NASA CONTRACTOR
REPORT

NASA CR-144247

(NASA-CR-144247) PROGRAMMER'S MANUAL FOR THE BANNING ARTWORK PROGRAM (M&S Computing, Inc.) 49 p HC $4.00 CSCL 09B

N76-22947

Unclas
G3/61 26775

PROGRAMMER'S MANUAL FOR THE BANNING ARTWORK PROGRAM

By J. F. Taylor

M&S Computing, Inc.
Huntsville, Alabama 35805

June 5, 1970

Prepared for

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

TECHNICAL REPORT STANDARD TITLE PAGE

1. REPORT NO. CR-144247	2. GOVERNMENT ACCESSION NO.	3. RECIPIENT'S CATALOG NO.

4. TITLE AND SUBTITLE
PROGRAMMER'S MANUAL FOR THE BANNING ARTWORK PROGRAM

5. REPORT DATE
February 19, 1976

6. PERFORMING ORGANIZATION CODE

7. AUTHOR(S)
J. F. Taylor

8. PERFORMING ORGANIZATION REPORT #
70-0013

9. PERFORMING ORGANIZATION NAME AND ADDRESS
M&S Computing, Inc.
P. O. Box 5183
Huntsville, Alabama 35805

10. WORK UNIT NO.

11. CONTRACT OR GRANT NO.
NAS8-25621

13. TYPE OF REPORT & PERIOD COVERED
Contractor Report
June 5, 1970

12. SPONSORING AGENCY NAME AND ADDRESS
National Aeronautics and Space Administration
George C. Marshall Space Flight Center

14. SPONSORING AGENCY CODE
EC45

15. SUPPLEMENTARY NOTES
Electronics Development Division, Electronics and Control Laboratory
Design Techniques Branch

16. ABSTRACT

This document is intended to serve as the programmer's Manual for the Banning Artwork Program. A detailed description of the internal logic and flow of the Artwork Program has been included in this report. It is recommended that every user review this manual in order to receive an appreciation of the functions performed by the Artwork Program; however, the User's Manual for the Artwork Program contains sufficient information to allow the designer/engineer to efficiently use the Artwork Program. Before beginning a detailed study of this Programmer's Manual, the reader should have a complete understanding of the contents of the User's Manual for the Banning Artwork Program. It is assumed that the programmer using this manual has a working knowledge of FORTRAN IV.

17. KEY WORDS

18. DISTRIBUTION STATEMENT
Unclassified-Unlimited

COR: John M. Gould

EC01 J. Brooks 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	49	NTIS

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

REFERENCE

This document is primarily a rewrite of information contained in the original documentation for the Artwork Program produced by the Radio Corporation of America during the contract for the initial definition and implementation of the Banning System of Programs (Contract Number DA-18-119-AMC-03460(X)).

This "Programmer's Manual" and the "User's Reference Manual" serve as the final documentation of the Banning Artwork Program as implemented by M&S Computing, Inc. under contract number NAS8-25621 for Marshall Space Flight Center's Astrionics Laboratory, Technology Division.

TABLE OF CONTENTS

TABLE OF CONTENTS - Continued

SECTION I

INTRODUCTION

The Banning Artwork Program generates the Gerber commands that select the appropriate apertures and determine the plotter motions required to completely expose of "fill-in" the specified areas of a circuit mask. The program generates the commands required to produce the artwork of one mask level at a time up to a maximum of nine (9) levels. After each level of artwork has been created, an end-of-file is written on the output magnetic tape and the program recycles to generate the commands needed to create the next level mask.

The structure of the Banning Artwork Program consists of an executive program maintaining control at all times, with subordinate subroutines being called upon to execute the specific requirements for updating the pattern library and creating circuit masks. Figure 1.1 illustrates the functional flow of the Artwork Program through the various subroutines. All program control cards are read and interpreted by the executive routine.

After the title card has been read, the available apertures have been defined to the program, and the fill techniques have been specified, control is passed to the utility subroutine to perform any utility operations required by the current job. The utility subroutine will normally return control immediately to the program executive, but utility requirements may dictate that the pattern library be updated or other utility functions be performed prior to returning control to the main program.

When control is returned to the executive routine, additional control cards are read to determine the number of levels of artwork to draw, and to define any pattern sets that will appear later in the component input data. After the pattern sets have been defined, the component input data is read and formatted into data arrays that can be easily accessed during the mask creation phase of the program.

The remainder of the artwork program consists of the major artwork processing loop that is repeated for each required mask level. The loop performs the following processing for each level:

- Extract the polygons from the pattern library that define the artwork required to construct the masks of each component;

ARTWORK PROGRAM SUBROUTINE FLOW

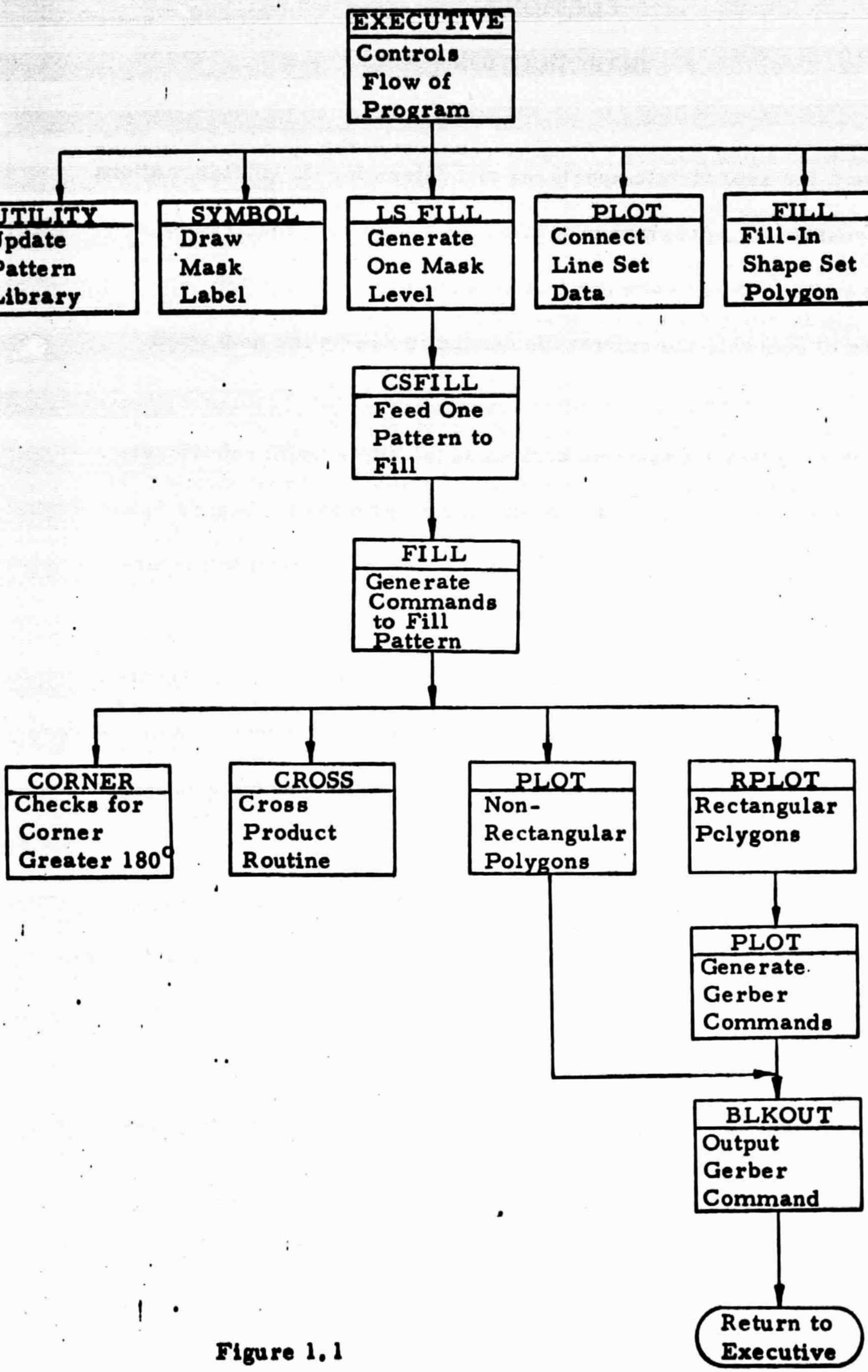


Figure 1.1

- Place the component pattern at the specified location and orientation on the chip;

- Create the Gerber commands to completely expose or "fill-in" every polygon;

- Draw the interconnect lines if line set data is specified;

- Draw the scribe lines around the chip as specified by the shape set data;

- Label the created mask as specified by the symbol data.

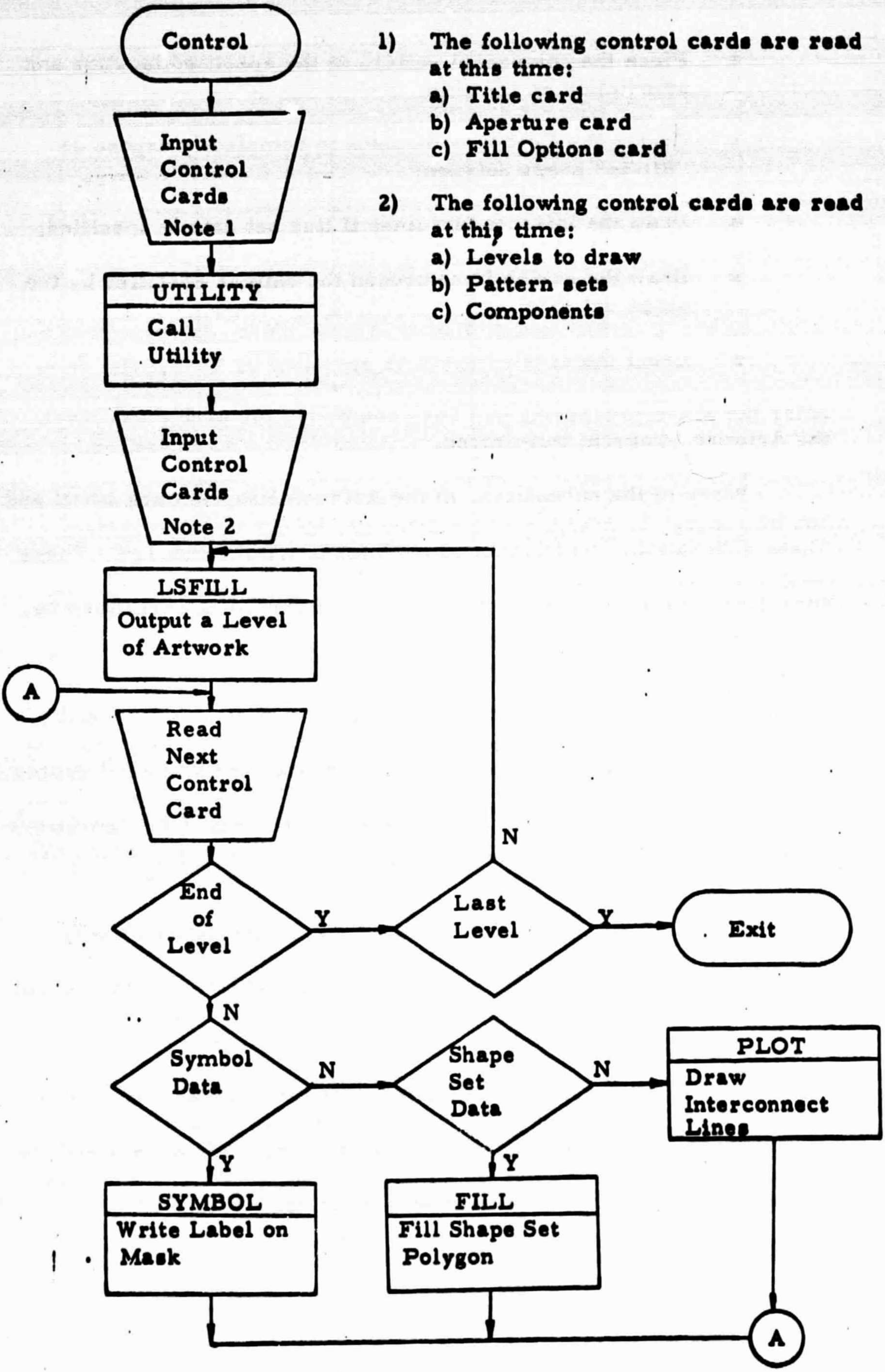
After the above processing has been completed for each mask level, the Artwork Program terminates.

Many of the subroutines in the Artwork Program are small and can be adequately described by a one-page functional flowchart. These subroutines are illustrated in Figures 1.2 through 1.9. These flowcharts and the comments incorporated into the program listing should provide a sufficient explanation of the functions performed by the subroutines.

Special sections have been included in this report to describe the more complicated algorithms used by the FILL subroutine and the UPDATE subroutine. The major portion of the logic required to fill-in the polygons of a given mask level is concentrated in the fill subroutine and the special plot options subroutines. Sections IV and V contain both detailed discussions and detailed flowcharts of the operations required to generate the Gerber commands necessary to produce the final artwork output.

Section II describes the format of the binary pattern library used by the Artwork Program. Any modifications to the format of this library will effect several subroutines, and therefore, care should be taken to insure that proper control is maintained over the format of the library.

Section III describes the format of the various arrays used to store, format, and manipulate the components and pattern sets used to create the artwork of each mask level of a chip. To understand the operations performed by the Artwork Program, it is imperative that the programmer become familiar with these arrays.



