CUT

8.1.23. This subroutine checks that the value held in accumulator lies between 0 and 1023, setting negative values to zero and values above 1023 to that figure.

UPDATE

8.1.24. This subroutine updates the display file following the deletion of a copper item.

GETRUT

8.1.25. This subroutine first gets the relative pointer to the route, RUTPTR and then calls DESTRY to read the route destructively into BUFF.

8.2. ZROUTE

For implementation on the PDP.15 (and 905/928) ZROUTE has been modified, made more modular, and, it is hoped, made faster. The program has been split into 3 basic programs called ZROUT1, ZROUT2 and ZROUT3. The functions of these programs are:-

ZROUT1 : Sets up the ZROUTE data structure and stores all pads in this data structure. Sorts the connections into increasing length order (length on a basis of mod x + mod y) and stores all these in the routes array as 5 point routes.

ZROUT2 : Attempts to route the routes in the routes array by scanning the array and using more powerful routing algorithms for each scan.

ZROUT3 : Scans the routes array and marks any segments that would cross any other segments if the original segment's side were swopped. Then it scans the routes array again and places all 'free' segments on side 1 or side 2 so that the number of P.T. holes for each individual route is minimized.

The ZROUTE Data Structure

The board is considered to be covered with a 50 thou grid of imaginary routes : x on side 1 and y on side 2. These routes are stored in the form starting value, length of segment. Thus, when no obstacles are present each one of these virtual routes starts at Ø and is 511 units long. Since the first segment in any particular route must always start at Ø this information is not stored; instead the space that would be used for it is used to store the number of segments in that virtual route. Thus a virtual route with an obstacle at 100 units would be stored as:-

| (Ø) | |
|---|---|
| 2 | 1ØØ |
| 1ØØ | 411 |

In the data structure these are packed as two 9 bit numbers per word. The next virtual route is (similarly) stored consecutively in the array.

The first part of ZROUTE (ZROUT1) simply stores pads as obstacles in the arrays representing the x and y virtual routes. Maintenance routines for accessing these arrays are FINDXY, used to find the appropriate virtual route, and REPAKX and REPAKY which create holes of 1 word in either the x or y (respectively) virtual routes.

8.2.1. ZROUTE Core Allocation and Organisation

Phase 1

The routes array is set up with routes sorted in order of length and seven words per route. Pointers to the routes array are inserted in the connections array. ZROUT1 is used for this.

The connections array is dumped to disc 1 (DAT + 3) as a binary file ZDUMPA BIN using ZDUMP(1).

Phase 2

X and Y cuts arrays are set up : the Y cuts array from the top of DFILE downwards and the X cuts array in its final position, i.e. 7.5K from the top of COMPON and downwards. Any pads and copper rectangles are entered into the X and/or Y cuts arrays. ZROUT1 carries out these operations.

The data structure (except ROUTES) is dumped as ZDUMPB BIN using ZDUMP(2).

Phase 3

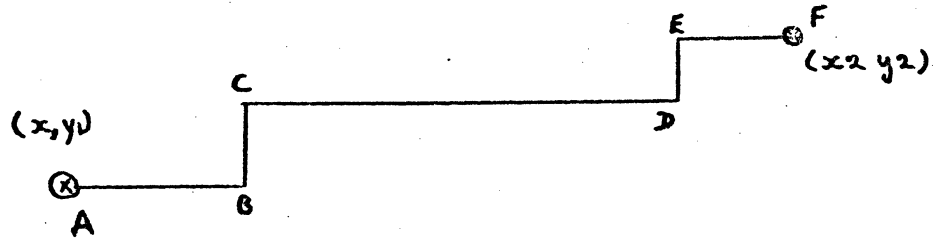
ZROUT2 is used to carry out the routing but it first moves the Y cuts array to a higher address (to the top of COMPON) so that approximately 15K is available for the routing phase.

Phase 4

ZDUMP(3) is used to recall the d.s. that was dumped to disc. ZROUT3 is used to tidy up the routes array and effect p.t.h. reduction.

8.2.2. The Basic Routing Algorithm

ZROUT2 tries to route each route as a 6 point route as shown:-




where AB, CD, EF are on side 1
and BC, DE, are on side 2.

In this description we will assume $X_2 - X_1 > Y_2 - Y_1$. If this is not true swop the sides and for X read Y and vice versa.

On each scan through the routes array the algorithm used is:-

SCAN 1:

AB = EF = $\emptyset$

Various positions for CD are tried to see if this 4 point route is satisfactory. First both single corner routes are tried and then all possible 'inside area' 2 point routes. The first ones tried are for the opposite type of route (CD vertical, on side 2) and if all these fail then all possible 'inside area' routes with CD horizontal are tried. If all fail the program moves to the next route. The first success is accepted by the program as the final route.

For this success the program deletes the segments of route used from the virtual routes array (using routines XSCRUB and YSCRUB) and stores the final route in place of the route as stored by the sorting program.

SCAN 2:

Tries all possible 'inside area' routes subject to the condition that CD must extend from X1 + $\Lambda$ to X2 - $\Lambda$, where $\Lambda = \frac{X2 - X1}{4}$. If all fails it tries all possible 'opposite type' routes (subject to the same conditions).

SCAN 3:

Tries all possible routes subject to the condition as for SCAN 2. However, EF and AB are allowed to lie outside this area. Similarly, if these fail, it tries all possible 'opposite type' routes. The limits imposed on these 'outside area' scans is the allowable routing area of the board.

These scans are illustrated below.

Copper Sharing

To save available routing space it is essential that electrically common routes going to the same pad and lying close to each other should, where possible, be common. In ZROUTE this electrical equivalence is achieved in the following way.

Segments of the form AB, EF:

If virtual copper has been deleted from the ZROUTE D.S. on either side of a pad any other route going to that pad will use that same virtual copper.

Segments of the form BC, DE:

If virtual copper either starts or finishes at B (for example) and AB has also been deleted from the virtual copper array the program will re-use the copper along BC.

Segments of the form CD:

No sharing is attempted unless AB or EF (or both) are zero. In this case sharing as for BC is allowed.

Glossary of subroutines used in ZROUT2:

JOINZ : checks to see if copper has been used from a line to a pad.

SETSWX,Y : looks at each line and sets switches for any special end conditions that may exist.

LUPSPY : checks to see if a virtual route is usable from ZL to ZU. Takes account of switches as set by SETSWX or SETSWY.

LUPSPX : checks to see if a virtual route is clear from ZL + delta to ZU-delta (where delta = (ZU-ZL)/4). Takes account of switches.

YLINES : given the line CD and a pad (A say) tries all segments of the type CB to see if any are suitable. Also used for trying the segments ED. When the route is completed it stores the final

solution in the routes array.

XLINES : as for YLINES.

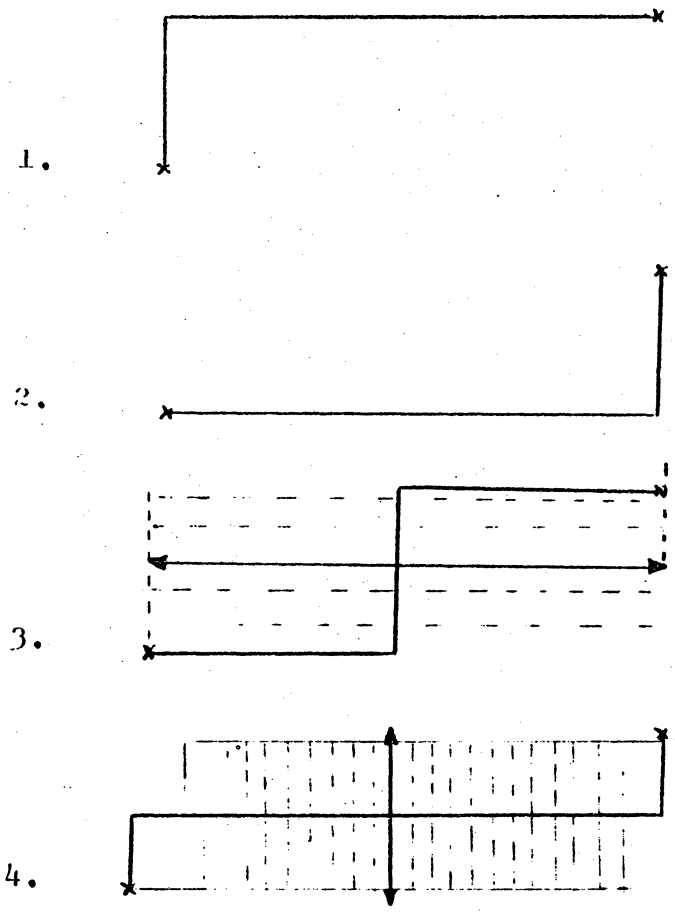
X,YSCRUB : deletes virtual copper from the ZROUTE data structure.

REPAKX,Y : as for ZROUT1.

FINDXY : as for ZROUT1.

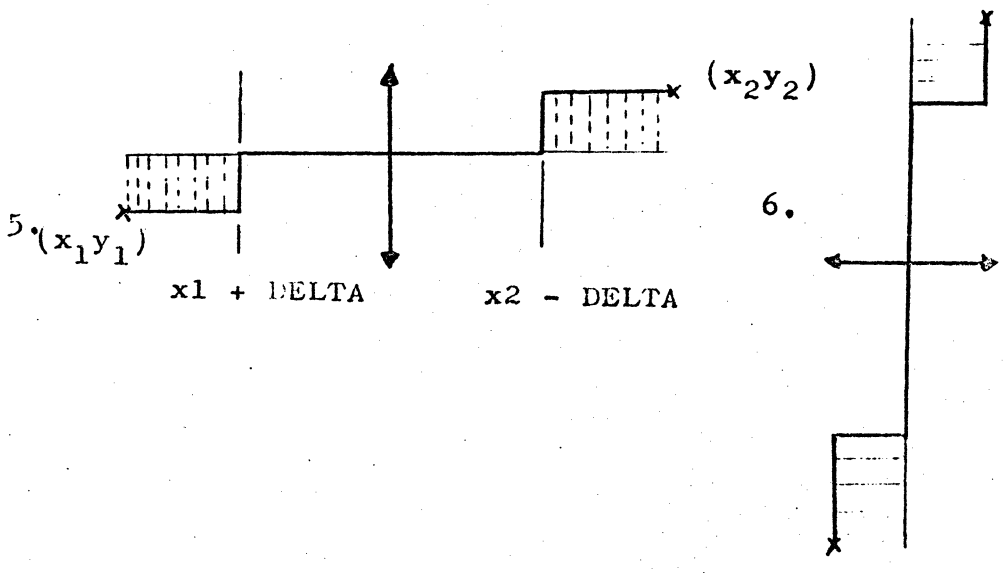
Inside Routing Phase

SCAN1

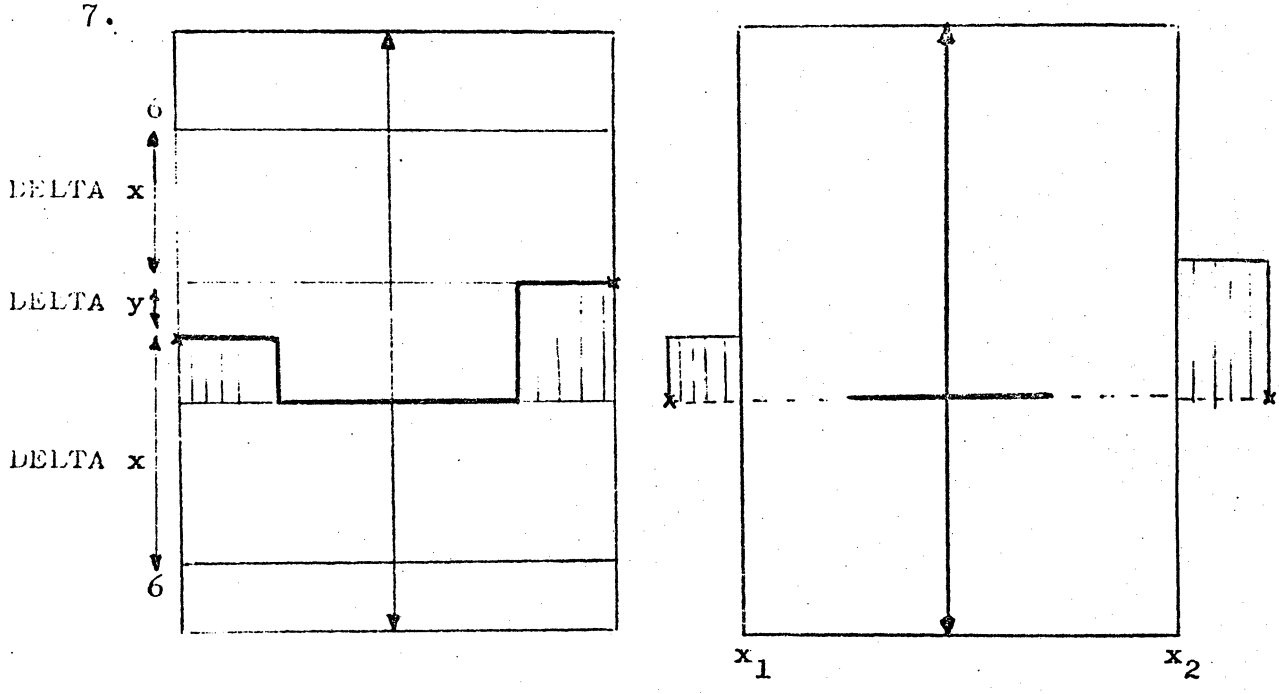



Outside Routing Phase

SCAN2



7.

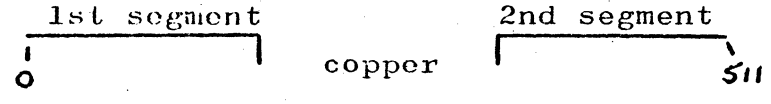



If x cuts segment is clear beyond X1 and X2 the routine tries to route as shown.

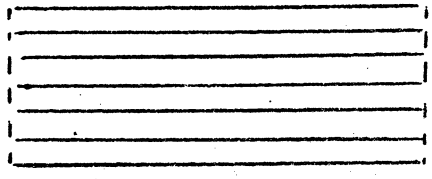
8. Opposite type routes to 7.

### 8.2.3. Storage of Obstacles in the Zroute Data Structure

(a) Copper is stored as a series of cuts each of the following form:-

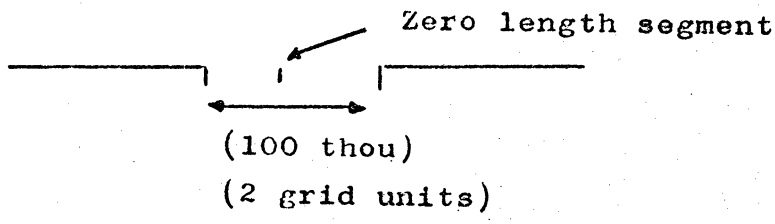



An area of copper on side 1 is indicated by two Y cuts and X cuts are shown below:

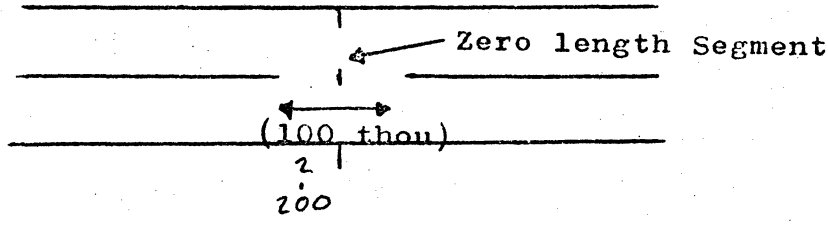


If the copper is on side 2 then the X and Y cuts are reversed.

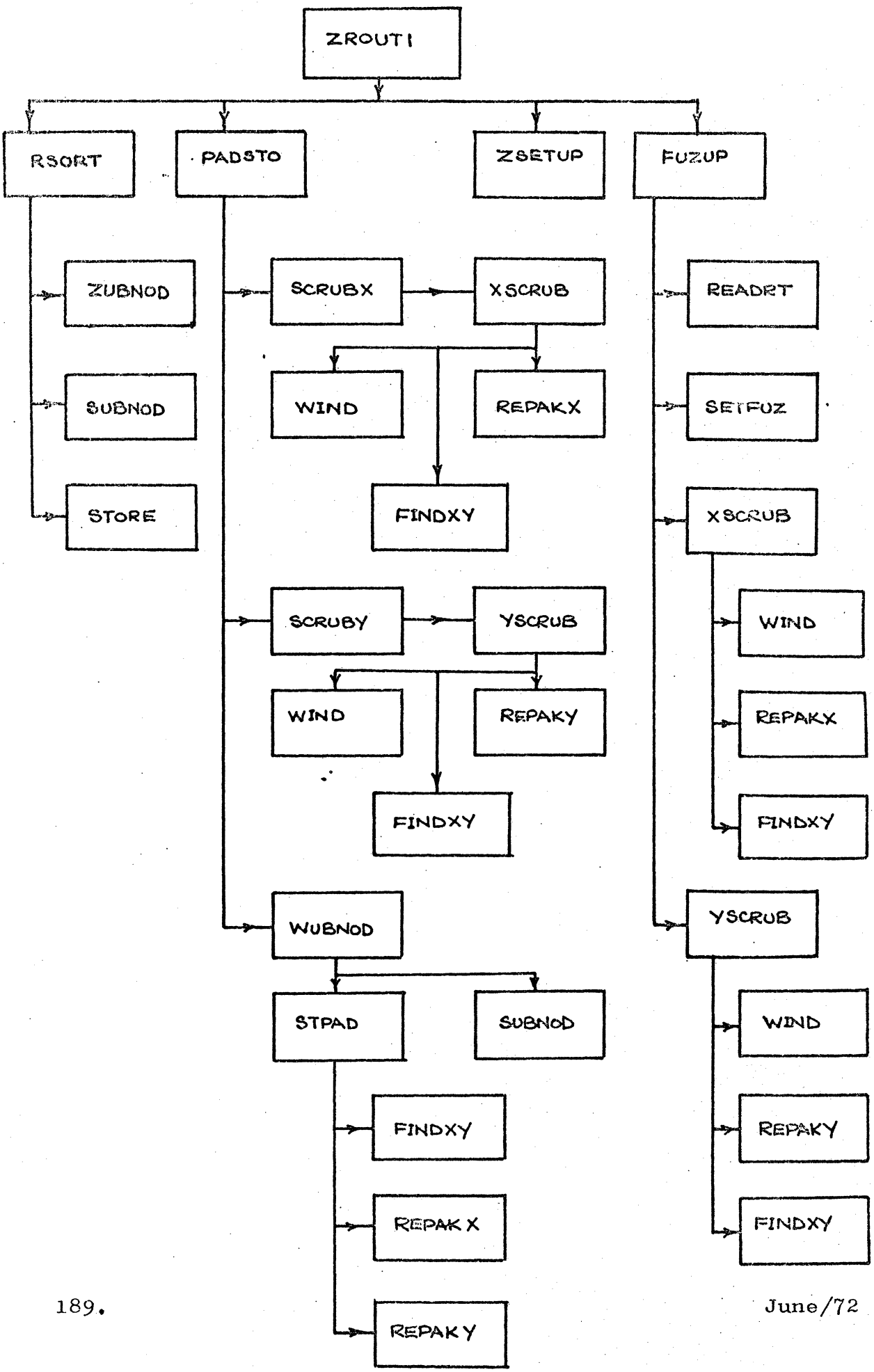
(b) Pads of 50 thou or less are stored as an X and Y cut each of which has three segments as shown belows:-



