NASA CONTRACTOR REPORT

N73-19230

NASA CR-2212

NASA CR-2212

COMPUTERIZED LOGIC DESIGN OF DIGITAL CIRCUITS

by Sidney Gussow and Rodney Oglesby

Prepared by
SPERRY RAND CORPORATION
SPACE SUPPORT DIVISION
Huntsville, Ala.
for George C. Marshall Space Flight Center

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION • WASHINGTON, D. C. • MARCH 1973

TECHNICAL REPORT STANDARD TITLE PAGE

1. REPORT NO. NASA CR-2212	2. GOVERNMENT ACCESSION NO.	3. RECIPIENT'S CATALOG NO.
4. TITLE AND SUBTITLE COMPUTERIZED LOGIC DESIGN OF DIGITAL CIRCUITS		5. REPORT DATE March 1973
		6. PERFORMING ORGANIZATION CODE M216
7. AUTHOR(S) Sidney Gussow and Rodney Oglesby		8. PERFORMING ORGANIZATION REPORT # 275-0678
9. PERFORMING ORGANIZATION NAME AND ADDRESS Sperry Rand Corporation Space Support Division Huntsville, Alabama		10. WORK UNIT NO.
		11. CONTRACT OR GRANT NO. NAS 8-21812
12. SPONSORING AGENCY NAME AND ADDRESS National Aeronautics and Space Administration Washington, D. C. 20546		13. TYPE OF REPORT & PERIOD COVERED Contractor
		14. SPONSORING AGENCY CODE

15. SUPPLEMENTARY NOTES

16. ABSTRACT

This manual presents a computer program that performs all the work required for the logic design of digital counters or sequential circuits and the simplification of Boolean logic expressions. The program provides both the experienced and inexperienced logic designer with a comprehensive logic design capability. The manual contains Boolean simplification and sequential design theory, detailed instructions for use of the program, a large number of illustrative design examples, and complete program documentation.

17. KEY WORDS	18. DISTRIBUTION STATEMENT

19. SECURITY CLASSIF. (of this report)	20. SECURITY CLASSIF. (of this page)	21. NO. OF PAGES	22. PRICE
Unclassified	Unclassified	165	$3.00

MSFC - Form 3292 (May

* For sale by the National Technical Information Service, Springfield, Virginia 22151

TABLE OF CONTENTS

Page

SECTION 1

INTRODUCTION

A computerized logic design procedure is presented for a specially developed computer program that performs all the work required for the logic design of digital counters or sequential circuits and the simplification of Boolean expressions. The program was developed by the Space Support Division of Sperry Rand Corporation under the sponsorship of the National Aeronautics and Space Administration (NASA). The program provides a simple, accurate, and comprehensive logic design capability to users both experienced and totally inexperienced in logic design. The manual is written in clear, easy to understand language and strong emphasis is placed on the use of illustrative design examples. Detailed instructions are given for use of the program and general design theory is included on the methods of Boolean simplification, sequential design procedures and the characteristics of all types of flip flops.

The program has been developed for two modes of operation: Counter design and Boolean simplification. In the counter design mode, the program provides the simplified flip flop input equations for any desired sequence and using any type of flip flop, JK, RS, D, T, or RST. The simplified logic equations can be obtained in either sum-of-product or product-of-sum form for any one or all of the flip-flop types. The program also provides printout of the intermediate design steps used in obtaining the flip flop input equations. In the Boolean simplification mode, the program simplifies Boolean logic functions that can be entered in either sum-of-product or product-of sum form.

The counter design mode can also be used for design of sequential circuits other than counters. The program can be used for any sequential design application where a group of flip flops are required to change states in a prescribed order. Here each flip flop is assigned a binary bit representation which provides an equivalent count sequence for the prescribed state changes. The equivalent count sequence is read into the computer the same as for regular counter designs. The program provides the logic equations that will cause the flip flops to change states in the prescribed order.

Considerable effort was expended in the development of these programs to reduce the input data requirements to the simplest level possible and to present the output results in a self-explanatory and instructive format. The required input to the program is the data a designer usually begins with for hand calculations. For counter designs, the input is the type flip flop and the count sequence in decimal numbers. The output is the simplified logic equations in proper form for direct implementation with the specified flip flops and the desired digital gates. For Boolean simplification, the function is read in and printed out in its normal hand written form.

The manual contains numerous design examples to illustrate all the uses and capabilities of the program. To make the examples as instructive and self-explanatory as possible, reduced replicas of the actual data cards used and the resulting computer printout for these data cards are included for each example. For several of the design examples it is shown how to implement the flip flop input equations with NAND, NOR, AND, and OR gates interconnected with the flip flops to show a complete counter design.

The program also instructs the user in counter design by printing out the intermediate design steps used in obtaining the simplified logic equations. These intermediate steps are the same as those that would be performed if the counter were designed by hand and include the flip flop transition input requirements and the counter truth table. Hence, each program printout also serves as an instructive design example and demonstrates the design procedure used. Printout of these intermediate steps can be suppressed by a simple option if desired.

The program greatly reduces the amount of time and effort required for logic design. Most counter design methods are fairly straightforward but do involve much laborious and tedious work for long count sequences or unordered sequences, especially when the logic equations are desired for more than one type of flip flop. A useful feature of the program is that for a given count sequence, the counter design equations can be obtained for several or all of the available type flip flops. The designer can compare these equations for simplicity or some other desired characteristic and select the optimum type flip flop for the given sequence.

The manual is composed of four main parts contained in Sections 2 through 5. Section 2 covers the general digital design concepts utilized by the program. Included is a description of the iterative and tabular Boolean simplification techniques used to simplify the Boolean expressions and the theory of sequential design. The flip flop truth table and input requirements are given for the JK, RS (Set-Reset), D (Delay), T (Trigger), and RST (Set-Reset-Trigger) flip flops.

Section 3 contains detailed instructions for the use of the program. It explains the input formats and how to enter the data onto the data cards and describes the various options that are available. The output formats are also explained in detail. Sample designs are given to illustrate the input and output formats. Section 4 contains design examples that demonstrate the various options, uses, and capabilities of the program. There are eleven examples given, six counter designs and five simplifications of Boolean expressions.

The program consists of a main-line program and ten subroutine programs. Complete documentation for these programs is given in Section 5. This section contains a description of the purpose and operation of each program along with the FORTRAN source listing and flow charts for each program. The programs are written in FORTRAN IV for the UNIVAC 1108 computer and can be adapted easily to any computer using FORTRAN IV or FORTRAN V. The use of the program stored on tape is covered in the Appendix.

The computer program presented here is far superior to other logic design programs in that the program begins at the most common or usual starting design point and terminates at the most logical end point, thereby performing all the required design work. A joint NASA/Sperry Rand research effort determined the need and inherent wide application for such a comprehensive design program. A survey of other available programs showed that nothing existed like what was needed. This program was then developed under the sponsorship of NASA to fulfill this need. The program should find wide use in all applications where digital logic design is required.

This manual is presented in belief that the quick, simple, and accurate logic design capability contained herein will be of tremendous value to all digital designers, both experienced and totally inexperienced.

SECTION 2

GENERAL DESIGN THEORY

2.1 BOOLEAN SIMPLIFICATION

2.1.1 Introduction

The purpose of the Boolean simplification is to simplify Boolean expressions in the form of sum-of-products to a minimum sum-of-products. A two step approach is used to accomplish this.

The first step is to obtain the prime implicants of the original Boolean expression. (When no further reduction in variables of each individual term can be accomplished by applying Boolean theorems to the original expression, the resulting irreducible terms are called prime implicants). The results of the first step is a sum-of-product expression containing only prime implicants. An iterative method will be used to obtain this expression.

Since a sum-of-product expression composed of only prime implicants is not necessarily a minimum sum-of-products, the second step is to assure that the final expression is a minimum sum-of-products. A tabular method will be used to assure the minimum sum-of-products expression.

2.1.2 Iterative Method

The iterative method used in obtaining the prime implicants makes use of three Boolean theorems:

1. $XY + \overline{X}Z = XY + \overline{X}Z + YZ$
2. $Y + Y = Y$
3. $Y + YZ = Y$

X is a single term variable

Y and Z contain one or more variables

The first theorem is applied systematically to all pairs of terms of the expression to generate all possible included terms (YZ). These new terms are added to the expression. The second and third theorems make use of the new terms to eliminate other terms. This process is continued until no more new terms can be generated by the first theorem, or until the new terms that are generated are immediately eliminated by the second and third theorems. After this process has been exhausted, the remaining terms in the expression are prime implicants.

For example, if

$$f = \bar{A}\bar{B}C + \bar{A}BC + ABC + AB\bar{C}.$$

The first theorem is applied to the first and second term which generates the new term $\bar{A}C$. This term is added to the expression, giving

$$f = \bar{A}\bar{B}C + \bar{A}BC + ABC + AB\bar{C} + \bar{A}C.$$

The third theorem makes use of the new term to eliminate the first and second terms. The expression is now

$$f = \bar{A}C + ABC + AB\bar{C}.$$

The first theorem is applied to the first and second term, forming the new term BC. With this term added, the expression is

$$f = \bar{A}C + ABC + AB\bar{C} + BC.$$

Applying the third theorem, the second term is eliminated, yielding

$$f = \bar{A}C + BC + AB\bar{C}.$$

The first theorem is applied to the second and third term, generating the new term AB. When AB is added and the third theorem applied, the expression becomes

$$f = \bar{A}C + BC + AB.$$

The remaining terms in the above expression are prime implicants.

2.1.3 Tabular Method

An expression containing only prime implicants is not necessarily a minimum sum-of-products. Each individual term cannot be reduced further since they are prime implicants; however, it may be possible to eliminate some of the terms completely.

The approach taken is to check each prime implicant individually to see if it can be eliminated. This is accomplished by first expanding the prime implicants under consideration into an expanded sum-of-product, then checking to see if every term in the expanded sum-of-product is contained in the remaining prime implicants. If every term is contained at least once, then that prime implicant is eliminated. This process is continued until all prime implicants have been considered and all unnecessary terms eliminated, resulting in a minimum sum-of-products.

The following is an example of the tabular method. First, a table will be constructed with a row at the top showing the expanded sum-of-products of the prime implicant under consideration. A column in the table lists the remaining prime implicants. If a term in the expanded sum-of-products is contained in one of the remaining prime implicants, a check is inserted in the table. If all terms of the expanded sum-of-product are accounted for, then that prime implicant is eliminated. A table is constructed for each prime implicant.

For continuity, the results of the iterative method section will be used where,

$$f = \bar{A}C + BC + AB.$$

The term $\bar{A}C$ will be considered first.

	$\bar{A}\bar{B}C$	$\bar{A}BC$
BC		x
AB		

All terms are not included;therefore, this term, $\bar{A}C$, cannot be eliminated.

The term BC will be considered next.

	$\bar{A}BC$	ABC
$\bar{A}C$	x	
AB		x

All terms are contained; therefore, this term is eliminated, reducing the expression to

$$f = \bar{A}C + AB.$$

The last term, AB, will be considered.

	$AB\bar{C}$	ABC
$\bar{A}C$		

None of the terms are contained so this term cannot be eliminated.

All terms have now been considered resulting in the minimum sum-of-products

$$f = \bar{A}C + AB.$$

2.2 SEQUENTIAL DESIGN

2.2.1 Introduction

The purpose of the sequential design is to develop the logic flip flop input equations for sequential counters. The design is accomplished through five steps; (1) word statement, (2) state diagram, (3) truth table, (4) flip flop inputs, and (5) flip flop input equations.

The first step in the design is to formulate a word statement. The word statement is a brief statement of the sequential circuit functions. The complete description of a sequential counter is given by the number of counts and the state assignments. From the word statement a state diagram is developed which shows the transitions from state to state. The state diagram has the same number of states as counts given in the word statement. At this point, the state assignments are made as specified in the word statement.

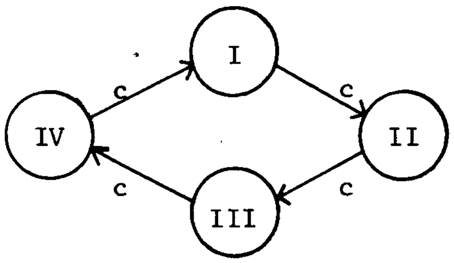
The next step in the design is the truth table which provides a means of bringing the complete design to a central point. The table is made up of three sections; present states, next states, and flip flop inputs. The present states and next states show every state transition, one at a time, as depicted by the state diagram. The flip flop input section shows the correct flip flop input to cause the desired state transition from the present state to the next state. The information to complete the flip flop input section is obtained from each individual flip flop truth table.

The last step in the design is obtaining the flip flop input equation. These equations are Boolean expressions and are obtained by associating the flip flop inputs with the present states in the truth table.

2.2.2 Example

*Word Statement. Design a 4 counter that counts in a binary sequence.

*State Diagram. The first step is to draw a state diagram showing only the number of states.



c = clock pulse

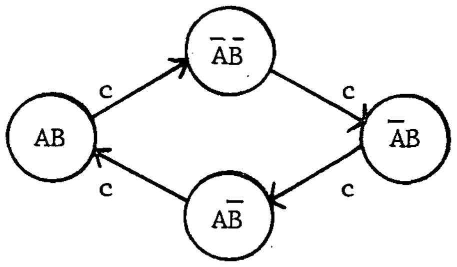
STATE DIAGRAM

The next step is to make the state assignments. First, the number of flip flops must be determined. This is done by the following equation.

$2^N \geq S$ N = number of flip flops
S = number of states

From the state diagram, S is equal to four; therefore, the number flip flops, N, is equal to two. The flip flops will be labeled A and B.

The state diagram is now completed by making the state assignment in a binary sequence as specified in the word statement.



c = clock pulse

STATE DIAGRAM

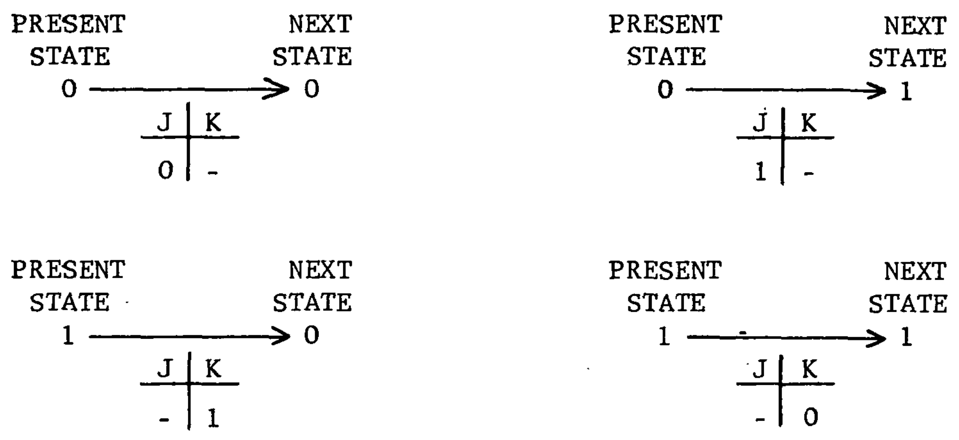
*Truth Table. The truth table is constructed with the three sections; present state, next state, and flip flop inputs.
(Only J-K flip flops will be considered in the example.)

PRESENT STATE		NEXT STATE		FLIP FLOP INPUTS			
				A FLIP FLOP		B FLIP FLOP	
AB		AB		JA	KA	JB	KB
00	I	01	II				
01	II	10	III				
10	III	11	IV				
11	IV	00	I				

TRUTH TABLE

The present state and next state columns are filled in with information obtained from the state diagram.

*Flip flop inputs. From a J-K flip flop truth table, it can be shown that the following flip flop inputs will cause the respective transitions.



- = Don't Care

With these flip flop inputs the truth table is completed.

PRESENT STATE		NEXT STATE		FLIP FLOP INPUTS			
				A FLIP FLOP		B FLIP FLOP	
AB		AB		JA	KA	JB	KB
00	I	01	II	0	-	1	-
01	II	10	III	1	-	-	1
10	III	11	IV	-	0	1	-
11	IV	00	I	-	1	-	1

TRUTH TABLE

*Flip flop input equations. Each flip flop input equation is made up of those present states associated with the one's and don't care's for each respective input.

These equations are as follows:

$$J_A = \bar{A}B + A\bar{B}* + AB*$$

$$K_A = AB + \bar{A}\bar{B}* + \bar{A}B*$$

$$J_B = \bar{A}\bar{B} + A\bar{B} + \bar{A}B* + AB*$$

$$K_B = \bar{A}B + AB + \bar{A}\bar{B}* + A\bar{B}*$$

* Don't care term.

The above equations are simplified, thereby completing the sequential design in the form of flip flop input equations.

2.3 FLIP FLOP CHARACTERISTICS

There will be five types of flip flops discussed;

1. J-K
2. RESET-SET (R-S)
3. RESET-SET-TRIGGER (R-S-T)
4. DELAY (D)
5. TRIGGER (T)

The characteristics of the flip flops will be given in the form of an operating characteristic table. From the operating characteristic table, the actual flip flop input requirements are found for the desired transitions.

The operating characteristics table gives all possible combinations of inputs into the flip flops and the corresponding transitions. The expression t_n will be used to designate time before the clock occurrence, hence before the transition. The expression t_{n+1} will be used to designate time after the clock occurrence, hence after the transition.

The input requirements portion will show essentially the same information as the operating characteristic table except that the "don't care" conditions are taken advantage of.

J-K FLIP FLOP

INPUT		TRANSITIONS			
J	K	t_n	t_{n+1}	t_n	t_{n+1}
0	0	0 →	0	1 →	1
0	1	0 →	0	1 →	0
1	0	0 →	1	1 →	1
1	1	0 →	1	1 →	0

OPERATING CHARACTERISTICS TABLE

From the operating characteristic table, the input requirements for the J-K flip flop are:

t_n	t_{n+1}	t_n	t_{n+1}	t_n	t_{n+1}	t_n	t_{n+1}
$0 \longrightarrow$	0	$0 \longrightarrow$	1	$1 \longrightarrow$	0	$1 \longrightarrow$	1
J	K	J	K	J	K	J	K
0	-	1	-	-	1	-	0

- = Don't Care

R-S FLIP FLOP

INPUT		TRANSITION			
R	S	t_n	t_{n+1}	t_n	t_{n+1}
0	0	$0 \rightarrow$	0	$1 \rightarrow$	1
0	1	$0 \rightarrow$	1	$1 \rightarrow$	1
1	0	$0 \rightarrow$	0	$1 \rightarrow$	0
1	1	NOT ALLOWABLE			

OPERATING CHARACTERISTICS TABLE

R-S flip flop input requirements are:

t_n	t_{n+1}	t_n	t_{n+1}	t_n	t_{n+1}	t_n	t_{n+1}
$0 \longrightarrow$	0	$0 \longrightarrow$	1	$1 \longrightarrow$	0	$1 \longrightarrow$	1
R	S	R	S	R	S	R	S
-	0	0	1	1	0	0	-

DELAY FLIP FLOP

INPUT	TRANSITION			
D	t_n	t_{n+1}	t_n	t_{n+1}
0	$0 \rightarrow$	0	$1 \rightarrow$	0
1	$0 \rightarrow$	1	$1 \rightarrow$	1

OPERATING CHARACTERISTICS TABLE

D flip flop input requirements are:

t_n	t_{n+1}	t_n	t_{n+1}	t_n	t_{n+1}	t_n	t_{n+1}
$0 \longrightarrow$	0	$0 \longrightarrow$	1	$1 \longrightarrow$	0	$1 \longrightarrow$	1
D		D		D		D	
0		1		0		1	

TRIGGER

INPUT	TRANSITION			
T	t_n	t_{n+1}	t_n	t_{n+1}
0	0 →	0	1 →	1
1	0 →	1	1 →	0

OPERATING CHARACTERISTICS TABLE

The T flip flop input requirements are:

t_n	t_{n+1}	t_n	t_{n+1}	t_n	t_{n+1}	t_n	t_{n+1}
0 →	0	0 →	1	1 →	0	1 →	1

T	T	T	T
0	1	1	0

R-S-T FLIP FLOP

INPUT	TRANSITION			
R S T	t_n	t_{n+1}	t_n	t_{n+1}
0 0 0	0 →	0	1 →	1
0 0 1	0 →	1	1 →	0
0 1 0	0 →	1	1 →	1
0 1 1	0 →	1	NOT ALLOWABLE	
1 0 0	0 →	0	1 →	0
1 0 1	NOT ALLOWABLE		1 →	0
1 1 0 1 1 1	NOT ALLOWABLE			

OPERATING CHARACTERISTICS TABLE

The R-S-T flip flop input requirement are:

t_n	t_{n+1}	t_n	t_{n+1}	t_n	t_{n+1}	t_n	t_{n+1}
0 →	0	0 →	1	1 →	0	1 →	1

R S T	R S T	R S T	R S T
- 0 0	0 - 1	1 0 -	0 - 0
	0 1 -	- 0 1	

SECTION 3

DESIGN BY COMPUTER PROGRAM

3.1 INTRODUCTION

A computer program was developed to perform all the logic design required for:

a. Design of digital counters of any desired sequence using the following type flip flops:

   1. JK
   2. RS (Set-Reset)
   3. D (Delay)
   4. T (Trigger)
   5. RST (Set-Reset-Trigger)

b. Simplification of Boolean expressions, including true and don't care terms.

The Boolean simplification capability is internally used in the counter design mode. For counter design, the computer derives the simplified Boolean input equations for each flip flop. The input equations can be obtained in either sum-of-product or product-of-sum form. The input required to the computer is the type of flip flops and the desired sequence in decimal numbers. For Boolean simplifications, the logic expression to be simplified can be entered in either sum-of-product or product-of-sum form but not a combination of both.

The program is specifically designed for simplified input data and self-explanatory output data, requiring the user to have only a basic knowledge of digital design. The program is also instructive, in that it provides all the main intermediate design steps used in obtaining the flip flop input equations. These intermediate steps are the same as would be performed if the counter were designed by hand, and include: the input sequence logic table, the flip flop transition input requirements, and the counter truth table. Hence, the program output actually serves as a design example by itself and is instructive to the designer. An input option is available to suppress printout of the intermediate steps if desired by the user. The program should be useful to both the experienced and inexperienced digital designer.

The program consists of a main program (DIGITL) and ten subroutine programs. The FORTRAN source listing and the flow chart of each program along with a description of each program are given in Section 5.

3.2 HOW TO USE PROGRAM

The program has two operating modes corresponding to its two functions of:

a. Digital counter design of any specified sequence.

b. Simplification of Boolean functions.

The operating mode is selected by a code letter in column one on the first data card of each data set. The code letter is "C" for counter design and "F" for Boolean function simplification.

3.2.1 Counter Design Input Format

To use the program in the counter design mode the input data cards are prepared in the following format.

1st Card, Col. 1 - Letter "C"

Col. 2-80 - List of flip flop code letters for each type FF desired, with commas separating FF codes. Output options 1 and 2 if desired, placed after last FF code letter.

2nd Card, Col. 1-5 - Number of counts in counter sequence in integer (I5) format.

Col. 5-80 - Count sequence in integer (I5) format. Use decimal number representation.

3rd thru Nth Card, Col. 1-80 - Continue count sequence in integer (I5) format until last count is listed.

(N+1)th Card - Repeat format of first N cards for next counter design.

For card #1, the code letters and symbols are as follows:

Flip Flop	Code Letters
JK	JK
Set-Reset	RS

Flip Flop	Code Letters
Delay	D
Trigger	T
Set-Reset-Trigger	RST
All Types (except RST)	All

Output Option	Code Symbol
1. Short Form Output	/ (Slash Mark)
2. Product-of-Sum Format	P

1. Short Form Output - Suppresses printout of the flip flop transition input requirements, the counter truth table, and the don't care terms of the flip flop input equations.
2. Product-of-Sum Format - Flip flop input equations are given in product-of-sum form instead of sum-of-product form which is normally used.

The flip flop code letters can be placed in any order anywhere within columns 2 through 80 of the first data card. Use a comma between each type flip flop specified. Any spacing can be used except that the code letters for each flip flop must appear together without any blank spaces between them. For example, the code letters for the Set-Reset-Trigger must be written as RST, and not R-S-T or R S T. However, the placing of the commas and the spacing between flip flop types is arbitrary. If all flip flop types (except RST) are desired, the code word "All" can be used in place of listing the code letters for each type flip flop. The code for RST can be included if desired. The RST flip flop is not included in code "All" because it is somewhat redundant to the RS and T flip flops. When RST is specified, the compůter prints out two sets of flip flop input equations; one where RST is used as an RS flip flop and another where RST is used as a T flip flop. When only true terms are considered, the two sets of input equations for the RST flip flop are identical to those of the RS and T flip flop respectively.

Either one or both of the output option codes can be added after the flip flop code listing. A comma need not be placed after the last flip flop code or between output option codes; although commas can be used if desired. For example, acceptable listings using both output options would be JK, RS, T /P or JK, RS, T, /, P.

The counts are placed on the data card as integer decimal numbers in I5 format. N o t e that each number is allocated five spaces and is right justified in its allocated columns. Right justified means that the least significant digit of the number is located in its most right column, for this case columns 5, 10, 15, 20, 25, etc.

The count sequence can be in any numerical order. The program will assign don't care conditions to all count numbers not used and will design a counter that will continuously repeat the specified sequence. To specify a counter that will stop after the last count is reached, list the last count of the sequence twice. For example, for the sequence 2, 5, 67, 8, 74, 14, 14, the counter would stop after reaching count 14. If the last count is not repeated, the counter designed will loop back to the first count and continue counting.

Placing a slash after the flip flop code letters suppresses printout of the flip flop transition input requirements, and the counter truth table, and the don't care terms of the flip flop input equations. This option permits a quick comparison of the input flip flop equations (true terms) for the specified flip flops.

3.2.2 Boolean Simplification Input Format

To use the program in the Boolean simplification mode the input data cards are prepared in the following format.

1st Card, Col. 1 - Letter "F"

Col. 2-80 - Boolean expression in either sum-of-product or product-of-sum form, but not a mixture of both. Use any desired letters A-Z. Follow letter with an apostrophe (') to signify a false or not condition.

2nd thru Nth Card, Col 1-80 - Continue Boolean expression as on 1st card. Use as many cards as necessary to complete expression. Place an asterisk (*) after the last term to terminate expression.

(N+1)th Card - Repeat format of first N cards for next Boolean expression.

The Boolean expressions can be placed anywhere in the allocated columns with blank spaces used wherever desired with the exception that an apostrophe must immediately follow a letter. For expressions in sum-of-product form, a plus (+) sign must be placed between products. Parentheses cannot be used for grouping because parentheses are used to signify the product-of-sum form. Don't care terms may be included separately when the sum-of-product form is used. To include don't care terms, place a slash (/) after the last true term and follow with the don't care terms. The program will simplify the don't care terms along with the complete expression.

When using the product-of-sum form, use plus signs between single letters and use parentheses around sums of letters to signify multiplication. For example, (A + B' + C) (A + B) (A' + B + C'). Don't care terms cannot be included separately.

For both input forms, an asterisk (*) must be placed after the last Boolean term (sum or product) to terminate the expression. Another Boolean function can be entered on the next data card beginning with a "F" in column one and followed by the Boolean expression as before.

Each Boolean expression must be in either sum-of-product form or product-of-sum form but not a mixture of both. Each product or sum need not contain all the letters (variables) used in the complete expression. However, each letter may only appear once in each product or sum term. Also, the letters used need not be in alphabetical order or used in the same order in each product or sum term. Examples of correct and incorrect Boolean expressions of both forms for input to the program are given below.

Correct expressions:

```
ABC' + BAC + C'B'  *
(A + B' + C) (C' + B') (C + A' + B)  *
```

Incorrect expressions:

```
ABB'C + AC' + BC  *
AB + (A'B + C)AC + BC  *
(A + B' + C) ( A' + BC) (A + C')  *
```

3.3 OUTPUT FORMAT

To aid in describing the input and output format, two sample design cases are illustrated showing the input data format and the output computer printout. Sample design 1 is for a 14 count counter using JK flip flops and sample design 2 is for simplification of a Boolean expression including don't care terms.

3.3.1 Counter Design Printout

The input data cards for a random 14 state counter using JK flip flops are shown in Figure 3.1. The computer printout for this input data is shown in Figure 3.2. The printout first gives the input data exactly as it appear on the data cards. The flip flop and code option designations in columns 2-80 of the first data card are printed out on the second line following the heading "flip flops". The input count sequence data on the following data cards appears after the heading "input count data".

Next is the computer calculated output data. The input count sequence gives the binary representation for each decimal count with letter "A" as the most significant binary bit. All count states not used in the count sequence are listed below the input sequence as don't care terms.

The flip flop transition input requirements appear next. The designation 0-------> 1 means that the Q output of the flip flop goes from state "0" to state "1". For the JK flip flop to go from state "0" to "1", the input required is a true or "1" on the J input and a don't care on the K input meaning either a "1" or "0" can be used for the K input.

Listed next is the counter truth table showing the input conditions of each flip flop for each binary count. For this counter, four flip flops designated by letters A, B, C, D are required. For the counter to advance from count 14 to count 11, flip flop B must change from state "1" to "0". Hence, the input for flip flop B during count 14 is a don't care on the J input and a "1" on the K input as shown in the truth table.

Finally, the flip flop input equations are given for each flip flop of the counter. The input equations are those Boolean functions that when applied to the flip flop causes the flip flop to be in its required binary state during each count of the specified count sequence. A block diagram

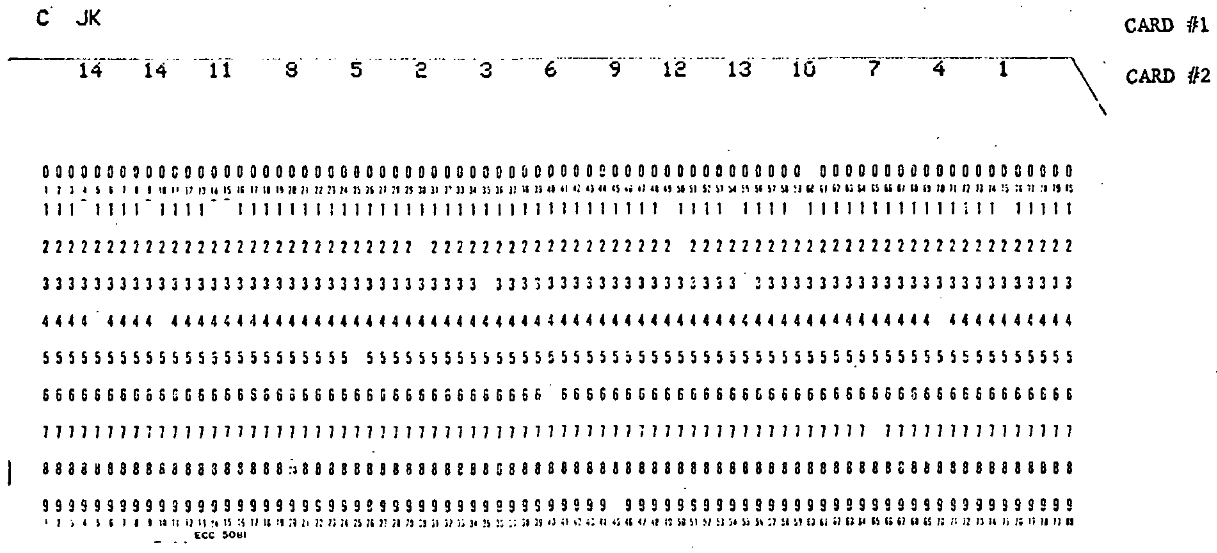


Figure 3.1. Data Cards For Sample Counter Design

```
FLIP-FLOP TYPES SPECIFIED FOR COUNTER

FLIP-FLOPS :  JK


INPUT COUNT DATA, FIRST NO. IS NUMBER OF COUNTS

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



        INPUT COUNT SEQUENCE LOGIC

         I      NO(I)    A B C D
         1       14      1 1 1 0
         2       11      1 0 1 1
         3        8      1 0 0 0
         4        5      0 1 0 1
         5        2      0 0 1 0
         6        3      0 0 1 1
         7        6      0 1 1 0
         8        9      1 0 0 1
         9       12      1 1 0 0
        10       13      1 1 0 1
        11       10      1 0 1 0
        12        7      0 1 1 1
        13        4      0 1 0 0
        14        1      0 0 0 1
        15       14      1 1 1 0
       --------------- DON'T CARE TERMS BELOW
        16        0      0 0 0 0
        17       15      1 1 1 1
```

Figure 3.2. Printout of Program DIGITL for Sample Counter Design (Sheet 1 of 3)

```
                    J K   FLIP-FLOP TRANSITION INPUT REQUIREMENTS

                   0 ----> 0     0 ----> 1     1 ----> 0     1 ----> 1
                     J K           J K           J K           J K
                     0 -           1 -           - 1           - 0

                '1' = TRUE,  '0' = FALSE,  '-' = DON'T CARE

                   TRUTH TABLE FOR COUNTER USING J K  FLIP-FLOPS

      NO     A B C D    JA KA JB KB JC KC JD KD

      14     1 1 1 0    -  0  -  1  -  0  1  -

      11     1 0 1 1    -  0  0  -  -  1  -  1

       8     1 0 0 0    -  1  1  -  0  -  1  -

       5     0 1 0 1    0  -  -  1  1  -  -  1

       2     0 0 1 0    0  -  0  -  -  0  1  -

       3     0 0 1 1    0  -  1  -  -  0  -  1

       6     0 1 1 0    1  -  -  1  -  1  1  -

       9     1 0 0 1    -  0  1  -  0  -  -  1

      12     1 1 0 0    -  0  -  0  0  -  1  -

      13     1 1 0 1    -  0  -  1  1  -  -  1

      10     1 0 1 0    -  1  1  -  -  0  1  -

       7     0 1 1 1    0  -  -  0  -  1  -  1

       4     0 1 0 0    0  -  -  1  0  -  1  -

       1     0 0 0 1    1  -  1  -  1  -  -  1

      14     1 1 1 0    0  0  0  0  0  0  0  0

       0     0 0 0 0    -  -  -  -  -  -  -  -

      15     1 1 1 1    -  -  -  -  -  -  -  -
```

Figure 3.2. Printout of Program DIGITL for Sample Counter Design (Sheet 2 of 3)

```
                              J K   FLIP-FLOP INPUT EQUATIONS
                                    (F = TRUE TERMS / DON'T CARE TERMS)

                                    CLOCK ---.
                                             V
                                    ..............
                                    :            :
                               ---->: J        Q :---->
                                    :            :
                                    :            :
                                    :            :
                               ---->: K        Q':---->
                                    :            :
                                    :............:

       FLIP-FLOP A

          JA = BCD' + B'C' / A

          KA = B'D' / A' + BCD

       FLIP-FLOP B

          JB = AD' + A'D + C' / B

          KB = CD' + C'D + A'D' / B' + AD

       FLIP-FLOP C

          JC = BD + A'D / C + A'B'

          KC = AD + A'B / C'

       FLIP-FLOP D

          JD = 1

          KD = 1
```

Figure 3.2. Printout of Program DIGITL for Sample Counter Design (Sheet 3 of 3)

of the flip flop showing its input and output terminals is included in the printout. The input equations list the simplified true terms followed by the simplified don't care terms. The true and don't care terms are separated by a slash mark. Only the true terms are required for the counter design. One or more of the don't care terms are sometimes used either to prevent the counter from hanging up in an unused count state on initial turn-on or because the don't care term is included in an available expression already generated.

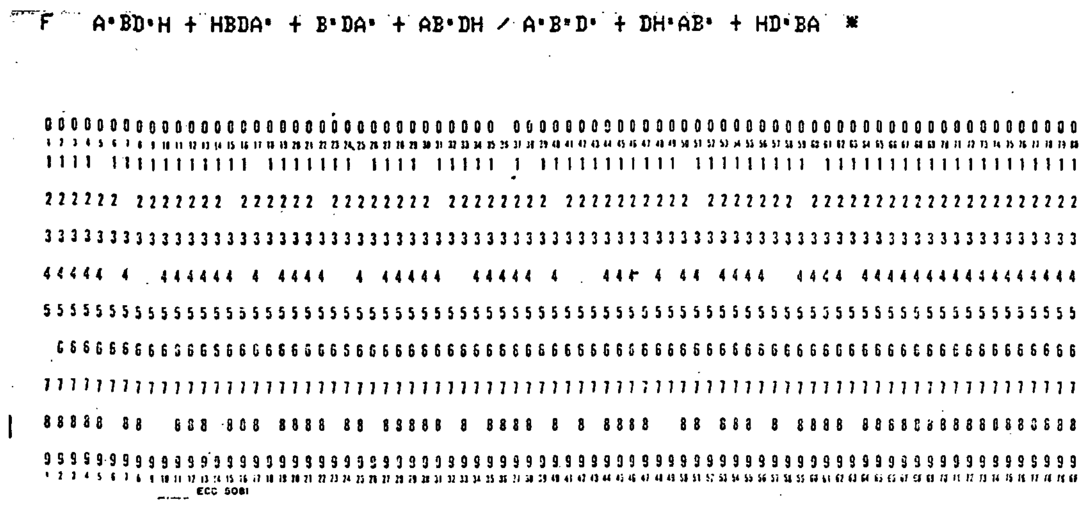
The counter design is for a synchronous counter with a common clock, meaning that all flip flops are clocked to change states at the same time. The input equations are derived directly from the counter truth table. The input equation for the J input of flip flop A (JA) is obtained from the true and don't care terms listed under column JA. For this example, there are two true terms for JA which are binary count 6 (A'BCD'), and count 1(A'B'C'D). There are 9 don't care terms which are binary counts 14, 11, 8, 9, 12, 13, 10, 0, and 15. The complete expression for JA when simplified is JA = BCD' + B'C' / A as shown in the printout.

If only the true terms of the input equations are desired, printout of the don't care terms along with printout of the flip flop transition input requirements and the counter truth table can be suppressed by use of the short form option.

3.3.2 Boolean Simplification Printout

The input data cards and the computer printout for simplification of a Boolean expression including both true and don't care terms are shown in Figure 3.3. Under the heading "input function", the Boolean expression is printed out exactly as it appears on the data cards. Each line of the printout is the data on one card. The first computer calculated output is an alphabetical list of all letters used in the Boolean expression. Next is the simplified Boolean expression. The simplified expression is in the form of true terms followed by don't care terms (if any) with a slash mark separating the true and don't care terms. The letters of each product term are listed in alphabetical order regardless of whether or not the input product terms are in alphabetical order.

CARD #1



```
F  A'BD'H + HBDA' + B'DA' + AB'DH / A'B'D' + DH'AB' + HD'BA  *
```

A. Data Card

```
------INPUT FUNCTION      (F = TRUE TERMS / DON'T CARE TERMS)
                          * TERMINATES BOOLEAN EXPRESSION

      F =     A'BD'H + HBDA' + B'DA' + AB'DH / A'B'D' + DH'AB' + HD'BA  *

    LETTERS USED = ABDH


------SIMPLIFIED FUNCTION ----(F = TRUE TERMS / DON'T CARE TERMS)

      F = A'H + A'B' + B'D / BD'H
```

B. Computer Printout

Figure 3.3. Data Card and Printout of Program DIGITL for Simplification of Sample Boolean Expression

SECTION 4

DESIGN EXAMPLES

Numerous design examples are presented to demonstrate the various uses and capabilities of program DIGITL. The examples were selected so as to illustrate all the operating modes and output options of the program. Each example includes a reduced replica of both the actual data cards used and the resulting computer printout for these data cards. This makes the examples as self explanatory as possible. Counter design examples are given first followed by Boolean Simplification examples. All counter designs are for synchronous binary type counters.

The counter design examples given in Section 4.1 and 4.4 include logic schematics that illustrate how to realize the flip flop input equations with various types of digital gates. Section 4.4 demonstrates how input equations in sum-of-product form can be directly implemented with NAND gates and equations in product-of-sum form with NOR gates.

The program uses the RST flip flop in two modes, as an RS flip flop and a T (trigger) flip flop. When the RST flip flop is specified, the program prints out a separate set of data for the RST used first as an RS flip flop and second as a T flip flop.

A flip flop equation printout of 0/1 is possible and means that either a logic "0" or logic "1" voltage level can be used for the flip flop input.

4.1 COUNTER, 12 COUNT 4 STAGE

This design is for a repetitive counter having a 12 count sequence of 3, 2, 6, 7, 5, 4, 12, 13, 15, 14, 10, 11. This particular sequence is a cyclic code sequence, meaning that the binary representation of each successive decimal number differs by only one binary digit. Cycle codes are sometimes used to eliminate flip flop cross-over spikes when decoding counts. Since each binary digit or bit is represented by a flip flop, only one flip flop changes states between successive counts. Repetitive means that the counter continually repeats the count sequence.

The counter design mode is specified by entering letter C in column one of the first data card.