1) The following control cards are read at this time:
   a) Title card
   b) Aperture card
   c) Fill Options card

2) The following control cards are read at thiş time:
   a) Levels to draw
   b) Pattern sets
   c) Components

Figure 1.2

LSFILL SUBROUTINE

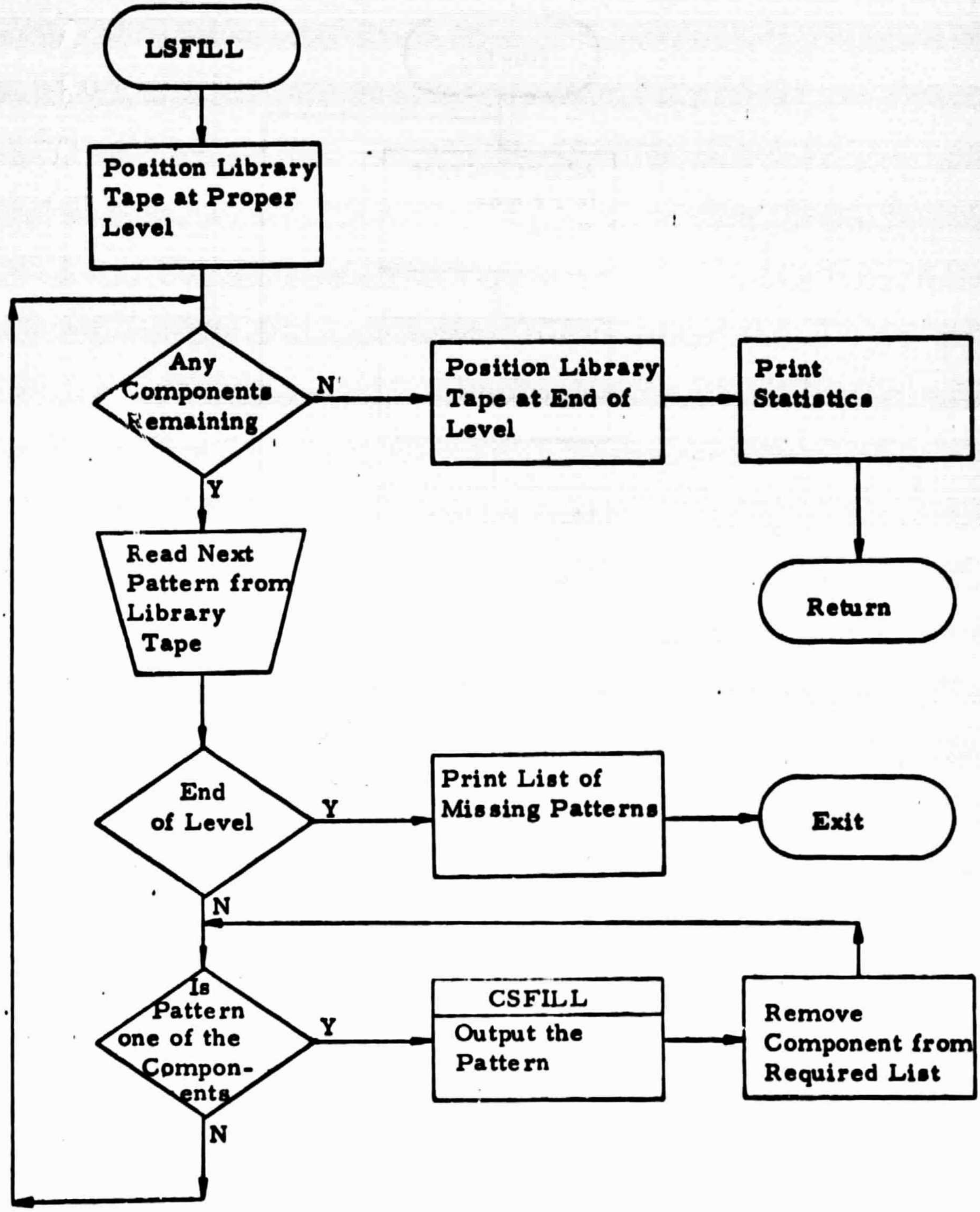

LSFILL
Position Library Tape at Proper Level
Any Components Remaining
N
Position Library Tape at End of Level
Print Statistics
Return
Y
Read Next Pattern from Library Tape
End of Level
Y
Print List of Missing Patterns
Exit
N
Is Pattern one of the Compon-ents
Y
CSFILL
Output the Pattern
Remove Component from Required List
N

Figure 1.3

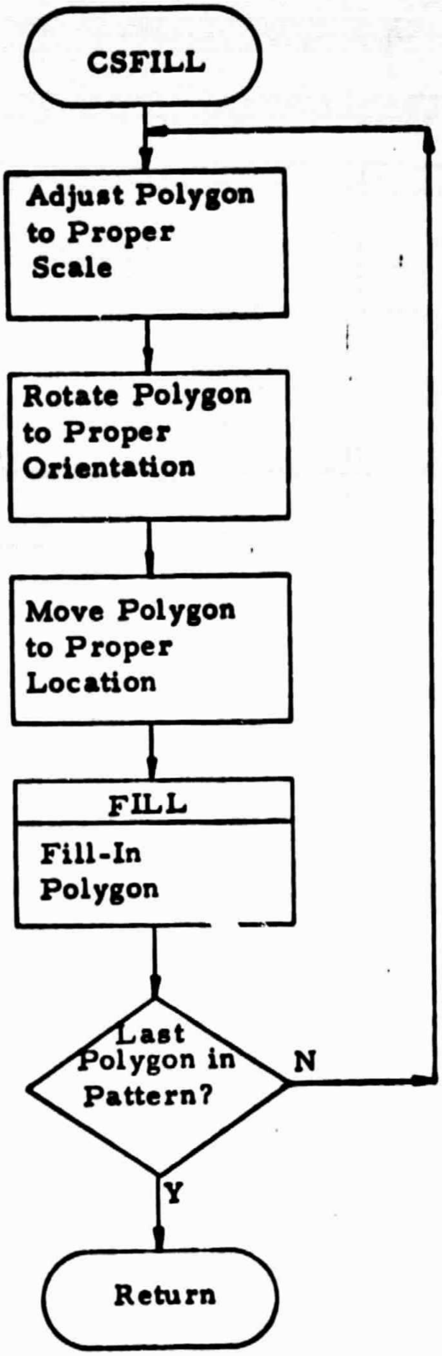

CSFILL
Adjust Polygon to Proper Scale
Rotate Polygon to Proper Orientation
Move Polygon to Proper Location
FILL
Fill-In Polygon
Last Polygon in Pattern?
N
Y
Return

Figure 1.4

PLOT SUBROUTINE

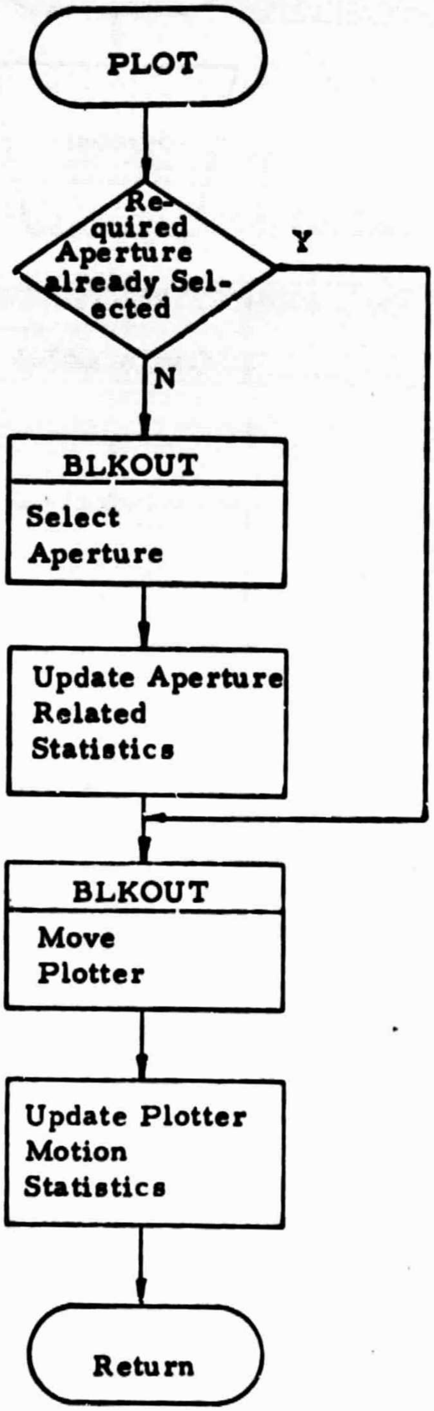

PLOT
Required Aperture already Selected
Y
N
BLKOUT
Select Aperture
Update Aperture Related Statistics
BLKOUT
Move Plotter
Update Plotter Motion Statistics
Return

Figure 1.5

SYMBOL SUBROUTINE

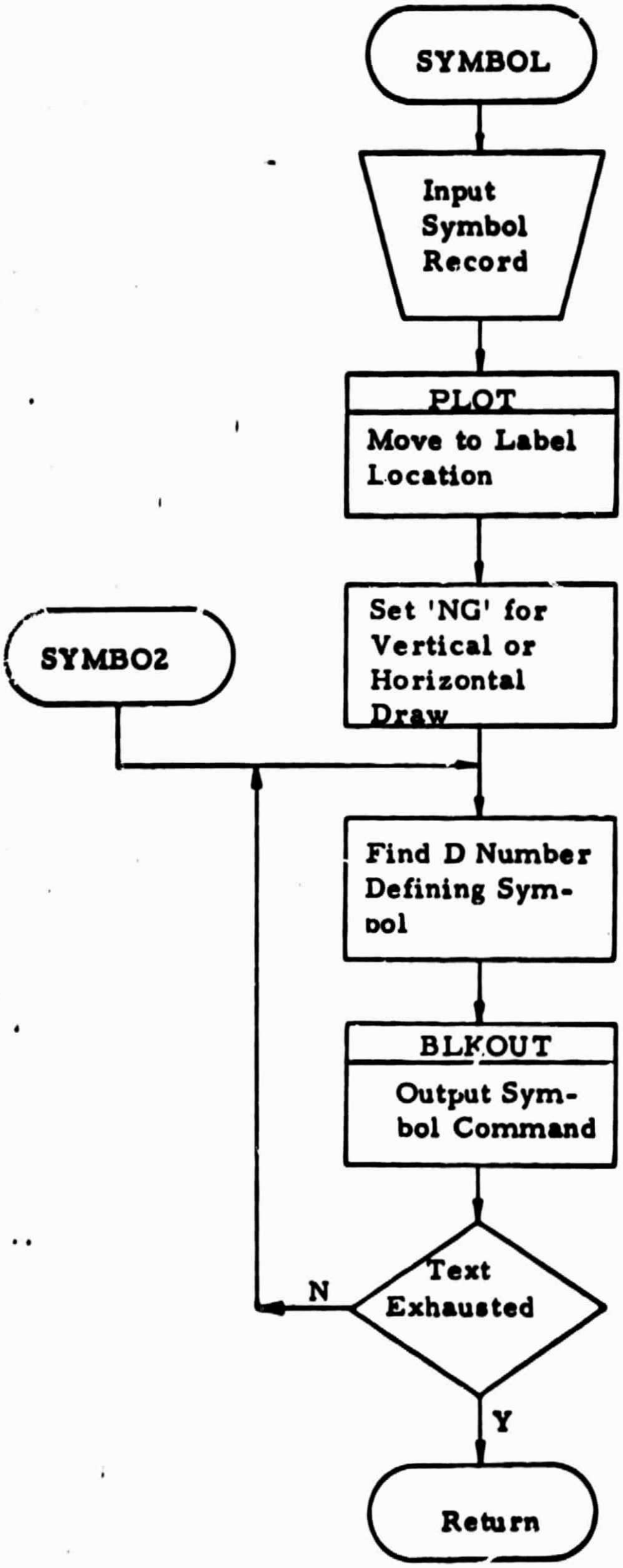


Figure 1.6

MARKER SUBROUTINE

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

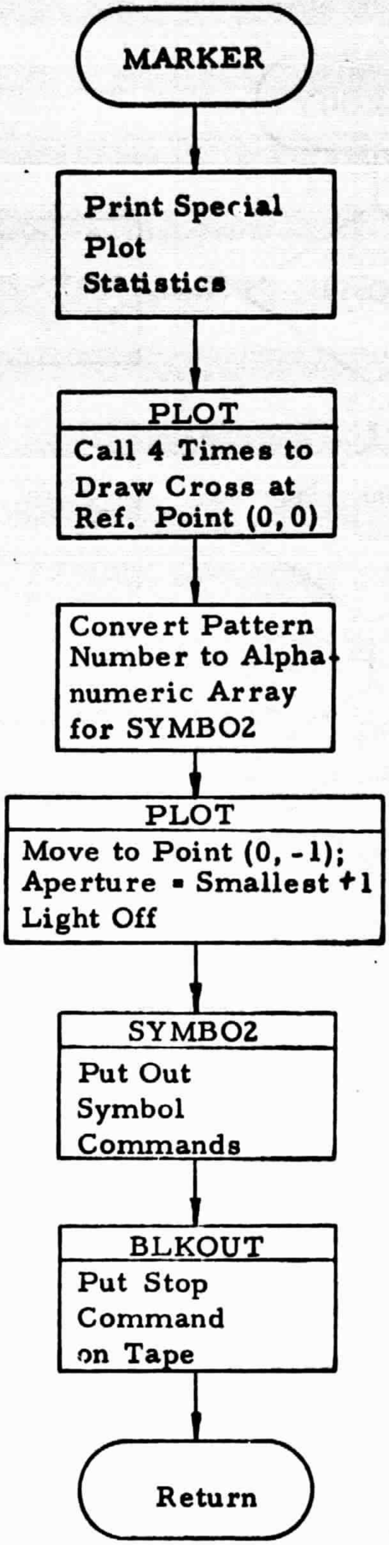
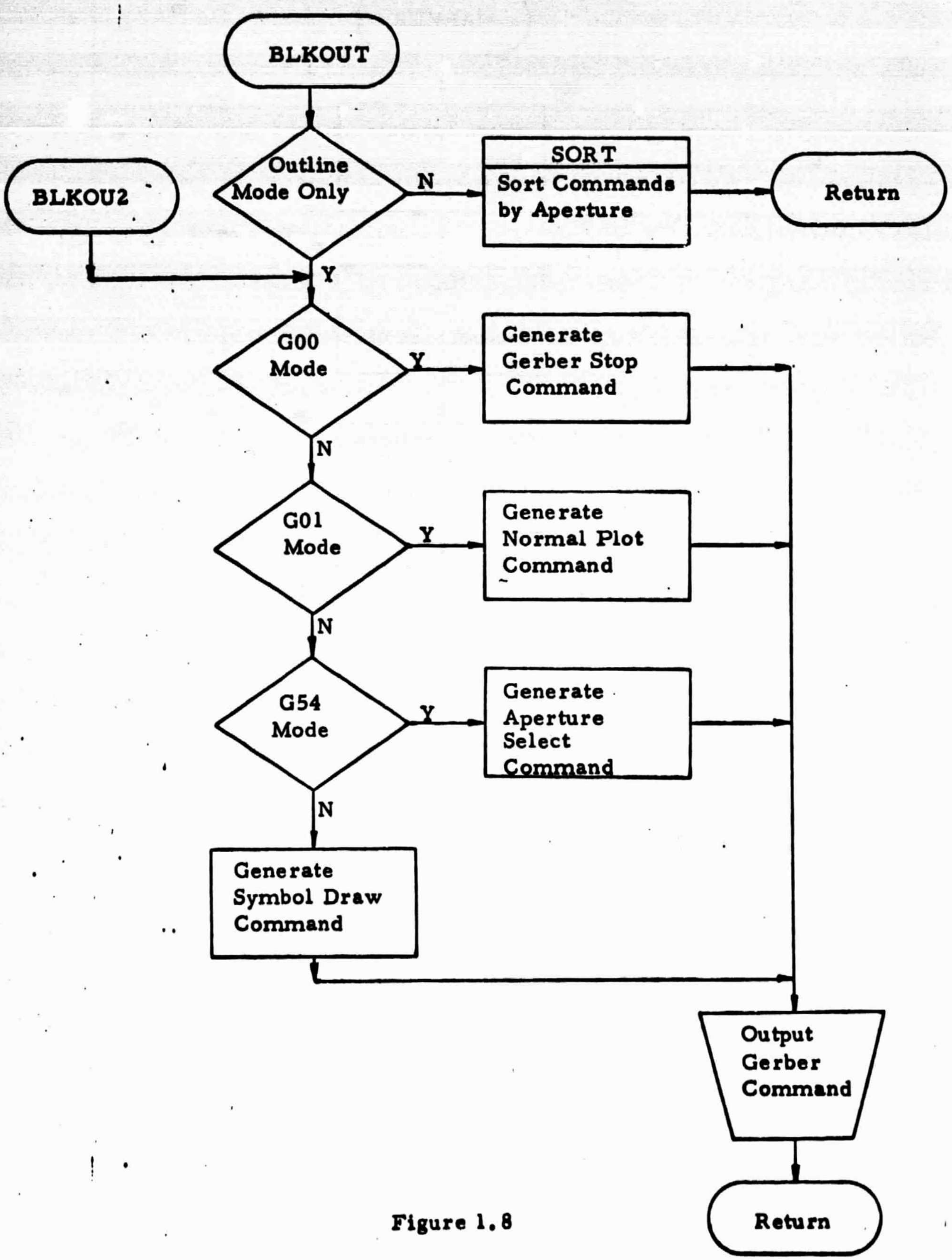

