

## N O T I C E

THIS DOCUMENT HAS BEEN REPRODUCED FROM  
MICROFICHE. ALTHOUGH IT IS RECOGNIZED THAT  
CERTAIN PORTIONS ARE ILLEGIBLE, IT IS BEING RELEASED  
IN THE INTEREST OF MAKING AVAILABLE AS MUCH  
INFORMATION AS POSSIBLE

"Made available under NASA sponsorship  
in the interest of early and wide dis-  
semination of Earth Resources Survey  
Program information and without liability  
for any use made thereof."

8.0 - 10217

NASA CR-

160621

JSC-13055

"AS-BUILT" DESIGN SPECIFICATION  
FOR  
BOUNDARY DETECTION AND REGISTRATION  
PROGRAM (BDARP1)

Job Order 71-695

(TIRF 76-0046)

(E80-10217) AS-BUILT DESIGN SPECIFICATION  
FOR BOUNDARY DETECTION AND REGISTRATION  
PROGRAM (BDARP1) (Lockheed Electronics Co.)  
116 p HC A06/MF A01 CSCL 05B

N80-29794

Unclas  
G3/43 00217

Prepared By  
Lockheed Electronics Company, Inc.  
Systems and Services Division  
Houston, Texas

Contract NAS 9-15200

For  
EARTH OBSERVATIONS DIVISION  
SCIENCE AND APPLICATIONS DIRECTORATE



*National Aeronautics and Space Administration*  
**LYNDON B. JOHNSON SPACE CENTER**

*Houston, Texas*

August 1977

LEC-11074

JSC-13055

"AS-BUILT" DESIGN SPECIFICATION  
FOR  
BOUNDARY DETECTION AND REGISTRATION  
PROGRAM (BDARP1)

Job Order 71-695

(TIRF 76-0046)

Prepared By

D. P. McKay

F. Collen

APPROVED BY

  
Philip L. Krumm, Supervisor  
Applications Software Section

Prepared By

Lockheed Electronics Company, Inc.

For

Earth Observations Division  
Science and Applications Directorate

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION  
LYNDON B. JOHNSON SPACE CENTER  
HOUSTON, TEXAS

August 1977

LEC-11074

## CONTENTS

Section	Page
1. SCOPE . . . . .	1-1
1.1 <u>GENERAL</u> . . . . .	1-1
2. APPLICABLE DOCUMENTS . . . . .	2-1
3. SYSTEM DESCRIPTION . . . . .	3-1
3.1 <u>HARDWARE DESCRIPTION</u> . . . . .	3-2
3.2 <u>SOFTWARE DESCRIPTION</u> . . . . .	3-2
3.2.1 SOFTWARE COMPONENT NO. 1 (DRVF) . . . . .	3-4
3.2.1.1 <u>Linkage</u> . . . . .	3-4
3.2.1.2 <u>Interface</u> . . . . .	3-4
3.2.1.3 <u>Input</u> . . . . .	3-4
3.2.1.4 <u>Output</u> . . . . .	3-4
3.2.1.5 <u>Storage Requirements</u> . . . . .	3-5
3.2.1.6 <u>Description</u> . . . . .	3-5
3.2.1.7 <u>Flowchart</u> . . . . .	3-6
3.2.1.8 <u>Listing</u> . . . . .	3-7
3.2.2 SOFTWARE COMPONENT NO. 2 (REAH) . . . . .	3-8
3.2.2.1 <u>Linkage</u> . . . . .	3-8
3.2.2.2 <u>Interface</u> . . . . .	3-8
3.2.2.3 <u>Input</u> . . . . .	3-8
3.2.2.4 <u>Output</u> . . . . .	3-8
3.2.2.5 <u>Storage Requirements</u> . . . . .	3-8
3.2.2.6 <u>Description</u> . . . . .	3-9
3.2.2.7 <u>Flowchart</u> . . . . .	3-10

Section	Page
3.2.2.8 <u>Listing</u> . . . . .	3-11
3.2.3 SOFTWARE COMPONENT NO. 3 (REAHD) . . . . .	3-13
3.2.3.1 <u>Linkage</u> . . . . .	3-13
3.2.3.2 <u>Interface</u> . . . . .	3-13
3.2.3.3 <u>Input</u> . . . . .	3-13
3.2.3.4 <u>Output</u> . . . . .	3-13
3.2.3.5 <u>Storage Requirements</u> . . . . .	3-13
3.2.3.6 <u>Description</u> . . . . .	3-13
3.2.3.7 <u>Flowchart</u> . . . . .	3-15
3.2.3.8 <u>Listing</u> . . . . .	3-16
3.2.4 SOFTWARE COMPONENT NO. 4 (INITN) . . . . .	3-19
3.2.4.1 <u>Linkage</u> . . . . .	3-19
3.2.4.2 <u>Interface</u> . . . . .	3-19
3.2.4.3 <u>Input</u> . . . . .	3-19
3.2.4.4 <u>Output</u> . . . . .	3-19
3.2.4.5 <u>Storage Requirements</u> . . . . .	3-19
3.2.4.6 <u>Description</u> . . . . .	3-19
3.2.4.7 <u>Flowchart</u> . . . . .	3-20
3.2.4.8 <u>Listing</u> . . . . .	3-21
3.2.5 SOFTWARE COMPONENT NO. 5 (CON79) . . . . .	3-23
3.2.5.1 <u>Linkage</u> . . . . .	3-23
3.2.5.2 <u>Interface</u> . . . . .	3-23
3.2.5.3 <u>Input</u> . . . . .	3-23
3.2.5.4 <u>Output</u> . . . . .	3-23
3.2.5.5 <u>Storage Requirements</u> . . . . .	3-23

Section	Page
3.2.5.6 <u>Description</u> . . . . .	3-23
3.2.5.7 <u>Flowchart</u> . . . . .	3-24
3.2.5.8 <u>Listing</u> . . . . .	3-25
3.2.6 SOFTWARE COMPONENT NO. 6 (RDLIN) . . . . .	3-28
3.2.6.1 <u>Linkage</u> . . . . .	3-28
3.2.6.2 <u>Interface</u> . . . . .	3-28
3.2.6.3 <u>Input</u> . . . . .	3-28
3.2.6.4 <u>Output</u> . . . . .	3-28
3.2.6.5 <u>Storage Requirements</u> . . . . .	3-28
3.2.6.6 <u>Description</u> . . . . .	3-28
3.2.6.7 <u>Flowchart</u> . . . . .	3-29
3.2.6.8 <u>Listing</u> . . . . .	3-30
3.2.7 SOFTWARE COMPONENT NO. 7 (ISET) . . . . .	3-34
3.2.7.1 <u>Linkage</u> . . . . .	3-34
3.2.7.2 <u>Interface</u> . . . . .	3-34
3.2.7.3 <u>Input</u> . . . . .	3-34
3.2.7.4 <u>Output</u> . . . . .	3-34
3.2.7.5 <u>Storage Requirements</u> . . . . .	3-34
3.2.7.6 <u>Description</u> . . . . .	3-34
3.2.7.7 <u>Flowchart</u> . . . . .	3-35
3.2.7.8 <u>Listing</u> . . . . .	3-36
3.2.8 SOFTWARE COMPONENT NO. 8 (FRAME) . . . . .	3-37
3.2.8.1 <u>Linkage</u> . . . . .	3-37
3.2.8.2 <u>Interface</u> . . . . .	3-37
3.2.8.3 <u>Input</u> . . . . .	3-37

