General Disclaimer

One or more of the Following Statements may affect this Document

- This document has been reproduced from the best copy furnished by the organizational source. It is being released in the interest of making available as much information as possible.

- This document may contain data, which exceeds the sheet parameters. It was furnished in this condition by the organizational source and is the best copy available.

- This document may contain tone-on-tone or color graphs, charts and/or pictures, which have been reproduced in black and white.

- This document is paginated as submitted by the original source.

- Portions of this document are not fully legible due to the historical nature of some of the material. However, it is the best reproduction available from the original submission.

adams associates

RESEARCH FOR THE DEVELOPMENT OF SOFTWARE TECHNIQUES FOR COMPUTER-DRIVEN DISPLAYS

By Edward N. Chase, Frank S. Greatorex, Robert M. Metcalfe and James H. Mittica

November 1968

Prepared under Contract No. NAS 12-601 by
ADAMS ASSOCIATES
A Division of Keydata & Adams Associates Incorporated
108 Water Street, Watertown, Massachusetts 02172

Electronics Research Center

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

CONTENTS

Appendixes

adams associates

Mr. David Kipping
Technical Monitor
NAS 12-601
Electronics Research Center
575 Technology Square
Cambridge, Massachusetts 02139

Requests for copies of this report
should be referred to:

NASA Scientific and Technical
Information Facility
Post Office Box 33
College Park, Maryland 20740

adams associates

RESEARCH FOR THE DEVELOPMENT OF SOFTWARE TECHNIQUES FOR COMPUTER-DRIVEN DISPLAYS

By Edward N. Chase, Frank S. Greatorex, Robert M. Metcalfe and James H. Mittica

ADAMS ASSOCIATES
A Division of Keydata & Adams Associates Incorporated
108 Water Street, Watertown, Massachusetts 02172

I. SUMMARY

The direct use of input data formats for interfacing hardware configurations is inconvenient for the user to program in FORTRAN. The Display Transformation Routines therefore provide a direct software interface of the source data format to the exact display commands, similar to the functions of an assembler.

In addition, the implementation of two-dimensional rotation; scaling and limited scissoring of points, lines and character origins; and the representation of three-dimensional information, including display of "wire frame" points, lines and character origins; and three-space to two-space projections, provide certain pseudo-hardware functions that are not otherwise readily available.

II. INTRODUCTION

Transformation of graphical data structures to display lists suitable for direct use by a display.

III. DISPLAY TRANSFORMATION ROUTINES

A. General

It is assumed that the reader is generally familiar with the display subsystem described in Section IV.B of the Graphics Reference Manual, especially with subsections 1 through 4.

The routines described in IV.B.3 allow a user's program to communicate with and control the display. However, all data placed in the display buffer must be in the formats described in IV.B.4. While convenient and efficient for the display hardware, such formats are inconvenient for the user to program, especially in FORTRAN. Therefore, these display transformation routines are provided to transform from a tractable source data format to the exact display commands, similar to the function of an assembler. The source format is convenient for FORTRAN data manipulation and organizes data as it is generally used without forfeiting ultimate control of any display functions.

The transformation routines described transform and move data from the user segment to either the display buffer or to another section of the user segment. In addition to format transformation, certain pseudo-hardware functions (such as two-dimensional transformation) are performed. If an error is detected, an error code is returned. If XFMSER and XFMGEV are not used, error codes are typed on the teletype as errors are detected.

B. Input Data Formats

The translation routines accept data in a number of formats: global display parameters, poly-items (i.e., multiple vectors, characters, etc.), mixed format or absolute display commands. The poly-items afford an efficient yet tractable coding of information while the mixed format provides more flexibility. The absolute format can be used when the most efficient coding is desired.

Global Parameters. For all parameters, if no value is specified (i.e., value = 0) no change is made to the current setting. If errors in values are detected, the default value is used.

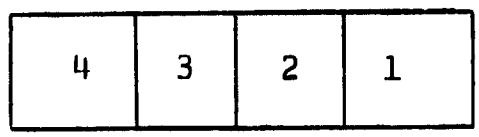
MODE = Not used

CHAN = Output channel control

The IDI (monochrome) display station is Channel 1. The color display uses Channel 2 (red), 3 (green) and 4 (blue). Each channel is controlled by one bit in the following format:

Bit = 0 Channel off

Bit = 1 Channel on



The parameter value is the decimal equivalent of the bit pattern. Since the value 0 is reserved to mean "no change," it is not possible to turn off all channels.

BRIT - Brightness

0 = No change

1 = Off

2 = Dim

3 = Bright (default)

4 = Extra bright

5 = Increase brightness

STRUCT - Line Structure

0 = No change

1 = Solid (default)

2 = Dotted

3 = Dashed

4 = Dot-dash

CSIZE - Character Size

0 = No change

1 = Small

2 = Medium (default)

3 = Large

4 = Extra large

5 = Increase character size

6 = Decrease character size

CROT - Character Rotation

0 = No change

1 = Horizontal (default)

2 = Vertical

LPVIZ - Light Pen Visibility

0 = No change

1 = Invisible (default)

2 = Visible

10	
MODE	CHAN
BRIT	STRUCT
CSIZE	CROT
LPVIZ	BLINK

BLINK - Blink

0 = No change

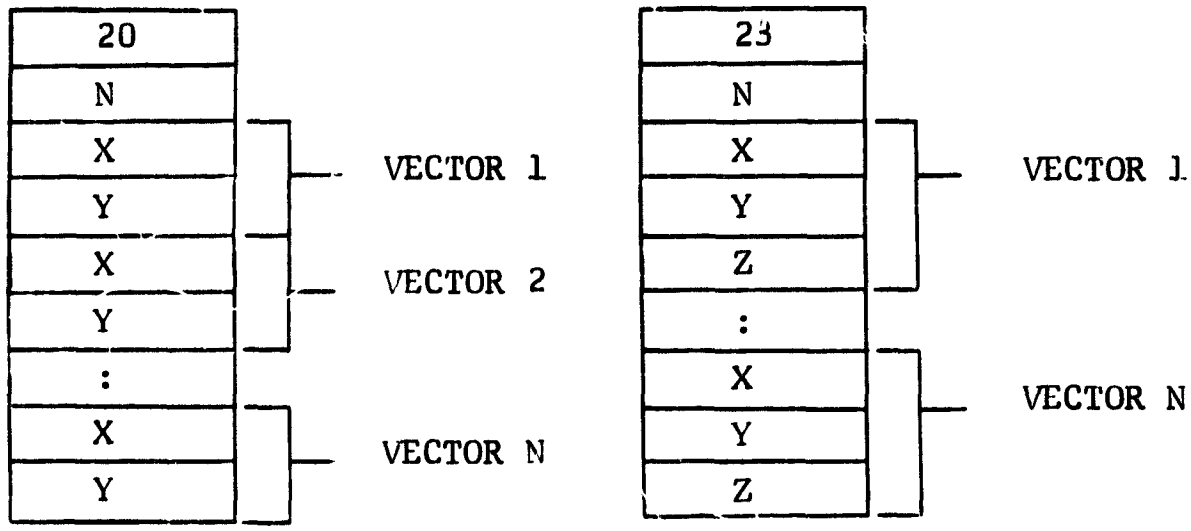
1 = No blink (default)

2 = Blink

Poly-Vector. Vectors are drawn from the present beam position to the given end point. This results in a sequence of N connected or "chained" vectors.

N = Number of vectors
X = X value of end point
Y = Y value of end point

N = Number of vectors
X = X value of end points
Y = Y value of end points
Z = Z value of end points

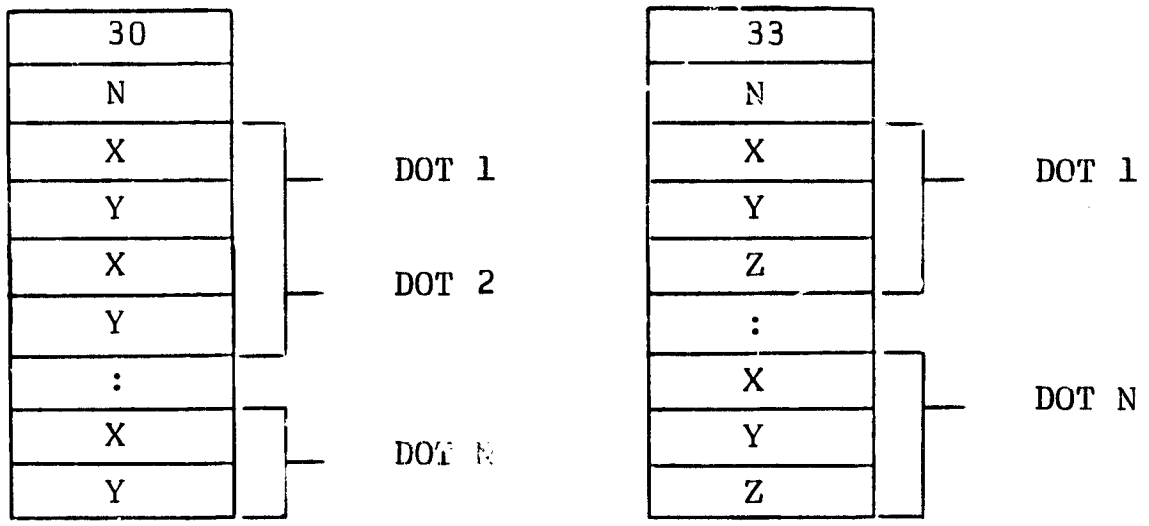


The range of X and Y after translation should be 0 to 1023. If a value is outside this range, the out-of-range edges are forced to the nearest edge (0 to 1023).

Poly-Dot. Dots are plotted from the present beam position to the given end point.

N = Number of dots
X = X value of dot coordinates
Y = Y value of dot coordinates

N = Number of dots
X = X value of dot coordinates
Y = Y value of dot coordinates
Z = Z value of dot coordinates



If either component distance between dots is from -3 to +3, incremental dots are drawn. If a value is outside this range, it is tested for range 0-1023.

In drawing incremental dots, if the number of dots drawn is odd, the last dot drawn will be somewhat brighter.

Poly-Circle. Circles are drawn using the radius from the X, Y, Z coordinate point.

N = Number of circles

X = X coordinate of center

Y = Y coordinate of center

RADIUS = Radius magnitude of circle

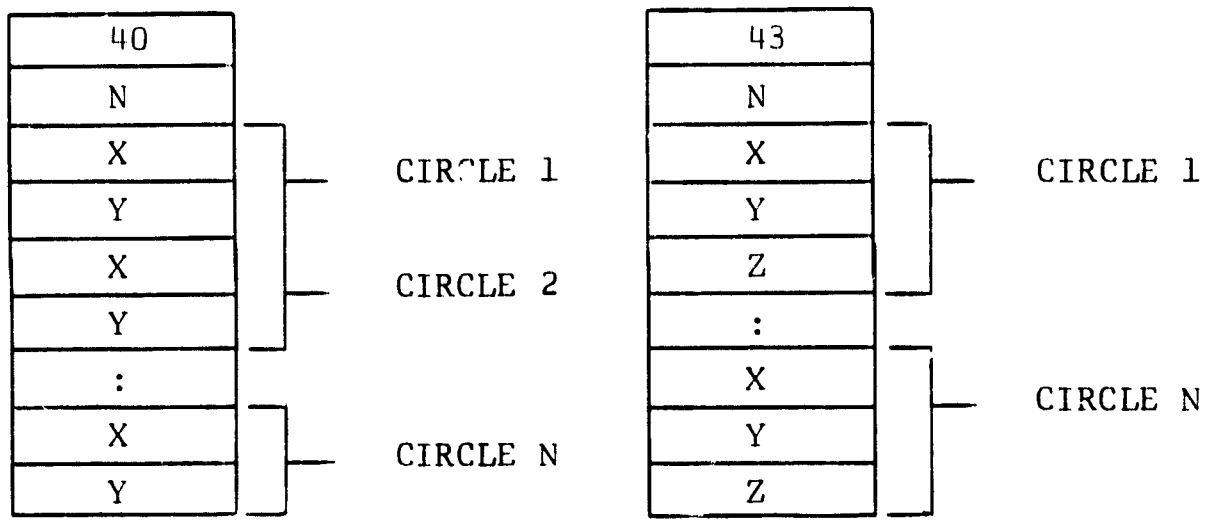
N = Number of circles

X = X coordinate of center

Y = Y coordinate of center

Z = Z coordinate of center

RADIUS = Radius magnitude of circle



The range of X and Y after translation is 0 to 1023 and the range of RADIUS is 0 to 511. If a value is outside its range, the out-of-range edges are forced to the nearest edge (0 to 1023).

Poly Character. Characters are drawn from the present beam position.

N = Number of characters (odd or even).

C_1, C_2 = Characters in system ASCII code. Characters are converted to the codes and functions given in Table IV.B.4.05 of the Graphics Reference Manual.

50	
N	C1
C2	C3
C4	--
--	--
:	:
--	--
--	Cn

Mixed Format. This format permits execution of a variety of display operations.

N = Number of entries.

Code = Function code.

Data = Value of data relating to the function. Meanings, ranges and error conditions are the same as for poly-items above.

100
N
Code
Data
Code
Data
:
Code
Data

CODE	FUNCTION	DATA
110	Define Beam X	Value of X
111	Set Beam Y	Value of Y
113	Define Beam Z	Value of Z
121	Draw Vector Y	Y end point
130	Draw circle	Radius of circle
140	Draw 2 char.	Two 8-bit system ASCII characters
150	No operation	Number of display commands NOP's
160	Chain	Pointer to new entity

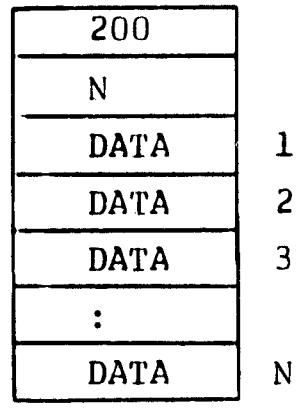
Note that code 150 will generate the specified number of NOP's. Code 160 permits processing non-contiguous global parameters or poly-items (but not mixed format) with a return to the next entry.

Absolute Format. This format can be used when the most efficient coding is desired.

N = Number of words of data.

DATA = Data in display format.

No translation or error checking takes place.



C. Program Description

From User to Display Subroutine

CALL XFMU2D (XBLOK,FROM,TO,NUMWDS). XFMU2D transforms and moves data from the user segment to the display buffer.

XBLOK = The name of a Transform Block, an array of 17 words.

FROM = The name of an array (of variable length) in the user segment to be transformed and moved to the display buffer.

TO = The name of a one-word subscript value; defines the first word in the display buffer (relative to a buffer origin of 1) to be filled. If TO = 0, data is appended to the present contents of the buffer.

NUMWDS = A one-word returned parameter indicating the number of words, in object format transferred to the display buffer.

From User to User Subroutine

CALL XFMU2U (XBLOK,FROM,TO,SPACE,NUMWDS). XFMU2U transforms and moves data from the user segment to another section of the user segment.

XBLOK = The name of a Transform Block, an array of 17 words.

FROM = The name of an array (of variable length) in the user segment to be transformed and moved.

TO = The name of an array in the user segment to be filled. (The array is of SPACE length.)

SPACE = A word defining the number of words available for transformed data in the user segment; i.e., length of the TO block.

NUMWDS = A one-word returned parameter indicating the number of words, in object format, transformed to the user segment.

Mainline Subroutine

CALL MAIN (XBLOK,WHERE,TO,FROM,SPACE,NUMWDS). MAIN transforms and moves data from the user segment to either the display buffer or to another section of the user segment. This is accomplished by stripping down each input entity table and performing the various positioning calls to either internal subroutines or subroutines in the Display Utility Package.

XBLOK = Name of Transform Block, an array of 17 words.

WHERE = Defines where data from all subsequent utility calls is to be placed. (A one-word parameter.)

WHERE = 1 Append to display buffer.

2 Insert in middle of display buffer.

3 Append to utility buffer (internal buffer of maximum length 200 words).

4 Insert in utility buffer.

TO = The name of an array or a one-word subscript value, depending on the type of call. (XFMU2D or XFMU2U)

FROM = The first address in the user segment to be transformed. (An array of unknown length.)

SPACE = A word defining the number of words available for transformed data in the user segment.

NUMWDS = A one-word returned parameter indicating the number of words in object format transformed to the user segment.

Entity Strip Subroutine

CALL ESTRIP (KEYLOC,KEYTYP,CURKEY,LENGTH). ESTRIP computes the entity length using the keyword type and N.

KEYLOC = The first address in the user segment to be transformed. (An array of unknown length - replaces FROM.)

KEYTYP = The current entity type. (A returned parameter.)

CURKEY = Not used at this time.

LENGTH = The length of the current entity table. (A returned parameter.)

10	Global parameters	Length = 5
20	Poly Vector - 2D	Length = 2*N+2
23	Poly Vector - 3D	Length = 3*N+2
30	Poly Dot - 2D	Length = 2*N+2
33	Poly Dot - 3D	Length = 3*N+2
40	Poly Circle - 2D	Length = 3*N+2
43	Poly Sphere - 3D	Length = 4*N+2
50	Poly Character	Length = N/2+1
100	Mixed Mode	Length = 2*N+2
200	Absolute Format	Length = N+2

Node Strip Subroutine

CALL NSTRIP (KEYLOC,XCOORD,YCOORD). NSTRIP strips the X, Y and Z nodes from the entity table.

KEYLOC = The first address in the user segment to be transformed.

XCOORD = The X-coordinate in the entity table. (A returned parameter.)

YCOORD = The Y-coordinate in the entity table. (A returned parameter.)

IZ3D = Although this is not in the calling arguments, the Z-coordinate is returned through COMMON.

Transform Routine

CALL TRFORM (XCOORD,YCOORD). TRFORM performs the two-dimensional rotation, scaling, the representation of three-dimensional information, including the display of "wire frame" points, lines and character origins, and three-space to two-space projection.

XCOORD = The X-coordinate in the entity table.

YCOORD = The Y-coordinate in the entity table.

Strip Word Subroutine

CALL STRPWD (KEYLOC,RESULT). STRPWD strips each word in the entity table, one by one, using STPLOC in COMMON as the indexing parameter.

KEYLOC = The first address in the user segment to be transformed.

RESULT = A returned parameter of one entity word.

Move Buffer Subroutine

CALL MOVBUF (WHERE,TO,WRDLEN,SPACE,MBFERR,NUMWDS). MOVBUF checks to see which buffer is being used (the display or utility). If the utility buffer is used, it is moved back to the user when transformation is completed or the buffer is full. In effect, if WHERE is equal to 3 or 4.

WHERE = Defines where data from all subsequent utility calls is to be placed.

WHERE = 1 Append to display buffer.

2 Insert in middle of display buffer.

3 Append to utility buffer (internal buffer of maximum length 200 words).

4 Insert in utility buffer.

TO = The name of an array in the user segment to be filled. (The array is of SPACE length.)

WRDLEN = The number of object words being appended to or inserted in the utility buffer.

SPACE = A word defining the number of words available for transformed data in the user segment.

MBFERR = A one-word returned parameter indicating the overflow condition of the user buffer.

NUMWDS = A one-word returned parameter indicating the number of words, in object format, transformed to the user segment.

Error Return Address Subroutine

CALL XFMSER (N). XFMSER allows the user to set an address to which control is returned following all display transformation errors. The call should be preceded by an ASSIGN statement.

N = The statement number to which control will be returned.

Get Error Value Subroutine

CALL XFMGEV (N). XFMGEV returns the code of the most recent transformation error.

N = Error code (a returned parameter).

The following is a list of the error code conditions resulting from calls to XFMU2D or XFMU2U:

XFMU2D ERROR CODES

1 = No error.

2 = No console assigned to user. No action taken.

3 = TO is out-of-range; negative or larger than present buffer. No action taken.

4 = Length of the array to be transformed is out-of-range; N in data format negative or length greater than 6000. No action taken.

5 = Zero scale word or view area edges exceed permissible scope coordinates. Out-of-range edges are forced to the nearest edge (0 or 1023); for zero scale, nothing is drawn.

6 = Illegal keyword found.

7 = Mixed format recursion.

XFMU2U ERROR CODES

1 = No error.

2 = TO is out-of-range of user's memory bound. No action taken.

3 = SPACE filled before transformation completed. Remaining source data is not transformed.

4 = Length of the array to be transformed is out-of-range; N in data format negative or length greater than 6000. No action taken.

5 = Zero scale word or view area exceed permissible scope coordinates. Out-of-range edges are forced to the nearest edge (0 or 1023); for zero scale, nothing is drawn.

6 = Illegal keyword is found.

7 = Mixed format recursion.

Line Scissor Routine

CALL SCISOR (IX,IY,IF). SCISOR calculates what the image of a vector drawn from the current beam position to the specified vector head will be on the viewing frame. It then returns a plot or no-plot flag and appropriately altered X,Y for a modified vector plot. The actual internal beam position is automatically updated. Five cases are considered and dealt with in the following manner:

Case 1. The vector is found to be entirely within the viewing frame. The X,Y values are left unchanged and the plot flag is set to 1.

Case 2. The vector is found to be entirely outside the viewing frame. The no-plot signal is set to 2.

Case 3. The vector is found to be entering the viewing frame. The beam is then reset to the frame entry point and the plot flag is set to 1. The X,Y values are left unchanged.

Case 4. The vector is found to be leaving the viewing frame. The plot flag is set to 1 and the X,Y values are set to the exit point.

Case 5. The vector is found to cross over the viewing frame. The beam is then reset to an entry point and the X,Y values are changed to be an exit point. The plot flag is set to 1.

IX = The X-coordinate input value.

IY = The Y-coordinate input value.

IF = The plot or no-plot flag. (A returned parameter.)

Extended Boundary Intercept Calculator and Positioner

CALL INCAL. INCAL calculates the X and Y positions of the intercepts involving the proposed beam vector. Using the extended frame boundaries, INCAL evaluates the position of these intercepts with respect to the frame. The frame boundaries on the vector foot, vector head, intercept second-coordinates and intercept positions all reside in COMMON.

Frame-Boundary-Extension Intercept Scanner

CALL INSCN (IX,IY,IF). INSCN scans the list of four frame-boundary-extension intercepts, searching for the first one that is on or within the viewing frame defined by the proposed beam vector.

If an intercept qualifies, the output flag is set to 1 and its coordinates are returned in the X,Y arguments passed. If no qualifying intercept is found, the values are returned unchanged.

IX = The X-coordinate input value.

IY = The Y-coordinate input value.

IF = Intercept flag, a returned parameter of value 1 or 2.

1 = Qualifying intercept

2 = Non-qualifying intercept

Reset Beam Position

CALL RESET (IX,IY,IF). RESET checks for in-frame beam reset positions and positions the beam legally if the request is made.

IX = The X-coordinate input value.

IY = The Y-coordinate input value.

IF = The input and output flag used in the following manner:

Input flag = 1 - Reset beam if legal reset position.

Input flag = 2 - Do not reset beam in any case.

Output flag = 1 - Transmitted beam position is legal.

Output flag = 2 - Illegal beam position transmitted.

Double Precision Integer Multiply Function

FUNCTION IMUL (IMULI). IMUL is a double-precision integer function. The multiplicand is in the integer word IAC located in COMMON, and the multiplier is passed as the argument to this function. The most significant left-half of the product is returned in IAC and as the function value, while the least significant right half of the product is left in IMQ. This routine takes advantage of the 1108 36-bit word length for 32-bit accuracy, and must be changed for 516 implementation.

IMULI = Multiplier.

Double-Precision Integer Divide Function

FUNCTION IDIV (IDIVI). IDIV is a double-precision integer divide function. The dividend is in the integer words IAC-IMQ located in COMMON, and the divisor is passed as the argument to this function. The quotient is returned as the function value and in IAC, while the remainder is left in IMQ. This routine takes advantage of the 1108 36-bit word length for 32-bit accuracy, and must be changed for 516 implementation.

IDIVI = Divisor.

In Frame Test Function

FUNCTION IFRME (IX,IY). IFRME computes the position of the specified point relative to the viewing frame specified in COMMON and returns this position as a zone code from 1 to 9.

IX = The X-coordinate input value.

IY = The Y-coordinate input value.

Vector Frame Test Function

FUNCTION IVFT (IX,IY). IVFT computes the position of a specified point relative to the rectangular frame generated by the coordinates of the head and tail of the proposed beam vector (IBPX,IBPY,IXN,IYN). If the point is in or on the frame, the function value is set to 1. If the point is outside the frame, the function value is set to 2.

IX = The X-coordinate input value.

IY = The Y-coordinate input value.

IV. ALGORITHM

A. General

The algorithm for rotation, scaling and three-space to two-space projection is centered around an input transform block of 17 words. The block consists of the following format:

IFLAGS = Flag used to signal a new transform block.

IDELTX = X-base value for transformation, data coordinates (signed integer).

IDELTY = Y-base value for transformation, data coordinates (signed integer).

IANGLE = Angle of rotation, expressed in multiples of .0001 radian; that is, +32767 = +3.2767 radians.

ISCALE = Multiplying or dividing factor as indicated by positive or negative numbers. For example: +200 is 200X; -3042 is $\frac{1}{3042}$ X.

IVXO = View origin X-coordinate.

IVYO = View origin Y-coordinate (VXO and VYO) specify the point in data coordinates which is at (511, 511) in scope coordinates.

IVXL = View left-edge X-coordinate (minimum 0).

IVXR = View right-edge X-coordinate (maximum 1023).

IVYB = View bottom-edge Y-coordinate (minimum 0).

IVYT = View top-edge Y-coordinate (maximum 1023). These four scope coordinates specify the area within which the transformed data is to appear. Scissoring to view occurs for points, lines, centers of circles and character origins.

ITX3D = View origin X'-coordinate.

ITY3D = View origin Y'-coordinate.

ITZ3D = View origin Z'-coordinate (ITX3D, ITY3D, and ITZ3D specify that point in user data coordinates relative to the original X, Y, Z viewing coordinates).

IGAMA = Primary angle rotation about X-axis, expressed in multiples of .0001 radian.

IBETA = Secondary angle, rotation about Y-axis.

IALPH = Tertiary angle, rotation about Z-axis. (The 3-D user coordinates are effected by Euler angle rotation and Z' made equal to zero for projection to 2-D view, X2D = X', Y2D = Y'.)