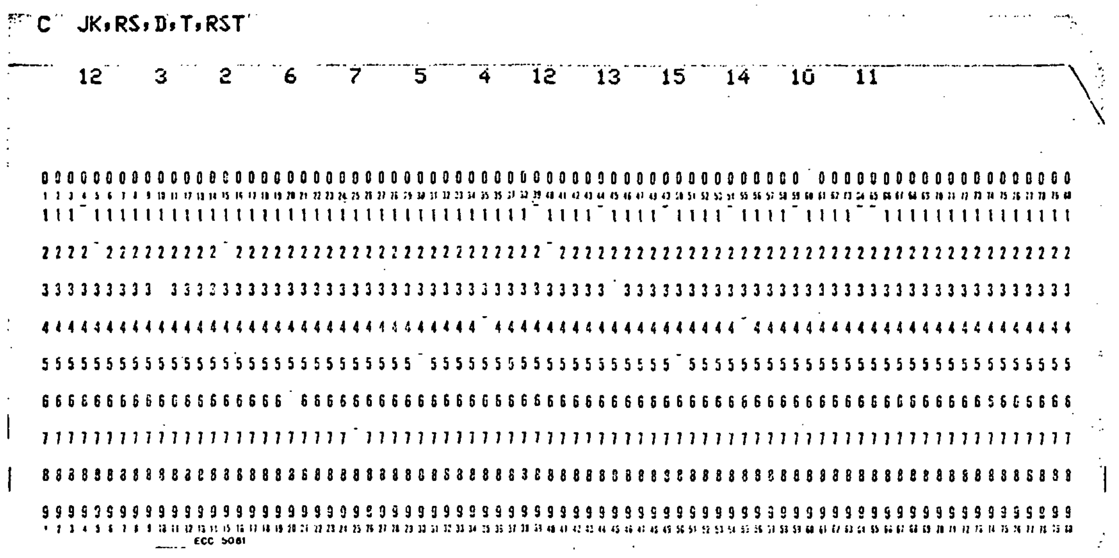
4.1.1 <u>JK, RS, D, T, RST</u>

Here the input flip flop equations for the 12 count 4 stage counter are to be calculated for the JK, RS, D, T, and RST flip flops. This data might be desired by a user who wants to compare the input equations of the different type flip flops for simplicity or some other characteristic. These five flip flop types could also be designated by the flip flop code letters: ALL, RST.

The input data cards for this example are shown in Figure 4.1. The computer printout of program DIGITL for these data cards is given in Figure 4.2. The normal (long form) output option is used which gives the flip flop transition input requirements, the counter truth table, and the true and don't care terms of the flip flop input equations. Only the true terms are necessary to implement the counter.

The program performs the counter logic design but does not specify how the input logic equations are to be implemented. This is left to the discretion of the designer. Several types of logic gates connected in any one of various configurations can be used. The commonly used gates are the NAND, NOR, AND, and OR gates. To illustrate how these gates can be used, the input equations for the JK flip flop have been realized for each type gate. The logic schematics of the counter implemented with NAND, NOR, and AND/OR gates are shown in Figures 4.3 through 4.5 respectively. These configurations are only one of many solutions that could have been used. Note that the true term equations for JD and KD are complements of each other, meaning JD = KD'. Hence, the configurations could be simplified for JK flip flops having inverting inputs (as many do) along with non-inverting inputs.

Notice that the NAND gate inputs in Figure 4.3 and the AND gate inputs in Figure 4.5 are obtained directly from the sum-of-product input equations for the JK flip flop. The NOR gate inputs are not so easily obtained from the sum-of-product form but can be obtained directly from



C JK,RS,D,T,RST

12 3 2 6 7 5 4 12 13 15 14 10 11

ECC 5081

CARD #1

CARD #2

Figure 4.1. Data Cards for 12 Count 4 Stage Counter for Example 4.1.1

```
FLIP-FLOP TYPES SPECIFIED FOR COUNTER

FLIP-FLOPS :   JK,RS,D,T,RST


INPUT COUNT DATA,  FIRST NO. IS NUMBER OF COUNTS

  12    3    2    6    7    5    4   12   13   15   14   10   11


      INPUT COUNT SEQUENCE LOGIC

                I      NO(I)      A B C D
                1        3        0 0 1 1
                2        2        0 0 1 0
                3        6        0 1 1 0
                4        7        0 1 1 1
                5        5        0 1 0 1
                6        4        0 1 0 0
                7       12        1 1 0 0
                8       13        1 1 0 1
                9       15        1 1 1 1
               10       14        1 1 1 0
               11       10        1 0 1 0
               12       11        1 0 1 1
               13        3        0 0 1 1
     -------------- DON'T CARE TERMS BELOW
               14        0        0 0 0 0
               15        1        0 0 0 1
               16        8        1 0 0 0
               17        9        1 0 0 1
```

Figure 4.2. Printout for Counter Design Example 4.1.1 (Sheet 1 of 13)

```
                    J K   FLIP-FLOP TRANSITION INPUT REQUIREMENTS

                    0 ----> 0      0 ----> 1      1 ----> 0      1 ----> 1
                      J K            J K            J K            J K
                      0 -            1 -            - 1            - 0

                    '1' = TRUE,  '0' = FALSE,  '-' = DON'T CARE

                    TRUTH TABLE FOR COUNTER USING J K   FLIP-FLOPS

      NO      A B C D    JA KA JB KB JC KC JD KD

       3      0 0 1 1    0  -  0  -  -  0  -  1

       2      0 0 1 0    0  -  1  -  -  0  0  -

       6      0 1 1 0    0  -  -  0  -  0  1  -

       7      0 1 1 1    0  -  -  0  -  1  -  0

       5      0 1 0 1    0  -  -  0  0  -  -  1

       4      0 1 0 0    1  -  -  0  0  -  0  -

      12      1 1 0 0    -  0  -  0  0  -  1  -

      13      1 1 0 1    -  0  -  0  1  -  -  0

      15      1 1 1 1    -  0  -  0  -  0  -  1

      14      1 1 1 0    -  0  -  1  -  0  0  -

      10      1 0 1 0    -  0  0  -  -  0  1  -

      11      1 0 1 1    -  1  0  -  -  0  -  0

       3      0 0 1 1    0  0  0  0  0  0  0  0

       0      0 0 0 0    -  -  -  -  -  -  -  -

       1      0 0 0 1    -  -  -  -  -  -  -  -

       8      1 0 0 0    -  -  -  -  -  -  -  -

       9      1 0 0 1    -  -  -  -  -  -  -  -
```

Figure 4.2. Printout for Counter Design Example 4.1.1 (Sheet 2 of 13)

```
                                     J-K FLIP-FLOP INPUT EQUATIONS
                                       (F = TRUE TERMS / DON'T CARE TERMS)

                                       CLOCK ----.
                                                 V
                                        ..............
                                        :            :
                                   ---->: J        Q :---->
                                        :            :
                                        :            :
                                        :            :
                                   ---->: K        Q':---->
                                        :            :
                                        :............:

      FLIP-FLOP A

         JA = C'D' / A + B'C'

         KA = B'D / A' + B'C'

      FLIP-FLOP B

         JB = A'D' / B + C'

         KB = ACD' / B'

      FLIP-FLOP C

         JC = AD / C + B'

         KC = A'BD / C'

      FLIP-FLOP D

         JD = A'BC + AC' + AB' / D + B'C'

         KD = A'B' + A'C' + ABC / D' + B'C'
```

Fugure 4.2. Printout for Counter Design Example 4.1.1 (Sheet 3 of 13)

```
                    R S  FLIP-FLOP TRANSITION INPUT REQUIREMENTS

              0 ----> 0     0 ----> 1     1 ----> 0     1 ----> 1
                S R           S R           S R           S R
                0 -           1 0           0 1           - 0

              '1' = TRUE,  '0' = FALSE,  '-' = DON'T CARE


                    TRUTH TABLE FOR COUNTER USING R S  FLIP-FLOPS

     NO     A B C D   SA RA SB RB SC RC SD RD

      3     0 0 1 1   0  -  0  -  -  0  0  1

      2     0 0 1 0   0  -  1  0  -  0  0  -

      6     0 1 1 0   0  -  -  0  -  0  1  0

      7     0 1 1 1   0  -  -  0  0  1  -  0

      5     0 1 0 1   0  -  -  0  0  -  0  1

      4     0 1 0 0   1  0  -  0  0  -  0  -

     12     1 1 0 0   -  0  -  0  0  -  1  0

     13     1 1 0 1   -  0  -  0  1  0  -  0

     15     1 1 1 1   -  0  -  0  -  0  0  1

     14     1 1 1 0   -  0  0  1  -  0  0  -

     10     1 0 1 0   -  0  0  -  -  0  1  0

     11     1 0 1 1   0  1  0  -  -  0  -  0

      3     0 0 1 1   0  0  0  0  0  0  0  0

      0     0 0 0 0   -  -  -  -  -  -  -  -

      1     0 0 0 1   -  -  -  -  -  -  -  -

      8     1 0 0 0   -  -  -  -  -  -  -  -

      9     1 0 0 1   -  -  -  -  -  -  -  -
```

Figure 4.2. Printout for Counter Design Example 4.1.1 (Sheet 4 of 13)

```
                                   R-S  FLIP-FLOP INPUT EQUATIONS
                                        (F = TRUE TERMS / DON'T CARE TERMS)

                                        CLOCK ----.
                                                  V
                                          ..............
                                          :            :
                                    ----->: S        Q :----->
                                          :            :
                                          :            :
                                          :            :
                                    ----->: R        Q':----->
                                          :            :
                                          :............:

          FLIP-FLOP A

               SA = C'D' / AB + AD' + B'C'

               RA = B'D / A'C + A'D + B'C'

          FLIP-FLOP B

               SB = A'D' / BD + C'

               RB = ACD' / B'D + B'C'

          FLIP-FLOP C

               SC = AD / B' + CD'

               RC = A'BD / C'D' + B'C'

          FLIP-FLOP D

               SD = A'BC + AC' + AB' / B'C'

               RD = A'B' + A'C' + ABC / B'C'
```

Figure 4.2. Printout for Counter Design Example 4.1.1 (Sheet 5 of 13)

```
                              D      FLIP-FLOP TRANSITION INPUT REQUIREMENTS

                         0 ----> 0     0 ----> 1     1 ----> 0     1 ----> 1
                           D             D             D             D
                           0             1             0             1

                         '1' = TRUE,  '0' = FALSE,  '-' = DON'T CARE


                              TRUTH TABLE FOR COUNTER USING D      FLIP-FLOPS

          NO     A B C D    DA DB DC DD

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

Figure 4.2. Printout for Counter Design Example 4.1.1 (Sheet 6 of 13)

```
                               D FLIP-FLOP INPUT EQUATIONS
                                 (F = TRUE TERMS / DON'T CARE TERMS)

                                 CLOCK ---.
                                          V
                               .............
                               :           :
                               :         Q :---->
                               :           :
                          ---->: D         :
                               :           :
                               :        Q' :---->
                               :           :
                               .............

      FLIP-FLOP A

          DA = C'D' + AB + AD' / B'C'

      FLIP-FLOP B

          DB = A'D' + BD + C'

      FLIP-FLOP C

          DC = B' + CD' + AD

      FLIP-FLOP D

          DD = A'BC + AC' + AB' / B'C'
```

Figure 4.2. Printout for Counter Design Example 4.1.1 (Sheet 7 of 13)

```
                T      FLIP-FLOP TRANSITION INPUT REQUIREMENTS

              0 ----> 0     0 ----> 1     1 ----> 0     1 ----> 1
                T             T             T             T
                0             1             1             0

              '1' = TRUE,   '0' = FALSE,   '-' = DON'T CARE


                TRUTH TABLE FOR COUNTER USING T      FLIP-FLOPS

       NO     A B C D    TA TB TC TD

        3     0 0 1 1    0  0  0  1

        2     0 0 1 0    0  1  0  0

        6     0 1 1 0    0  0  0  1

        7     0 1 1 1    0  0  1  0

        5     0 1 0 1    0  0  0  1

        4     0 1 0 0    1  0  0  0

       12     1 1 0 0    0  0  0  1

       13     1 1 0 1    0  0  1  0

       15     1 1 1 1    0  0  0  1

       14     1 1 1 0    0  1  0  0

       10     1 0 1 0    0  0  0  1

       11     1 0 1 1    1  0  0  0

        3     0 0 1 1    0  0  0  0

        0     0 0 0 0    -  -  -  -

        1     0 0 0 1    -  -  -  -

        8     1 0 0 0    -  -  -  -

        9     1 0 0 1    -  -  -  -
```

Figure 4.2. Printout for Counter Design Example 4.1.1 (Sheet 8 of 13)

```
                              T FLIP-FLOP INPUT EQUATIONS
                              (F = TRUE TERMS / DON'T CARE TERMS)

                              CLOCK ---.
                                       V
                                ...............
                                :             :
                                :         Q   :---->
                                :             :
                           ---->:  T          :
                                :             :
                                :         Q'  :---->
                                :             :
                                :.............:

     FLIP-FLOP A

        TA = A'C'D' + AB'D / B'C'

     FLIP-FLOP B

        TB = A'B'D' + ABCD' / B'C'

     FLIP-FLOP C

        TC = A'BCD + AC'D / B'C'

     FLIP-FLOP D

        TD = A'B'D + A'BCD' + A'C'D + AC'D' + ABCD + AB'D' / B'C'
```

Figure 4.2. Printout for Counter Design Example 4.1.1 (Sheet 9 of 13)

```
                         R S T FLIP-FLOP TRANSITION INPUT REQUIREMENTS

                   0 ----> 0     0 ----> 1     1 ----> 0     1 ----> 1
                    S R T         S R T         S R T         S R T
                    0 - 0         1 0 -         0 1 -         - 0 0

                   '1' = TRUE,  '0' = FALSE,  '-' = DON'T CARE

                         TRUTH TABLE FOR COUNTER USING R S T FLIP-FLOPS

      NO     A B C D   SA RA TA SB RB TB SC RC TC SD RD TD

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

Figure 4.2. Printout for Counter Design Example 4.1.1 (Sheet 10 of 13).

```
                    R-S-T FLIP-FLOP INPUT EQUATIONS
                    (F = TRUE TERMS / DON'T CARE TERMS)

                    CLOCK ---.
                             V
                     ..............
                     :            :
                ---->: S        Q :---->
                     :            :
                ---->: T          :
                     :            :
                ---->: R        Q':---->
                     :            :
                     :............:

     FLIP-FLOP A

        SA = C'D' / AB + AD' + B'C'

        RA = B'D / A'C + A'D + B'C'

        TA = 0 / A'C'D' + AB'D + B'C'

     FLIP-FLOP B

        SB = A'D' / BD + C'

        RB = ACD' / B'D + B'C'

        TB = 0 / A'B'D' + ABCD' + B'C'

     FLIP-FLOP C

        SC = AD / B' + CD'

        RC = A'BD / C'D' + B'C'

        TC = 0 / A'BCD + AC'D + B'C'

     FLIP-FLOP D

        SD = A'BC + AC' + AB' / B'C'

        RD = A'B' + A'C' + ABC / B'C'

        TD = 0 / A'B'D + A'BCD' + A'C'D + AC'D' + ABCD + AB'D' + B'C'
```

Figure 4.2. Printout for Counter Design Example 4.1.1 (Sheet 11 of 13)

```
                    R S T FLIP-FLOP TRANSITION INPUT REQUIREMENTS

                  0 ----> 0     0 ----> 1     1 ----> 0     1 ----> 1
                    S R T         S R T         S R T         S R T
                    0 - 0         - 0 1         0 - 1         - 0 0

                  '1' = TRUE,  '0' = FALSE,  '-' = DON'T CARE

                  TRUTH TABLE FOR COUNTER USING R S T FLIP-FLOPS

        NO     A B C D   SA RA TA SB RB TB SC RC TC SD RD TD

         3     0 0 1 1   0  -  0  0  -  0  -  0  0  0  -  1

         2     0 0 1 0   0  -  0  -  0  1  -  0  0  0  -  0

         6     0 1 1 0   0  -  0  -  0  0  -  0  0  -  0  1

         7     0 1 1 1   0  -  0  -  0  0  0  -  1  -  0  0

         5     0 1 0 1   0  -  0  -  0  0  0  -  0  0  -  1

         4     0 1 0 0   -  0  1  -  0  0  0  -  0  0  -  0

        12     1 1 0 0   -  0  0  -  0  0  0  -  0  -  0  1

        13     1 1 0 1   -  0  0  -  0  0  -  0  1  -  0  0

        15     1 1 1 1   -  0  0  -  0  0  -  0  0  0  -  1

        14     1 1 1 0   -  0  0  0  -  1  -  0  0  0  -  0

        10     1 0 1 0   -  0  0  0  -  0  -  0  0  -  0  1

        11     1 0 1 1   0  -  1  0  -  0  -  0  0  -  0  0

         3     0 0 1 1   0  0  0  0  0  0  0  0  0  0  0  0

         0     0 0 0 0   -  -  -  -  -  -  -  -  -  -  -  -

         1     0 0 0 1   -  -  -  -  -  -  -  -  -  -  -  -

         8     1 0 0 0   -  -  -  -  -  -  -  -  -  -  -  -

         9     1 0 0 1   -  -  -  -  -  -  -  -  -  -  -  -
```

Figure 4.2. Printout for Counter Design Example 4.1.1 (Sheet 12 of 13)

```
                            R-S-T FLIP-FLOP INPUT EQUATIONS
                               (F = TRUE TERMS / DON'T CARE TERMS)

                               CLOCK ---.
                                        V
                               ..............
                               :            :
                          ---->: S        Q :---->
                               :            :
                          ---->: T          :
                               :            :
                          ---->: R        Q':---->
                               :            :
                               ..............

        FLIP-FLOP A

           SA = 0 / C'D' + AB + AD' + B'C'

           RA = 0 / B'D + A'C + A'D + B'C'

           TA = A'C'D' + AB'D / B'C'

        FLIP-FLOP B

           SB = 0 / A'D' + BD + C'

           RB = 0 / B'D + ACD' + B'C'

           TB = A'B'D' + ABCD' / B'C'

        FLIP-FLOP C

           SC = 0 / B' + CD' + AD

           RC = 0 / A'BD + C'D' + B'C'

           TC = A'BCD + AC'D / B'C'

        FLIP-FLOP D

           SD = 0 / A'BC + AC' + AB' + B'C'

           RD = 0 / A'B' + A'C' + ABC + B'C'

           TD = A'B'D + A'BCD' + A'C'D + AC'D' + ABCD + AB'D' / B'C'
```

Figure 4.2. Printout for Counter Design Example 4.1.1 (Sheet 13 of 13)

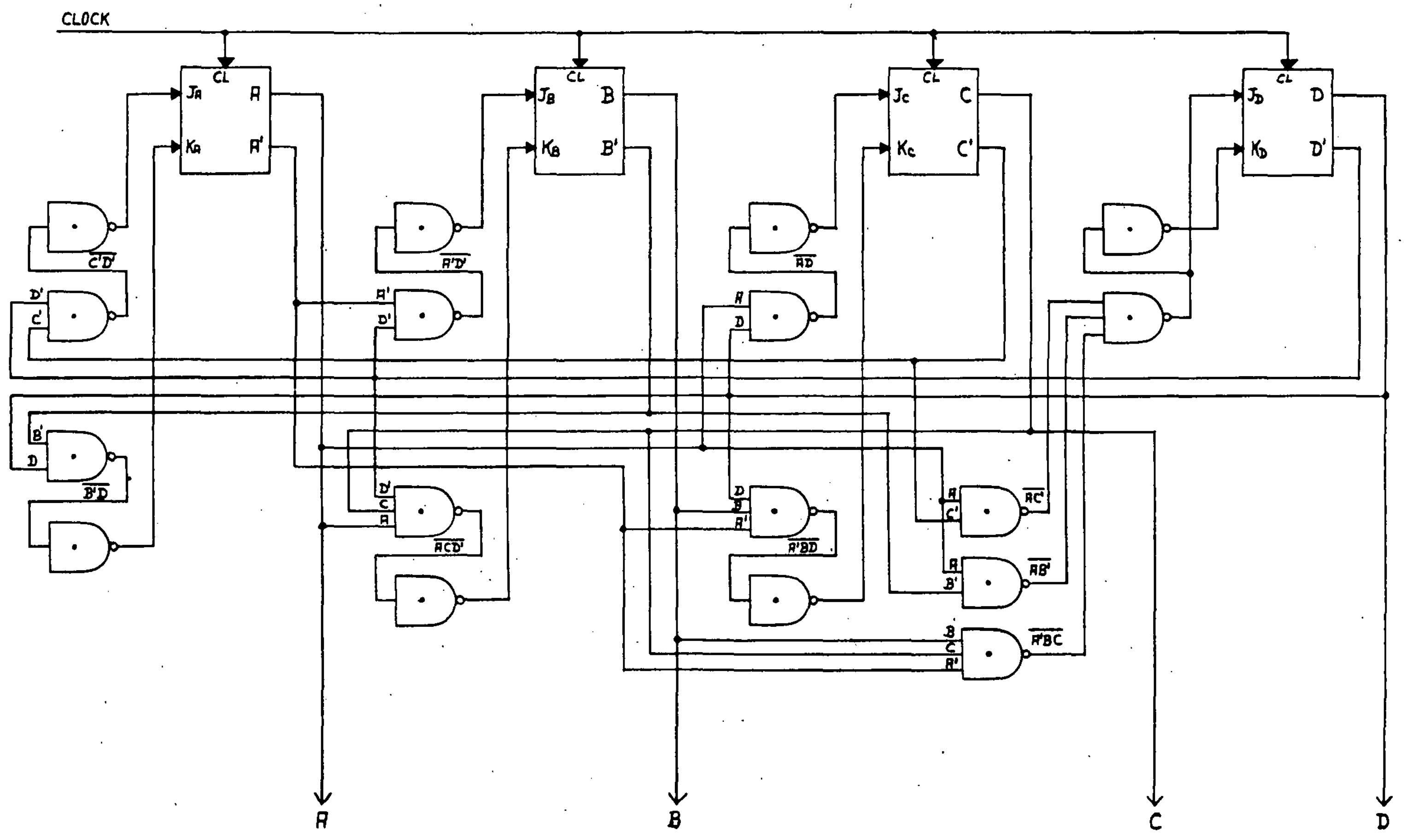


Figure 4.3. Schematic of JK Flip Flop Counter Using NAND Gates for Example 4.1.1

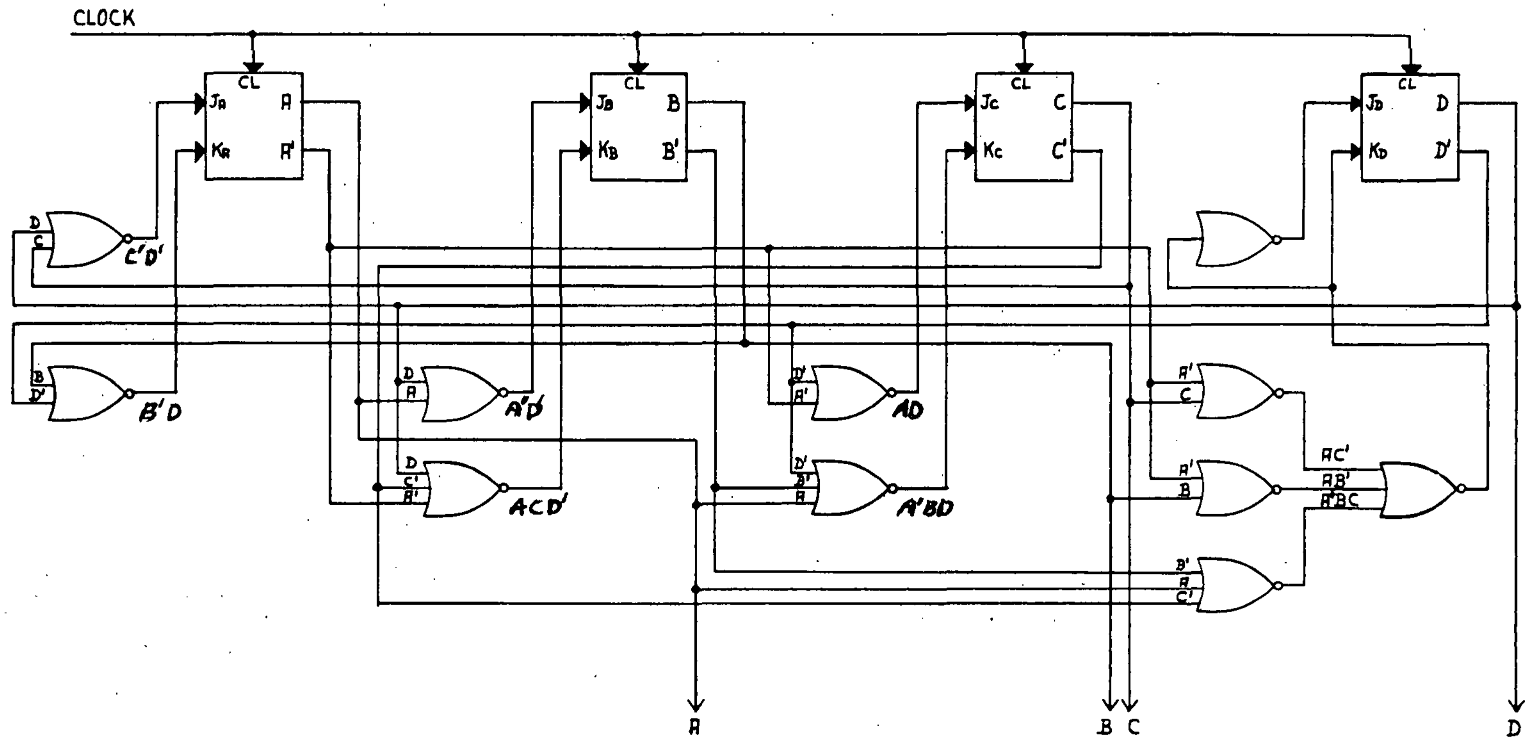


Figure 4.4. Schematic of JK Flip Flop Counter Using NOR Gates for Example 4.1.1

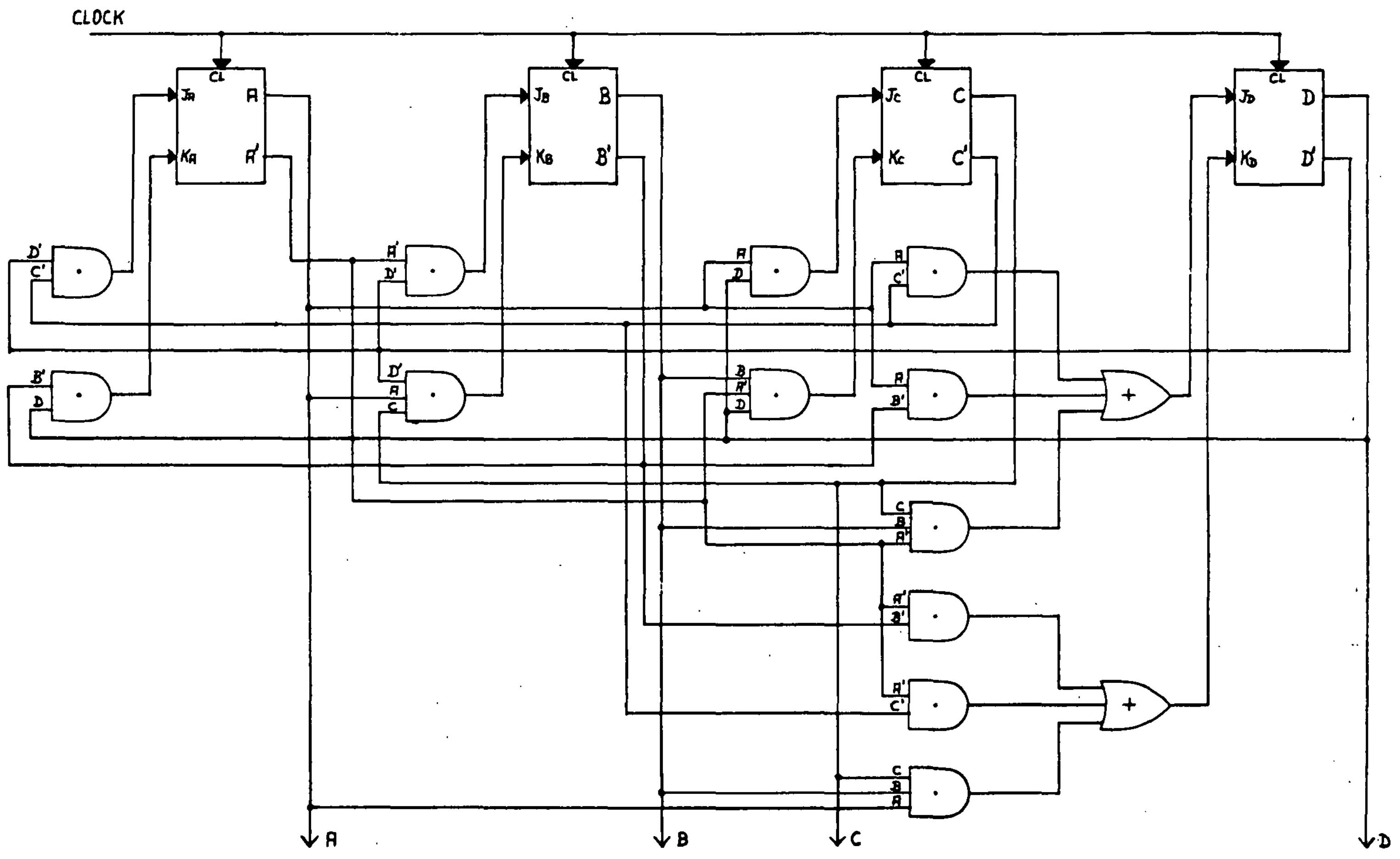


Figure 4.5 Schematic for JK Flip Flop Counter Using AND and OR Gates for Example 4.1.1

the product-of-sum form. This is demonstrated in Section 4.4 which shows how sum-of-product and product-of-sum equations directly provide the inputs to NAND and NOR gates respectively.

4.2 COUNTER, 40 COUNT 6 STAGE

This example is for a 40 state 6 bit counter meaning a 6 flip flop counter having 40 repetitive count states. The sequence used is a cyclic code meaning the binary representation of each successive decimal number differs by only one binary digit.

The program for the count sequence is run for the following two cases given in Sections 4.2.1 and 4.2.2.

(1) JK, RS, D, T, RST flip flops
Sum-of-product input equations
Short form output

(2) JK, RS, D, T flip flops
Product-of-sum input equations
Short form output

The input data cards for these two cases are given in Figure 4.6.

The input equations obtained in sum-of-product form and product-of-sum form are not always equal to each other. This is because there may be more than one correct solution.

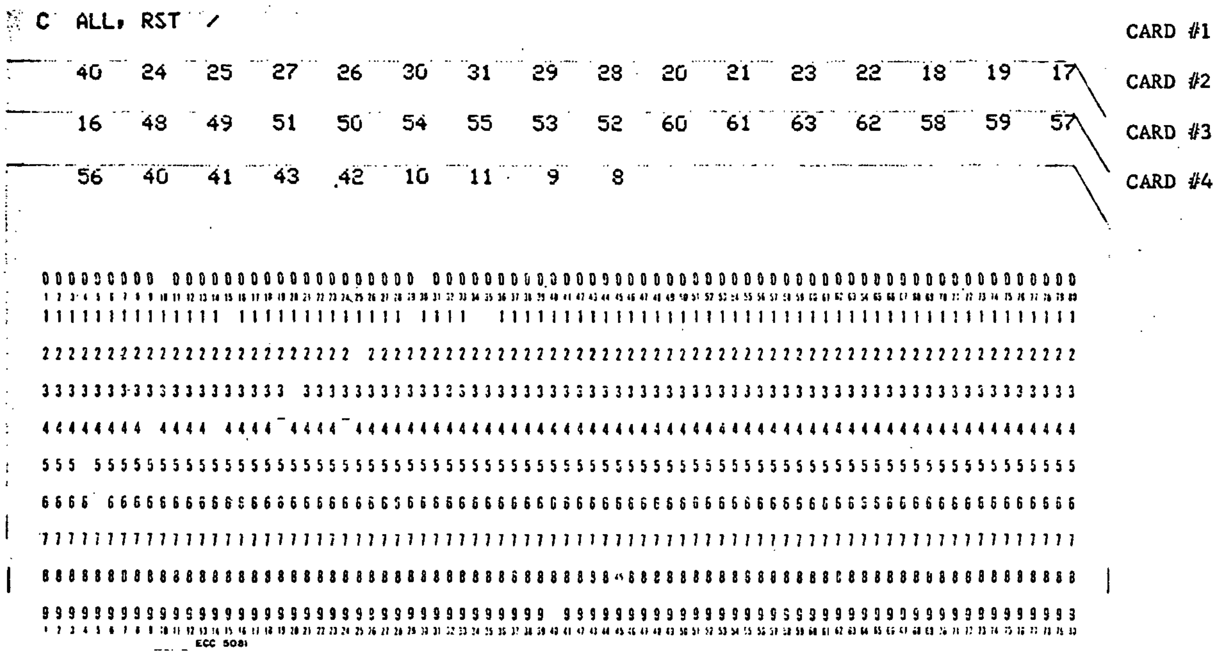
4.2.1 ALL, RST /

The flip flop code letters ALL, RST specify the JK, RS, D, T, and RST flip flops, and the output option / (slash mark) specifies the short form output option. The counter logic equations are calculated for each type flip flop specified. The short form output option provides printout of only the input data and input count sequence logic, and the input equations (true terms only) for each flip flop type specified. The short form option suppresses printout of the flip flop transition input requirements, the counter truth table, and the don't care terms of the flip flop input equations.

The computer printout for this case (data cards of Figure 4.6A) is shown in Figure 4.7.

4.2.2 ALL, P /

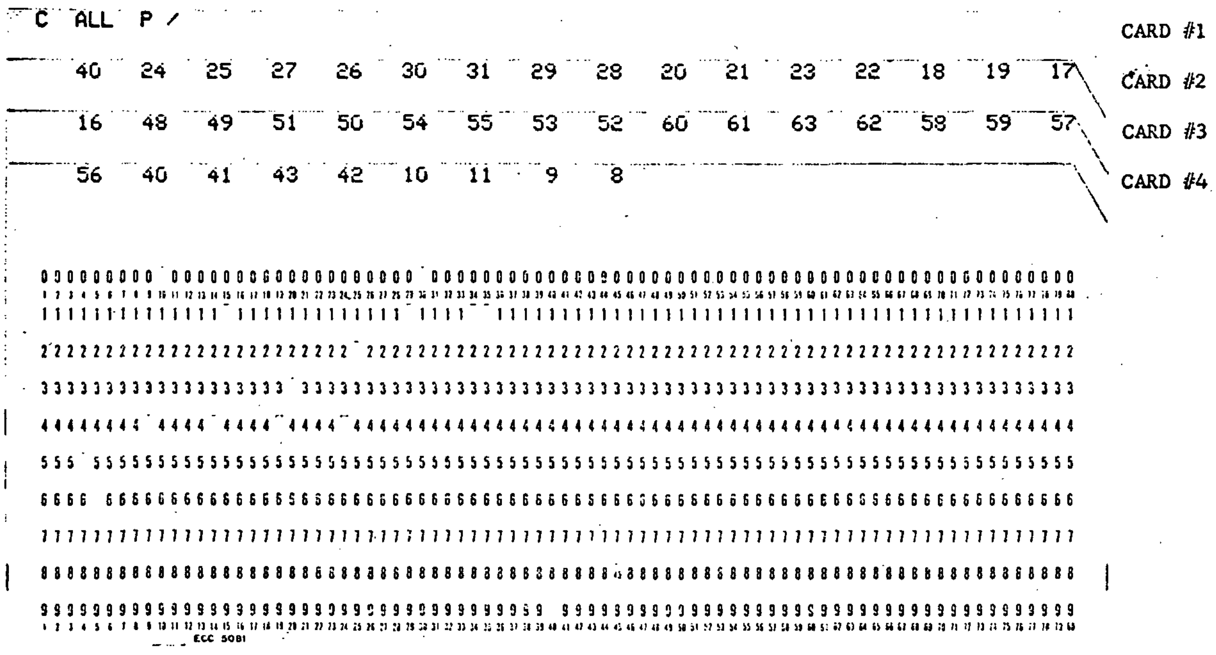
The flip flop code letters ALL specify the JK, RS, D, and T flip flops. Output option P specifies that the flip flop input equations are to be printed out in product-of-sum form and option / specifies the short form output as in Section 4.2.1. The data cards for this case are shown in Figure 4.6B. The computer printout for these data cards is given in Figure 4.8.



```
C ALL, RST /                                                        CARD #1
  40  24  25  27  26  30  31  29  28  20  21  23  22  18  19  17    CARD #2
  16  48  49  51  50  54  55  53  52  60  61  63  62  58  59  57    CARD #3
  56  40  41  43  42  10  11   9   8                                CARD #4
```

ECC 5081

A. Data Cards for Example 4.2.1



```
C ALL P /                                                           CARD #1
  40  24  25  27  26  30  31  29  28  20  21  23  22  18  19  17    CARD #2
  16  48  49  51  50  54  55  53  52  60  61  63  62  58  59  57    CARD #3
  56  40  41  43  42  10  11   9   8                                CARD #4
```

ECC 5081

B. Data Cards for Example 4.2.2

Figure 4.6. Data Cards for 40 Count 6 Stage Counter for Example 4.2.1 and Example 4.2.2

```
FLIP-FLOP TYPES SPECIFIED FOR COUNTER

FLIP-FLOPS :   ALL, RST  /


INPUT COUNT DATA, FIRST NO. IS NUMBER OF COUNTS
    40  24  25  27  26  30  31  29  28  20  21  23  22  18  19  17
    16  48  49  51  50  54  55  53  52  60  61  63  62  58  59  57
    56  40  41  43  42  10  11   9   8


         INPUT COUNT SEQUENCE LOGIC

               I    NO(I)   A B C D E F
               1     24     0 1 1 0 0 0
               2     25     0 1 1 0 0 1
               3     27     0 1 1 0 1 1
               4     26     0 1 1 0 1 0
               5     30     0 1 1 1 1 0
               6     31     0 1 1 1 1 1
               7     29     0 1 1 1 0 1
               8     28     0 1 1 1 0 0
               9     20     0 1 0 1 0 0
              10     21     0 1 0 1 0 1
              11     23     0 1 0 1 1 1
              12     22     0 1 0 1 1 0
              13     18     0 1 0 0 1 0
              14     19     0 1 0 0 1 1
              15     17     0 1 0 0 0 1
              16     16     0 1 0 0 0 0
              17     48     1 1 0 0 0 0
```

Figure 4.7. Printout for Counter Design Example 4.2.1 (Sheet 1 of 9)

```
     18      49     1 1 0 0 0 1
     19      51     1 1 0 0 1 1
     20      50     1 1 0 0 1 0
     21      54     1 1 0 1 1 0
     22      55     1 1 0 1 1 1
     23      53     1 1 0 1 0 1
     24      52     1 1 0 1 0 0
     25      60     1 1 1 1 0 0
     26      61     1 1 1 1 0 1
     27      63     1 1 1 1 1 1
     28      62     1 1 1 1 1 0
     29      58     1 1 1 0 1 0
     30      59     1 1 1 0 1 1
     31      57     1 1 1 0 0 1
     32      56     1 1 1 0 0 0
     33      40     1 0 1 0 0 0
     34      41     1 0 1 0 0 1
     35      43     1 0 1 0 1 1
     36      42     1 0 1 0 1 0
     37      10     0 0 1 0 1 0
     38      11     0 0 1 0 1 1
     39       9     0 0 1 0 0 1
     40       8     0 0 1 0 0 0
     41      24     0 1 1 0 0 0
   **************DON'T CARE TERMS BELOW
     42       0     0 0 0 0 0 0
     43       1     0 0 0 0 0 1
     44       2     0 0 0 0 1 0
```

Figure 4.7. Printout for Counter Design Example 4.2.1 (Sheet 2 of 9)

```
45     3    0 0 0 0 1 1
46     4    0 0 0 1 0 0
47     5    0 0 0 1 0 1
48     6    0 0 0 1 1 0
49     7    0 0 0 1 1 1
50    12    0 0 1 1 0 0
51    13    0 0 1 1 0 1
52    14    0 0 1 1 1 0
53    15    0 0 1 1 1 1
54    32    1 0 0 0 0 0
55    33    1 0 0 0 0 1
56    34    1 0 0 0 1 0
57    35    1 0 0 0 1 1
58    36    1 0 0 1 0 0
59    37    1 0 0 1 0 1
60    38    1 0 0 1 1 0
61    39    1 0 0 1 1 1
62    44    1 0 1 1 0 0
63    45    1 0 1 1 0 1
64    46    1 0 1 1 1 0
65    47    1 0 1 1 1 1
```

Figure 4.7. Printout for Counter Design Example 4.2.1 (Sheet 3 of 9)

```
                          J K   FLIP-FLOP INPUT EQUATIONS
                                (F = TRUE TERMS / DON'T CARE TERMS)

                                CLOCK ---.
                                         V
                                  ..............
                                  :            :
                             ---->: J      Q   :---->
                                  :            :
                                  :            :
                                  :            :
                             ---->: K      Q'  :---->
                                  :            :
                                  :............:

   FLIP-FLOP A

      JA = C'D'E'F'

      KA = B'EF'

   FLIP-FLOP B

      JB = A'E'F'

      KB = ACD'E'F'

   FLIP-FLOP C

      JC = ADE'F'

      KC = A'DE'F'

   FLIP-FLOP D

      JD = A'BCEF' + AC'EF'

      KD = A'C'EF' + ACEF'

   FLIP-FLOP E

      JE = A'BCD'F + A'C'DF + AC'D'F + ACDF + AB'F

      KE = A'CDF + A'C'D'F + AC'DF + ABCD'F + A'B'F

   FLIP-FLOP F

      JF = A'BCD'E' + A'CDE + A'C'DE' + A'C'D'E + AC'D'E' + AC'DE + ACDE' + ABCD'E + AB'E'
           + A'B'E

      KF = A'BCD'E + A'CDE' + A'C'DE + A'C'D'E' + AC'D'E + AC'DE' + ACDE + ABCD'E' + AB'E
           + A'B'E'
```

