

AD-A093 490

PRC INFORMATION SCIENCES CO MCLEAN VA  
RASTER TO LINEAL CONVERSION SOFTWARE.(U)  
OCT 80 S L FARRANCE

F/6 9/2

UNCLASSIFIED

PRC-R-3266

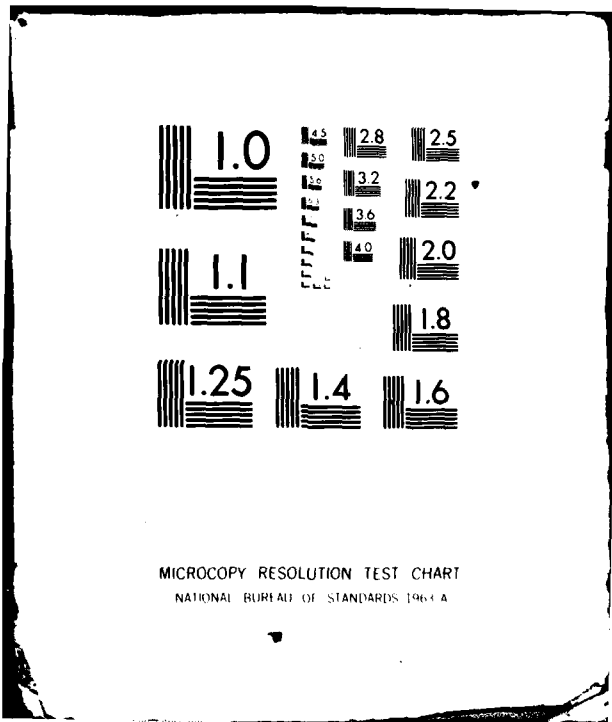
RADC-TR-80-324

F30602-79-C-0150

NL

1 - 1  
AT  
4 68300

END
DATE
FILED
2 81
DTIC



MICROCOPY RESOLUTION TEST CHART  
NATIONAL BUREAU OF STANDARDS 1963-A

RADC-TR-80-324  
Final Technical Report  
October 1980

LEVEL

AD A093490

# RASTER TO LINEAL CONVERSION SOFTWARE

Planning Research Corporation

Susan L. Farrance

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

DTIC  
ELECT  
JAN 6 1981


ROME AIR DEVELOPMENT CENTER  
Air Force Systems Command  
Griffiss Air Force Base, New York 12041


DDC FILE COPY

81 1 25 80

This report has been approved for release to the public in accordance with the provisions of Executive Order 13526, dated 12/18/2001. All information contained herein is unclassified unless otherwise indicated.

Approved for Release by NSA on 05-08-2014 pursuant to E.O. 13526

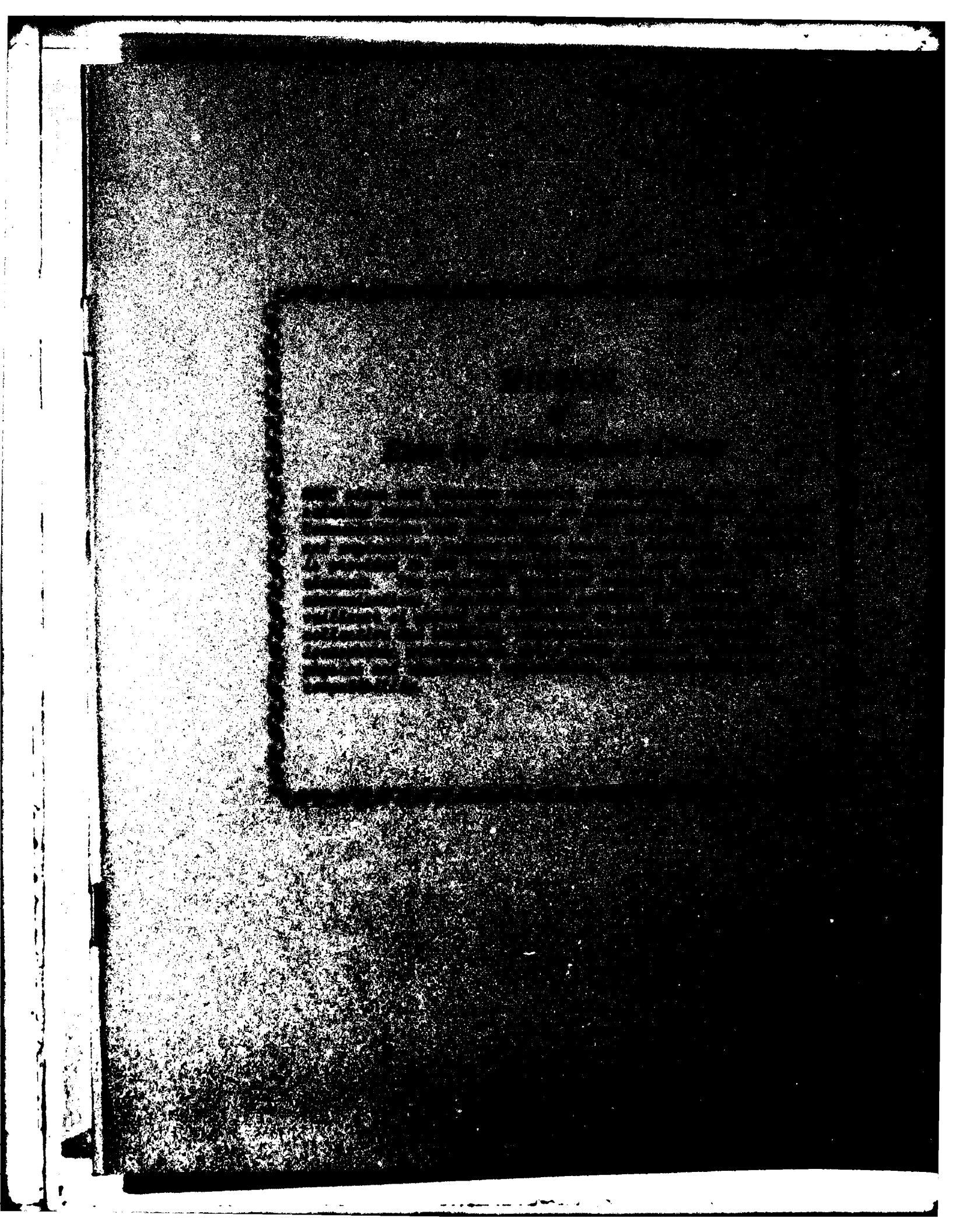
APPROVED:   
JOHN R. BAUMAN  
Project Engineer

APPROVED:   
OWEN R. LAWLER, Colonel, USAF  
Chief, Intelligence and Reconnaissance Division

FOR THE CHIEF:   
JOHN P. [unclear]  
Acting Chief, [unclear]

If your address has changed or if you are no longer receiving this mailing list, or if the information is incorrect, please notify [unclear] at [unclear] maintaining a current mailing list.

Do not return this copy. Thank you.



UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

19 REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM	
1. REPORT NUMBER RADC TR-80-324	2. GOVT ACCESSION NO. AD-AC93 490	3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle) RASTER TO LINEAL CONVERSION SOFTWARE		5. TYPE OF REPORT & PERIOD COVERED Final Technical Report 18 Jun 79 - 15 Aug 80	
6. AUTHOR(s) Susan L. Farrance		7. PERFORMING ORG. REPORT NUMBER R-3266	8. CONTRACT OR GRANT NUMBER(s) F30602-79-C-0150
9. PERFORMING ORGANIZATION NAME AND ADDRESS Planning Research Corporation/Information Sciences Company, 7600 Old Springhouse Road McLean VA 22102		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 64701B 43030334	17. $\emptyset$
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (IRRP) Griffiss AFB NY 13441		12. REPORT DATE October 1980	13. NUMBER OF PAGES 60
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Same		15. SECURITY CLASS. (of this report) UNCLASSIFIED	
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A	
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report) Same			
18. SUPPLEMENTARY NOTES RADC Project Engineer: John R. Baumann (IRRP)			
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Raster to Lineal Conversion Raster Processing Skeletonization Automated cartography			
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) The purpose of this document is to detail the current status of the Raster to Lineal Conversion Software. Also provided is a summary of work performed and recommendations for the future of the software.			

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

407126

TABLE OF CONTENTS

	<u>PAGE</u>
1.0 INTRODUCTION . . . . .	1-1
1.1 PURPOSE . . . . .	1-1
1.2 SCOPE . . . . .	1-1
1.3 SUMMARY OF WORK PERFORMED . . . . .	1-1
1.4 ORGANIZATION . . . . .	1-2
2.0 SYSTEM STATUS . . . . .	2-1
2.1 OVERVIEW OF ORIGINAL SYSTEM . . . . .	2-1
2.2 PHASE I . . . . .	2-2
2.2.1 PROBLEMS ENCOUNTERED . . . . .	2-2
2.2.2 SOLUTIONS . . . . .	2-2
2.2.3 STATUS . . . . .	2-2
2.3 PHASE II . . . . .	2-5
2.3.1 PROBLEMS ENCOUNTERED . . . . .	2-5
2.3.2 SOLUTIONS . . . . .	2-5
2.3.3 STATUS . . . . .	2-5
2.4 PHASE III . . . . .	2-5
2.4.1 PROBLEMS ENCOUNTERED . . . . .	2-7
2.4.2 SOLUTIONS . . . . .	2-7
2.4.3 STATUS . . . . .	2-7
2.5 PHASE IV . . . . .	2-7
2.5.1 PHASE IV - ORIGINAL . . . . .	2-10
2.5.2 PHASE IV - MODIFIED . . . . .	2-10
2.6 SUMMARY OF SYSTEM STATUS . . . . .	2-10
3.0 RECOMMENDATIONS . . . . .	3-1
APPENDIX A	A-1
APPENDIX B	B-1
APPENDIX C	C-1

Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By _____	
Distribution/ _____	
Availability Codes	
Dist	Avail and/or Special
A	

LIST OF FIGURES

		<u>PAGE</u>
FIGURE 2-1	RASTER TO LINEAL CONVERSION SOFTWARE . . . . .	2-3
FIGURE 2-2	RASTER-LINEAL PHASE I FLOW DIAGRAM . . . . .	2-4
FIGURE 2-3	RASTER-LINEAL PHASE II FLOW DIAGRAM . . . . .	2-6
FIGURE 2-4	RASTER-LINEAL PHASE III FLOW DIAGRAM . . . . .	2-8
FIGURE 2-5	RASTER-LINEAL PHASE IV (ORIGINAL) FLOW DIAGRAM . .	2-11
FIGURE 2-6	RASTER-LINEAL PHASE IV (MODIFIED) FLOW DIAGRAM . .	2-12
FIGURE C-1	FLOWCHART FOR SUBROUTINE AGDSF . . . . .	C-4
FIGURE C-2	FLOWCHART FOR SUBROUTINE CALMA . . . . .	C-9
FIGURE C-3	FLOWCHART FOR SUBROUTINE CHKWD . . . . .	C-12
FIGURE C-4	FLOWCHART FOR SUBROUTINE LIS . . . . .	C-16
FIGURE C-5	FLOWCHART FOR SUBROUTINE OUTPUT . . . . .	C-19
FIGURE C-6	FLOWCHART FOR SUBROUTINE PACK . . . . .	C-21



## Evaluation

Under this contract, raster to lineal data conversion software which had been developed at the Defense Mapping Agency Aerospace Center was implemented at the DMA Hydrographic/Topographic Center.

The software delivered provides DMAHTC with a base system upon which more comprehensive raster processing methodologies can be built. Future raster systems developed for production use will be able to take advantage of experience gained with this baseline software.

  
JOHN R. BAUMANN  
Project Engineer

## 1.0 INTRODUCTION

↙ The purpose of the Raster to Lineal Conversion Software is to produce lineal feature files from data which has been digitized by raster scanning of the cartographic source. The particular objectives of this effort are to identify and correct shortcomings of the existing Raster to Lineal Conversion Software, and, secondly, to design and implement output modules to provide multiple output formats to facilitate editing of manuscripts.

### 1.1 PURPOSE

This report will present the current status of the Raster to Lineal Conversion Software and provide a comprehensive report of the progress made during this effort. ↗

### 1.2 SCOPE

The Raster to Lineal Software system is structured into four modules, described as Phases. The current status of each of the four phases will be addressed with a list of the routines for each phase. All problems encountered in the software during the effort will be discussed and any solutions given. This report will also provide recommendations concerning future development of the software.

### 1.3 SUMMARY OF WORK PERFORMED

The original Raster to Lineal Software was obtained from DMAAC in July, 1979, and loaded on the UNIVAC system at DMAHTC during the period of late August to early September 1979. PRC was denied access to the UNIVAC system from October 1, 1979, until March 3, 1980, because of personnel and funding problems internal to DMAHTC. During this period the output modules for LIS, CALMA and AGDS were coded and documented. Also, the LIS module was unit tested on the Honeywell system at RADC in Rome, New York. Two test data tapes were located in the beginning of March. However, they were not considered viable test data because no identification of the type of data could be made. These tapes were used during the time period of March to June, 1980, to install the software on the UNIVAC system. Viable test data was received by PRC on June 25, 1980. Rigorous testing of the system was begun on this date. Section 2 of this report details the testing that was completed.

1.4 ORGANIZATION

Section 2 describes the original system, the current status of Phases I-IV, and the new output modules.

Section 3 will provide some recommendations for the future of the software.

## 2.0 SYSTEM STATUS

This section will describe in detail the current status of Phases I-IV of the Raster to Lineal Conversion Software.

### 2.1 OVERVIEW OF ORIGINAL SYSTEM

The purpose of the Raster to Lineal Conversion Software is the timely and cost-effective conversion of raster scanned cartographic data to a feature-oriented lineal format. To this end a system was originally developed at RADC on a DEC PDP-11/45 under the DOS operating system. This system provided proof of the validity of the approach. The next step was to implement the system on the UNIVAC 1108 under EXEC-8 for the Defense Mapping Agency Aerospace Center (DMAAC) in St. Louis. This was done via a bootstrap system that was developed and tested on the Honeywell 6180 at RADC using the GCOS operating system. These two systems are very similar, the difference being that the system running on the Honeywell 6180 processed data derived from the Automatic Color Separating Device (ACSD) while the system implemented on the UNIVAC 1108 accepts input from the RAPS raster scanner. The processing of the raster data and the final output was identical in both systems.

The Raster to Lineal Conversion Software was developed for RADC under Contract No. F30602-74-C-0334. The Final Technical Report for this system was delivered to RADC June, 1977.

The Raster to Lineal Conversion Software is modular in nature, and is organized into four distinct phases:

- o Phase I - Input. This phase reads raster formatted scanner data from magnetic tape, sections it, if requested, and writes the data onto disk. This phase also processes other pertinent user-supplied parameters. Phase I consists of 7 routines.
- o Phase II - Skeletonization. Phase II contains 2 routines. It performs the important task of reducing the raster lines to unit thickness. It also finds and classifies junction points.
- o Phase III - Linealization. This phase converts the raster data to linked strings of coordinates each representing a segment of a lineal feature. Phase III contains 23 routines.