MARKER
Print Special Plot Statistics
PLOT
Call 4 Times to Draw Cross at Ref. Point (0, 0)
Convert Pattern Number to Alpha-numeric Array for SYMBO2
PLOT
Move to Point (0, -1); Aperture = Smallest +1 Light Off
SYMBO2
Put Out Symbol Commands
BLKOUT
Put Stop Command on Tape
Return

Figure 1.7

BLKOUT SUBROUTINE



BLKOUT
BLKOU2
Outline Mode Only
N
SORT
Sort Commands by Aperture
Return
Y
G00 Mode
Y
Generate Gerber Stop Command
N
G01 Mode
Y
Generate Normal Plot Command
N
G54 Mode
Y
Generate Aperture Select Command
N
Generate Symbol Draw Command
Output Gerber Command
Return

Figure 1.8

APERTURE SORT SUBROUTINE

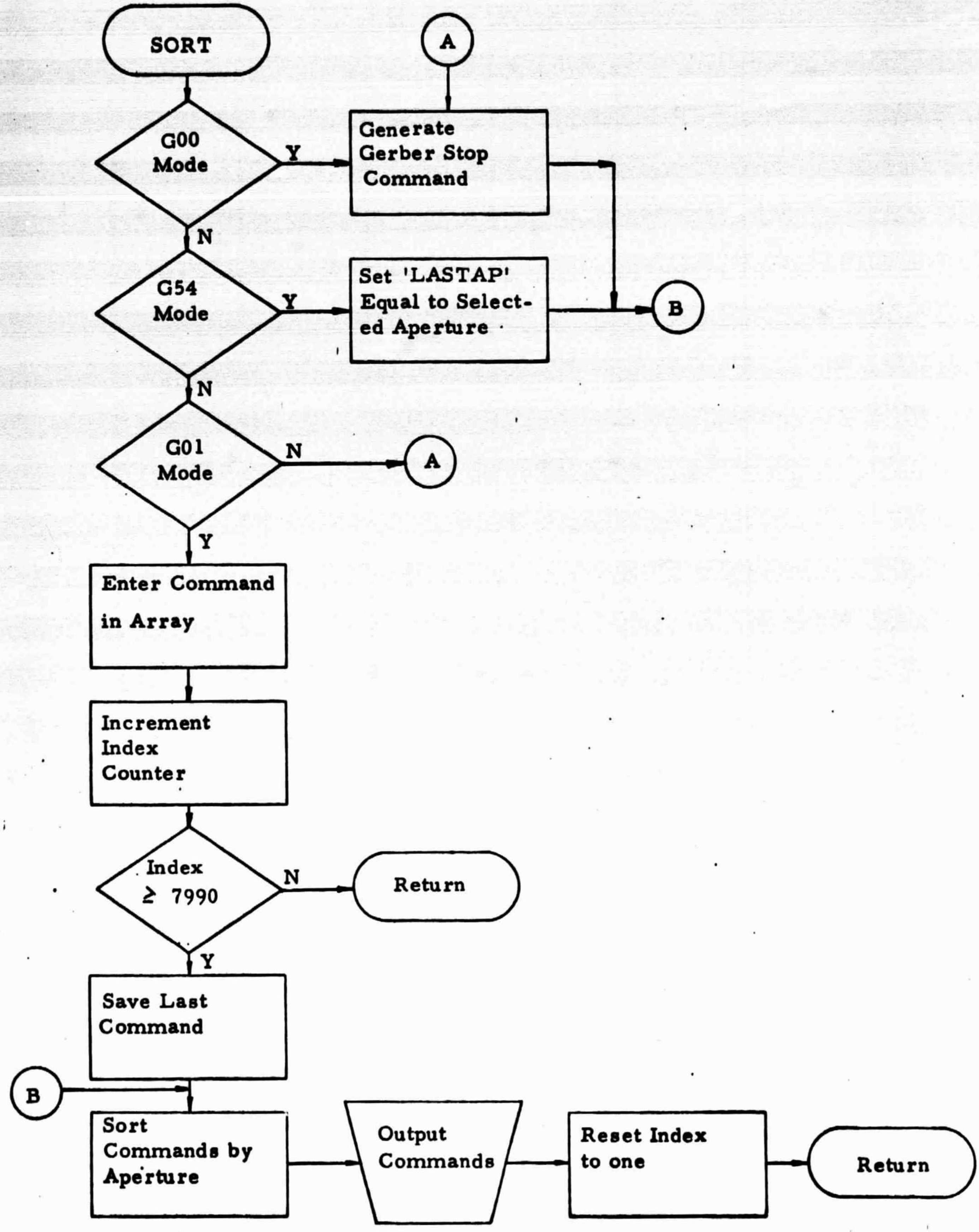


Figure 1.9

SECTION II

BINARY PATTERN LIBRARY

Within the Banning System, the term pattern is used to identify the polygon or set of polygons defining the artwork required to construct a given mask. Therefore, the Banning Pattern Library may be defined as the collection of polygons defining the artwork required to construct each mask of every standard Banning cell. In order to understand the method in which the Artwork Program processes the pattern definitions, it is important to understand the arrangement of patterns within the Pattern Library.

Since the Artwork Program produces the output artwork commands for one mask level at a time, the pattern library has been formatted into levels where each level contains all of the patterns defining the particular mask level of each standard Banning cell. The Artwork Program is capable of supporting up to nine levels of circuit masks, and therefore the pattern library can theoretically be divided into nine major sections. However, there are presently only six levels defined within the Banning system, resulting in the following divisions in the pattern library:

LEVEL 1 - P-REGION
LEVEL 2 - CONTACT & GATE
LEVEL 3 - CONTACT
LEVEL 4 - METAL
LEVEL 5 - SAME AS LEVEL 2 (DEEP DIFFUSION)
LEVEL 6 - PROTECTIVE COVERING

This multi-level division of the pattern library assumes that each standard Banning Cell has a particular design for each mask level. However, there are Banning cells that appear on only one level mask or appear on every level, but use the same shape on each level. In order to facilitate the definition of cells of this type, the following numbering scheme was developed:

- **<u>0000-9999</u>: All Banning cells must be assigned a number within this range.**

- **<u>0000-8999</u>: Banning cells with numbers within this range will be considered Family Patterns. That is each cell** will be assigned a number with the units digit equal to zero and be broken down into mask levels with the value of the units digit specifying the level. For example, the Banning 2 Input Nor cell is defined as a family pattern and referred to as cell number 2070. The cell contains a unique pattern for each level mask with each level indicated by the unit digit of the number: 2071 for level one, 2072 for the level 2, etc.

- 9000-9599: Banning cells with numbers within this range are considered One-Level Patterns. Each cell will consist of only one pattern definition which is included in the pattern library at the level indicated by the units digit of the number assigned to the cell. For example, cell number 9034 refers to a Banning Pad that consists of one pattern on the fourth level of the pattern library.

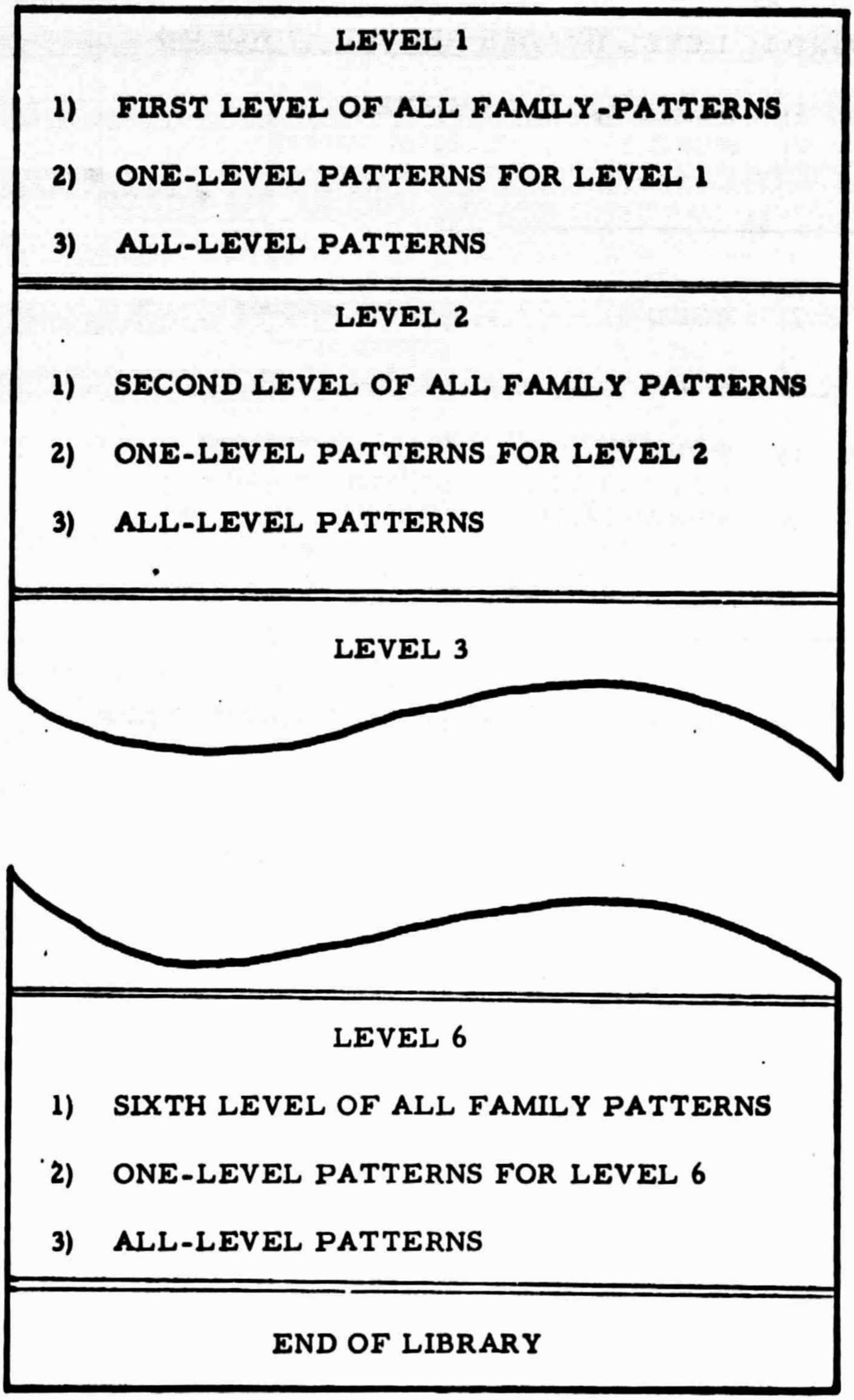
- 9599-9999: Banning cells with numbers within this range are considered All-Level-Patterns. Each cell will consist of only one pattern definition, but it will be included on each level of the pattern library. The units digit has no significance for All-Level-Patterns. An off-chip alignment mark is a good example of an all-level-pattern.

Figure 2.1 illustrates the format of the pattern library. With this arrangement of patterns, the Artwork Program can position the library tape at the level being processed and have immediate access to all predefined patterns. The family-pattern concept is the most important and includes the majority of the patterns encountered. Family-patterns are specified to the Artwork Program by number, for example 2070, to be drawn at a desired location with a specified orientation. The Artwork Program then places pattern 2071 at this location for level 1 of the artwork, pattern 2072 for level 2 of the artwork, etc., continuing to the last level.

Table 2.1 illustrates the format of a level of the binary pattern library,. This format is repeated for every level of the library. A two-word record of the following format terminates the library:

ENDLIB

BANNING PATTERN LIBRARY



LEVEL 1

1) FIRST LEVEL OF ALL FAMILY-PATTERNS

2) ONE-LEVEL PATTERNS FOR LEVEL 1

3) ALL-LEVEL PATTERNS

LEVEL 2

1) SECOND LEVEL OF ALL FAMILY PATTERNS

2) ONE-LEVEL PATTERNS FOR LEVEL 2

3) ALL-LEVEL PATTERNS

LEVEL 3

LEVEL 6

1) SIXTH LEVEL OF ALL FAMILY PATTERNS

2) ONE-LEVEL PATTERNS FOR LEVEL 6

3) ALL-LEVEL PATTERNS

END OF LIBRARY

Figure 2.1

FORMAT OF THE BINARY PATTERN LIBRARY

RECORD 1: LEVEL HEADER RECORD (3 WORDS)

1)	WORDS 1, 2	: 'LEVEL'
2)	WORD 3	: level number

RECORD 2: PATTERN HEADER RECORD (16 WORDS)

1)	WORDS 1, 2	: blanks
2)	WORD 3	: pattern number
3)	WORD 4	: pattern scale
4)	WORD 5	: number of words in vertex record
5)	WORD 6	: X reference point
6)	WORD 7	: Y reference point
7)	WORDS 8-13	: pattern identification
8)	WORDS 14, 15	: 'POLY'
9)	WORD 16	: number of polygons in vertex record

RECORD 3: VERTEX RECORD (MAXIMUM OF 2,000 WORDS)

1)	WORD 1	: number of sides of first polygon
2)	WORDS 2-n	: vertices of first polygon

•
•
•
definition of remaining polygons
•
•
•

RECORD 4: END LEVEL RECORD (16 WORDS)

1)	WORDS 1, 2	: 'ENDLEV'
2)	WORDS 3-16	: blanks

Table 2.1

SECTION III

KEY ARRAYS IN THE ARTWORK PROGRAM

A group of key arrays, set up in the main program, governs the placing of patterns from the library tape. A brief description of each array is given below.

NDIR
Index to NPATM array. Plus value denotes standard pattern. Zero means pattern has been put out. Minus denotes pattern set.

Size limit is 500.

NPATM	NPTATM	PATXM	PATYM
Component number to be placed.	Orientation of component. If component is a pattern set, the value, excluding units digit, is the index of pattern set in NPS array.	Gives the X and Y locations at which patterns and pattern sets are to be placed.	

