NASA CONTRACTOR
REPORT

NASA CR-149969

(NASA-CR-149969) BANNING PRF PROGRAMMER'S MANUAL (M&S Computing, Inc., Huntsville, Ala.) 165 p HC $6.75 CSCL 09B

N76-30847

Unclas
G3/61 50422

BANNING PRF PROGRAMMER'S MANUAL

By R. L. Kuelthau
M&S Computing, Inc.
Post Office Box 5183
Huntsville, Alabama 35804

December 30, 1970

Prepared for
NASA - GEORGE C. MARSHALL SPACE FLIGHT CENTER
Marshall Space Flight Center, Alabama 35812

TECHNICAL REPORT STANDARD TITLE PAGE

1. REPORT NO.	2. GOVERNMENT ACCESSION NO.	3. RECIPIENT'S CATALOG NO.
NASA CR-149969		

4. TITLE AND SUBTITLE	5. REPORT DATE
BANNING PRF PROGRAMMERS MANUAL	December 30, 1970
	6. PERFORMING ORGANIZATION CODE

7. AUTHOR(S)	8. PERFORMING ORGANIZATION REPORT #
R. L. Kuehlthau	70-0036

9. PERFORMING ORGANIZATION NAME AND ADDRESS	10. WORK UNIT NO.
M&S Computing, Inc. P.O. Box 5183 Huntsville, AL 35805	11. CONTRACT OR GRANT NO. NAS8-25621

12. SPONSORING AGENCY NAME AND ADDRESS	13. TYPE OF REPORT & PERIOD COVERED
National Aeronautics and Space Administration Marshall Space Flight Center, Alabama 35812	Contractor Report
	14. SPONSORING AGENCY CODE EC45

15. SUPPLEMENTARY NOTES

Electronics Development Division, Electronics and Control Laboratory
Design Techniques Branch

16. ABSTRACT

This manual describes a modification of the Banning Placement-Routing-Folding Program written by Radio Corporation of America and transmitted to the Marshall Space Flight Center in May 1970. The modifications to this program have been made by M&S Computing, Inc. to implement it on MSFC's Sigma 5 computer and in support of their activities using the program. The input requirements of this program are detailed in the "Banning Placement-Routing-Folding User's Manual", while the required output, which is used as input for the Banning ARTWORK Generation Program, can be found described in detail in the "Banning ARTWORK Program User's Manual". This document presumes the reader is familiar with both of these documents and has a working knowledge of the FORTRAN IV language.

Included in this manual are flowcharts of various levels, beginning with high level functional diagrams and working down to the level of detail deemed necessary to understand the operations of the various sections of the program. Along with the flowcharts of each subroutine is a narrative description of its functional operation and definitions of its arrays and key variables. A section is also included to assist the programmer in dimensioning the program's arrays.

17. KEY WORDS	18. DISTRIBUTION STATEMENT
	Unclassified-Unlimited COR: John R. Gould EC01 R. M. Aden for F. B. Moore Director, E&C Lab

19. SECURITY CLASSIF. (of this report)	20. SECURITY CLASSIF. (of this page)	21. NO. OF PAGES	22. PRICE
Unclassified	Unclassified	164	NTIS

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

SECTION I

INTRODUCTION

The Placement-Routing-Folding Program (PRF) is part of the Banning MOS large scale integrated circuit production system. The input to this program is the logic design of an integrated circuit specified by a net list of required interconnections between Banning Standard Cells. The output of PRF is a magnetic tape of data defining the physical locations of the Standard Cells and the interconnections between these cells, which are utilized in implementing the logic design. This output is in a format which may be used as input for the Banning ARTWORK Program which would normally be used to generate the plotter coding used to produce the integrated circuit's masks. The Banning Standard Cell Engineering Notebook and the Placement-Routing-Folding User's Manual contain detailed information on formulation of logic designs using the Banning Standard Cells and on the translation of these designs into PRF program inputs. Details of the output requirements for driving ARTWORK can be found in the Banning ARTWORK Program User's Manual.

This document describes the operations of the PRF program from input of the logic design to output of ARTWORK data. Throughout the program, control is maintained by the executive titled MAIN, which calls a sequence of subroutines to process data stored in COMMON memory (COMMON memory being that portion of core memory which is saved for access by all subroutines of the program). Section II of this manual discusses the function performed by each call in this sequence, and contains definitions of COMMON variables and references to the COMMON arrays, as well as a flow diagram of MAIN.

The subroutines called by MAIN are described individually in Section III, along with the secondary and tertiary subroutines which may be called in turn. Following a brief functional description of each subroutine is a list of definitions of key variables and arrays which are constructed or modified by the routine. Flowcharts provided are of the picture-on-a-page format, allowing the programmer to reference the level of detail he may require. The lowest level of detail for which flowcharts are provided varies with the complexity of subroutine coding, as considerably greater detail is required to give complete and accurate pictures of the more difficult routines.

Section IV describes the PRF Program's overlay scheme which is used to reduce core memory requirements. It also contains a detailed analysis of the relationship between the complexity of the logic design being run and the required dimensions of the individual arrays. This analysis has been included since these arrays comprise the bulk of the program's core memory requirements, and it is anticipated their overflow will be the cause of the majority of the programming problems encountered.

The final section of this manual is a glossary of terms, abbreviations, and conventions used throughout the manual. A list of references to other documents of the Banning System is also included. Familiarity with the definitions of this section and with these references will save the reader from the necessity of making numerous later references while studying Sections II, III, and IV of this manual.

SECTION II

PROGRAM EXECUTIVE MAIN

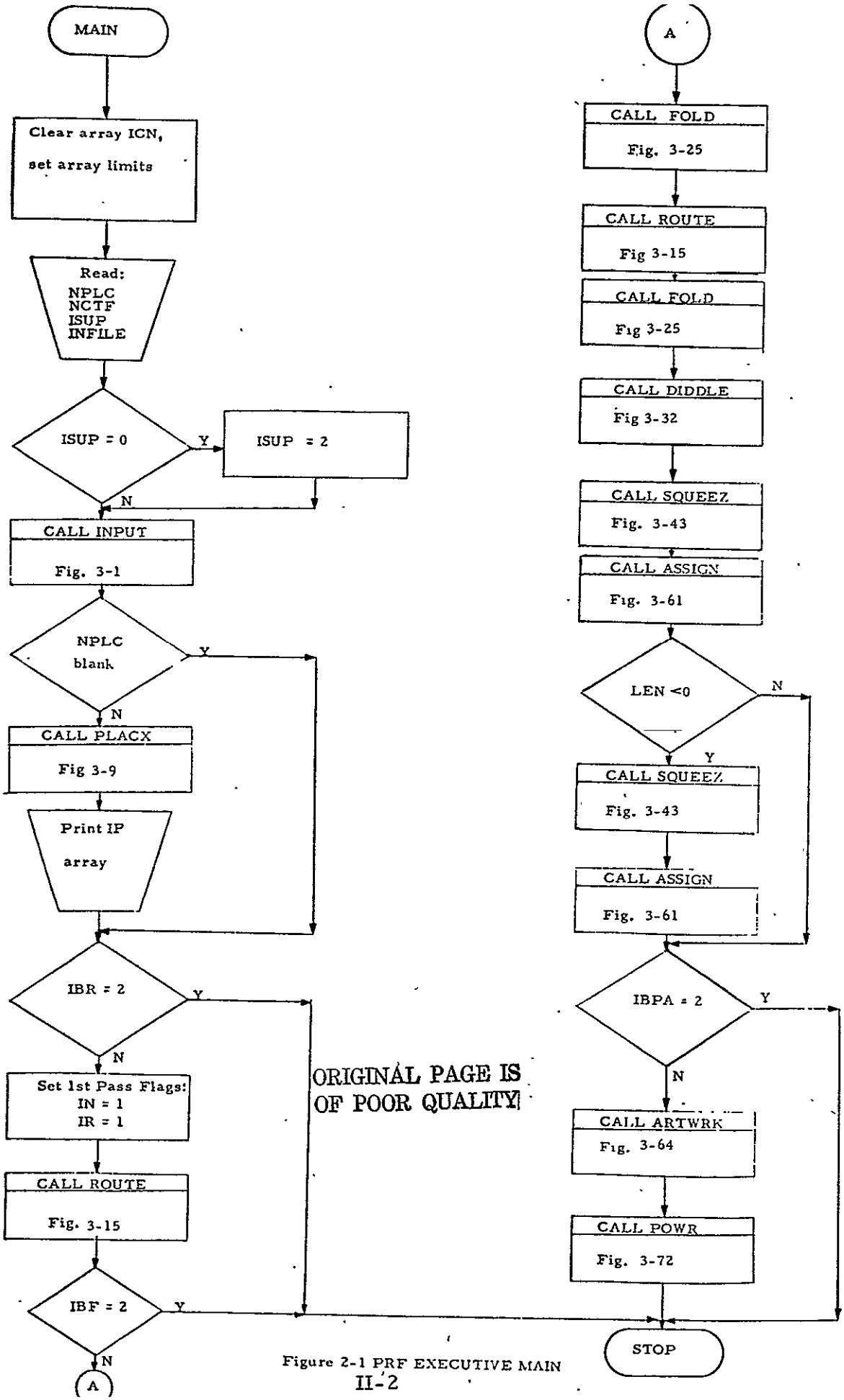
The PRF program is constructed with a simple executive program MAIN which maintains control at all times. Most of the program's operations are performed by subroutines called sequentially by this executive. It will be useful to reference the details of this calling sequence in the MAIN flowchart (Figure 2-1) while reading the following discussion of its functions.

2.1 Functional Description

The PRF program performs the translation from the logic designer's cell interconnection requirements to physical definition of the MOS chip by manipulating data arrays which usually remain in COMMON core memory. The user controls this translation by altering the values of those input parameters which are recognized by the program as flags. The first data card is read by MAIN and contains four such parameters (NPLC, NCTF, ISUP, and INFILE). The remaining 75 input parameters, the initial relative placement of logic cells, the list of required interconnections between logic pins of these cells, and the assignment of standard BANNING circuit numbers to these cells comprises the remainder of the user's normal input data. MAIN calls INPUT to read, check, and format all of this data into the COMMON arrays which are referenced throughout the translation process. Arrays describing the logic pins of the BANNING circuits are also formed by this subroutine with data read from the Circuit Type File.

If the NPLC flag was set on the first data card, PLACX will be called to attempt to reduce the total length of interconnections by interchanging cell numbers in the array into which the user's initial placement has been mapped. The final placement of cells is printed before the program proceeds with the routing of connections between the cell pins. This final placement is in a form identical to that of the user's initial placement and consists of a list showing the order in which the logic cells would appear if they were to be connected together in a straight line. It is in this linear form that the cells will be interconnected and mapped during the preliminary phase of PRF.

The chip is mapped in memory by letting each element of a large two dimensional array represent a position on the chip relative to the other elements of the array. The number which is entered in an element indicates whether metal, P-material (tunnel), feedthrough (tunnel end) or some Banning cell occupies that position. The ROUTE subroutine is called to form such a



ORIGINAL PAGE IS
OF POOR QUALITY

Figure 2-1 PRF EXECUTIVE MAIN

map of the final linear cell placement and enter the required interconnections between these cells. FOLD is then called to examine the widths of the interconnections in this map to determine the points at which this linear array should be folded upon itself to form a square chip.

The folds, or sections of the linear map which are bounded by these points, are then individually rerouted during a second pass through ROUTE. By routing a fold's wiring independent of the other folds, space is utilized more efficiently and the required number of wiring channels can usually be reduced. A second pass through FOLD follows this pass through ROUTE during which the fold points are refined and the linear map is broken into maps of the individual folds which are then stored on scratch storage.

The DIDDLE subroutine proceeds by reading the individual maps into a large two dimensional map which is representative of the final chip. As each of the individual fold maps is read into this final map, the signal connections required between it and the previously entered folds are routed. When this has been completed, MAIN proceeds by calling SQUEEZ to refine the mapped chip. This is accomplished by individually clearing and rerouting each fold's interconnections. Several wire channel widths may be saved in this process since the interconnections between adjacent folds are now defined and can be integrated into the pin to pin wiring.

The map of the relative locations of the chip's components is now complete. ASSIGN is now called to assign absolute coordinates to each map array element in accordance with the specified spacing parameters and the spacing requirements of the standard cell components. This subroutine then uses the chip map and these coordinate assignments to produce the chip picture on the line printer.

A normal PRF run proceeds from this point to the ARTWRK subroutine where output for use by the BANNING ARTWORK program is produced. Here pattern set data is generated to produce standard cells, tunnel ends, alignment marks, power pads, and test transistor. Line set data is generated to produce the metal and tunnel interconnections and shape set data may also be generated to produce a chip border. The POWR subroutine follows to calculate the power bus requirements and generate line set data to produce them. This completes the PRF operation.

There are two possible deviations from this normal PRF sequence which the user may select. By setting the LEN input parameter (see PRF

User's Manual) non-zero, he may make insertions or deletions of metal or tunnel connection segments. When this option has been selected, the program will loop back to SQUEEZ after completing the first pass through ASSIGN. SQUEEZ then inputs and implements the change specifications before ASSIGN is called once again to revise coordinate assignments and restore control to the normal program sequence. It is necessary to complete the first pass through ASSIGN before entering the manual changes, since the change specifications are referenced by the absolute chip coordinates assigned by the subroutine. Details of the manual change option can be found in Section VI of the PRF User's Manual.

The other option which may be selected is that of simply bypassing generation of output for the ARTWORK program. If input parameter IBPA is set to zero, this option terminates the program immediately after the ASSIGN subroutine has been completed.

2.2 Program Structure

Figures 2-2 and 2-3 contain a diagram showing the levels of the various subroutines and coding blocks of the PRF program. Distinct subroutines have underlined titles in this tree structure, while the remaining titles are merely blocks of coding which have been flowcharted on the pages having that title as the entry point. The number of the figure on which a subroutine or coding block is flowcharted appears directly below its title. All PRF coding may be referenced by this diagram with the exception of the LOOK subroutine. LOOK, described in Section 3.10, is of such a low level that no specific calls are entered in the flowcharts, although it is referenced at numerous points in the DIDDLE, SQUEEZ, PADMVE, MOVE, ASSIGN, and ARTWRK subroutines.

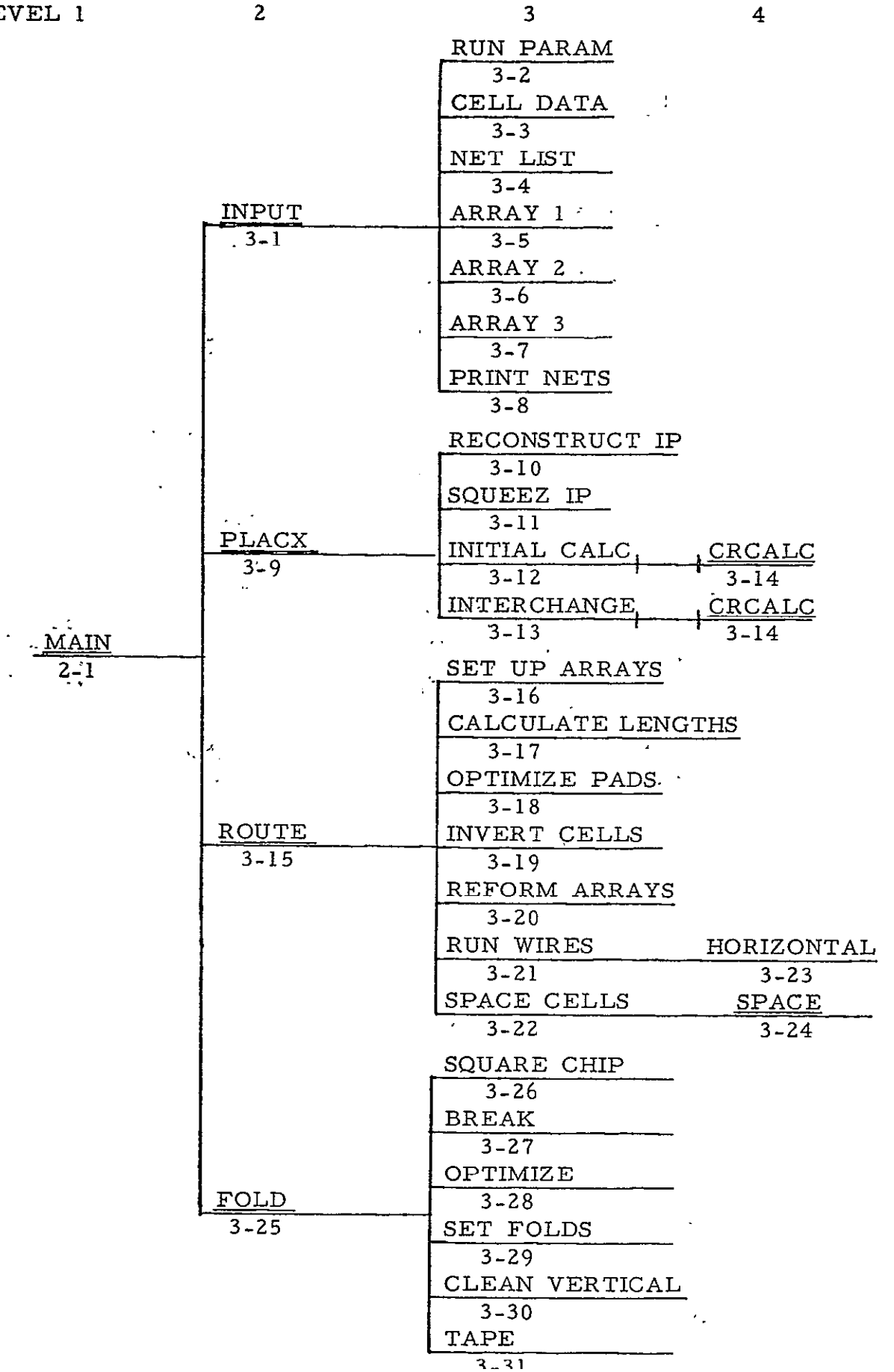


Figure 2-2 PRF CODING LEVELS

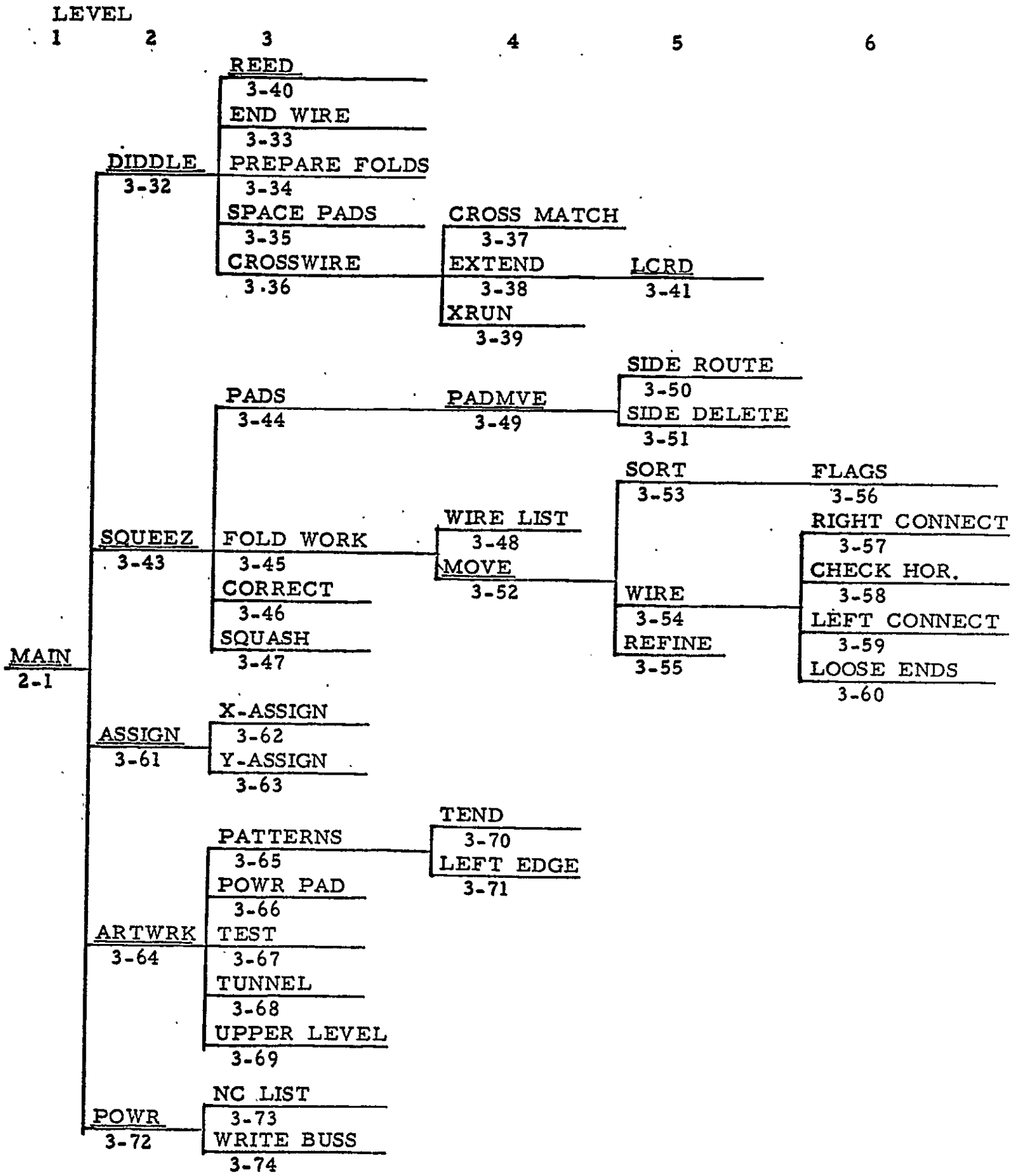


Figure 2-3 PRF CODING LEVELS

2.3 Variables and Arrays

As mentioned previously, most of the data describing the MOS chip being constructed passes between subroutines in COMMON variables and arrays. The following list defines usual use of COMMON variables as well as the few local variables used by MAIN:

COMMON Variables

CAP - Array

IAX - Maximum X-dimension of IFA (X, Y)

IAY - Maximum Y-dimension of IFA (X, Y)

IBF - FOLD bypass flag

IBP - Not used

IBPA - ARTWRK bypass flag

IBPI - Not used

IBR - ROUTE bypass flag

IC - Array

ICK - Array

ICL - Array

ICN - Array

IDBG - Not used

IDNEG - Not used

IDP - Not used

IEC - Array

IEL - Element number used as argument by CRCALC.

IF - Array

IFA	-	Array
IFX	-	Max. X-dimension of IF (X, Y); set to 10.
IHF	-	Cell height.
IJ	-	Number of folds.
IN	-	During PLACX: value of new total pseudo wire length. During ROUTE - FOLD sequence: flag indicating 1st or 2nd pass through sequence.
INEQ	-	Not used.
INT	-	Array
IO	-	Total of old pseudo wire lengths calculated by CRCALC.
IPAD	-	Pattern number of connection pads.
IPTRN	-	Array
IR	-	Flag set to 1 or 2 for 1st or 2nd pass through ROUTE.
IRAMX	-	Index on last pin of IRA.
IRX	-	Maximum X-dimension of IRA (X, Y).
IRY	-	Maximum Y-dimension of IRA (X, Y).
ISWC	-	Not used.
IW	-	Net weight assigned to wire whose pseudo length is being evaluated by PLACX.
IXW	-	Array
IYW	-	Array
JD	-	Array
J3	-	Number of logical unit on which the C. T. F. is to be read.

LE - Length element used by PLACX

LEN - Manual correction option flag

LIMIT - Interchange limit used by PLACX

LL - Number of the fold being rewired by MOVE

MM - Vertical channel spacing; equal to 2xICN (73)

MN - Pointer on vertical channel immediately to left of folds.

M1 - Number of cells on chip

NAR - Not used

NC - Array

NCT - Number of circuit types

NET - Array

NNETS - Number of nets

NP - Number of pins

NWRK - Number of work tape

NX - Array

N1 - Set to 2

PR - Array

MAIN Local Variables

INFILE - Contains the logical number of the input unit on which the Circuit Type File is to be read when cards are not used.

ISUP - Flag which must be set to 1 if circuit pin parameters are to be printed from the C. T. F.

NCTF - Alphanumeric constant used to indicate which input device is to be read for the C. T. F.

NCTF = "CARDS" for card input

NCTF = "TAPE" or "ALTR" for input on device INFILE

NPLC - Alphanumeric constant indicating whether PLACX subroutine is to be run.

NPLC = (Blank) for bypass of PLACX

The PRF COMMON arrays are often used for different purposes at different points in the program. Every subroutine in which an array is constructed or modified to perform a different function contains a description of the new array format as a part of the subroutine's description in Section III. Figure 2-4 indicates which subroutines so alter the various arrays, as well as where each array is referenced. By using this table, the reader can reference the subroutine describing the format of any COMMON array at any point in the program.

The following two arrays are referenced directly by MAIN:

IP (I, J) - MAIN prints the placement from this array which has been set up by INPUT and modified by PLACX.

MAP (I, J) - MAIN uses several discrete locations of this array to pass the three local variables INFILE, ISUP, NCTF on to the INPUT subroutine.

2.4 Debug Printout

The debug flags mentioned in the PRF User's Manual (page IV-11) produce output dumps from the routines listed below:

ICN (36) - PLACX

ICN (37) - INPUT

ICN (38) - ROUTE

ICN (39) - FOLD

SUBROUTINE

ARRAY	INPUT	PLACX	CRCALC	ROUTE	SPACE	FOLD	DIDDLE	REED	LCRD	SQUEEZ	PADMVE	MOVE	ASSIGN	ARTWRK	POWR	LOOK
CAP	A			B										A		
IC	A	C		B	B	A	C			C		C	C	C		C
ICK	A			C			C							C		
ICN	A	C				B					C			B		
IEC	A	C		C		C	C			C			C	C	C	
IF						A	C	C		B		C	B	B	C	
IFA							A	A		B	B	B	B	C	C	
INT				A	C											
IPR						A							A			
IPTRN	A			C			C							C		
IRA		A		A		B										
IXW										A			A	A		
IYW										A			A	A	C	
JD			A	A			B			B	C	B	C	B	C	
NC				A		A				B		B	A	C	A	
NET	A	C	C	B		C								C		
NX				A			A			B	B	B	C	C		C
PR						A							A			

A - ARRAY FORMED BY SUBROUTINE
B - ARRAY MODIFIED BY SUBROUTINE
C - ARRAY REFERENCED BY SUBROUTINE

EQUIVALENT ARRAYS: IFA ↔ IRA
IPR ↔ PR

COMMON ARRAY USAGE MAP

Figure 2-4

ICN (40) - DIDDLE, SQUEEZ

ICN (41) - REED, MOVE, LOOK

ICN (42) - ASSIGN

ICN (43) - ARTWRK

ICN (44) - POWR

None of these dumps are labeled and some of the array dumps are not formatted, making interpretation difficult. Parameter dumps generally print many irrelevant variables along with the few whose values are critical at the particular point in the program. The programmer must use the listing to determine exactly which quantities of which dump are of interest.

SECTION III

SUBROUTINES

3.1 Subroutine INPUT

The INPUT routine reads most of the PRF input data, makes various checks on its validity, and formats it for storage in COMMON arrays.

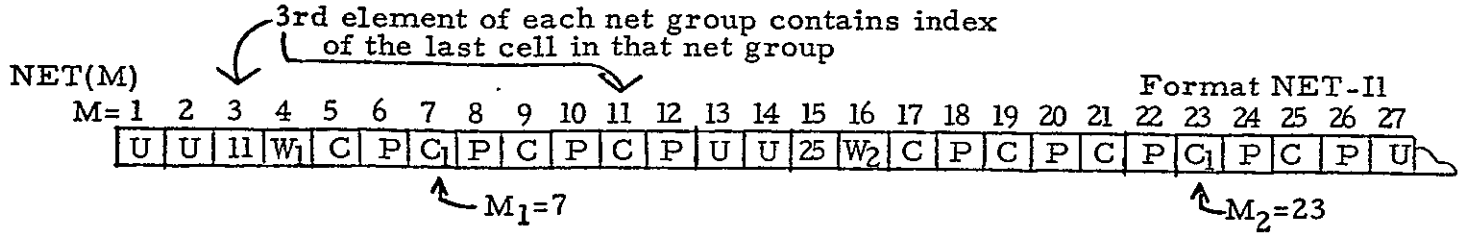
3.1.1 INPUT Functional Description

Since the formats of inputs are described in detail in the PRF User's Manual, as are input options and possible error messages for invalid data, this discussion is concerned primarily with the subroutine's construction of arrays in COMMON memory. After the first 15 control parameters (M1, LEN, LIMINT, IHF, IBR, IBF, IRP, IBP1, IBPA, IDP, IDBG, NWRK, NAR, IPAD, I) have been read as COMMON discretes, the remaining control parameters are read sequentially into the first of these arrays, ICN. Following this, the assignment list is input directly into the IEC array, the initial placement directly into IP, and the net list is formatted and entered into NET. The entire Circuit Type File is input into arrays IPTRA, ICA, and ICB. After these arrays are searched for the data describing the cells to be used in the particular PRF run, and this data is transferred into IPTRN, ICL, and ICK for permanent reference, the original arrays IPTRA, ICA, and ICB have served their purpose and are no longer referenced.

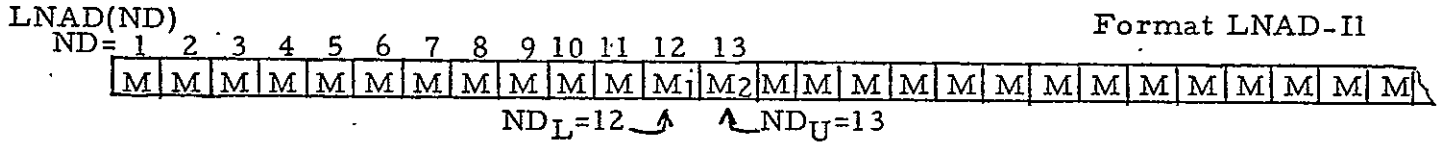
A number of cross-referencing arrays are constructed to enable easy access of cell type data, interconnection data, and placement data. These arrays, INF, LNAD, MAP, IC, and CAP are constructed at the same time that a number of validity checks are made on the data. If an error is detected, an error message, as indicated in the PRF User's Manual and the INPUT flowcharts (Section 3.1.3) is printed and local variable IIE is set to 2, causing an eventual recycle of the INPUT subroutine. The five arrays constructed in this section have a particularly interesting cross-reference structure which is shown in Table 3-1.

The formats of arrays ICK, ICL, ICN, IEC, INF, IP, IPTRN, LNAD, MAP, and NET are significant since this is the form in which they will be referenced throughout the remainder of the program. Maps of all of these arrays are included in Tables 3-1 through 3-3.

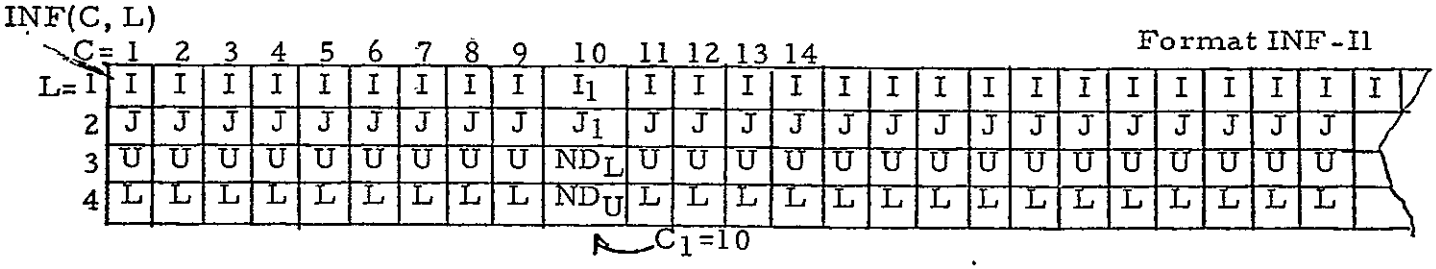
INPUT NET CROSS-REFERENCE ARRAYS



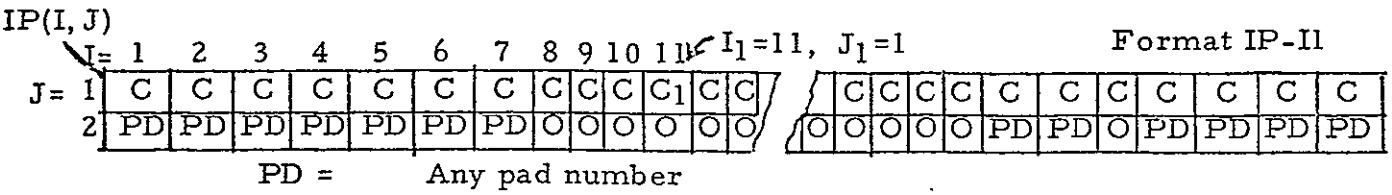
C_1 = Some specific cell number indexed by M_1 and M_2.
U = Unknown number--these elements have not been set.
W_1 = Weighting factor assigned to Net 1.
C = Any cell number
P = Any pin number



M_1 = Some index to cell C_1 in NET
M_2 = Some index to cell C_1 in NET
M = An index to any cell in NET
ND_U = Highest index in LNAD of a reference M to cell C_1
ND_L = Lowest index in LNAD of a reference M to cell C_1



(I, J) = Coordinates locating some cell C in IP
U = Highest index in LNAD of reference to cell C
L = Lowest index in LNAD of reference to cell C
(I, J) = Coordinates locating cell C_1 in IP



PD = Any pad number

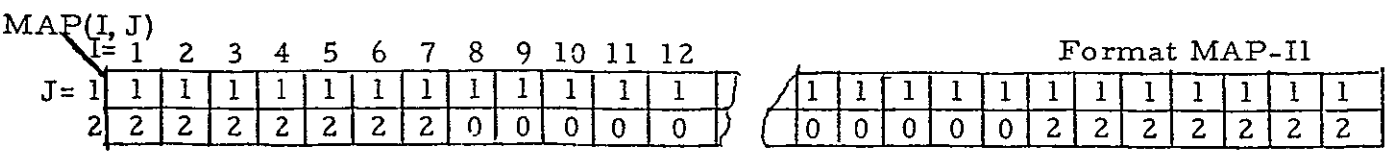


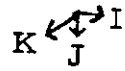
Table 3-1 INPUT NET CROSS REFERENCE ARRAYS

IC(ME, MP)

Format IC-I1

MP \ ME	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	...
1	0	0	0	0	0	0													
2	1	1	1	1	1														
3	1	0	1	1	1														
4	1	1	0	1	0														
5	0	0	0	0	1														
6	0	0	0	0	0														
7	0	0	0	0	0														
8	0	0	0	0	0														
9	0	0	0	0	0														
10	0	0	0	0	0														

1 = Connected Pin
0 = Unused Pin



ICK(I, J, K)

Format ICK-I1

I \ J	1	2	3	4	5	6	7	8	9
1	0	0	0	0	0	0	0		
2	300	201	301	200	200				
3	300	201	351	250	200				
4	350	250	0	200	200				
5	0	0	0	0	250				
6	0	0	0	0	0				
7	0	0	0	0	0				
8	0	0	0	0	0				
9	0	0	0	0	0				
10	0	0	0	0	0				

K = 1	K = 2	K = 3	K = 4
0	19	0	0
4	14	622	300
14	14	619	300
14	4	489	350
19	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

I Index is the Element Number
J Index is the Pin Number
K Index is the Parameter Number
K = 1 Indexes reassignment flags
K = 2 Indexes pin capacitance
K = 3 Indexes right to left pin spacing
K = 4 Indexes left to right pin spacing

Table 3-2 INPUT ARRAYS

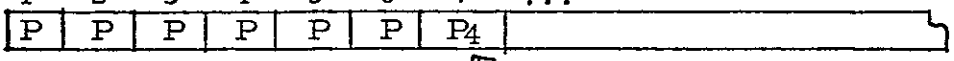
IEC is indexed by the cell number tag assigned in the assignment list.

IEC(I) I = 1 2 3 4 5 ...

Format IEC-I1

P_4 is the Banning Cell Library Pattern Number which has been assigned to Element 4.

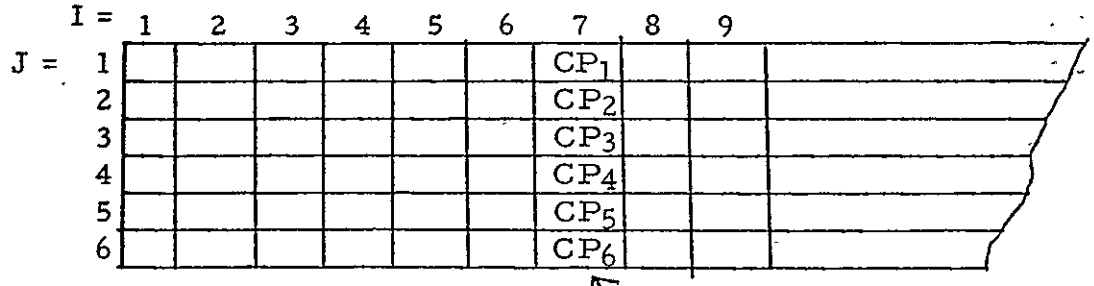
IPTRN(I) I = 1 2 3 4 5 6 7 ...



Format IPTRN-I1

IPTRN(I) is a list of all the patterns read in from the Circuit Type File in order in which they were read. Thus Pattern P_4 is the 7th pattern in the file.

ICL(I, J)



Format ICL-I1

CP_1 - CP_6 are the cell parameters of the 7th pattern in the Circuit Type File.

Table 3-3 INPUT ARRAY MAPS

3.1.2 INPUT Variables and Arrays

Since most of the key variables of this subroutine are in COMMON, there are only a few significant local variables. These are:

I - Dummy variable used as the number of the cell being processed.

IIE - Error indication flag which causes recycle of the INPUT subroutine.

INFILE - Logical number of the alternate input file.

ISUP - Pin data suppression flag; data will only be printed if ISUP = 1.

ND - Dummy variable used as a pointer in forming LNAD and INF.

NP - Dummy variable indicating the number of pins on the cell being processed.

The following arrays are formed by the INPUT subroutine:

A (I) - Temporary storage of alphaneumeric cell name data.

CAP (I) - Contains the total capacitance of cell pins connected to net I.

IC (ME, MP) - Map of pins referenced by net list;

For: IC(ME, MP) = 1 - Pin connected
IC(ME, MP) = 0 - Pin not connected

Where: ME = Element number referenced
MP = Pin number of element MP

ICK (I, J, K) - Contains circuit type file pin parameters for cell types specified in the assignment list:

I = Element number in Circuit Type File
J = Pin number
K = Parameter type index

For: K=1 - Reassignment flag
K=2 - Pin Capacitance
K=3 - Right to left pin spacing
K=4 - Left to right pin spacing

ICL (I, J) - Contains circuit type cell parameters.

I = Element index number in Circuit Type File
J = Cell parameter number

ICN (I) - Program input parameter array.

Note that only 75 parameters are utilized by PRF. Parameters 76-111 are reserved for use by SIGNAL TRACE. Some of the elements between 112 and 200 are used as common data links between subroutines later in the program.

IEC (I) - Pattern assignment array.

Element Number I is assigned the Banning Pattern Number entered in the location it indexes.

INF (C, J) - Contains cross-reference information between array IP, which describes a cell's placement position, and array LNAD, which contains an ordered list of addresses of NET array elements containing the particular cell. See INPUT array maps, Table 3-1.

INF (C, 1) = Contains the X-index locating cell C in IP

INF (C, 2) = Contains the Y-index locating cell C in IP

INF (C, 3) = Contains index to lowest element of LNAD referencing cell C.

INF (C, 4) = Contains index to highest element of LNAD which references cell C.

IP (I, J) - Element position placement array containing the element numbers in the relative positions specified by initial placement. Bonding pads are placed in a separate column.

I = Index to relative position in linear array

J=1 - Column containing logic elements
J=2 - Column containing pad cells. See INPUT array maps.

IPTRN (I) - List of pattern numbers read in from Circuit Type File.

I = Index on position of pattern in Circuit Type File.

ITITLE(I) - Array used to store the title of the PRF run and titles of input parameters.

LNAD(ND) - Contains ordered list of address of NETS in NET array containing particular cells. This list is referenced by INF to provide a cell to net cross-reference. See INPUT array maps, Table 3-1.

ND = Index referenced by INF

NET (M) - Linear array containing all net list data. Net lists are stored in NET in the order they are read from input data cards. Cells in INF may be referenced by utilizing the LNAD and INF arrays. The format of net list data stored in NET is shown by the INPUT array maps.

M = Index referenced by LNAD

MAP (I, J) - Map of available cell position in IP (I, J)

For: Map (I,1) = 1 - Logic cell position which may be utilized.
= 0 - Logic cell position to be vacant.
Map (I, 2) = 2 - Pad position which may be utilized.
= 0 - Pad position to be vacant.

Where: I = Relative position index

J=1 - Logic cell
J=2 - Pad cell

NETOT (150) - Used as temporary storage of pattern/element pairs during the print of the initial assignment list.

ICA (I, J) - Temporary storage of Circuit Type File cell parameters. Format identical to ICL.

ICB (I, J, K) - Temporary storage of Circuit Type File pin parameters. Format identical to ICK.

3.1.3 INPUT Flowcharts

Figures 3-1 through 3-8 contain flowcharts of the INPUT subroutine.