- o Phase IV - Output. Phase IV reads and concatenates the lineal segment strings produced in the previous phase into logical features which are output in the appropriate format. Information concerning possible errors on the lineal output file is gathered, as well as statistical data about the features on the file. Phase IV is made up of 7 routines.

Thus there are a total of 32 separate routines which make up the four phases of the Raster to Lineal Conversion Software. These routines are all written in Fortran V. A diagram of the original system is shown in Figure 2-1.

It was this original software that was tested and modified in this effort. A description of the current status of the four phases described above is given in the following sections.

## 2.2 PHASE I

This group of routines accepts input from two 9-track tapes that are produced by the RAPS raster scanner. The raster data from these tapes is reformatted and output to a random access disk file. Sectioning is performed if required. User-supplied parameters such as resolution, scale, etc., are read in and put into a header file along with statistics determined by the Phase I processing.

### 2.2.1 PROBLEMS ENCOUNTERED

Only one problem surfaced during installation and testing of Phase I. The phase terminated abnormally and the system would not catalog the files when file space was exhausted.

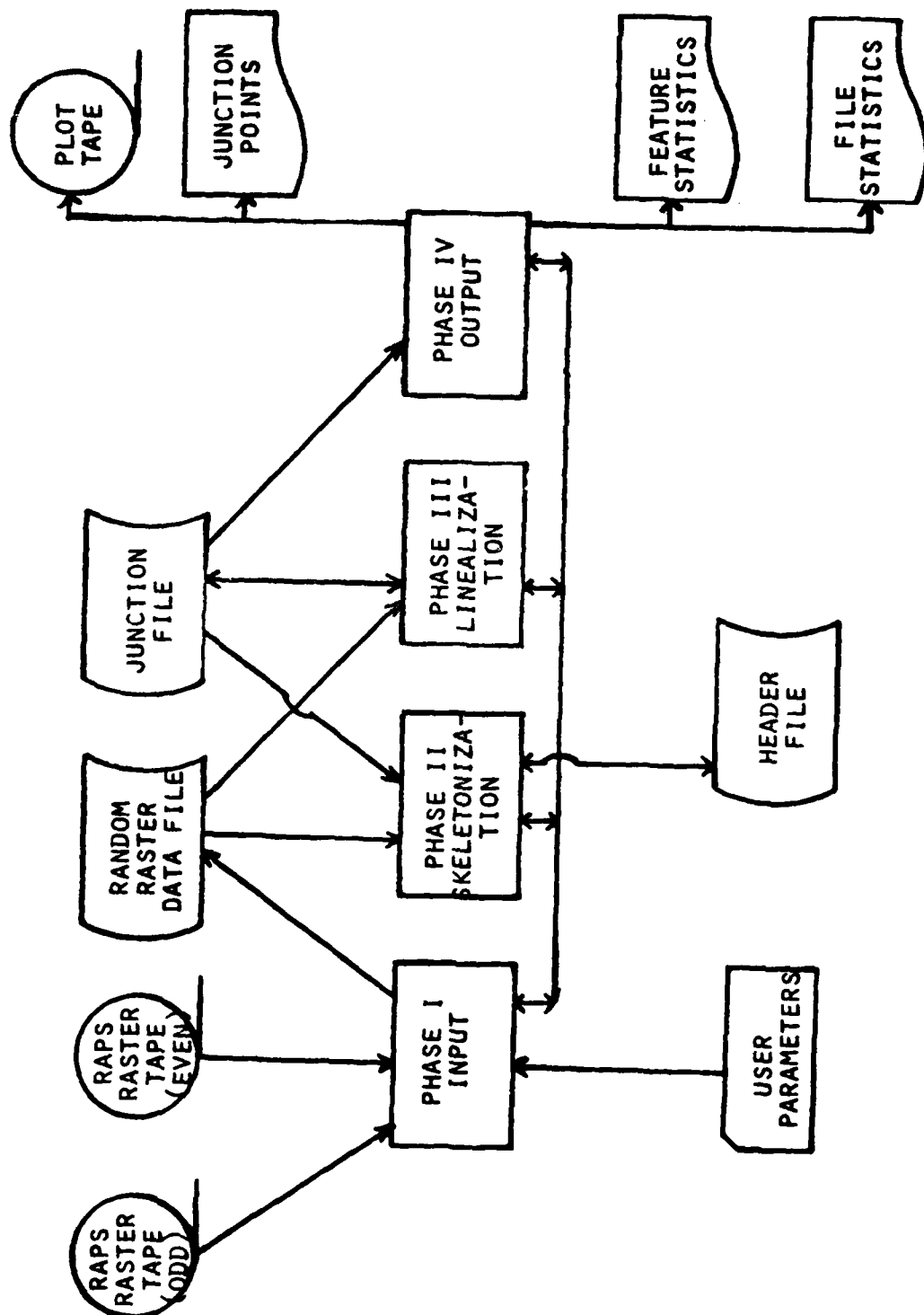
### 2.2.2 SOLUTIONS

To correct this problem statements were added to routine OUTPT1 to determine when this error occurs. The phase is then terminated normally, the files saved and an appropriate message is printed.

### 2.2.3 STATUS

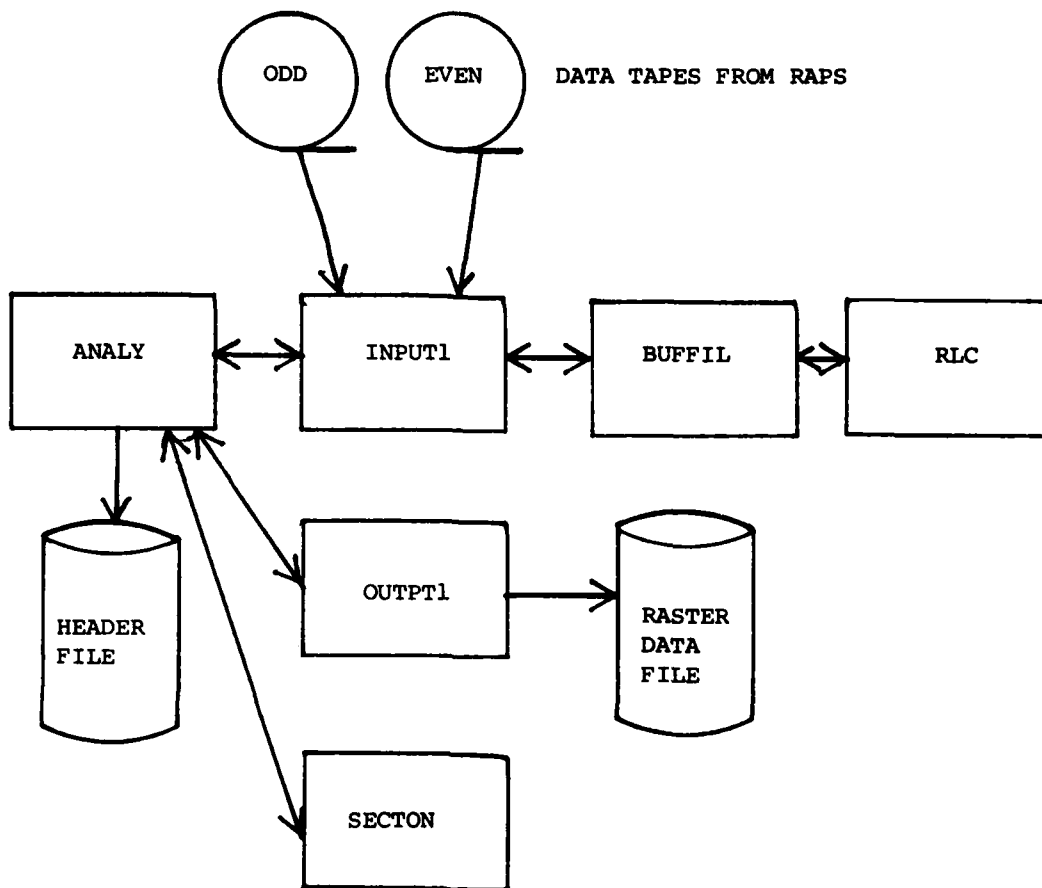
Phase I has been installed and completely tested on the UNIVAC at DMAHTC. See Appendix A for a list and short description of the modules in Phase I. A diagram of this phase is shown in Figure 2-2.

To process a section of data of 1000 x 1000 pixels of a relatively dense manuscript (maximum of 30 line crossings per scan line) a processing



RASTER TO LINEAL CONVERSION SOFTWARE

FIGURE 2-1



RASTER-LINEAL PHASE I FLOW DIAGRAM

FIGURE 2-2

time of approximately 4 minutes was required (36 seconds of CPU time and approximately 3 minutes of I/O time).

### 2.3 PHASE II

The second phase of the system applies the skeletonization algorithm to the raster data file and is repeated until all raster images of the cartographic features are reduced to lines of unit thickness. Since the amount of memory available has practical limits, only a limited piece of the raster file may be in memory at one time. Therefore, a sophisticated "staircasing" function is used to control the transfer of data between memory and the disk to minimize the amount of disk I/O necessary. Nevertheless, the CPU and I/O resources needed by Phase II are a considerable part of the total requirements for the entire Raster to Lineal Conversion Software.

During Phase II, all junction points are identified as to type; this information is output onto a disk file.

#### 2.3.1 PROBLEMS ENCOUNTERED

The only problem encountered during installation and testing of Phase II was the large amount of processing time needed to run. For the data section described above (1000 x 1000 pixels) Phase II required approximately 58 minutes of time, of which 53 minutes was I/O time.

#### 2.3.2 SOLUTIONS

The only practical solution to this problem using the current version of Phase II is to begin Phase I with a smaller section of the data. A version of the skeletonization routine, SKELIN, has been written and compiled on the UNIVAC but not tested because it was not part of the original system. This version has been designed to speed up the execution of this phase.

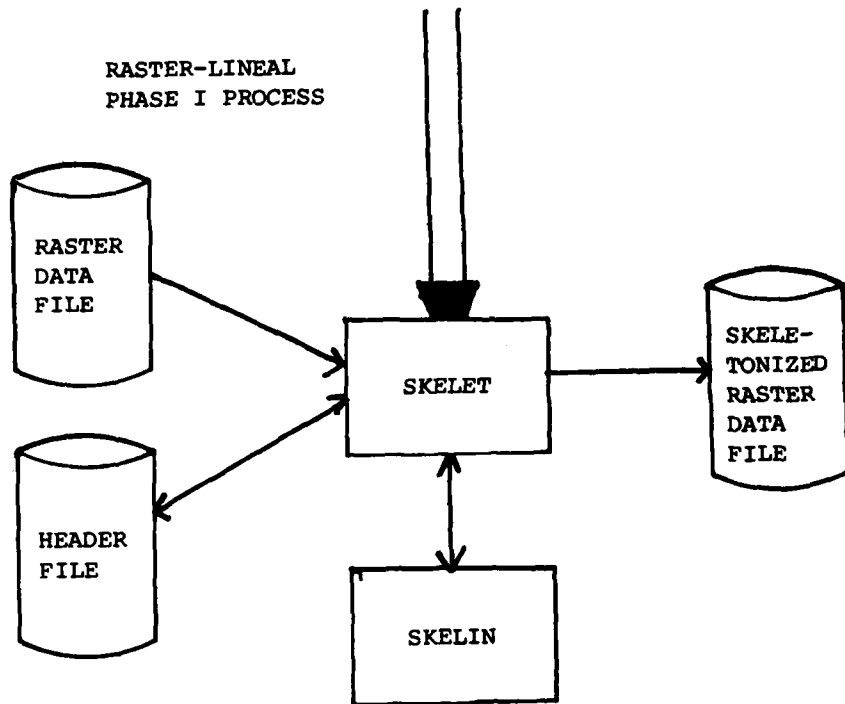
#### 2.3.3 STATUS

Phase II has been installed and completely tested on the UNIVAC at DMAHTC. See Appendix A for a list and short description of the modules in this phase. A diagram of Phase II is shown in Figure 2-3.

### 2.4 PHASE III

The linealization phase first calculates the number of raster scan lines that it can hold in memory at once, taking into account the memory





RASTER-LINEAL PHASE II FLOW DIAGRAM

FIGURE 2-3

available and the feature density of the manuscript. Strips of this size are read from the raster file and the raster elements are linked into segments by the line following algorithm. Information concerning these segments is saved in a master list until all segments belonging to a cartographic feature have been found, linked and output to disk.

The amount of bookkeeping necessary to create and maintain these linked feature segment lists is extensive. Thus the greatest amount of code is found in the Phase III Linealization module.

#### 2.4.1 PROBLEMS ENCOUNTERED

When a very dense section of data was used (74 line crossings per scan line), Phase III terminated abnormally. The parameter CORE, defined in the executive routine EXEC3, is currently set to 30000. This parameter is the size of an array that is used by many routines in Phase III. The storage allocation for the array was exceeded when the data was very dense resulting in termination of the run.

