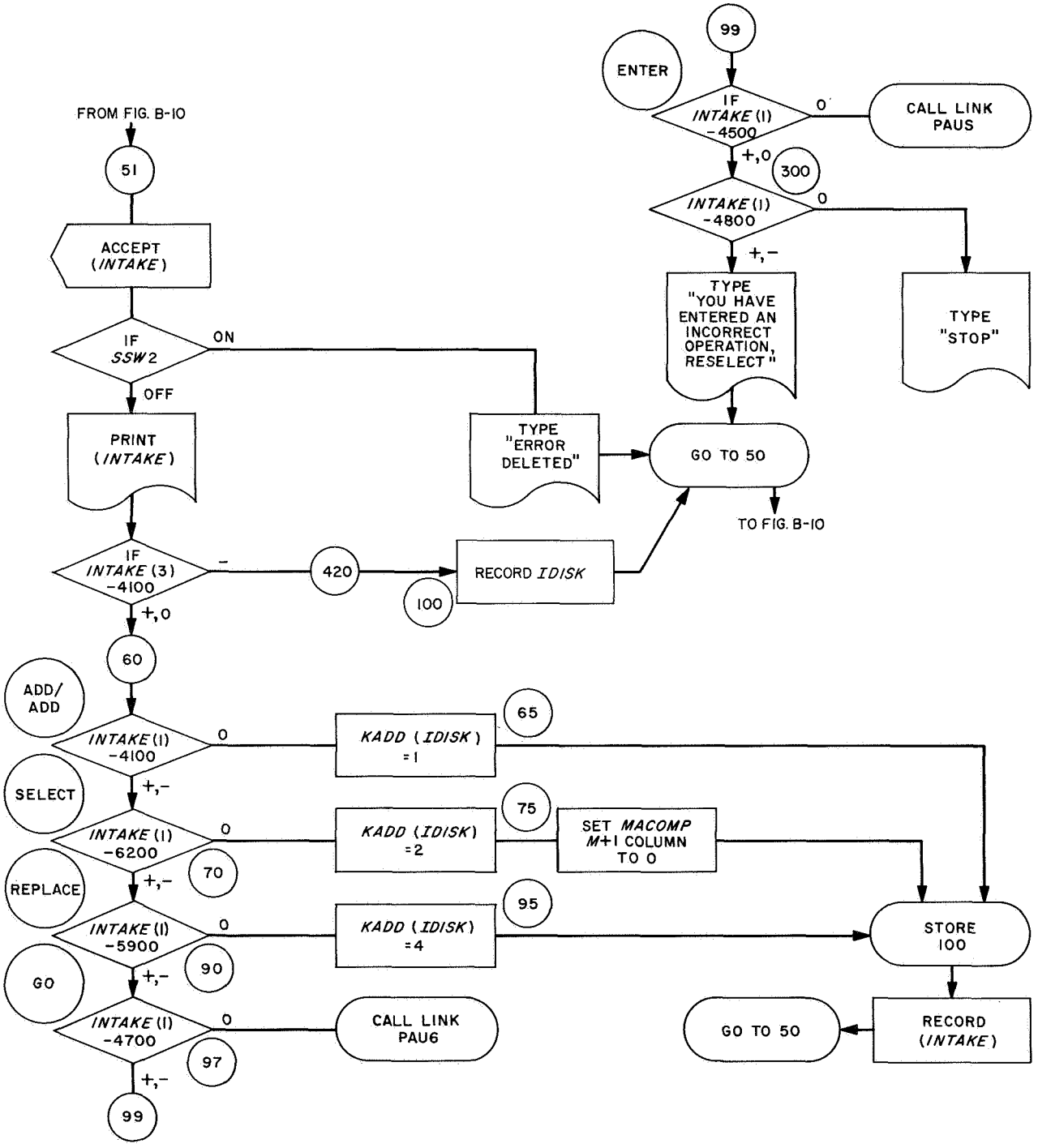


Fig. B-11. PAU4, Part 2

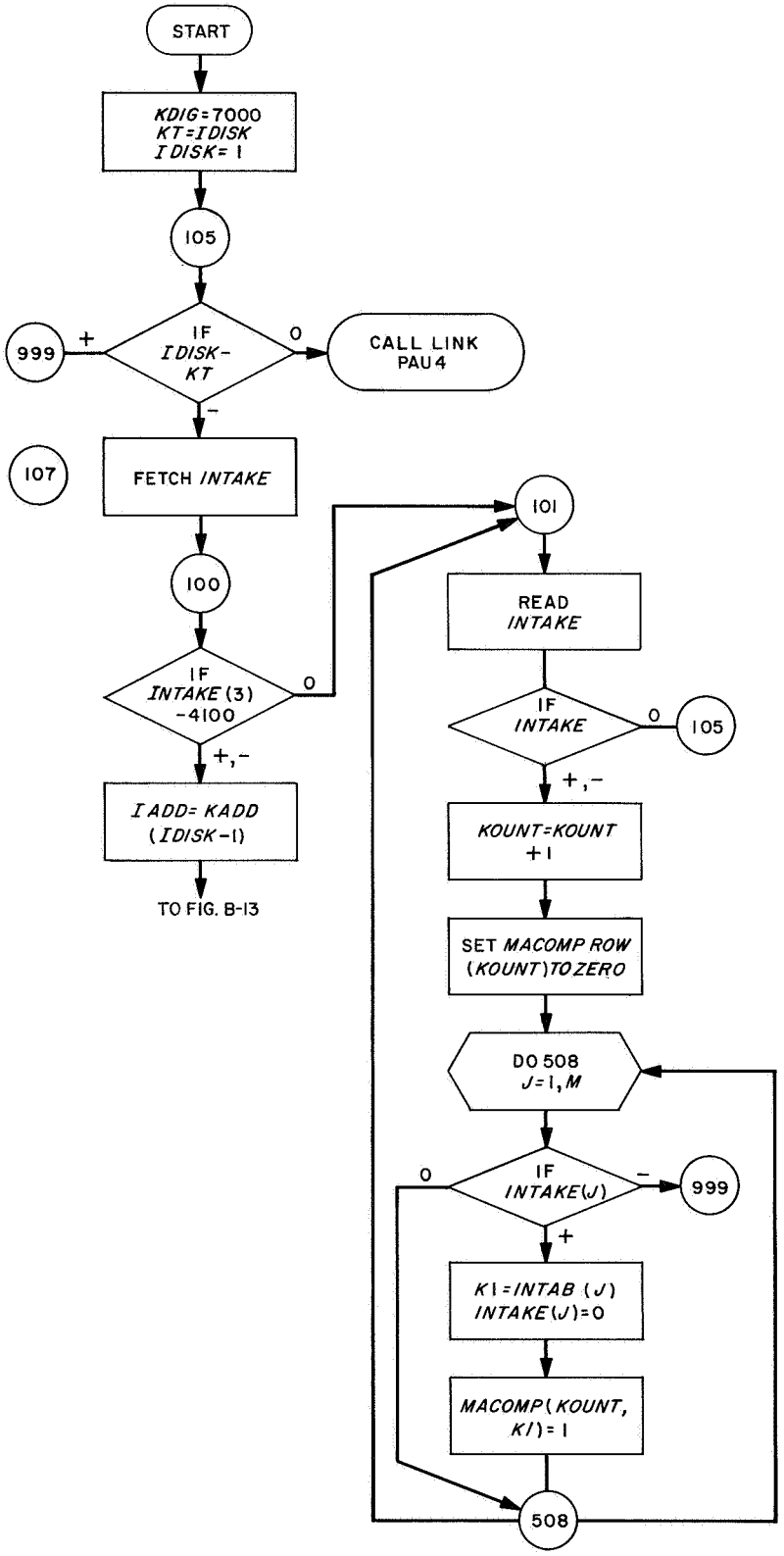


Fig. B-12. PAU5, Part 1

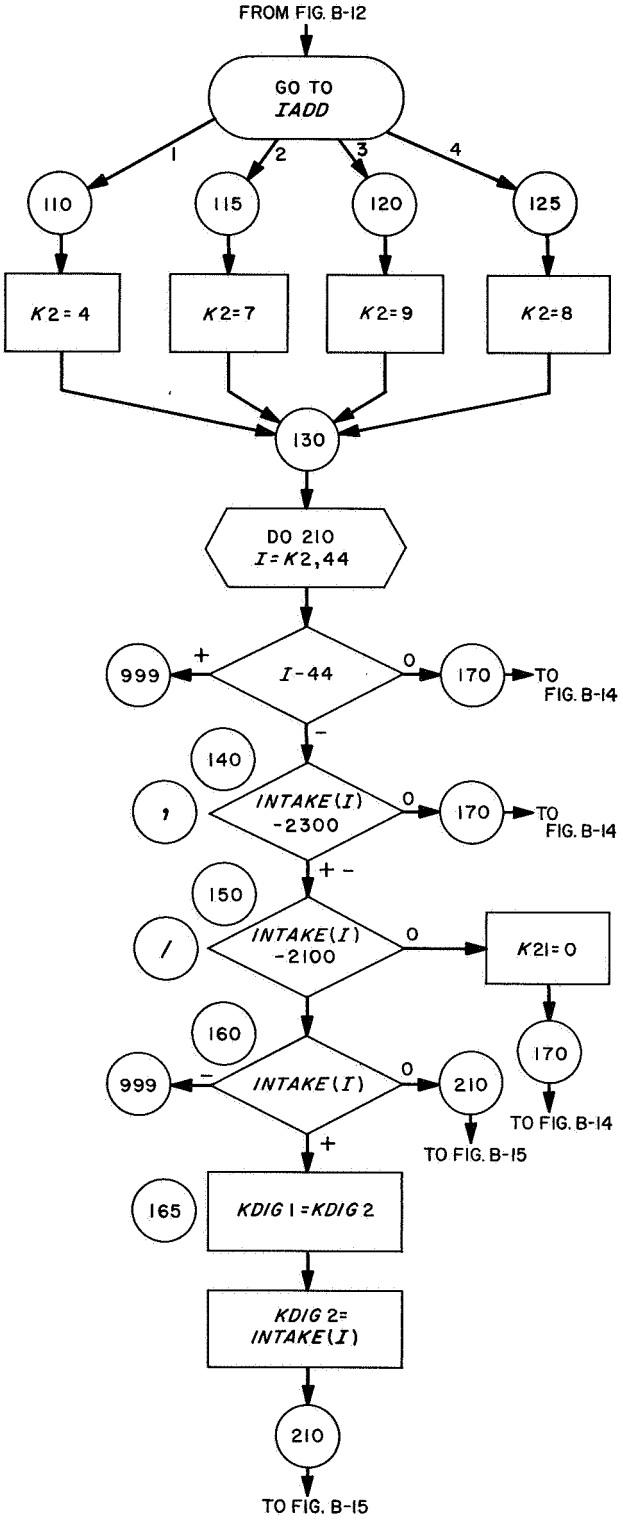


Fig. B-13. PAU5, Part 2

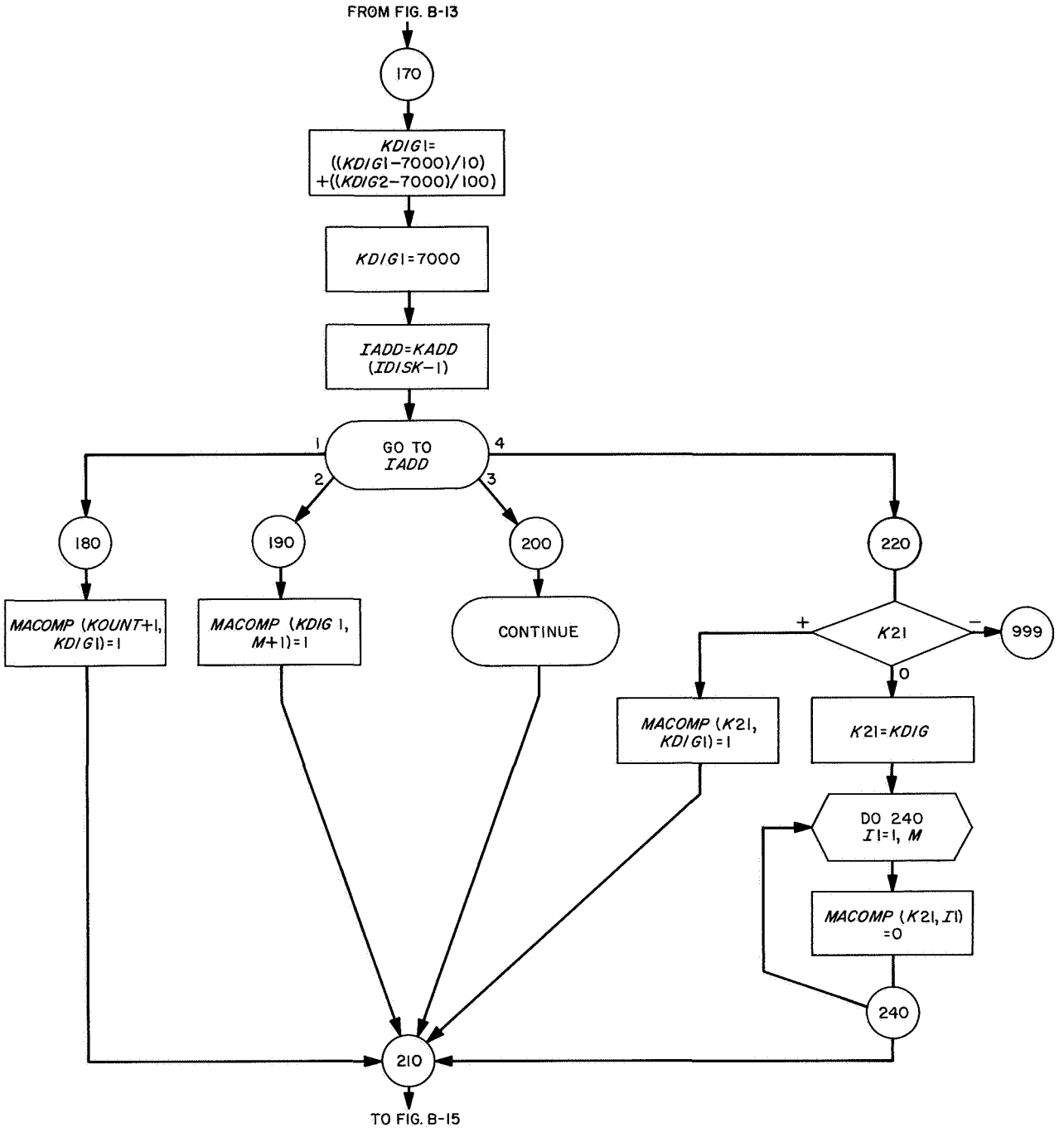


Fig. B-14. PAU5, Part 3

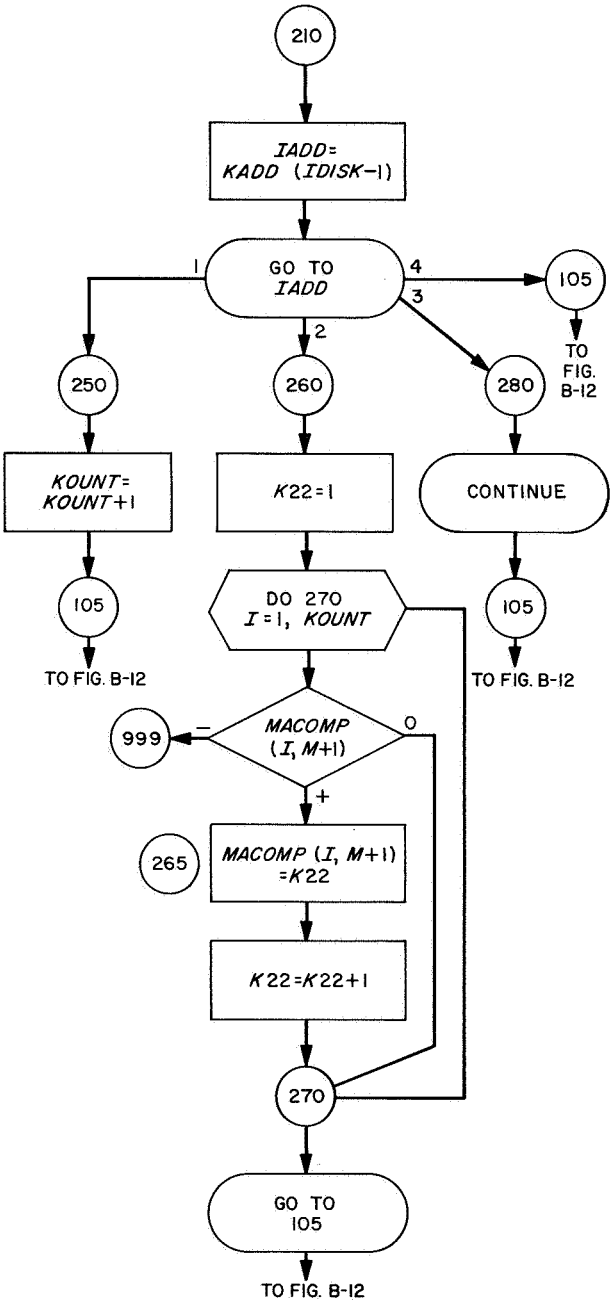


Fig. B-15. PAU5, Part 4

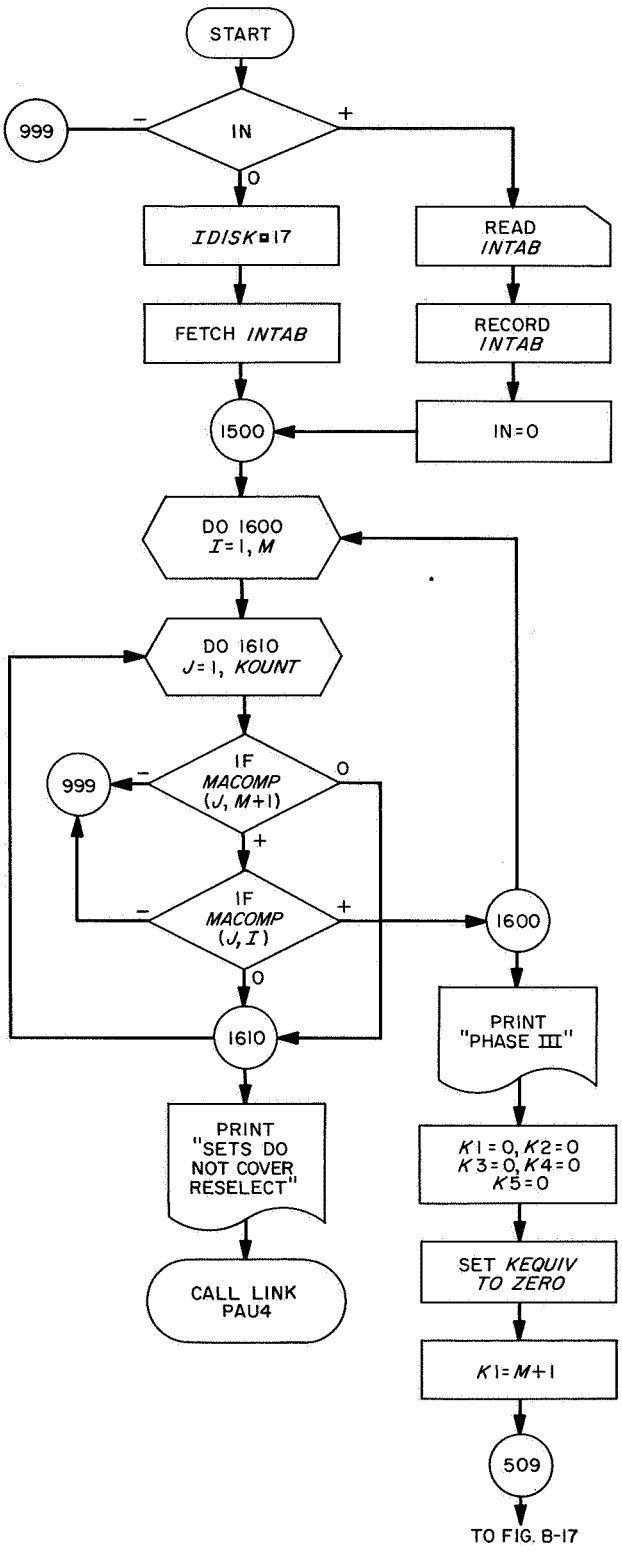


Fig. B-16. PAU6, Part 1

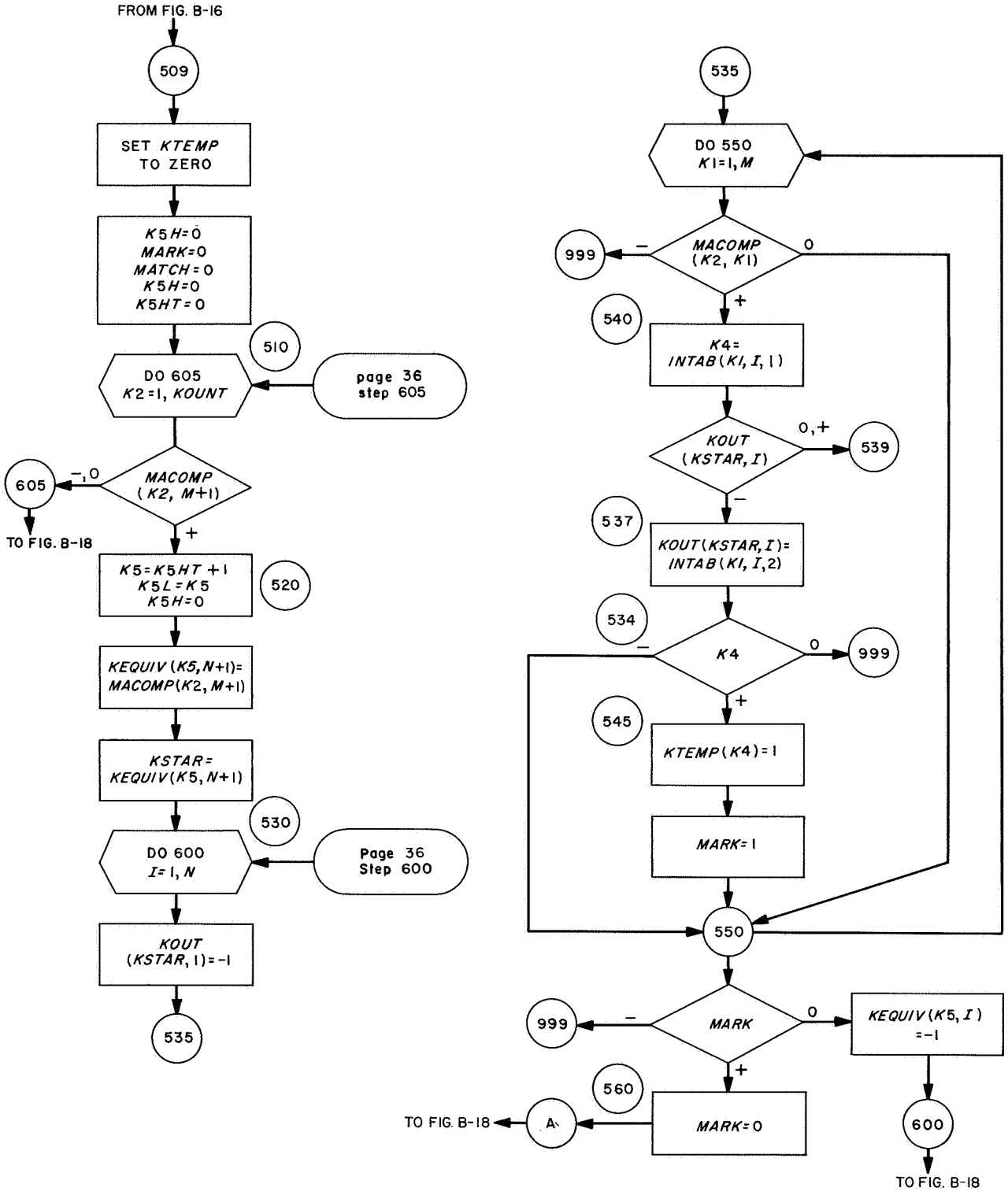


Fig. B-17. PAU6, Part 2

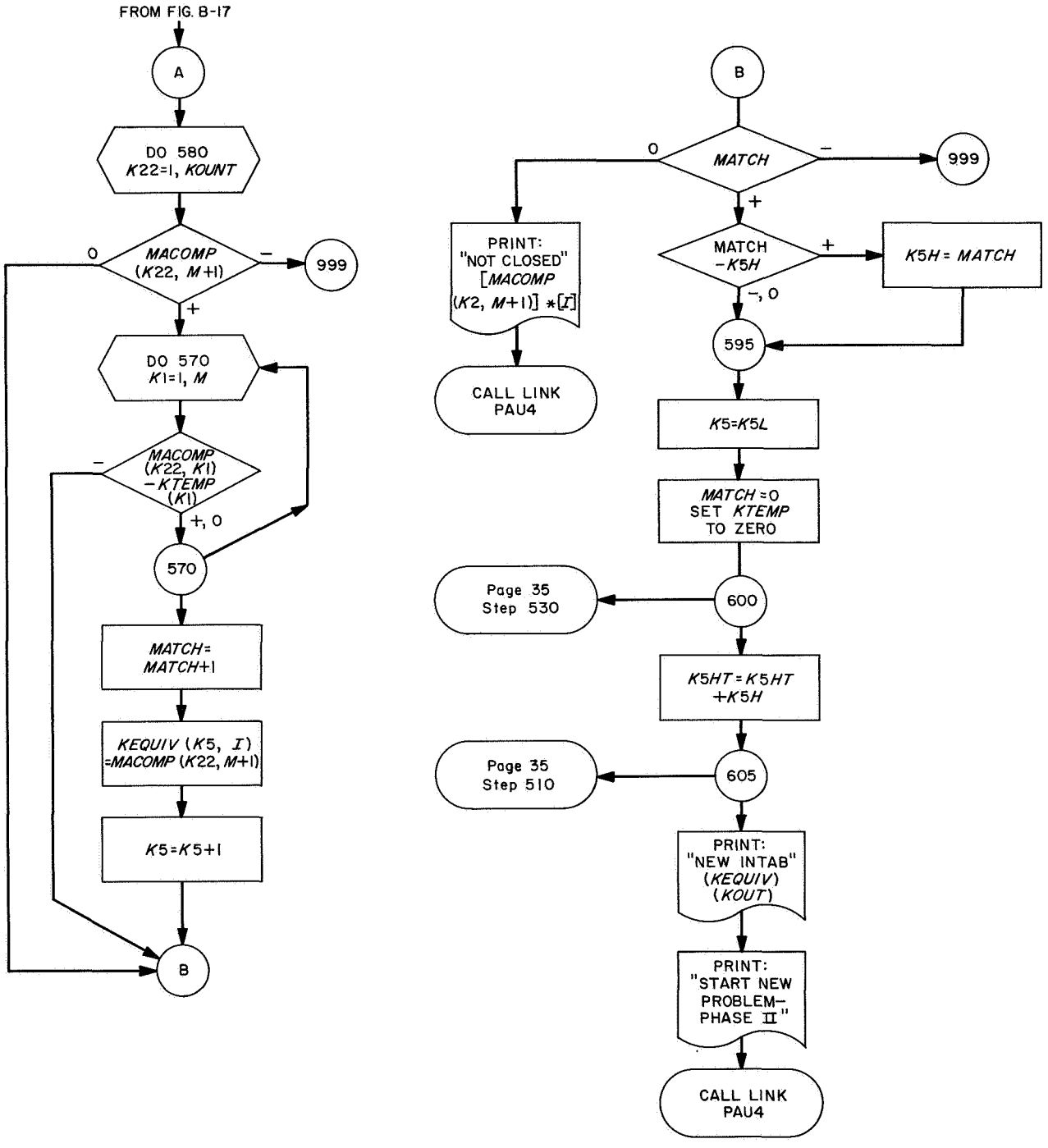


Fig. B-18. PAU6, Part 3

APPENDIX C

Program Listing

```
††FOR  2
*LDISKPAU1
*FANDK0204
      DIMENSION INTAB(20,10,2),ITTAB(4,10),LOOKUP(19),INDEX(190,3)
      COMMON M,N,IDISK,JUMP,LOOKUP,INDEX,NUMBER,K3,K3T,K1,K2,MUSE,ITTAB
        DEFINE DISK   (40,26)
  807  FORMAT  (10I4)
  805  FORMAT  (//10X, 14H BEGIN PHASE I/)
       PRINT  805
       TYPE 805
       READ 1, M,N
    1  FORMAT (20I4)
        DO 5 I=1,M
    5 READ 1, ((  INTAB(I,J,K),  K=1,2),  J= 1,N)
       IDISK = 17
       FIND (IDISK)
   10  MUSE = (M*(M-1))/2
         DO 11 I=1,19
   11    LOOKUP(I) = 0000
       RECORD   (IDISK) (((INTAB(I,J,K), K=1,2), J=1,N), I=1,M)
           IDISK = 1
       FIND (IDISK)
         I=0002
   20  LOOKUP(I)= LOOKUP(I-1) +M-I+1
       IF (I-M) 21,22,999
   21    I= I+0001
       GO TO  20
   22  DO  25  I=1,MUSE
   25 INDEX(I,1) = 0
       PRINT 2
    2  FORMAT (6H INTAB/)
       PRINT 4, (I, I=1,N)
    4  FORMAT (32(4X,I4) )
        DO 15  I=1,M
   15  PRINT 3,  ((INTAB(I,J,K),  K=1,2),  J=1,N)
    3  FORMAT  (20I4)
        J=1
       K3=1
       K3T=1
   40 K=J+1
C      FILL IN ITTAB
   44  INDEX(K3,2) = J
       INDEX(K3,3) =K
  801 FORMAT(3H J=, I4, 3H K=, I4)
  800 FORMAT (12H STEP NUMBER, I4)
   45 DO 65 I=1,N
          K1=   INTAB(J,I,1)
         K2=  INTAB(K,I,1)
C       SET POINTERS IN ROW K3
   50 CONTINUE
   56   IF (K1)   52,51,51
   51 IF (K2) 52,53,53
   52 ITTAB(K3T,I)=0
       GO TO 60
   53   IF (K1-K2)   55,52,54
