AD 744492

INFORMATION PROCESSING

Technical Report Number AU-T-24

A FAST ALGORITHM FOR COMPLETE MINIMIZATION OF BOOLEAN FUNCTIONS

PREPARED BY
S. G. SHIVA AND H. T. NAGLE, JR.
DIGITAL SYSTEMS LABORATORY
OF ELECTRICAL ENGINEERING

JUNE, 1972

CONTRACT DAAH01-68-C-0296
ARMY MISSILE COMMAND
HUNTSVILLE, ALABAMA

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED.

PROJECT THEMIS AUBURN UNIVERSITY

77

UNCLASSIFIED
Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) Vice President for Research Auburn University Auburn, Alabama 36830	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP N/A

3. REPORT TITLE

A FAST ALGORITHM FOR COMPLETE MINIMIZATION OF BOOLEAN FUNCTIONS

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)
Technical Report

5. AUTHOR(S) (First name, middle initial, last name)

S. G. Shiva
H. T. Nagle, Jr.

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
June, 1972	76	4

8a. CONTRACT OR GRANT NO. DAAH01-68-C-0296 b. PROJECT NO. N/A c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) AU-T-24 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report) None

10. DISTRIBUTION STATEMENT

Distribution of this document is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
None	Army Missile Command

13. ABSTRACT

Properties of the cellular n-cube representation are used to advantage in developing a fast algorithm for finding the Prime Implicants, Essentail Prime Implicants and Non-essential Prime Implicants of a Boolean function. The algorithm is discussed and several examples are included showing computer solutions to selected Boolean function minimization problems. The complete FORTRAN source program listing for the automated algorithm is included.

DD FORM 1 NOV 65 1473 REPLACES DD FORM 1473, 1 JAN 64, WHICH IS OBSOLETE FOR ARMY USE.

Unclassified
Security Classification

A

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Logic design						
Minimization						
Boolean functions						
Cellular n-Cube						
Switching algebra						
Prime Implicants						
Combinational circuits						

TECHNICAL REPORT NUMBER AU-T-24

A FAST ALGORITHM FOR COMPLETE MINIMIZATION OF BOOLEAN FUNCTIONS

PREPARED BY

S. G. SHIVA AND H. T. NAGLE, JR.
DIGITAL SYSTEM LABORAOTRY
OF ELECTRICAL ENGINEERING

JUNE, 1972

PROJECT THEMIS: INFORMATION PROCESSING

CONTRACT DAAH01-C-0296
ARMY MISSILE COMMAND
HUNTSVILLE, ALABAMA

APPROVED BY:

C. C. Carroll
Professor and Head of
Electrical Engineering
THEMIS Project Director

H. T. Nagle, Jr.
Associate Professor of
Electrical Engineering

Ben T. Lanham, Jr.
Vice-President for Research
Auburn University

FOREWORD

This report is a technical summary reporting the progress of a study conducted by the Digital Systems Laboratory of the Electrical Engineering Department, Auburn University. The study is focused toward fulfilment of contract No. DAAH01-68-C-0296, granted to Auburn University, Auburn, Alabama, by the Army Missile Command, Huntsville, Alabama.

A FAST ALGORITHM FOR COMPLETE MINIMIZATION OF BOOLEAN FUNCTIONS

S. G. Shiva and H. T. Nagle, Jr.

ABSTRACT

Properties of the cellular n-cube representation are used to advantage in developing a fast algorithm for finding the Prime Implicants, Essential Prime Implicants and Non-essential Prime Implicants of a Boolean function. The algorithm is discussed and several examples are included showing computer solutions to selected Boolean function minimization problems. The complete FORTRAN source program listing for the automated algorithm is included.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

An automated design algorithm was developed in [1] to determine the list of Prime Implicants that cover any given Boolean function. The properties of cellular-n dimensional cubic representation [3] were used in formulating this algorithm. This was extended to take care of "don't care" terms and to determine the essential prime implicants in [2].

This report presents a FORTRAN program capable of determining the non-essential prime implicants. A "weight" is given to each of the constraint terms that cover a minterm not covered by essential Prime Implicants. The maximum weighted term is selected as a non-essential Prime Implicant. The present form of the program can handle Boolean functions of 8th order. To change the storage requirements needed for higher order functions, subscripted arrays must be altered as shown later.

An additional feature of the algorithm is that it can also minimize the complementary Boolean function using the same don't cares. This arises frequently in logic design problems.

Since the program was developed as a general Computer Aided Design (CAD) tool, it is designed to accept da in integer formats and binary formats.

Chapter II discusses the cellular-n cube and the theorems connected with it, as applied to the algorithm formulation. Chapter III presents the algorithm with a general flow chart and selected parts of the flow chart in detail. Chapter IV presents some examples solved using the algorithm.

Appendix I contains the complete listing of FORTRAN program. Appendix II has an assembly language subroutine for LOGICAL AND, used in the main program.

II. THE CELLULAR STRUCTURE AND PROPERTIES OF BOOLEAN FUNCTIONS [3]

The variables in a Boolean product can be conveniently represented as,

$$x_i^{(0)} = \bar{x}_i$$

$$x_i^{(1)} = x_i$$

$$x_i^{(-)} = 1 \text{ (Absence of } x_i)$$

That is, a variable in complemented form is represented by "0", variable in "uncomplemented form" by "1" and the absence of a variable by "-". Then any Boolean product term P may be written as

$$P = x_n^{(J_n)} x_{n-1}^{(J_{n-1})} x_{n-2}^{(J_{n-2})} \ . \ . \ . \ x_1^{(J_1)}$$

where $J_i \in C$ $C = \{0,1,-\}$. $i = 1,2,\ldots,n$.

The Boolean product corresponds to a cell "$\underline{c}$" in the n-cube c^n.

$$P \equiv \underline{c}$$

where $\underline{c} = (c_n, c_{n-1} \ . \ . \ . \ c_1)$, $c_i \in C$.

Example 1: Consider the Boolean function

$$F(A, B, C, D) = P_1 + P_2 + P_3 + P_4$$

where $P_1 = AB\bar{C}D$

$P_2 = A\bar{B}D$

$P_3 = \bar{A}\bar{D}$

$P_4 = \bar{B}C$

Corresponding cellular notations are

$\underline{c}_1 = 1101$

$\underline{c}_2 = 10\text{-}1$

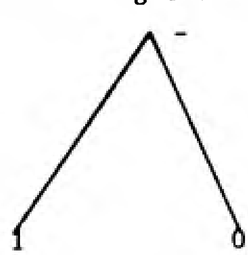
$\underline{c}_3 = 0\text{--}0$

$\underline{c}_4 = \text{-}01\text{-}$

Each of these is a "cell". In $\underline{c}_1$, $c_1 = 1$, $c_2 = 1$, $c_3 = 0$, $c_4 = 1$, etc.

The n-cube

The set $C = \{0,1,-\}$ is a partially order set with the order relation $\geq$ defined as in the figure.



That is, $- \geq 1$, $- \geq 0$, $1 \geq 1$, $0 \geq 0$.

The partially ordered set $(c^n, \leq)$ where $c^n = \{\underline{c}\}$ with the realtion

$$(c_n, c_{n-1}, \ldots, c_1) \leq (c'_n, c'_{n-1}, \ldots, c'_1) \Leftrightarrow c_i \leq c'_i$$

$$c_i \in \{0,1,-\}$$

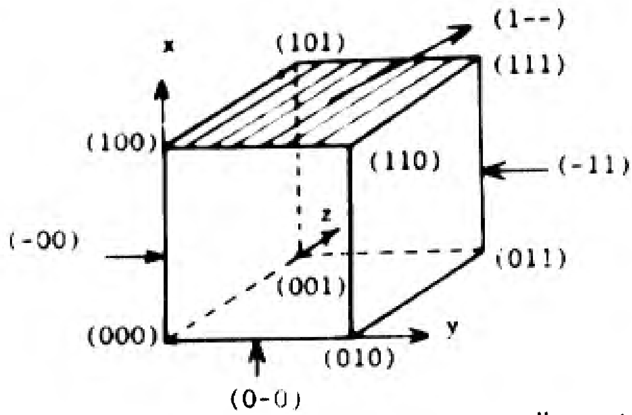
is called a cellular n-cube.

The order of a cell $\underline{c}$ in c^n is equal to the number of components c_i such that c_i = "-". Thus, a "one cell" has one component equal to "-" etc. A "zero cell" is called a "vertex" $\underline{v}$ of c^n. There are 2^n vertices in the n-cube c^n.

Example 2:

Consider a third order function. Here n=3 so, we have c^3.

There are 2^3 = 8 vertices; each vertex corresponds to the vertex of a cube.



"Zero cells" 000, 010, 011, . . . , 111 are "vertices". Each edge corresponds to a "one-cell": 0-0, -11, -00, etc. Each plane corresponds to a "two-cell". Ex.: (1--)

Two vertices $\underline{v}$ that are extremely useful in representing a cell of c^n are the minimum and maximum vertices. Consider the cell

$\underline{c} = (c_n, c_{n-1}, \ldots, c_1)$ where $c_i = 0, 1$ or $-$ $\quad i = 1, 2, 3, \ldots, n.$

The minimum vertex of $\underline{c}$ is given by,

$$\min(\underline{c}) = (v_n, v_{n-1}, \ldots, v_1)$$

where

$$v_i = c_i \quad \text{if } c_i = 0 \text{ or } 1$$

$$v_i = 0 \quad \text{if } c_i = -$$

The maximum vertex is given by

$$\max(\underline{c}) = (v'_n, v'_{n-1}, \ldots, v'_1)$$

where

$$v'_i = c_i \quad \text{if } c_i = 0 \text{ or } 1$$

$$v'_i = 1 \quad \text{if } c_i = -$$

In other words, max $(\underline{c})$ is found by replacing each "-" by a "1" and min $(\underline{c})$ is found by replacing "-" by "0".

Example 3:

Consider $\underline{c}_2 = 10\text{-}1$ in example 1.

$\max(\underline{c}_2) = 1011$

$\min(\underline{c}_2) = 1001$

The Decimal Transform

Consider the vertex $\underline{v} = (v_n, v_{n-1}, \ldots, v_1)$. This vertex has a decimal representation defined as

$$\delta(\underline{v}) = \sum_{i=1}^{N} v_i \, 2^{i-1}$$

Example 4: $\underline{v} = (1011)$

$$\delta(\underline{v}) = 1x2^0 + 1\,x2^1 + 0\,x2^2 + 1x2^3 = 11$$

Similarly, each cell $\underline{c}$ in an n-cube c^n has a decimal representation defined as,

$$D(\underline{c}) = (\delta(\min(\underline{c})), \delta(\max(\underline{c}))$$

Example 5: For the cell in example 3,

$\max(\underline{c}) = 1011$ $\quad\quad \delta(\max(\underline{c})) = 11$

$\min(\underline{c}) = 1001$ $\quad\quad \delta(\min(\underline{c})) = 9$

$$D(\underline{c}) = (9, 11)$$

Notation: Hereafter, a capital letter indicates the decimal transform of the corresponding "lower case letter" in cellular notation.

If $\underline{v} = (v_n, v_{n-1}, \ldots, v_1)$ is a vertex, $V = \delta(\underline{v})$.

Cell $(I, J) \Rightarrow$ cell $\underline{c}$ with $\delta(\min(\underline{c})) = I$

and $\delta(\max(\underline{c})) = J$

Example 6: Consider $\underline{c}$ = (0, -, 1, -, 0, 1)

This is a "two-cell", with min($\underline{c}$) = (0 0 1 0 0 1)

and max($\underline{c}$) = (0 1 1 1 0 1)

(Note: Commas are ommitted for simplicity)

This two-cell has 2^2 vertices. The other two vertices can be obtained by replacing "-" by the remaining combinations of 1 and 0.

That is,

$\underline{v}_1$ = min($\underline{c}$) = (0 0 1 0 0 1) $V_1 = 9$

$\underline{v}_2$ = = (0 0 1 1 0 1) $V_2 = 13$

$\underline{v}_3$ = = (0 1 1 0 0 1) $V_3 = 25$

$\underline{v}_4$ = max($\underline{c}$) = (0 1 1 1 0 1) $V_4 = 29$

Here 9 < 13 < 25 < 29

It is seen that

$$\delta(\min(\underline{c})) \leq \delta(\underline{v}) \leq \delta(\max(\underline{c})) \quad \text{for every } \underline{v} \text{ of } \underline{c}.$$

where $\leq$ means "less than or equal to".

Some of the properties of decimal representation of c^n are useful in defining the relations existing between cells and vertices. Those properties which have been used in the algorithm to arrive at a minimum, non-redundant cover for the Boolean function are now discussed.

The AND operation "." is defined as

·	0	1
0	0	0
1	0	1

The "logical AND" operation "$\wedge$" for two vertices $\underline{v}$ and $\underline{v}'$ is defined as

$$(v_n, v_{n-1}, \ldots, v_1) \wedge (v'_n, v'_{n-1}, \ldots, v'_1)$$

$$= (v_n \cdot v'_n, v_{n-1} \cdot v'_{n-1}, \ldots, v_1 \cdot v'_1)$$

This is essentially the logical intersection of two n-element sets, such that if both the sets contain a "1". in a given position, the resulting set will contain a "1" in that position. An assembly language function subroutine (IAND) is used in the program to perform this operation.

Note: $V \wedge V' \longrightarrow (v_n, v_{n-1}, \ldots, v_1) \wedge (v'_n, v'_{n-1}, \ldots, v'_1)$

<u>Example 7:</u> $\underline{v}$ = (1 0 0 1)

$\underline{v}'$ = (1 0 1 0)

$\underline{V} \wedge \underline{V}'$ = (1 0 0 1) $\wedge$ (1 0 1 0)

= (1·1, 0·0, 0·1, 1·0)

= (1 0 0 0)

Note, by looking at the result we can say, in which positions both the vertices contain a'1'.

If two vertices $\underline{v}_1$ and $\underline{v}_2$ of the n-cube are such that $\underline{v}_1 \wedge \underline{v}_2 = \underline{v}_1$, then $\underline{v}_1$ is related to $\underline{v}_2$ as $\underline{v}_1 \leqslant \underline{v}_2$ meaning that $\underline{v}_1$ is contained in $\underline{v}_2$; i.e., for every co-ordinate of $\underline{v}_1$ equal to 1 a corresponding co-ordinate of $\underline{v}_2$ will also be equal to 1. We may also say that if

$$V_1 \wedge V_2 = V_1$$

where

$$V_1 = \delta(\underline{v}_1),\ V_2 = \delta(\underline{v}_2),$$

then

$$V_1 \leqslant V_2$$

Theorem 1: If $\underline{c} \in c^n$, then $\min(\underline{c}) \leqslant \max(\underline{c})$

This theorem provides a method to determine if a minterm pair of a Boolean function maps on to a cell of c^n. If I and J are two minterms of an nth order Boolean function, each of them correspond to a vertex of the n-cube c^n. Now, if $I \wedge J = I$, $I \leqslant J$. So, from the theorem (I, J) is a cell with minimum vertex I and maximum vertex J.

Example 8: Consider two minterms 15 and 10 of a 4th order function.

$I = 10 \qquad \underline{i} = (1\ 0\ 1\ 0)$

$J = 15 \qquad \underline{j} = (1\ 1\ 1\ 1)$

$I \wedge J \rightarrow \underline{i} \wedge \underline{j} = (1\ 0\ 1\ 0) \wedge (1\ 1\ 1\ 1)$

$\qquad = 1\ 0\ 1\ 0 \qquad = \underline{i}$

(10, 15) is a cell of c^4, represented by (1-1-)

Note that, knowing a minterm pair which forms a cell, the cellular notation for the cell can be found by replacing the position in which they differ by "-" and retaining the same variable in the positions they agree.

i_i	j_i	c_i
0	0	0
0	1	-
1	1	1
1	0	-

Theorem 2: Vertex $\underline{v}$ is contained in cell $\underline{c}$ [$\underline{v} \subseteq \underline{c}$] iff $\underline{v} \leq \max(\underline{c})$ and $\min(\underline{c}) \leq \underline{v}$.

This theorem is used to determine if a vertex [a minterm] $\underline{v}$ of the n-cube [Boolean function] is covered by a cell (I, J) [group of minterms].

If

$\underline{i} \wedge \underline{v} = \underline{i}$ $\qquad$ $I \wedge V = I$

$\underline{j} \wedge \underline{v} = \underline{v}$ $\qquad$ $J \wedge V = V$

then

$V \subseteq (I, J)$

Example 9: Consider minterm 11 and the pair (10, 15)

$V = 11 \quad \underline{v} = (1\ 0\ 1\ 1)$

$\underline{i} \wedge \underline{v} = (1\ 0\ 1\ 0) \wedge (1\ 0\ 1\ 1)$

$= 1\ 0\ 1\ 0 \qquad = \underline{i}$

$\underline{1} \wedge \underline{v} = (1\ 1\ 1\ 1) \wedge (1\ 0\ 1\ 0)$

$= 1\ 0\ 1\ 0 \qquad = \underline{v}$

Hence, $\underline{v}$ is contained in (I.J).

Theorem 3: If $\min(\underline{c}_1) \leqslant \min(\underline{c}_2)$ and $\max(\underline{c}_2) \leqslant \max(\underline{c}_1)$ then $\underline{c}_1 \subseteq c_2$.

This gives the critenion for containment of a cell in another cell. For a cell (I_1, J_1) to cover (I_2, J_2)

$I_1 \wedge I_2 = I_1$

$J_1 \wedge J_2 = J_2$

This theorem is used to test whether a cell is covered by another cell already listed as a Prime Implicant, before a new Prime Implicant is identified. This is employed in forming Prime Implicant and Essential Prime Implicant tables.