#### 2.4.2 SOLUTIONS

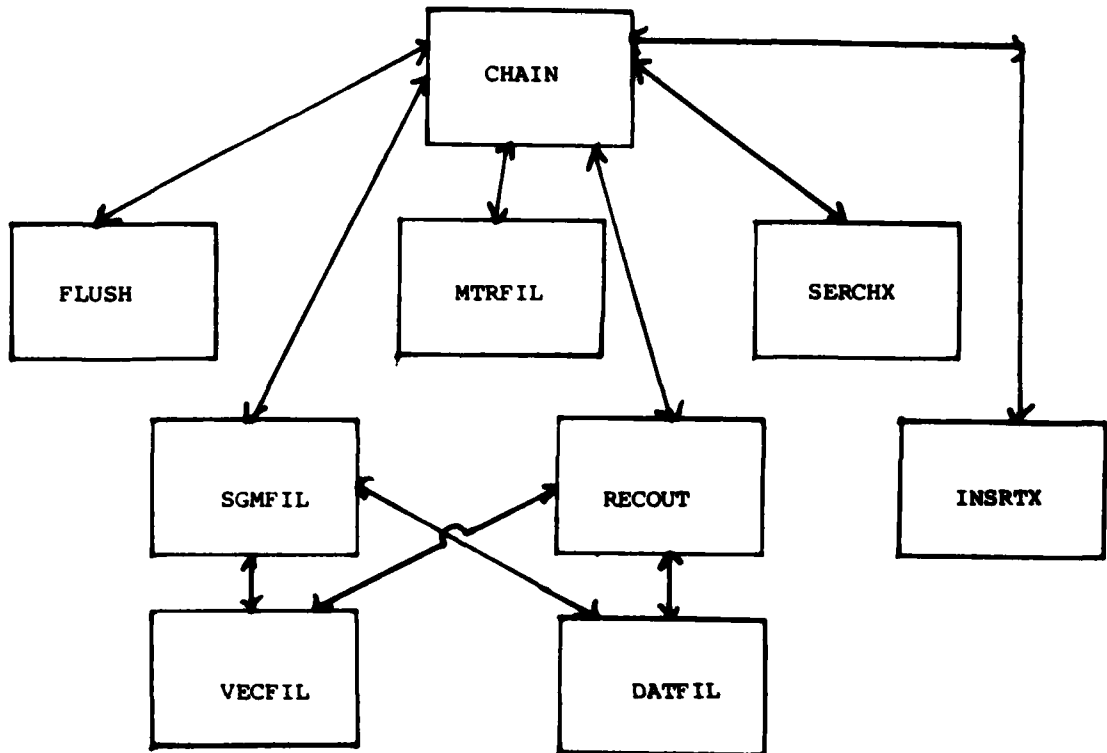
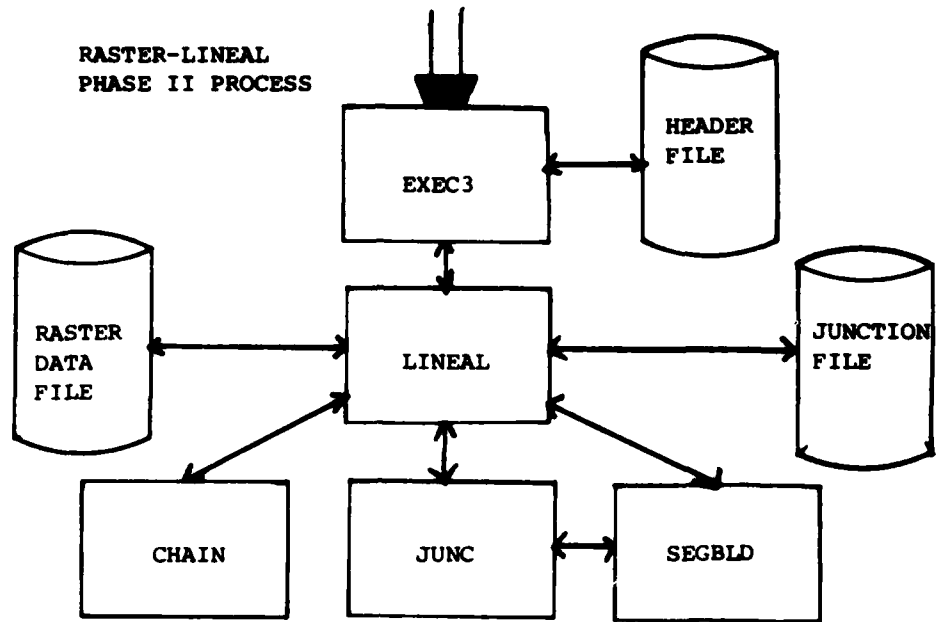
To correct this problem an initial attempt was made to increase the value of CORE to 40000. However, in doing this, Phase III exceeded the allowable size of 65K words of memory. Thus the only solution is again to begin with a smaller section of data in Phase I.

#### 2.4.3 STATUS

Phase III has been installed but not completely tested. A set of RAPS test data for which there was no original manuscript was run through this phase. This run terminated normally. Appendix A provides a list and description of all modules in Phase III and a diagram of the phase is shown in Figure 2-4.

#### 2.5 PHASE IV

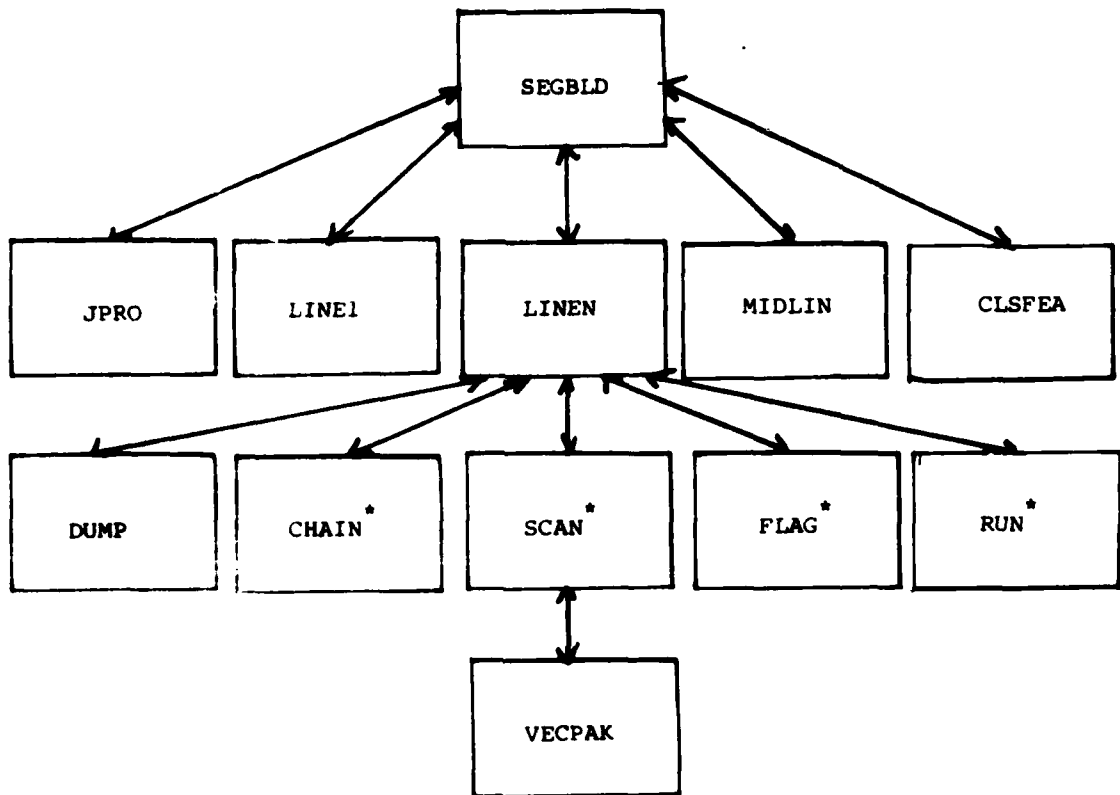
This final system phase formats output. Segment records output in the previous phase are retrieved by following the chains. These segments are then sorted to match up the proper segment end points. When an entire feature has been assembled, it is formatted for the appropriate output format written to the output tape. During output, statistics concerning the features are generated and reported along with a list of all junction points in the file.



RASTER-LINEAL PHASE III FLOW DIAGRAM

FIGURE 2-4

(Page 1 of 2)



\* These routines are called by each routine above,  
i.e., JPRO, LINE1, LINEN, MIDLIN and CLSFEA

RASTER-LINEAL PHASE III FLOW DIAGRAM

FIGURE 2-4

(Page 2 of 2)

#### 2.5.1 PHASE IV - ORIGINAL

The original version of Phase IV has been installed at DMAHTC. It will produce an output tape in Xynetics plotter format. This version was not completely tested. However, the phase ran to successful termination using the unknown test data described above. This run produced a successful plot. A diagram of this version of the phase is shown in Figure 2-5. A list of the routines is given in Appendix A.

#### 2.5.2 PHASE IV - MODIFIED

A modified version of Phase IV exists that has been compiled but not tested. This version will produce output in Xynetics plotter format, CALMA plotter format, AGDS format, and LIS format, based on a user-defined parameter. The modules for these output formats have been included in the compiled listing of this version of the phase. The CALMA and AGDS routines have not been tested. The LIS module has been unit tested on the HIS 6000 computer at RADC. A list of the routines in this version of Phase IV can be found in Appendix A, and a diagram is shown in Figure 2-6. Documentation of the output modules and routine OUTPUT can be found in Appendix C.

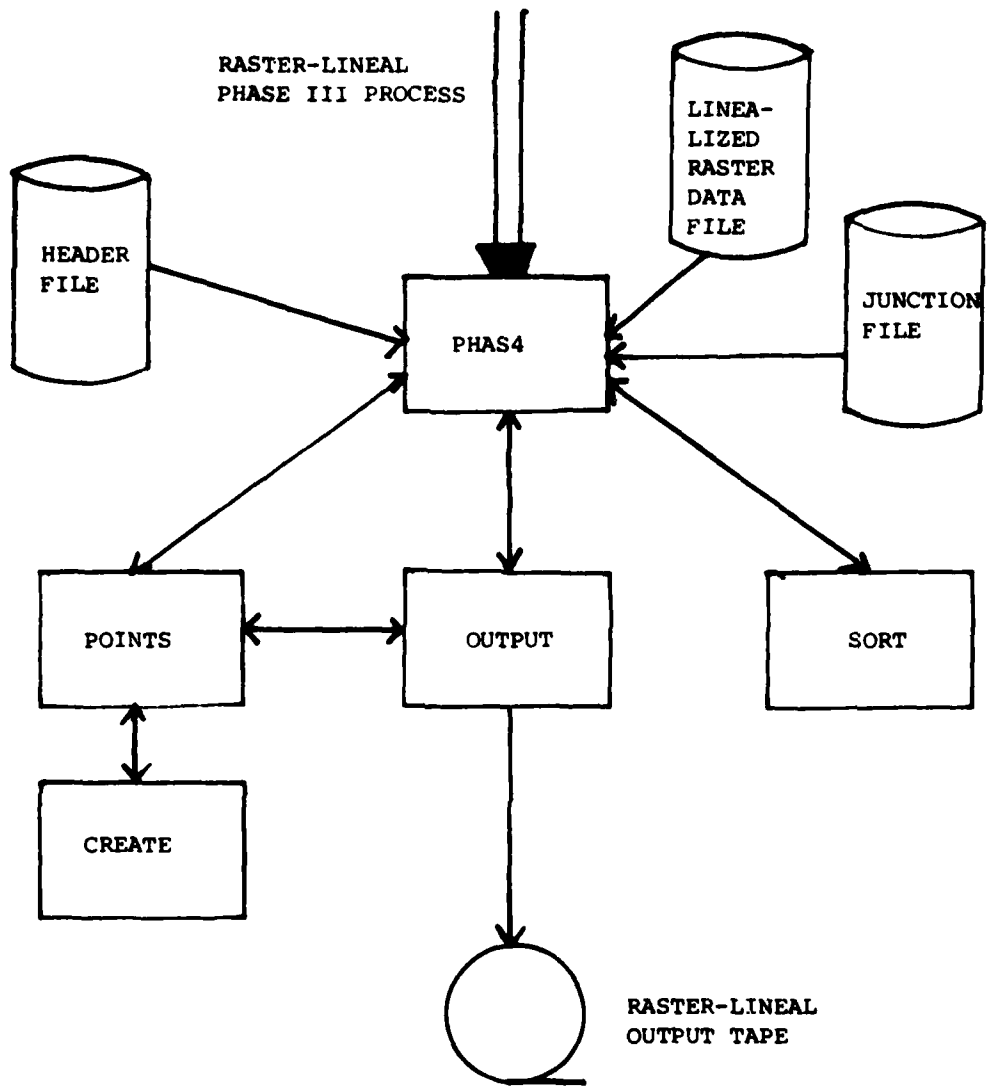
#### 2.6 SUMMARY OF SYSTEM STATUS

In summary, the following modules have been installed on the UNIVAC at DMAHTC:

- o Phase I
- o Phase II
- o Phase III
- o Phase IV - Original
- o Phase IV - With new output routines.

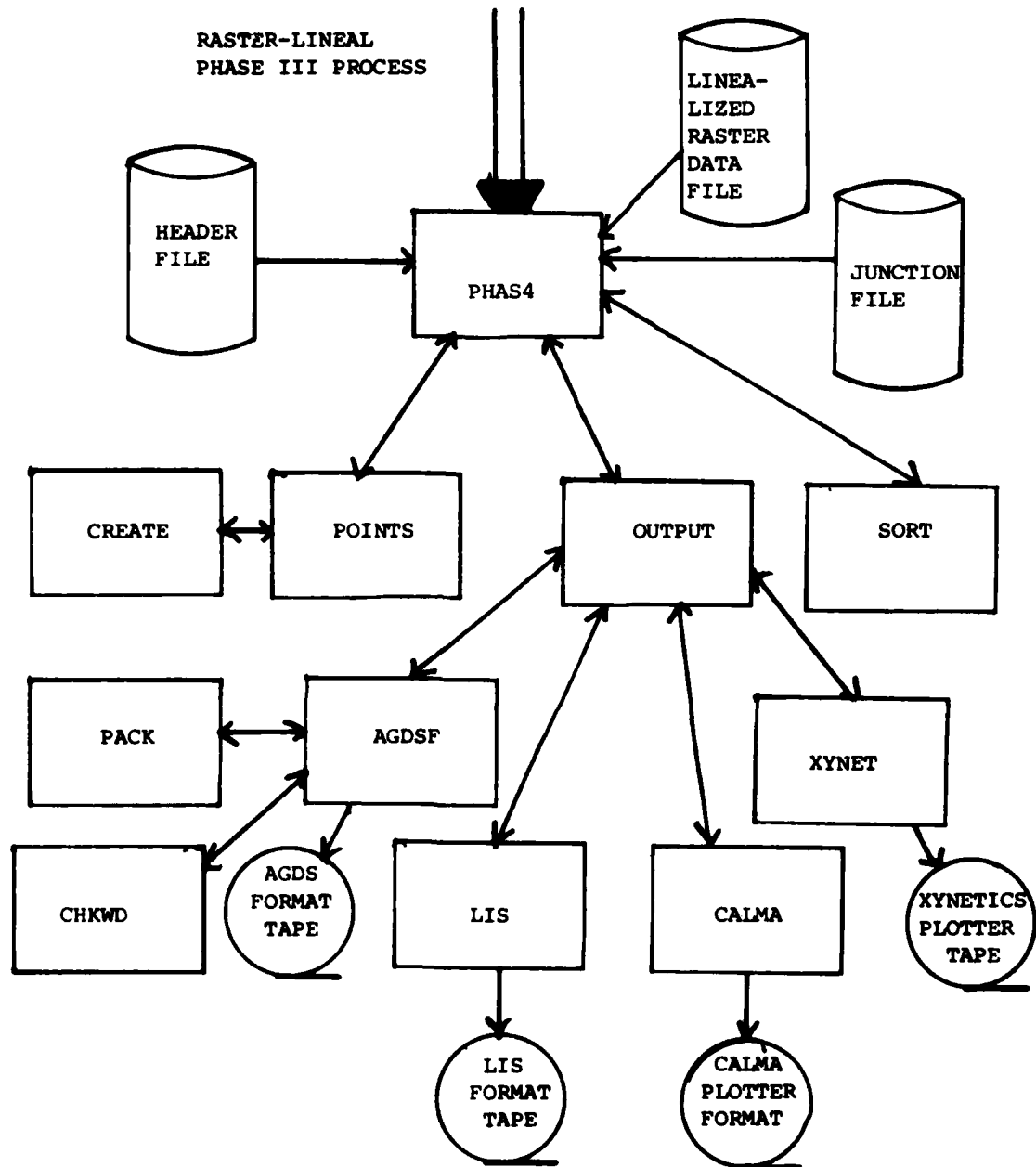
The runstreams to compile and execute these modules have been included on the software tape and all the execution runstreams are shown in Appendix B.