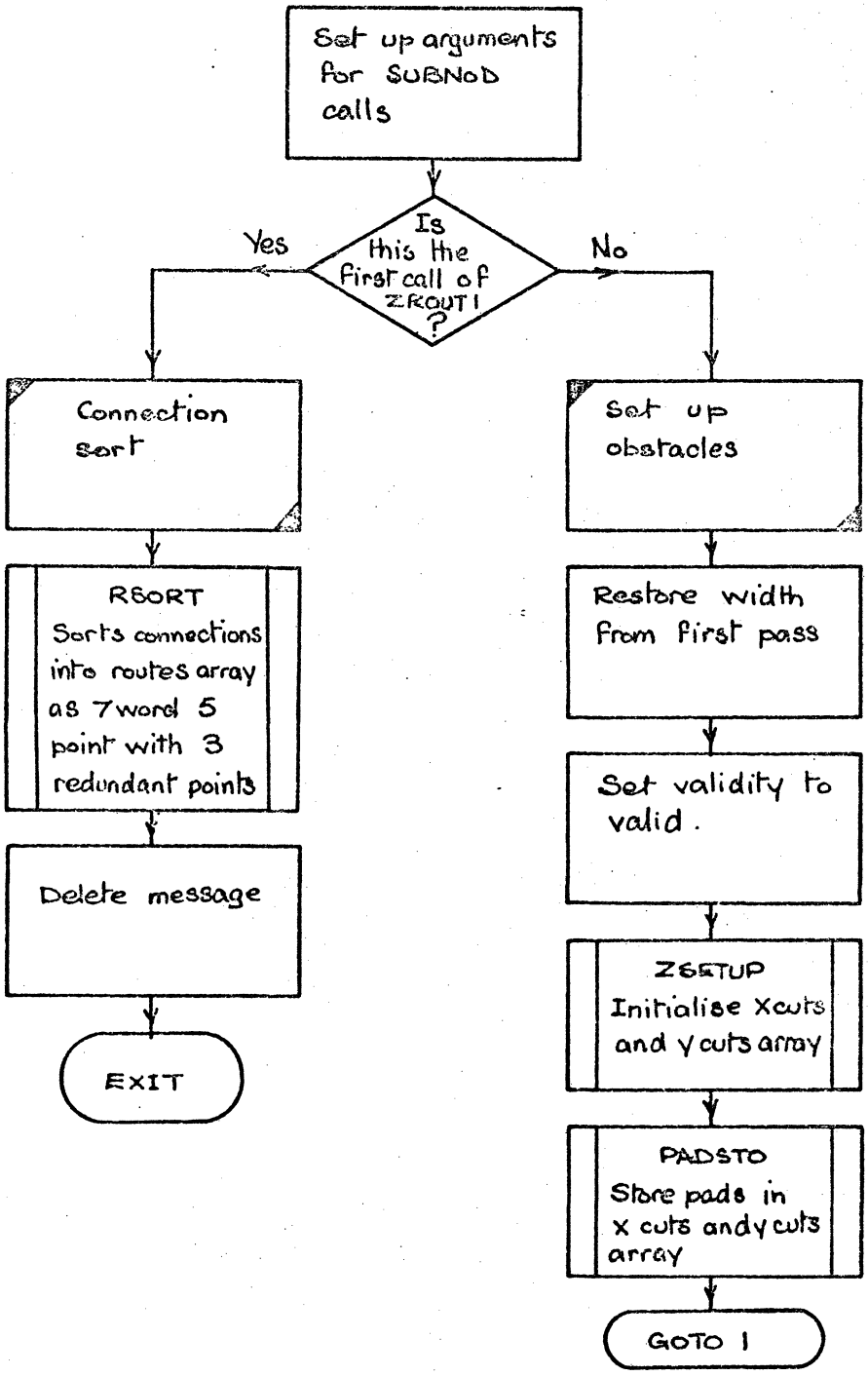

If the pad is larger than 50 thou the X and Y cuts have the following form:-

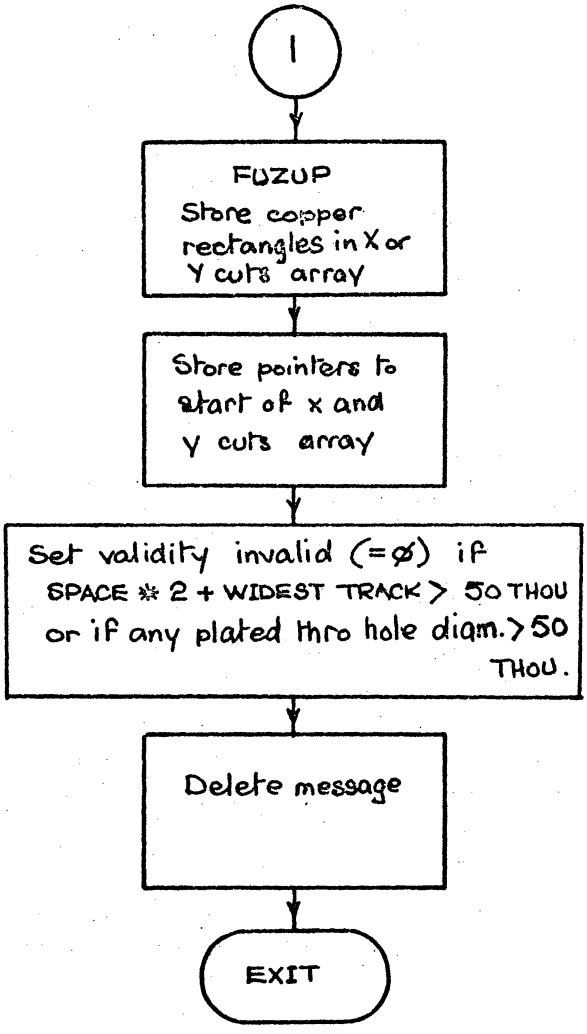


SUBROUTINE ZROUT1 STRUCTURE

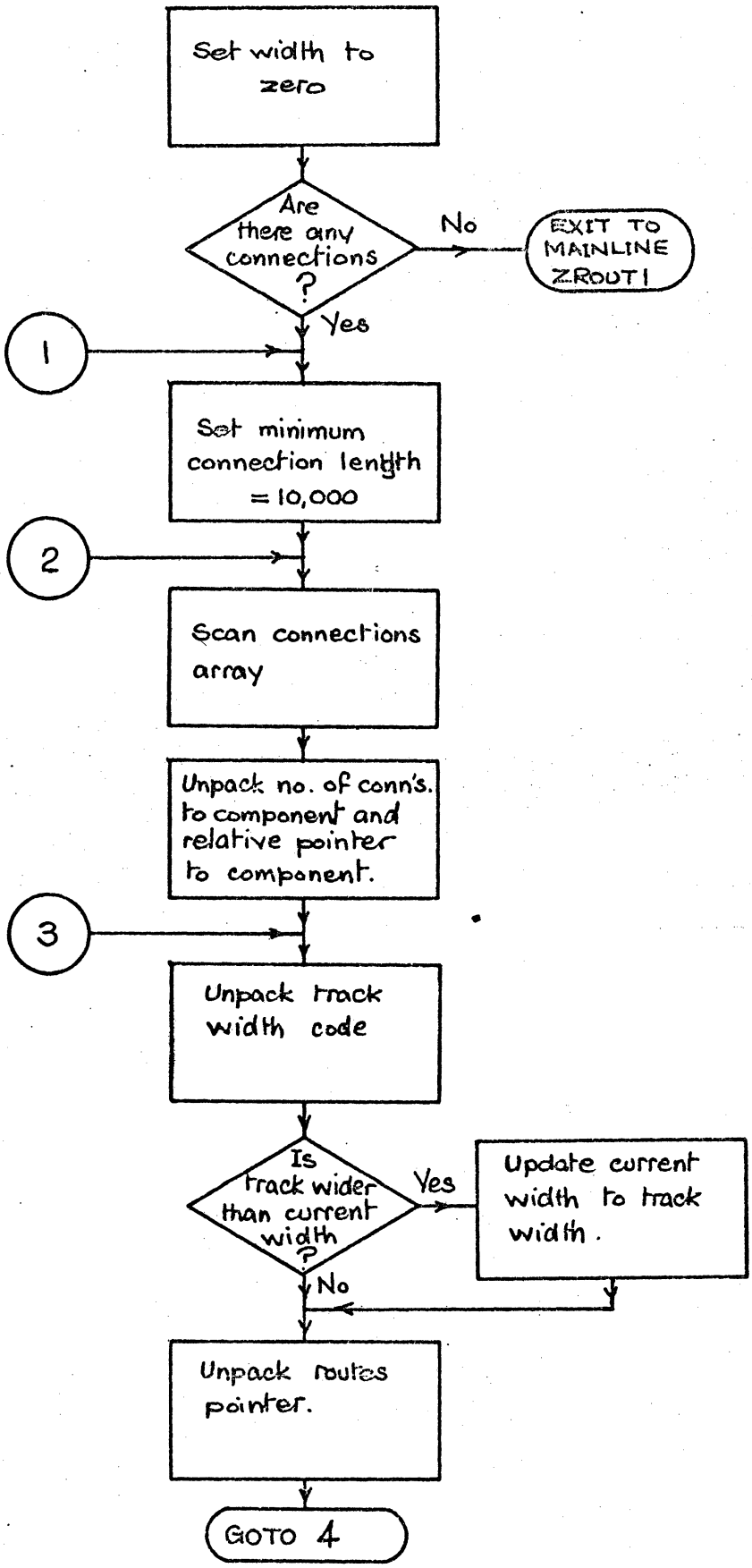


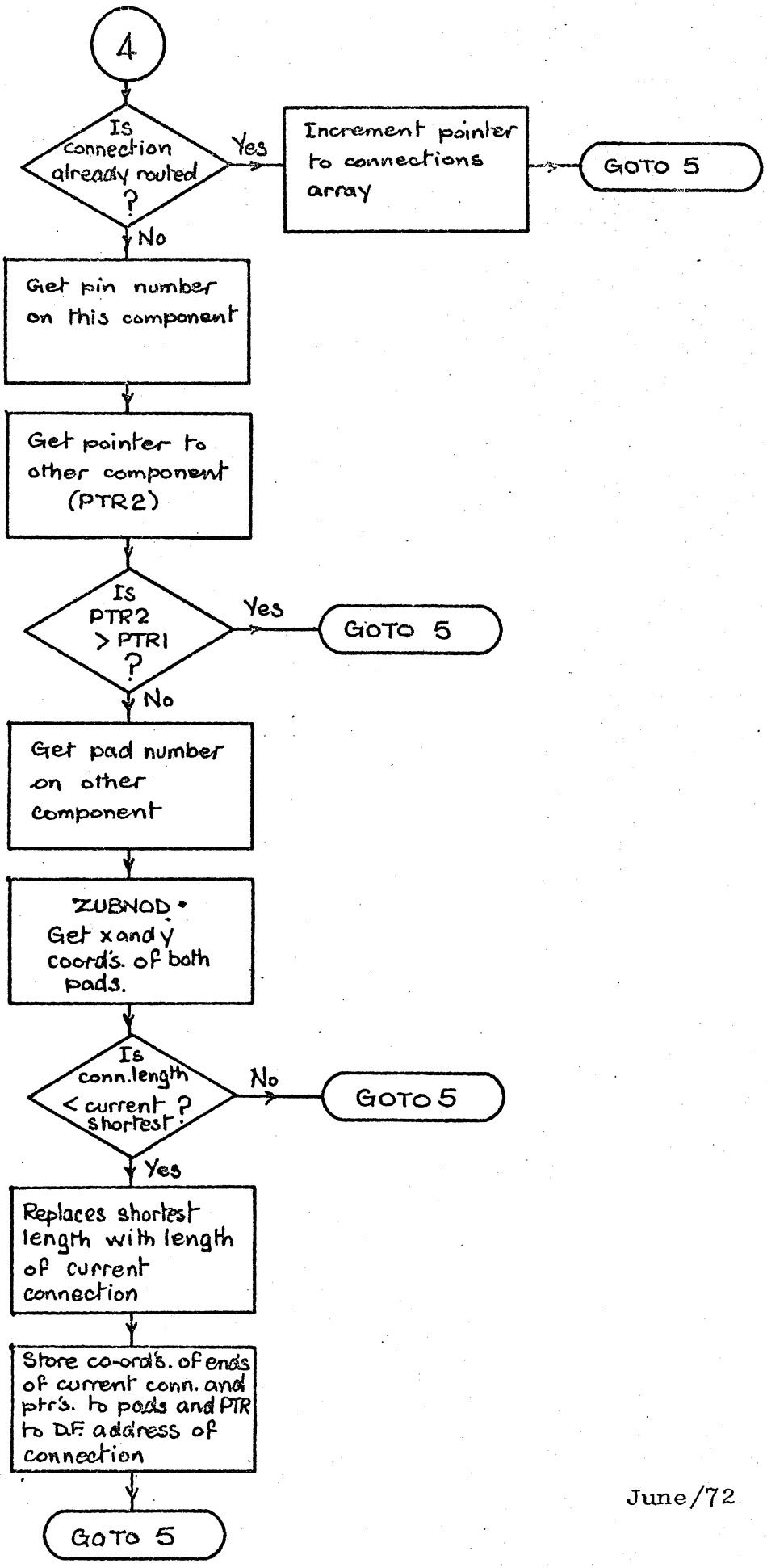
SUBROUTINE ZROUT1 FLOWCHART PART 1 OF 2



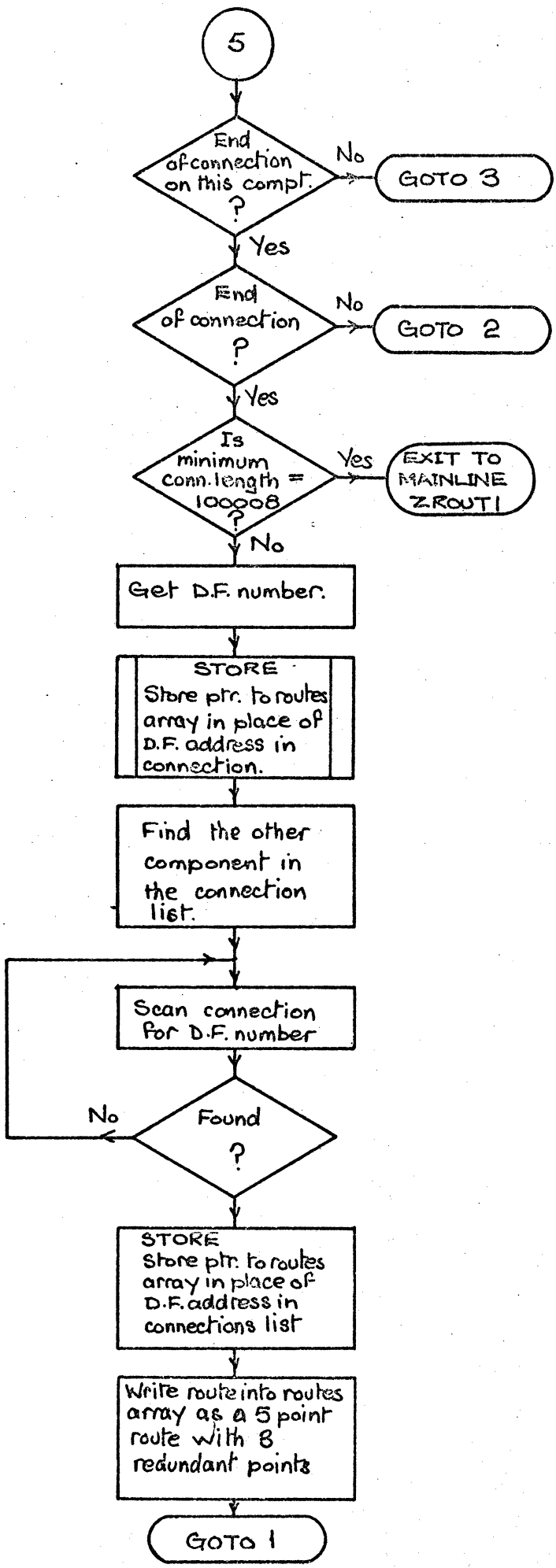

1
FUZUP
Store copper rectangles in X or Y cuts array
Store pointers to start of x and y cuts array
Set validity invalid (=ø) if SPACE * 2 + WIDEST TRACK > 50 THOU or if any plated thro hole diam. > 50 THOU.
Delete message
EXIT

SUBROUTINE RSORT FLOWCHART PART 1 OF 3

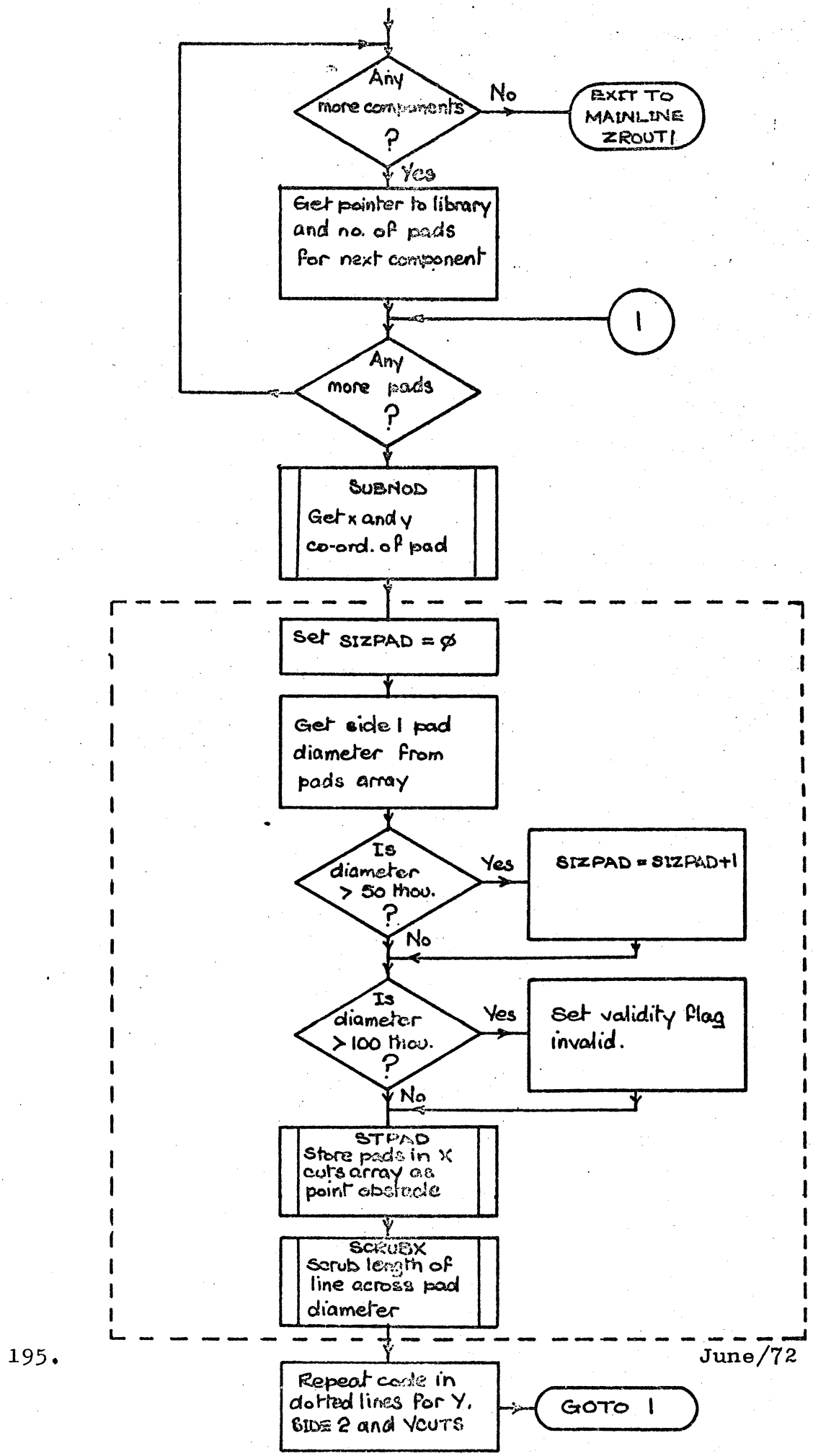



4
Is connection already routed ?
Yes
Increment pointer to connections array
GOTO 5
No
Get pin number on this component
Get pointer to other component (PTR2)
Is PTR2 > PTR1 ?
Yes
GOTO 5
No
Get pad number on other component
'ZUBNOD' Get x and y coord's. of both pads.
Is conn.length < current shortest ?
No
GOTO 5
Yes
Replaces shortest length with length of current connection
Store co-ord's. of ends of current conn. and ptr's. to pads and PTR to DF address of connection
GOTO 5