Example 10: Consider

$\underline{c}_1 = (1\ -\ 1\ -) \rightarrow (10, 15)$

$\underline{c}_2 = (1\ 1\ 1\ -) \rightarrow (14, 15)$

$10 \wedge 14 = (1\ 0\ 1\ 0) \wedge (1\ 1\ 1\ 0) = 1\ 0\ 1\ 0 = 10$

$15 \wedge 15 = (1\ 1\ 1\ 1) \wedge (1\ 1\ 1\ 1) = 1\ 1\ 1\ 1 = 15$

$\underline{c}_2$ is covered by $\underline{c}_1$

By observing the two cells one notes that they agree in 3 positions, and in the second position from left $\underline{c}_1$ has a "-" and $\underline{c}_2$ has a "1". Since, $- \geq 1$, $\underline{c}_1 \supseteq \underline{c}_2$

<u>Theorem 4</u>: If $\underline{v}_1 \leq \underline{v}_2$ then $\delta(\underline{v}_1) \leq \delta(\underline{v}_2)$ or $V_1 \leq V_2$. For one vertex to contain another, its decimal transform must be greater than the contained vertex.

<u>Lemma 4·1</u>: If $\underline{c}_1 \subseteq \underline{c}_2$, then, $\delta(\min(\underline{c}_1)) \leq \delta(\min(\underline{c}_2))$ and $\delta(\max(\underline{c}_1)) \geq \delta(\max(\underline{c}_2))$.

This provides a criterion for testing the containment of one cell in another just by comparing their minimum and maximum vertices. For (I_1, J_1) to contain (I_2, J_2)

$$I_1 \leq I_2 \leq J_2 \leq J_1.$$

This condition is used before using Theorem 3, when testing for containment of one cell in the other.

<u>Example 11</u>: Considering the cells (10, 15) and (14, 15) of Example 10,

$$10 < 14 < 15 \leq 15.$$

In short, for a vertex V to be covered by a cell (I, J)

$$I \leq V \leq J$$

$$I \wedge V = I$$

$$J \wedge V = V$$

and, for a cell (I_1, J_1) to cover (I_2, J_2)

$$I_1 \leq I_2 \leq J_2 \leq J_1$$

$I_1 \wedge I_2 = I_1$

$J_1 \wedge J_2 = J_2$

Example 12: Consider $f(w,x,y,z) = \sum m(1,3,5,7)$

Test for cells between all possible minterm pairs.

(1, 7)	:	(0 0 0 1) ∧ (0 1 1 1) = (0 0 0 1)	A cell (0 - - 1)
(1, 5)	:	(0 0 0 1) ∧ (0 1 0 1) = (0 0 0 1)	A cell (0 - 0 1)
(1, 3)	:	(0 0 0 1) ∧ (0 0 1 1) = (0 0 0 1)	A cell (0 0 - 1)
(3, 7)	:	(0 0 1 1) ∧ (0 1 1 1) = (0 0 1 1)	A cell (0 - 1 1)
(3, 5)	:	(0 0 1 1) ∧ (0 1 0 1) = (0 0 0 1)	Not a cell.
(5, 7)	:	(0 1 0 1) ∧ (0 1 1 1) = (0 1 0 1)	A cell (0 1 - 0)

Test for containment of cells.

(1, 7) and (1, 5) $1 \leq 1 \leq 5 \leq 7$

Also $1 \wedge 1 = 1$; $7 \wedge 5 = 5$ (1, 7) Contains (1, 5)

Similarly, (1, 7) contains (1, 3), (3, 5) and (5, 7)

Cell (1, 7) covers the complete function f.

$\underline{c}$ = (0 - - 1) is the Prime Implicant.

i.e., $f = \overline{w}z$

Note: By organizing the minterms in increasing order of their decimal representation, and comparing the first with largest, next largest and so on, (as in the example) we generate the larger cells first. This guarantees the entry of largest Prime Implicants into the P. I. table [corresponding to looking for largest block of cells on a k-map]. This ordering is adopted in the algorithm.

III. ALGORITHM

The theorems on the cellular n-cube are used to set up a computer program to select Prime Implicants, Essential P. I's and non-essential P. I.'s. A complete flow chart is given in Figure 1. Detailed flow charts of several sections of the program are given in subsequent figures.

Step 1: The first step in the program is to read in the program flags, N, D, ID, minterms and don't cares.

N = order of the function..

D = program flag, to select the mode for reading in the data.

D	Minterms	Don't Cares
0	Binary	No Don't Cares
1	Binary	Binary
2	Integer	Integer
3	Binary	Integer
4	Interger	Binary
5	Integer	No Don't Cares

Fig. 2. gives the flow chart for selecting the mode in which data has to be read in.

ID = Program flag to select the complemented or uncomplemented form of the function for minimization. (with the same don't cares, in both the cases).

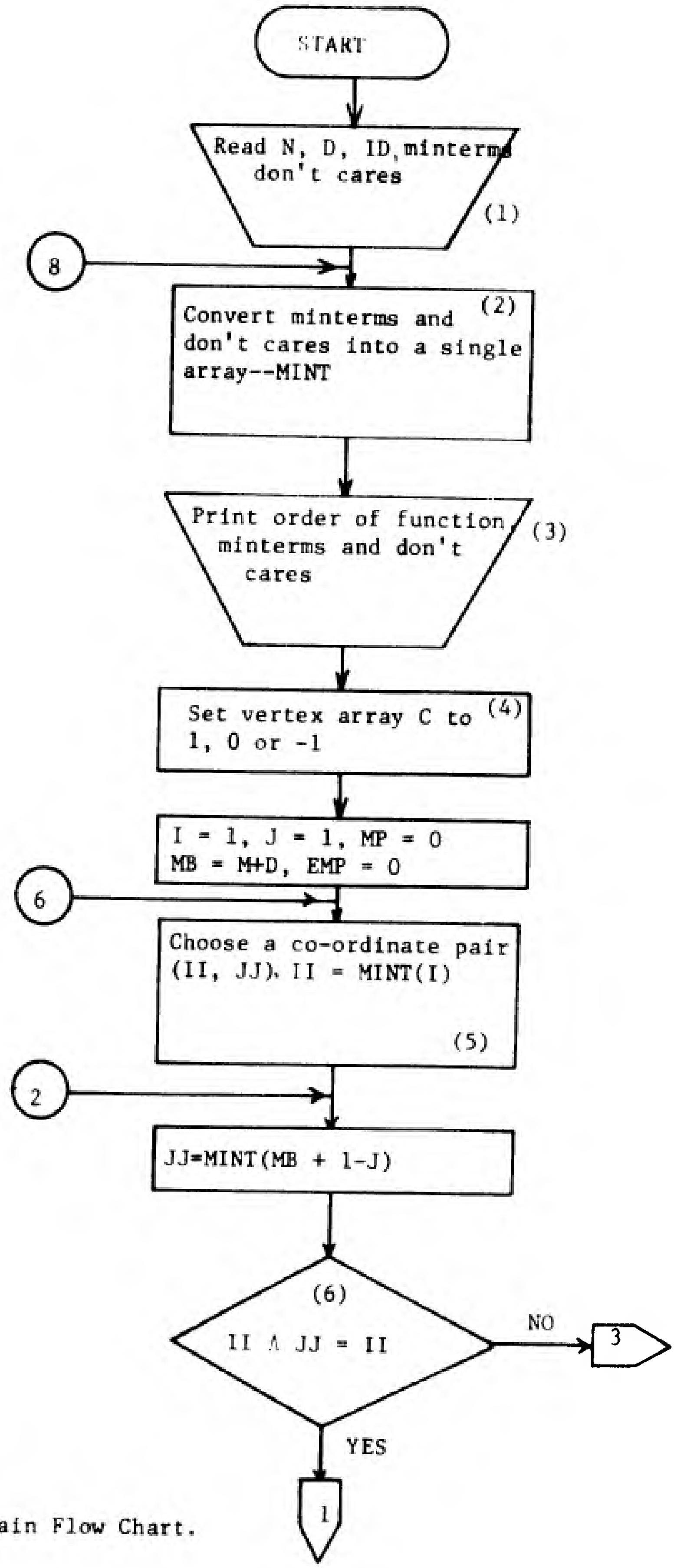


Figure 1. Main Flow Chart.

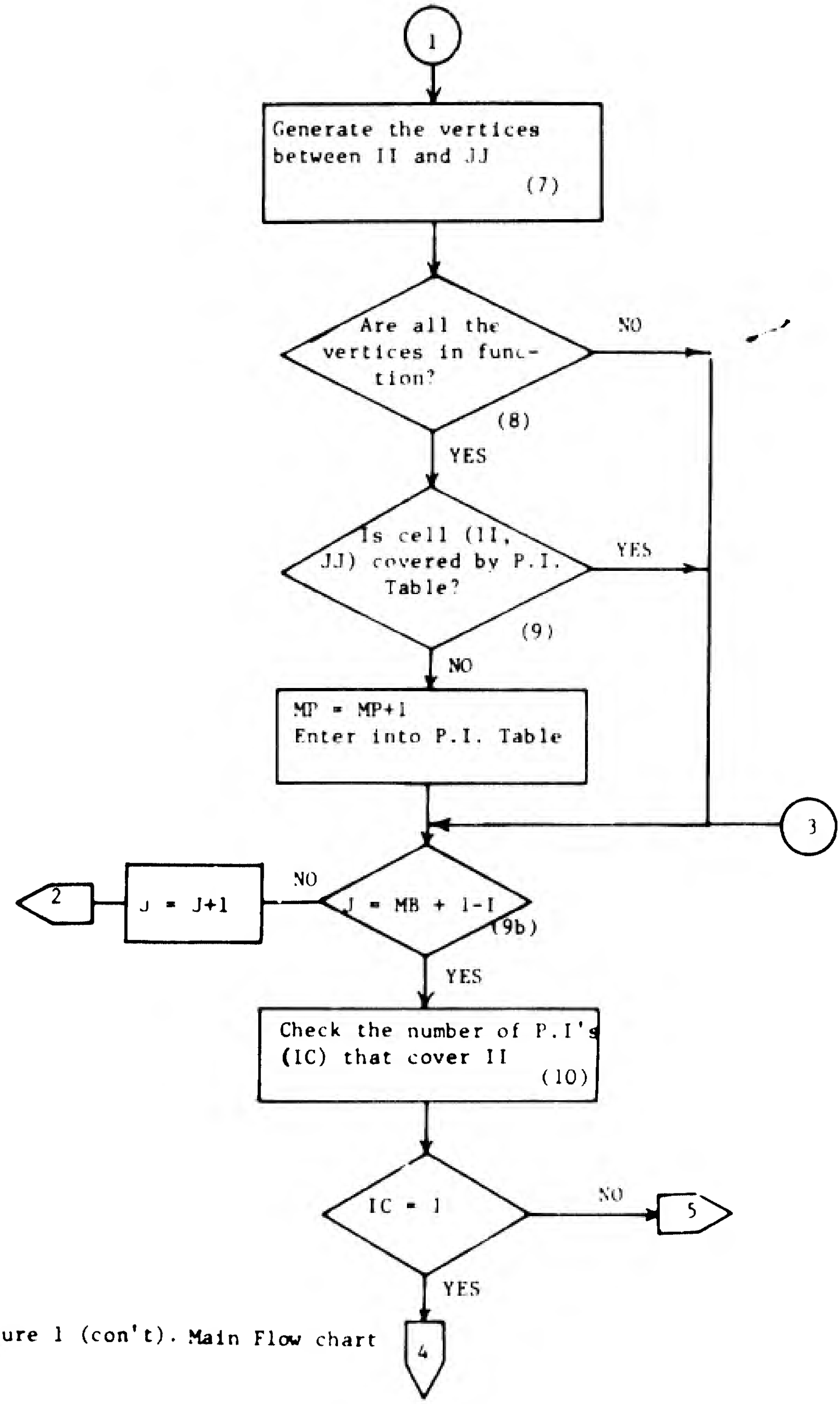


Figure 1 (con't). Main Flow chart

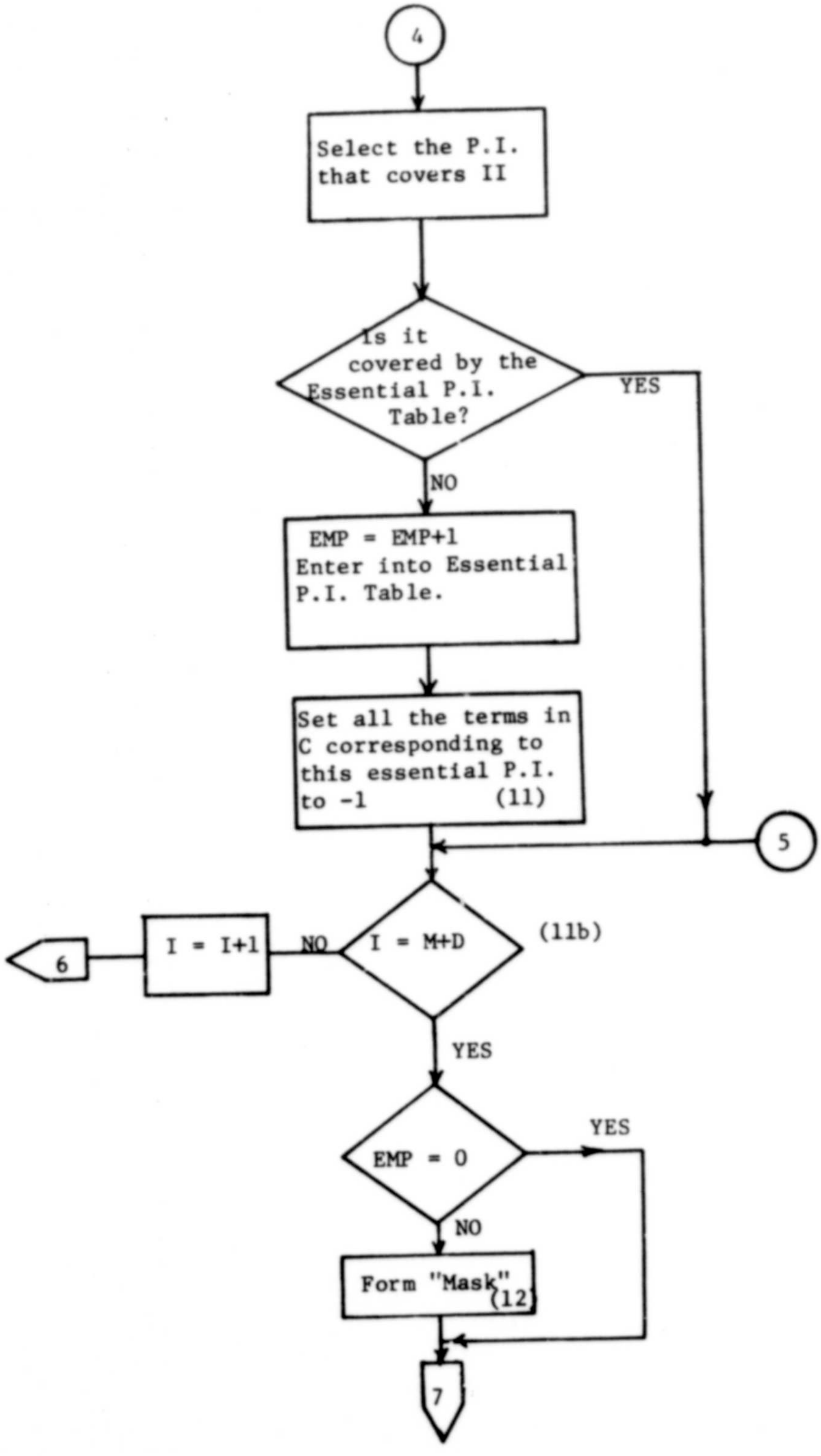


Figure 1 (con't). Main Flow Chart.

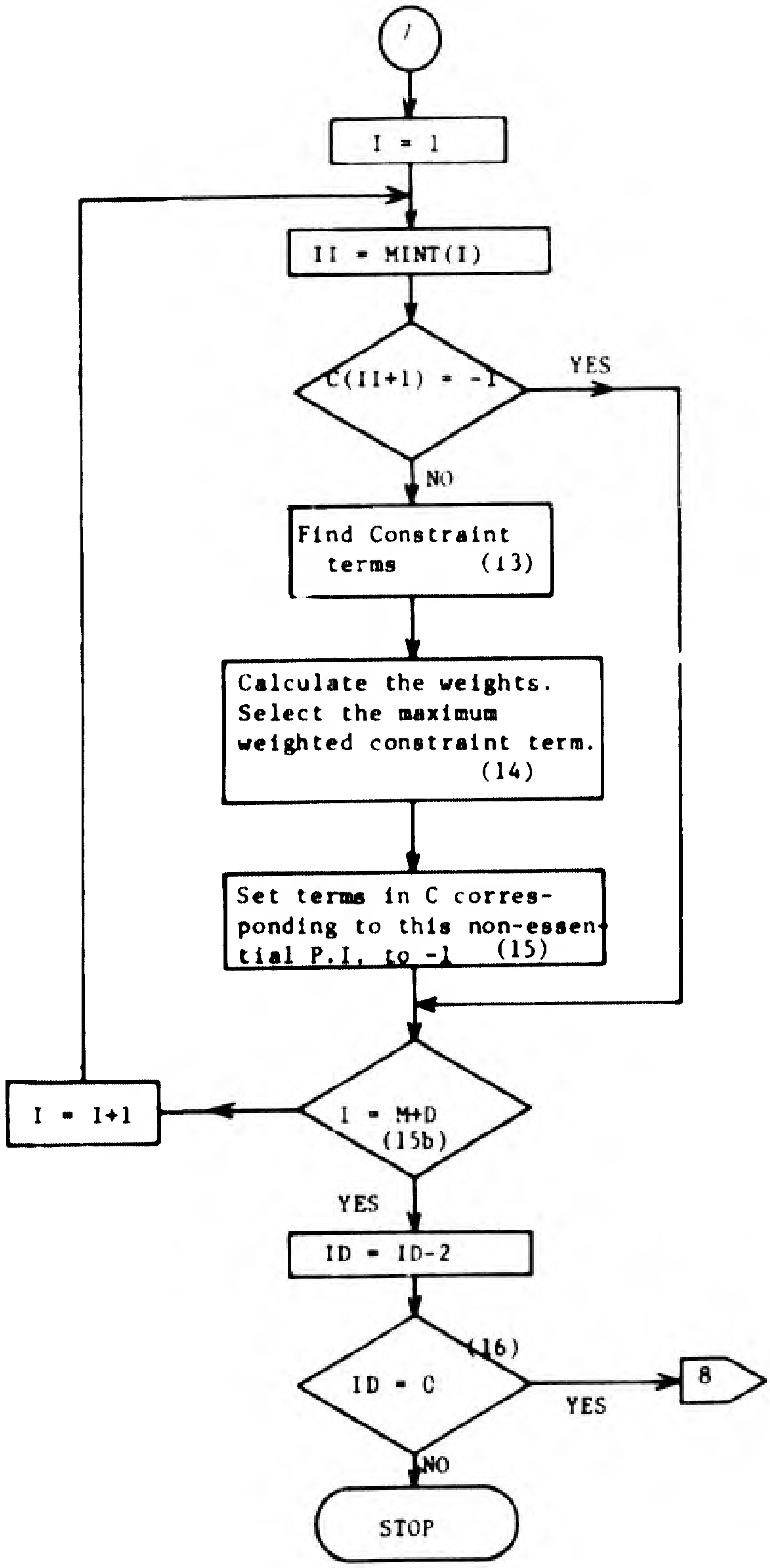


Figure 1 (con't). Main Flow chart.