Phases I and II have been tested and minor modifications made. Phase III and the two versions of Phase IV have not been completely tested.



RASTER-LINEAL PHASE IV (ORIGINAL) FLOW DIAGRAM

FIGURE 2-5



RASTER-LINEAL PHASE IV (MODIFIED) FLOW DIAGRAM

FIGURE 2-6

### 3.0 RECOMMENDATIONS

PRC recommends two approaches to the future use of the Raster to Lineal Software. The first approach would be to continue working with the current system. The system as it stands is still in need of a certain amount of testing and improvement. The completely tested output modules would greatly increase the flexibility of the system and add to its value.

Some improvements to the current system should be made. The most important of these is optimization of the code. Optimization of the revised version of routine SKELIN in Phase II would be one such improvement.

Another improvement to the system could be to make Phase I more general, i.e., to accept input from a variety of scanners. The purpose of this is to remove the system dependency on the RAPS scanner and to provide flexibility. This would result in a more useful system, one that could provide multiple types of output from multiple input types. PRC recommends a system that would handle multiple input, i.e., RAPS, Hamilton-Standard color Raster Scanner, etc., and multiple output, i.e., AGDS, LIS, CALMA, XYNETICS, etc.

The second approach involves the redirection of the implementation philosophy to achieve the level of efficiency required in a production environment.

The processing workload on the UNIVAC processing system used to test the Raster to Lineal Conversion Software precluded comprehensive measurement of software efficiency. It is readily apparent, however, that the input/output to the SKELIN routine placed a major burden on this general purpose multiprocessing system. Major performance improvements in this operating environment are questionable.

The repetitive nature of the pattern recognition algorithm in SKELIN suggests the use of microprocessor technology. This routine tests each 3 x 3 pixel matrix for the existence of 256 possible combinations. For the 1000 x 1000 pixel test cell, this comparison was made in 5 minutes of CPU time, but consumed 53 minutes of I/O. Optimization of the raster to lineal conversion function clearly requires a configuration which matches I/O operations with processor throughput.



Image processing hardware has made significant advances through the effective balancing of high speed memory and processors. These configurations provide high throughput at minimal cost.

It is recommended that the raster to lineal conversion function be addressed through the use of dedicated processors similar in configuration to other image processing systems. Such a configuration incorporating microprocessors or possibly a small array processor can incorporate the design philosophy of the existing raster to lineal conversion software and achieve throughputs consistent with the data collection rates of current scanning devices.

In conclusion, PRC believes that raster processing in general, and specifically raster to lineal efforts, should be continued. The future of this technology seems very promising from the hardware viewpoint and software is needed to support this developing hardware. Despite setbacks in specific hardware development efforts, other raster scanners and plotters have been developed which demonstrated a capability to support cartographic production. The efficiency of these devices and their potential for eliminating labor intensive cartographic tasks warrants a concerted effort to incorporate raster scan technology in the production environment. A concerted effort to match the efficiency of the Raster to Lineal Conversion Software function is both necessary and feasible.

## APPENDIX A

The following is a list and brief description of all routines in the Raster to Lineal Software system.

### Phase I

- ANALY - executive routine for the phase; calculates minimums, maximums and highest number of line crossings per scan line.
- BIN - converts a string of binary mode data into start-stop pair data.
- BUFFIL - reads data from tape into buffer.
- INPUT1 - input routine for phase; acquires start-stop pairs from buffer.
- OUTPT1 - output routine for phase; will fill output buffer and then write to raster data file
- RLC - converts run-length coded data into start-stop pair data.
- SECTION - called when data is to be sectioned.

### Phase II

- SKELET - executive routine for the phase; allocates buffer space and does input and output.
- SKELIN - performs skeletonization on each line on a point by point basis, locates junction points.

### Phase III

- EXEC3 - executive routine for the phase; allocates buffer space based on calculations using Phase I and II statistics.
- LINEAL - input routine for linealization.
- JUNC - orders the processing of junction points.
- SEGBLD - sets up junction start points.
- JPRO - sets up the start point in junction following for SCAN routine.

LINE1 - handles all segments singularly crossing the first line of each scan block.  
 LINEN - resolves segments which begin and/or end on the N-th scan line but do not cross line 1.  
 MIDLIN - handles open ended segment that does not start or end on lines 1 or N.  
 CLSFEA - resolves closed features lying within the boundaries of the scan block.  
 SCAN - main segment following routine.  
 FLAG - builds the data buffer for routine CHAIN.  
 VECPAK - packs data into the buffer.  
 RUN - sets up the vector to be packed by VECPAK.  
 CHAIN - attempts to string segments together to form features and outputs them to a random file.  
 DATFIL - fills output buffer.  
 FLUSH - closes features that have had no activity, eliminates zeroed cells and zeroes activity flags.  
 INSRTX - packs data into the segment file array.  
 MTRFIL - files the master record buffer.  
 RECOUT - writes the master record buffer to the random file.  
 SERCHX - searches the segment directory file for a match for an X-coordinate.  
 SGMFIL - performs output of chain and segment records.  
 VECFIL - fills segment output buffer.

Phase IV - Original

PHAS4 - executive routine for the phase; reads the segment data and outputs it in X-Y format.  
 CREATE - creates a set of points from a starting point and a vector.  
 POINTS - places points in output buffer.  
 SORT - orders information in array.  
 OUTPUT - feeds the X-Y pairs to the XYNETICS plotter.

Phase IV - Modified

PH4 - executive routine; same as described above.  
CREATE - as described above.  
POINTS - as described above.  
SORT - as described above.  
OUTPUT - calls appropriate output module with X-Y pairs.  
XYNET - XYNETICS plotter output module.  
CALMA - CALMA plotter output module.  
LIS - Lineal Input System output module.  
AGDSF - AGDS output module.  
PACK - packs data into buffer for AGDSF routine.  
CHKWD - checks for end-of-block for AGDSF routine and will write  
block to temporary disk file.

```

PRC*WORKING(1),CMPHAS1
1 @PUN,A/TPR 110SUF,78AD00000001/PERCSE,8120SSCFAPPA,3,3D
2 @SYM PRINTS,RM1620 * COMPILFS AND MAPS PHASE 1
3 @FREE TPF$.
4 @ASG,T TPF$,F//8000
5 @USE A., PRC*WOPKING.
6 @COPY,S A.ANALY,TPF$.
7 @COPY,S A.BIN,TPF$.
8 @COPY,S A.BUFFIL,TEF$.
9 @COPY,S A.INPUT1,TPF$.
10 @COPY,S A.OUTPUT1,TPF$.
11 @COPY,S A.RLC,TPF$.
12 @COPY,S A.SECTION,TPF$.
13 @FOR,S TPF$.ANALY,TPF$.ANALY
14 @FOR,S TPF$.BIN,TPF$.BIN
15 @FOR,S TPF$.BUFFIL,TPF$.BUFFIL
16 @FOR,S TPF$.INPUT1,TPF$.INPUT1
17 @FOR,S TPF$.OUTPUT1,TPF$.OUTPUT1
18 @FOR,S TPF$.RLC,TPF$.RLC
19 @FOR,S TPF$.SECTION,TPF$.SECTION
20 @MAP,S A.MPHAS1,A.MPHAS1
21 @FREE A.
22 @FIN

```

```

PRC*WORKING(1),MPHAS1
1 IN ANALY
2 IN FIN
3 IN BUFFIL
4 IN INPUT1
5 IN OUTPUT1
6 IN RLC
7 IN SECTION

```

Phase I - Compile and Map

```

PRC*WORKING(1).EXPHAS1
1 @RUN,C/TPP 110SUF,1CDNYEGS0001/DPPCSF,5120SSSFAPPA,15,10
2 @SYM PRINT,1,1,RMI620 . EXECUTES PHASE 1
3 @ASG,T EVEN,0,U9S,C99K4P . EVEN TAPE 625C RP:
4 @ASG,T ODD,0,U9S,C9R70R . ODD TAPE 625R RP:
5 @USE 21,0,EVEN.
6 @USE 22,0,ODD.
7 @REWIND EVEN.
8 @REWIND ODD.
9 @DELETE,C PRC*RDATA.
10 @DELETE,C PRC*HEADER.
11 @ASG,CP PRC*RDATA,0,F//12800
12 @USF 2,0,PRC*RDATA.
13 @ASG,CP PRC*HEADER,0,F//1
14 @USF 3,0,PRC*HEADER.
15 @ASG,A PRC*WORKING.
16 @XOT PRC*WORKING.MPHAS1
17 $UCFR
18 RFS=1.
19 SCALE=100.0.
20 SECT=1.
21 SMINX=1.
22 SMAXX=1000.
23 SMINY=1.
24 SMAXY=1000.
25 $FND
26 @REWIND EVEN.
27 @REWIND ODD.
28 @FREE EVEN.
29 @FREE ODD.
30 @FIN

```

Phase I - Execution

```

PRC*WORKING(1).CMPHAS2
  1 @PUN.A/TPR 12DSUF.7BA000000001/DPRCSF.5120SSGFAPPA.2.FG
  2 @SYM PRINT$.PM1620 . COMPILES AND MAPS PHASE 2
  3 @FREE TPF$.
  4 @ASG.T TPF$.F//R000
  5 @USE A..PRC*WORKING.
  6 @COPY.S A.SKFLET.TPF$.
  7 @COPY.S A.SKELIN.TPF$.
  8 @FOR.S TPF$.SKELET.TPF$.SKELET
  9 @FOR.S TPF$.SKFLIN.TPF$.SKELIN
 10 @MAP.S A.MPHAS2.A.MPHAS2
 11 @FREE A.
 12 @FIN

```

```

PRC*WORKING(1).MPHAS2
  1 IN SKELIN
  2 IN SKFLIN

```

Phase II - Compile and Map

```

PRC*WORKING(1)*EXPHAS2
1  BRUN*B/TPR 120SUF*79A0000S0001/DPPCSF*F120*SSSFARFA*100*100
2  RSYM PRINTS*RM1622 * EXECUTES PHASE 2
3  PASS*A PRC*RDATA. * PASTER DATA FILE FROM PHASE 1
4  GASS*A PRC*HEADER.
5  @DELETE*C PRC*JUNC.
6  GASS*CP PFC*JUNC*F//7000 * JUNCTION FILE
7  @DELETE*C PRC*RSDATA.
8  GASS*CP PRC*RSDATA.*F//5800 * SKELETONIZED PASTER DATA FILE
9  @USE 1.*PRC*RSDATA.
10 @USE 2.*PPC*PDATA.
11 @USE 3.*PRC*HEADER.
12 @USE 4.*PRC*JUNC.
13 @XGT PRC*WORKING.MCHAS2
14 @FIN

```

Phase II - Execution



```

PRC*WORKING(1).CMPHASE
1  @PUN,P/TPR 130SUF,7BA70C0S0001/DFPCSF,5120SSSF,AFR1,5,100
2  @SYM PRINT$,RM1620 . COMPILES AND MAPS PHASE 3
3  @FREE TPF$.
4  @ASG,T TPF$.F//R000
5  @USE A,PRC*WORKING.
6  @COPY,S A.EXEC3,TPF$.
7  @COPY,S A.LINEAL,TPF$.
8  @COPY,S A.JUNC,TPF$.
9  @COPY,S A.SEGALD,TPF$.
10 @COPY,S A.JPRO,TPF$.
11 @COPY,S A.LINE1,TPF$.
12 @COPY,S A.LINE2,TPF$.
13 @COPY,S A.MIDLIN,TPF$.
14 @COPY,S A.CLSFEAT,TPF$.
15 @COPY,S A.SCAN,TPF$.
16 @COPY,S A.FLAG,TPF$.
17 @COPY,S A.VECPAK,TPF$.
18 @COPY,S A.RUN,TPF$.
19 @COPY,S A.CHAIN,TPF$.
20 @COPY,S A.DATFIL,TPF$.
21 @COPY,S A.FLUSH,TPF$.
22 @COPY,S A.INSRTX,TPF$.
23 @COPY,S A.MTRFIL,TPF$.
24 @COPY,S A.RECOUT,TPF$.
25 @COPY,S A.SERCHY,TPF$.
26 @COPY,S A.SGMFIL,TPF$.
27 @COPY,S A.VECFIL,TPF$.
28 @COPY,S A.DUMP,TPF$.
29 @FOR,S TPF$.EXEC3,TPF$.EXEC3
30 @FOR,S TPF$.LINEAL,TPF$.LINEAL
31 @FOR,S TPF$.JUNC,TPF$.JUNC
32 @FOR,S TPF$.SEGBLD,TPF$.SEGBLD
33 @FOR,S TPF$.JPRO,TPF$.JPRO
34 @FOR,S TPF$.LINE1,TPF$.LINE1
35 @FOR,S TPF$.LINE2,TPF$.LINE2
36 @FOR,S TPF$.MIDLIN,TPF$.MIDLIN
37 @FOR,S TPF$.CLSFEAT,TPF$.CLSFEAT

```

Phase III - Compile and Map

38	@FOR,S	TPFS	SCAN	TPFS	SCAN
39	@FOR,S	TPFS	FLAG	TPFS	FLAG
40	@FOR,S	TPFS	VECPAK	TPFS	VECPAK
41	@FOR,S	TPFS	RUN	TPFS	RUN
42	@FOR,S	TPFS	DUMP	TPFS	DUMP
43	@FOR,S	TPFS	CHAIN	TPFS	CHAIN
44	@FOR,S	TPFS	DATFIL	TPFS	DATFIL
45	@FOR,C	TPFS	FLUSH	TPFS	FLUSH
46	@FOR,S	TPFS	INSRTX	TPFS	INSRTX
47	@FOR,S	TPFS	MTRFIL	TPFS	MTRFIL
48	@FOR,S	TPFS	RECCUT	TPFS	RECCUT
49	@FOR,S	TPFS	SEPCHY	TPFS	SEPCHY
50	@FOR,S	TPFS	SGMFIL	TPFS	SGMFIL
51	@FOR,S	TPFS	VECFIL	TPFS	VECFIL
52	@MAP,S	A	MPHAS3	A	MPHAS3
53	@RCE	A			
54	@FIN				

Phase III - Compile and Map (cont.)

```

PRC*WORKING(1)*MPHAS3
1 IN EXEC3
2 IN LINEAL
3 IN JUNC
4 IN SEGPLD
5 IN JPRO
6 IN LINE1
7 IN LINEN
8 IN M*PLIN
9 IN CLSFEAT
10 IN SCAN
11 IN FLAG
12 IN VECPRK
13 IN RUN
14 IN CHAIN
15 IN DATFIL
16 IN FLUSH
17 IN INSRTX
18 IN MTRFIL
19 IN PECOUT
20 IN SERCHX
21 IN SGMFIL
22 IN VECFIL
23 IN DUMP