Section	Page
3.2.8.4 <u>Output</u> . . . . .	3-37
3.2.8.5 <u>Storage Requirements</u> . . . . .	3-37
3.2.8.6 <u>Description</u> . . . . .	3-37
3.2.8.7 <u>Flowchart</u> . . . . .	3-38
3.2.8.8 <u>Listing</u> . . . . .	3-39
3.2.9 SOFTWARE COMPONENT NO. 9 (LINIT) . . . . .	3-40
3.2.9.1 <u>Linkage</u> . . . . .	3-40
3.2.9.2 <u>Interface</u> . . . . .	3-40
3.2.9.3 <u>Input</u> . . . . .	3-40
3.2.9.4 <u>Output</u> . . . . .	3-40
3.2.9.5 <u>Storage Requirements</u> . . . . .	3-40
3.2.9.6 <u>Description</u> . . . . .	3-40
3.2.9.7 <u>Flowchart</u> . . . . .	3-42
3.2.9.8 <u>Listing</u> . . . . .	3-43
3.2.10 SOFTWARE COMPONENT NO. 10 (BDT3) . . . . .	3-44
3.2.10.1 <u>Linkage</u> . . . . .	3-44
3.2.10.2 <u>Interface</u> . . . . .	3-44
3.2.10.3 <u>Input</u> . . . . .	3-44
3.2.10.4 <u>Output</u> . . . . .	3-44
3.2.10.5 <u>Storage Requirements</u> . . . . .	3-44
3.2.10.6 <u>Description</u> . . . . .	3-44
3.2.10.7 <u>Flowchart</u> . . . . .	3-46
3.2.10.8 <u>Listing</u> . . . . .	3-48
3.2.11 SOFTWARE COMPONENT NO. 11 (READAT) . . . . .	3-55
3.2.11.1 <u>Linkage</u> . . . . .	3-55
3.2.11.2 <u>Interface</u> . . . . .	3-55

Section	Page
3.2.11.3 <u>Input</u> . . . . .	3-55
3.2.11.4 <u>Output</u> . . . . .	3-55
3.2.11.5 <u>Storage Requirements</u> . . . . .	3-55
3.2.11.6 <u>Description</u> . . . . .	3-55
3.2.11.7 <u>Flowchart</u> . . . . .	3-57
3.2.11.8 <u>Listing</u> . . . . .	3-58
3.2.12 SOFTWARE COMPONENT NO. 12 (IGET) . . . . .	3-59
3.2.12.1 <u>Linkage</u> . . . . .	3-59
3.2.12.2 <u>Interface</u> . . . . .	3-59
3.2.12.3 <u>Input</u> . . . . .	3-59
3.2.12.4 <u>Output</u> . . . . .	3-59
3.2.12.5 <u>Storage Requirements</u> . . . . .	3-59
3.2.12.6 <u>Description</u> . . . . .	3-59
3.2.12.7 <u>Flowchart</u> . . . . .	3-60
3.2.12.8 <u>Listing</u> . . . . .	3-61
3.2.13 SOFTWARE COMPONENT NO. 13 (FILL) . . . . .	3-62
3.2.13.1 <u>Linkage</u> . . . . .	3-62
3.2.13.2 <u>Interface</u> . . . . .	3-62
3.2.13.3 <u>Input</u> . . . . .	3-62
3.2.13.4 <u>Output</u> . . . . .	3-62
3.2.13.5 <u>Storage Requirements</u> . . . . .	3-62
3.2.13.6 <u>Description</u> . . . . .	3-62
3.2.13.7 <u>Flowchart</u> . . . . .	3-63
3.2.13.8 <u>Listing</u> . . . . .	3-65
3.2.14 SOFTWARE COMPONENT NO. 14 (FINDAR) . . . . .	3-67



Section	Page
3.2.14.1 <u>Linkage</u> . . . . .	3-67
3.2.14.2 <u>Interface</u> . . . . .	3-67
3.2.14.3 <u>Input</u> . . . . .	3-67
3.2.14.4 <u>Output</u> . . . . .	3-67
3.2.14.5 <u>Storage Requirements</u> . . . . .	3-67
3.2.14.6 <u>Description</u> . . . . .	3-67
3.2.14.7 <u>Flowchart</u> . . . . .	3-68
3.2.14.8 <u>Listing</u> . . . . .	3-69
3.2.15 SOFTWARE COMPONENT NO. 15 (CONECT) . . .	3-72
3.2.15.1 <u>Linkage</u> . . . . .	3-72
3.2.15.2 <u>Interface</u> . . . . .	3-72
3.2.15.3 <u>Input</u> . . . . .	3-72
3.2.15.4 <u>Output</u> . . . . .	3-72
3.2.15.5 <u>Storage Requirements</u> . . . . .	3-72
3.2.15.6 <u>Description</u> . . . . .	3-72
3.2.15.7 <u>Flowchart</u> . . . . .	3-73
3.2.15.8 <u>Listing</u> . . . . .	3-74
3.2.16 SOFTWARE COMPONENT NO. 16 (CONALL) . . .	3-76
3.2.16.1 <u>Linkage</u> . . . . .	3-76
3.2.16.2 <u>Interface</u> . . . . .	3-76
3.2.16.3 <u>Input</u> . . . . .	3-76
3.2.16.4 <u>Output</u> . . . . .	3-76
3.2.16.5 <u>Storage Requirements</u> . . . . .	3-76
3.2.16.6 <u>Description</u> . . . . .	3-76
3.2.16.7 <u>Flowchart</u> . . . . .	3-77

Section	Page
3.2.16.8 <u>Listing</u> . . . . .	3-79
3.2.17 SOFTWARE COMPONENT NO. 17 (JOIN) . . . . .	3-81
3.2.17.1 <u>Linkage</u> . . . . .	3-81
3.2.17.2 <u>Interface</u> . . . . .	3-81
3.2.17.3 <u>Input</u> . . . . .	3-81
3.2.17.4 <u>Output</u> . . . . .	3-81
3.2.17.5 <u>Storage Requirements</u> . . . . .	3-81
3.2.17.6 <u>Description</u> . . . . .	3-81
3.2.17.7 <u>Flowchart</u> . . . . .	3-82
3.2.17.8 <u>Listing</u> . . . . .	3-83
3.2.18 SOFTWARE COMPONENT NO. 18 (CLSTST) . . . . .	3-85
3.2.18.1 <u>Linkage</u> . . . . .	3-85
3.2.18.2 <u>Interface</u> . . . . .	3-85
3.2.18.3 <u>Input</u> . . . . .	3-85
3.2.18.4 <u>Output</u> . . . . .	3-85
3.2.18.5 <u>Storage Requirements</u> . . . . .	3-85
3.2.18.6 <u>Description</u> . . . . .	3-85
3.2.18.7 <u>Flowchart</u> . . . . .	3-86
3.2.18.8 <u>Listing</u> . . . . .	3-87
3.2.19 SOFTWARE COMPONENT NO. 19 (AREAL) . . . . .	3-88
3.2.19.1 <u>Linkage</u> . . . . .	3-88
3.2.19.2 <u>Interface</u> . . . . .	3-88
3.2.19.3 <u>Input</u> . . . . .	3-88
3.2.19.4 <u>Output</u> . . . . .	3-88
3.2.19.5 <u>Storage Requirements</u> . . . . .	3-88