   54    ITTAB(K3T,I)=  LOOKUP(K2)  +  K1  -  K2
         GO TO  60
   55 ITTAB(K3T,I) = LOOKUP(K1) + K2 -K1
C      CHECK OUTPUTS AND SET INDEX ACCORDING TO OUTPUT AGREEMENT
```

```
   60  CONTINUE
       IF (INTAB(J,I,2))       65,62,62
   62 IF (INTAB(K,I,2)) 65,63,63
   63 IF(INTAB(J,I,2)  -  INTAB(K,I,2)) 64,65,64
   64 INDEX(K3,1)  =(-1)
   65  CONTINUE
       IF(INDEX(K3,1)) 70,66,999
   66 INDEX(K3,1) =(+1)
C     IF NECES. RECORD ITTAB ON DISK AND START NEW ITTAB TABLE
   70   CONTINUE
      K3 = K3+1
       K3T = K3T + 1
        K=K+1
       IF (K3T-4)  85,85,75
   75 RECORD (IDISK)   ((ITTAB(K1,K2), K1=1,4), K2=1,N)
         JUMP =1
  850 NUMBER = IDISK - 1
       IF (SENSE SWITCH 3) 852,76
  852  PRINT 806, NUMBER
          DO 77  L1=1,4
   77  PRINT 807,  (ITTAB(L1,L2),  L2= 1,N)
  806 FORMAT (14H ITTAB,RECORD,I4//)
   76  K3T=1
   85 IF( K-(M+1))  44,86,86
   86 J=J+1
  804 FORMAT (3H J=, I4)
       IF (J-M) 40,87,87
   87  IF (K3T-1) 999,90,88
   88 RECORD (IDISK)   ((ITTAB(K1,K2), K1=1,4), K2=1,N)
   90    CALL LINK (PAU2)
  999   PRINT 998
  998     FORMAT (10H GOOF STOP)
         STOP
         END
††††
```

```
††FOR 5
*LDISKPAU2
*FANDK0204
        DEFINE DISK   (40,26)
      DIMENSION                ITTAB(4,10),LOOKUP(19),INDEX(190,3)
      COMMON M,N,IDISK,JUMP,LOOKUP,INDEX,NUMBER,K3,K3T,K1,K2,MUSE,ITTAB
C     ITTERATION OF ITTAB
   90 IDISK = 1
  800 FORMAT (12H STEP NUMBER, I4)
  802  FORMAT (14H ITTAB(K3T,I)=, I4,4H K3=,I4,5H K3T=,I4,3H I=,2I4)
  803   FORMAT (13H INDEX(K3,I)=,  3(I4,2X), 4H K3=, I4)
       NUMBER=090
        K3=1
  806 FORMAT (14H ITTAB,RECORD,I4//)
        K3T=1
        MODIF =0
      FETCH  (IDISK)   ((ITTAB(K1,K2), K1=1,4), K2=1,N)
         JUMP = 2
       GO TO 850
  100  CONTINUE
      IF( K3T-4)  110,110,150
  110 IF (INDEX(K3,1))  130,999,112
  112 DO 120 I=1,N
       IF (ITTAB(K3T,I))      999,120,113
  113     K1= ITTAB(K3T,I)
       IF (INDEX(K1,1))  114,999,120
  114 INDEX(K3,1)  = (-1)
         MODIF = 1