Transforming operations are performed in the following order:

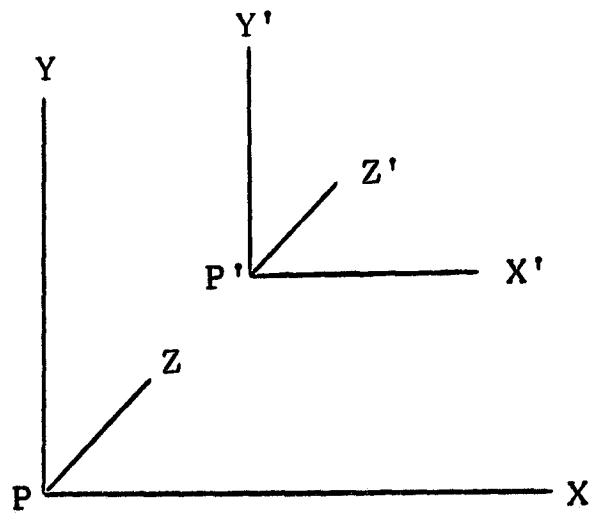
1. Three-space to two-space projection.
2. Rotation about (0,0).
3. Translation by ΔX, ΔY.
4. Scaling relative to (IVXO, IVYO), the resultant coordinate +511 giving scope coordinates.

B. Three Dimensional Algorithm

Translation of Coordinate Areas. Let X, Y, Z be the coordinates of any point P with respect to the right-handed rectangular cartesian reference system. Let X', Y', Z' be the coordinates of the same point P with respect to a second right-handed

rectangular cartesian reference system. If these axes have the same directions as the corresponding axes of the X, Y, Z system, and the origin has the coordinates $X = X_o$, $Y = Y_o$, $Z = Z_o$ in the X, Y, Z system, and also if equal scales are used to measure the coordinates in both systems, the coordinates X', Y', Z' are related to the coordinates X, Y, Z by the transformation equations:

$$X = X' + X_o$$
$$Y = Y' + Y_o$$
$$Z = Z' + Z_o$$



Rotation of Coordinate Axes. Given any point $P \equiv (X,Y,Z)$, let X', Y', Z' be the coordinates of the same point with respect to a second right-handed rectangular cartesian reference system having the same origin $\emptyset$ as the X,Y,Z system and oriented with respect to the latter so that:

The X-axis has the direction cosines IT_{11}, IT_{12}, IT_{13}.
The Y-axis has the direction cosines IT_{21}, IT_{22}, IT_{23}.
The Z-axis has the direction cosines IT_{31}, IT_{32}, IT_{33}.

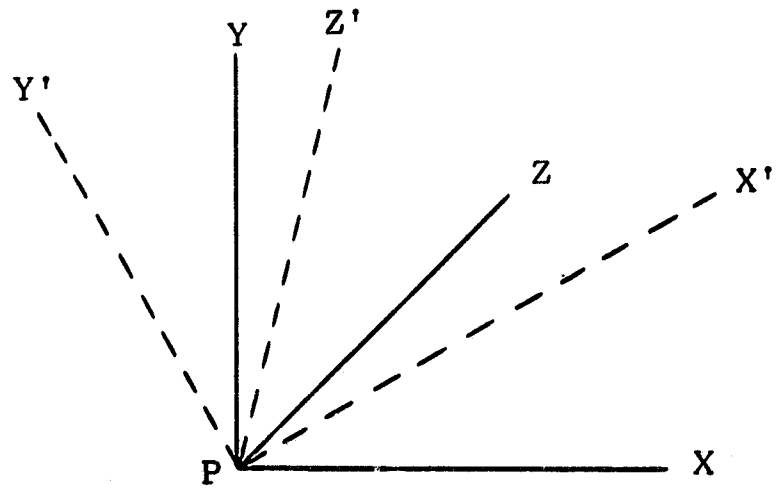
If equal scales are used to measure X,Y,Z, X', Y', and Z', the transformation equations relating the coordinates X', Y', Z', to the coordinates X,Y,Z are:

$$X = IT_{11}X'+IT_{12}Y'+IT_{13}Z'$$

$$Y = IT_{21}X'+IT_{22}Y'+IT_{23}Z'$$

$$Z = IT_{31}X'+IT_{32}Y'+IT_{33}Z'$$

It should be noted at this point that the transformations above are orthogonal projections.

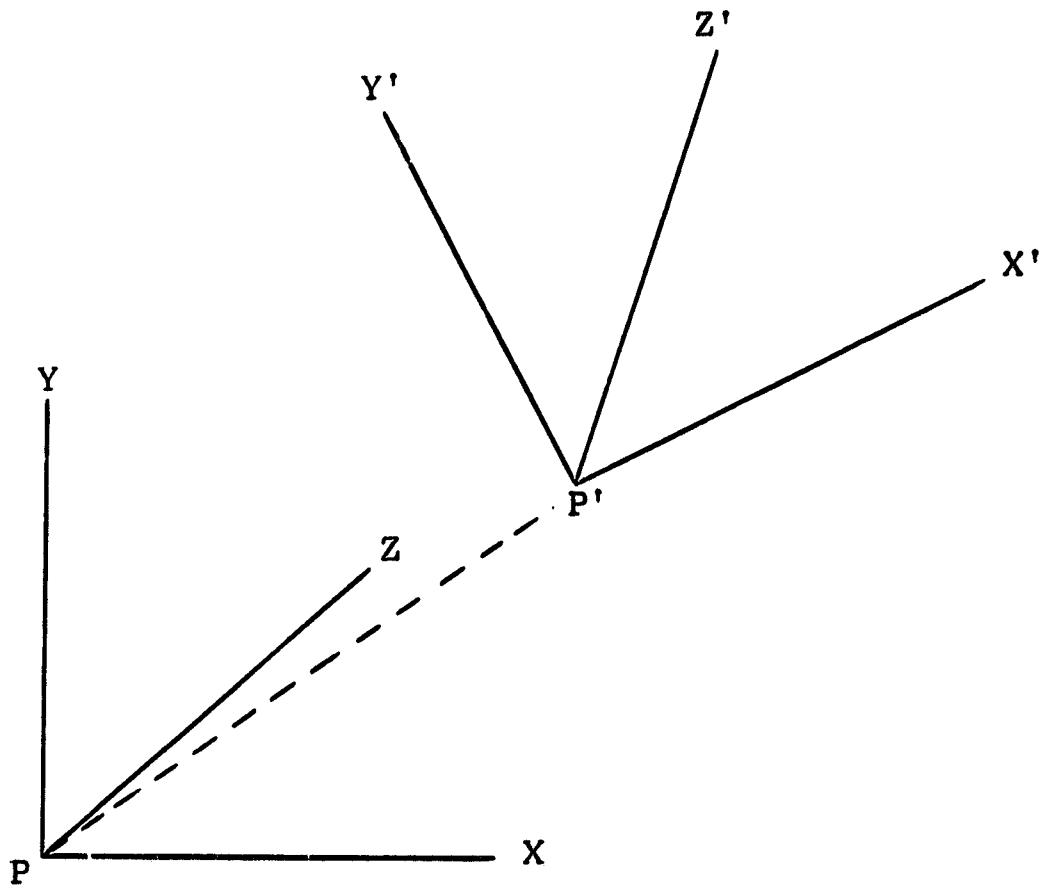


Simultaneous Translation and Rotation. If the origin of the X', Y', Z' system is not that of the X,Y,Z system but has the coordinates $X = X_o$, $Y = Y_o$, $Z = Z_o$ in the X,Y,Z system, the transformation equations become:

$$X = IT_{11}X'+IT_{12}Y'+IT_{13}Z'+X_o$$

$$Y = IT_{21}X'+IT_{22}Y'+IT_{23}Z'+Y_o$$

$$Z = IT_{31}X'+IT_{32}Y'+IT_{33}Z'+Z_o$$



Euler Angles. Given a matrix $A \equiv [a_{ik}]$ representing a proper rotation in three-dimensional Euclidean space, it can be variously expressed as a product of three matrices as follows:

$$IT_{11} = \cos \alpha \cos \beta \cos \gamma - \sin \alpha \sin \gamma$$

$$IT_{12} = -(\sin \alpha \cos \beta \cos \gamma + \cos \alpha \sin \gamma)$$

$$IT_{13} = \sin \beta \cos \gamma$$

$$IT_{21} = \cos \alpha \cos \beta \sin \gamma + \sin \alpha \cos \gamma$$

$$IT_{22} = -\sin \alpha \cos \beta \sin \gamma + \cos \alpha \cos \gamma$$

$$IT_{23} = \sin \beta \sin \gamma$$

$$IT_{31} = -\cos \alpha \sin \beta$$

$$IT_{32} = \sin \alpha \sin \beta$$

$$IT_{33} = \cos \beta$$

Geometical Interpretation of the Euler Angles. A set of cartesian X, Y, Z axes, initially aligned with X, Y, Z can be rotated into alignment with X', Y', Z' by three successive rotations:

1. Rotate about Z-axis through the Euler angle α.
2. Rotate about Y-axis through the Euler angle β.
3. Rotate about Z-axis through the Euler angle γ.

C. Two-Dimensional Algorithm

Rotation of the Coordinate Axes. Given X,Y are the coordinates of any point P with respect to a right-handed rectangular cartesian reference system. Let X', Y' be the coordinates of the same point P with respect to a second right-handed rectangular cartesian reference system having the same origin and rotated with respect to the X,Y system. If equal scales are used to measure all four coordinates X, Y, X', Y', the coordinates X', Y' are related to the coordinates X, Y by the transformation equations:

$$X' = X \cos \phi + Y \sin \phi$$

$$Y' = -X \sin \phi + Y \cos \phi$$

Simultaneous Translation and Rotation of Coordinate Area.

If the origin of the X', Y' system is not the same as the origin of the X, Y system but has the coordinates $X = X_o$ and $Y = Y_o$ in the X, Y system, the transformation equations become:

$$X = X' \cos \phi - Y' \sin \phi + X_o$$

$$Y = X' \sin \phi + Y' \cos \phi + Y_o$$

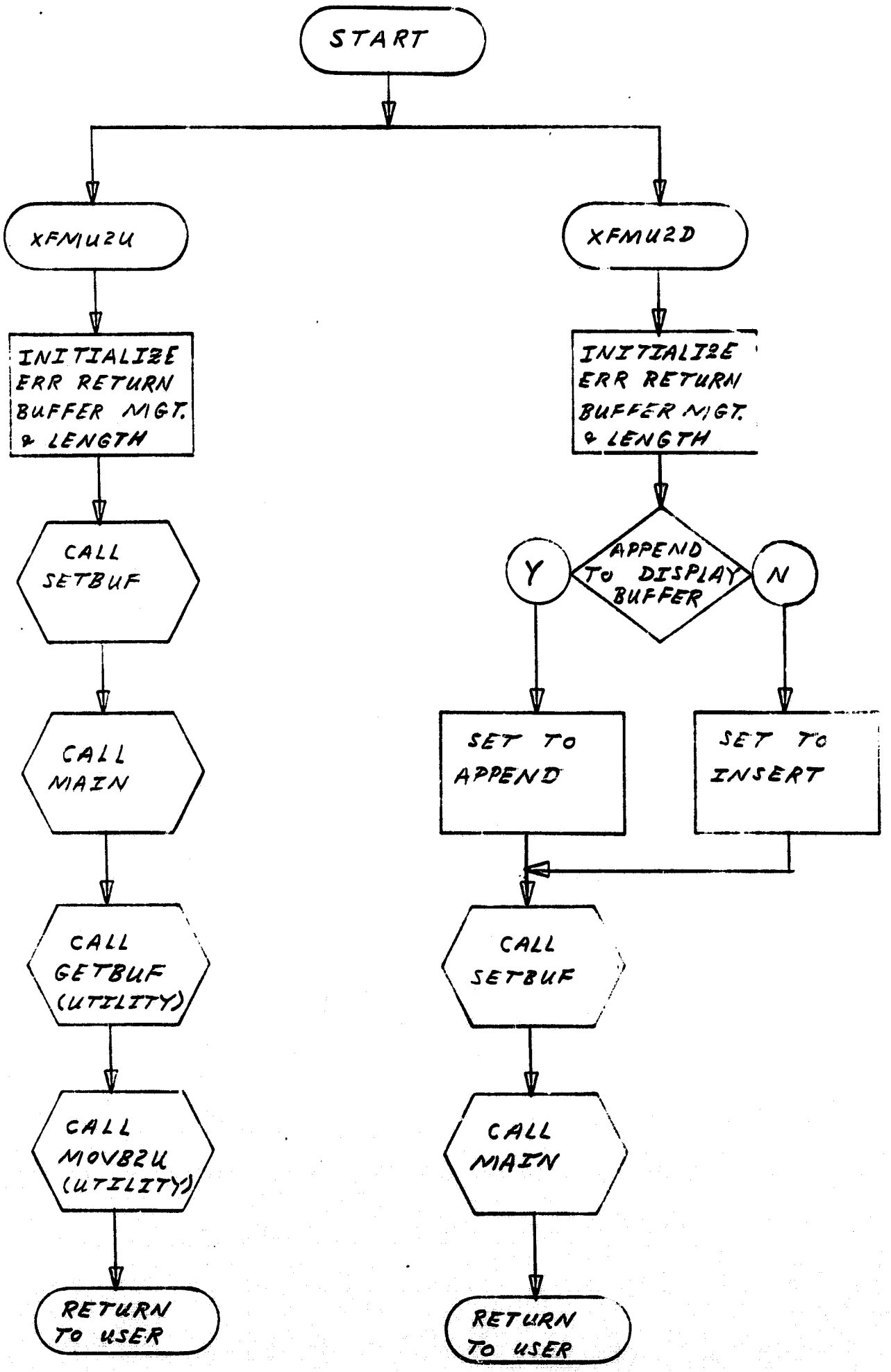
D. Reference

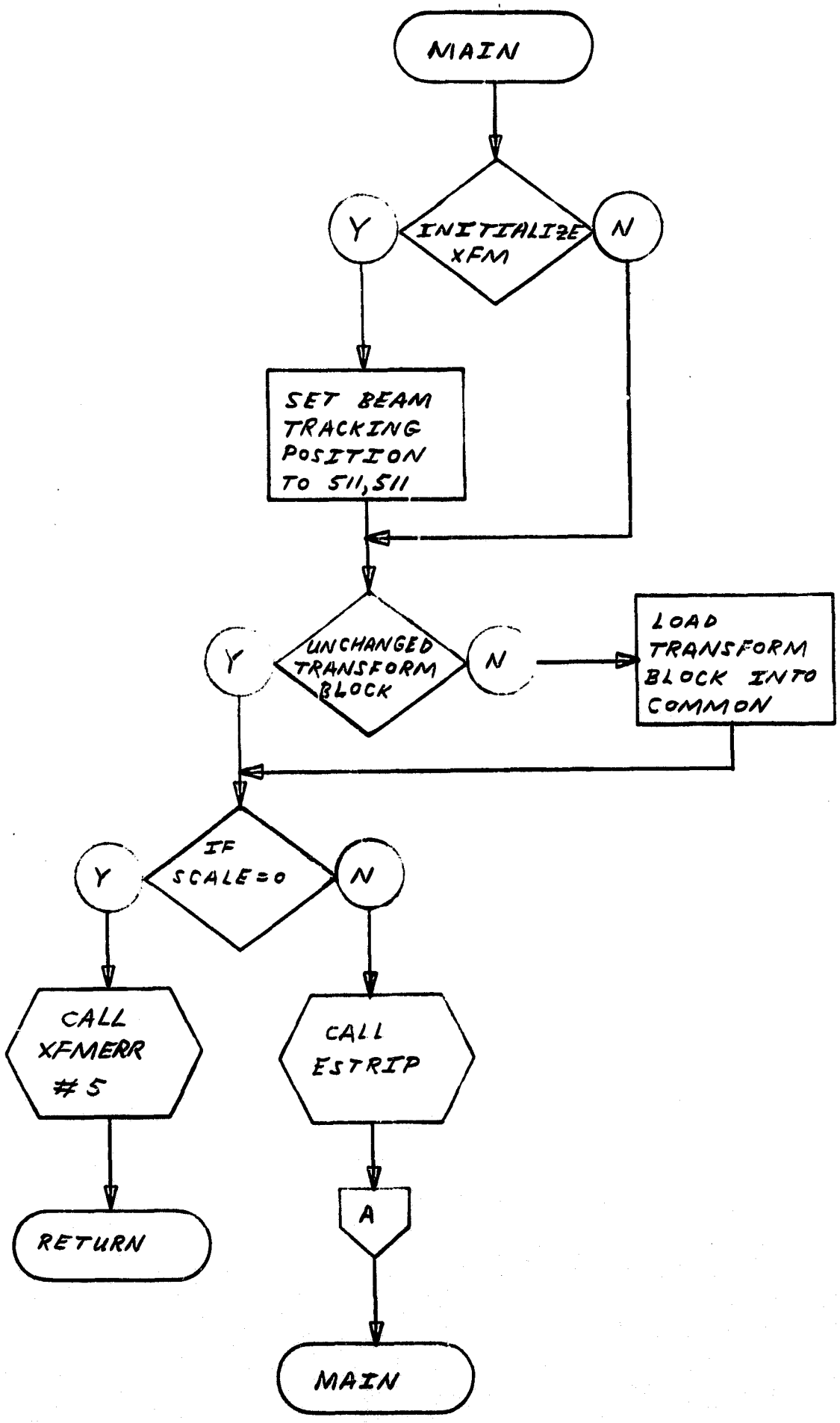
Korn, G. A., and Korn, T. M.: "Mathematical Handbook for Scientist and Engineers." McGraw-Hill Book Co., Inc., 1961.