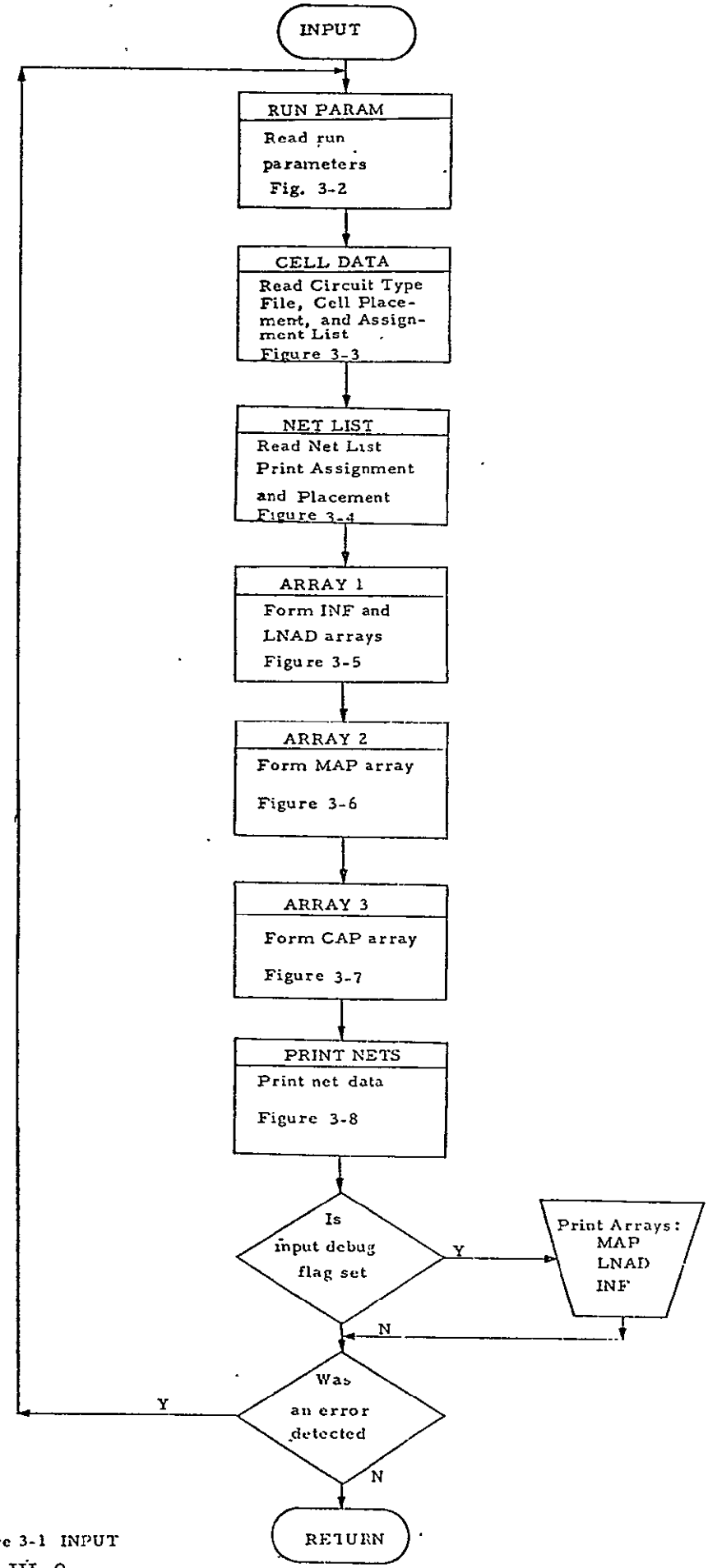


Figure 3-1 INPUT

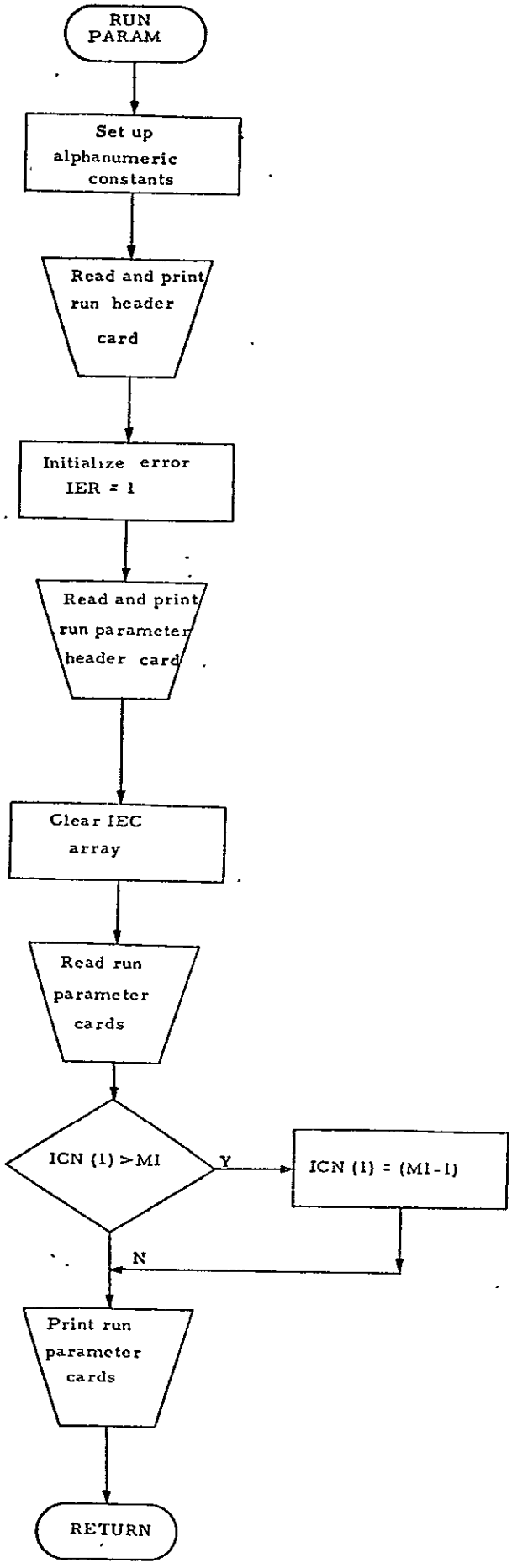
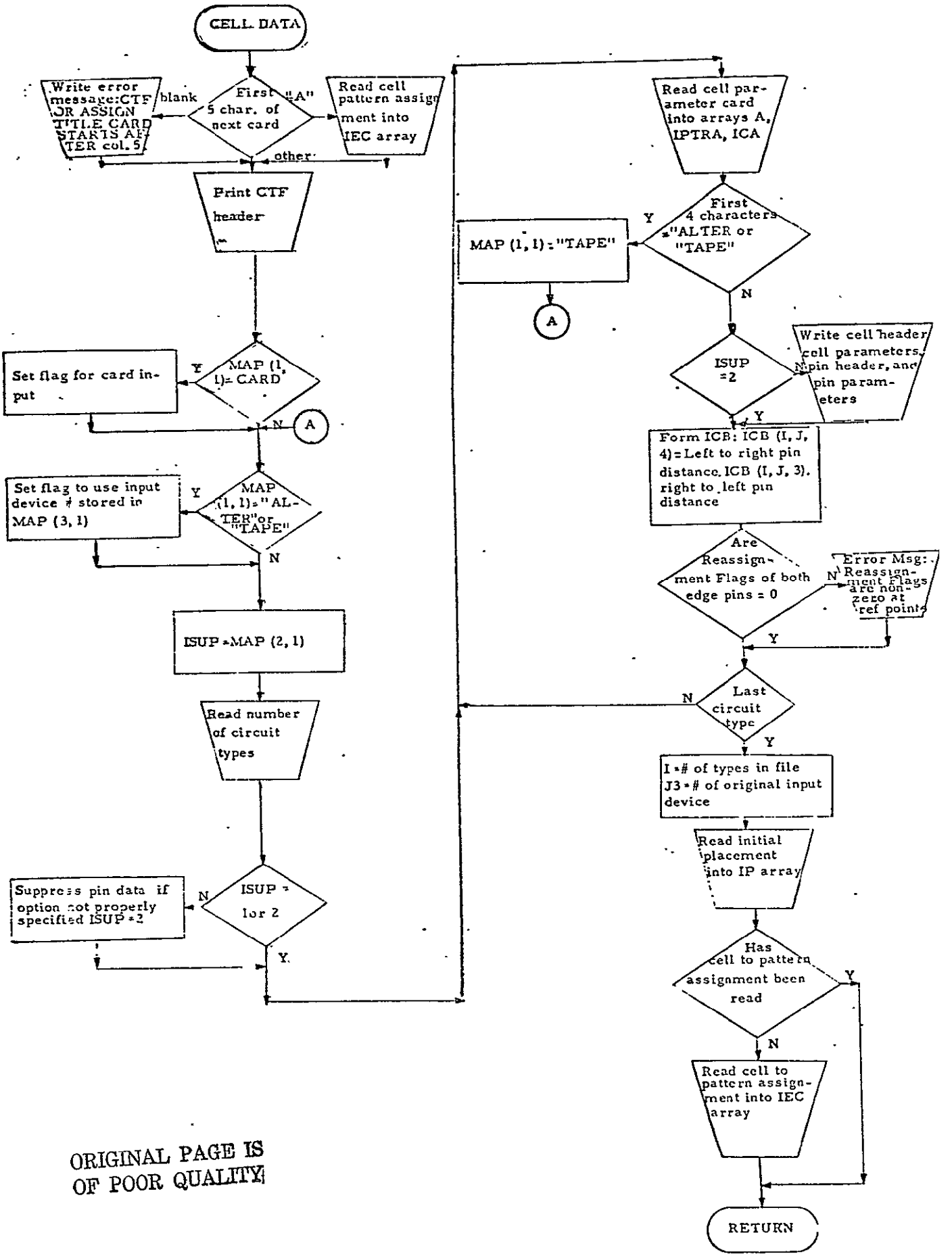


Figure 3-2 RUN PARAM



ORIGINAL PAGE IS
OF POOR QUALITY

Figure 3-3 CELL DATA

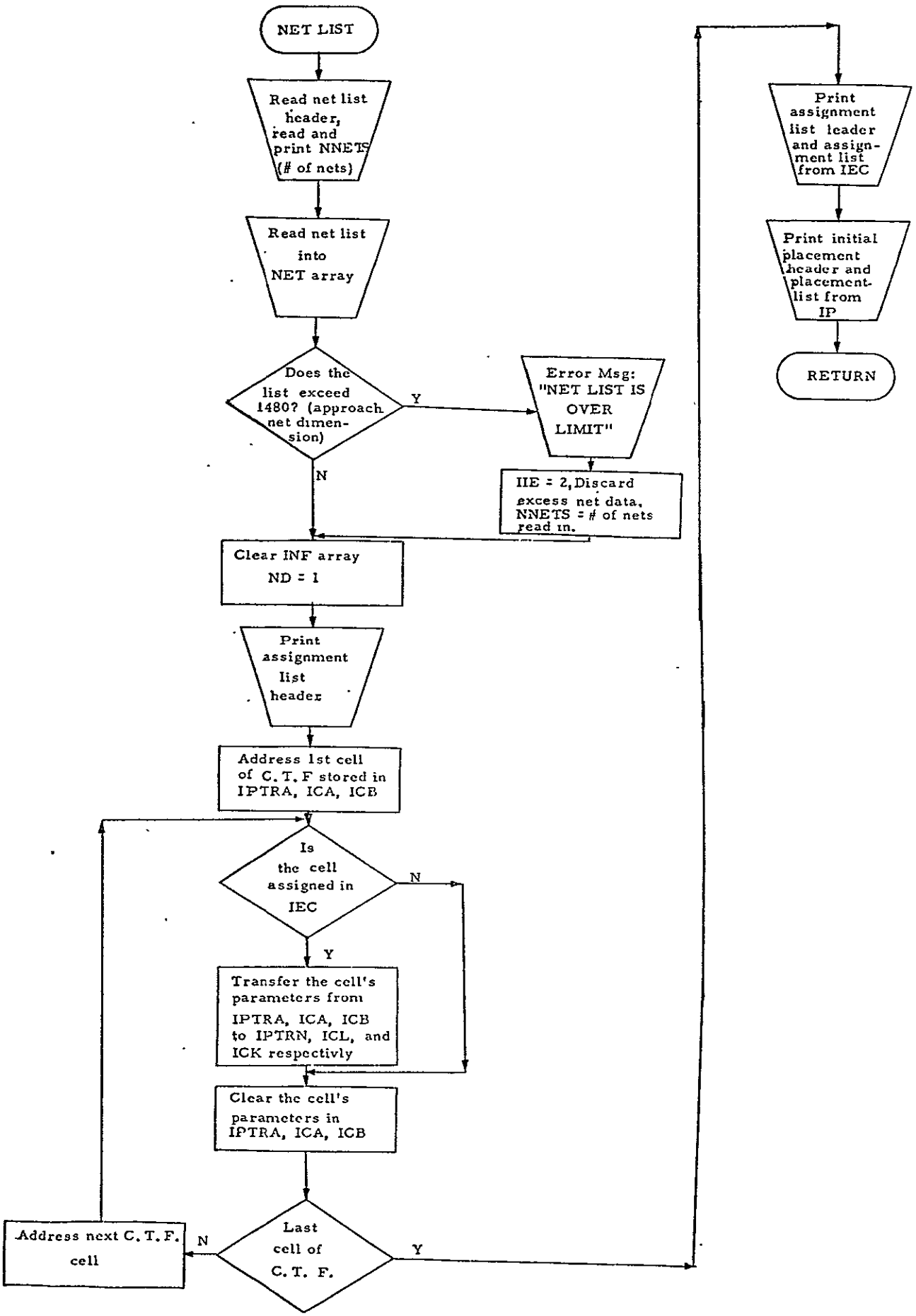
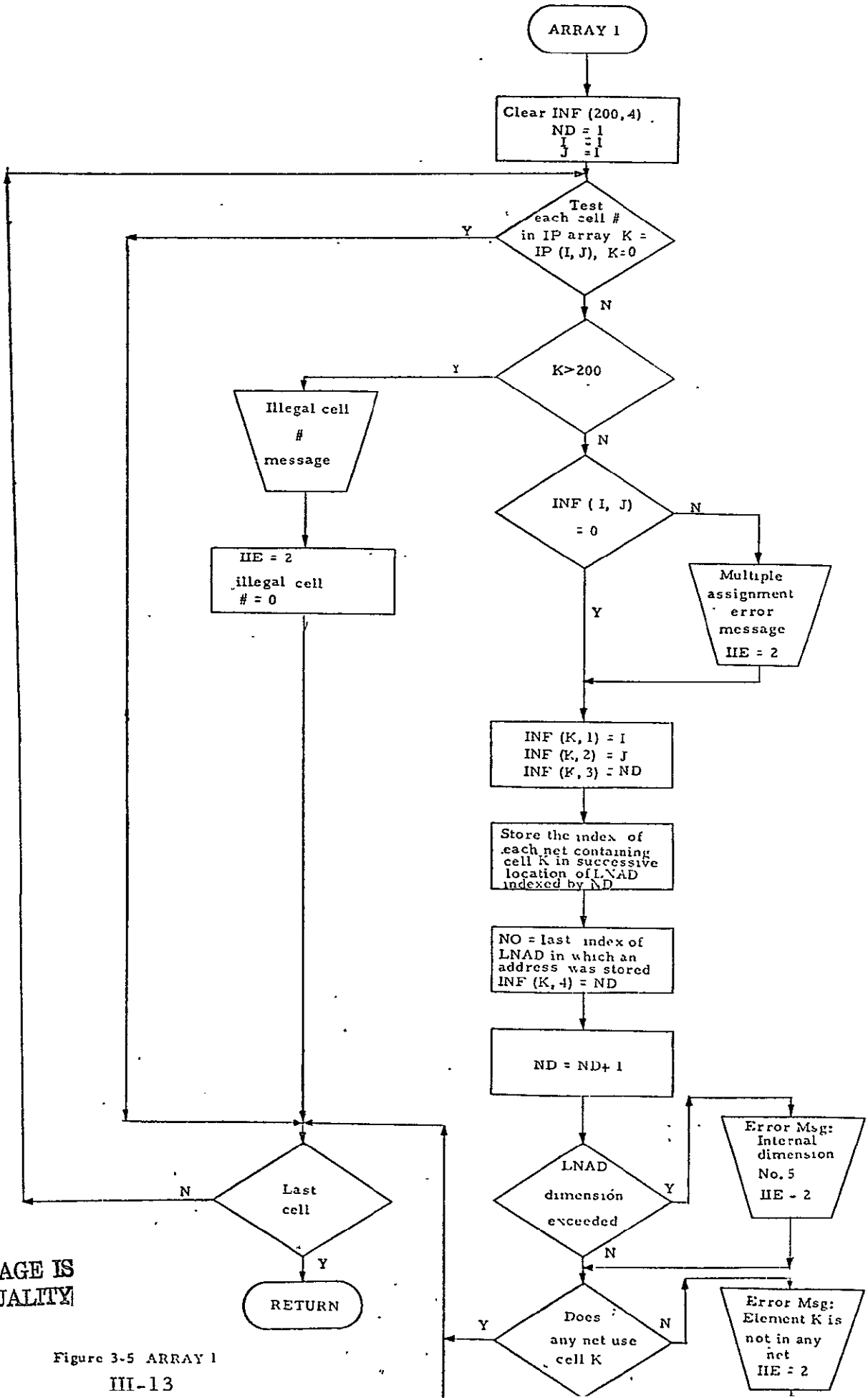


Figure 3-4 NET LIST



ORIGINAL PAGE IS
OF POOR QUALITY

Figure 3-5 ARRAY 1

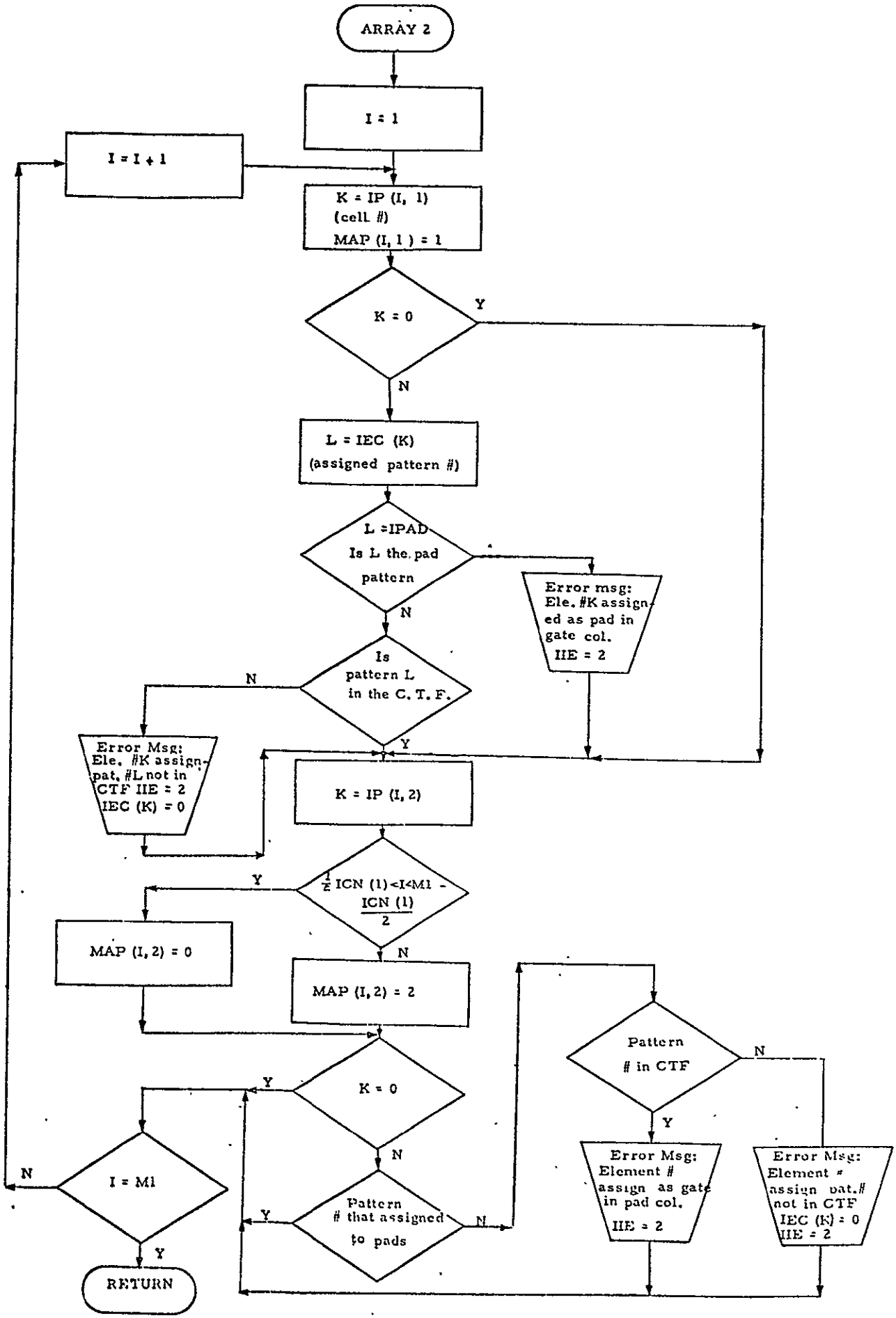
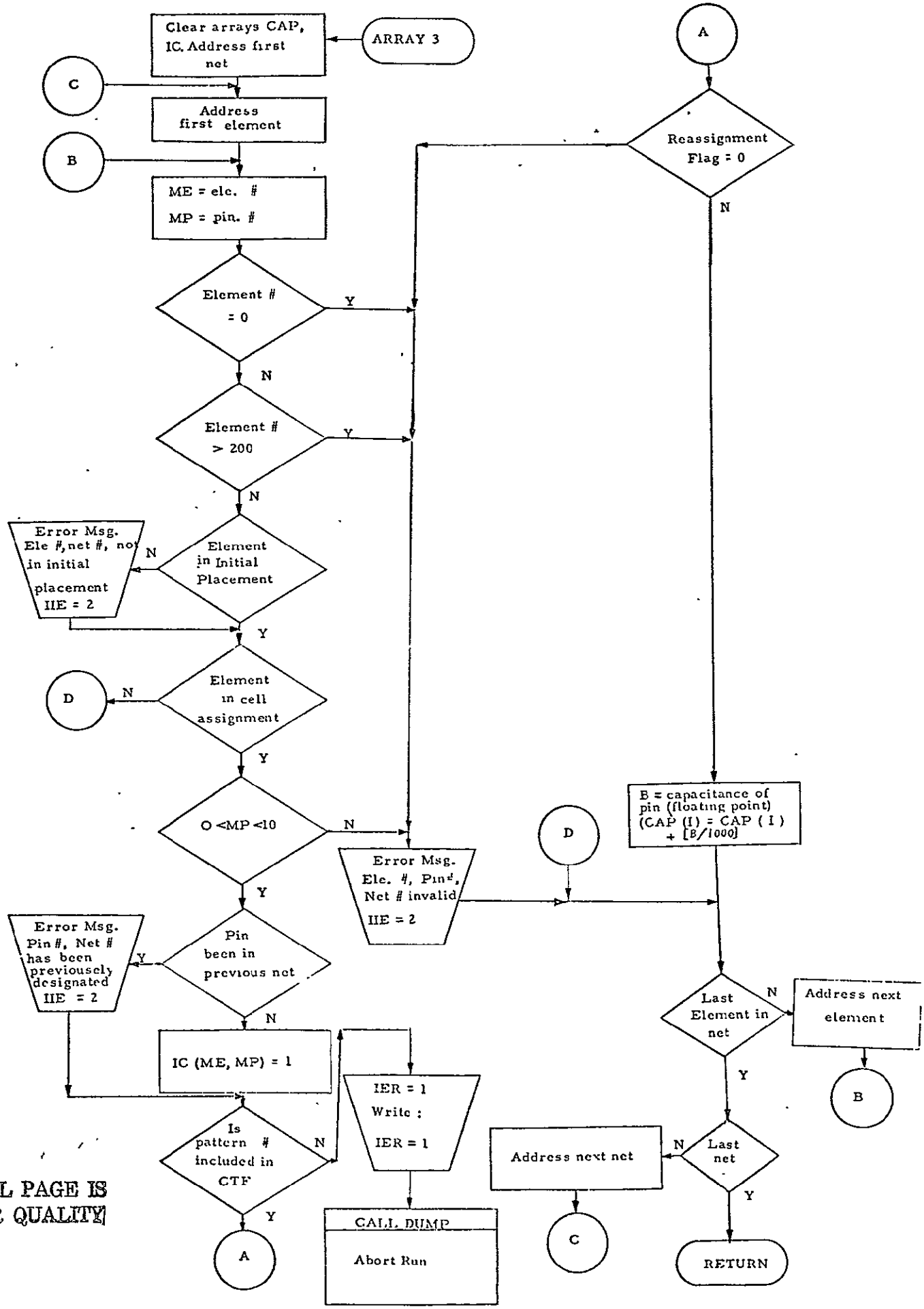


Figure 3-6 ARRAY 2



ORIGINAL PAGE IS
OF POOR QUALITY

Figure 3-7 ARRAY 3

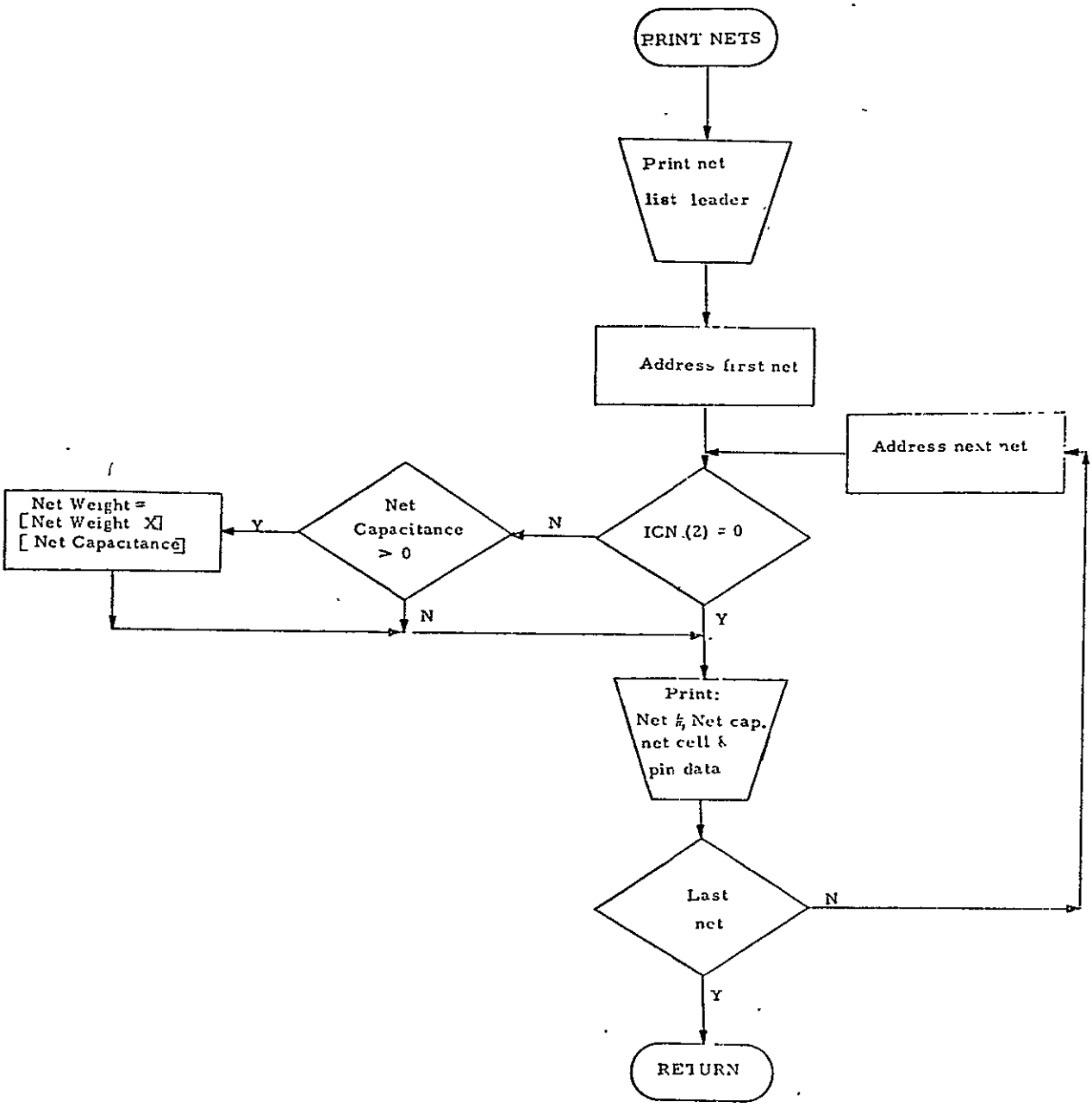


Figure 3-8 PRINT NETS

3.2 Subroutine PLACX

PLACX is used to reduce the length of cell interconnections by changing the relative positions of the cells in the placement array specified by the user.

3.2.1 Functional Description

Since examination of all possible combinations of cell positions would require a prohibitive amount of computer time, the subroutine attempts to optimize placement by running the designer's initial placement through a complex iterative loop which makes desirable cell interchanges until it can detect no more possible improvements. Key variables controlling this iteration sequence are LE and KSW. They are initialized so as to route the first pass through the flow block labeled SQUEEZ IP before any calculations or cell interchanges are made. Here IPTMP, which has been loaded with the user's initial placement from IP, is formatted into a more nearly square array, having a Y-dimension of 6 (see Table 3-4). The array will be constructed so that there will be fewer non-zero elements in IPTMP than there are positions, and these will be blocked near the array's center.

Following this initialization, the program enters a loop where the effect of each of a series of cell position interchanges is evaluated. This is accomplished by calling the CRCALC subroutine to calculate a rough approximation to cell interconnection lengths of each net, based on relative cell positions in the IPTMP array. The total of these lengths is termed the pseudo-wire length.

The sequence in which different trial interchanges are evaluated is detailed on the INTERCHANGE flowchart, Figure 3-13. If one or more good interchanges were made in the interchange loop, it is recycled. This process continues until no more good interchanges can be found. Once this state has been reached, the key variables are tested and modified, and the complete interchange loop is recycled. Variables and switches of this complex optimization cycle, as listed in Table 3-5, sequence through the outer loop a total of 12 passes. On each of these passes, the short loop is recycled as many times as necessary to reach the no good interchange state. By tracing the variables of this table through the PLACX flowchart, the detailed path sequence of the loop can be determined.

It should be noted that N1, the Y-dimension of IPTMP, is decremented until it eventually returns to the original IP dimension of 2. When the loop

IPTMP(I, J)

Format IPTMP-P1

J= \ I=	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	O	O	O	C	C	C	C	C	C	C	C	C	C	C	B	B	B	O	O	O
2	O	O	O	B	B	B	C	C	C	C	C	C	C	C	C	C	C	O	O	O
3	O	O	O	C	C	C	C	C	C	C	C	C	C	C	B	B	B	O	O	O
4	O	O	O	B	B	B	C	C	C	C	C	C	C	C	C	C	O	O	O	O
5	O	O	O	C	C	C	C	C	C	C	C	C	C	B	B	B	O	O	O	O
6	O	O	O	B	B	B	C	C	C	C	C	C	C	C	C	C	O	O	O	O

B = Bonding Pad Number
C = Cell Number

MAPTMP(I, J)

Format MAPTMP-P1

J= \ I=	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
6	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ABOVE ARRAY MAPS OF STATUS IMMEDIATELY AFTER RETURNING FROM SQUEEZ IP CODING BLOCK.

3rd element of each net group contains index of the last cell in that net group.

M=	1	2	3	4	5	6	7	8	8	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
NET(M)	OW_1	NW_1	11	W_1	C	P	C	P	C	P	C	P	OW_2	NW_2	25	W_2	C	P	C	P	C	P	C	P	C	P	OW_3

Format NET-P1

OW_1 = Old pseudo wire length of Net 1.
NW_1 = New pseudo wire length of Net 1.
W_1 = Weighting factor assigned to Net 1.
C = Any cell number.
P = Any pin number

Table 3-4 PLACX ARRAYS

OUTER LOOP PASS COUNT	SHORT LOOP PASS COUNT	PARAMETERS								
		LE	NOINT	KSW	ITOT	INGG	INEE	M_1	N_1	LIMINT
1	2	6	361	2	0	0	361	7	6	3
2	2	5	919	1	0	0	819	7	7	4
3	3	5	1241	2	0	0	1028	7	5	4
4	4	4	1745	1	0	0	1357	7	6	6
5	5	4	2096	2	0	0	1503	9	4	6
	6	3	2562	1	100	3*	1642	7	4	9
6	7	3	2793	1	0	3	1716	7	4	9
	8	3	3066	2	0	4*	1759	11	3	9
7	9	3	3358	2	0	4	1805	11	3	9
	10	2	3740	1	300	7*	1838	11	3	13
	11	2	3937	1	400	8*	1870	11	3	13
8	12	2	4208	1	200	10	1905	11	3	13
9	13	2	4505	1	100	10	1935	11	3	13
	14	2	4647	2	1100	11*	1940	13	2	13
	15	2	4783	2	1000	12*	1943	13	2	13
10	16	2	4936	2	1000	12	1946	13	2	13
	17	1	5025	2	2900	17*	1963	9	2	13
	18	1	5124	2	2700	19*	1981	9	2	13
11	19	1	5245	2	2700	19	1996	9	2	13
12	20	0	5366	2	4700	19	2012	9	2	13

*Note that each pass in which a good interchange was made (indicated by an increment in the value of INGG), the loop took the short return path, causing the control variables to remain unchanged.

The parameters of this table were printed by the debug print option near the end of the short loop while a small 18 cell test chip was being run.

PLACX PARAMETERS PRINTED AT COMPLETION OF CELL INTERCHANGE LOOP

Table 3-5

has sequenced to this point and when KSW=1, the elements of IPTMP are again segregated into rows of logic cells and pad cells. The original MAP array is restored before the final passes through the interchange loop to maintain this segregation. When good interchanges can no longer be made and LE has sequenced down to zero, PLACX prints the pseudo-wire lengths of each net and returns with the new cell placement stored in IP.

3.2.2 PLACX Variables and Arrays

The following key variables are referenced in PLACX:

INEE	-	Equal interchange count
INGG	-	Good interchange count
IELL	-	Left cell of interchange used as argument in calling CRCALC.
IELR	-	Right cell of interchange used as argument in calling CRCALC.
INTE	-	Equal interchange flag
INTG	-	Good interchange flag
ITOT	-	Total of net pseudo-wiring lengths
IX	-	X-Index on base cell
IY	-	Y-Index on base cell
KSW	-	Sequencing switch
LE	-	Length element sequencing variable
LIMINT	-	Limit on the distance from base cell within which cells will be tested for interchange.
M	-	X-Index on interchange candidate

M1 - X-Dimension of IPTRN

M1S - Number of logic cells

N - Y-Index on interchange candidate

N1 - Y-Dimension of IPTRN

NOINT - Number of interchanges count

The following arrays are either formed by PLACX or altered from their initial configuration in the INPUT subroutine:

IC(I, J) - Rows 1, 2, and 3 are used for temporary storage of MAPTMP and IPTMP.

IPTMP(I, J) - Array used to indicate relative positions of cells. IPTMP and MAPTMP are loaded from IP and MAP on entry to PLACX and their first two rows are restored in these arrays on return.

MAPTMP(I, J) - Map denoting legal interchange positions. All rows corresponding to IPTMP rows which contain pads or cells are initially set to 1, enabling interchanges between pads and cells. After final reformation of IPTMP, the original map array (as shown in the INPUT array maps) is entered. Since interchanges between only those elements having equal non-zero MAPTMP references are allowed, the final format of IPTMP is preserved.

NAT(I) - Used to temporarily store the elements of IPTMP.

NET(M) - Identical to format formed by INPUT with the addition of OW and NW for each net as shown in Table 3-4.

NETTOT(I) - Array containing the total pseudo-wire length of each net I.

3.2.3 PLACX Flowcharts

Figures 3-9 through 3-13 contain flowcharts of the PLACX subroutine. Flowcharts of the external subroutine CRCALC, which is referenced by PLACX, are presented in Section 3.3.3.

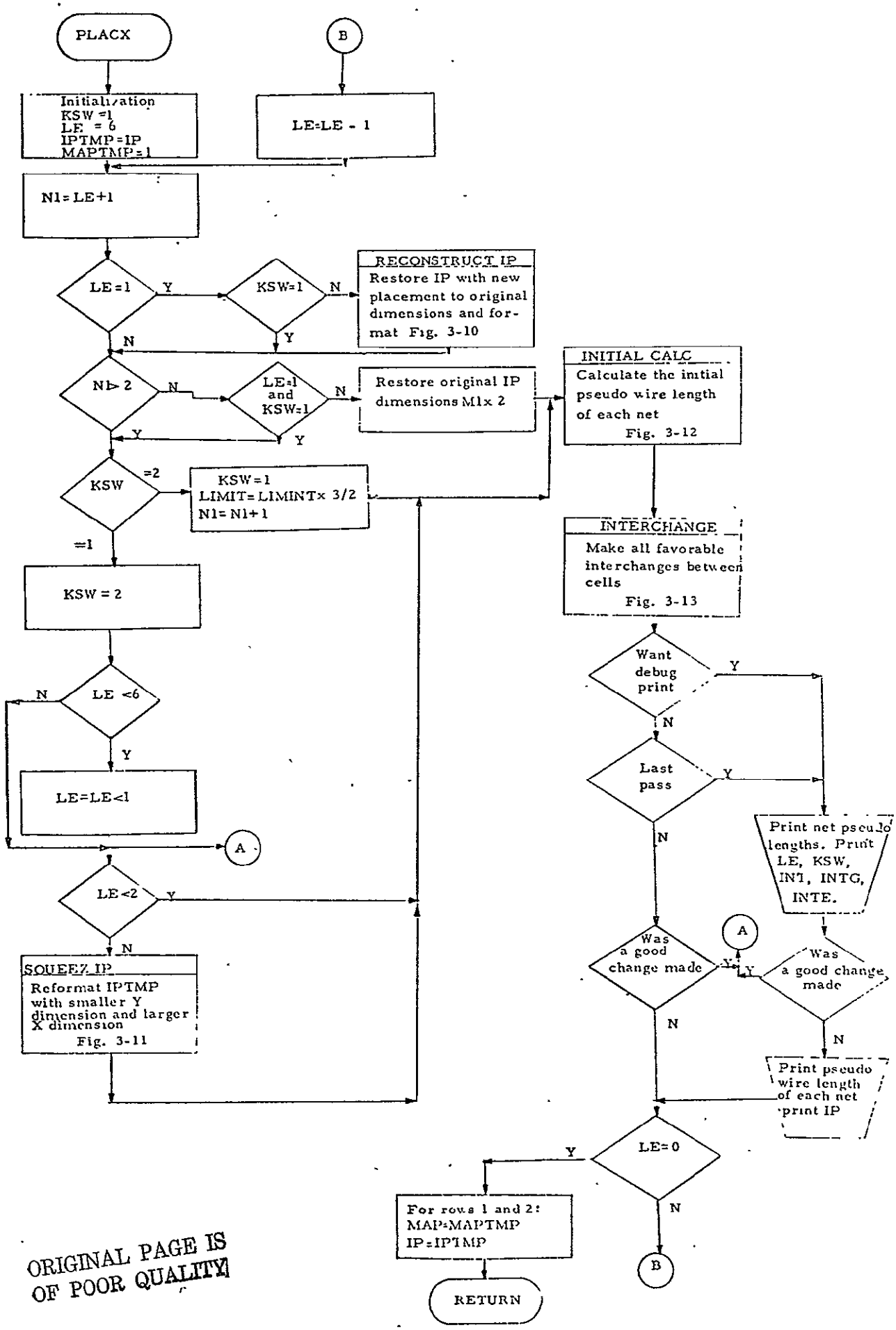


Figure 3-9 PLACX

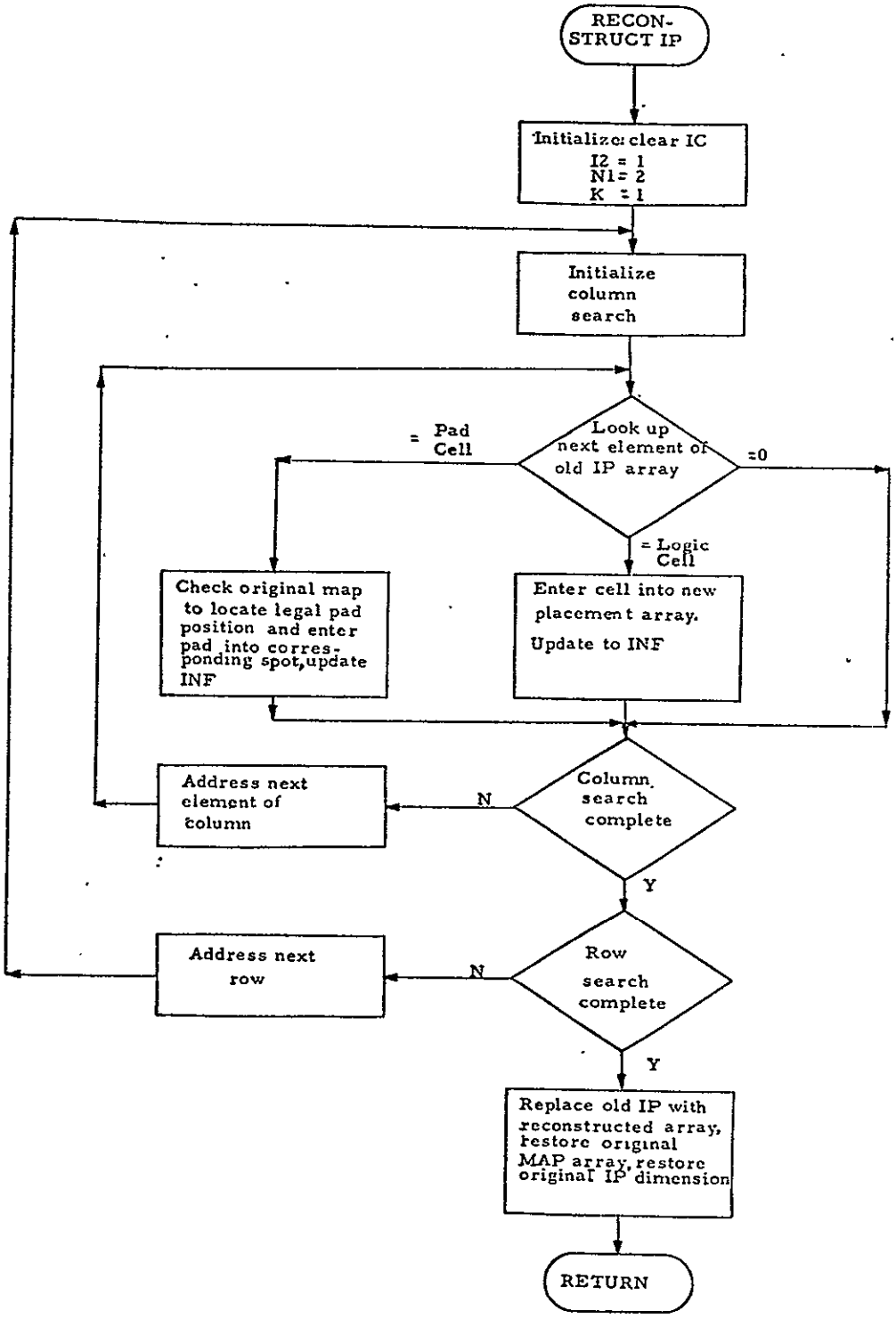


Figure 3-10 RECONSTRUCT IP

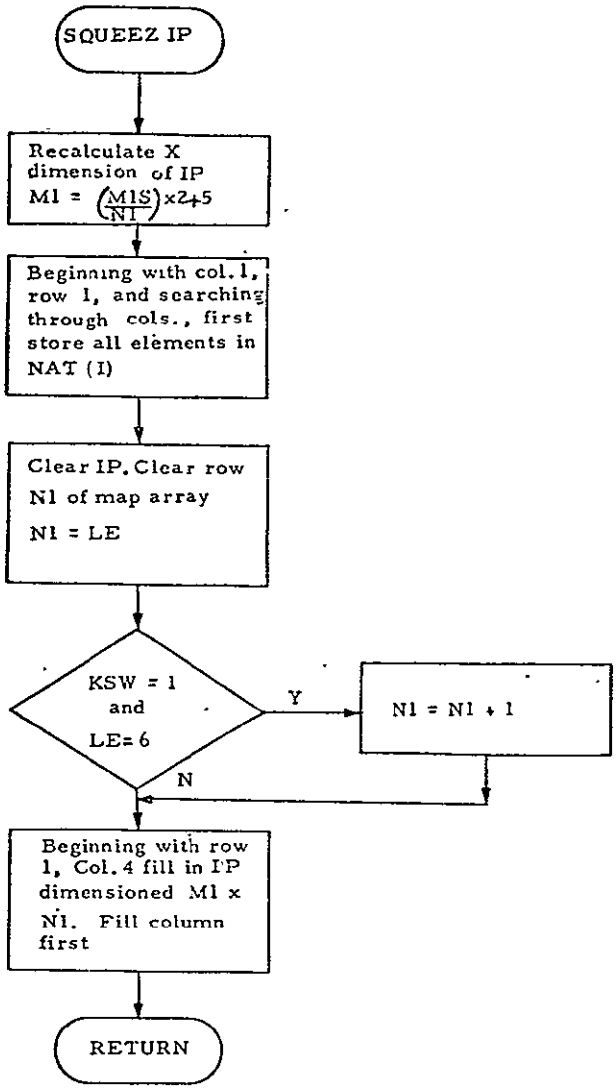


Figure 3-11 SQUEEZ IP

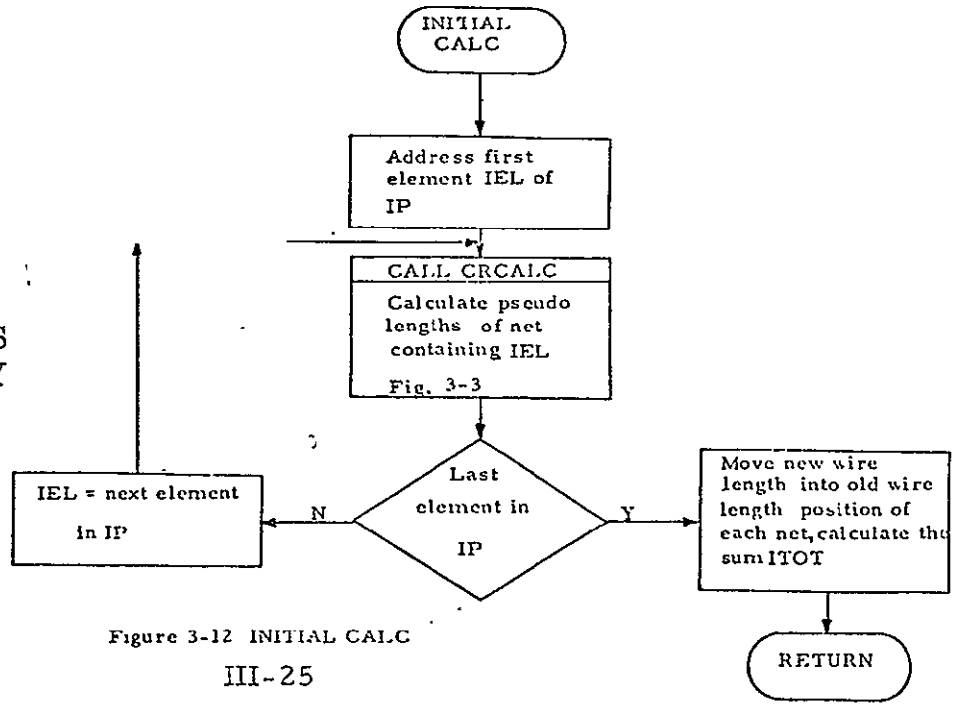


Figure 3-12 INITIAL CALC

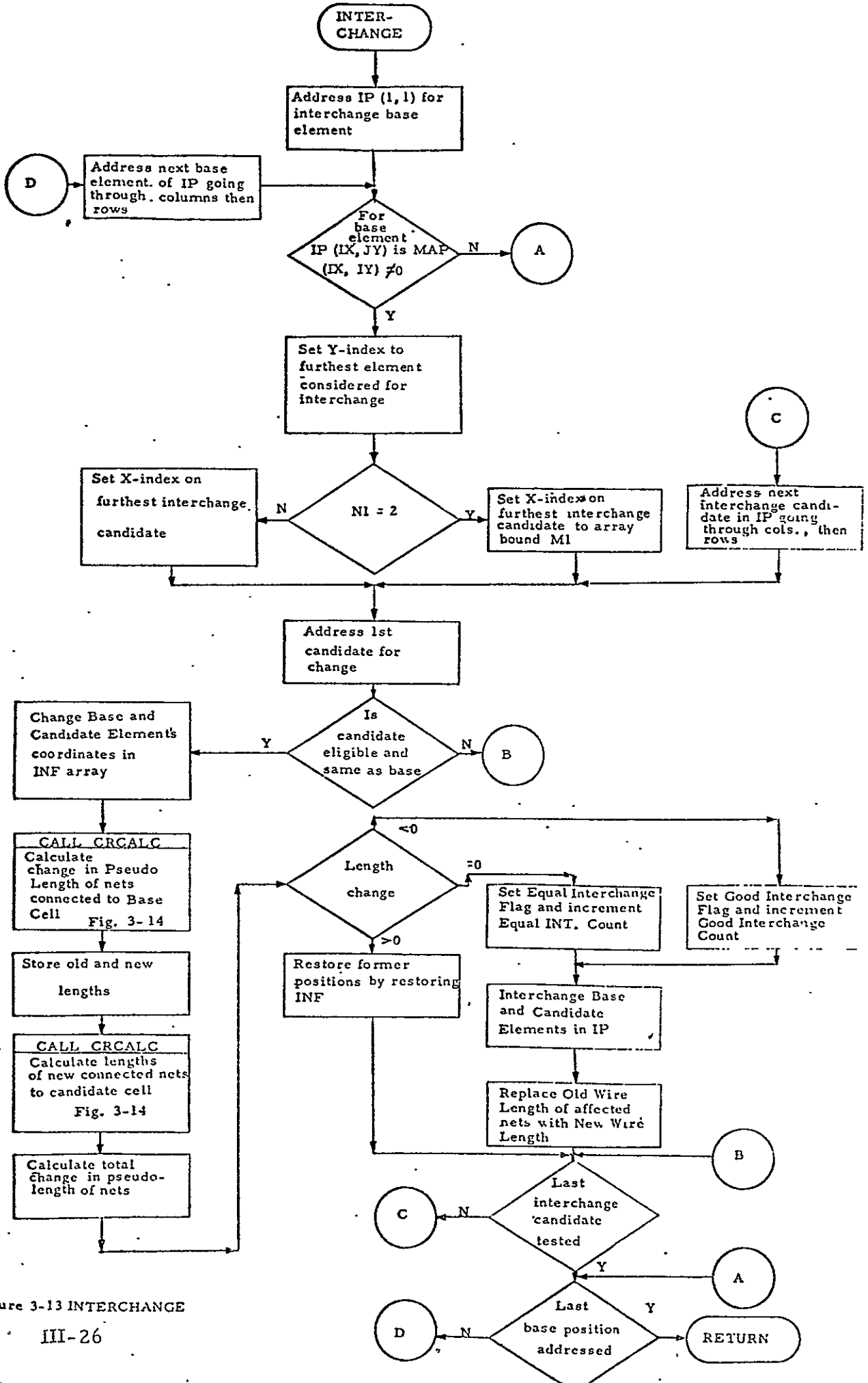


Figure 3-13 INTERCHANGE

3.3 Subroutine CRCALC

PLACX calls this subroutine to make the wire length calculations on which cell interchanges are based.

3.3.1 Functional Description

Common variable IEL is set by PLACX to an element number which CRCALC is to use as its primary argument. The old pseudo-wire lengths, as set in NET (see Table 3-4) by PLACX, and the new pseudo-wire lengths, as calculated by this routine, of each net containing cell IEL are added to form the total new pseudo-wire length, IN. These are the common variables passed back to PLACX for use in making the interchange decision.

The new pseudo-wire length of a net can be calculated by either of two subroutine branches, one using a fast and simple method for 2 pin nets, and another using a more general method for more than two pins. In either case, the calculation begins by determining the maximum index separation between the net's elements in array IP for both the X and Y dimensions. The variable LE, which is varied by PLACX, is then subtracted from both of these values. If either value is negative, it is set to zero before being added to the other value to form the new pseudo-wire length of the net, NW. If LE=0, NW is limited to 20. The only case in which NW is a true measure of a net's span in IP is when LE=0.

3.3.2 CRCALC Variables and Arrays

The following list contains important local variables of the CRCALC subroutine:

ID	-	Address of last net index in LNAD which references IEL.
IEL	-	Argument element of IEL for which length is calculated.
IMN	-	Net array index on the cell whose connection length is being added to the total net length.
IN	-	New wire length for element IEL.
IO	-	Old wire length for element IEL.