```

```

PRC*WORKING(1)*EXPHAS3
1 @RUN@/TPF 130SUF@78A0000S0001/DPRCSF@F120SSSFARPA
2 @SYM PRINTS@*PM1K20 * EXECUTES PHASE 3
3 @ASG@A PRC*HEADER.
4 @ASG@A PRC*JUNC.
5 @ASG@A PRC*RSDATA.
6 @USE 3@*PRC*HEADER.
7 @USE 2@*PRC*RSDATA.
8 @USE 4@*PRC*JUNC.
9 @XOT PRC*WOPKING.M=HAS3
10 @PMO@DE
11 @FIN

```

Phase III - Compile and Map (cont.)  
Execution

```

PRC*WORKING(1),CMEHAS4
1  BRUN,A/TYPE 140SUF,7B4000000001/DPRCSF,5120SSSFARPA,3,25
2  CCYM FRINT$,RM1620 . COMPILFS AND MAPS ORIGINAL VERSION PHASE 4
3  @FREE TPF$.
4  CASC,T TPF$,F//6000
5  FUSE A,PRC*WORKING.
6  @COPY,S A,PHAS4,TPF$.
7  @COPY,S A,CREATE,TPF$.
8  @COPY,S A,POINTS,TPF$.
9  @COPY,S A, SORT,TPF$.
10 @COPY,S A,OUTPUT,TPF$.
11 @FOR,S TPF$,PHAS4,TPF$,PHAS4
12 @FOR,S TPF$,CREATE,TPF$,CREATE
13 @FOR,S TPF$,POINTS,TPF$,POINTS
14 @FOR,S TPF$,SORT,TPF$,SORT
15 @FOR,S TPF$,OUTPUT,TPF$,OUTPUT
16 @SETC 30
17 CASC,A LIP3,PLTPAK.
18 @ADD,P LIP3,PLTPAK,PLTPAK
19 @MAP,S A,MPHAS4,A,MPHAS4
20 IN PHAS4
21 IN POINTS
22 IN SORT
23 LIF FTFILE
24 !! OUTPUT
25 IN CREATE
26 END
27 @FREE A.
28 @FIN

```

Phase IV - Compile and Map

```

PRC*WORKING(1)*EXPHAS4
1  @KUN*B/TPR 140SLF.78A000050001/00PCSF.5120SSSFARFA.10.3E
2  @SYM PRINT$*PMI620 . EXECUTE ORIGINAL PHASE 4
3  @AS$*T OUTTP**UCH.097*7M . OUTPUT TAPE (1F00PFI) FOR PLOT
4  @ASG*A PRC*HEADER.
5  @ASG*A PRC*JUNC.
6  @ASG*A PRC*RSDATA.
7  @UCF 3**PRC*HEADER.
8  @USE 2**PRC*RSDATA.
9  @USE 4**PRC*JUNC.
10 @USE 23**OUTTP.
11 @REWIND OUTTP.
12 @XOT PRC*WORKING.MPHAS4 . WILL OUTPUT A XYNETICS PLOT TAPE
13 $USER
14   OUTTP=1.
15 $END
16 @REWIND OUTTP.
17 @FREE OUTTP.
18 @FIN

```

Phase IV - Execution

```

PRC*WORKING(1).C.MNEW4
1 CRUN.A/TPR 100SF.7EAD0000S001/D.F120SSSFAPRA.7.50 . THIS WILL COMPILE&MVA
2 @SYM PRINT$.RM1620 . NEW PHASE 4
3 @FREE TPF$.
4 @ASG.T TPF$.F//P000
5 RUSF A.PRC*WORKING.
6 @COPY.S A.PH4.TPF$. . EXECUTIVE ROUTINE
7 @COPY.S A.CREATE.TPF$.
8 @COPY.S A.PTS.TPF$. . MODIFIED VERSION OF PGINTS
9 @COPY.S A.SORT.TPF$.
10 @COPY.S A.OUTFT.TPF$. . MODIFIED VERSION OF OUTPUT
11 @COPY.S A.XYNET.TPF$. . XYNET:CS OUTPUT MODULE
12 @COPY.S A.CALMA.TPF$. . CALMA OUTPUT MODULE
13 @COPY.S A.LIS.TPF$. . LIS OUTPUT MODULE
14 @COPY.S A.AGDS.TPF$. . AGDS OUTPUT MODULE
15 @COPY.S A.APACK.TPF$.
16 @COPY.S A.CHKWD.TPF$.
17 @FOR.S TPF$.PH4.TPF$.PH4
18 @FOR.S TPF$.CREATE.TPF$.CREATE
19 @FOR.S TPF$.PTS.TPF$.PTS
20 @FOR.S TPF$.SORT.TPF$.SORT
21 @FOR.S TPF$.OUTPT.TPF$.OUTPT
22 @FOR.S TPF$.XYNET.TPF$.XYNET
23 @FOR.S TPF$.CALMA.TPF$.CALMA
24 @FOR.S TPF$.LIS.TPF$.LIS
25 @FOR.S TPF$.AGDS.TPF$.AGDS
26 @FOR.S TPF$.APACK.TPF$.APACK
27 @FOR.S TPF$.CHKWD.TPF$.CHKWD
28 @SETC 30
29 @ASG.A LIB3*PLTPAK. . XYNETICS PLOT ROUTINE LIBRARY MUST BE ADDED
30 @ADD.P LIB3*PLTPAK*PLTPAK
31 @MAP.IS A.MNEW4.A.MNEW4

```

Phase IV (Modified) - Compile and Map

```

PRC*WORKING(1).MNEW4
 1 IN PH4
 2 IN PTS
 3 IN SORT
 4 LIB PTFILF
 5 IN OUTPT
 6 IN XYNET
 7 IN CALMA
 8 IN LIS
 9 IN AGDS
10 IN APACK
11 IN CHKWD
12 IN CREATE
13 END

```

```

PRC*WORKING(1).EXNEW4
 1 @RUN,B/TPR 140SUF,78A0000S0001/DPPCSF,5120SSSFARPA,10,25
 2 ASYM PRINT$,RM1620 . EXECUTFS NEW VERSION OF PHASE 4
 3 @ASG,T OUTTP,,UPH,C9797
 4 @ASG,A PRC*HEADER.
 5 @ASG,A PRC*JUNC.
 6 @ASG,A PRC*RSDATA.
 7 @USE 3,,PRC*HEADER.
 8 @USE 2,,PRC*RSDATA.
 9 @USE 4,,PRC*JUNC.
10 @USE 23,,OUTTP.
11 @PEWIND OUTTP.
12 @XQT PRC*WORKING.MNEW4
13 $USFR
14 OUTTP=2.
15 $FND
16 @REWIND OUTTP.
17 @FREE OUTTP.
18 @FIN

```

Phase IV (Modified) - Compile and Map (cont.)  
Execution

## AGDSF

### (1) Function

This routine will produce an output tape in the AGDS format. It receives X,Y pairs from subroutine OUTPUT and writes the tape.

### (2) Input/Output

The input to AGDSF is the X,Y coordinates of the linealized features contained in the common areas XLIST and YLIST. The tape in AGDS format is the output from the routine.

### (3) Process Description

One of four functions is performed on each call to AGDSF. These four functions are: initialization of the tape, begin a new feature, continue with the previous feature, and end-of-file processing. The function is determined by the value in common area WHAT.

If initialization is requested the subroutine begins by writing the tape header. This header contains information such as the file name, the user code, and the protection code. This information is input as part of the namelist in Phase IV. This tape header is logical block 0 and is 14 bytes long.

The initialization section next sets up the Group Header, Type 16 data, for the first block of the tape. This information is stored in array BUF16. The last position will be filled during the end-of-file processing.

When a new feature is encountered the routine first checks to see if this is the first feature. If it is, the beginning information is stored in the array TBUF, one item per word. The word and block number for the location of the word offset to the next group header is retained. Also, the block and word locations for storing the range rectangle is held in array R. If the new feature is not the first this information is filled in. If the block number retained is not the same as the current block, then that block is read and the data stored in the correct word locations.

Each time an item is stored in array TBUF, which holds all data (one item per word), the subroutine CHKWD is called. If this routine determines that the end of the block has been reached it writes this block to a temporary random disk file. Each "block" is a 256 word record on this file.



After the group header information has been processed, the routine begins to process the X,Y pairs contained in XLIST and YLIST. These values are stored in TBUF one per word. Each X and Y value is compared against the current range rectangle for that feature. If the values are larger than the maximum X and Y, or smaller than the minimum X and Y, the maximum and minimum values are adjusted accordingly.

When the continuation of a feature is processed it is handled as described in the above paragraph.

The end-of-file processing begins by filling in the offset and range rectangle for the last feature processed. Then the last position pointer for the Type 16 data is filled in and this block written as logical block 1 to the temporary disk file. The scan-graphic end-of-file is then stored in the last block and this block written.

The data is now packed using subroutine PACK. The blocks are sequentially read from the temporary disk file, packed and then written to the tape. After all blocks are processed the end-of-file mark is written on the tape and the control returns to the calling routine.

(4) Program Variables

Common Variables:

AGBUF	- buffer holding packed data
TBUF	- temporary buffer holding data 1 item/word
BUF	- temporary buffer
BUF16	- buffer holding data for Group 16
HDWD	- holds word location for offset to next group
HDBLK	- holds block location for offset
R	- hold block and word locations for current feature range rectangle
RAN	- holds range rectangle values for current feature
FIRST	- set to 1 after first feature
OFFSET	- word offset to next group header
BLK	- current block
WD	- current wd
ICT	- feature counter

Variables:

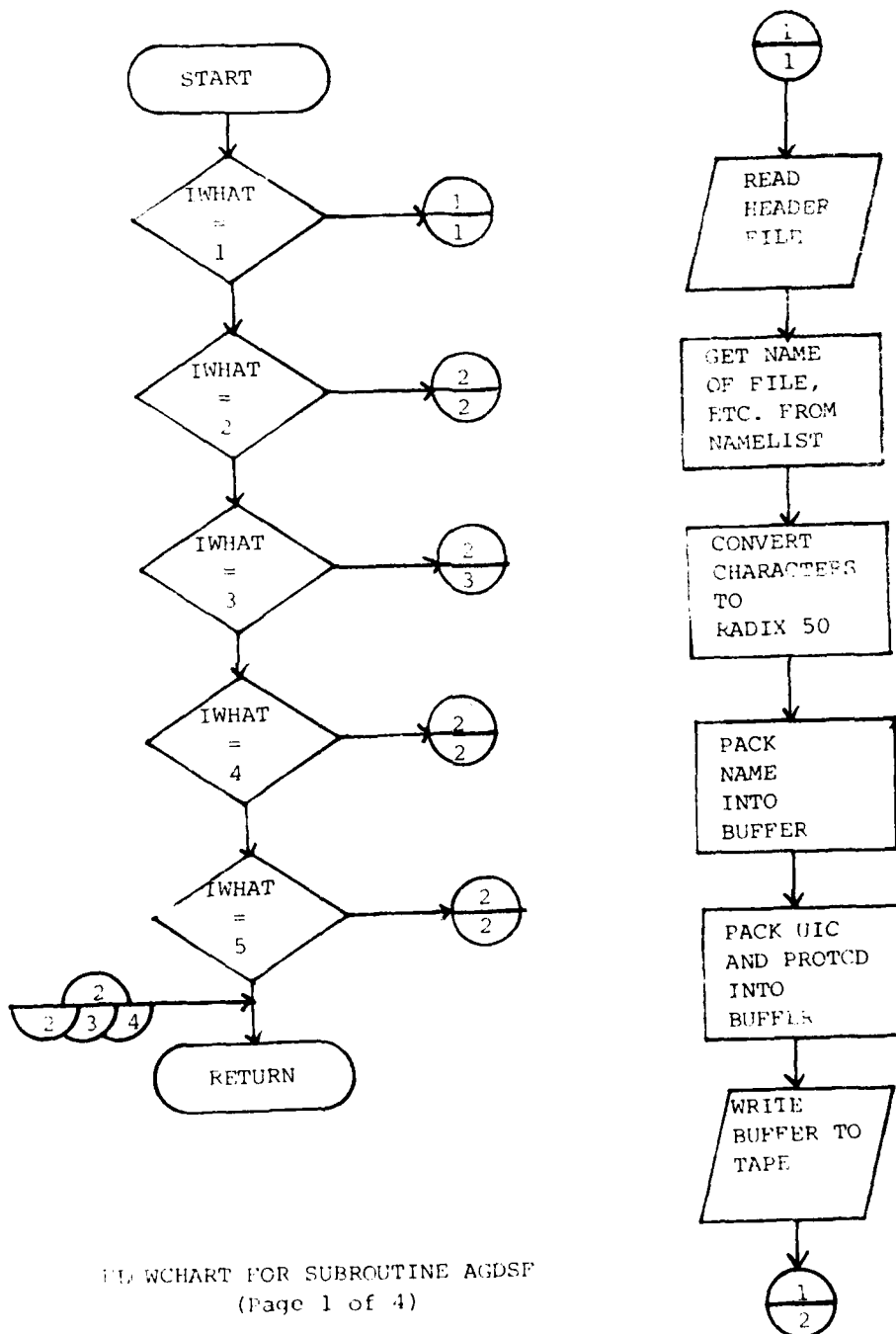
SET           - all bits set  
CLEAR         - all bits clear  
X             - X coordinate \* resolution  
Y             - Y coordinate \* resolution

Also COMMON areas:

RESOLU        }  
BOUREC        }  
UNIT           }  
XLIST         }  
YLIST         }  
HMANY         }  
WHAT           } - see OUTPUT for a description of these areas