  120 CONTINUE
  130  CONTINUE
  810 FORMAT (13H INDEX(K3,1)=, I4, 4H K3=, I4)
      K3= K3+1
       K3T= K3T+1
  132 IF (K3-MUSE) 100,100,134
  134 IF (MODIF) 999,135,90
  135 IDISK = IDISK - 1
          JUMP = 3
        GO TO 850
  136 RECORD (IDISK)   ((ITTAB(K1,K2), K1=1,4), K2=1,N)
    5  FORMAT(/17H COMPATIBLE PAIRS)
    6 FORMAT (I4,2H ,, I4)
        MODIF = 0
        PRINT 5
         DO 139 I=1,MUSE
       IF (         INDEX(I,1)) 139,999,137
  137 PRINT 6, INDEX(I,2), INDEX(I,3)
        MODIF = 1
  139  CONTINUE
  141 IF (MODIF)  999,142,145
  142  PRINT 143
  143  FORMAT (30H THERE ARE NO COMPATABLE PAIRS)
  145  CALL LINK (PAU3)
C   FETCH AND RECORD IF NECESARY
  150 IDISK = K3/4
  151 RECORD (IDISK)   ((ITTAB(K1,K2), K1=1,4), K2=1,N)
      FETCH  (IDISK)   ((ITTAB(K1,K2), K1=1,4), K2=1,N)
        K3T=1
         JUMP = 4
```

```
      GO TO  850
850 NUMBER = IDISK - 1
     IF (SENSE SWITCH 3)  852,808
852  PRINT 806, NUMBER
        DO 855  L1=  1,4
855  PRINT 807,  (ITTAB(L1,L2), L2=1,N)
807   FORMAT (10I4)
808   GO TO(999,100,136,110), JUMP
999   PRINT 998
998     FORMAT (10H GOOF STOP)
        STOP
      END
††††
```

```
††FOR 5
*LDISKPAU3
*FANDK0204
        DEFINE DISK    (40,26)
       DIMENSION MACOMP(20,11),ISET(50)
      DIMENSION                                  LOOKUP(19),INDEX(190,3)
       COMMON M,N,IDISK,KOUNT,LOOKUP,INDEX
    1  FORMAT  (20I4)
  811  FORMAT  (/16H MACOMP FOR KOL=,  I4/ 20I4)
  812  FORMAT  (6H ERASE,  2I4)
       IDISK = 1
       FIND (IDISK)
    7  INC = -1
    6  MACOMP(1,1) = 0
       K3=1
       KOL=1
       IND=1
   10  DO 20 I=2,M
       MACOMP(1,I)=1
   20  CONTINUE
       KOUNT = 2
       MACOMP(2,1)=1
   30  DO 40 I=2,M
       IF (INC) 35,37,37
   37  IF (INDEX(I-1,1)) 31,999,32
   35  IF (INDEX(I-1,1)) 32,999,31
   31  MACOMP(2,I) =1
       GO TO 40
   32  MACOMP(2,I) =0
   40  CONTINUE
       IF (SENSE SWITCH 3) 42,60
   42  DO 41 I=1,2
   41  PRINT 1, (MACOMP(I,J), J=1,M)
   60  CONTINUE