IW	-	Net weight of current object net.
JD	-	Address of first net index in LNAD which references IEL.
KD	-	Index to LD in LNAD.
LD	-	Variable which points to the net for which lengths are currently being calculated.
LD24	-	Index on new wire length of net in net array.
LE	-	Minimum segment of X or Y distance in IP which will be summed in calculation of wiring length.
NEX	-	X-Span of wiring length in placement array.
NEY	-	Y-Span of wiring length in placement array.

All arrays referenced by CRCALC are in a format identical to that described for the PLACX subroutine in Section 3.2.2.

3.3.3 CRCALC Flowcharts

Figure 3-14 contains the CRCALC subroutine flowchart.

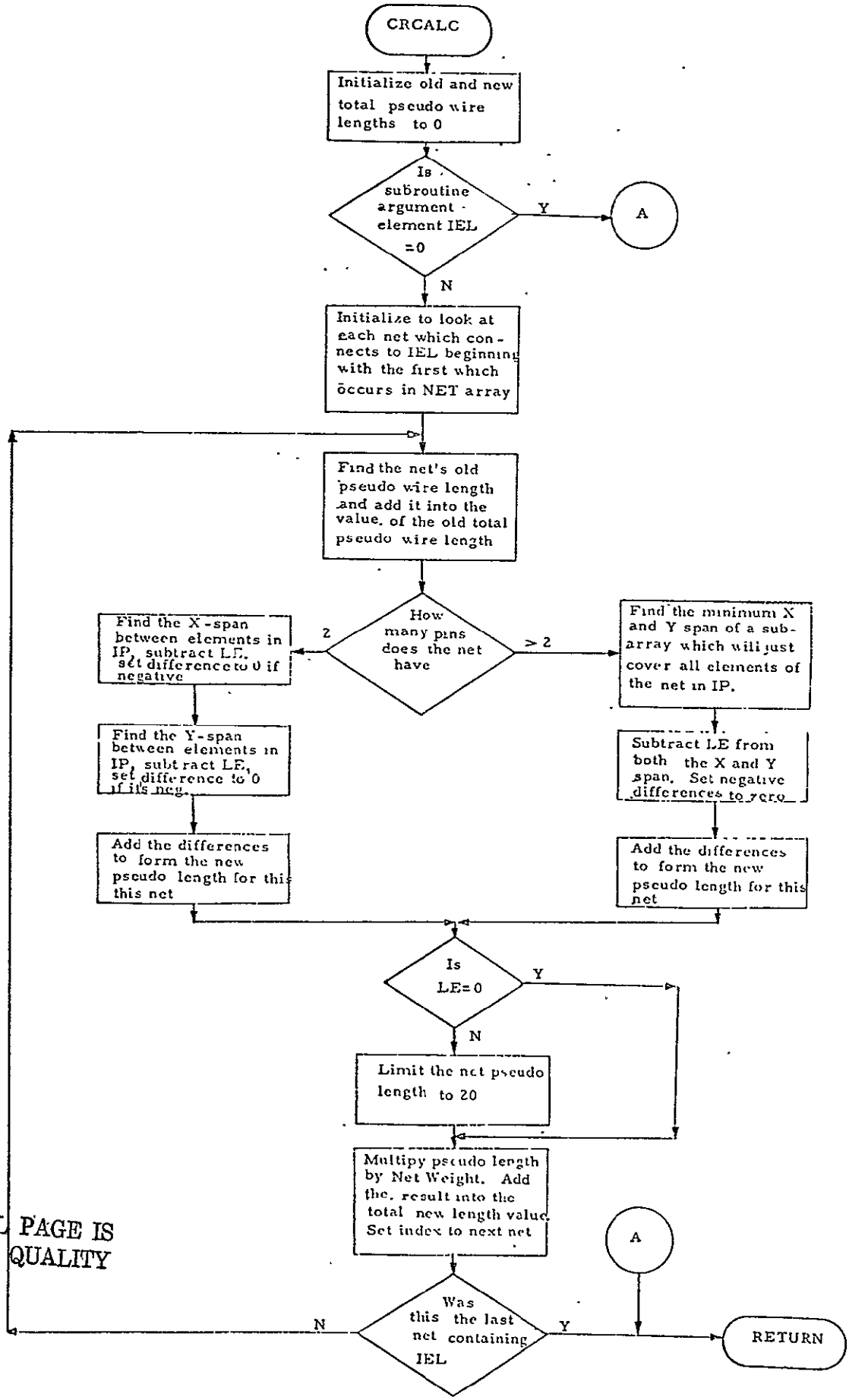


Figure 3-14 CRCALC

3.4 Subroutine ROUTE

The primary function of the ROUTE subroutine is to form a map of the interconnections between cells specified by the NET array. These interconnections are made by "unfolding" the relative positions of the cells in the IP array and assigning each pin of each cell in this placement a position along an edge column of a linear chip map in array IRA (Table 3-8). Connections between the pins are then made by entering codes into the array elements which separate the pins, denoting the location of P-material, tunnel segment, wire segment, or both.

3.4.1 Functional Description

Before the actual routing function takes place, several secondary tasks are performed by the subroutine. The first of these is the optimization of pad placement, occurring on the first pass through ROUTE. Flags are set during this first pass so that a large bias length is added to the estimated connection length of each net containing a pad. After these lengths have been calculated and sorted on length, control is switched to coding which clears the initial pad placement and shuffles pad positions so that on either end of the linear array, those pads having longer connections to logic pins are placed farther from the array's center.

When this has been completed, the subroutine starts anew, calculating connection lengths for the end elements of each net and using these to determine which cells should be inserted by having the order of their pins in the linear array reversed. Each cell is also examined to reassign pins by interchanging the connections of functionally identical logic pins when the connection length can thereby be reduced. Upon completion of desirable pin interchanges, the NET array and others are updated to reflect these changes.

The horizontal wire length required for each connection between pins is now calculated, and the NC array describing each connection is formed and sorted on increasing length. The actual running of connections between pins mapped in IRA is made from the list of connections in NC, shortest first for most efficient utilization of wiring area. Horizontal segments of the connection are run by searching into the wiring area from the connection's left pin until a clear channel is found to the right pin. Elements of this channel are then filled with the net number of the connection to indicate a horizontal wire has been run. The vertical connections between the left and right pins and this horizontal channel are now entered by filling the intervening elements with either a 1, to indicate vertical metal, or a 999, indicating a P-material

tunnel under a previously run horizontal wire. Note that all horizontal connections will be metal, while vertical connections may be either P-material or metal.

After every connection has been run another secondary function, that of determining the proper spacing between cells, is performed. This is done by adding the specified pad to pad spacing to the running length of pins following pad interfaces, and adding the spacing calculated by the SPACE subroutine to running lengths following all other interfaces.

After a pass through FOLD, MAIN calls ROUTE for a second time. During this second pass, flags are set to bypass pad optimization and to reroute all wiring taking advantage of knowing the points at which the array is to be folded. The common variable IJ is referenced during this second pass to determine the number of folds, while array IF (Table 3-6) is referenced for the X-indexes in IRA at which the array is to be folded. By utilizing these indexes, each fold is now wired independently, resulting in a saving of horizontal channels over the wiring on the first pass.

3.4.2 ROUTE Variables and Arrays

The following list contains key variables of the ROUTE subroutine:

IAD - Running length of pins in linear array.

IB - X-index on right pin of horizontal wire segment in IRA.

ID - Number of pins of an element being considered for inversion.

II - X-index on first logic pin of element in IRA.

IJ - Number of folds.

IK - Index on number of elements in IRA.

IM - Number of net end point connections.

IN - Flag which indicates whether FOLD subroutine has been run.

For: IN=1 - FOLD has not been run
IN=2 - FOLD has been run

INC - Inversion coding increment.

For: INC=1 - Inverted cell
INC=2 - Normal cell

IPAD - Pattern number of a pad.

IQ - Index on vertical wiring channel.

IQ1 - Vertical channel wiring switch.

IR - Flag denoting whether pad placement has been checked to put pads with longest connections on outside of linear array.

For: IR=1 - Check has not been completed
IR=2 - Check completed and pads moved accordingly.

IRAMX - X-index on last pin in IRA.

IRX - Maximum X-dimension of IRA.

IRY - Maximum Y-dimension of IRA.

IRYY - Set to (IRY-1).

IS - Space between adjacent cells calculated by SPACE Subroutine.

ISW - Inversion coding flag.

For: ISW=1 - Normal cell processing
ISW=2 - Inverted cell processing

IT - X-index on left pin of horizontal wire segment in IRA.

I5 - Y-index on column in which horizontal wire is to be run.

JM - Flag indicating status of interfold wiring.

KPK - Number of elements in column 1 of NC array.

K10 - During routing - number of connections to route.

K12 - Location of true bottom pin number where connection wire terminates on fold edge.

NN - Net number of wire being run.

NNETS - Number of nets specified by user.

NSTRT - Index on first channel to be checked to run horizontal wiring.

The following arrays are constructed or modified by the ROUTE subroutine. The more complex arrays are mapped in Tables 3-6 through 3-9.

IRA(I, J) - Format IRA-R1:

A linear list of all the pins of all the elements in their proper sequence is constructed in the first four columns in format IRA-R1. The running length in column 1 contains the physical distance in tenths of mils which would separate a common origin at the top of the array and the associated pin. Except for the shuffling of pins within cells during cell inversion and pin reassignment, this list remains unchanged through the ROUTE and FOLD routines. Following the first pass through FOLD, the fold points are flagged during ROUTE wiring by setting the last element number of each fold negative.

- Format IRA-R2:

Columns 5 through 20 are used as temporary storage of various indexes, flags, and connection lengths during the ordering of connections, inversion of cells, and reassignment of pins. Table 3-8 shows the details of this format. This storage area is cleared when the wiring map, Format IRA-R3 is constructed.

- Format IRA-R3:

This map specifies the route to be followed and the types of conductors utilized by preliminary connections between cells.

IP(I, J)
Format IP-I1

J= \ I=	1	2	3	4	5	6		30	31	32	33	34
1	C_1	C_2	C_3	C_4	O	C_6		C_{30}	C_{31}	C_{32}	C_{33}	C_{34}
2	P_1	P_2	O	O	O	O		O	O	P_{32}	P_{33}	P_{34}

NC(I, J)
Format NC-R1

J= \ I=	1	2	3	4	5	6	7	8		30	31	32	33	34	35	36	37	38
1	P_2	P_1	C_1	C_2	C_3	C_4	C_6	C_7		C_{29}	C_{30}	C_{31}	C_{32}	C_{33}	C_{34}	P_{34}	P_{33}	P_{32}
2																		
3																		
4																		
5																		

IC(I, J)
Format IC-R1

J= \ I=	1	2	3	4	5	6						
1		17		9	1							
2		21		13	5							
3		35		17	9							
4		49										
5		53										
6												
7												
8												
9												
10												

Pin# IC(I, J) contains running length corresponding to Pin J of Element I.

IRA(I, J)
Format IRA-R1

J= \ I=	1	2	3	4	5	6	7	8	9	10	11	12	13					
1	1	1	5	9	9	13	17	17	21	35	49	53						
2	O	E_1	E_1	E_1	E_2	E_2	E_2	E_3	E_3	E_3	E_3	E_3						
3	O	1	2	3	1	2	3	1	2	3	4	5						
4	O	O	346	O	O	346	O	O	290	296	291	O						

Pad P_2 Pad P_1 $E_3=C_1=2070$

Table 3-6 ROUTE ARRAYS

IC(I, J)
Format IC-R2

J= \ I=	1	2	3	4	5	6	7	8	9	10	11	12
1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	4	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3	3	3	3
4	4	4	2	4	4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5	5	5	5	5

The pin sequence deviations show Pins 2 and 4 of Element 3 have been reassigned to one another.

NX(I, J)*
Format NX-R1

J= \ I=	1	2	3	4	5	6	7	8	9	10	11	12	
1													Vertical Length
2													Horizontal Length
3													# of Crossovers

NC(I, J)*
Format NC-R2

J= \ I=	1	2	3	4	5	6	7	
1								Net #
2								Index to upper pin
3								Index to lower pin
4								Connection length
5								X-Index of end of last fold containing net

IC(I, J)
Format IC-R3

J= \ I=	1	2	3	4	5	6
1	X	X	X	X	X	
2	X	X	X	X	X	
3	X	X	X	X	X	
4	X	X	X	X	X	
5	O	X	O	O	O	
6	O	X	O	O	O	
7	O	O	O	O	O	
8	O	O	O	O	O	
9	O	O	O	O	O	
10	O	O	O	O	O	

Element IC(I, J) contains the X-Index to Element I, Pin J in IRA.

*Final ROUTE output configuration.

Table 3-7 ROUTE ARRAYS

ROUTE SUBROUTINE ARRAY MAP - IRA(I, J)

I \ J	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		IRY
1	1	0	0	0	0	0		46	4	1	16	2	46	2	1		
2	1	102	1	0	0	0		46	2	1	36	1	46	4	1		
3	5	102	2	346	0	0											
4	9	102	3	0	0	0											
5	9	101	1	0	0	0											
6	13	101	2	346	0	0											
7	17	101	3	0	0	0											
8	17	46	1	0	0	0											
9	21	46	2	290	4	0											
10	35	46	3	296	0	0											
11	49	46	4	290	2	0											
12	53	46	5	0	0	0											
13																	
14																	
15																	
16																	
17																	
	Running length	Element #	Pin #	Reassignment flag	Pin reassignment in normal orientation	Pin reassignment for inverted orientation		Element # - position ordered	Pin # - position ordered	Up = 1, Down = 2	Length of connection	Connection index	Ordered Ele. # - length	Ordered Pin # - length	Up - Down		
IRX																	

End point connect list

Ele. #46 - 2070 Nor.
#101 - 9034 Pad
#102 - 9034 Pad

Format of Columns 1 - 4 referenced as Format IRA-R1.
Format of Columns 5 - 20 referenced as Format IRA-R2.

Table 3-8 ROUTE ARRAYS

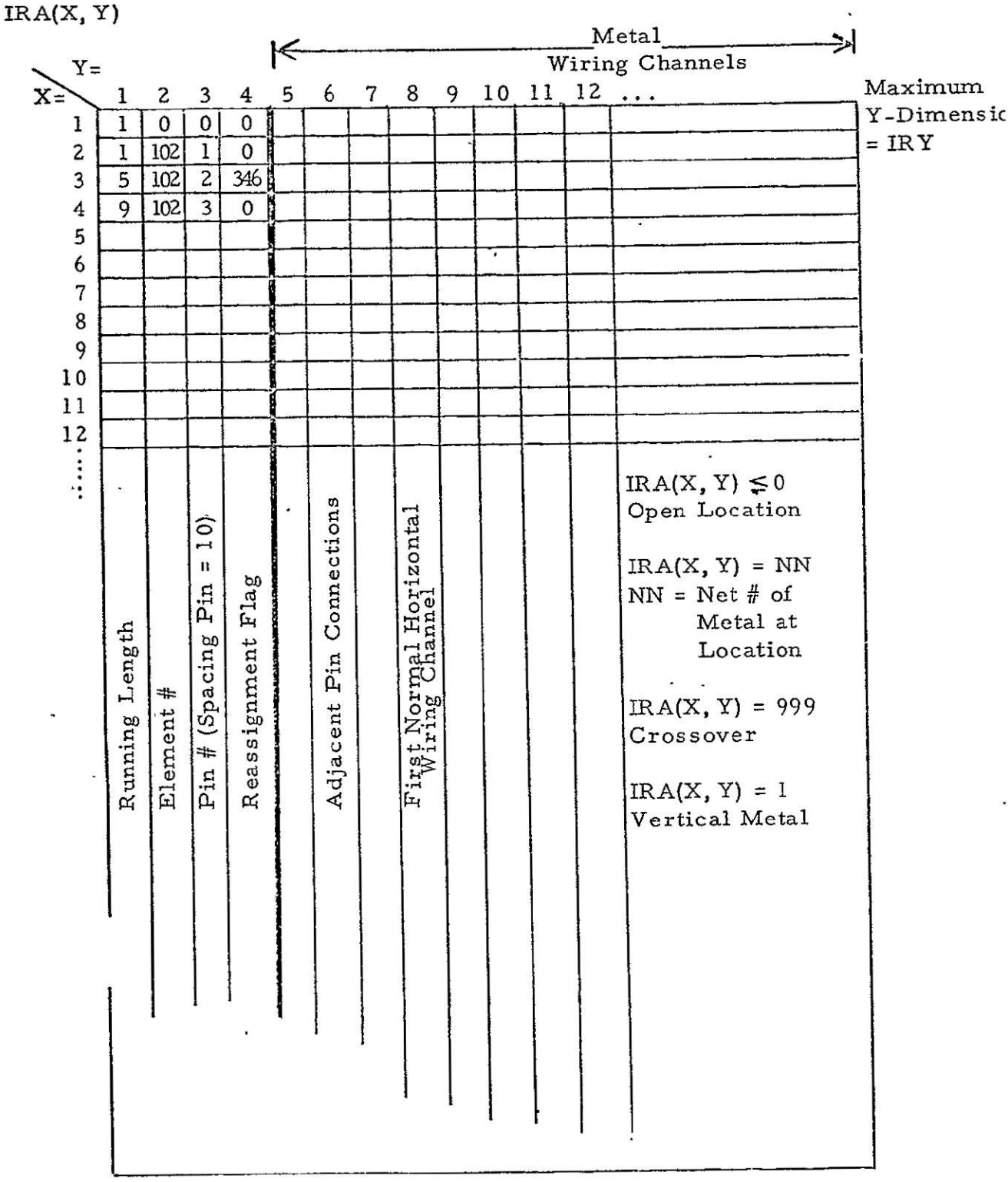


Table 3-9 ROUTE ARRAYS

For some element IRA(I, J):

IRA(I, J) = 0 - Position (I, J) is not used.
IRA(I, J) = 1 - Position (I, J) contains a vertical metal wire.
IRA(I, J) = N - Position (I, J) contains a horizontal metal wire of net number N, where $1 < N < 998$.
IRA(I, J) = 999 - Position (I, J) contains a vertical P-material wire.

JD(I) - JD(1) contains the total length of connections from a cell's pins to its edges for the cell in normal orientation.

JD(2) contains the total length of connections from a cell's pins to its edges for the cell in inverted orientation.

IP(I) - This array is not changed by ROUTE except for the interchanging of pads located on the same end of the array during pad optimization.

NET(I) - Pins which are reassigned are replaced by the new pin number. No other changes are made.

ISR(I) - Array used to transfer input parameters to SPACE subroutine.

ISR(I) = ICN(I+61) For I = 1, 2, 3.

IEC(I) - On entry to ROUTE, IEC is reconstructed to contain in each element, IEC(I), the index to the Circuit Type File arrays ICK, ICL, and IPTRN for cell I. Before ROUTE returns to MAIN, the original format is restored, in which IEC(I) contains the pattern number of cell I.

IC(I, J) - Format IC-R1:

Each element of IC(I, J) contains the running length assigned to pin J of cell I in IRA.

- Format IC-R2:

Element IC(I, J) contains the new pin number to which connections formerly made to cell I, pin J have been assigned. Thus IC(I, J) = J except when pin J has been reassigned.

NC(I, J) - Format NC-R1:

Column 1 contains the placement array elements in a linear, "unfolded" order (see map, Table 3-6).

- Format NC-R2:

Each row I contains information describing a connection between two pins.

For connection I:

NC(I, 1) = Net number to which the connection belongs.
NC(I, 2) = X-index to the left pin of the connection in IRA.
NC(I, 3) = X-index to the right pin of the connection in IRA.
NC(I, 4) = The horizontal length of this connection in tenths of mils. Connections in NC are eventually ordered on ascending length.
NC(I, 5) = Flag used when running wires after FOLD has been run to determine fold points. If the connection runs down past the end of a fold, the X-index on the end of that fold is entered into this location.

INT(I) - Used to pass cell edge data describing adjacent cells to SPACE subroutine. It is equivalent to the array IDT(I) described in Section 3.5.2.

NX(I, J) - Array used to accumulate the horizontal and vertical length and the number of crossovers encountered by each net. For net I:

NX(I, 1) = Length of horizontal wiring in mils X10.
NX(I, 2) = Number of vertical channels spanned by net's vertical wiring.
NX(I, 3) = Number of points at which this net has crossed over or under another net's connections.

3.4.3 ROUTE Flowcharts

Figures 3-15 through 3-23 contain the ROUTE subroutine flowcharts.

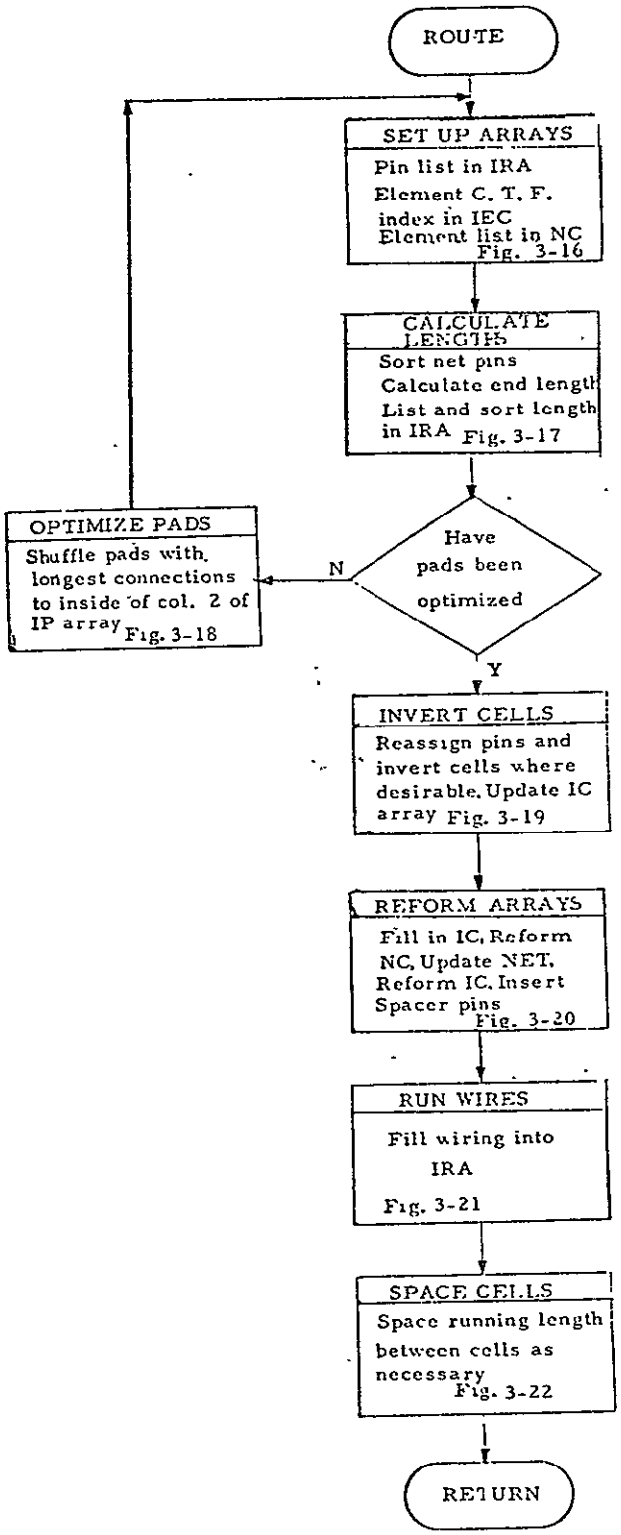


Figure 3-15 ROUTE

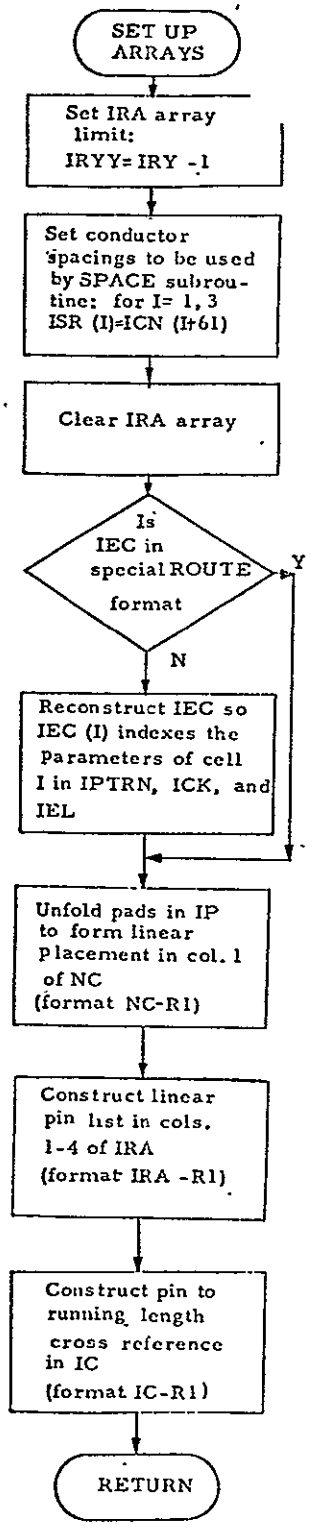


Figure 3-16 SET UP ARRAYS

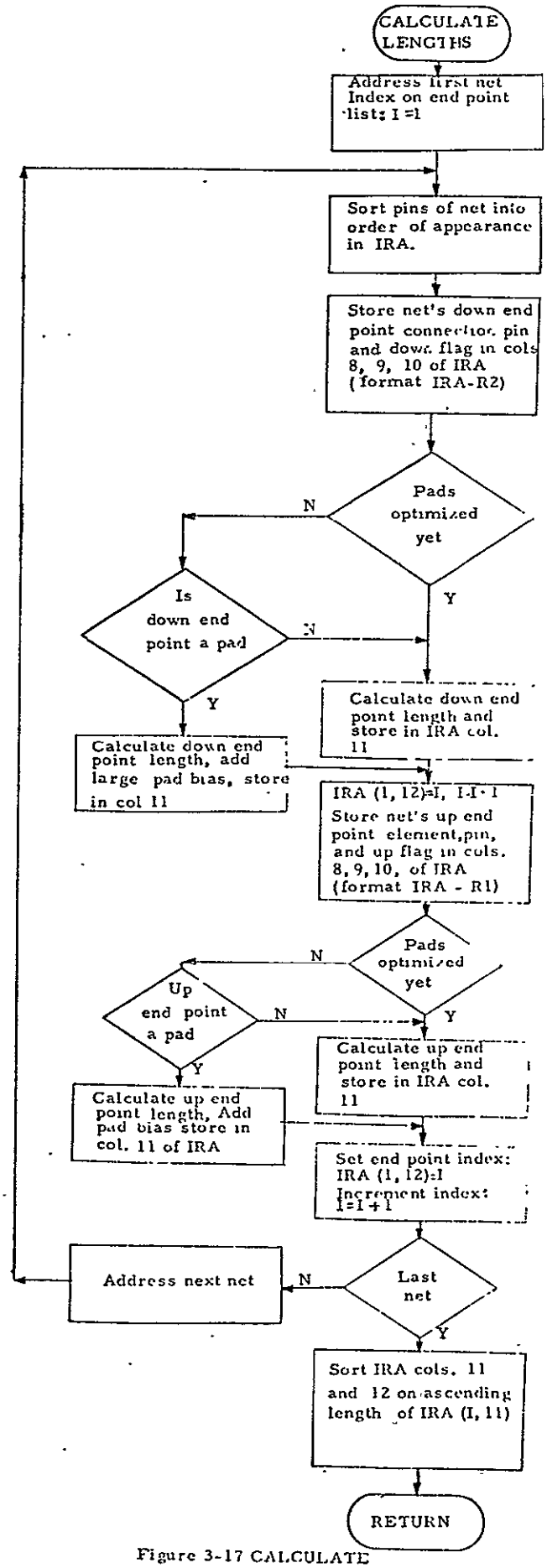


Figure 3-17 CALCULATE LENGTHS

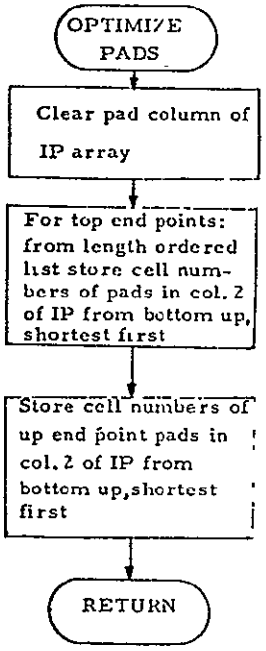


Figure 3-18 OPTIMIZE PADS

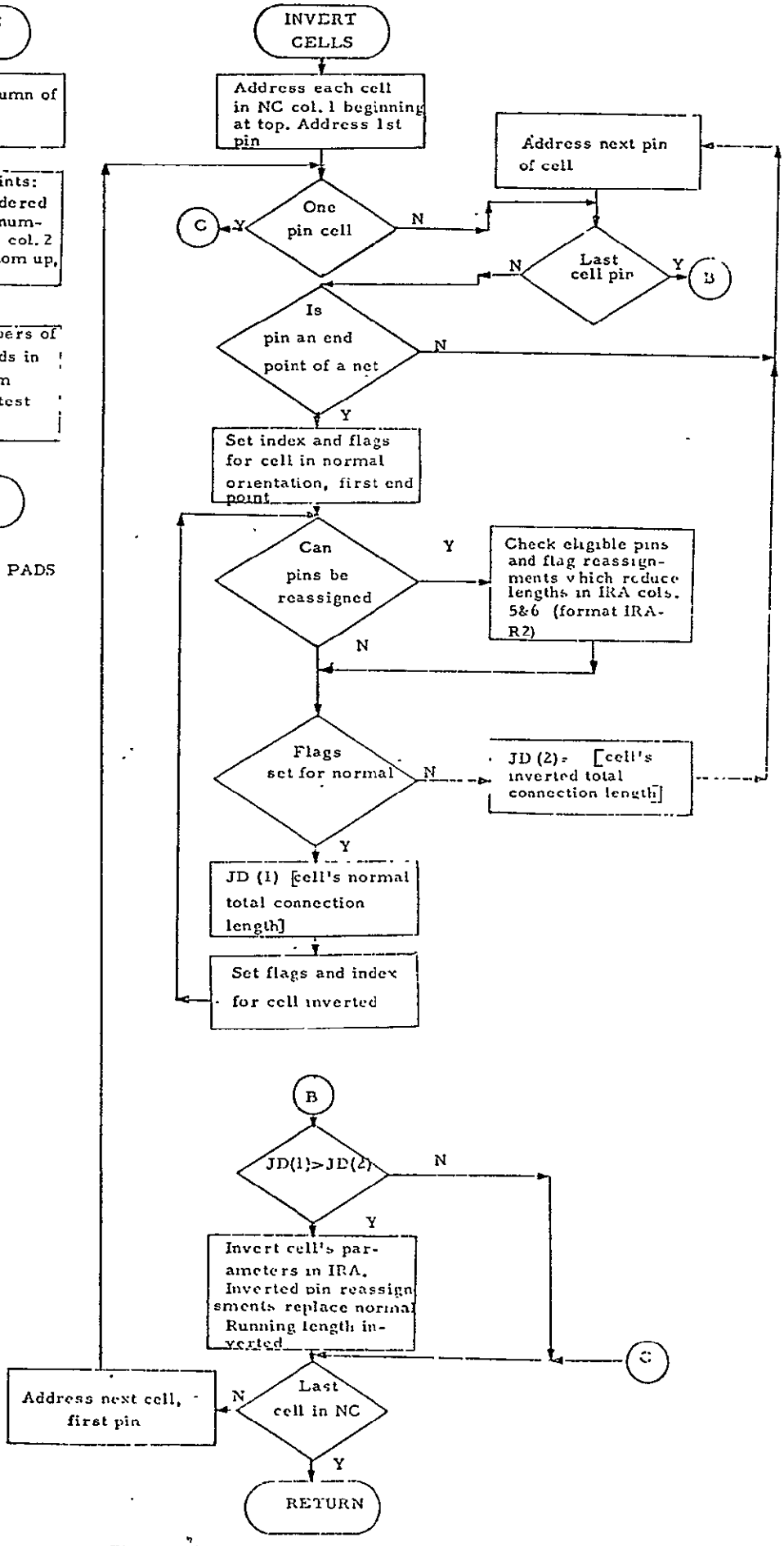


Figure 3-19 INVERT CELLS

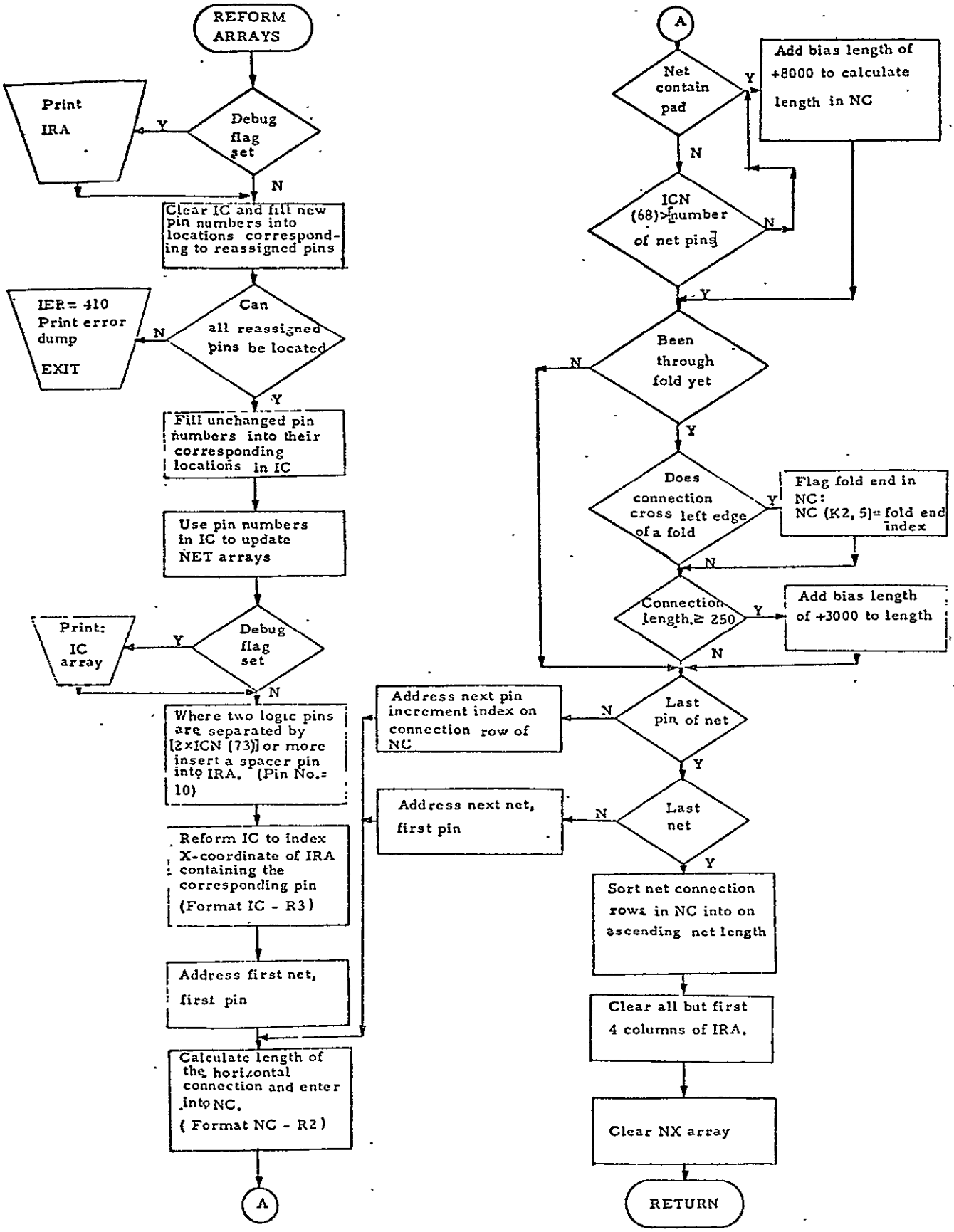


Figure 3-20 REFORM ARRAYS

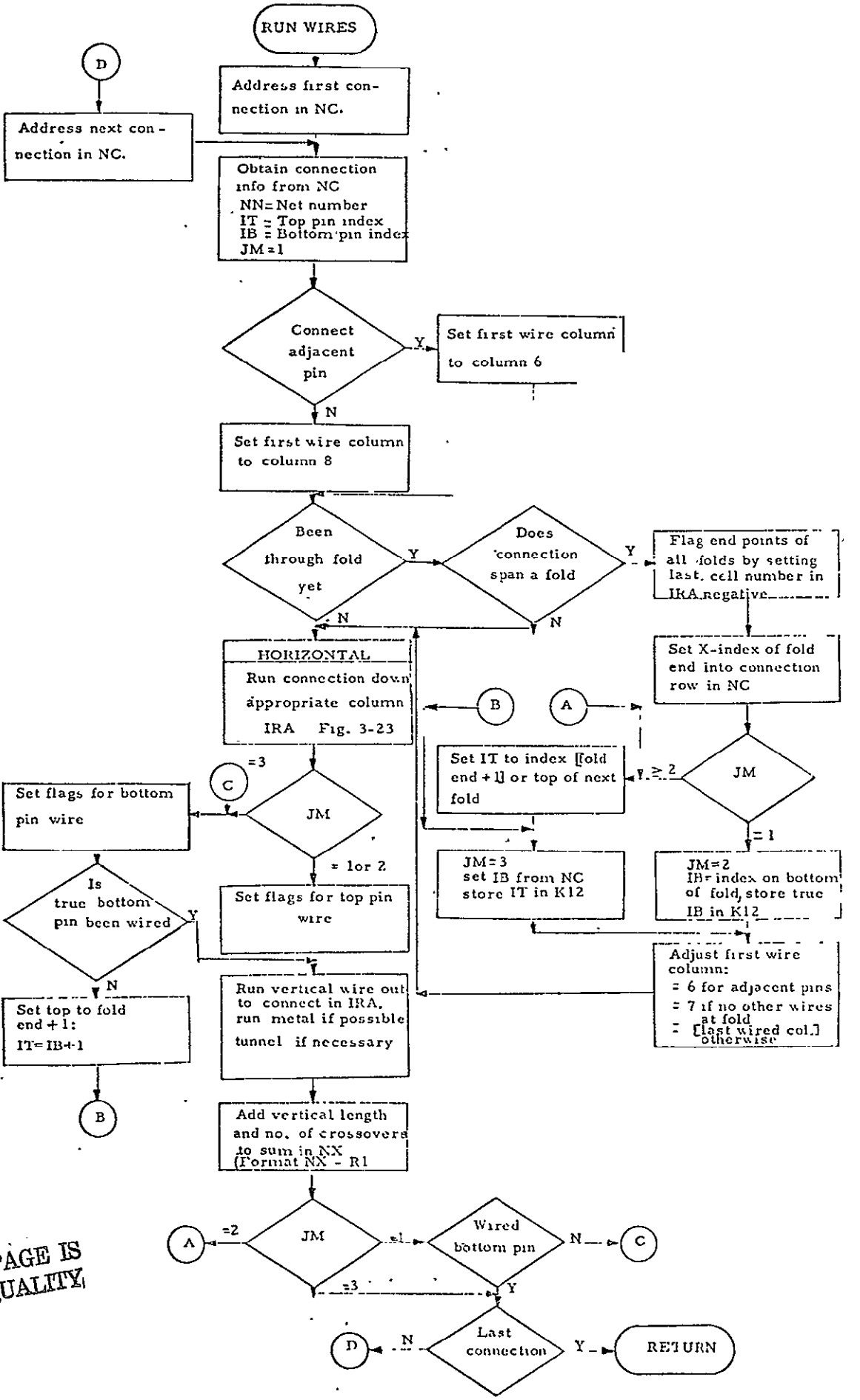


Figure 3-21 RUN WIRES

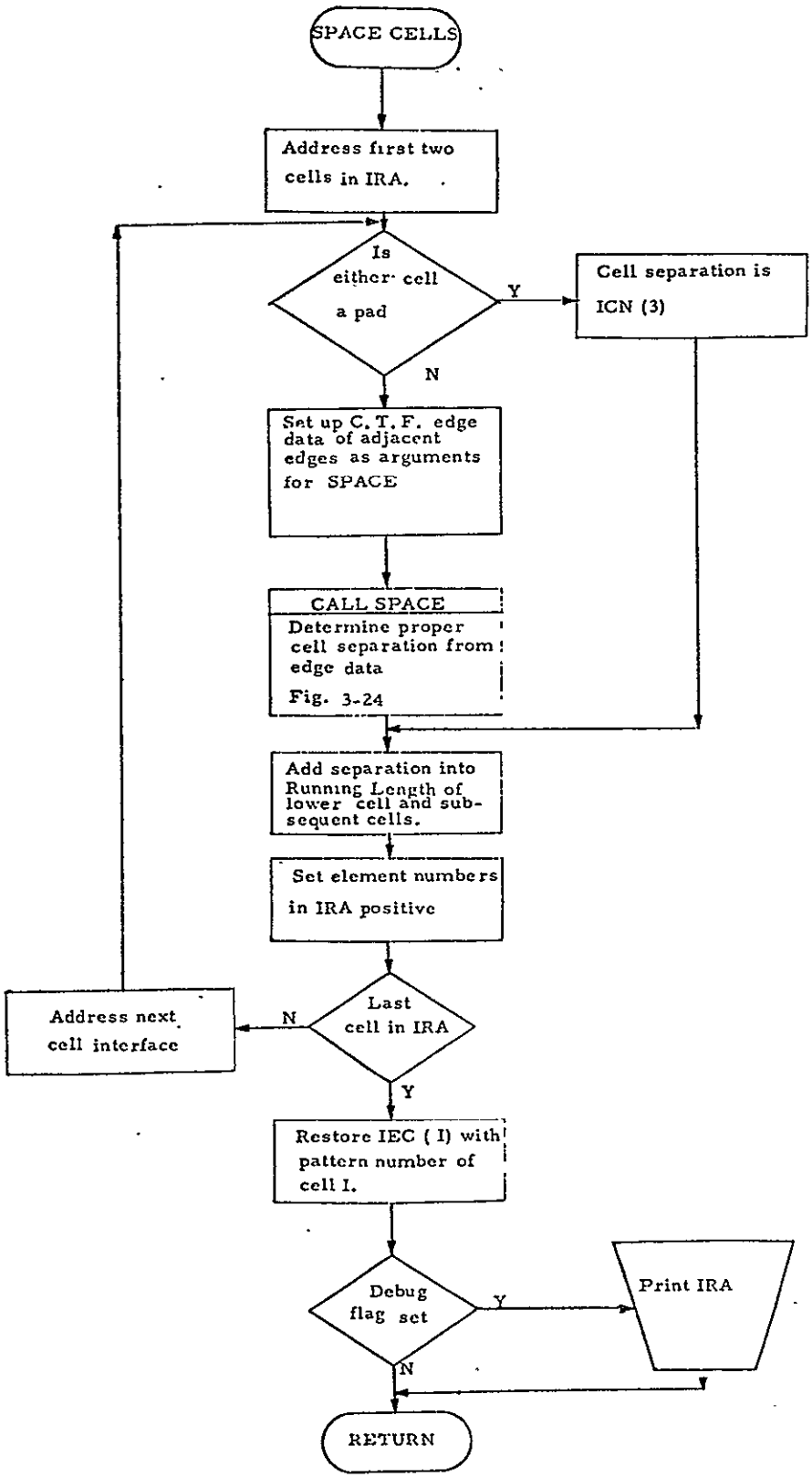


Figure 3-22 SPACE CELLS

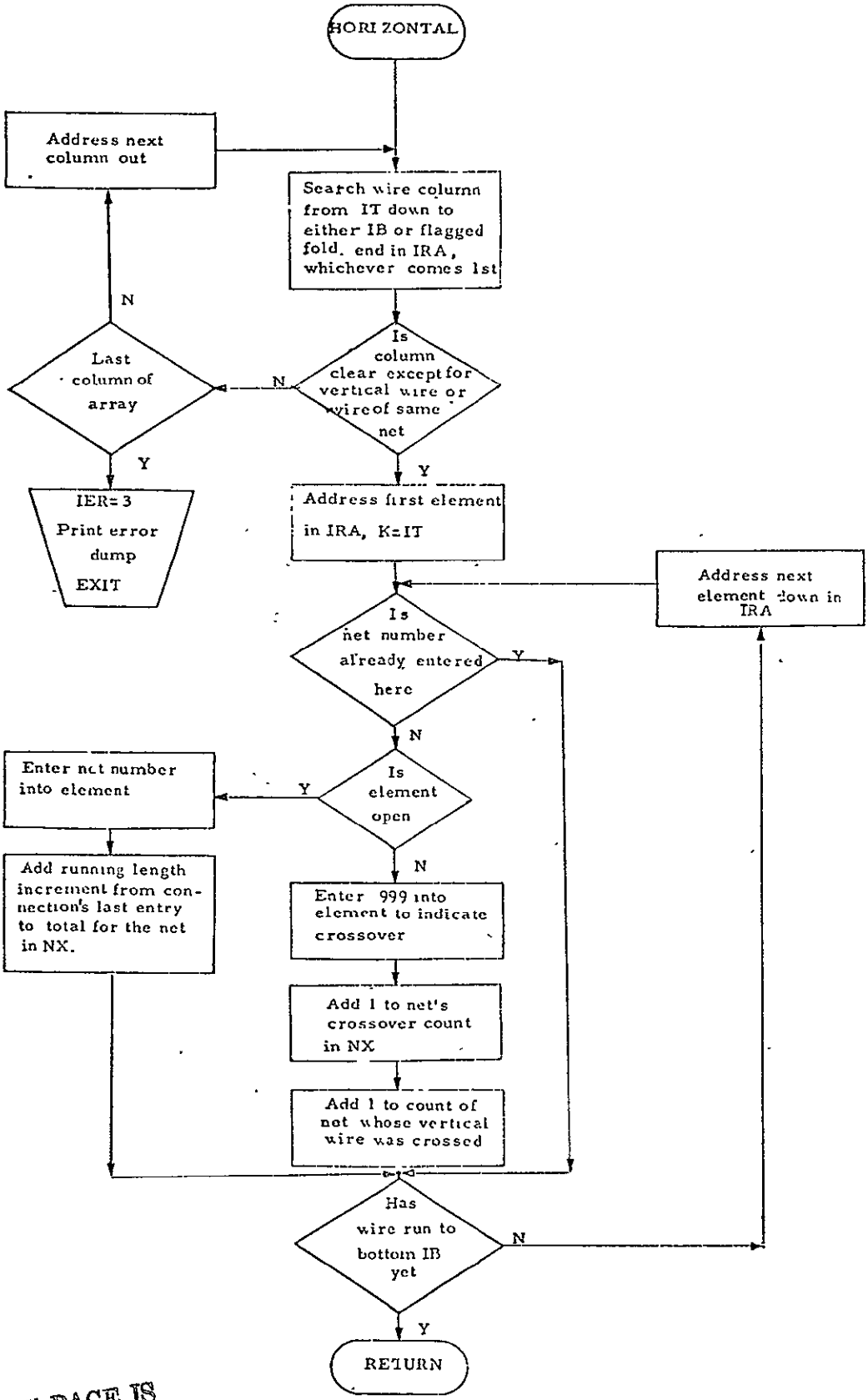


Figure 3-23 HORIZONTAL

3.5 Subroutine SPACE

The SPACE subroutine is called by ROUTE to calculate the distance in tenths of a mil which must be added to the running length of a cell edge pin to separate it from the adjacent cell.