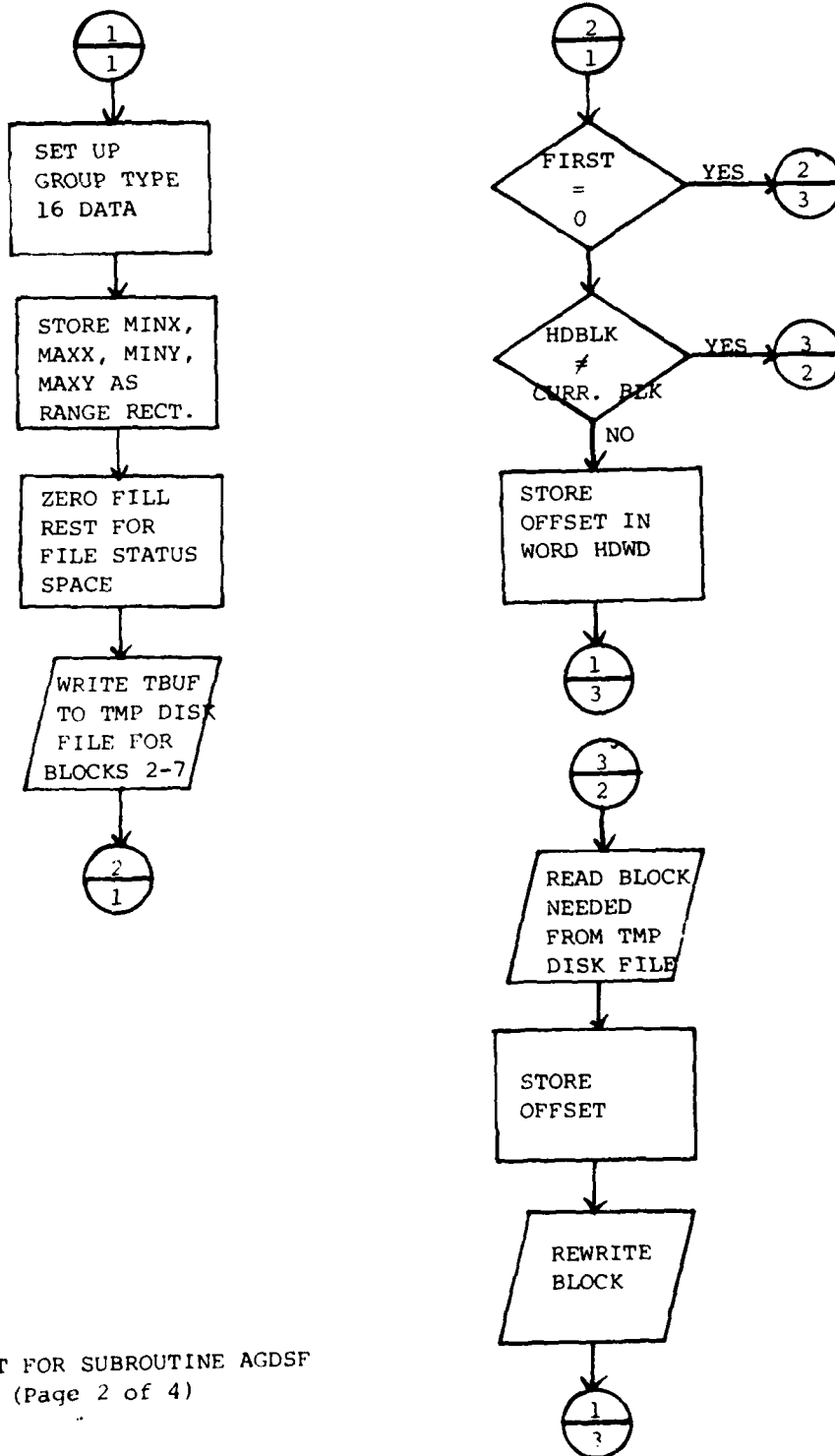
(5) Relationship to Other Routines

AGDSF is called by routine OUTPUT when the value of IFORM is four (4). AGDSF uses NTRAN to write to the tape. Subroutines CHKWD and PACK are called from this routine only.



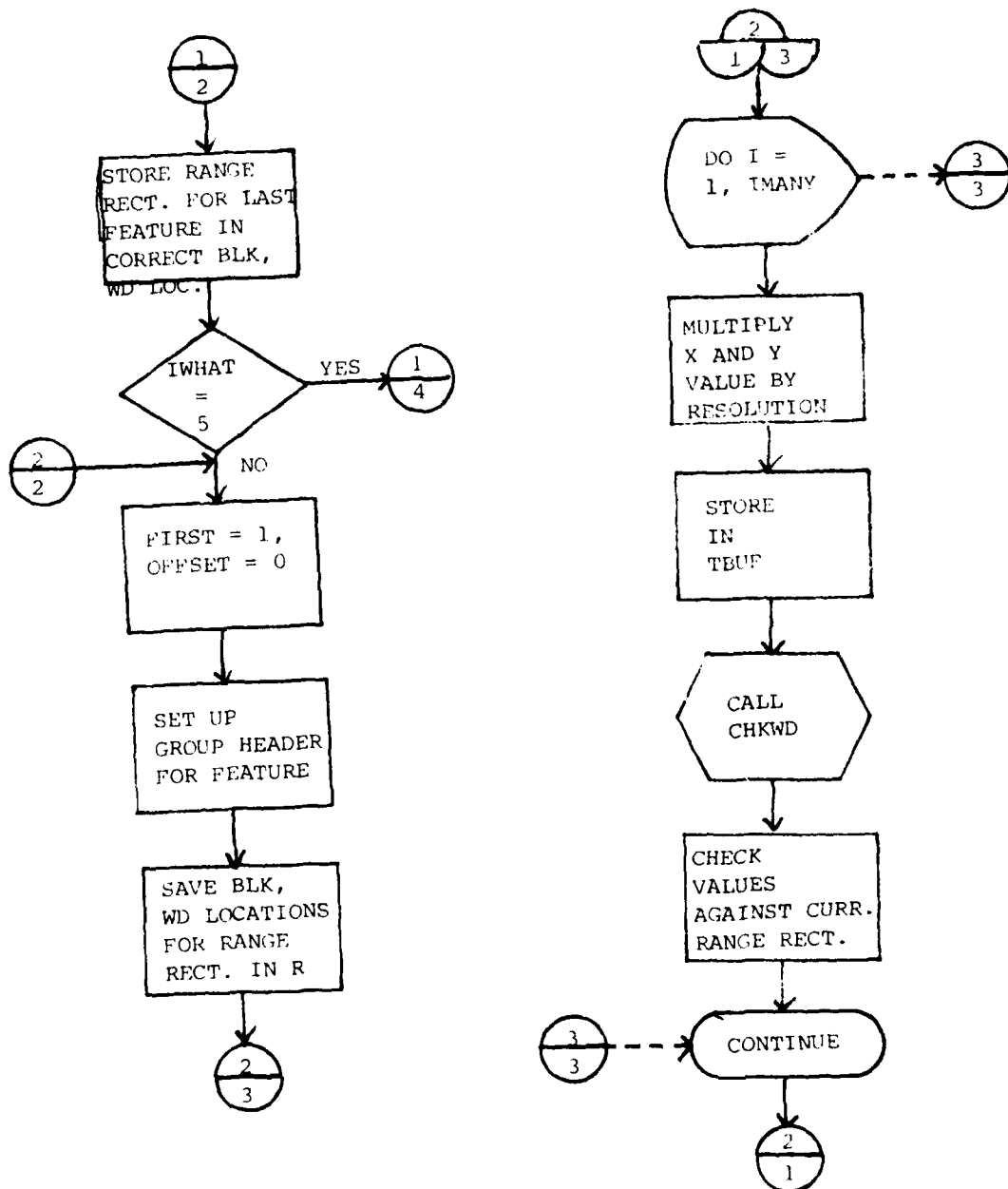
FLOWCHART FOR SUBROUTINE AGDSF  
(Page 1 of 4)

FIGURE C-1



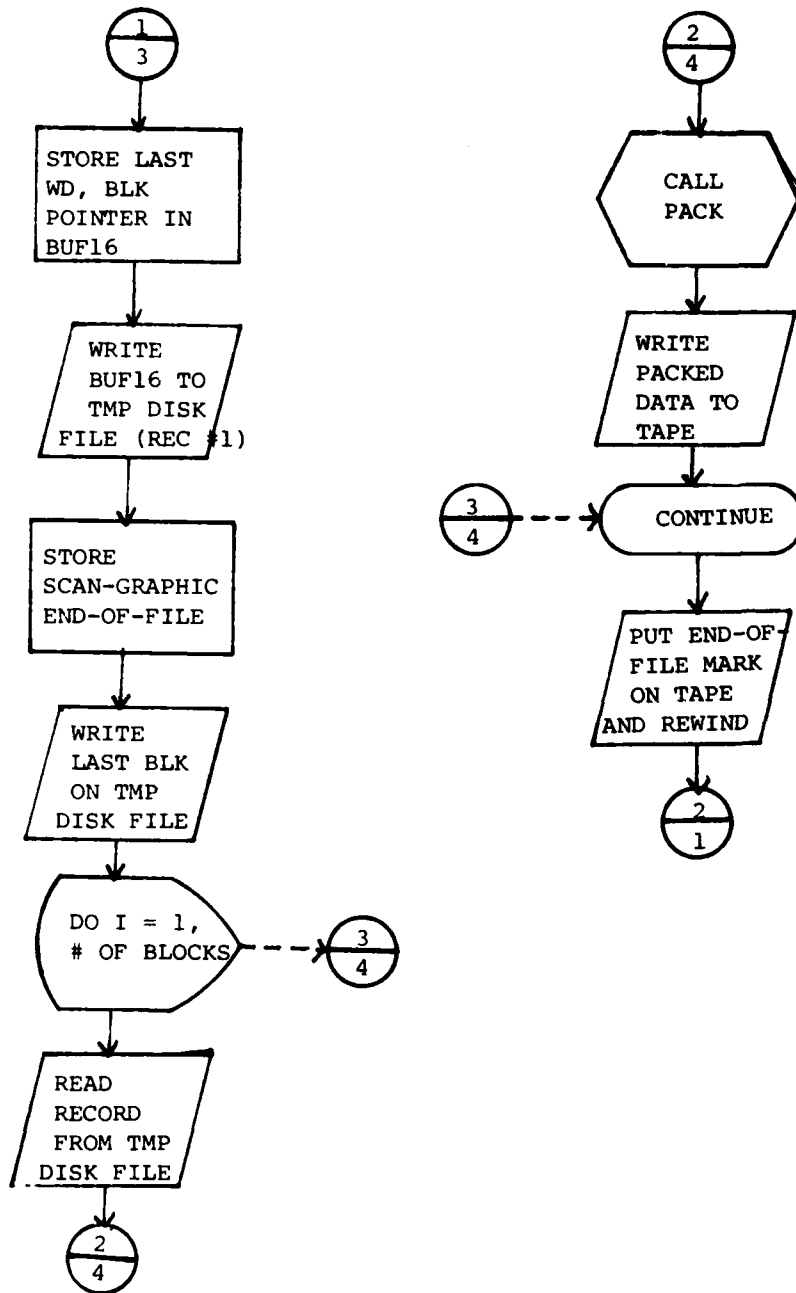
FLOWCHART FOR SUBROUTINE AGDSF  
(Page 2 of 4)

FIGURE C-1



FLOWCHART FOR SUBROUTINE AGDSF  
(Page 3 of 4)

FIGURE C-1



FLOWCHART FOR SUBROUTINE AGDSF  
(Page 4 of 4)

FIGURE C-1

## CALMA

### (1) Function

CALMA converts the integer X and Y coordinates of each lineal feature into the CALMA output format and writes them to the output tape.

### (2) Input/Output

Input to CALMA is a list of coordinates in COMMON areas XLIST and YLIST. Output consists of a 9-track tape in CALMA format.

### (3) Process Description

Routine CALMA performs one of four functions depending on the value in the COMMON area WHAT. These are initialization, start new feature, continue with previous feature, and end-of-file processing.

If initialization is called for, the logical unit is defined.

If a new feature start is requested, counters are initialized and processing is continued as described in the next paragraph.

If a continuation of the feature previously started is requested, processing starts here. The total number of points remaining in the feature is calculated and the buffer CALBUF is filled alternately with X and Y values until this total is reached. Each time thirty-eight (38) X,Y values have been stored in the buffer it is written on the tape. If the last group of X and Y values is less than 38, these pairs are put into the buffer separately, then output to the tape. When all pairs are processed control is returned to the calling program.

If the end-of-file is indicated, the tape has an end-of-file written on it and it is then rewound. A return is then made to the calling program.

### (4) Program Variables

CALBUF - Buffer holding the information output to the CALMA tape

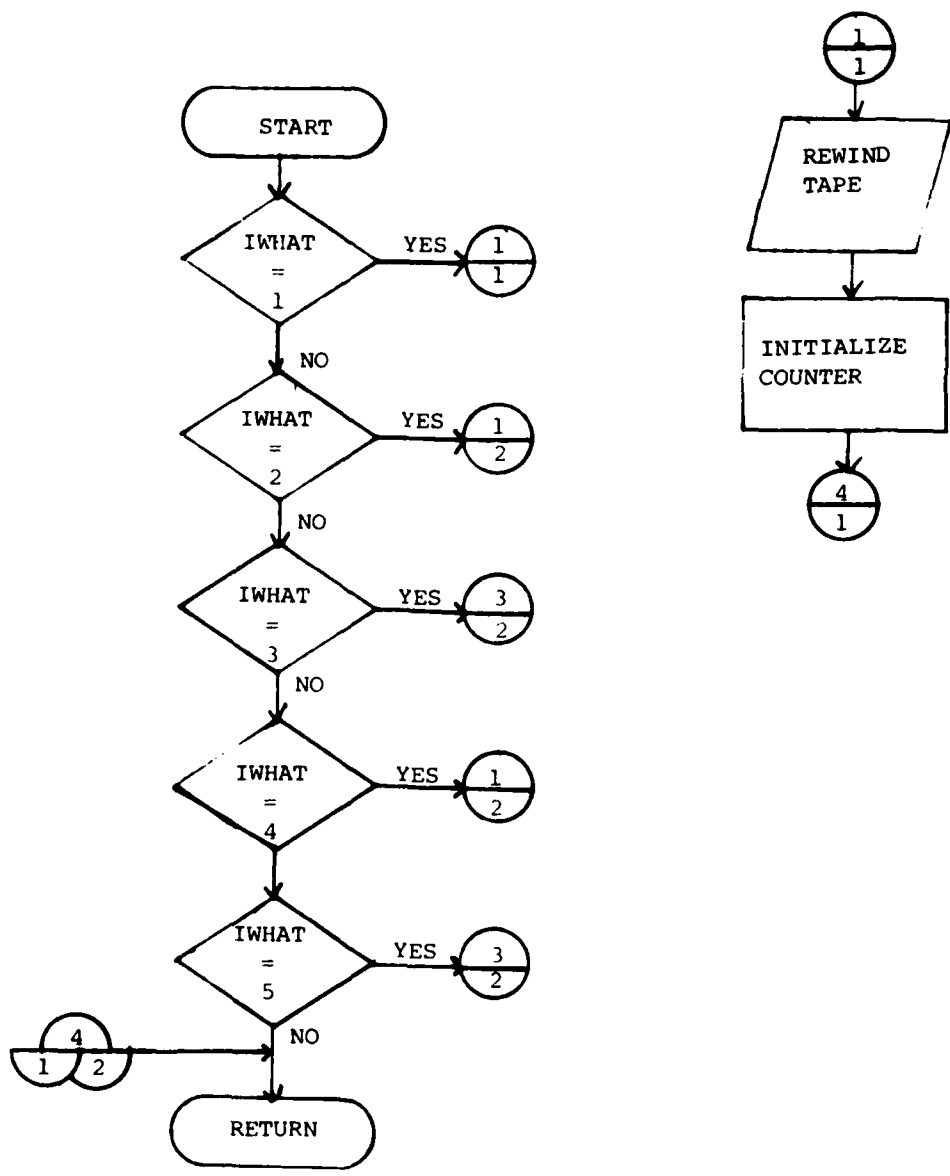
REM - number of remaining X and Y pairs to process

L - counter used in stepping through IXLIST and IYLIST

TOT - total number of pairs in feature

### (5) Relationship to Other Routines

CALMA is called by OUTPUT then the value of IFORM is two (2). CALMA uses NTRAN to write to the output tape.