Figure 4.7. Printout for Counter Design Example 4.2.1 (Sheet 4 of 9)

```
                              R S   FLIP-FLOP INPUT EQUATIONS
                                    (F = TRUE TERMS / DON'T CARE TERMS)


                                    CLOCK ---,
                                             V
                                    .............
                                    :           :
                              ----> : S       Q : ---->
                                    :           :
                                    :           :
                                    :           :
                              ----> : R       Q': ---->
                                    :           :
                                    :...........:


   FLIP-FLOP A

       SA = C'D'E'F'

       RA = B'EF'

   FLIP-FLOP B

       SB = A'E'F'

       RB = ACD'E'F'

   FLIP-FLOP C

       SC = ADE'F'

       RC = A'DE'F'

   FLIP-FLOP D

       SD = A'BCEF' + AC'EF'

       RD = A'C'EF' + ACEF'

   FLIP-FLOP E

       SE = A'BCD'F + A'C'DF + AC'D'F + ACDF + AB'F

       RE = A'CDF + A'C'D'F + AC'DF + ABCD'F + A'B'F

   FLIP-FLOP F

       SF = A'BCD'E' + A'CDE + A'C'DE' + A'C'D'E + AC'D'E' + AC'DE + ACDE' + ABCD'E + AB'E'
            + A'B'E

       RF = A'BCD'E + A'CDE' + A'C'DE + A'C'D'E' + AC'D'E + AC'DE' + ACDE + ABCD'E' + AB'E
            + A'B'E'
```

Figure 4.7. Printout for Counter Design Example 4.2.1 (Sheet 5 of 9)

```
                              D   FLIP-FLOP INPUT EQUATIONS
                                  (F = TRUE TERMS / DON'T CARE TERMS)

                                  CLOCK ---.
                                           V
                                  ..............
                                  :            :
                                  :          Q :---->
                                  :            :
                             ---->: D          :
                                  :            :
                                  :          Q':---->
                                  :            :
                                  :............:

FLIP-FLOP A

    DA = C'D'E'F' + ABF + ABE + AD + ACF + ACE'

FLIP-FLOP B

    DB = A'E'F' + BF + BE + D + C'

FLIP-FLOP C

    DC = CD' + CE + CF + ADE'F'

FLIP-FLOP D

    DD = A'BCEF' + DF + DE' + AC'EF'

FLIP-FLOP E

    DE = A'BCD'F + EF' + A'C'DF + AC'D'F + ACDF + AB'F

FLIP-FLOP F

    DF = A'BCD'E' + A'CDE + A'C'DE' + A'C'D'E + AC'D'E' + AC'DE + ACDE' + ABCD'E + AB'E'
         + A'B'E
```

Figure 4.7. Printout for Counter Design Example 4.2.1 (Sheet 6 of 9)

```
                              FLIP-FLOP INPUT EQUATIONS
                              (F = TRUE TERMS / DON'T CARE TERMS)

                              CLOCK ---.
                                       V
                              ...............
                              :             :
                              :           Q :---->
                              :             :
                         ---->: T           :
                              :             :
                              :          Q' :---->
                              :             :
                              :.............:

    FLIP-FLOP A

         TA = A'C'D'E'F' + AB'EF'

    FLIP-FLOP B

         TB = ABCD'E'F' + A'B'E'F'

    FLIP-FLOP C

         TC = A'CDE'F' + AC'DE'F'

    FLIP-FLOP D

         TD = A'BCD'EF' + A'C'DEF' + AC'D'EF' + ACDEF'

    FLIP-FLOP E

         TE = A'BCD'E'F + A'CDEF + A'C'DE'F + A'C'D'EF + AC'D'E'F + AC'DEF + ACDE'F + ABCD'EF
              + AB'E'F + A'B'EF

    FLIP-FLOP F

         TF = A'BCD'E'F' + A'BCD'EF + A'CDEF' + A'CDE'F + A'C'DE'F' + A'C'DEF + A'C'D'EF' + A'C'D'E'F
              + AC'D'E'F' + AC'D'EF + AC'DEF' + AC'DE'F + ACDE'F' + ACDEF + ABCD'EF' + ABCD'E'F
              + AB'E'F' + AB'EF + A'B'EF' + A'B'E'F
```

Figure 4.7. Printout for Counter Design Example 4.2.1 (Sheet 7 of 9)

```
                    R.S.T FLIP-FLOP INPUT EQUATIONS
                    (F = TRUE TERMS / DON'T CARE TERMS)

                         CLOCK ---,
                                  V
                         '''''''''''''''
                         :             :
                    ---->: S         Q :---->
                         :             :
                    ---->: T           :
                         :             :
                    ---->: R         Q':---->
                         :             :
                         :.............:

     FLIP-FLOP A

        SA = C'D'E'F'

        RA = B'EF'

        TA = 0

     FLIP-FLOP B

        SB = A'E'F'

        RB = ACD'E'F'

        TB = 0

     FLIP-FLOP C

        SC = ADE'F'

        RC = A'DE'F'

        TC = 0

     FLIP-FLOP D

        SD = A'BCEF' + AC'EF'

        RD = A'C'EF' + ACEF'

        TD = 0

     FLIP-FLOP E

        SE = A'BCD'F + A'C'DF + AC'D'F + ACDF + AB'F

        RE = A'CDF + A'C'D'F + AC'DF + ABCD'F + A'B'F

        TE = 0

     FLIP-FLOP F

        SF = A'BCD'E' + A'CDE + A'C'DE' + A'C'D'E + AC'D'E' + AC'DE + ACDE' + ABCD'E + AB'E'
             + A'B'E

        RF = A'BCD'E + A'CDE' + A'C'DE + A'C'D'E' + AC'D'E + AC'DE' + ACDE + ABCD'E' + AB'E
             + A'B'E'

        TF = 0
```

Figure 4.7. Printout for Counter Design Example 4.2.1 (Sheet 8 of 9)

```
                    R S T FLIP-FLOP INPUT EQUATIONS
                        (F = TRUE TERMS / DON'T CARE TERMS)

                          CLOCK ---.
                                   V
                        ..............
                        :            :
                   ---->: S        Q :---->
                        :            :
                   ---->: T          :
                        :            :
                   ---->: R        Q' :---->
                        :            :
                        :............:

   FLIP-FLOP A

      SA = 0

      RA = 0

      TA = A'C'D'E'F' + AB'EF'

   FLIP-FLOP B

      SB = 0

      RB = 0

      TB = ABCD'E'F' + A'B'E'F'

   FLIP-FLOP C

      SC = 0

      RC = 0

      TC = A'CDE'F' + AC'DE'F'

   FLIP-FLOP D

      SD = 0

      RD = 0

      TD = A'BCD'EF' + A'C'DEF' + AC'D'EF' + ACDEF'

   FLIP-FLOP E

      SE = 0

      RE = 0

      TE = A'BCD'E'F + A'CDEF + A'C'DE'F + A'C'D'EF + AC'D'E'F + AC'DEF + ACDE'F + ABCD'EF
           + AB'E'F + A'B'EF

   FLIP-FLOP F

      SF = 0

      RF = 0

      TF = A'BCD'E'F' + A'BCD'EF + A'CDEF' + A'CDE'F + A'C'DE'F' + A'C'DEF + A'C'D'EF' + A'C'D'E'F
           + AC'D'E'F' + AC'D'EF + AC'DEF' + AC'DE'F + ACDE'F' + ACDEF + ABCD'EF' + ABCD'E'F
           + AB'E'F' + AB'EF + A'B'EF' + A'B'E'F
```

Figure 4.7. Printout for Counter Design Example 4.2.1 (Sheet 9 of 9)

```
FLIP-FLOP TYPES SPECIFIED FOR COUNTER
FLIP-FLOPS :  ALL  P /

INPUT COUNT DATA, FIRST NO. IS NUMBER OF COUNTS
  40   24   25   27   26   30   31   29   28   20   21   23   22   18   19   17
  16   48   49   51   50   54   55   53   52   60   61   63   62   58   59   57
  56   40   41   43   42   10   11    9    8

         INPUT COUNT SEQUENCE LOGIC

           I       NO(I)    A B C D E F
           1        24      0 1 1 0 0 0
           2        25      0 1 1 0 0 1
           3        27      0 1 1 0 1 1
           4        26      0 1 1 0 1 0
           5        30      0 1 1 1 1 0
           6        31      0 1 1 1 1 1
           7        29      0 1 1 1 0 1
           8        28      0 1 1 1 0 0
           9        20      0 1 0 1 0 0
          10        21      0 1 0 1 0 1
          11        23      0 1 0 1 1 1
          12        22      0 1 0 1 1 0
          13        18      0 1 0 0 1 0
          14        19      0 1 0 0 1 1
          15        17      0 1 0 0 0 1
          16        16      0 1 0 0 0 0
          17        48      1 1 0 0 0 0
```

Figure 4.8. Printout for Counter Design Example 4.2.2 (Sheet 1 of 7)

```
 18        49      1 1 0 0 0 1
 19        51      1 1 0 0 1 1
 20        50      1 1 0 0 1 0
 21        54      1 1 0 1 1 0
 22        55      1 1 0 1 1 1
 23        53      1 1 0 1 0 1
 24        52      1 1 0 1 0 0
 25        60      1 1 1 1 0 0
 26        61      1 1 1 1 0 1
 27        63      1 1 1 1 1 1
 28        62      1 1 1 1 1 0
 29        58      1 1 1 0 1 0
 30        59      1 1 1 0 1 1
 31        57      1 1 1 0 0 1
 32        56      1 1 1 0 0 0
 33        40      1 0 1 0 0 0
 34        41      1 0 1 0 0 1
 35        43      1 0 1 0 1 1
 36        42      1 0 1 0 1 0
 37        10      0 0 1 0 1 0
 38        11      0 0 1 0 1 1
 39         9      0 0 1 0 0 1
 40         8      0 0 1 0 0 0
 41        24      0 1 1 0 0 0
--------------- DON'T CARE TERMS BELOW
 42         0      0 0 0 0 0 0
 43         1      0 0 0 0 0 1
 44         2      0 0 0 0 1 0
```

Figure 4.8. Printout for Counter Design Example 4.2.2 (Sheet 2 of 7)

```
45     3    0 0 0 0 1 1
46     4    0 0 0 1 0 0
47     5    0 0 0 1 0 1
48     6    0 0 0 1 1 0
49     7    0 0 0 1 1 1
50    12    0 0 1 1 0 0
51    13    0 0 1 1 0 1
52    14    0 0 1 1 1 0
53    15    0 0 1 1 1 1
54    32    1 0 0 0 0 0
55    33    1 0 0 0 0 1
56    34    1 0 0 0 1 0
57    35    1 0 0 0 1 1
58    36    1 0 0 1 0 0
59    37    1 0 0 1 0 1
60    38    1 0 0 1 1 0
61    39    1 0 0 1 1 1
62    44    1 0 1 1 0 0
63    45    1 0 1 1 0 1
64    46    1 0 1 1 1 0
65    47    1 0 1 1 1 1
```

Figure 4.8. Printout for Counter Design Example 4.2.2 (Sheet 3 of 7)

```
                              J-K  FLIP-FLOP INPUT EQUATIONS
                              (F = TRUE TERMS / DON'T CARE TERMS)

                                   CLOCK ----,
                                             V
                                  ...............
                                  :             :
                            ----->: J         Q :----->
                                  :             :
                                  :             :
                                  :             :
                            ----->: K         Q':----->
                                  :             :
                                  :.............:

FLIP-FLOP A

    JA = (C')(E')(F')(D')

    KA = (B')(F')(E)

FLIP-FLOP B

    JB = (A')(F')(E')

    KB = (A)(F')(E')(D')(C)

FLIP-FLOP C

    JC = (A)(F')(E')(D)

    KC = (D)(E')(F')(A')

FLIP-FLOP D

    JD = (E)(F')(A+C)(A'+C')(B)

    KD = (A+C')(F')(E)(A'+C)

FLIP-FLOP E

    JE = (F)(A+C'+D')(A+C+D)(A'+C+D')(A'+B'+C'+D)(A+B)

    KE = (A+B'+C'+D)(F)(A+C+D')(A'+C+D)(A'+C'+D')(A'+B)

FLIP-FLOP F

    JF = (A+B'+C'+D+E')(A+C'+D'+E)(A+C+D'+E')(A+C+D+E)(A'+C+D+E')(A'+C+D'+E)(A'+C'+D'+E')
         (A'+B'+C'+D+E)(A'+B+E')(A+B+E)

    KF = (A+B'+C'+D+E)(A+C'+D'+E')(A+C+D'+E)(A+C+D+E')(A'+C+D+E)(A'+C+D'+E')(A'+C'+D'+E)
         (A'+B'+C'+D+E')(A'+B+E)(A+B+E')
```

Figure 4.8. Printout for Counter Design Example 4.2.2 (Sheet 4 of 7)

```
                              R-S  FLIP-FLOP INPUT EQUATIONS
                                   (F = TRUE TERMS / DON'T CARE TERMS)

                                   CLOCK ----.
                                             V
                                   ...............
                                   :             :
                              ---->: S         Q :---->
                                   :             :
                                   :             :
                                   :             :
                              ---->: R        Q' :---->
                                   :             :
                                   :.............:

FLIP-FLOP A

    SA = (C')(E')(F')(D')

    RA = (E)(F')(B')

FLIP-FLOP B

    SB = (A')(F')(E')

    RB = (A)(F')(E')(D')(C)

FLIP-FLOP C

    SC = (A)(F')(E')(D)

    RC = (D)(E')(F')(A')

FLIP-FLOP D

    SD = (E)(F')(A+C)(A'+C')(B)

    RD = (A+C')(F')(E)(A'+C)

FLIP-FLOP E

    SE = (F)(A+C'+D')(A+C+D)(A'+C+D')(A'+B'+C'+D)(A+B)

    RE = (A+B'+C'+D)(F)(A+C+D')(A'+C+D)(A'+C'+D')(A'+B)

FLIP-FLOP F

    SF = (A+B'+C'+D+E')(A+C'+D'+E)(A+C+D'+E')(A+C+D+E)(A'+C+D+E')(A'+C+D'+E)(A'+C'+D'+E')
         (A'+B'+C'+D+E)(A'+B+E')(A+B+E)

    RF = (A+B'+C'+D+E)(A+C'+D'+E')(A+C+D'+E)(A+C+D+E')(A'+C+D+E)(A'+C+D'+E')(A'+C'+D'+E)
         (A'+B'+C'+D+E')(A'+B+E)(A+B+E')
```

Figure 4.8. Printout for Counter Design Example 4.2.2 (Sheet 5 of 7)

```
                          D    FLIP-FLOP INPUT EQUATIONS
                               (F = TRUE TERMS / DON'T CARE TERMS)

                               CLOCK ----.
                                         V
                                ..............
                                :            :
                                :          Q :---->
                                :            :
                           ---->: D          :
                                :            :
                                :          Q':---->
                                :            :
                                :............:

  FLIP-FLOP A

     DA = (A+C')(A+B'+E')(A+B'+F')(A+D')(B+E'+F)

  FLIP-FLOP B

     DB = (A'+C'+D+E+F)(B+F')(B+E')

  FLIP-FLOP C

     DC = (A+D'+E+F)(C+F')(C+E')(C+D)

  FLIP-FLOP D

     DD = (D+E)(D+F')(A+C+E'+F)(A'+C'+E'+F)(B)

  FLIP-FLOP E

     DE = (E+F)(A+C'+D'+F')(A+C+D+F')(A'+C+D'+F')(A'+B'+C'+D+F')(A+B+F')

  FLIP-FLOP F

     DF = (A+B'+C'+D+E')(A+C'+D'+E)(A+C+D'+E')(A+C+D+E)(A'+C+D+E')(A'+C+D'+E)(A'+C'+D'+E')
          (A'+B'+C'+D+E)(A'+B+E')(A+B+E)
```

Figure 4.8. Printout for Counter Design Example 4.2.2 (Sheet 6 of 7)

```
                         T        FLIP-FLOP INPUT EQUATIONS
                                 (F = TRUE TERMS / DON'T CARE TERMS)

                                  CLOCK ---.
                                           V
                                 ..............
                                 :            :
                                 :        Q   :---->
                                 :            :
                            ---->: T          :
                                 :            :
                                 :        Q'  :---->
                                 :            :
                                 :............:

  FLIP-FLOP A

     TA = (C'+E)(F')(B'+E')(D')(A'+B')(A+C')

  FLIP-FLOP B

     TB = (A+B')(F')(E')(D')(C)(A'+B)

  FLIP-FLOP C

     TC = (D)(E')(F')(A+C)(A'+C')

  FLIP-FLOP D

     TD = (E)(F')(A+C'+D')(A+C+D)(A'+C+D')(A'+C'+D)(B)

  FLIP-FLOP E

     TE = (F)(A+B'+C'+D+E')(A+C'+D'+E)(A+C+D'+E')(A+C+D+E)(A'+C+D+E')(A'+C+D'+E)(A'+C'+D'+E')
          (A'+B'+C'+D+E)(A'+B+E')(A+B+E)

  FLIP-FLOP F

     TF = (A+B'+C'+D+E+F')(A+B'+C'+D+E'+F)(A+C'+U'+E'+F')(A+C'+D'+E+F)(A+C+D'+E+F')(A+C+D'+E'+F)
          (A+C+D+E'+F')(A+C+D+E+F)(A'+C+D+E+F')(A'+C+D+E'+F)(A'+C+D'+E'+F')(A'+C+D'+E+F)(A'+C'+D'+E+F')
          (A'+C'+D'+E'+F)(A'+B'+C'+D+E'+F')(A'+B'+C'+D+E+F)(A'+B+E+F')(A'+B+E'+F)(A+B+E'+F')
          (A+B+E+F)
```

Figure 4.8. Printout for Counter Design Example 4.2.2 (Sheet 7 of 7)

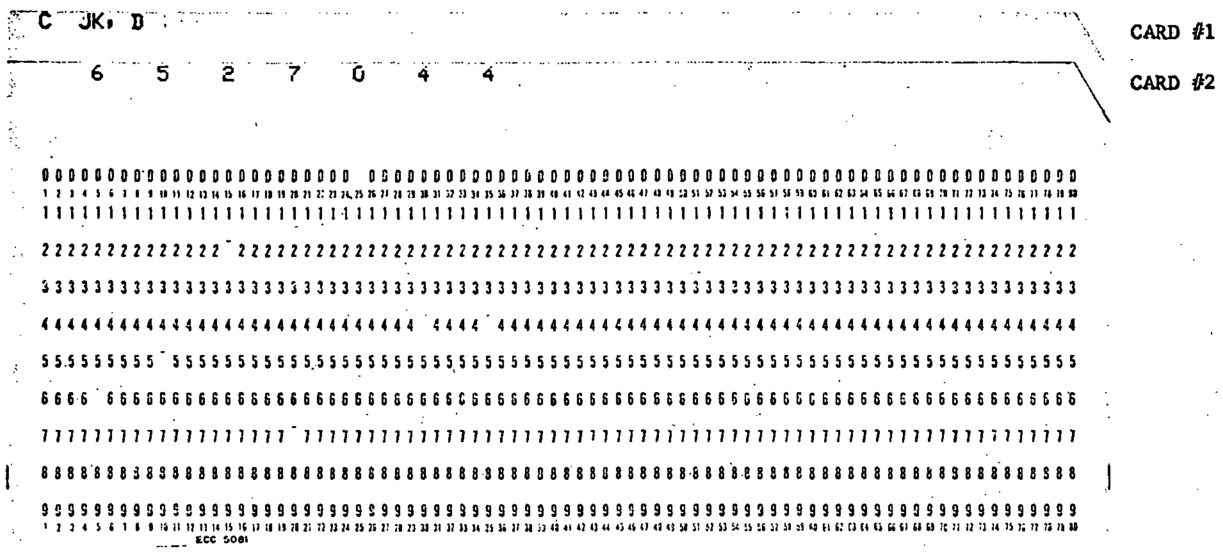
4.3 COUNTER, NON-REPETITIVE, 5 COUNT 3 STAGE

This counter design is for a non-repetitive 5 count sequence using 3 flip flops. The count sequence is 5, 2, 7, 0, 4. Non-repetitive means that the counter stops when the last count of the sequence is reached, for this case count 4. The counter remains in the last count state until reset to the first count or some other count of the sequence. Then the counter sequences from the reset state to the last count state and again stops.

A non-repetitive counter is specified by repeating the last count of the sequence twice. The reason the counter stops is that the flip flop inputs that make a synchronous counter remain in any one state for more than one clock period will permanently keep the counter in that state. Since the last count is repeated twice, the first number on the data card is equal to one plus the number of different count states in the sequence. This is illustrated by the data cards in Figure 4.9.

4.3.1 JK, D

Here the JK and Delay flip flops are specified. The input data cards for design of the non-repetitive 5 state counter using JK and D flip flops are given in Figure 4.9. The computer printout for these data cards is shown in Figure 4.10.


C JK, D

6 5 2 7 0 4 4

CARD #1
CARD #2
ECC 5081

Figure 4.9. Data Cards for Non-Repetitive 6 Count 3 Stage Counter for Example 4.3.1

```
                FLIP-FLOP TYPES SPECIFIED FOR COUNTER

                FLIP-FLOPS :   JK, D


                INPUT COUNT DATA, FIRST NO. IS NUMBER OF COUNTS
                 6    5    2    7    0    4    4


                   INPUT COUNT SEQUENCE LOGIC

                     I      NO(I)    A B C
                     1        5      1 0 1
                     2        2      0 1 0
                     3        7      1 1 1
                     4        0      0 0 0
                     5        4      1 0 0
                     6        4      1 0 0
                   -------------- DON'T CARE TERMS BELOW
                     7        1      0 0 1
                     8        3      0 1 1
                     9        6      1 1 0
```

Figure 4.10. Printout for Counter Design Example 4.3.1 (Sheet 1 of 5)

```
J K  FLIP-FLOP TRANSITION INPUT REQUIREMENTS

          0 ----> 0     0 ----> 1     1 ----> 0     1 ----> 1
           J K           J K           J K           J K
           0 -           1 -           - 1           - 0

      '1' = TRUE,  '0' = FALSE,  '-' = DON'T CARE

        TRUTH TABLE FOR COUNTER USING J K  FLIP-FLOPS

   NO      A B C    JA KA JB KB JC KC

   5       1 0 1    -  1  1  -  -  1

   2       0 1 0    1  -  -  0  1  -

   7       1 1 1    -  1  -  1  -  1

   0       0 0 0    1  -  0  -  0  -

   4       1 0 0    -  0  0  -  0  -

   4       1 0 0    0  0  0  0  0  0

   1       0 0 1    -  -  -  -  -  -

   3       0 1 1    -  -  -  -  -  -

   6       1 1 0    -  -  -  -  -  -
```

Figure 4.10. Printout for Counter Design Example 4.3.1 (Sheet 2 of 5)

```
                          J K  FLIP-FLOP INPUT EQUATIONS
                               (F = TRUE TERMS / DON'T CARE TERMS)

                               CLOCK ----.
                                         V
                                 ..............
                                 :            :
                            ---->:  J      Q  :---->
                                 :            :
                                 :            :
                                 :            :
                            ---->:  K      Q' :---->
                                 :            :
                                 :............:

      FLIP-FLOP A

          JA = 1

          KA = C / B + A'

      FLIP-FLOP B

          JB = C / B

          KB = C / B' + A

      FLIP-FLOP C

          JC = B / C

          KC = 1
```

Figure 4.10. Printout for Counter Design Example 4.3.1 (Sheet 3 of 5)

```
                         D        FLIP-FLOP TRANSITION INPUT REQUIREMENTS

                        0 ----> 0     0 ----> 1     1 ----> 0     1 ----> 1
                          D             D             D             D
                          0             1             0             1

                        '1' = TRUE,  '0' = FALSE,  '-' = DON'T CARE

                         TRUTH TABLE FOR COUNTER USING D      FLIP-FLOPS

           NO      A B C    DA  DB  DC

           5       1 0 1    0   1   0

           2       0 1 0    1   1   1

           7       1 1 1    0   0   0

           0       0 0 0    1   0   0

           4       1 0 0    1   0   0

           4       1 0 0    0   0   0

           1       0 0 1    -   -   -

           3       0 1 1    -   -   -

           6       1 1 0    -   -   -
```

Figure 4.10. Printout for Counter Design Example 4.3.1 (Sheet 4 of 5)

```
                              D  FLIP-FLOP INPUT EQUATIONS
                                 (F = TRUE TERMS / DON'T CARE TERMS)

                                 CLOCK ---.
                                          V
                                   .............
                                   :           :
                                   :         Q :---->
                                   :           :
                              ---->: D         :
                                   :           :
                                   :        Q' :---->
                                   :           :
                                   :...........:

      FLIP-FLOP A

         DA = C' / A'

      FLIP-FLOP B

         DB = B'C + BC' / A'C

      FLIP-FLOP C

         DC = BC' / A'C
```

Figure 4.10. Printout for Counter Design Example 4.3.1 (Sheet 5 of 5)

4.4 COUNTER, 10 COUNT 6 STAGE

This example is for a 6 bit counter having a 10 count sequence of 8, 17, 26, 13, 62, 5, 42, 9, 59, 37. Normally a 10 state counter is implemented with only 4 flip flops. However, in this case 6 bits or flip flops are required to represent the highest designated count (count 59) in binary form.

This design is an example of when subroutine ARRAY operates in its alternate mode as described in Section 5.1.8. The alternate mode is used whenever the count sequence contains 1/6 or less of the maximum count states possible for the number of flip flops used. The alternate mode uses less computer run time for this type of sequence by ignoring the don't care count states. For 6 flip flops, 2^6 or 64 states are possible. Since the sequence uses 10 states, there are 54 don't care states that would normally have to be included in development of the input equations.

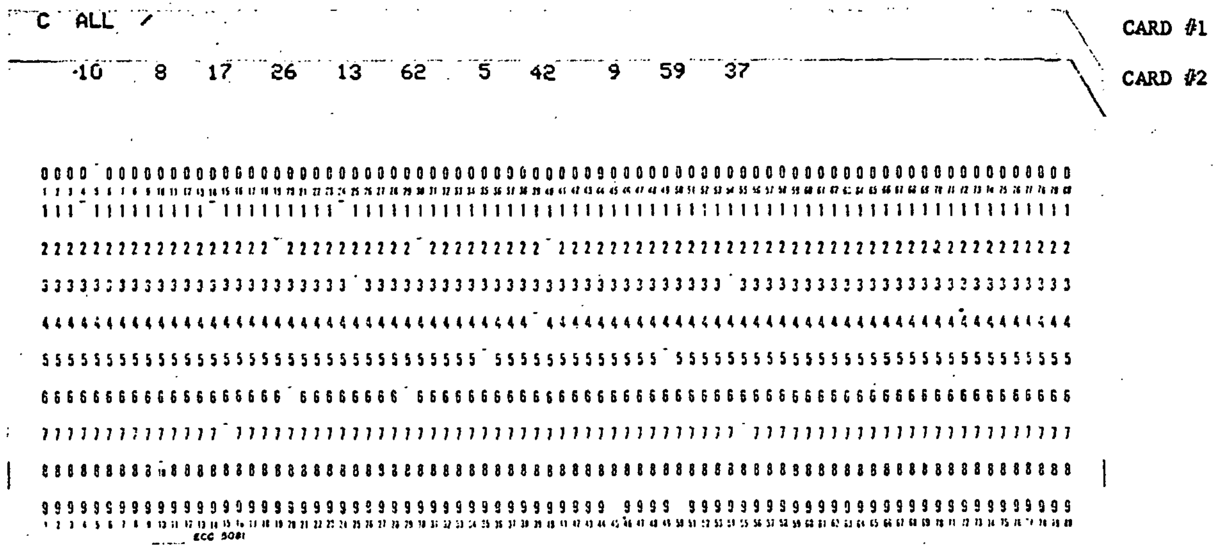
The program is run twice for the given 10 count sequence. First to obtain the flip flop input equations in sum-of-product form (Section 4.4.1) and then in product-of-sum form (Section 4.4.2). The input data cards for these two program runs are shown in Figure 4.11.

Section 4.4.1 illustrates how NAND gates can be used to realize the sum-of-product equations where the gate inputs are obtained directly from the equations. Section 4.4.2 illustrates how NOR gates are used in similar fashion to realize product-of-sum equations.

4.4.1 <u>ALL /</u>

The flip flop code letters ALL specify the JK, RS, D, and T flip flops and output code option / specifies the short form output mode. The input flip flop equations are in sum-of-product form which is the normal printout form. This design is implemented with the data cards shown in Figure 4.11A. The computer printout for these data cards is given in Figure 4.12.

Sum-of-product equations are in proper form for easy implementation with two levels of NAND gates as illustrated in Figure 4.13 for the Delay flip flop. There is a first level gate for each product term of


C ALL /

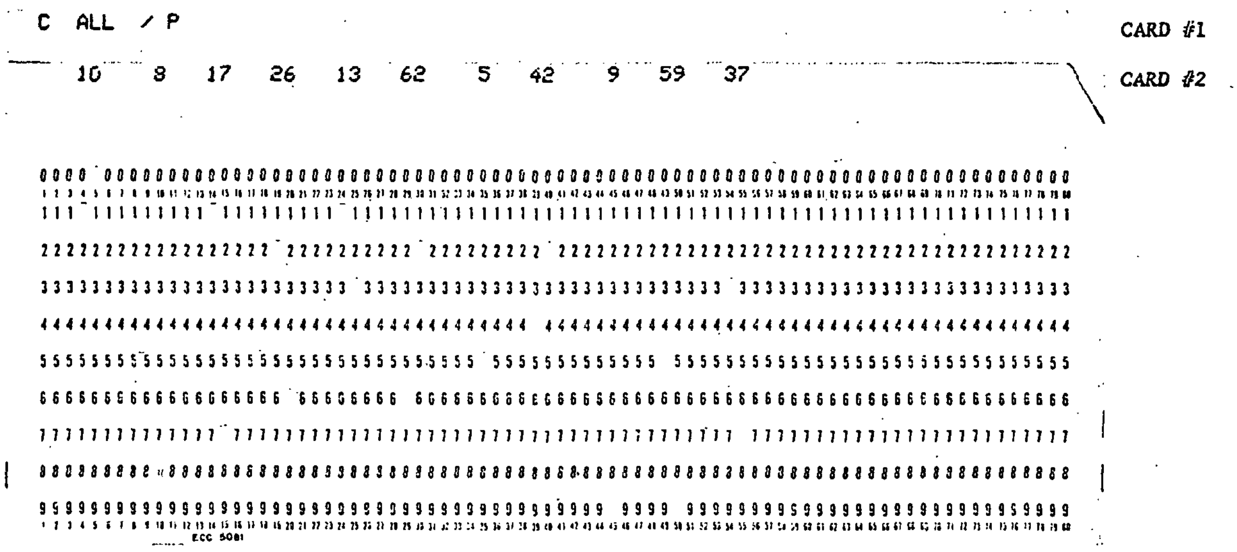
CARD #1

10 8 17 26 13 62 5 42 9 59 37

CARD #2

ECC 5081

A. Data Cards for Example 4.4.1


C ALL / P

CARD #1

10 8 17 26 13 62 5 42 9 59 37

CARD #2

ECC 5081

B. Data Cards for Example 4.4.2

Figure 4.11. Data Cards for 10 Count 6 Stage Counter for Example 4.4.1 and Example 4.4.2

```
FLIP-FLOP TYPES SPECIFIED FOR COUNTER

FLIP-FLOPS :   ALL  /


INPUT COUNT DATA,  FIRST NO. IS NUMBER OF COUNTS

 10    8    17    26    13    62    5    42    9    59    37



        INPUT COUNT SEQUENCE LOGIC

          I      NO(I)      A B C D E F

          1        8        0 0 1 0 0 0

          2       17        0 1 0 0 0 1

          3       26        0 1 1 0 1 0

          4       13        0 0 1 1 0 1

          5       62        1 1 1 1 1 0

          6        5        0 0 0 1 0 1

          7       42        1 0 1 0 1 0

          8        9        0 0 1 0 0 1

          9       59        1 1 1 0 1 1

         10       37        1 0 0 1 0 1

         11        8        0 0 1 0 0 0

        -------------- DON'T CARE TERMS BELOW

         12        0        0 0 0 0 0 0

         13        1        0 0 0 0 0 1

         14        2        0 0 0 0 1 0

         15        3        0 0 0 0 1 1

         16        4        0 0 0 1 0 0

         17        6        0 0 0 1 1 0

         18        7        0 0 0 1 1 1
```

Figure 4.12. Printout for Counter Design Example 4.4.1 (Sheet 1 of 7)

```
19          10          0 0 1 0 1 0
20          11          0 0 1 0 1 1
21          12          0 0 1 1 0 0
22          14          0 0 1 1 1 0
23          15          0 0 1 1 1 1
24          16          0 1 0 0 0 0
25          18          0 1 0 0 1 0
26          19          0 1 0 0 1 1
27          20          0 1 0 1 0 0
28          21          0 1 0 1 0 1
29          22          0 1 0 1 1 0
30          23          0 1 0 1 1 1
31          24          0 1 1 0 0 0
32          25          0 1 1 0 0 1
33          27          0 1 1 0 1 1
34          28          0 1 1 1 0 0
35          29          0 1 1 1 0 1
36          30          0 1 1 1 1 0
37          31          0 1 1 1 1 1
38          32          1 0 0 0 0 0
39          33          1 0 0 0 0 1
40          34          1 0 0 0 1 0
41          35          1 0 0 0 1 1
42          36          1 0 0 1 0 0
43          38          1 0 0 1 1 0
44          39          1 0 0 1 1 1
45          40          1 0 1 0 0 0
46          41          1 0 1 0 0 1
```

Figure 4.12. Printout for Counter Design Example 4.4.1 (Sheet 2 of 7)

```
47      43      1 0 1 0 1 1
48      44      1 0 1 1 0 0
49      45      1 0 1 1 0 1
50      46      1 0 1 1 1 0
51      47      1 0 1 1 1 1
52      48      1 1 0 0 0 0
53      49      1 1 0 0 0 1
54      50      1 1 0 0 1 0
55      51      1 1 0 0 1 1
56      52      1 1 0 1 0 0
57      53      1 1 0 1 0 1
58      54      1 1 0 1 1 0
59      55      1 1 0 1 1 1
60      56      1 1 1 0 0 0
61      57      1 1 1 0 0 1
62      58      1 1 1 0 1 0
63      60      1 1 1 1 0 0
64      61      1 1 1 1 0 1
65      63      1 1 1 1 1 1
```

Figure 4.12. Printout for Counter Design Example 4.4.1 (Sheet 3 of 7)

```
                              J K   FLIP-FLOP INPUT EQUATIONS
                                    (F = TRUE TERMS / DON'T CARE TERMS)

                                    CLOCK ---.
                                             V
                                    ..............
                                    :            :
                               ---->: J        Q :---->
                                    :            :
                                    :            :
                                    :            :
                               ---->: K        Q':---->
                                    :            :
                                    :............:

      FLIP-FLOP A

          JA = B'F

          KA = B' + D

      FLIP-FLOP B

          JB = A'C

          KB = C

      FLIP-FLOP C

          JC = 1

          KC = AB + E'F'

      FLIP-FLOP D

          JD = BC

          KD = C'

      FLIP-FLOP E

          JE = A'F

          KE = 1

          KF = C' + D
```

Figure 4.12. Printout for Counter Design Example 4.4.1 (Sheet 4 of 7)

```
                         R S   FLIP-FLOP INPUT EQUATIONS
                               (F = TRUE TERMS / DON'T CARE TERMS)

                               CLOCK ---.
                                        V
                                 .............
                                 :           :
                            ---->: S       Q :---->
                                 :           :
                                 :           :
                            ---->: R       Q':---->
                                 :           :
                                 :...........:

   FLIP-FLOP A

      SA = A'B'F

      RA = F' + AB'

   FLIP-FLOP B

      SB = CE'

      RB = E

   FLIP-FLOP C

      SC = C'

      RC = AB + E'F'

   FLIP-FLOP D

      SD = BC

      RD = C'

   FLIP-FLOP E

      SE = A'F

      RE = E

      RF = C' + B'D
```

Figure 4.12. Printout for Counter Design Example 4.4.1 (Sheet 5 of 7)

```
                         D      FLIP-FLOP INPUT EQUATIONS
                                (F = TRUE TERMS / DON'T CARE TERMS)

                                CLOCK ---.
                                         V
                                 ::::::::::::::
                                 :            :
                                 :        Q   :---->
                                 :            :
                            ---->: D          :
                                 :            :
                                 :        Q'  :---->
                                 :            :
                                 ::::::::::::::

       FLIP-FLOP A

         DA = CF + A'B'F

       FLIP-FLOP B

         DB = CE' + D'E'

       FLIP-FLOP C

         DC = A'B + AB' + E'F

       FLIP-FLOP D

         DD = BC + CD

       FLIP-FLOP E

         DE = A'F

       FLIP-FLOP F

         DF = E + CD'
```

Figure 4.12. Printout for Counter Design Example 4.4.1 (Sheet 6 of 7)

```
                  T      FLIP-FLOP INPUT EQUATIONS
                         (F = TRUE TERMS / DON'T CARE TERMS)

                         CLOCK ---.
                                  V
                            ..............
                            :            :
                            :         Q  :---->
                            :            :
                       ---->: T          :
                            :            :
                            :         Q' :---->
                            :            :
                            :............:

      FLIP-FLOP A

         TA = D + AB' + B'F

      FLIP-FLOP B

         TB = BC + A'C

      FLIP-FLOP C

         TC = C' + AB + E'F'

      FLIP-FLOP D

         TD = B'C' + BCD'

      FLIP-FLOP E

         TE = B + E + A'F

      FLIP-FLOP F

         TF = C' + D + F'
```

Figure 4.12. Printout for Counter Design Example 4.4.1 (Sheet 7 of 7)

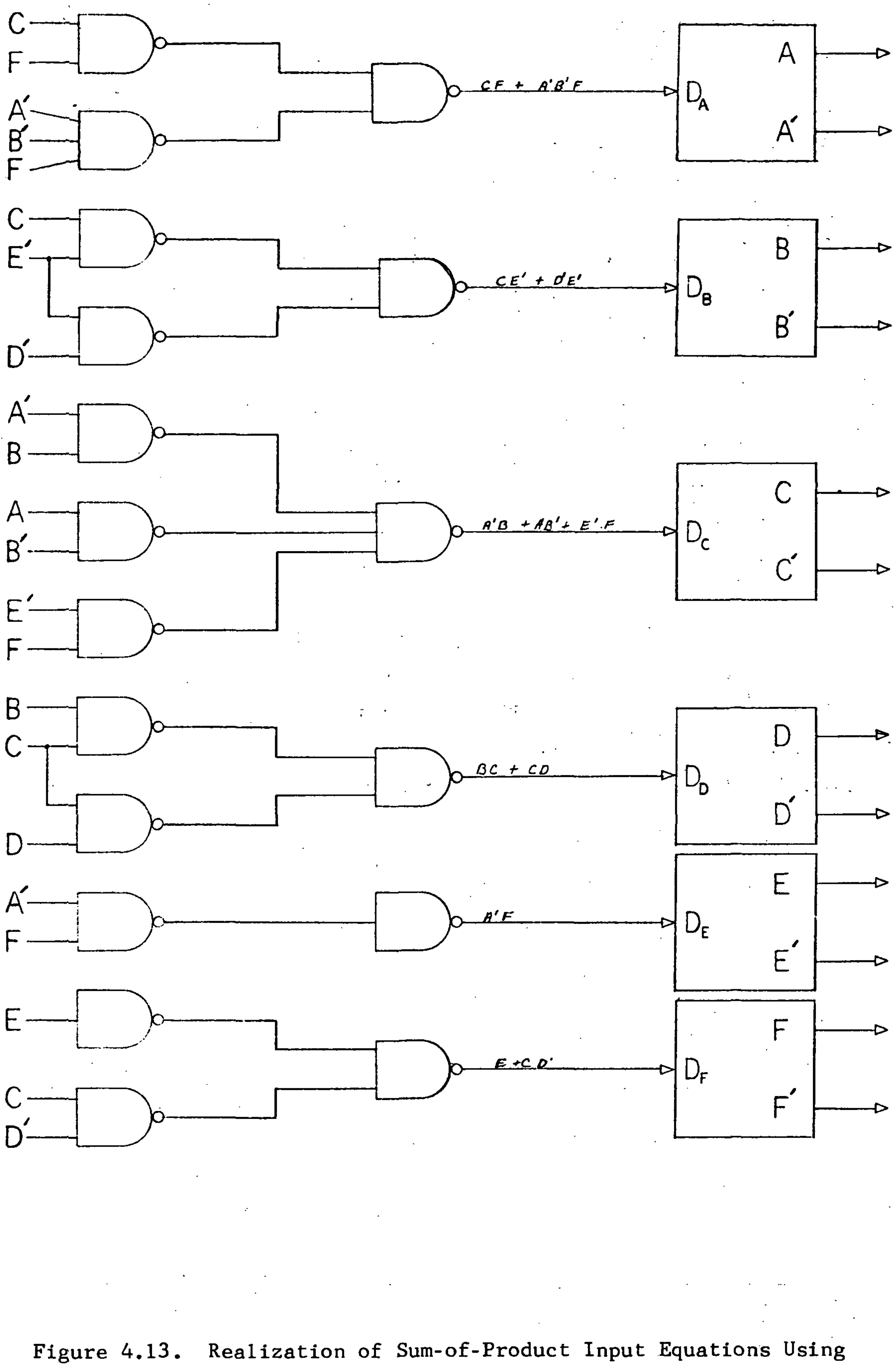


Figure 4.13. Realization of Sum-of-Product Input Equations Using NAND Gates for Delay Flip Flops of Example 4.1.1

each flip flop equation where the gate inputs are the product term variables. There is one second level gate for each flip flop equation which essentially sums the product terms. For example, the equation DA = CF + A'B'F is implemented using two first level NAND gates as shown in Figure 4.13. The inputs to one gate are C and F and the inputs to the other gate are A', B' and F. The gate input variables are of course the flip flop outputs.