       KOUNT1= KOUNT
       KOL = KOL + 1
   72  DO 100 J=1,KOUNT1
       IF (MACOMP(J,KOL))  999,100,73
   73  KOUNT = KOUNT+ 1
       IND = LOOKUP(KOL)
   80  DO 90 I= 1,M
       IF (I-KOL) 83,83,81
   81  IND = IND + 1
       IF (INC) 86,85,85
   85  IF (INDEX(IND,1))  83,82,82
   86  IF (INDEX(IND,1)) 82,83,83
   82  MACOMP(KOUNT,I) = 0
       GO TO 89
   83  MACOMP(KOUNT,I) = MACOMP(J,I)
   89  CONTINUE
   90  CONTINUE
       MACOMP(J,KOL) = 0
  100  CONTINUE
  120  CONTINUE
       J=1
  130  CONTINUE
  131  DO 200 JC=1,KOUNT
       IF (J-JC) 140,200,140
```

```
 140  DO 150 I=1,M
      IF (MACOMP(J,I)-MACOMP(JC,I))  150,150,200
 150  CONTINUE
      GO TO 152
 200  CONTINUE
      J= J+1
      GO TO 210
 152  KOUNT = KOUNT - 1
      IF (SENSE SWITCH 3)  153,154
 153 PRINT 812, KOL,J
 154  IF (J-KOUNT) 160,160,60
 160  DO 180  JS=J,KOUNT
 162  DO 170 I=1,M
      MACOMP(JS,I) = MACOMP(JS+1,I)
 170  CONTINUE
 180  CONTINUE
 210  CONTINUE
      IF (J-KOUNT)  130,130,220
 220  IF (KOL - M + 1) 60,222,999
 222  IF (INC)  224,225,225
 224  PRINT 242
 242  FORMAT (20H MAXIMAL COMPATIBLES/)
 225  DO 250 JS = 1,KOUNT
      I1=0
 230  DO 240 I=1,M
      IF (MACOMP(JS,I)) 232,240,239
 232  PRINT 233,JS
 233 FORMAT  (11H ERROR SET,, I4)
      GO TO 250
 239  I1 = I1 + 1
 235  ISET(I1) =   I
 240  CONTINUE
      IF (INC) 241,243,243
 243  IF (INC-I1)  245,250,250
 245  INC = I1
      GO TO 250
 241  PRINT 2,JS,(ISET(I),  I=1,I1)
   2  FORMAT  (I4,2H /,20I4)
 250  CONTINUE
      IF (INC)  255,260,260
 255  INC = 0
      RECORD   (IDISK)  ((MACOMP(I,J),J=1,M), I=1,KOUNT)
      MACSIZ  =  KOUNT
      GO TO 6
 260  PRINT 244, INC
      IDISK = 0
      KOUNT = MACSIZ
      CALL LINK (PAU4)
 999  PRINT 998
 998  FORMAT  (10H GOOF STOP)
      STOP
 244 FORMAT (13H MINIMUM SIZE/I4//)
      END
††††
```

```
‡‡FOR 5
*LDISKPAU4
*FANDK0204
        DEFINE DISK    (40,26)
       DIMENSION INTAKE(43) ,MACOMP(20,11),KADD(10)
C      PAU COMMUNICATOR