Size limit is 500.

NPSA	NPSTAT	PSAX	PSAY
Pattern numbers in pattern set.	Orientation of patterns in pattern sets plus one. Made negative where pattern is outputted for each level.	Location to place pattern in pattern set.	

Size limit is 500.

NPS	NPPS	NPSDIR
Component number of pattern set.	Number of patterns in set.	Index of start of pattern set in NPSA.

Size limit is 50.

The NDIR and NPSTAT arrays are set up for each level in the main program. When a pattern is read from the library tape by LSFILL, NDIR is scanned and used as an index to NPATM. In the case of simple patterns, this determines if and where the pattern should be placed. After placing, the NDIR entry is set to zero. For pattern sets, the absolute value of the negative entry in NDIR indexes the pattern set in NPATM. The NPTATM entry in this case, when divided by 10, gives the index of the pattern set in NPS. The corresponding NPPS and NPSDIR entries give the range and index of the patterns in the pattern set for the NPSA array. When a pattern in a pattern set is output, the number in NPSTAT is set to a negative value. When all the orientations in NPSTAT are negative, the entry in NDIR is set to zero. Before the next pattern is read from the library tape, NDIR is shortened by eliminating the zero entries.

SECTION IV

OUTLINE AND FILL SUBROUTINE

The following paragraphs, together with the flowcharts and commented listing, are intended to be a detailed explanation of the outline and fill subroutine, hereafter referred to as "FILL." The supporting subroutines, CROSS, CORNER, and RPLOT, are also described in order to present a total picture of the functions performed by the fill-in subroutine.

The FILL subroutine can be best described by orienting the discussion toward the five major functions performed by the subroutine: SEQUENCE DETECTOR, SHRINKER, REJECTOR, SMASHER, FILL-IN.

4.1 INPUT DATA

Input data are supplied to FILL through common. The common variables that must be set prior to calling FILL are:

SOVLP Slit overlap constant, set to 0.001 by the main program.

KSMA
KOUTL Smash, outline, and fill-smash option keys.
KFSMA

XB Coordinates of polygon vertices. The first point need
YB not be repeated.

NSIDES The number of sides of the input polygon.

BORMAX The width of the border desired on the polygon outline. If set to zero, the outline is generated with the smallest round aperture.

NLR The number of round apertures available for drawing lines. NLR must not exceed 18.

SIZLR A list of the sizes of round apertures available. The sizes must appear in the SIZLR array in ascending order.

NAPLR A corresponding list of Gerber D-numbers for the sizes in the SIZLR array.

NSQ The number of available horizontal slit apertures. This must be the same as the number of available vertical slit apertures. NSQ must not exceed 9. It may be set to zero, in which case no slit apertures will be used.

SIZQH
NAPQH Arrays of horizontal slit sizes and D-numbers arranged in ascending order by size.

SIZQV
NAPQV Similar arrays for vertical slits. Vertical slits have their longest dimension in the Y-direction.

NFIRST The array subscript for the smallest desired round aperture. This is usually one, but can be set to larger integers if high accuracy is not required on outlines.

4.2 SEQUENCE DETECTOR

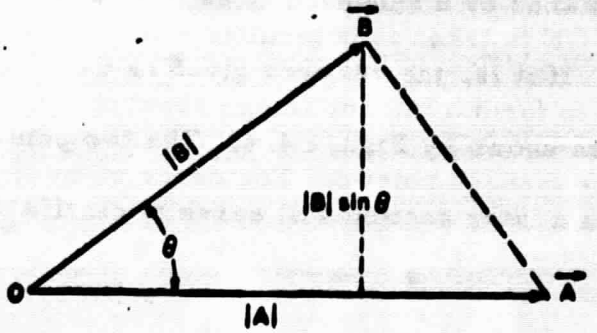
Polygons used as input to FILL must have no more than 200 sides and the number of interior angles larger than 180 degrees (reentrant angles) must not exceed 100. An input polygon must not intersect itself; it must not look like a figure eight; and it must not have three successive points in a straight line. It must be large enough so that after it is shrunk by the radius of the smallest aperture to be used it will not intersect itself.

The purpose of the sequence detector is to assure that the coordinates of the polygon vertices stores in XB and YB are in a clockwise sequence. If the sequence is known to be clockwise, it is a simple matter to determine whether a particular angle is reentrant.

The quantity QB is used as a summing variable to sum cross products of vectors with ends XB and YB. The final QB is twice the area of the polygon, as shown by Figures 4.1 and 4.2. A negative area implies that the sequence is clockwise. A positive area causes the sequence of vertices to be reversed.

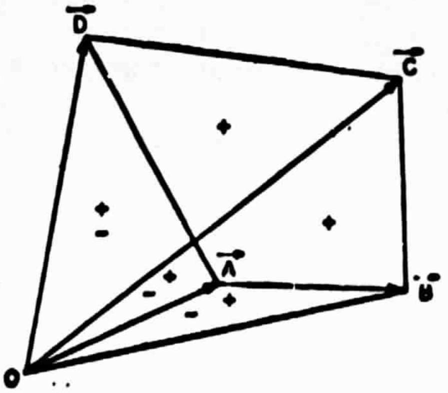
4.3 SHRINKER

SHRINKER receives and changes the vertes information stored in XB and YB to produce a new polygon with vertices moved inward along angle bisectors in a distance such that the new sides are half



$$\bar{A} \times \bar{B} = |A| \cdot |B| \sin\theta$$
$$= 2 \text{ area of } \Delta OAB$$

Figure 4.1 Area of Triangle



$$\bar{A} \times \bar{B} + \bar{B} \times \bar{C} + \bar{C} \times \bar{D} + \bar{D} \times \bar{A}$$
$$= 2 \text{ area of } ABCD$$

Figure 4.2 Sequence Detector

the diameter of the smallest aperture inside the original sides. An example of a shrunken polygon is shown by Figure 4.3

Shrinking is accomplished by a sequence of calls to subroutine CORNER. A cross product QE is taken to determine whether a given vertex is a reentrant angle. If it is, the vertices given as CORNER arguments must be changed to form a mirror image about the vertex before CORNER is called, as shown by Figure 4.4. The two points A' and B' are mirror images through point 0. The description of subroutine CORNER given in a later section will serve to clarify this.

4.4 REJECTOR

The purpose of the REJECTOR part of the program is to check a shrunken polygon to see if it intersects itself. Bad input data for polygons are thus found and print-outs are made to describe possible error points in the polygon. The possibility of an illegal polygon's being processed by the FILL program is thus avoided. Such an illegal polygon would cause an error which might require the computer to stop. A knowledge of the inputs and operations of subroutine CROSS, which is documented in a later paragraph, is useful in understanding this test.

4.5 SMASHER

The SMASHER part of the program breaks up a legal polygon stored in XB and YB to produce a number of smaller polygons in XA and YA so that none has a reentrant angle. This is accomplished by "cutting" or breaking up polygons at the reentrant vertices until all reentrant vertices have been eliminated.

It will be important in following this section to know how groups of polygons are stored in XA and YA. Comments supplied in the listing explain the variables used to store groups of polygons.

The KSMA option key is sensed in this part of the program. The smash line referred to in the comments is the line of the "cut" used to remove a reentrant angle. This line must be through the reentrant vertex and can be either the bisector of the reentrant angle or an extension of one of the reentrant angle sides, depending on the value of KSMA. When dealing with polygons made mostly of horizontal and vertical lines, it is useful to have smash lines that are extensions of reentrant angle sides, so that no unnecessary diagonal lines are added to intermediate stages of the artwork.

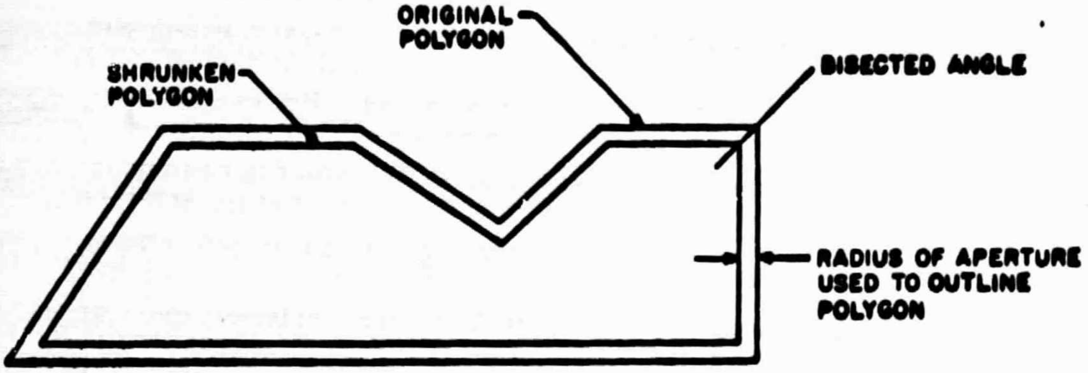


Figure 4.3 Shrinker Program

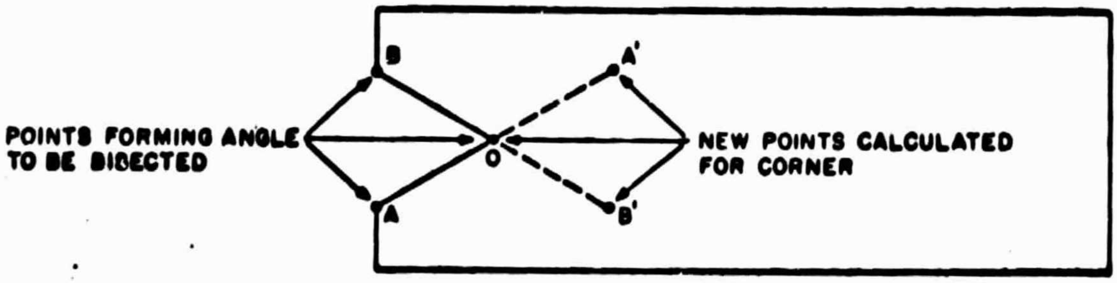

B
A'
POINTS FORMING ANGLE TO BE BISECTED
O
NEW POINTS CALCULATED FOR CORNER
A
B'

Figure 4.4 Shrinking at Reentrant Angles

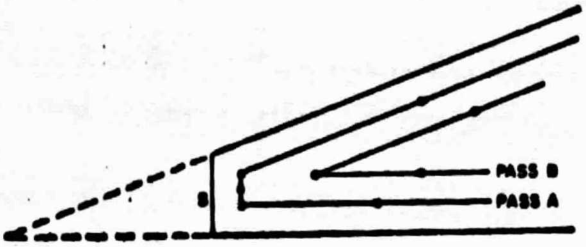
REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

4.6 FILL-IN

This part of the program determines the actual plotter motions needed to fill in, one at a time, the convex polygons stores in XA and YA. If slit apertures are available, each polygon is first tested to see if it is a rectangle and vertical and horizontal sides. Rectangles are filled in by setting upper and lower X and Y boundaries and calling RPLOT, which generates the short sequence of movements needed to fill in the rectangle. If a polygon is not a rectangle and if the KFSMA (secondary smash) option has been specified, the program will check the polygon in an attempt to find three successive sides that are either horizontal or vertical. If the polygon satisfies this condition, part of the polygon is filled with slits by calling RPLOT. The remainder of the polygon is then returned to XA and YA for further tests. The outline option, KOUTL, applies in any case where RPLOT is used to fill all or part of a polygon. Outlining of rectangular areas is generally done after RPLOT is called, in case a condition exists which will make use of slits impossible. This would happen if a rectangle were too small to be filled by the available slits.

In the general case, where there are no suitable slit apertures available, a polygon is filled using round apertures. The largest aperture that will fit into the smallest internal angle without leaving a gap is chosen and used around the polygon. This process continues until a side becomes "over-run" and is eliminated, as shown by Figure 4.5. The process of filling and eliminating sides continues until the polygon is filled completely, or until the limit on the border, BORMAX, is reached. It will be necessary to study the commented program listing at the end of this report to acquire a full understanding of FILL-IN details.

A question might arise at comment 104 in the listing concerning formation of a triangle when sides (J, J+1) and (J+2, J+3) intersect on the "wrong" or "abnormal" side of the polygon. Figure 4.6 shows the "normal" orientation of the intersection point. In the normal case, side (J+1, J+2) is removed at this stage of the FILL program, but a special situation occurs when (J, J+1) and (J+2, J+3) intersect as shown in Figure 4.7. In this situation side (J+1, J+2) should not be removed. Instead, a triangle is formed from vertices J+1, J+2, and the intersection point. The triangle is a sufficient definition of the unfilled part of the original polygon. FILL-IN proceeds from there.



Pass A is made before side "s" is eliminated.

Pass B is made after side "s" is eliminated.

Figure 4.5 Side Elimination

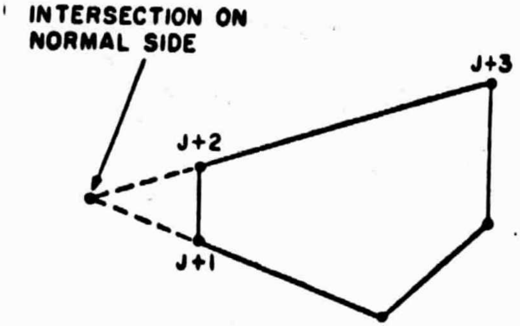


Figure 4.6 Normal Orientation of Intersection Point

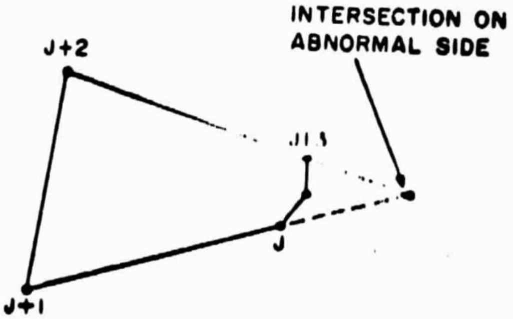


Figure 4.7 Abnormal Orientation of Intersection Point

4.7 AUXILIARY SUBROUTINES

Three other short subroutines are used by FILL. A description of each subroutine follows. Details can be easily obtained from the listing.

4.7.1 Subroutine CROSS

Given four points that define two straight lines, subroutine CROSS determines:

- **If there is an intersection of the two lines;**
- **The intersection point;**
- **If the intersection point is on the second line segment.**

If an intersection anywhere on the two lines is desired, PTS (12) is set to +1. If an intersection on the second line segment is desired, PTS(12) is set to -1. If no suitable intersection can be found, PTS(11) is set to -1 by the program. Otherwise it is +1 for a suitable intersection. Coincident lines are considered to have no intersection. The details of the subroutine are easily followed by a study of the program listing.

4.7.2 Subroutine CORNER

Given three points which describe an angle, subroutine CORNER determines the sine of the half angle and the plotting point (XVER, YVER) knowing the existing border width and selected aperture size. Figure 4.8 shows this point.