3.5.1 Functional Description

The process spacing requirements, as set by input parameters ICN(62), ICN(63), and ICN(64), are used as subroutine arguments through array ISR. The distances between the cell's edge and the metal and P-material within are specified by the edge parameters of the Circuit Type File, and are input via argument array IDT. The subroutine breaks down the data words of IDT so that the parameters may be referenced individually in array IC, as shown in Table 3-10. By using the ND array, sets of corresponding parameters are referenced and checked against the minimum spacing requirements of the process and the current value of IS. IS contains the largest separation requirement which has been determined at any point during the checks, being immediately replaced by a larger requirement when one is found. Thus when all the parameter sets have been checked, we will return with IS being the required space between cell edge pins.

A list of the edge parameter sets which are checked and the process parameters they are checked against is included in Table 3-11. These variables are defined in the description of the Circuit Type File in the Banning Engineering Notebook.

3.5.2 SPACE Variables and Arrays

The following key variables are referenced by SPACE:

IS - Distance calculated by SPACE by which the edge pins of the two argument cells are to be separated in tenths of a mil.

ISS - Temporary storage of space required by specific parameters.

The following arrays are referenced by SPACE:

IC(I, J) - Array constructed from IDT containing each of the 24 decimal characters of the input word in individual elements of the array (see Table 3-10). Note this array is not in COMMON, and hence is not related to the COMMON IC array used elsewhere in the program.

I =	1	2	3	4	5	6
IDT(I)	221611	101210	Not used	031013	101226	Not used

Format IDT-S1

IC(K)

K =	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
	2	2	1	6	1	1	1	0	1	2	1	0	0	3	1	0	1	3	1	0	1	2	2	6

Format IC-S1

Note this is not the same array as the array labeled IC in COMMON.

I =	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
ND(I)	4	16	8	20	2	14	6	18	12	24	2	16	6	20	4	14	8	18

Format ND-S1

ND is an array referencing elements of IC. All elements are set by a data statement.

Table 3-10 SPACE ARRAYS

Parameter of Cell #1	Parmater of Cell #2	Process Spacing Checked
DP	DP	ICN(62)
DLP	DLP	ICN(62)
DUP	DUP	ICN(62)
DLM	DLM	ICN(63)
DUM	DUM	ICN(63)
DMB	DMB	ICN(63)
DLM	DPM	ICN(64)
DUM	DUP	ICN(64)
DLP	DLM	ICN(64)
DUP	DUM	ICN(64)
TB	TB	N/A
DB	DB	ICN(62)

SPACE PARAMETER CHECKS

Table 3-11

IDT(I) - Argument array containing the edge parameters of the adjacent cells. IDT is equivalent to the INT array of the ROUTE subroutine.

IDT(1) = INT(1) - 1st cell, edge parameter word 1
IDT(2) = INT(2) - 1st cell, edge parameter word 2
IDT(4) = INT(4) - 2nd cell, edge parameter word 1
IDT(5) = INT(5) - 2nd cell, edge parameter word 2

ISR(I) - Argument array containg the process spacing requirements.

ISR(I) = ICN(I+61); for I=1, 2, 3

ND(I) - Array which is set by a data statement and is used to sequentially reference parameter sets from IC.

3.5.3 SPACE Flowchart

The SPACE subroutine flowchart is contained in Figure 3-24.

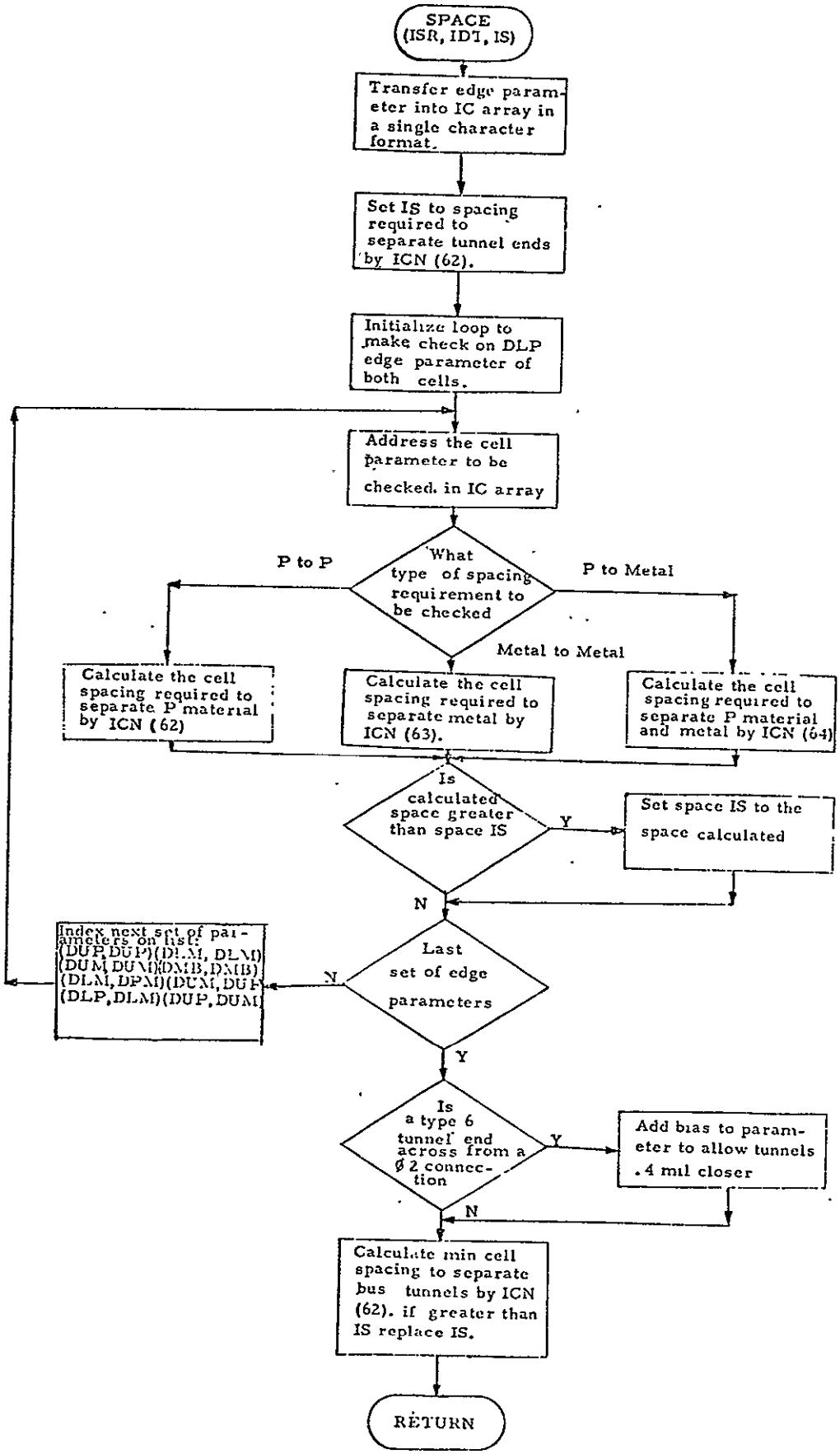


Figure 3-24 SPACE

3.6 Subroutine FOLD

FOLD subroutine is called to determine the points at which the linear collection of cells in IRA is to be broken into smaller fold segments.

3.6.1 Functional Description

As explained in the MAIN description, FOLD is called twice; the first time just to locate these fold points and the second to attempt to improve them and to put the folds in appropriate form on a scratch storage for latter reference. The number of folds to be made is calculated during the first pass and stored in ICN(67) to force the same number of folds on the second pass. This calculation takes into account estimates of width of wiring located between trial folds, the width of wiring around the left edge of trial folds, and the actual widths of cells in the tentative folds. The algorithm determines the number of folds which form a chip with the smallest Y-dimension that is not less than its X-dimension. Thus if fold determination is left to the program, chips will always have greater height than width.

The PRF User's Manual describes options to force the number of folds to a specified number, or to specify the cells which terminate each fold. If the latter option is not selected, the program will attempt to locate the points at which the string of logic cells must be broken to give the required number of logic folds having the most uniform length. This is done by breaking the array into tentative folds and then attempting to modify the number of cells in each of these to decrease the length of the array's longest fold. Pads are always folded separately in end folds regardless of relative fold lengths.

Once the breakpoints have been determined, they are flagged in the IRA array by setting the elements of the first and last column of each fold negative. Each fold is also described by a four element column entered into the IF array. All the information on folding required by ROUTE to run wires on an individual fold basis is now contained in these arrays, and the subroutine will return if flags indicate this is the first pass through.

On the final pass through FOLD, several operations are performed on the IRA array to refine and reformat it. All the array's wiring is searched for locations which will require a connection between metal and tunnel, and 998 is entered into these locations to indicate tunnel end positions. The search also looks for tunnel segments having five or more elements which can be replaced by metal, and enters the necessary tunnel ends and metal for these connections.

IRA is placed in the format utilized by the remainder of PRF by identifying those folds which must be inverted (folds which will have interconnections above their cells) and interchanging their elements about both the X and Y axes of symmetry. Since the top pad fold must be normal and the bottom pad fold inverted, an odd number of logic folds will cause a dummy fold having no cells to be inserted in the array. Each fold is finally transferred to the scratch storage from IRA, completing FOLD operations.

3.6.2 FOLD Variables and Arrays

The following list defines key variables referenced by FOLD:

IAH	-	Average height of cell interconnections of a fold.
IARL	-	Length of logic elements of a fold.
IB	-	Index on bottom logic element pin total running length.
IFL	-	Nominal running length span of a fold.
IIF	-	Current number of folds which have been determined.
IN	-	ROUTE pass number flag; initialized to 1 by MAIN.
IPH	-	An approximation to the height of pad folds and associated wiring.
IR	-	ROUTE pass number flag; initialized to 1 by MAIN.
IRYY	-	Y-index of second last row in IRA.
IT	-	Index on top logic element pin of fold.
J1	-	Number of edge pins included in logic cells; hence also the number of columns in NC during fold optimization.
J3	-	During fold point optimization: J3 = The total number of folds.
KTOP	-	2 x (number of top pad fold pins) - (number of logic fold pins).

LNGF - Length of longest fold of a particular folding configuration.

LNGL - Length of left fold being compared for length.

LNGR - Length of right fold being compared for length.

LNGST - Shortest of the longest folds found during optimization.

NF - Number of folds for trial array layout.

NF1 - (Number of folds) - 1.

TRH - Trial height of array.

TRW - Trial width of array.

The following arrays are constructed or modified by FOLD:

IC(I, J) - Format IC - F1: Element IC(I, J) contains the net number of a wire which is to be connected to element I, pin J.

IF(I, J) - Format IF - F1: See Table 3-12. For Fold I:

IF(I, 1) = Index to pin in NC which is the last pin of fold I.
IF(I, 4) = Set to 0 initially; used as a modification to the index in IF(I, 1) during fold optimization.

Format IF - F2: See Table 3-12. For Fold I:

IF(I, 1) = Y - Index on last horizontal wire of fold in IRA.
IF(I, 2) = X - Index on top pin of fold in IRA.
IF(I, 3) = Y - Index on bottom pin of fold in IRA.
IF(I, 4) = 1 - Indicates fold I inverted.

INT(I) - Linear array containing the X-index to the bottom pin of every other logic fold in IRA in normal sequence, beginning with the first logic fold.

NC(I, J)

Format NC-F1

J=	I= 1 2 3 4 ...
1	X-index to right edge pin of cell in IRA
2	IRA running length at pin
3	Width of channels at pin
4	0 = Normal pin flag 1 = Fold point pin flag
5	Not used

IF(I, J)

Format IF-F1

J=	I= 1 2 ...
1	Fold ref. pt. index to NC
2	Not used
3	Not used
4	Fold variation increment

Where I = Fold number referenced

IF(I, J)

Format IF-F2

J=	I= 1 2 ...
1	Fold width
2	X-index on top of fold
3	X-index on bottom of fold
4	0 = Normal fold flag 1 = Inverted fold flag

Where I = Fold number referenced

IRA(I, J)

Format IRA-F1

J=	I= 1	2	3	...		
1	1	0				
2						
3						
4						
5						
⋮	Running total length	100*(cell #)+(pin #)	Interconnect info.	Interconnect info.	Interconnect info.	Interconnect Format 999 = Crossover 998 = Tap 1 = Vertical metal 2 → 200 = Net # of metal 0 = Open Data is blocked by folds which are alternately row and column inverted.

IRA is output in this format on the work tape after FOLD has been run the second time.

Table 3-12 FOLD ARRAYS

IPR(I) - Linear array used to store element numbers of breakpoints when the breakpoints are specified by user.

IRA(I, J) - Format IRA - F1: See Table 3-12. For column I of IRA:

IRA(I, 1) = Running length (same as Format IRA-R1).
IRA(I, 2) = [(100 x element number) + (pin number)].

All remaining rows of IRA contain the wiring date entered by ROUTE and shifted down two rows to utilize the rows formerly occupied by the pin number and reassignment flag. The wiring data is modified and tunnel ends are entered where required by setting elements to 998.

NC(I, J) - Reference to right edge pins of cells in IRA. See Table 3-12. For edge pin I in IRA:

NC(I, 1) = X-index on edge pin I in IRA.
NC(I, 2) = Running length of edge pin from IRA.
NC(I,3) = Width in mils x 10 of horizontal wire crossing edge pin.
NC(I,4) = Fold pin flag: 1 = last pin of logic fold
0 = normal pin

3.6.3 FOLD Flowcharts

Flowcharts of the FOLD subroutine are contained in Figure 3-25 through 3-31.

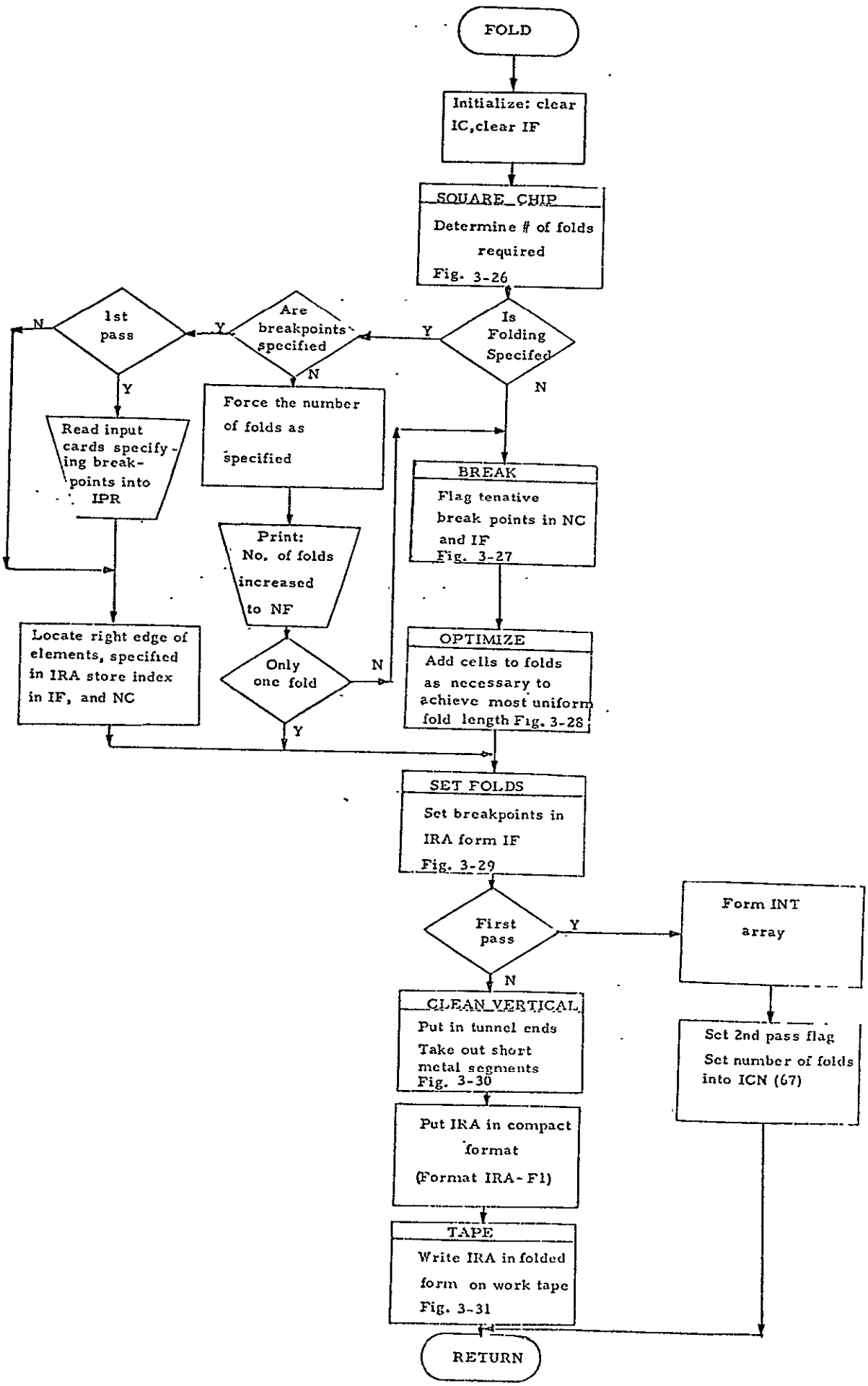
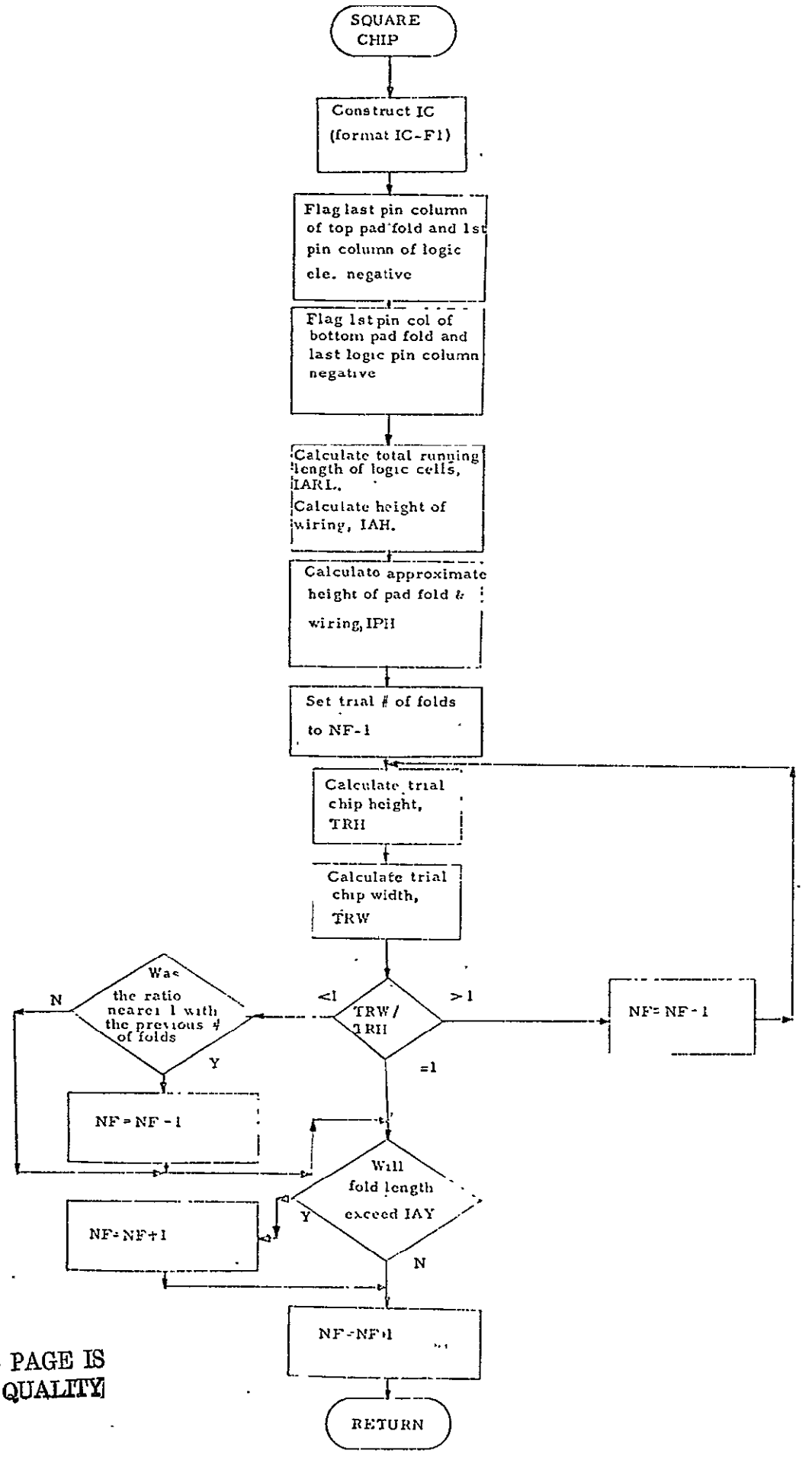


Figure 3-25 FOLD



ORIGINAL PAGE IS
OF POOR QUALITY

Figure 3-26 SQUARE CHIP

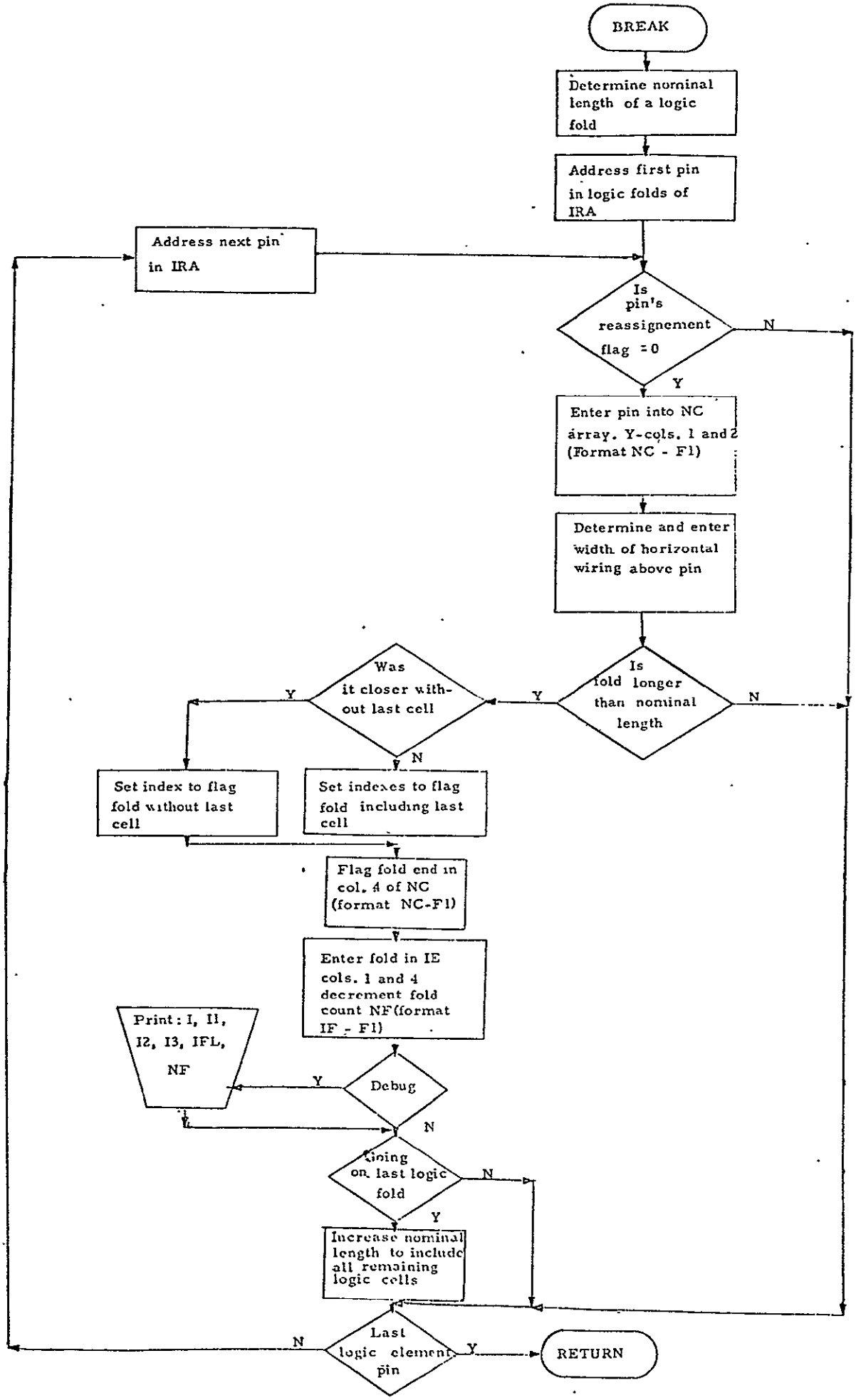
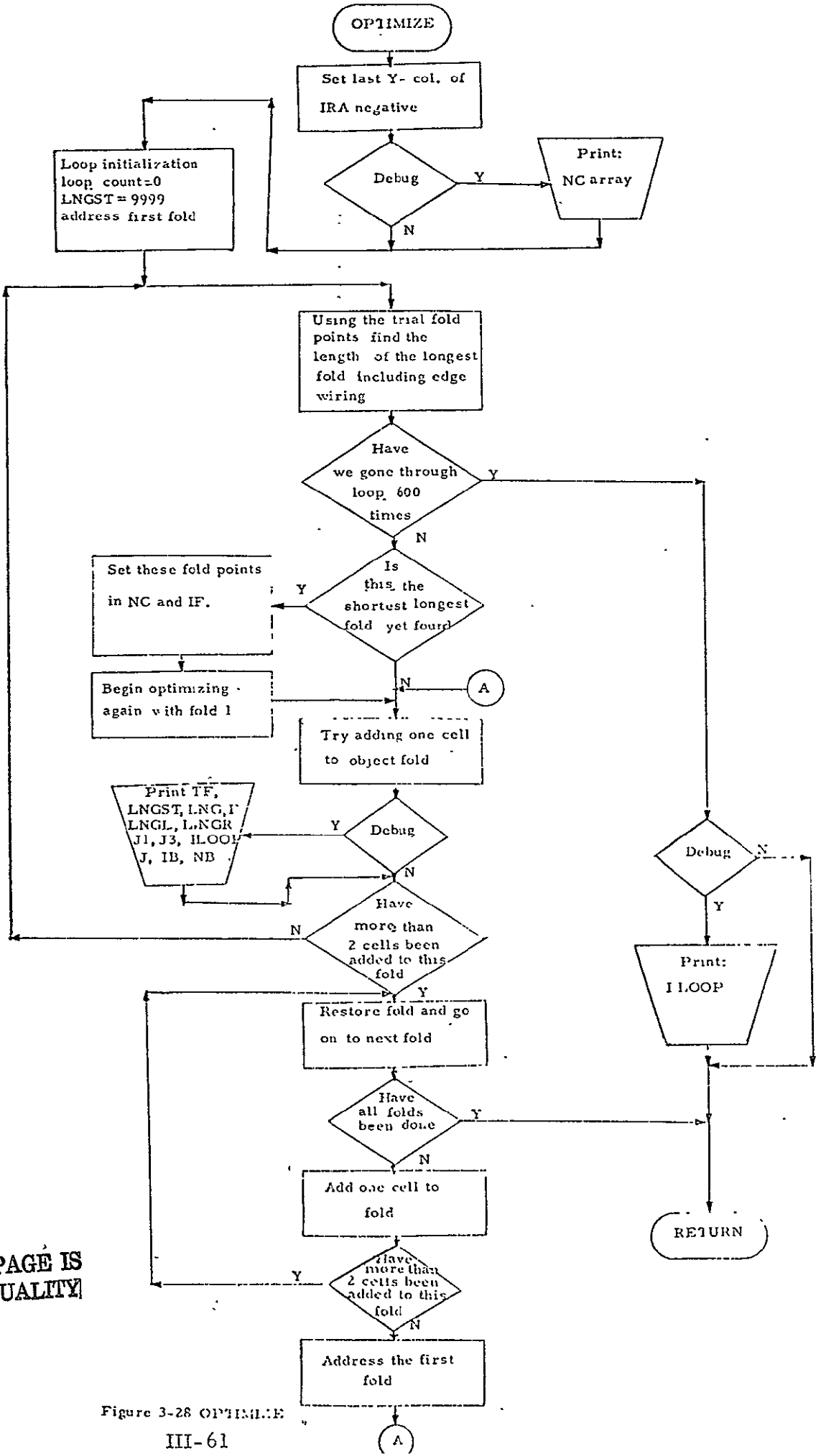


Figure 3-27 BREAK



ORIGINAL PAGE IS OF POOR QUALITY

Figure 3-28 OPTIMIZE

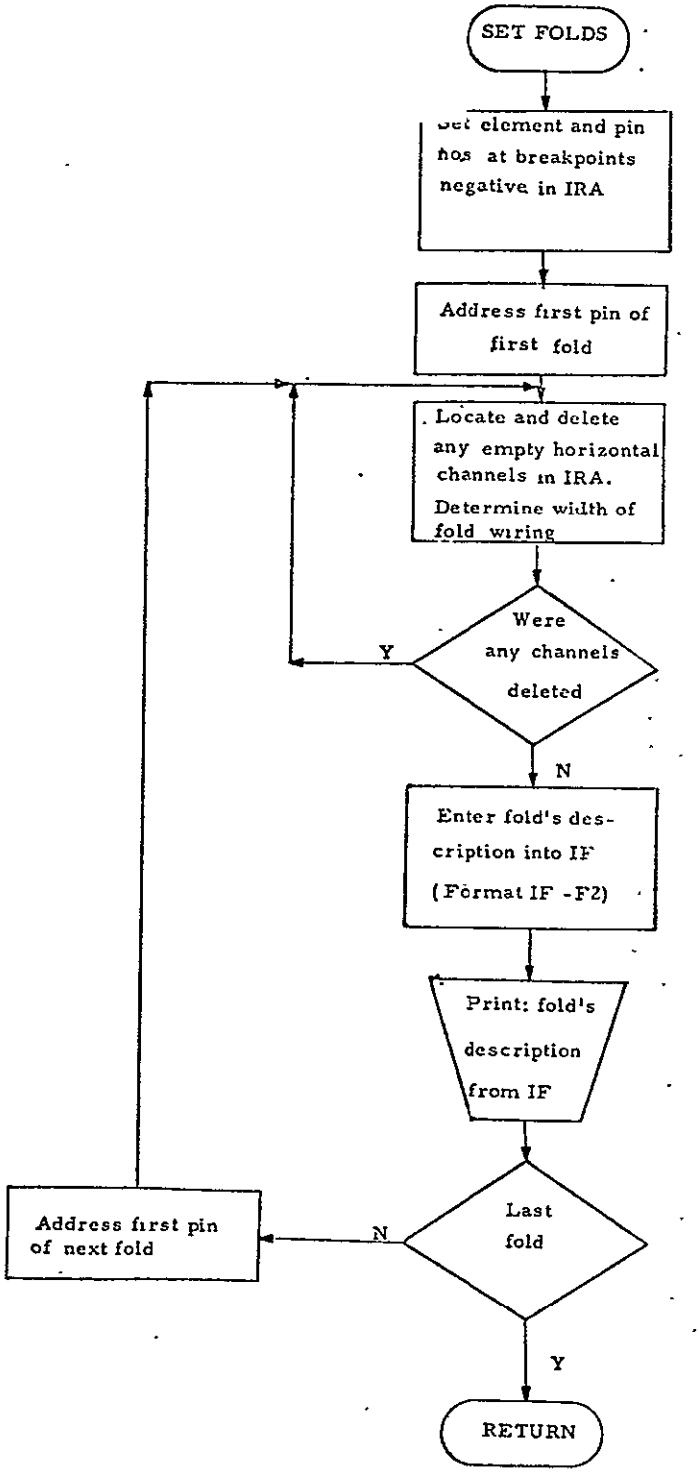


Figure 3 - 29 SET FOLDS

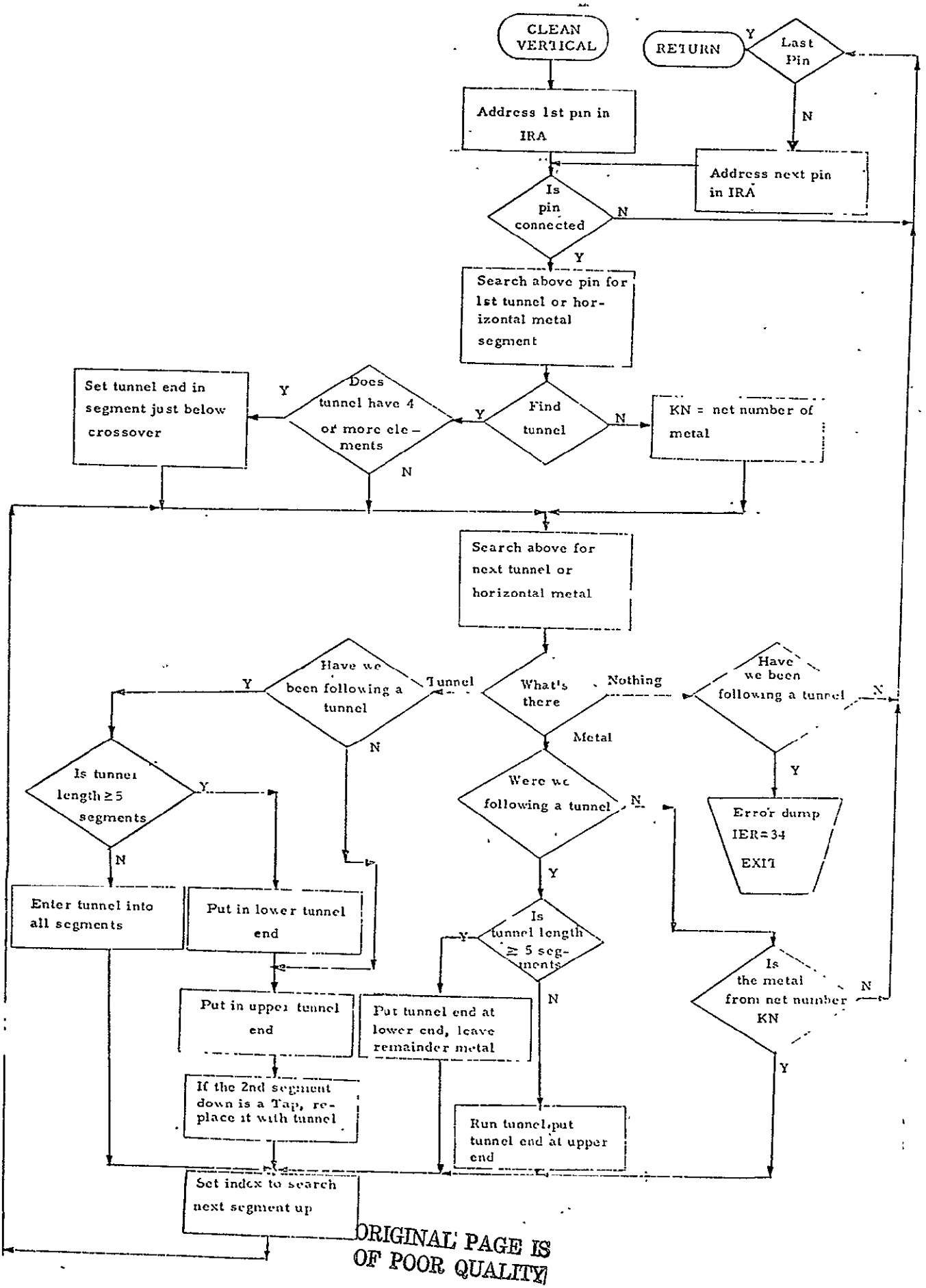


Figure 3-30 CLEAN VERTICAL

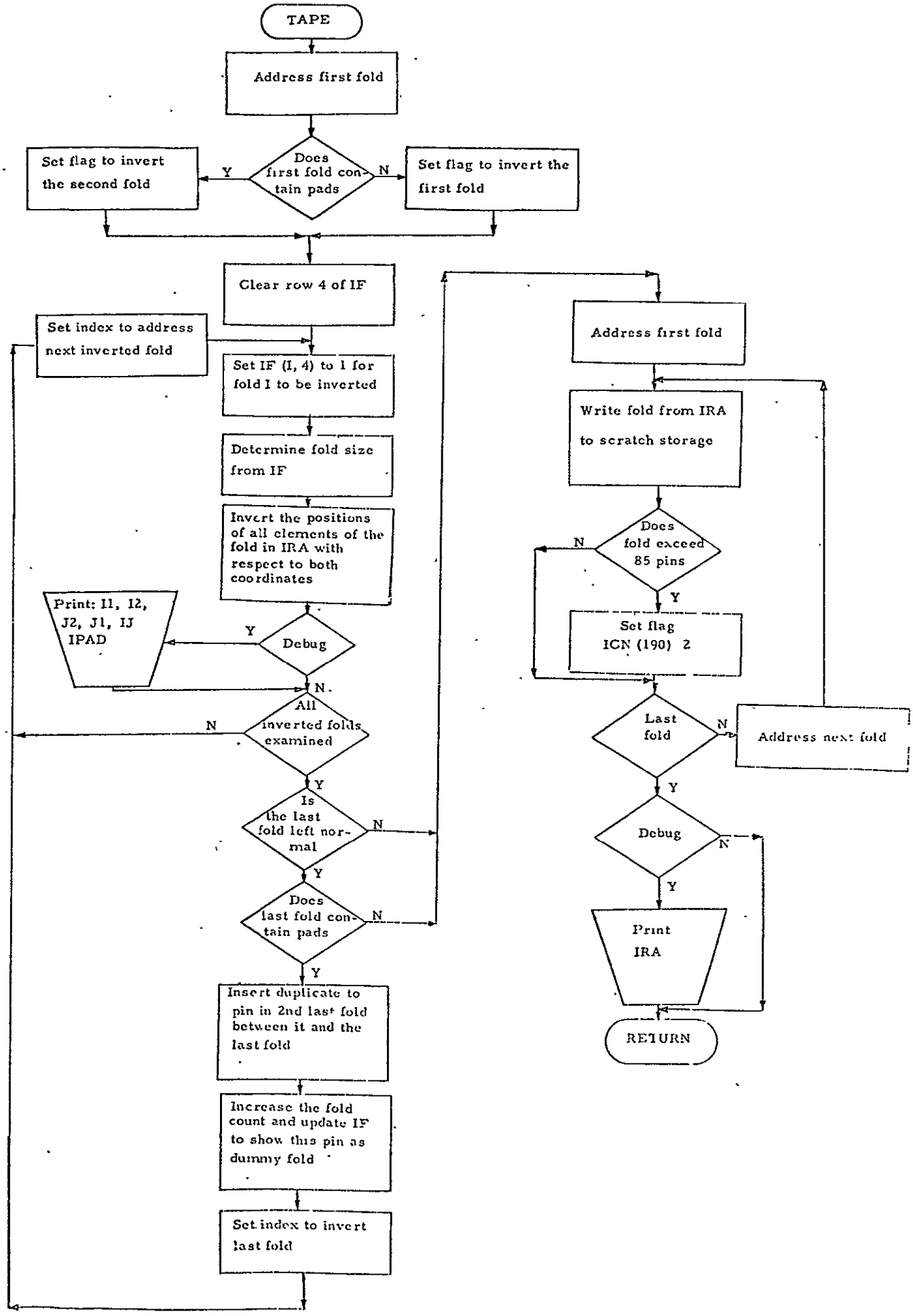


Figure 3-31 TAPE

3.7 Subroutine DIDDLE

The DIDDLE subroutine forms a square map of the chip in array IFA from the previously formed fold segments and makes the required interconnections between these segments.

3.7.1 Functional Description

DIDDLE calls REED to read individual folds into IFA and maintain pointers on the locations of both the last fold and the second last fold read into the array. If the last fold read was inverted, DIDDLE proceeds to make the cross-fold connections required between it and the normal fold read in previously. If it was normal, the fold is connected to the previous fold by left end-around wires. Once it has been connected, the loop recycles to read and connect the next fold in sequence.

Cross-fold wire routing begins by extending the right edge wires far to the right of the longest fold of the pair (the top fold of this pair is normal, the lower fold is inverted). Beginning with the centermost wires and working out, extensions of each wire broken by the folding are then matched and the folds are searched to determine the rightmost and leftmost connection pins. The coding block headed CROSSWIRE determines the leftmost vertical channel through which the crosswire can be run. Since this channel may be to the right of normal fold wiring, the EXTEND coding may be required to assign running length values to these new vertical channels.

Horizontal wiring which is no longer functional in the connection is then deleted as the cross-fold wire is run by coding block XRUN. Horizontal wires having no connection to pins of the fold pair are deleted entirely from the folds. Cross wiring is completed when all of the extended right edge wires have been matched and either crosswired or deleted.

The program branch which performs cross-fold connections also performs bonding pad spacing calculations for pad folds. The space calculated is based on the total length of the adjacent logic fold minus the length required for pads. The bottom fold also allows a space for the test transistor to be inserted on its extreme right. If the spacing calculated is not greater than the minimum set in the input parameters, it is forced to this minimum value.

Left end-around wiring, performed by the other major branch of DIDDLE, occurs immediately after a normal fold is read into IFA. Left edge wires of the normal fold are matched with those of the inverted fold and both are extended left to be connected in the furthest available vertical channel. Each wire is also checked to determine if it has been deleted from the previous

fold pair and must run up to folds above them. The required tunnel is entered if such a connection is indicated.

DIDDLE is completed when the last fold has been entered into IFA and connected to the previous fold. A dump of IFA in format 60I2 in Table 3-13 shows the interconnections of a small chip, consisting of a pad fold, two logic folds, and another pad fold (note that the dump indexes are not in the same directions as generally referenced in this manual). Cross-fold wires have been run between Folds 1 and 2, and left end-around wires are run between Folds 2 and 3, while the required cross-fold wires from Fold 3 to Fold 4 have yet to be run.

3.7.2 DIDDLE Variables and Arrays

The following list defines variables referenced by DIDDLE.

IM1 - Right edge of right end wiring. Indexes channel 19 columns to right of folds.

IST - Starting vertical channel of left end-around wiring; set to ICN(8).

ISW - Switch is normally set to 1. When the right edge of the normal fold is searched and no more wires can be found, ISW is set to 3, forcing the routine to make a second search after the folds have been extended as required.

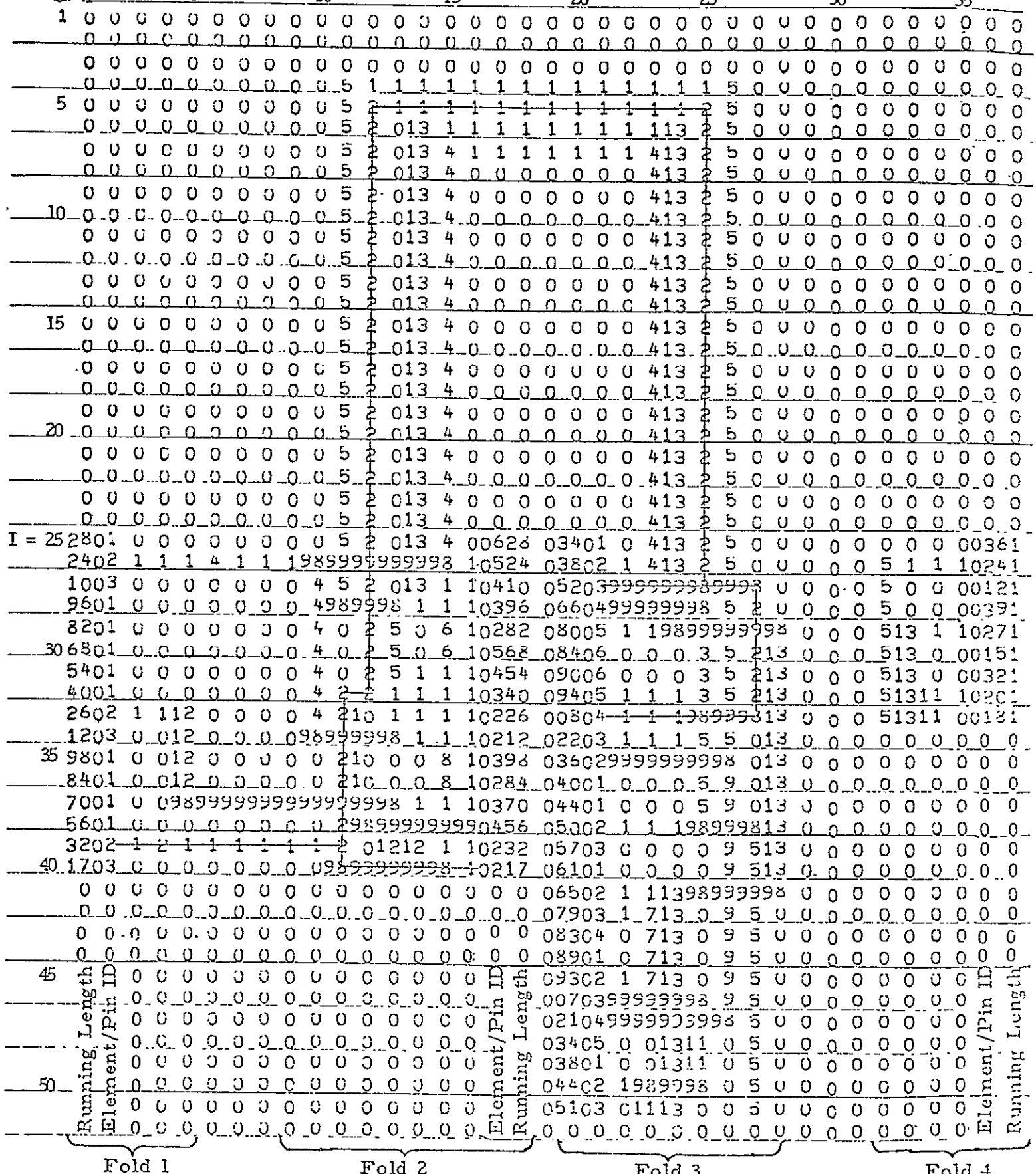
IS1 - Switch initialized at start of each crosswire pass to 3. Controls whether crosswire is to be run during the pass.

IS2 - Switch used to sequence normal fold, then inverted fold through reference pin removal coding.

IS2 = 1 - Normal fold processing
IS2 = 2 - Inverted fold processing

IS3 - Flag used to indicate one of the folds of a fold pair contains pads.

IS3 = 1 - No pads in fold pair
IS3 = 2 - Normal fold contains pad
IS3 = 3 - Inverted fold contains pad



DIDDLE ARRAY DUMP, IFA(I,J), FORMAT IFA-D1

Table 3-13

I1	-	X-index on right edge of normal fold.
I6	-	Pointer on candidate channel of normal fold for running cross-fold wire.
I6M	-	Conditional cross-fold wire's X-coordinate.
I7	-	Pointer on candidate channel of inverted fold for running cross-fold wire.
JNC	-	Net number of end wire.
JNCM	-	Net number of trial crosswire.
J1	-	Y-coordinate of top of normal fold's running length row.
J2	-	(Y-coordinate of bottom of normal fold) - 1.
J3	-	(Y-coordinate of top of inverted fold) - 1.
J4	-	Y-coordinate of bottom of inverted fold's running length row.
J6	-	Y-coordinate of normal fold's right edge wire.
J6M	-	Y-coordinate of normal fold's trial crosswire end point.
J7	-	Y-coordinate of inverted fold's right edge wire.
J7M	-	Y-coordinate of inverted fold's trial crosswire end point.
KL1	-	Width of normal fold's right edge wires calculated at K7 by LCRD.
KL2	-	Width of normal fold's right edge wires calculated at K71 by LCRD.
KR1	-	Width of inverted fold's right edge wires calculated at K7 by LCRD.