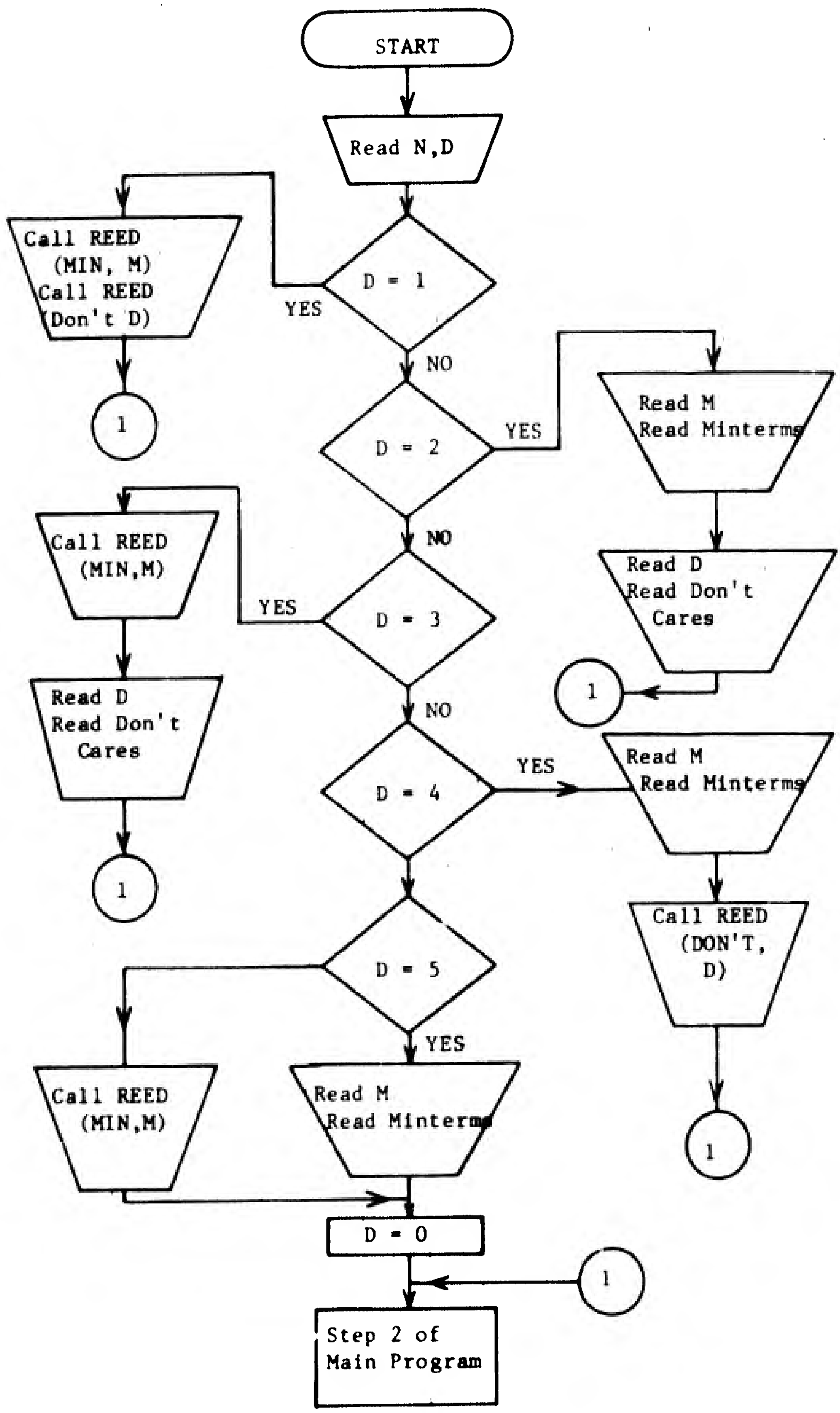


Figure 2. Selecting the mode of data.

ID	Type of Function	
0	Complemented	$\bar{f}$
1	Uncomplemented	f
2	Both	$\bar{f}$, f

Data in Binary (0, 1, - form) is read in by a subroutine REED. The detailed flow chart is given in Fig. 3. This routine generates all the minterms in integer format and arranges them in increasing order for algorithm execution.

When data is in integer format the minterms and don't cares are read in increasing order of their decimal representation.

Step 2: In this step, as indicated by "ID", the minterms or complementary terms are selected. These are combined with don't cares and a single array of terms is formed with the lowest first and highest last. (Fig. 4)

Step 3: Order of the function minterms, don't cares are printed. Same details are printed, whether the function minimized is complemented or uncomplemented.

Step 4: This step sets the vertex array. The vertex array has 2^n elements. The ith element is set to "1", if (i-1) is a numerical representation of a minterm; to "-1" if a don't care and to "0" otherwise. A detailed flow chart is given in Figure 5.

Step 5: In this step a co-ordinate pair (II, JJ) is selected from the combined array of minterms and don't cares. This pair is to be tested for being a "cell". By selecting II as smallest and

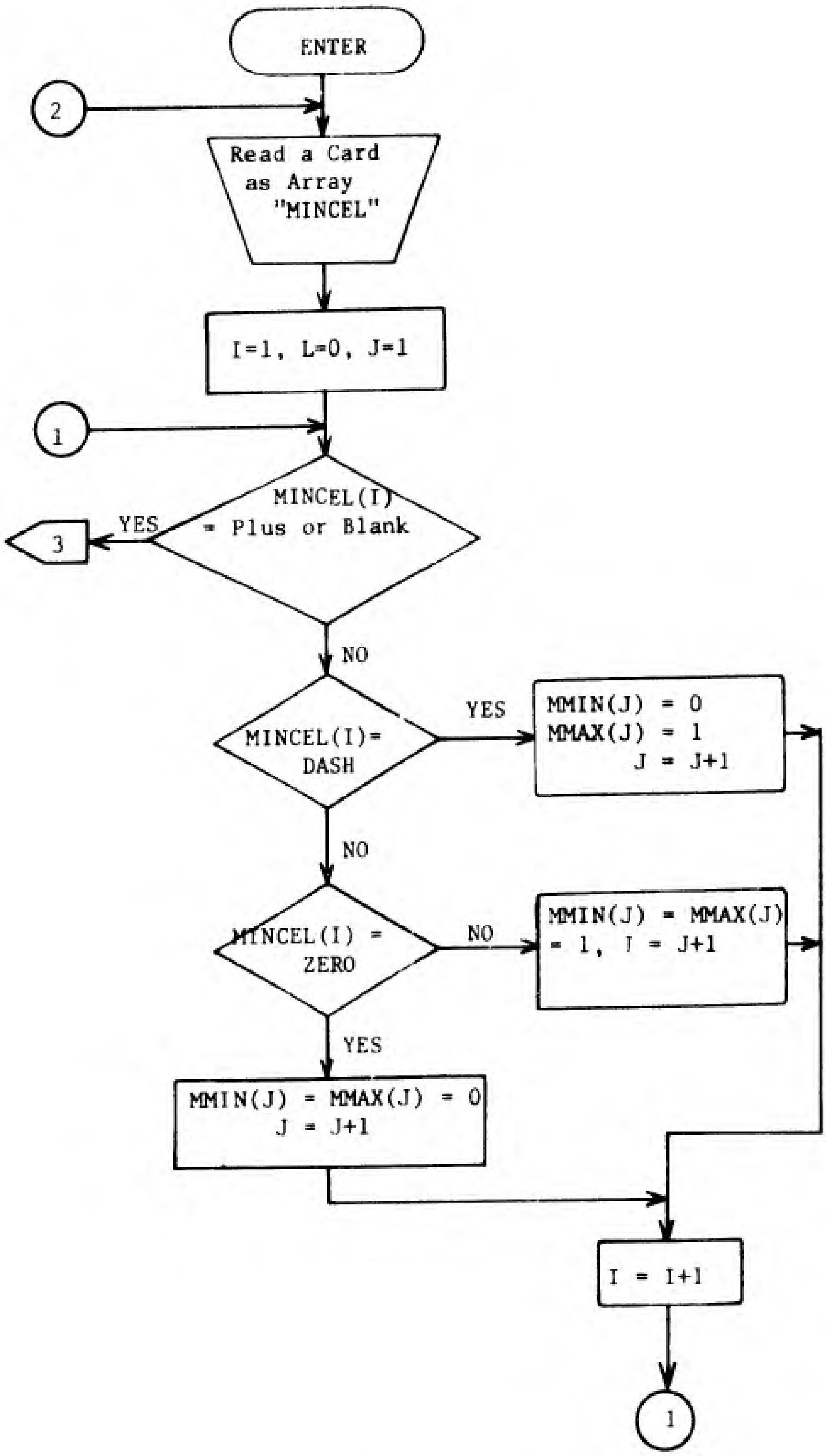


Figure 3. Subroutine REED, to read data in Binary format.

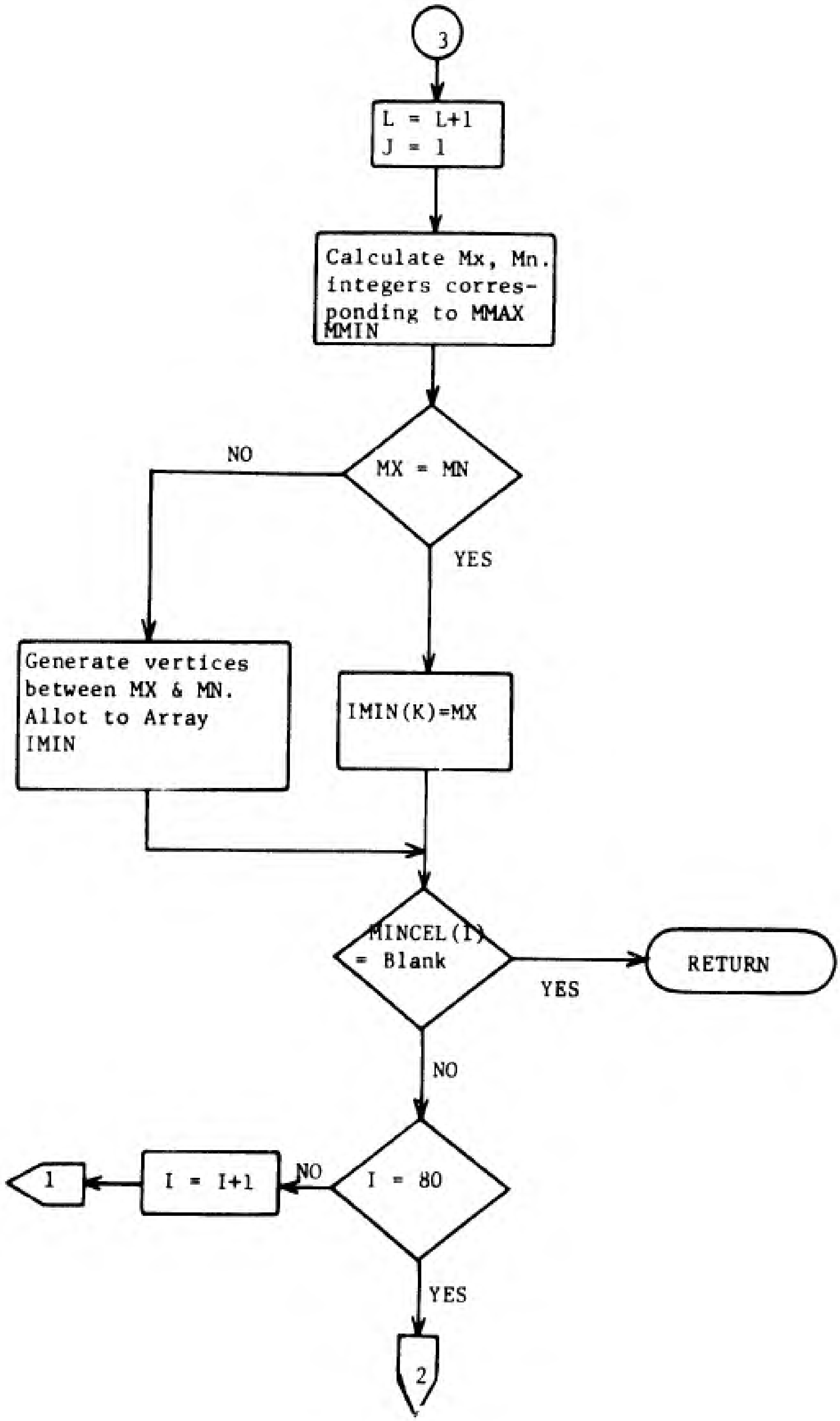


Figure 3 (con't). Subroutine REED, to read Data in Binary Format.

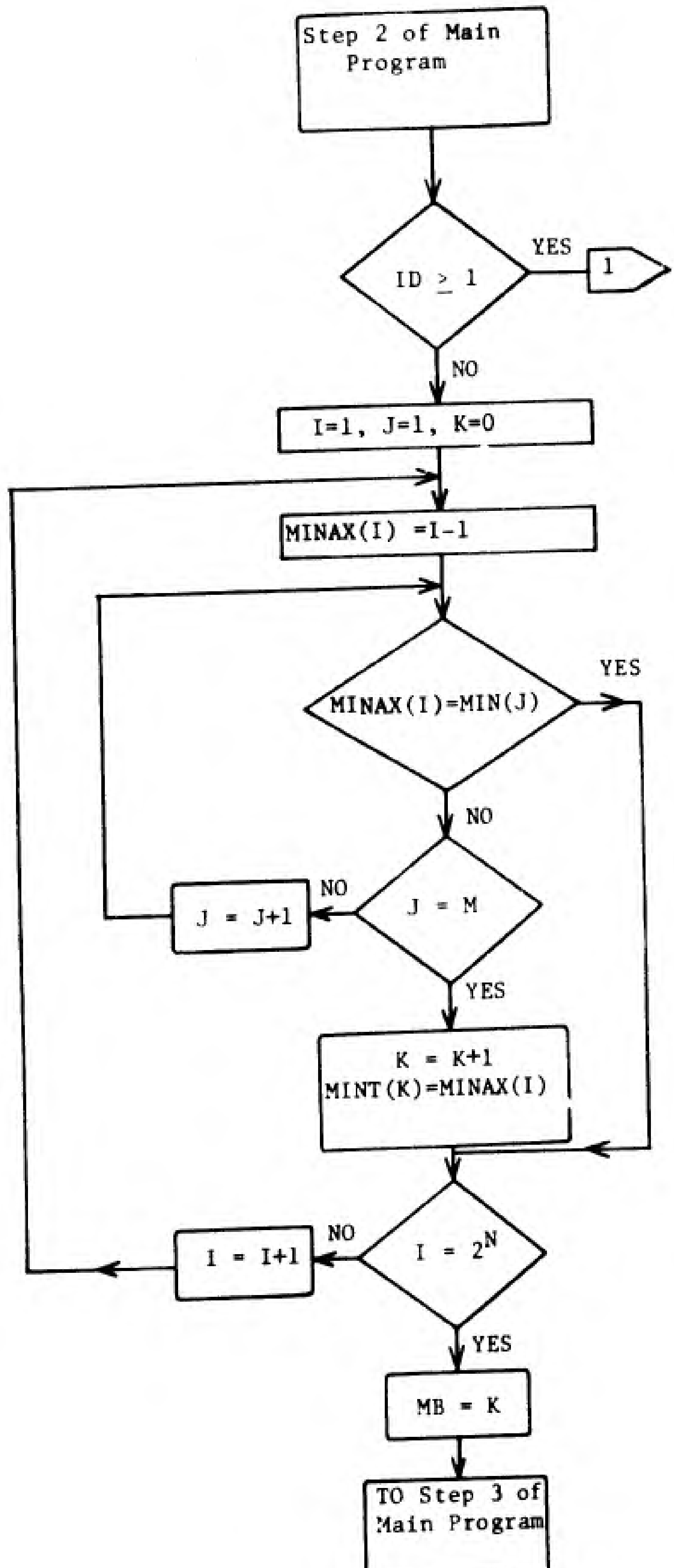


Figure 4. Combine Minterms and Don't Cares into a Single Array MINT.