The subroutine first selects a point on the angle bisector by summing two unit vectors along the sides of the angle. The sine of the half angle is found by using the cross product of the unit vectors, and the point (XVER, YVER) is found by using a formula which is easily derived from Figure 4.8.

4.7.3 Subroutine RPLOT

Given the X and Y coordinates of the boundaries for a rectangle, as shown by Figure 4.9, and a list of available slit apertures (their sizes and D-numbers), RPLOT generates the Gerber Artwork Generator commands needed to fill in the rectangle.

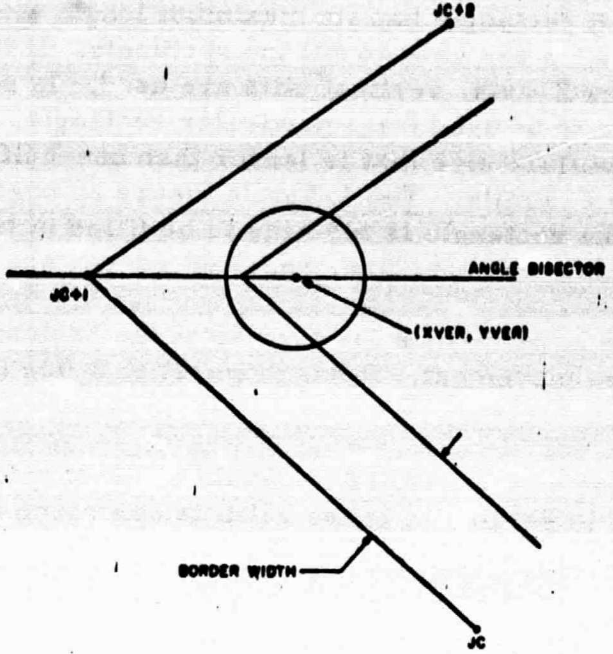


Figure 4.8 Determination of Plotting Point

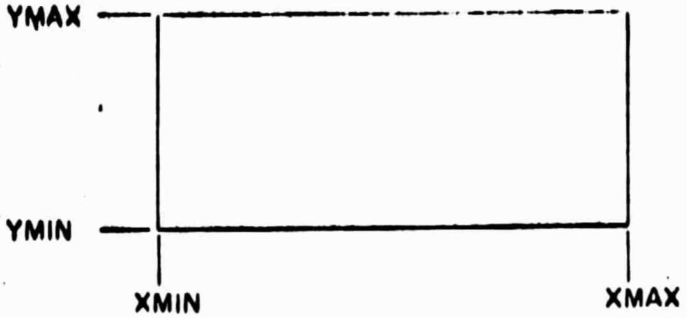


Figure 4.9 Rectangle Definition

If the input rectangle has its maximum length along the Y-axis, horizontal slits are used to fill the rectangle. If the greatest length is along the X-axis, vertical slits are used. In selecting a slit aperture size to be used for a particular rectangle, the program picks the first aperture size that is larger than one-half of the rectangle width plus 1.5 mils. The 1.5 mils insure an overlap of the two slit passes. If the rectangle is too wide to be filled in two passes by the largest available slit aperture, repeated passes are made.

SOLVP is an allowance for overlap at the beginning and end of each slit aperture movement. It has been set to 0.001 (1 mil) with good results.

In any case where RPLOT is unable to fill a rectangle properly, the variable IGO is set to 1 to cause FILL to use round apertures on the rectangle.

4.8 OUTPUT

All of the tape output from subroutine FILL is made through subroutine PLOT. This subroutine performs many tasks independent of those required by FILL. For PLOT to produce valid Gerber commands, and to be compatible with FILL, it must have these capabilities:

- PLOT must obtain information about new plotter motions from X and Y in common and from its argument list;

- The first argument must be either 1 or 2, depending on whether the light is to be on or off during the motion;

- The second argument must be the aperture D-number. It may be zero for a light-off command.

- PLOT must sense a change in the aperture being used and must generate an aperture select command when such a change occurs.

- Each call to PLOT must generate a Gerber command (or two in the case that an aperture select is necessary) and either add the command to a buffer or write it on a tape.

For further details about the PLOT subroutine used for BANNING see the description of the other artwork routines.

4.9 FLOW CHARTS

The following pages constitute the flow charts for subroutine FILL. Flowcharts of the auxiliary routines were not included since they are short, and simple enough to be understandable from a description and the commented listing.

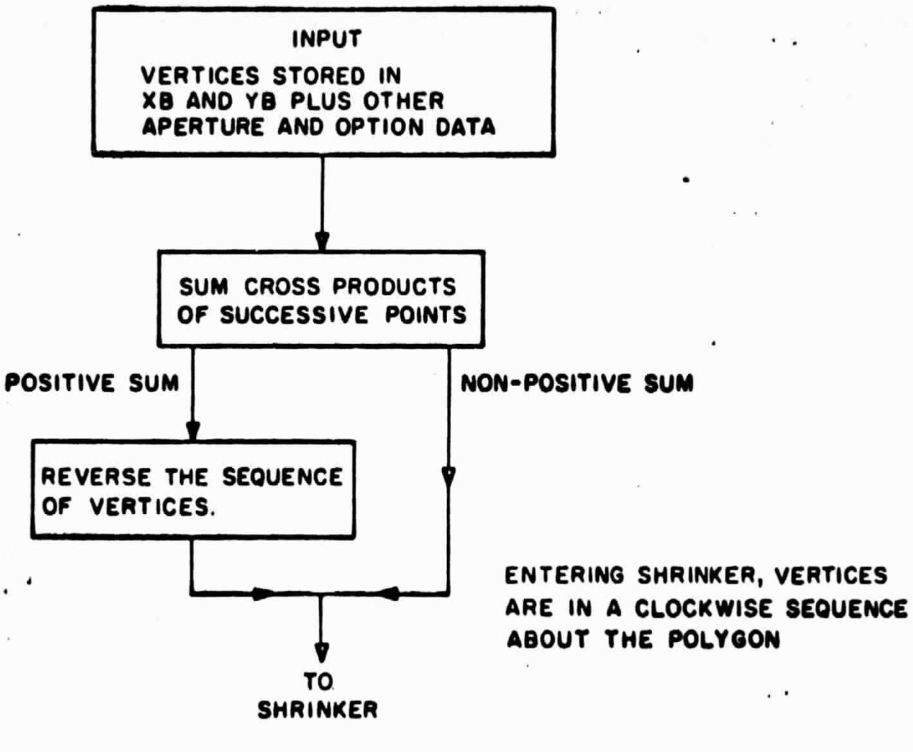
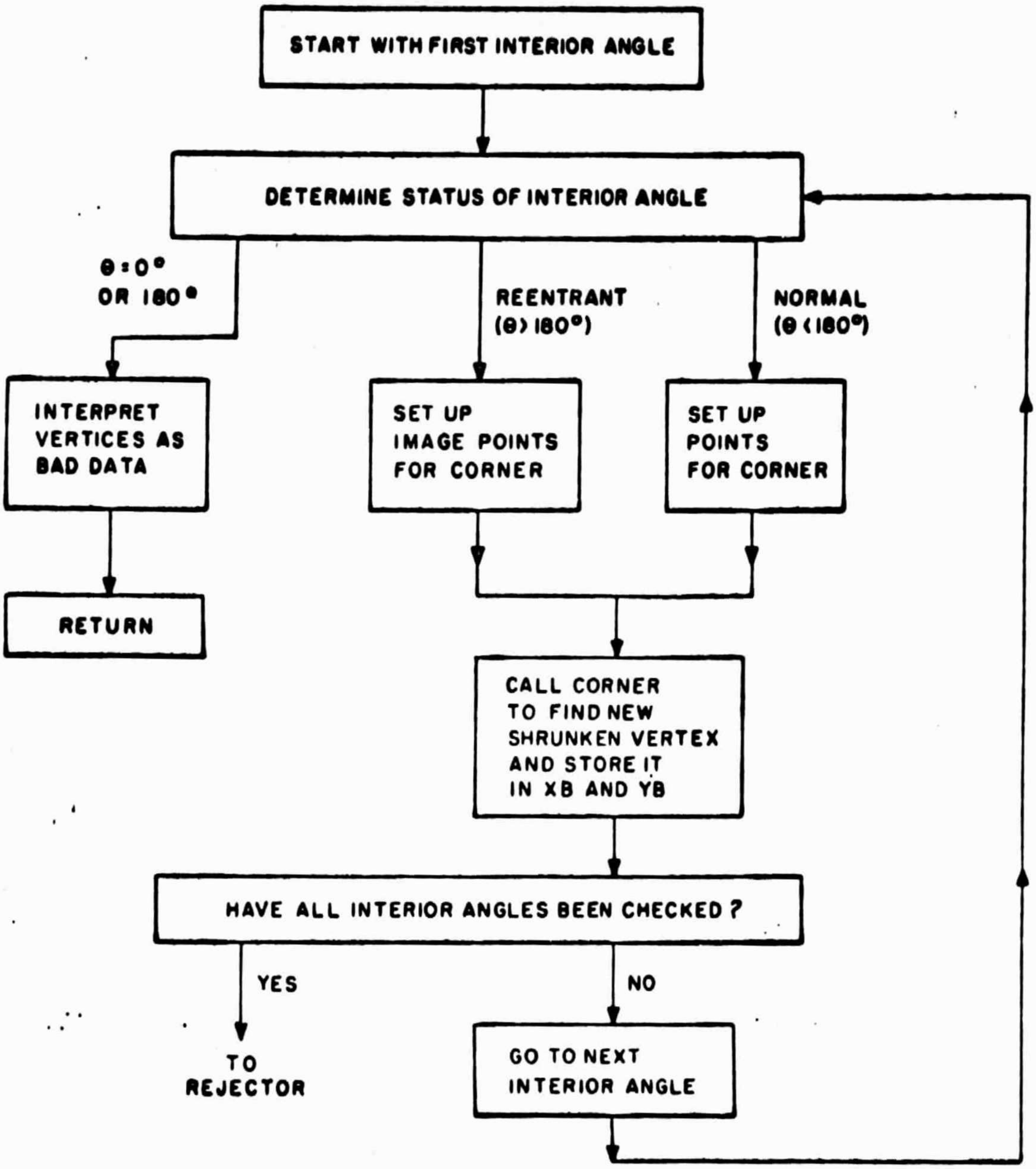
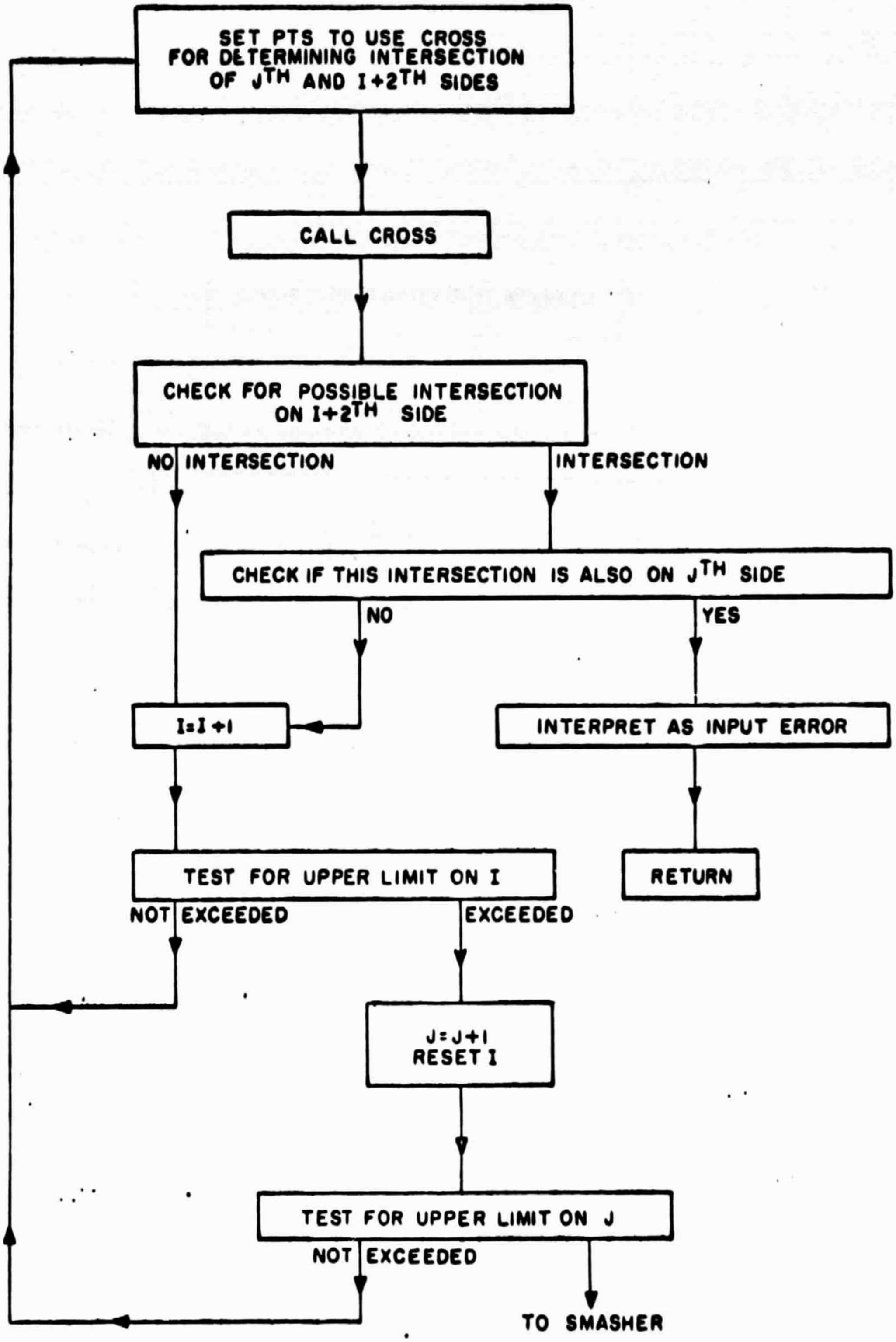

INPUT
VERTICES STORED IN XB AND YB PLUS OTHER APERTURE AND OPTION DATA
SUM CROSS PRODUCTS OF SUCCESSIVE POINTS
POSITIVE SUM
NON-POSITIVE SUM
REVERSE THE SEQUENCE OF VERTICES.
ENTERING SHRINKER, VERTICES ARE IN A CLOCKWISE SEQUENCE ABOUT THE POLYGON
TO SHRINKER

Figure 4.10 Sequence Detector


START WITH FIRST INTERIOR ANGLE
DETERMINE STATUS OF INTERIOR ANGLE
θ = 0°
OR 180°
REENTRANT
(θ > 180°)
NORMAL
(θ < 180°)
INTERPRET VERTICES AS BAD DATA
SET UP IMAGE POINTS FOR CORNER
SET UP POINTS FOR CORNER
RETURN
CALL CORNER TO FIND NEW SHRUNKEN VERTEX AND STORE IT IN XB AND YB
HAVE ALL INTERIOR ANGLES BEEN CHECKED?
YES
NO
TO REJECTOR
GO TO NEXT INTERIOR ANGLE

ENTERING REJECTOR, THE STORED VERTICES ARE THOSE OF A SHRUNKEN POLYGON CORRESPONDING TO THE STARTING POLYGON.