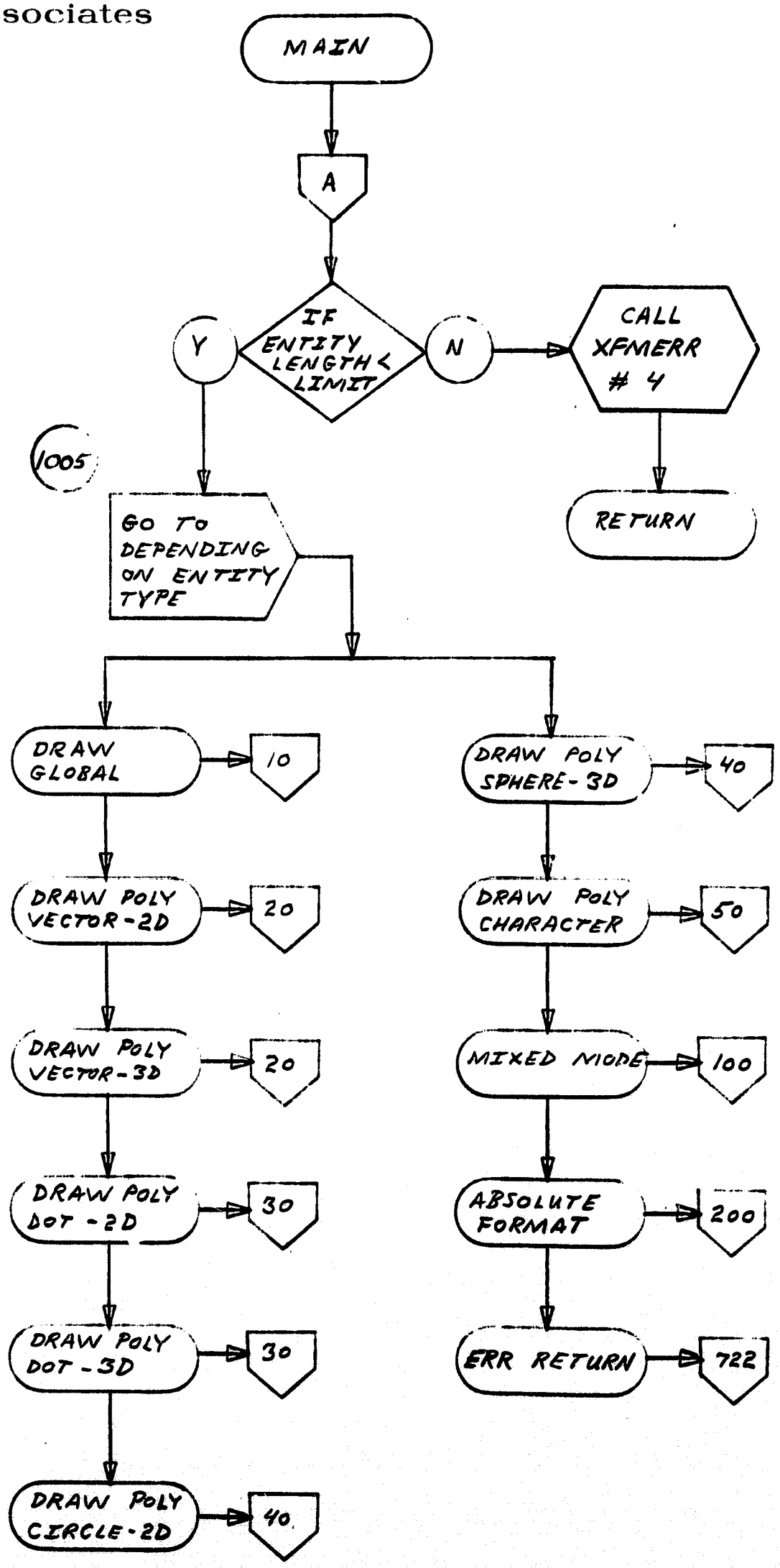
adams associates

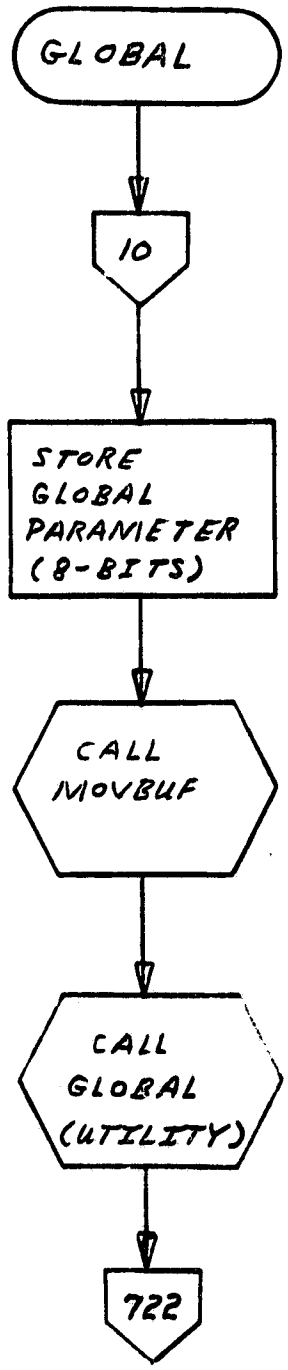
Appendix A

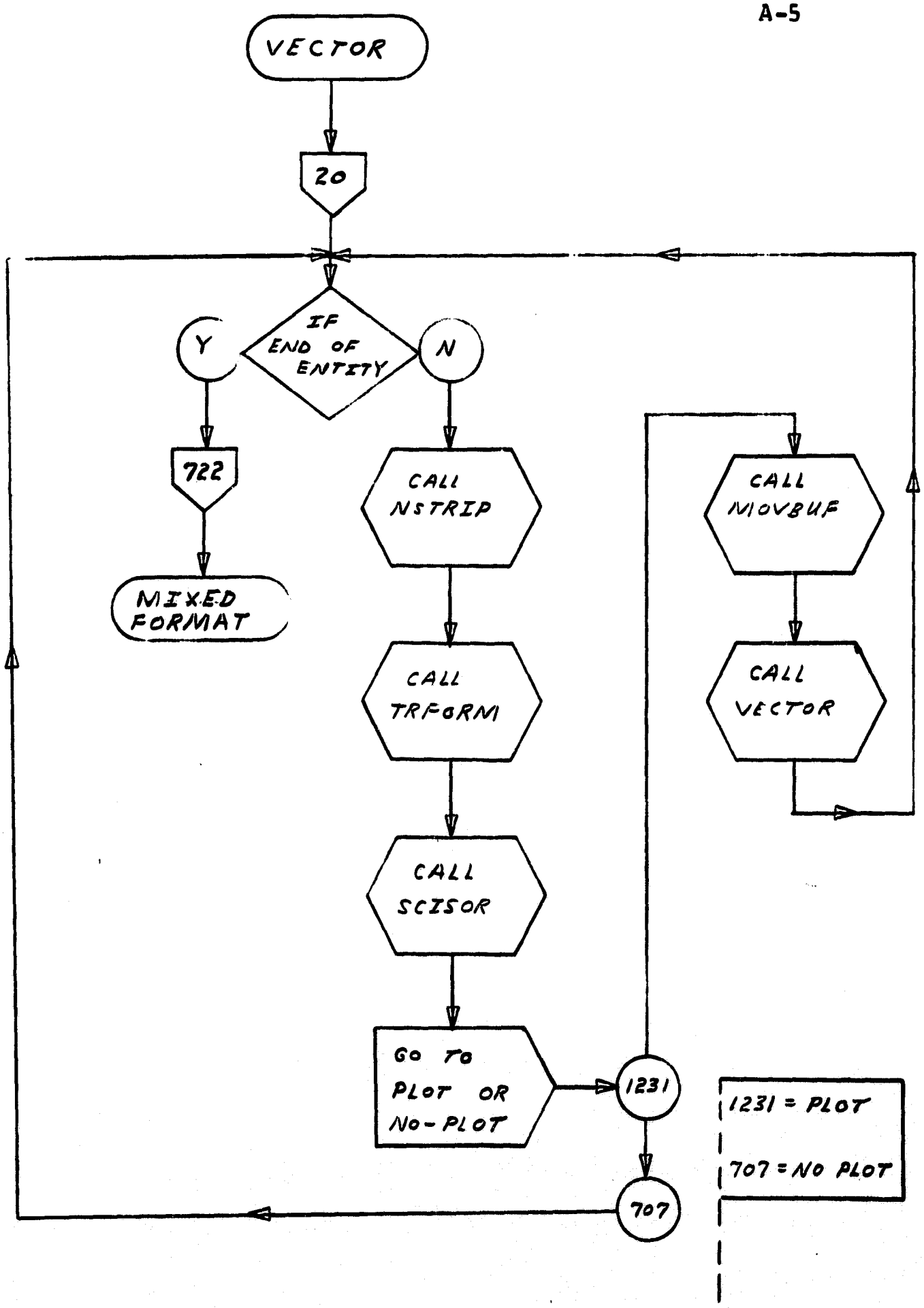
MAINLINE ROUTINE FLOWCHARTS

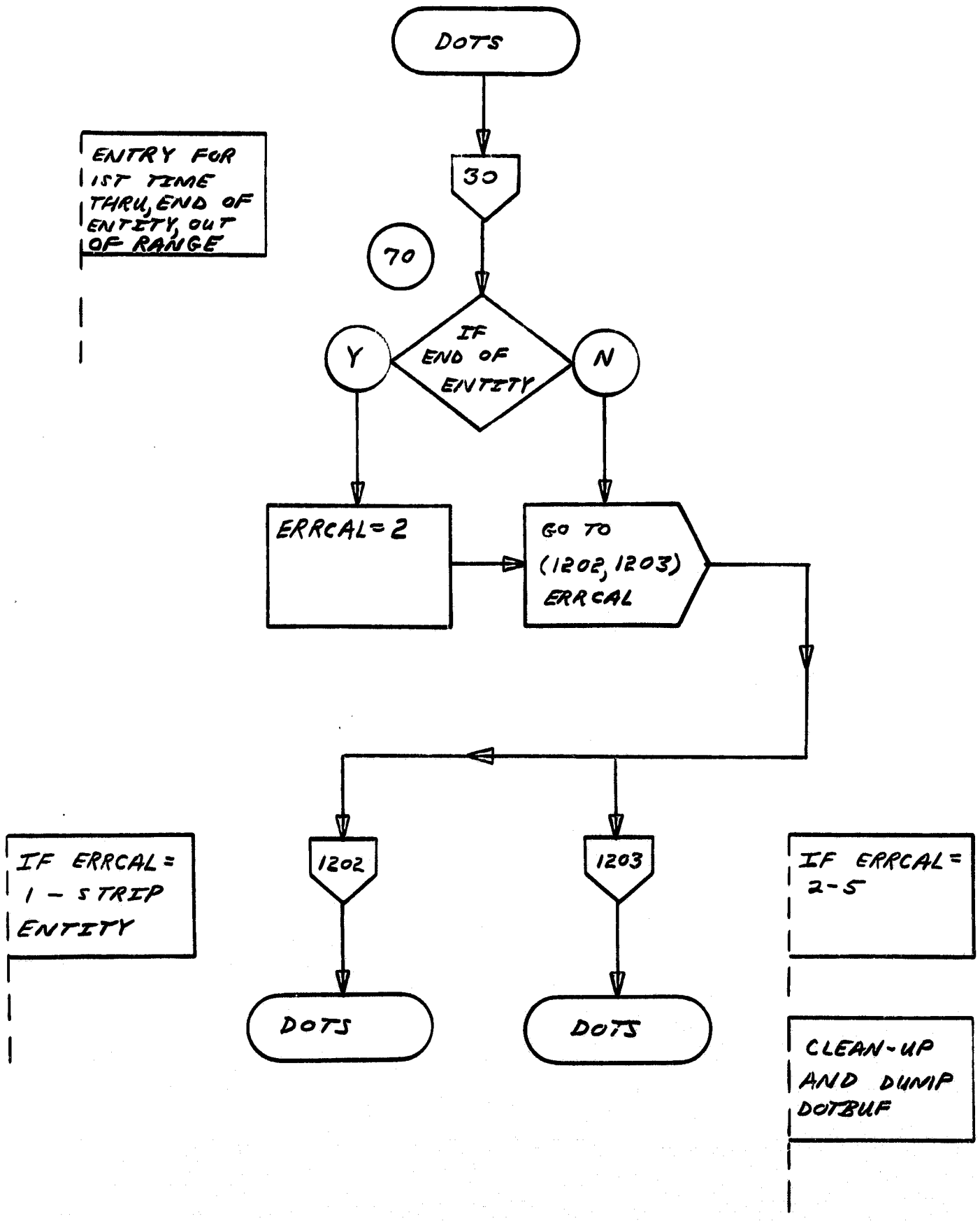

START
XFMU2U
XFMU2D
INITIALIZE ERR RETURN BUFFER MGT. & LENGTH
INITIALIZE ERR RETURN BUFFER MGT. & LENGTH
CALL SETBUF
APPEND TO DISPLAY BUFFER
Y
N
CALL MAIN
SET TO APPEND
SET TO INSERT
CALL GETBUF (UTILITY)
CALL SETBUF
CALL MOVB2U (UTILITY)
CALL MAIN
RETURN TO USER
RETURN TO USER

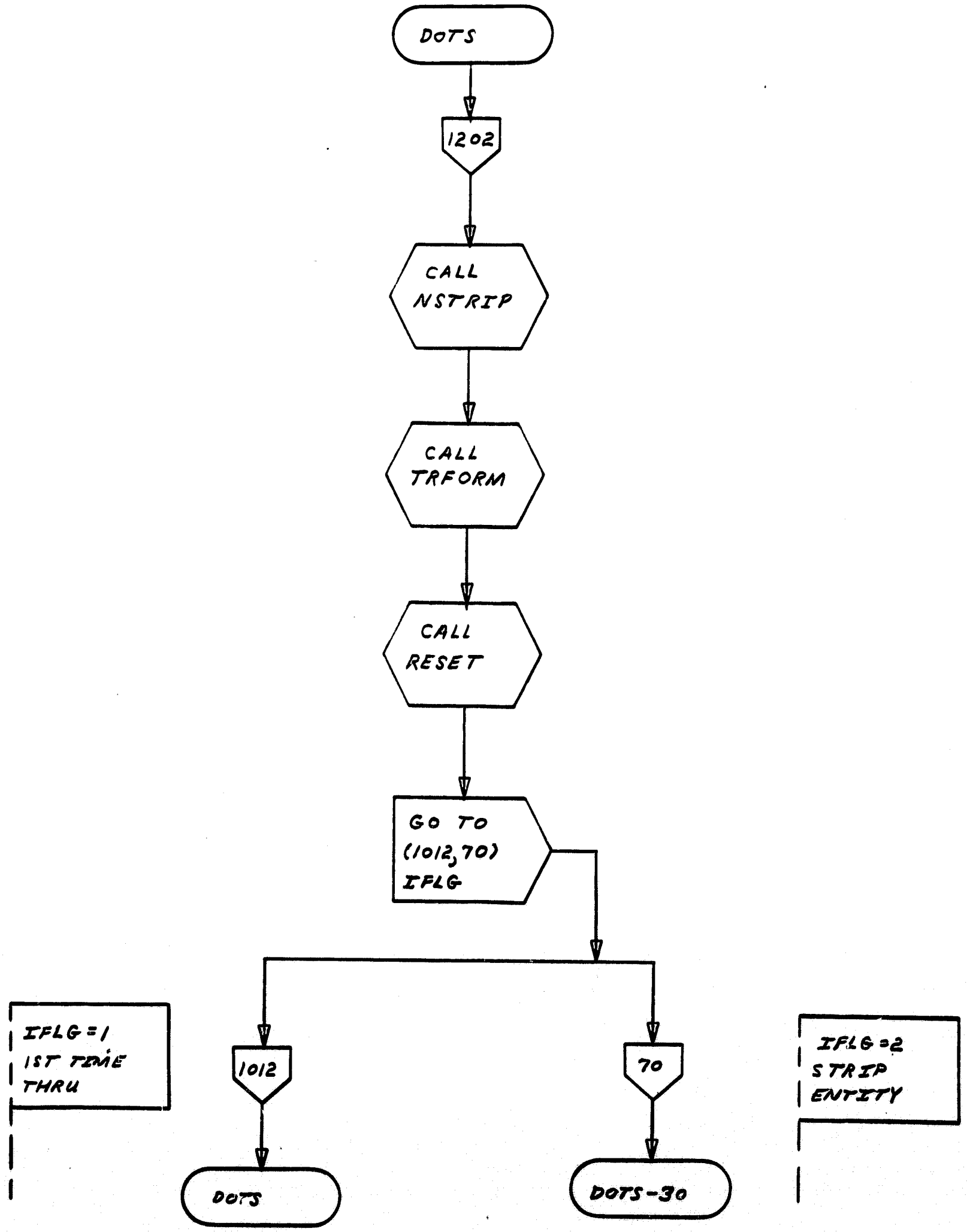

MAIN
INITIALIZE XFM
Y
N
SET BEAM TRACKING POSITION TO 511,511
UNCHANGED TRANSFORM BLOCK
Y
N
LOAD TRANSFORM BLOCK INTO COMMON
IF SCALE=0
Y
N
CALL XFMERR #5
CALL ESTRIP
RETURN
A
MAIN

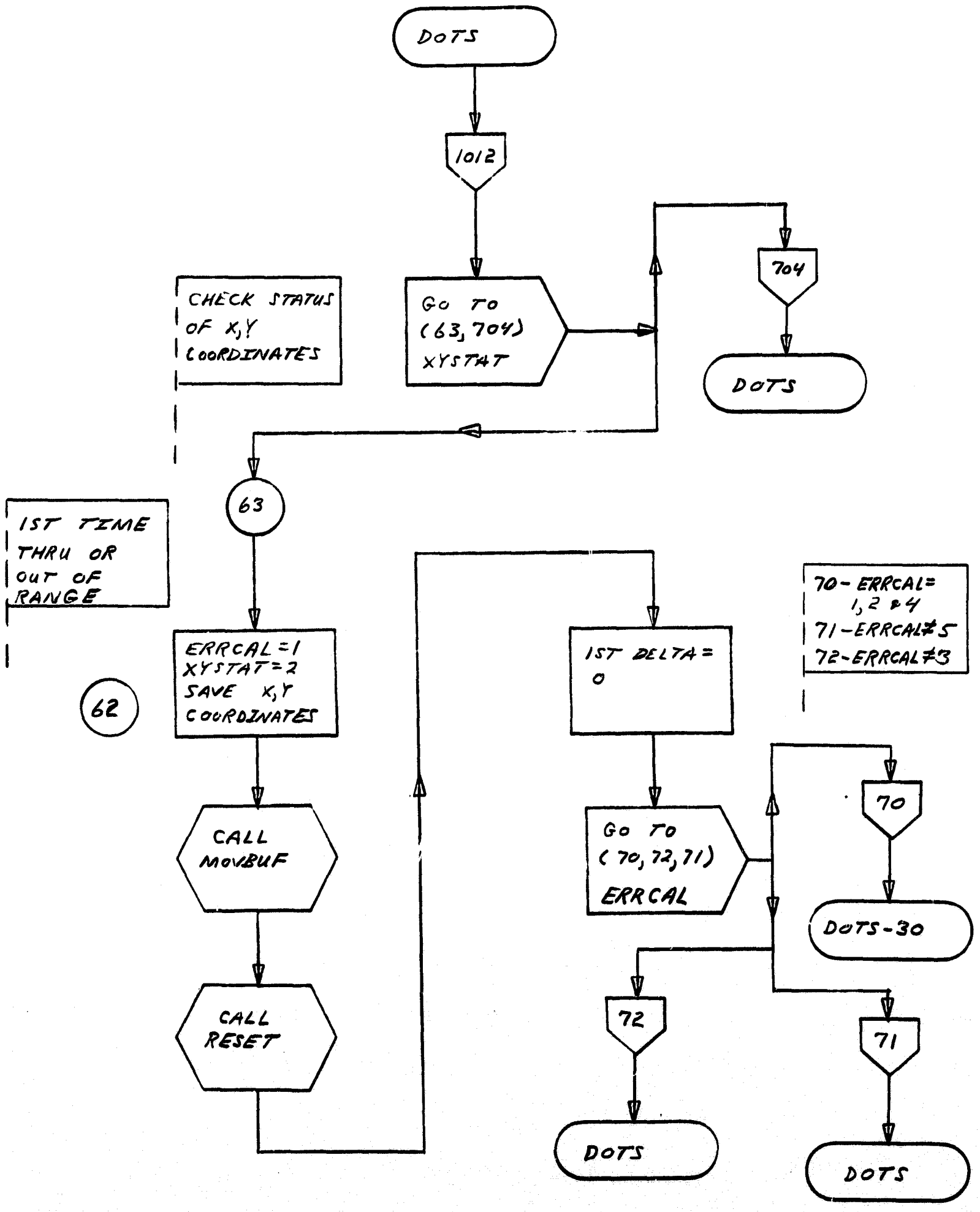

MAIN
A
IF ENTITY LENGTH < LIMIT
Y
N
CALL XFMERR # 4
RETURN
1005
GO TO DEPENDING ON ENTITY TYPE
DRAW GLOBAL
10
DRAW POLY VECTOR-2D
20
DRAW POLY VECTOR-3D
20
DRAW POLY DOT-2D
30
DRAW POLY DOT-3D
30
DRAW POLY CIRCLE-2D
40
DRAW POLY SPHERE-3D
40
DRAW POLY CHARACTER
50
MIXED MODE
100
ABSOLUTE FORMAT
200
ERR RETURN
722

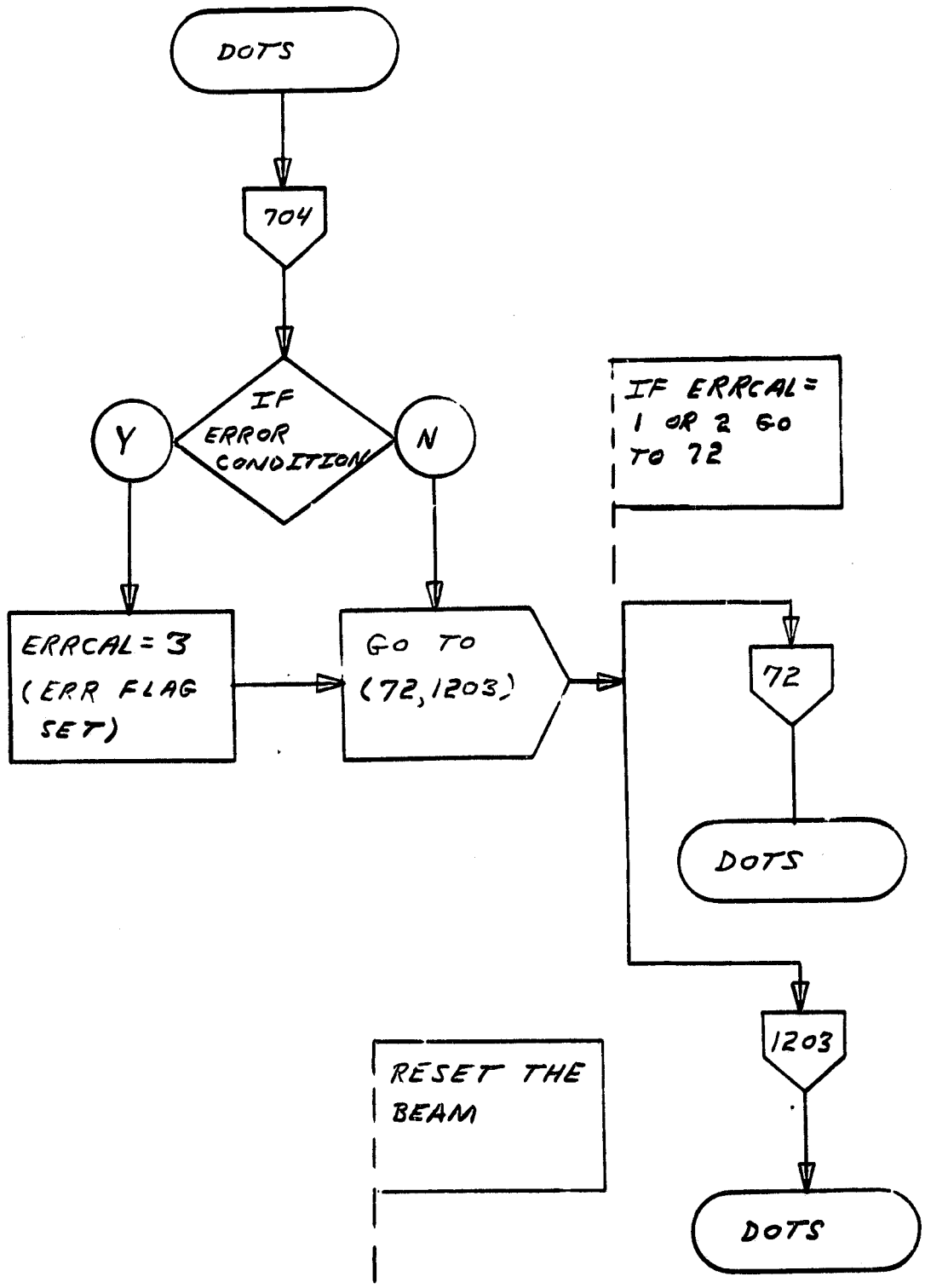

GLOBAL
10
STORE
GLOBAL
PARAMETER
(8-BITS)
CALL
MOVBUF
CALL
GLOBAL
(UTILITY)
722

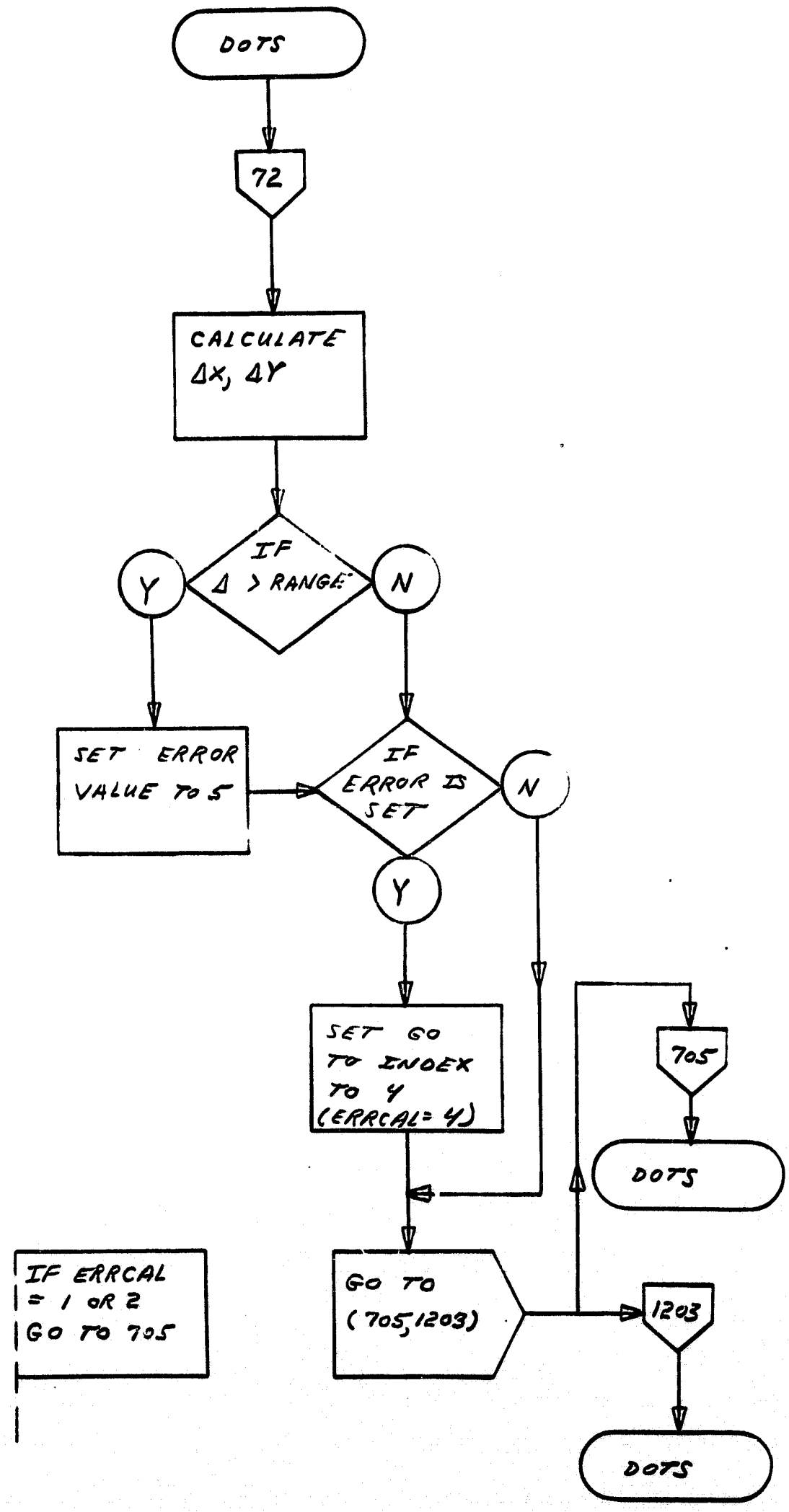

VECTOR
20
IF END OF ENTITY
Y
N
722
MIXED FORMAT
CALL NSTRIP
CALL TRFORM
CALL SCISOR
GO TO PLOT OR NO-PLOT
1231
707
CALL MOVBUF
CALL VECTOR
1231 = PLOT
707 = NO PLOT

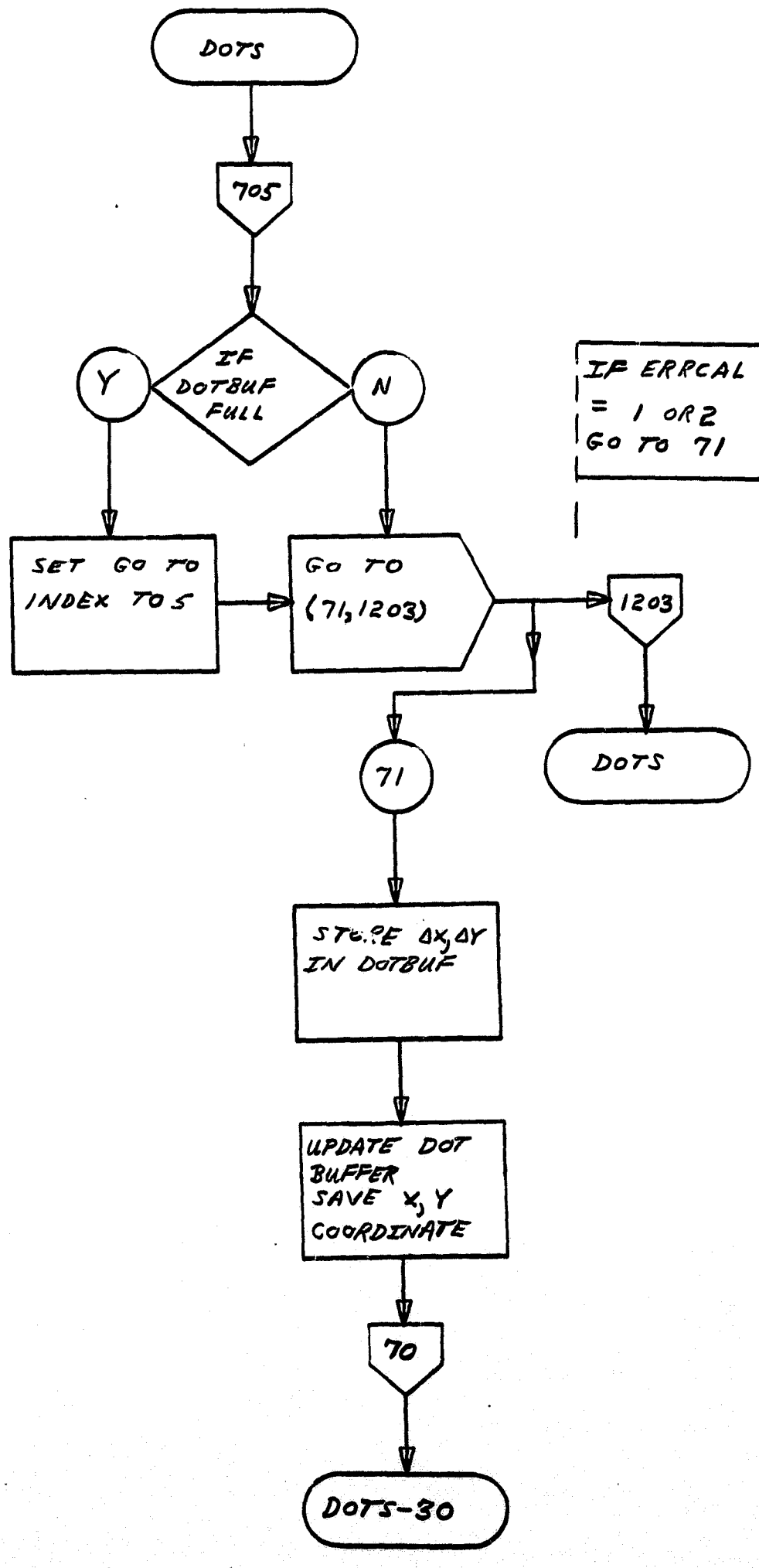

DOTS
30
70
ENTRY FOR 1ST TIME THRU, END OF ENTITY, OUT OF RANGE
IF END OF ENTITY
Y
N
ERRCAL=2
GO TO (1202, 1203) ERRCAL
IF ERRCAL= 1 - STRIP ENTITY
1202
1203
IF ERRCAL= 2-5
DOTS
DOTS
CLEAN-UP AND DUMP DOTBUF