4.4.2 ALL / P

The flip flop code letters ALL specify the JK, RS, D, and T flip flops. Output code options / and P specify the short form output and product-of-sum input equations respectively. The data cards for this design are shown in Figure 4.11B. The computer printout for these data cards is shown in Figure 4.14.

The product-of-sum equations are in proper form for easy implementation with two levels of NOR gates as illustrated in Figure 4.15 for the Delay flip-flop. There is a first level gate for each sum term of each flip flop equation where the gate inputs are the sum term variables. The concepts for this configuration are the same as for Figure 4.13 where NAND gates are used for sum-of-product equations. For example, the product-of-sum equation DA = (F)(A' + B)(A + B') is implemented with three first level NOR gates as shown in Figure 4.15. The gate inputs are F, A' and B, A and B'.

```
FLIP-FLOP TYPES SPECIFIED FOR COUNTER

FLIP-FLOPS :    ALL  / P


INPUT COUNT DATA,  FIRST NO. IS NUMBER OF COUNTS

 10     8    17    26    13    62    5    42    9    59    37



        INPUT COUNT SEQUENCE LOGIC

          I      NO(I)     A B C D E F
          1        8       0 0 1 0 0 0
          2       17       0 1 0 0 0 1
          3       26       0 1 1 0 1 0
          4       13       0 0 1 1 0 1
          5       62       1 1 1 1 1 0
          6        5       0 0 0 1 0 1
          7       42       1 0 1 0 1 0
          8        9       0 0 1 0 0 1
          9       59       1 1 1 0 1 1
         10       37       1 0 0 1 0 1
         11        8       0 0 1 0 0 0
        ---------------  DON'T CARE TERMS BELOW
         12        0       0 0 0 0 0 0
         13        1       0 0 0 0 0 1
         14        2       0 0 0 0 1 0
         15        3       0 0 0 0 1 1
         16        4       0 0 0 1 0 0
         17        6       0 0 0 1 1 0
         18        7       0 0 0 1 1 1
```

Figure 4.14. Printout for Counter Design Example 4.4.2 (Sheet 1 of 7)

```
19      10      0 0 1 0 1 0
20      11      0 0 1 0 1 1
21      12      0 0 1 1 0 0
22      14      0 0 1 1 1 0
23      15      0 0 1 1 1 1
24      16      0 1 0 0 0 0
25      18      0 1 0 0 1 0
26      19      0 1 0 0 1 1
27      20      0 1 0 1 0 0
28      21      0 1 0 1 0 1
29      22      0 1 0 1 1 0
30      23      0 1 0 1 1 1
31      24      0 1 1 0 0 0
32      25      0 1 1 0 0 1
33      27      0 1 1 0 1 1
34      28      0 1 1 1 0 0
35      29      0 1 1 1 0 1
36      30      0 1 1 1 1 0
37      31      0 1 1 1 1 1
38      32      1 0 0 0 0 0
39      33      1 0 0 0 0 1
40      34      1 0 0 0 1 0
41      35      1 0 0 0 1 1
42      36      1 0 0 1 0 0
43      38      1 0 0 1 1 0
44      39      1 0 0 1 1 1
45      40      1 0 1 0 0 0
46      41      1 0 1 0 0 1
```

Figure 4.14. Printout for Counter Design Example 4.4.2 (Sheet 2 of 7)

```
 47         43       1 0 1 0 1 1
 48         44       1 0 1 1 0 0
 49         45       1 0 1 1 0 1
 50         46       1 0 1 1 1 0
 51         47       1 0 1 1 1 1
 52         48       1 1 0 0 0 0
 53         49       1 1 0 0 0 1
 54         50       1 1 0 0 1 0
 55         51       1 1 0 0 1 1
 56         52       1 1 0 1 0 0
 57         53       1 1 0 1 0 1
 58         54       1 1 0 1 1 0
 59         55       1 1 0 1 1 1
 60         56       1 1 1 0 0 0
 61         57       1 1 1 0 0 1
 62         58       1 1 1 0 1 0
 63         60       1 1 1 1 0 0
 64         61       1 1 1 1 0 1
 65         63       1 1 1 1 1 1
```

Figure 4.14. Printout for Counter Design Example 4.4.2 (Sheet 3 of 7)

```
                        J K   FLIP-FLOP INPUT EQUATIONS
                              (F = TRUE TERMS / DON'T CARE TERMS)

                              CLOCK ---.
                                       V
                                 ..............
                                 :            :
                            ---->: J        Q :---->
                                 :            :
                                 :            :
                                 :            :
                            ---->: K        Q':---->
                                 :            :
                                 :............:

        FLIP-FLOP A

           JA = (B')(F)

           KA = (B'+D)

        FLIP-FLOP B

           JB = (A')(C)

           KB = (C)

        FLIP-FLOP C

           JC = 1

           KC = (A+F')(D+E'+F)

        FLIP-FLOP D

           JD = (B)(E)

           KD = (C')

        FLIP-FLOP E

           JE = (A')(F)

           KE = 1

           KF = (C'+D)
```

Figure 4.14. Printout for Counter Design Example 4.4.2 (Sheet 4 of 7)

```
                         R S   FLIP-FLOP INPUT EQUATIONS
                               (F = TRUE TERMS / DON'T CARE TERMS)

                               CLOCK ---.
                                        V
                                 ..............
                                 :            :
                            ---->: S        Q :---->
                                 :            :
                                 :            :
                                 :            :
                            ---->: R        Q':---->
                                 :            :
                                 :............:

     FLIP-FLOP A

          SA = (A')(B')(F)

          RA = (A)(C'+F')

     FLIP-FLOP B

          SB = (C)(E')

          RB = (E)

     FLIP-FLOP C

          SC = (C')

          RC = (A+B')(A'+B)(E+F')

     FLIP-FLOP D

          SD = (B)(E)

          RD = (C')

     FLIP-FLOP E

          SE = (A')(F)

          RE = (E)

          RF = (E')(C'+D)
```

Figure 4.14. Printout for Counter Design Example 4.4.2 (Sheet 5 of 7)

```
                D      FLIP-FLOP INPUT EQUATIONS
                       (F = TRUE TERMS / DON'T CARE TERMS)

                       CLOCK ---.
                                V
                         ..............
                         :            :
                         :        Q   :---->
                         :            :
                    ---->: D          :
                         :            :
                         :        Q'  :---->
                         :            :
                         :............:

    FLIP-FLOP A

       DA = (F)(A'+B)(A+B')

    FLIP-FLOP B

       DB = (A')(E')(B+C)

    FLIP-FLOP C

       DC = (A'+B')(E+F)

    FLIP-FLOP D

       DD = (C)(B+D)

    FLIP-FLOP E

       DE = (A')(F)

    FLIP-FLOP F

       DF = (C)(B+D')
```

Figure 4.14. Printout for Counter Design Example 4.4.2 (Sheet 6 of 7)

```
                              T   FLIP-FLOP INPUT EQUATIONS
                                  (F = TRUE TERMS / DON'T CARE TERMS)

                                  CLOCK ---.
                                           V
                                  ..............
                                  :            :
                                  :         Q  :---->
                                  :            :
                             ---->: T          :
                                  :            :
                                  :         Q' :---->
                                  :            :
                                  :............:

     FLIP-FLOP A

        TA = (B'+D)(A+F)

     FLIP-FLOP B

        TB = (C)(A'+B)

     FLIP-FLOP C

        TC = (D+E'+F)(A+C'+F')

     FLIP-FLOP D

        TD = (A'+F)(B+C')(D+E)

     FLIP-FLOP E

        TE = (A'+C)(E+F)

     FLIP-FLOP F

        TF = (C'+D+F')
```

Figure 4.14. Printout for Counter Design Example 4.4.2 (Sheet 7 of 7)

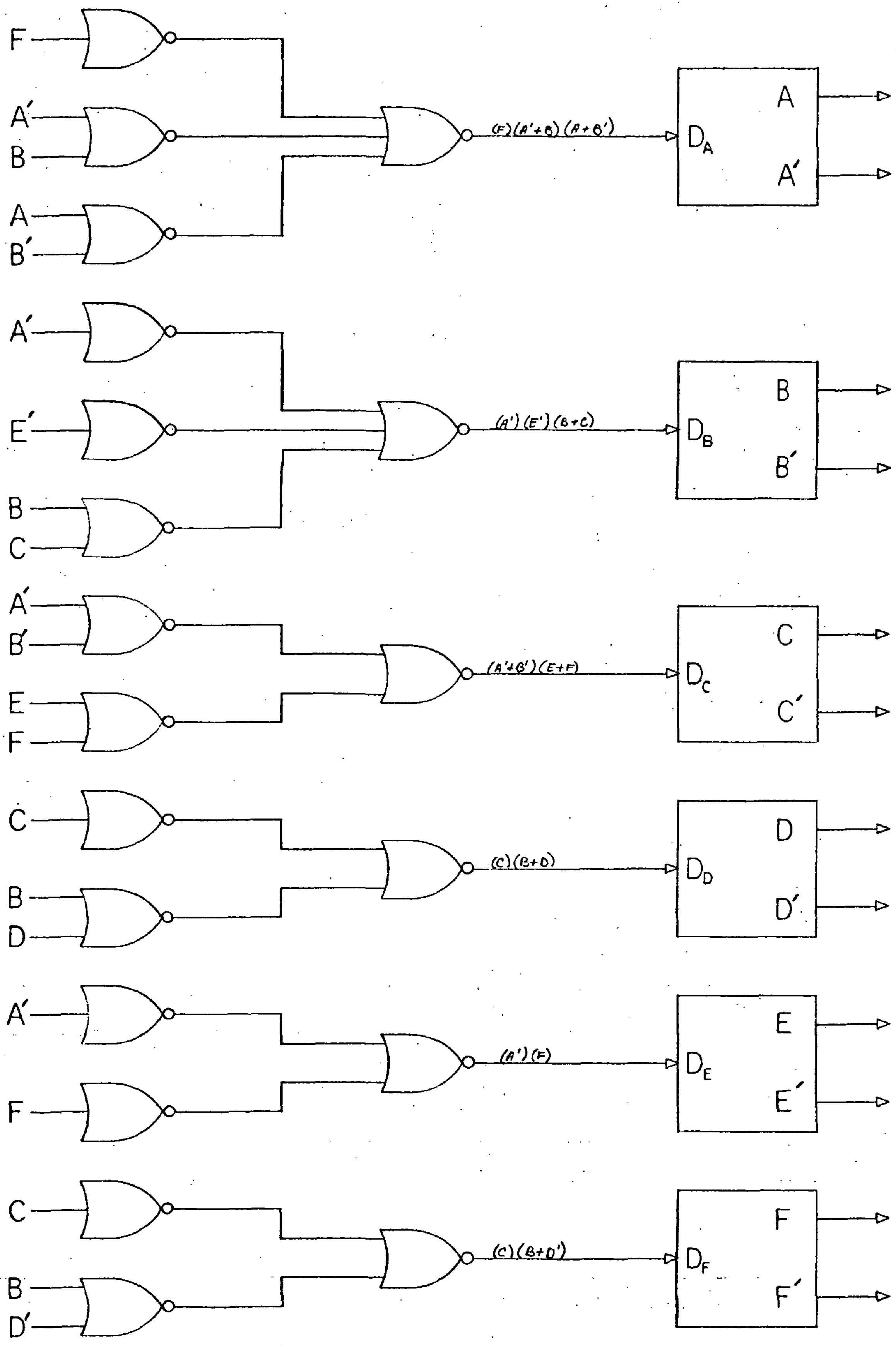


Figure 4.15. Realization of Product-of-Sum Input Equations Using NOR Gates for Delay Flip Flop of Example 4.1.2

4.5 BOOLEAN SIMPLIFICATION, SUM-OF-PRODUCT INPUT

This section provides three examples of Boolean simplification where the function to be simplified is in sum-of-product form. The Boolean function is specified by entering letter F in column one of the first data card and is terminated by an asterisk (*). Refer to Section 3.2.2 for details on how the function is entered on data cards. Each example illustrates different features of the program. The first two examples contain only true terms while the third example contains both true and don't care terms.

4.5.1 Twelve Term 6 Variable Function

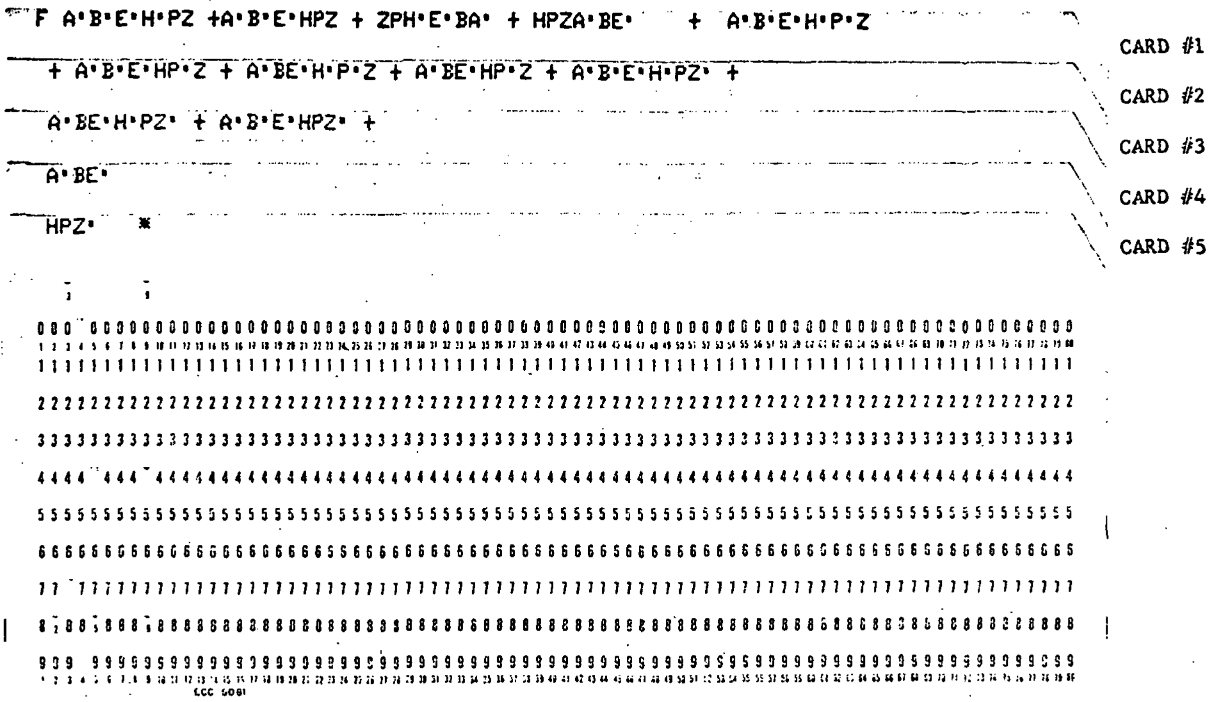
The Boolean function to be simplified contains 12 true terms, each consisting of 6 variables. The data cards for this function along with the computer printout for these data cards are shown in Figure 4.16. The function is entered on 5 data cards. In this example, each term contains all 6 variables although this is not required. The variables are represented by letters A, B, E, H, P and Z to demonstrate that any letter A through Z can be used. Notice that the variables of the third and fourth terms are not entered in alphabetical order and that the last term is divided between data cards 4 and 5.

4.5.2 Eight Term 3 Variable Function

This Boolean function consists of 8 true terms of 3 variables each. The data card and computer printout for this function are shown in Figure 4.17. Since the 8 terms contain every possible combination of the 3 variables, the simplified function is equal to unity.

4.5.3 Sixteen Term 6 Variable Function

Here the Boolean function to be simplified contains 10 true terms plus 6 don't care terms, each term consisting of 6 variables. The five data cards used for this function and the computer printout for these data cards are shown in Figure 4.18. The true terms are always entered first and a slash mark (/) is used to separate the true terms from the don't care terms. For this example, the simplified function also contains don't care terms. This is not always the case as the simplified true terms may contain all the don't care terms.



```
F A•B•E•H•PZ +A•B•E•HPZ + ZPH•E•BA• + HPZA•BE•   +  A•B•E•H•P•Z
  + A•B•E•HP•Z + A•BE•H•P•Z + A•BE•HP•Z + A•B•E•H•PZ• +
  A•BE•H•PZ• + A•B•E•HPZ• +
  A•BE•
  HPZ•   *
```

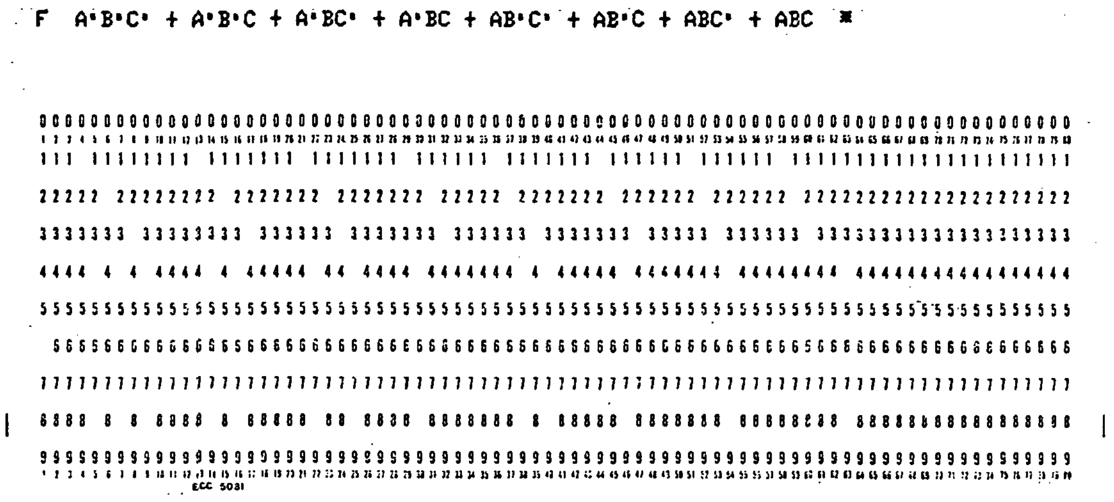
CARD #1
CARD #2
CARD #3
CARD #4
CARD #5

A. Data Cards

```
INPUT FUNCTION    (F = TRUE TERMS / DON'T CARE TERMS)
                  • TERMINATES BOOLEAN EXPRESSION

    F =   A'B'E'H'PZ  +A'B'E'HPZ + ZPH'E'BA' + HPZA'BE'    +  A'B'E'H'P'Z
              + A'B'E'HP'Z + A'BE'H'P'Z + A'BE'HP'Z + A'B'E'H'PZ' +
              A'BE'H'PZ' + A'B'E'HPZ' +
              A'BE'
              HPZ'   •

LETTERS USED = ABEHPZ


SIMPLIFIED FUNCTION   (F = TRUE TERMS / DON'T CARE TERMS)

    F = A'E'Z + A'E'P
```

B. Computer Printout

Figure 4.16. Data Cards and Printout for Simplification of Sum-of-Product Boolean Function, Example 4.5.1


F A'B'C' + A'B'C + A'BC' + A'BC + AB'C' + AB'C + ABC' + ABC *

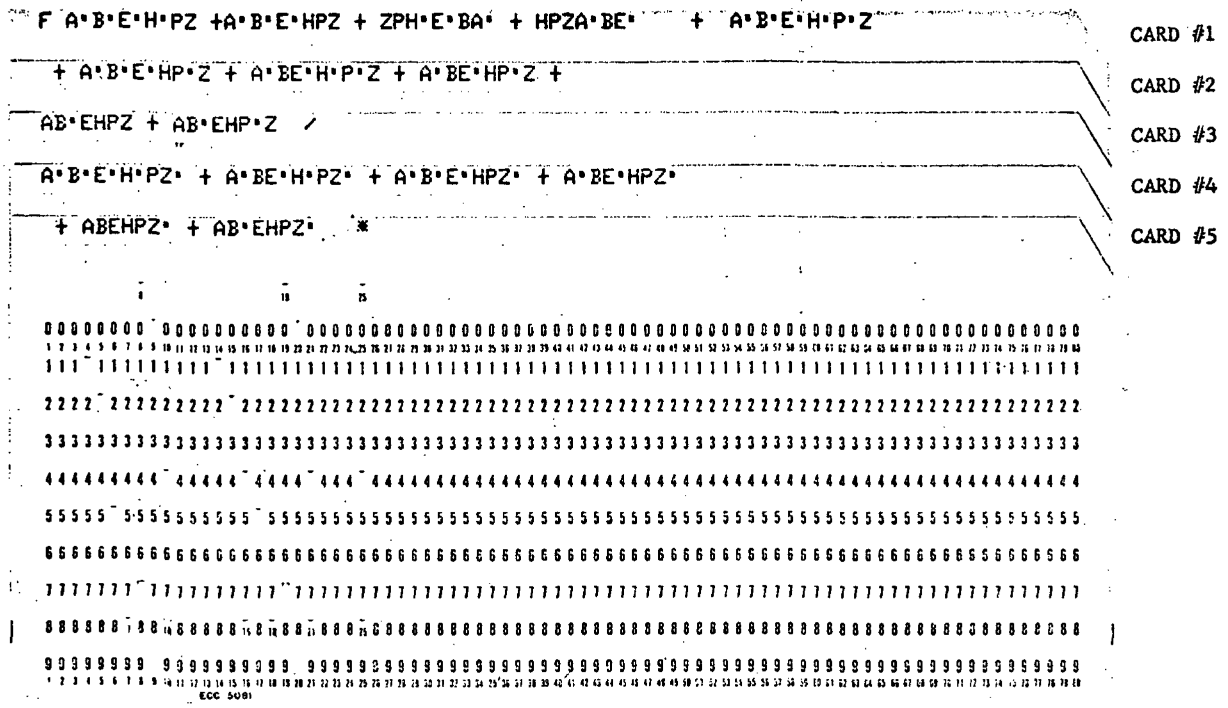
CARD #1

A. Data Card

```
----INPUT FUNCTION-----(F = TRUE TERMS / DON'T CARE TERMS)--------------------
                        * TERMINATES BOOLEAN EXPRESSION

          F =    A'B'C' + A'B'C + A'BC' + A'BC + AB'C' + AB'C + ABC' + ABC *

   LETTERS USED = ABC


----SIMPLIFIED FUNCTION-----(F = TRUE TERMS / DON'T CARE TERMS)---------------

          F = 1
```

B. Computer Printout

Figure 4.17. Data Card and Printout for Simplification of Sum-of-Product Boolean Function, Example 4.5.2



```
F A'B'E'H'PZ +A'B'E'HPZ + ZPH'E'BA' + HPZA'BE'   +  A'B'E'H'P'Z        CARD #1
 + A'B'E'HP'Z + A'BE'H'P'Z + A'BE'HP'Z +                              CARD #2
AB'EHPZ + AB'EHP'Z  /                                                 CARD #3
A'B'E'H'PZ' + A'BE'H'PZ' + A'B'E'HPZ' + A'BE'HPZ'                     CARD #4
 + ABEHPZ' + AB'EHPZ'  *                                              CARD #5
```

A. Data Cards

```
INPUT FUNCTION    (F = TRUE TERMS / DON'T CARE TERMS)
                  * TERMINATES BOOLEAN EXPRESSION

   F =   A'B'E'H'PZ +A'B'E'HPZ + ZPH'E'BA' + HPZA'BE'     + A'B'E'H'P'Z
         + A'B'E'HP'Z + A'BE'H'P'Z + A'BE'HP'Z +
      AB'EHPZ + AB'EHP'Z  /
      A'B'E'H'PZ' + A'BE'H'PZ' + A'B'E'HPZ' + A'BE'HPZ'
         + ABEHPZ' + AB'EHPZ'  *

LETTERS USED = ABEHPZ


SIMPLIFIED FUNCTION    (F = TRUE TERMS / DON'T CARE TERMS)

   F = A'E'Z + AB'EHZ / A'E'P + AEHPZ'
```

B. Computer Printout

Figure 4.18. Data Cards and Printout for Simplification of Sum-of-Product Boolean Function, Example 4.5.3

4.6 BOOLEAN SIMPLIFICATION, PRODUCT-OF-SUM INPUT

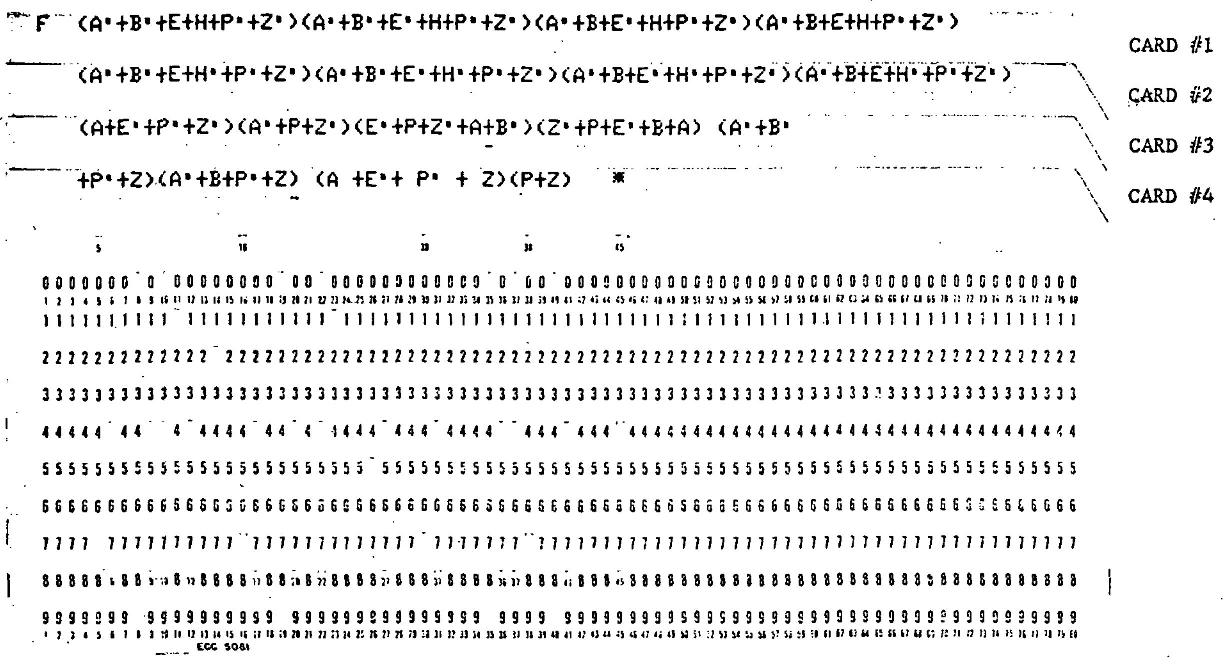
This section provides two examples of Boolean simplification where the function to be simplified is in product-of-sum form. The simplified function is always printed out in sum-of-product form regardless of the input form. Only true terms are permitted for product-of-sum inputs. To simplify a function containing true and don't care terms, the true and don't care terms are split into two functions and simplified separately. Then the simplified sum-of-product results of both functions are summed together and run as a sum-of-product function containing true and don't care terms.

4.6.1 Sixteen Term 6 Variable Function

This Boolean function contains 16 product-of-sum terms, each term consisting of a maximum of 6 variables. Each term can contain one or all of the 6 variables. The data cards and the computer printout for this function are shown in Figure 4.19. Notice that the third and fourth data cards contain terms consisting of from two to five variables. This function is equivalent to the sum-of-product function given in Figure 4.16. Notice that both functions simplify to the same function.

4.6.2 Eight Term 3 Variable Function

This function contains 8 product-of-sum terms of 3 variables each. The data card and computer printout for this function is shown in Figure 4.20. The simplified function is equal to zero as the 8 terms contain every possible combination of the 3 variables. If the terms were multiplied out, each resulting term would contain at least one complement pair and would therefore equal zero. Notice that this function is the complement of the sum-of-product function given in Figure 4.17.



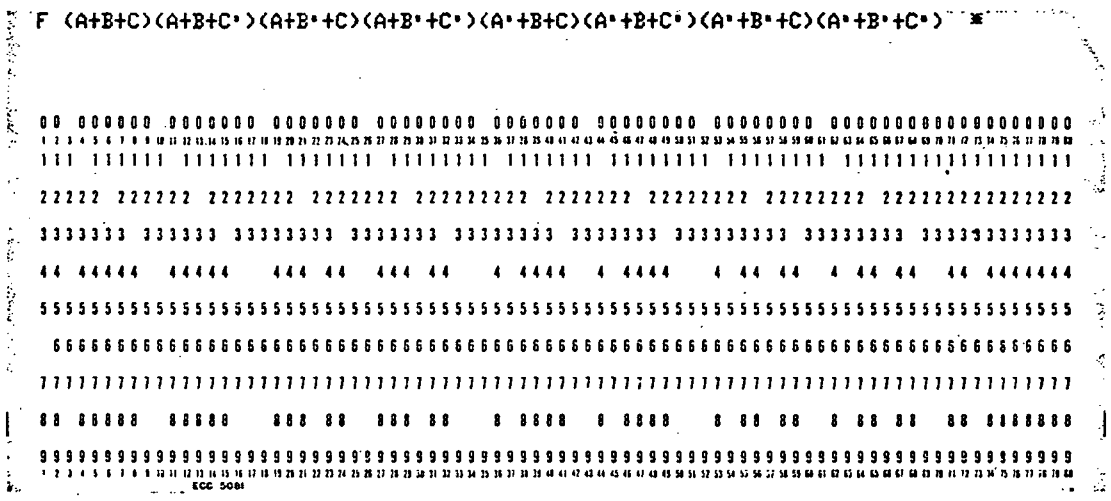
```
F  (A'+B'+E+H+P'+Z')(A'+B'+E'+H+P'+Z')(A'+B+E'+H+P'+Z')(A'+B+E+H+P'+Z')          CARD #1
   (A'+B'+E+H'+P'+Z')(A'+B'+E'+H'+P'+Z')(A'+B+E'+H'+P'+Z')(A'+B+E+H'+P'+Z')      CARD #2
   (A+E'+P'+Z')(A'+P+Z')(E'+P+Z'+A+B')(Z'+P+E'+B+A) (A'+B'                       CARD #3
   +P'+Z)(A'+B+P'+Z) (A +E'+ P' + Z)(P+Z)   *                                    CARD #4
```

A. Data Cards

```
INPUT FUNCTION    (F = TRUE TERMS / DON'T CARE TERMS)
                  * TERMINATES BOOLEAN EXPRESSION

     F =   (A'+B'+E+H+P'+Z')(A'+B'+E'+H+P'+Z')(A'+B+E'+H+P'+Z')(A'+B+E+H+P'+Z')
           (A'+B'+E+H'+P'+Z')(A'+B'+E'+H'+P'+Z')(A'+B+E'+H'+P'+Z')(A'+B+E+H'+P'+Z')
           (A+E'+P'+Z')(A'+P+Z')(E'+P+Z'+A+B')(Z'+P+E'+B+A) (A'+B'
           +P'+Z)(A'+B+P'+Z) (A +E'+ P' + Z)(P+Z)   *

LETTERS USED = ABEHPZ

SIMPLIFIED FUNCTION    (F = TRUE TERMS / DON'T CARE TERMS)

     F = A'E'Z + A'E'P
```

B. Computer Printout

Figure 4.19. Data Cards and Printout for Simplification of Product-of-Sum Boolean Function, Example 4.6.1



```
F (A+B+C)(A+B+C')(A+B'+C)(A+B'+C')(A'+B+C)(A'+B+C')(A'+B'+C)(A'+B'+C') *
```

CARD #1

A. Data Cards

```
INPUT FUNCTION    (F = TRUE TERMS / DON'T CARE TERMS)
                   * TERMINATES BOOLEAN EXPRESSION

      F =    (A+B+C)(A+B+C')(A+B'+C)(A+B'+C')(A'+B+C)(A'+B+C')(A'+B'+C)(A'+B'+C')  *

   LETTERS USED = ABC


SIMPLIFIED FUNCTION    (F = TRUE TERMS / DON'T CARE TERMS)

      F = 0
```

B. Computer Printout

Figure 4.20. Data Card and Printout for Simplification of Product-of-Sum Boolean Function, Example 4.6.2

SECTION 5

COMPUTER PROGRAM DOCUMENTATION

5.1 DESCRIPTION OF PROGRAMS

A brief description is given for the main-line program and each of the 10 subroutines used. The purpose is to explain the function of each sub-program along with the main concepts involved. The sub-programs are listed below giving the section in which they are described.

Section	Main-line Program
5.1.1	DIGITL - Digital Logic Design
Section	**Subroutine Programs**
5.1.2	BLNIN - Boolean Input
5.1.3	CTRIN - Counter Input
5.1.4	JKRS - JK + RS FF's
5.1.5	DLTRG - Delay + Trigger FF's
5.1.6	RST - RST FF
5.1.7	CTDC - Counter Don't Care Terms
5.1.8	ARRAY - Boolean Expression Array
5.1.9	DETRU - Determine True Terms
5.1.10	BOOL - Boolean Simplification
5.1.11	DGLOT - Digital Output

The program flow charts and source listings are given in Section 5.2.

5.1.1 Program DIGITL - Digital Logic Design

Program DIGITL is the main-line program which acts as master control for all program functions by calling the appropriate subroutines in proper sequence. A block diagram of the program is shown in Figure 5.1. As the diagram shows, the program is divided into two modes, the Boolean function simplification mode and the counter design mode. The code letter in column one of the first data card, in each data set, determines the mode. A blank space for the code letter terminates the program. A description of the function and operation of each of the subroutine programs is given in the following sections.

5.1.2 Subroutine BLNIN - Boolean Input

Subroutine BLNIN takes the Boolean function that is read in for simplification and converts the expression into a two dimensional numerical array called the "A" array. The read in function consists of alphabetical

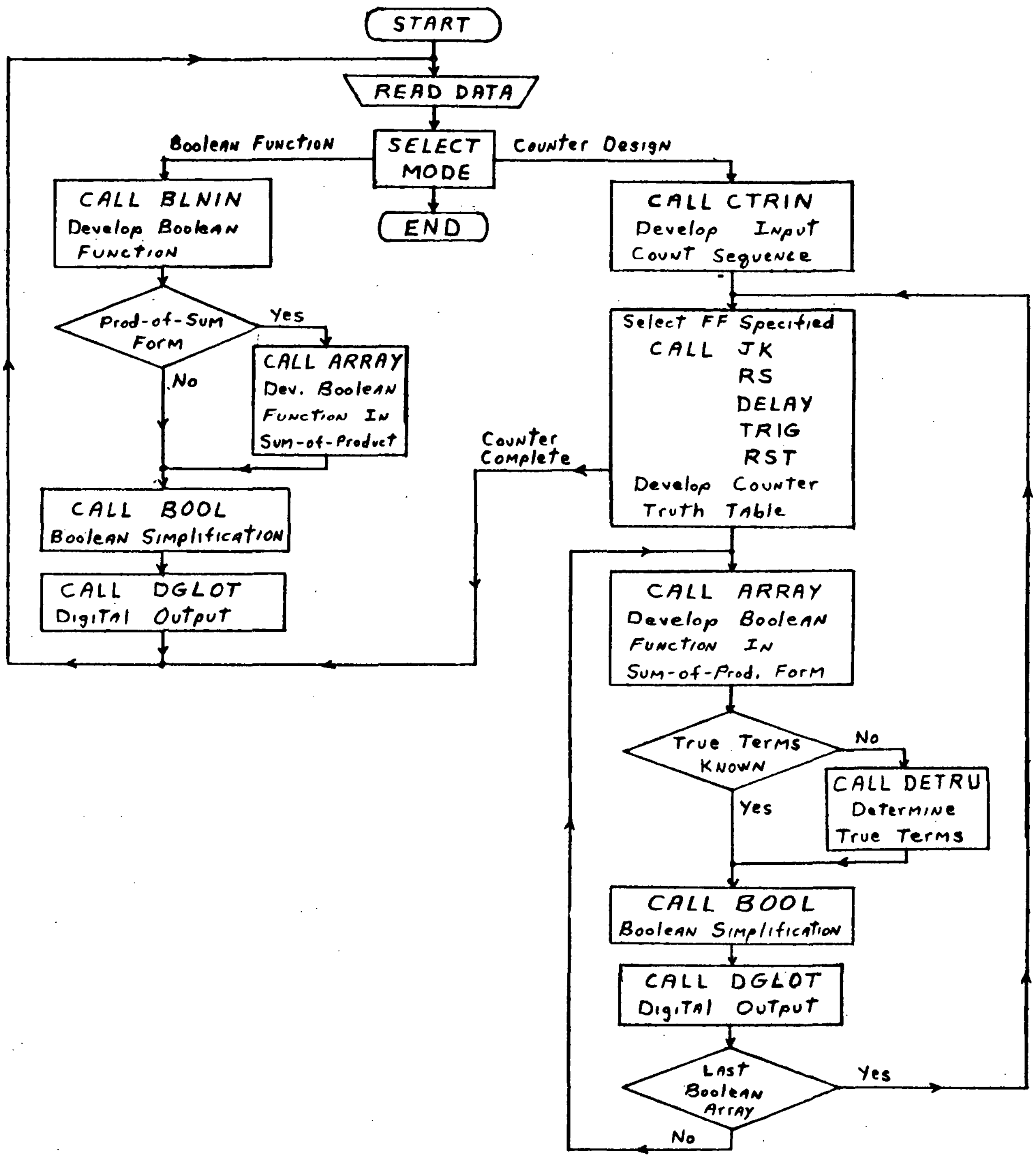


Figure 5.1. Block Diagram of Program DIGITL

characters and other symbols that are converted into a binary numerical array used by the Boolean simplification subroutine BOOL. The read in function can be either in sum-of-product or product-of-sum form but not a combination of both. The first left-hand parenthesis is used to detect the product-of-sum form. A product-of-sum array is first sent to subroutine ARRAY for conversion to a sum-of-product array before going on to subroutine BOOL for simplification.

Subroutine BLNIN begins by searching each data card to determine the number of Boolean true and don't care terms, the number of variables involved, and the alphabetical letters used to represent the variables. On the data cards, the letters (variables) used in each term do not have to appear in alphabetical order nor must each variable appear in each term. The subroutine prints out the Boolean function exactly as it appears on the data cards and also prints out an alphabetical listing of the letters used.

By knowledge of the total number of variables involved and what letter represents each variable, the A(I,J) array is then generated. Term A(I,1) designates the first term with the rows (I's) representing the variables of the term. The number representation used is "1" for true, "0" for false or not condition, and "3" for variable missing or no data. For example, if the Boolean function on the data card was ABC' + C'B' + BCA' + B', the "A" array generated would be:

$$\begin{array}{c|c|c|c|c|l} & 1 & 2 & 3 & 4 & \leftarrow \text{term number} \\ \hline A & 1 & 3 & 0 & 3 & \\ \hline B & 1 & 0 & 1 & 0 & \\ \hline C & 0 & 0 & 1 & 3 & \\ \hline \end{array}$$

5.1.3 <u>Subroutine CTRIN</u> - <u>Counter Input</u>

The main function of this subroutine is to take the input count sequence and generate the two dimensional binary array "NOBI" containing the input count sequence plus the unused or don't care counts. Array "NOBI" is the same as array "B" used in other subroutines. The subroutine also detects the flip flop types specified and stores this in a one dimensional array called "FCODE".

The program begins by detecting the flip flop code letters specified on the first data card and generates the "FCODE" array. Flip flop types not specified are signified by FCODE(I)=7. Next the data cards containing the input count sequence are read. The subroutine prints out the flip flop code letters and the input count data exactly as they appear on the data cards.

The subroutine then generates the input count sequence logic which is stored in binary array "NOBI(I,J)". The largest number is found to determine the maximum number of binary bits required. If MX is the number of binary bits, then there are 2^{MX} possible count states with the highest count equal to $2^{MX}-1$. The input count sequence read in decimal numbers (modulo 10) is converted to binary numbers (modulo 2), and stored in array "NOBI". Next the count states not used, called don't care states, are converted to binary and are stored in "NOBI".

The subroutine then prints out all the count states in both decimal and binary form under the heading of "Input Count Sequence Logic". The letter "A" is always used to represent the most significant bit of each binary number. For example, if the count sequence on the data card were 0, 1, 2, 3, 5, the array "NOBI" generated would be as follows:

	A	B	C	
0	0	0	0	
1	0	0	1	
2	0	1	0	
3	0	1	1	
5	1	0	1	
0	0	0	0	
4	1	0	0	
6	1	1	0	don't care terms
7	1	1	1	

The array "NOBI" is to be used later by one of the flip flop subroutines.