Figure 4.11 Shrinker

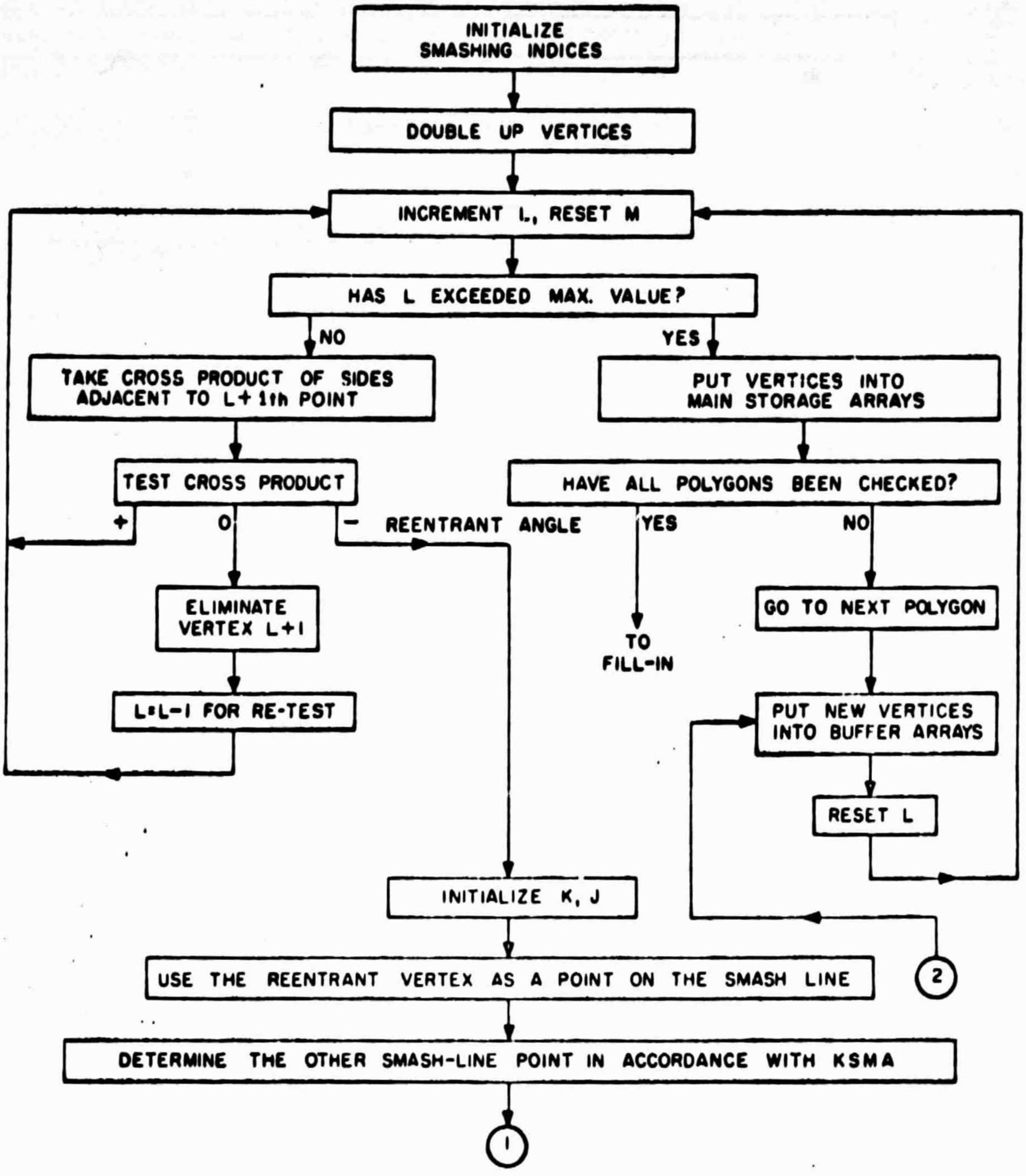

SET PTS TO USE CROSS FOR DETERMINING INTERSECTION OF J^TH AND I+2^TH SIDES
CALL CROSS
CHECK FOR POSSIBLE INTERSECTION ON I+2^TH SIDE
NO INTERSECTION
INTERSECTION
CHECK IF THIS INTERSECTION IS ALSO ON J^TH SIDE
NO
YES
I=I+1
INTERPRET AS INPUT ERROR
TEST FOR UPPER LIMIT ON I
RETURN
NOT EXCEEDED
EXCEEDED
J=J+1 RESET I
TEST FOR UPPER LIMIT ON J
NOT EXCEEDED
TO SMASHER

AS SMASHER IS ENTERED, POLYGON VERTICES ARE IN THE BUFFER STORAGE ARRAYS XB AND YB

Figure 4.12 Rejector

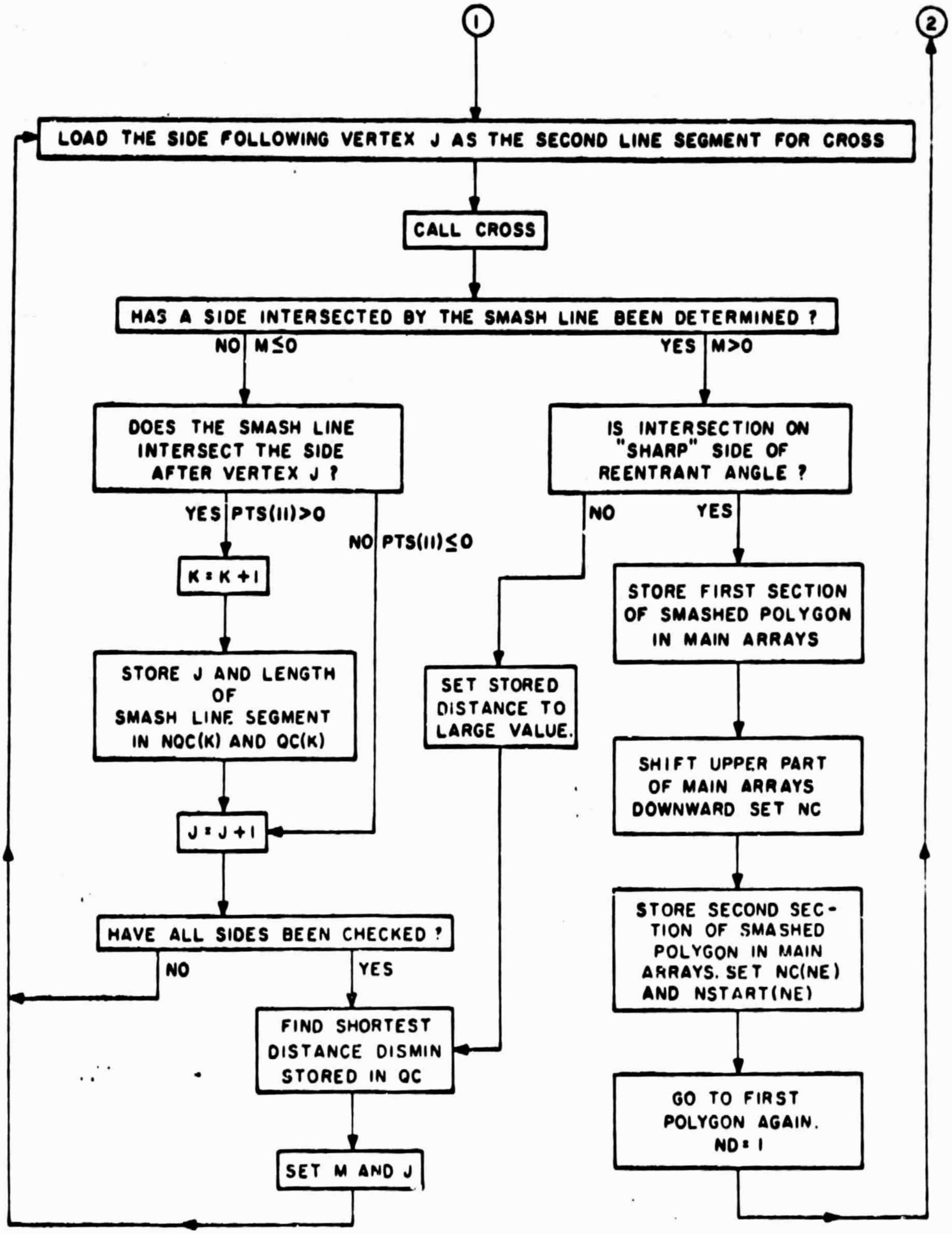
L = INDEX OF THE VERTEX BEFORE THE REENTRANT ANGLE

M = STARTING INDEX OF THE SIDE INTERSECTED BY THE SMASH LINE


INITIALIZE SMASHING INDICES
DOUBLE UP VERTICES
INCREMENT L, RESET M
HAS L EXCEEDED MAX. VALUE?
NO
YES
TAKE CROSS PRODUCT OF SIDES ADJACENT TO L+1th POINT
PUT VERTICES INTO MAIN STORAGE ARRAYS
TEST CROSS PRODUCT
+
0
− REENTRANT ANGLE
HAVE ALL POLYGONS BEEN CHECKED?
YES
NO
ELIMINATE VERTEX L+1
TO FILL-IN
GO TO NEXT POLYGON
L=L−1 FOR RE-TEST
PUT NEW VERTICES INTO BUFFER ARRAYS
RESET L
INITIALIZE K, J
USE THE REENTRANT VERTEX AS A POINT ON THE SMASH LINE
2
DETERMINE THE OTHER SMASH-LINE POINT IN ACCORDANCE WITH KSMA
1

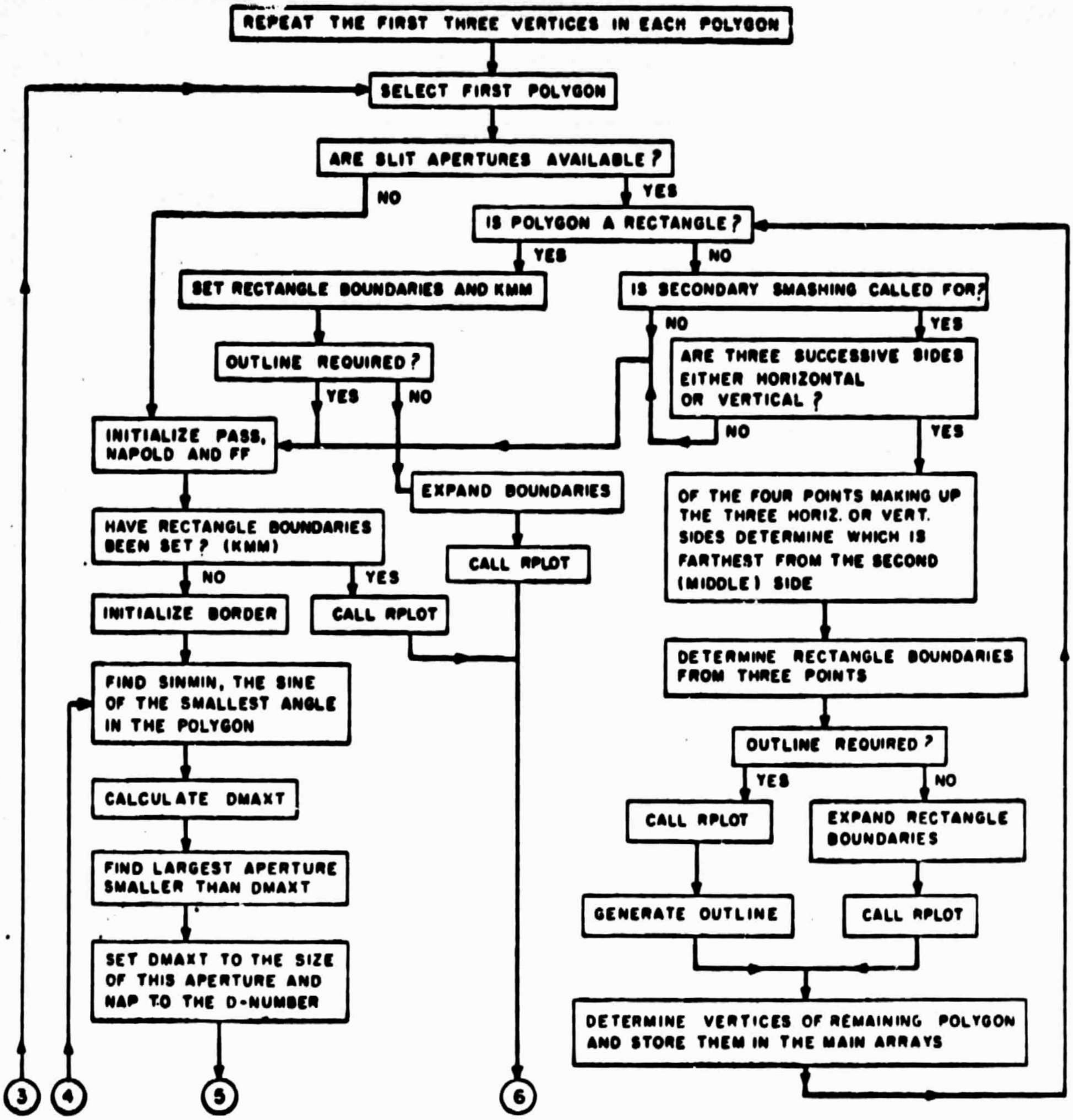
(CONTINUED ON NEXT PAGE)

Figure 4.13 Smasher


1
2
LOAD THE SIDE FOLLOWING VERTEX J AS THE SECOND LINE SEGMENT FOR CROSS
CALL CROSS
HAS A SIDE INTERSECTED BY THE SMASH LINE BEEN DETERMINED ?
NO M≤0
YES M>0
DOES THE SMASH LINE INTERSECT THE SIDE AFTER VERTEX J ?
YES PTS(II)>0
NO PTS(II)≤0
K=K+1
STORE J AND LENGTH OF SMASH LINE SEGMENT IN NOC(K) AND QC(K)
J=J+1
HAVE ALL SIDES BEEN CHECKED ?
NO
YES
FIND SHORTEST DISTANCE DISMIN STORED IN QC
SET M AND J
IS INTERSECTION ON "SHARP" SIDE OF REENTRANT ANGLE ?
NO
YES
SET STORED DISTANCE TO LARGE VALUE.
STORE FIRST SECTION OF SMASHED POLYGON IN MAIN ARRAYS
SHIFT UPPER PART OF MAIN ARRAYS DOWNWARD SET NC
STORE SECOND SECTION OF SMASHED POLYGON IN MAIN ARRAYS. SET NC(NE) AND NSTART(NE)
GO TO FIRST POLYGON AGAIN. ND=1

AS FILL-IN IS ENTERED THE MAIN ARRAYS CONTAIN NE CONVEX POLYGONS WHICH WILL BE FILLED ONE AT A TIME BY FILL-IN.