Section	Page
3.2.19.6 <u>Description</u> . . . . .	3-88
3.2.19.7 <u>Flowchart</u> . . . . .	3-89
3.2.19.8 <u>Listing</u> . . . . .	3-90
3.2.20 SOFTWARE COMPONENT NO. 20 (ENDTST) . . .	3-91
3.2.20.1 <u>Linkage</u> . . . . .	3-91
3.2.20.2 <u>Interface</u> . . . . .	3-91
3.2.20.3 <u>Input</u> . . . . .	3-91
3.2.20.4 <u>Output</u> . . . . .	3-91
3.2.20.5 <u>Storage Requirements</u> . . . . .	3-91
3.2.20.6 <u>Description</u> . . . . .	3-91
3.2.20.7 <u>Flowchart</u> . . . . .	3-92
3.2.20.8 <u>Listing</u> . . . . .	3-93
4. OPERATION . . . . .	4-1
4.1 <u>USER DOCUMENTATION</u> . . . . .	4-1
4.2 <u>OPERATOR DOCUMENTATION</u> . . . . .	4-1
 Appendix	
A. BDARPI COMMON TABLE . . . . .	A-2

**FIGURES**

<b>Figure</b>	<b>Page</b>
1. Functional Diagram of Phase I Implementation of the Software System . . . . .	4-2
2. Bendix System 100 Core Utilization Map . . . . .	4-3
3. User's Procedure . . . . .	4-4
4. The Functional Block Diagram of BDARPl . . . . .	4-6

**APPENDIX**

1. BDARPl Common Table . . . . .	A-2
----------------------------------	-----

## 1. SCOPE

This document describes the detailed design characteristics of the Boundary Detection and Registration Program (BDARPl), as built for the Bendix 100 Interactive Drafting System. The BDARPl is an unsophisticated version of the final software system, yet it provides the user with the basic capabilities of obtaining classified data boundary plots, editing, and registration of the final boundary plot to a user-selected base.

## 2. APPLICABLE DOCUMENTS

The following documents form a part of this specification to the extent specified herein:

- Technical Memorandum Software Specifications for Automated Thematic Plotting of Classified Digital Data, LEC-8289
- Technical Memorandum Project Development Plan for the Bendix Interactive Drafting System Modification, LEC-8968
- Design Specification for Automated Thematic Plotting of Classified Digital Data, LEC-9506
- Technical Memorandum Acceptance Test Plan for Boundary Detection and Registration Program (BDARPl), LEC-10672
- TIRF 76-0046

### 3. SYSTEM DESCRIPTION

The Boundary Detection and Registration Program (BDARPl) was designed and implemented as an addition to the basic Bendix 100 Drafting Program. The BDARPl consists of three overlays:

USER08 - the classified tape initialization module

T2 - the tape read and data storage routine

T3 - the boundary detection and registration algorithm

To begin processing, USER08 accepts the user's options and reads the header record from a 7-track, 800 BPI, even parity universal formatted classified tape directly or indirectly obtained from the GE Interactive Multispectral Image Analyst System (Image 100), the Earth Resources Interactive Processing System (ERIPS) or the UNIVAC 1100 Software.

USER08 then calls overlay T2, which reads the required number of data records from the magnetic tape. The data are processed, packed and written on a temporary disk file, TDATA. Corner reference ticks are placed on the drawing file.

Overlay T2 calls the third and last overlay - T3. Overlay T3 reads the data stored in TDATA, one line at a time, and performs the boundary detection and registration algorithm. The resultant boundary information is written into a standard format drawing file, and control is then returned to the basic Bendix 100 Drafting Program. Editing and write tape functions are now available to prepare the boundary data for plotting.

BDARPl is designed to process one class at a time. For the case of multiple classes, BDARPl has to run as many times as the number of classes. Each execution of BDARPl under the Drafting Program is initialized by selecting USER OPTION:8 on the menu.

When processing is completed, BDARPl informs the user by sounding the tone on the display device (Tektronix) and illuminating the red indicator light on the digitizer cursor.

Note that the editing and tape write functions are currently available under Bendix System 100 and can be used as long as the drawing file format used to store the boundary strings by the boundary detection routine is identical to the one employed by the Bendix System 100 software. Since no additional software is required for the editing and write tape routines, these two are not included in the software description. However, as a result, it imposes a restriction on the file format to be used to store the boundary strings.

### 3.1 HARDWARE DESCRIPTION

Bendix System 100 configuration.

### 3.2 SOFTWARE DESCRIPTION

In this section each of the three overlays which form an integral part of BDARPl is further broken down into subroutines. Brief functional descriptions of each subroutine as well as inter-subroutine relationships are discussed.

Overlay USER08 is the initialization module for BDARPl, and consists of the following routines:

DRVF - the driver routine for this overlay

INPBD - the subroutine which interacts with the operator to accomplish input of the control parameters

REAHD - subroutine which reads the header record on the classified input tape

INITN - subroutine which error checks header record data and positions the tape for reading the image data



CON79 - subroutine which converts unformatted input data to byte data

The second overlay, T2, which performs input of classified data, consists of the following routines:

RDLIN - reads the classified data tape and packs the data into a temporary disk file, TDATA

ISSET - sets the appropriate bits in 16 bit words to indicate which pixels belong to the class being examined. These words are the packed data which RDLIN packs into TDATA

CON79 - same as CON79 in overlay USER08

FRAME - subroutine which inserts corner ticks in the drawing file

LINIT - subroutine which performs 8-parameter transformation to the data and sends it to the System 100 drawing file

The third and final overlay in BDARPl is designated T3. This overlay is the boundary detection algorithm, which examines the packed data in TDATA, creates boundary strings to represent the boundaries of the specified data class, and writes these boundary strings into a drawing file formatted for the Bendix system. The routines which comprise overlay T3 are:

BDT3 - This is the main routine for T3 and the principal routine for the boundary detection algorithm.

READAT - This subroutine reads bit images of line data from the temporary disk file, TDATA.

IGET - This subroutine unpacks the bit data read into READAT for the boundary detection algorithm.

FILL - This routine redefines appropriate pixels as "classified" to facilitate connectivity as defined by the user input parameter Epsilen.

FINDAR - Subroutine which finds the appropriate boundary string to which a boundary line segment belongs.

CONECT; CONALL; JOIN - Subroutines which link appropriate boundary strings.