FLOWCHART FOR SUBROUTINE CALMA  
 (Page 1 of 2)

FIGURE C-2



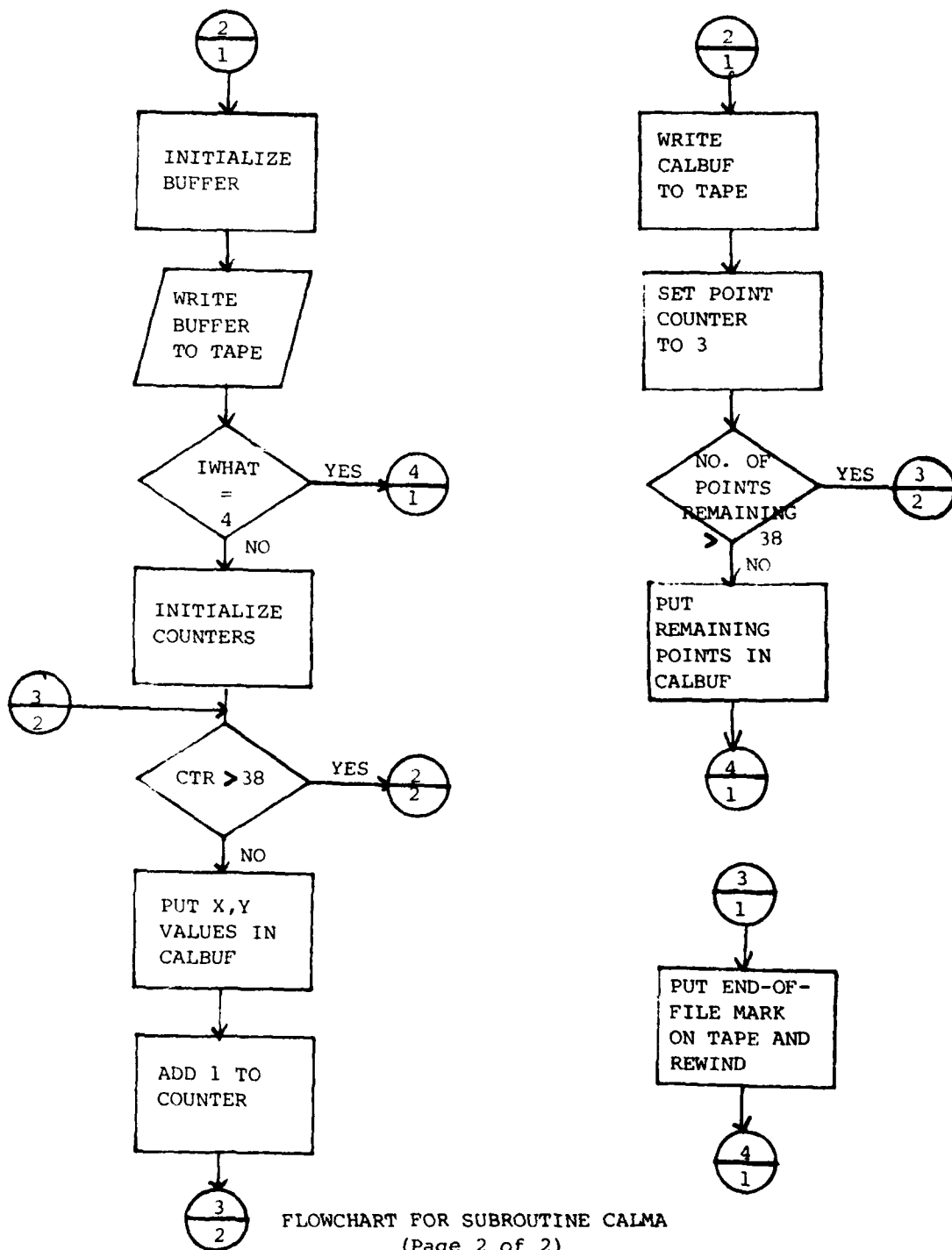


FIGURE C-2

CHKWD

(1) Function

This routine keeps track of the current word and block for routine AGDSF.

(2) Input/Output

No external input is performed by this routine. All information comes in the common block AGDS. The routine will write a record to the temporary disk file when the end of the 'block' is reached.

(3) Process Description

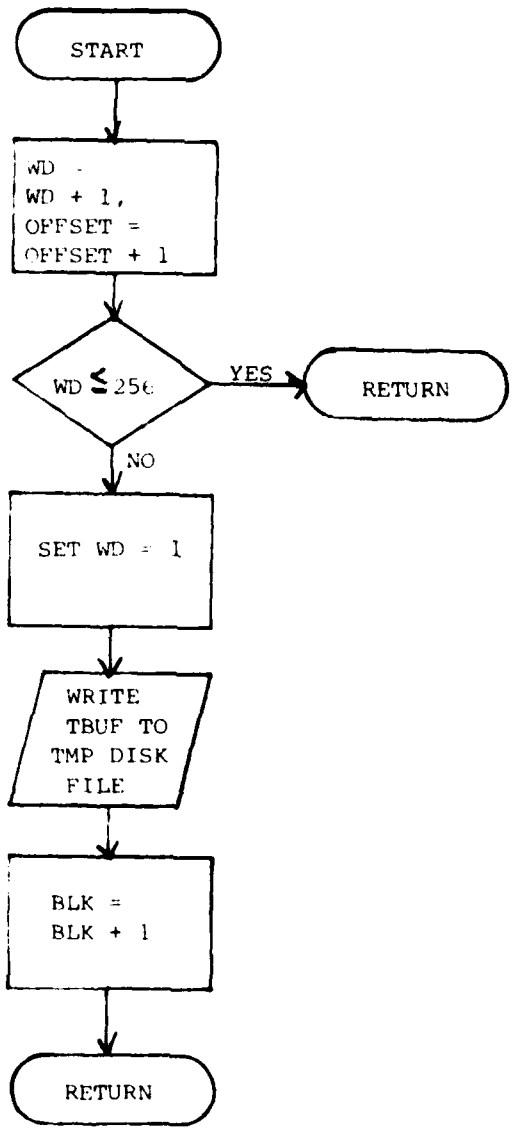
The routine begins by incrementing the word counter, WD, and the offset counter, OFFSET, by 1. If the value of WD is greater than 256 the block is written to the temporary disk file and the block counter, BLK, increased by 1.

(4) Program Variables

Common block AGDS - see routine AGDSF for a description.

(5) Relationship to Other Routines

CHKWD is called by AGDSF. It calls no other routines.



FLOWCHART FOR SUBROUTINE CHKWD  
(Page 1 of 1)

FIGURE C-3

## SUBROUTINE LIS

### (1) Function

Subroutine LIS is called by OUTPUT, a subroutine of PHASE IV. It is used to create a skeleton input tape for the Lineal Input System or LIS.

### (2) Input/Output

Subroutine LIS has no external input. It outputs the linealized features and necessary header information in Lineal Input System format to tape.

### (3) Process Description

Subroutine LIS determines first what kind of call is being made to it from subprogram OUTPUT of PHASE IV. This is done by examining the value of IWHAT from OUTPUT. The four possible types of calls are start file, new feature, continue previous feature, and end file.

The Lineal Input System uses a tape consisting of 4 file control blocks, a feature header block for each feature, feature data blocks, and a file summary block. If the value of IWHAT corresponds to the START FILE command, subroutine LIS initializes the four file control blocks. This is done by writing all available file control information into arrays RT0, RT201, RT202, and RT203 and outputting these arrays to tape. Upon completion, control returns to subprogram OUTPUT.

If the value of IWHAT corresponds to the command START NEW FEATURE, LIS determines whether this is the first feature from PHASE IV OUTPUT. If not, LIS writes the header and data block from the previous feature to a temporary disk file. LIS then takes the feature data from OUTPUT in IXLIST and IYLIST and converts the point values to Lineal Input System vector format. Feature header information is stored in array RT30 and feature data is stored in RT31. When IXLIST and IYLIST have been exhausted, the last points of IXLIST and IYLIST are saved for possible feature continuatuion.

If the value of IWHAT corresponds to the command CONTINUE PREVIOUS FEATURE, subroutine LIS retrieves the stored last points and continues the vector formatting of the previous feature using the new values in IXLIST and IYLIST supplied by subprogram OUTPUT.

If the value of IWHAT corresponds to the command END FILE, subroutine LIS outputs the feature header (RT30) and feature data (RT31) blocks for the previous and last feature. LIS then sets up RT90, an array that stores summary file information in Lineal Input System format. When RT90 has been completed, it is determined whether updating of RT0 is necessary. If so, RT0 is corrected and rewritten on tape, then RT90, the file summary record, is written on tape.

(4) Variables

COMMON areas:

SCAFAC - SCALEF  
 RESOLU - RES  
 XLIST - IXLIST (1000)  
 YLIST - IYLIST (1000)  
 HMANY - IMANY  
 WHAT - IWHAT  
 LISOUT - XFIRST, YFIRST, XLAST, YLAST, N31, FTRNUM, TNUMPT,  
 FLKNUM, STRVEC, RES1, RESOCD, RT0, RT30, RT31,

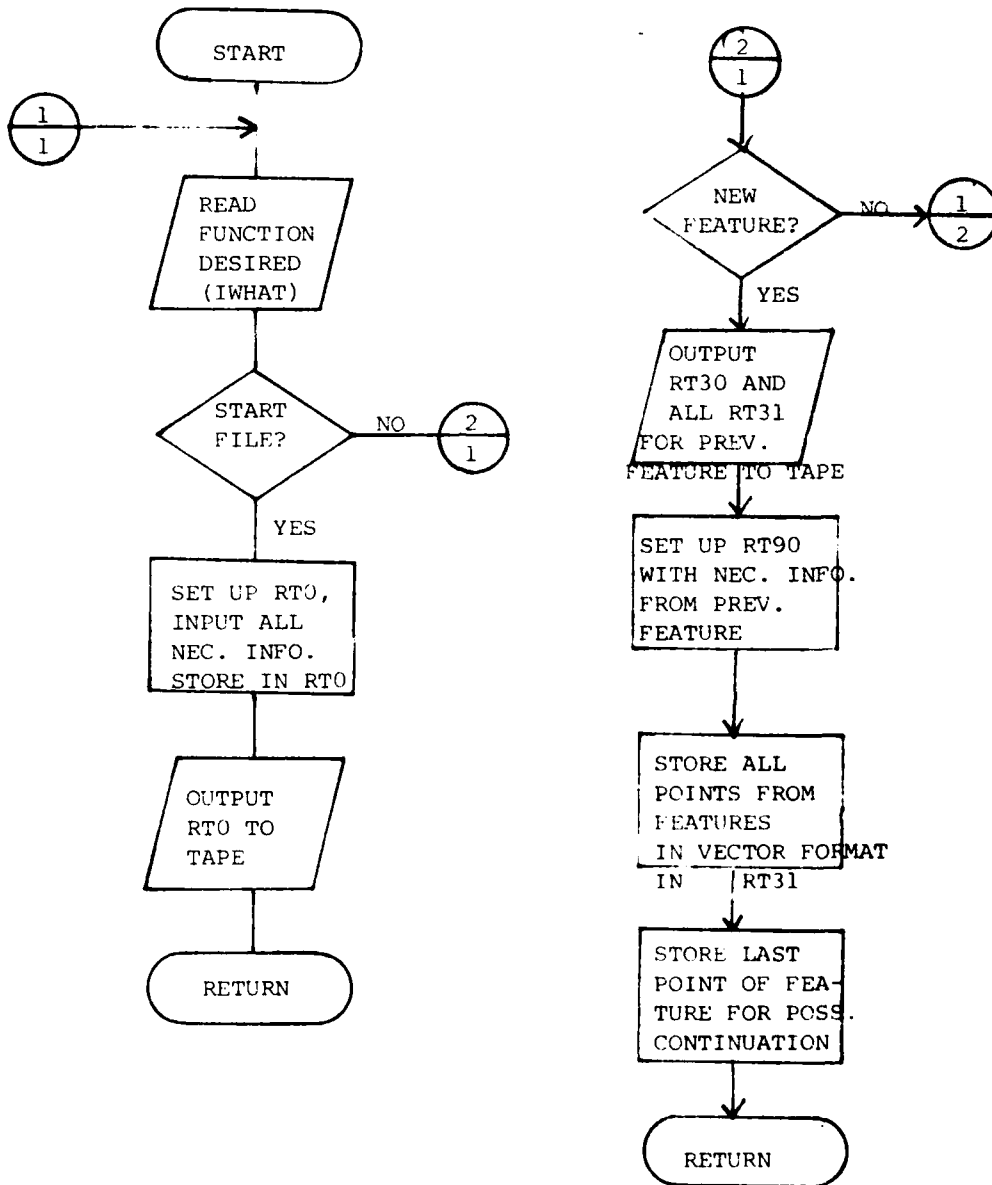
Variables: RT90, WORD, VCOUNT

XFIRST, YFIRST - first point of a feature  
 XLAST, YLAST - last points of a feature  
 HOLDX, HOLDY - last value of arrays IXLIST and IYLIST held  
 for possible continuation of a feature  
 N31 - counter for number of RT31 (feature data)  
 blocks  
 FTRNUM - binary feature number (RT0, RT30, RT90)  
 TNUMPT - total number of features  
 BLKNUM - relative block number in file  
 RESOCD - LIS format resolution code  
 TL - trace length of each feature  
 TRACEL - total trace length of file  
 STRVEC - starting vector position of each feature  
 NUMPNT - number of points in file  
 NUMVEC - number of vectors in feature

FTR                   - array for storing feature length  
AVLENG                - average feature length for entire file

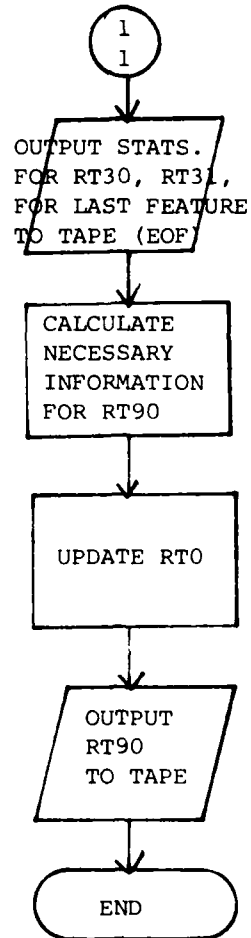
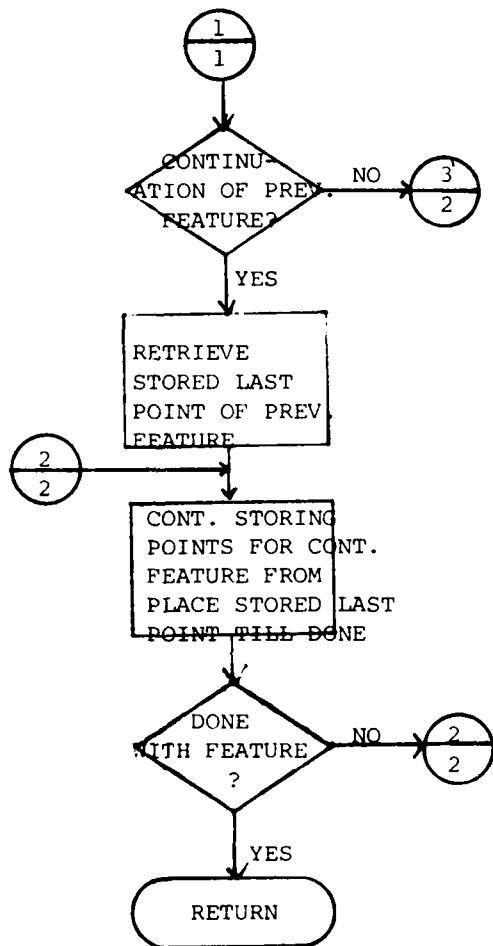
(5) Relationship to Other Routines

LIS is called by subprogram OUTPUT of PHASE IV. LIS calls no other routine.