Figure 4.14 Smasher


REPEAT THE FIRST THREE VERTICES IN EACH POLYGON
SELECT FIRST POLYGON
ARE SLIT APERTURES AVAILABLE ?
NO
YES
IS POLYGON A RECTANGLE ?
YES
NO
SET RECTANGLE BOUNDARIES AND KMM
IS SECONDARY SMASHING CALLED FOR?
NO
YES
OUTLINE REQUIRED ?
YES
NO
ARE THREE SUCCESSIVE SIDES EITHER HORIZONTAL OR VERTICAL ?
NO
YES
INITIALIZE PASS, NAPOLD AND FF
EXPAND BOUNDARIES
OF THE FOUR POINTS MAKING UP THE THREE HORIZ. OR VERT. SIDES DETERMINE WHICH IS FARTHEST FROM THE SECOND (MIDDLE) SIDE
HAVE RECTANGLE BOUNDARIES BEEN SET ? (KMM)
CALL RPLOT
NO
YES
INITIALIZE BORDER
CALL RPLOT
DETERMINE RECTANGLE BOUNDARIES FROM THREE POINTS
FIND SINMIN, THE SINE OF THE SMALLEST ANGLE IN THE POLYGON
OUTLINE REQUIRED ?
YES
NO
CALCULATE DMAXT
CALL RPLOT
EXPAND RECTANGLE BOUNDARIES
FIND LARGEST APERTURE SMALLER THAN DMAXT
GENERATE OUTLINE
CALL RPLOT
SET DMAXT TO THE SIZE OF THIS APERTURE AND NAP TO THE D-NUMBER
DETERMINE VERTICES OF REMAINING POLYGON AND STORE THEM IN THE MAIN ARRAYS
3
4
5
6

(CONTINUED ON NEXT PAGE)

Figure 4.15 Fill-In

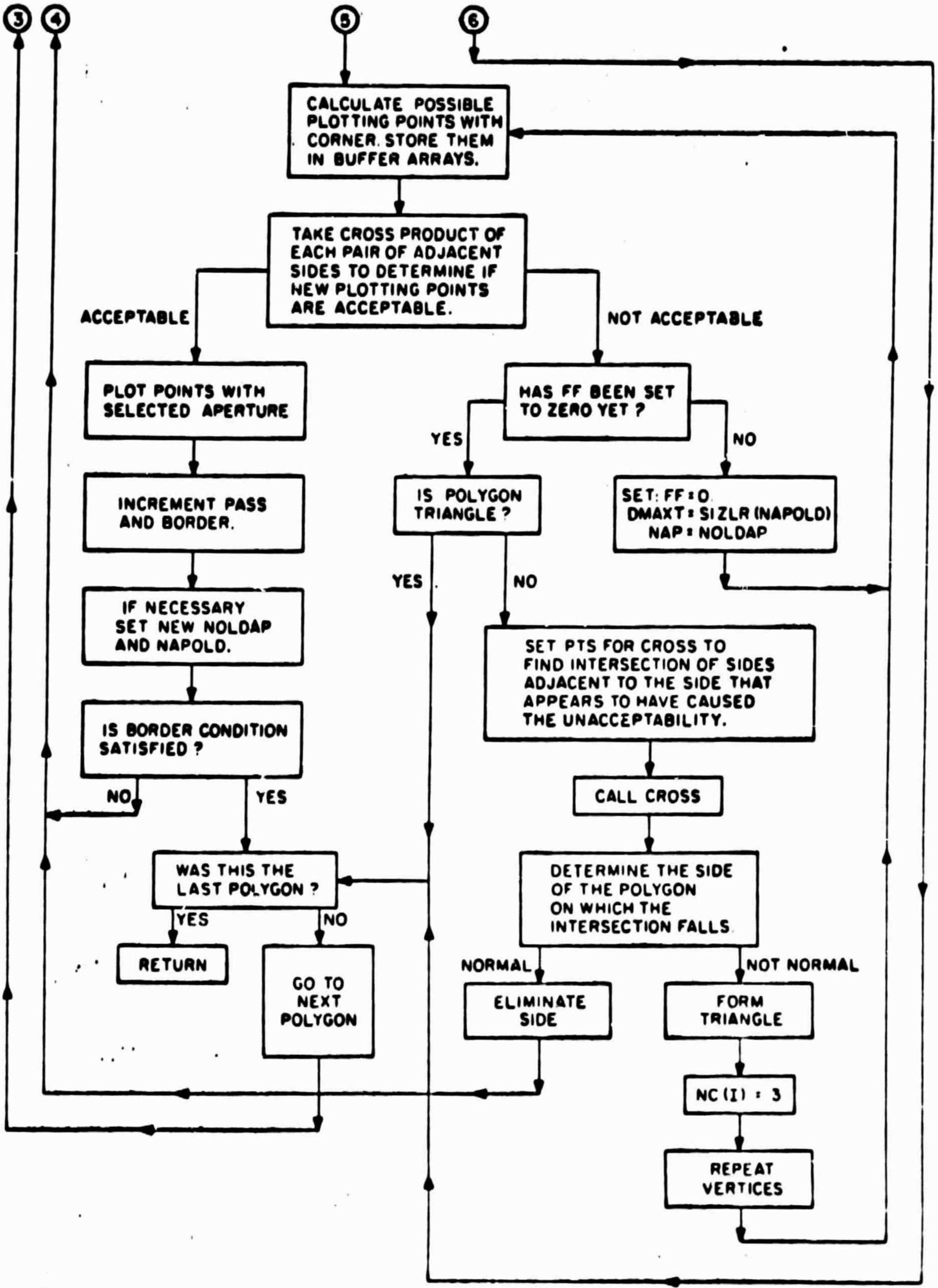

3
4
5
6
CALCULATE POSSIBLE PLOTTING POINTS WITH CORNER. STORE THEM IN BUFFER ARRAYS.
TAKE CROSS PRODUCT OF EACH PAIR OF ADJACENT SIDES TO DETERMINE IF NEW PLOTTING POINTS ARE ACCEPTABLE.
ACCEPTABLE
NOT ACCEPTABLE
PLOT POINTS WITH SELECTED APERTURE
HAS FF BEEN SET TO ZERO YET ?
YES
NO
INCREMENT PASS AND BORDER.
IS POLYGON TRIANGLE ?
SET: FF = 0
DMAXT = SIZLR (NAPOLD)
NAP = NOLDAP
YES
NO
IF NECESSARY SET NEW NOLDAP AND NAPOLD.
SET PTS FOR CROSS TO FIND INTERSECTION OF SIDES ADJACENT TO THE SIDE THAT APPEARS TO HAVE CAUSED THE UNACCEPTABILITY.
IS BORDER CONDITION SATISFIED ?
NO
YES
CALL CROSS
WAS THIS THE LAST POLYGON ?
DETERMINE THE SIDE OF THE POLYGON ON WHICH THE INTERSECTION FALLS
YES
NO
RETURN
GO TO NEXT POLYGON
NORMAL
NOT NORMAL
ELIMINATE SIDE
FORM TRIANGLE
NC (I) = 3
REPEAT VERTICES

Figure 4.16 Fill-In

SECTION V

SPECIAL PLOT OPTIONS

Access to the Special Plot Options is made through the Utility Routine. There are five special plots that can be specified:

1) **Dashed outling,**

2) **Cross hatch,**

3) **Normal fill,**

4) **Outline only, and**

5) **Cross hatch.**

Options 2 and 5 are identical, but the angle of inclination of the cross hatch lines may be positive for one and negative for the other, resulting in a criss-cross pattern. See the Artwork Program User's Manual for convention definitions and examples.

The subroutines associated with the special plot options are NOTL, NPLOT, and SCAN. The function NBR is used by SCAN. Subroutine FILL is used in the normal fill option (option 3), but since its use has been described in the Outline-and-Fill section, it is not described here.

5.1 SUBROUTINE NOTL (NAP, DS, DL)

Subroutine NOTL is a new outline routine that is used in the outline and the dashed outline options (options 4 and 1). Its purpose is to use NPLOT, a new plot routine, to make plot outlines of polygons from XB and YB arrays in COMMON. The arguments of NOTL are not used in NOTL itself, but they must be passed on to NPLOT. Outlines generated by NOTL are always centered over the "ideal" polygon sides. These are different from the outlines generated by the FILL subroutine with BORMAX = 0.0 since there the outlines generated by FILL are always "shrunk" inside the polygons enough to correct for the outline width.

5.2 SUBROUTINE NPLOT (NAP, DS, DL)

When the space between dashes, DS, is set to zero NPLOT calls PLOT, the regular output routine, immediately. If DS is not zero a dashed line is constructed between the last point plotted at coordinates (XOLDE, YOLDE) and the new end point at coordinates (X, Y). Using DS and DL (the dash length), the overall length of the dashed line to be drawn, an estimate is made of the number of dashes that will be necessary to draw the complete line segment. A certain amount of scaling is used on the dash length and dash spacing to insure that a dash occurs at each end of the segment to be drawn. This routine is short and can be understood easily from the commented listed.

5.3 SUBROUTINE SCAN (DY, THETA, NAP)

The SCAN subroutine generates the cross-hatch lines within a polygon. To define the arguments of SCAN, DY is the spacing between the cross-hatch lines, THETA is the angle of inclination of the cross hatch with the horizontal, and NAP is the aperture D-number that must be used during cross hatching.

Subroutine SCAN accepts polygon information through the COMMON arrays XB and YB. COMMON arrays XA and YA are given new names X and Y for use by subroutine SCAN. The COMMON variables that pass coordinates to the PLOT routine are called XI and YI to distinguish them from the normal names X and Y in the other routines.

The SCAN routine scans a polygon in a series of horizontal passes. The cross-hatch angle is introduced by rotating the entire polygon. Then, after horizontal scans are generated, they are rotated back to return the polygon to its original orientation.

In the method used to generate horizontal scans, vertices are processed starting with the highest and proceeding monotonically downward to the lowest. Scans are generated between vertices using information in the special arrays NHI and NLO. Prior to generation of a scan sequence, the NHI and NLO arrays are loaded with numbers identifying which polygon sides intersect the scan lines.

Figure 5.1 is a flowchart showing the sequence of operations in Subroutine SCAN.

5.4 CROSS-HATCH EXAMPLE

The processing of vertices consists of updating the arrays NHI and NLO. Before any sequence of scan lines can be generated, NHI

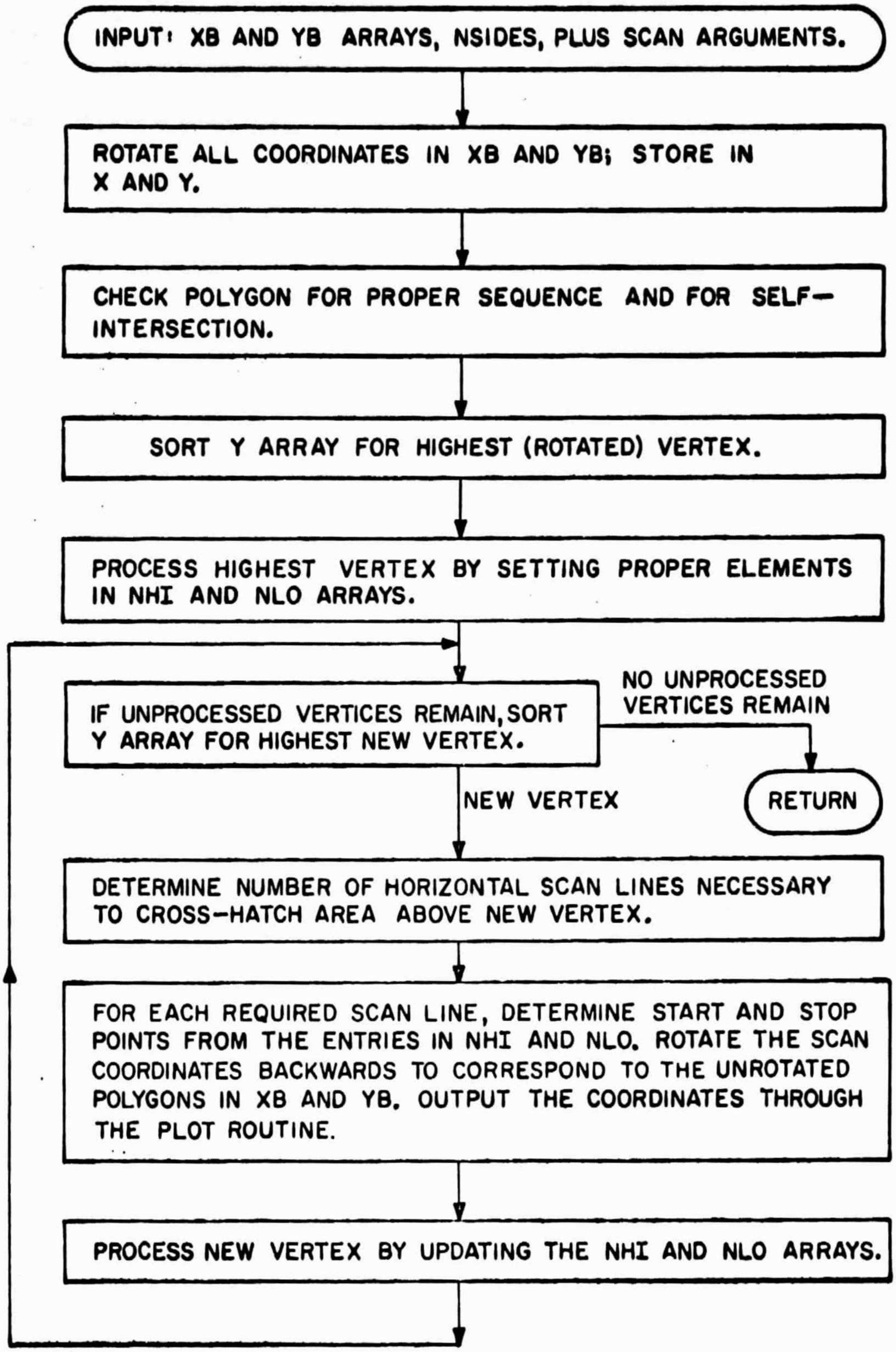

INPUT: XB AND YB ARRAYS, NSIDES, PLUS SCAN ARGUMENTS.
ROTATE ALL COORDINATES IN XB AND YB; STORE IN X AND Y.
CHECK POLYGON FOR PROPER SEQUENCE AND FOR SELF-INTERSECTION.
SORT Y ARRAY FOR HIGHEST (ROTATED) VERTEX.
PROCESS HIGHEST VERTEX BY SETTING PROPER ELEMENTS IN NHI AND NLO ARRAYS.
IF UNPROCESSED VERTICES REMAIN, SORT Y ARRAY FOR HIGHEST NEW VERTEX.
NO UNPROCESSED VERTICES REMAIN
RETURN
NEW VERTEX
DETERMINE NUMBER OF HORIZONTAL SCAN LINES NECESSARY TO CROSS-HATCH AREA ABOVE NEW VERTEX.
FOR EACH REQUIRED SCAN LINE, DETERMINE START AND STOP POINTS FROM THE ENTRIES IN NHI AND NLO. ROTATE THE SCAN COORDINATES BACKWARDS TO CORRESPOND TO THE UNROTATED POLYGONS IN XB AND YB. OUTPUT THE COORDINATES THROUGH THE PLOT ROUTINE.
PROCESS NEW VERTEX BY UPDATING THE NHI AND NLO ARRAYS.