CLSTST - Subroutine which periodically checks the status of boundary strings for completeness, and processes the complete ones.

AREAL - Subroutine which computes the area in pixel units of each classified group.

LINIT - Subroutine which performs 8-parameter transformation to the data and sends it to the System 100 drawing file.

ENDTST - Subroutine which handles segmentation of large plot string arrays.

### 3.2.1 SOFTWARE COMPONENT NO. 1 (DRVF)

#### 3.2.1.1 Linkage

Subroutine DRVF calls user subroutines INPBD and INITN, and calls the system subroutine FRNOV.

#### 3.2.1.2 Interface

DRVF is linked with the common block ICONS (see Appendix A) which houses all the basic control parameters for BDARPl.

#### 3.2.1.3 Input

None

#### 3.2.1.4 Output

An error message is output including an error code whenever the system subroutine FRNOV fails.

3.2.1.5 Storage Requirements

Subroutine DRVF requires 184 words in core.

3.2.1.6 Description

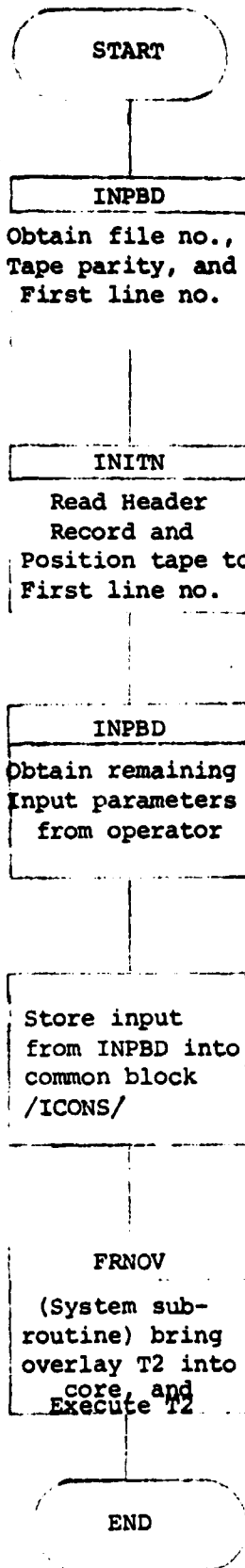
DRVF is the driver for the initial overlay USER08, and calls overlay T2 into core after USER08 has been executed.

3.2.1.7 Flowchart

3.2.1.8 Listing

(DRVF)

ORIGINAL PAGE IS  
OF POOR QUALITY



```
COMMON /ICONS/ IO(14), OPTNS(16), ISET
DIMENSION IG0(5), IOENT(20), IALPH(5)
IALPH(1)="T2"
IALPH(2)="/1"
CALL INPBD(1)
CALL INITN
CALL INPBD(2)
IALPH(3)=0
IER=0
CALL FRNOU(IALPH,IER)
PAUSE DIDNT USE FRNOU SUCCESSFULLY
WRITE(10,1001) IER
1001 FORMAT(10X,"IER =",I4)
C PAUSE OVERLAY ERROR-NO RETURN TO SYSTEM 101
END

READY
```

### 3.2.2 SOFTWARE COMPONENT NO. 2 (INPBD)

#### 3.2.2.1 Linkage

Subroutine INPBD is called by DRVF.

#### 3.2.2.2 Interface

The basic common block ICONS (see Appendix A) which houses all the necessary control parameters is created by subroutine INPBD.

#### 3.2.2.3 Input

All the basic information which BDARPl needs for execution is requested and received by INPBD via the teletype or display screen and keyboard. The operator is queried for the following:

1. Tape file no.
2. Parity (0 or 1)
3. First line no.
4. Last line no.
5. First pixel no.
6. Last pixel no.
7. Channel no.
8. Class value
9. Epsilon value
10. Kappa value
11. Eight coefficients for registration (optional)

#### 3.2.2.4 Output

The above control parameter queries are displayed on the screen.

#### 3.2.2.5 Storage Requirements

Subroutine INPBD requires 584 words in core.

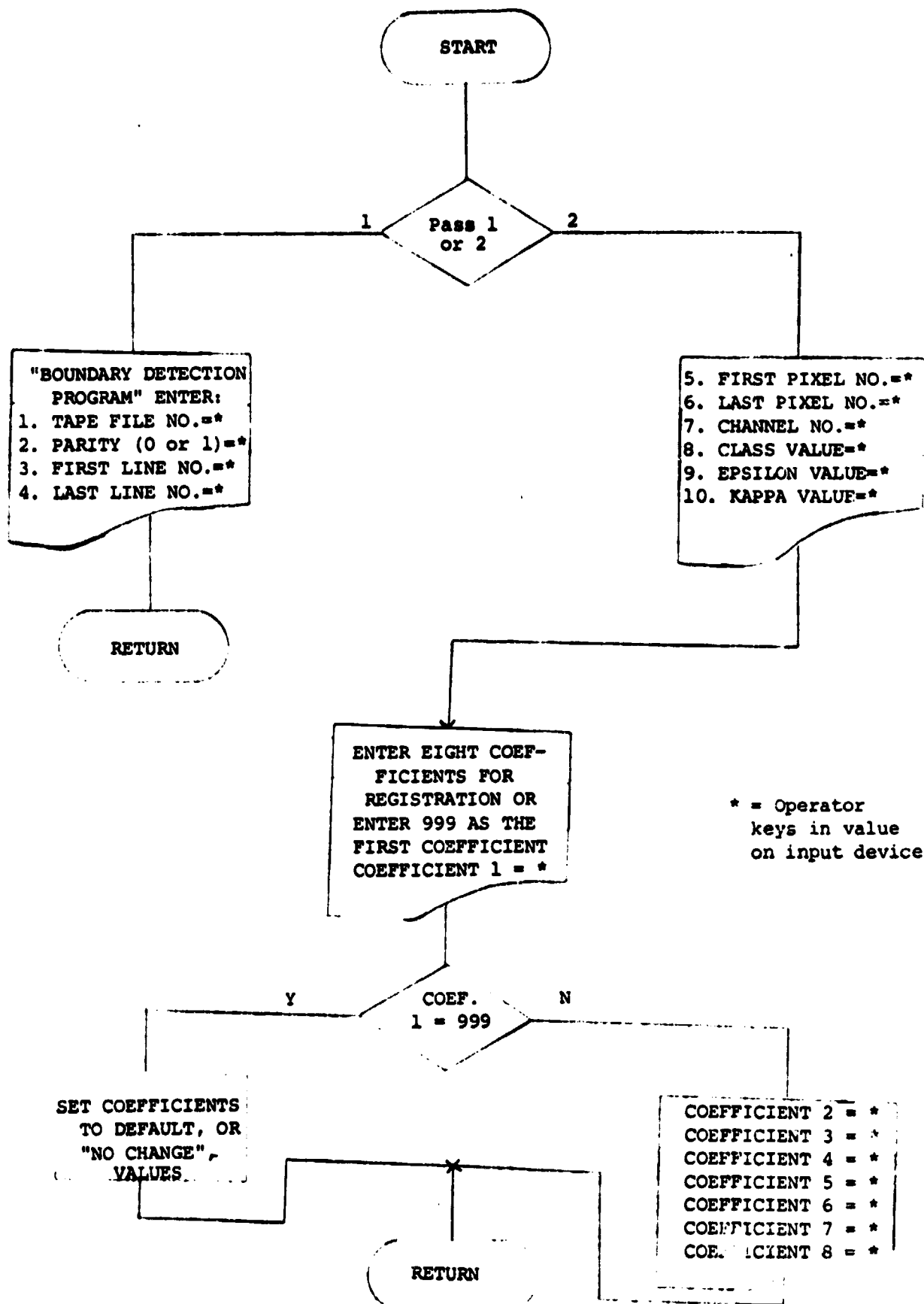
#### 3.2.2.6 Description

Subroutine INPBD interacts with the operator to bring in the basic control parameters for BDARPI execution, and defines them as components of the vector OPTNS (see Appendix A) which is part of the common block ICONS.