       COMMON M,N,IDISK,KOUNT,MACOMP,IN,KADD
    1 FORMAT (20I4)
  850  FORMAT (25H SELECT SETS OR ADDITIONS)
  851  FORMAT  ( 8H PROCEED,I4)
  852  FORMAT (50H YOU HAVE ENTERED AN INCORRECT OPERATION, RESELECT)
  853  FORMAT (26H SET SENSE SWITCH 1 TO OFF)
  856  FORMAT (//10X, 15H BEGIN PHASE II)
  859  FORMAT (43A1)
  860  FORMAT (13H CALL CLOSURE)
  862  FORMAT  (11H NEW ROW NO,  I4/10I4)
  863  FORMAT (10X,  14H ERROR DELETED)
  864  FORMAT (20H MAXIMAL COMPATIBLES)
  865  FORMAT   (I4,2H /,  I4, 2H *,  20I4)
  866  FORMAT (10X,6H INPUT,I4,2H ,, 43A1)
       IF (SENSE SWITCH 1)  20,35
   20  READ 1, M,N
       KOUNT = 0
       IN = 1
       IDISK = 1
       TYPE 853
       GO TO 40
   35  IF (IDISK)  999,36,49
   36  IDISK=1
       IN = 0
       FETCH    (IDISK)  ((MACOMP(I,J),J=1,M), I=1,KOUNT)
       DO 37  I=1,20
   37  MACOMP(I,M+1)  = 0
       TYPE 856
       PRINT 856
   40  TYPE 850
   49  IDISK = 1
       FIND (IDISK)
       I1=0
       IF (KOUNT)  999,57,52
   52  PRINT 864
       TYPE 864
       DO 55 I=1,KOUNT
       DO 54 K1=1,M
       IF (MACOMP(I,K1))  999,54,53
   53  I1 =I1+1
       INTAKE(I1) = K1
   54  CONTINUE
       PRINT            865, I, MACOMP(I,M+1),(INTAKE(K1), K1=1,I1)
                 TYPE   865, I, MACOMP(I,M+1),(INTAKE(K1), K1=1,I1)
       I1=0
   55  CONTINUE
   50  CONTINUE
   57  DO  51 I = 1,43
   51  INTAKE(I) =0
       TYPE 851, IDISK
       ACCEPT 859,  (INTAKE(K1),K1 =1,43    )
       IF ( SENSE SWITCH 2)  400,410
```

```
  400  TYPE 863
       GO TO 50
C      READ MACOMP
  410  CONTINUE
       PRINT 866 ,  IDISK,  INTAKE
       IF  (INTAKE(3)  - 4100)  60,420,60
  420  CONTINUE
       GO TO 100
C      ADD/AND
   60  IF (INTAKE(1)-4100) 70,65,70
   65  KADD(IDISK)=1
       GO TO   100
C      SELECT
   70  IF  (INTAKE(1)-6200) 80,75,80
   75  KADD(IDISK)=2
       DO 77 I=1,20
   77  MACOMP(I,M+1)  = 0
       GO TO 100
C      REPLACE
   80  IF  (INTAKE(1)-5900)97,95,97
   95  KADD(IDISK)=4
       GO TO 100
C      GO