Figure 5.1 Subroutine SCAN

and NLO must identify the upper and lower end points of every polygon side that will intersect the lines to be generated. To simplify the explanation, consider the polygon shown in Figure 5.2.

The rotated coordinates X and Y would be stored and indexed by vertex number as shown by Table 5.1.

Table 5.1 Coordinates of Vertices

Vertex Number	1	2	3	4	5	6	7	8	9	10
X	2	1	5	6	8	8.5	5.5	8	7	4
Y	3	4	8	7	9	8.5	5.5	3	2	5

A search through the Y array reveals that vertex number 5 has the highest ordinate. The processing of this first vertex would result in the NHI and NLO entries of Table 5.2, where NLO shows the vertex number at the other ends of the line segments intersecting at vertex No. 5.

Table 5.2 Update Vertex No. 5

Entry No.	1	2
NHI	5	5
NLO	4	6

The two pairs entered indicate that scan lines immediately below the point just processed must intersect line segments from vertices 5 to 4 and from vertices 5 to 6.

Another search of the Y array would give the next lower vertex, No. 6. Using the information already stored in NHI and NLO, scan lines can be generated down to the level of vertex No. 6. The next update would result in Table 5.3.

Table 5.3 Update Vertex No. 6

Entry No.	1	2
NHI	5	6
NLO	4	7

Subsequent updates would bring about the following tables.

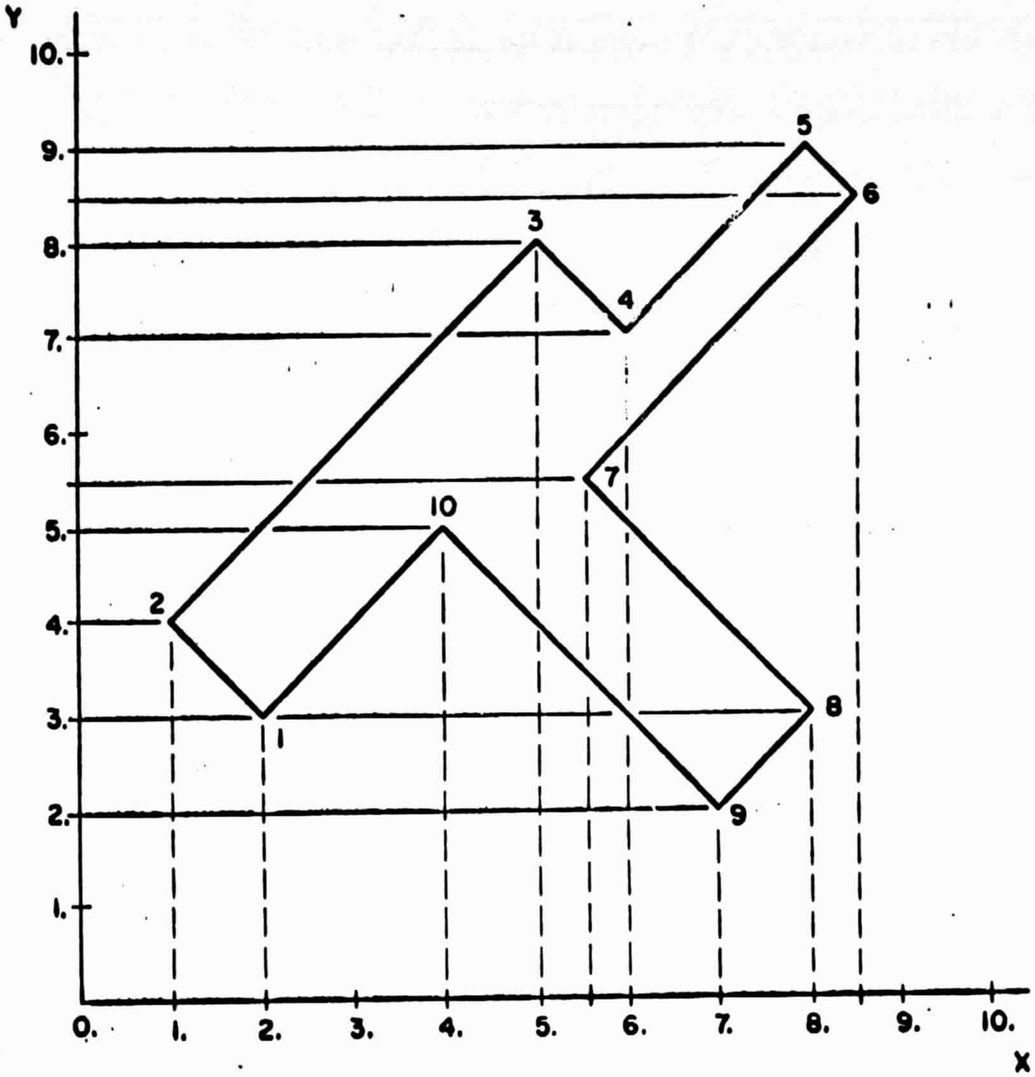


Figure 5.2 Rotated Polygon

Table 5.4 Update Vertex No. 3

Entry No.	1	2	3	4
NHI	3	3	5	6
NLO	2	4	4	7

Table 5.5 Update Vertex No. 4

Entry No.	1	2
NHI	3	6
NLO	2	7

Table 5.6 Update Vertex No. 7

Entry No.	1	2
NHI	3	7
NLO	2	8

Table 5.7 Update Vertex No. 10

Entry No.	1	2	3	4
NHI	3	10	10	7
NLO	2	1	9	8

5.5 THE UPDATE PROCESS

The updating of the NHI and NLO arrays, as each new vertex is encountered, is relatively complex. The action taken for a particular update depends on the orientation of the vertex causing the update. For instance, if the vertex causing the update is on an upward-pointing peak, the update consists of introducing two new element pairs into the NHI and NLO arrays. The two new pairs correspond to the two line segments (polygon sides) immediately below the vertex. A downward pointing peak causes two element pairs in NHI and NLO to be deleted.

Figure 5.3 shows all of the configurations that require different update procedures. Each one of the cases illustrated in Figure 5.3 contains either one isolated or two horizontal vertex points. In the three cases where two horizontal vertices are involved in an update, the update vertices are end points of a horizontal side.

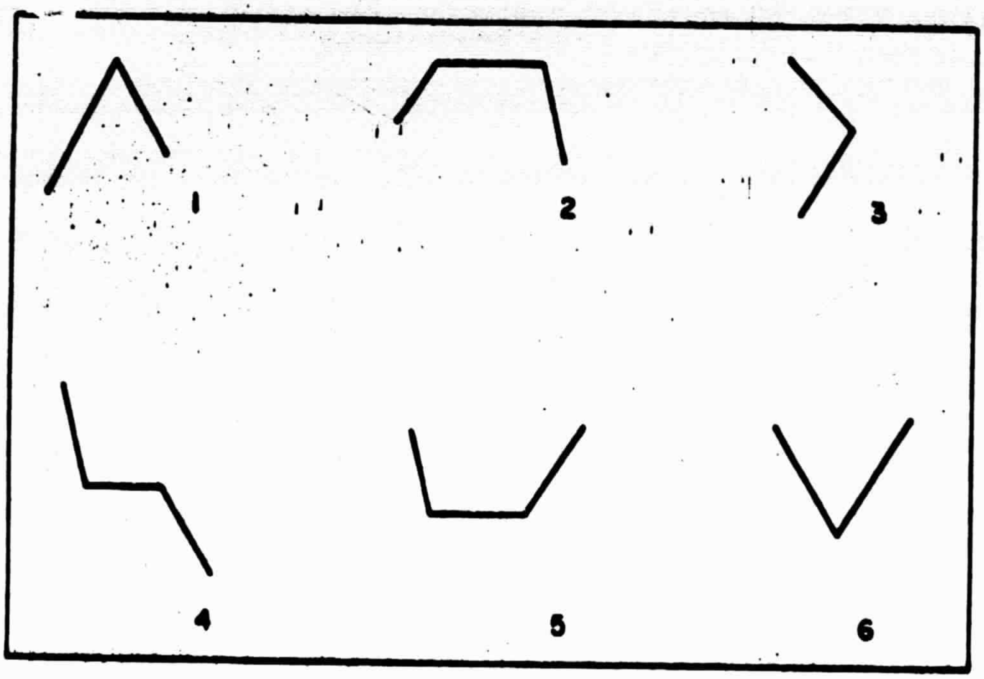


Figure 5.3 Update Configurations, Cases 1-6

When selection of a vertex makes an update necessary, it must be decided which of the six configurations of Figure 5. 3 apply. Then the procedure that corresponds to that configuration must be used to bring about the update.

Figure 5. 4 is an expansion of the lowest block in the flowchart of Figure 5. 2. It demonstrates the procedure used to detect and update the cases of Figure 5. 3. The procedure used here detects case 4 in one of two ways (causes 4A and 4B), depending upon which end point of the horizontal segment of case 4 is selected by the computer during the Y-coordinate search operation. Further details can be obtained from the comments in the SCAN listing.

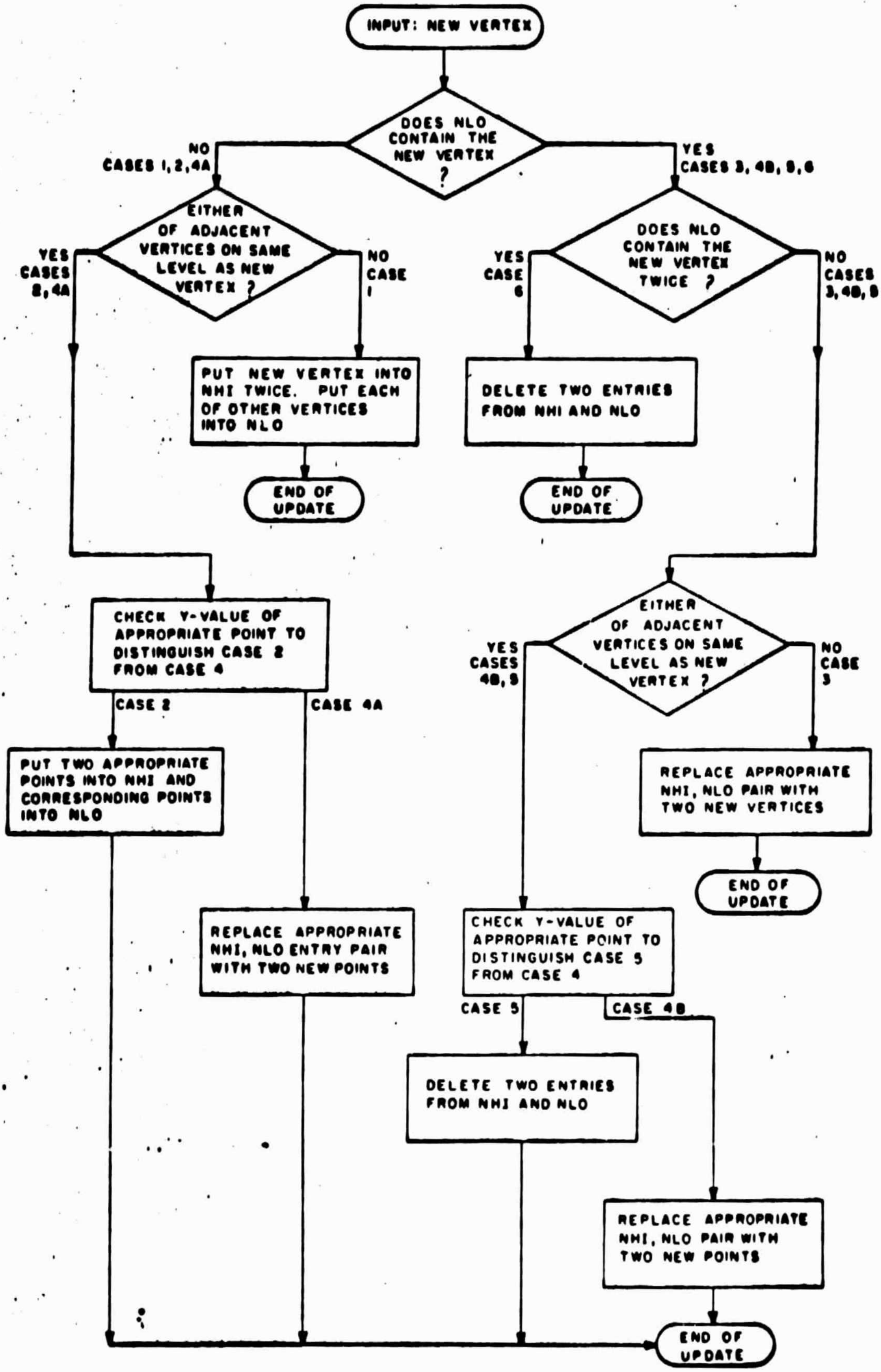

INPUT: NEW VERTEX
DOES NLO CONTAIN THE NEW VERTEX ?
NO
CASES 1, 2, 4A
YES
CASES 3, 4B, 5, 6
EITHER OF ADJACENT VERTICES ON SAME LEVEL AS NEW VERTEX ?
YES
CASES 2, 4A
NO
CASE 1
PUT NEW VERTEX INTO NHI TWICE. PUT EACH OF OTHER VERTICES INTO NLO
END OF UPDATE
DOES NLO CONTAIN THE NEW VERTEX TWICE ?
YES
CASE 6
NO
CASES 3, 4B, 5
DELETE TWO ENTRIES FROM NHI AND NLO
END OF UPDATE
CHECK Y-VALUE OF APPROPRIATE POINT TO DISTINGUISH CASE 2 FROM CASE 4
CASE 2
CASE 4A
PUT TWO APPROPRIATE POINTS INTO NHI AND CORRESPONDING POINTS INTO NLO
REPLACE APPROPRIATE NHI, NLO ENTRY PAIR WITH TWO NEW POINTS
EITHER OF ADJACENT VERTICES ON SAME LEVEL AS NEW VERTEX ?
YES
CASES 4B, 5
NO
CASE 3
REPLACE APPROPRIATE NHI, NLO PAIR WITH TWO NEW VERTICES
END OF UPDATE
CHECK Y-VALUE OF APPROPRIATE POINT TO DISTINGUISH CASE 5 FROM CASE 4
CASE 5
CASE 4B
DELETE TWO ENTRIES FROM NHI AND NLO
REPLACE APPROPRIATE NHI, NLO PAIR WITH TWO NEW POINTS
END OF UPDATE

Figure 5.4 Update Flowchart

U.S. GOVERNMENT PRINTING OFFICE 1976-641-255/392 REGION NO 4