#### 3.2.2.7 Flowchart

#### 3.2.2.8 Listing

(INPBD)



\* = Operator keys in value on input device



LIST LINES - 33

READY

```
SUBROUTINE INPBD (LL)
COMMON /ICONS/ ID(14),OPTNS(16),ISET
IF(LL NE 1) GO TO 101

WRITE(10,1)
1  FORMAT(10X,"****      BOUNDARY DETECTION PROGRAM      ****")
   /28X,"VERSION 1",//////)
WRITE(10,2)
2  FORMAT(20X,"ENTER:",/23X,"1. TAPE FILE NO.  =")
   READ (11) OPTNS(2)
   WRITE(10,21)
21  FORMAT(23X,"2. PARITY(0 OR 1) =")
   READ (11) OPTNS(3)
   WRITE(10,22)
22  FORMAT(23X,"3. FIRST LINE NO. =")
   READ (11) OPTNS(1)
   RETURN
101 CONTINUE
   WRITE(10,3)
3  FORMAT(23X,"4. LAST LINE NO.  =")
   READ (11) OPTNS(2)
   WRITE(10,4)
4  FORMAT(23X,"5. FIRST PIXEL NO.=")
   READ (11) OPTNS(3)
   WRITE(10,5)
5  FORMAT(23X,"6. LAST PIXEL NO. =")
   READ (11) OPTNS(4)
   WRITE(10,6)
6  FORMAT(23X,"7. CHANNEL NO.   =")
   READ (11) OPTNS(5)
   WRITE(10,7)
7  FORMAT(23X,"8. CLASS VALUE   =")
   READ (11) OPTNS(6)
```

LIST LINES - 33

```
WRITE(10,8)
5 FORMAT(23X,"9. EPSILON VALUE =")
READ(11) OPTNS(7)
WRITE(10,9)
5 FORMAT(22X,"10. KAPPA VALUE =")
READ(11) OPTNS(8)
WRITE(10,10)
10 FORMAT(10X,"ENTER EIGHT COEFFICIENTS FOR REGISTRATION",
10X,"OR",10X,"ENTER 999 AS THE FIRST COEFFICIENT")
DO 100 I=1,8
WRITE(10,11) I
11 FORMAT(15X,"COEFFICIENT ",I1," =")
J=I+8
READ(11) OPTNS(J)
NOP=OPTNS(J)
IF(NOP-999) 100,120,100
100 CONTINUE
GO TO 130
120 DO 200 N=10,16
200 OPTNS(N) = 0.0
OPTNS(9) = 1.0
OPTNS(15) = 1.0
130 CALL FCNOT("!")
RETURN
END
```

READY

### 3.2.3 SOFTWARE COMPONENT NO. 3 (REAHD)

#### 3.2.3.1 Linkage

Subroutine REAHD is called in overlay USER08 by subroutine INITN, and calls subroutines CON79 and RDTAPE (a system subroutine which affects magnetic tape reading).

#### 3.2.3.2 Interface

The control information from the header record, and the information needed to read the header record, is transmitted through the common block ICONS.

#### 3.2.3.3 Input

The subroutine reads the header record on the data tape.

#### 3.2.3.4 Output

Error messages may be displayed to the operator if subroutine REAHD encounters ambiguities in the header information.

#### 3.2.3.5 Storage Requirements

Subroutine REAHD requires 650 words in core.

#### 3.2.3.6 Description

The main purpose of subroutine REAHD is to read the header record from the designated file on the data tape. A conversion using CON79 is required to obtain descriptive values. These elements are tested and, if valid, are stored in the vector ID in the common block ICONS. If adequate information to process the data is not available, a message to that effect alerts the operator and the run is terminated. Under certain conditions, however, when only one or two parameters are in error, the subroutine will supply "standard" values for the one or two in error, and execution

of BDARPI will be attempted. An appropriate warning will be communicated to the operator under this condition.

3.2.3.7 Flowchart

3.2.3.8 Listing

(REAH)

START

SET UP BYTE  
REFERENCE TABLES  
IN ITEMP  
AND ILIST

RDTAPE  
(SYSTEM SUB-  
ROUTINE) READ  
ONE RECORD OF  
DATA INTO IA

CON79  
CONVERT 7-  
TRACK INPUT  
INTO 9-TRACK  
IMAGE BYTES

LOOKS  
LIKE A HEADER  
RECORD

N

Y

SET UP STANDARD  
HEADER VALUES  
IN ID ARRAY

USING BYTE  
REFERENCE TABLES,  
PLACE BYTE DATA  
INTO ID ARRAY

RETURN

~~3-15~~

17

READY

```
SUBROUTINE REAMD (NOHD)
COMMON /ICONS/ ID(14), OPTNS(16), IFLAG
DIMENSION IAK(4000), INK(2), IB(3)
DATA IHD /0/
DATA LIST /5,7,11,11,2,3,1,4,4,8,12,12,9,10,10,6,6/
DATA IBYTE /90,91,92,93,102,103,104,105,106,107,108,109,1778,
1785,1786,1787,1788/
IF (IHD) 160, 20, 160
20  NIT = 0
    N = 4000
    IPAR = OPTNS(3)
    CALL ROTAPE(NIT, IA, N, IPAR, KSHR, JCON)
    INK(1) = IAK(1)
    INK(2) = IAK(2)
    CALL CON79 (IN, IB)
    IF (IB(1)) 70, 30, 70
30  IF (IB(2) - 1) 70, 35, 70
    35  WRITE (10,535) IB(2)
535  FORMAT (1X, " SINCE FIRST WORD = ", I3, ", THIS RECORD APPEARS TO
    BE DATA INSTEAD HEADER. WILL TRY TO USE STANDARD VALUES.")
C** NOHD = 1 FOR I-100, 2 FOR LARSYS AND 3 FOR 1100.
    IF (NOHD - 2) 40, 50, 60
40  ID(1) = 1
    ID(2) = 0
    ID(3) = 0
    ID(4) = 70
    ID(5) = 3
    ID(6) = 500
    ID(7) = 8
    ID(8) = 0
    ID(9) = 1
    ID(10) = 3
    ID(11) = 1
```