5.1.4 Subroutine JKRS - JK and RS Flip Flops

Subroutine JKRS contains the flip flop transition input requirements, derived from the counter truth table, for both the JK and RS flip flops as given in Section 2.3. For a given binary count sequence (developed in subroutine CTRIN) the subroutine enters into the counter truth table the input transition requirements for each flip flop. The count sequence from subroutine CTRIN is stored in a two dimensional array "B(I,J)". The flip flop transition requirements are added to this array which then contains the counter truth table logic. The transition requirements for the don't care count states, which are the same for all type flip flops, are not entered here but are entered by subroutine CTDC. The number 4 is used to represent a don't care condition in the truth table and is indicated in the printout by a dash mark (-).

For example, for the count sequence of 0, 1, 2, 3, 5 (used as example in subroutine CTRIN), the "B" array for the JK flip flop would appear as follows:

	A	B	C	JA	KA	JB	KB	JC	KC
0	0	0	0	0	4	0	4	1	4
1	0	0	1	0	4	1	4	4	1
2	0	1	0	0	4	4	0	1	4
3	0	1	1	1	4	4	1	4	0
5	1	0	1	4	1	0	4	4	1
0	0	0	0	0	0	0	0	0	0
4	1	0	0	4	4	4	4	4	4
6	1	1	0	4	4	4	4	4	4
7	1	1	1	4	4	4	4	4	4

generated by subroutine CTDC

don't care terms

5.1.5 Subroutine DLTRG - Delay and Trigger Flip Flops

Subroutine DLTRG generates the delay and trigger flip flop transition input requirements for a given count sequence. The transition requirements complete the counter truth table logic which is stored in the two dimensional array "B(I,J)". For description of this subroutine, refer to subroutine JKRS (Section 5.1.4) which is similar in function and operation.

5.1.6 Subroutine RST - RST Flip Flop

This subroutine generates the RST flip flop transition input requirements for a given count sequence. The operation is similar to that described for subroutine JKRS (Section 5.1.4). The input transition requirements for each flip flop in the counter truth table are generated and stored in a two dimensional array "B(I,J)".

The subroutine considers the RST flip flop for use as either an RS or a T flip flop. When an RST flip flop is specified, the subroutine is used for two different counter designs. The first design uses the "RST" as a RS flip flop and the second design uses the "RST" as a T (trigger) flip flop. The value of variable ICODE determines the operating mode of the RST flip flop.

5.1.7 Subroutine CTDC - Counter Don't Care Terms

Subroutine CTDC generates the don't care transition input requirements for the don't care terms of all counter designs. These don't care conditions complete the two dimensional array "B(I,J)" as illustrated for the JK flip flop subroutine in Section 5.1.4.

5.1.8 Subroutine ARRAY - Boolean Expression Array

This subroutine generates the Boolean input equations (one at a time) in sum-of-product form for each flip-flop and converts Boolean functions in product-of-sum form to sum-of-product form. These Boolean equations are later sent to subroutine BOOL for simplification. The subroutine has a normal and an alternate mode of operation. Basically, the normal mode develops the Boolean equation from sum-of-product terms while the alternate mode develops an equivalent equation from product-of-sum terms. The alternate mode is used when the number of counts specified is $\leq$ 1/6 of the maximum count states possible for the number of binary bits used, or when the Boolean expression read in is in product-of-sum form. Both modes produce Boolean equations in sum-of-product form regardless of the form to be used in printout.

Operation of the normal mode is as follows. The array "B(I,J)" from a flip flop subroutine (e.g. JKRS) is transfered into array "NBI(I,J)".

For output in sum-of-product form, the subroutine generates the Boolean function array "A(I,J)" for each flip flop input from the count states specified by the true and don't care transition requirements for that input. For example, for the JK flip flop array shown at the end of Section 5.1.4, input JA has one true term (count 3) and four don't care terms (counts 5, 4, 6, 7). The "A(I,J)" array generated for the J input of flip flop "A"(JA) is as follows:

$$\begin{array}{c|c|cccc}
 & 1 & 2 & 3 & 4 & 5 \\
\hline
A & 0 & 1 & 1 & 1 & 1 \\
B & 1 & 0 & 0 & 1 & 1 \\
C & 1 & \underbrace{1}_{} & 0 & 0 & 1
\end{array}$$

True Term Don't Care Terms

For output in product-of-sum form, the "A" array is obtained from the false and don't care transition requirements. For JA there are three false terms (counts 0, 1, 2).

The alternate mode for counter designs is used when the sequence contains only a small portion ($\leq$ 1/6) of the total possible count states. The terms specified by the false ("0") transition requirements are taken, and the entire expression is then inverted. The inverted "0" term expression is equivalent to the expression obtained from the true and don't care transition requirements. The true terms are then later separated from the don't care terms in subroutine DETRU. The reason for the alternate method is to save computer operating time by reducing the size of the "A" array which would otherwise contain a large number of don't care terms.

The inverted expression consists of product-of-sum terms which are converted into sum-of-product terms by a special expansion technique. The product-of-sum terms cannot simply be multiplied out as this would yield a voluminous number of terms. For example, eight terms each containing six variables would yield 6^8 or 1,680,000 terms. The technique used involves factoring out common variables beginning with the most common variable in alphabetical order. Also, the alternate mode handles Boolean expressions that are initially read in product-of-sum-form.

The operation of the alternate mode, for sum-of-product printout, is as follows. First, the "1" transition terms are stored in array "AT(I,J)" for later use in subroutine DETRU. Then the inverted "0" terms are stored in "AT". The inverted "0" terms are processed within the "AT" array producing reduced sum-of-product terms which are stored in array "A(I,J)". The resulting sum-of-product expression is usually close to its most simplified form. Usually each term is reduced to its simplest form but some unnecessary terms may be included. For printout in product-of-sum form, the role of the "0" terms and "1" terms is reversed. Here the "1" terms would be inverted and then processed.

The expansion technique used is illustrated by the following example. Let "JA" represent the expression generated in array "A" and "F" represent the product-of-sum expression contained in array "AT". Initially JA = 0. As "F" is reduced, the sum-of-product terms generated are transferred into "JA". The expansion is complete when F = 0. The expression shown for "F" is expanded as follows:

$$F = (A + B + C' + D')(A' + B + C' + D)(A + B + C + D')(A' + B + C + D)$$
$$JA = 0$$

Factoring out B gives,

$$F = B + (A + C' + D')(A' + C' + D)(A + C + D')(A' + C + D).$$

Transferring B from F into JA gives,

$$F = (A + C' + D')(A' + C' + D)(A + C + D')(A' + C + D)$$
$$JA = B.$$

Factoring out A gives,

$$F = A(C' + D)(C + D) + (C' + D')(A' + C' + D)(C + D')(A' + C + D).$$

When a variable is factored out, its complement is eliminated from the term factored out. Next the first term is further expanded by factoring out variable D giving,

$$F = AD + A(C')(C) + (C' + D')(A' + C' + D)(C + D')(A' + C + D).$$

Since $AC'C = 0$, only term AD is transferred into JA giving,

$F = (C' + D')(A' + C' + D)(C + D')(A' + C + D)$

$JA = B + AD.$

Factoring out A' gives,

$F = A'(C' + D')(C + D') + (C' + D')(C' + D)(C + D')(C + D).$

Then, the first term is further expanded by factoring out D' giving,

$F = A'D' + A'(C')(C) + (C' + D')(C' + D)(C + D')(C + D).$

Transferring A'D' into JA and eliminating $A'C'C = 0$ gives,

$F = (C' + D')(C' + D)(C + D')(C + D)$

$JA = B + AD + A'D'.$

Factoring out C gives,

$F = C(D')(D) + (C' + D')(C' + D)(D')(D) = 0.$

Since $F = 0$, the expansion is complete and the resulting input equation is $JA = B + AD + A'D'$.

Each term produced by factoring out a variable is further expanded by factoring out the next most common variable and so on for five times or until only sum-of-product terms remain. Only if the expression contains more than 32 product-of-sum terms is it possible for one expanded term to still contain product-of-sum factors after five repetitions of factoring out common variables. When this occurs, the subroutine multiplies out the remaining factors.

The multiplication section is complete by itself and can be used to expand product-of-sum terms directly if desired for some other application. The multiplication section consists of cards 131-169 on the source program listing.

5.1.9 Subroutine DETRU - Determine True Terms

When the alternate mode is used in subroutine ARRAY, the true and don't care terms generated in the "A" array are intermingled. Subroutine DETRU selects the minimum required true terms from the array and then places the true terms at the front of the array. The "A" array is now in proper form for further simplification by subroutine BOOL.

The subroutine functions as follows. Array "A(I,J)" is transferred into array "AT(I,J)" which already contains the "1" terms entered in subroutine ARRAY. A true term matrix is generated in array "IA(I,J)" which cross references the "A" array terms to the "1" terms that they contain. The rows correspond to "1" terms and the columns to "A" array terms that contain at least one "1" term. As an example, assume the "1" terms are counts 2, 10, 42, 65, 107 and the "A" array contains expression JE given as:

JE = B' + E + AG' + CG + DG' + FG + A'G + C'G' + D'G + F'G'.

The true term matrix would then appear as follows:

B'	DG'	FG	C'G'	D'G	
①			8		A'B'C'D'E'FG' (2)
1	5		8		A'B'C'DE'FG' (10)
	5		⑧		A'BC'DE'FG' (42)
1				9	AB'C'D'E'F'G (65)
		⑥			ABC'DE'FG (107)

The position of the numbers in a column signify the "1" terms that are contained by the JE term listed above the column. The value of the number signifies the position of the column term in the JE expression. For example, C'G' is the 8th term in JE and contains the first three "1" terms.

The required true terms are determined by the following technique. First, the minimum row is determined. Then from the column terms appearing in this row, the column which contains the largest number of "1" terms is selected. If several columns contain the same number of "1" terms, the column term having the least number of variables is the one selected. All the rows ("1" terms) contained by the selected term are then eliminated from the matrix. The next minimum row is determined and the process repeated until all rows are eliminated. The terms selected now contain all the "1" terms and are therefore the true terms.

Applying this technique to the previous example gives row 5 as the first minimum row since only one column appears in this row. Hence FG is the first true term. The reason for selecting the minimum row is made clearer by noting that FG is the only term that contains the fifth "1" term (ABC'DE'FG) and therefore must be one of the true terms. The next minimum row is row 1. Both terms B' and C'G' in this row contain three "1" terms and term B' is selected since it contains one less variable. Rows 1, 2, 4, 5 have now been eliminated. Obviously, row 3 is the next minimum row and term C'G' is selected over DG' since it contains more "1" terms. All rows are now eliminated and the resulting true terms are FG, B', and C'G'. The remaining terms in JE are don't care terms. The true term expression obtained in this alternate mode will always be at least as minimized (fewest number of terms and variables) as the expression obtained in the normal mode.

The subroutine then transfers the expression in array "AT" into array "A" with the true terms being transferred first. For the previous example, JE would now appear as:

JE = FG + B' + C'G'/E + AG' + CG + DG' + A'G + D'G + F'G'.

5.1.10 <u>Subroutine BOOL</u> - <u>Boolean Simplification</u>

The Boolean simplification subroutine implements the Boolean simplification theory discussed in Section 2.1. The subroutine follows the sequence of steps described in this section. The first step applies the three theorems of the iterative method to obtain the prime implicants. Then, the tabular method is used to assure that the final expression is a minimum sum-of-product.

The subroutine transfers the Boolean expression that is to be simplified into a two dimensional array called the "A(I,J)" array. The true variables of the expression are read into the array as 1's and the complements as 0's. For example, the expression

$$\overline{A}BC + ABC + A\overline{B}C$$

would appear in the "A(I,J)" array as follows:

	1	2	3
A	0	1	1
B	1	1	0
C	1	1	1

Once the Boolean expression has been read into the array, the first step is to apply the first theorem of the iterative method to the array. The first theorem of the iterative method is:

$$1.\quad XY + \bar{X}Z = XY + XZ + YZ.$$

The function of this theorem is to generate the new term YZ out of the variables associated with X and its complement. The complement pairs are the 1's and 0's on each row, and the variables associated with a complement pair are the remaining 1's and 0's in those two colums.

An example of the first theorem is as follows:

Complement Pair New YZ Term

3 = No Data

A	0	1	1		3
B	1	1	0		1
C	1	1	1		1

The next step in the iterative method is to apply the second and third theorems. These are:

$$2.\quad Y + Y = Y$$
$$3.\quad Y + YZ = Y$$

These two theorems use the new term YZ to reduce or eliminate certain terms. For example, it can be seen from the previous array that the new term YZ will replace the first and second terms of the array. This is an application of the third theorem. After application of the third theorem, the first and second terms of the array are the same; therefore, the second theorem can be applied to eliminate the second term.

After the application of the second and third theorems, the previous array would appear as follows.

	1	2	3
A	3	3	1
B	1	3	0
C	1	3	1

3 = No Data

As the title, iterative method, implies, these three theorems are applied over and over to the array in the same order until all possible applications are completely exhausted. The complete iterative method is applied to the array three times. The first time only the true terms are considered. This is to group the true terms together before further simplification with the don't care terms. The second time, both the true and don't care terms are considered together. Because of the initial grouping of the true terms, the second application of the iterative method may not completely reduce all terms. Hence the iterative method is applied a third time. When this is done, the remaining terms in the array are prime implicants. Before the iterative method is applied for the third time, the array is compressed by eliminating all "no data (all 3's)" terms from the array. This provides working space in the array for application of the tabular method and reduces the time required to implement the remaining portion of the simplification technique.

The second part of the subroutine is the tabular method. The tabular method takes the results of the iterative method and eliminates all possible terms that are not required for the minimum Boolean expression. This is accomplished by expanding each term into an expanded sum-of-product, then checking to see if every term in the expanded sum-of-product is contained in the remaining terms. If every term is contained at least once, then the original term that was expanded is eliminated from the array.

For example, if the expression

$$BC + \bar{A}C + AB$$

were the final result of the iterative method, then the array would appear as follows:

	1	2	3
A	3	0	1
B	1	3	1
C	1	1	3

The first term in the "A" array is expanded into two terms in the working portion of the array which is behind the last term. The array now appears as follows:

$$\begin{array}{c|c|c|c|c|c|}
 & 1 & 2 & 3 & \multicolumn{2}{c}{\overbrace{\qquad\qquad}^{\text{Expanded Terms}}} \\
\hline
A & 3 & 0 & 1 & 0 & 1 \\
\hline
B & 1 & 3 & 1 & 1 & 1 \\
\hline
C & 1 & 1 & 3 & 1 & 1 \\
\hline
\end{array}$$

The arrows show what terms of the expanded terms are contained in the remaining two terms. The term expanded is obviously not considered.

Since all expanded terms are contained in the remaining terms, the term that was originally expanded is eliminated from the array.

The "A" array now appears as follows:

$$\begin{array}{c|c|c|c|}
 & \overset{\text{Eliminated}}{\overset{\downarrow}{1}} & 2 & 3 \\
\hline
A & 3 & 0 & 1 \\
\hline
B & 3 & 3 & 1 \\
\hline
C & 3 & 1 & 3 \\
\hline
\end{array}$$

This process of expanding each successive term and then comparing the expanded terms with the remaining terms is continued until all terms of the array have been considered. When this is done the final expression is minimal and the tabular method is complete.

The tabular method is applied to the array twice. The first time true terms are eliminated using only the true terms. The second time don't care terms are eliminated using both the true and don't care terms. This is because true terms can be used to help eliminate a don't care term but don't care terms can't eliminate a true term.

5.1.11 <u>Subroutine DGLOT</u> - <u>Digital Output</u>

This subroutine prints out most of the calculated digital data. In the counter design mode, the subroutine prints out the flip flop transition input requirements, the counter truth table, the flip flop block diagram, and the flip flop input equations in either sum-of-product or product-of-sum form. In the Boolean simplification mode, the subroutine prints out the simplified Boolean function in sum-of-product form.

In the counter design mode, the operation of the subroutine is as follows. First the flip flop code letters (e.g. JK) are generated for the flip flops in use from variable FCODE. Next the transition input requirements and the counter truth table are printed out. If the short form output option (NRT = 1) is specified, these printouts are bypassed. The array "NBI(I,J)", which contains the counter truth table, cannot be directly printed out because of different spacing requirements for the binary sequence portion and the input transition portion of the array. Hence, a new array "ROW(I)" is generated and printed out with the proper spacing between characters for each row of array "NBI(I,J)".

Next, the Boolean input flip flop equations are printed out. Each time the subroutine is called, one equation brought in through array "A(I,J)" is printed out. The Boolean variables contained in array "A" in binary form are converted into alphabetical letters for printout. The variables designate flip flops which are assigned letters in alphabetical order, with flip flop A always representing the most significant binary bit of the count sequence.

The Boolean equation printout is normally in sum-of-product form. In this form, the equation can contain both true and don't care terms. If variable MPS = 1, the printout is in product-of-sum form. When this form is used, the complement of the variables in array "A" are printed out and only true terms are used. When the printout requires more than one line, the last term on each line is completed before beginning the next line. A term is never split up between two lines.

In the Boolean simplification mode, the subroutine branches directly to the Boolean equation printout section used in the counter design mode. The Boolean variables are assigned the same alphabetical letters as used in the expression read in. The letters read in are stored in array LETR(I). Only sum-of-product printout is used in the simplification mode.

For an example of the Boolean equation generated from array "A(I,J)" under different output modes, assume array "A(I,J)", containing three true terms and one don't care term, is given as follows:

$$\begin{array}{c|c|c|c|c|} & 1 & 2 & 3 & 4 \\ \hline A & 0 & 1 & 1 & 3 \\ \hline B & 1 & 3 & 0 & 0 \\ \hline C & 1 & 0 & 3 & 0 \\ \hline \end{array}$$

True Terms   Don't Care Terms

For sum-of-product printout with true and don't care terms,

$$F = A'BC + AC' + AB' \ / \ B'C'.$$

For sum-of-product printout with true terms only,

$$F = A'BC + AC' + AB'.$$

For product-of-sum printout,

$$F = (A + B' + C')(A' + C)(A' + B)$$

5.2 SOURCE LISTINGS AND FLOW CHARTS

The listing of the computer main-line program and subroutine programs along with their flow charts are given in Figure 5.2 through 5.23. The programs are written in FORTRAN IV for the UNIVAC 1108 computer and can be adapted easily to any computer using FORTRAN IV or FORTRAN V.

Main-line Program

1.	DIGITL	-	Digital Logic Design

Subroutine Program

1.	BLNIN	-	Boolean Input
2.	CTRIN	-	Counter Input
3.	JKRS	-	JK and RS FF's
4.	DLTRG	-	Delay
5.	RST	-	RST FF
6.	CTDC	-	Counter Don't Care Terms
7.	ARRAY	-	Boolean Expression Array
8.	DETRU	-	Determine True Terms
9.	BOOL	-	Boolean Simplification
10.	DGLOT	-	Digital Output

```
C  MAX. OF 10 VARIABLES AND APPROX. 250 TERMS FOR BOOLEAN INPUT.         DGTL
C  HIGHEST NO. AND MAX. NO. OF STATES FOR COUNTER INPUT IS 255.          DGTL
C  PROGRAM FOR 255 COUNTS USES ABOUT 25,500 WORDS MEMORY                 DGTL
C  NO(I) = DECIMAL NUMBER SEQUENCE FOR COUNTER                           DGTL
C  A(I,J) = BOOLEAN LOGIC ARRAY,  B(I,J) = BINARY SEQUENTIAL ARRAY       DGTL
C  K = NUMBER OF VARIABLES,  L = NUMBER OF INPUT COUNT STATES            DGTL
C  N = NUMBER OF TRUE + DON'T CARE TERMS,  M = NUMBER OF TRUE TERMS      DGTL
C  NH = MAX. NO. OF COLUMNS IN B ARRAY,  NC = COLUMN NO. OF B ARRAY      DGTL
C  LETR = LETTER ARRAY,  FCODE(I) = FLIP-FLOP CODE NUMBER                DGTL
C  NRT = CODE NUMBER FOR SHORT FORM OUTPUT                               DGTL
C  AT(I,J) = PRODUCT OF SUM TERM ARRAY, MPS = INPUT/OUTPUT FORM CODE NO.DGTL
      INTEGER BLANK, DATR, FCODE, A, C, F, AT                           DGTL
      DIMENSION A(10,257),NBI(257,32),DATR(80),LETR(26),FCODE(9),NO(257)DGTL
      DIMENSION AT(10,100)                                              DGTL
      COMMON A                                                          DGTL
      DATA BLANK,C,F/1H ,1HC,1HF/                                       DGTL
   10 READ(5,15) (DATR(I),I = 1,80)                                     DGTL
   15 FORMAT (80A1)                                                     DGTL
      MODE = DATR(1)                                                    DGTL
      IF(MODE .EQ. BLANK) CALL EXIT                                     DGTL
      IF(MODE .EQ. C) GO TO 20                                          DGTL
      IF(MODE .EQ. F) GO TO 60                                          DGTL
      GO TO 10                                                          DGTL
   20 CALL CTRIN (DATR, NBI, FCODE, K, L, LETR, NO, NRT, MPS)           DGTL
      I = 1                                                             DGTL
   22 ICODE = FCODE(I)                                                  DGTL
      GO TO (30,30,34,34,38,38,10,24), ICODE                            DGTL
   24 I = I + 1                                                         DGTL
      GO TO 22                                                          DGTL
   30 CALL JKRS  (L,K,ICODE,NBI,NH,NC)                                  DGTL
      GO TO 40                                                          DGTL
   34 CALL DLTRG (L,K,ICODE,NBI,NH,NC)                                  DGTL
      GO TO 40                                                          DGTL
   38 CALL RST   (L,K,ICODE,NBI,NH,NC)                                  DGTL
   40 CALL CTDC  (L,NH,NC,K,NBI)                                        DGTL
   50 CALL ARRAY (NBI, K, NC, L, MPS, A, N, M, AT, MIN)                 DGTL
      IF(L .LE. 2**K/6) CALL DETRU (K,MIN,AT,NRT,A,N,M)                 DGTL
      CALL BOOL (N,K,M,NRT,A)                                           DGTL
   54 CALL DGLOT (A,LETR,K,N,M,MODE,FCODE(I),NC,NO,NBI,NH,NRT,MPS)      DGTL
      IF(NC .EQ. NH + 1) GO TO 24                                       DGTL
      GO TO 50                                                          DGTL
   60 CALL BLNIN (DATR, LETR, A, K, N, M, NRT, MPS)                     DGTL
      IF(MPS .EQ. -1) CALL ARRAY(NBI,K,NC,L,MPS,A,N,M,AT,MIN)           DGTL
      CALL BOOL (N,K,M,NRT,A)                                           DGTL
      CALL DGLOT (A,LETR,K,N,M,MODE,FCODE(I),NC,NO,NBI,NH,NRT,MPS)      DGTL
      GO TO 10                                                          DGTL
      END                                                               DGTL
```

Figure 5.2. Program DIGITL

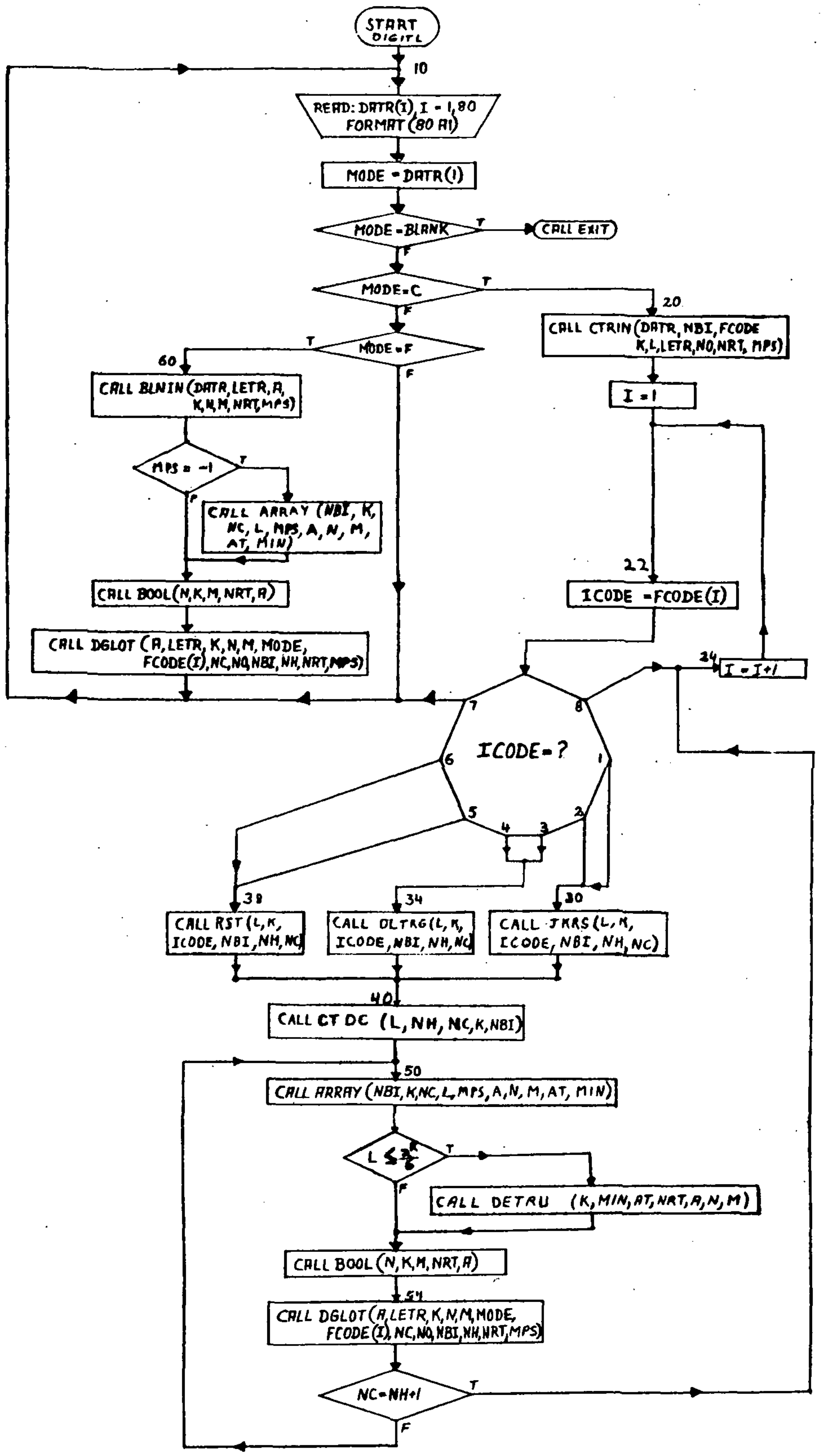

START DIGITL
10
READ: DATR(I), I = 1,80 FORMAT (80 A1)
MODE = DATR(1)
MODE = BLANK
CALL EXIT
MODE = C
20
CALL CTRIN (DATR, NBI, FCODE K, L, LETR, NO, NRT, MPS)
I = 1
MODE = F
60
CALL BLNIN (DATR, LETR, A, K, N, M, NRT, MPS)
MPS = -1
CALL ARRAY (NBI, K, NC, L, MPS, A, N, M, AT, MIN)
CALL BOOL (N, K, M, NRT, A)
CALL DGLOT (A, LETR, K, N, M, MODE, FCODE(I), NC, NO, NBI, NH, NRT, MPS)
22
ICODE = FCODE(I)
24
I = I+1
ICODE = ?
38
CALL RST (L, K, ICODE, NBI, NH, NC)
34
CALL DLTRG (L, K, ICODE, NBI, NH, NC)
30
CALL JKRS (L, K, ICODE, NBI, NH, NC)
40
CALL CT DC (L, NH, NC, K, NBI)
50
CALL ARRAY (NBI, K, NC, L, MPS, A, N, M, AT, MIN)
L ≤ 2^K/6
CALL DETRU (K, MIN, AT, NRT, A, N, M)
CALL BOOL (N, K, M, NRT, A)
54
CALL DGLOT (A, LETR, K, N, M, MODE, FCODE (I), NC, NO, NBI, NH, NRT, MPS)
NC = NH+1

Figure 5.3. Flow Chart of Program DIGITL

```
      SUBROUTINE BLNIN (DATR, LETR, A, K, N, M, NRT, MPS)
      INTEGER A,DATA,SYMB,BLANK,BAR,PLUS,STAR,DATR,SLH,RPAR
      DIMENSION DATR(1),LETR(1),A(10,1),SYMB(26),DATA(500)
      DATA (SYMB(I),I =1,26)/1HA,1HB,1HC,1HD,1HE,1HF,1HG,1HH,1HI,1HJ,
     1  1HK,1HL,1HM,1HN,1HO,1HP,1HQ,1HR,1HS,1HT,1HU,1HV,1HW,1HX,1HY,1HZ/
      DATA BLANK,BAR,PLUS,STAR/1H ,1H',1H+,1H*/,SLH/1H//
      DATA LPAR,RPAR/1H(,1H)/
      MPS = 0
      N = 1
      IM = 0
      NRT = 0
      M = 0
      DO 10 I = 1,26,1
      LETR(I) = BLANK
   10 CONTINUE
      IN = 5
      IO = 6
      DATR(1) = BLANK
      WRITE (IO,12)
   12 FORMAT (1H1, 10X,'INPUT FUNCTION    (F = TRUE TERMS /'
     1    18H DON'T CARE TERMS),/30X,
     2      '* TERMINATES BOOLEAN EXPRESSION',///15X,'F = ')
   17 IBK = 0
      WRITE (IO,18) (DATR(J),J = 1,80)
   18 FORMAT (1H+,18X,80A1,//)
      DO 20 J = 1,80
      IF(DATR(J) .EQ. BLANK) GO TO 24
      IM =IM + 1
      DATA(IM) = DATR(J)
      IF(MPS .EQ. -1) GO TO 21
      IF(DATA(IM) .EQ. LPAR) GO TO 22
      IF(DATA(IM) .EQ. PLUS) GO TO 26
      IF(DATA(IM) .EQ. SLH) GO TO 27
   21 IF(DATA(IM) .EQ. BAR .OR. DATA(IM) .EQ. RPAR) GO TO 20
      IF(DATA(IM) .EQ. LPAR) GO TO 26
      IF(DATA(IM) .EQ. STAR) GO TO 30
      GO TO 28
   22 MPS = -1
      GO TO 20
   24 IBK = IBK + 1
      IF(IBK .GE. 80) GO TO 30
      GO TO 20
   27 M = N
   26 N = N +1
      GO TO 20
   28 DO 19 I = 1,26
      IF (SYMB(I) .EQ. DATA(IM)) GO TO 29
      GO TO 19
   29 LETR(I) = SYMB(I)
      GO TO 20
   19 CONTINUE
   20 CONTINUE
   15 READ (IN,16) (DATR(J),J = 1,80)
   16 FORMAT (80A1)
      GO TO 17
```

Figure 5.4. Subroutine BLNIN (Sheet 1 of 2)

```
 30   IF(M .EQ. 0) M = N
      K = 0
      DO 34 I = 1,26
      IF(LETR(I) .EQ. BLANK) GO TO 34
      K = K +1
      LETR(K) = LETR(I)
 34   CONTINUE
      WRITE (IO,36) (LETR(I),I = 1,K)
 36   FORMAT (1H ,10X,'LETTERS USED = '30A1)
      DO 38 J = 1,N
      DO 38 I = 1,K
      A(I,J) = 3
 38   CONTINUE
      L = 1
      J = 0
 40   J = J + 1
      IF(MPS.EQ.0.AND.(DATA(J).EQ.PLUS.OR.DATA(J).EQ.SLH)) GO TO 48
      IF(MPS.EQ.-1.AND.DATA(J).EQ.RPAR) GO TO 48
      DO 42 I = 1,K
      IF(LETR(I) .EQ. DATA(J)) GO TO 44
 42   CONTINUE
      IF(IM .GT. J) GO TO 40
      GO TO 50
 44   IF(DATA(J+1) .EQ. BAR) GO TO 46
      A(I,L) = 1
      GO TO 40
 46   A(I,L) = 0
      J = J + 1
      GO TO 40
 48   L = L + 1
      GO TO 40
 50   CONTINUE
      RETURN
      END
```

Figure 5.4. Subroutine BLNIN (Sheet 2 of 2)

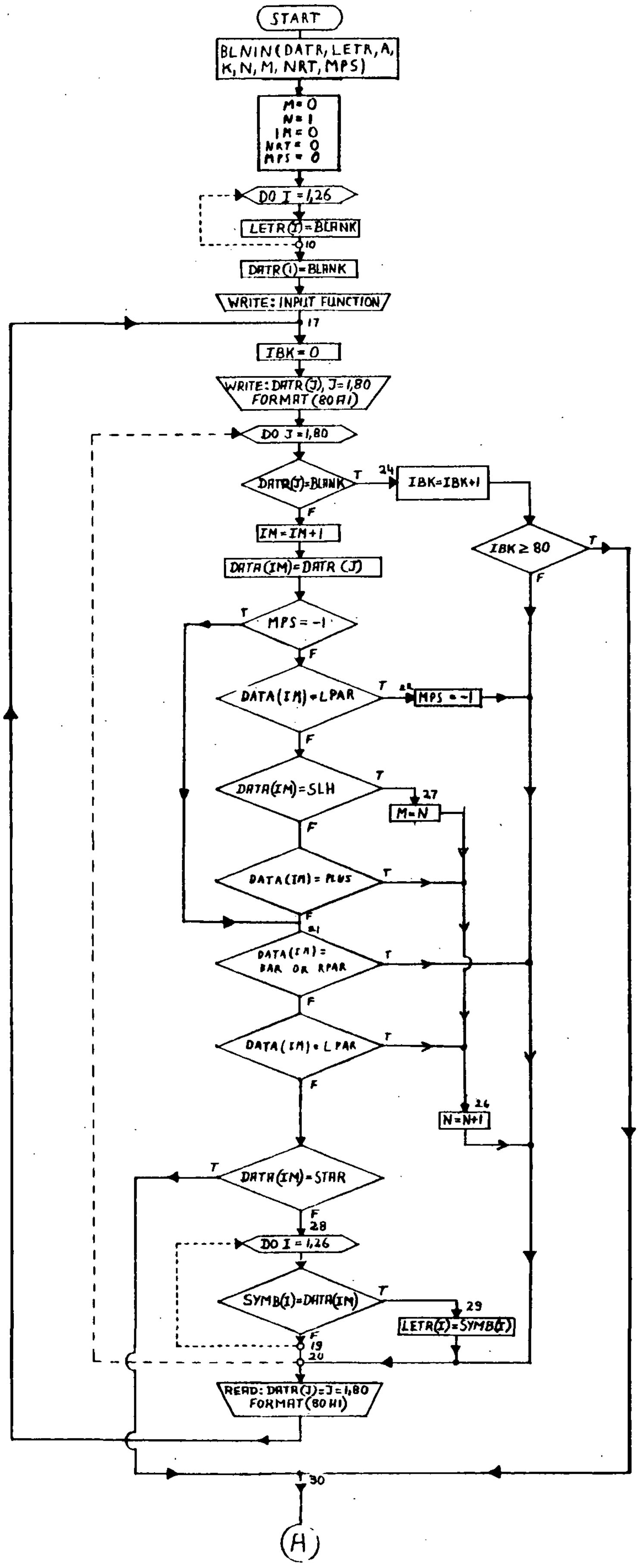


Figure 5.5. Flow Chart of Subroutine BLNIN (Sheet 1 of 2)

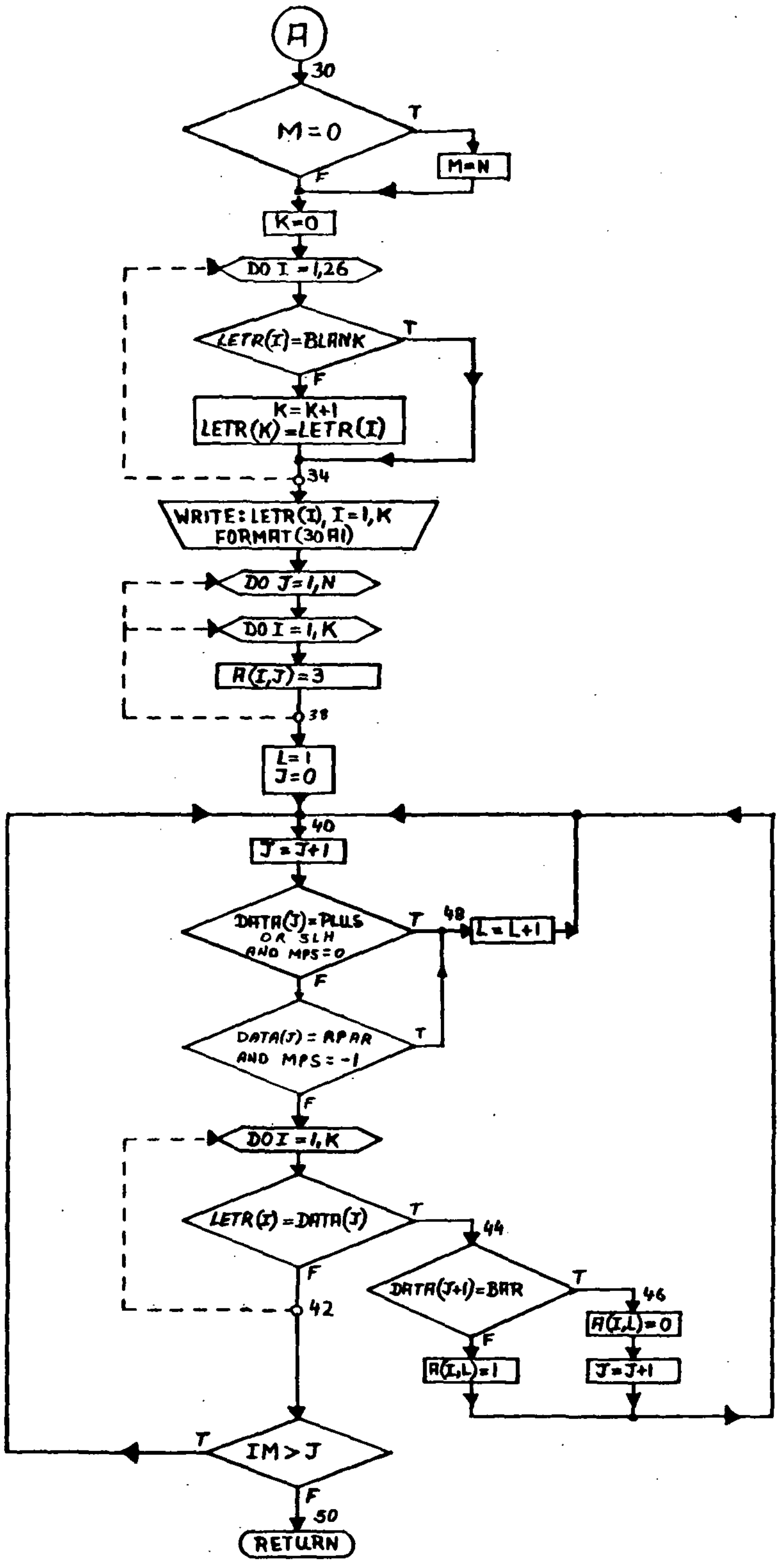


Figure 5.5. Flow Chart of Subroutine BLNIN (Sheet 2 of 2)

```
      SUBROUTINE CTRIN (DATR, NOBI, FCODE, MX, M, LETR, NO, NRT, MPS)
      INTEGER DATR,FCODE,BLANK,COMMA,A,D,R,T,CLR,SLH
      DIMENSION DATR(1),NOBI(257,1),FCODE(1),LETR(1),NO(1),NBLE(26)
      DATA BLANK,COMMA,A,D,JJ,R,T,SLH/1H ,1H,,1HA,1HD,1HJ,1HR,1HT,1H//
      DATA (NBLE(I),I =1,26)/1HA,1HB,1HC,1HD,1HE,1HF,1HG,1HH,1HI,1HJ,
     1  1HK,1HL,1HM,1HN,1HO,1HP,1HQ,1HR,1HS,1HT,1HU,1HV,1HW,1HX,1HY,1HZ/
      IN = 5
      IO = 6
      MPS = 0
      DO 7 I = 1,26
      LETR(I) = NBLE(I)
    7 CONTINUE
      WRITE (IO,8) (DATR(I), I = 2,80)
    8 FORMAT (1H1,15X'FLIP-FLOP TYPES SPECIFIED FOR COUNTER'//
     1    16X'FLIP-FLOPS I '79A1//)
      FCODE(7) = 7
      CLR = 1
      NRT = 0
      DO 10 I = 1,6
      FCODE(I) = 8
   10 CONTINUE
      DO 38 J = 2,80
      IF(DATR(J) .EQ. BLANK) GO TO 38
      IF(DATR(J) .EQ. SLH) NRT = 1
      IF(DATR(J) .EQ. NBLE(16)) MPS = 1
      IF(DATR(J) .EQ. COMMA) GO TO 12
      IF(CLR .EQ. 0) GO TO 38
      GO TO 14
   12 CLR = 1
      GO TO 38
   14 IF(DATR(J) .EQ. A) GO TO 16
      IF(DATR(J) .EQ.JJ) GO TO 20
      IF(DATR(J) .EQ. R) GO TO 22
      GO TO 26
   16 DO 18 I = 1,4
      FCODE(I) = I
   18 CONTINUE
      GO TO 38
   20 FCODE(1) = 1
      GO TO 36
   22 IF(DATR(J+2) .EQ.T) GO TO 24
      FCODE(2) = 2
      GO TO 36
   24 FCODE(5) = 5
      FCODE(6) = 6
      GO TO 36
   26 IF(DATR(J) .EQ. T) GO TO 28
      IF(DATR(J) .EQ. D) GO TO 30
      GO TO 38
   28 FCODE(4) = 4
      GO TO 36
   30 FCODE(3) = 3
   36 CLR = 0
   38 CONTINUE
   40 READ(IN,42) M, (NO(I),I = 1,M)
```