KR2 - Width of inverted fold's right edge wires calculated at K71 by LCRD.

KSK - Switch initialized to 1 after reading an inverted fold into IFA. When the crosswire search is to be recycled, KSK is set to 2, preventing more than two complete restarts on the same fold pair.

K7 - X-index on inverted fold's right edge channel containing the wire being run.

K71 - K71 = (K7-1).

LL1 - Previous value of KL1.

LR1 - Previous value of KR1.

MINSP - Minimum space between pads in mils x 10.

MR - X-index on right edge of inverted fold.

MST - X-index on first pad pin which is to remain in fold.

NR - Count of the number of folds which have been read into IFA.

NWAL - Minimum width of normal fold's right edge wires.

NWAR - Minimum width of inverted fold's right edge wires.

The following arrays are constructed or modified by DIDDLE:

IF - See REED arrays, Section 3.8.2.

IFA - Contains the folded wiring run by ROUTE and broken into individual folds as specified by FOLD. The folds are read into the array by REED, stacked in the Y-dimension, and interconnected by DIDDLE as required to restore connections. See Format IFA-D1 in Table 3-13.

NX - Contains reference to a net N associated with an element in IFA(I, J); where for array index K: (Table 3-14)

NX(K, 1) = I
NX(K, 2) = J
NX(K, 3) = N

3.7.3 DIDDLE Flowcharts

Flowcharts of the FOLD subroutine are presented in Figures 3-32 through 3-39.

NX(I, J)

Format NC-D1

J= \ I=	1	2	...	K4K
1			X-Coord. of Tunnel End	
2			Y-Coord. of Tunnel End	
3			Net Number of Tunnel End	

K4K indexes last entry made in array.

Table 3-14 DIDDLE ARRAY

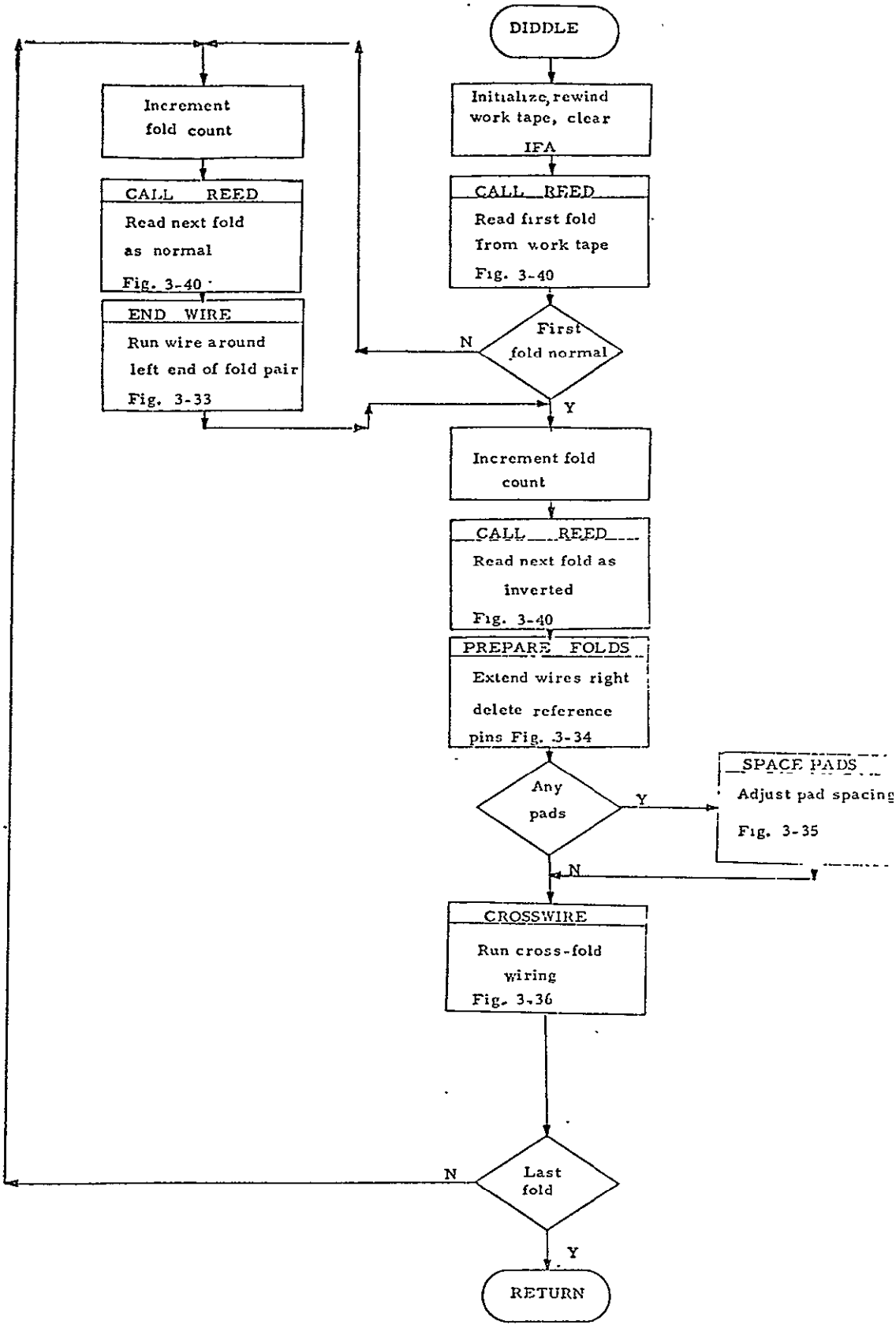


Figure 3-32 DIDDLE

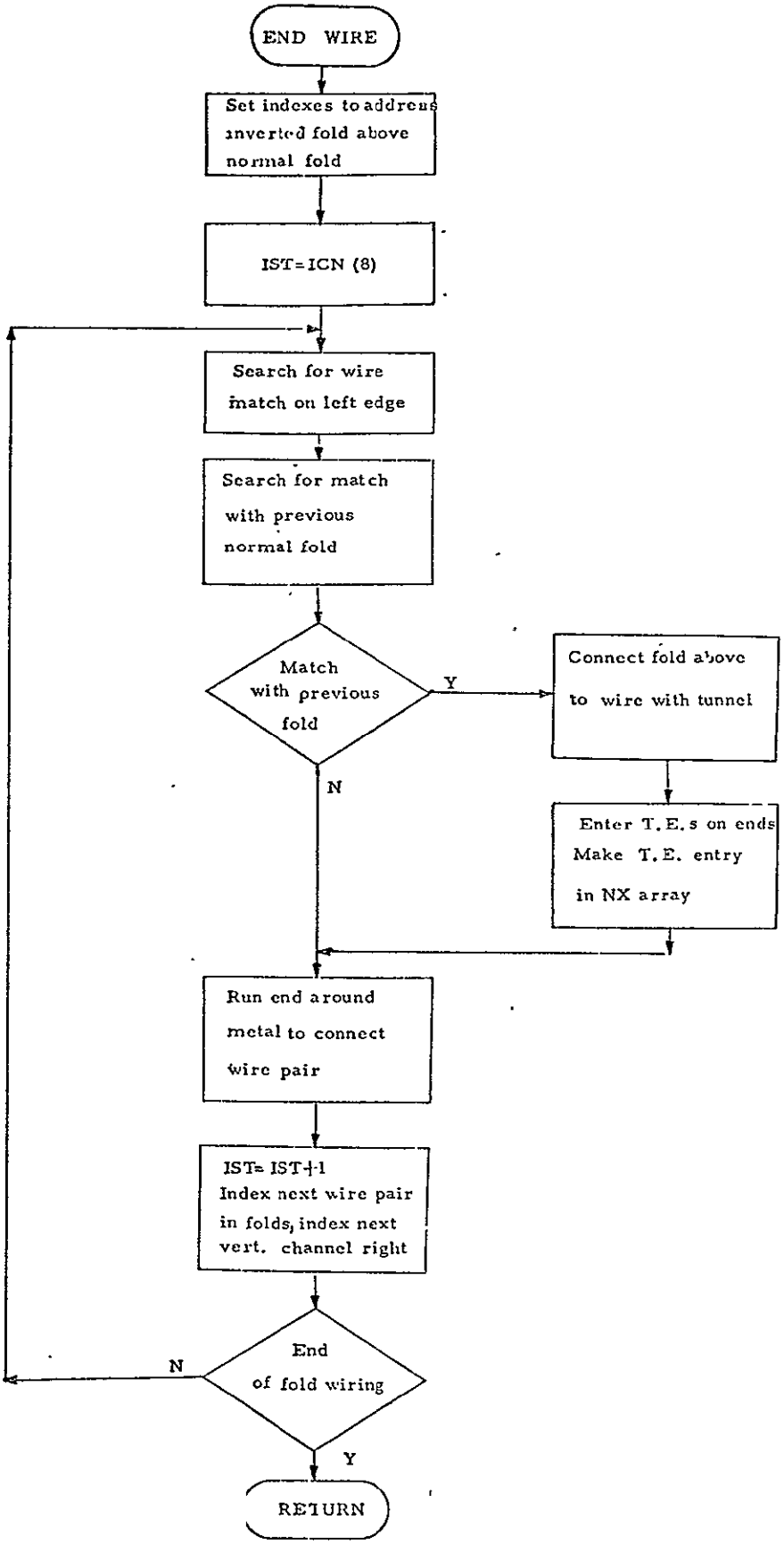


Figure 3-33 END WIRE

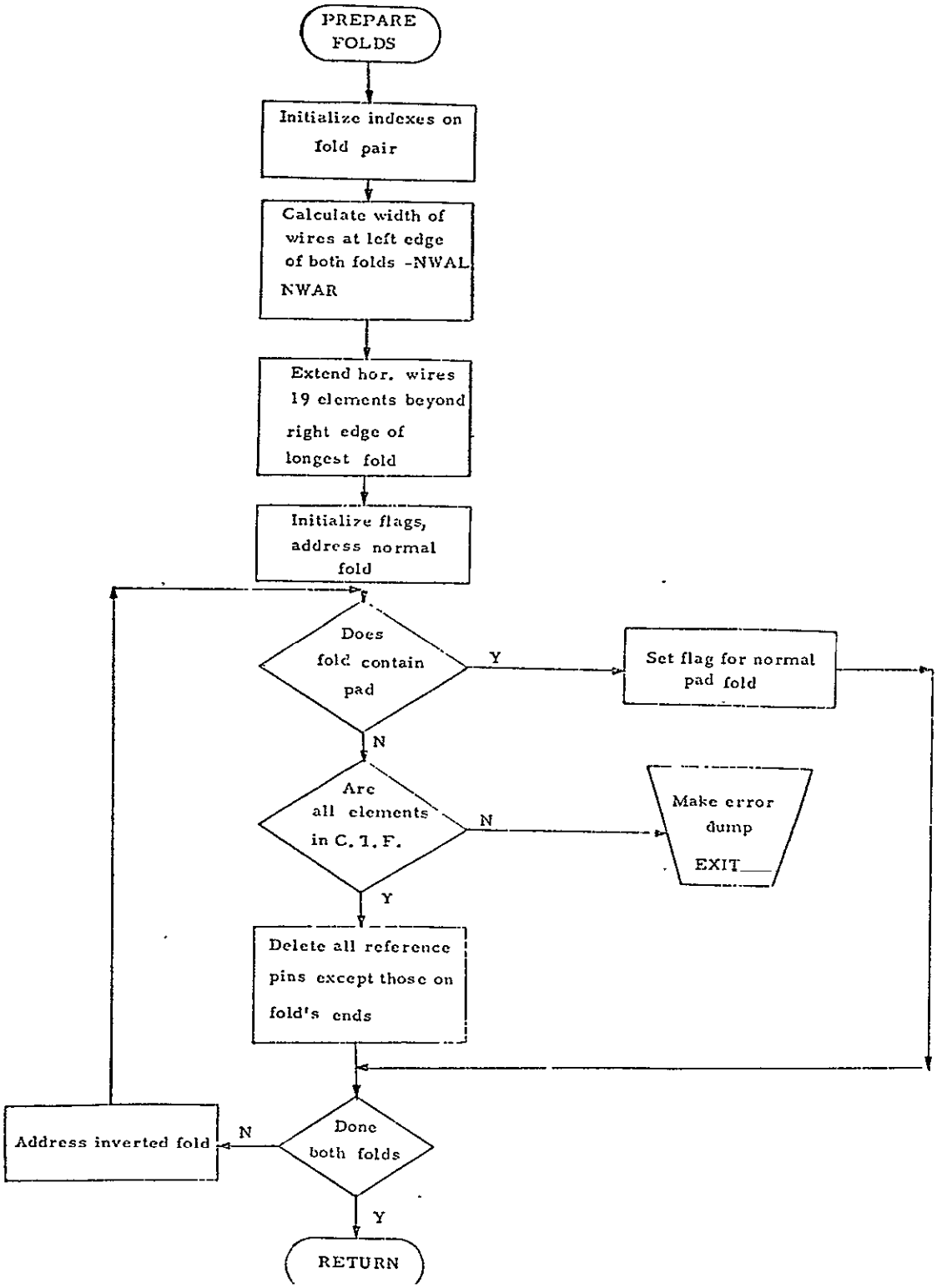


Figure 3-34 PREPARE FOLDS

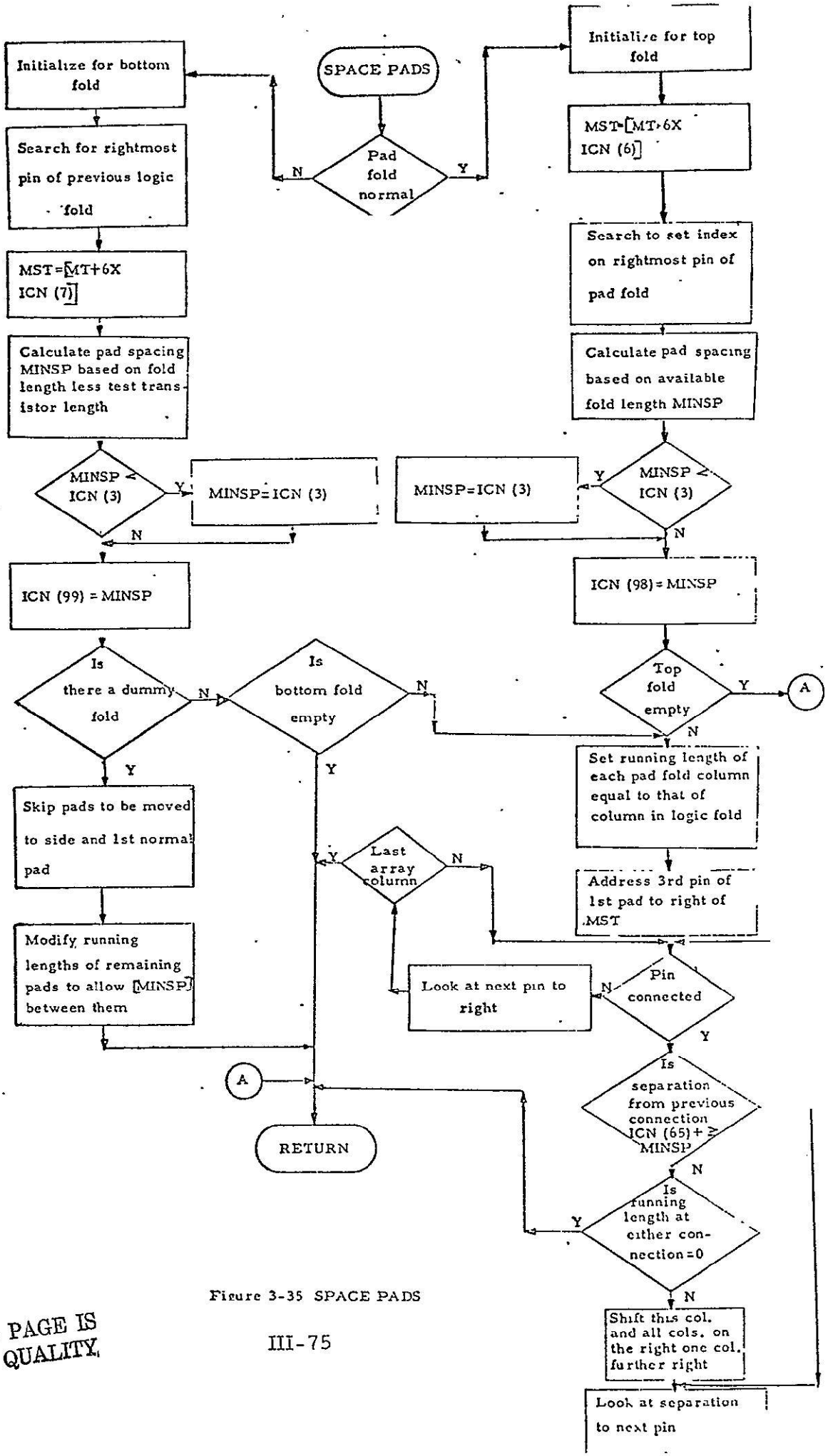


Figure 3-35 SPACE PADS

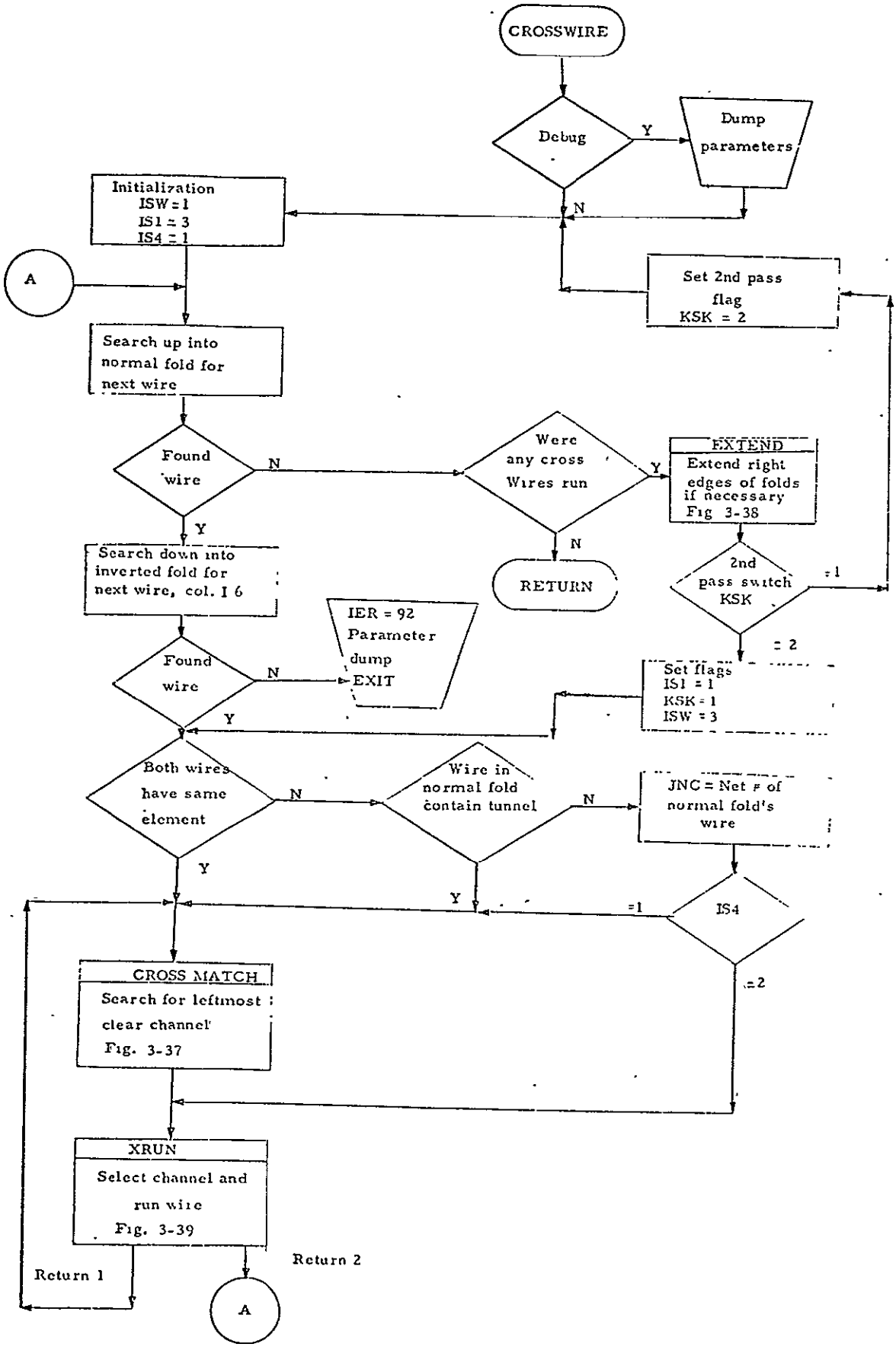


Figure 3-36 CROSSWI

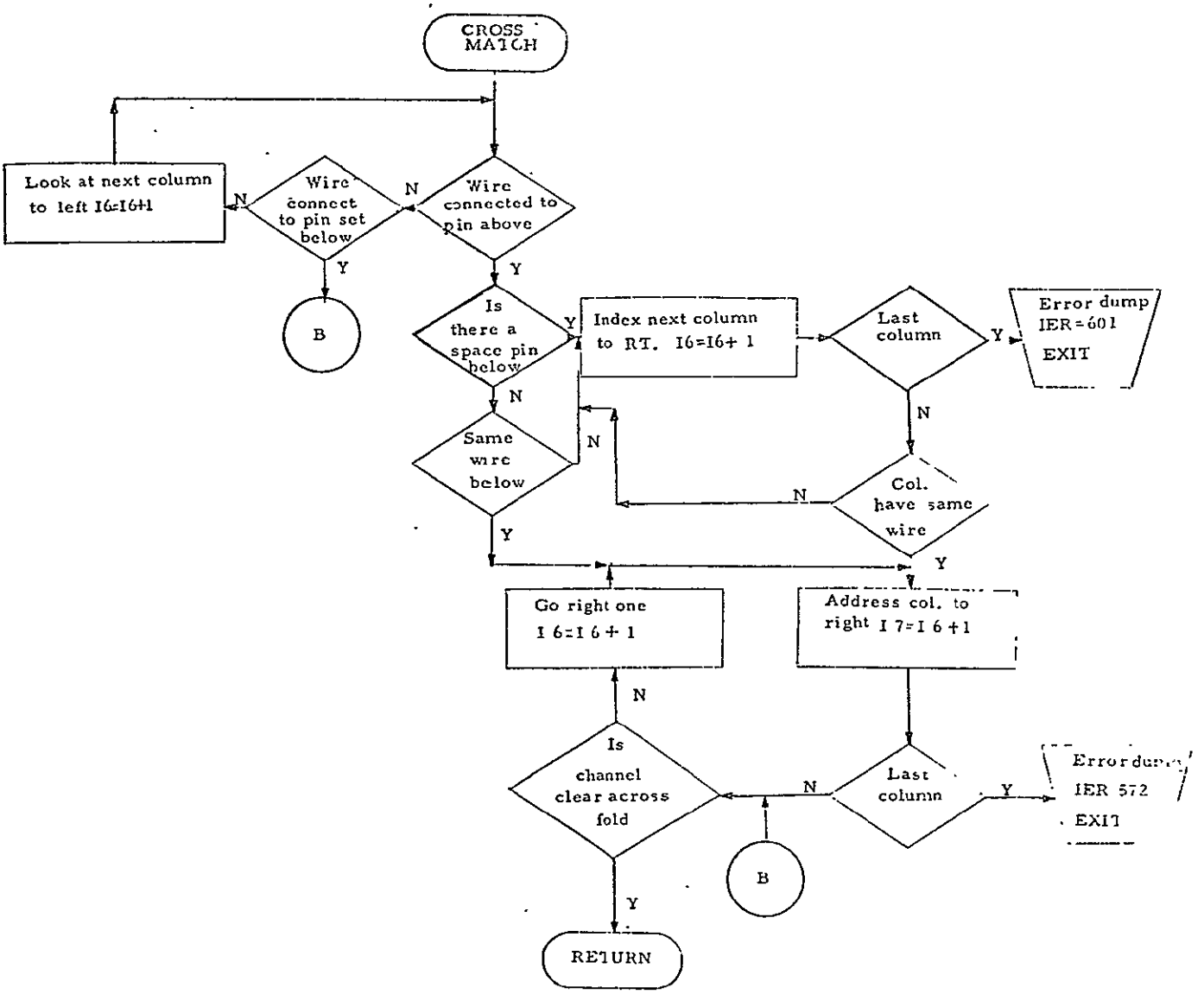
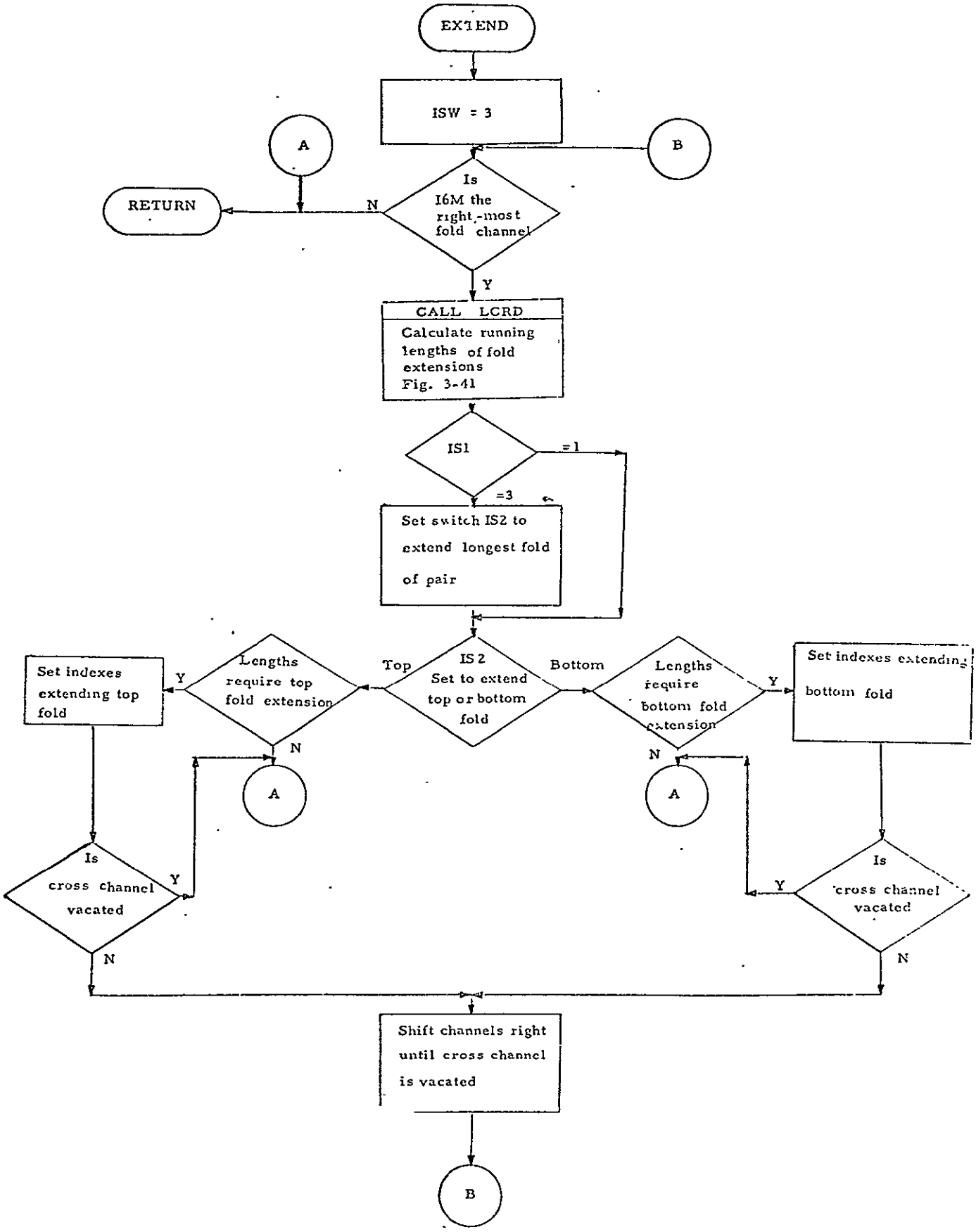


Figure 3-37 CROSS MATCH


EXTEND
ISW = 3
A
B
RETURN
N
Is I6M the right-most fold channel
Y
CALL LCRD
Calculate running lengths of fold extensions Fig. 3-41
IS1
=1
=3
Set switch IS2 to extend longest fold of pair
Top
IS 2 Set to extend top or bottom fold
Bottom
Lengths require top fold extension
Y
N
Set indexes extending top fold
Lengths require bottom fold extension
Y
N
Set indexes extending bottom fold
A
A
Is cross channel vacated
Y
N
Is cross channel vacated
Y
N
Shift channels right until cross channel is vacated
B

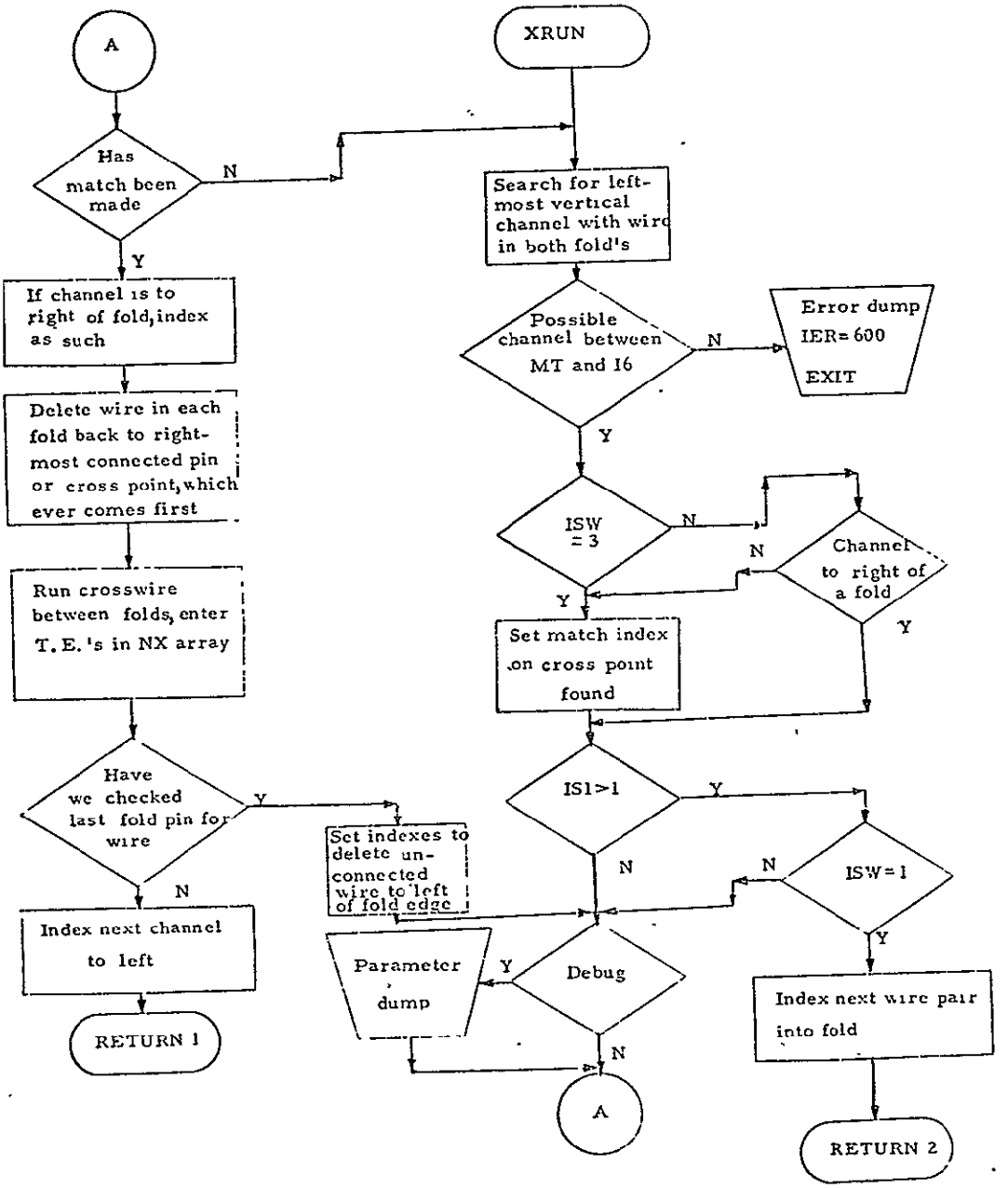


Figure 3-39 XRUN

3.8 Subroutine REED

DIDDLE calls REED to read individual folds from the scratch storage into IFA.

3.8.1 Functional Description

In addition to its function of simply transferring data into IFA, REED maintains indexes on the locations of the currently inputed fold and the previously inputed fold, and makes a row entry in IF for the previously inputed fold as a permanent reference to its location in IFA. Before reading a fold, REED increments the indexes so that it is placed immediately beneath the previous fold in IFA.

3.8.2 REED Variables and Arrays

Reed utilizes the following variables as either input or output arguments:

Input Arguments

J3	-	Y-index on top of previous fold in IFA.
J4	-	Y-index on bottom of previous fold in IFA.
MT	-	X-index on the column in which the left edge of the fold is to be entered in IFA.
NR	-	Number of the fold to be inputed.

Output Arguments

I1	-	Right X-index of previous fold (previous value of MR).
J1	-	Y-index on top of second last inputed fold in IFA.
J2	-	Y-index on bottom of second last inputed fold in IFA.
J3	-	Y-index of IFA on top of last inputed fold.
J4	-	Y-index of IFA on bottom of last inputed fold.
MR	-	X-index to IFA on the last column of the last fold inputed to IFA.
NR	-	Number of last inputed fold.

C.2

The following array is constructed in Format IF-RE1 and is shown in an array map in Table 3-15:

IF(I, J) - For Fold I:

IF(I, 1) - Y-index on top of fold in IFA
IF(I, 2) - Y-index on bottom of fold in IFA
IF(I, 3) = 1 - Fold I is inverted
= 0 - Fold I is normal

3.8.3 REED Flowchart

The REED subroutine is flowcharted in Figure 3-40.

IF(I, J)

Format IF-RE1

J=	I= 1	2	...	10
1			Fold Top Index	
2			Fold Bottom Index	
3			0 = Normal Flag 1 = Inverted Flag	
4			Not Used	

Where I = The number of the fold described.

Table 3-15 REED ARRAY

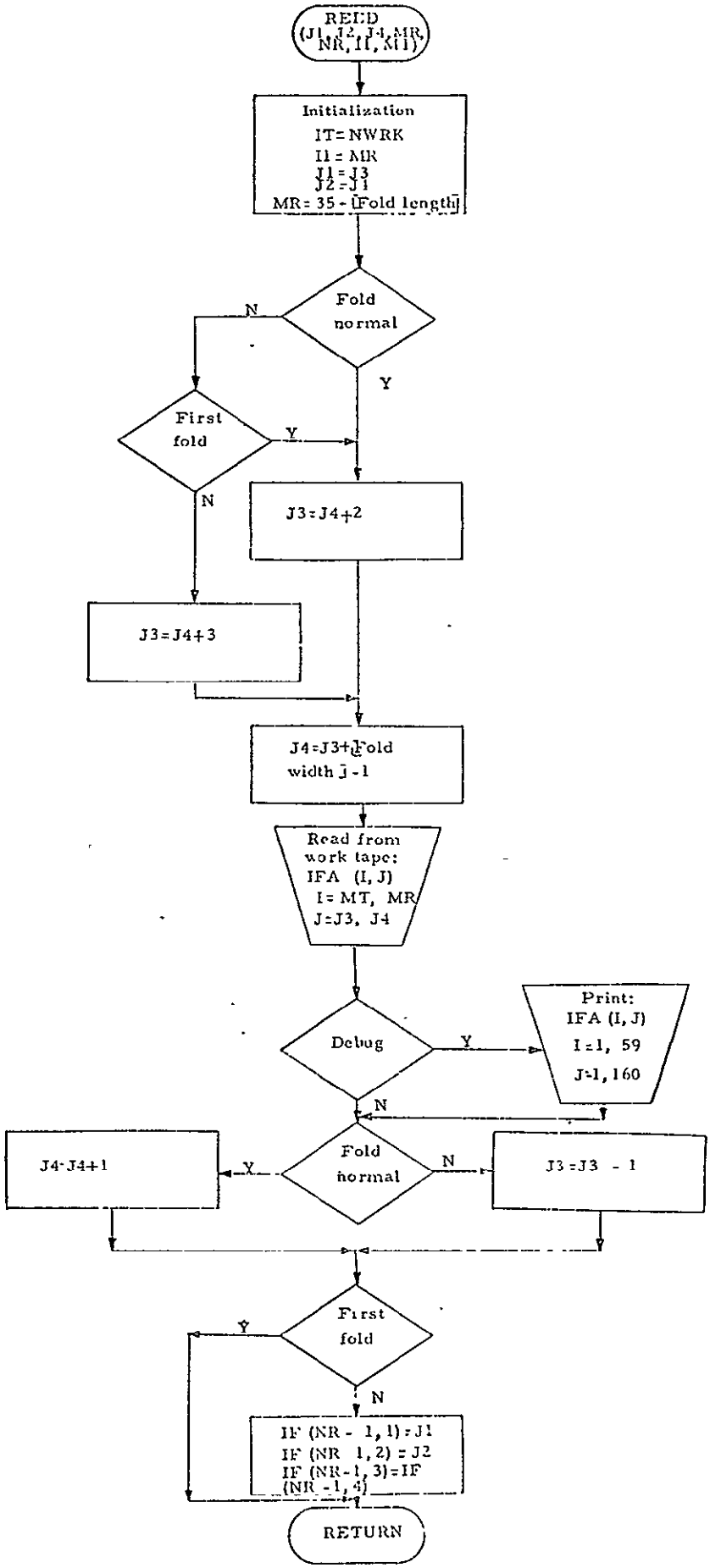


Figure 3-40 REED

3.9 Subroutine LCRD

LCRD is called by DIDDLE to determine a X-direction running length value for a wiring column of IFA located to the right of fold columns which have previously been assigned such values.

3.9.1 Functional Description

The LCRD subroutine is entered with a X-index on the column in IFA whose running length value is to be determined, and a Y-index on the row of IFA which contains those running length values which have been assigned for the fold of interest. It will return with the value K calculated as follows:

$$K = \lceil (\text{Center to center spacing}) + (\text{left edge wiring width}) + (\text{length of fold's previously assigned columns}) \rceil .$$

3.9.2 LCRD Variables

The following variables comprise the arguments of the LCRD subroutine:

Input Arguments

J	-	Y-index on IFA row containing running length assignment data.
K7	-	X-index on the column in IFA for which the running length value is to be determined.
MM	-	Spacing required between adjacent tunnel ends.
MT	-	X-index on left edge of folds in IFA.
NW	-	Value of left edge wiring to be added in IFA.

Output Arguments

K	-	Calculated for running length value for column K7.
K7	-	X-index on rightmost running length value which has been assigned to this fold.

3.9.3 LCRD Flowchart

The LCRD subroutine is flowcharted in Figure 3-41.

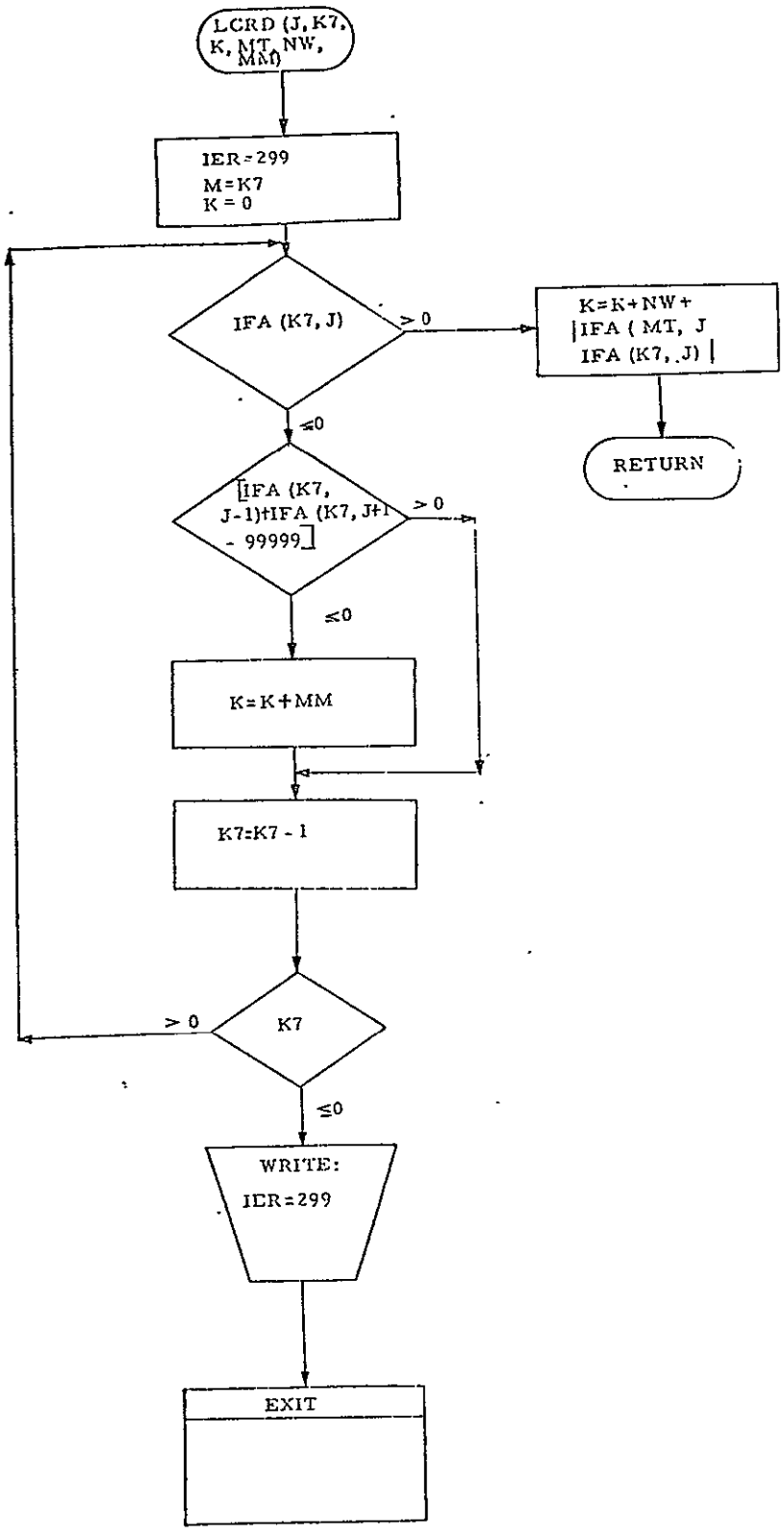


Figure 3-41 LCRD

3.10 Subroutine LOOK

LOOK is called to identify the net number associated with an element of array IFA.

3.10.1 Functional Description

The NX array, which will be in format NX-D1 is first searched to find an entry specifically identifying the net number of element (I, J) of IFA. If no such entry is found, the element is assumed to belong to the net connected to the logic cell pin having the same column and contained in the same fold as the element. This net number is found by locating the proper pin in IFA and then finding this pin's reference in IC (format IC-F1). If the element cannot be located in NX and a fold pin cannot be associated with the element, the subroutine returns with a non-existent net number.

3.10.2 LOOK Variables

The following variables are used as arguments of the LOOK subroutine:

Input Arguments

I	-	X-coordinate of element whose net is to be identified.
J	-	Y-coordinate of element whose net is to be identified.
K	-	Y-index to row in IFA containing cell/pin identity of pins in fold containing unknown element.
L	-	Number of column entries which have been made in NX.
MR	-	Y-index on left edge of fold 1.

Output Arguments

N	-	Net number associated with element IFA(I, J).

3.10.3 LOOK Flowchart

The LOOK subroutine is flowcharted in Figure 3-42.

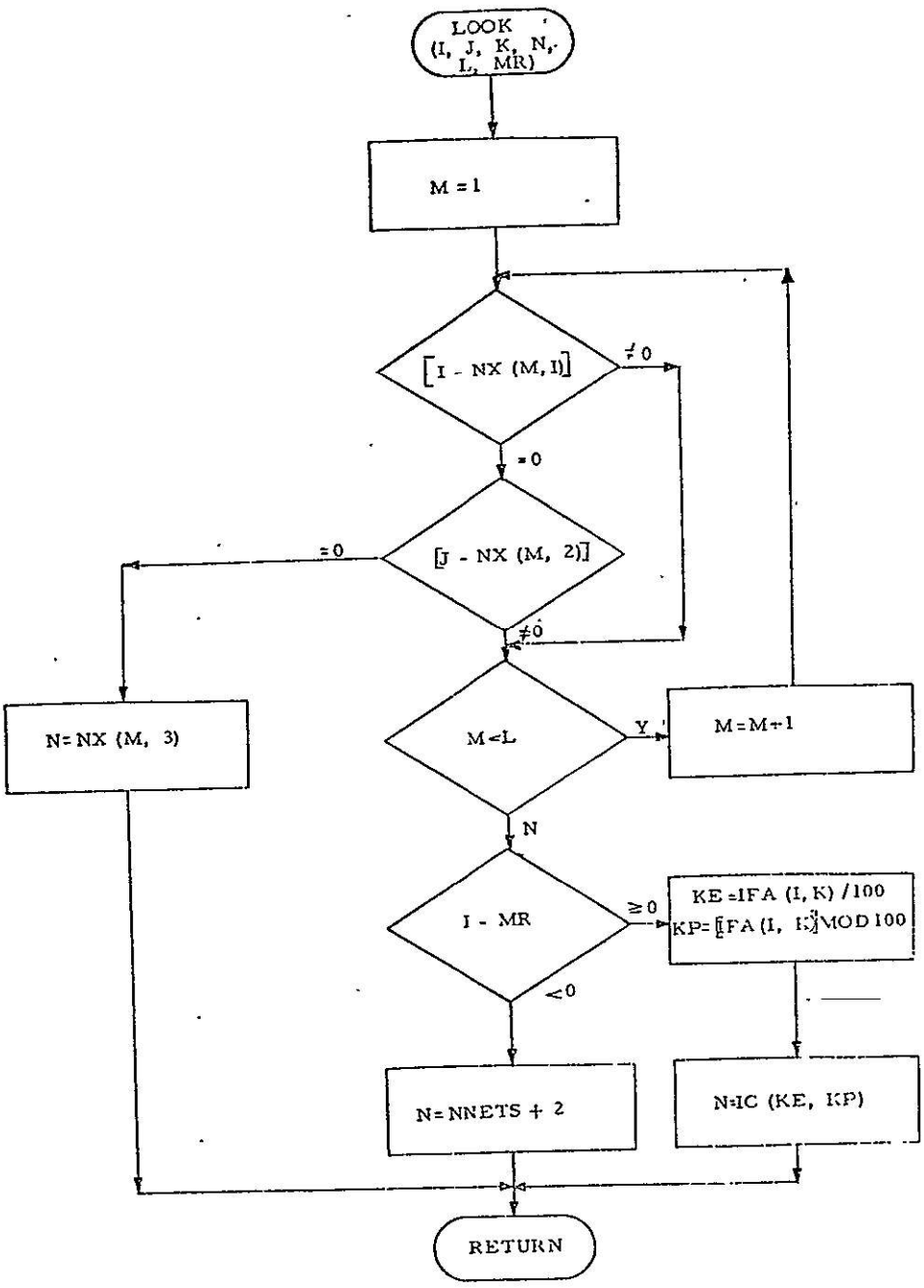


Figure 3-42 LOOK

3.11 Subroutine SQUEEZ

SQUEEZ takes the wiring map formed in IFA by DIDDLE and attempts to compact it in the Y-dimension.