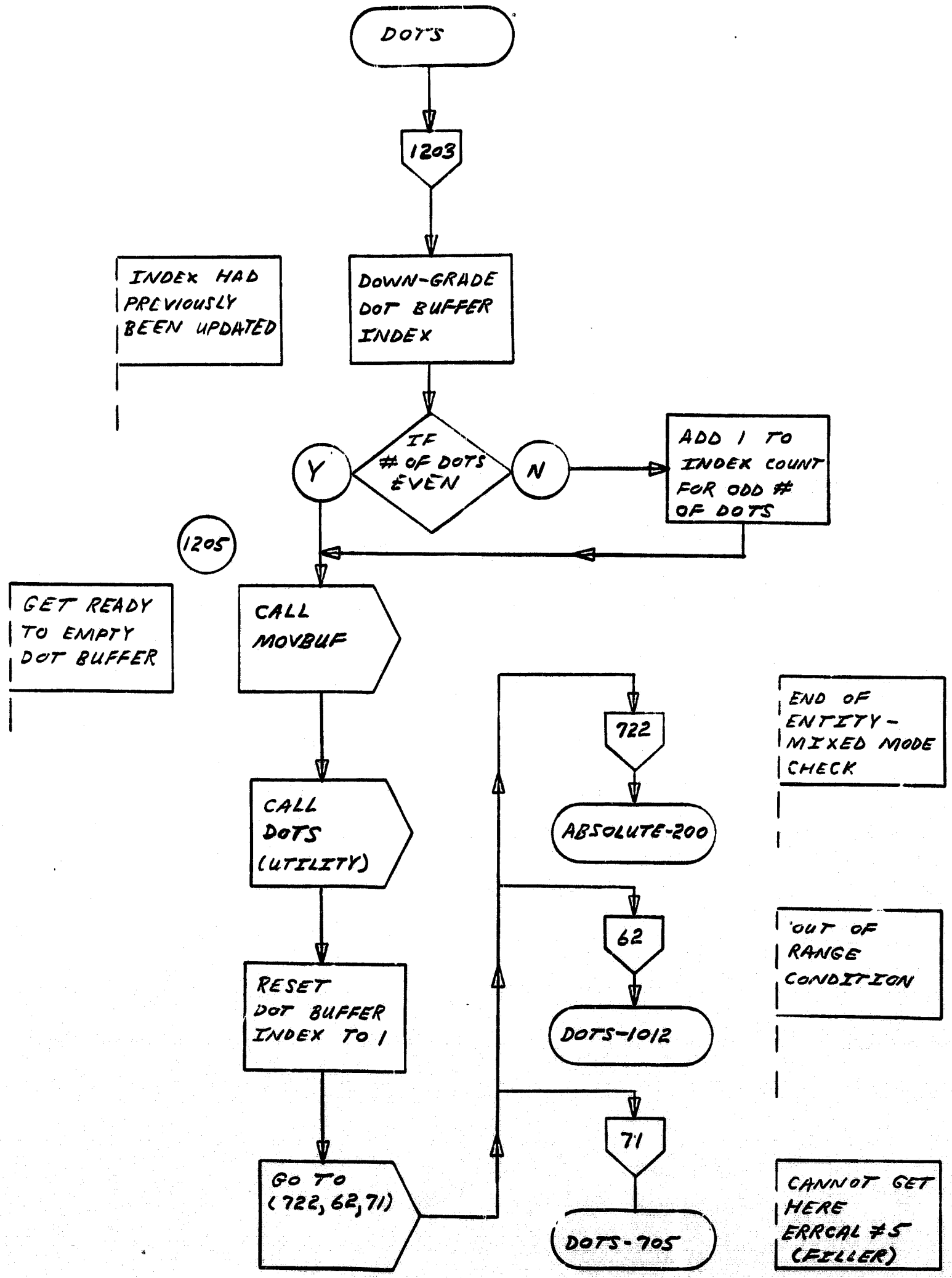

DOTS
1202
CALL NSTRIP
CALL TRFORM
CALL RESET
GO TO (1012,70) IFLG
IFLG=1 1ST TIME THRU
1012
70
IFLG=2 STRIP ENTITY
DOTS
DOTS-30

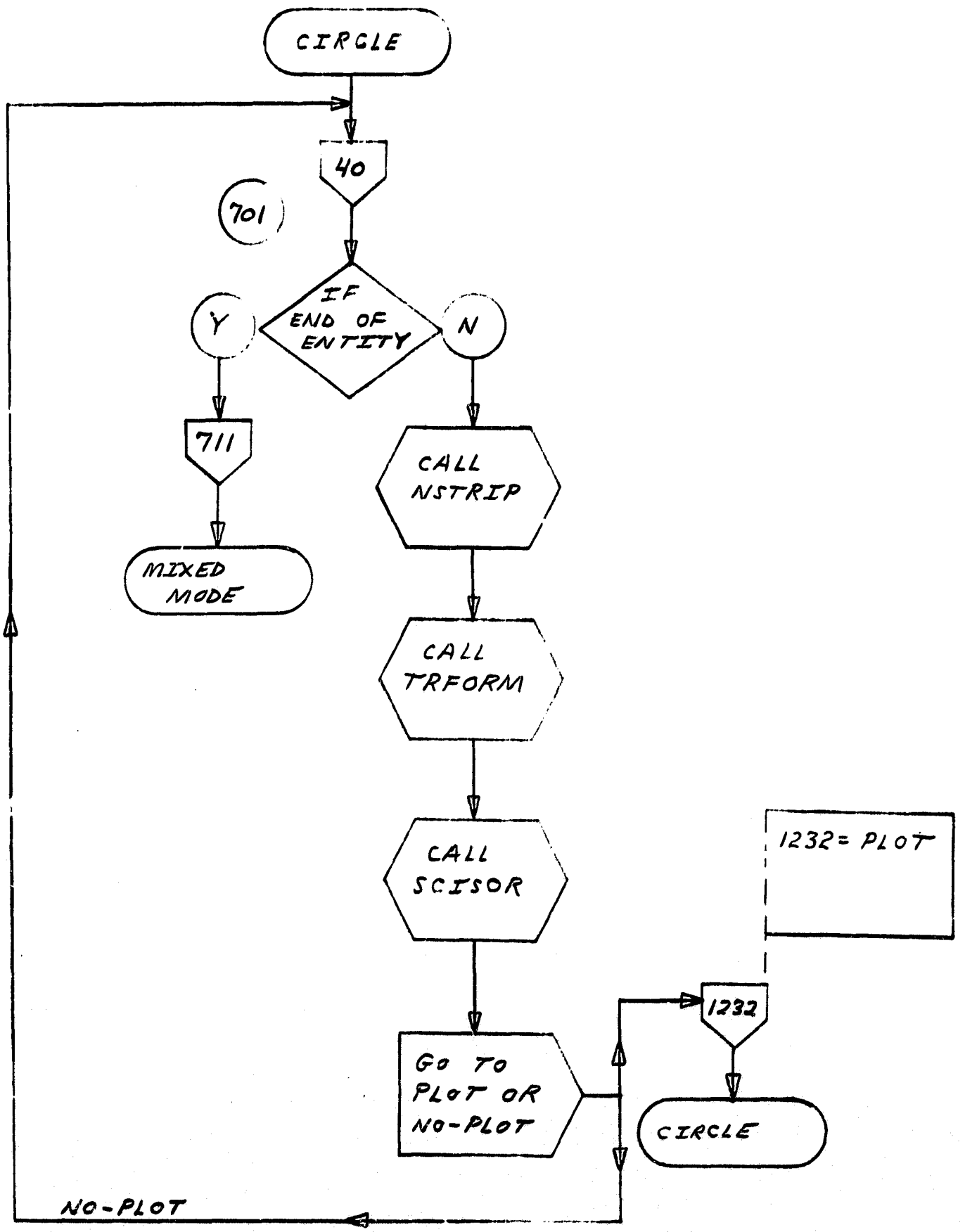

DOTS
1012
CHECK STATUS OF X,Y COORDINATES
GO TO (63, 704) XYSTAT
704
DOTS
63
1ST TIME THRU OR OUT OF RANGE
62
ERRCAL=1
XYSTAT=2
SAVE X,Y COORDINATES
CALL MOVBUF
CALL RESET
1ST DELTA = 0
70-ERRCAL= 1, 2 & 4
71-ERRCAL≠5
72-ERRCAL≠3
GO TO (70, 72, 71) ERRCAL
70
DOTS-30
72
DOTS
71
DOTS

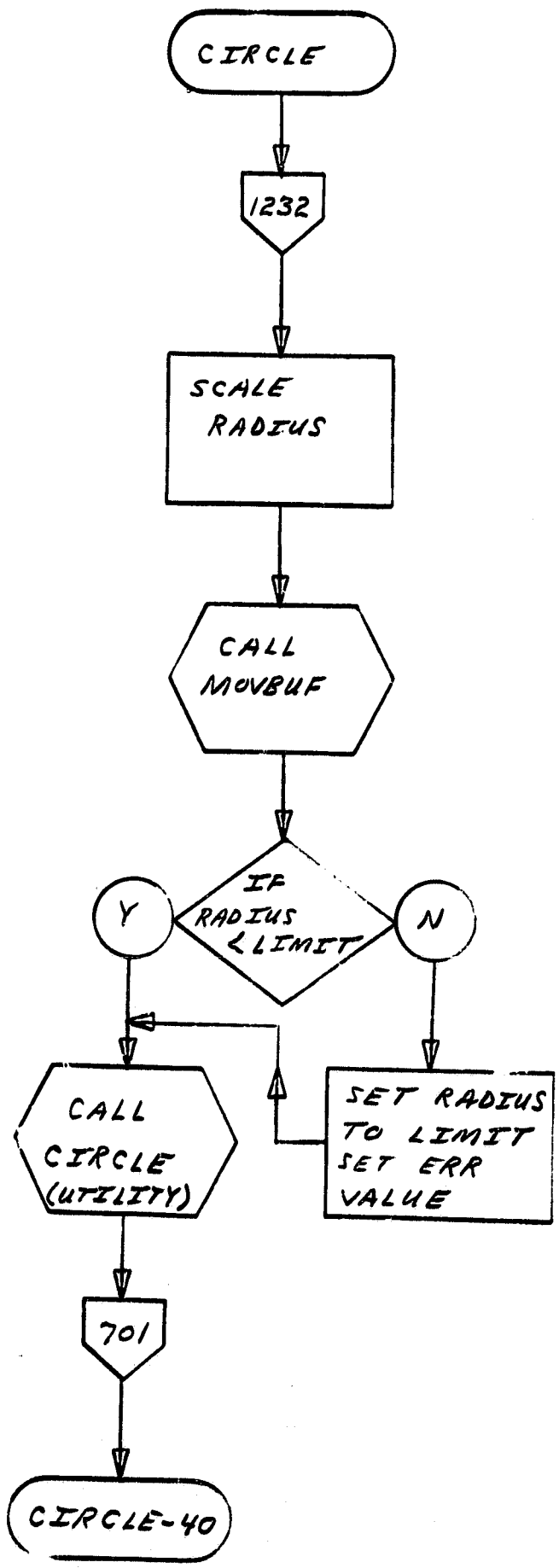

DOTS
704
IF ERROR CONDITION
Y
N
IF ERRCAL = 1 OR 2 GO TO 72
ERRCAL = 3 (ERR FLAG SET)
GO TO (72,1203)
72
DOTS
1203
RESET THE BEAM
DOTS

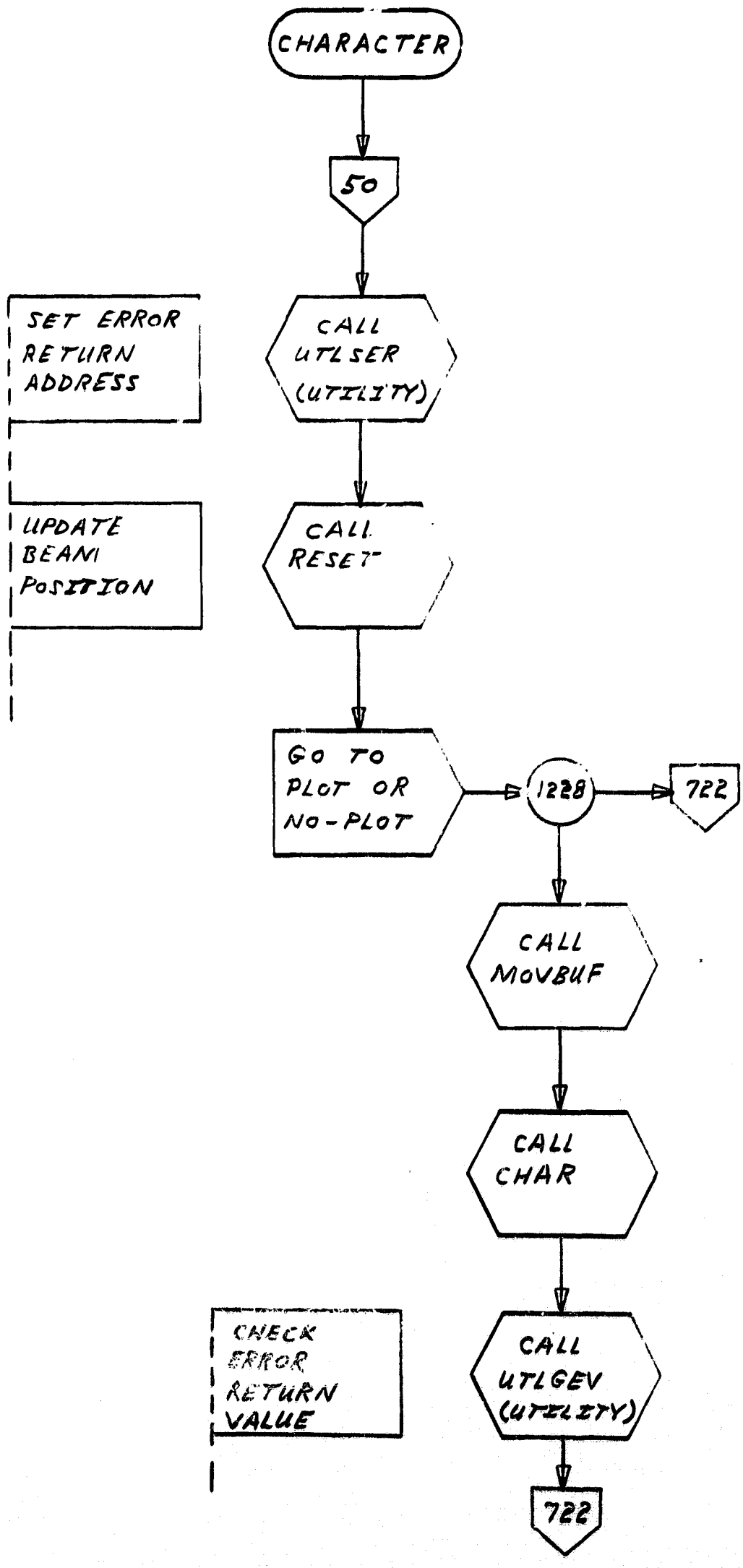

DOTS
72
CALCULATE
Δx, Δy
IF
Δ > RANGE
Y
N
SET ERROR
VALUE TO 5
IF
ERROR IS
SET
N
Y
SET GO
TO INDEX
TO 4
(ERRCAL=4)
705
DOTS
IF ERRCAL
= 1 OR 2
GO TO 705
GO TO
(705,1203)
1203
DOTS

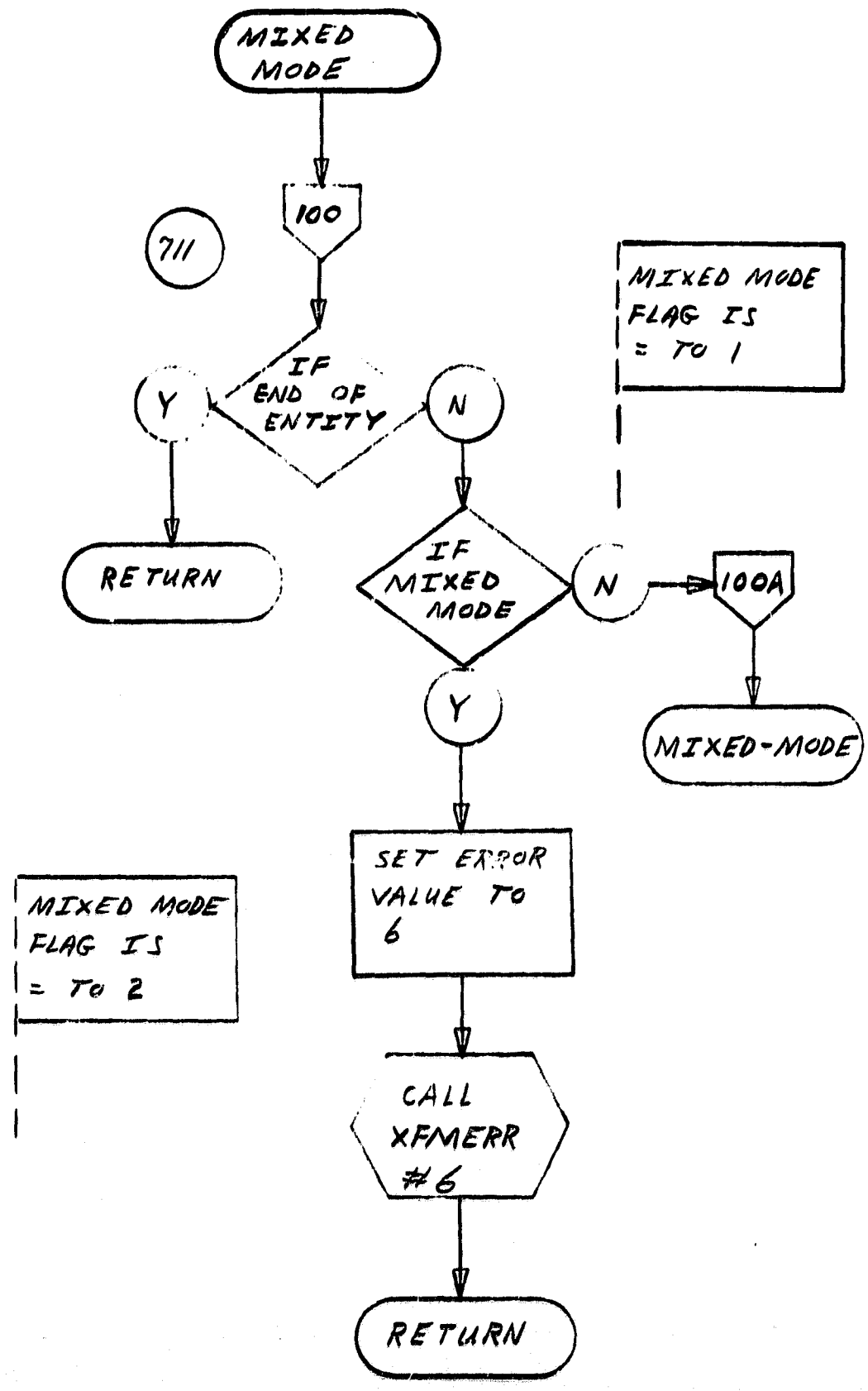

DOTS
705
IF DOTBUF FULL
Y
N
IF ERRCAL = 1 OR 2 GO TO 71
SET GO TO INDEX TO 5
GO TO (71,1203)
1203
DOTS
71
STORE ΔX, ΔY IN DOTBUF
UPDATE DOT BUFFER SAVE X, Y COORDINATE
70
DOTS-30

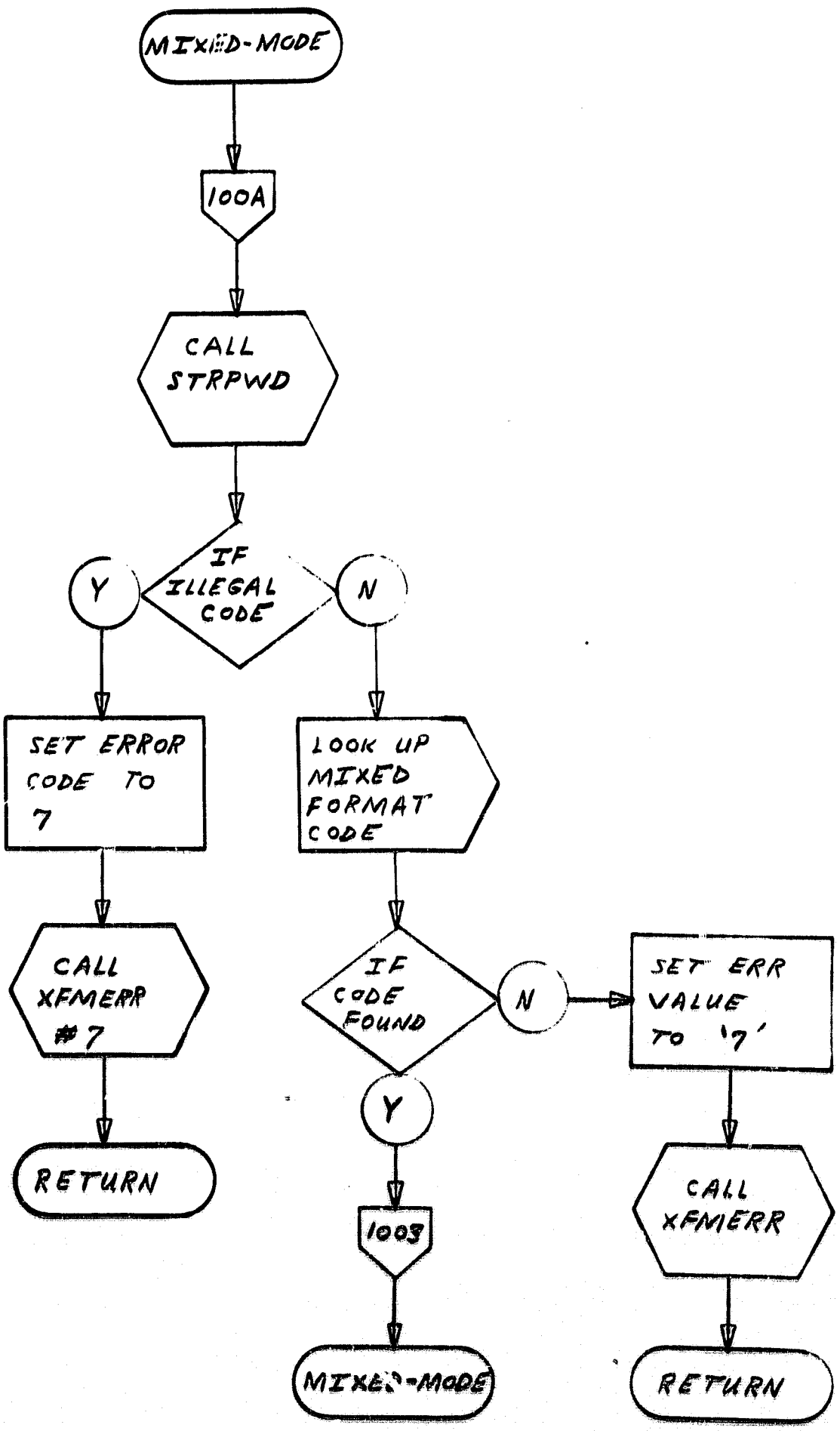

DOTS
1203
INDEX HAD PREVIOUSLY BEEN UPDATED
DOWN-GRADE DOT BUFFER INDEX
IF # OF DOTS EVEN
Y
N
ADD 1 TO INDEX COUNT FOR ODD # OF DOTS
1205
GET READY TO EMPTY DOT BUFFER
CALL MOVBUF
CALL DOTS (UTILITY)
RESET DOT BUFFER INDEX TO 1
GO TO (722, 62, 71)
722
ABSOLUTE-200
END OF ENTITY - MIXED MODE CHECK
62
DOTS-1012
'OUT OF RANGE CONDITION
71
DOTS-705
CANNOT GET HERE ERRCAL #5 (FILLER)

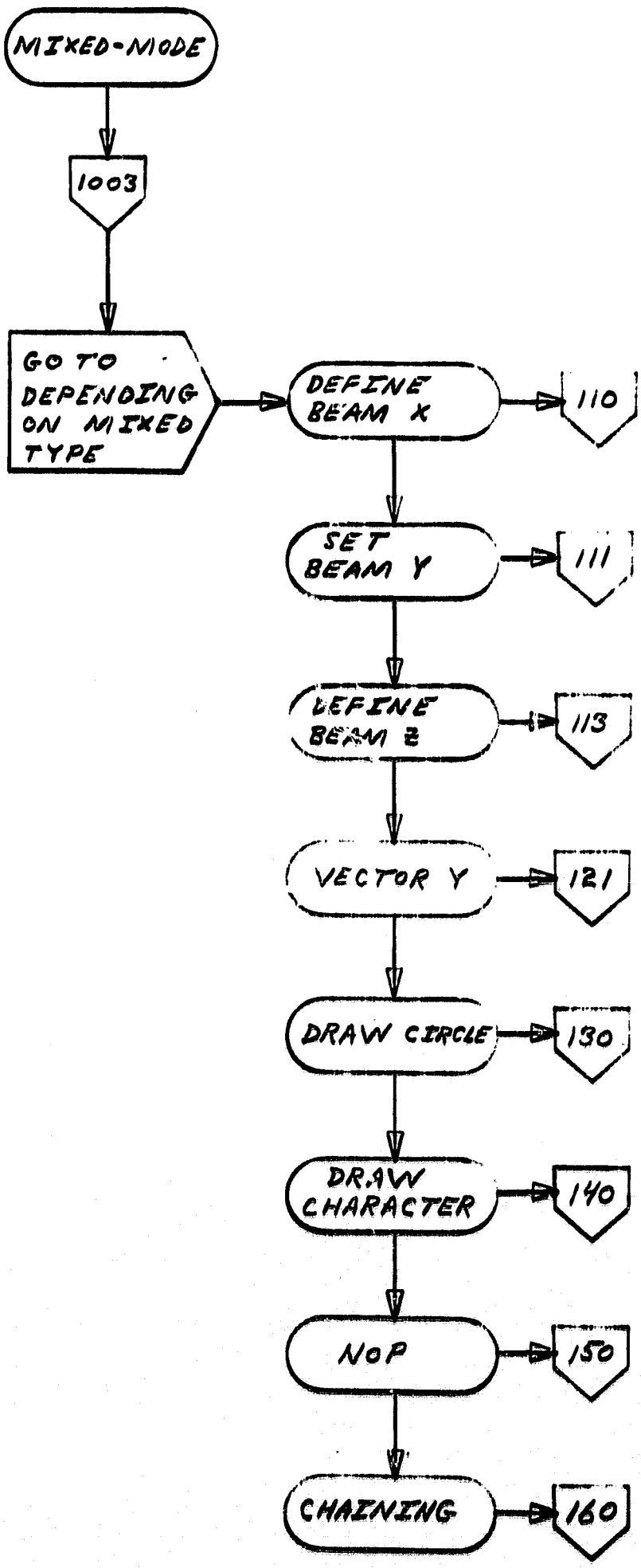

CIRCLE
40
701
IF END OF ENTITY
Y
N
711
CALL NSTRIP
MIXED MODE
CALL TRFORM
CALL SCISOR
1232= PLOT
1232
GO TO PLOT OR NO-PLOT
CIRCLE
NO-PLOT