Figure 5.6. Subroutine CTRIN (Sheet 1 of 2)

```
   42 FORMAT (16I5)
      WRITE(10,44) M,(NO(I),I = 1,M)
   44 FORMAT (1H0,15X'INPUT COUNT DATA,  FIRST NO. IS NUMBER OF COUNTS'
     1  //(14X,16I5/))
      IF(NO(M) .EQ. NO(M-1)) GO TO 46
      L = M + 1
      NO(L) = NO(1)
      GO TO 48
   46 L = M
      M = M - 1
   48 NBIG = NO(M)
      DO 50 I = 1,M
      IF(NO(I) .GT. NBIG) NBIG = NO(I)
   50 CONTINUE
      BIG = NBIG
      MX = ALOG10(BIG)/ALOG10(2.0)+ 1.000001
      NOMX = 2**MX - 1
      NOS = 0
   52 DO 54 I = 1,M
      IF(NO(I) .EQ. NOS) GO TO 56
   54 CONTINUE
      L = L + 1
      NO(L) = NOS
   56 IF(NOS .GE. NOMX) GO TO 60
      NOS = NOS + 1
      GO TO 52
   60 CONTINUE
      DO 64 I = 1,L
      NU = NO(I)
      DO 62 J = 1,MX
      NOBI(I,MX-J+1) = NU - (NU/2)*2
      NU = NU/2
   62 CONTINUE
   64 CONTINUE
      WRITE(10,72)  (LETR(I),I=1,MX)
   72 FORMAT (///1H0,20X'INPUT COUNT SEQUENCE LOGIC'///
     1    (23X'I'5X'NO(I)'3X26(1XA1)))
      DO 76 I = 1,L
      IF(I .EQ. M+2) WRITE (10,73)
   73 FORMAT (1H0,20X'--------------',23H DON'T CARE TERMS BELOW)
      WRITE(10,74) I,NO(I),(NOBI(I,J),J=1,MX)
   74 FORMAT (1H0,19XI4,5XI4,4X26I2)
   76 CONTINUE
      RETURN
      END
```

Figure 5.6. Subroutine CTRIN (Sheet 2 of 2)

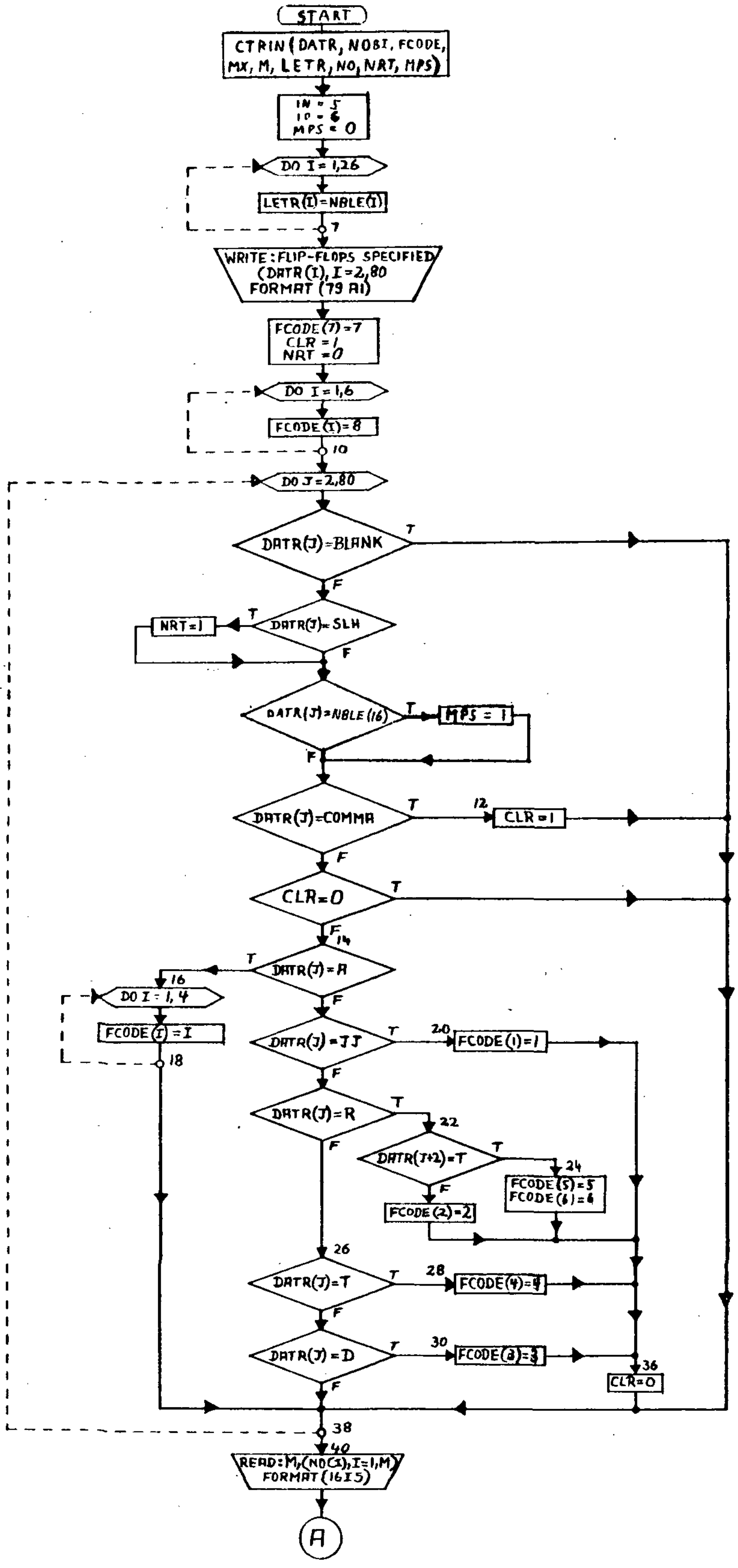


Figure 5.7. Flow Chart of Subroutine CTRIN (Sheet 1 of 2)

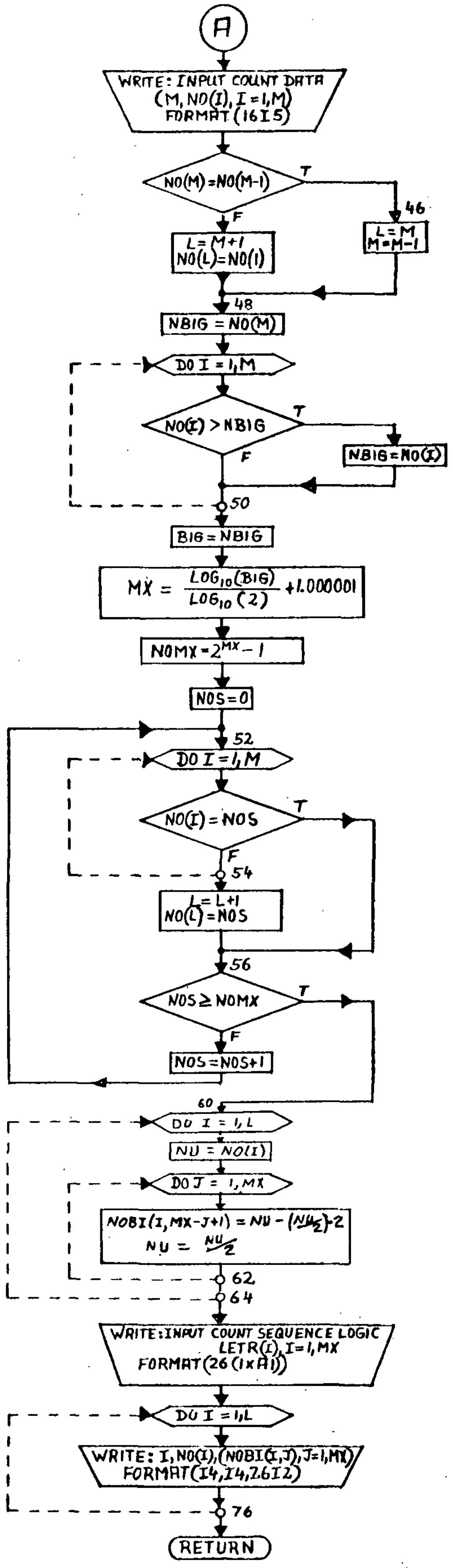

A
WRITE: INPUT COUNT DATA (M, NO(I), I=1,M) FORMAT(16I5)
NO(M)=NO(M-1)
T
F
46
L=M
M=M-1
L=M+1
NO(L)=NO(1)
48
NBIG=NO(M)
DO I=1,M
NO(I)>NBIG
T
F
NBIG=NO(I)
50
BIG=NBIG
MX = LOG10(BIG)/LOG10(2) +1.000001
NOMX=2^MX-1
NOS=0
52
DO I=1,M
NO(I)=NOS
T
F
54
L=L+1
NO(L)=NOS
56
NOS≥NOMX
T
F
NOS=NOS+1
60
DO I=1,L
NU=NO(I)
DO J=1,MX
NOBI(I,MX-J+1)=NU-(NU/2)*2
NU=NU/2
62
64
WRITE: INPUT COUNT SEQUENCE LOGIC LETR(I), I=1,MX FORMAT(26(1xA1))
DO I=1,L
WRITE: I,NO(I),(NOBI(I,J),J=1,MX) FORMAT(I4,I4,26I2)
76
RETURN

Figure 5.7. Flow Chart of Subroutine CTRIN (Sheet 2 of 2)

```
      SUBROUTINE JKRS  (M,X,ICODE,B,N,C)
      INTEGER B,C,F,H,P,T,W,X,Z,SUM
      DIMENSION B(257,1)
      N=3*X
      C=X+1
      F=2**X
      P=C
      Z=0
   23 Z=Z+1
      IF(Z .EQ. X + 1) RETURN
      W=0
      T=P+1
   16 W=W+1
      IF(W.EQ.M+1) GO TO 21
      H=W+1
      SUM=B(W,Z) - B(H,Z)
      IF(SUM)18,19,20
   18 IF(ICODE.EQ.1) GO TO 80
      B(W,P)=1
      B(W,T)=0
      GO TO 16
   80 B(W,P)=1
      B(W,T)=4
      GO TO 16
   20 IF(ICODE.EQ.1) GO TO 81
      B(W,P)=0
      B(W,T)=1
      GO TO 16
   81 B(W,P)=4
      B(W,T)=1
      GO TO 16
   19 IF(B(W,Z).EQ.1) GO TO 17
      B(W,P)=0
      B(W,T)=4
      GO TO 16
   17 B(W,P)=4
      B(W,T)=0
      GO TO 16
   21 P=P+2
      GO TO 23
      END
```

Figure 5.8. Subroutine JKRS

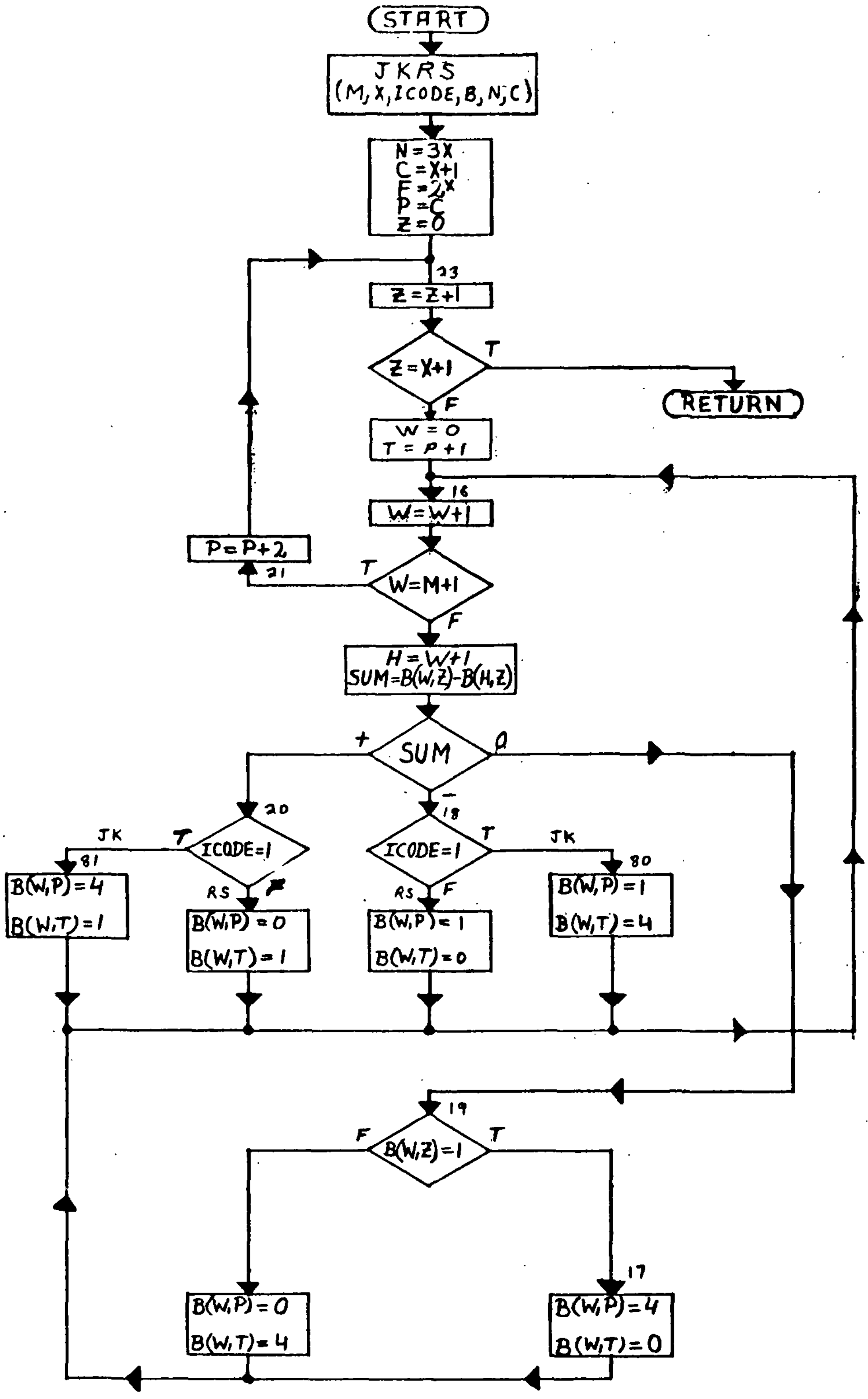


Figure 5.9. Flow Chart of Subroutine JKRS

```
      SUBROUTINE DLTRG (M,X,ICODE,B,N,C)                                DLTG   1
      INTEGER B,C,F,H,P,W,X,Z,SUM                                       DLTG   2
      DIMENSION B(257,1)                                                DLTG   3
      N=2*X                                                             DLTG   4
      C=X+1                                                             DLTG   5
      F=2**X                                                            DLTG   6
      P=C                                                               DLTG   7
      Z=0                                                               DLTG   8
   23 Z=Z+1                                                             DLTG   9
      IF(Z .EQ. X + 1) RETURN                                           DLTG  10
      W=0                                                               DLTG  11
   16 W=W+1                                                             DLTG  12
      IF(W.EQ.M+1) GO TO 21                                             DLTG  13
      H=W+1                                                             DLTG  14
      IF(ICODE.EQ.3) GO TO 80                                           DLTG  15
      SUM=B(W,Z)-B(H,Z)                                                 DLTG  16
   82 IF(SUM.EQ.0) GO TO 81                                             DLTG  17
      B(W,P)=1                                                          DLTG  18
      GO TO 16                                                          DLTG  19
   80 SUM=B(H,Z)                                                        DLTG  20
      GO TO 82                                                          DLTG  21
   81 B(W,P)=0                                                          DLTG  22
      GO TO 16                                                          DLTG  23
   21 P=P+1                                                             DLTG  24
      GO TO 23                                                          DLTG  25
      END                                                               DLTG  26
```

Figure 5.10. Subroutine DLTRG

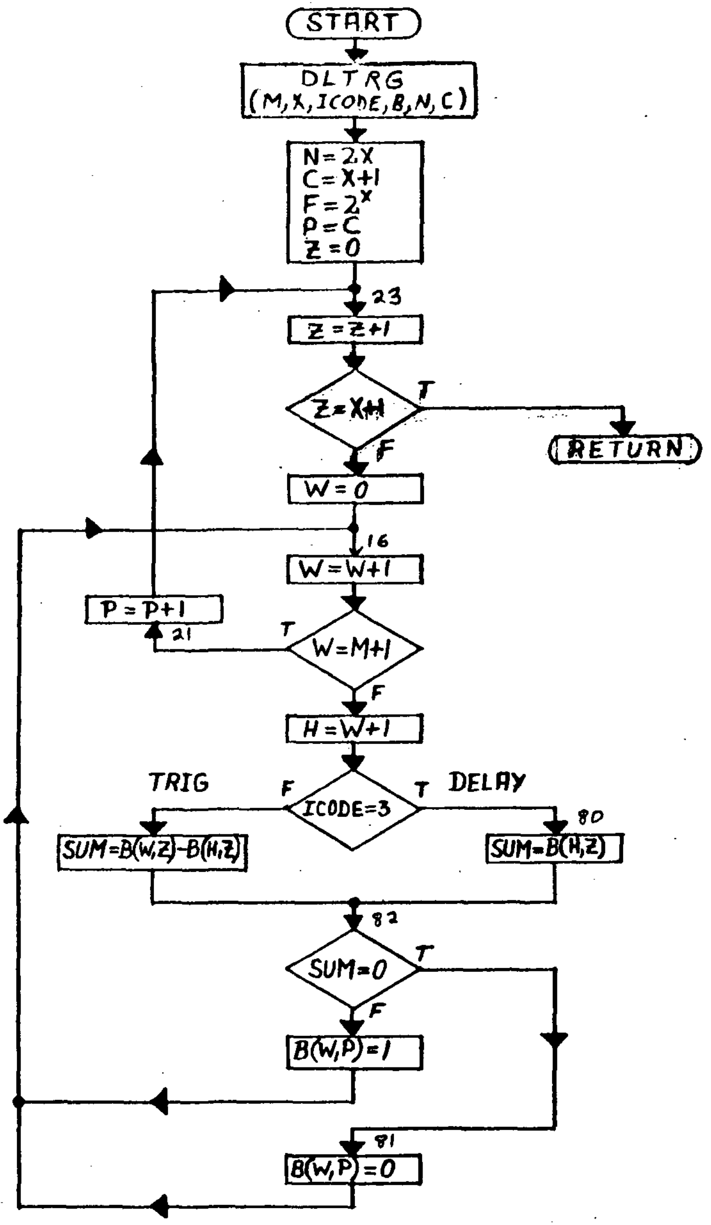


Figure 5.11. Flow Chart of Subroutine DLTRG

```
      SUBROUTINE RST    (M,X,ICODE,B,N,C)
      INTEGER C,X,F,P,Z,W,H,SUM,T,TT,B
      DIMENSION B(257,1)
      N=4*X
      C=X+1
      F=2**X
      P=C
      Z=0
   23 Z=Z+1
      IF(Z .EQ. X + 1) RETURN
      W=0
      T=P+1
      TT=T+1
   16 W=W+1
      IF(W.EQ.M+1) GO TO 21
      H=W+1
      SUM=B(W,Z)-B(H,Z)
      IF(SUM)18,19,20
   18 IF(ICODE.EQ.5) GO TO 80
      B(W,P)=4
      B(W,T)=0
      B(W,TT)=1
      GO TO 16
   80 B(W,P)=1
      B(W,T)=0
      B(W,TT)=4
      GO TO 16
   20 IF(ICODE.EQ.5) GO TO 81
      B(W,P)=0
      B(W,T)=4
      B(W,TT)=1
      GO TO 16
   81 B(W,P)=0
      B(W,T)=1
      B(W,TT)=4
      GO TO 16
   19 IF(B(W,Z).EQ.1) GO TO 17
      B(W,P)=0
      B(W,T)=4
      B(W,TT)=0
      GO TO 16
   17 B(W,P)=4
      B(W,T)=0
      B(W,TT)=0
      GO TO 16
   21 P=P+3
      GO TO 23
      END
```

Figure 5.12. Subroutine RST

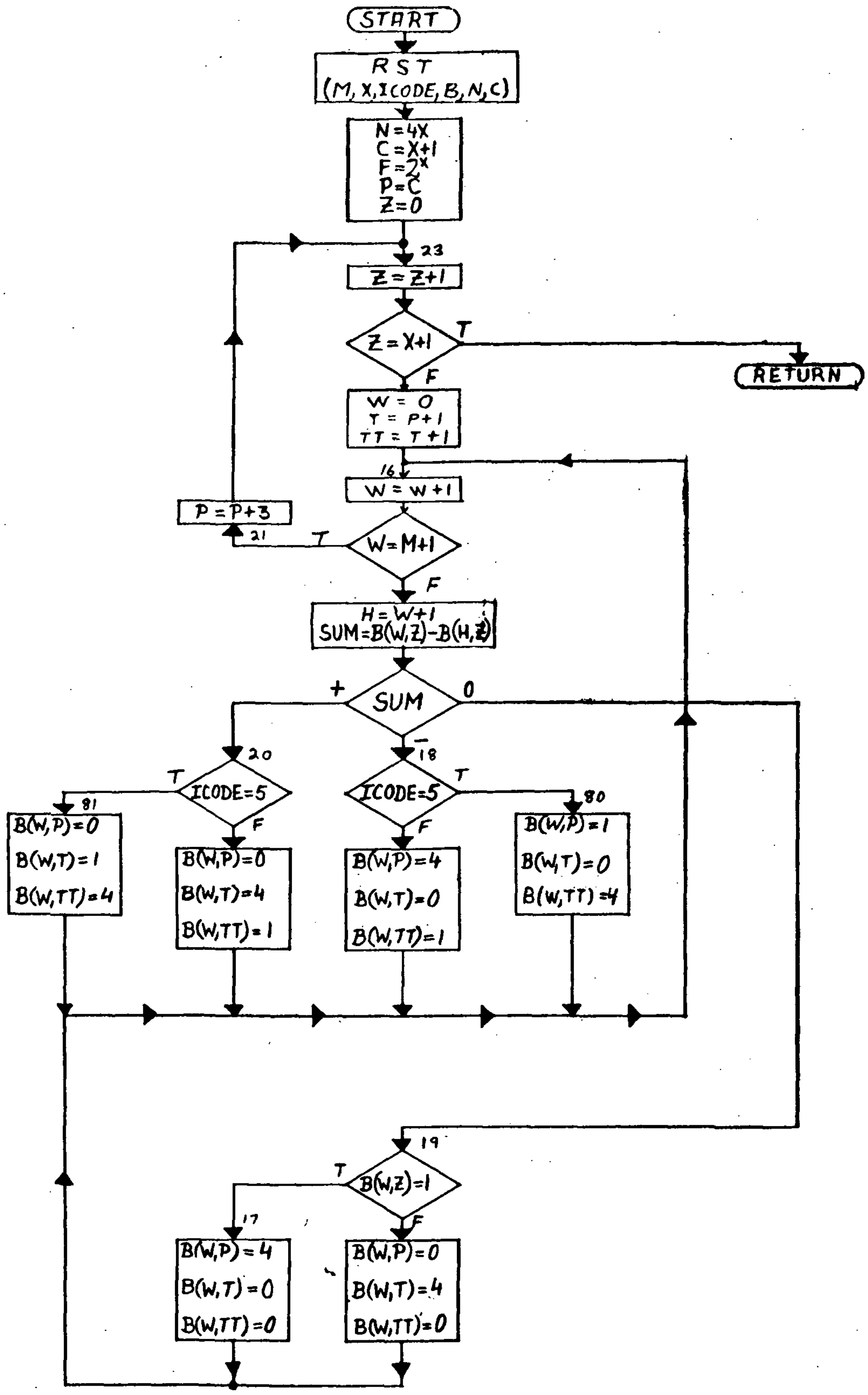


Figure 5.13. Flow Chart of Subroutine RST

```
      SUBROUTINE CTDC  (M,N,C,X,B)
      INTEGER B,C,G,O,Q,U,V,X
      DIMENSION B(257,1)
      G=M+1
      L=C
      DO 14 L=C,N
      B(G,L)=0
   14 CONTINUE
      U=G+1
      V = 2**X + 1
      DO 11 O=U,V
      DO 11 Q=C,N
      B(O,Q)=4
   11 CONTINUE
      RETURN
      END
```

Figure 5.14. Subroutine CTDC

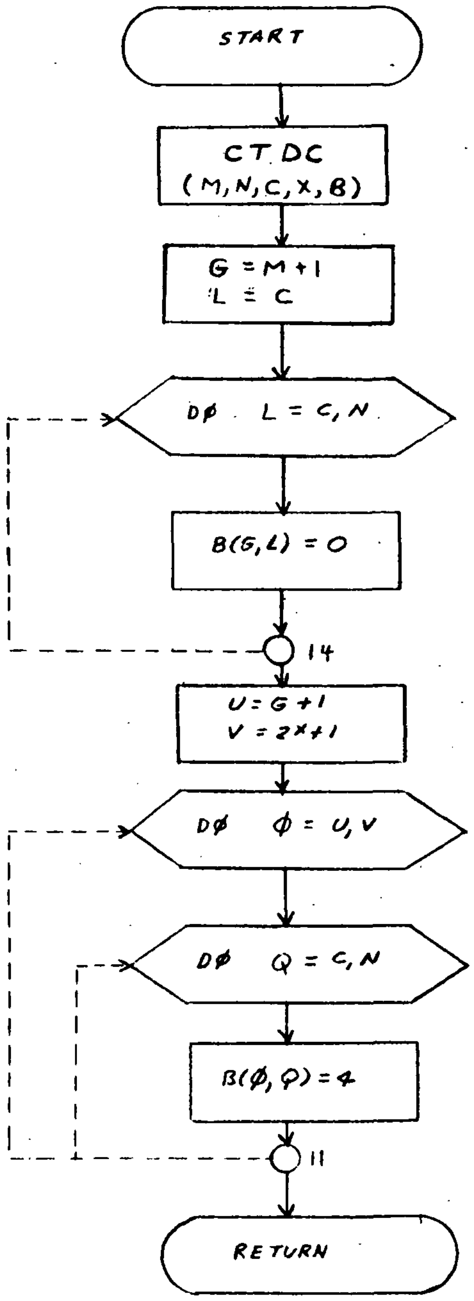


Figure 5.15. Flow Chart of Subroutine CTDC

```
      SUBROUTINE ARRAY (NBI, KK, NC, LN, MPS, A, N, M, AT, MIN)
      INTEGER A, AT, ROW, WK
      DIMENSION NBI(257,1),A(10,1),AT(10,1),ROW(100),WK(25)
      ITU = 1
      IV = 2**KK + 1
      L = 0
      IF (MPS) 201, 202, 203
  203 ITU = 0
  202 IF (LN .LE. 2**KK/6) GO TO 204
      DO 205 I = 1, LN
      IF (NBI(I, NC) - ITU )  205, 206, 205
  206 L = L + 1
      DO 207 K = 1, KK
  207 A(K, L) = NBI(I, K)
  205 CONTINUE
      M = L
      DO 208 I = 1, IV
      IF (NBI(I, NC) - 4 ) 208, 209, 208
  209 L = L + 1
      DO 211 K = 1, KK
  211 A(K, L) = NBI(I, K)
  208 CONTINUE
      N = L
      IR = N + 1
      DO 212 I = IR, IV
      DO 212 K = 1, KK
      A(K,I) = 3
  212 CONTINUE
      RETURN
  201 DO 213 I = 1, M
      DO 213 K = 1, KK
      AT(K, I) = A(K, I)
  213 CONTINUE
      MIN = 1
      MAX = M
      GO TO 214
  204 DO 215 I = 1, LN
      IF (NBI(I, NC) - ITU )  215, 220, 215
  220 L = L + 1
      DO 216 K = 1, KK
  216 AT(K,L) = NBI(I, K)
  215 CONTINUE
      MIN = L + 1
      IF(MIN .EQ. 1 .AND. MPS .EQ. 1) GO TO 180
      DO 217 I = 1, LN
      IF (NBI(I, NC) - (1 - ITU))  217, 221, 217
  221 L = L + 1
      DO 218 K = 1, KK
  218 AT (K, L) = 1 - NBI (I, K)
  217 CONTINUE
      MAX = L
  214 NRG = MAX - MIN + 1
      MW = MAX + 1
      LA = 0
      II = 1
```

Figure 5.16. Subroutine ARRAY (Sheet 1 of 3)

```
      DO 219 I = 1, KK
      DO 219 J = 1, 100
      A(I, J) = 3
  219 CONTINUE
      IF(MAX .LE. MIN + 1) GO TO 170
   34 IK = 1
      MRG = 0
      ITS = 0
   36 DO 44 K = IK,KK
      N1 = 0
      NZ = 0
      DO 40 I = MIN,MAX
      IF(AT(K,I) .EQ. 1) N1 = N1 + 1
      IF(AT(K,I) .EQ. 0) NZ = NZ + 1
   40 CONTINUE
      IF(N1 - NRG  ) 41,50,50
   41 IF(NZ - NRG  ) 43,52,52
   43 IF (N1 .GT. MRG) MRG = N1
      IF (NZ .GT. MRG) MRG = NZ
   44 CONTINUE
   45 IF(MRG) 46,46,145
  145 NRG = MRG
      GO TO 34
   46 IF(ITS) 34,70,34
   50 AT(K,MW) = 1
      IF(NZ .GT. MRG) MRG = NZ
      GO TO 54
   52 AT(K,MW) = 0
      IF(N1 .GT. MRG) MRG = N1
   54 ITS = 1
      L = MW
      DO 60 I = MIN,MAX
      IF(AT(K,I) .EQ. AT(K,MW)) GO TO 58
      L = L + 1
      NCOL = 0
      DO 56 J = 1,KK
      IF(AT(J,I) .EQ. 3) NCOL = NCOL + 1
      AT(J,L) = AT(J,I)
   56 CONTINUE
      IF(AT(K,L) .NE. 3 .AND. NCOL .EQ. KK - 1) ITS = 0
  164 IF(NCOL - KK) 57,70,57
   57 AT(K,L) = 3
      GO TO 60
   58 AT(K,I) = 3
   60 CONTINUE
      WK(II+2) = K
      IF(ITS) 168,166,168
  166 ITS = 1
      GO TO 72
  168 WK(II+4) = MAX
      IF(L - MW - 1) 66,76,64
   64 IF(II .GE. 21)GO TO 76
      WK(II) = MIN
      WK(II+1) = MRG
      WK(II+3) = NRG
      II = II + 5
```

Figure 5.16. Subroutine ARRAY (Sheet 2 of 3)

```
      MIN = MW + 1
      MAX = L
      NRG = L - MW
      MW = L + 1
      GO TO 34
   66 LA = LA + 1
      DO 68 IW = 1,II,5
      KW = WK(IW+2)
      MW = WK(IW+4) +1
      A(KW,LA) = AT(KW,MW)
   68 CONTINUE
      GO TO 72
   70 IF(II .EQ. 1) GO TO 200
      II = II - 5
      MIN = WK(II)
      MRG = WK(II +1)
      NRG = WK(II+3)
      MAX = WK(II+4)
      MW = MAX + 1
   72 IK = WK(II+2) + 1
      IF (IK - KK) 36, 36, 45
   76 MMW = MW + 1
      IM = L - MW
      DO 78 LW = 1,IM
   78 ROW(LW) = 1
      LST = LA + 1
   80 LA = LA + 1
      DO 88 I = MMW,L
      IW = I - MW
      IR = ROW(IW)
      DO 84 J = IR,KK
      IF(AT(J,I) - 3) 82,84,82
   82 IF(AT(J,I) + A(J,LA) - 1) 86,84,86
   84 CONTINUE
      NL = I - MW
      LA = LA - 1
      GO TO 92
   86 ROW(IW) = J
      A(J,LA) = AT(J,I)
   88 CONTINUE
      NL = 0
   90 NL = NL + 1
      IF(ROW(NL) .EQ. KK) GO TO 92
      ROW(NL) = ROW(NL) + 1
      GO TO 80
   92 IF(NL .EQ. IM) GO TO 94
      ROW(NL) = 1
      GO TO 90
   94 DO 96 J = 1,KK
   96 A(J,LA+1) = 3
      IF(LST - LA) 97,97,72
   97 DO 98 IW = 1,II,5
      KW = WK(IW+2)
      MW = WK(IW+4) + 1
      DO 95 LW = LST,LA
      A(KW,LW) = AT(KW,MW)
   95 CONTINUE
   98 CONTINUE
      GO TO 72
  170 LA = 1
  200 N = LA
      M = LA
  180 IF(MIN .LE. 1 .AND. MPS .NE. -1) M = 0
      RETURN
      END
```

Figure 5.16. Subroutine ARRAY (Sheet 3 of 3)

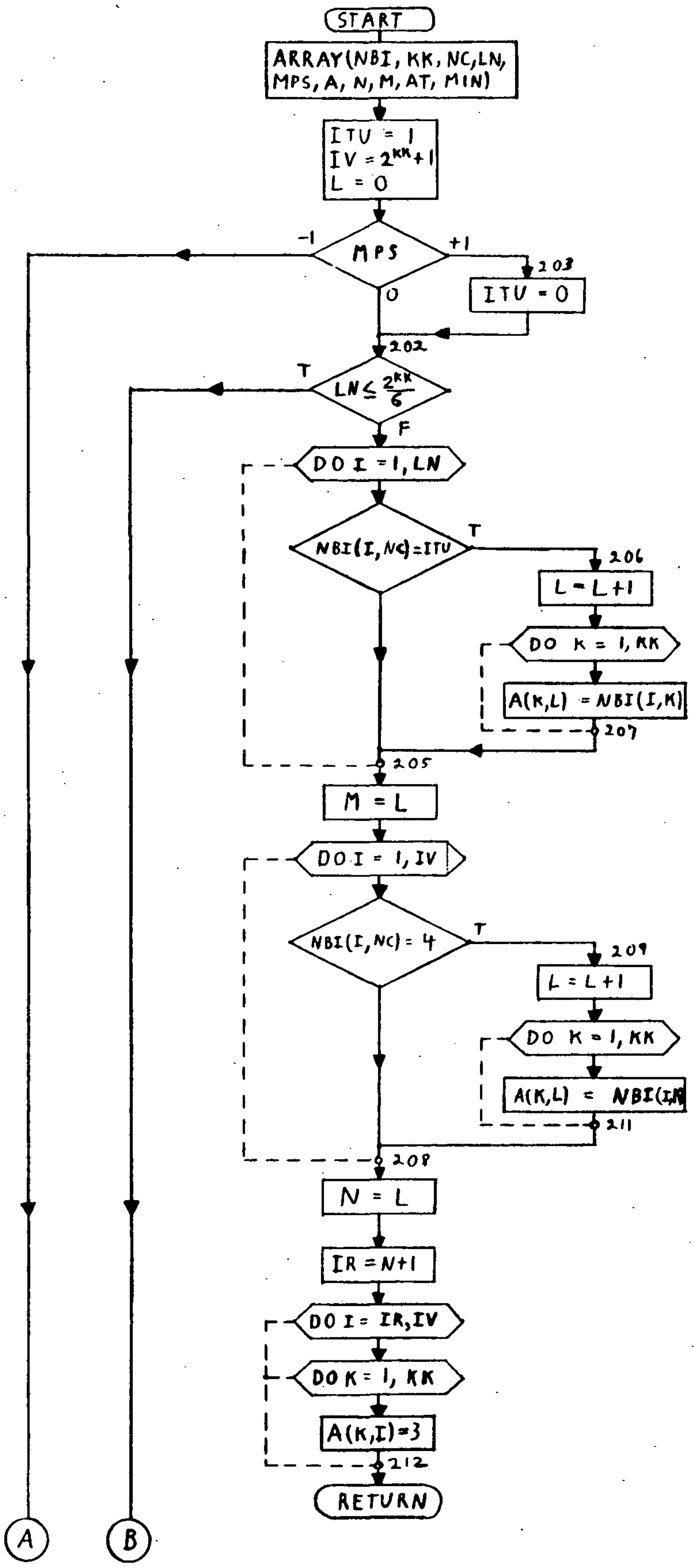


Figure 5.17. Flow Chart of Subroutine ARRAY (Sheet 1 of 5)

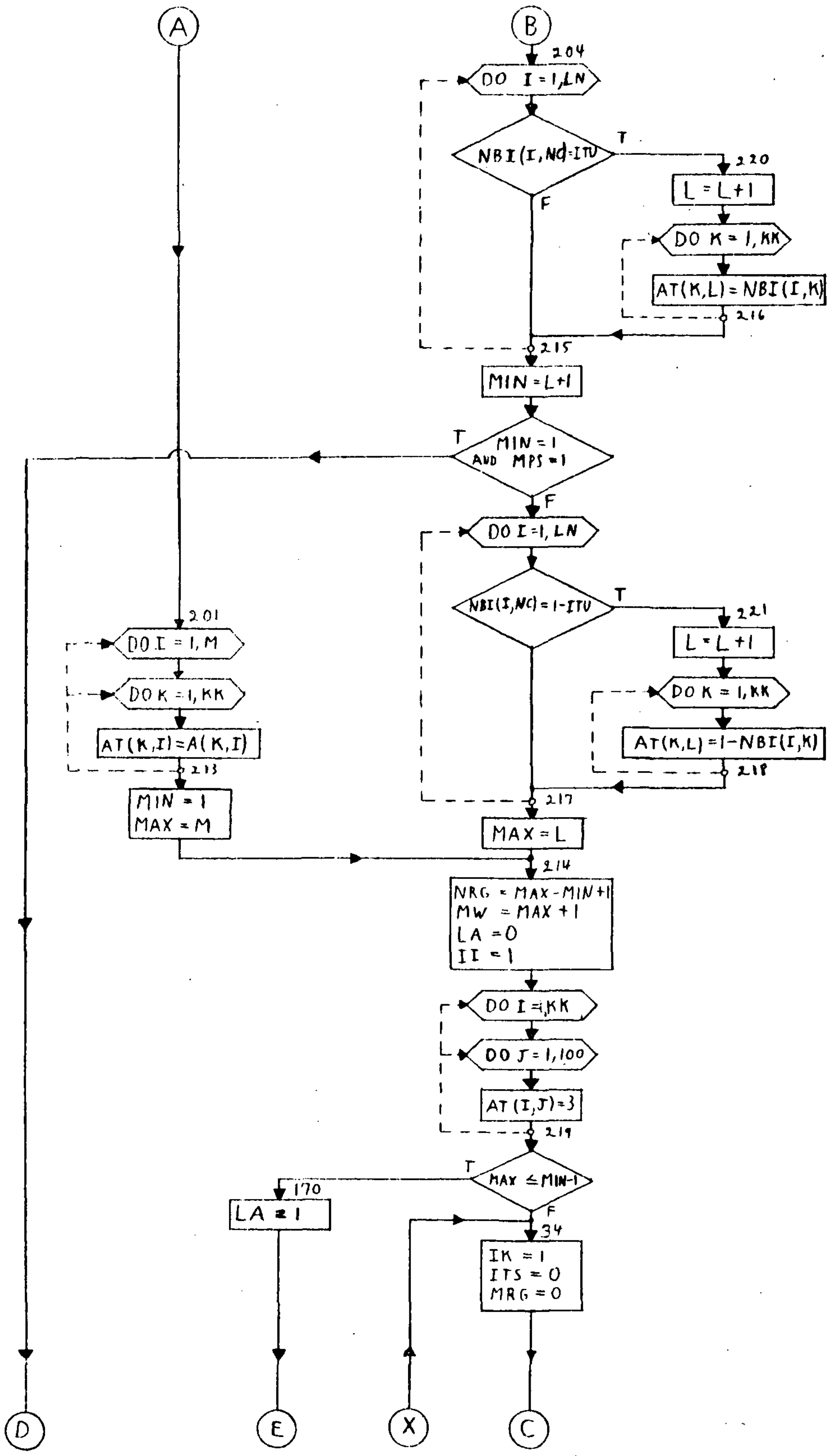


Figure 5.17. Flow Chart of Subroutine ARRAY (Sheet 2 of 5)