ORIGINAL PAGE IS  
OF POOR QUALITY

READY

```
      ID (12) = 1
      GO TO 160
50    GO TO 40
C***  USE I-100 UNTIL VALUES FOR LARSYS AVAILABLE
60    GO TO 40
C ***  INSERT UNIVAC CONSTANTS WHEN AVAILABLE
70    I = 57
      K = 87
      L = 1
80    I = I + 2
      IF (I - 75) 90, 85, 90
85    I = 1185
      K = 1776
90    IN(1) = IA(I)
      IN(2) = IA(I + 1)
      CALL CON79 (IN, IB)
      IMD = IA(I + 1)
C     WRITE (10,930) IN,IB, IA(I), IMD, I, K, L
C ***  PUT VALUES FROM TAPE INTO BYTE
      DO 100 N = 1,3
      K = K + 1
      IF (K - IBYTE(L)) 100, 95, 105
95    IBYTE(L) = IB(N)
      L = L + 1
100   CONTINUE
      IF (I - 1191) 80, 110, 110
105   WRITE (10,605) K, IBYTE(L)
605   FORMAT (1X, " HOW CAN K = ", I3, " WHICH IS LARGER THAN", I4)
C ***  TEST FOR TWO-BYTE WORDS AND STORE IN ID
110   L = 1
120   IF (L - 16) 130, 130, 160
130   N = LIST(L)
      K = L + 1
```

3-27  
19

```
IF (LIST(L) - LIST(K)) 140, 150, 140
140 ID(N) = IBYTE(L)
    L = L + 1
    GO TO 120
150 IWD = IBYTE(L) * 400K + IBYTE(K)
    ID(N) = IWD
    L = L + 2
    GO TO 120
160 IHD = 1
C WRITE (10,650) ID
C 650 FORMAT (1X, 10X, "HEADER VALUES FROM TAPE."//2(7I6//))
C *** IHD FLAG SHOWS HEADER RECORD HAS BEEN READ.
RETURN
END
```

READY

~~3-18~~  
20



### 3.2.4 SOFTWARE COMPONENT NO. 4 (INITN)

#### 3.2.4.1 Linkage

Subroutine INITN is called by the driver subroutine DRVF in overlay USER08, and in turn calls user subroutines REAHD and CON79 and system subroutines RDTAPE and SPACE.

#### 3.2.4.2 Interface

The common block ICONS transmits control information to INITN.

#### 3.2.4.3 Input

See 3.2.4.2.

#### 3.2.4.4 Output

None

#### 3.2.4.5 Storage Requirements

Subroutine INITN requires 396 words in core.

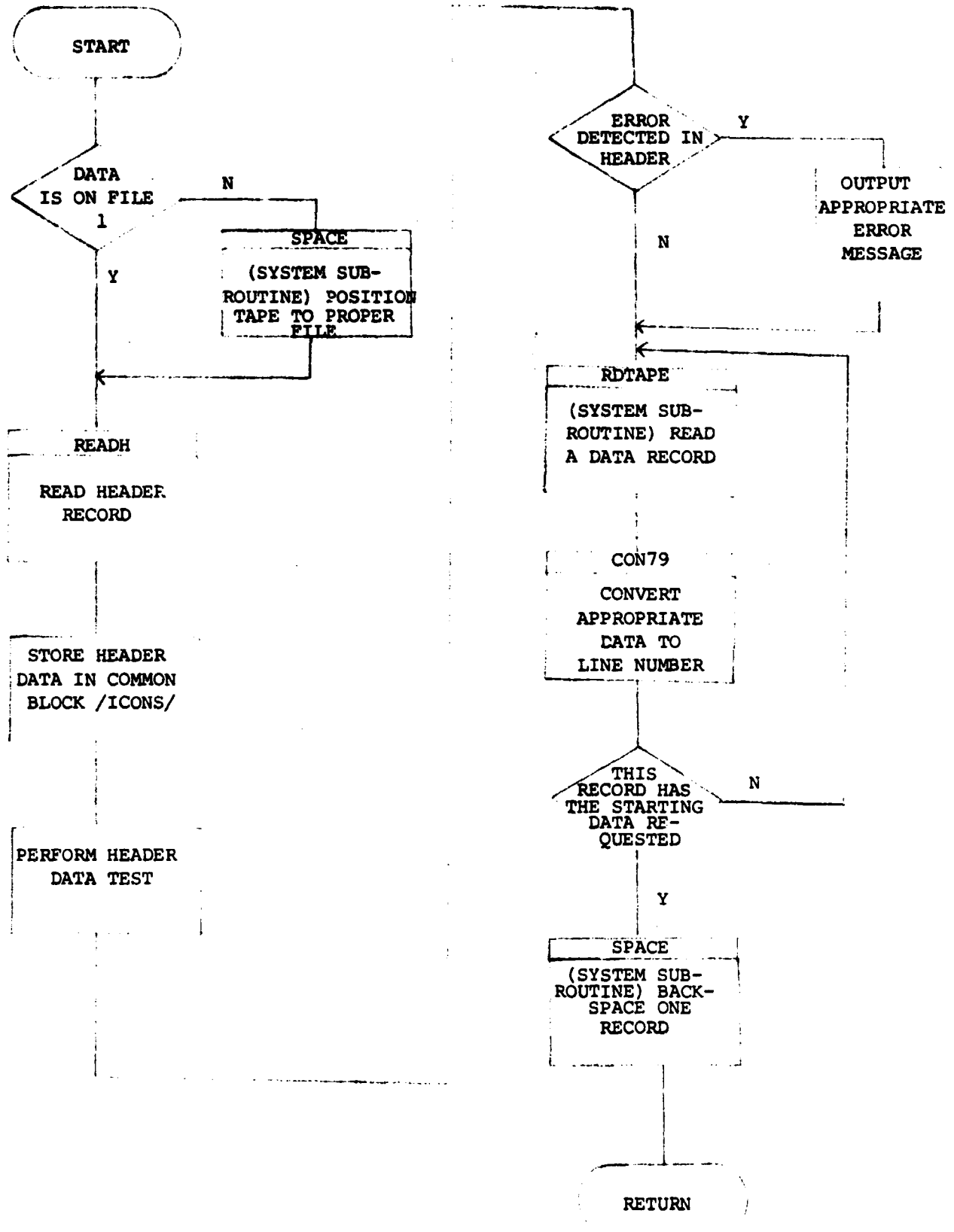
#### 3.2.4.6 Description

Subroutine INITN begins by positioning the input tape to the requested file and reading the header record via subroutine REAHD. Additional error checks are performed on the header data, then the input tape is positioned to the record containing the first data line requested by the user.