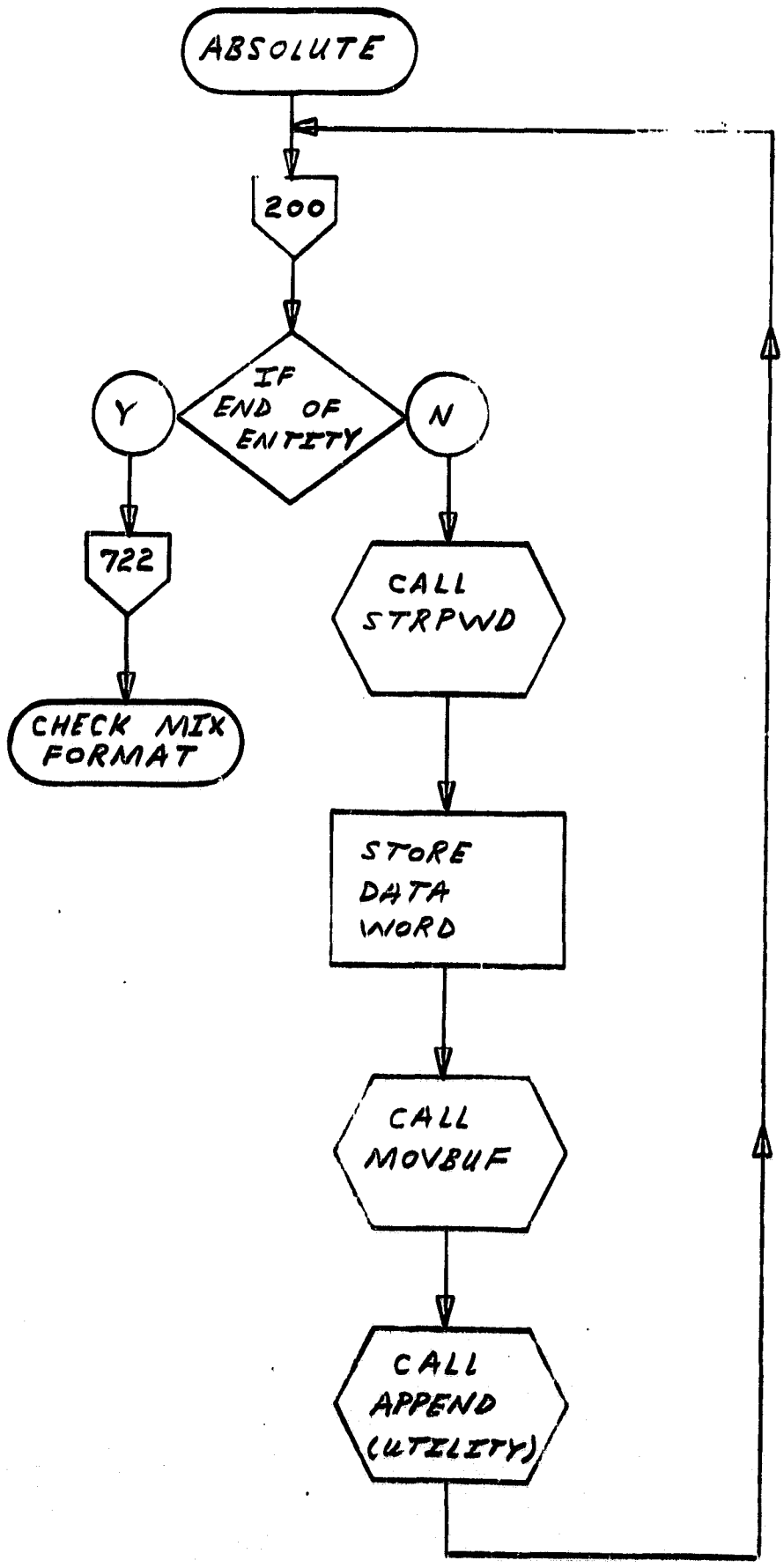

CIRCLE
1232
SCALE
RADIUS
CALL
MOVBUF
IF
RADIUS
LLIMIT
Y
N
CALL
CIRCLE
(UTILITY)
SET RADIUS
TO LIMIT
SET ERR
VALUE
701
CIRCLE-40

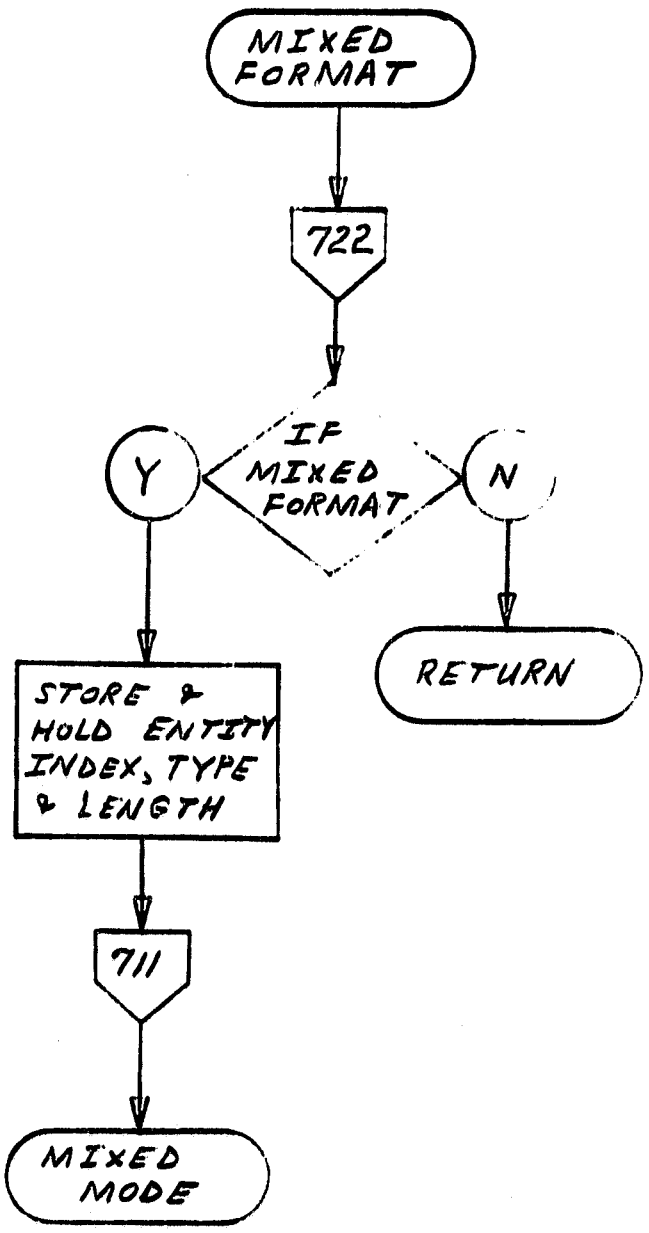

CHARACTER
50
SET ERROR RETURN ADDRESS
CALL UTLSER (UTILITY)
UPDATE BEAM POSITION
CALL RESET
GO TO PLOT OR NO-PLOT
1228
722
CALL MOVBUF
CALL CHAR
CHECK ERROR RETURN VALUE
CALL UTLGEV (UTELITY)
722

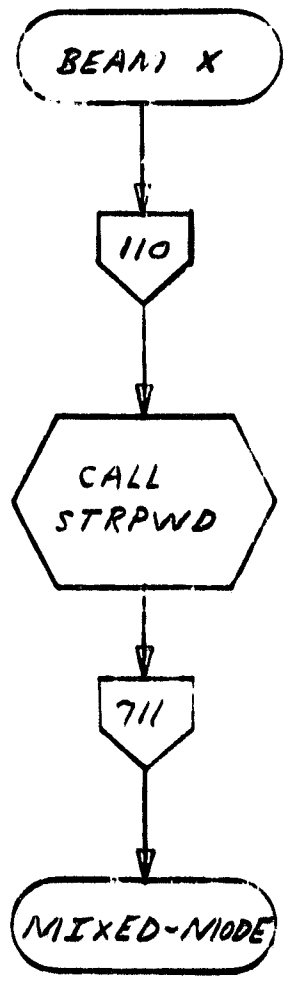

MIXED MODE
100
711
IF END OF ENTITY
Y
N
RETURN
MIXED MODE FLAG IS = TO 1
IF MIXED MODE
N
100A
MIXED-MODE
Y
SET ERROR VALUE TO 6
MIXED MODE FLAG IS = TO 2
CALL XFMERR #6
RETURN

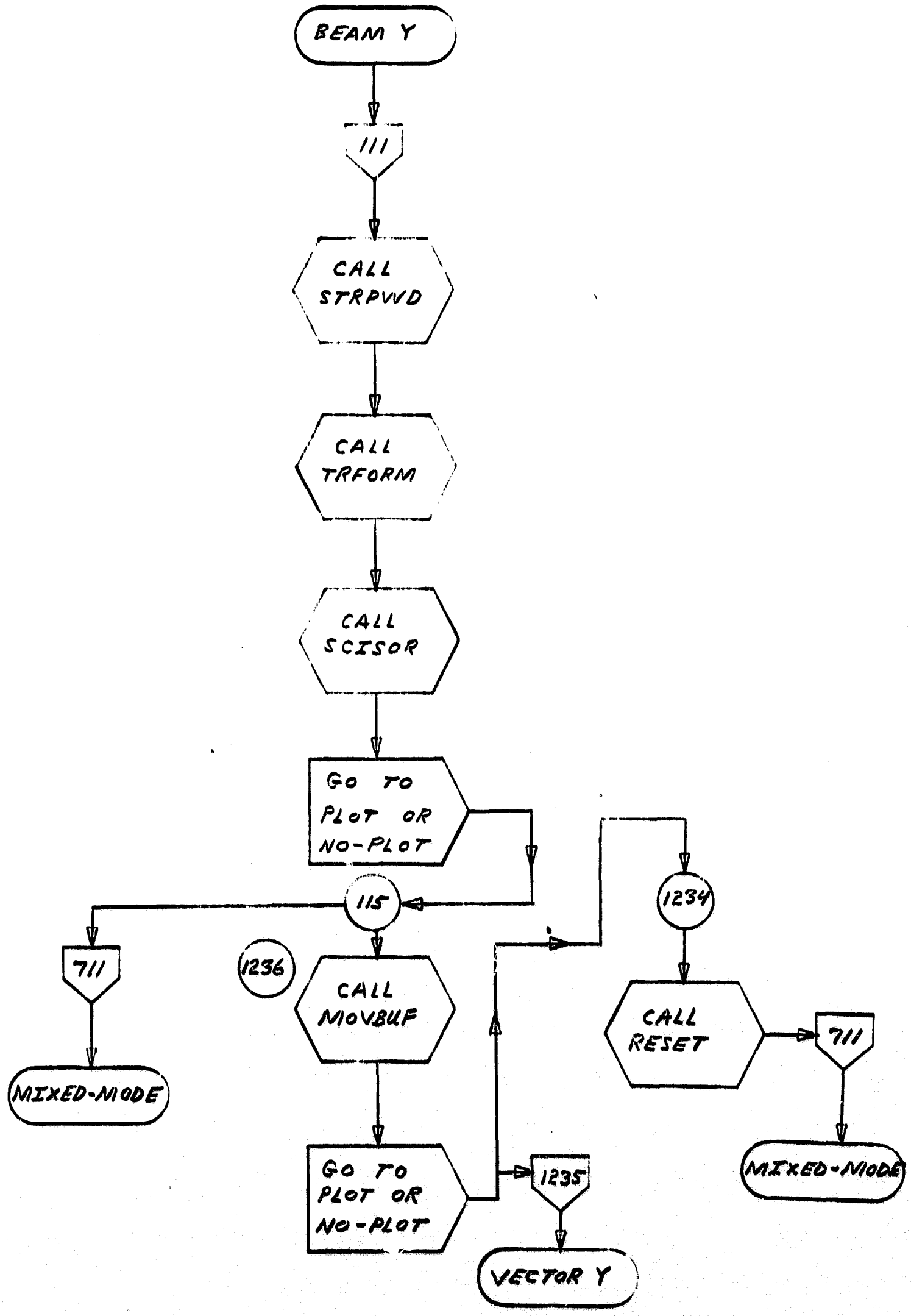

MIXED-MODE
100A
CALL STRPWD
IF ILLEGAL CODE
Y
N
SET ERROR CODE TO 7
CALL XFMERR #7
RETURN
LOOK UP MIXED FORMAT CODE
IF CODE FOUND
N
SET ERR VALUE TO '7'
Y
1003
MIXED-MODE
CALL XFMERR
RETURN

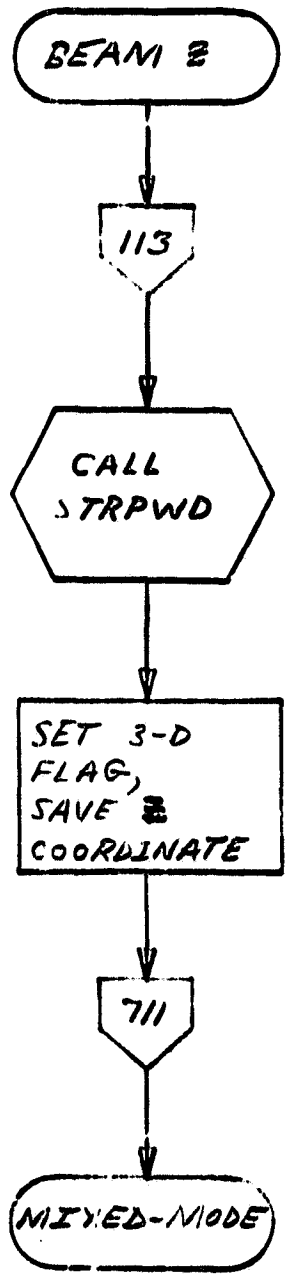

MIXED-MODE
1003
GO TO DEPENDING ON MIXED TYPE
DEFINE BEAM X
110
SET BEAM Y
111
DEFINE BEAM Z
113
VECTOR Y
121
DRAW CIRCLE
130
DRAW CHARACTER
140
NOP
150
CHAINING
160

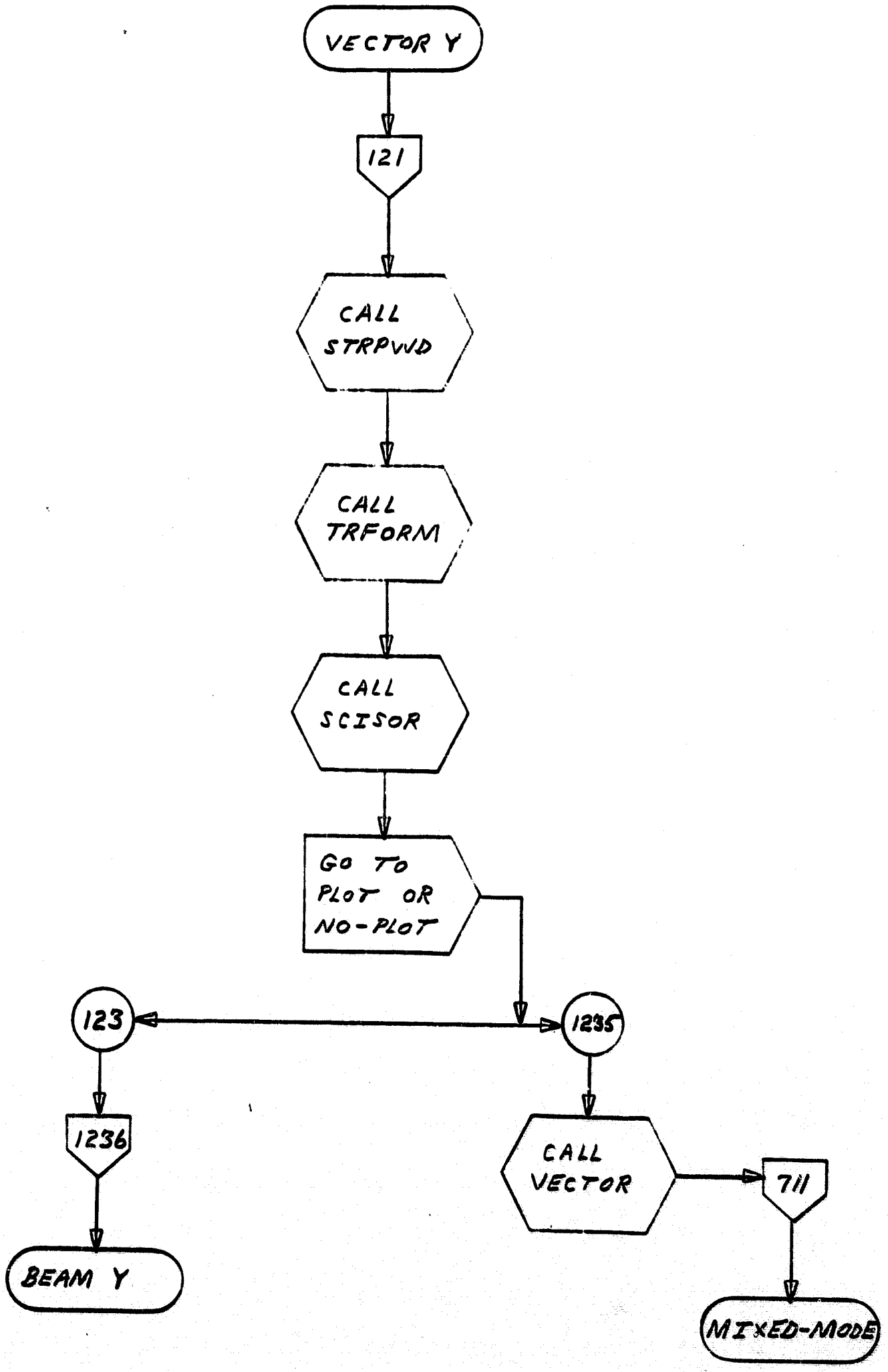

ABSOLUTE
200
IF END OF ENTITY
Y
N
722
CHECK MIX FORMAT
CALL STRPWD
STORE DATA WORD
CALL MOVBUF
CALL APPEND (UTILITY)

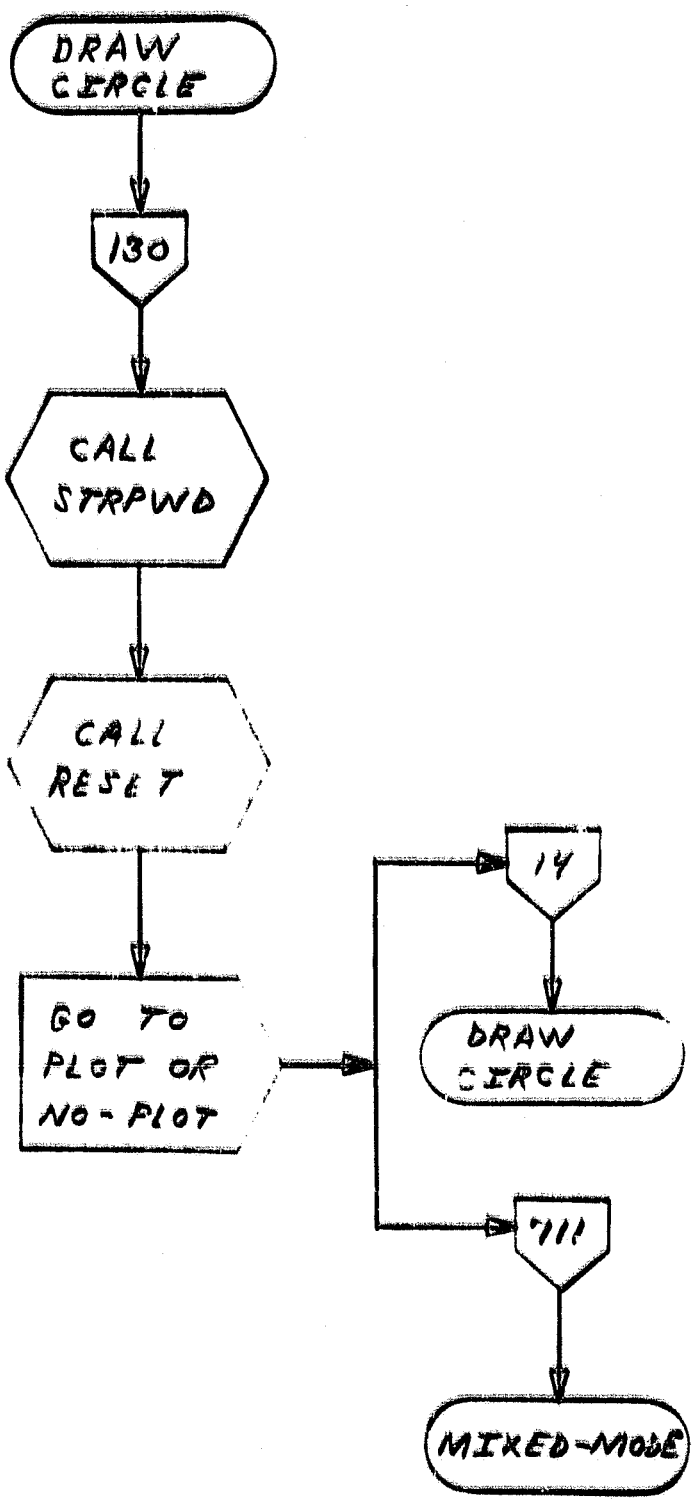

MIXED FORMAT
722
IF MIXED FORMAT
Y
N
RETURN
STORE & HOLD ENTITY INDEX, TYPE & LENGTH
711
MIXED MODE

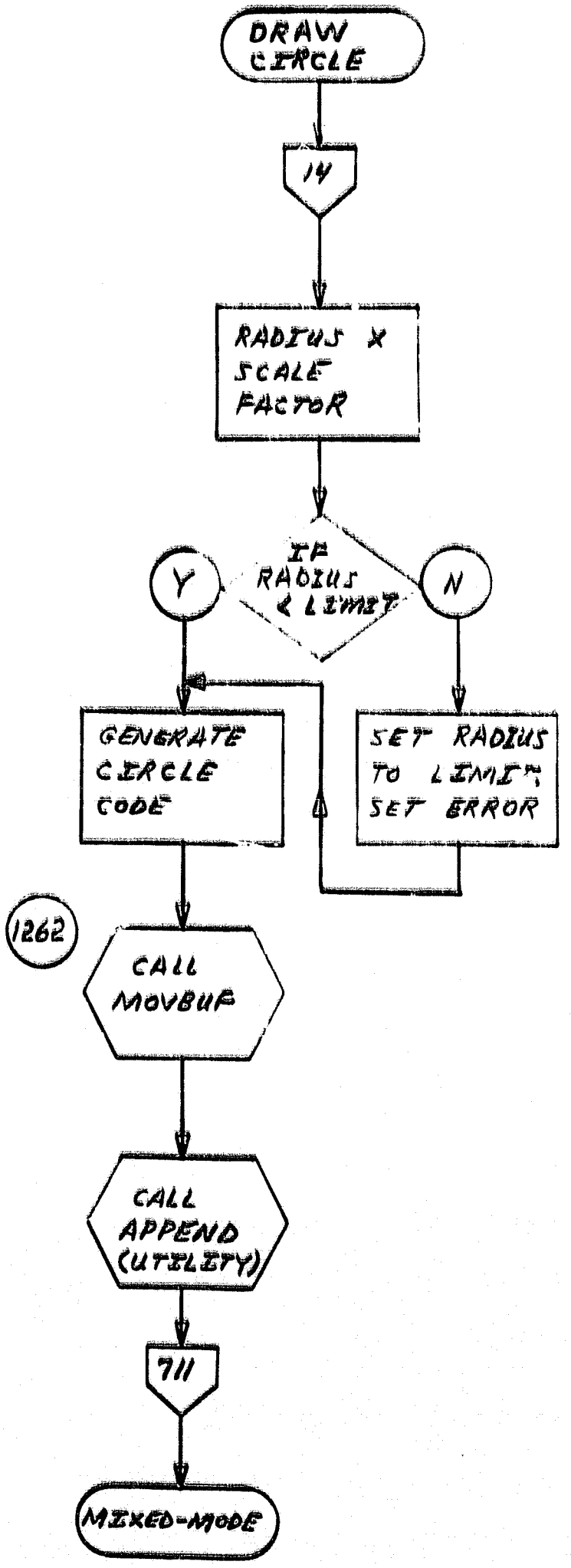

BEAM X
110
CALL
STRPWD
711
MIXED-MODE

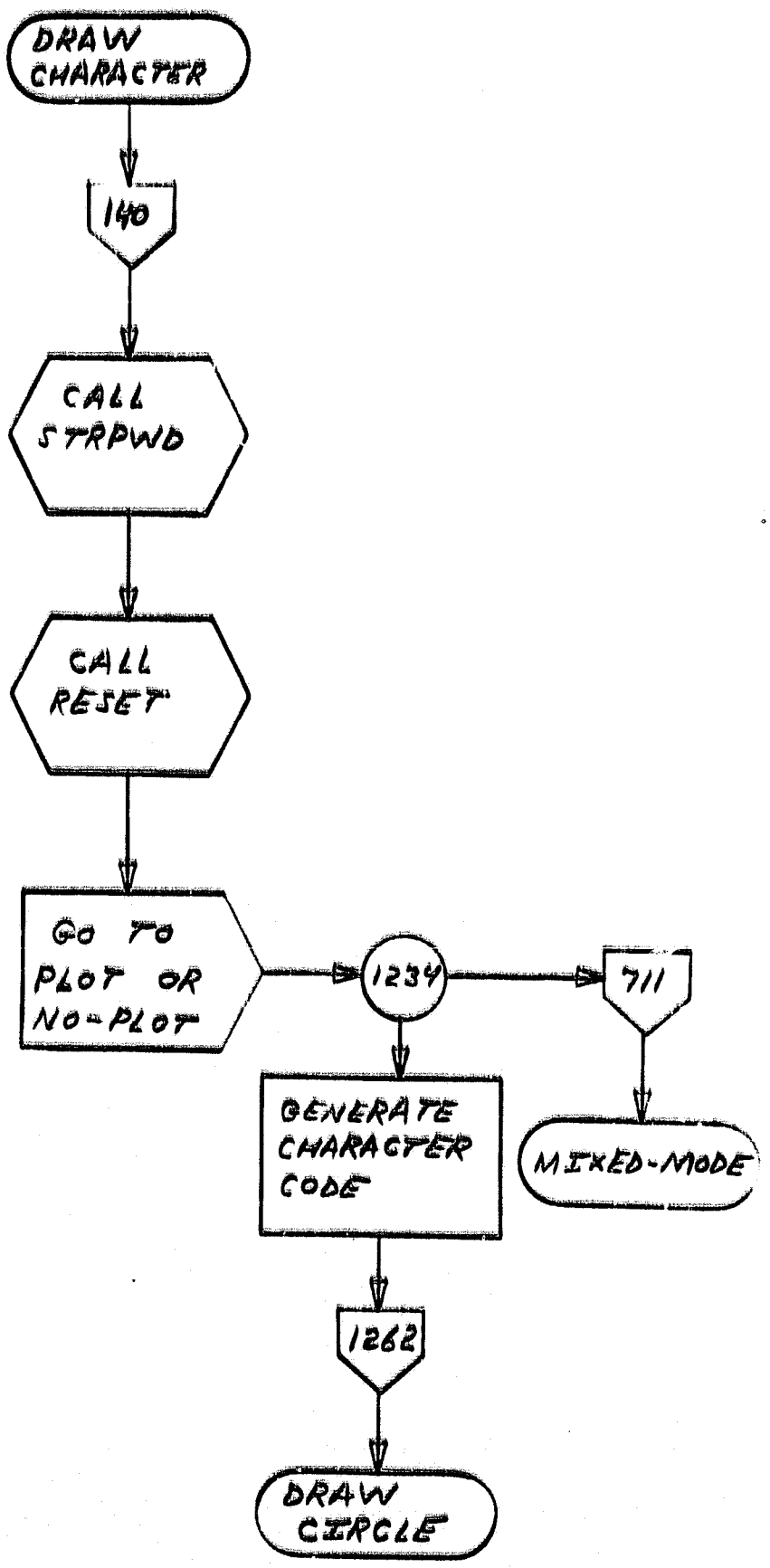

BEAM Y
111
CALL STRPVVD
CALL TRFORM
CALL SCISOR
GO TO PLOT OR NO-PLOT
115
711
MIXED-MODE
1236
CALL MOVBUF
GO TO PLOT OR NO-PLOT
1235
VECTOR Y
1234
CALL RESET
711
MIXED-MODE