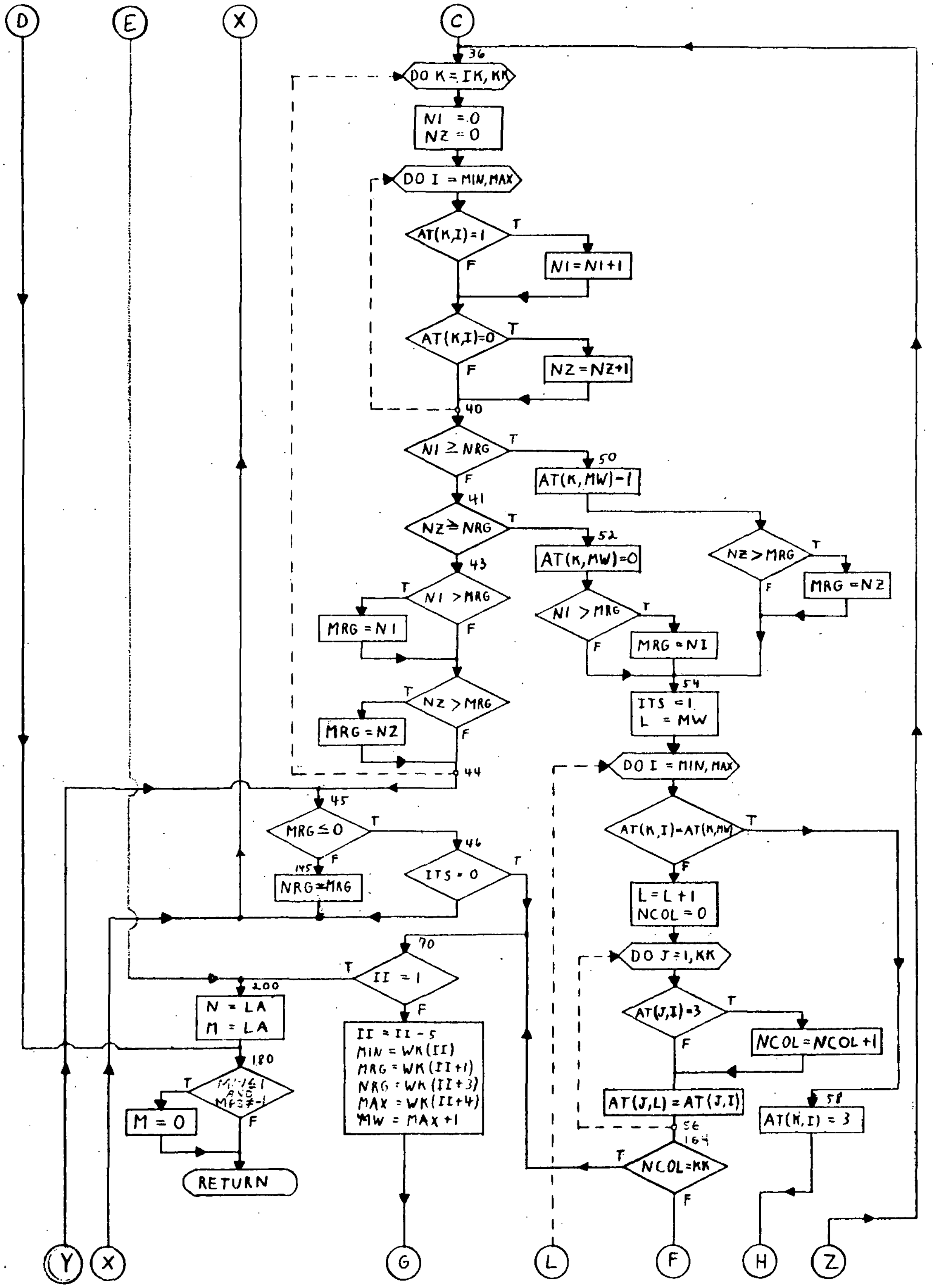


Figure 5.17. Flow Chart of Subroutine ARRAY (Sheet 3 of 5)

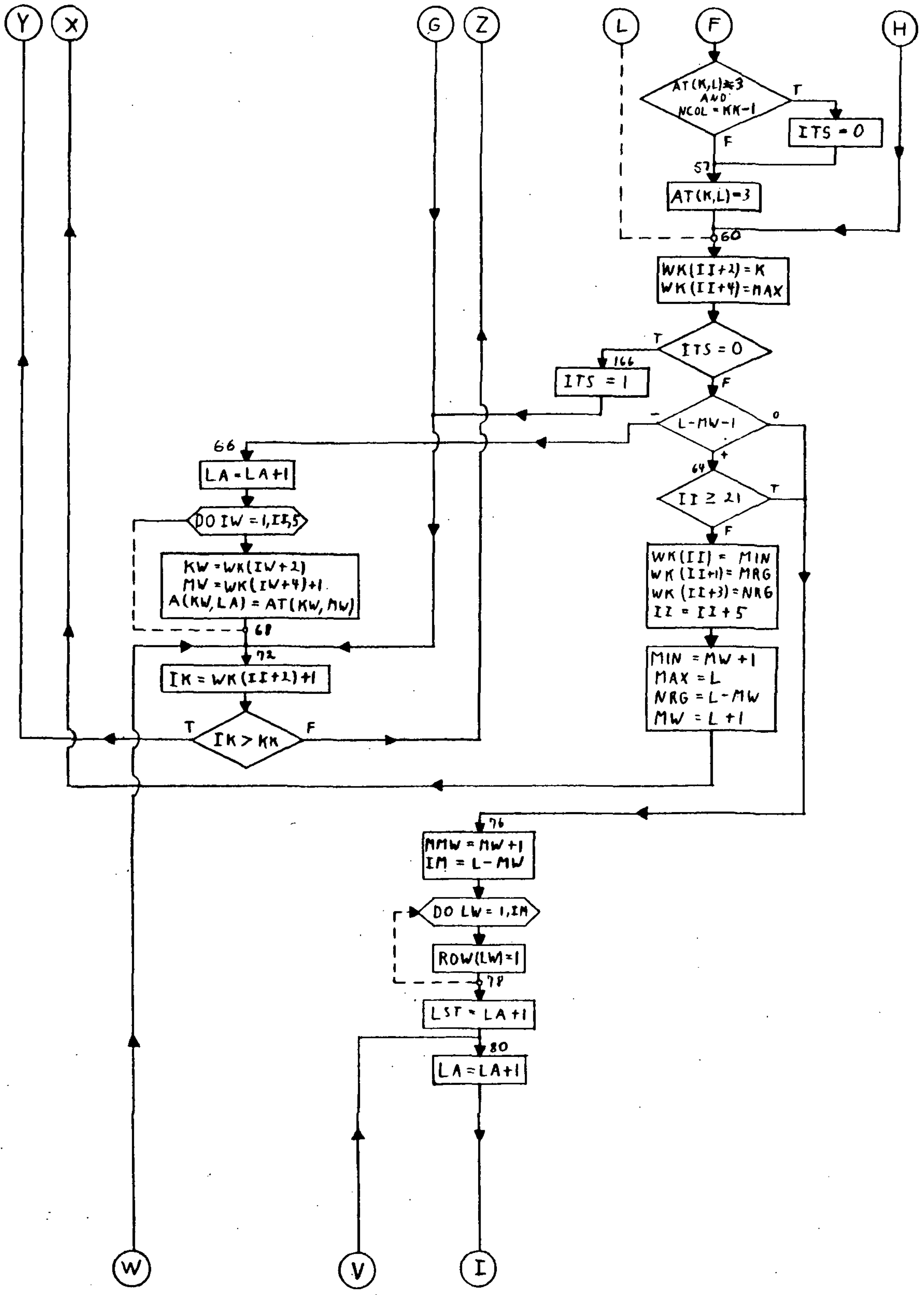


Figure 5.17. Flow Chart of Subroutine ARRAY (Sheet 4 of 5)

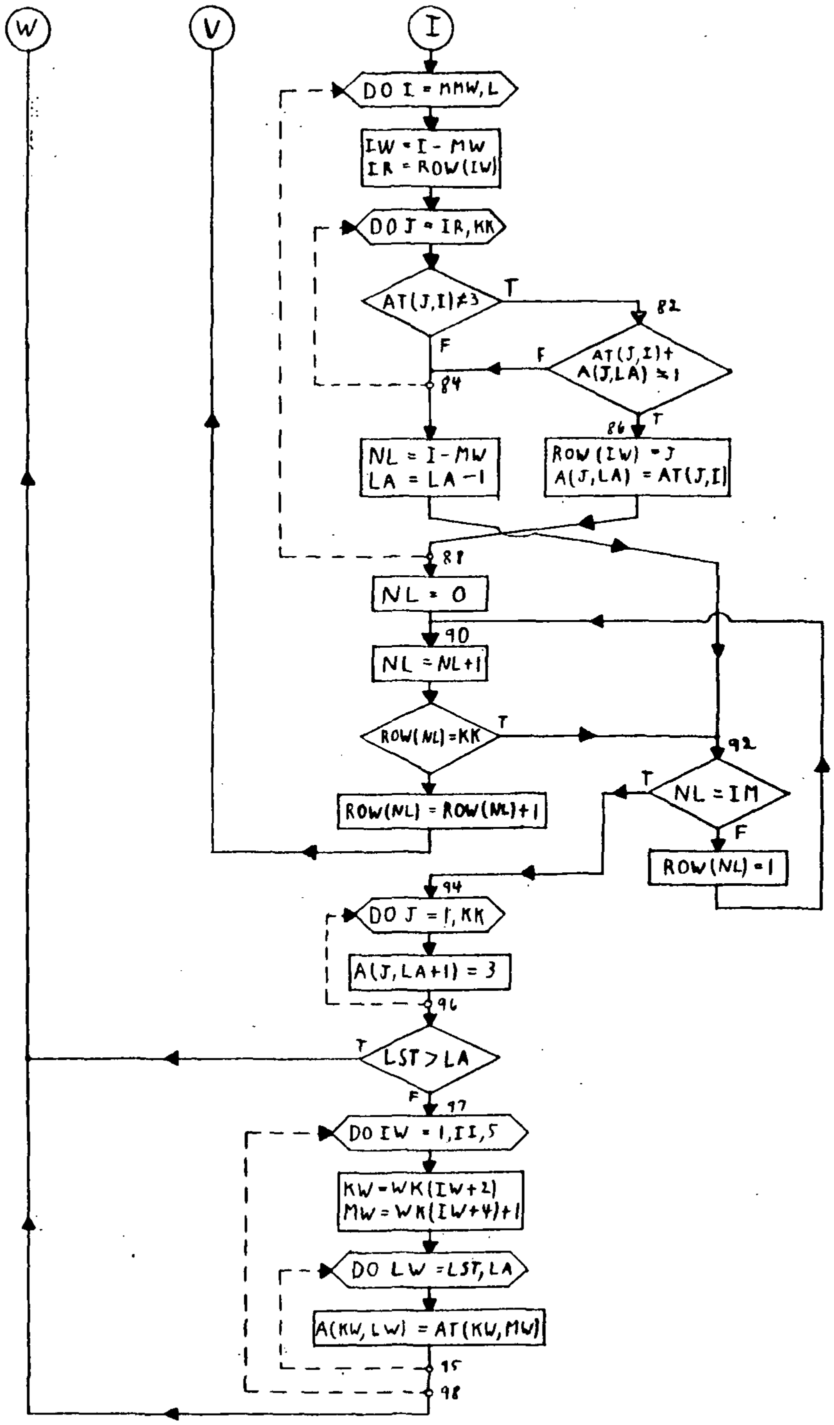


Figure 5.17. Flow Chart of Subroutine ARRAY (Sheet 5 of 5)

```
      SUBROUTINE DETRU  (KK, MIN, AT, NRT, A, N, M)
      INTEGER A, AT, ROW
      DIMENSION AT(10,1),IA(28,92),ROW(100),A(10,257)
      COMMON IA
      IF(M .LE. 1) RETURN
      L = MIN
      DO 22 I = 1, N
      ROW(L) = 0
      DO 20 K = 1, KK
      AT(K,L) = A(K,I)
      IF(A(K,I) .NE. 3) ROW(L) = ROW(L) + 1
   20 CONTINUE
      IF(ROW(L) .NE. 0) L = L + 1
      IF(I .EQ. M) LM = L - 1
   22 CONTINUE
      L = L - 1
      MI = MIN - 1
      JW = 1
      DO 24 I = 1, MI
   24 IA(I,1) = 0
      DO 36 J = MIN,LM
      JW = JW + 1
      NCOL = 0
      DO 34 I = 1, MI
      DO 30 K = 1, KK
      IF(AT(K,I) + AT(K,J) .EQ. 1)GO TO 32
   30 CONTINUE
      IA(I,JW) = J
      NCOL = NCOL + 1
      IA(I,1) = IA(I,1) + 1
      GO TO 34
   32 IA(I,JW) = 0
   34 CONTINUE
      IF(NCOL .EQ. 0) GO TO 35
      IA(MIN,JW) = NCOL
      GO TO 36
   35 JW = JW - 1
   36 CONTINUE
      IF (JW .GT. 92 .OR. MIN .GT. 28) WRITE(6,38)
   38 FORMAT (1H0,'THE FOLLOWING BOOLEAN FUNCTION MAY BE IN ERROR'/
     1    1X,'INCREASE DIMENSIONS OF IA(-,-) IN SUBROUTINE DETRU')
   39 LR = 0
      MROW = 90
      DO 42 I = 1, MI
      IF(IA(I, 1)  - MROW) 40, 42, 42
   40 MROW = IA(I,1)
      LR = I
   42 CONTINUE
      IF(LR .EQ. 0) GO TO 50
      NCOL = 0
      LC = 0
      ITS = 12
      DO 46 J = 2, JW
      NL = IA(LR,J)
      IF(IA(LR,J) .GE. 1 .AND. IA(MIN,J) .GT. NCOL) GO TO 44
```

Figure 5.18. Subroutine DETRU (Sheet 1 of 2)

```
      IF(IA(LR,J).GE.1.AND.IA(MIN,J).EQ.NCOL.AND.ROW(NL).LT.ITS)GO TO 44
      GO TO 46
   44 LC = J
      NCOL = IA(MIN,J)
      ITS = ROW(NL)
   46 CONTINUE
      NL = IA(LR,LC)
      ROW(NL) = 20
      DO 48 I = 1, MI
      IF(IA(I,LC) .GE. 1) IA(I, 1) = 90
   48 CONTINUE
      GO TO 39
   50 LA = 0
      DO 56 I = MIN,LM
      IF(ROW(I) - 20) 56, 52, 56
   52 LA = LA + 1
      DO 54 K = 1, KK
   54 A(K,LA) = AT(K,I)
   56 CONTINUE
      M = LA
      IF(NRT .EQ. 1) GO TO 64
      DO 62 I = MIN,L
      IF(ROW(I) .EQ. 20) GO TO 62
      LA = LA + 1
      DO 60 K = 1, KK
   60 A(K,LA) = AT(K,I)
   62 CONTINUE
   64 N = LA
      RETURN
      END
```

Figure 5.18. Subroutine DETRU (Sheet 2 of 2)

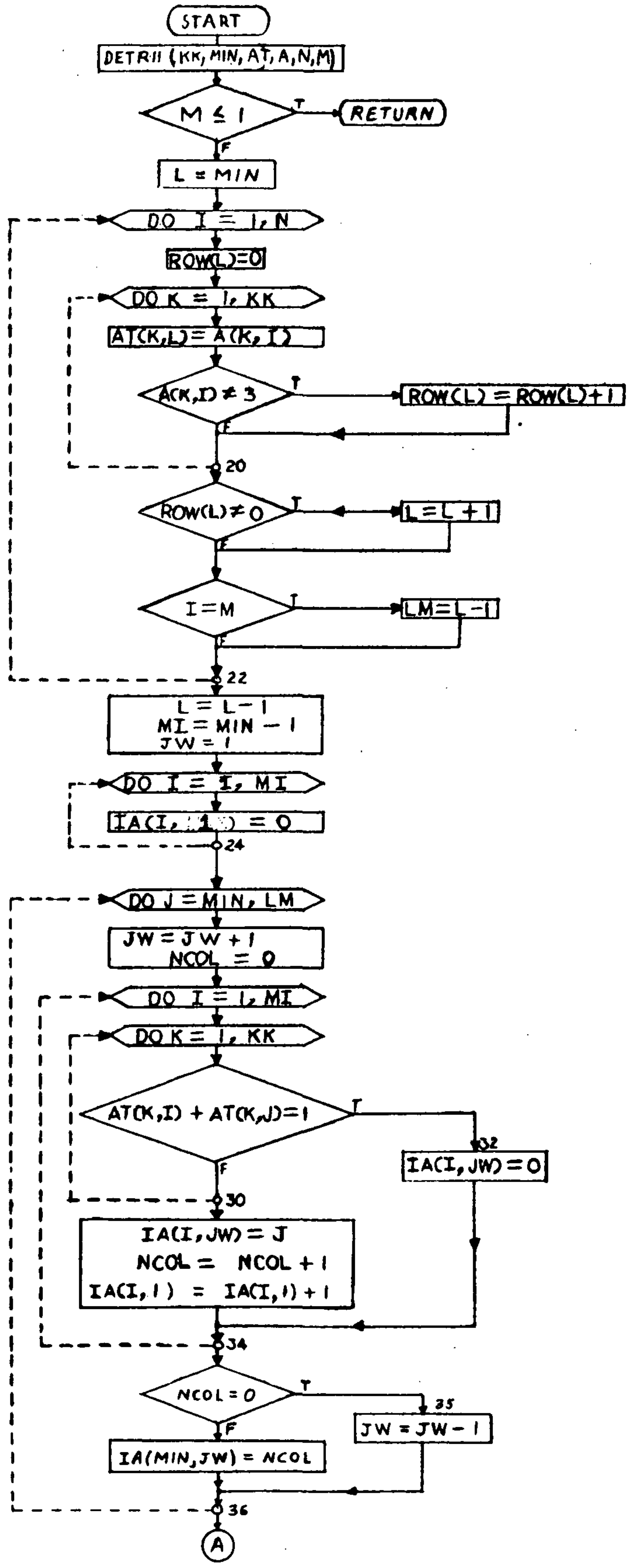


Figure 5.19. Flow Chart of Subroutine DETRU (Sheet 1 of 3)

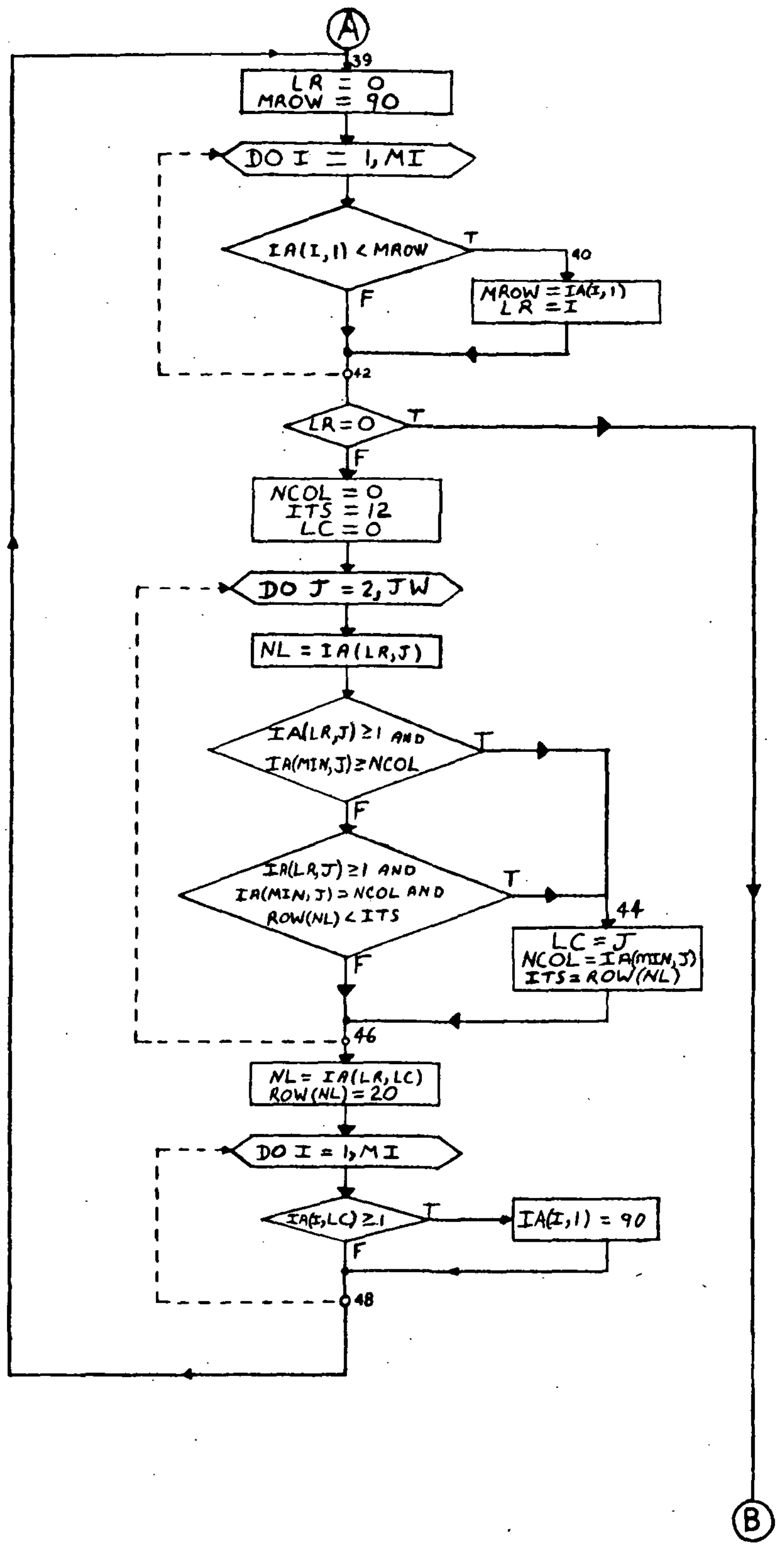


Figure 5.19. Flow Chart of Subroutine DETRU (Sheet 2 of 3)

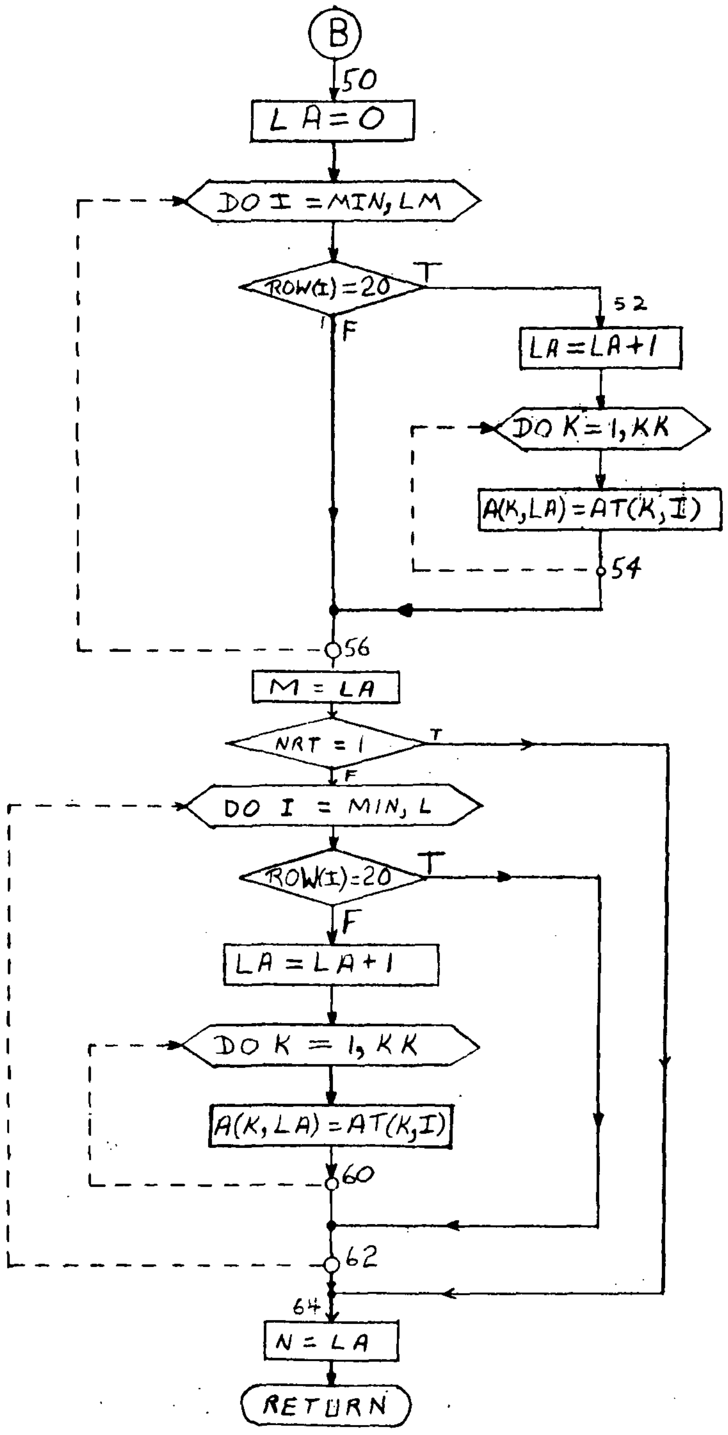


Figure 5.19. Flow Chart of Subroutine DETRU (Sheet 3 of 3)

```
      SUBROUTINE BOOL (R,X,S,NRT,A)
      INTEGER A,C,F,G,S,R,T,U,V,W,X,Y,Z,SUM,RT,RR,XX
      DIMENSION A(10,1)
      IF(R .LE. 1) RETURN
      IP=0
      I=0
      IF(S .EQ. 1 .OR. S .EQ. R) GO TO 940
      IR = S
      GO TO 818
  940 IP = 1
      IR = R
  818 I=I+1
      IF(I - X - 1) 254,251,254
  251 IP=IP+1
      IF(IP .EQ. 2) GO TO 502
      IF(IP - 3) 514,120,514
  502 JJ = 1
      DO 510 J = 2,R
      DO 504 I = 1,X
      IF(A(I,J) .NE. 3) GO TO 506
  504 CONTINUE
      GO TO 509
  506 JJ = JJ + 1
      DO 508 II = 1,X
  508 A(II,JJ) = A(II,J)
  509 IF(J .EQ. S) S = JJ
  510 CONTINUE
      R = JJ
  514 IR = R
      I = 1
  254 J = 0
  817 J=J+1
      IF(J.GE.IR) GO TO 818
      IF(A(I,J).EQ.3) GO TO 817
      K = J + 1
  809 SUM=A(I,J) + A(I,K)
      IF(SUM-1)808,807,808
  808 K=K+1
      IF(K.EQ.IR+1)GO TO 817
      GO TO 809
  807 C=R+1
      DO 810 L=1,X
      A(L,C) = A(L,J)
  810 CONTINUE
  811 A(I,C) = 3
      RR=0
  812 RR=RR+1
      IF(RR.EQ.X+1) GO TO 814
      SUM = A(RR,K) + A(RR,C)
      IF(SUM-1) 812,813,812
  813 DO 819 T=1,X
      A(T,C) =3
  819 CONTINUE
      GO TO 808
  814 XX=0
```

Figure 5.20. Subroutine BOOL (Sheet 1 of 3)

```
815 XX=XX+1
    IF(XX.EQ.X+1) GO TO 816
    IF(A(XX,C).LT.A(XX,K)) GO TO 815
    A(XX,C) = A(XX,K)
    GO TO 815
816 A(I,C) = 3
    W = 0
800 W= W+1
    IF(W.EQ.IR+1)GO TO 808
    Z = 1
910 IF(Z.EQ.X+1) GO TO 908
    SUM=A(Z,C) + A(Z,W)
    IF(SUM-1)909,800,909
909 Z=Z+1
    GO TO 910
908 RT=1
905 IF(RT.EQ.X+1) GO TO 906
    SUM=A(RT,W) - A(RT,C)
    IF(SUM) 907,907,800
907 RT=RT+1
    GO TO 905
906 DO 6 U=1,X
    A(U,W) = A(U,C)
  6 CONTINUE
904 Y=W
860 Y=Y+1
    IF(Y.EQ.C+1) GO TO 808
    V=1
890 IF(V.EQ.X+1) GO TO 900
    SUM=A(V,W) + A(V,Y)
    IF(SUM-1) 880,860,880
880 V=V+1
    GO TO 890
900 F=1
902 IF(F.EQ.X+1) GO TO 870
    SUM=A(F,Y) - A(F,W)
    IF(SUM) 901,901,860
901 F=F+1
    GO TO 902
870 DO 7 G=1,X
    A(G,Y) = 3
  7 CONTINUE
    GO TO  860
120 J = 0
    MK = S
602 J = J + 1
    NTH = 0
    IF(J - MK - 1) 606,604,606
604 IF(NRT .EQ. 1 .OR. MK .EQ. R) RETURN
    MK = R
606 DO 608 I = 1,X
    IF(A(I,J) .EQ. 3) NTH = NTH + 1
608 CONTINUE
    IF(NTH -X +1) 610,602,602
610 NCL = 2**NTH
    JT = R + NCL
```

Figure 5.20. Subroutine BOOL (Sheet 2 of 3)

```
      MC = R + 1
      MT = -1
      DO 620 I = 1,X
      IF(A(I,J) .EQ. 3) GO TO 616
      DO 614 JJ = MC,JT
  614 A(I,JJ)= A(I,J)
      GO TO 620
  616 MT = MT + 1
      IOZ = 0
      MOD = 2**MT
      DO 618 JJ = MC,JT
      A(I,JJ) = IOZ
      JM = JJ - R
      IF(JM - (JM/MOD)*MOD .EQ. 0) IOZ = 1 - IOZ
  618 CONTINUE
  620 CONTINUE
      DO 650 JJ = MC,JT
      DO 640 K = 1,MK
      IF(K .EQ. J) GO TO 640
      ITS = 0
      DO 630 I = 1,X
      IF(A(I,K) .EQ. 3) GO TO 630
      IF(A(I,K) - A(I,JJ) .NE. 0) GO TO 640
      ITS = 1
  630 CONTINUE
      IF(ITS .EQ. 1) GO TO 650
  640 CONTINUE
      GO TO 602
  650 CONTINUE
      DO 654 I = 1,X
  654 A(I,J) = 3
      GO TO 602
      END
```

Figure 5.20. Subroutine BOOL (Sheet 3 of 3)

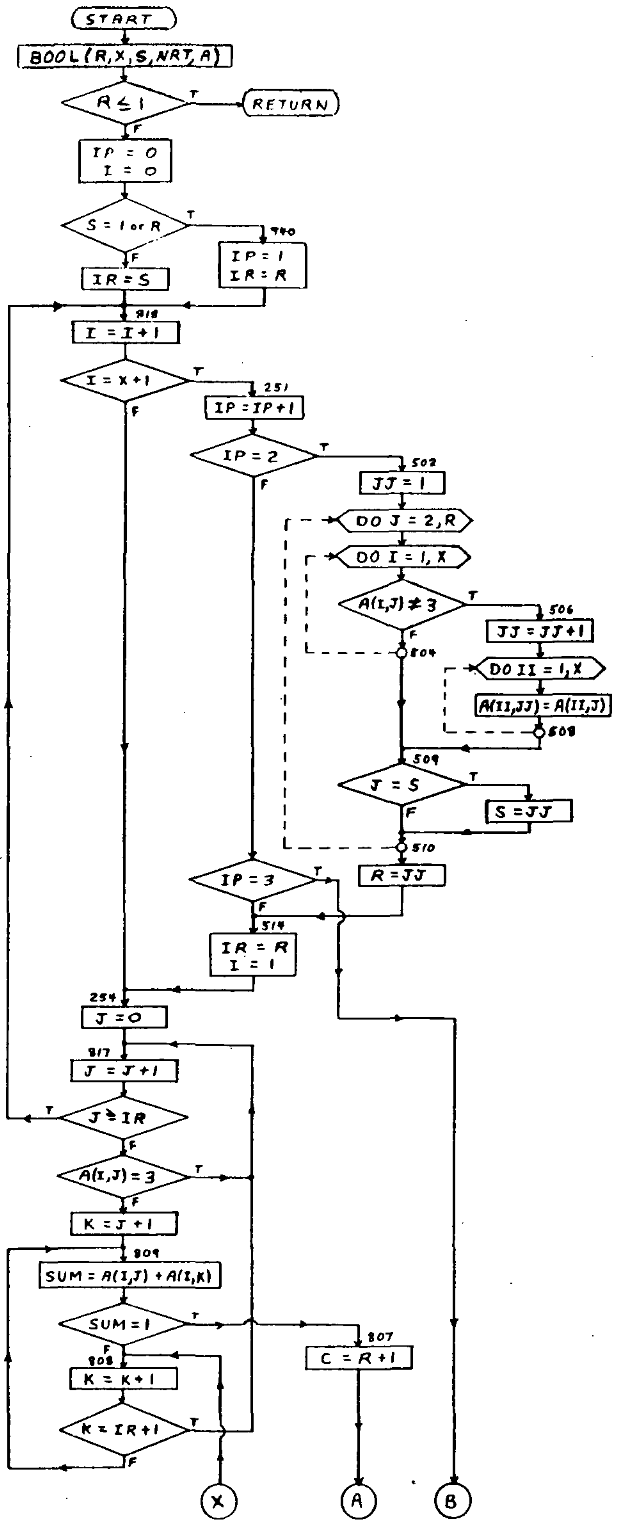


Figure 5.21. Flow Chart of Subroutine BOOL (Sheet 1 of 4)

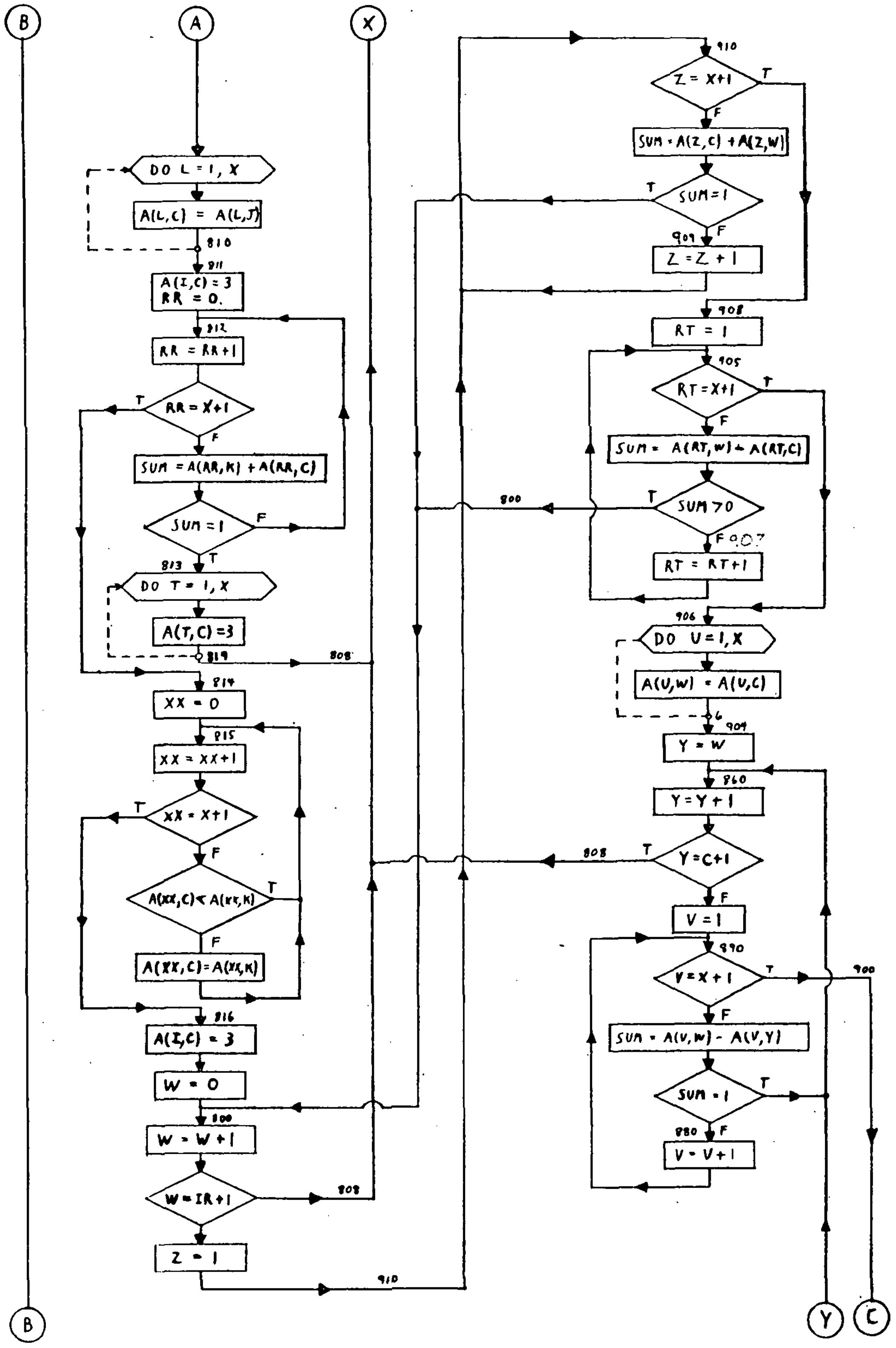


Figure 5.21. Flow Chart of Subroutine BOOL (Sheet 2 of 4)

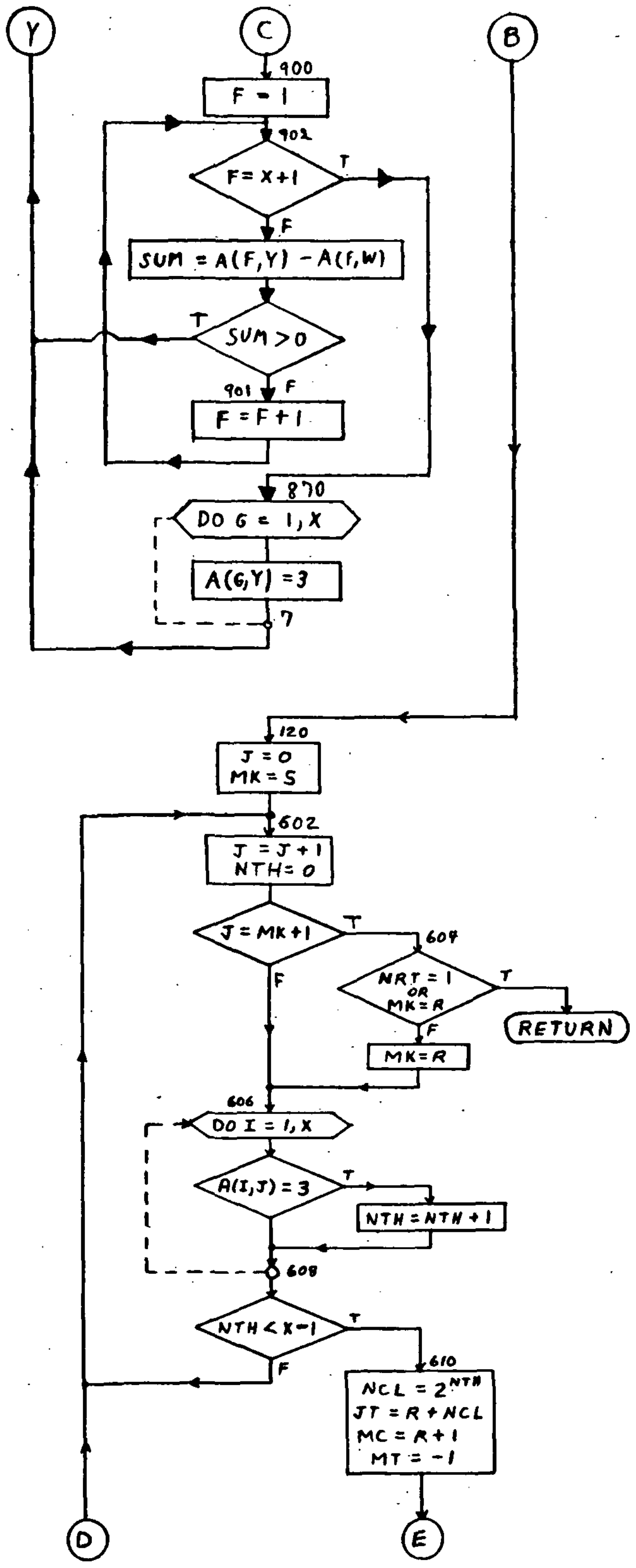


Figure 5.21. Flow Chart of Subroutine BOOL (Sheet 3 of 4)