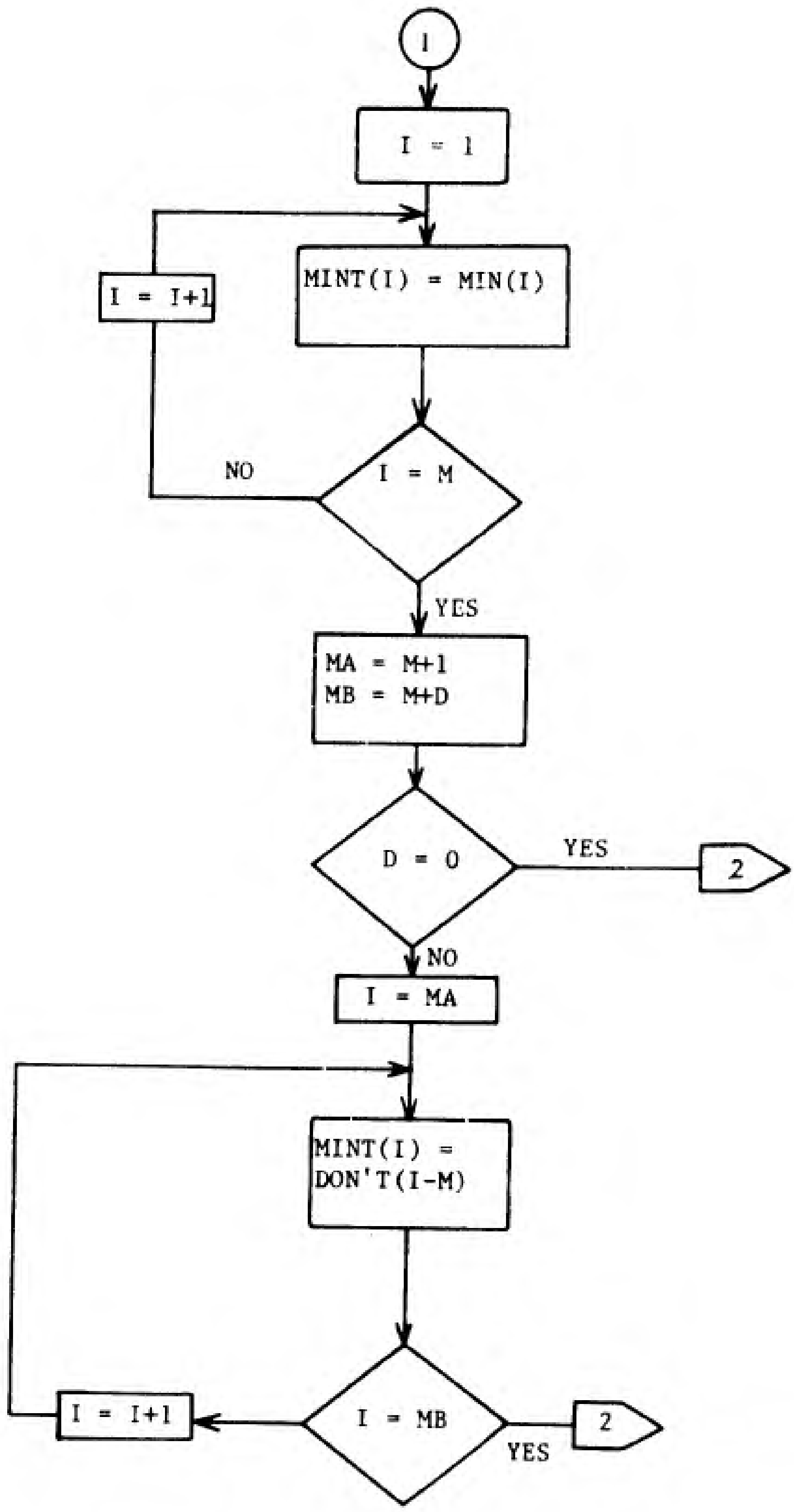


Figure 4.(con't). Combine Minterms and Don't Cares into a Single Array MINT.

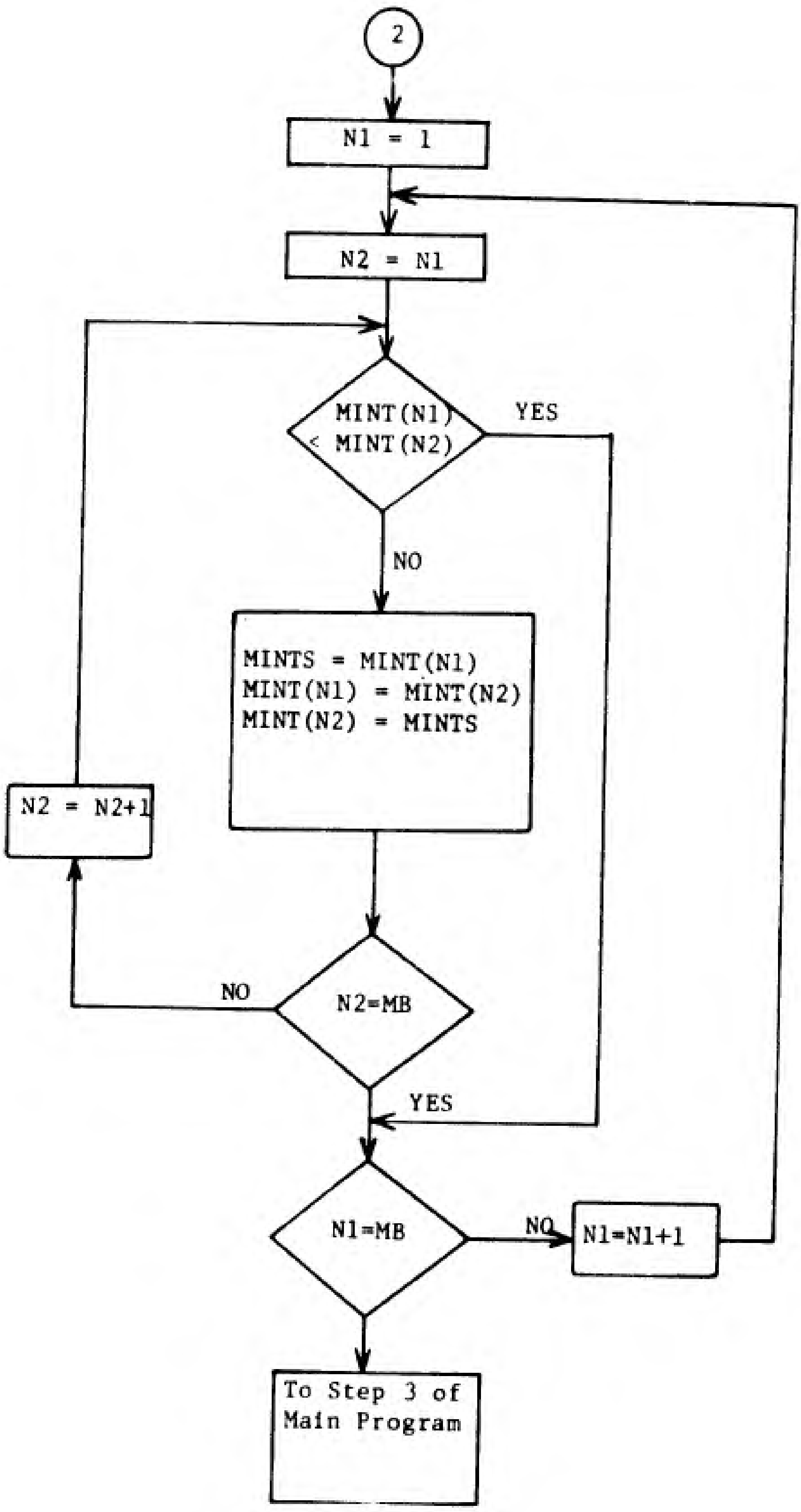


Figure 4 (con't). Combine Minterms and Don't Cares into a Single Array MINT.

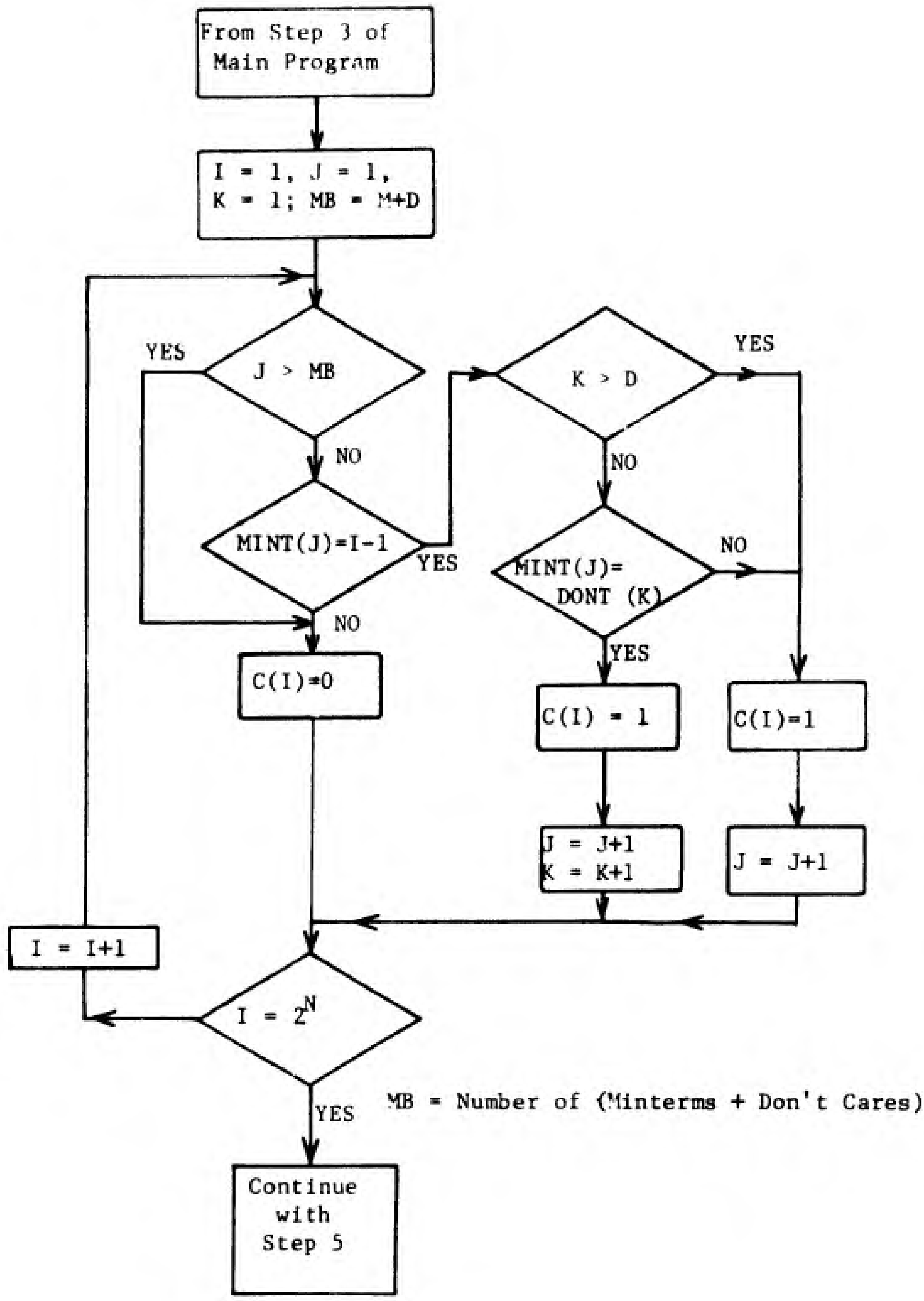


Figure 5. Set Vertex Array "C'.

"JJ" the largest, higher order cells of the function are selected first. This organization prevents the entry of cells into the P. I. table, which are later found to be covered by other cells. This is a result of Lemma 4·1.

Step 6: The selected co-ordinate pair is tested for being a "cell." This is done by forming the "logical AND" of minimum and maximum vertices (II, JJ). If the result is the minimum vertex, this pair forms a "cell." This is a result of Theorem 1.

Step 7: If the selected co-ordinate is a cell, this step generates all the vertices of the cell. A detailed flow chart is given in Figure 6.

Step 8: The vertices generated are tested for containment in the function to be minimized. From Theorem 2, if $\mathbf{v} \preccurlyeq \max(\mathbf{c})$ and $\min(\mathbf{c}) \preccurlyeq \mathbf{v}$, then $\mathbf{v} \leq \mathbf{c}$. Therefore all vertices $\mathbf{v}$ such that $\mathbf{v} \wedge \min(\mathbf{c}) = \min(\mathbf{c})$ and $\mathbf{v} \wedge \max(\mathbf{c}) = \mathbf{v}$ are found. Then corresponding elements of the vertex array are examined. If all these elements are non-zero (1 or -1), the function contains all the vertices; otherwise, this cell is not a cell of the function. (Fig. 6)

Step 9: If the "cell" is a cell of the function, it is compared with the P. I. table to see if any of the P. I.'s contain this "cell". From Theorem 3, for a cell $\mathbf{c}$ to contain $\mathbf{c}'$, $\min(\mathbf{c}) \preccurlyeq \min(\mathbf{c}')$ and $\max(\mathbf{c}') \preccurlyeq \max(\mathbf{c})$. So, if (II', JJ') is a term in the P.I. table and (II,JJ) is the cell under consideration, II' $\wedge$ II = II' and JJ' $\wedge$ JJ = JJ for (II,JJ) to be contained in (II',JJ'). If not, (II,JJ) is entered as a Prime Implicant. (Fig. 7)

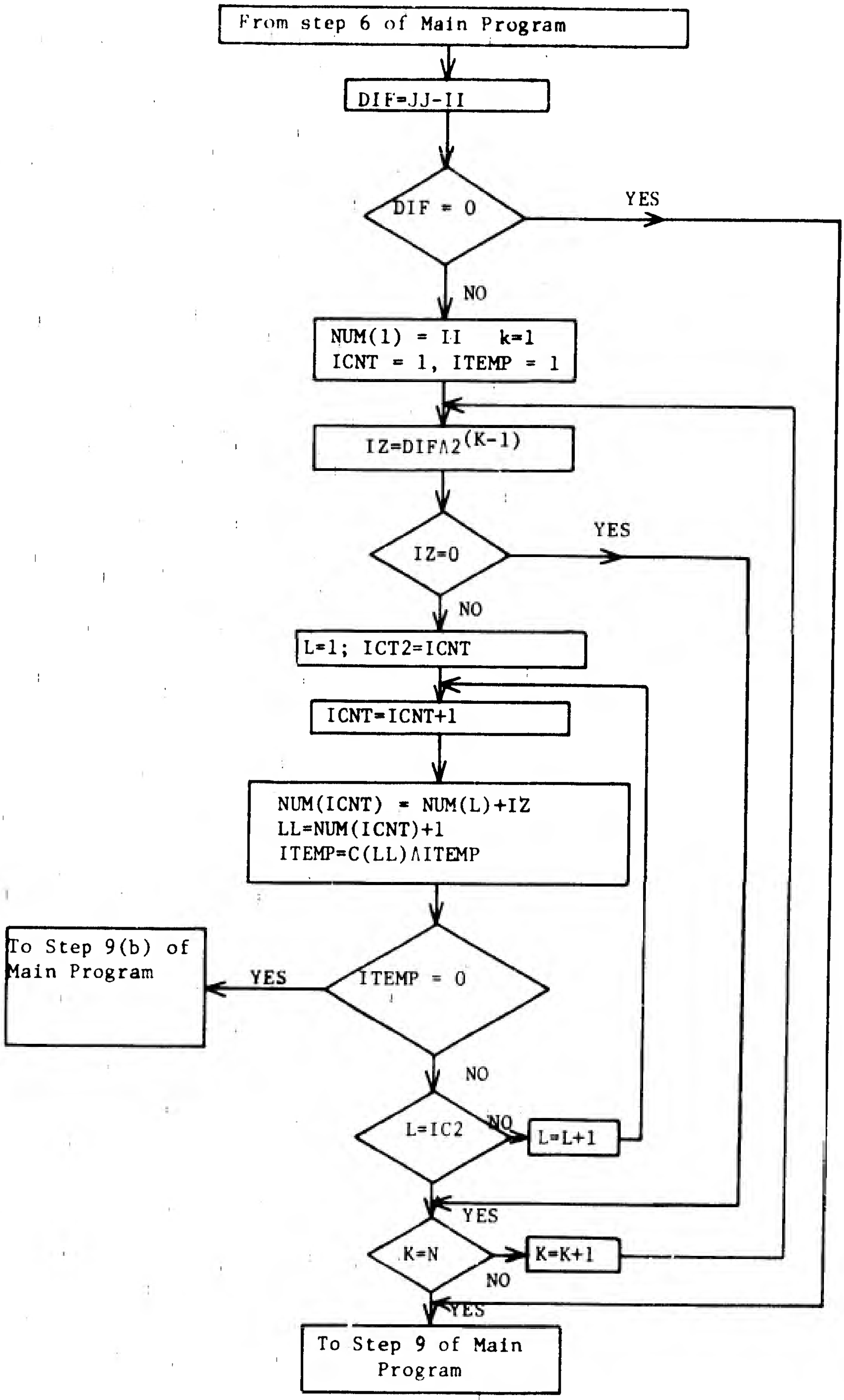


Figure 6. Generate vertices of (II,JJ);
Test if (II,JJ) is a cell of the function.

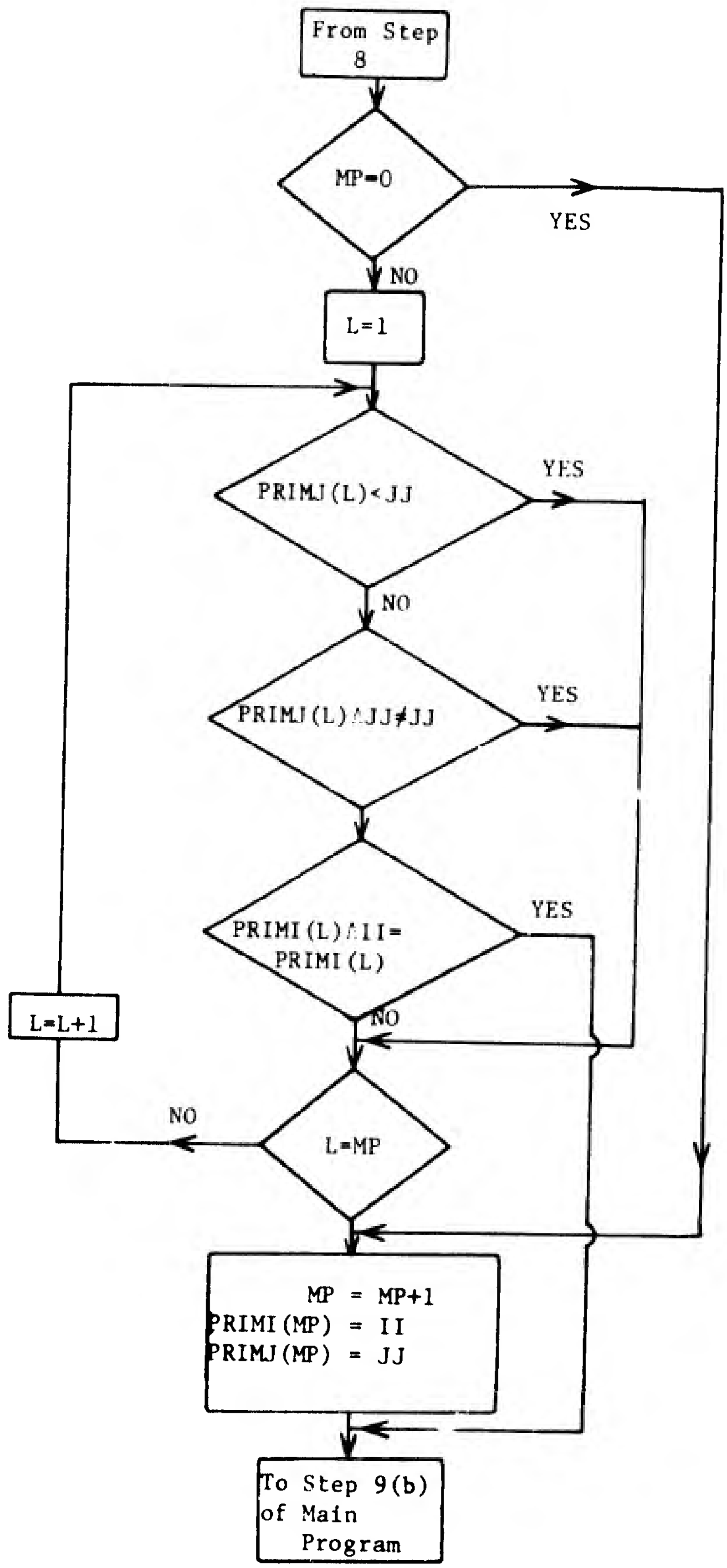


Figure 7. Check if (II, JJ) is a Prime Implicant.