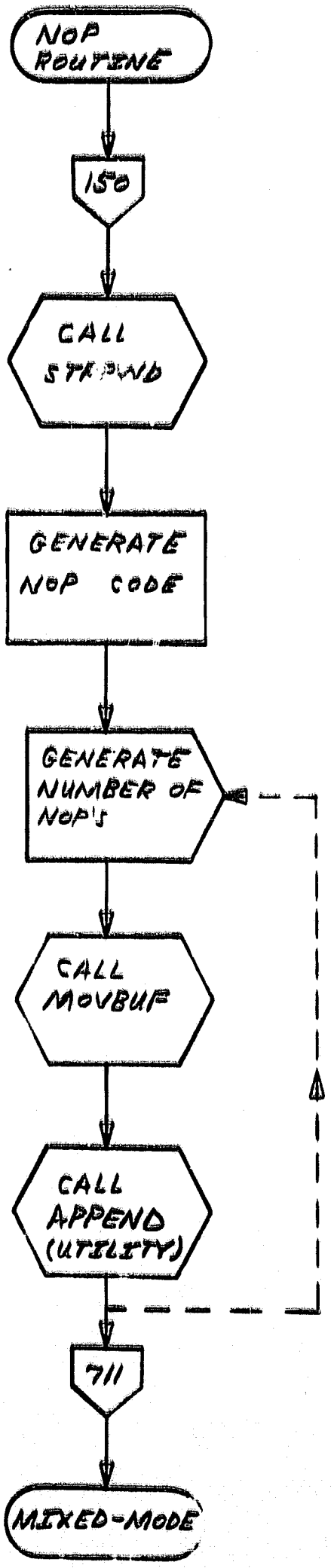

BEAM 2
113
CALL
STRPWD
SET 3-D
FLAG,
SAVE
COORDINATE
711
MIXED-MODE

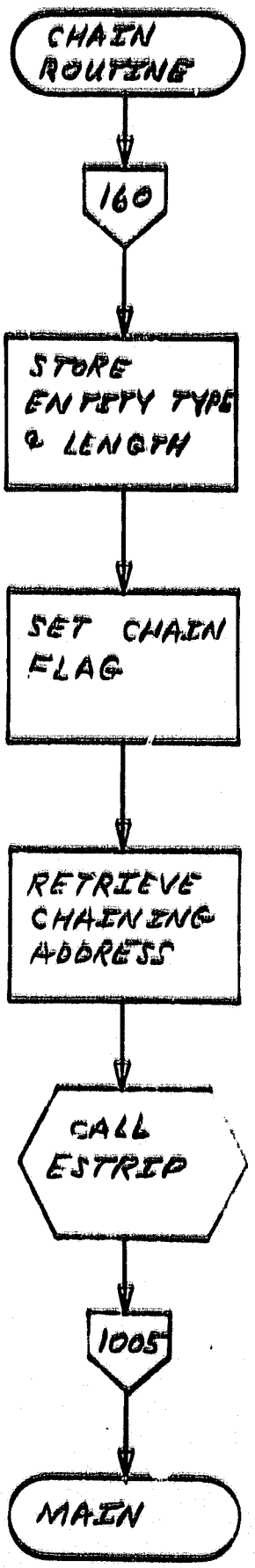

VECTOR Y
121
CALL STRPVVD
CALL TRFORM
CALL SCISOR
GO TO PLOT OR NO-PLOT
123
1235
1236
CALL VECTOR
711
BEAM Y
MIXED-MODE


DRAW CIRCLE
130
CALL STRPWD
CALL RESET
GO TO PLOT OR NO-PLOT
14
DRAW CIRCLE
111
MIXED-MODE


DRAW CIRCLE
14
RADIUS X SCALE FACTOR
IF RADIUS < LIMIT
Y
N
GENERATE CIRCLE CODE
SET RADIUS TO LIMIT; SET ERROR
1262
CALL MOVBUF
CALL APPEND (UTILITY)
711
MIXED-MODE


DRAW CHARACTER
140
CALL STRPWD
CALL RESET
GO TO PLOT OR NO-PLOT
1234
711
GENERATE CHARACTER CODE
MIXED-MODE
1262
DRAW CIRCLE

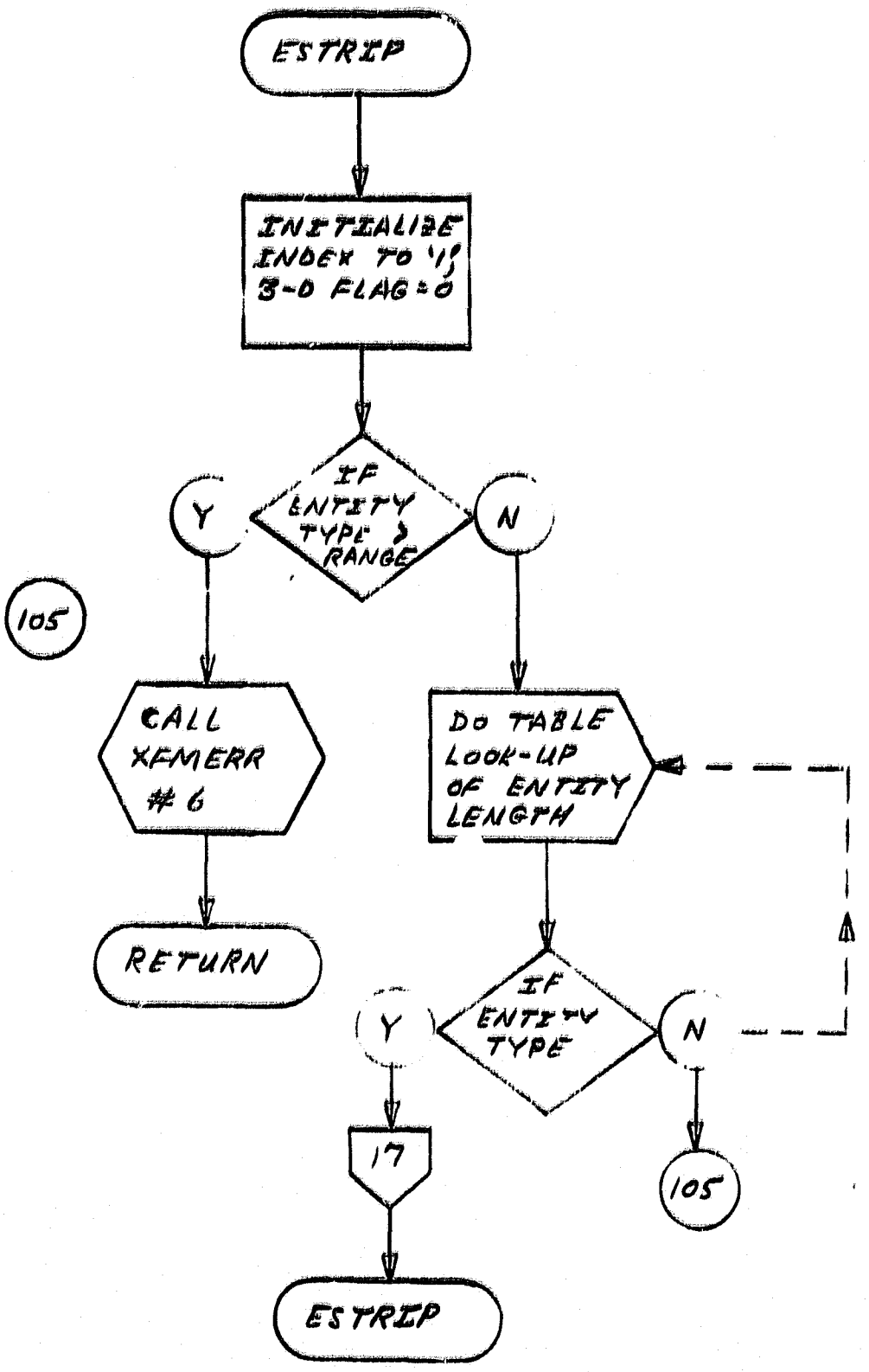

NOP ROUTINE
150
CALL STRPWD
GENERATE NOP CODE
GENERATE NUMBER OF NOP's
CALL MOVBUF
CALL APPEND (UTILITY)
711
MIXED-MODE

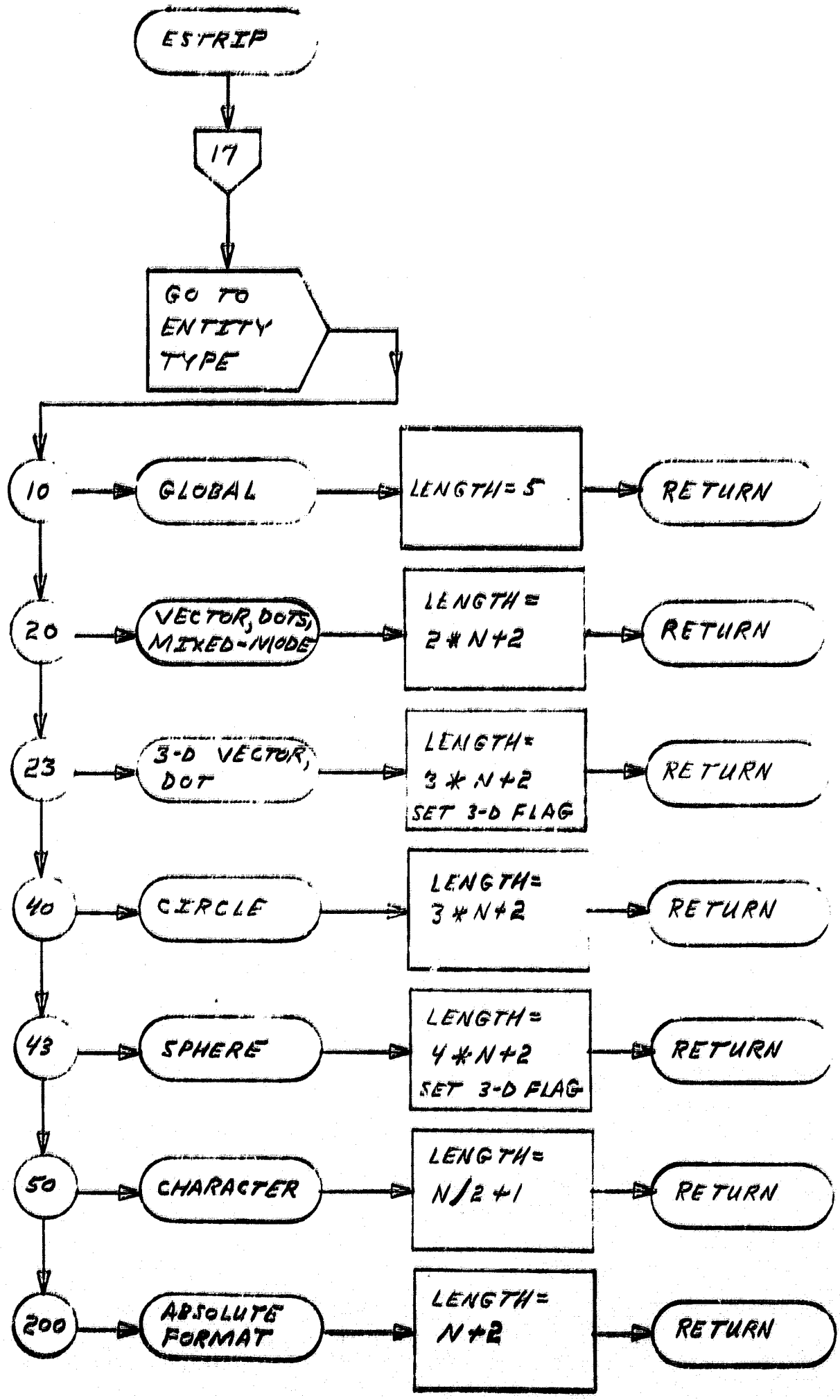

CHAIN ROUTINE
160
STORE ENTITY TYPE & LENGTH
SET CHAIN FLAG
RETRIEVE CHAINING ADDRESS
CALL ESTRIP
1005
MAIN

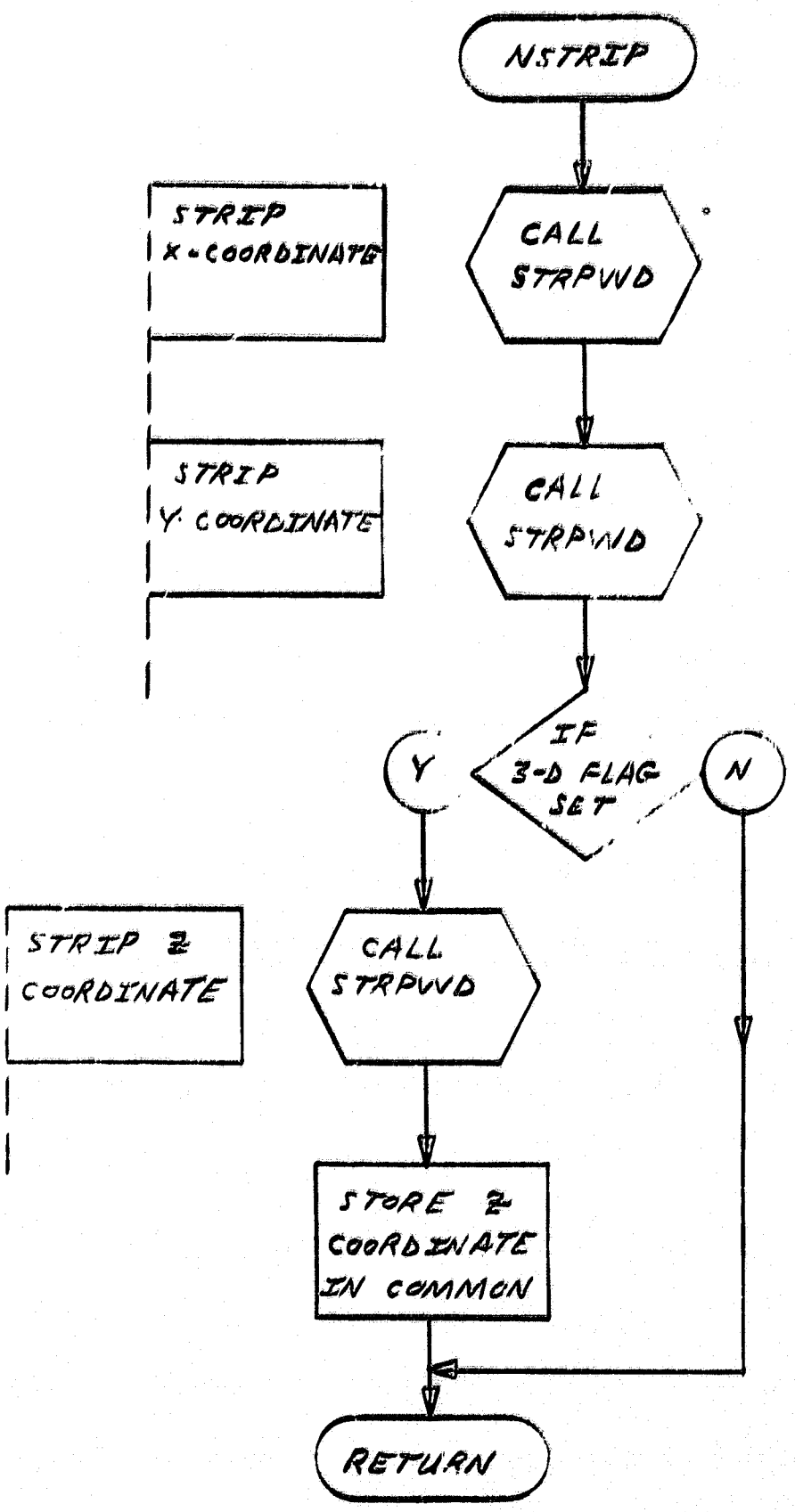
adams associates

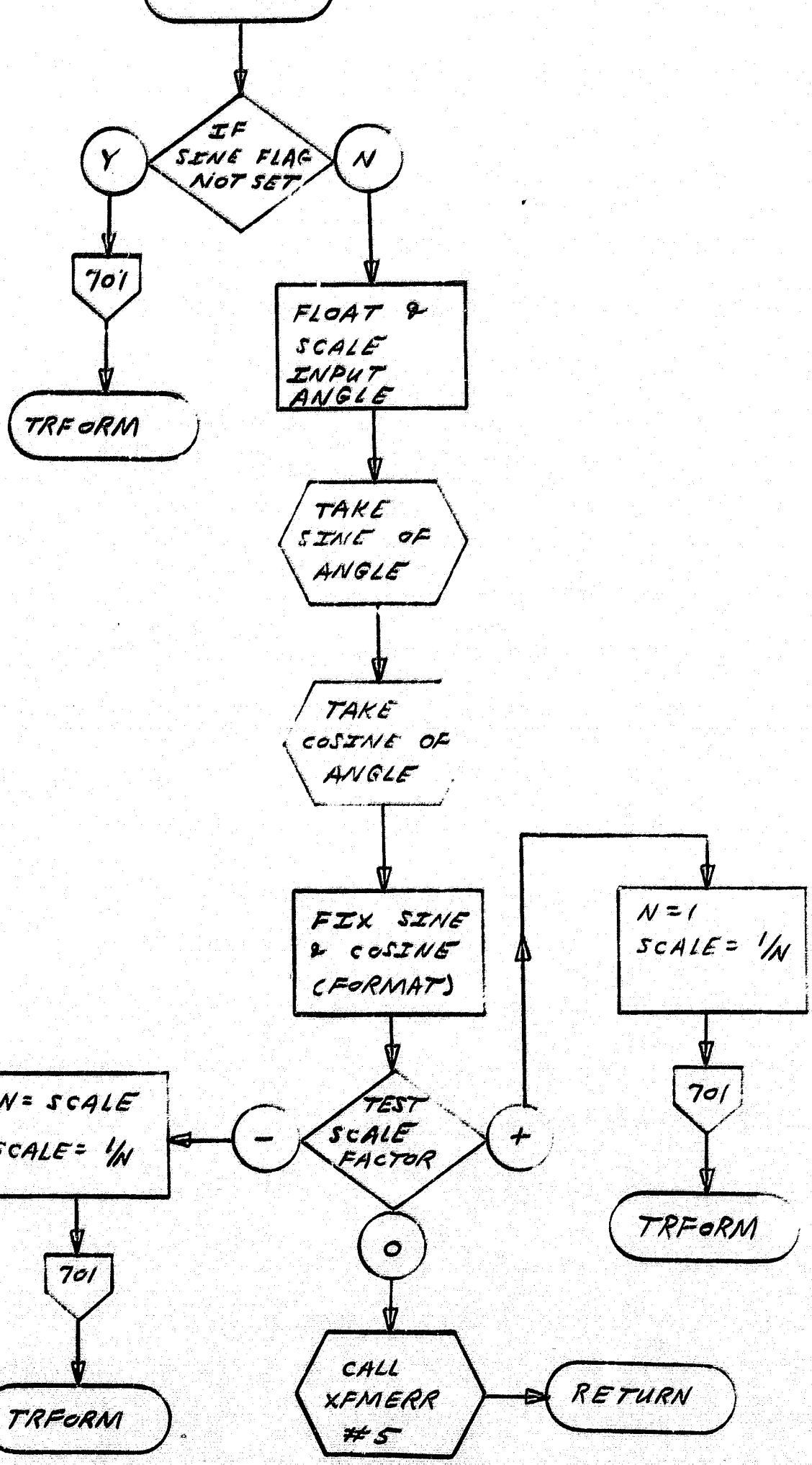
Appendix B

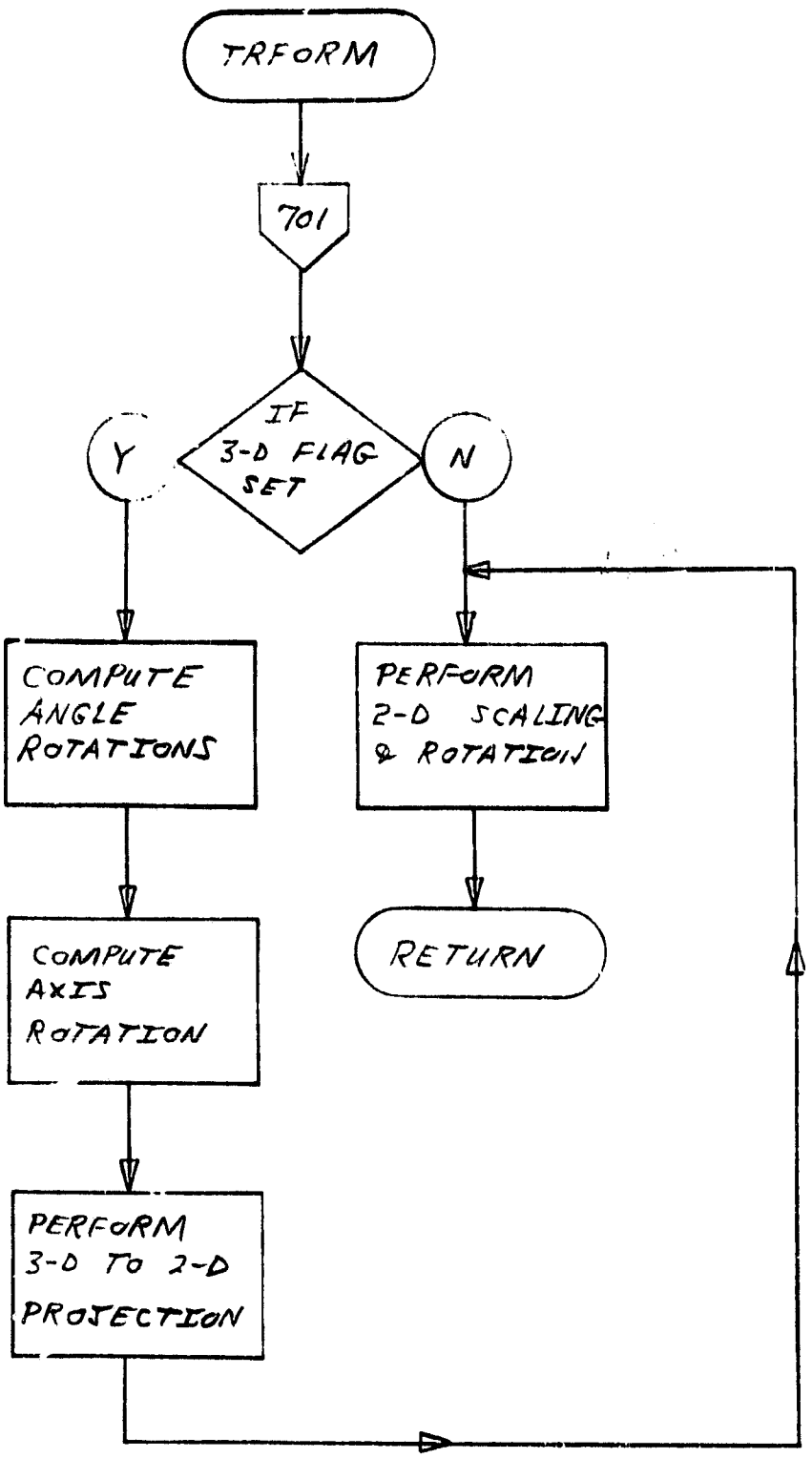
SERVICE ROUTINE FLOWCHARTS

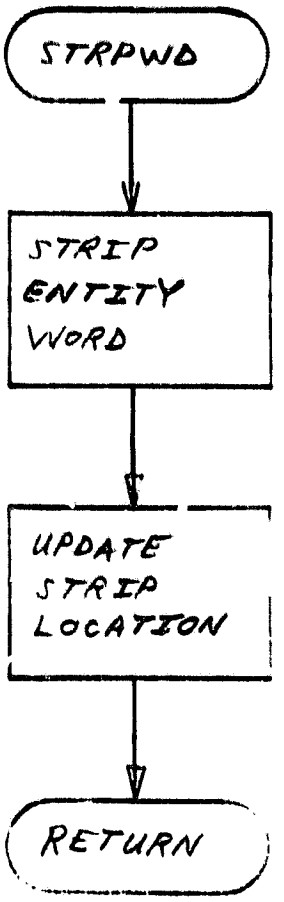

ESTRIP
INITIALIZE INDEX TO '1' 3-0 FLAG = 0
IF ENTITY TYPE > RANGE
Y
N
105
CALL XFMERR #6
RETURN
DO TABLE LOOK-UP OF ENTITY LENGTH
IF ENTITY TYPE
Y
N
17
105
ESTRIP