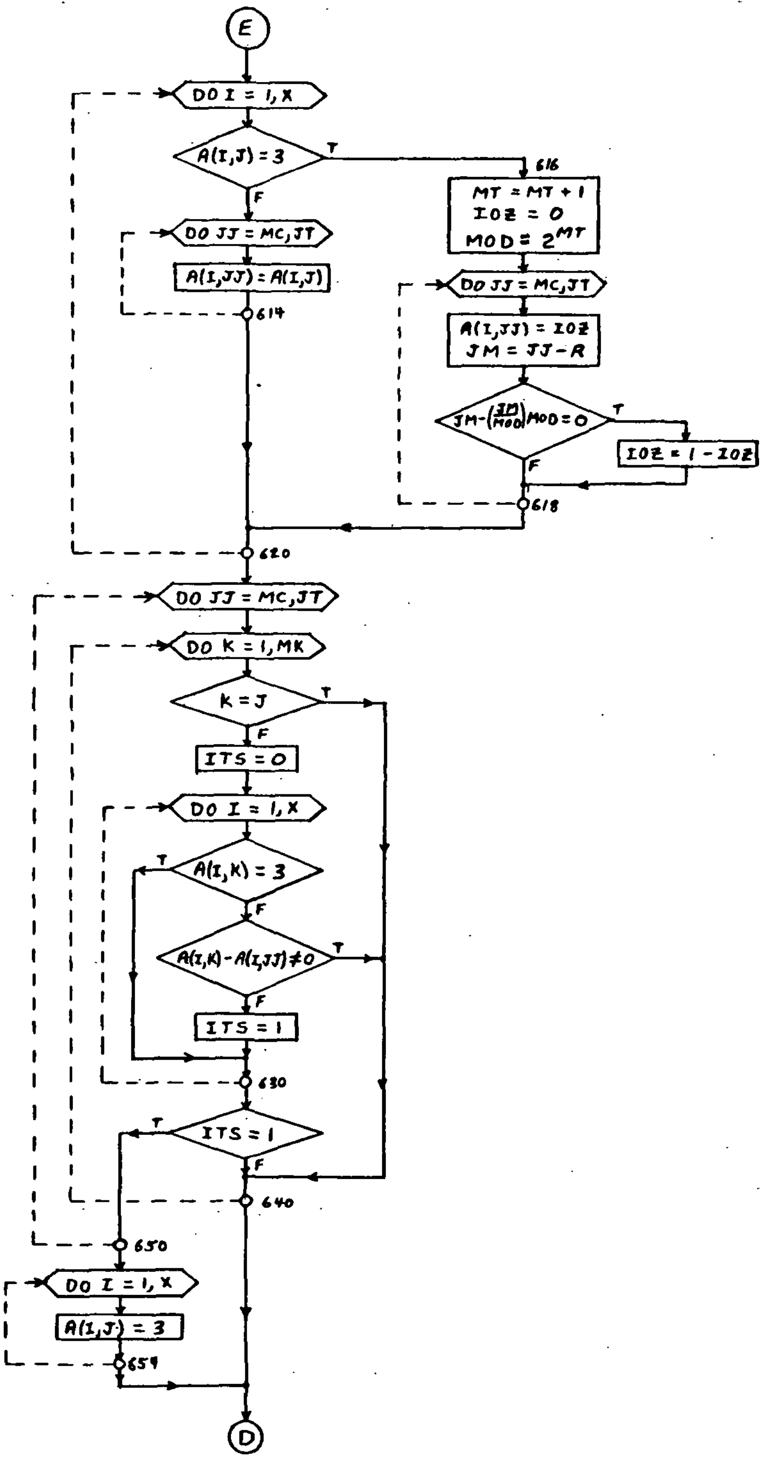


Figure 5.21. Flow Chart of Subroutine BOOL (Sheet 4 of 4)

```
      SUBROUTINE DGLOT (A,LETR,KK,N,M,MODE,FCODE,NC,NO,NBI,NH,NRT,MPS)
      INTEGER A,DATA,BLANK,BAR,PLUS,SLH,ONE,FCODE,FIN,C,D,F,R,S,T,ROW
      INTEGER RPAR
      DIMENSION A(10,1),LETR(1),DATA(110),FIN(3)
      DIMENSION ROW(120),NO(1),NBI(257,1),IRW(75)
      DATA LPAR,RPAR/1H(,1H)/
      DATA BLANK,BAR,PLUS,SLH,ONE/1H ,1H',1H+,1H/,1H1/,IDSH/1H-/
      DATA C,D,F,J,K,R,S,T/1HC,1HD,1HF,1HJ,1HK,1HR,1HS,1HT/,IZRO/1HO/
      DATA (IRW(I),I =1,72)/1HO,1H-,1H ,1H1,1H-,1H ,1H-,1H1,1H ,1H-,1HO,
     1  1H ,1HO,1H-,1H ,1H1,1HO,1H ,1HO,1H1,1H ,1H-,1HO,1H ,
     2  1HO,1H ,1H ,1H1,1H ,1H ,1HO,1H ,1H ,1H1,1H ,1H ,
     3  1HO,1H ,1H ,1H1,1H ,1H ,1H1,1H ,1H ,1HO,1H ,1H ,
     4  1HO,1H-,1HO,1H1,1HO,1H-,1HO,1H1,1H-,1H-,1HO,1HO,
     5  1HO,1H-,1HO,1H-,1HO,1H1,1HO,1H-,1H1,1H-,1HO,1HO/
      IN = 5
      IO = 6
      IF(MODE .EQ. F) GO TO 52
      IF(MODE .EQ. C) GO TO 10
      GO TO 99
   10 IF(NC-KK .EQ. 1) GO TO 12
/     GO TO 44
   12 NIN = 2
      I1 = 1
      I2 = 2
      I3 = 3
      FIN(3) = BLANK
      IF(FCODE .EQ. 1) GO TO 14
      IF(FCODE .EQ. 5 .OR. FCODE .EQ. 6) GO TO 16
   18 IF(FCODE .EQ. 2 .OR. FCODE .EQ. 5 .OR. FCODE .EQ. 6) GO TO 20
      NIN = 1
      FIN(2) = BLANK
      IF(FCODE  .EQ. 4) GO TO 22
      IF(FCODE .EQ. 3) GO TO 24
      GO TO 28
   14 FIN(1) = J
      FIN(2) = K
      GO TO 28
   16 FIN(3) = T
      NIN = 3
      GO TO 18
   20 FIN(1) = S
      FIN(2) = R
      I1 = 2
      I2 = 1
      GO TO 28
   22 FIN(1) = T
      GO TO 26
   24 FIN(1) = D
   26 I3 = 2
   28 NOF = 1
      NI = 1
      IF(NRT .EQ. 1) GO TO 31
      J1 = I2*FCODE - I1
      J2 = J1 + I1
      WRITE(IO,102) FIN(I1),FIN(I2),FIN(I3)
```

Figure 5.22. Subroutine DGLOT (Sheet 1 of 4)

```
102 FORMAT (1H126X3(A1,1X),'FLIP-FLOP TRANSITION INPUT REQUIREMENTS',
    1   ///25X'0 ----> 0     0 ----> 1     1 ----> 0     1 ----> 1')
      WRITE(IO,104) ((FIN(I),I =1,3),N =1,4),(IRW(I),I = J1,J2)
104 FORMAT (27X,4(A1,1XA1,1XA1,8X)/27X,4(A1,1XA1,1XA1,8X)///
    1   25X,43H'1' = TRUE,  '0' = FALSE,  '-' = DON'T CARE//)
      WRITE(IO,110) FIN(I1),FIN(I2),FIN(I3)
110 FORMAT (1H0,26X,'TRUTH TABLE FOR COUNTER USING'3(1X,A1),
    1   ' FLIP-FLOPS'/)
      IL = 0
      DO 112 L = 1,KK
      ROW(IL+1) = LETR(L)
      ROW(IL+2) = BLANK
      IL = IL + 2
112 CONTINUE
      DO 114 I = 1,2
      IL = IL + 1
114 ROW(IL) = BLANK
      DO 118 L = 1,KK
      DO 116 I = 1,NIN
      ROW(IL+1) = FIN(I)
      ROW(IL+2) = LETR(L)
      ROW(IL+3) = BLANK
      IL = IL + 3
116 CONTINUE
118 CONTINUE
      WRITE(IO,120) (ROW(I),I = 1,IL)
120 FORMAT (1H0,10X'NO'5X,130A1)
      NLG = 2**KK + 1
      DO 140 I = 1,NLG
      IL = 0
      DO 130 L = 1,NH
      ROW(IL+1) = IDSH
      IF(NBI(I,L) .EQ. 1) ROW(IL+1) = ONE
      IF(NBI(I,L) .EQ. 0) ROW(IL+1) = IZRO
      ROW(IL+2) = BLANK
      IL = IL + 2
      IF(L .EQ. KK) GO TO 122
      GO TO 126
122 DO 124 II = 1,2
      IL = IL + 1
124 ROW(IL) = BLANK
126 IF(L .GE. KK) GO TO 128
      GO TO 130
128 IL = IL + 1
      ROW(IL) = BLANK
130 CONTINUE
      WRITE(IO,134) NO(I),(ROW(L),L = 1,IL)
134 FORMAT (1H0,8XI4,5X130A1)
140 CONTINUE
 31 WRITE (IO,32) FIN(I1),FIN(I2),FIN(I3)
 32 FORMAT (1H1,31X,A1,1X,A1,1X,A1,' FLIP-FLOP INPUT EQUATIONS'/
    1    38X'(F = TRUE TERMS /'18H DON'T CARE TERMS)//
    2    1H037X'CLOCK ---,'/47X':'/1H+,46X'V'/
    3    40X'..............'/40X':'12X
    4  ':'/40X':'9X'Q   :---->')
      IF(FIN(I) .NE. BLANK .AND. I3 .NE. 2) WRITE(IO,34) FIN(I)
```

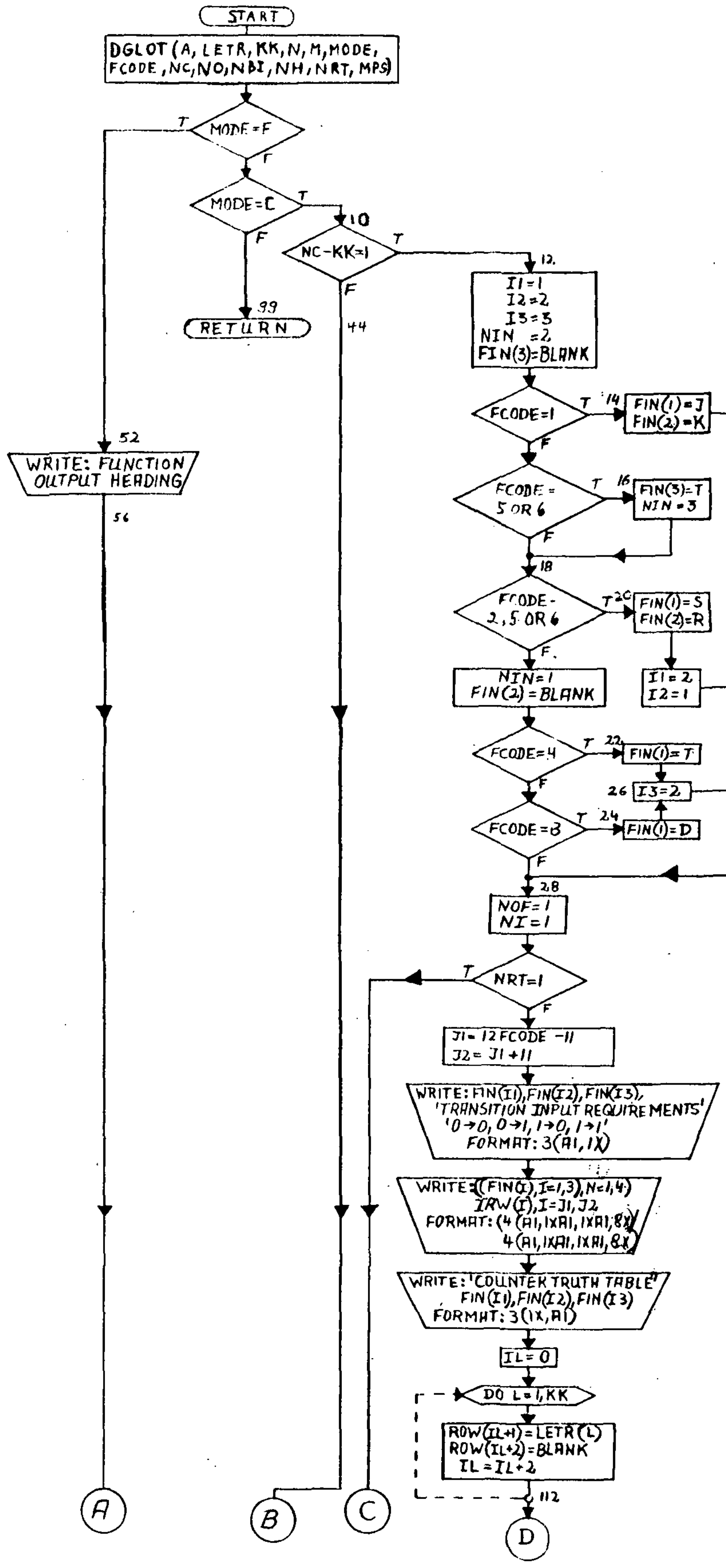
Figure 5.22. Subroutine DGLOT (Sheet 2 of 4)

```
   34 FORMAT (1H+,34X'---->  'A1)
      WRITE (IO,35)
   35 FORMAT (40X':'12X':'/40X':'12X':')
      IF(FIN(3) .NE. BLANK) WRITE(IO,34) FIN(3)
      IF(I3 .EQ. 2) WRITE(IO,34) FIN(1)
      WRITE (IO,36)
   36 FORMAT (40X':'12X':'/40X':'9X2HQ',' :---->')
      IF(FIN(2) .NE. BLANK) WRITE(IO,34) FIN(2)
      WRITE (IO,38)
   38 FORMAT (40X':'12X':'/40X':..............:')
      WRITE (IO,40) LETR(NOF)
   40 FORMAT (1H0/11X'FLIP-FLOP 'A1)
   44 IF(NI .GT. NIN) GO TO 50
   45 WRITE (IO,46) FIN(NI), LETR(NOF)
   46 FORMAT (1H0,13X,2A1,' = ')
      NI = NI + 1
      NC = NC + 1
   48 GO TO 56
   50 NOF = NOF + 1
      NI = 1
      WRITE (IO,40) LETR(NOF)
      GO TO 45
   52 WRITE (IO,60)
   60 FORMAT (////11X,'SIMPLIFIED FUNCTION'4X,'(F = TRUE TERMS /'
     1     18H DON'T CARE TERMS)///15X,'F = ')
   56 L = 0
      JCLR = 1
      IF(MPS .EQ. 1) GO TO 200
      IF(M .EQ. 0) GO TO 57
      GO TO 58
  200 IF(M .EQ. 0) GO TO 257
      L = L + 1
      DATA(L) = LPAR
      DO 272 JJ = 1,M
      LX = L
      DO 264 I = 1,KK
      IF(A(I,JJ) .EQ. 1 .OR. A(I,JJ) .EQ. 0) GO TO 266
      GO TO 264
  266 L =L+ 1
      DATA(L) = LETR(I)
      IF(A(I,JJ) - 1) 270,268,270
  268 L = L + 1
      DATA(L) = BAR
  270 L = L + 1
      DATA(L) = PLUS
  264 CONTINUE
      IF(L - LX) 272,272,276
  276 DATA(L) = RPAR
  277 L = L + 1
      DATA(L) = LPAR
      IF(L - 80) 272,278,278
  278 L = L - 1
      WRITE (IO,74) (DATA(I),I = 1,L)
      WRITE (IO,80)
      L = 0
      JCLR = 0
```

Figure 5.22. Subroutine DGLOT (Sheet 3 of 4)

```
      GO TO 277                                                         DGOT 168
  272 CONTINUE                                                          DGOT 169
      IF(L .LE. 1 .AND. JCLR .EQ. 1) GO TO 284                          DGOT 170
      DATA(L) = BLANK                                                   DGOT 171
      GO TO 95                                                          DGOT 172
  284 DATA(L) = IZRO                                                    DGOT 173
      GO TO 95                                                          DGOT 174
  267 L = L + 1                                                         DGOT 175
      DATA(L) = ONE                                                     DGOT 176
      GO TO 95                                                          DGOT 177
   67 DATA(L+1) = IZRO                                                  DGOT 178
      DATA(L+2) = BLANK                                                 DGOT 179
      DATA(L+3) = SLH                                                   DGOT 180
      DATA(L+4) = BLANK                                                 DGOT 181
      L = L + 4                                                         DGOT 182
      IF(NRT .EQ. 1) GO TO 73                                           DGOT 183
   60 DO 72 JJ= 1,N                                                     DGOT 184
      LX = L                                                            DGOT 185
      DO 64 I = 1,KK                                                    DGOT 186
      IF(A(I,JJ).EQ. 1 .OR. A(I,JJ).EQ. 0) GO TO 66                     DGOT 187
      GO TO 64                                                          DGOT 188
   66 L = L + 1                                                         DGOT 189
      DATA(L) = LETR(I)                                                 DGOT 190
      IF(A(I,JJ)) 64,68,64                                              DGOT 191
   68 L = L + 1                                                         DGOT 192
      DATA(L) = BAR                                                     DGOT 193
   64 CONTINUE                                                          DGOT 194
   70 IF(L .GT. LX) GO TO 76                                            DGOT 195
      IF(JJ.EQ. M .OR. JJ.EQ. N) GO TO 82                               DGOT 196
   71 IF(JJ .EQ. M .AND. NRT .EQ. 1) GO TO 73                           DGOT 197
      GO TO 72                                                          DGOT 198
   82 IF(((L.EQ.4 .AND. M.EQ.0).OR.L.LE.3).AND. JCLR.EQ.1) GO TO 84     DGOT 199
      DATA(L+1) = SLH                                                   DGOT 200
      GO TO 71                                                          DGOT 201
   84 L = L + 1                                                         DGOT 202
      DATA(L) = ONE                                                     DGOT 203
   76 L = L + 1                                                         DGOT 204
      DATA(L) = BLANK                                                   DGOT 205
   77 DATA(L+1) = PLUS                                                  DGOT 206
      IF(JJ.EQ. M) DATA(L+1) = SLH                                      DGOT 207
      DATA(L+2) = BLANK                                                 DGOT 208
      L = L + 2                                                         DGOT 209
      IF(L - 80) 71,78,78                                               DGOT 210
   78 L = L - 3                                                         DGOT 211
      WRITE (10,74) (DATA(I),I = 1,L)                                   DGOT 212
      WRITE (10,80)                                                     DGOT 213
   80 FORMAT (1H0)                                                      DGOT 214
      L = 0                                                             DGOT 215
      JCLR = 0                                                          DGOT 216
      GO TO 77                                                          DGOT 217
   72 CONTINUE                                                          DGOT 218
   73 DATA(L+1) = BLANK                                                 DGOT 219
   95 WRITE (10,74) (DATA(I),I = 1,L)                                   DGOT 220
   74 FORMAT (1H+,18X,110A1)                                            DGOT 221
   99 RETURN                                                            DGOT 222
      END                                                               DGOT 223
```

Figure 5.22. Subroutine DGLOT (Sheet 4 of 4)



START
DGLOT (A, LETR, KK, N, M, MODE, FCODE, NC, NO, NBI, NH, NRT, MPS)
MODE=F
T
F
MODE=C
T
F
10
NC-KK=1
T
F
99
RETURN
44
12
I1=1
I2=2
I3=3
NIN =2
FIN(3)=BLANK
FCODE=1
14
FIN(1)=J
FIN(2)=K
52
WRITE: FUNCTION OUTPUT HEADING
56
FCODE = 5 OR 6
16
FIN(3)=T
NIN=3
18
FCODE-2, 5 OR 6
20
FIN(1)=S
FIN(2)=R
I1=2
I2=1
NIN=1
FIN(2)=BLANK
FCODE=4
22
FIN(1)=T
26
I3=2
FCODE=8
24
FIN(1)=D
28
NOF=1
NI=1
NRT=1
J1=12FCODE -11
J2=J1+11
WRITE: FIN(I1),FIN(I2),FIN(I3), 'TRANSITION INPUT REQUIREMENTS' '0→0, 0→1, 1→0, 1→1' FORMAT: 3(A1,1X)
WRITE: ((FIN(I),I=1,3),N=1,4) IRW(I),I=J1,J2 FORMAT: (4(A1,1XA1,1XA1,8X)/ 4(A1,1XA1,1XA1,8X))
WRITE: 'COUNTER TRUTH TABLE' FIN(I1),FIN(I2),FIN(I3) FORMAT: 3(1X,A1)
IL=0
DO L=1,KK
ROW(IL+1)=LETR(L)
ROW(IL+2)=BLANK
IL=IL+2
112
A
B
C
D

Figure 5.23. Flow Chart of Subroutine DGLOT (Sheet 1 of 5)

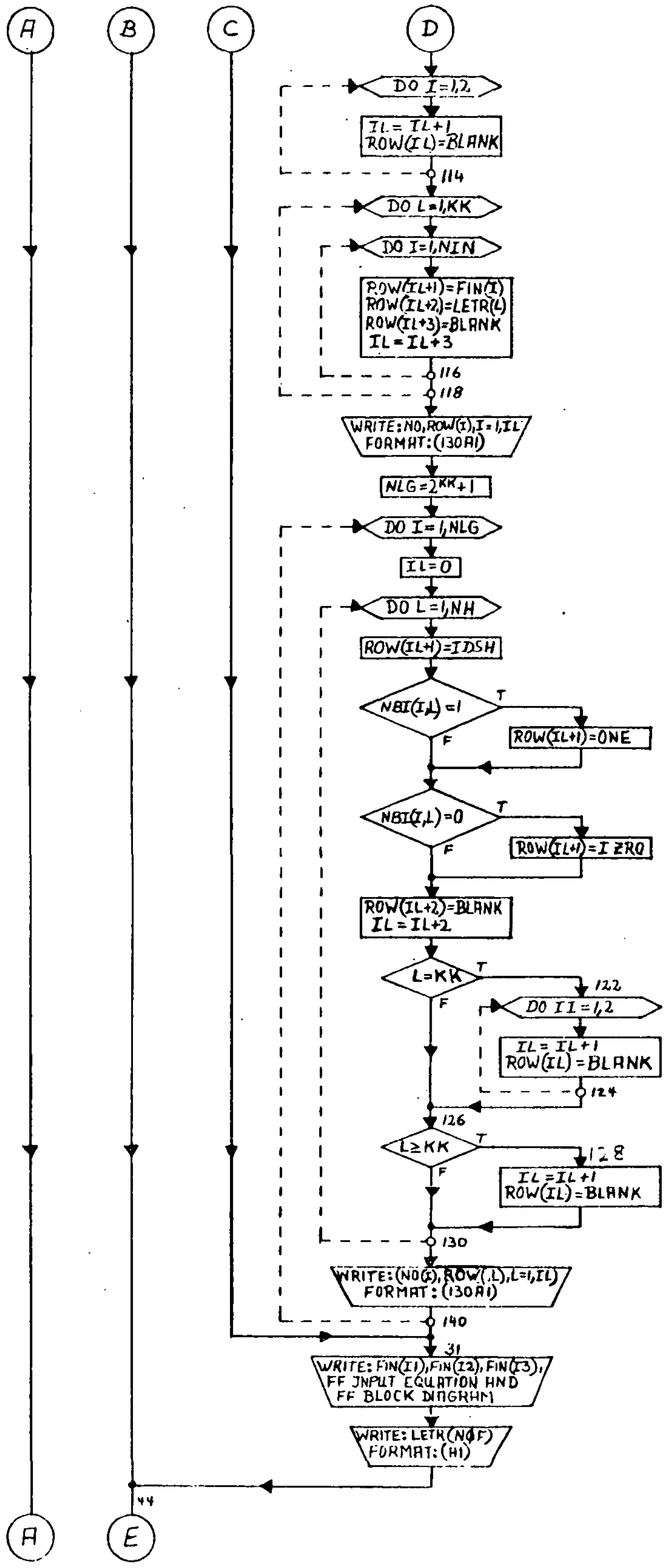


Figure 5.23. Flow Chart of Subroutine DGLOT (Sheet 2 of 5)

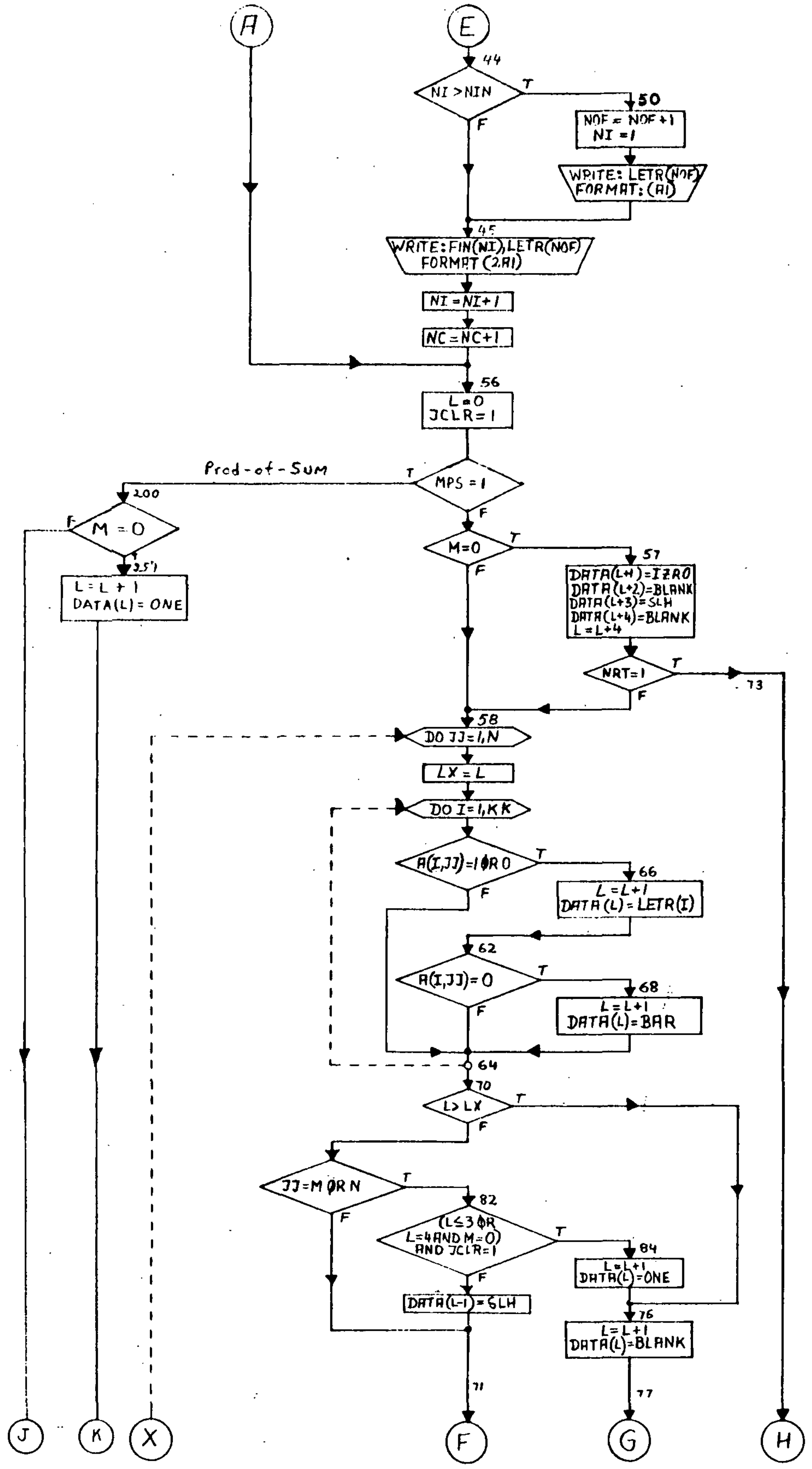


Figure 5.23. Flow Chart of Subroutine DGLOT (Sheet 3 of 5)

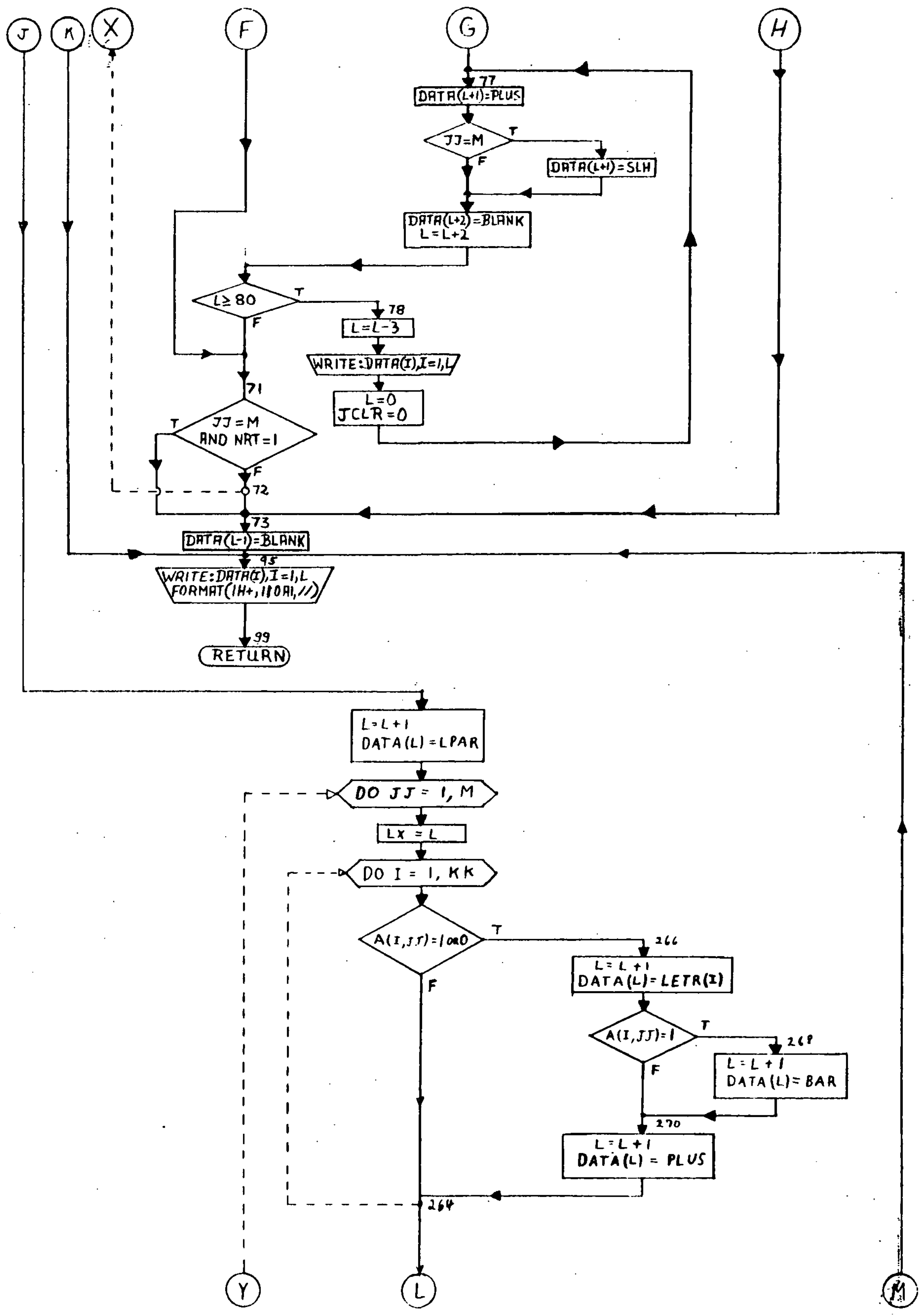


Figure 5.23. Flow Chart of Subroutine DGLOT (Sheet 4 of 5)

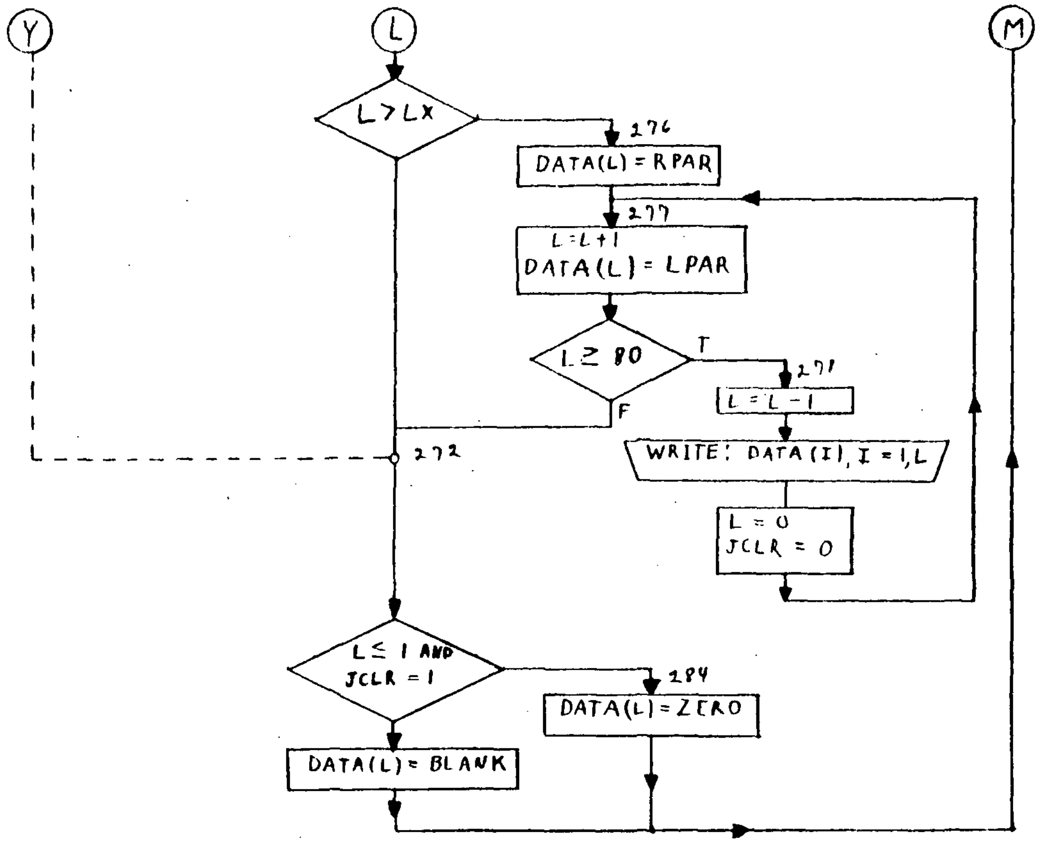


Figure 5.23. Flow Chart of Subroutine DGLOT (Sheet 5 of 5)

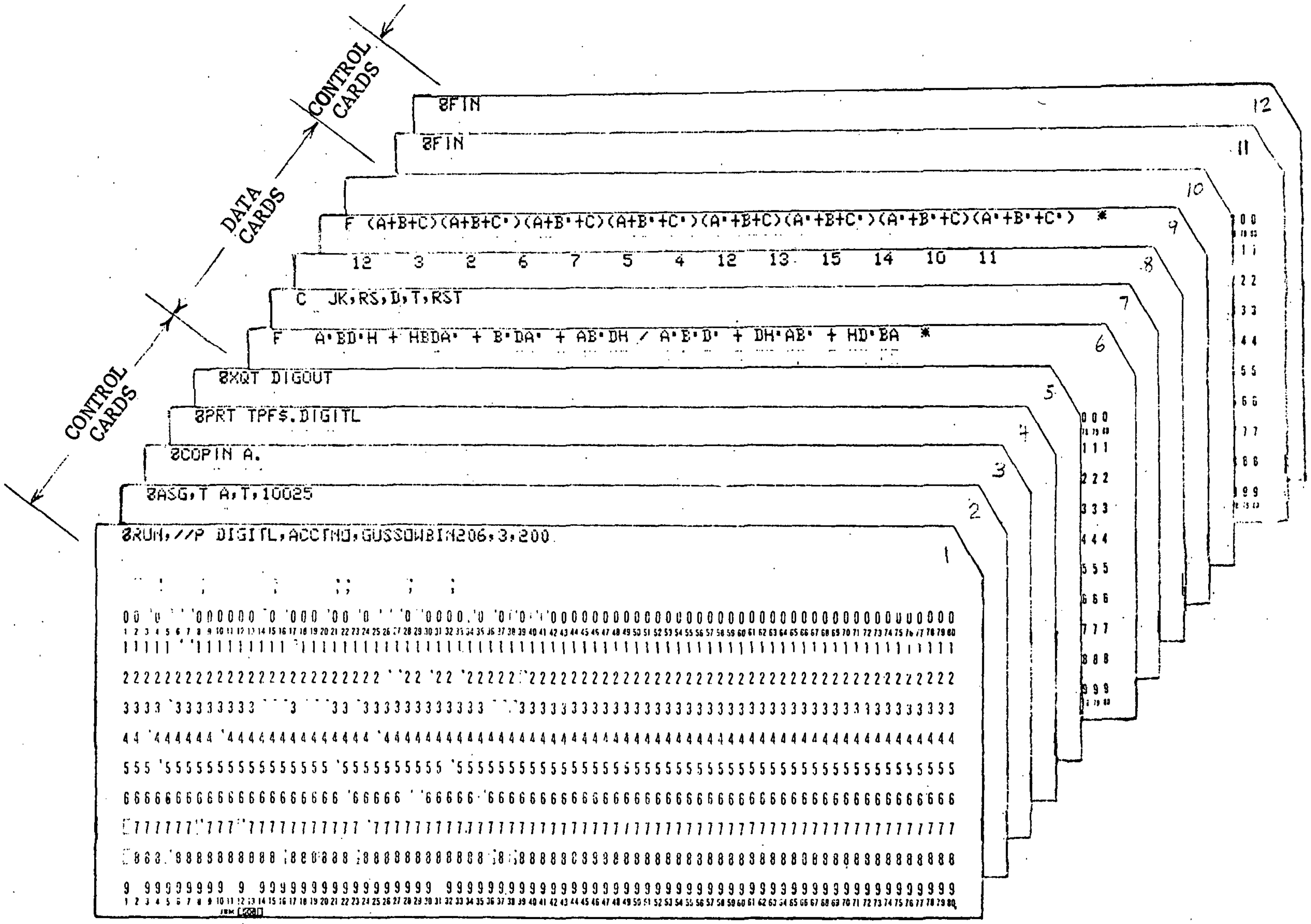
APPENDIX

USE OF PROGRAM STORED ON TAPE

The complete program deck, consisting of one main-line program, ten subroutines, and control cards, contains about 1000 cards. Unless the program is stored on a tape or disc, these 1000 cards must be submitted each time the program is run. Hence, where the program is often used (a most probable situation), it is convenient to store part or all of the program.

An example of the program stored on tape is given for the program stored for use by NASA personnel located at Astrionics Laboratory, Bldg. 4487, Marshall Space Flight Center (MSFC). The program was stored on tape for use on the Univac 1108 computer at the Computation Laboratory, Bldg. 4663, MSFC. The program deck, including sample data cards, required to run the stored program is given in Figure A.1. MSFC personnel can submit this deck either through a terminal station or directly to the Computation Laboratory. One such terminal station is located at Astrionics Laboratory, Room A288.

The function of the program deck illustrated in Figure A.1 is as follows. Card #1 initiates the program run and provides accounting, identification, and scheduling information. On card #1 each user should substitute their own charge account number in columns 17-22, name in columns 24-29, and bin number in columns 33-35. Control cards # 2 and 3 call the stored program. For this example, the program is stored on tape number 10025 as indicated on card #2. Card #3 reads the elements from the tape into the program file. Card #4 prints out the source listing of main-line program DIGITL exactly as given in Figure 5.2. If this source listing is not desired, card #4 can be left out. Control card #5 initiates execution of the program. Cards #6 through 10 are sample data cards. Any number of design cases consisting of any number of data cards can be included. Note that the last data card must be a blank card. The last two cards (cards #11 and 12) are control cards that terminate the program run. All seven control cards shown begin with a multiple 7-8 punch in column one.



```
@FIN
@FIN

F (A+B+C)(A+B+C')(A+B'+C)(A+B'+C')(A'+B+C)(A'+B+C')(A'+B'+C)(A'+B'+C')  *
 12   3   2   6   7   5   4   12   13   15   14   10   11
C  JK,RS,D,T,RST
F    A'BD'H + HBDA' + B'DA' + AB'DH / A'B'D' + DH'AB' + HD'BA  *
@XQT DIGOUT
@PRT TPF$.DIGITL
@COPIN A.
@ASG,T A,T,10025
@RUN,//P DIGITL,ACCTNO,GUSSOWBIN206,3,200
```

Figure A.1. Program Deck Required to Run Program DIGITL Stored on Tape For Univac 1108 Computer at MSFC

All the thirteen design examples contained in this manual (sections 3.3 and 4) were run at the same time on the UNIVAC 1108 computer using this program deck. With all the thirteen design cases submitted on the same run, the program used a total of 1.7 minutes computer run time and 25,000 words memory. The typical cost for this computer usage is $10. The 25,000 words memory is required for a counter design having 8 bits or 255 count states. Since all the design examples were for counters having 6 bits or less, the required memory for this run could have been substantially reduced.

BIBLIOGRAPHY

Chu, Y., "Digital Computer Design Fundamentals", McGraw-Hill Book Co., 1962.

Marcus, M. P., "Switching Circuits For Engineers", Prentice-Hall, Inc., 1967.

Miller, R. E., "Switching Theory, Vol. II, Sequential Circuits and Machines", John Wiley & Sons, Inc., 1965.

Phister, Montgomery, "Logical Design of Digital Computers", John Wiley & Sons, Inc., 1958.

Richards, R. K., "Arithemetic Operations in Digital Computers", D. Van Nostrand Co., 1955.

"The aeronautical and space activities of the United States shall be conducted so as to contribute . . . to the expansion of human knowledge of phenomena in the atmosphere and space. The Administration shall provide for the widest practicable and appropriate dissemination of information concerning its activities and the results thereof."

—NATIONAL AERONAUTICS AND SPACE ACT OF 1958

NASA SCIENTIFIC AND TECHNICAL PUBLICATIONS

TECHNICAL REPORTS: Scientific and technical information considered important, complete, and a lasting contribution to existing knowledge.

TECHNICAL NOTES: Information less broad in scope but nevertheless of importance as a contribution to existing knowledge.

TECHNICAL MEMORANDUMS: Information receiving limited distribution because of preliminary data, security classification, or other reasons. Also includes conference proceedings with either limited or unlimited distribution.

CONTRACTOR REPORTS: Scientific and technical information generated under a NASA contract or grant and considered an important contribution to existing knowledge.

TECHNICAL TRANSLATIONS: Information published in a foreign language considered to merit NASA distribution in English.

SPECIAL PUBLICATIONS: Information derived from or of value to NASA activities. Publications include final reports of major projects, monographs, data compilations, handbooks, sourcebooks, and special bibliographies.

TECHNOLOGY UTILIZATION PUBLICATIONS: Information on technology used by NASA that may be of particular interest in commercial and other non-aerospace applications. Publications include Tech Briefs, Technology Utilization Reports and Technology Surveys.

Details on the availability of these publications may be obtained from:

SCIENTIFIC AND TECHNICAL INFORMATION OFFICE

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Washington, D.C. 20546