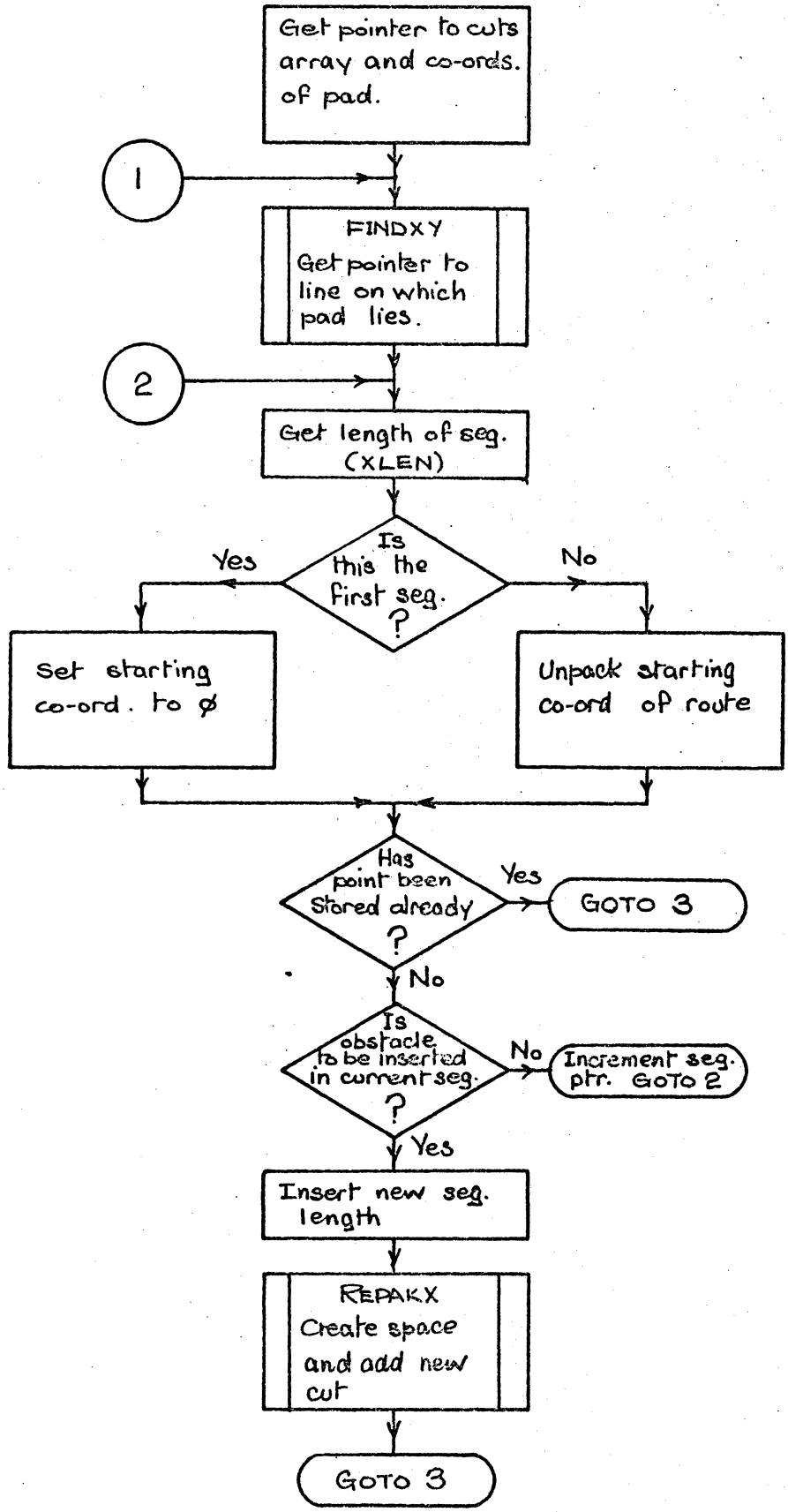
SUBROUTINE RSORT FLOWCHART PART 3 OF 3



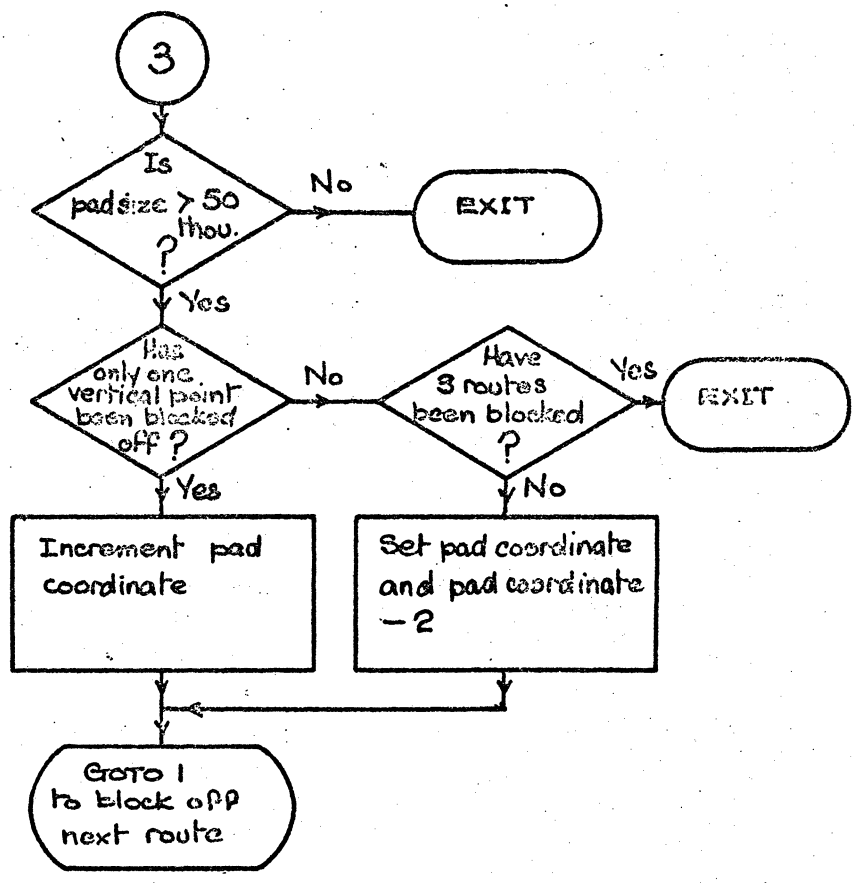
SUBROUTINES PADSTO AND WUBNOD FLOWCHART



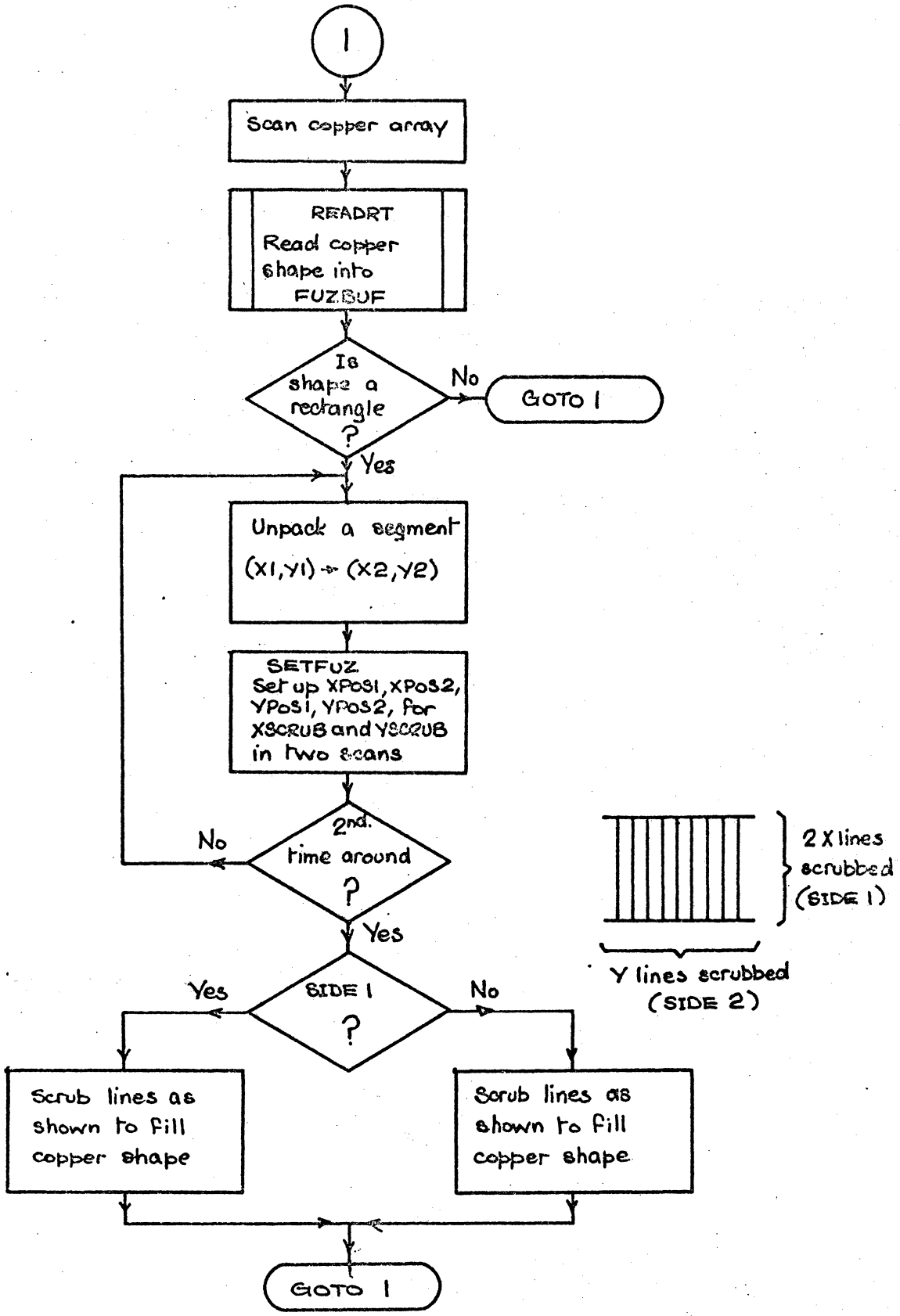
SUBROUTINE STPAD FLOWCHART PART 1 OF 2



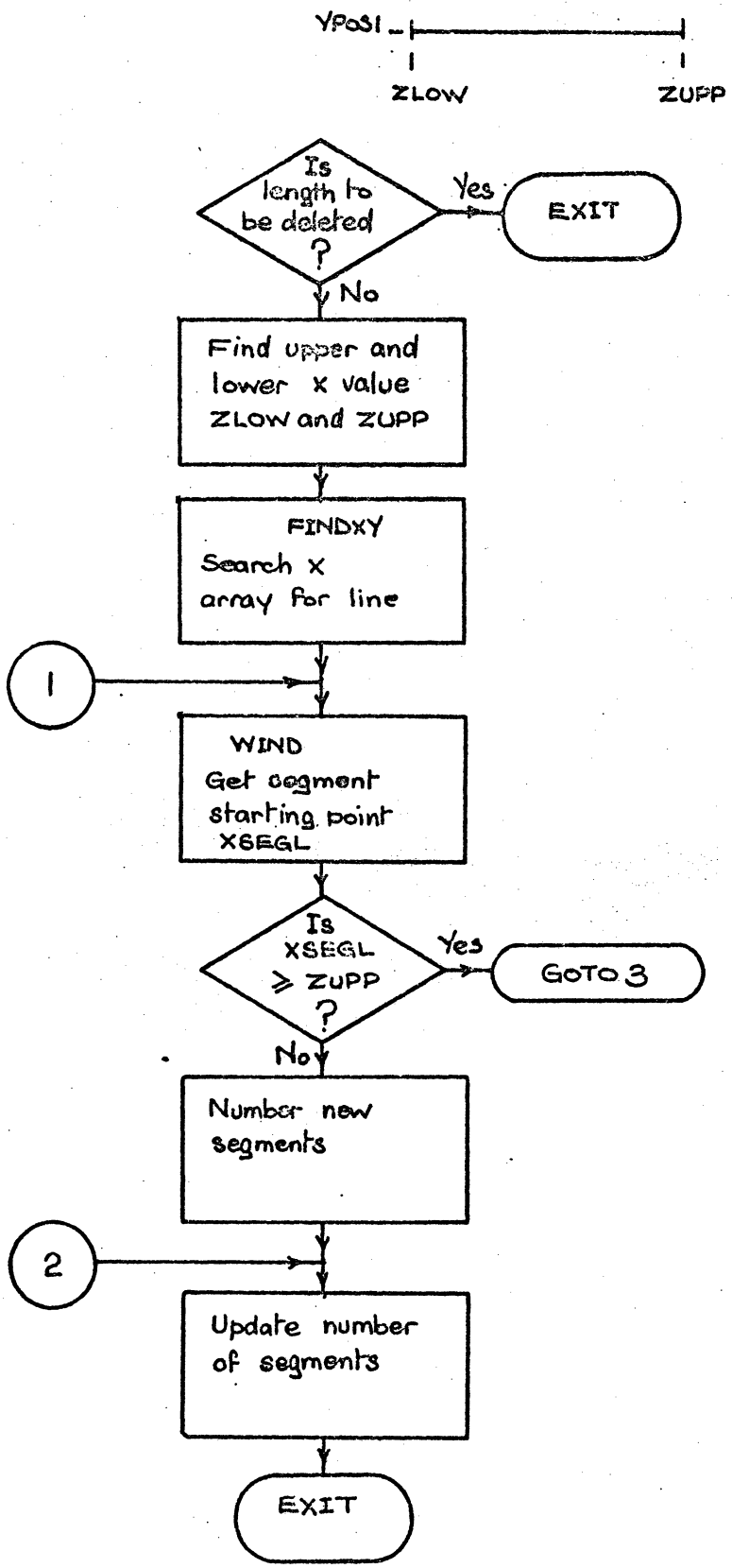
SUBROUTINE STPAD FLOWCHART PART 2 OF 2



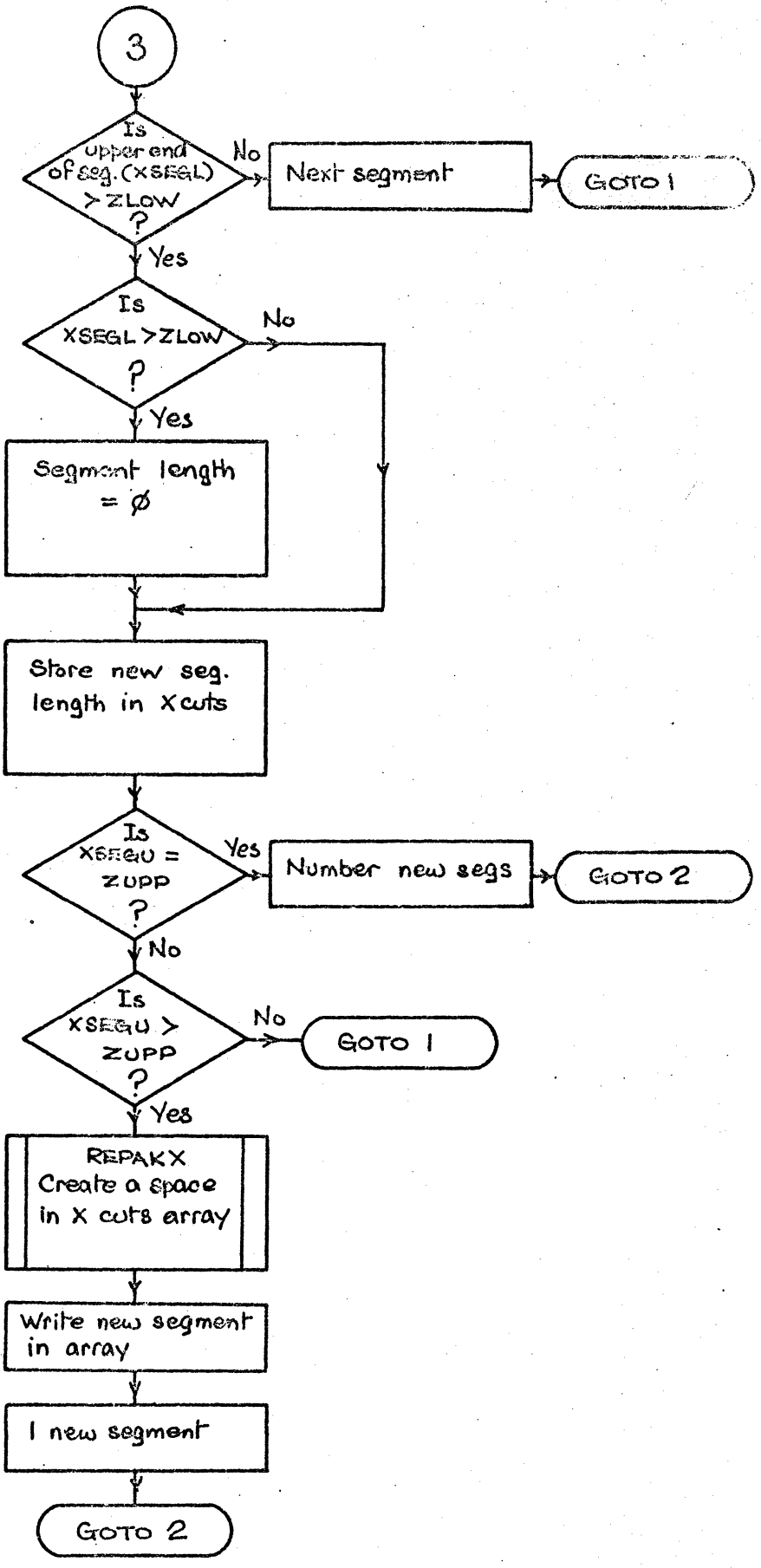
SUBROUTINE TUZUP FLOWCHART



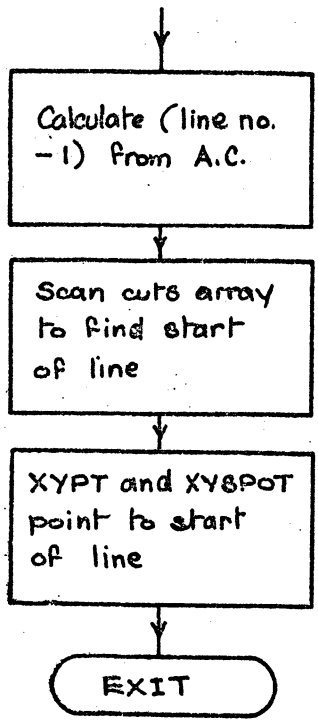
SUBROUTINE XSCRUB FLOWCHART PART 1 OF 2



SUBROUTINE XSCRUB FLOWCHART PART 2 OF 2



SUBROUTINE FINDXY FLOWCHART




On entry :-

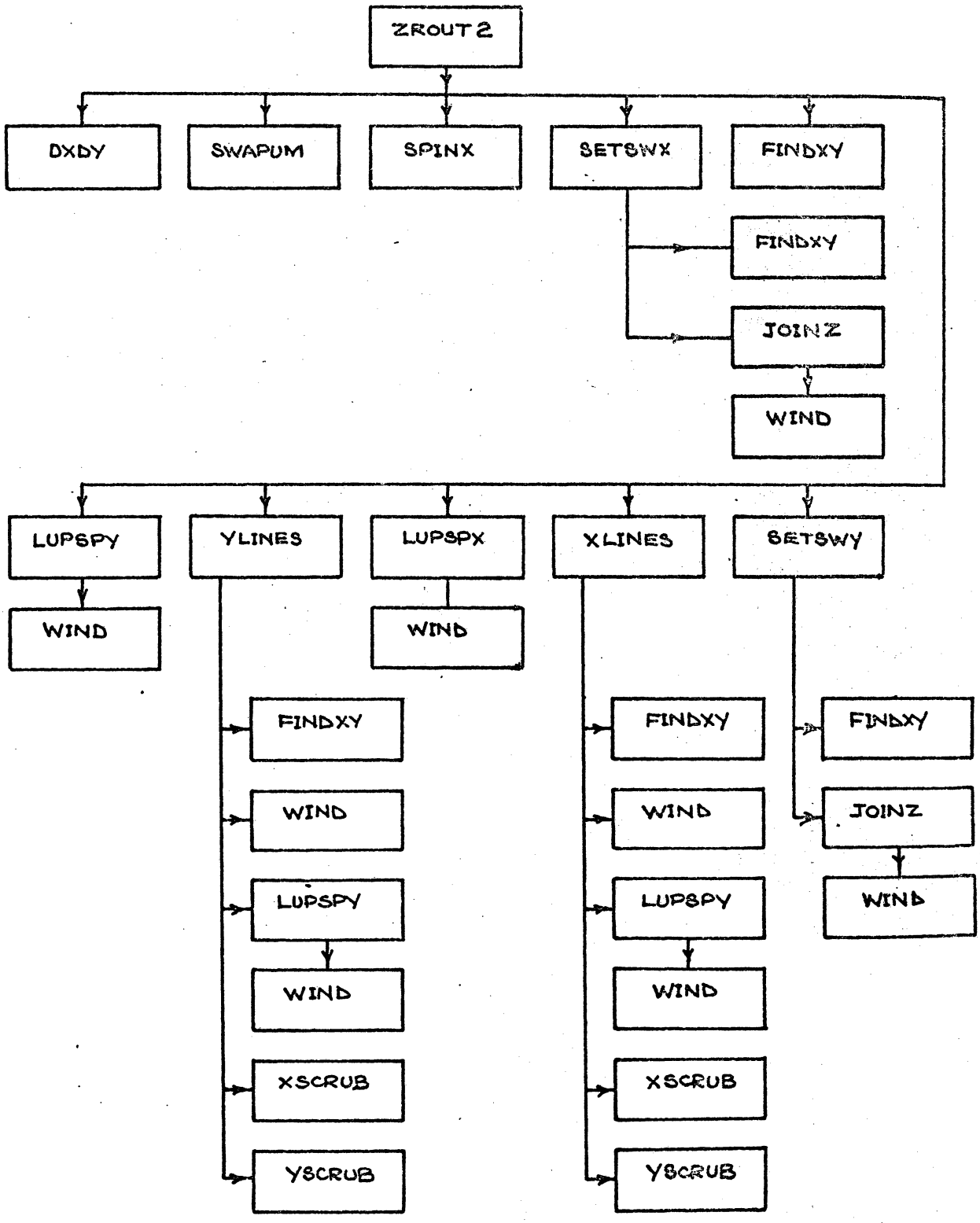
A.C. contains pad coordinate.

MQ contains pointer to X or Y cuts array.

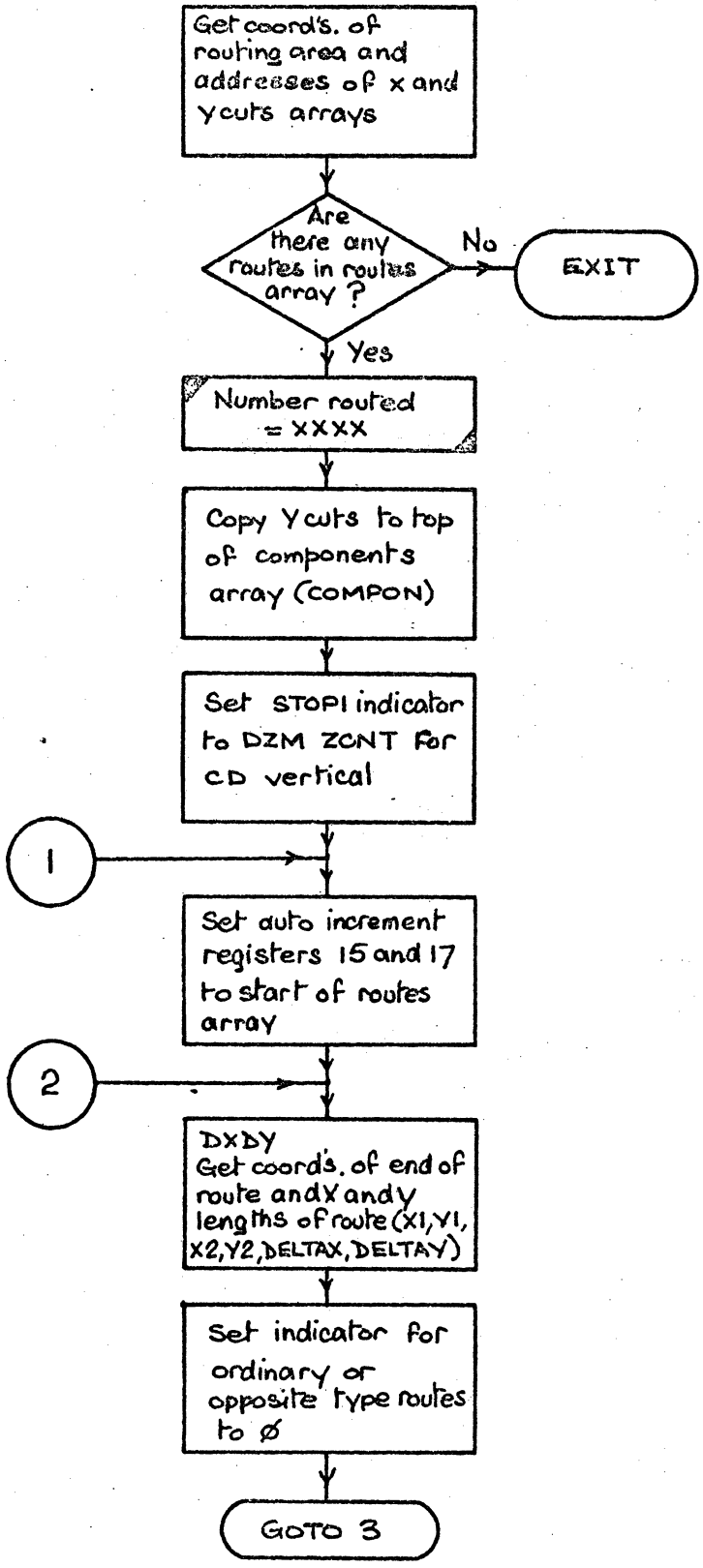
ZSETUP

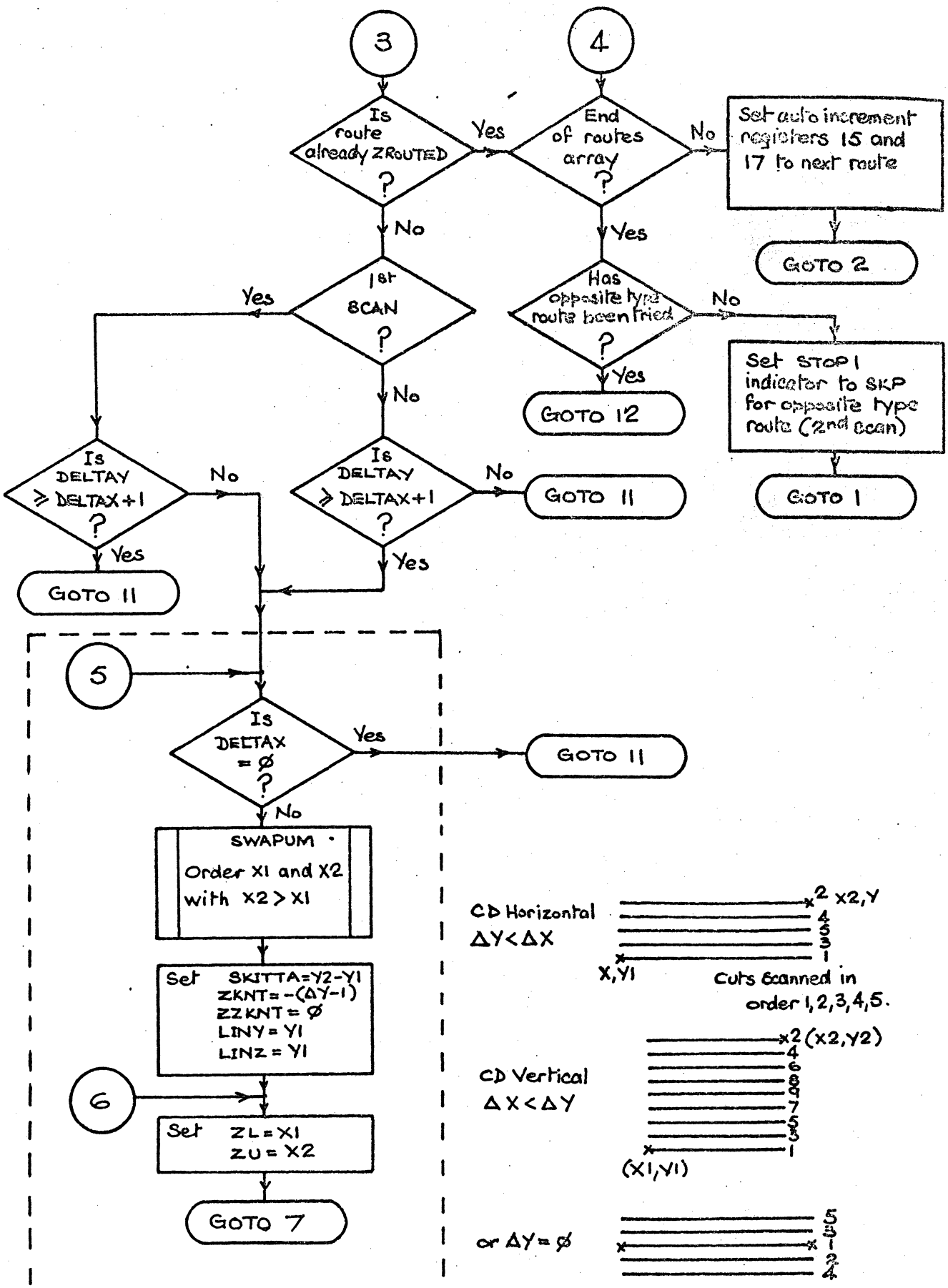
This routine initialises the X and Y cuts arrays by setting the first 512 locations in each to 1777, i.e. 1 segment 511 units long. Pointers to the start of each array are LOCX and LOCY.