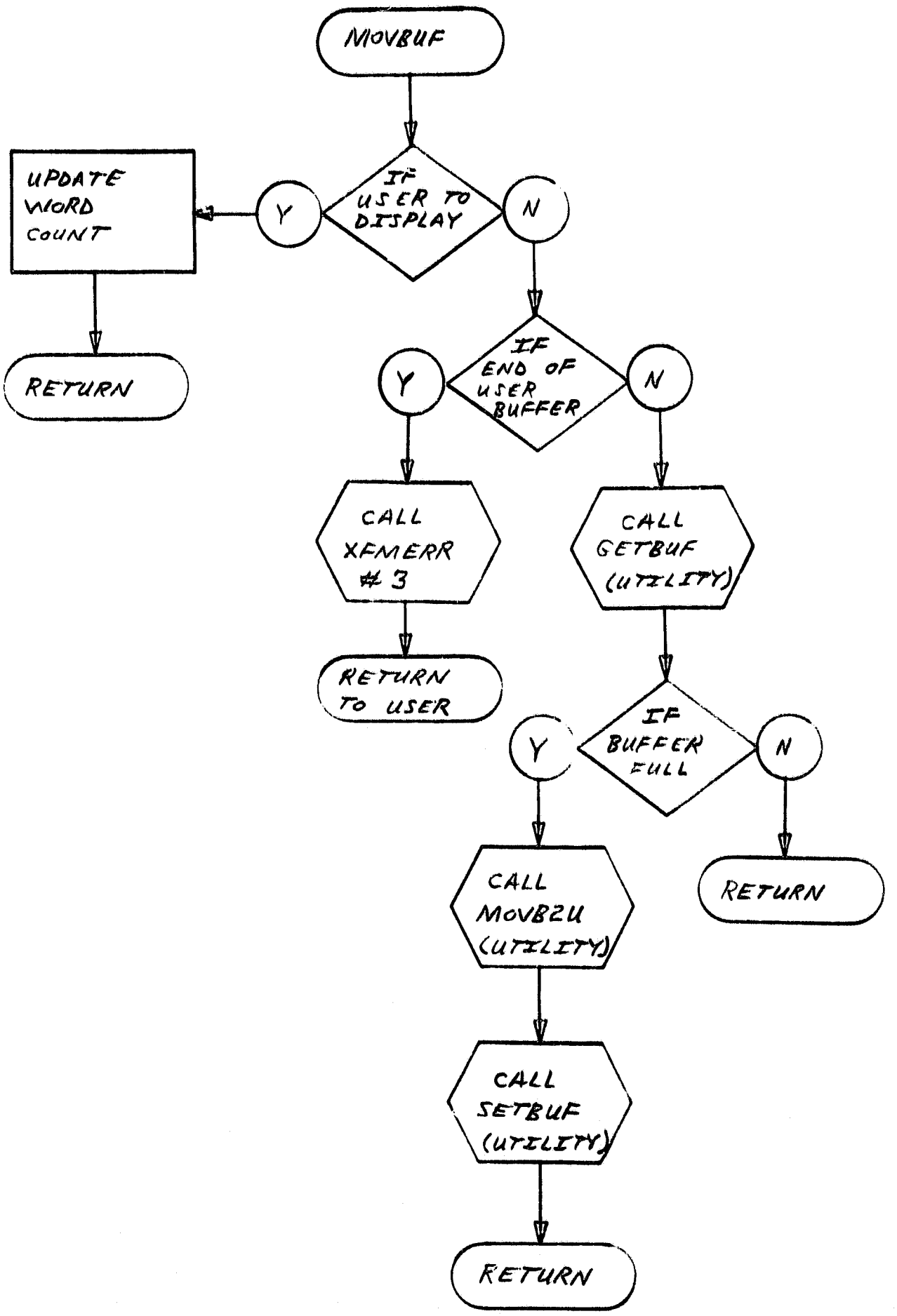

ESTRIP
17
GO TO ENTITY TYPE
10
GLOBAL
LENGTH=5
RETURN
20
VECTOR, DOTS, MIXED-MODE
LENGTH= 2*N+2
RETURN
23
3-D VECTOR, DOT
LENGTH= 3*N+2 SET 3-D FLAG
RETURN
40
CIRCLE
LENGTH= 3*N+2
RETURN
43
SPHERE
LENGTH= 4*N+2 SET 3-D FLAG
RETURN
50
CHARACTER
LENGTH= N/2+1
RETURN
200
ABSOLUTE FORMAT
LENGTH= N+2
RETURN

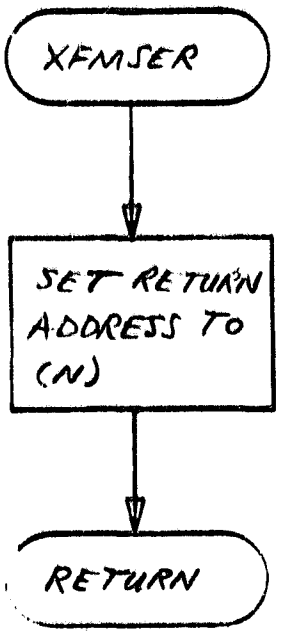

NSTRIP
CALL STRPVVD
STRIP X-COORDINATE
CALL STRPVVD
STRIP Y-COORDINATE
IF 3-D FLAG SET
Y
N
CALL STRPVVD
STRIP Z COORDINATE
STORE Z COORDINATE IN COMMON
RETURN

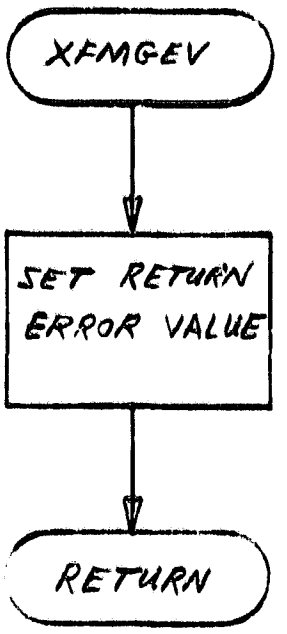

TRFORM
IF SINE FLAG NOT SET
Y
N
701
TRFORM
FLOAT & SCALE INPUT ANGLE
TAKE SINE OF ANGLE
TAKE COSINE OF ANGLE
FIX SINE & COSINE (FORMAT)
TEST SCALE FACTOR
-
+
0
N = SCALE
SCALE = 1/N
701
TRFORM
N = 1
SCALE = 1/N
701
TRFORM
CALL XFMERR #5
RETURN

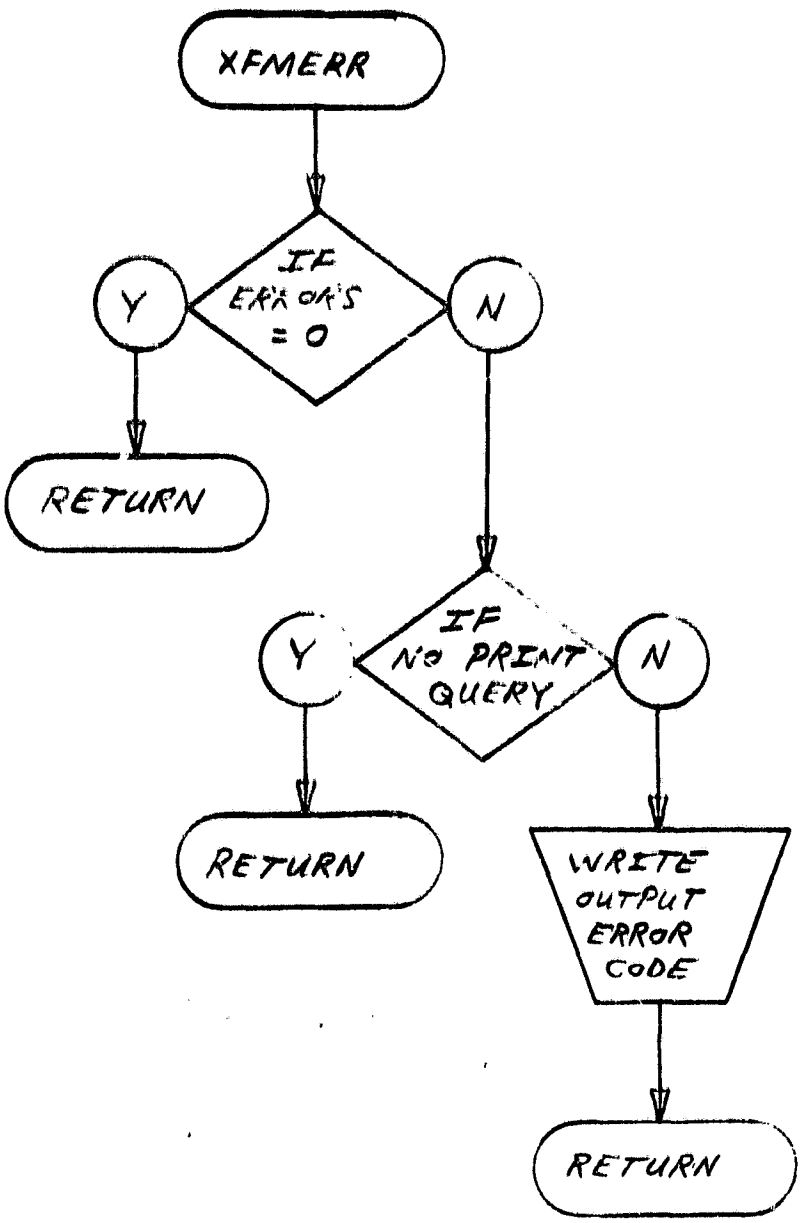

TRFORM
701
IF
3-D FLAG
SET
Y
N
COMPUTE
ANGLE
ROTATIONS
PERFORM
2-D SCALING
& ROTATION
COMPUTE
AXIS
ROTATION
RETURN
PERFORM
3-D TO 2-D
PROJECTION


STRPWD
STRIP
ENTITY
VWORD
UPDATE
STRIP
LOCATION
RETURN


MOVBUF
IF USER TO DISPLAY
Y
N
UPDATE WORD COUNT
RETURN
IF END OF USER BUFFER
Y
N
CALL XFMERR #3
RETURN TO USER
CALL GETBUF (UTILITY)
IF BUFFER FULL
Y
N
CALL MOVB2U (UTILITY)
RETURN
CALL SETBUF (UTILITY)
RETURN