#### 3.2.4.7 Flowchart

#### 3.2.4.8 Listing

(INITN)



3-20  
22

READY

```
SUBROUTINE INITN
COMMON /ICONS/ ID(14), OPTNS(16), IFLG1
DIMENSION IA(4080), INK(2), IB(3)
ISYS=1
IFLG1=0
NIT = 0
IFLSK = OPTNS(2)
IFLG1 = 1
N = IFLSK - 1
IF (N) 50,50,40
40 N = N - 1
CALL SPACE (NIT, IFLG1, NIT, ISTAT)
IF (N) 50,50,40
50 CALL REHD (ISYS)
IANS = 1
C*** HEADER DATA TEST BY MINTER.
C*** ONE CHANNEL MUST NOT BE LARGER THAN ONE RECORD.
IF (ID(3) - 1) 80,80,75
75 WRITE (10,575) ID(3)
C*** OR SUM ERRORS USING IERR = IERR + 2
575 FORMAT (1X," FLAG3 = ", I4,". INDICATES CHANNEL LARGER THAN "
"RECORD.")
ID(3) = 1
C*** START OF VIDEO DATA SHOULD BE GREATER THAN ZERO.
80 IF (ID(1)) 85,85,90
85 ID(1) = 1
C*** NUMBER OF DATA SETS PER RECORD IS GREATER THAN ZERO.
90 IF (ID(9)) 95,95,100
95 ID(9) = 1
C*** EXPECT 8 BITS FROM ORIGINAL DATA IN BYTES.
100 IF (ID(7) - 8) 105,110,105
C*** AGAIN FOR ERROR SUM, IERR = IERR + 4
105 WRITE (10,600) ID(7)
```

```

600  FORMAT (1X," NO OF BITS = ", I5)
      IO(7) = 8
C*** POSITION TAPE TO START OF REQUESTED DATA.
110  ITEM = OPTNS(1)
C   610  WRITE (10,610) NIT, ISTAT
C   610  FORMAT (1X," READ DATA RECORD NEXT.  PARITY = ", I5, "  STAT=", I6)
      ISTAT = 8192
      N = 4080
120  CALL ROTAPE (NIT, IA, N, NIT, KSHRT, ISTAT)
      IN(1) = IA(47)
      IN(2) = IA(48)
      CALL CON79 (IN, IB)
      IF (IB(3) - ITEM) 120,140,130
130  WRITE (10,630) IFLSK, IB(3)
630  FORMAT (1X," ON FILE ", I4," FIRST LINE IS", I5)
140  MNUS = -1
      CALL SPACE (NIT, NIT, MNUS, ISTAT)

      RETURN
      END

```

READY

### 3.2.5 SOFTWARE COMPONENT NO. 5 (CON79)

#### 3.2.5.1 Linkage

Subroutine CON79 is called by subroutines INITN and REHD in overlay USER08 and by RDLIN in overlay T2.

#### 3.2.5.2 Interface

Interface is accomplished by one input argument and one output argument.

#### 3.2.5.3 Input

The input argument IA is a two-word array read from 7-track tape.

#### 3.2.5.4 Output

The argument IB is a 3-word output array, one byte/word, right justified.

#### 3.2.5.5 Storage Requirements

Subroutine CON79 requires 64 words in core.

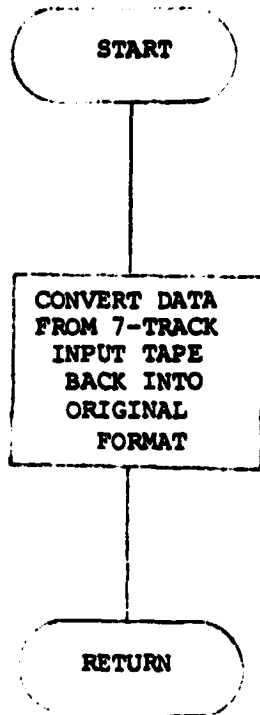
#### 3.2.5.6 Description

Subroutine CON79 is designed to convert 7-track unformatted input data to formatted information in the form it originally appeared in a 9-track tape format. It is specifically designed to restore the data to its form as it appears on a Universally formatted classified tape.

#### 3.2.5.7 Flowchart

#### 3.2.5.8 Listing

(CON79)



~~3-24~~  
26

LIST LINES - 33

READY

TITLE CON79  
PROGRAM ID-SUBROUTINE CON79  
PROGRAMMER-PAUL LINK(LEC 626-45 SOFTWARE  
DEVELOPMENT SECTION)  
DATE-SEPT 3, 1976  
FUNCTION-CONVERT 2 WORDS READ FROM 7 TRACK TAPE  
TO 3 WORDS(1 BYTE/WORD, RIGHT JUSTIFIED)

EXAMPLE:

FROM WORD 1:00NNNNNN 00UUUUUU  
WORD 2:00ZZZZZZ 00YYYYYY  
TO WORD 1:00000000 NNNNNNUU  
WORD 2:00000000 UUUZZZZZ  
WORD 3:00000000 ZZYYYYYY

SOURCE=<CON79:A>  
OBJECT=<CON79:R>

CALLING SEQUENCE:

CALL CON79(IA, IB)

WHERE IA IS A 2-WORD INPUT ARRAY READ FROM 7 TRACK TAPE

IB IS A 3-WORD OUTPUT ARRAY, 1 BYTE/WORD, RIGHT JUSTIFIED

ENT CON79  
EXTD .CPYL, .FRET  
.NREL

CON79 2  
JSR @CPYL  
STA 3,SAVE  
LDA @FTSTR,3  
LDA 2,FTSTR+1,3

LIST LINES - 65

```

      MOV 0,3
      PROCESS 1ST OUTPUT BYTE
      LDA 0,0,3
      LDA 1,MASK1
      AND 1,0
      LDA 1,SHFT4
CON01  MOVZR 0,0
      INC 1,1,SZR
      JMP CON01
      STA 0,TEMP

      LDA 0,0,3
      LDA 1,MASKR
      AND 1,0
      LDA 1,SHFT6
CON02  MOVZR 0,0
      INC 1,1,SZR
      JMP CON02

      LDA 1,TEMP
      ADD 1,0
      STA 0,0,2

      PROCESS 2ND OUTPUT BYTE

      LDA 0,0,3
      LDA 1,MASK2
      AND 1,0
      LDA 1,SHFT4
CON03  MOVZL 0,0
      INC 1,1,SZR

      JMP CON03
      STA 0,TEMP
      LDA 0,1,3
      LDA 1,MASKR
      AND 1,0
      MOVZ 0,0
      MOVZR 0,0
      LDA 1,TEMP
      ADD 1,0
      STA 0,1,2

      PROCESS 3RD OUTPUT BYTE

      LDA 0,1,3
      LDA 1,MASKL
      AND 1,0
      STA 0,TEMP
      LDA 0,1,3
      LDA 1,MASK3
      AND 1,0
      MOVZR 0,0
      MOVZR 0,0

      LDA 1,TEMP
      ADD 1,0
      STA 0,2,2

      LDA 3,SAVE
      JSR 0,FRET

```

READY

~~326~~  
28



LIST LINES - 65

MASK1 000060 ;GET BITS 10,11  
MASK2 000017 ;GET BITS 12-15  
MASK3 001400 ;GET BITS 6,7  
MASK4 000377 ;GET BITS 8-15  
MASKR 177400 ;GET BITS 0-7  
SHFT4 -4  
SHFT6 -6  
SHVE 0  
TEMP 0  
END

READY

~~3-27~~  
29

### 3.2.6 COMPONENT NO. 6 (RDLIN)

#### 3.2.6.1 Linkage

Subroutine RDLIN is the driver (main) routine in overlay T2. RDLIN calls the user subroutines ISET, CON79, and FRAME, as well as various system subroutines which read the input tape and create the temporary data file TDATA. After execution, RDLIN calls in overlay T3.