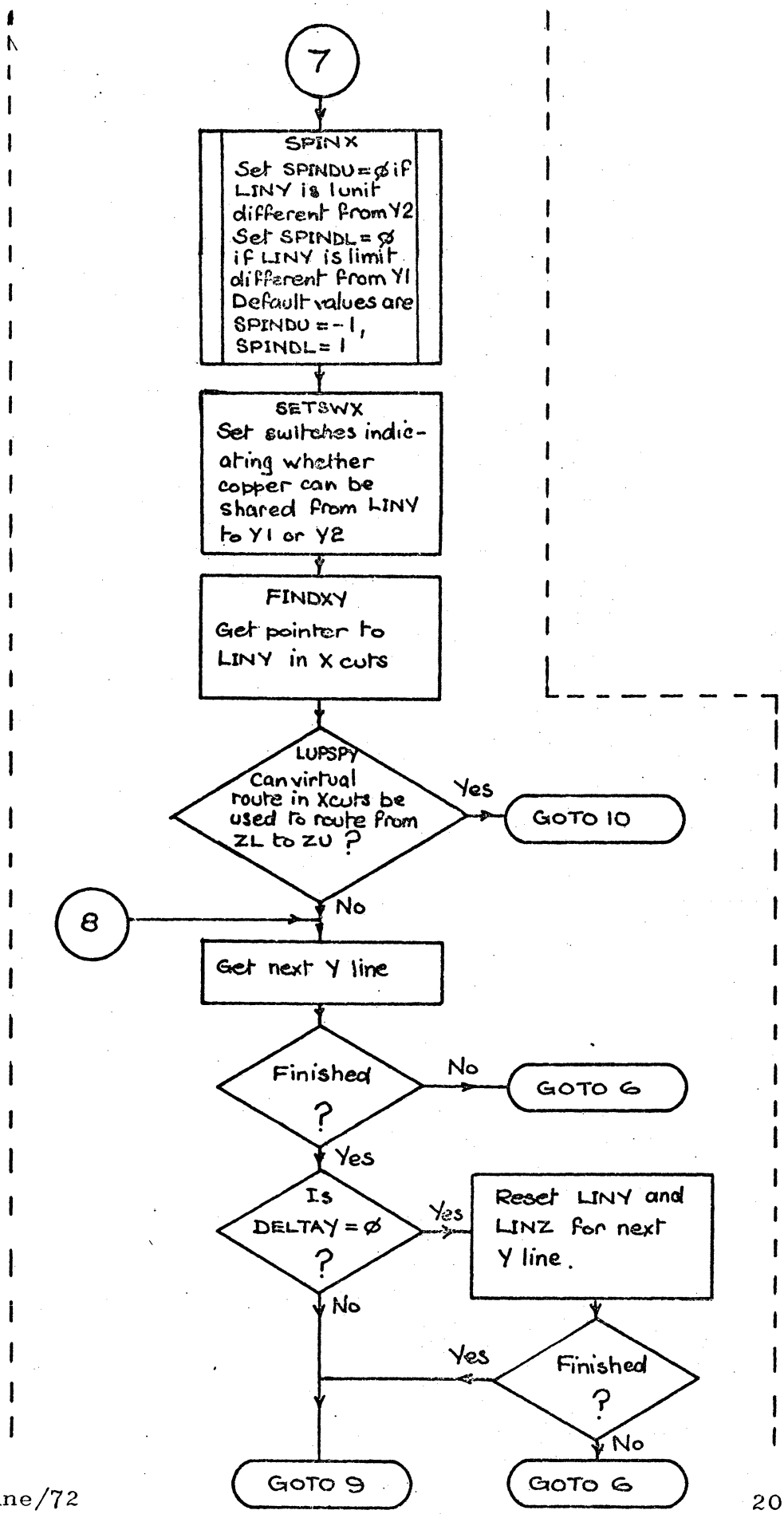
SUBROUTINE ZROUT2 STRUCTURE

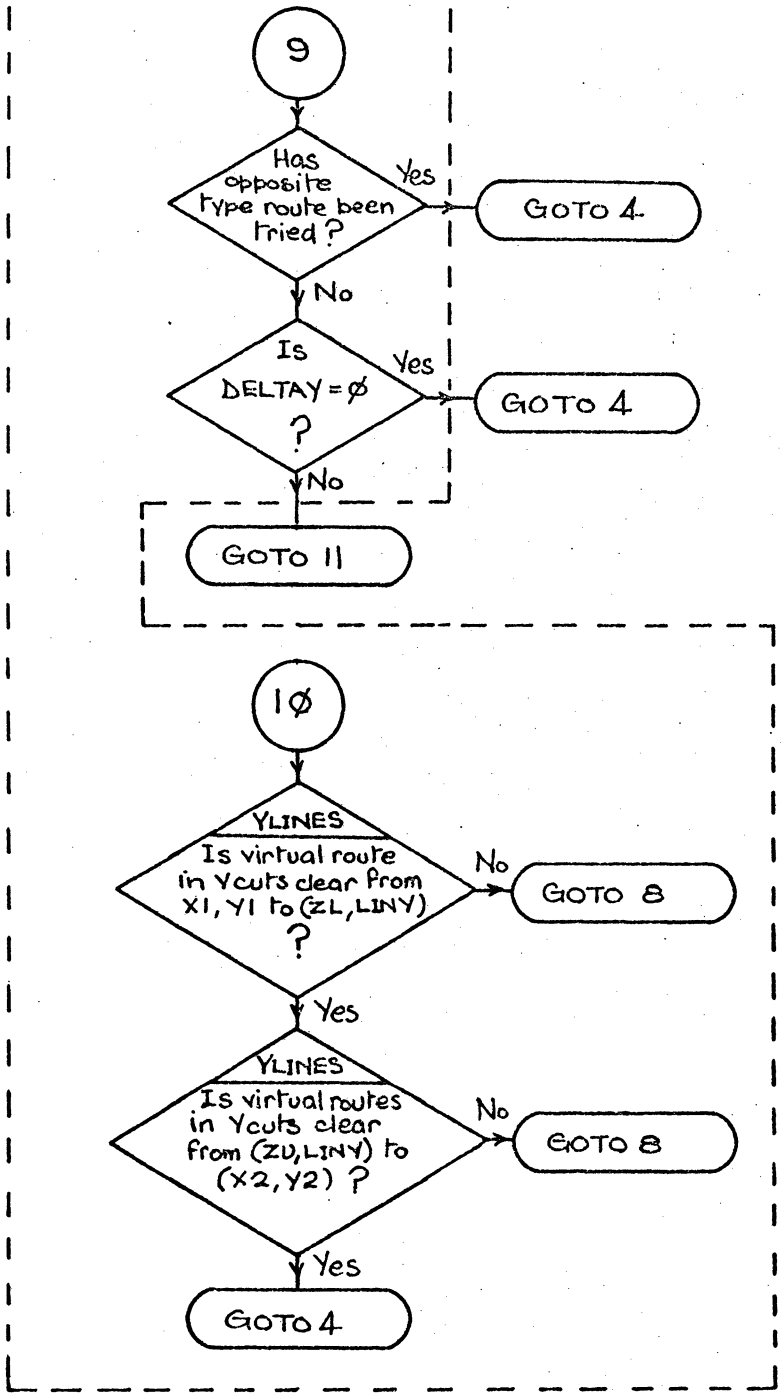


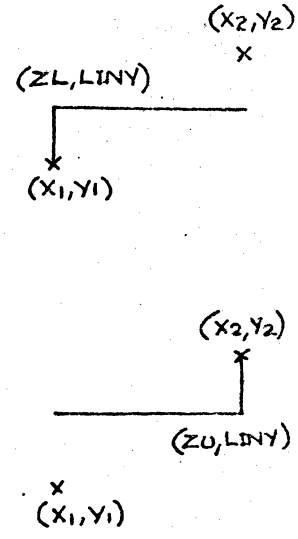
SUBROUTINE ZROUT2 FLOWCHART PART 1 OF 9

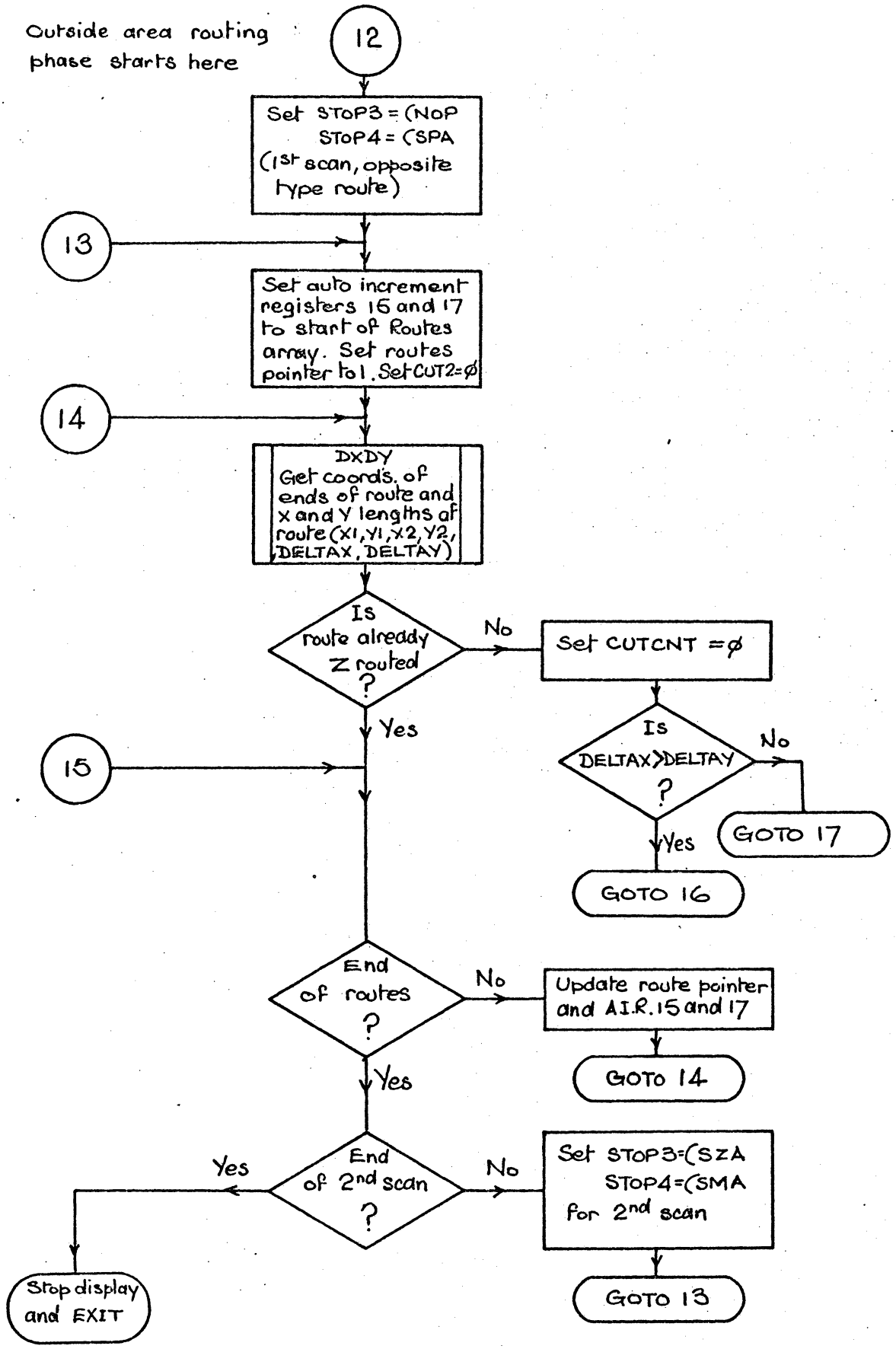


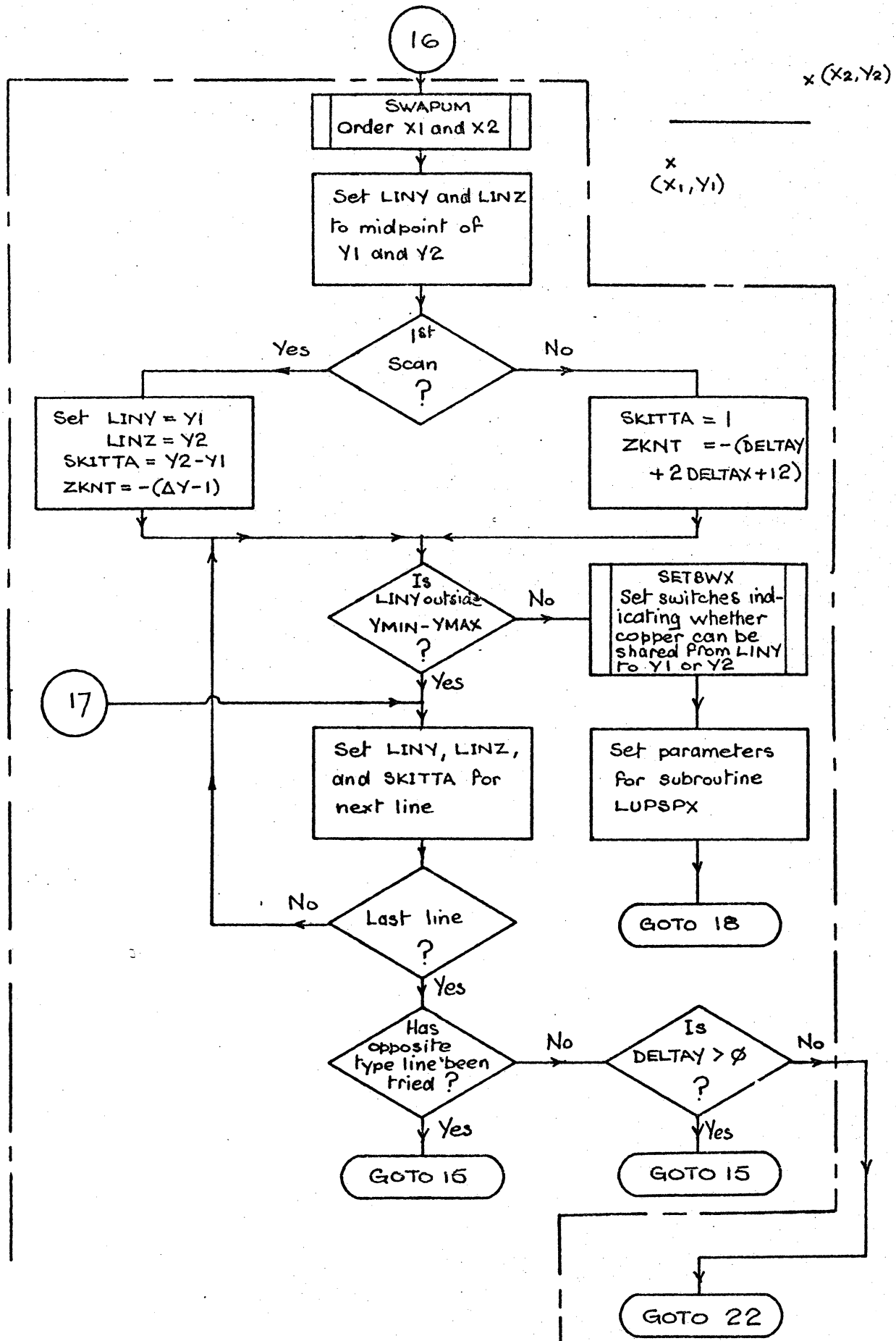

3
4
Is route already ZROUTED ?
Yes
End of routes array ?
No
Set auto increment registers 15 and 17 to next route
GOTO 2
No
Yes
1st SCAN ?
Yes
Has opposite type route been tried ?
No
Set STOP1 indicator to SKP for opposite type route (2nd scan)
GOTO 1
No
Yes
GOTO 12
Is DELTAY ≥ DELTAX+1 ?
No
Is DELTAY ≥ DELTAX+1 ?
No
GOTO 11
Yes
GOTO 11
Yes
5
Is DELTAX = Ø ?
Yes
GOTO 11
No
SWAPUM
Order X1 and X2 with X2 > X1
Set SKITTA=Y2-Y1
ZKNT=-(ΔY-1)
ZZKNT=Ø
LINY=Y1
LINZ=Y1
6
Set ZL=X1
ZU=X2
GOTO 7
CD Horizontal
ΔY<ΔX
2 X2,Y
4
5
3
1
X,Y1
Cuts scanned in order 1,2,3,4,5.
CD Vertical
ΔX<ΔY
2 (X2,Y2)
4
6
8
9
7
5
3
1
(X1,Y1)
or ΔY=Ø
5
3
1
2
4

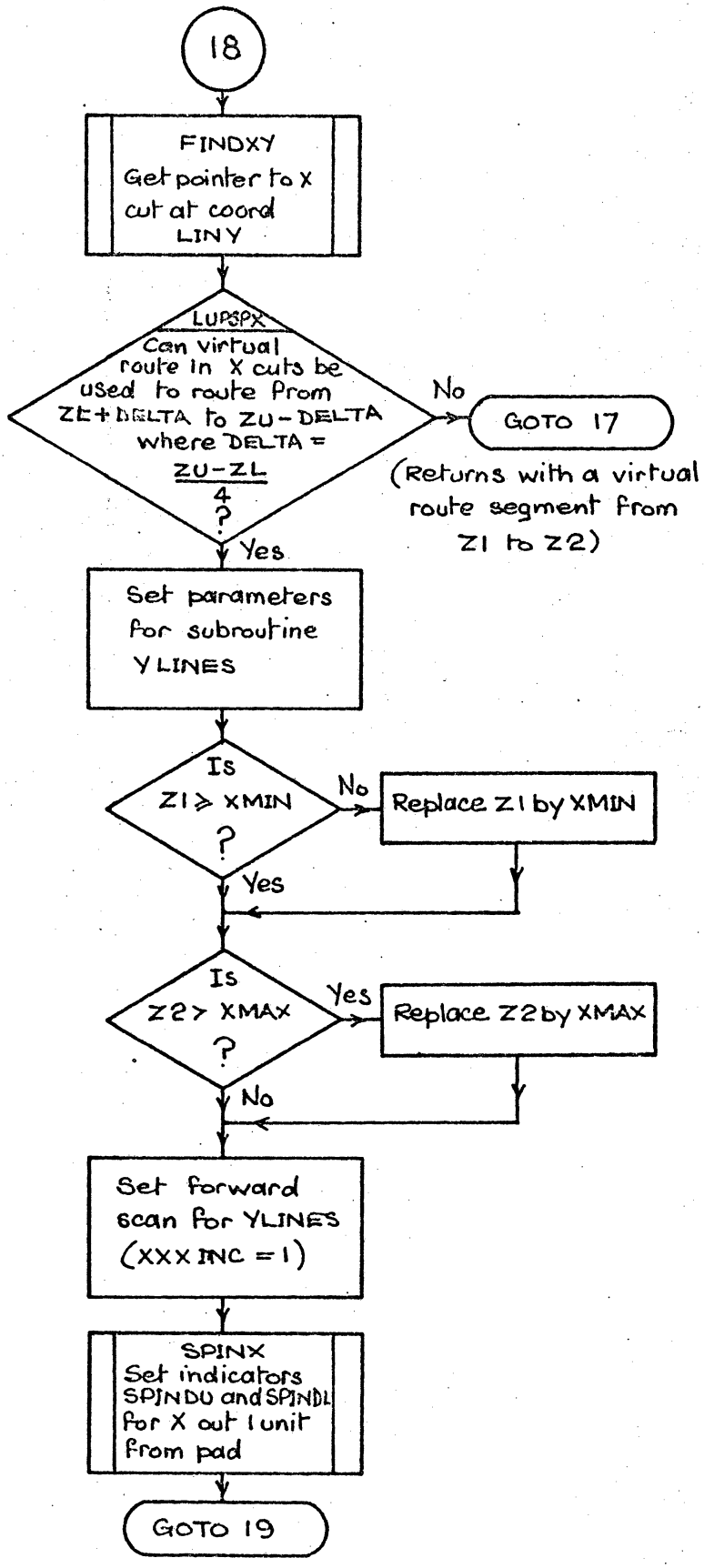

7
SPINX
Set SPINDU=Ø if LINY is 1unit different from Y2
Set SPINDL=Ø if LINY is limit different from Y1
Default values are SPINDU=-1, SPINDL=1
SETSWX
Set switches indicating whether copper can be shared from LINY to Y1 or Y2
FINDXY
Get pointer to LINY in X cuts
LUPSPY
Can virtual route in Xcuts be used to route from ZL to ZU ?
Yes
GOTO 10
No
8
Get next Y line
Finished ?
No
GOTO 6
Yes
Is DELTAY = Ø ?
Yes
Reset LINY and LINZ for next Y line.
No
Yes
Finished ?
No
GOTO 9
GOTO 6

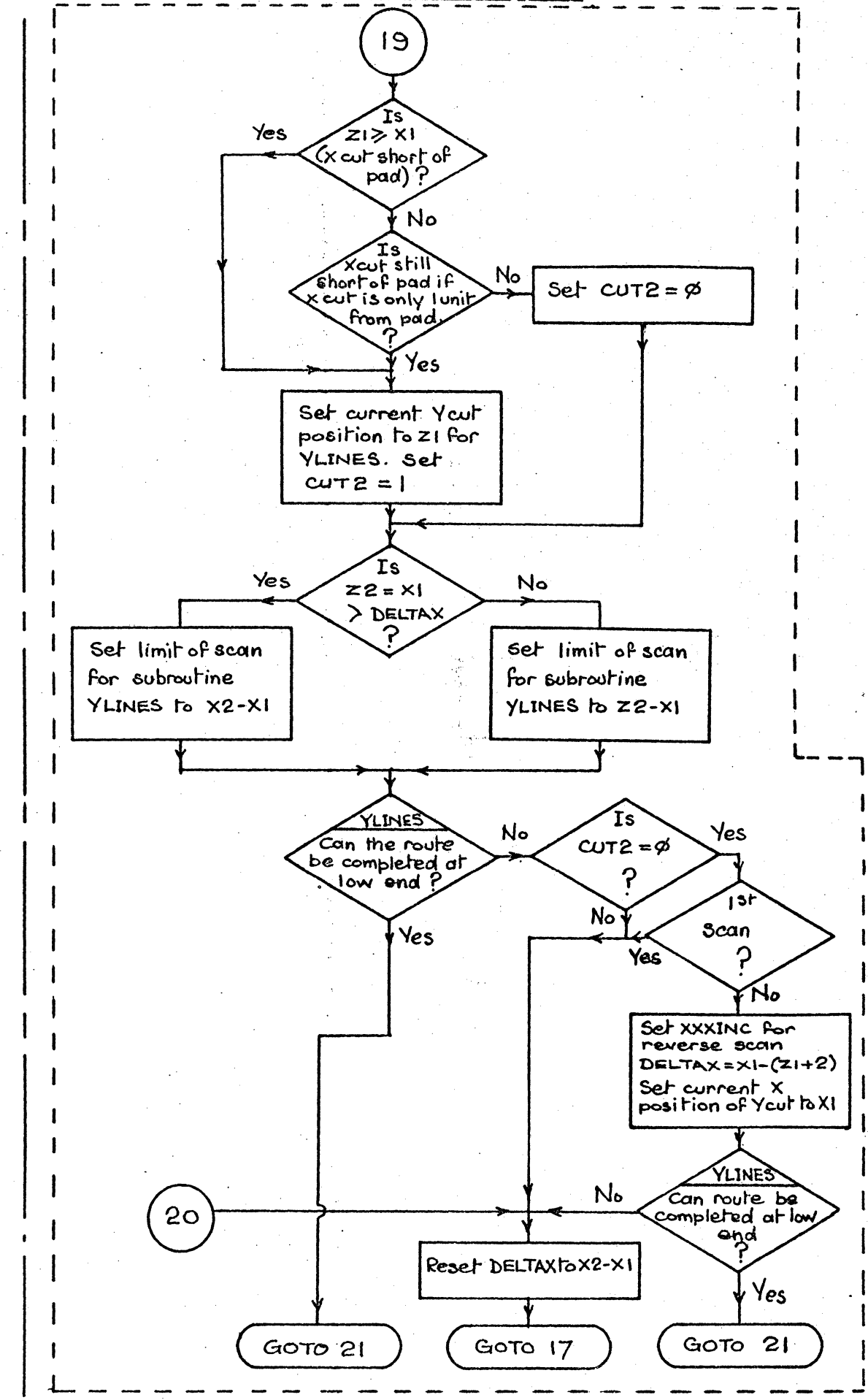

9
Has opposite type route been tried ?
Yes
GOTO 4
No
Is DELTAY = ø ?
Yes
GOTO 4
No
GOTO 11
1ø
YLINES
Is virtual route in Ycuts clear from X1, Y1 to (ZL, LINY) ?
No
GOTO 8
Yes
YLINES
Is virtual routes in Ycuts clear from (ZU, LINY) to (X2, Y2) ?
No
GOTO 8
Yes
GOTO 4
11
Repeat code above in dotted lines for Y cuts substituting ⑤ for ⑪

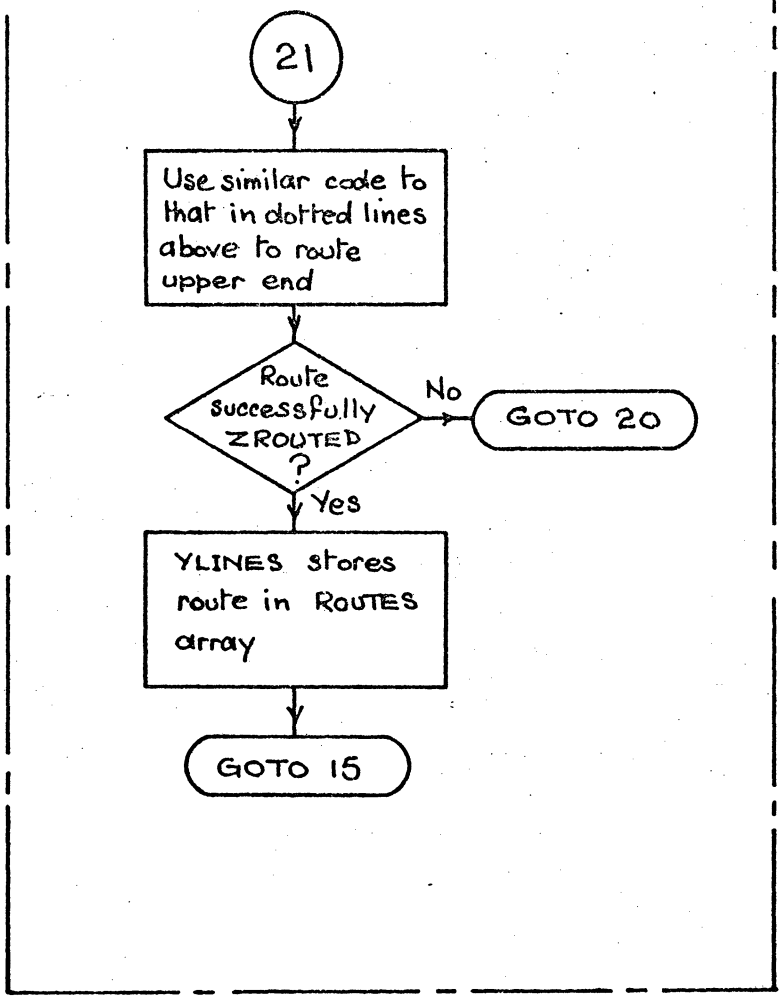

(X2,Y2)
(ZL,LINY)
(X1,Y1)
(X2,Y2)
(ZU,LINY)
(X1,Y1)

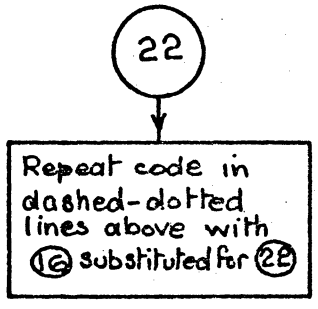

Outside area routing phase starts here
12
Set STOP3 = (NOP
STOP4 = (SPA
(1st scan, opposite type route)
13
Set auto increment registers 16 and 17 to start of Routes array. Set routes pointer to 1. Set CUT2=ø
14
DXDY
Get coord's. of ends of route and x and y lengths of route (X1, Y1, X2, Y2, DELTAX, DELTAY)
Is route already Z routed ?
No
Set CUTCNT = ø
Is DELTAX>DELTAY ?
No
GOTO 17
Yes
GOTO 16
Yes
15
End of routes ?
No
Update route pointer and A.I.R. 15 and 17
GOTO 14
Yes
End of 2nd scan ?
Yes
Stop display and EXIT
No
Set STOP3=(SZA
STOP4=(SMA
for 2nd scan
GOTO 13


16
SWAPUM
Order X1 and X2
Set LINY and LINZ to midpoint of Y1 and Y2
1st Scan ?
Yes
No
Set LINY = Y1
LINZ = Y2
SKITTA = Y2-Y1
ZKNT = -(ΔY-1)
SKITTA = 1
ZKNT = -(DELTAY + 2 DELTAX+12)
Is LINY outside YMIN-YMAX ?
No
SETBWX
Set switches indicating whether copper can be shared from LINY to Y1 or Y2
Yes
17
Set LINY, LINZ, and SKITTA for next line
Set parameters for subroutine LUPSPX
Last line ?
No
Yes
GOTO 18
Has opposite type line been tried ?
No
Is DELTAY > Ø ?
No
Yes
Yes
GOTO 16
GOTO 15
GOTO 22
x (X2,Y2)
x
(X1,Y1)

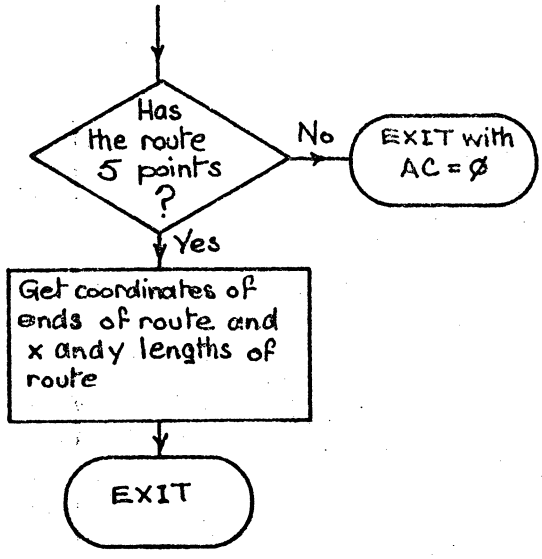

18
FINDXY
Get pointer to X cut at coord LINY
LUPSPX
Can virtual route in X cuts be used to route from ZL+DELTA to ZU-DELTA where DELTA = ZU-ZL / 4 ?
No
GOTO 17
(Returns with a virtual route segment from Z1 to Z2)
Yes
Set parameters for subroutine YLINES
Is Z1 ≥ XMIN ?
No
Replace Z1 by XMIN
Yes
Is Z2 > XMAX ?
Yes
Replace Z2 by XMAX
No
Set forward scan for YLINES (XXX INC = 1)
SPINX
Set indicators SPINDU and SPINDL for X out 1 unit from pad
GOTO 19