All these steps are repeated selecting the next highest term as JJ with II and so on. Now the P. I. table covering II is complete.

Step 10: The number of P. I's that cover "II" is checked in this step. If only one of these P.I.'s cover II, it is an essential prime implicant. It is compared with the essential P.I. table for containment. If it is not contained by any term in the essential P.I. table it is entered into the table. (Fig. 8)

Step 11: All the terms in vertex array corresponding to minterms covered by this essential P. I. are set to "-1". The details of this step are shown in Figure 9. The boxes shown in dotted lines set the vertex array to "-1".

These steps are repeated with next term as "II", thus covering all the terms.

Step 12: The essential P.I.'s in cellular notation are converted into binary form; (each term into an array of 1, 0, -). This is accomplished by subroutine CELBIN (Fig. 13). A "mask" is formed by scanning these arrays term by term. If a "literal" exists both complemented and uncomplemented, the corresponding term in Mask is a "2"; if the "literal exists only uncomplemented it is a "1"; if only complemented it is a "0", if does not exist it is a "-". (Fig. 10)

Example: Essential P.I. Table

Cellular Notation	Binary
(0,6)	0 0 - - 0
(4,15)	0 - 1 - -
(9,15)	0 1 - - 1
Mask	0 2 1 - 2

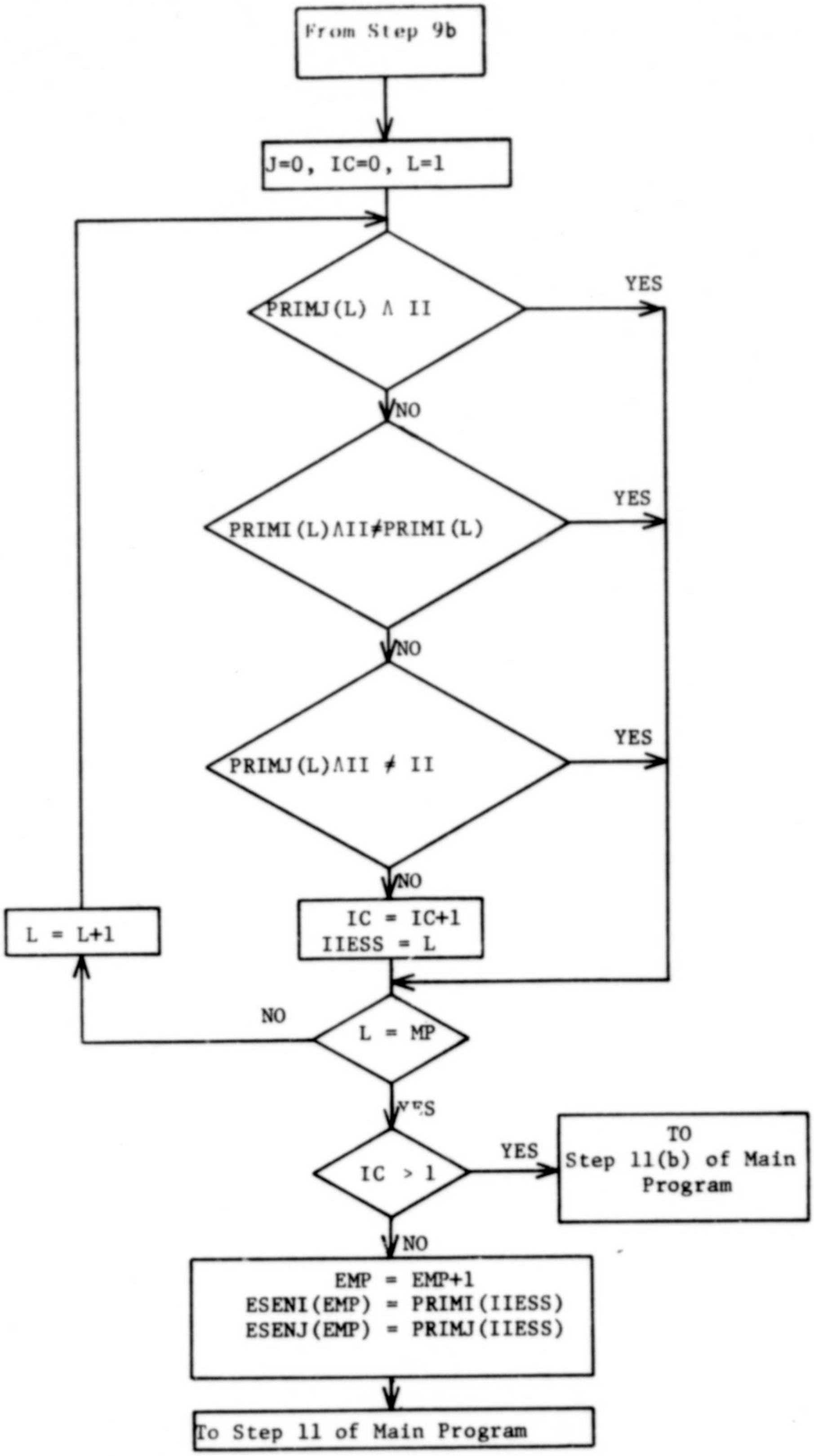


Figure 8. Enter Essential Prime Implicants.

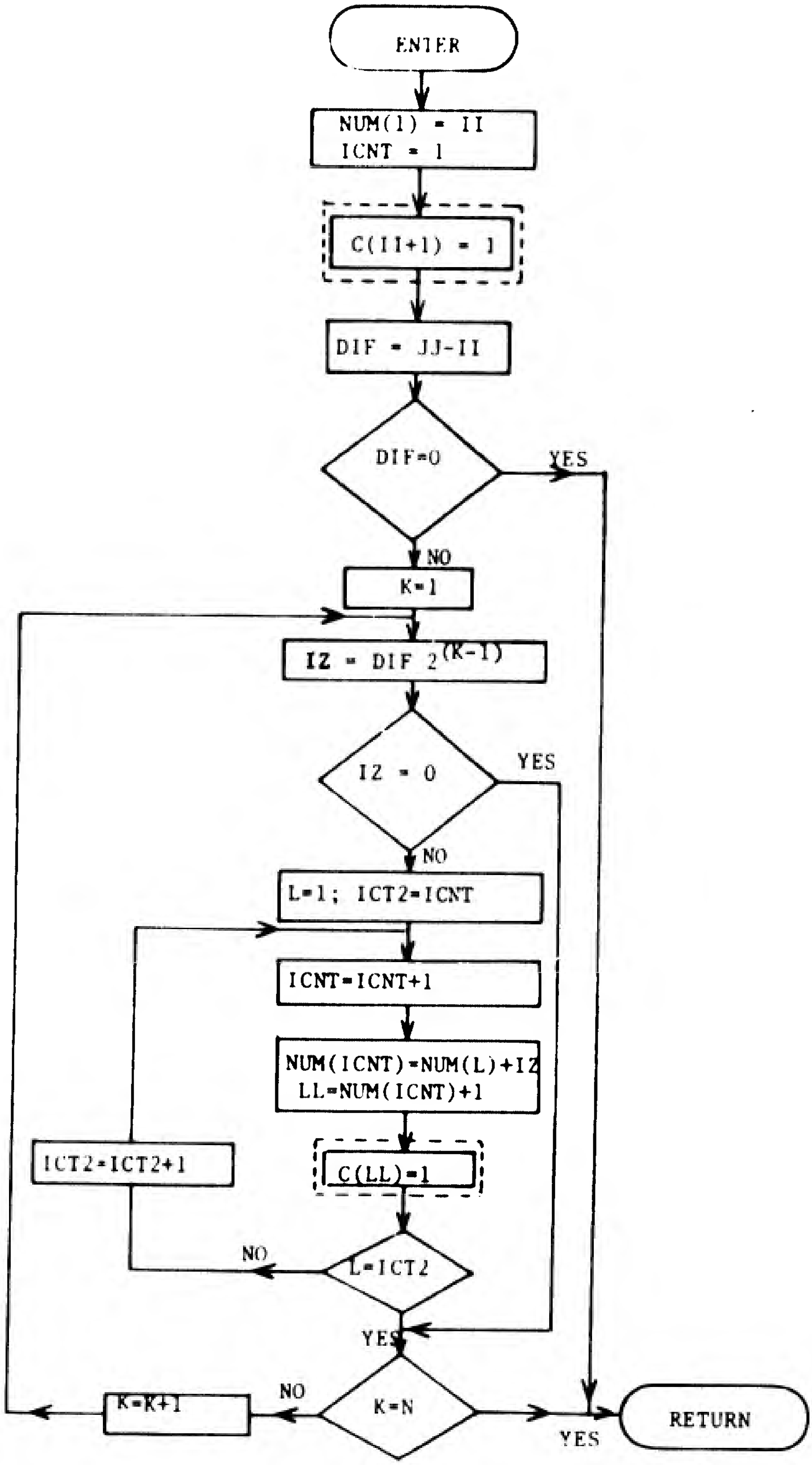


Figure 9. Subroutine RESETC, to set elements of C to -1.

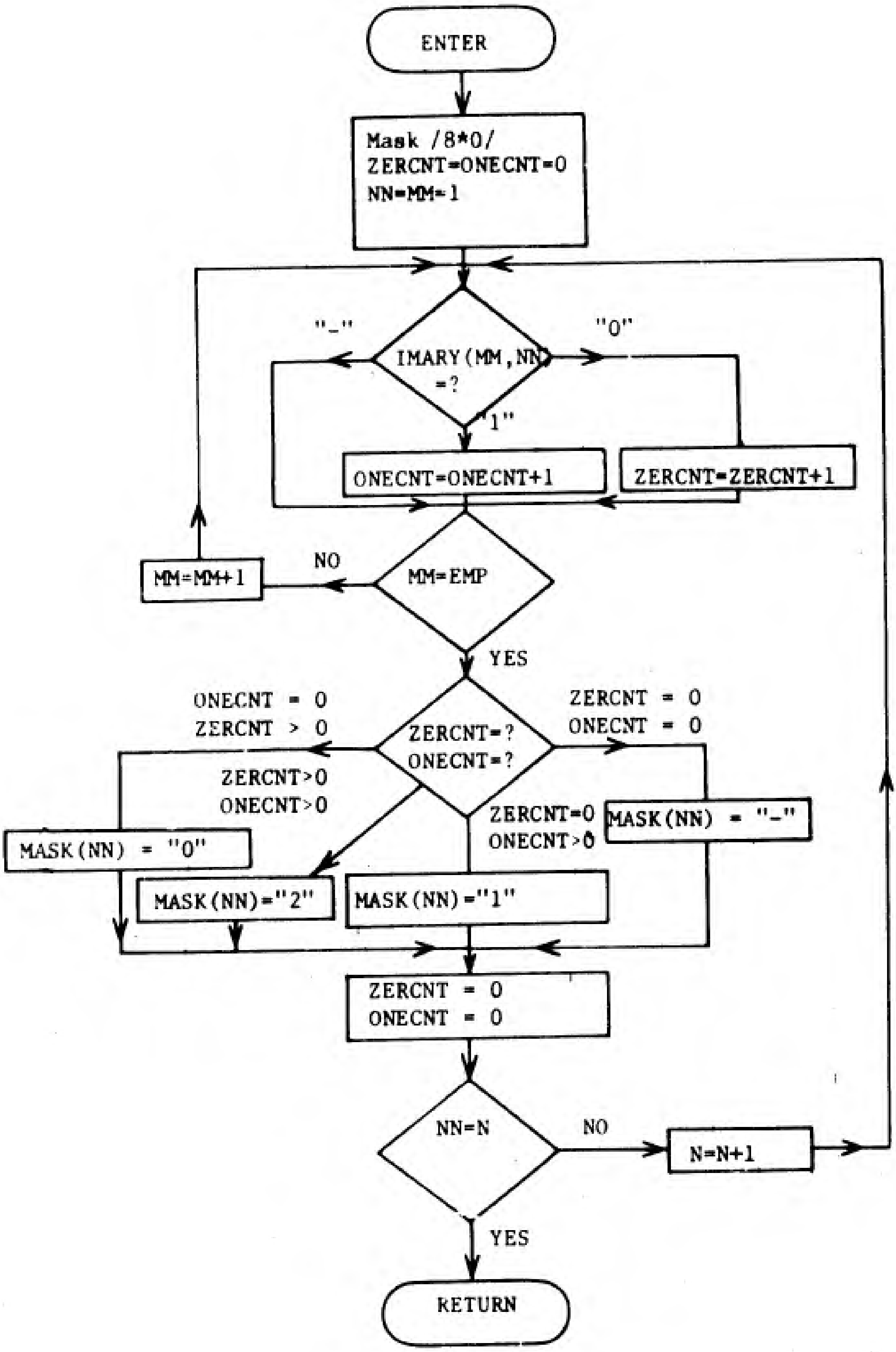


Figure 10. Subroutine IMASK.

Step 13: All the terms not covered by essential P.I. table are found in this step. This is done by selecting a minterm; if the corresponding term in vertex array is a "-1", the minterm has already been covered; if not, all the terms in P.I. table that cover this minterm are found. Each term is a constraint term. (Fig. 11)

Step 14: The "weight" of each constraint term is determined. Three factors are taken into consideration.

(a) The number of uncovered minterms, covered by the constraint term: Each minterm whose corresponding vertex element is not "-1" is compared with the constraint term for containment. Thus, the number of uncovered minterms covered by this term are determined. [Fig. 12]

(b) The number of literals: The binary array representing each constraint term is scanned to find the total number of "1" and "0" in the array. If a "literal" is absent it is a "-" in the binary array. [Fig. 12]

(c) The number of matching literals: Each constraint term is compared with "Mask" element by element.

Constraint Element	Mask Element	Count
1 or 0	2	1
1	1	1
0	0	1
-	-	1

Only the cases when the matching count is incremented are shown in the above table. [Fig. 14]

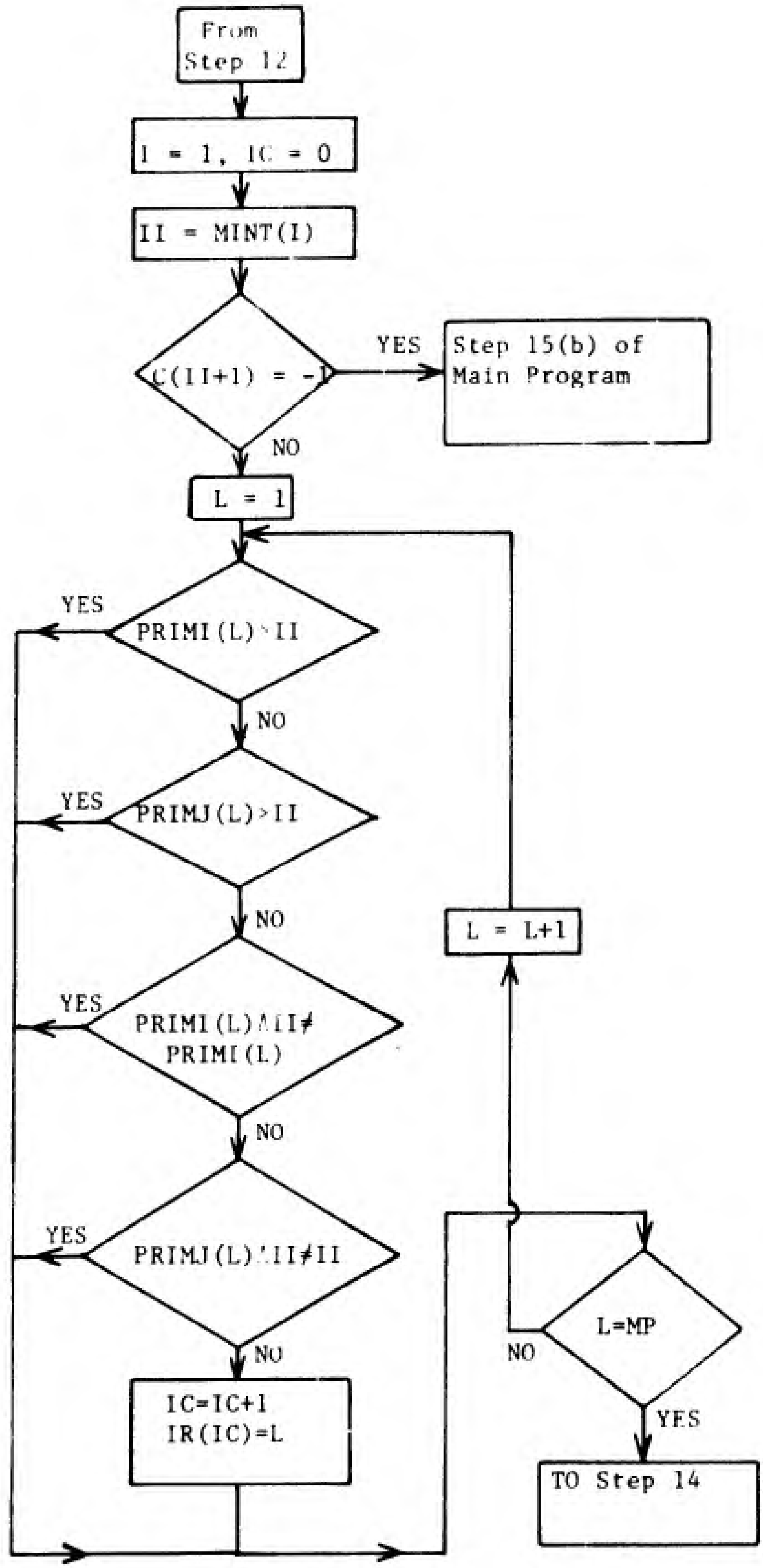


Figure 11. Find the constraint terms for II.