3.11.1 Functional Description

In addition to the primary function of reducing Y-channel requirements, SQUEEZ also performs the secondary functions of sequencing the calling of PADMVE to place side pads and making manual corrections if they are specified. Since the manual corrections reference wires in IFA by their absolute chip locations, the corrections can only be performed on a second pass through SQUEEZ after the ASSIGN subroutine has made absolute coordinate assignments to the wire channels in IFA. The LEN flag is switched by ASSIGN and checked by MAIN to control this sequence.

The first call to SQUEEZ begins with tests for pad folds which are to be referenced by indexes as argument folds of a call to PADMVE. Although this subroutine is called for all pad folds, the fold is not altered unless the input parameters specify that pads are to be moved. The program proceeds to a loop through which each fold passes once. Here a list of all the pin connections contained in the fold is formed in the NC array (Table 3-16). Each point at which a wire connects to a pin or runs to an adjacent fold is designated a wire segment end point, and the appropriate array entry is made for each segment connecting such end points. Only those connections which are found within the fold are identified; left end connections, which are external to folds, are not altered by the routine.

The loop proceeds to call the MOVE subroutine to clear and rewire all of the fold's connections from this list in NC. By sorting and individually running each segment of the fold's final wiring requirements, including those making cross-fold connections, MOVE reduces the number of horizontal wire channels required by the fold. This completes the rewire loop, which is then recycled until the last fold has been processed through. When this occurs, the program proceeds directly to the array refinement coding.

SQUEEZ will be called a second time to insert manual changes, if this option has been selected, and flags direct execution directly to the coding performing this function. This coding individually reads and enters each change from the change cards (whose format is detailed in the PRF User's Manual). Only a limited check is made on the validity of these cards, but errors will show on the array picture output printed by the ASSIGN subroutine. After all change cards are read into IFA, the refinement section of the subroutine is executed for a second time.

NC(I, J)

Format NC-SQ1

J= \ I=	1	2	3	4	5	...
1	0	0	0	3011	0	Cross-Fold Tap Flag
2	9	9	10	10	10	Index to row containing wire segment
3	4	7	16	3	3	Net number of segment
4	7	32	6	29	40	Index to left end of segment
5	30	49	27	40	46	Index to right end of segment

Cross-Fold Tap Flag { = 0 For no cross-fold connection
= 0 For connection to adjacent fold

Typical values describing five fold connections have been entered in the above map.

Table 3-16 SQUEEZ ARRAY

The refinement section of this subroutine is primarily concerned with searching for empty channels which may have been cleared by PADMVE, MOVE, or the manual changes, and deleting these channels from the IFA array. This search is also used to clear the wiring of redundant tunnel ends and other extraneous entries. When horizontal channels are deleted, the vertical wires running across them are restored and the IF and NX arrays which reference the folds are updated.

3.11.2 SQUEEZ Variables and Arrays

The following variables and arrays are referenced by SQUEEZ:

IB - Net number of right end of wire segment.

IFPAD - Although this variable is the argument of the SQUEEZ subroutine, it is not functional in the current program.

IRET - Flag used to sequence through pad adjustments.

IRET = 1 - Top pad fold
IRET = 2 - Bottom pad fold

IT - Net number of left end of wire segment.

KTAP - Flag indicating that a wire segment for which an entry is made in the NC array is connected at one end to the adjacent fold, and should be so flagged in NC(I, 1).

KTAP = 0 - Wire has cross-fold connections
KTAP = 1 - Wire has no cross-fold connections

K12 - X-index to right end of wire segment in IFA.

K7 - X-index to left end of wire segment in IFA.

K9 - Y-index on row of IFA containing wire segment.

LEN - Flag controlling whether or not the subroutine is to be used to make manual changes on this pass.

LEN $\geq$ 0 - Put in manual corrections
LEN $<$ 0 - Bypass manual changes

LL4 - Running count of the number of connections for which rows have been entered in the NC array, format NC-SQ1.

NOP - Number of manual corrections to be executed.

The following arrays are constructed or modified by SQUEEZ:

IXW(I) - Array used in format IXW-A1 to find X-coordinates in IFA of manual changes in the wiring to the left of the folds. The array is cleared after corrections have been made.

IYW(I) - Array used in format IYW-A1 to find Y-coordinate in IFA of manual correction channels. The array is cleared after the corrections have been completed.

NC(I, J) - Array used to store the description of each required horizontal wire of a fold for later use in rewiring by the MOVE subroutine. Each horizontal wire segment which connects any two vertical wires has a column of the array describing it (see array map, Table 3-16). The indexes contained in this column give the location of the original segment in IFA.

3.11.3 SQUEEZ Flowcharts

Flowcharts of the SQUEEZ subroutine are presented in Figures 3-43 through 3-48.

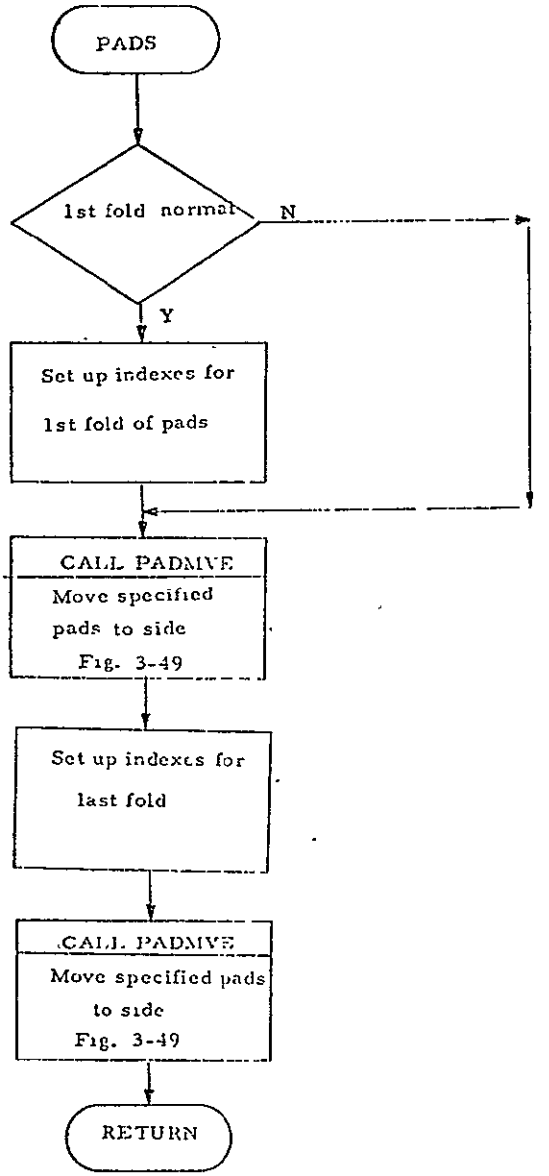


Figure 3-44 PADS

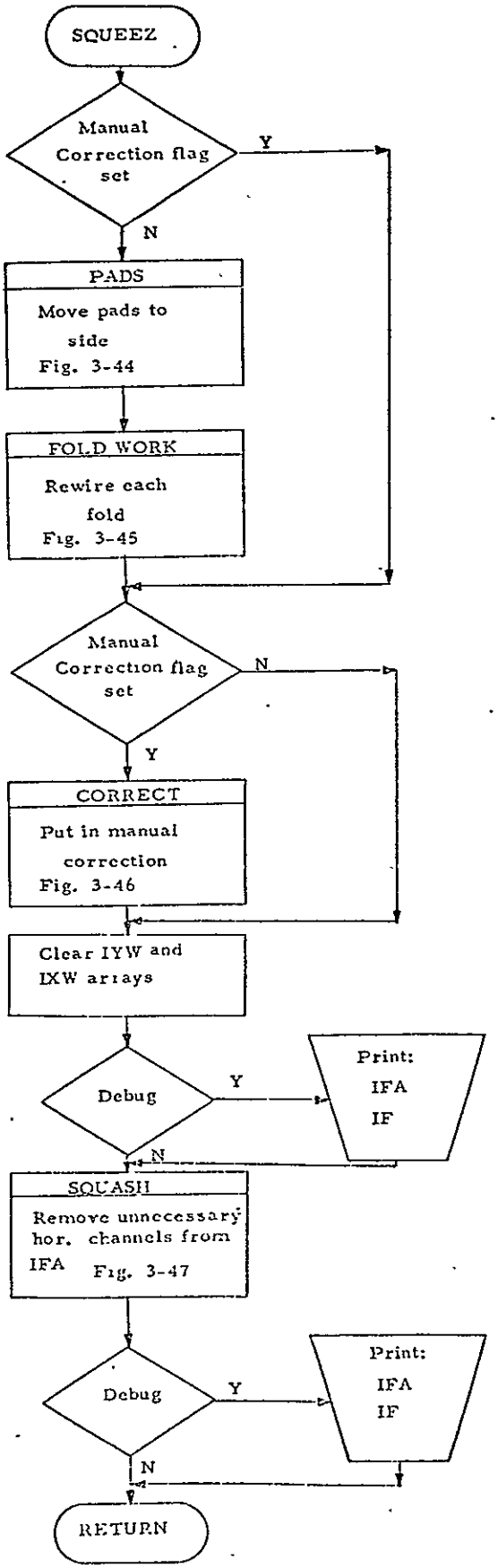


Figure 3-43 SQUEEZ

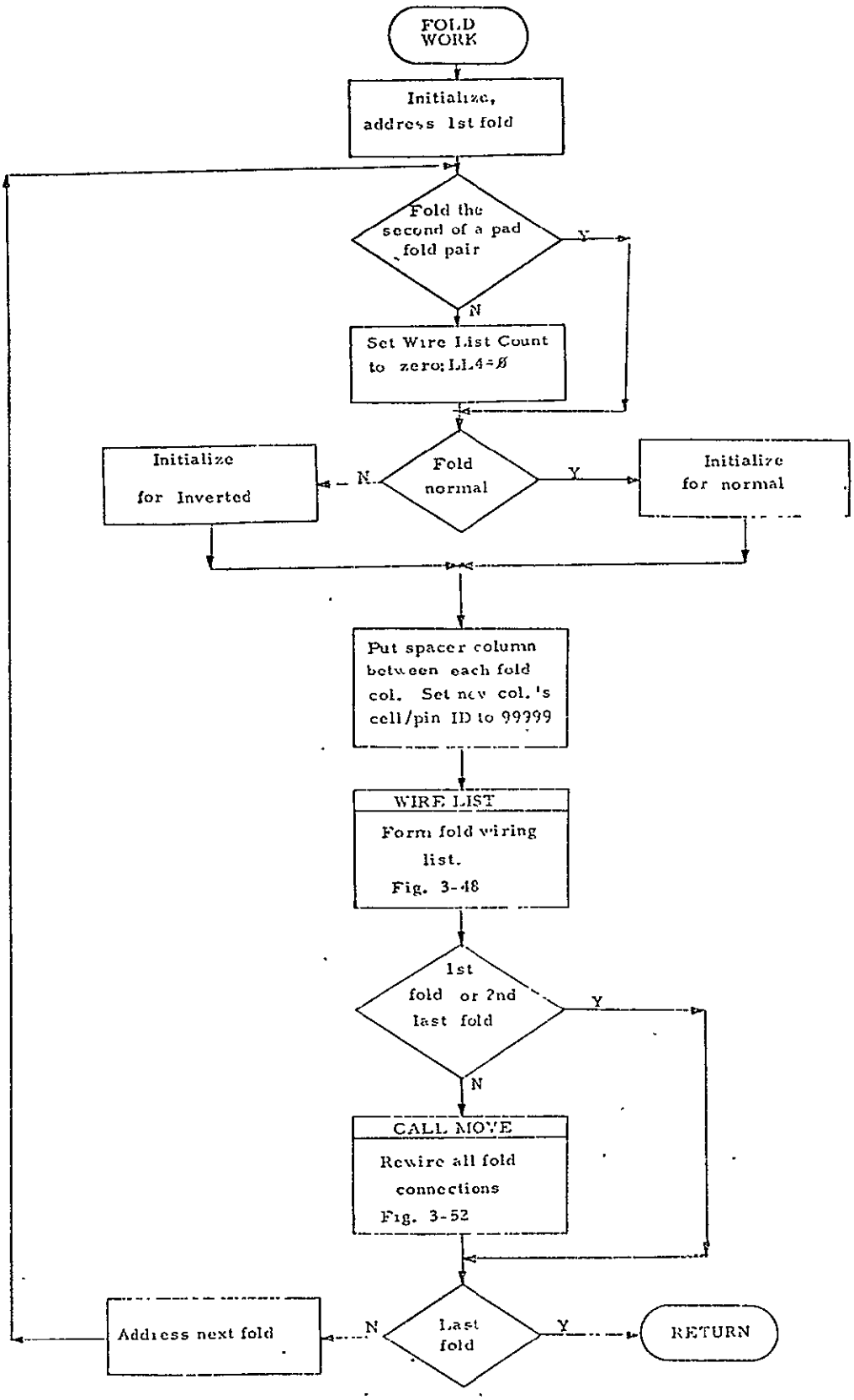


Figure 3-45 FOLD WORK

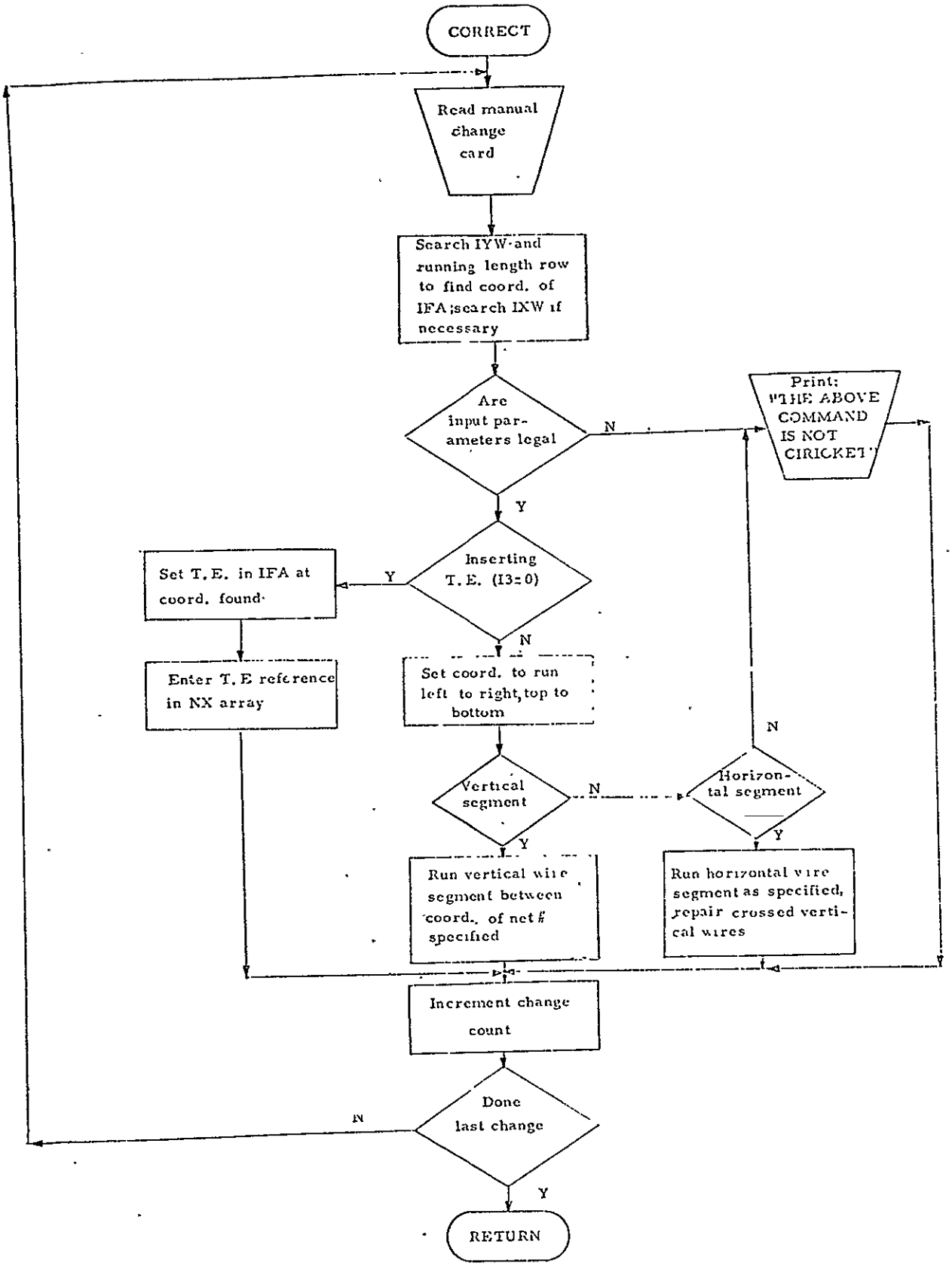


Figure 3-46 CORREC

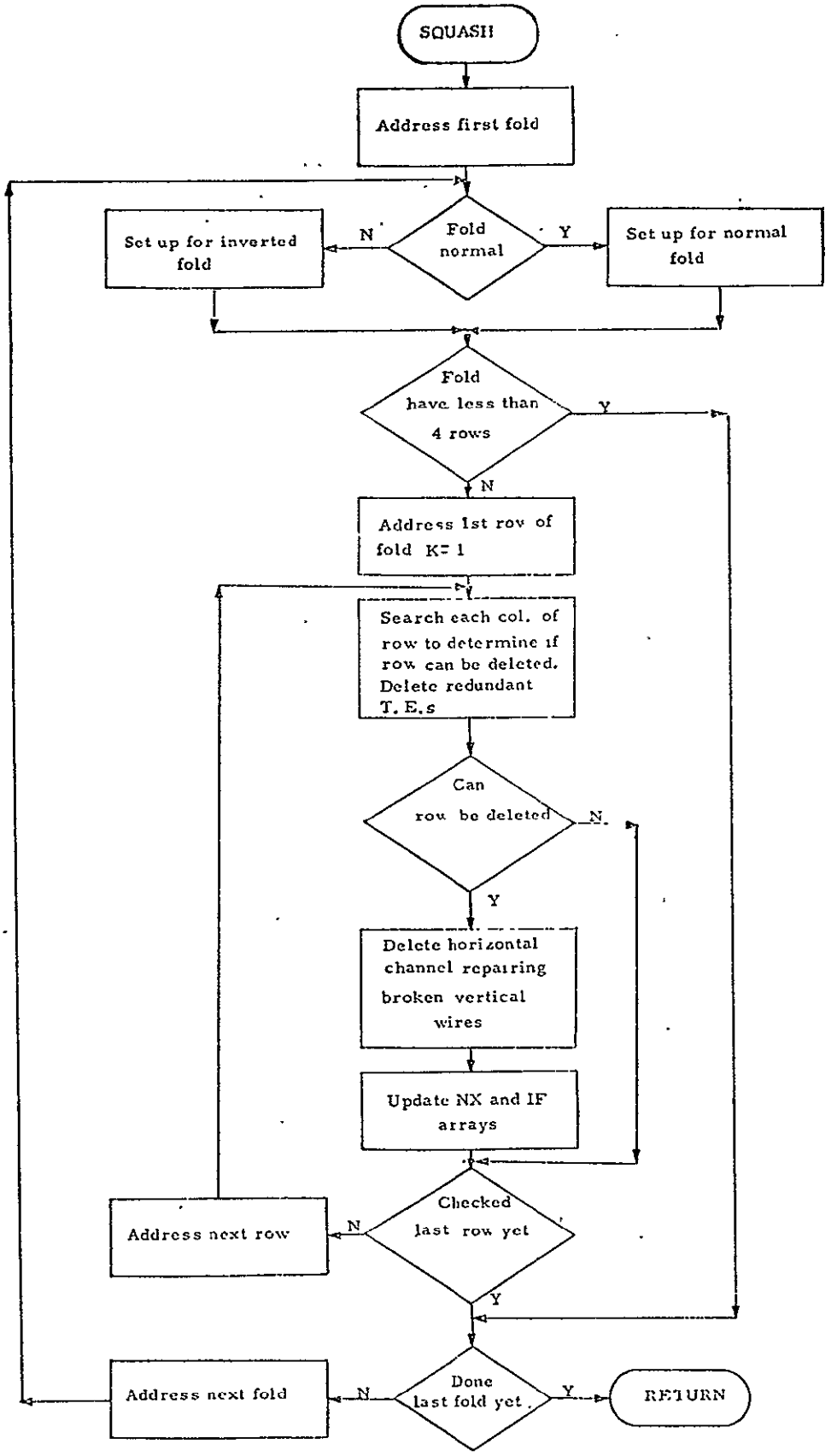


Figure 3-17 SQUASH

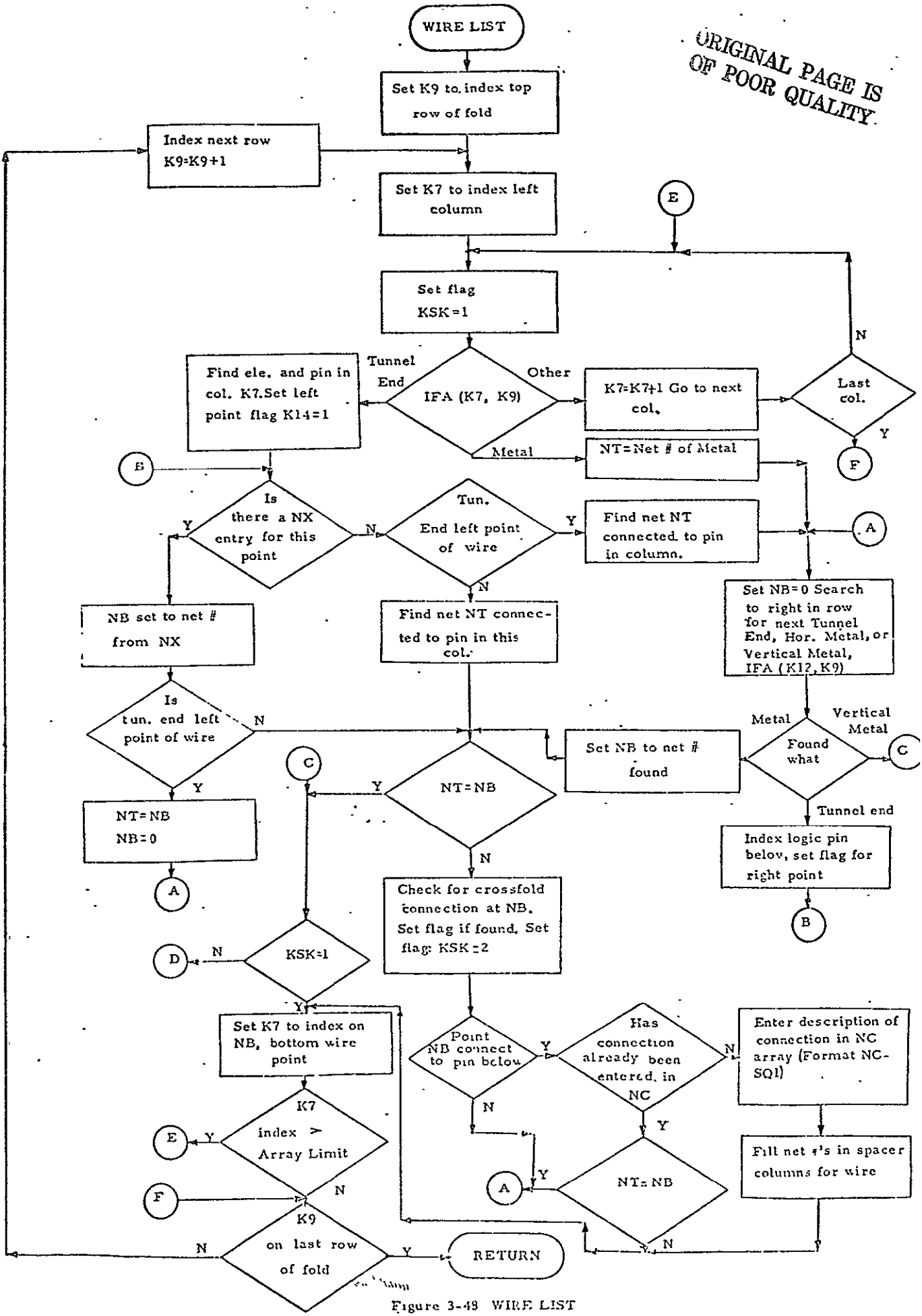


Figure 3-48 WIRE LIST

3.12 Subroutine PADMVE

PADMVE is a short subroutine which is called by SQUEEZ to move pads from the top or bottom of the chip to the left side.

3.12.1 Functional Description

The PADMVE subroutine arguments specify whether the top or bottom fold is to be altered, while the number of pads to be moved are specified in the input parameter array ICN. IFA contains negative numbers in column I which flag the end of each fold. These flags are used by PADMVE to locate the new pads.

There are two types of pad networks which are considered. The first type, handled by the flowchart block SIDE DELETE, consists of a pad which has no other connections in the fold adjacent to the pad fold. In this case the old pad is flagged as being disconnected by setting its former connection pin to 1, and all the unnecessary wiring between the new side pad location and the old pad is deleted.

The second type of network contains connections between the old pad and the adjacent logic fold. In this case the old pad is flagged as being disconnected as before, but only the short vertical wire between the old pad and the adjacent fold can be deleted. A wire from the net connections in this fold is then run out to the left edge and down to the new side pad location.

3.12.2 PADMVE Variables

The following variables are referenced by PADMVE:

ID - Increment to J-index:

ID = -1 for inverted fold
ID = +1 for normal fold

J - Y-index on first wiring row of pad fold.

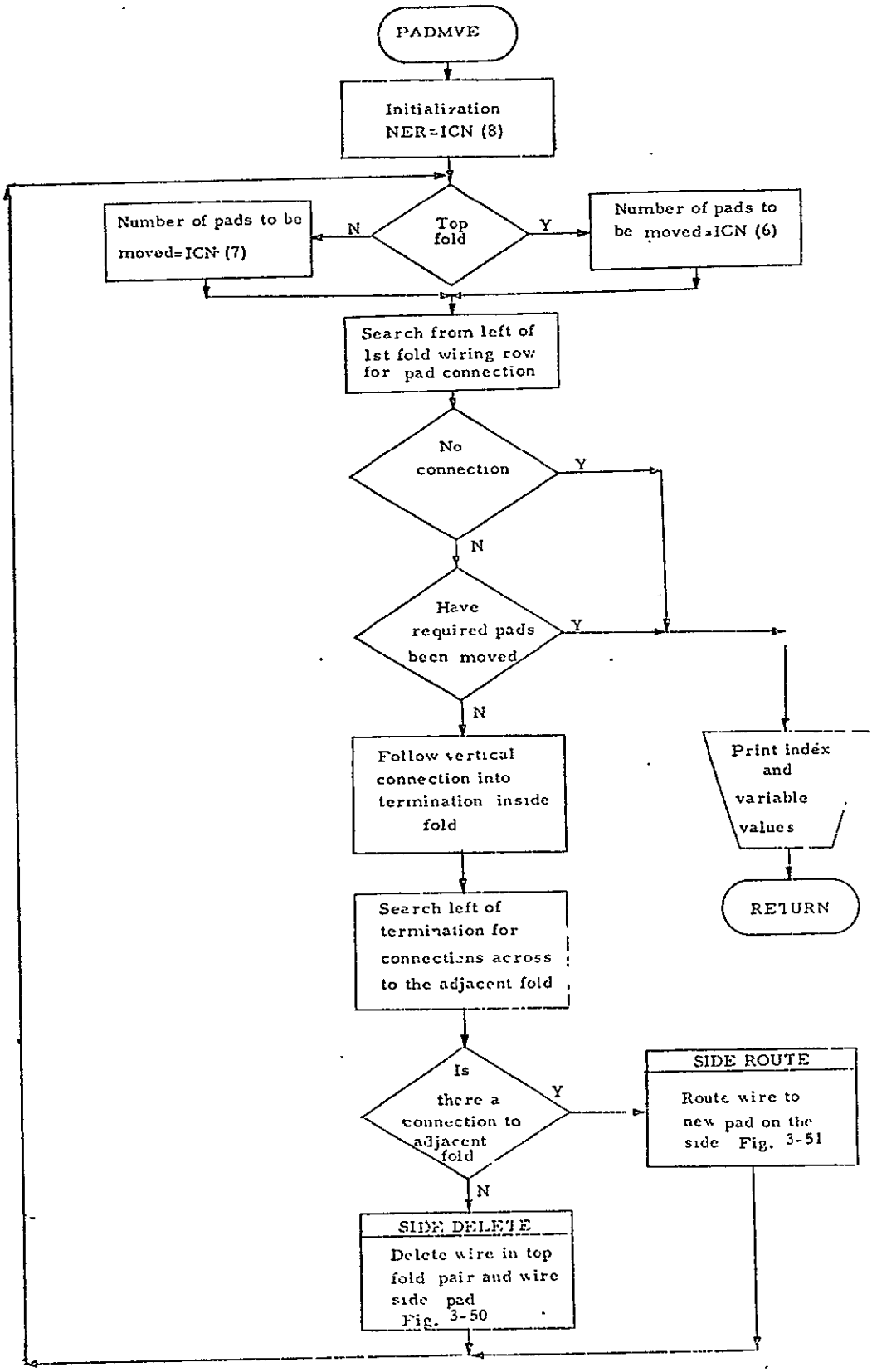
J2 - Y-index on horizontal connection of moved pad in pad fold.

J7 - Same as J2 in the case where the pad is connected to the adjacent fold.

KEC	-	Index to cell/pin ID of the adjacent logic fold.
KM	-	Y-index on new side pad position.
KPC	-	Index to cell/pin ID row of fold.
K4K	-	Index on the last cross-fold wire which has been entered in the NX array.
MT	-	Left edge of fold wiring in IFA.
NA	-	Net number to which displaced pad is connected.
NER	-	X-index on vertical side channels available for side pad wiring. Initialized to ICN(8).
NUM	-	Count on the number of pads yet to be moved.

3.12.3 PADMVE Flowcharts

Flowcharts of the PADMVE subroutine are presented in Figures 3-49 through 3-51.



ORIGINAL PAGE IS
OF POOR QUALITY

Figure 3-49 PADMVE

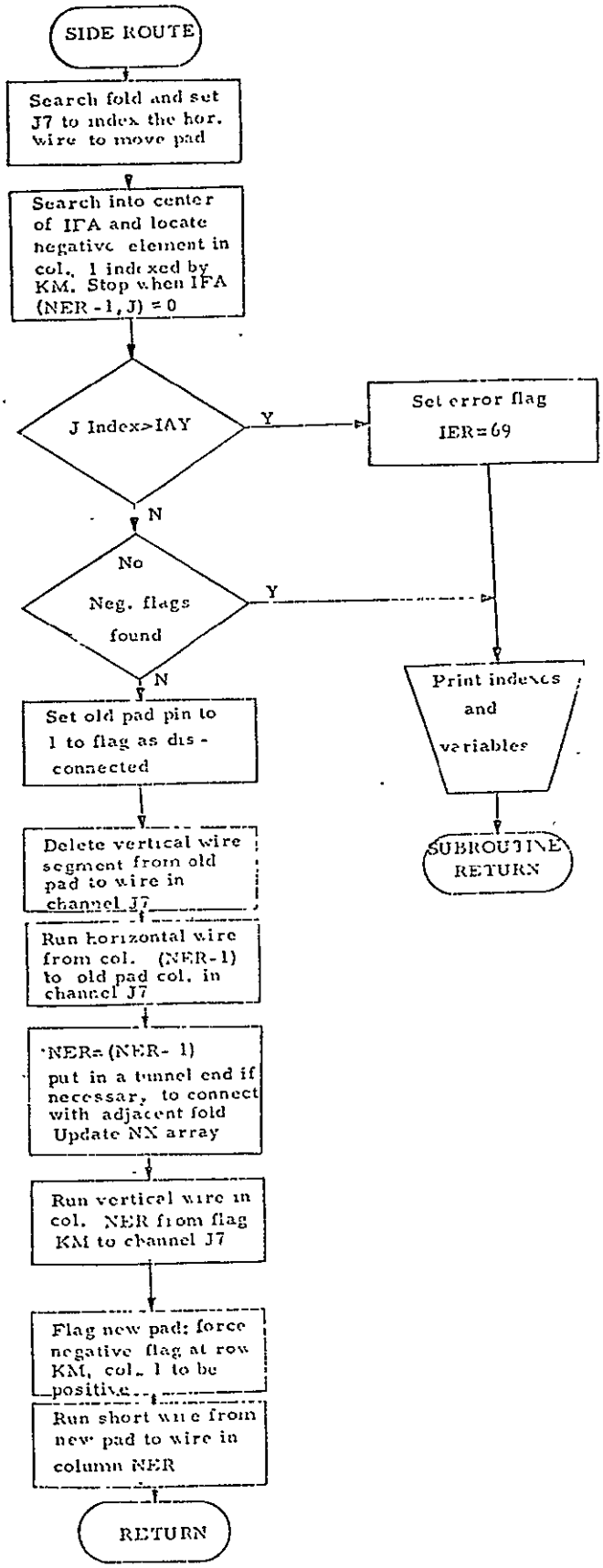


Figure 3-50 SIDE ROUTE

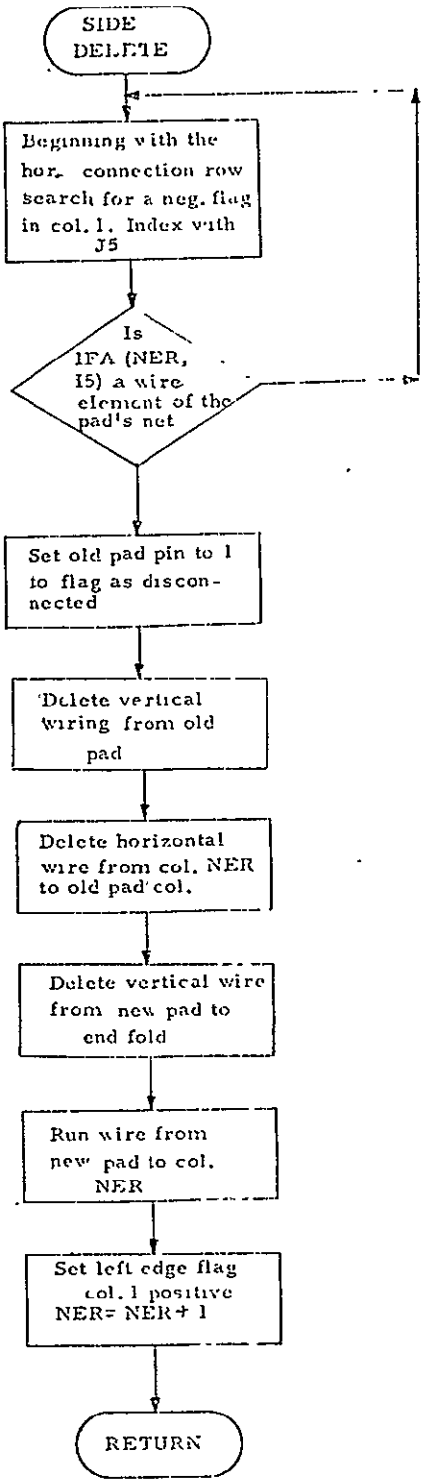


Figure 3-51 SIDE DELETE

3.13 Subroutine MOVE

MOVE is called by SQUEEZ to rewire for each fold of logic cells.

3.13.1 Functional Description

MOVE's function is to clear and rewire the connections within the argument fold from a connection list which has been set up in the NC array. Each column entry in this array represents a horizontal wire segment running between either the left fold edge and a vertical channel, or between two vertical channels. Before the actual wiring is begun, the entries are flagged to indicate which type of wire segment (pad on left end, cross-fold connections, etc.) is described and then ordered in NC by type and increasing length among types.

IFA elements within the fold boundaries are now cleared. In the case of a logic fold adjacent to a pad fold, these boundaries are specially defined to include all pad wiring. The connections of a particular type wire are always run from their order of appearance in the NC array, hence the shortest are entered first. The wiring loop first runs all those wires which connect to left end wiring before going on to the pin to pin wires. If a connection is flagged as a cross-fold type in NC, the routine searches for and runs a connection to the adjacent fold before searching for and running a possible connection to the fold's own cells. The search and run procedure is done first for the right wire segment termination, and then for the left.

If at some time during this process the routine finds it does not have sufficient horizontal channels to make the required connections within the fold, another channel is added and the fold is recycled back to the beginning of the subroutine. Normally, however, MOVE can make the required connections in fewer horizontal channels than previously required. Once all entries in NC are run, the adjoining fold is searched to determine if all required cross-fold connections have been completed. The adjacent fold is searched for connections which are to run directly to the object fold's pins and such wires are run where required.

After all wires have been run, each vertical wiring channel of the fold is examined and cleared of very short metal or tunnel segments and unnecessary tunnel ends. The channel between folds is also cleaned of unnecessary entries and proper cross-fold connections entered before MOVE returns to SQUEEZ.

3.13.2 MOVE Variables and Arrays

The following variables are referenced by MOVE:

IE - Channel increment element which is to be added to the Y-index of a channel in IFA to index channels above or below the base channel.

IE = +1 or IE = -1

IJRL - Number of folds in chip.

INC - Unit increment by which Y-index on fold channels must be increased to look at wires further from fold's cells.

INC = +1 - for normal folds
INC = -1 - inverted folds

ISFE - Switch used to sequence every sixth search of the column into the NX reference deletion coding and back again.

KG - Flag used to indicate whether the last column of IFA has been processed through the tunnel checking loop.

KINSW - Switch indicating whether a recycle pass has been executed.

KINSW = 1 - No recycle
KINSW = 0 - Have done a recycle

KSW - Switch used to indicate whether the wire coming in from the fold's left end, or cap wires, have been completed yet.

KSW = 1 - Cap wires are being run
KSW = 2 - Cap wires have all been run

K4K - Number of entries in the NX array.

K4L - Number of tunnel end references in NX array at start of tunnel checking loop.

LL - Number of the fold being worked upon.

LL4 - Number of entries in the NC array.

MT - X-index on left edge of folds in IFA.

NB - X-index on right end of wire being connected in IFA.

NNO - Number of the net containing the wire being connected in IFA.

NT - X-index on left end of wire being connected in IFA.

The only array format changes made by MOVE are the length/connection type entries made in row one of the NC array:

For connection I:

NC(I, 1) = $[NC(I, 5) - NC(I, 4)]$ - Connection I is a normal wire segment having a horizontal length ≤ 6.

NC(I, 1) = $[NC(I, 4)]$ - Connection I is a normal wire segment having a horizontal length > 6.

NC(I, 1) = $[10,000 - NC(I, 4)]$ - Left end of connection I runs to a pad.

NC(I, 1) = $[3,000 + NC(I, 5) - NC(I, 4)]$ - Connection I has a cross-fold wire tap.

3.13.3 MOVE Flowcharts

Flowcharts of the MOVE subroutine are presented in Figures 3-52 through 3-60.