XFMSER
SET RETURN ADDRESS TO (N)
RETURN

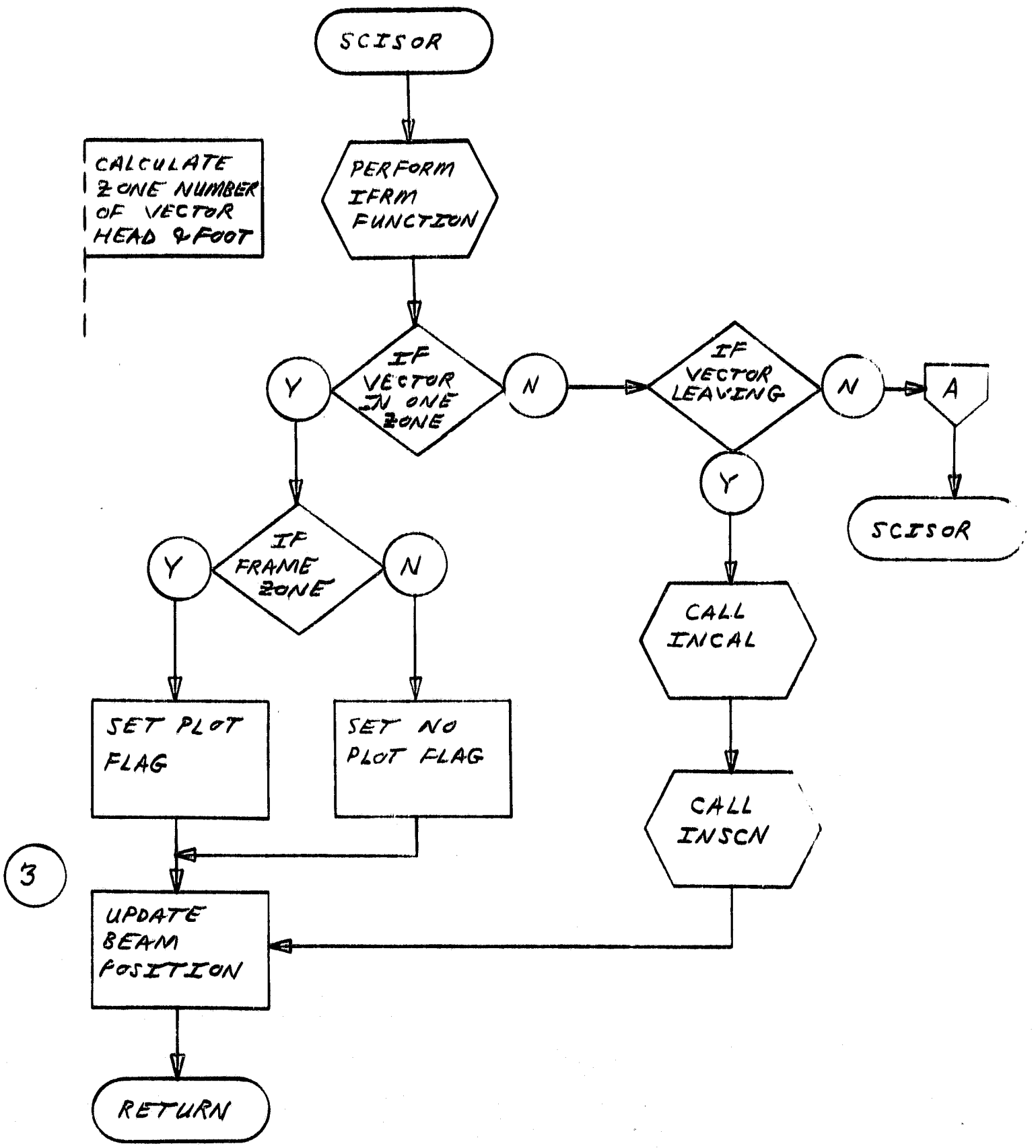

XFMGEV
SET RETURN ERROR VALUE
RETURN

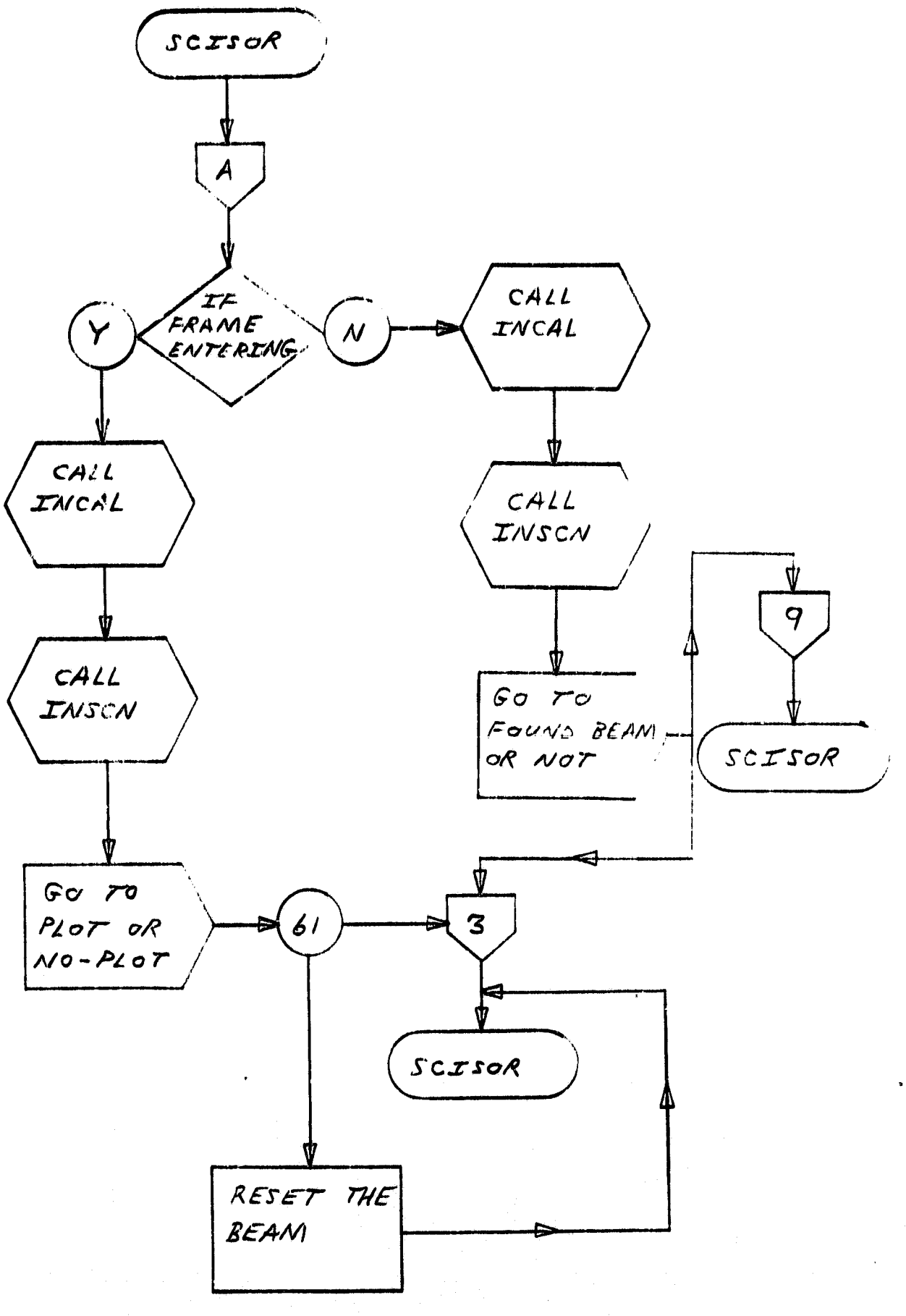

XFMERR
IF ERROR'S = 0
Y
N
RETURN
IF NO PRINT QUERY
Y
N
RETURN
WRITE OUTPUT ERROR CODE
RETURN

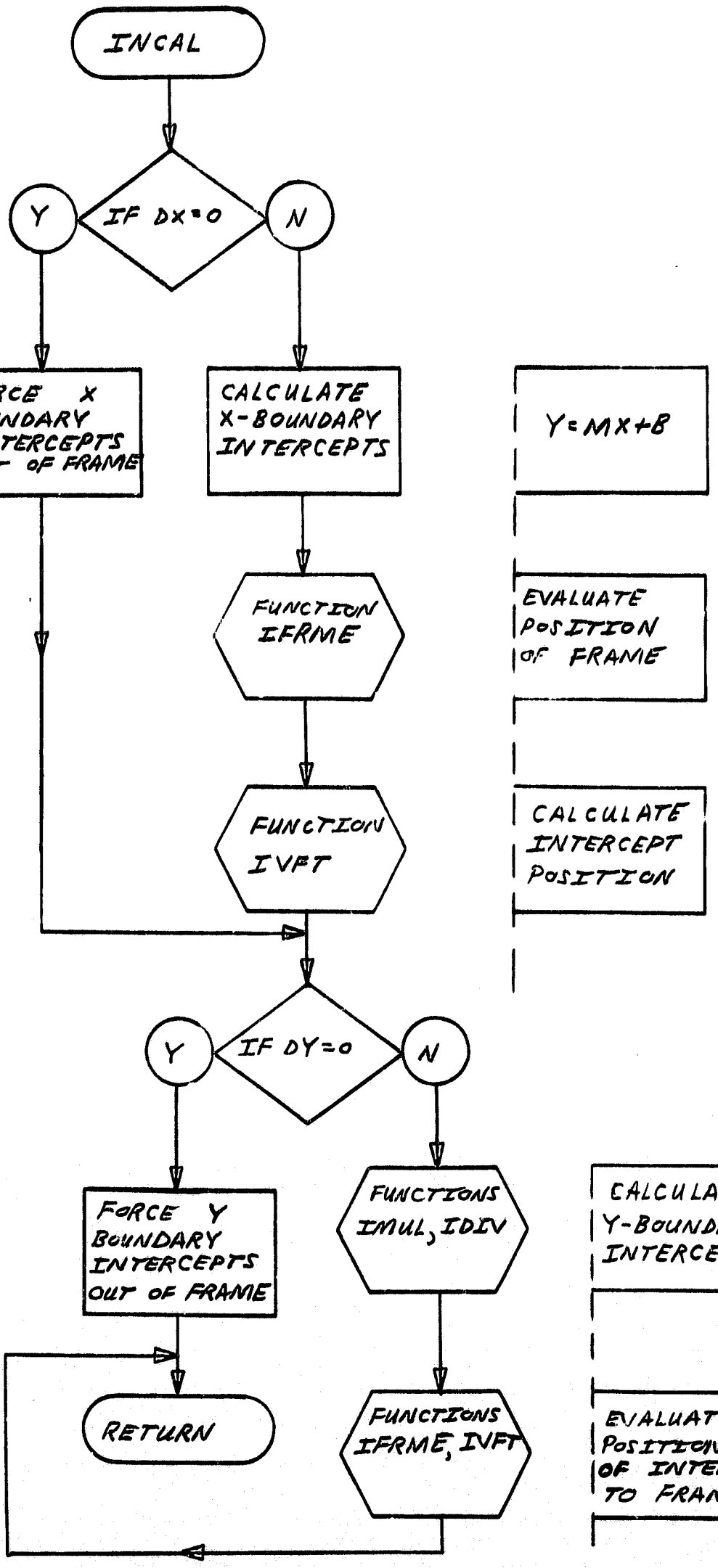
adams associates

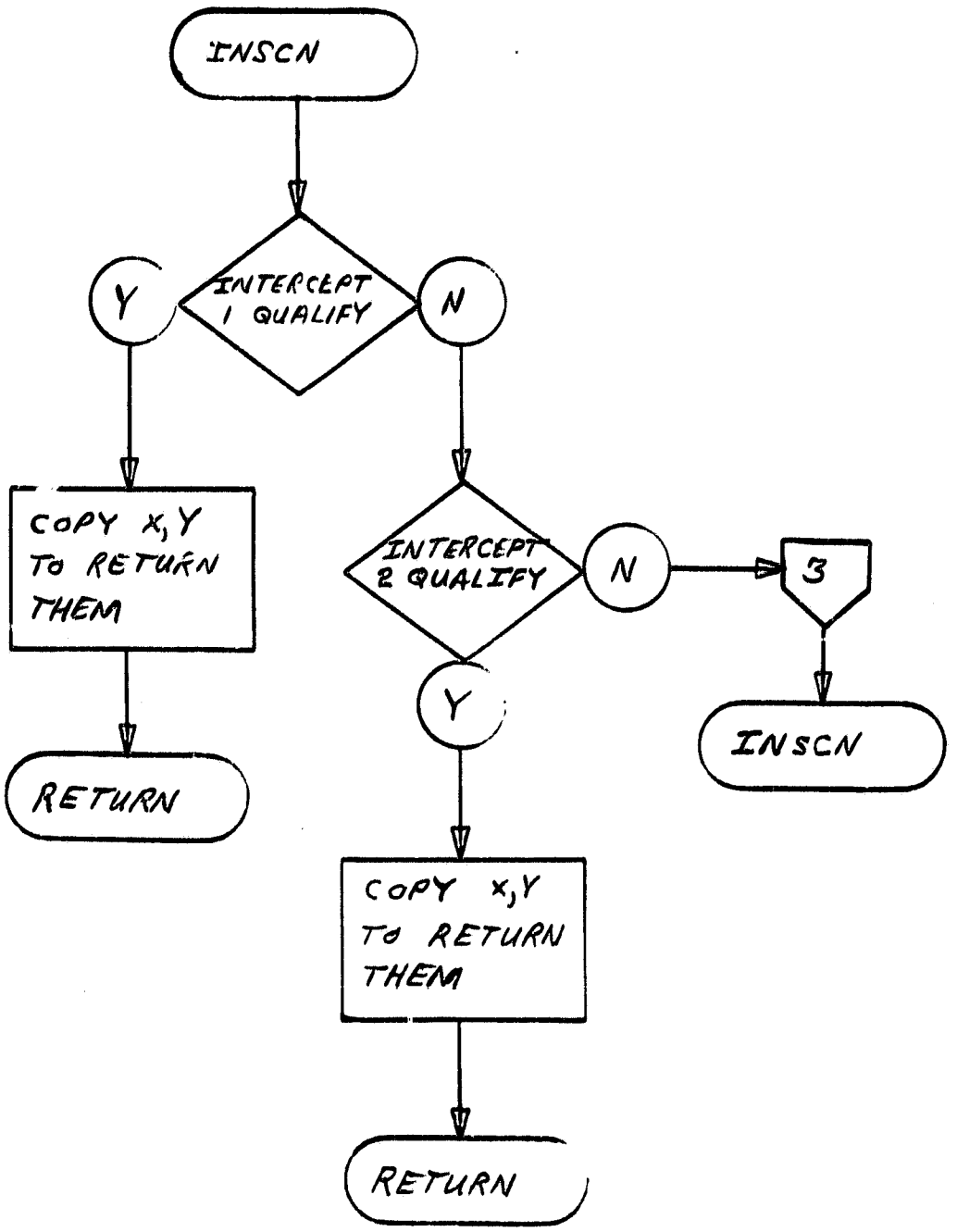
Appendix C

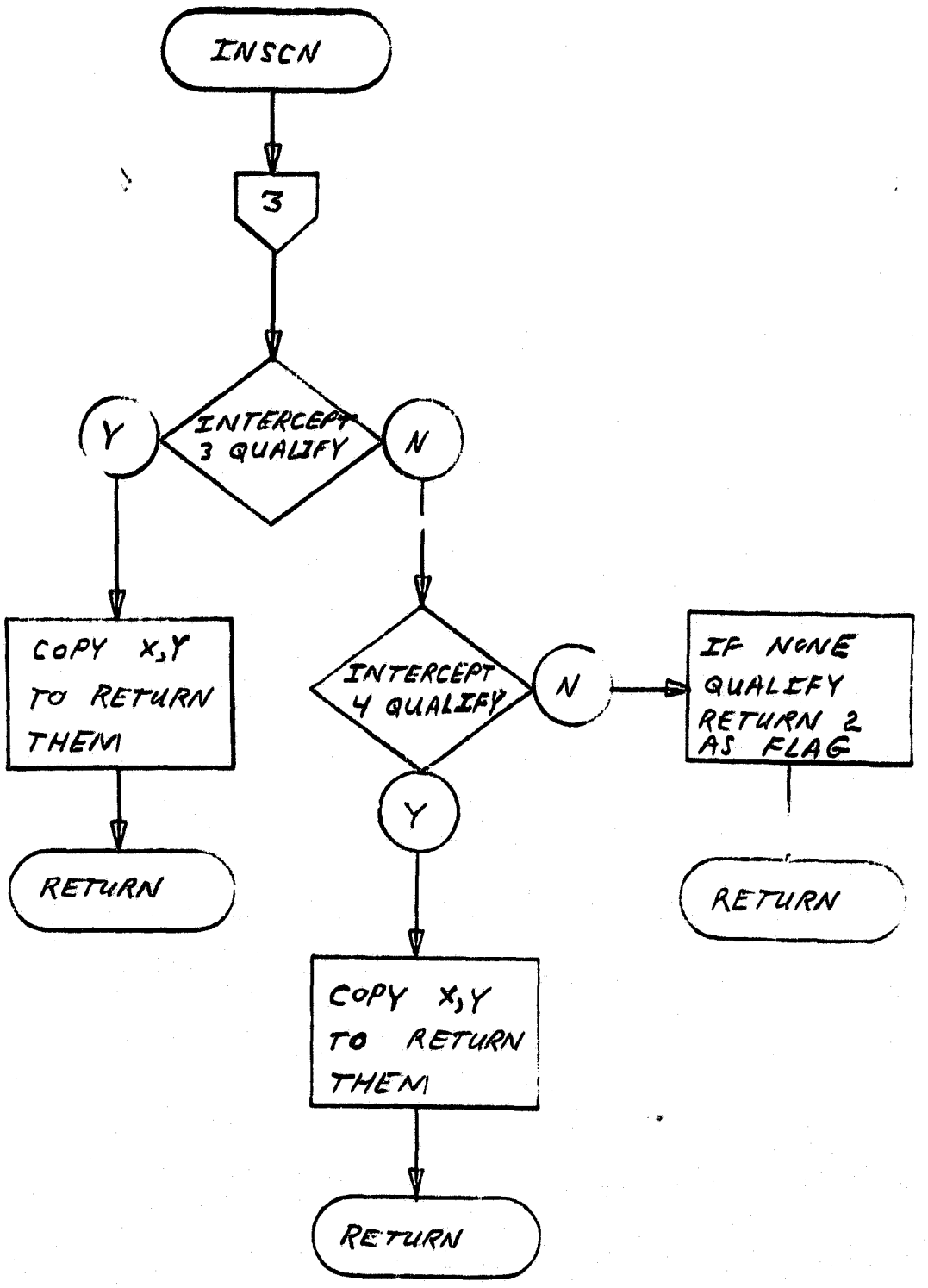
SCISSOR ROUTINE FLOWCHARTS

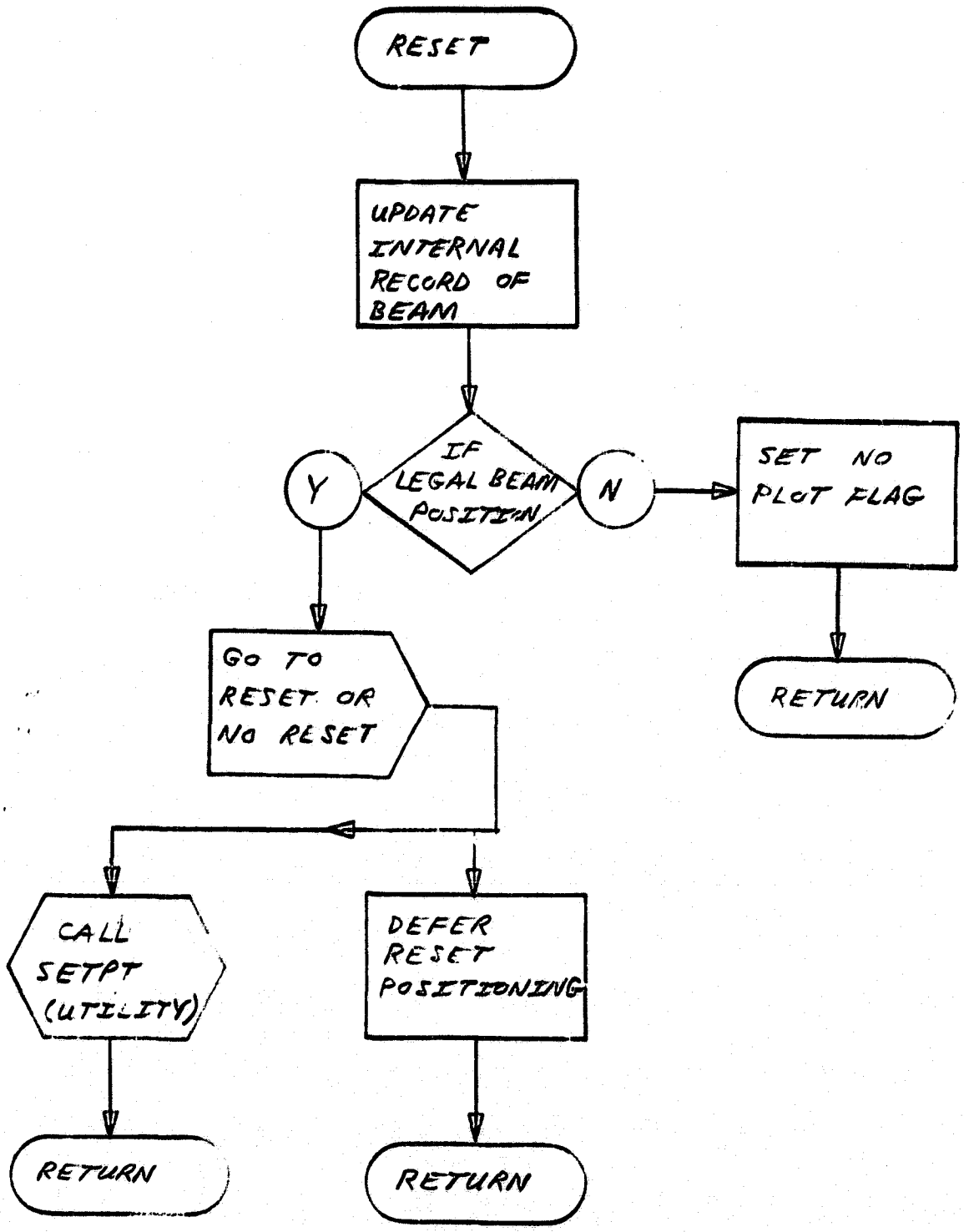

SCISOR
PERFORM IFRM FUNCTION
CALCULATE ZONE NUMBER OF VECTOR HEAD & FOOT
IF VECTOR IN ONE ZONE
Y
N
IF VECTOR LEAVING
N
A
Y
SCISOR
IF FRAME ZONE
Y
N
CALL INCAL
SET PLOT FLAG
SET NO PLOT FLAG
CALL INSCN
3
UPDATE BEAM POSITION
RETURN

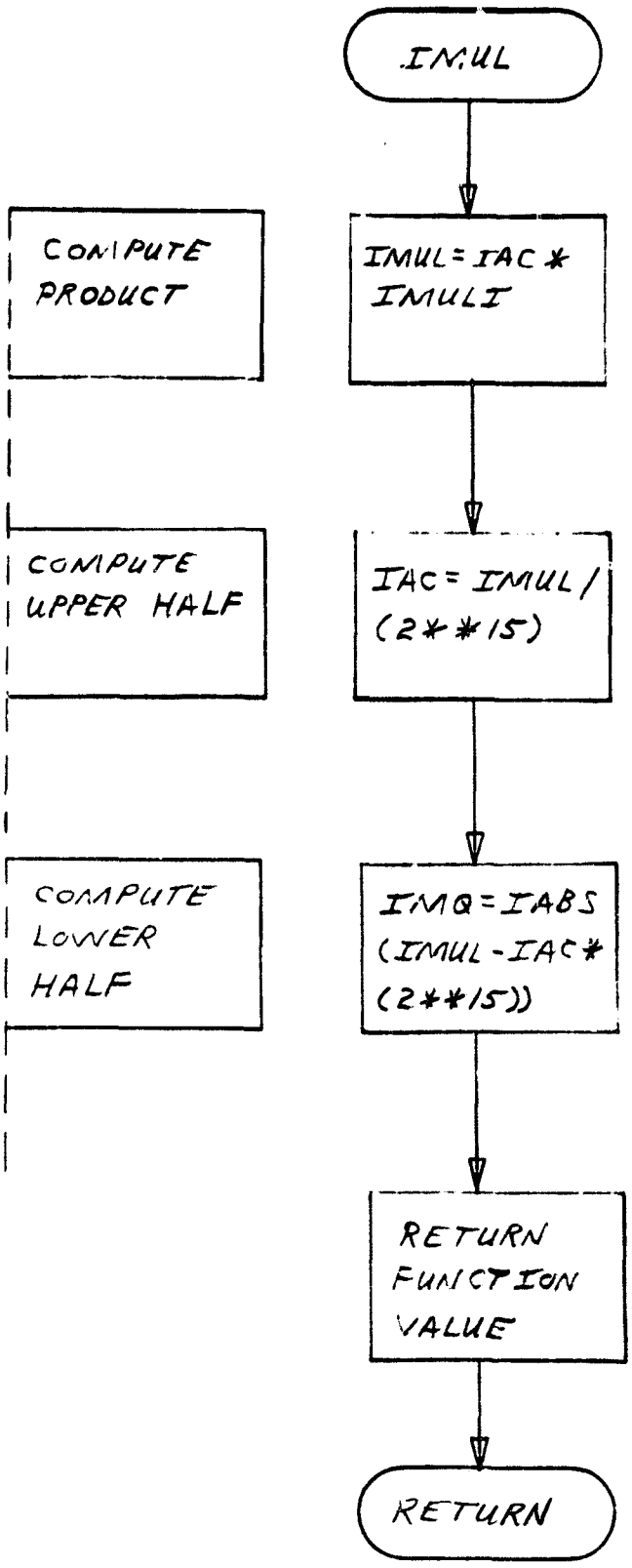

SCISOR
A
IF FRAME ENTERING
Y
N
CALL INCAL
CALL INCAL
CALL INSCN
CALL INSCN
GO TO FOUND BEAM OR NOT
9
SCISOR
GO TO PLOT OR NO-PLOT
61
3
SCISOR
RESET THE BEAM

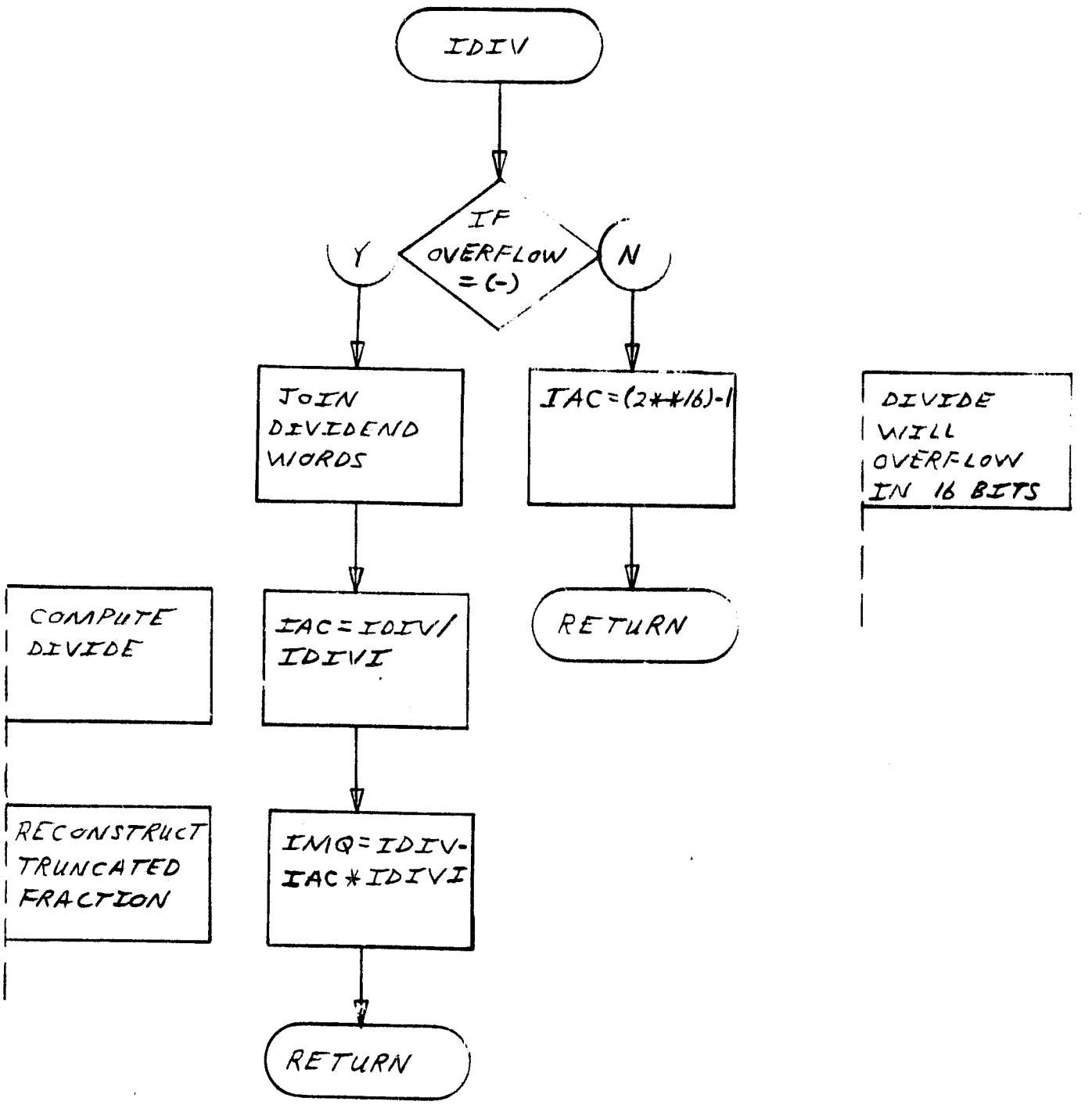


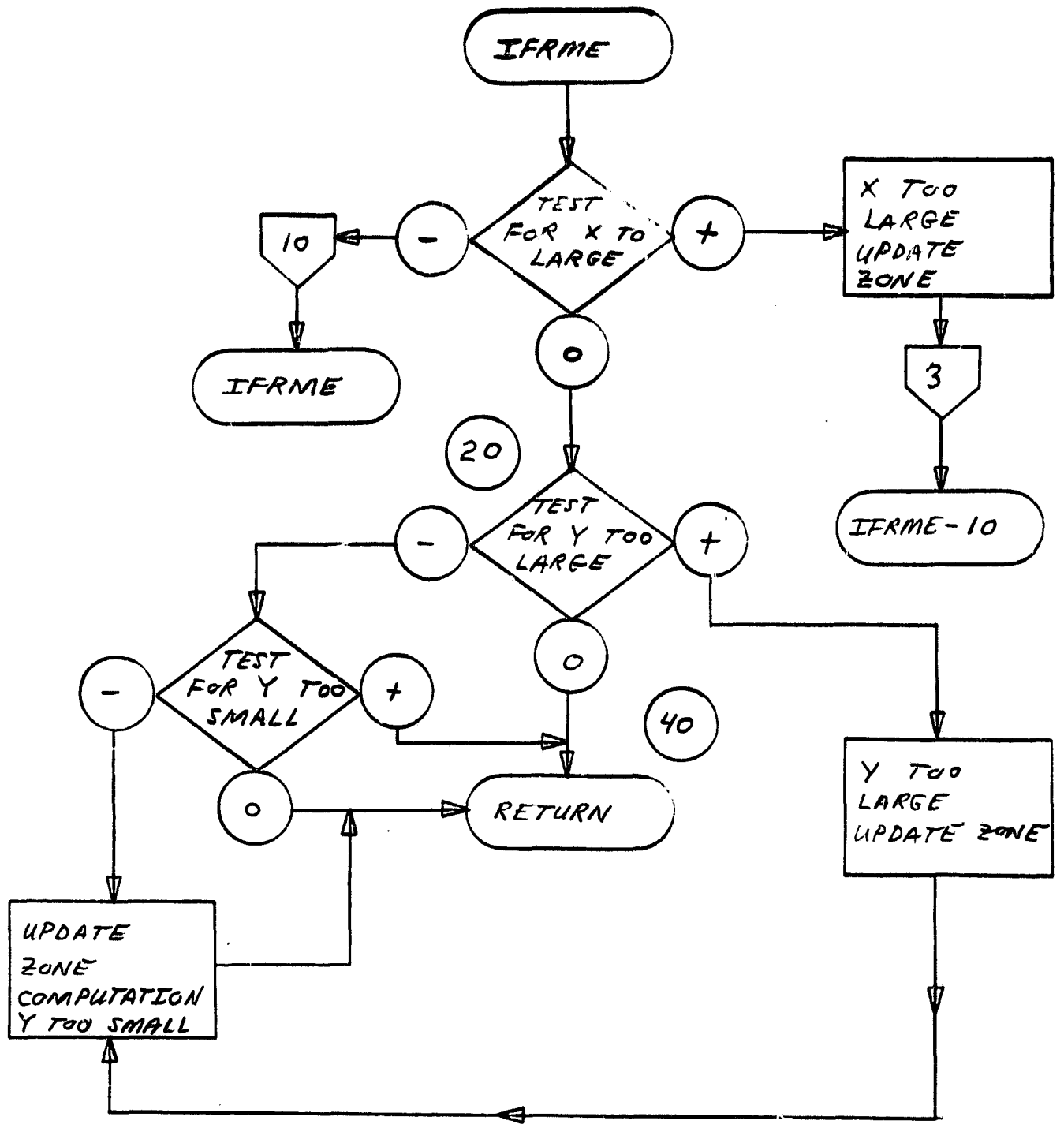

INSCN
INTERCEPT 1 QUALIFY
Y
N
COPY X,Y TO RETURN THEM
RETURN
INTERCEPT 2 QUALIFY
N
3
INSCN
Y
COPY X,Y TO RETURN THEM
RETURN

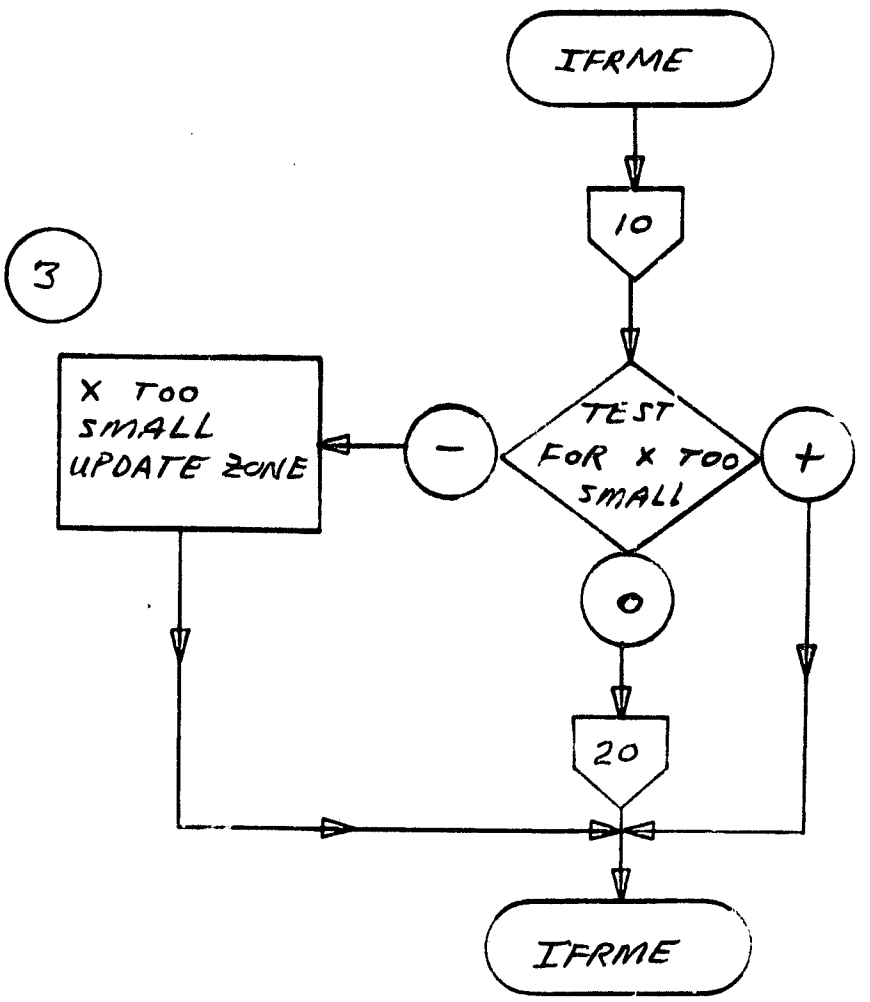

INSCN
3
INTERCEPT 3 QUALIFY
Y
N
COPY X,Y TO RETURN THEM
RETURN
INTERCEPT 4 QUALIFY
N
IF NONE QUALIFY RETURN 2 AS FLAG
Y
RETURN
COPY X,Y TO RETURN THEM
RETURN

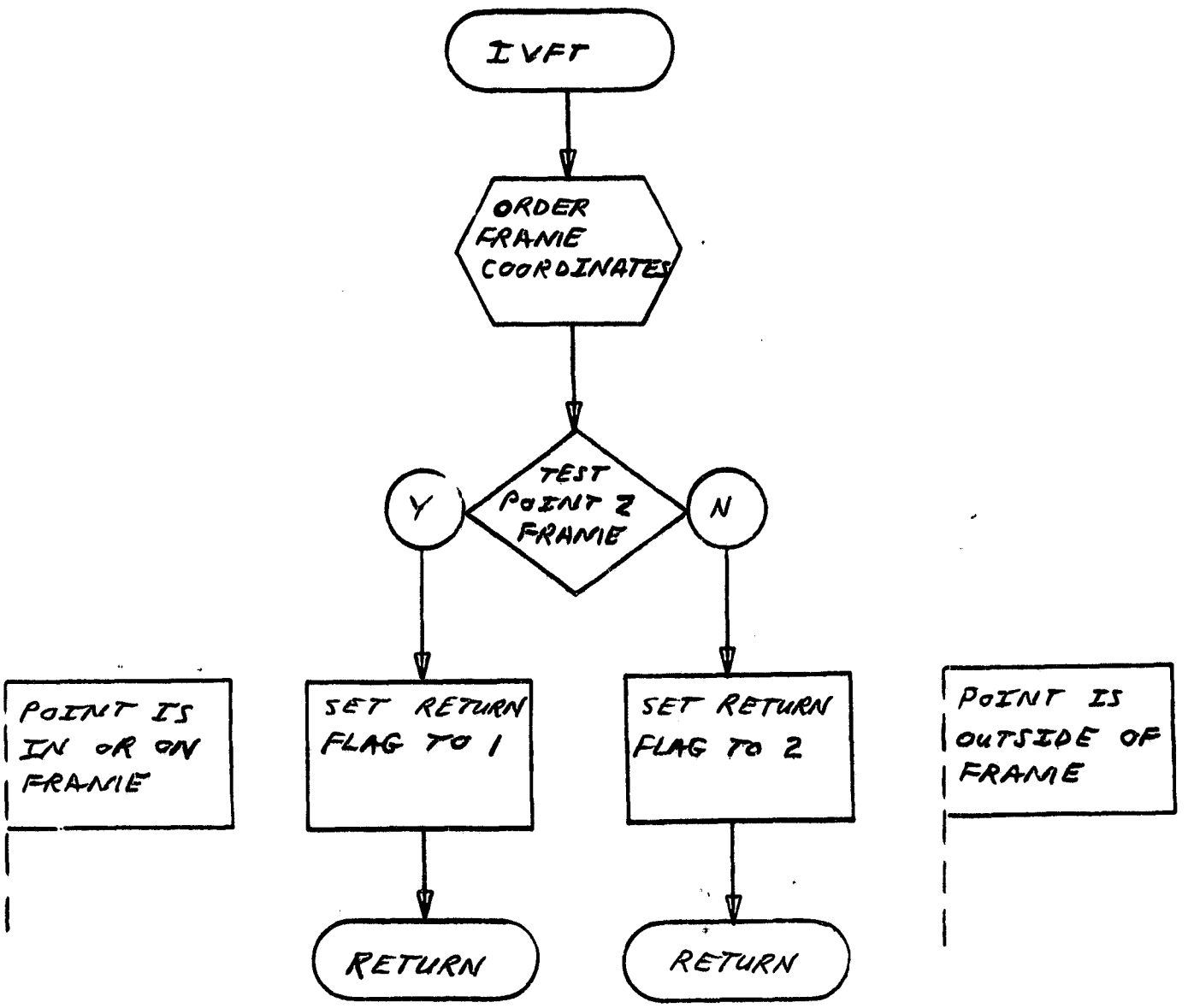

RESET
UPDATE INTERNAL RECORD OF BEAM
IF LEGAL BEAM POSITION
Y
N
SET NO PLOT FLAG
RETURN
Go TO RESET OR NO RESET
CALL SETPT (UTILITY)
DEFER RESET POSITIONING
RETURN
RETURN


IMUL
COMPUTE PRODUCT
IMUL=IAC*IMULI
COMPUTE UPPER HALF
IAC=IMUL/(2**15)
COMPUTE LOWER HALF
IMQ=IABS(IMUL-IAC*(2**15))
RETURN FUNCTION VALUE
RETURN


IDIV
IF OVERFLOW =(-)
Y
N
JOIN DIVIDEND WORDS
IAC=(2**16)-1
DIVIDE WILL OVERFLOW IN 16 BITS
COMPUTE DIVIDE
IAC=IDIV/IDIVI
RETURN
RECONSTRUCT TRUNCATED FRACTION
IMQ=IDIV-IAC*IDIVI
RETURN


IFRME
TEST FOR X TO LARGE
-
+
0
10
IFRME
X TOO LARGE UPDATE ZONE
3
IFRME-10
20
TEST FOR Y TOO LARGE
-
+
0
TEST FOR Y TOO SMALL
-
+
0
40
RETURN
Y TOO LARGE UPDATE ZONE
UPDATE ZONE COMPUTATION Y TOO SMALL


IFRME
10
3
X TOO SMALL UPDATE ZONE
−
TEST FOR X TOO SMALL
+
0
20
IFRME


IVFT
ORDER FRAME COORDINATES
TEST POINT 2 FRAME
Y
N
POINT IS IN OR ON FRAME
SET RETURN FLAG TO 1
SET RETURN FLAG TO 2
POINT IS OUTSIDE OF FRAME
RETURN
RETURN