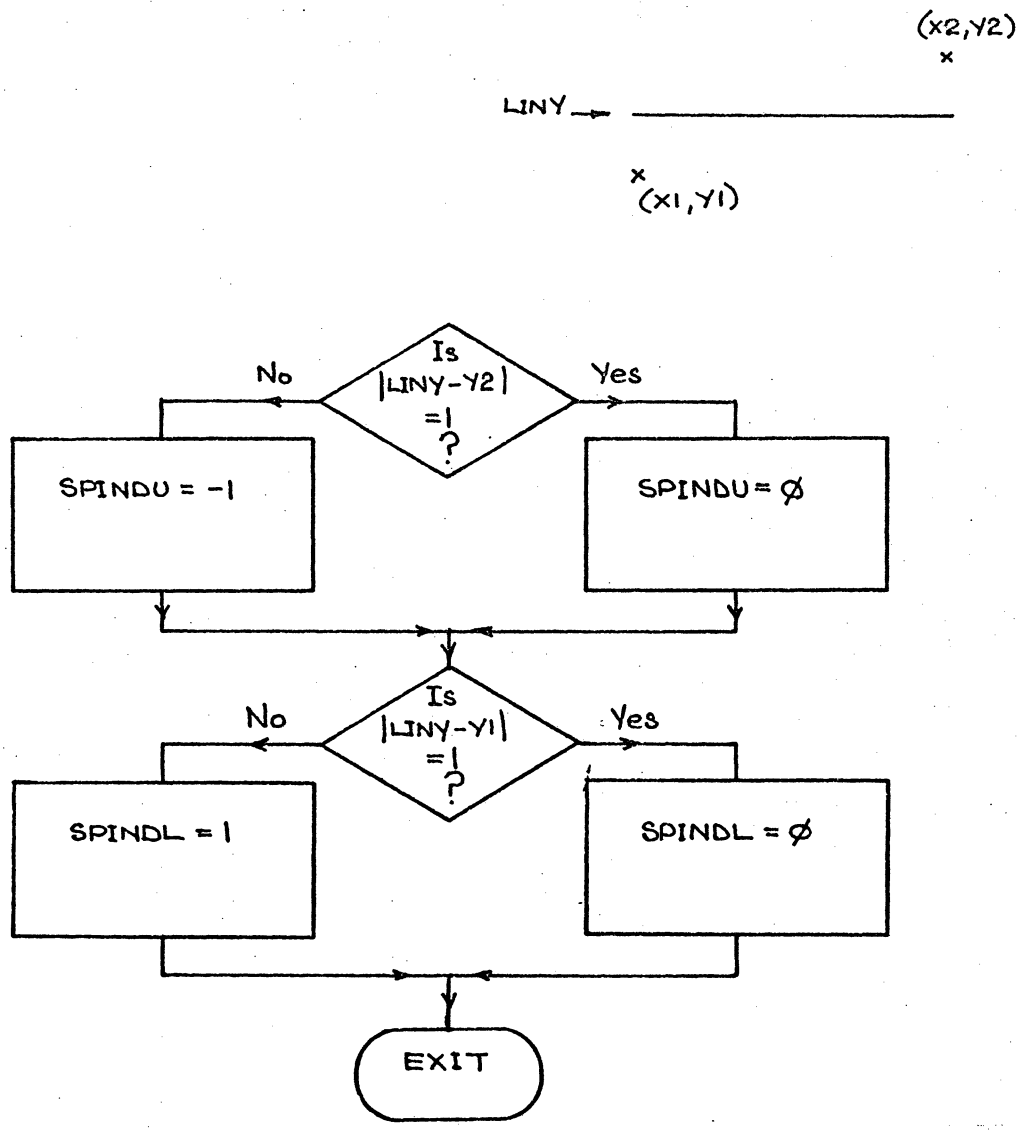

19
Is z1 ≥ x1 (x cut short of pad) ?
Yes
No
Is x cut still short of pad if x cut is only 1 unit from pad. ?
No
Set CUT2 = ø
Yes
Set current Y cut position to z1 for YLINES. Set CUT2 = 1
Is z2 = x1 > DELTAX ?
Yes
No
Set limit of scan for subroutine YLINES to X2-X1
Set limit of scan for subroutine YLINES to Z2-X1
YLINES Can the route be completed at low end ?
No
Yes
Is CUT2 = ø ?
Yes
No
1st Scan ?
Yes
No
Set XXXINC for reverse scan DELTAX = X1-(Z1+2) Set current X position of Y cut to X1
YLINES Can route be completed at low end ?
No
Yes
20
Reset DELTAX to X2-X1
GOTO 21
GOTO 17
GOTO 21


21
Use similar code to that in dotted lines above to route upper end
Route successfully ZROUTED ?
No
GOTO 20
Yes
YLINES stores route in ROUTES array
GOTO 15