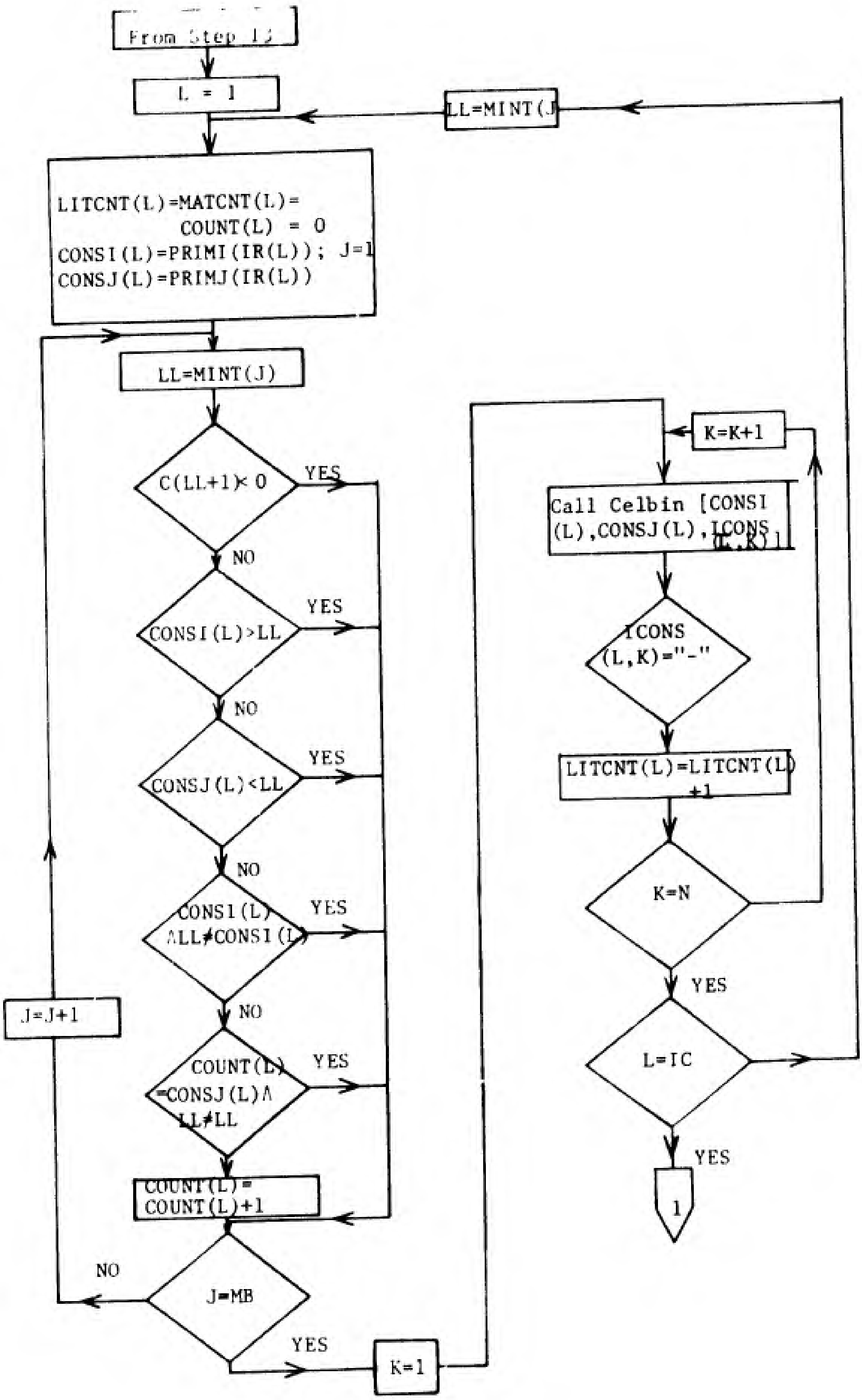


Figure 12. Weights for constraint terms.

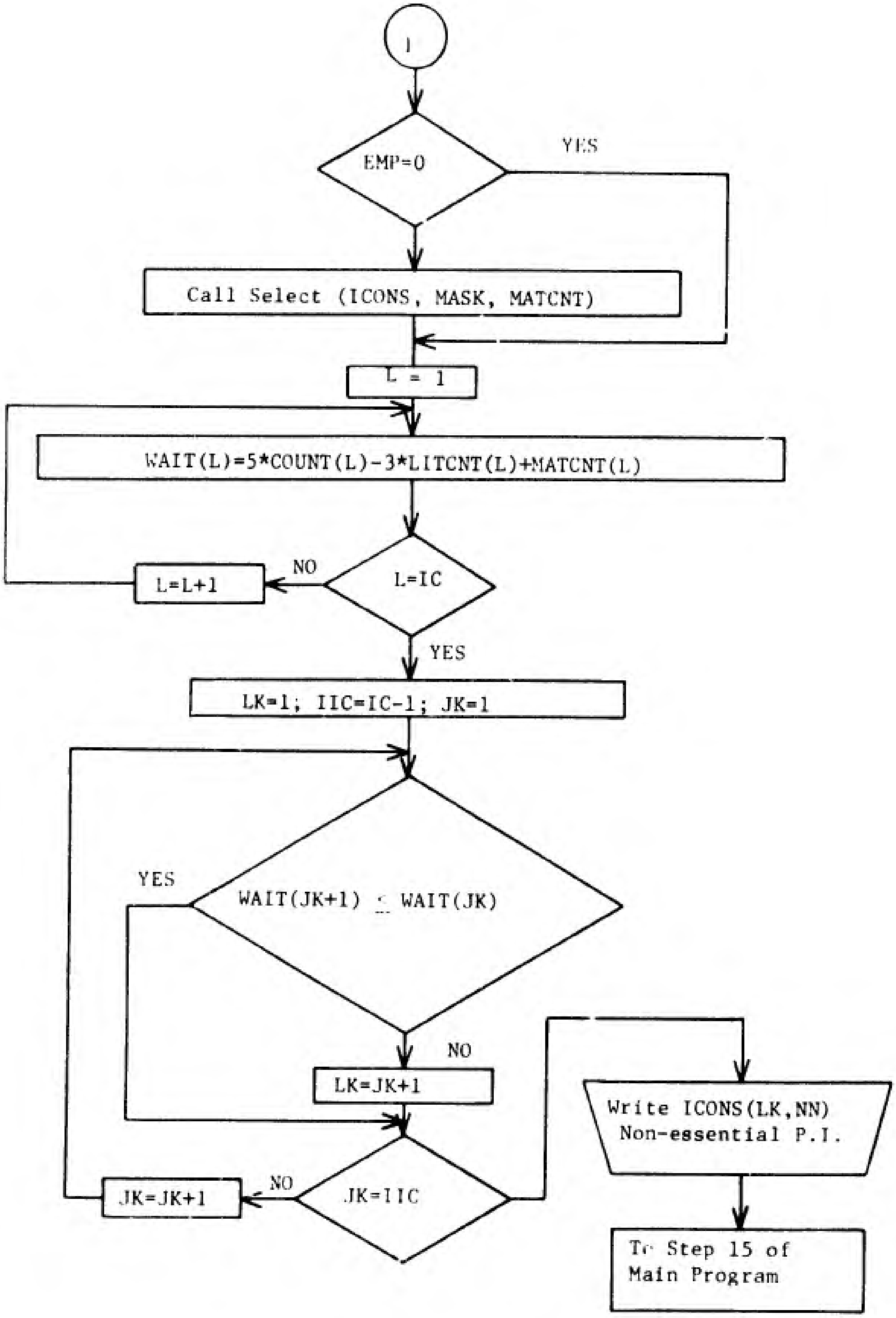


Figure 12 (con't). Weights for constraint terms.

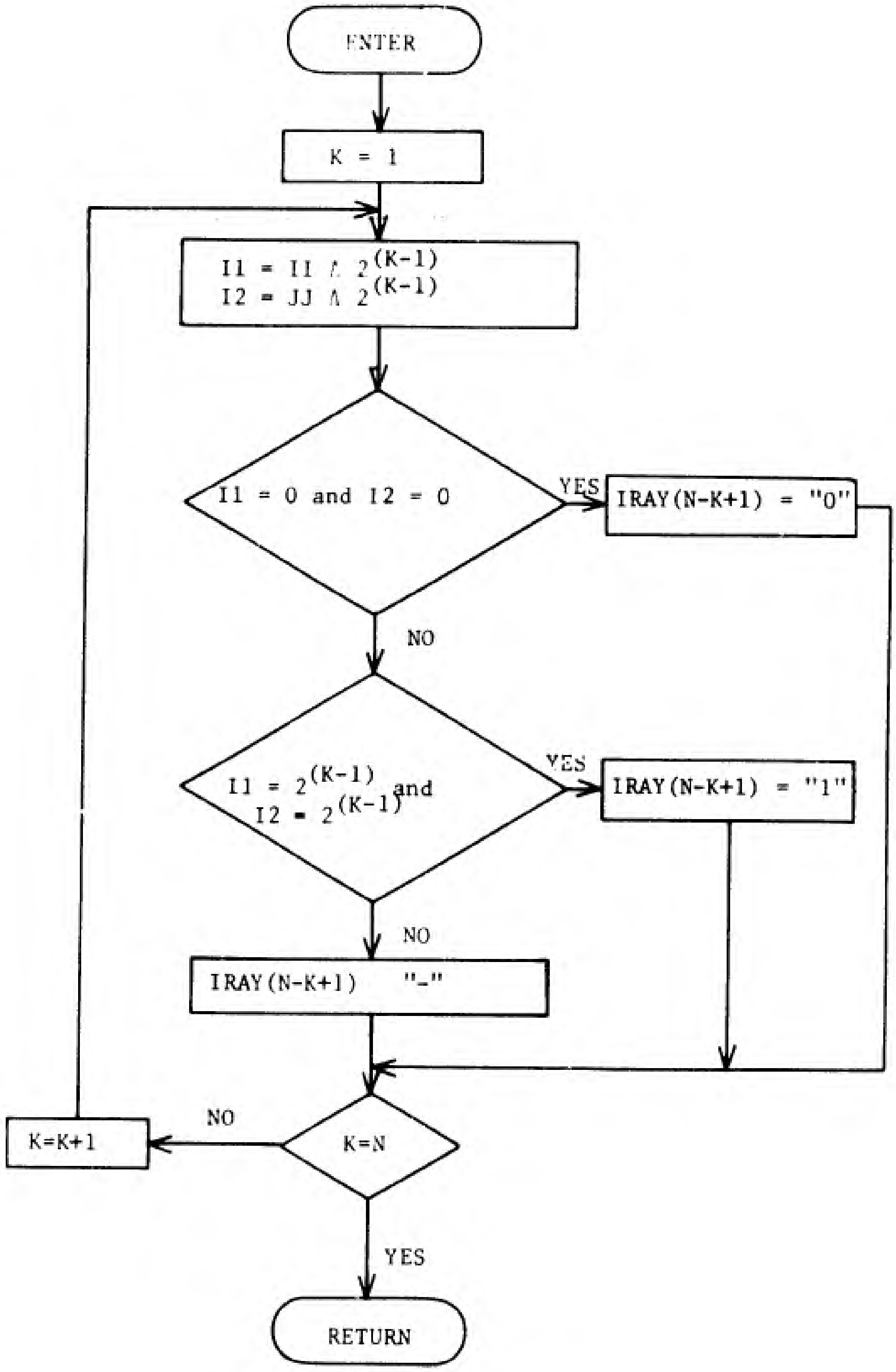


Figure 13. Subroutine CELBIN, to convert (II,JJ) to IRAY in cellular (0,1,-) format.

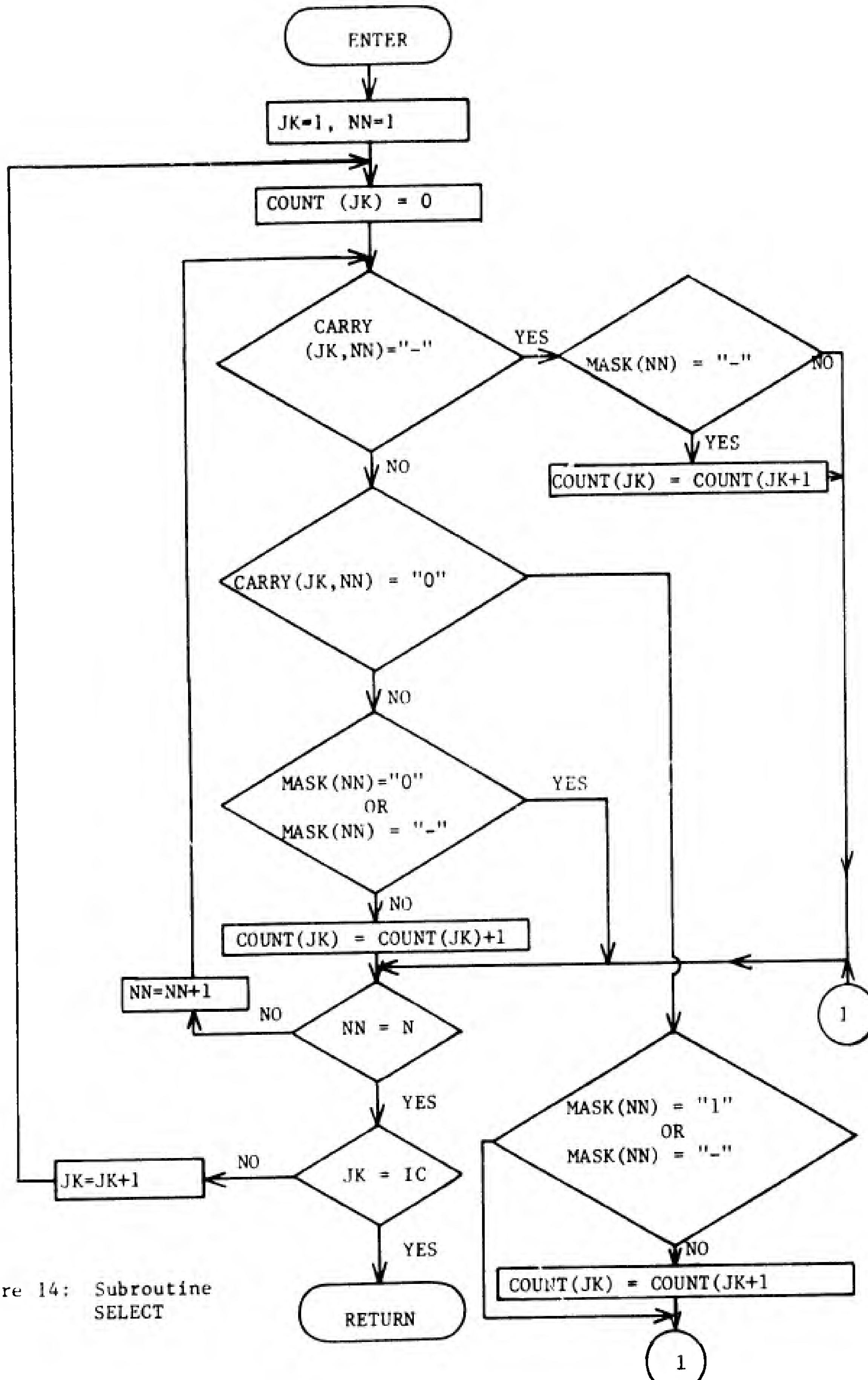


Figure 14: Subroutine SELECT

Total weight: Total weight for each constraint term is calculated as:

weight = K_1(No. of uncovered minterms covered) + K_2 (No. of literal + K_3 (No. of matching literals).

Constants K_1, K_2, and K_3 can be varied as desired by the designer who is minimizing the function. The maximum weighted term is a "non-essential P.I.". In this program $K_1 = 5$, $K_2 = -3$, $K_3 = 1$. (Fig. 12b)

Step 15: The terms in the vertex array corresponding to minterms covered by the non-essential P.I. are set to "-1". This is accomplished by a subroutine RESETC (Fig. 9). This process is repeated with all the uncovered minterms.

This completes the minimization of one function.

Step 16: Here "ID" is decremented by "2". If the new "ID" is zero, the process is repeated with the minterms of the complemented function. If the original "ID" were either "1" or "0", this repetition is not necessary as can be seen by the convention associated with "ID".

IV. EXAMPLE RESULTS

The general flow of the algorithm can be summarized as follows:

As each minterm pair is selected, the program checks:

(A) Each coordinate pair:

(a) Is the coordinate pair a cell?

(b) Is the cell a cell of the function?

(c) Is the cell (if in the function) a Prime Implicant?

(d) Is it an essential Prime Implicant?

(B) Each minterm:

(a) Is the term a don't care?

(b) If not, has it been covered by an essential Prime Implicant?

Then, the program determines

(a) All the minterms not covered by essential Prime Implicants.

(b) All the P.I.'s covering each of these minterms.

and selects,

the non-essential prime implicant to cover each of these terms.

If the "complementary" function is also to be minimized, the algorithm is repeated with same don't cares and complementary minterms.

Three example results are given.