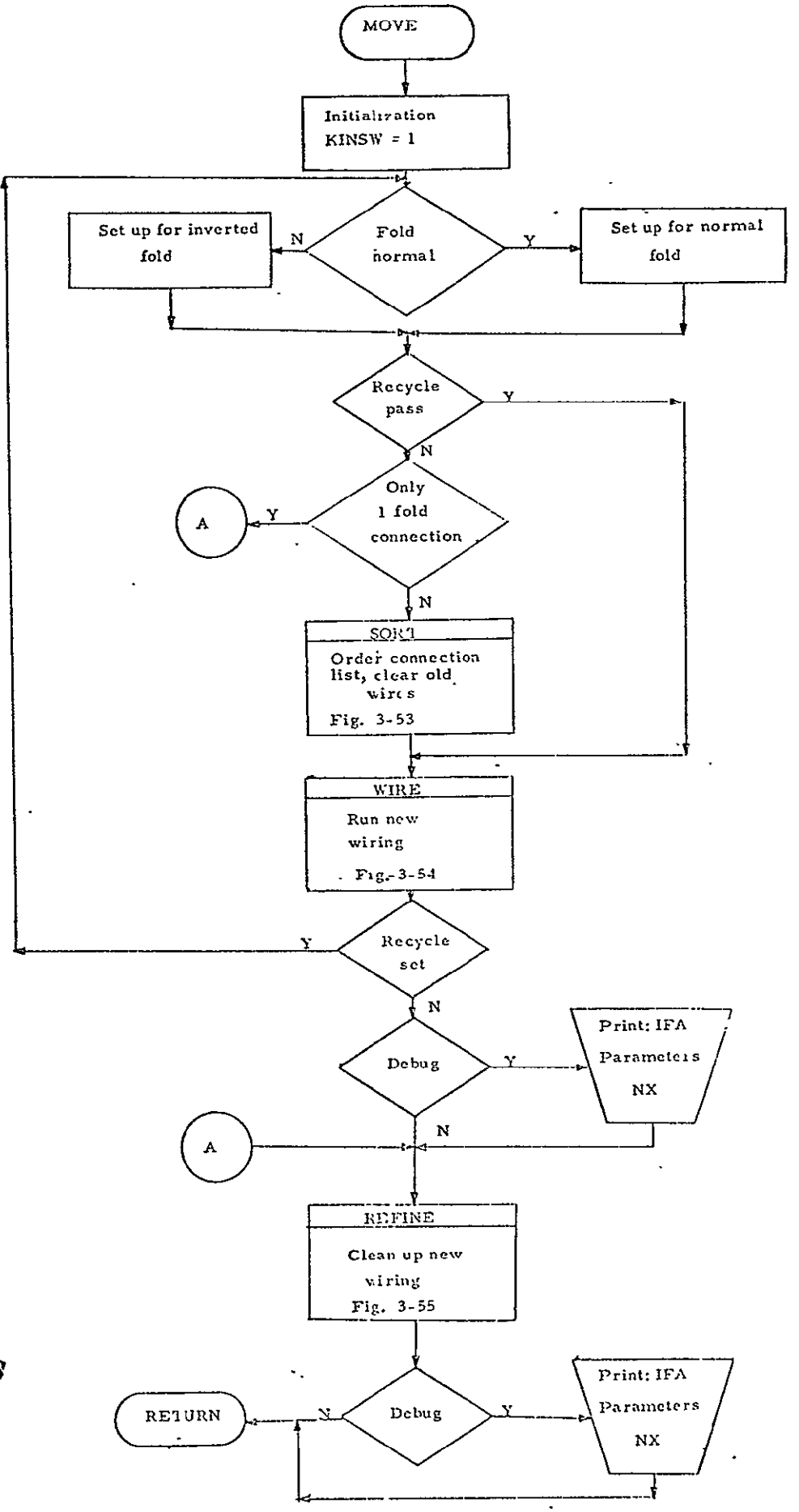


Figure 3-52 MOVE

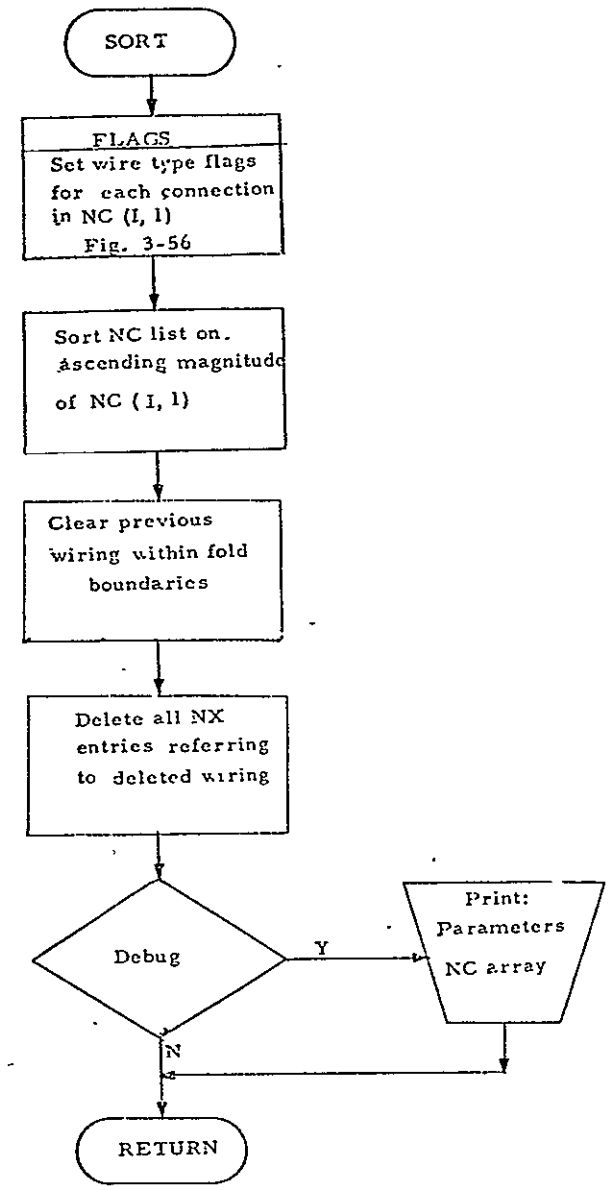


Figure 3-53 SORT

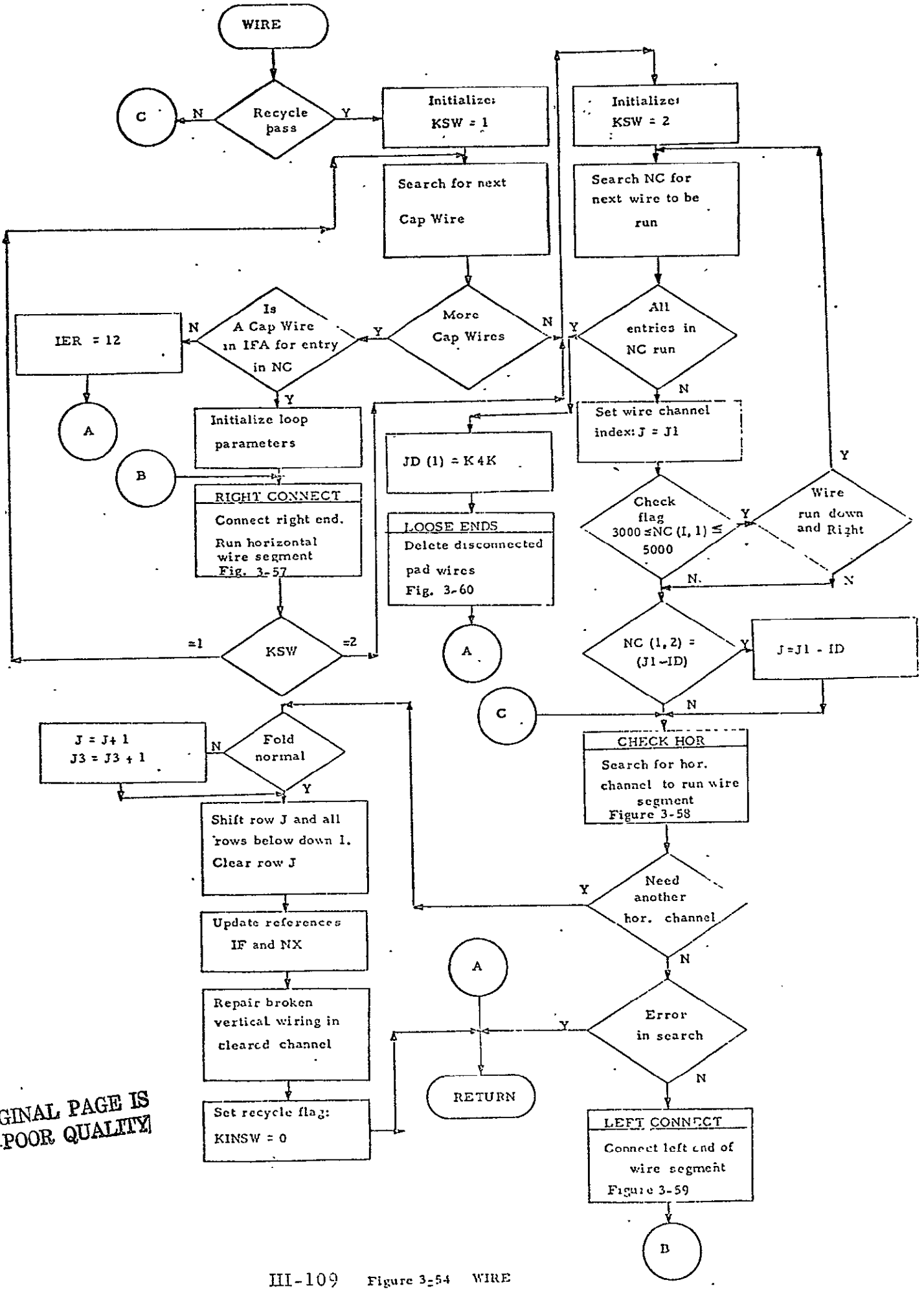


 Figure 3-54 WIRE

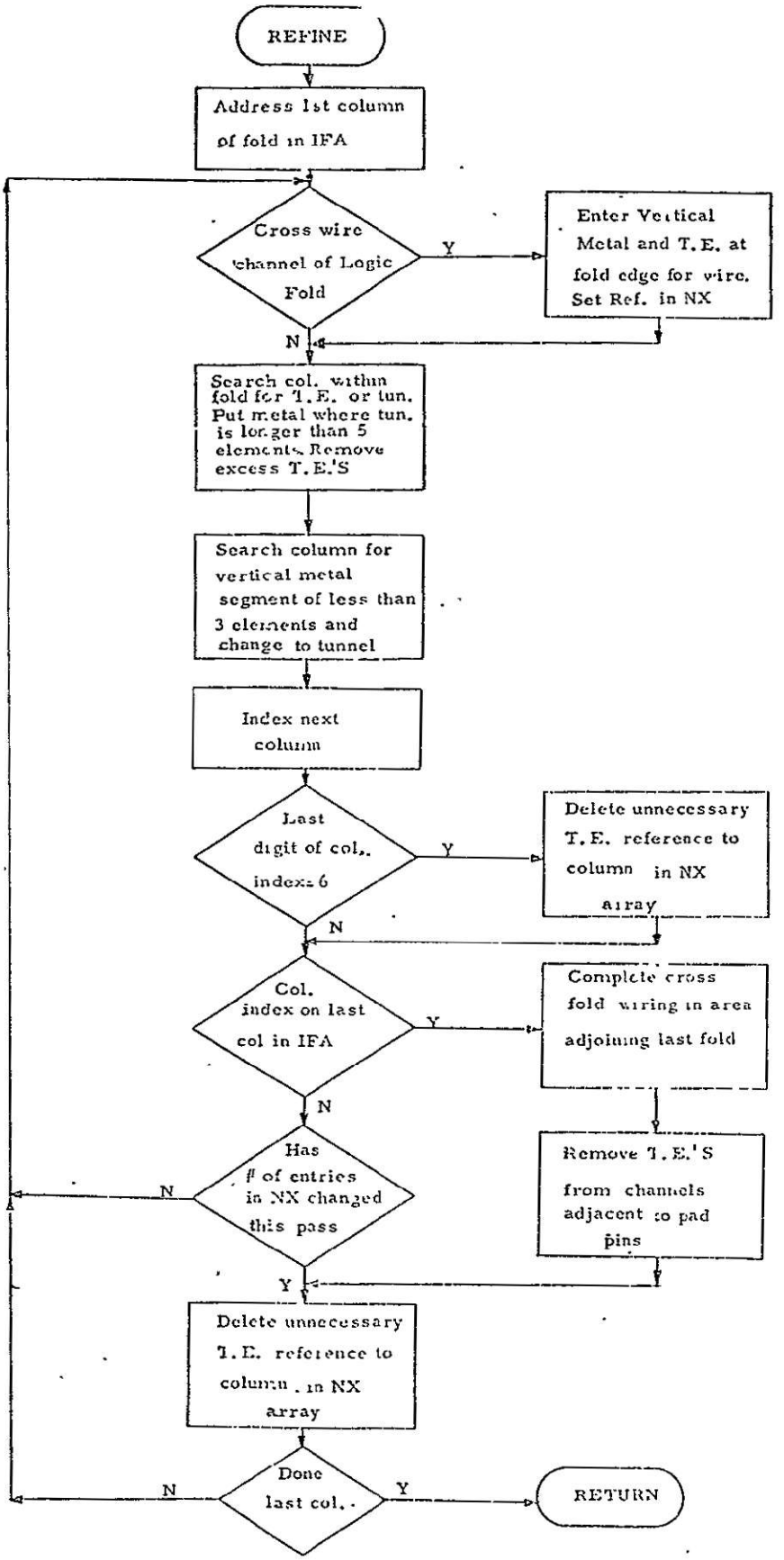


Figure 3-55 REFINE

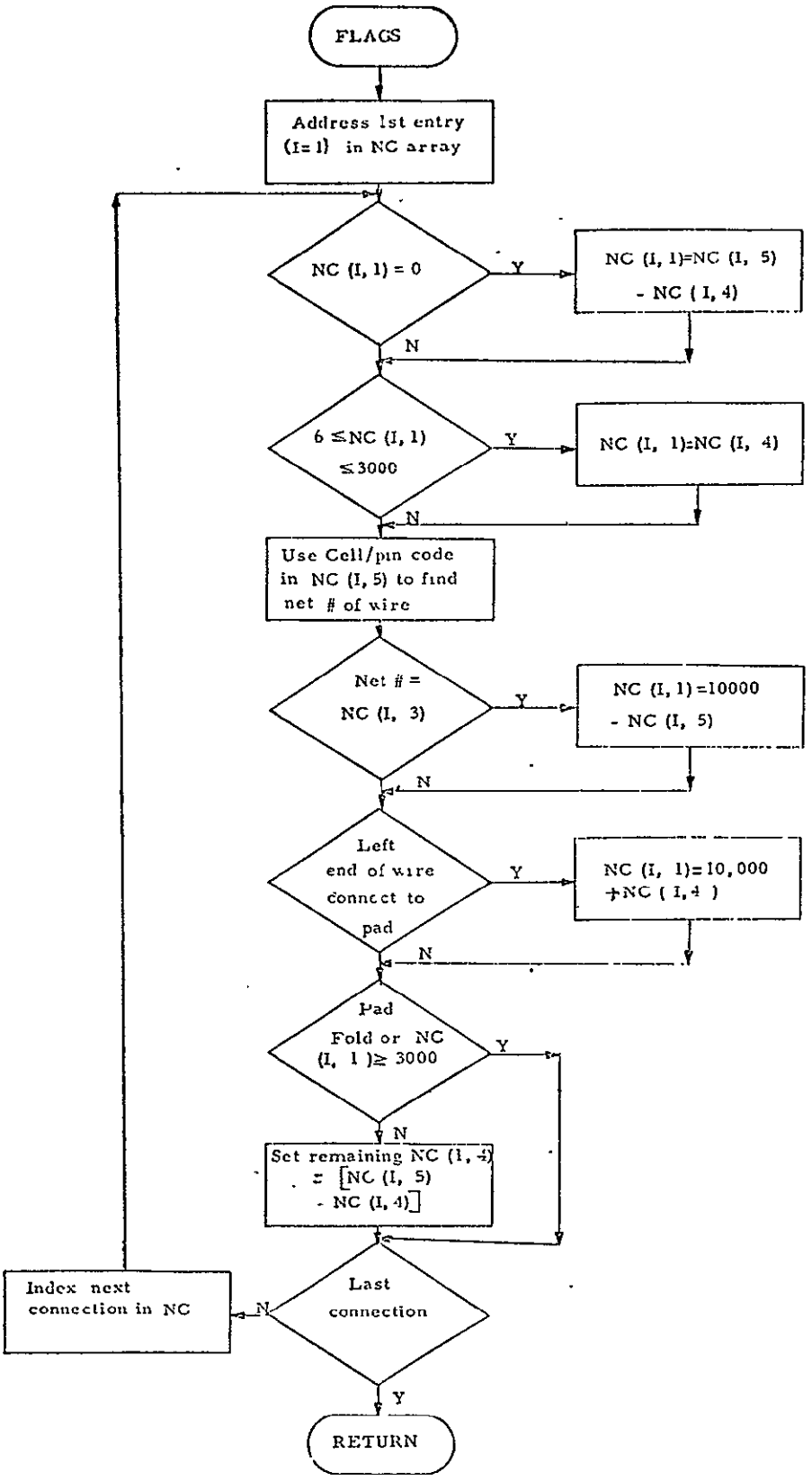


Figure 3-56 FLAGS

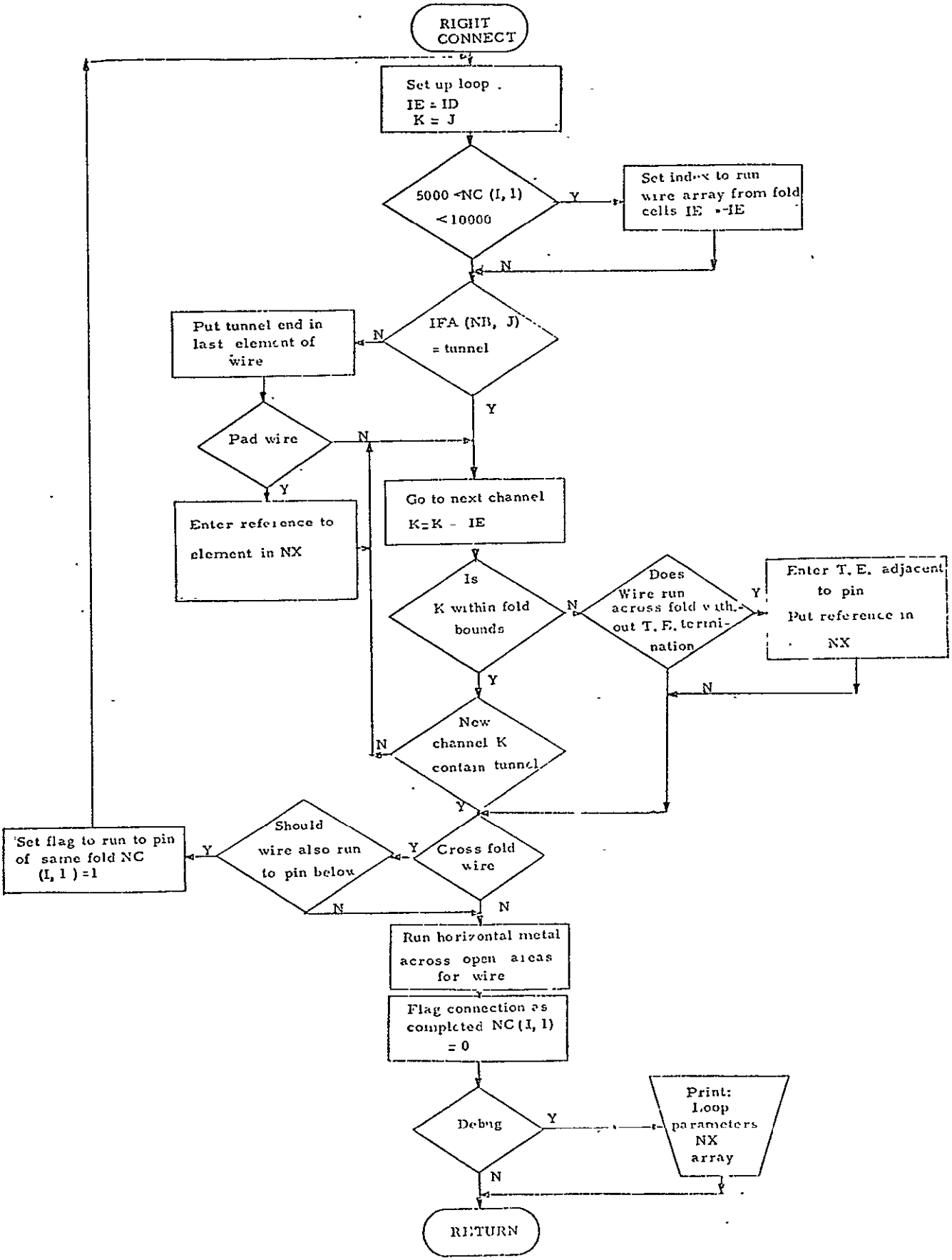
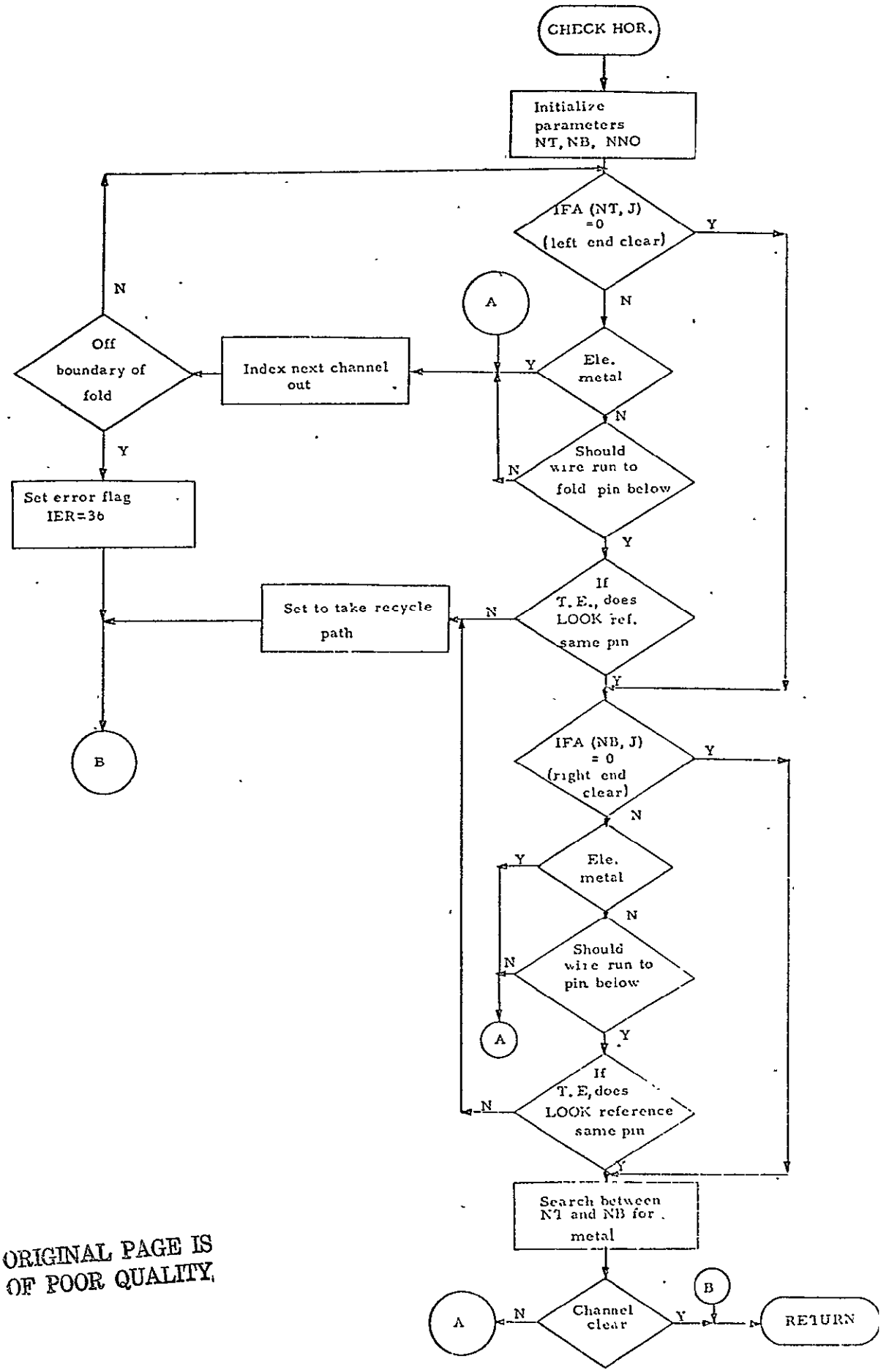


Figure 3-57 RIGHT CONNECT



ORIGINAL PAGE IS
OF POOR QUALITY

Figure 3-58 CHECK HOR

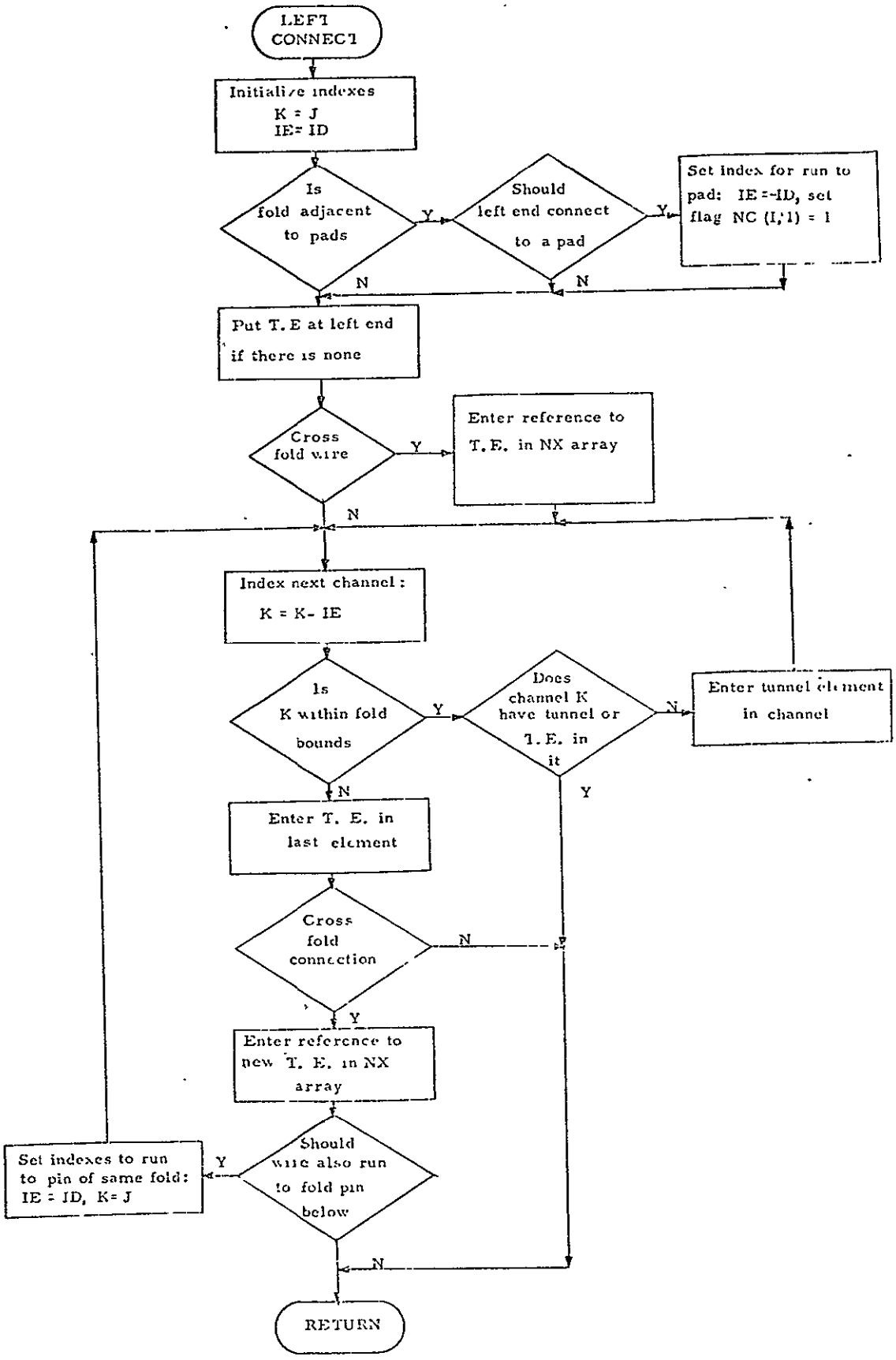


Figure 3-59 LEFT CONNECT

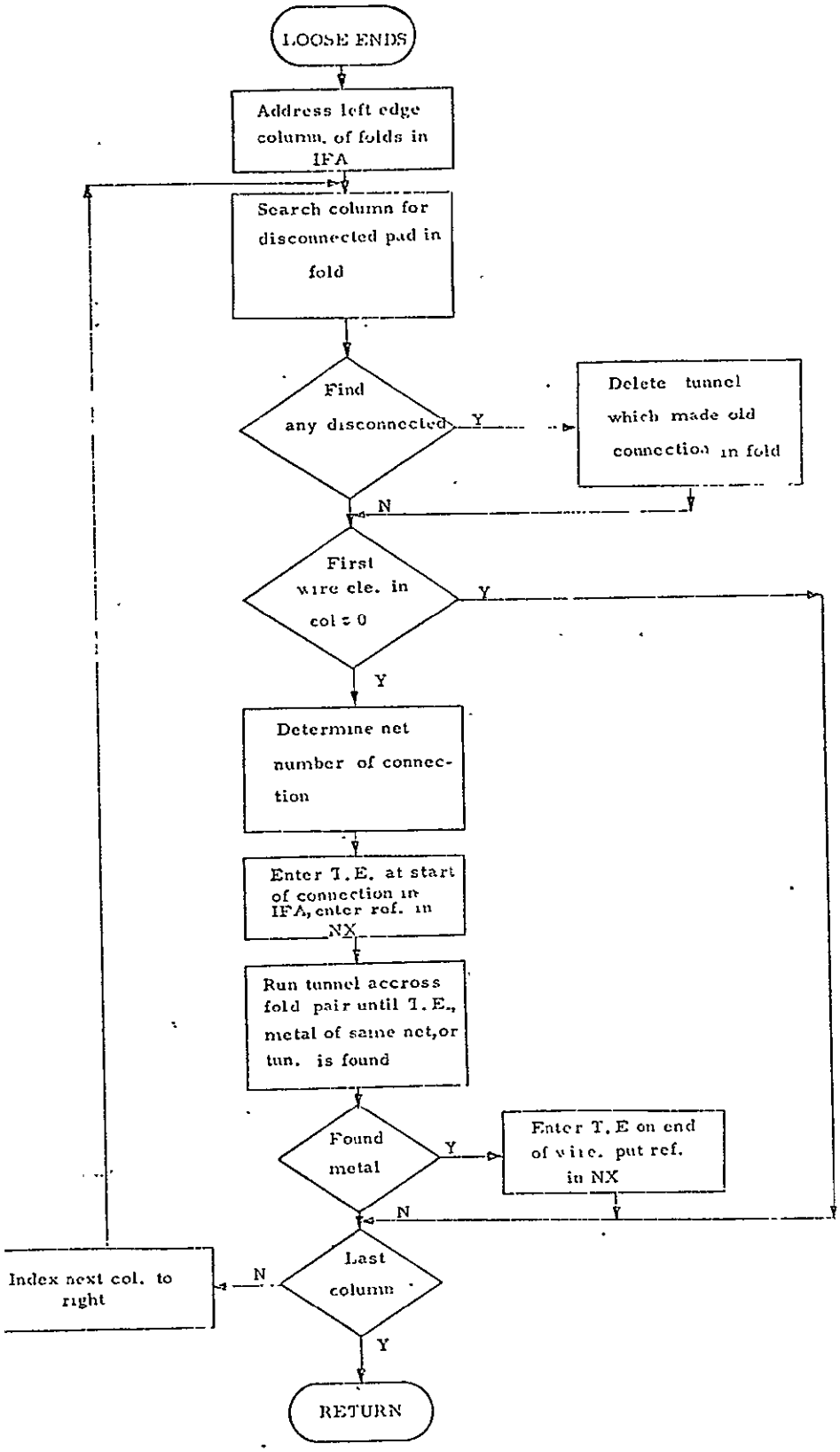


Figure 3-60 LOOSE ENDS

3.14 Subroutine ASSIGN

The ASSIGN subroutine assigns absolute coordinates to the various wiring and pattern components of the chip.

3.14.1 Functional Description

Previous PRF routines have mapped the wiring components into relative locations in the IFA array. All that remains to be done to fully define them on the chip is to utilize the input parameters which specify spacings and wire widths to make coordinate assignments relative to the physical origin of the chip. X-dimension coordinates are first assigned to the left side pads and wiring channels in the IYW array. Fold assignments follow with the translation of running length values into the chip X-coordinates. During this translation, the wiring is examined to insure that no violations of process spacing requirements are made. Wire channels which are to the right of folds proper are assigned coordinates in extensions of fold running length rows.

The IYW array is used to store the Y-coordinates assigned to the rows of the IFA array. An entry is made for each row of the chip picture printout which is assigned a coordinate. The assignments are made by beginning at the top of IFA and utilizing cell size and channel spacing parameters to make assignments to the rows of successive pairs of folds. If crosswires run between the folds of a pair, entries are made in the NC array at this time to describe the wiring of any horizontal adjustment which must be made to compensate for assignment of different X-coordinates in different folds to the same array column. If such a horizontal adjustment wire must be run, Y-coordinates are assigned to allow space for these channels.

After absolute coordinates have been assigned, ASSIGN prints the contents of the IFA array as the chip picture to clearly show the relative locations of wiring and cells. The absolute coordinates assigned to the various channels are also printed, and these must be used if one is to determine the true locations of components on the chip.

3.14.2 ASSIGN Variables and Arrays

The following variables are referenced by the ASSIGN subroutine.

ICH - Cell height; set to IHF.

IFFP - Set to 1.

IFMX - Set to the total number of folds.

ILFP - Set to 1.

IPH - Pad edge to metal wire spacing.

IPWR - Width of channel reserved for power lines.

IWVDD - Distance by which folds are overlapped.

IISP - Correction distance by which space MMSP is reduced in spacing the first horizontal wire above cells.

MAXX - Maximum X-coordinate assigned to a fold.

MMSP - Center to center spacing of horizontal metal lines.

MSLSP - Distance from metal to scribe lines.

MT - X-index to left edge of fold cells.

NMPA - Pass number.

The following arrays are constructed by ASSIGN:

IF(I, J) - For Fold Number I:

IF(I, 1) - Y-index on top of fold in IFA.
IF(I, 2) - Y-index on bottom of fold in IFA.
IF(I, 3) = 0 - Fold I is normal.
= 1 - Fold I is inverted.
IF(I, 4) - Index to first crosswire entry in NC which connects to Fold I (for inverted folds only).

IXW(I) - Array in which the X-coordinate of each vertical channel left of the fold is stored.

IYW(I) - Array in which the Y-coordinate of each row I of the printed chip picture is assigned. IFA rows containing running length values are assigned the Y-coordinate of the top of the cells of a normal fold (bottom if fold is inverted). Cell/pin ID rows have the bottom cell coordinate for normal folds (top for inverted folds).

NC(I, J) - Array containing a column entry to describe each crosswire running between folds. See array map, Table 3-17.

For wire I:

NC(I, 1) - X-coordinate of channel of normal fold containing crosswire.
NC(I, 2) - X-coordinate of channel of inverted fold containing crosswire.
NC(I, 3) - The number of channels which must be used to run wire horizontally to correct for the length displacement between folds.
NC(I, 4) - Direction flag
.= 1 - For NC(I, 1) < NC(I, 2)
= 2 - For NC(I, 1) > NC(I, 2)
NC(I, 5) = [1,000 x crosswire (fold number) + (X-index to channel in IFA)]

PR(I) - Linear array used as buffer storage of one line of data during I/O operations in the generation of the chip picture.

3.14.3 ASSIGN Flowcharts

Flowcharts of the ASSIGN subroutine are presented in Figures 3-61 through 3-63.

NC(I, J)

Format NC-A1

J=	I=1 2 3 4 5 6
1	X-coord. of channel in top fold of pair
2	X-coord. of channel in bottom fold of pair
3	Number of horizontal channels displaced
4	Direction flag
5	Fold/Channel I.D.

Where: Fold/Channel I.D. = [1000 x (fold number) + (X-index to channel in IFA)]

Direction Flag = $\begin{cases} 1 \text{ for } NC(I,1) \leq NC(I,2) \\ 2 \text{ for } NC(J,1) > NC(I,2) \end{cases}$

IF(I, J)

Format IF-RE1

J=	I=1 2 ... 10
1	Fold top index
2	Fold bottom index
3	0 = Normal flag 1 = Inverted flag
4	Crosswire index

Where I = The number of the fold described.

Table 3-17 ASSIGN ARRAYS

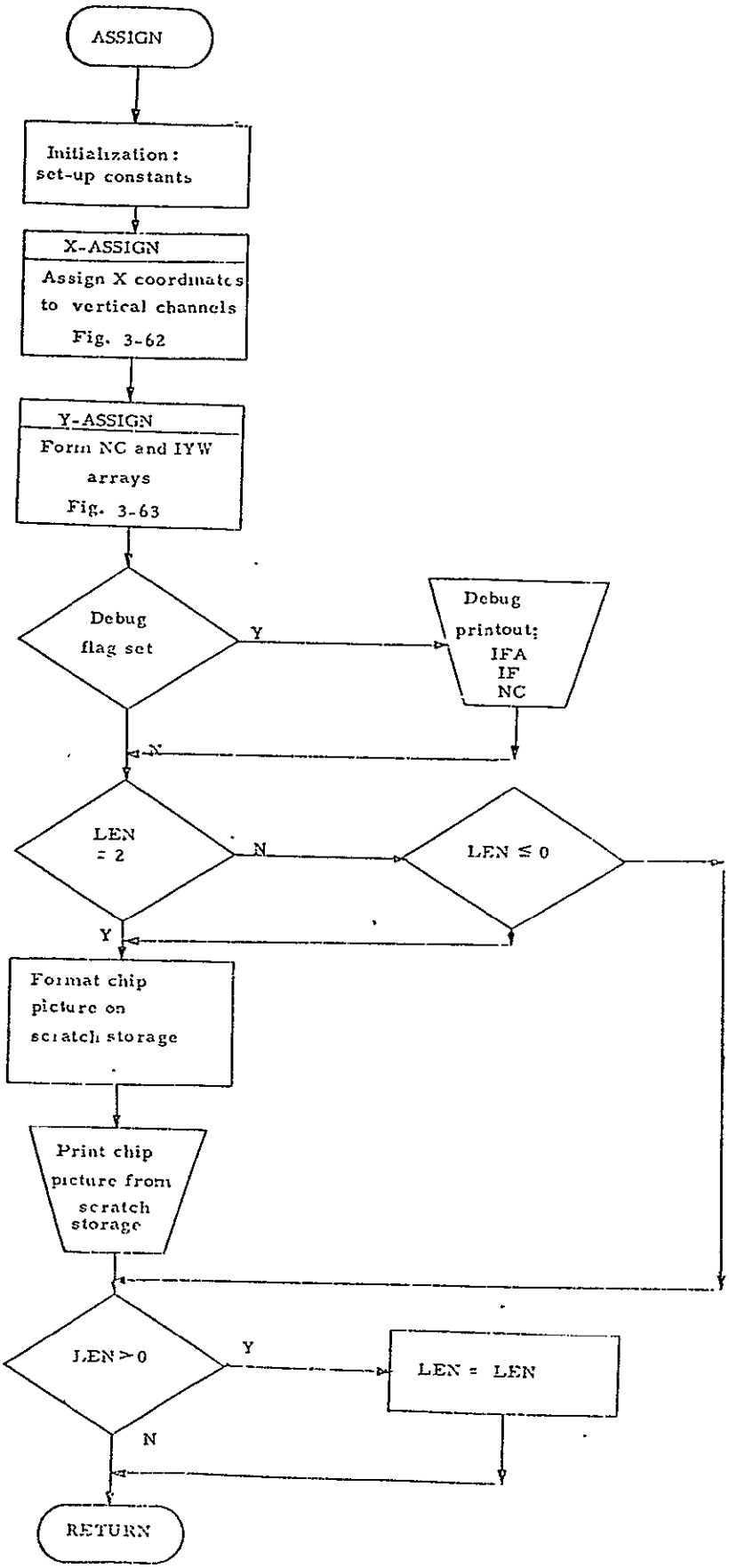


Figure 3-61 ASSIGN

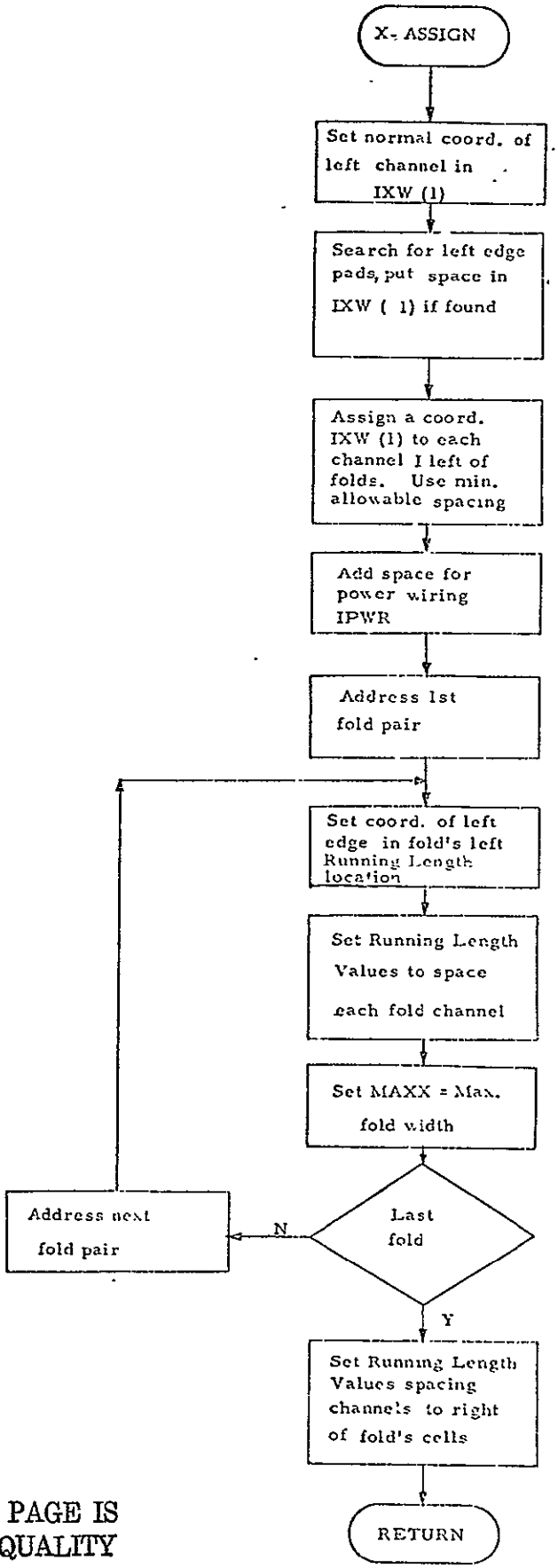


Figure 3-62 X-ASSIGN

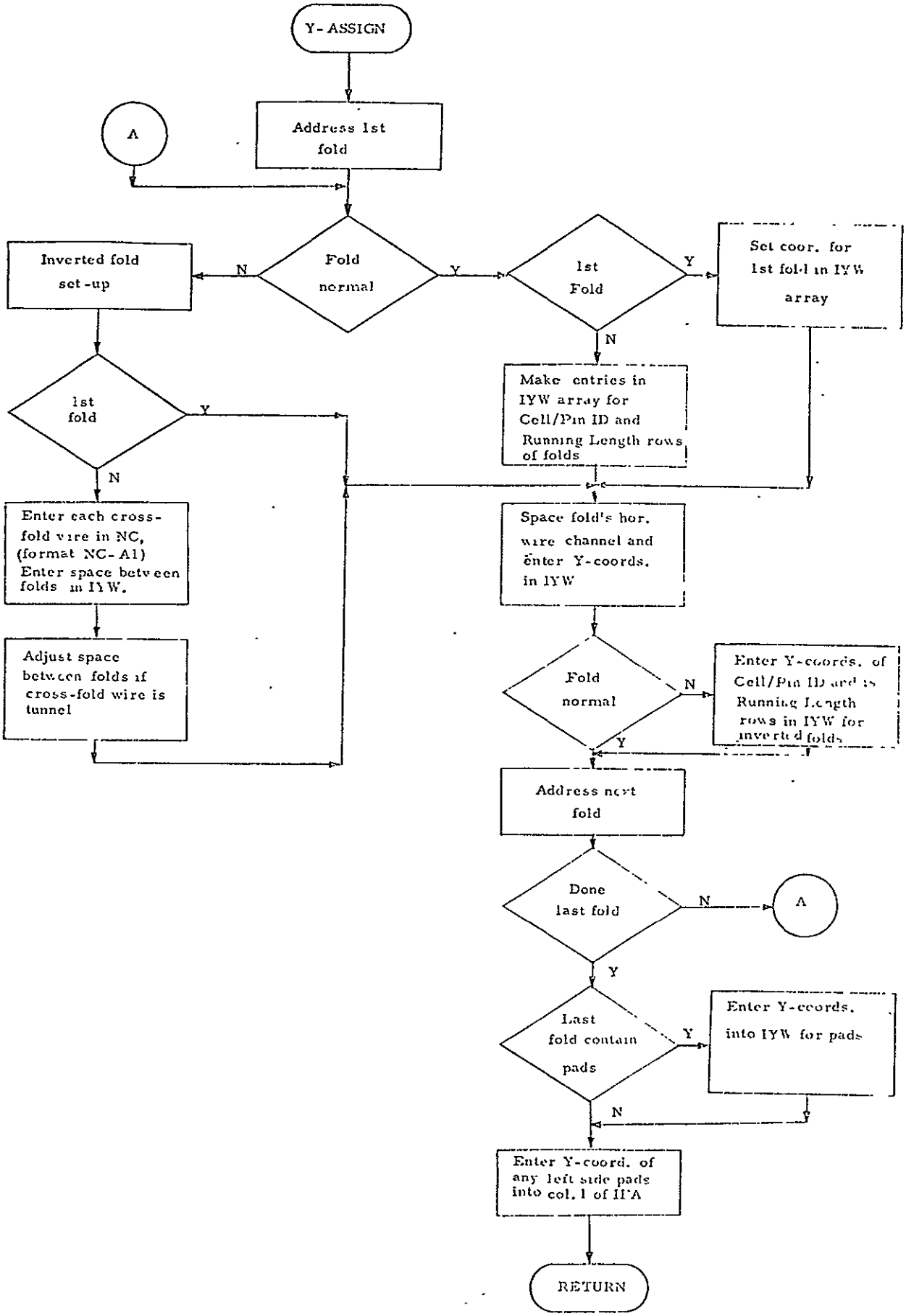


Figure 3-63 Y-ASSIGN

3.15 Subroutine ARTWRK

The ARTWRK subroutine performs the translation from the representation of the chip in PRF arrays into pattern set, line set, and shape set data which can be used by the Banning ARTWORK program to produce plotter commands.

3.15.1 Functional Description

ARTWRK is a relatively straightforward routine which operates sequentially to produce pattern set data and line and shape set commands for each of the four levels of MOS masks. The details of these commands for ARTWORK can be found in the ARTWORK User's Manual. Since the absolute coordinates of all points in IFA have previously been calculated and stored in arrays IXW and IYW, ARTWRK merely searches the array for the various components and uses these arrays to formulate the proper output data in the proper sequence.

The only calculations performed by the routine are those required to determine the position of the test transistor, the power pads, and the chip border, and those which compute the total capacitance of each net. The net capacitances are calculated by determining the net number of each metal segment, tunnel segment, and tunnel end, and adding the associated capacitances to that net's running total stored in the CAP array. This array is printed along with the net list at the end of the subroutine, and can be used as the input for the Banning SIGNAL TRACE program.

3.15.2 ARTWRK Variables

The following variables are referenced by the ARTWRK subroutine:

ICNT - Running count of line set number.

IFMX - Set to IJ = Number of folds.

IG - Flag indicating whether an ARTWORK tape is to be made:

IG $\neq$ 0 - Produce tape on logical unit IG
IG = 0 - Printer output only

ILH - Y-coordinate of lower end of vertical wire.

IRH - Y-coordinate of upper end of vertical wire.

ISW - Switch used in search for vertical metal to indicate which end of a wire is being sought.

0 - Lower end
1 - Upper end

K2K - Set to JD(2) = Number of cross wire entries in NC.

K4K - Set to JD(1) = Number of tunnel end entries in NX.

LEVEL - Flag designating the mask level of the ARTWORK input being generated.

NMPA - Set to ICN(190) - Specify left edge - 15 or 35.

NNO - Net number of wire currently being run.

3.15.3 ARTWRK Flowcharts

Flowcharts of the ARTWRK subroutine are presented in Figures 3-64 through 3-71.