   97  IF  (INTAKE(1) -  4700)   99,94,99
   94  CALL LINK  (PAU6)
C      ENTER
   99  IF  (INTAKE(1) - 4500)  300,96,300
   96  CALL LINK (PAU5)
C      HALT
  300  IF  (INTAKE(1) -4800) 340,325,340
  325  STOP
  340    TYPE 852
       GO TO 50
  100  RECORD(IDISK)  (INTAKE(I),  I=1,40)
       GO TO 50
  999  PRINT 998
  998   FORMAT  (10H GOOF STOP)
       END
††††
```

```
‡‡FOR 5
*LDISKPAU5
*FANDK0204
        DEFINE DISK   (40,26)
       DIMENSION INTAKE(43) ,MACOMP(20,11),KADD(10)
       COMMON M,N,IDISK,KOUNT,MACOMP,IN,KADD
    1  FORMAT (20I4)
    2  FORMAT (20A1)
       KDIG2 = 7000
       KT = IDISK
IDISK = 1
  105  IF (IDISK - KT)  107,106,999
  106  CALL LINK (PAU4)
  107  FETCH (IDISK)  (INTAKE(I),  I=1,40)
C      STRIP
  100  CONTINUE
       IF(INTAKE(3)-4100)  102,101,102
  101 READ 1,(INTAKE(J), J=1,M)
       IF  (INTAKE(1) - 3123)  5083,105,5083
 5083  KOUNT = KOUNT + 1
       DO 5082 J=1,M
 5082  MACOMP(KOUNT , J)  = 0
      DO 508  J=1,M
       IF   (INTAKE(J))  999,508,5081
 5081 K1=INTAKE(J)
       INTAKE(J) = 0
       MACOMP(KOUNT,K1)  = 1
  508  CONTINUE
       GO TO 101
  102   IADD = KADD(IDISK - 1)
       GO TO (110,115,120,125), IADD
  110  K2=4
       DO 112  I=1,M
  112  MACOMP(KOUNT+1,I) = 0
       GO TO 130
  115  K2=7
       GO TO 130
  120  K2=9
       GO TO 130
  125  K2=8
  130  CONTINUE
       DO 210 I=K2,44
       IF  (I-44)  140,170,999
C      COMMA
  140  IF  (INTAKE(I)-2300)  150,170,150
C      SLASH
  150  IF  (INTAKE(I)-2100)  160,155,160
  155  K21 = 0
       GO TO 170
  160  IF  (INTAKE(I))   999,210,165
  165  KDIG1 = KDIG2
       KDIG2 = INTAKE(I)
    4  FORMAT  (5H KDIG, 2I4)
       GO TO 210
  170  CONTINUE
       KDIG1=   ((KDIG1-7000)/10)  +  ((KDIG2-7000)/100)
       KDIG2=7000
        IADD = KADD(IDISK - 1)
```

```
      GO TO (180,190,200,220), IADD
  180 MACOMP(KOUNT+1,KDIG1)  =1
      GO TO 210
  190 MACOMP(KDIG1,M+1)  =1
      GO TO 210
  200 CONTINUE
      GO TO 210
  220 IF  (K21) 999,230,225