Example 1:

This is a fourth order function with the minterms and don't cares shown in the printout. There are five Prime Implicants printed in both cellular and binary formats. The following table shows the Prime Implicants and the minterm covering.

Example 1

```
THE BOOLEAN FUNCTION IS OF ORDER      4
MINTERMS
   0    2    4    5    6    9   10
DONTCARES
   7   11   12   13   14   15
PRIME IMPLICANTS
                 (    0,    6)                 0--0
                 (    2,   14)                 --10
                 (    4,   15)                 -1--
                 (    9,   15)                 1--1
                 (   13,   15)                 1-1-



                 THE ESSENTIAL P.I. ARE
                                               0--0
                                               -1--
                                               1--1
                 NON ESSENTIAL P.I. ARE
                                               --10




*********** MINIMIZED COMPLEMENTARY FUNCTION ******
MINTERMS
   0    2    4    5    6    9   10
DONTCARES
   7   11   12   13   14   15
PRIME IMPLICANTS
                 (    1,    3)                 00-1
                 (    3,   15)                 --11
                 (    8,   12)                 1-00
                 (   12,   15)                 11--



                 THE ESSENTIAL P.I. ARE
                                               00-1
                                               1-00
                 NON ESSENTIAL P.I. ARE
                                               *** NONE ***
```

PRIME IMPLICANT	MINTERMS							DON'T CARES					
	0	2	4	5	6	9	10	7	11	12	13	14	15
(0,6)	X	X	X		X								
(4,15)			X	X	X			X		X	X	X	X
(9,15)						X			X		X		X
(2,14)		X			X		X					X	
(10,15)							X		X			X	X

It can be seen from the table that, (0,6), (4,15) and (9,15) are the Essential Prime Implicants. Minterm "10" is not covered by these Essential Prime Implicants. Both (2,14) and (10,15) cover minterm 10. Comparing these with the Essential Prime Implicants, both have same number of matching literals (2) and both have the same number of literals (2). So, both have the same weight. The first one is selected as non-essential Prime Implicant.

The Printout shows the minimized complementary function whose min-terms are (1,3,8) and same don't cares. There are four Prime Implicants out of which two are found to be essential. Since these two cover all the minterms, no non-essential Prime Implicant is needed.

This example illustrates all the features of the program. The Printouts for two more functions are included.

The algorithm is fast and flexible. It suits any automated logic design procedure. Following table shows the execution time on system 360/50 for different functions in the examples.

ORDER OF THE FUNCTION	NUMBER OF MINTERMS	NUMBER OF DON'T CARES	TIME (SECONDS) TO MINIMIZE		
			F	$\overline{F}$	F and $\overline{F}$
4	7	6	1.27	1.09	1.34
6	27	11	2.17	2.29	4.10
8	45	157	31.90	39.39	69.84

Total length of the Program = 46 K bytes of core memory?

Example 2

```
THE BOOLEAN FUNCTION IS OF ORDER     6
MINTERMS
   3   7  12  14  15  19  27  28  29  31  35  39  44  45  46  48  49  50  52  53
  55  56  57  59  60  62  63
DONTCARES
   0  11  13  23  30  32  43  47  51  54  61
PRIME IMPLICANTS
              (   0,  32)            -00000
              (   3,  63)            ----11
              (  12,  63)            --11--
              (  32,  48)            1-0000
              (  48,  61)            11--0-
              (  48,  55)            110---
              (  49,  53)            11---1
              (  52,  53)            11-1--


          THE ESSENTIAL P.I. ARE
                                      ----11
                                      --11--
                                      110---
                                      11--0-
          NON ESSENTIAL P.I. ARE
                                      *** NONE ***
```

Example 2 (continued)

```
************ MINIMIZED COMPLEMENTARY FUNCTION ******
MINTERMS
   3   7  12  14  15  19  27  28  29  31  35  39  44  45  46  48  49  50  52  53
  55  56  57  59  60  62  63
DONTCARES
   0  11  13  23  30  32  43  47  51  54  61
PRIME IMPLICANTS
                (    0,  42)                     -0-0-0
                (    0,  41)                     -0-00-
                (    0,  38)                     -00--0
                (    0,  37)                     -00-0-
                (    0,  26)                     0--0-0
                (    0,  25)                     0--00-
                (    0,  22)                     0-0--0
                (    0,  21)                     0-0-0-
                (    1,  13)                     00--01
                (    5,  54)                     --0110
                (    8,  43)                     -010--
                (   10,  58)                     --1010
                (   18,  30)                     01--10
                (   20,  23)                     0101--
                (   43,  47)                     101-11
                (   51,  51)                     110011
                (   51,  51)                     111101



                THE ESSENTIAL P.I. ARE
                                                 0--00-
                                                 -00-0-
                                                 --1010
                NON ESSENTIAL P.I. ARE
                                                 0-0--0
                                                 0-0-0-
                                                 -0-0-0
                                                 -00--0
                                                 -0-00-
```

Example 3

```
THE BOOLEAN FUNCTION IS OF ORDER      8
MINTERMS
  17  20  21  23  25  32  34  35  38  39  48  49  53  54  64  65  66  70  71  72
  73  84  85  86  87  98  99 100 101 102 114 115 116 117 118 119 132 133 134 135
 136 137 151 152 153
DONTCARES
   0  10  11  12  13  14  15  26  27  28  29  30  31  42  43  44  45  46  47  58
  59  60  61  62  63  74  75  76  77  78  79  90  91  92  93  94  95 106 107 108
 109 110 111 122 123 124 125 126 127 138 139 140 141 142 143 154 155 156 157 158
 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178
 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198
 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218
 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238
 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 239
PRIME IMPLICANTS
          (   0,  64)            0-000000
          (   0,  32)            00-00000
          (  10, 255)            ----1-1-
          (  12, 255)            ----11--
          (  17,  53)            00-10-01
          (  17,  29)            0001--01
          (  20,  93)            0-01-10-
          (  21, 125)            0--1-101
          (  21,  95)            0-01-1-1
          (  23, 223)            --01-111
          (  25, 159)            -0011--1
          (  32, 175)            -01-0000
          (  32, 162)            -01000-0
          (  34, 238)            --10--10
          (  34, 235)            --10-01-
          (  34, 175)            -010--1-
          (  38, 254)            --1--110
          (  48, 177)            -011000-
          (  49, 181)            -0110-01
          (  53, 253)            --11-101
          (  64, 202)            -100-0-0
          (  64, 201)            -100-00-
          (  66, 238)            -1-0 -10
          (  70, 254)            -1---110
          (  70, 223)            -10--11-
          (  72, 207)            -1001---
          (  84, 255)            -1-1-1--
          (  98, 254)            -11---10
          (  98, 251)            -11--01-
          ( 100, 254)            -11--1-0
          ( 100, 253)            -11--10-
          ( 114, 255)            -111--1-
          ( 132, 239)            1--0-1--
          ( 135, 255)            1----111
          ( 136, 255)            1---1---

          ( 160, 255)            1-1-----
          ( 192, 255)            11------
```

```
THE ESSENTIAL P.I. ARE
                              0-01-10-
                              -010--1-
                              --1--110
                              -1000-00-
                              -10--11-
                              -11--10-
                              1--0-1--
                              1---1---
NON ESSENTIAL P.I. ARE
                              00-10-01
                              --01-111
                              -0011--1
                              -01-0000
                              -1-0--10
                              -11--01-
                              -1-1-1--
```

Example 3 (continued)

```
************ MINIMIZED COMPLEMENTARY FUNCTION ******
MINTERMS
  17  20  21  23  25  32  34  35  38  39  48  49  53  54  64  65  66  70  71  72
  73  84  85  86  87  98  99 100 101 102 114 115 116 117 118 119 132 133 134 135
 136 137 151 152 153
DONTCARES
   0  10  11  12  13  14  15  26  27  28  29  30  31  42  43  44  45  46  47  58
  59  60  61  62  63  74  75  76  77  78  79  90  91  92  93  94  95 106 107 108
 109 110 111 122 123 124 125 126 127 138 139 140 141 142 143 154 155 156 157 158
 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178
 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198
 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218
 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238
 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 239
PRIME IMPLICANTS
              (   0, 146)               -00-00-0
              (   0, 131)               -00000--
              (   0,  26)               000--0-0
              (   0,  15)               0000----
              (   1, 151)               -0-00001
              (   1,  45)               00-0--01
              (   2, 155)               -00--01-
              (   2,  30)               000---10
              (   3, 219)               --0--011
              (   4,  77)               0-00-10-
              (   4,  45)               00-0-10-
              (   8,  62)               00--1--0
              (   8,  47)               00-01---
              (  10, 255)               ----1-1-
              (  12, 255)               ----11--
              (  16, 210)               --0100-0
              (  16,  90)               0-01-0-0
              (  18, 219)               --01-01-
              (  18, 187)               -0-1-01-
              (  18, 158)               -001--10
              (  24, 126)               0--11--0
              (  33, 233)               --10-001
              (  33, 173)               -010--01
              (  36, 188)               -01--100
              (  36, 173)               -010-10-
              (  40, 255)               --1-1---
              (  51, 191)               -011--11
              (  68, 205)               -100-10-
              (  80, 249)               -1-1-00-
              (  80, 219)               -101-0--
              (  88, 255)               -1-11---
              (  96, 249)               -11--00-
              ( 103, 239)               -110-111
              ( 128, 243)               1---00--
              ( 130, 251)               1----01-
              ( 144, 246)               1--10--0
              ( 144, 245)               1--10-0-
              ( 146, 254)               1--1--10
              ( 148, 254)               1--1-1-0
              ( 148, 253)               1--1-10-
              ( 160, 255)               1-1-----
              ( 192, 255)               11------
```

Example 3 (continued)

```
THE ESSENTIAL P.I. ARE
                                        0000----
                                        -0-1-01-
                                        -01--100
                                        -011--11
                                        --1-1---
                                        --0--011
                                        -11--00-
                                        -110-111
NON ESSENTIAL P.I. ARE
                                        0-01-0-0
                                        -001--10
                                        -010--01
                                        0-00-10-
                                        -1-1-00-
                                        1---00--
                                        1--1-10-
```

Example 3 (continued)

V. Conclusion

The algorithm described in this report is basically a Quine-McCluskey type procedure. But, the containment properties of cellular n-cube representation greatly minimize the number of comparisons. The ordering of minterms and don't cares (lowest first, highest last) helps in finding the largest Prime Implicant first, thus avoiding the possibility of entering a P. I. into the table, which may be found later to be covered by a new cell.

The weights associated with the number of uncovered minterms, the number of literals and the number of matching literals can be varied as desired, to select the most suitable non-essential P. I.

The algorithm can also minimize the complementary function whereever such minimization is needed.

REFERENCES

1. C. C. Carroll and G. E. Jordan, "A Fast Algorithm for Boolean Function Minimization," Project THEMIS Tech. Report No. AU-T-3, Dec. 1968.

2. C. C. Carroll and W. A. Hornfeck, "An Algorithm for Fast Boolean Function Minimization Using Properties of Cellular N-cube," Project THEMIS Tech Report No. AU-T-16, Aug. 70.