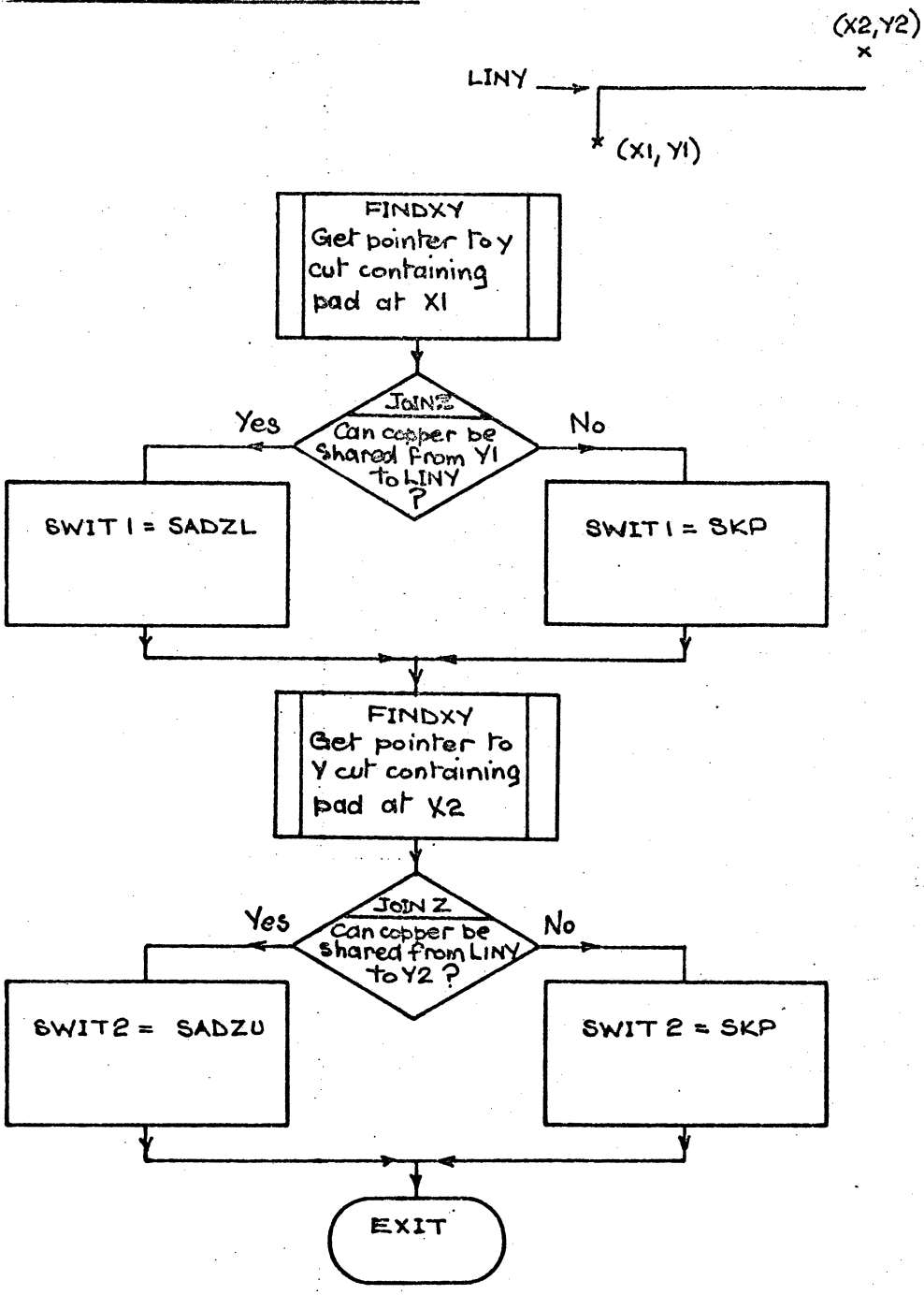

22
Repeat code in dashed-dotted lines above with (16) substituted for (22)

SUBROUTINE DXDY FLOWCHART

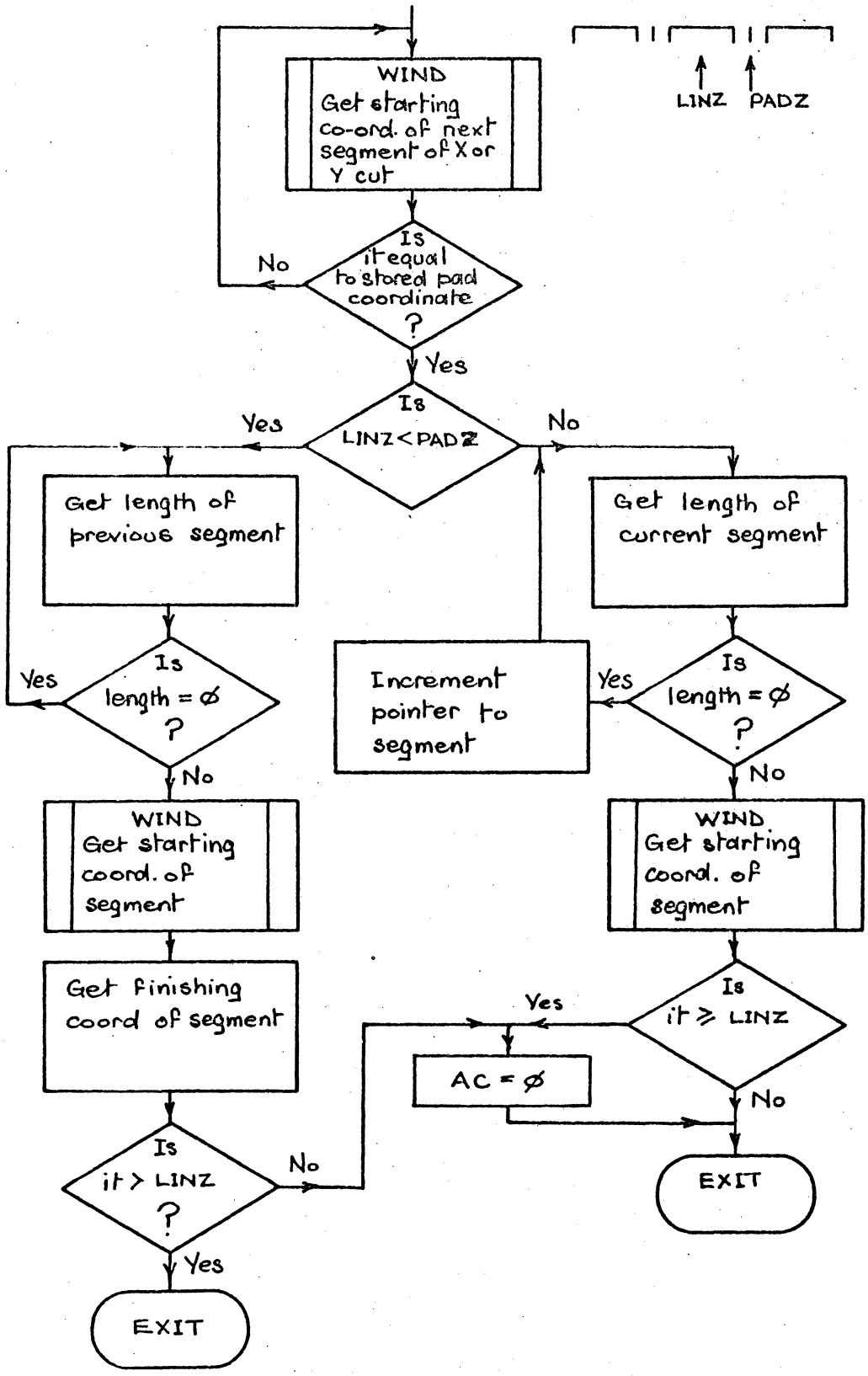


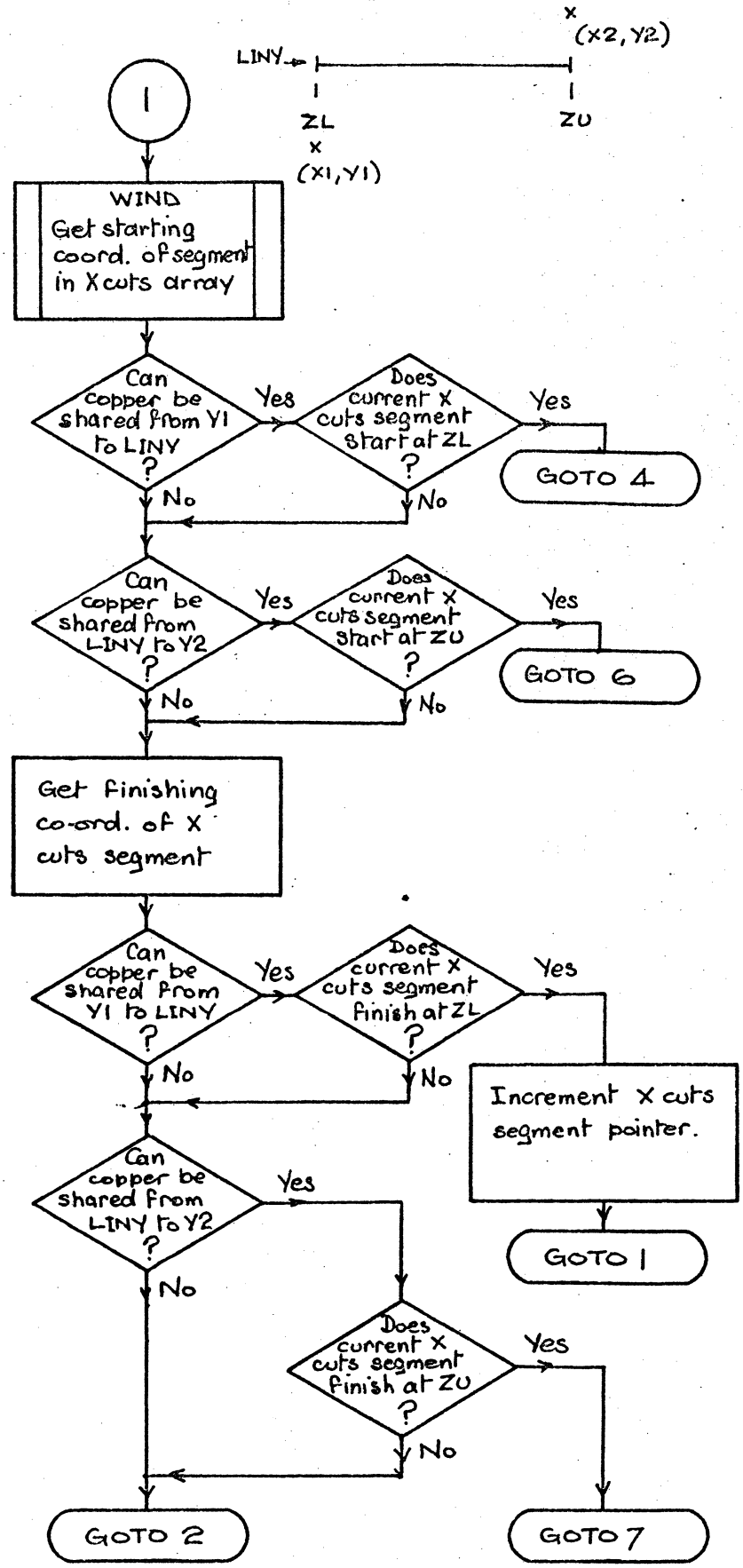

(x2,Y2)
LINY
(x1,Y1)
Is
|LINY-Y2|
=1
?
No
Yes
SPINDU = -1
SPINDU = Ø
Is
|LINY-Y1|
=1
?
No
Yes
SPINDL = 1
SPINDL = Ø
EXIT

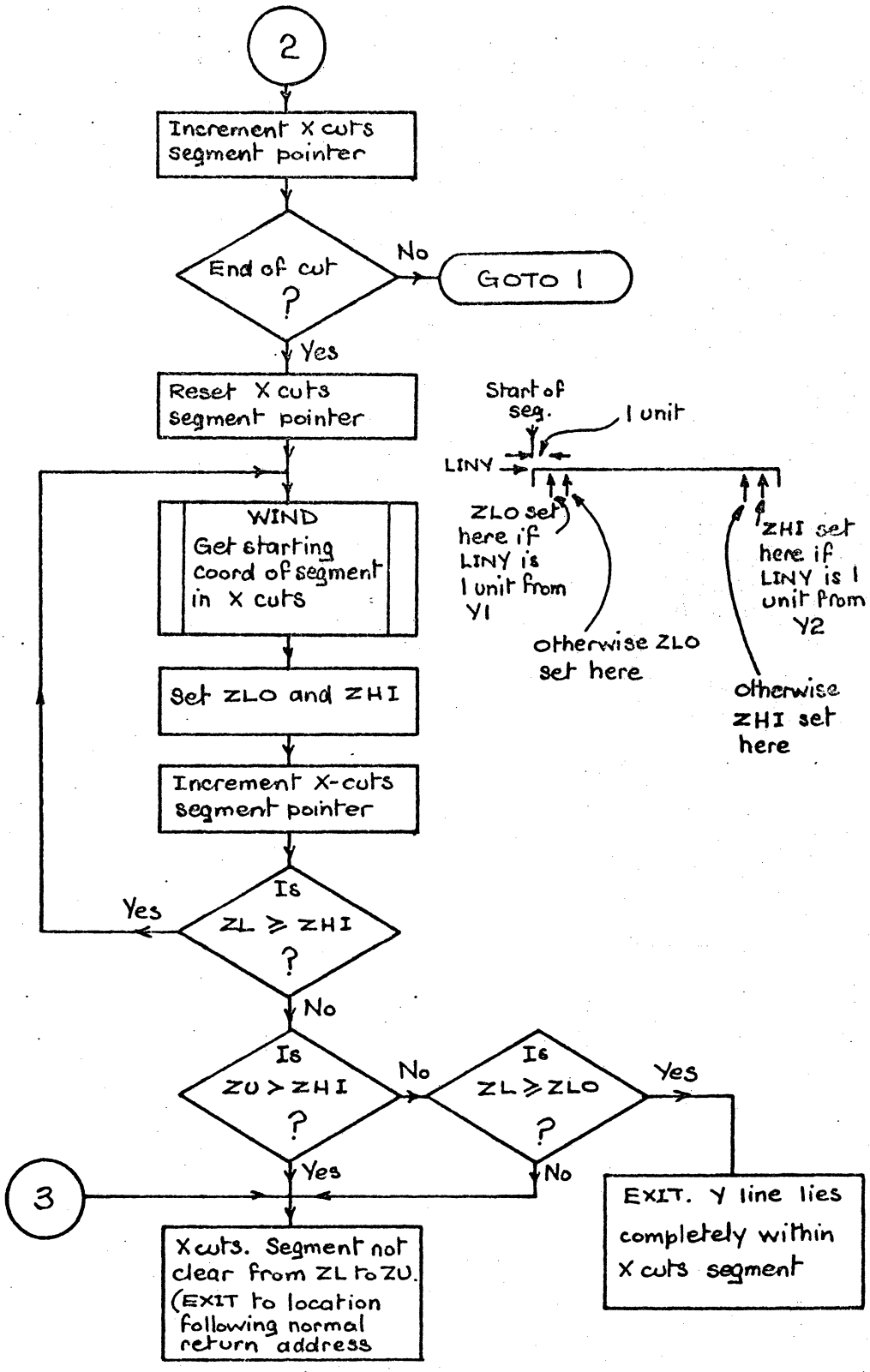
SUBROUTINE SETSWX FLOWCHART

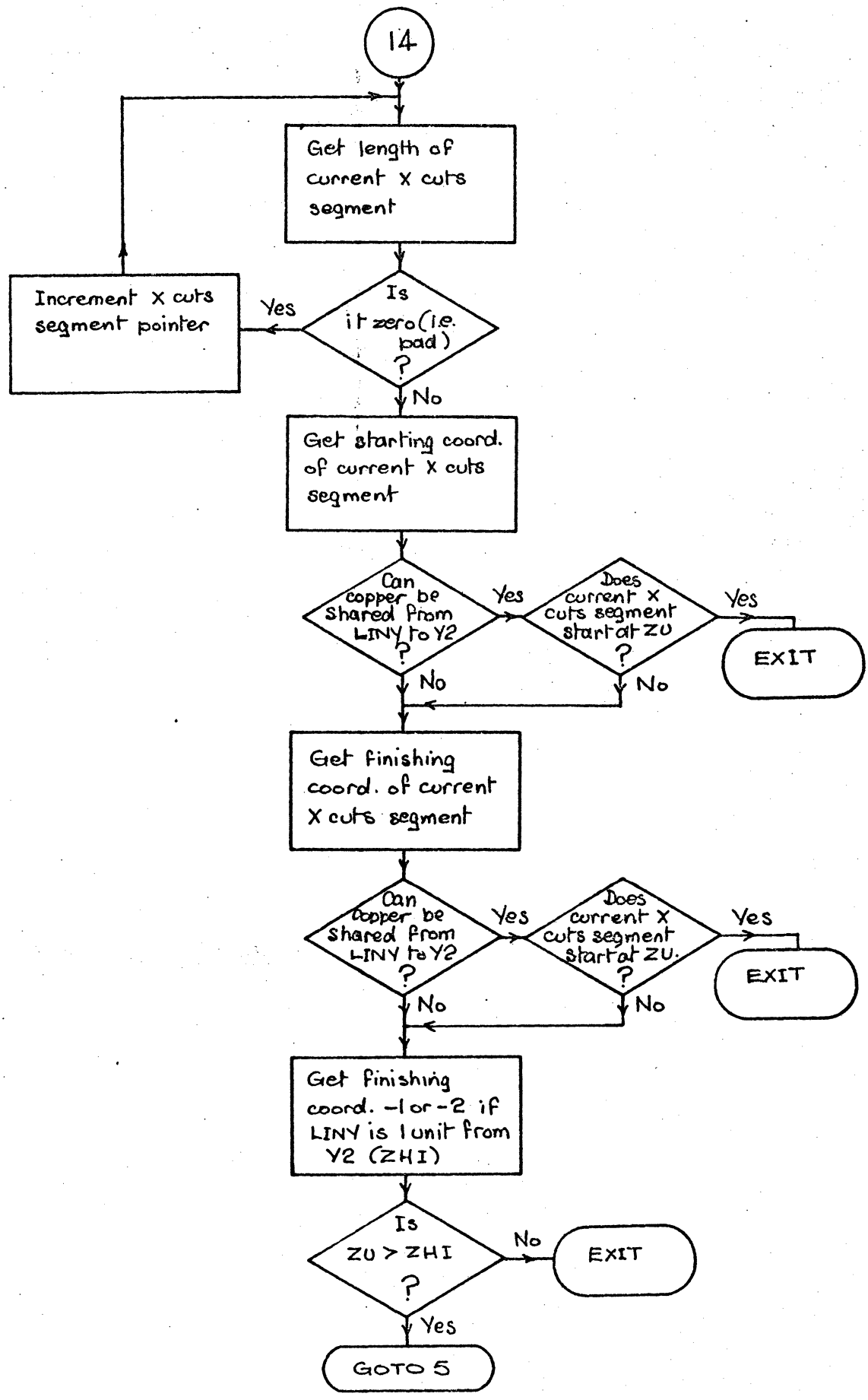


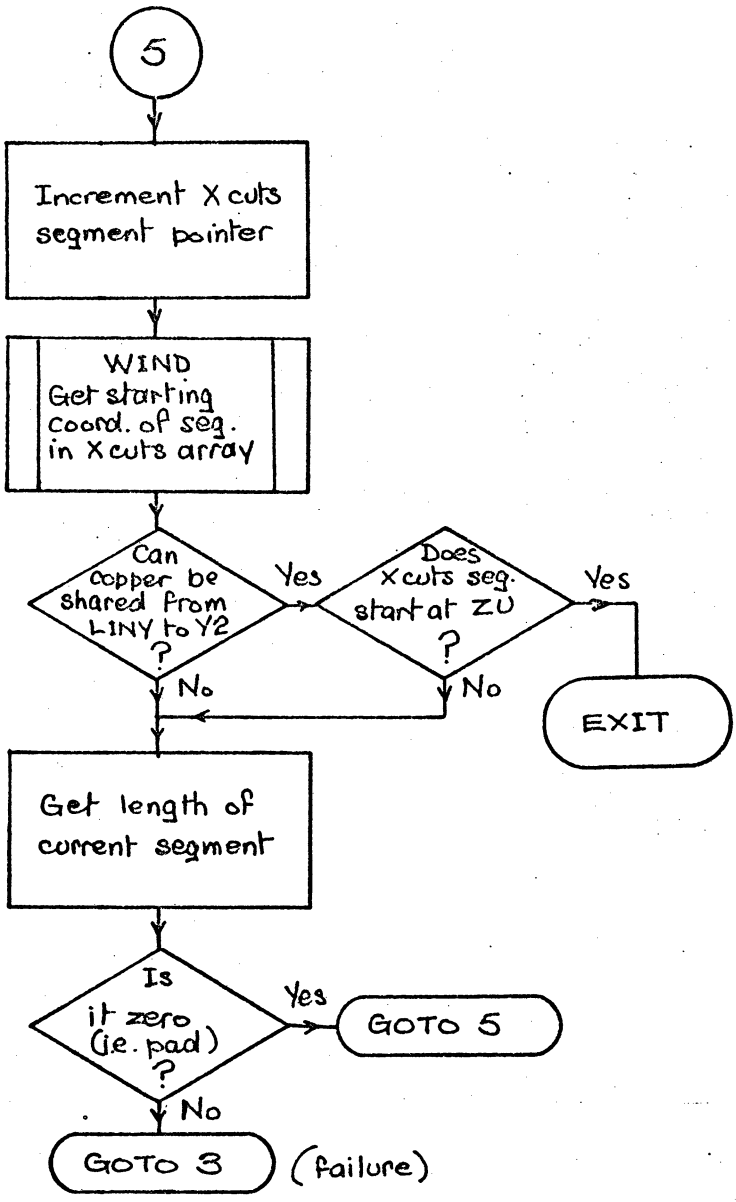
SUBROUTINE JOINZ FLOWCHART

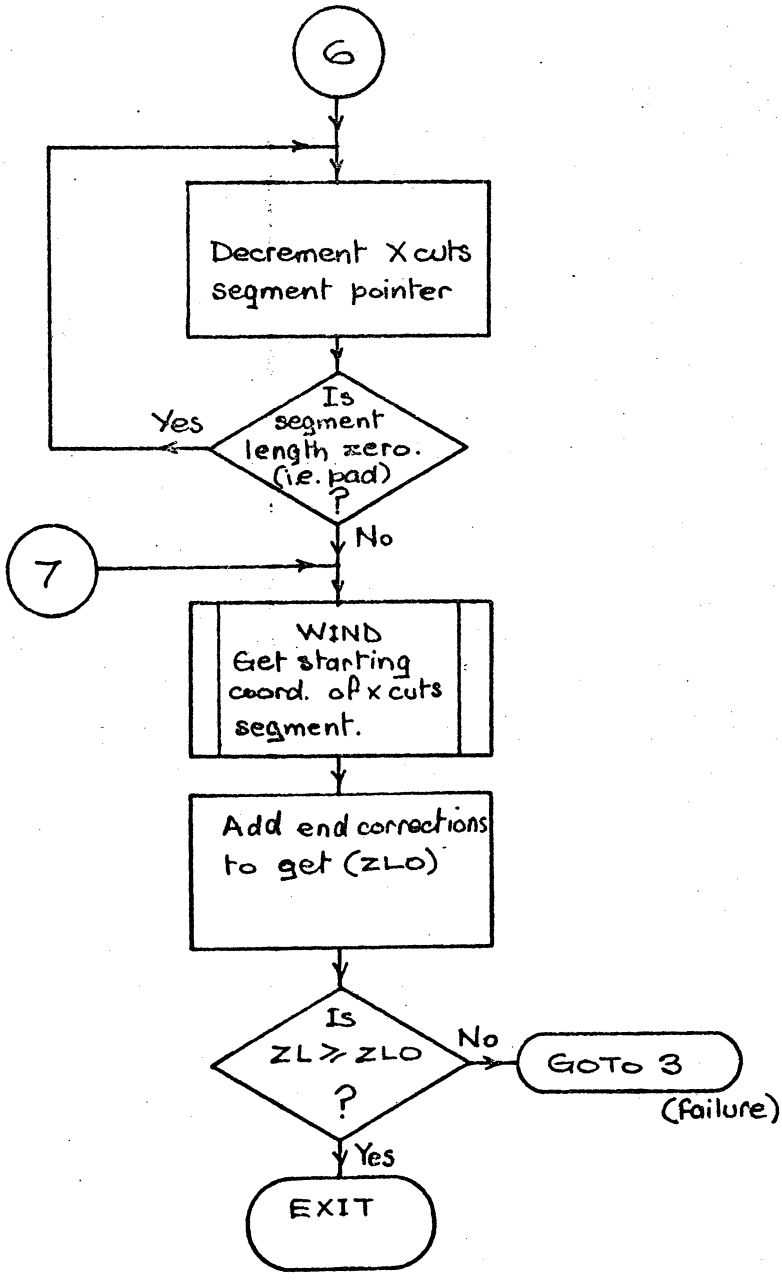


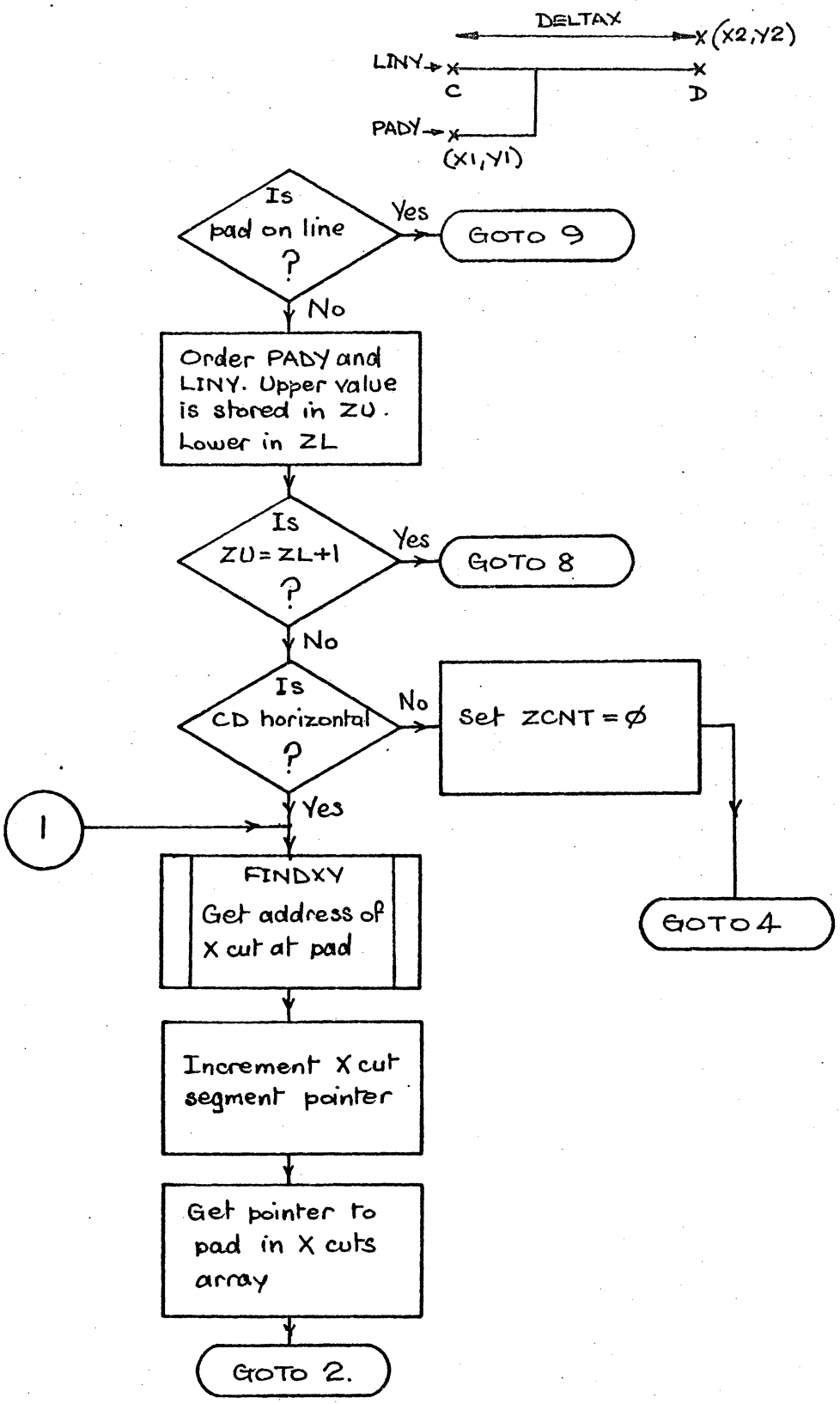

(X2,Y2)
LINY
ZL
ZU
(X1,Y1)
1
WIND
Get starting coord. of segment in X cuts array
Can copper be shared from Y1 to LINY ?
Yes
No
Does current X cuts segment start at ZL ?
Yes
No
GOTO 4
Can copper be shared from LINY to Y2 ?
Yes
No
Does current X cuts segment start at ZU ?
Yes
No
GOTO 6
Get finishing co-ord. of X cuts segment
Can copper be shared from Y1 to LINY ?
Yes
No
Does current X cuts segment finish at ZL ?
Yes
No
Increment X cuts segment pointer.
GOTO 1
Can copper be shared from LINY to Y2 ?
Yes
No
Does current X cuts segment finish at ZU ?
Yes
No
GOTO 2
GOTO 7

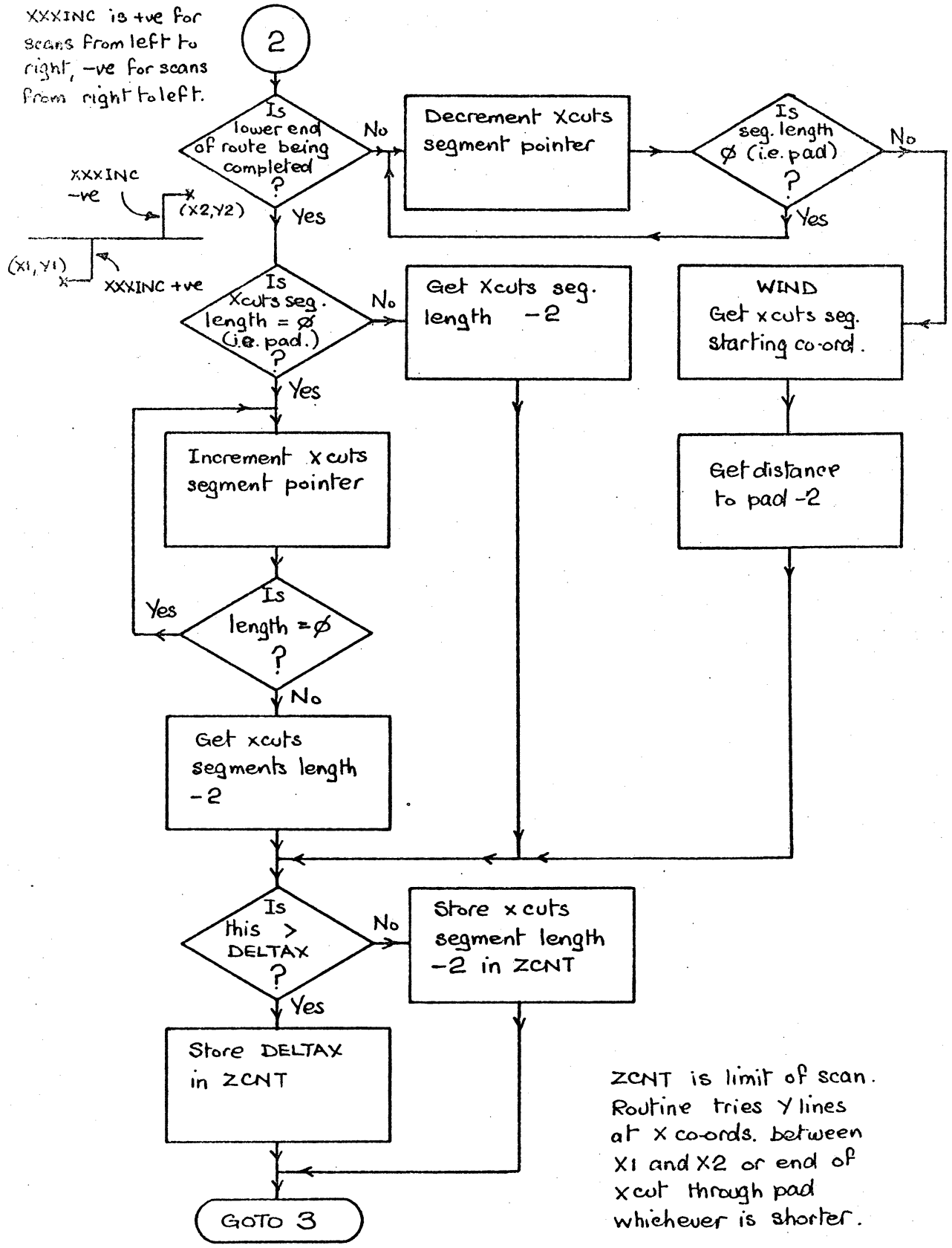

2
Increment X cuts segment pointer
End of cut ?
No
GOTO 1
Yes
Reset X cuts segment pointer
WIND
Get starting coord of segment in X cuts
Set ZLO and ZHI
Increment X-cuts segment pointer
Is ZL ≥ ZHI ?
Yes
No
Is ZU > ZHI ?
No
Is ZL ≥ ZLO ?
Yes
Yes
No
3
X cuts. Segment not clear from ZL to ZU. (EXIT to location following normal return address
EXIT. Y line lies completely within X cuts segment
Start of seg.
1 unit
LINY
ZLO set here if LINY is 1 unit from Y1
otherwise ZLO set here
ZHI set here if LINY is 1 unit from Y2
otherwise ZHI set here


14
Get length of current X cuts segment
Is it zero (i.e. pad) ?
Yes
Increment X cuts segment pointer
No
Get starting coord. of current X cuts segment
Can copper be shared from LINY to Y2 ?
Yes
Does current X cuts segment start at ZU ?
Yes
EXIT
No
No
Get finishing coord. of current X cuts segment
Can copper be shared from LINY to Y2 ?
Yes
Does current X cuts segment start at ZU. ?
Yes
EXIT
No
No
Get finishing coord. -1 or -2 if LINY is 1 unit from Y2 (ZHI)
Is ZU > ZHI ?
No
EXIT
Yes
GOTO 5


5
Increment X cuts segment pointer
WIND
Get starting coord. of seg. in X cuts array
Can copper be shared from LINY to Y2 ?
Yes
Does X cuts seg. start at ZU ?
Yes
No
No
EXIT
Get length of current segment
Is it zero (ie. pad) ?
Yes
GOTO 5
No
GOTO 3
(failure)

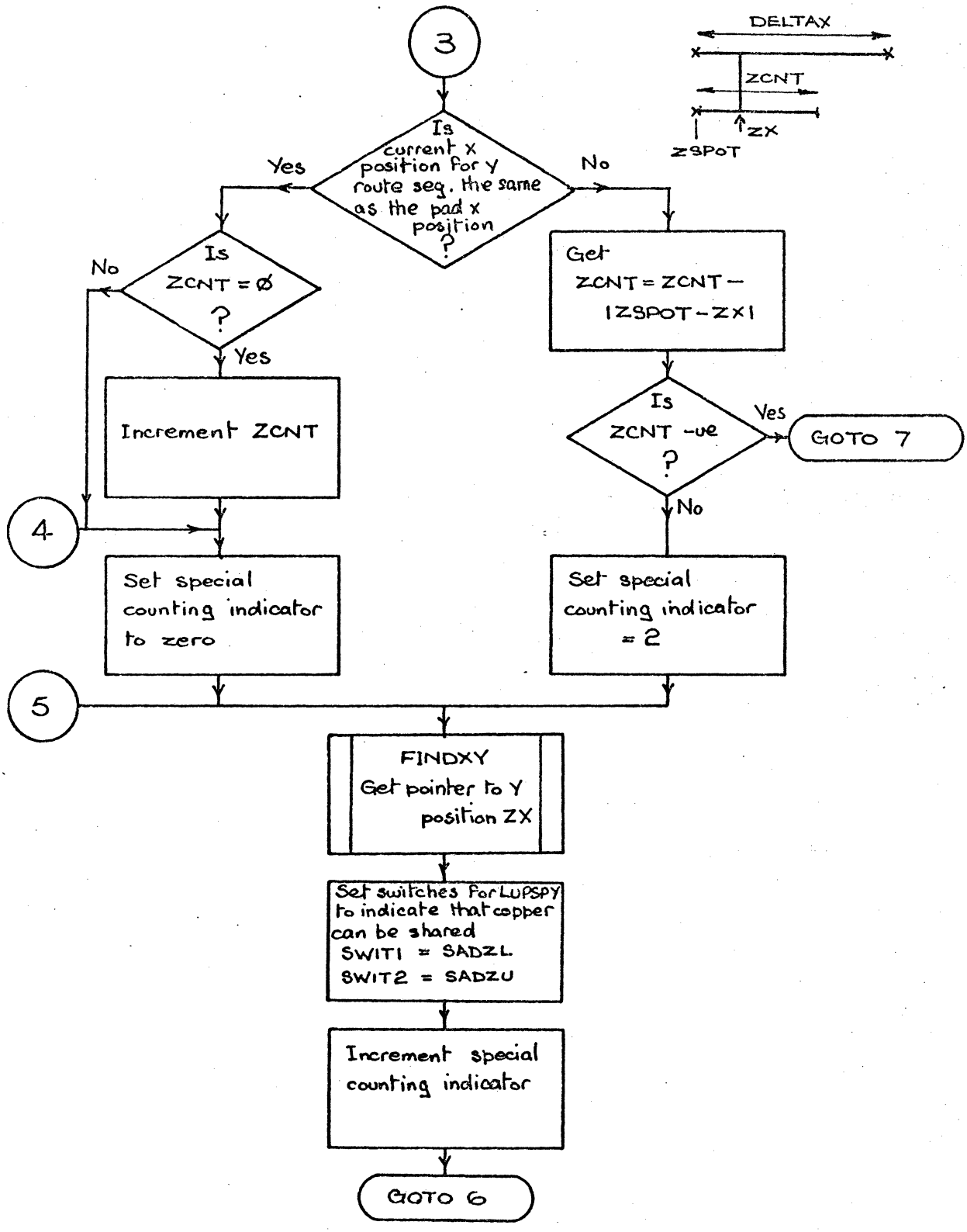

6
Decrement X cuts segment pointer
Is segment length zero. (i.e. pad) ?
Yes
No
7
WIND
Get starting coord. of x cuts segment.
Add end corrections to get (ZLO)
Is ZL ≥ ZLO ?
No
GOTO 3
(failure)
Yes
EXIT


DELTAX
(X2,Y2)
LINY
C
D
PADY
(X1,Y1)
Is pad on line ?
Yes
GOTO 9
No
Order PADY and LINY. Upper value is stored in ZU. Lower in ZL
Is ZU=ZL+1 ?
Yes
GOTO 8
No
Is CD horizontal ?
No
Set ZCNT = ∅
Yes
1
FINDXY
Get address of X cut at pad
GOTO 4
Increment X cut segment pointer
Get pointer to pad in X cuts array
GOTO 2.

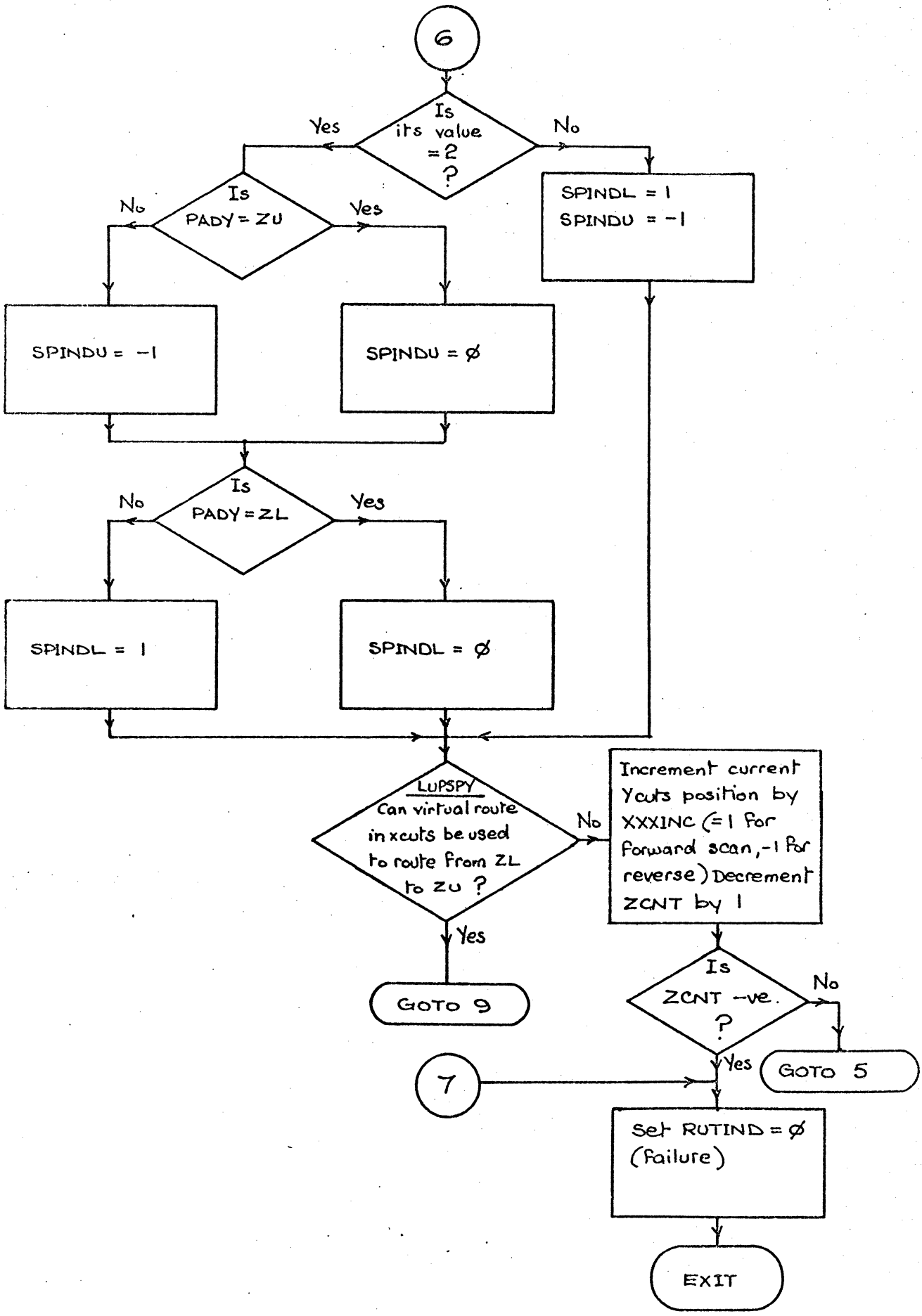



ZCNT is limit of scan. Routine tries Y lines at X co-ords. between X1 and X2 or end of xcut through pad whichever is shorter.