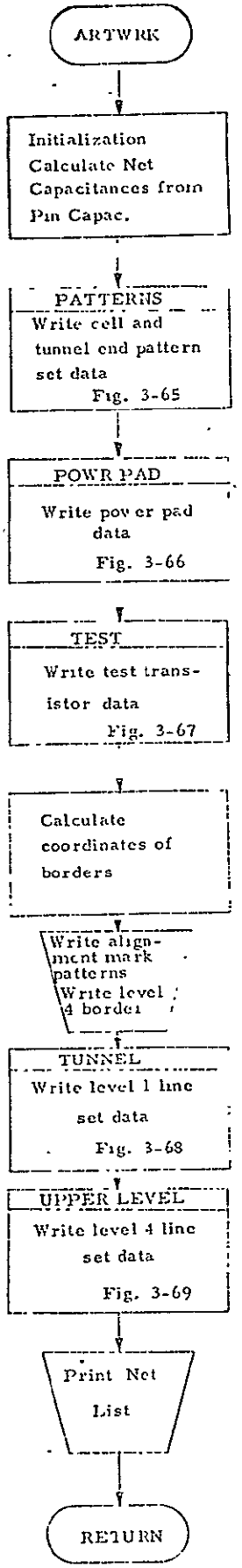


Figure 3-64 ARTWRK

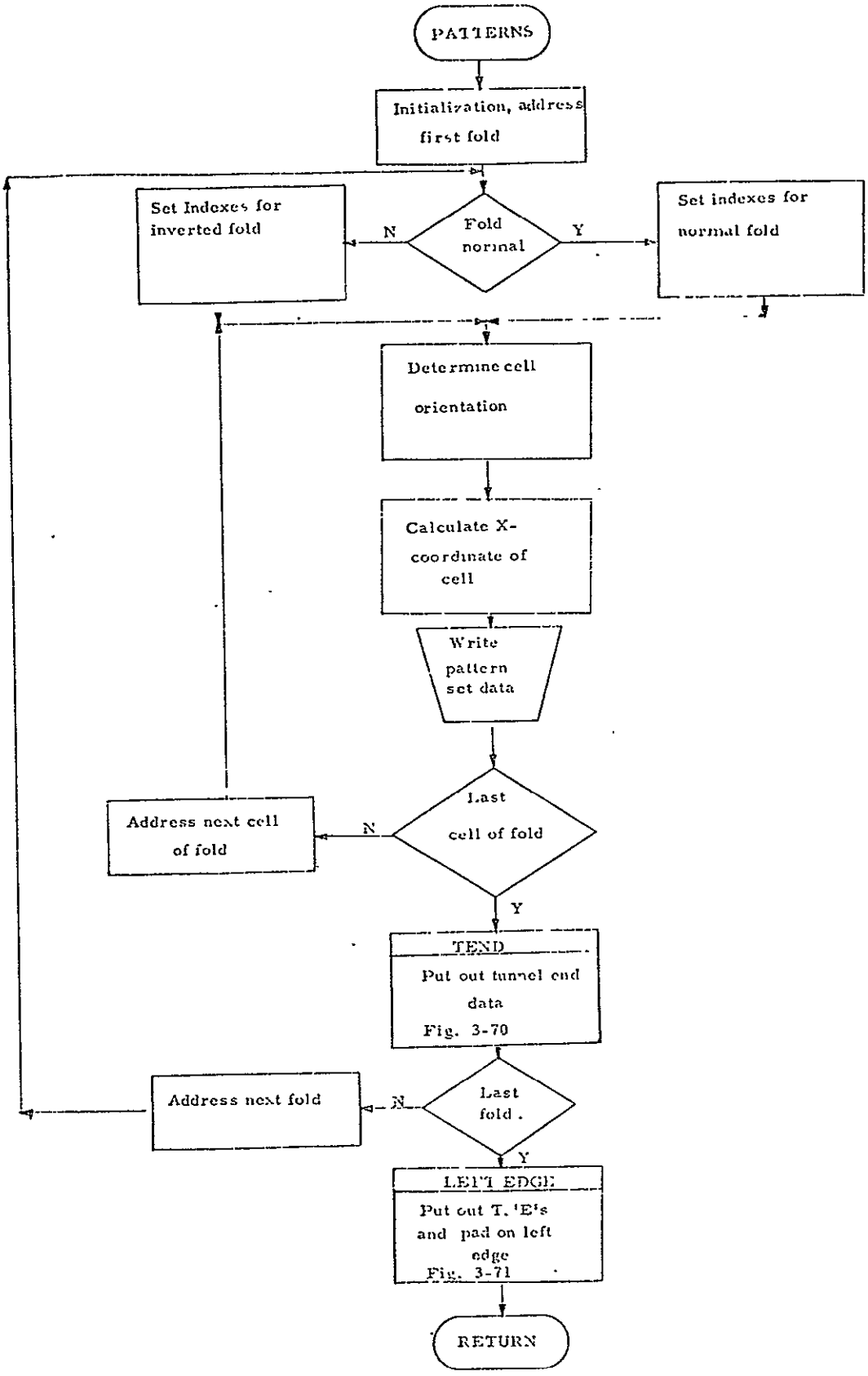


Figure 3-65 PATTERNS

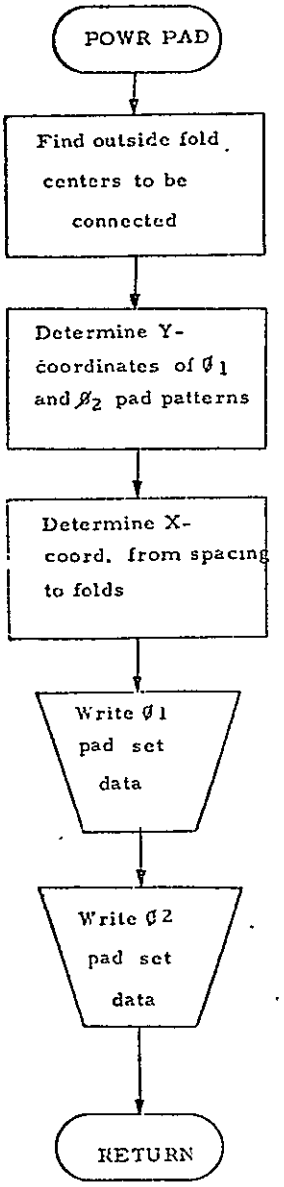


Figure 3-66 POWR PAD

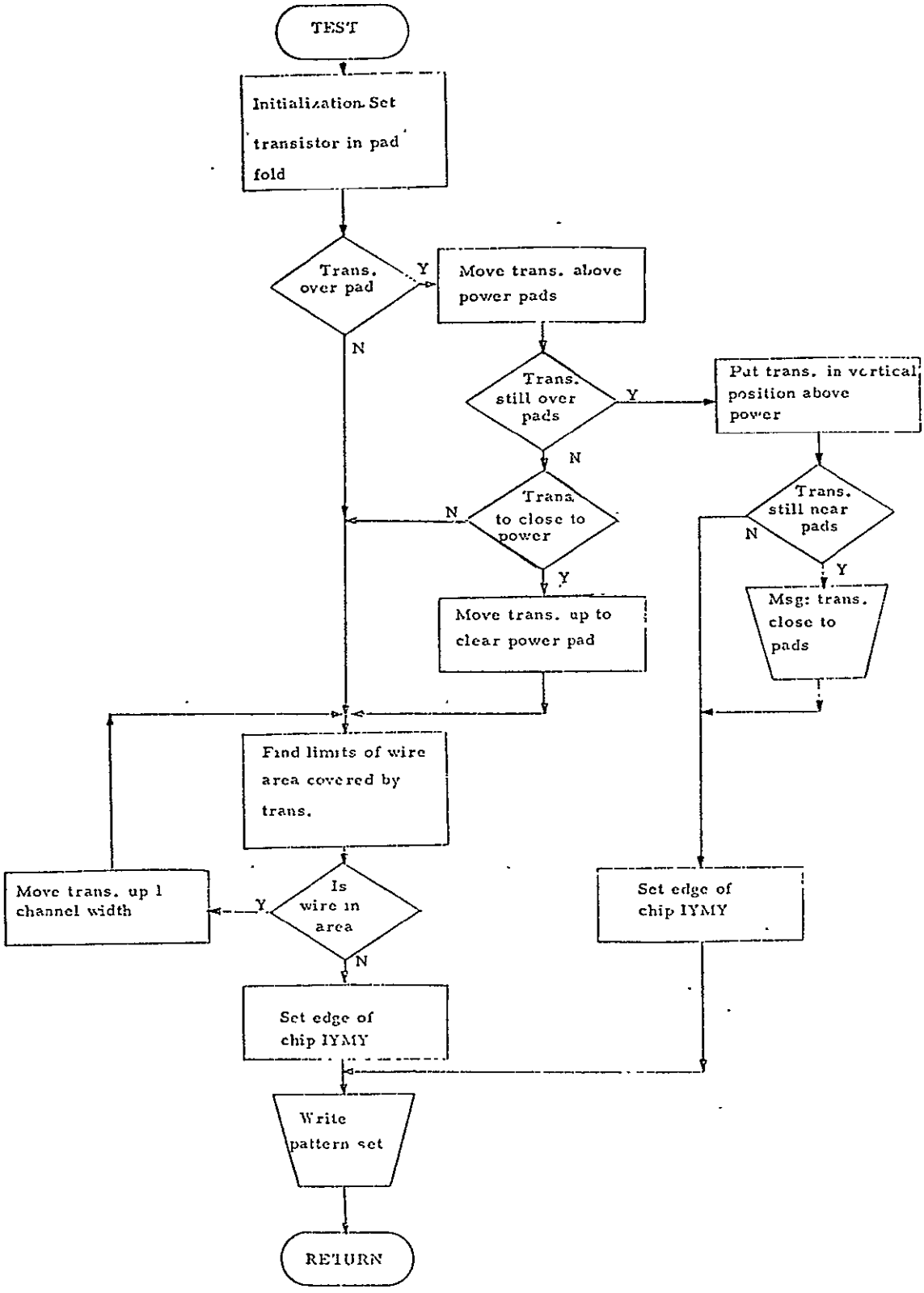


Figure 3-67 TEST

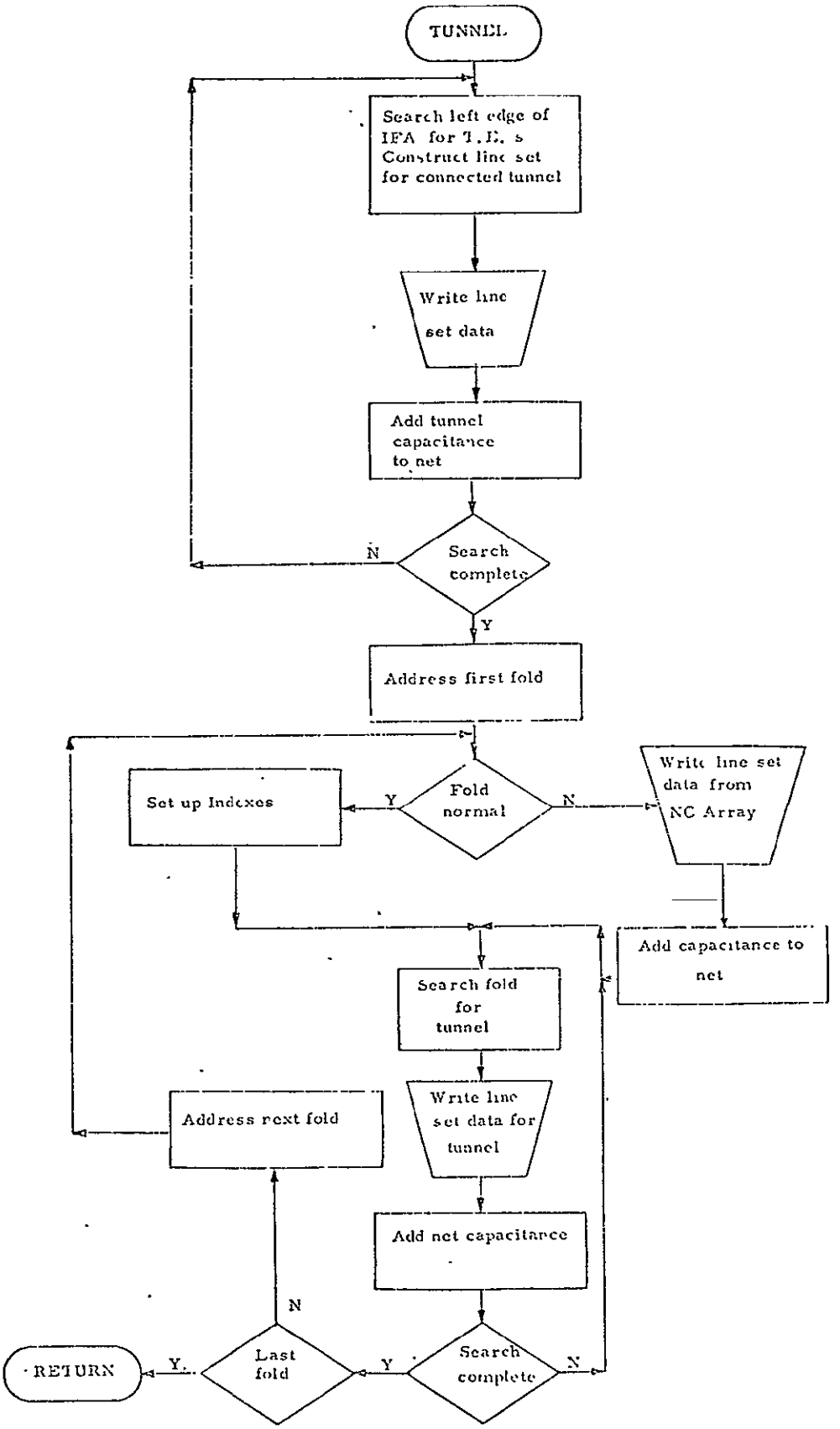


Figure 3-68 TUNNEL

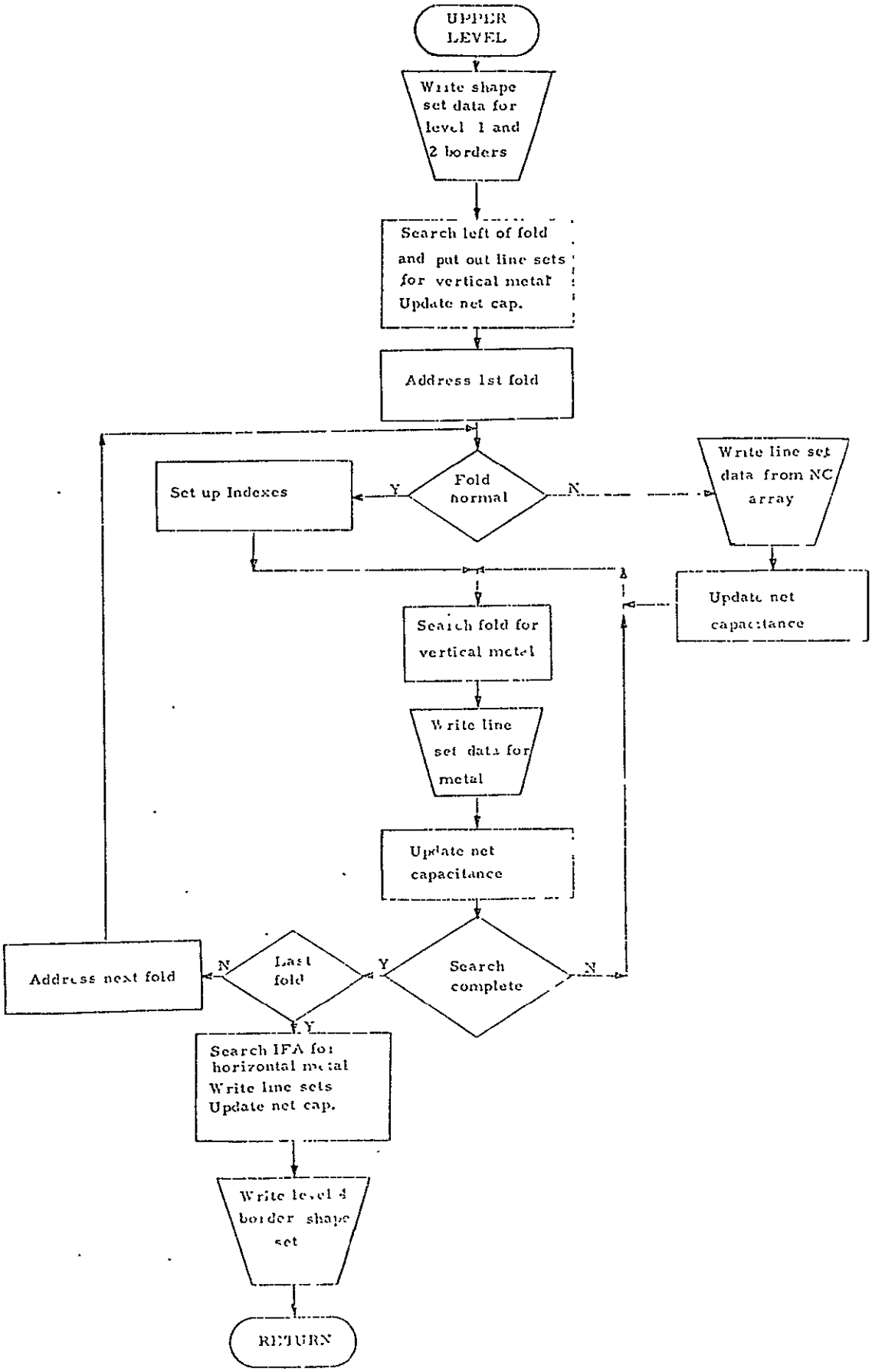


Figure 3-69 UPPER LEVEL

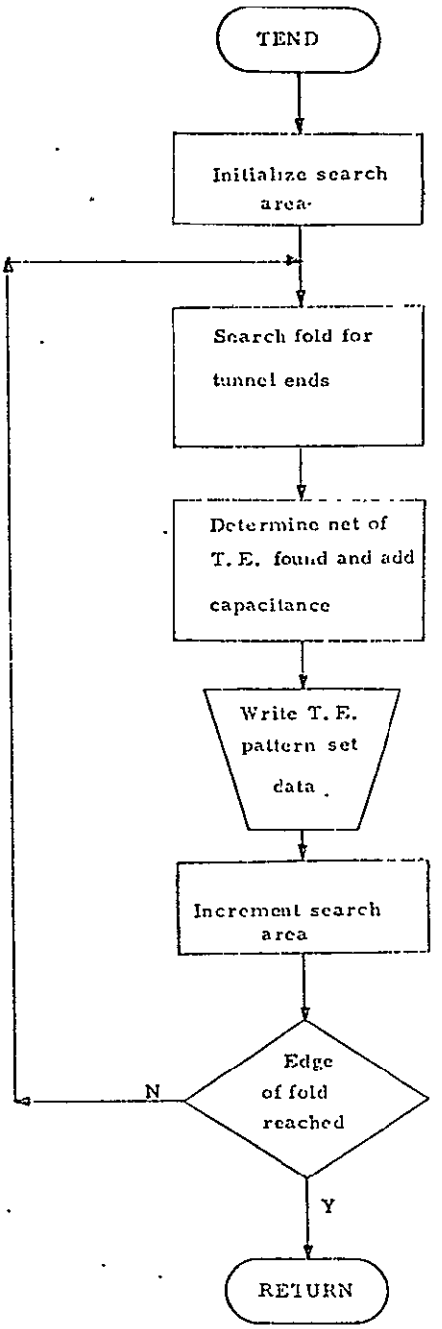


Figure 3-70 TEND

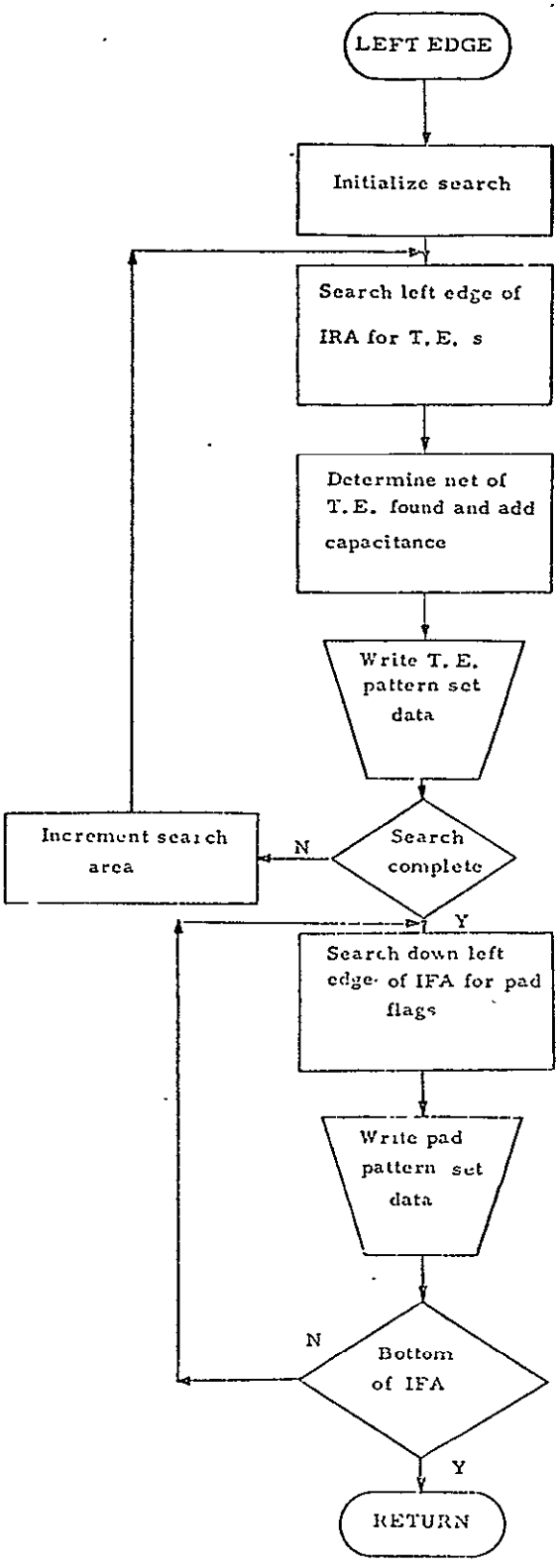


Figure 3-71 LEFT EDGE

3.16 Subroutine POWR

The ARTWORK program inputs required to produce the chip's power busses are generated by the POWR subroutine.

3.16.1 Functional Description.

POWR utilizes coordinate information in array IFA, fold information in array IF, power pad location data passed from ARTWRK in the ICN array, and input parameters stored in ICN to produce the metal line set data for the power busses. Connections to the power pads and between folds are run, as well as the lines running through the fold centers. The basic wiring structure is outlined in the NC array (format NC-P1) prior to wiring. During the formation of this array, nearly all horizontal components of the power system are defined, taking into account the extensions required on both fold ends for interconnection.

The output of line set data then begins with the short connection required to connect power pads to the position where busses will be drawn. A loop producing line set data for each type bus (ϕ_1, ϕ_2, V_{DD1} or GND) is now entered. The operature specified by the associated input parameter is selected, the horizontal lines are drawn from the NC array, and the vertical connection to the previous fold is made, if required. When all busses have been run, the ARTWORK output tape is rewound and control is returned to MAIN to terminate the PRF program.

3.16.2 POWR Variables and Arrays

The following variables are referenced by POWR:

ICN(178) - Flag set during positioning of power pads by ARTWRK.

ICN(178) = 0 - Not enough room for ϕ_2 protection device.
ICN(178) = 1 - Sufficient room for device available.

ICN(179) - Set to NTF in ARTWRK.

NTF = 0 - More than one fold pair.
NTF = 1 - Only one fold pair.

ICN(180) - X-coordinate of origin of ϕ_1 power pad pattern.

ICN(181) - X-coordinate of origin $\emptyset_2$ power pad pattern.

ISW - Switch used to sequence normal or inverted folds through formation of NC array.

ISW = 1 - For first fold inverted
ISW = 2 - For other inverted folds
ISW = 3 - For normal folds

ITX - X-coordinate of left edge of fold to which busses are to be run.

N - Running X-index on entries to NC array.

The following arrays are constructed by POWR:

IAP(I) - Equivalent to input parameters ICN(32), ICN(33), ICN(34), ICN(35).

IBI(I) - Equivalent to input parameters ICN(28), ICN(29), ICN(30), ICN(31) for I - 1, 2, 3, 4.

ITI(I) - Equivalent to input parameters ICN(24), ICN(25), ICN(26), ICN(27) for I - 1, 2, 3, 4.

NC(I, J) - Array used to list the coordinates of the power busses of each fold. Each logic fold will have three columns entered in format NC-P1, illustrated in the array map, Table 3-18.

3.16.3 POWR Flowcharts

Flowcharts of the POWR subroutine are presented in Figures 3-72 through 3-74.

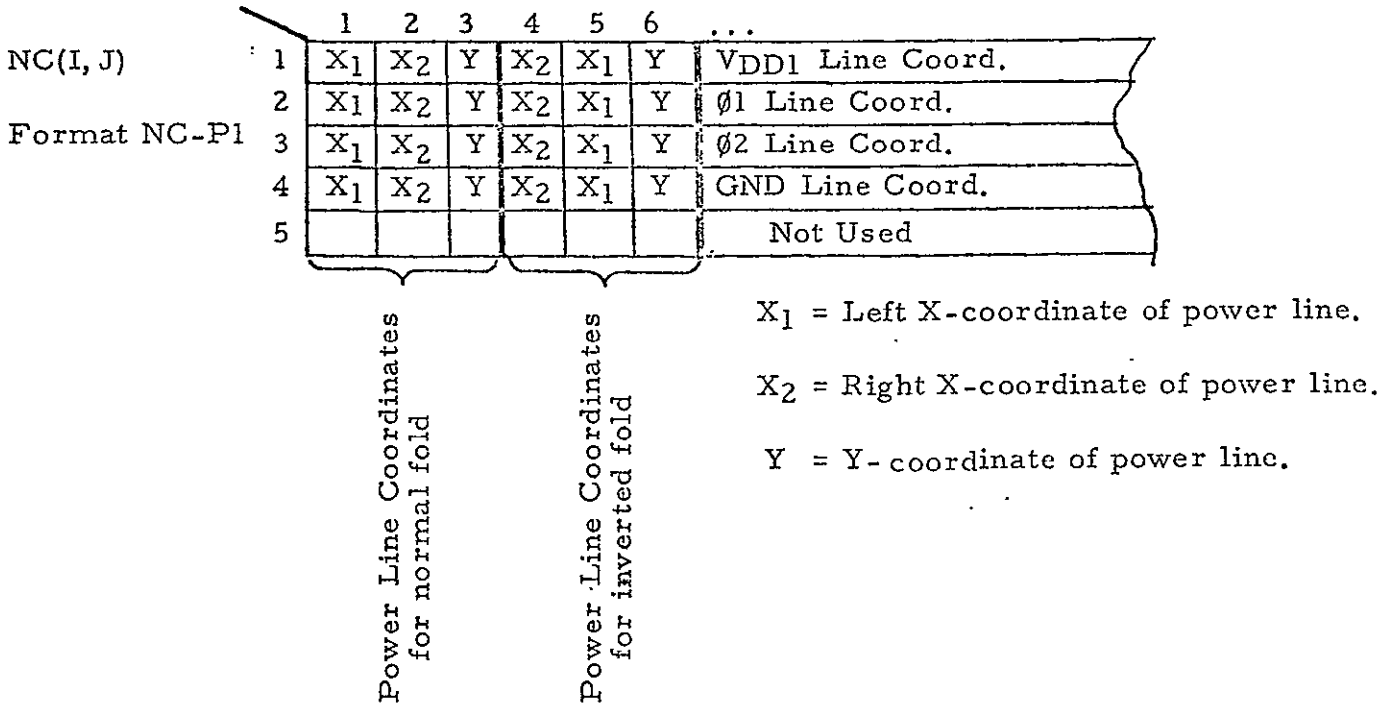


Table 3-18 POWR ARRAYS

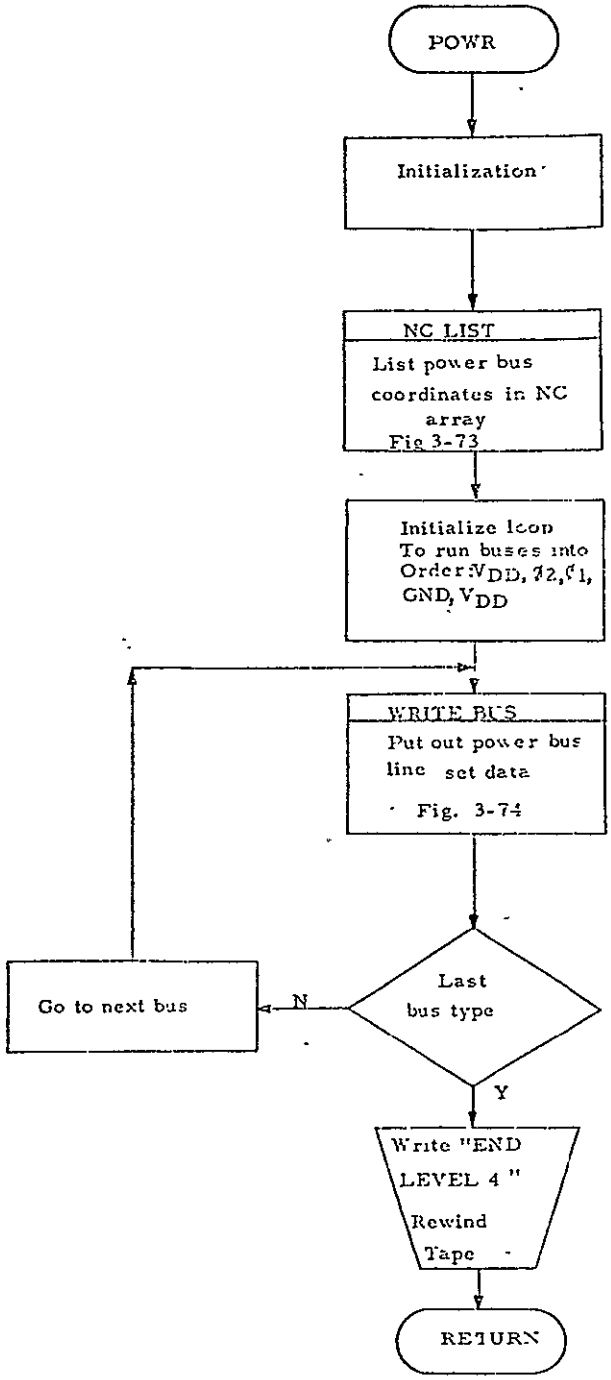


Figure 3-72 POWR

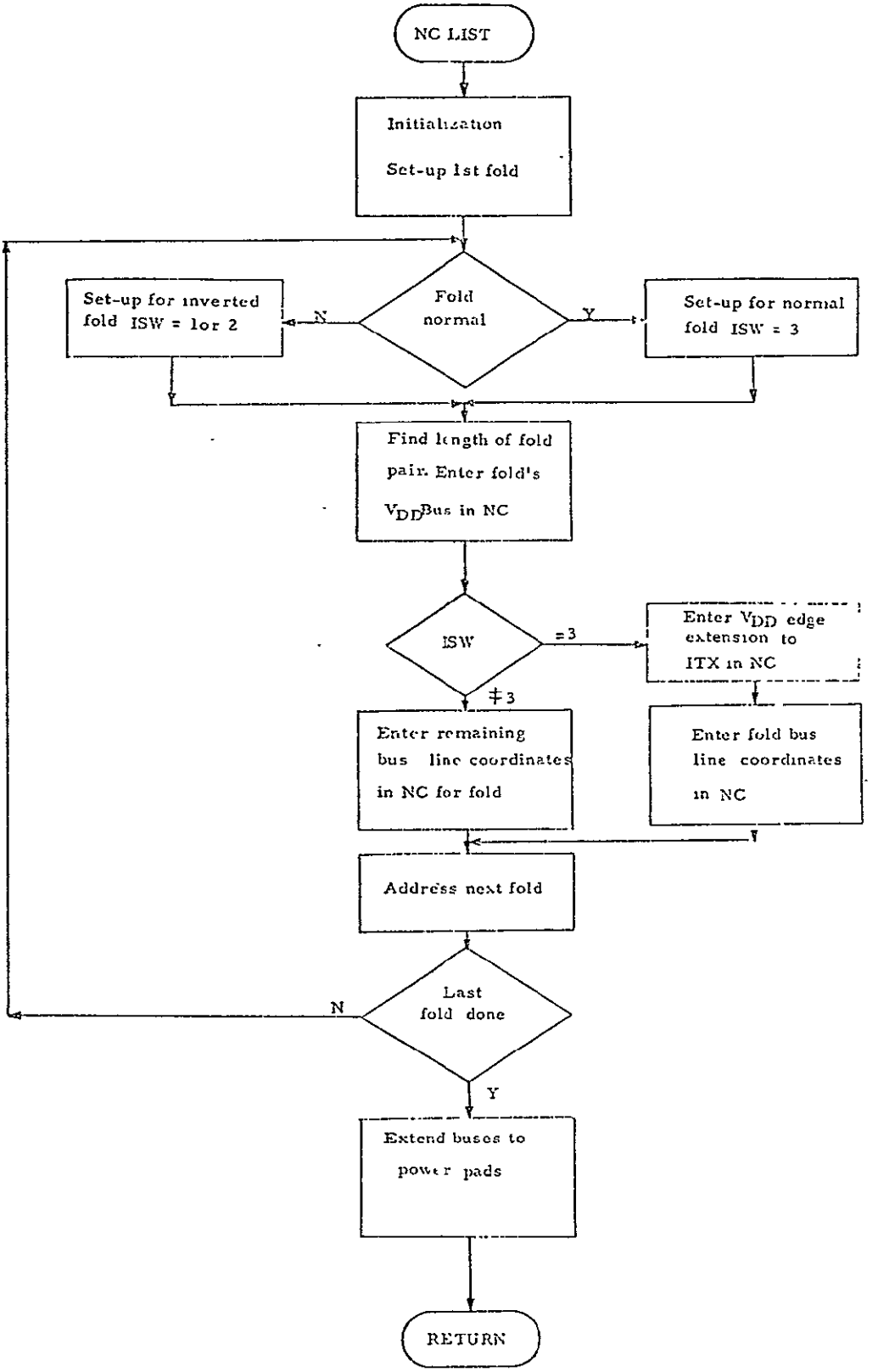


Figure 3-73 NC LIST

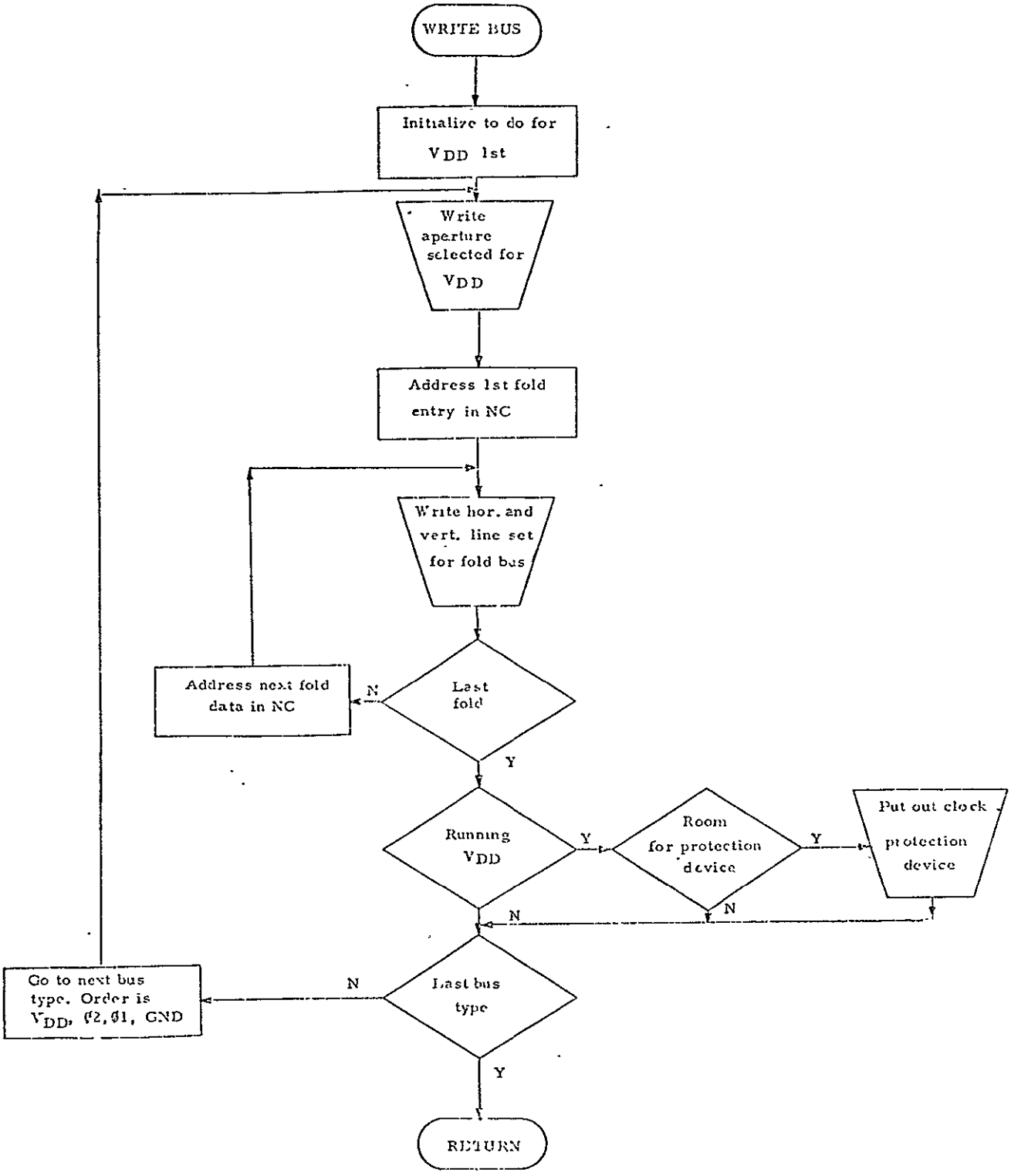


Figure 3-74 WRITE BUS

SECTION IV

PROGRAM SIZING

4.1 Program Overlays

Since the PRF program is relatively large, a series of program overlays has been devised to reduce the core storage which must be dedicated to program instructions. The sequential nature of the program allows each of the bracketed segments of Figure 4-1 to overlay the preceding segment when its execution has been completed. The only instructions which must remain resident in core throughout the program are those of the executive MAIN and those of the small utility routine LOOK. The core required for program instruction is thus the sum of that required by MAIN and LOOK and that required by the largest overlay block. The largest block of the current PRF configuration is SQUEEZ-PADMVE-MOVE, requiring roughly 5.6K. The total core which must be reserved for instructions is thus a bit less than 6K, while the total without overlays is more than 20K.

4.2 Array Dimensions

The bulk of the core memory required by PRF is used for array storage. These arrays are used as chip maps, component lists, cross-referencing data tables, and input data tables, and, generally, their size requirements are directly dependent upon the complexity of the chip designs being run. Optimal use of core can be obtained by carefully scaling each individual array to the complexity of the circuit being run. Although the program makes a few checks for array overflows, generally there will not be an error message when an array dimension has been exceeded, and such an error must be traced back from seemingly unrelated problems using core memory dumps and debug printout.

Table 4-1 lists the dimensions of the major chip dependent arrays in both the 200 cell program used for system test and the 129 cell version implemented by M&S Computing. Complete descriptions of the formats of these arrays may be found in Section III in the descriptions of the subroutines which form them. These subroutines can be referenced from Figure 2-4 for common arrays or from Table 4-2 for local arrays. The latter figure also indicates those local arrays which are equivalent or sectionally equivalent to common arrays, and where these equivalent statements must be adjusted if the dimensions of either array are altered.

The following lists attempt to define the relationship between chip complexity and minimum array dimensions.

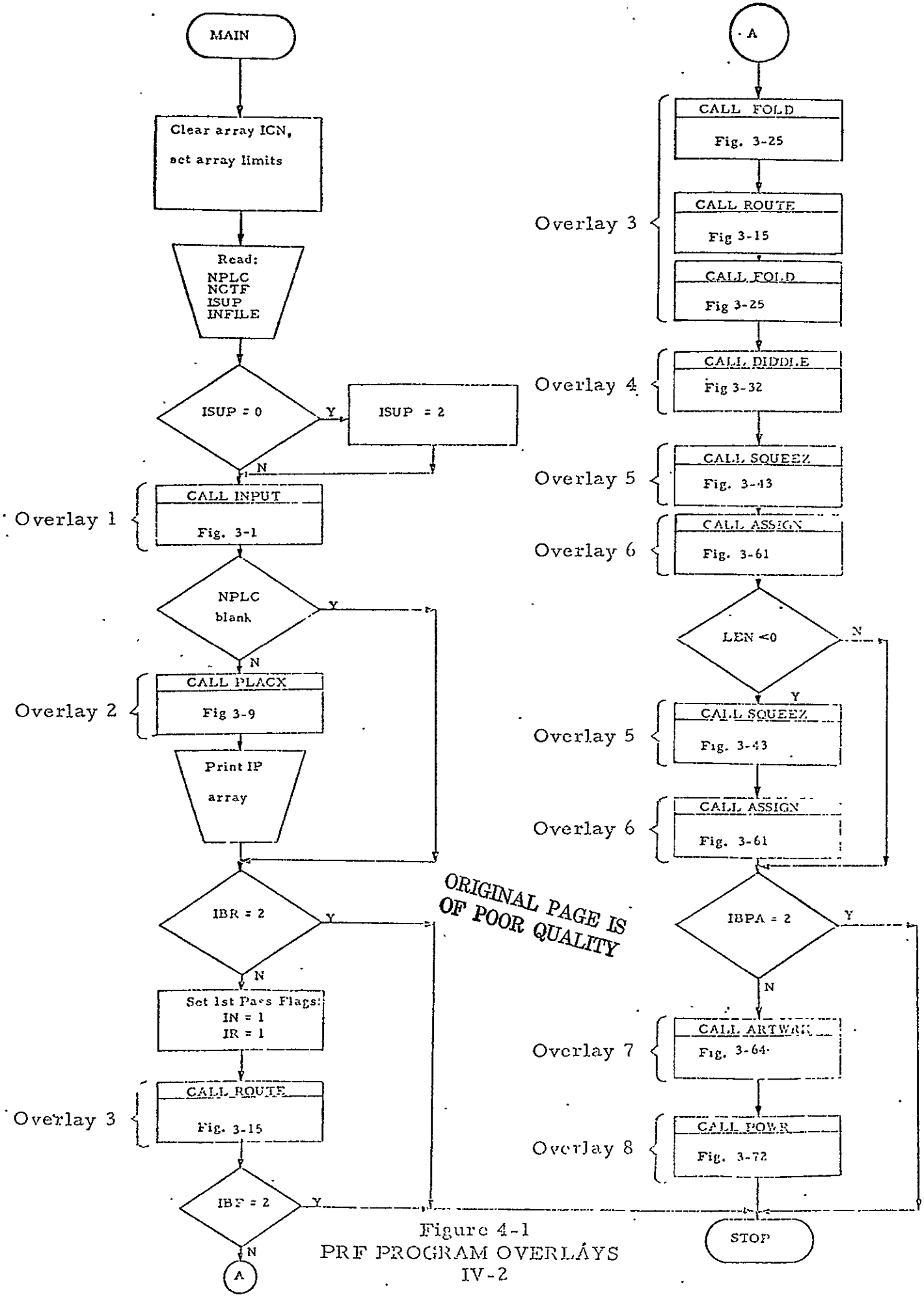


Figure 4-1
PRF PROGRAM OVERLAYS

Array	Subroutine Forming Array	Equivalent to Array	Subroutines Which Reference Array
ICA	INPUT	IRA	INPUT
ICB	INPUT	IRA	INPUT
ICL	INPUT	IRA	INPUT, ROUTE
INF	INPUT	IRA	INPUT, PLACX, CRCALC, FOLD
IP	INPUT	IRA	INPUT, PLACX, ROUTE
IPTMP	PLACX	---	PLACX
IPTRA	INPUT	IRA	INPUT
LNAD	INPUT	IRA	INPUT, PLACX, CRCALC
MAP	INPUT	IRA	INPUT, PLACX, FOLD
MAPTMP	PLACX	---	PLACX
NAT	PLACX	---	PLACX
NETTOT	PLACX	---	PLACX

Table 4-1 MAJOR LOCAL ARRAYS

		200 Cells		220	
		Dimensions	Core	Dimensions	Core
COMMON ARRAYS	CAP	(200)	200	(220)	220
	IC	(200,10)	2,000	(220,11)	2,420
	ICK	(120,10,13)	15,600	(50, 11, 4)	2,200
	ICN	(200)	200	(200)	200
	IEC	(200)	200	(200)	200
	IF	(10,4)	40	(20, 4)	80
	IFA	(200,200)	40,000	(218, 220)	23,980**
	INT	(6)	6	(6)	6
	IPR	(250)	250	(250)	250
	IPTRN	(120)	120	(50)	50
	IRA	(800,50)	40,000	(800, 51)	20,400**
	IXW	(35)	35	(45)	45
	IYW	(200)	200	(235)	235
	JD	(2)	2	(2)	2
	NC	(600,5)	3,000	(850, 5)	4,250
	NET	(2000)	2,000	(2250)	2,250
	NX	(200,3)	600	(400, 3)	1,200
	PR	(250)	250	(250)	250
LOCAL ARRAYS	ICA	(200,6)	1,200	(50, 6)	300
	ICB	(200,10,13)	26,000	(50, 11, 13)	7,150
	ICL	(120,6)	720	(50, 6)	300
	INF	(200,4)	800	(220, 4)	880
	IP	(200,2)	400	(220, 2)	440
	IPTMP	(200,7)	1,400	(220, 7)	1,540
	IPTRA	(200,2)	400	(150)	150
	LNAD	(1000)	1,000	(1300)	1,300
	MAP	(200,2)	400	(220, 2)	440
	MAPTMP	(200,7)	1,400	(220, 7)	1,540
	NAT	(2000)	2,000	(1540)	1,540
	NETTOT	(200)	200	(220)	220

*Although it may be possible to run up to 220 cells with these dimensions, chips of this size have not yet been attemped.

**Half word arrays

EFFECTIVE ARRAY DIMENSIONS VS. NUMBER OF CELLS

Table 4-2

Fixed Dimension Arrays:

ICN(200)
IF(10, 4)
INT(6)
IPR(250)
IXW(35)
JD(2)

Arrays directly dependent on the number of cells (including bonding pads) of the logic design. Where C = (number of logic cells + bonding pads), these arrays are dimensioned as:

IC(C, 10)
IEC(C)
INF(C, 4)

Arrays directly dependent on the number of logic cells of the design. Where C is the number of logic cells, these arrays are dimensional as:

IP(C, 2)
MAP(C, 2)
IPTMP(C, 7)
MAPTMP(C, 7)
NAT(7xC)

Arrays directly dependent on the number of different cells in the Circuit Type File. They are dimensioned as follows, where F is the number of file cells:

IPTRA(F)
ICA(F, 6)
ICB(F, 10, 13)

Arrays directly dependent on D, the number of different circuit types utilized by the logic design being run:

ICK(D, 10, 13)
ICL(D, 6)
IPTRN(D)

Arrays directly dependent on N, the number of independent interconnection nets in the logic design being run:

CAP(N)
NETTOT(N)

The following arrays have the unique dimension requirements described:

IFA(X, Y) - The minimum X-dimension is roughly twice the maximum number of pins in any fold plus 45. The minimum Y-dimension is the total number of horizontal wiring channels plus 4x(the number of folds).

IRA(X, Y) - The required X-dimension equals the total number of cell pins, including cell edge reference pins, plus one. The Y-dimension must be 7 plus the maximum number of horizontal wire channels required at any one point in the linear wiring array, not to be less than 20.

IYW(Y) - Dimensions should be the same as the Y-dimensions of IFA.

LNAD(I) - Dimension must equal the number of pins connected in the design being run.

NC(I, 5) - Dimension I must equal the number of connected pins minus the number of nets.

NET(1) - Dimension I $\geq \lceil$4x(number of nets) + 2x(number of connected pins)$\rceil$.

NX(1, 3) - Dimension I must equal whichever is greater, the number of nets, the number of cross-fold wires, or the number of tunnel ends in the PRF output.

SECTION V

DEFINITIONS AND REFERENCES

5.1 Map Array Referencing Conventions

The following convention corresponds to the orientation printed in the chip picture by the PRF program:

Linear Chip Array Map IRA(X, Y)

Up	-	Increasing Y
Right	-	Increasing X

Square Chip Array Map IFA(X, Y)

Up	-	Decreasing Y
Right	-	Increasing X

Although this convention is used consistently throughout this manual, debug dumps and many of the listing comments use the following convention for both IRA(X, Y) and IFA(X, Y):

Up	-	Decreasing X
Right	-	Increasing Y

All output data coordinates follow yet another convention where the origin is located at the lower left of a plot, resulting in these axis directions:

Up	-	Increasing Y
Right	-	Increasing X

Some of the listing comments in ASSIGN, ARTWRK, and POWR reference IFA using this last convention.

5.2 Abbreviations

C. T. F.	-	Circuit Type File
Cap.	-	Capacitance
Coord.	-	Coordinate
Hor.	-	Horizontal

I.	-	Inverted
L.	-	Left
Msg.	-	Message
N.	-	Normal
P. R. F.	-	Placement-Routing-Folding Program
Ref.	-	Reference
Rt.	-	Right
Seg.	-	Segment
T. E.	-	Tunnel End
Trans.	-	Transistor
Tun.	-	Tunnel
Vert.	-	Vertical

5.3 Term Definitions

CAP WIRES - Same as LEFT END-AROUND WIRES.

CELL/PIN ID - A number which identifies both the cell number and the pin number of a pin position in IRA or IFA. Set to [100 x (cell number) + (pin number)].

CROSS-FOLD WIRES - Wire interconnections between a pair of adjacent folds whose cells are connected to different power bus branches.

CROSSWIRE - Same as CROSS-FOLD WIRE.

DUMMY FOLD - A fold containing no cells which is inserted above the bottom pad fold to make the pad fold inverted.

EDGE PIN - Either the first or last pin of a cell which is used to provide a reference to the location of the cell edge, and may not be used as a connection point.

FOLD - A row of Banning standard cells to be placed side by side on the chip, and the wire interconnections associated with these cells.

HORIZONTAL WIRE - An interconnection segment which runs parallel to the fold axes of the chip. Such a segment will be represented in chip map IRA and IFA by elements having the same Y-coordinate and containing either a number N, $\angle 1 \angle$ N 998 (metal of net N), or 998 (tunnel end).

INVERTED CELL - A cell which was oriented in the linear chip map IRA so that its first pin was referenced by a larger X-coordinate than its last pin.

INVERTED FOLD - A fold whose cells are oriented with their connection pins above.

LEFT END-AROUND WIRES - Wire interconnections between a pair of adjacent folds whose cells share a common power bus branch.

LOGIC FOLD - A fold containing at least one cell, but no bonding pad cells.

NORMAL CELL - A cell which was oriented in the linear chip map IRA so that its first pin was referenced by a smaller X-coordinate than its last pin. Cells are drawn in normal orientation in the Banning Engineering Notebook.

NORMAL FOLD - A fold whose cells are oriented with their connection pins below, as they are drawn in the Banning Engineering Notebook.

PAD FOLD - A fold which contains one or more bonding pad cells.

REASSIGNED PIN - A pin whose specified connection has been interchanged with an equivalent pin of the same cell to reduce total connection length.

REFERENCE PIN - Same as EDGE PIN.

RUNNING LENGTH - A value which specifies the distance in tenths of a mil between a particular cell pin or wiring column and some reference point. This reference point may be either the left end of the linear chip map, the right or left end of a fold, or the origin of the chip coordinates.

SPACER PIN - A pin number of 10 added to a cell by the PRF program where the cell has space for a wiring channel but does not have a connection pin.

TAP An intermediate connection between the end points of a horizontal wire segment.

TUNNEL END - A point at which pattern set data is to be used to create a connection between metal and P-material.

VERTICAL WIRE An interconnection segment which runs perpendicular to the fold axes of the chip. The representation of such a segment in chip maps IRA and IFA will consist of elements having the same X-coordinate and containing either a 1 (vertical metal), 999 (tunnel) or 998 (tunnel end).

5.4 References

Banning Placement-Routing-Folding User's Manual (M&S Computing Report No. 70-0021).

Banning ARTWORK Program User's Manual (M&S Computing Report No. 70-0012).

☆ U.S. GOVERNMENT PRINTING OFFICE 1976—641-255/555 REGION NO. 4