  225 MACOMP(K21,KDIG1) =1
      GO TO 210
  230 K21=KDIG1
      DO 240  I1=1,M
  240 MACOMP(K21,I1)=0
  210 CONTINUE
       IADD = KADD(IDISK - 1)
      GO TO (250,260,280,105), IADD
  250 KOUNT = KOUNT +1
      GO TO 105
  260 K22=1
      DO  270 I= 1,KOUNT
      IF  (MACOMP(I,M+1))  999,270,265
  265 MACOMP(I,M+1) = K22
      K22=K22+1
  270 CONTINUE
      GO TO 105
  280 CONTINUE
      GO TO 105
  999 PRINT 998
  998  FORMAT  (10H GOOF STOP.)
      END
++++
```

```
††FOR 5
*LDISKPAU6
*FANDK0204
        DEFINE DISK   (40,26)
        DIMENSION INTAB(20,10,2)
       DIMENSION KTEMP(20), KEQUIV(20,10), KOUT(20,10), MACOMP(20,11)
       COMMON M,N,IDISK,KOUNT,MACOMP,IN
  820  FORMAT(//10X, 10H PHASE III/)
       IF  (IN)  999,1530,1518
 1518  DO 1520 K1=1,M
 1520  READ 1, (((INTAB(K1,K2,K3), K3=1,2), K2=1,N)
       IDISK  = 17
       RECORD   (IDISK) (((INTAB(I,J,K), K=1,2), J=1,N), I=1,M)
       IN = 0
       GO TO 1500
 1530  IDISK  = 17
       FETCH    (IDISK) (((INTAB(I,J,K), K=1,2), J=1,N), I=1,M)
 1500  DO 1600 I=1,M
        DO 1610  J=1,KOUNT
       IF  (MACOMP(J,M+1))  999,1610,1510
 1510  IF  (MACOMP(J,I)) 999, 1610,1600
 1610  CONTINUE
       TYPE  1609
 1609  FORMAT  (28H SETS DO NOT COVER, RESELECT)
       CALL LINK  (PAU4)
 1600  CONTINUE
       PRINT 820
       TYPE 820
       K1=0
       K2=0
       K3=0
       K4=0
       K5=0
       K1=N+1
       DO  504 I= 1,20
       DO 504 J= 1,K1
       KEQUIV(I,J)  = 0
  504  CONTINUE
    1   FORMAT (20I4)
       K1=M+1
       DO 509 I =1,M
  509  KTEMP(I) =0
       K5HT =0
       K5H =0
       MARK =0
       MATCH =0
  510   DO 605 K2=1,KOUNT
       IF(MACOMP(K2, M+1))  605  ,605  ,  520
  520  K5= K5HT +1
       K5L=K5
       K5H=0
       KEQUIV(K5, N+1)=MACOMP(K2,M+1)
       KSTAR = KEQUIV(K5,N+1)
  530  DO 600  I=1,N
       KOUT(KSTAR,I)  = (-1)
  535  DO  550 K1=1,M
       IF(MACOMP(K2,K1))  999 ,  550  ,540
  540  K4=INTAB(K1,I,1)
```

```
      IF (KOUT(KSTAR,I))   537,539,539