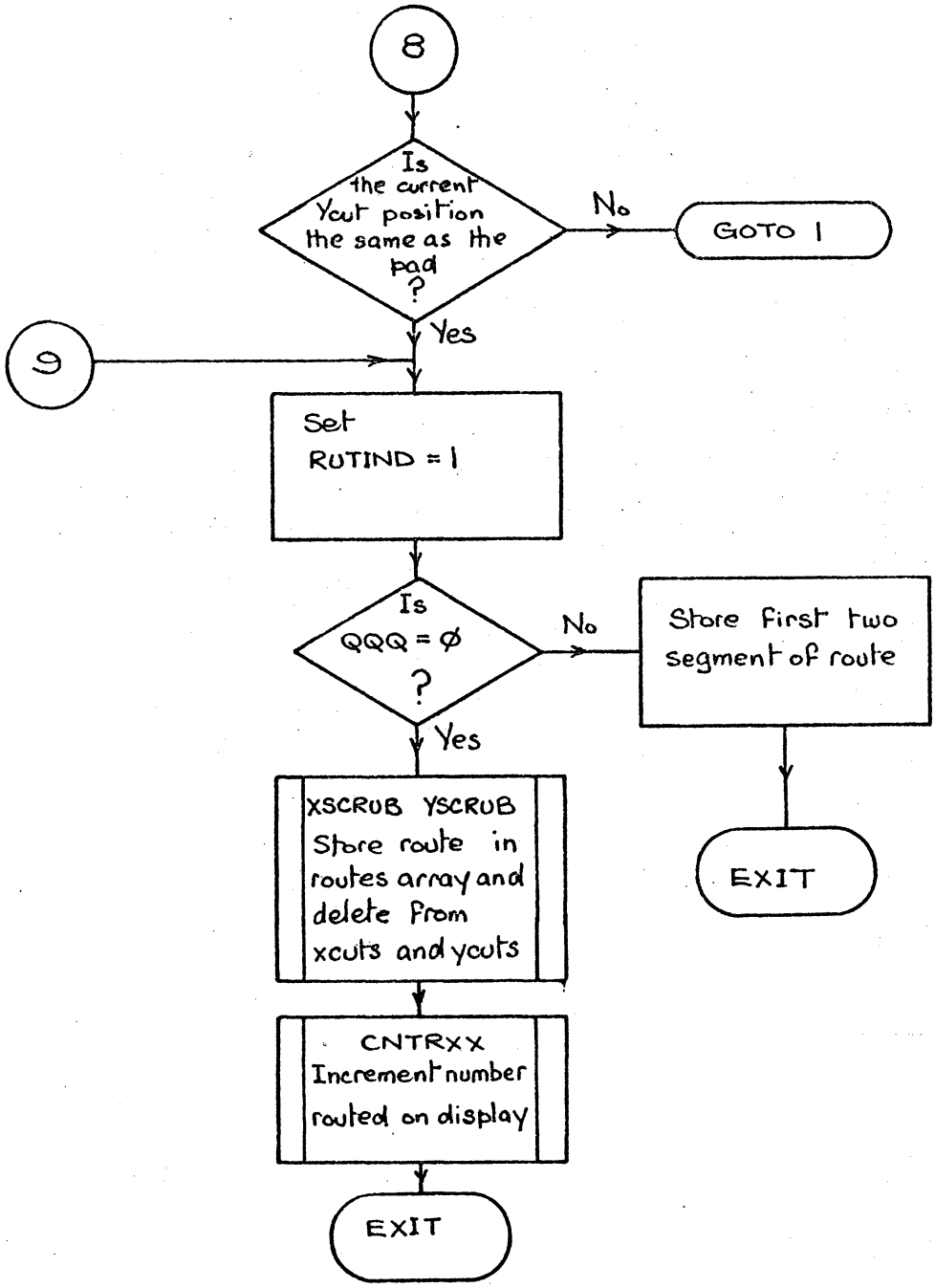
SUBROUTINE YLINES FLOWCHART PART 3 OF 5



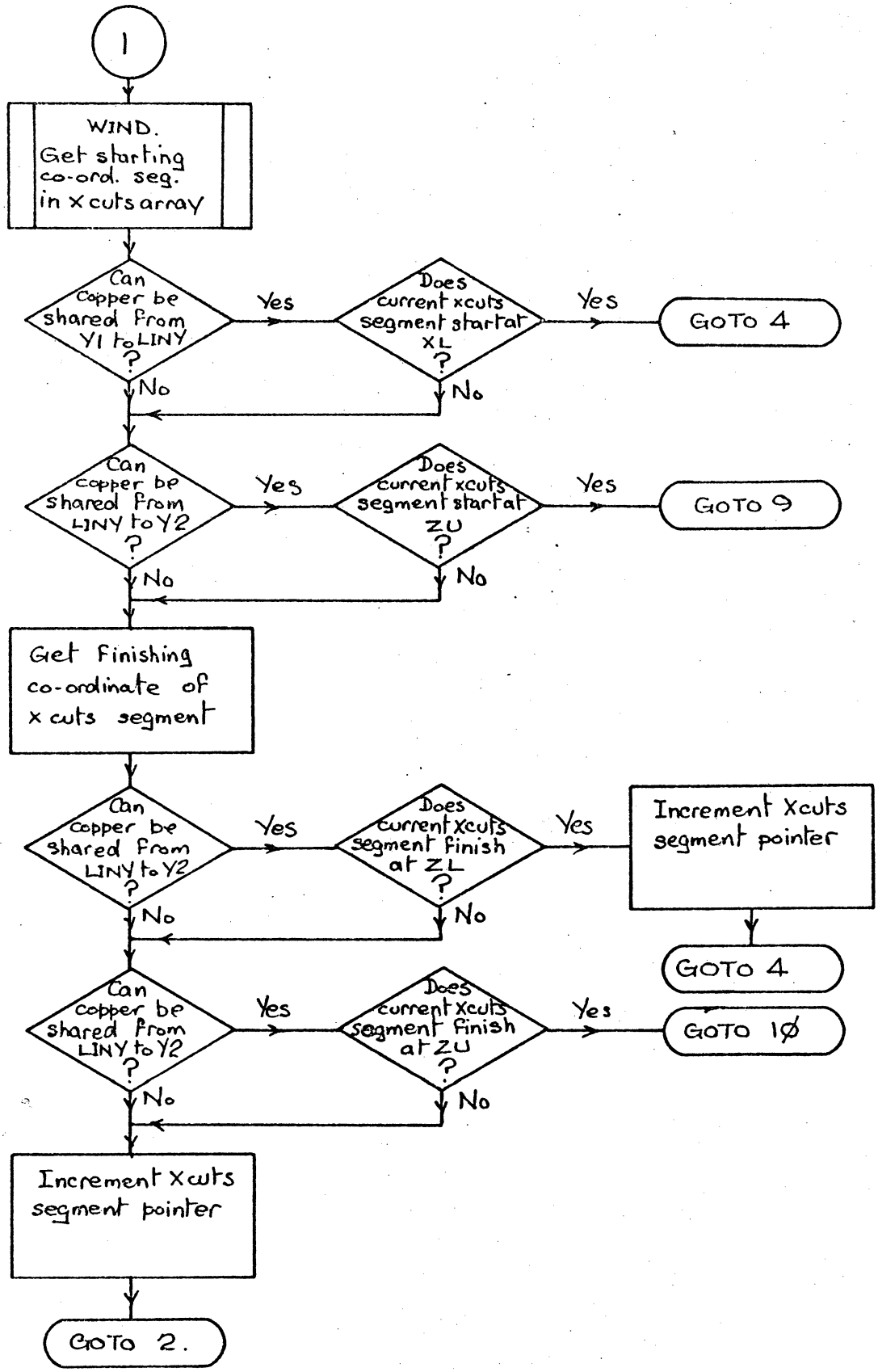
SUBROUTINE YLINES FLOWCHART PART 4 OF 5

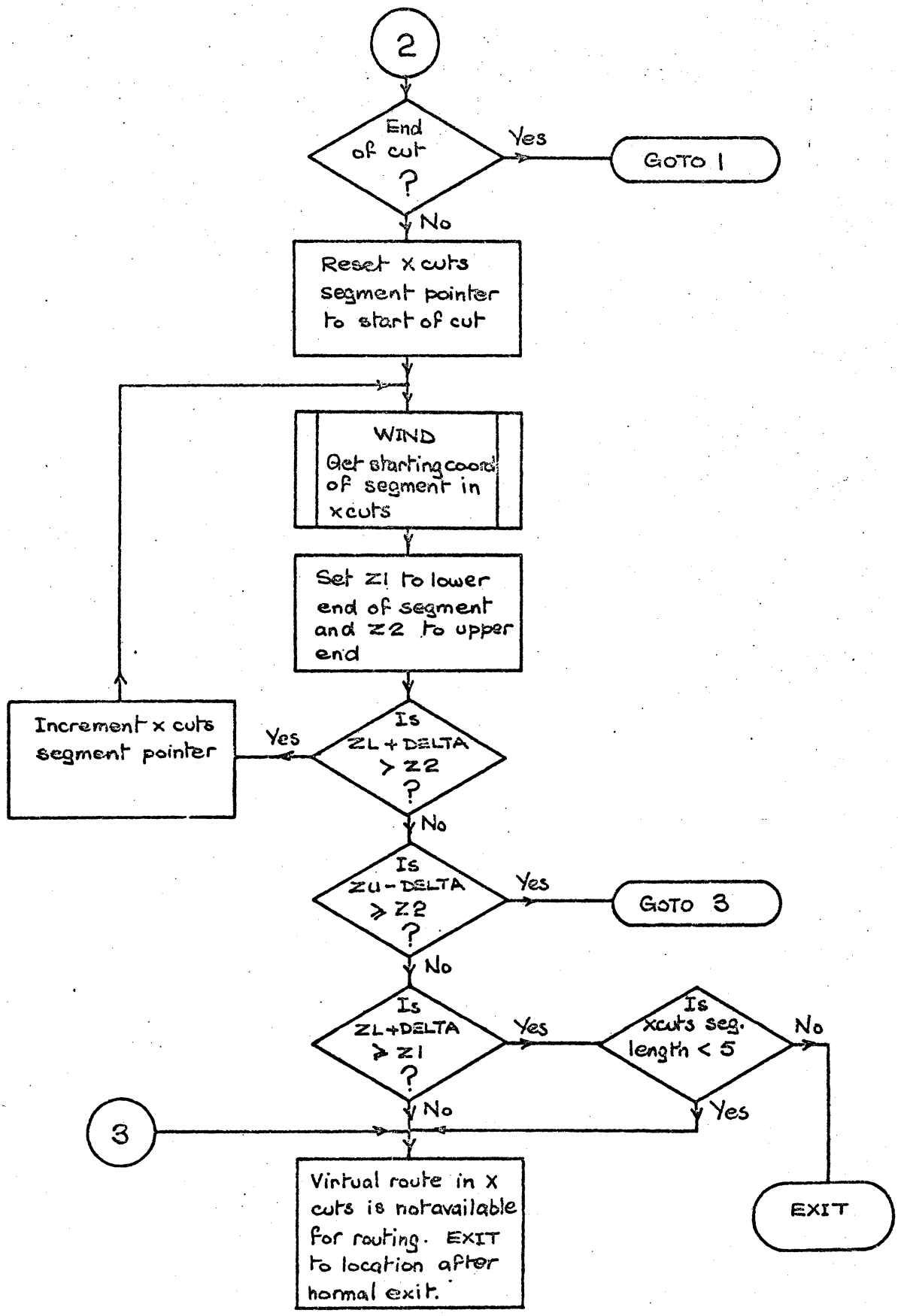


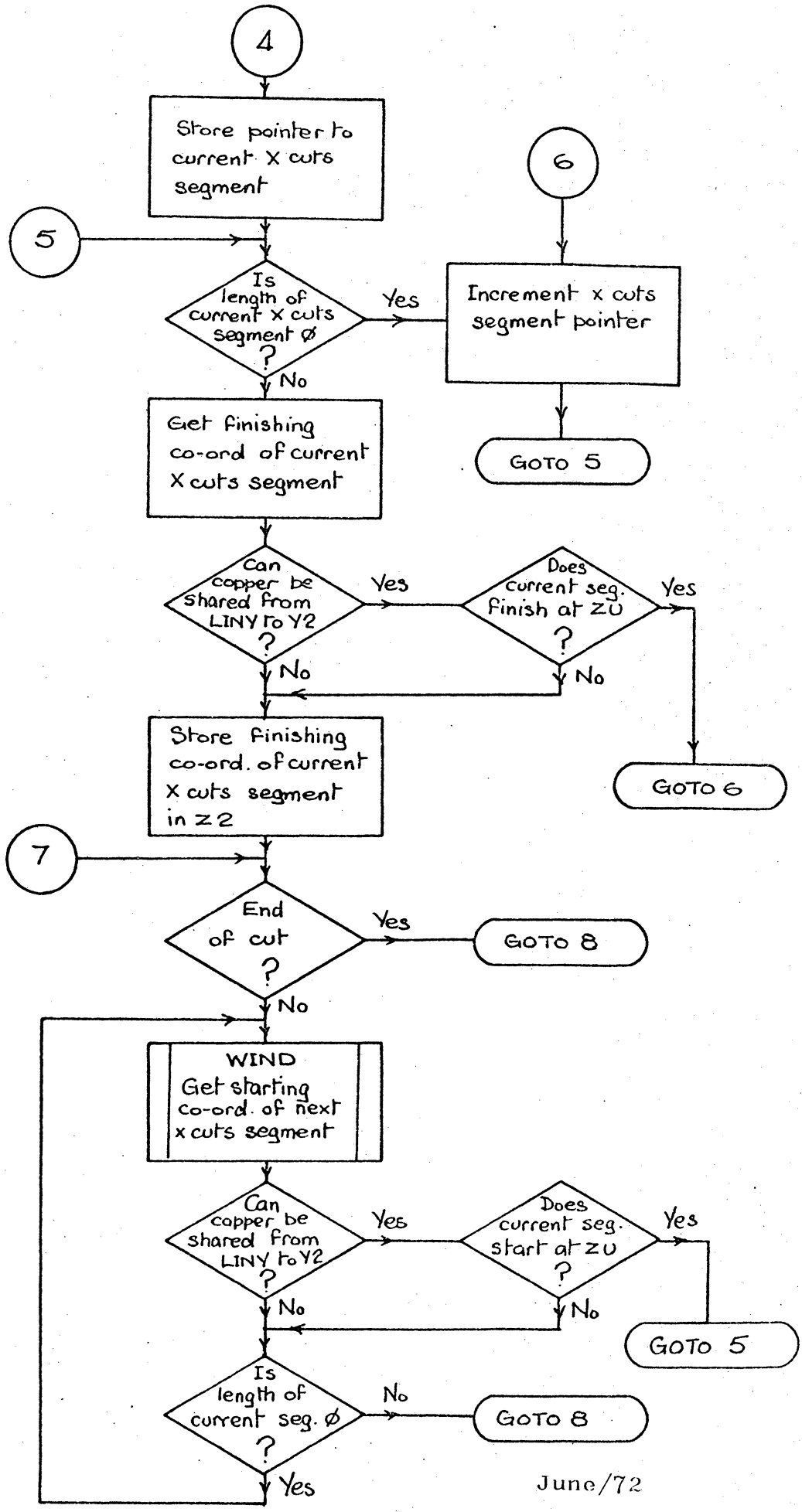
SUBROUTINE YLINES FLOWCHART PART 5 OF 5

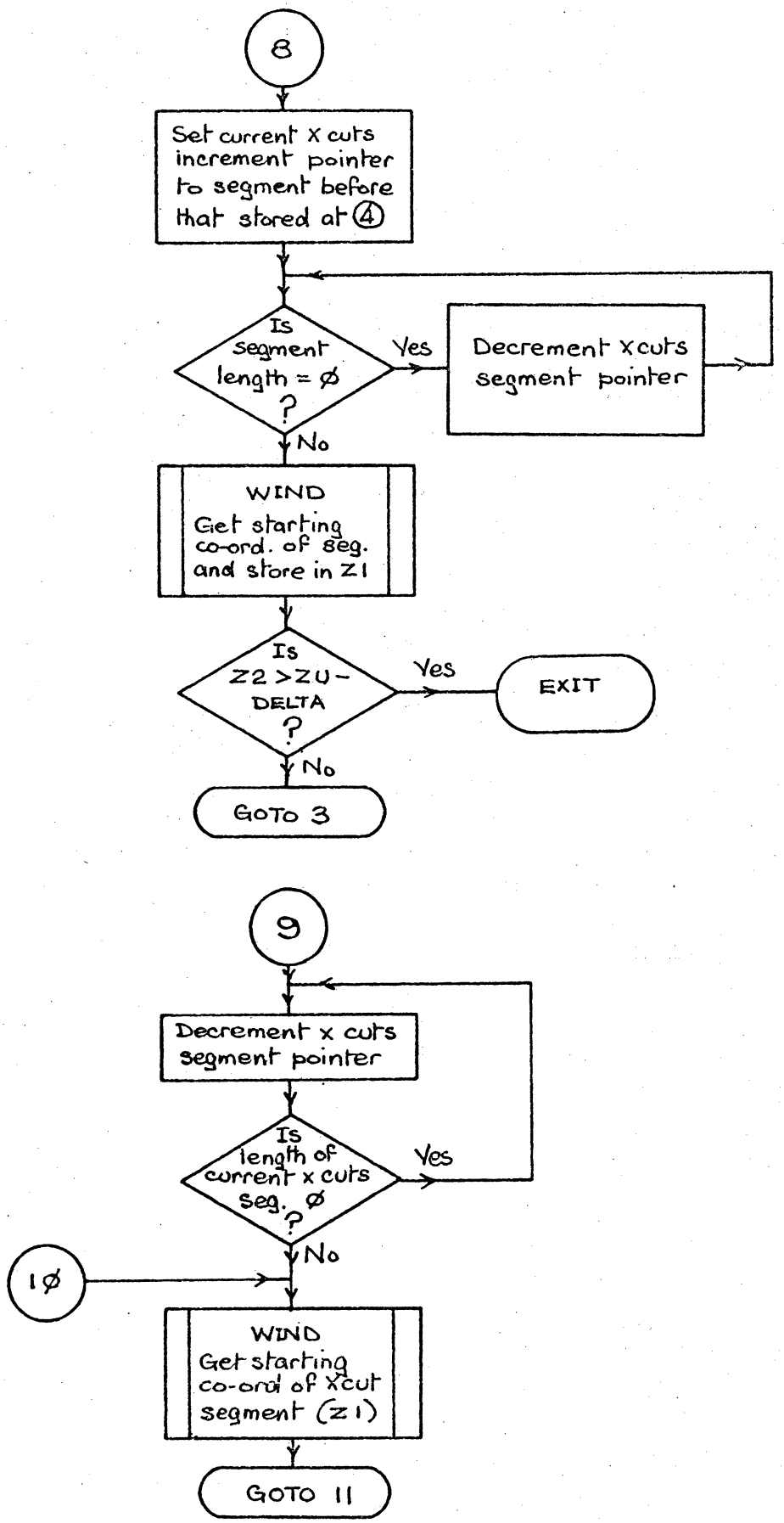


SUBROUTINE LUPSPX FLOWCHART PART 1 OF 5

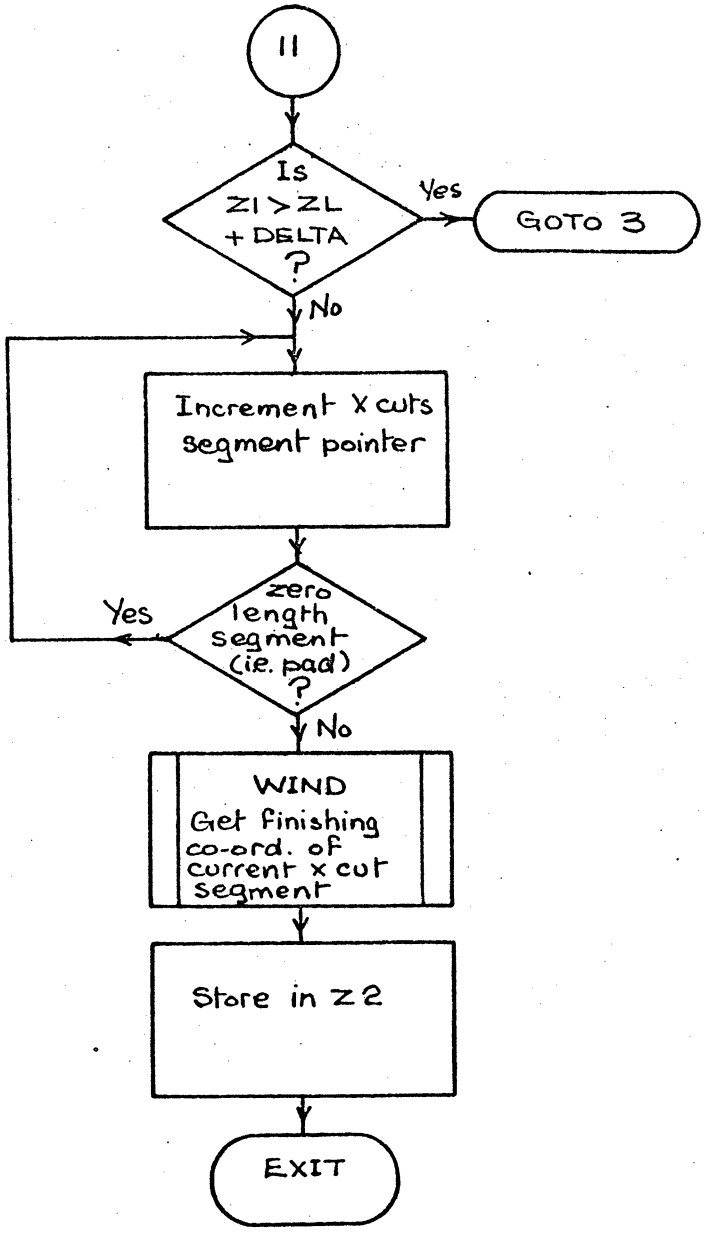



2
End of cut ?
Yes
GOTO 1
No
Reset x cuts segment pointer to start of cut
WIND
Get starting coord of segment in x cuts
Set Z1 to lower end of segment and Z2 to upper end
Is ZL + DELTA > Z2 ?
Yes
Increment x cuts segment pointer
No
Is ZU - DELTA ≥ Z2 ?
Yes
GOTO 3
No
Is ZL+DELTA ≥ Z1 ?
Yes
Is xcuts seg. length < 5
No
Yes
No
3
Virtual route in X cuts is not available for routing. EXIT to location after normal exit.
EXIT

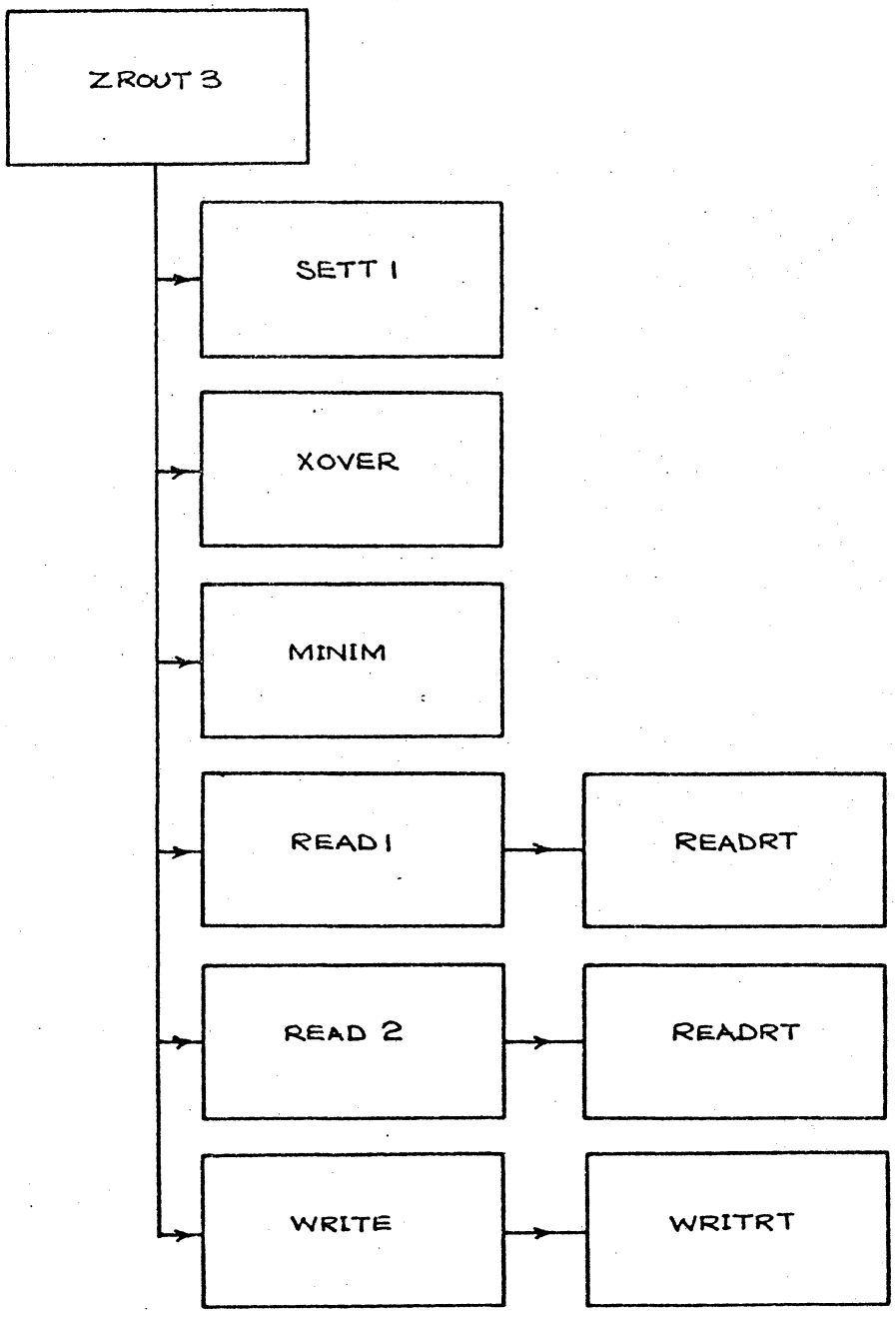

4
Store pointer to current X cuts segment
5
Is length of current X cuts segment ø ?
Yes
No
6
Increment x cuts segment pointer
GOTO 5
Get finishing co-ord of current X cuts segment
Can copper be shared from LINY to Y2 ?
Yes
No
Does current seg. Finish at ZU ?
Yes
No
GOTO 6
Store finishing co-ord. of current X cuts segment in Z2
7
End of cut ?
Yes
GOTO 8
No
WIND
Get starting co-ord. of next x cuts segment
Can copper be shared from LINY to Y2 ?
Yes
No
Does current seg. start at ZU ?
Yes
No
GOTO 5
Is length of current seg. ø ?
No
GOTO 8
Yes


8
Set current x cuts increment pointer to segment before that stored at ④
Is segment length = ø ?
Yes
Decrement xcuts segment pointer
No
WIND
Get starting co-ord. of seg. and store in Z1
Is Z2 > ZU - DELTA ?
Yes
EXIT
No
GOTO 3
9
Decrement x cuts segment pointer
Is length of current x cuts seg. ø ?
Yes
No
1ø
WIND
Get starting co-ord of xcut segment (z1)
GOTO 11