FLOWCHART FOR SUBROUTINE LIS  
(Page 1 of 2)

FIGURE C-4



FLOWCHART FOR SUBROUTINE LIS  
(Page 2 of 2)

FIGURE C-4



## OUTPUT

### (1) Function

OUTPUT calls the routine to convert the integer X and Y coordinates of each lineal feature into the appropriate output format.

### (2) Input/Output

Input to OUTPUT is a list of coordinates. There is no external output from this routine.

### (3) Process Description

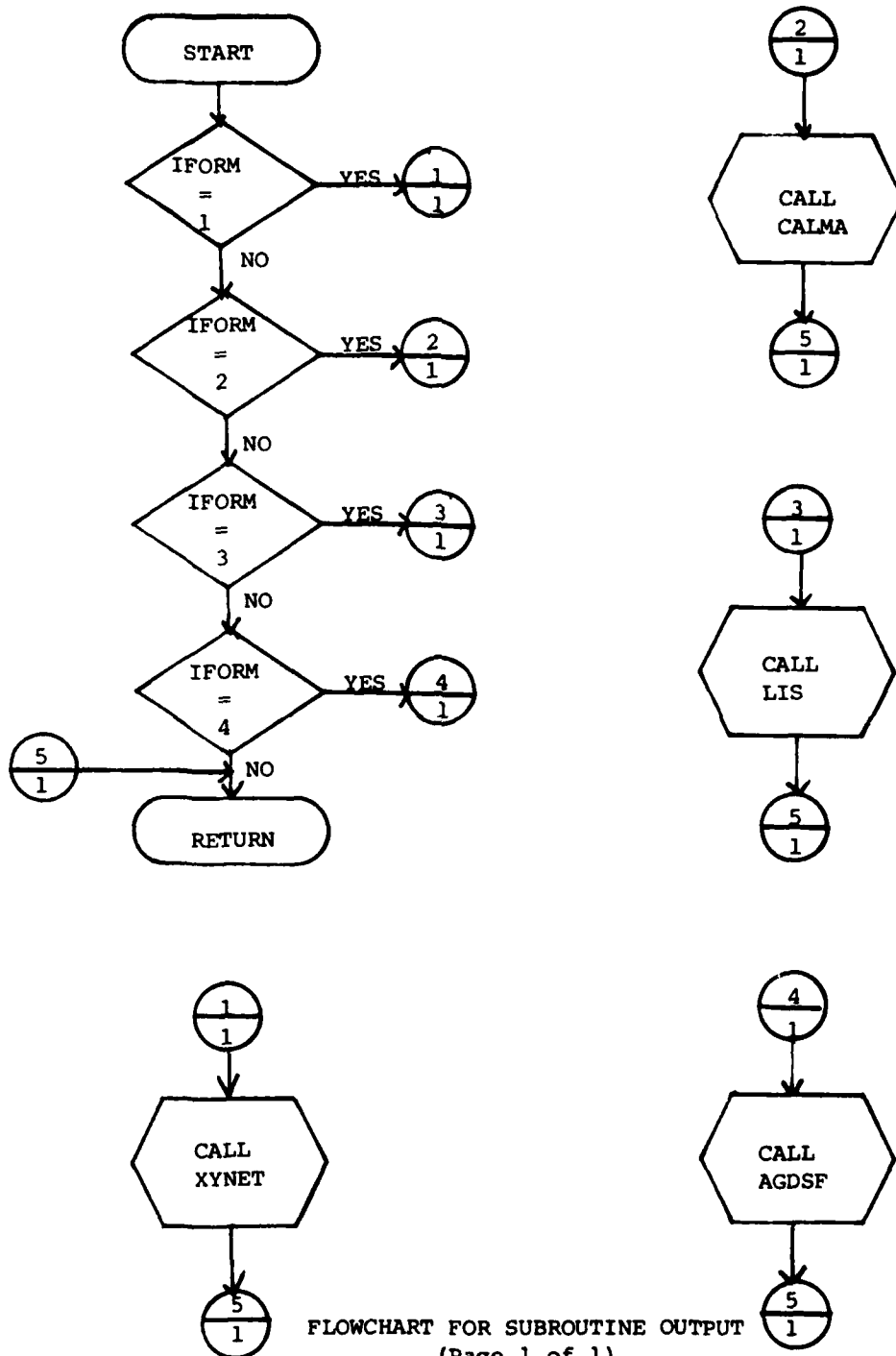
OUTPUT calls the appropriate output module based on the value of IFORM.

### (4) Program Variables

FORMAT	-	IFORM
SCAFAC	-	SCALEF
RESOLU	-	RES
BOUREC	-	MINY
	-	MAXX
	-	MINY
	-	MAXY
UNIT	-	IUNIT
XLIST	-	IXLIST (1000)
YLIST	-	IYLIST (1000)
HMAN Y	-	IMANY
WHAT	-	IWHAT
HEAD	-	HEADER (20)

### (5) Relationship to Other Routines

OUTPUT calls LIS, AGDSF, CALMA or XYPNET depending on the value of IFORM. It is called by PHAS4 and by POINTS.



FLOWCHART FOR SUBROUTINE OUTPUT  
(Page 1 of 1)

FIGURE C-5

PACK

(1) Function

This subroutine will pack data contained in array TBUF into AGBUF simulating a 16-bit word.

(2) Input/Output

PACK does no external input/output.

(3) Process Description

PACK takes the top 16 bits (20-36) of each word in TBUF and packs them into array AGBUF. If the entire 16 bits cannot be fit into the current word in AGBUF, the bits are split over word boundaries in the array.

(4) Program Variables

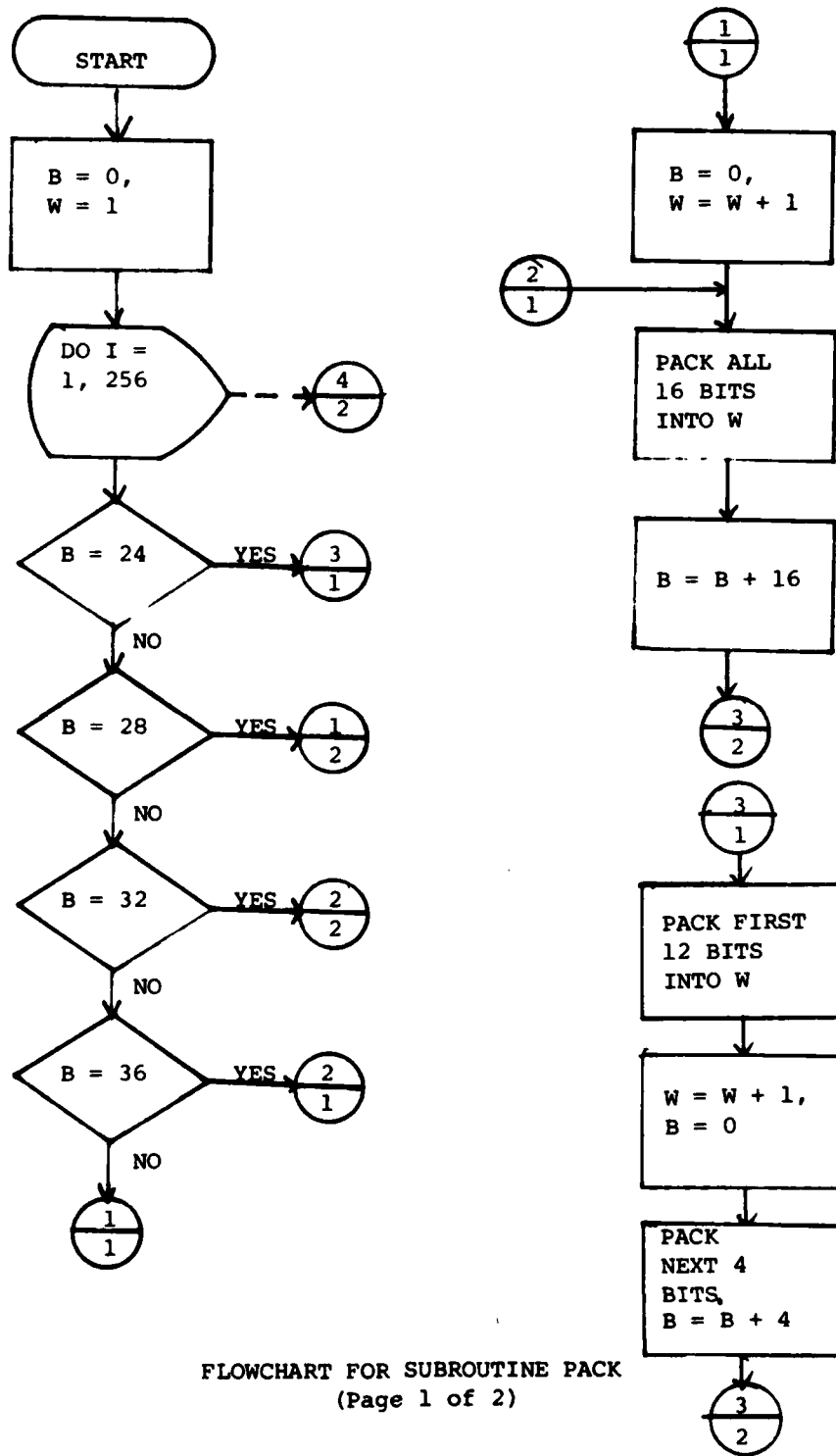
COMMON area AGDS - See routine AGDSF for description.

W - current word in AGBUF

B - current bit in the word

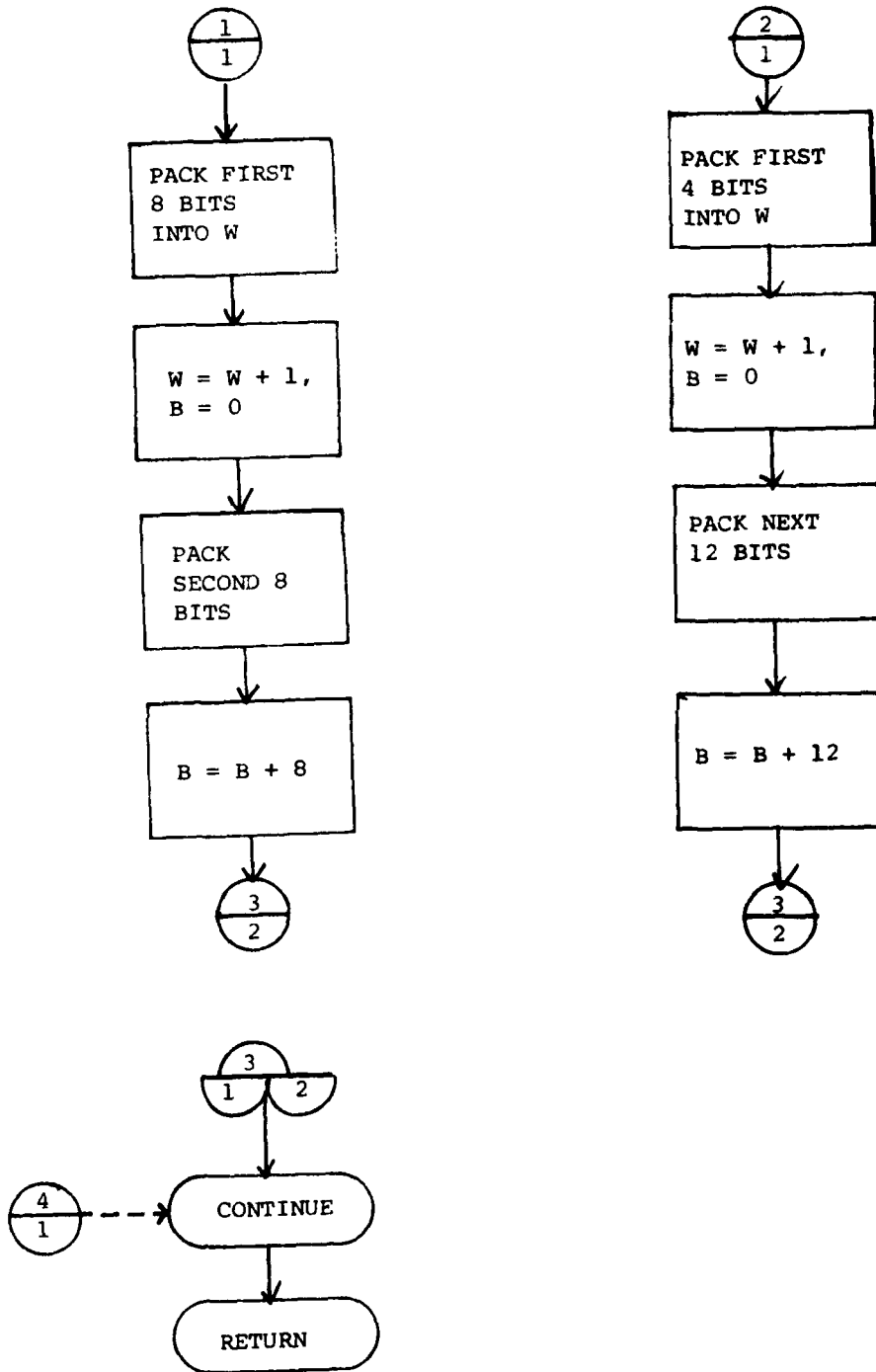
(5) Relationship to Other Routines

PACK is called by AGDSF. It calls no other routines.



FLOWCHART FOR SUBROUTINE PACK  
(Page 1 of 2)

FIGURE C-6



FLOWCHART FOR SUBROUTINE PACK  
(Page 2 of 2)

FIGURE C-6