3. R. E. Prather, *Introduction to Switching Theory: A Mathematical Approach*, Boston, Allyn & Bacon, 1967.

4. H. Mott and C. C. Carroll, "Numerical Procedures for Boolean Function Minimization," *IEEETEC*, Aug. 1964.

APPENDIX I

FORTRAN source program listing of the Boolean function minimization program described in this report.

```
      C     N IS THE ORDER OF BOOLEAN FUNCTION.
      C     M=NO. OF MINTERMS, D= NO. OF DONTCARES
      C     MIN  -- ARRAY OF  M  MINTERMS
      C     DONT -- ARRAY OF  D  DONTCARES
      C     MINT -- COMBINED ARRAY OF (M+D) TERMS
      C     EMP  -- NO. OF ESSENTIAL P.I.S
      C     MP   -- NO. OF P.I.S
      C     IC   -- NO. OF CONSTRAINT TERMS FOR EACH UNCOVERED TERM
      C     C    -- ARRAY OF LENGTH 2**N
      C     LITCNT  -- NO. OF LITERALS IN CONSTRAINT TERM
      C     COUNT   -- NO. OF UNCOVERED TERMS COVERED BY CONSTRAINT TERM
      C     MATCNT   -- NO. OF MATCHING LITERALS
      C     WAIT   --- TOTAL WEIGHT OF CONSTRAINT TERM
      C     PRIME IMPLICANTS.....
      C         LOW--PRIMI,    HIGH---PRIMJ
      C     ESSENTIAL PRIME IMPLICANTS....
      C         LOW--ESENI,   HIGH--ESENJ
      C         EACH ROW OF 'ESARY' CORR. TO ONE ESSENTIAL P.I.
      C     CONSTRAINT TERMS.....
      C         LOW--CONSI,   HIGH--CONSJ
      C         EACH ROW OF 'ICONS'  CORR. TO ONE CONSTRAINT TERM
      C
      C******************
0001          DIMENSION IRAY(8),ICONS(10,8),LITCNT(10)
0002          DIMENSION NUM(200),IR(200),MINAX(256)
0003          INTEGER C(256),D,MIN(256),DONT(256),MINT(256),EMP
0004          INTEGER DIF,PRIMI(256),PRIMJ(256),ESENI(200),ESENJ(200)
0005          INTEGER CONSI(10),CONSJ(10),COUNT(10)
0006          INTEGER ESARY(100,8),MASK(8),MATCNT(10)
0007          INTEGER ONE,ZERO,DASH,WAIT(10)
0008          DATA DASH/1H-/
0009          COMMON N,EMP,IC
0010          COMMON/COM2/C
      C******************** DATA CARDS  ***********
      C     FIRST DATA CARD HAS N,D,ID ON IT IN FORMAT(3I1)
      C*****************************************************
      C      D   MINTERMS    DONTCARES
      C*****************************************************
      C      0   BINARY      NONE
      C      1   BINARY      BINARY
      C      2   INTEGER     INTEGER
      C      3   BINARY      INTEGER
      C      4   INTEGER     BINARY
      C      5   INTEGER     NONE
      C*****************************************************
      C      ID      FUNCTION TO BE MINIMIMIZED
      C*****************************************************
      C       0            COMPLEMENTED FUNCTION
      C       1            TRUE FUNCTION
      C       2             BOTH
      C*****************************************************
      C     WHEN DATA IS  IN INTEGER FORM, THE CARD BEFORE SHOULD
      C       HAVE THE NO. OF TERMS IN FORMAT(I4)
      C       DATA IS IN FORMAT(20I4) ON EACH CARD,  CONTINUE ON NEXT CARD
      C       IF EXEEDS 20 TERMS.
      C     WHEN DATA IS IN BINARY FORM, NO SPECIAL CARD IS NEEDED.
      C      A '+' SEPARATES EACH TERM
      C******************
```

```
C      READ N, MINTERMS AND DONTCARES
C

             READ(5,10)N,D,ID
   10        FORMAT(3I1)
             IF(D.EQ.1)GO TO 121
             IF(D.EQ.2)GO TO 122
             IF(D.EQ.3)GO TO 123
             IF(D.EQ.4)GO TO 124
             IF(D.EQ.5)GO TO 125
C
C   IF D=0,  NO DONTCARES  MINTERMS IN BINARY(0,1,-) FORM
             CALL REED(MIN,M)
             GO TO 5
C
C   IF D=1, MINTERMS AND DONTCARES IN BINARY FORM(0,1,-)
  121        CALL REED(MIN,M)
             CALL REED(DONT,D)
             GO TO 5
C
C   IF D=2, MINTERMS AND DONTCARES IN INTEGER FORMAT
  122        READ(5,126)M
  126        FORMAT(I4)
             READ(5,127)(MIN(I),I=1,M)
  127        FORMAT(20I4)
             READ(5,126)D
             READ(5,127)(DONT(I),I=1,D)
             GO TO 5
C
C   IF D=3, MINTERMS IN BINARY,  DONTCARES IN INTEGER
  123        CALL REED(MIN,M)
             READ(5,126)D
             READ(5,127)(DONT(I),I=1,D)
             GO TO 5
C
C   IF D=4, MINTERMS IN INTEGER FORM,  DONTCARES IN  BINARY
  124        READ(5,126)M
             READ(5,127)(MIN(I),I=1,M)
             CALL REED(DONT,D)
             GO TO 5
C   IF D=5,  MINTERMS IN INTEGER FORMAT,  NO DONTCARES.
  125        READ(5,126)M
             READ(5,127)(MIN(I),I=1,M)
             D=0
C
C      CONVERT MINTERMS AND DONTCARES INTO A SINGLE ARRAY        ***
C      IN INCREASING ORDER OF THEIR DECIMAL REPRESENTATION
C
    5        MM=2**N
             K=0
             WRITE(6,30)N
   30        FORMAT(' ','THE BOOLEAN FUNCTION IS OF ORDER ',I4)
             IF(IC.GE.1)GO TO 805
   20        K=0
             DO 809 I=1,MM
             MINAX(I)=I-1
             DO 811 J=1,M
             IF(MINAX(I).EQ.MIN(J))GO TO 809
```

```
      811 CONTINUE
          K=K+1
          MINT(K)=MINAX(I)
      809 CONTINUE
          MB=K
          WRITE(6,810)
      810 FORMAT('1*********** MINIMIZED COMPLEMENTARY FUNCTION ******')
          GO TO 815
      805 DO 15 I=1,M
          MINT(I)=MIN(I)
       15 CONTINUE
          MA=M+1
          MB=M+D
          IF(D.EQ.0) GO TO 6
          DO 25 I=MA,MB
       25 MINT(I)=DONT(I-M)
        6 DO 35 N1=1,MB
          DO 35 N2=N1,MB
          IF(MINT(N1).LT.MINT(N2)) GO TO 35
          MINTS=MINT(N1)
          MINT(N1)=MINT(N2)
          MINT(N2)=MINTS
       35 CONTINUE
C
C     PRINT MINTERMS AND DONTCARES
C
      815 WRITE(6,40)(MIN(I),I=1,M)
       40 FORMAT(' ','MINTERMS'/(' ',20I4))
          IF(D.EQ.0) GO TO 7
          WRITE(6,50)(DONT(I),I=1,D)
       50 FORMAT(' ','DONTCARES'/(' ',20I4))
C
C
C     SET VERTICES IN  C  CORRESPONDING TO MINTRMS =1, DONTCARES=-1,
C     AND ALL OTHERS TO ZEROS
C
        7 J=1
          K=1
          MM=2**N
          DO 333 I=1,MM
          IF(J.GT.MB) GO TO 1
          IF((I-1).EQ.MINT(J)) GO TO 3
        1 C(I)=0
          GO TO 333
        3 IF(K.GT.D)GO TO 9
          IF(MINT(J).NE.DONT(K)) GO TO 9
          C(I)=-1
          J=J+1
          K=K+1
          GO TO 333
        9 C(I)=1
          J=J+1
      333 CONTINUE
C
C     CHOOSE A COORDINATE PAIR    ****
C     II IS SMALLER AND  JJ IS LARGER
C
          MP=0
```

```
0098            MP=0
0099            WRITE(6,555)
0100      555   FORMAT(' ','PRIME IMPLICANTS')
0101            DO 200 I=1,MB
0102            II=MINT(I)
0103            M1=MB-I+1
0104            DO 300 J=1,M1
0105            J1=MB-J+1
0106            JJ=MINT(J1)
          C
          C     CHECK TO SEE THE COORDINATE PAIR IS A CELL
          C
0107            IF(IAND(II,JJ).NE.II) GO TO 300
0108            DIF=JJ-II
0109            IF(DIF.EQ.0) GO TO 700
          C
          C     CALCULATE VERTICES OF THE CELL
          C     BETWEEN II  AND  JJ
          C
0110            ITEMP=1
0111            ICNT=1
0112            NUM(1)=II
0113            DO 400 K=1,N
0114            IZ=IAND((2**(K-1)),DIF)
0115            IF(IZ.EQ.0)GO TO 400
0116            IC2=ICNT
0117            DO 600 L=1,IC2
0118            ICNT=ICNT+1
0119            NUM(ICNT)=NUM(L)+IZ
0120            LL=NUM(ICNT)+1
0121            ITEMP=IAND(C(LL),ITEMP)
          C
          C    CHECK CELL TO SEE IF IT IS IN FUNCTION
          C
0122            IF(ITEMP.EQ.0)GO TO 300
0123        600 CONTINUE
0124        400 CONTINUE
          C
          C    CHECK CELL TO SEE IF IT IS COVERED BY ONE IN THE P.I.TABLE
          C
0125        700 IF(MP.EQ.0) GO TO 800
0126            DO 900 L=1,MP
0127            IF(PRIMJ(L).LT.JJ)GO TO 900
0128            IF(IAND(PRIMJ(L),JJ).NE.JJ) GO TO 900
0129            IF(IAND(PRIMI(L),II).EQ.PRIMI(L)) GO TO 300
0130        900 CONTINUE
0131        800 MP=MP+1
0132            PRIMI(MP)=II
0133            PRIMJ(MP)=JJ
          C
          C    WRITE P.I. ******
          C
0134            WRITE(6,70)II,JJ
0135       70   FORMAT(' ',10X,'(',I4,',',I4,')')
0136            CALL CELBIN(II,JJ,IRAY)
0137            WRITE(6,996)(IRAY(L),L=1,N)
0138        300 CONTINUE
          C
```

```
            C      CHECK IF II IS INCLUDED IN AN ESSENTIAL TERM
            C
0139                 IF(C(II+1).LT.0) GO TO 200
            C
            CHECK FOR THE NO. OF P.I.S THAT COVER  II
            C
0140                 J=0
0141                 IC=0
0142                 DO 11 L=1,MP
0143                 IF(PRIMJ(L).LT.II) GO TO 11
0144                 IF(IAND(PRIMI(L),II).NE.PRIMI(L))GO TO 11
0145                 IF(IAND(PRIMJ(L),II).NE.II) GO TO 11
0146                 IC=IC+1
0147                 IIESS=L
0148            11   CONTINUE
            C
            C      IC  IS THE NO. OF P.I.S THAT COVER  II. IF  IC=1
            C      AN ESSENTIAL P.I. IS DISCOVERED
            C
0149                 IF(IC.GT.1) GO TO 200
0150                 EMP=EMP+1
0151                 ESENI(EMP)=PRIMI(IIESS)
0152                 ESENJ(EMP)=PRIMJ(IIESS)
            C
            CHECK OFF ALL THE TERMS COVERED BY THIS ESSENTIAL P.I.
            C
0153                 CALL RESETC(ESENI(EMP),ESENJ(EMP))
0154           200 CONTINUE
            C
            C      PRINT ESSENTIAL P.I.
            C
0155                 IF(EMP.EQ.0) GO TO 111
0156                 WRITE(6,80)
0157            80   FORMAT('1',10X,'THE ESSENTIAL P.I. ARE')
0158           112 DO 113 I=1,EMP
            C
            CONVERT ESSENTIAL P.I  INTO ROWS OF MATRIX ESARY
            C
0159                 CALL CELBIN(ESENI(I),ESENJ(I),IRAY)
0160                 WRITE(6,999)(IRAY(L),L=1,N)
0161                 DO 115 J=1,N
0162           115 ESARY(I,J)=IRAY(J)
0163           113 CONTINUE
            C
            C     FORM MASK
            C
0164                 CALL  IMASK(ESARY,MASK)
0165                 GO TO 114
            C
            C        IF THERE ARE NO ESSENTIAL P.I.S
            C
0166           111 WRITE(6,90)
0167           90  FORMAT(' ','THERE ARE NO ESSENTIAL P.I. ')
            C
            C        FIND ALL MINTERMS NOT INCLUDED IN ESSENTIAL P.I.
            C
0168          114  WRITE(6,110)
0169           110 FORMAT(' ',10X,'NON ESSENTIAL P.I. ARE')
```

```
            MESS=0
            DO 2000 I=1,MB
            II=MINT(I)
            IC=0
      C
      C     IF C  CORR. TO II  IS -1  II HAS ALREADY BEEN COVERED
      C
            IF(C(II+1).LT.0) GO TO 2000
      C
      C     IF  C  IS NOT EQ. -1 , FIND WHAT P.I.S COVER THIS TERM
      C     THESE ARE CONSTRAINT TERMS
      C
            DO 1000 L=1,MP
            IF(PRIMI(L).GT.II) GO TO 1000
            IF(PRIMJ(L).LT.II)GO TO 1000
            IF(IAND(PRIMI(L),II).NE.PRIMI(L))GO TO 1000
            IF(IAND(PRIMJ(L),II).NE.II) GO TO 1000
            IC=IC+1
            IR(IC)=L
       1000 CONTINUE
            DO 222 L=1,IC
            CONSI(L)=PRIMI(IR(L))
            CONSJ(L)=PRIMJ(IR(L))
      C
      CHECK HOW MANY UNCOVERED MINTERMS ARE COVERED BY EACH CONSTRAINT TERM
      C
            COUNT(L)=0
            DO 66 J=1,MB
            LL=MINT(J)
            IF(C(LL+1).LT.0)GO TO 66
            IF(CONSI(L).GT.LL)GO TO 66
            IF(CONSJ(L).LT.LL)GO TO 66
            IF(IAND(CONSI(L),LL).NE.CONSI(L))GO TO 66
            IF(IAND(CONSJ(L),LL).NE.LL)GO TO 66
            COUNT(L)=COUNT(L)+1
         66 CONTINUE
      C
      CHECK HOW MANY LITERALS EACH CONSTRAINT TERM HAS
      C
            LITCNT(L)=0
            CALL CELBIN(CONSI(L),CONSJ(L),IRAY)
            DO 522 K=1,N
            ICONS(L,K)=IRAY(K)
            IF(ICONS(L,K).EQ.DASH)GO TO 522
            LITCNT(L)=LITCNT(L)+1
        522 CONTINUE
            MATCNT(L)=0
        222 CONTINUE
      C
      COMPARE CONSTRAINT ARRAY WITH MASK FOR MATCHING LITERALS
      C
            IF(EMP.EQ.0)GO TO 76
            CALL SELECT(ICONS,MASK,MATCNT)
         76 DO 8 L=1,IC
          8 WAIT(L)=5*COUNT(L)-3*LITCNT(L)+MATCNT(L)
      C
      CHOOSE MAXIMUM WEIGHTED CONSTRAINT TERM
      C
```

```
0209             LK=1
0210             IIC=IC-1
0211             DO 4 JK=1,IIC
0212             IF(WAIT(JK+1).LE.WAIT(JK))GO TO 4
0213             LK=JK+1
0214       4     CONTINUE
0215             WRITE(6,999)(ICONS(LK,NN),NN=1,N)
0216     996     FORMAT('+',35X,8A1)
0217     999     FORMAT(' ',35X,8A1)
0218             MESS=1
           C
           C     SET VERTICES CORRESPONDING TO THE SELECTED TERM TO -1 IN C
           C
0219             CALL RESETC(CONSI(LK),CONSJ(LK))
0220    2000     CONTINUE
0221             IF(MESS.NE.0)GO TO 42
0222             WRITE(6,41)
0223      41     FORMAT(' ',35X,'*** NONE ***')
0224      42     IC=IC-2
0225             IF(IC.EQ.0)GO TO 20
0226
0227             STOP
0228             END
```

```
      SUBROUTINE CELBIN(II,JJ,IRAY)
C
C
C     THIS CONVERTS II  AND  JJ INTO AN ARRAY
C     IN BINARY FORM CONTAINING 1,0,-
C
C
      COMMON N
      DIMENSION IRAY(8)
      INTEGER ONE,ZERO,DASH
      DATA ONE,ZERO,DASH/1H1,1H0,1H-/
      DO 997 K=1,N
      I1=IAND(II,2**(K-1))
      I2=IAND(JJ,2**(K-1))
      IF(I1.EQ.0.AND.I2.EQ.0)GO TO 999
      IF(I1.EQ.(2**(K-1)).AND.I2.EQ.(2**(K-1)))GO TO 998
      IRAY(N-K+1)=DASH
      GO TO 997
  999 IRAY(N-K+1)=ZERO
      GO TO 997
  998 IRAY(N-K+1)=ONE
  997 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE IMASK(IMARY,MASK)
C
C
C     THIS SCANS THROUGH THE MATRIX IMARY EACH ROW OF WHICH
C     CORRESPONDS TO A P.I.) AND FORMS A  MASK
C
C
      DIMENSION IMARY(100,8)
      INTEGER ZERCNT,ONECNT,ONE,ZERO,DASH,TWO,MASK(8),EMP
      COMMON N,EMP,IC
      DATA ONE,ZERO,DASH,TWO/1H1,1H0,1H-,1H2/
      DATA ZERCNT,ONECNT/2*0/
      DO 15 NN=1,N
      DO 1 MM=1,EMP
      IF(IMARY(MM,NN).EQ.DASH)GO TO 1
      IF(IMARY(MM,NN).EQ.ZERO)GO TO 2
      IF(IMARY(MM,NN).EQ.ONE)GO TO 3
    2 ZERCNT=ZERCNT+1
      GO TO 1
    3 ONECNT=ONECNT+1
    1 CONTINUE
      IF(ZERCNT.EQ.0.AND.ONECNT.EQ.0)GO TO 10
      IF(ZERCNT.EQ.0.AND.ONECNT.GT.0)GO TO 11
      IF(ZERCNT.GT.0.AND.ONECNT.GT.0)GO TO 12
C     MASK(NN)=0, IF THE LITERAL EXISTS ONLY COMPLEMENTED.
      IF(ZERCNT.GT.0.AND.ONECNT.EQ.0)MASK(NN)=ZERO
      GO TO 13
C     AN ELEMENT OF MASK IS -, IF THE LITERAL DOES NOT
C     EXIST IN ANY ESSENTIAL P.I.
   10 MASK(NN)=DASH
      GO TO 13
C     MASK(NN)=1 , IF THE LITERAL EXISTS ONLY UNCOMPLEMENTED
   11 MASK(NN)=ONE
      GO TO 13
C     MASK(NN)=2, IF THE LITERAL EXISTS IN BOTH COMPLEMENTED
C     AND UNCOMPLEMENTED FORMS
   12 MASK(NN)=TWO
   13 ZERCNT=0
      ONECNT=0
   15 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE RESETC(II,JJ)
C
C
C     THIS GENERATES THE VERTICES BETWEEN II AND JJ
C     AND SETS THE CORR. TERMS IN C  TO -1
C
C
      INTEGER NUM(200),C(256),DIF
      COMMON N,EMP,IC
      COMMON/COM2/C
C     GENERATE VERTICES BETWEEN  II  AND  JJ
      NUM(1)=II
      PRISS=NUM(1)+1
      C(PRISS)=-1
      ICNT=1
      DIF=JJ-II
      IF(DIF.EQ.0)GO TO 200
      DO 13 K=1,N
      IZ=IAND((2**(K-1)),DIF)
      IF(IZ.EQ.0)GO TO 13
      ICT2=ICNT
      DO 14 L=1,ICT2
      ICNT=ICNT+1
      NUM(ICNT)=NUM(L)+IZ
      LL=NUM(ICNT)+1
C     SET THE TERM IN C CORR. TO THE VERTEX TO -1
   14 C(LL)=-1
   13 CONTINUE
  200 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE SELECT(CARRY,MASK,COUNT)
C
C
C     THIS COMPARES EACH ROW OF CARRY WITH MASK.
C     COUNT IS THE NO. OF MATCHING LITERALS.
C
C
      INTEGER CARRY(10,8),MASK(8),COUNT(10)
      INTEGER ONE,ZERO,DASH,TWO
      COMMON N,EMP,IC
      DATA ONE,ZERO,DASH/1H1,1H0,1H-/
      DO 10 JK=1,IC
      COUNT(JK)=0
      DO 1 NN=1,N
      IF(CARRY(JK,NN).EQ.DASH)GO TO 11
      IF(CARRY(JK,NN).EQ.ZERO)GO TO 2
      IF(MASK(NN).EQ.ZERO.OR.MASK(NN).EQ.DASH)GO TO 1
      COUNT(JK)=COUNT(JK)+1
      GO TO 1
    2 IF(MASK(NN).EQ.ONE.OR.MASK(NN).EQ.DASH)GO TO 1
      COUNT(JK)=COUNT(JK)+1
      GO TO 1
   11 IF(MASK(NN).EQ.DASH)COUNT(JK)=COUNT(JK)+1
    1 CONTINUE
   10 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE REED(IOUT,K)
C
C
C     THIS ROUTINE READS THE DATA IN BINARY FORM, GENERATES ALL
C     THE MINTERMS AND ARRANGES IN INCREASING ORDER
C     K  IS THE NO. OF TERMS IN IOUT
C
C
      DIMENSION MINCEL(80),MMIN(8),MMAX(8),NUM(200)
      DIMENSION IMIN(300),IOUT(256)
      INTEGER ZERO,ONE,DASH,BLANK,PLUS,DIF
      DATA ZERO,DASH,BLANK,PLUS/1H0,1H-,1H ,1H+/
      COMMON N,EMP,IC
      J=1
      L=0
C     READ ONE DATA CARD
    1 READ(5,10)MINCEL
   10 FORMAT(80A1)
      DO 100 I=1,80
      IF(MINCEL(I).EQ.PLUS.OR.MINCEL(I).EQ.BLANK)GO TO 15
      IF(MINCEL(I).EQ.DASH)GO TO 5
      IF(MINCEL(I).EQ.ZERO)GO TO 2
      MMIN(J)=1
      MMAX(J)=1
      J=J+1
      GO TO 100
C     CHARACTER IS A ZERO
    2 MMIN(J)=0
      MMAX(J)=0
      J=J+1
      GO TO 100
C     CHARACTER IS A DASH.GENERATE MINIMUM AND MAXIMUM
    5 MMIN(J)=0
      MMAX(J)=1
      J=J+1
      GO TO 100
C     CHARACTER IS A PLUS. INDICATES THE END OF A MINTERM OR CELL.
   15 J=1
      L=L+1
      MX=0
      MN=0
      DO 200 K=1,N
      MX=MX+MMAX(K)*2**(N-K)
      MN=MN+MMIN(K)*2**(N-K)
  200 CONTINUE
      DIF=MX-MN
      IF(DIF.NE.0)GO TO 25
      IMIN(L)=MX
      IF(MINCEL(I).EQ.BLANK)GO TO 35
      GO TO 100
C     GENERATE THE VERTICES BETWEEN MIN. AND MAX.
   25 ITEMP=1
      ICNT=1
      NUM(1)=MN
      DO 400 K=1,N
      IZ=IAND((2**(K-1)),DIF)
      IF(IZ.EQ.0)GO TO 400
      IC2=ICNT
```

```
      DO 600 M=1,IC2
      ICNT=ICNT+1
      NUM(ICNT)=NUM(M)+IZ
      IMIN(L)=NUM(ICNT)
      L=L+1
  600 CONTINUE
  400 CONTINUE
      IMIN(L)=MN
      IF(MINCEL(I).EQ.BLANK)GO TO 35
  100 CONTINUE
      GO TO 1
C     ELIMINATE REPEATED TERMS AND ARRANGE IN INCREASING ORDER
   35 K=0
      DO 300 I=1,L
      IF(K.EQ.0)GO TO 650
      DO 500 J=1,K
      IF(IMIN(I).EQ.IOUT(J))GO TO 300
  500 CONTINUE
  650 K=K+1
      IOUT(K)=IMIN(I)
  300 CONTINUE
      DO 302 I=1,K
      DO 302 J=1,K
      IF(IOUT(I).GT.IOUT(J))GO TO 302
      MT=IOUT(I)
      IOUT(I)=IOUT(J)
      IOUT(J)=MT
  302 CONTINUE
      RETURN
      END
```

APPENDIX II

Assembly language program (System 360/50) for LOGICAL AND operation.

```
*
*    SUBROUTINE FOR LOGICAL 'AND'.
*  USE IAND(II,JJ)
*
*
IAND   START
  STM   14,12,12(13)            SAVE REGISTERS
  L   2,0(1)                    LOAD ADDRESS OF II
  L   3,4(1)                    LOAD ADDRESS OF JJ
  L   0,0(2)                    LOAD II
  N   0,0(3)                    AND JJ
  LM   14,15,12(13)             RESTORE REGISTERS
  LM   1,12,24(13)
  BR   14
  END
```