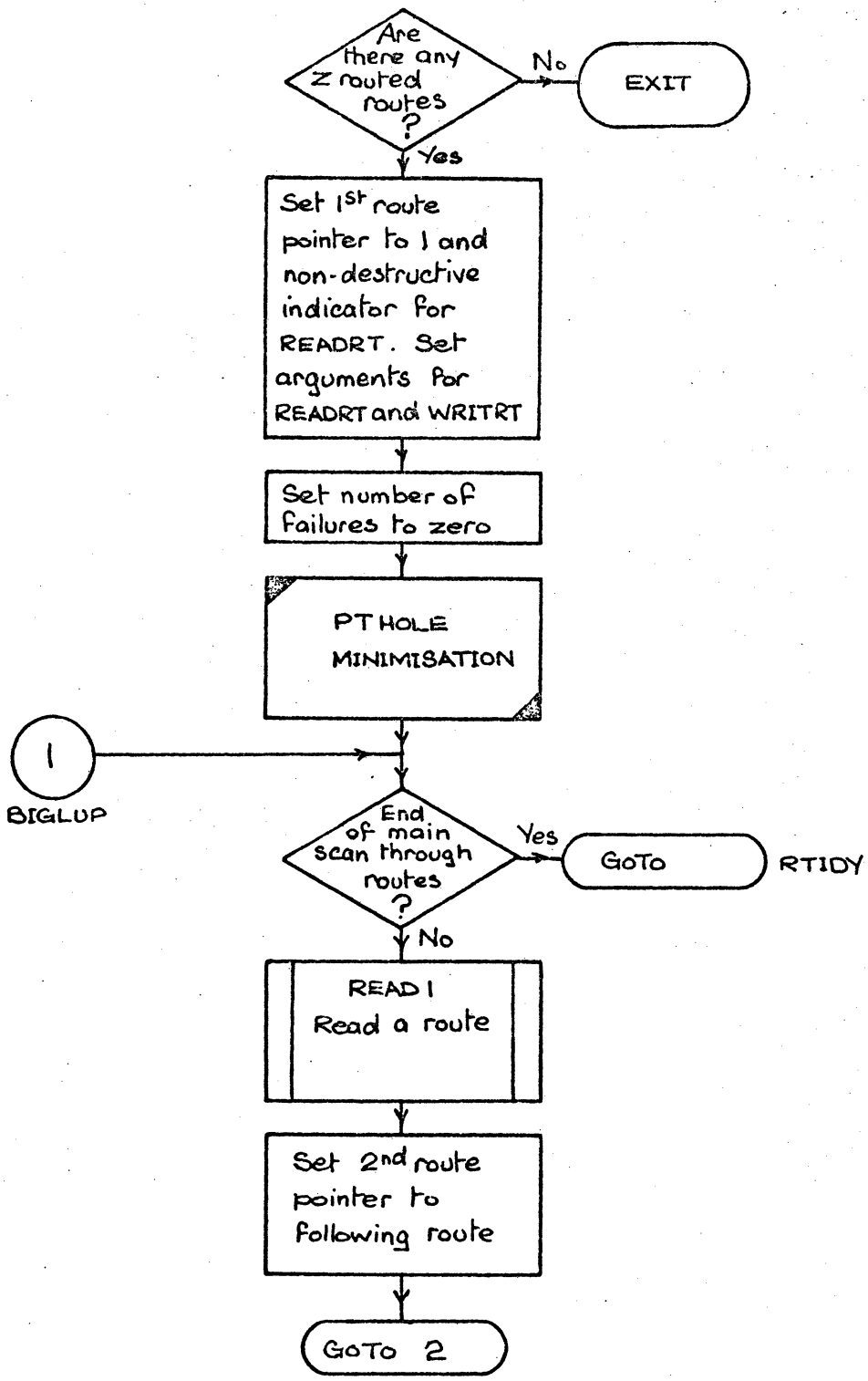
SUBROUTINE LUPSPX FLOWCHART PART 5 OF 5



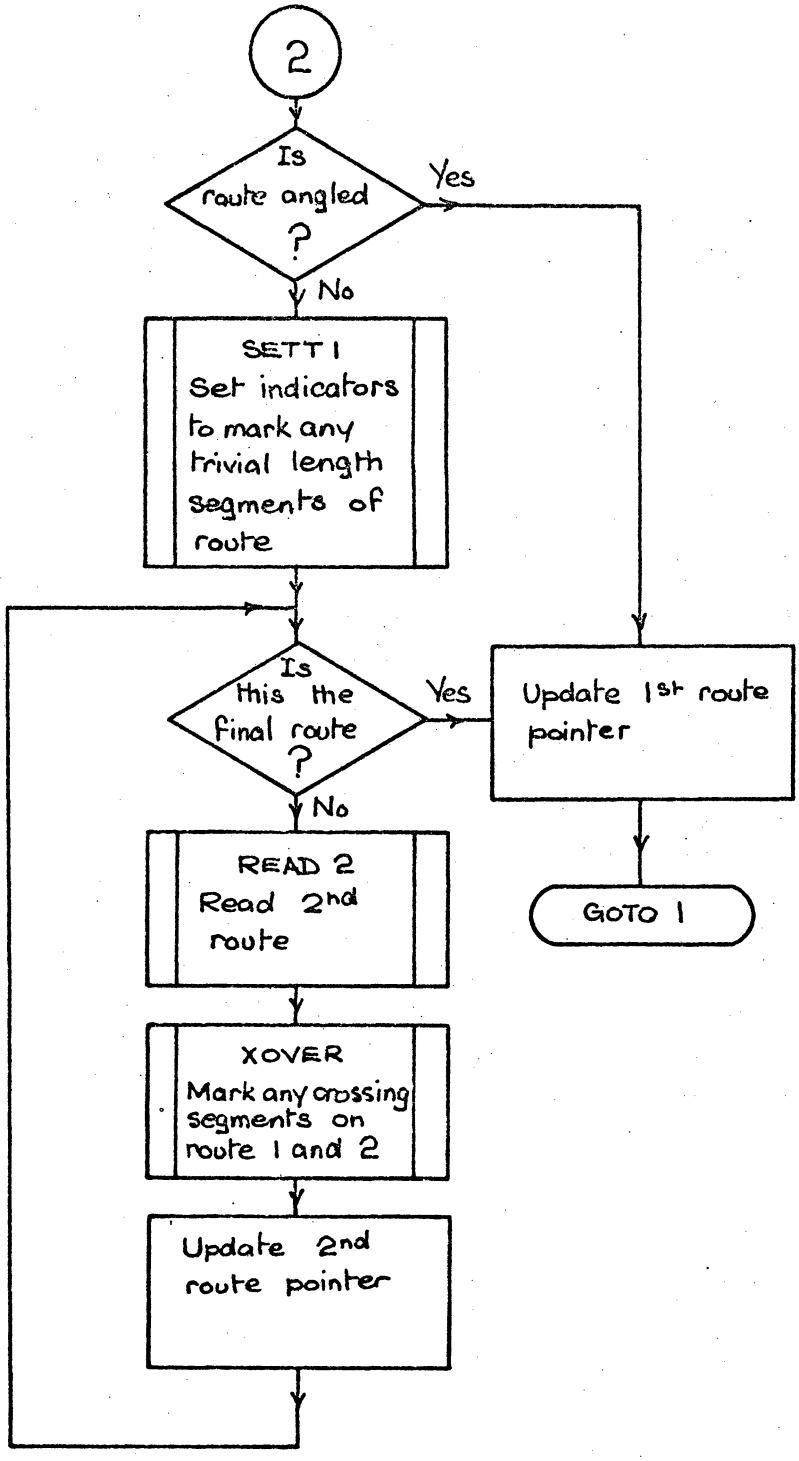
SUBROUTINE ZROUT3 STRUCTURE

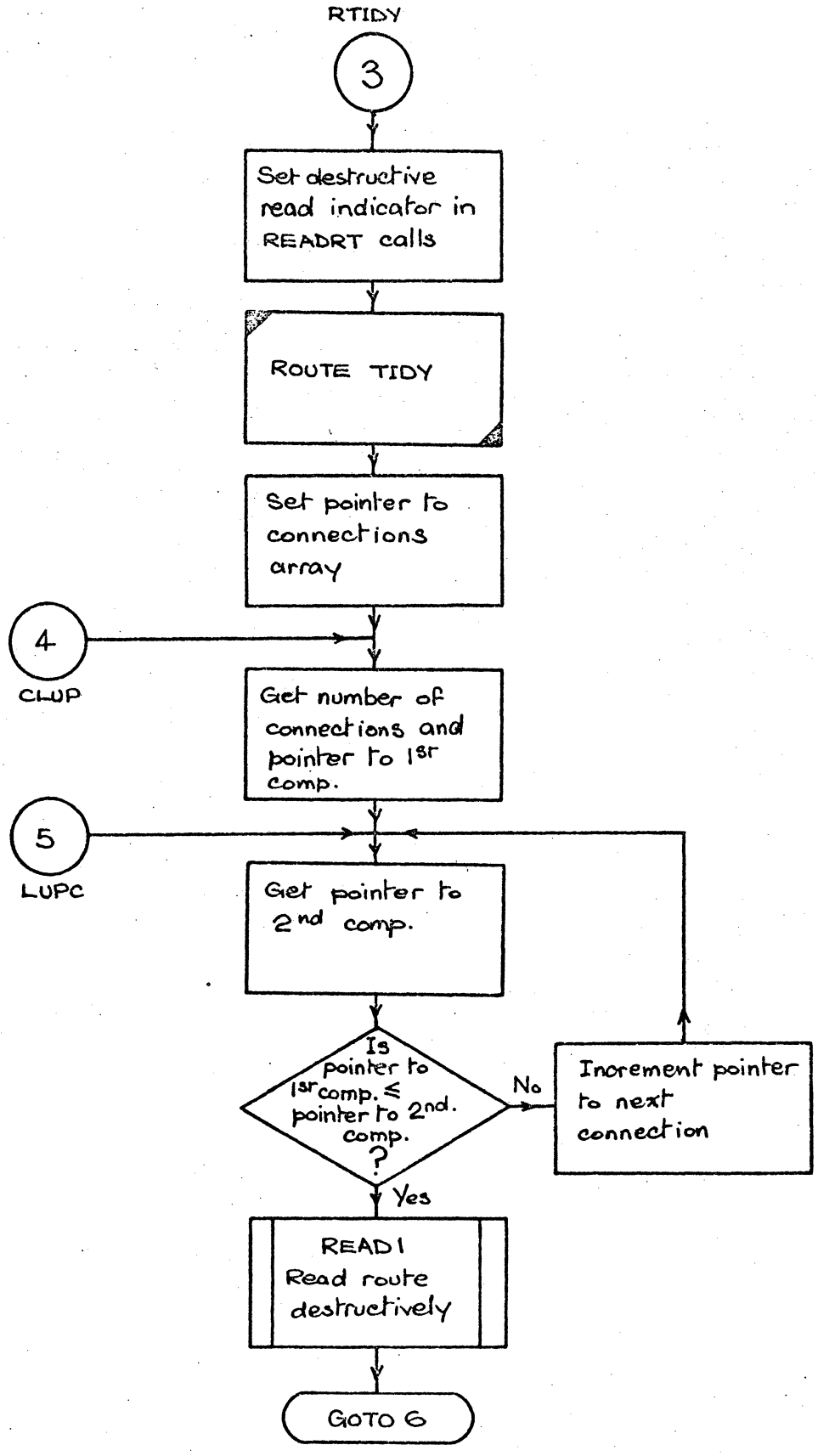


SUBROUTINE ZROUT3 FLOWCHART PART 1 OF 4

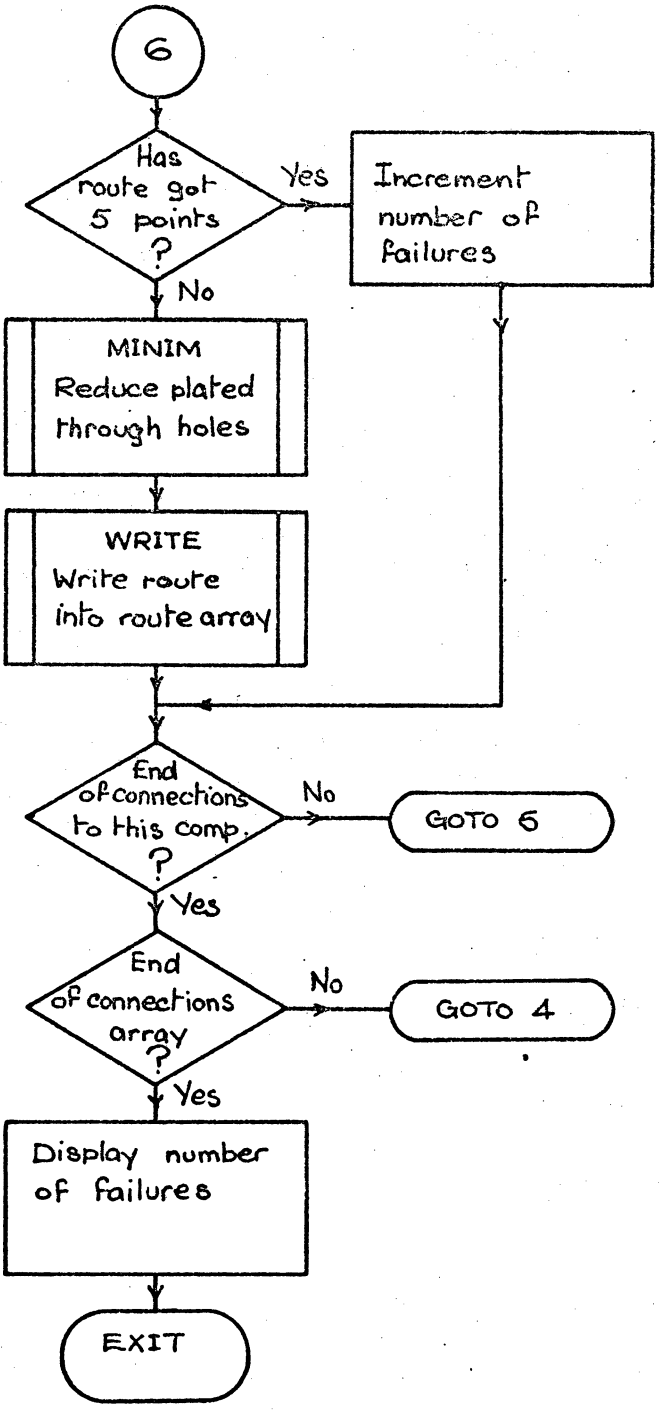


SUBROUTINE ZROUT3 FLOWCHART PART 2 OF 4

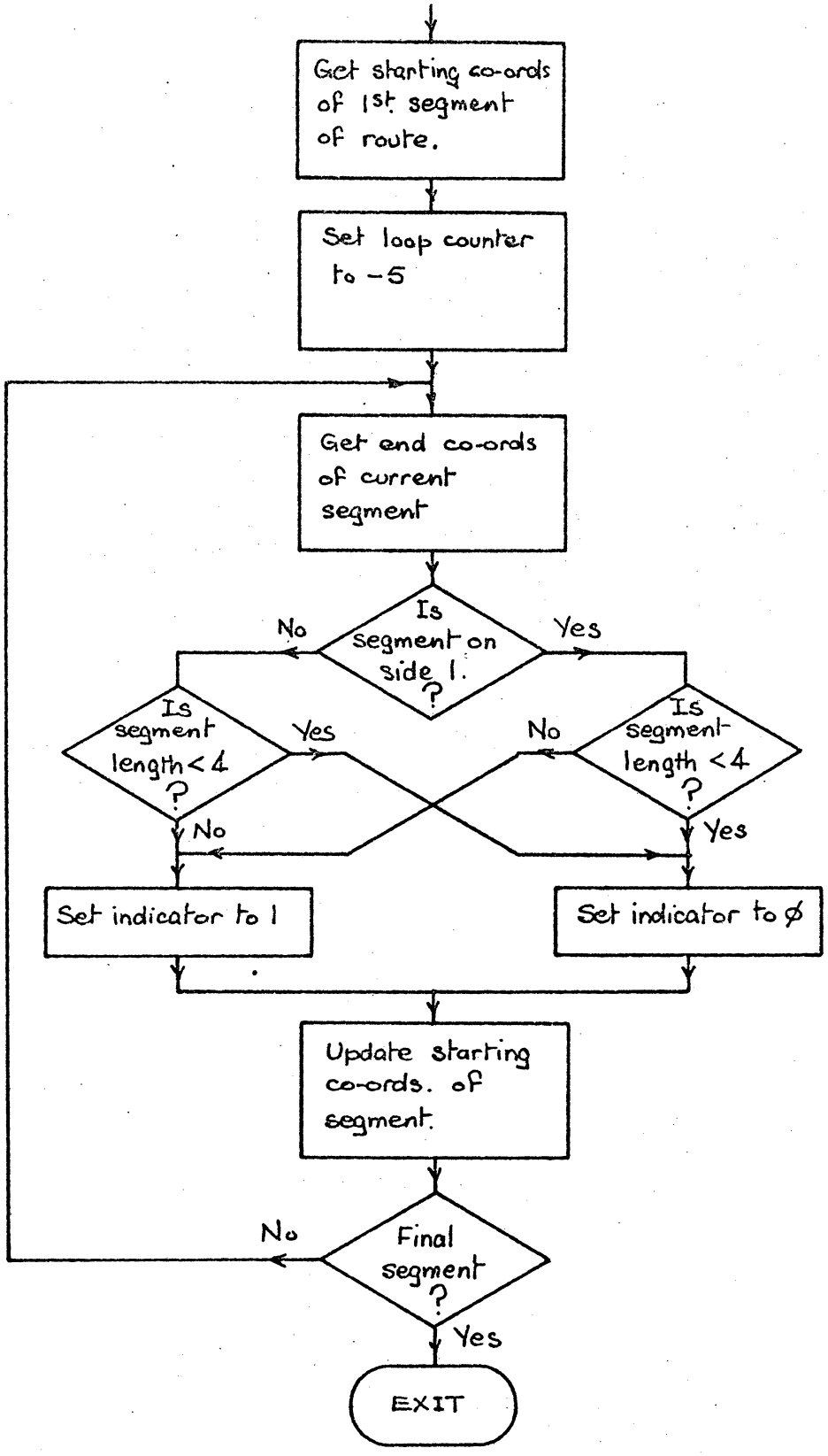



RTIDY
3
Set destructive read indicator in READRT calls
ROUTE TIDY
Set pointer to connections array
4
CLUP
Get number of connections and pointer to 1st comp.
5
LUPC
Get pointer to 2nd comp.
Is pointer to 1st comp. ≤ pointer to 2nd. comp. ?
No
Increment pointer to next connection
Yes
READ1
Read route destructively
GOTO 6

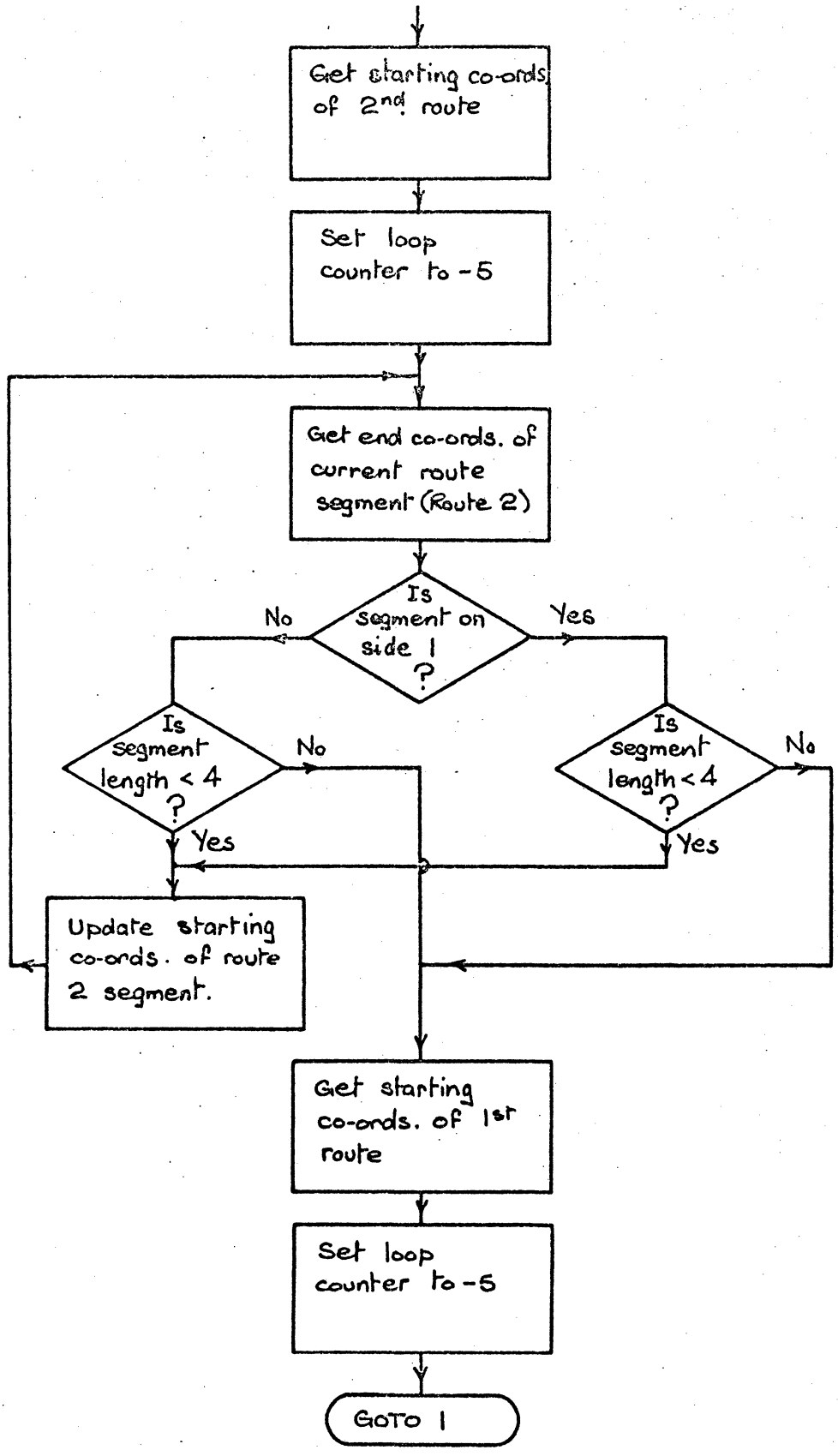
SUBROUTINE ZROUT3 FLOWCHART PART 4 OF 4



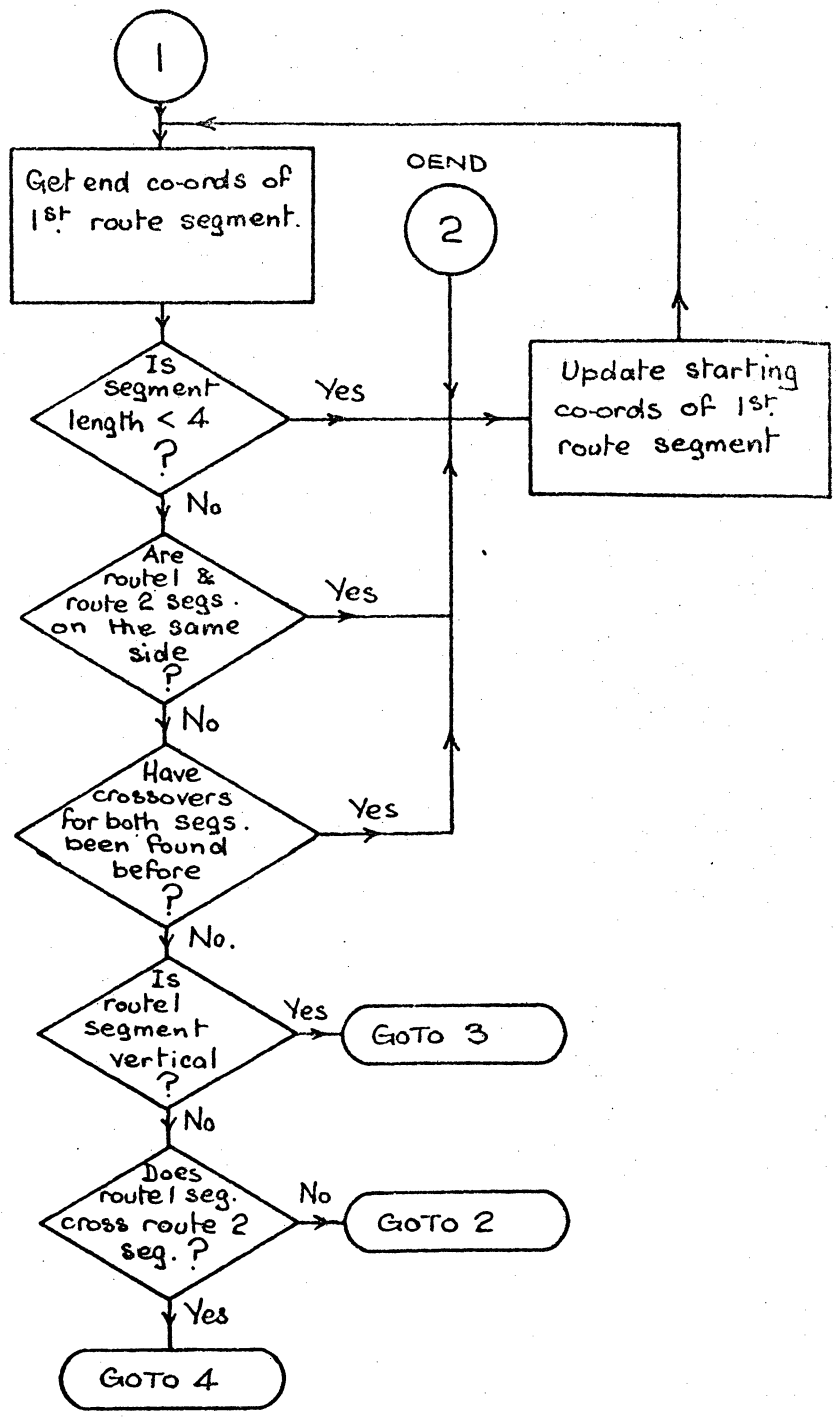
SUBROUTINE SETT1 FLOWCHART

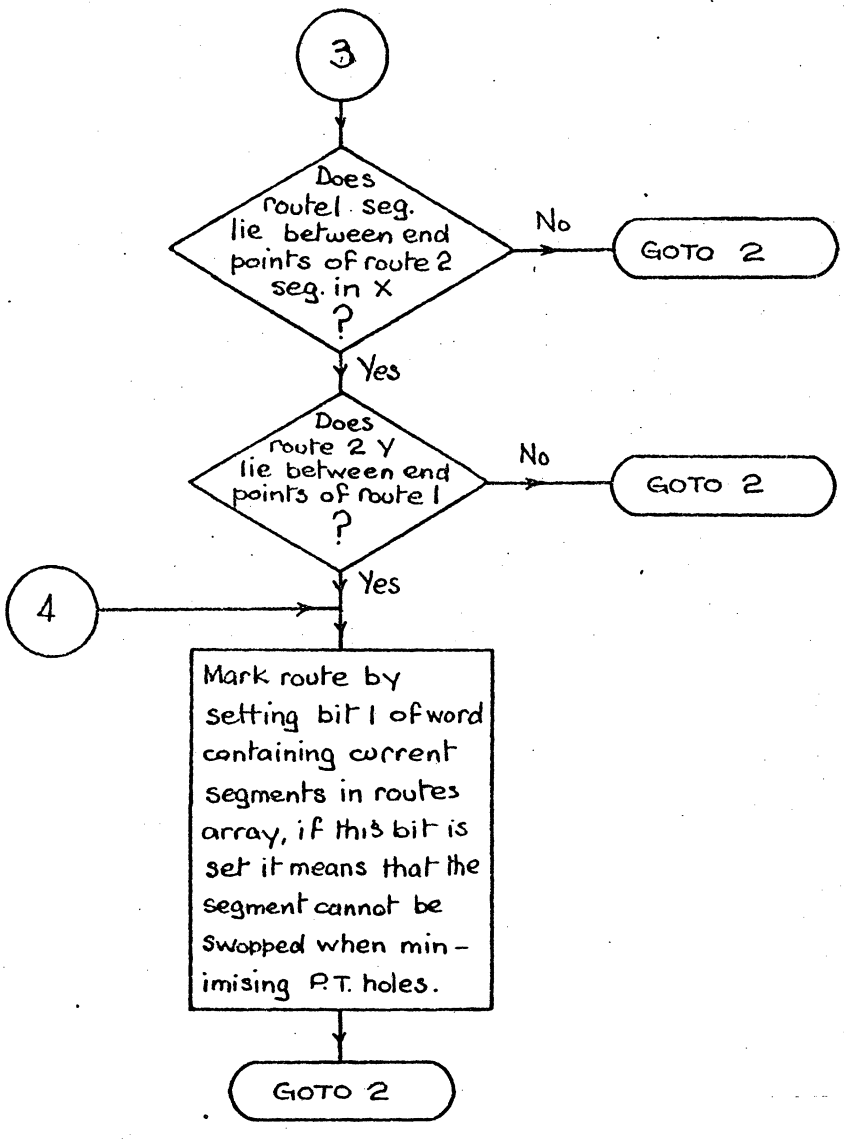


SUBROUTINE ZROUT3 FLOWCHART PART 1 OF 3



SUBROUTINE XOVER FLOWCHART PART 2 OF 3




3
Does route1 seg. lie between end points of route 2 seg. in X ?
No
GOTO 2
Yes
Does route 2 Y lie between end points of route 1 ?
No
GOTO 2
Yes
4
Mark route by setting bit 1 of word containing current segments in routes array, if this bit is set it means that the segment cannot be swopped when min-imising P.T. holes.
GOTO 2

SUBROUTINE MINIM FLOWCHART