  537 KOUT(KSTAR,I)  = INTAB(K1,I,2)
  539 IF( K4) 550,999,   545
  545 KTEMP(K4) =1
      MARK =1
  550 CONTINUE
      IF (MARK)  999,555,560
  555 KEQUIV(K5,I) =(-1)
      GO TO 600
  560 MARK  = 0
      DO 580 K22=1,KOUNT
      IF (MACOMP(K22,M+1))   999,580,565
  565 DO 570 K1=1,M
      IF  (MACOMP(K22,K1)  - KTEMP(K1))  580,570,570
  570 CONTINUE
      MATCH = MATCH  + 1
       KEQUIV(K5,I)   = MACOMP(K22,M+1)
      K5 =K5+1
  580 CONTINUE
      IF  (MATCH)   999,  582,585
  582 PRINT 800, MACOMP(K2,M+1),I
      CALL LINK  (PAU4)
  585 IF (MATCH-K5H)  595,595,590
  590 K5H =MATCH
  595 K5 = K5L
      MATCH = 0
      DO 597 I1=1,M
  597 KTEMP(I1) =0
  600 CONTINUE
      K5HT = K5HT + K5H
  605 CONTINUE
  800 FORMAT (21H NOT CLOSED, STAR NO.,  I4,  3H I=, I4)
  610 PRINT 802, (I ,      I=1,N )
      DO 630 K2=1,K5HT
      PRINT 801,  KEQUIV(K2,N+1),(KEQUIV(K2,I), I=1,N)
  630 CONTINUE
  802 FORMAT (/10H NEW INTAB// 6X, 10I4)
  801 FORMAT  (I4,2H *, 10I4)
      PRINT 803,  (I,I=1,N)
  803 FORMAT (/17H NEW INTAB OUTPUT//6X,10I4)
      DO 640  K2=1,KSTAR
      PRINT 801, K2, (KOUT(K2,I),  I=1,N)
  640 CONTINUE
  804 FORMAT   (10X,  28H BEGIN NEW PROBLEM, PHASE II/)
  805 FORMAT(1H1)
      PRINT 805
      PRINT 804
      TYPE 804
      CALL LINK  (PAU4)
  999 PRINT 998
  998 FORMAT   (10H GOOF STOP)
      STOP
       END
++++
```

REFERENCES

1. Maley, G., and Earle, J., *The Logic Design of Transistor Digital Computers,* pp. 271–272, Prentice Hall, 1964.

2. Caldwell, S. H., *Switching Circuits and Logic Design,* pp. 470–479, John Wiley & Sons, Inc., 1958.

3. Paull, M. C., and Unger, S. H., "Minimizing the Number of States in Incompletely Specified Sequential Switching Functions," *IRE Transactions on Electronic Computers,* pp. 346–367, September 1959.

4. Grasseli, A., and Luccio, F., "A Method for Minimizing the Number of Internal States in Incompletely Specified Sequential Networks," *IEEE Transactions on Electronic Computers,* pp. 350–359, June 1965.