#### 3.2.6.2 Interface

Subroutine RDLIN communicates with its associate subroutines via the common parameter block ICONS.

#### 3.2.6.3 Input

Subroutine RDLIN accepts input from the 7-track input data tape.

#### 3.2.6.4 Output

RDLIN creates a temporary data file TDATA on the system disk.

#### 3.2.6.5 Storage Requirements

Subroutine RDLIN requires 1054 words in core.

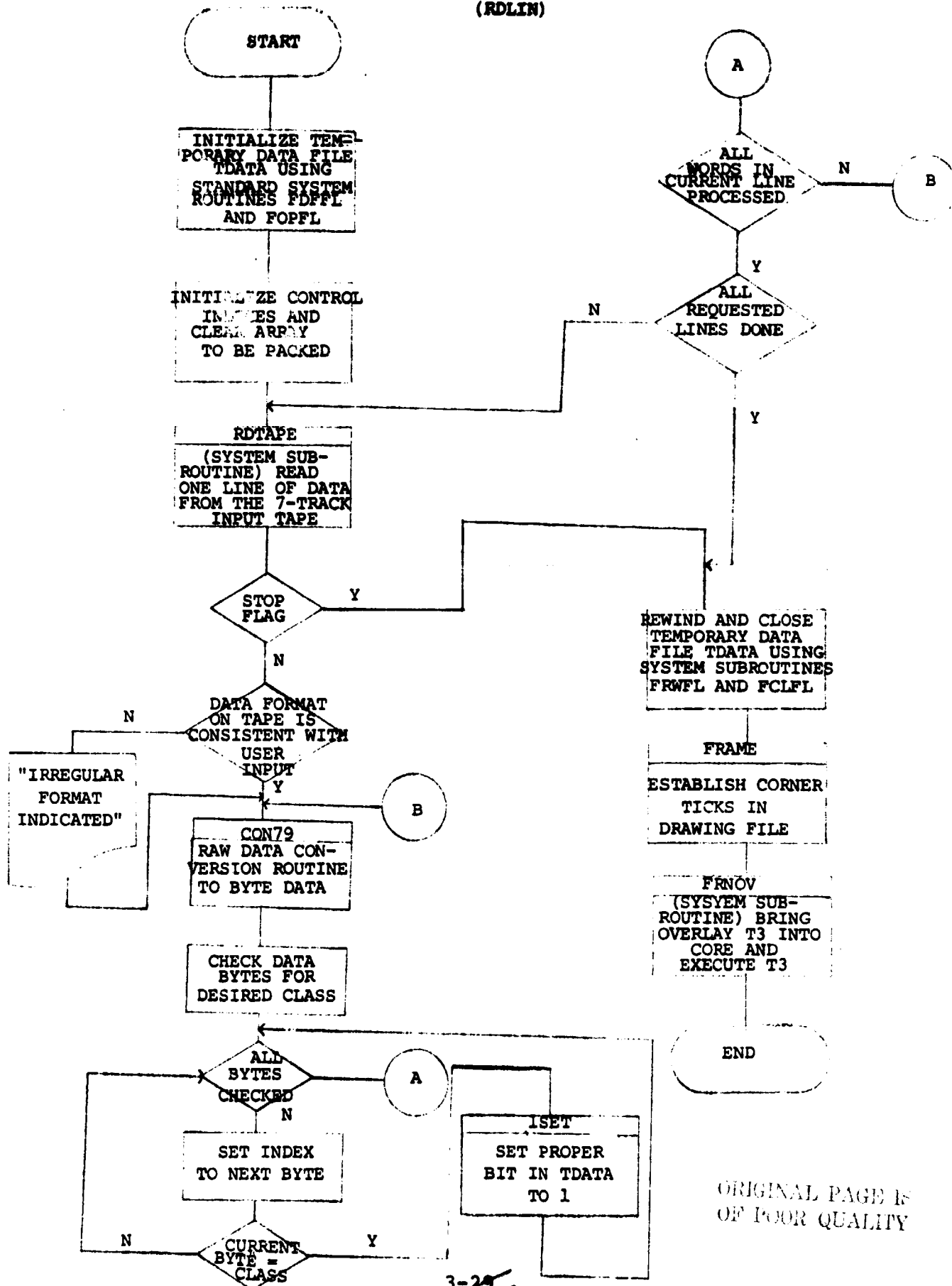
#### 3.2.6.6 Description

Subroutine RDLIN reads the classified data tape, packs the classified data 16 pixels per word, and stores these data on a temporary disk file, TDATA.

#### 3.2.6.7 Flowchart

#### 3.2.6.8 Listing

(RDLIN)



ORIGINAL PAGE IS OF POOR QUALITY

```

                                READY
DIMENSION IN (2), IB(3), IS(50), IA(4000)
COMMON /ICONS/ ID(14), OPTNS(16), IFLG1
DATA LNCNT, IEND /0,0/
NAME (1) = "T3"
NAME (2) = "/1"
NAME (3) = 0
IER = 0
CALL FOPFL ("TDATA", IER)
CALL FOPFL ("TDATA", 1, 1, IER)
50  KT = 1
    NIT = 0
    IPAR = ID(14)
    LINRC = 0
    ISTAT = 0
    NWDS = 0
    DO 60 K = 1, 50
    IS(K) = 0
60  CONTINUE
    IF (IEND - 1) 70, 390, 390
70  ICLS = OPTNS(6)
    N = 4000
80  CALL ROTAPE (NIT, IA, N, IPAR, NSHR, ISTAT)
    NOL = OPTNS (4) - OPTNS (3) + 1
    IF (NOL) 90, 90, 100
90  NOL = ID (6)
    GO TO 115
100 IF (NOL - ID(6)) 115,115,110
110 WRITE (10,615) NOL, ID (6)
615 FORMAT (1X," WANT",I5," PIXELS PER LINE? WILL TRY",I5)
    NOL = ID(6)
    OPTNS (4) = 0.
115 IF (ISTAT - 4) 160, 140, 120
120 IEND = IEND + 1

```

```

                                READY
125 IF (IEND - 1) 140, 125, 390
620 WRITE (10, 620) ISTAT, LNCNT
140 FORMAT (1X, " STATUS WORD =", I3, " TOTAL LINES DONE IS", I5)
160 WRITE (10, 620) ISTAT, LNCNT
160 IF (ID(8) - 1) 170, 400, 400
170 IADD = MOD (IFLG1, 10)
180 IF (IADD - 5) 200, 200, 180
180 IFLG1 = 5 - IADD
    IOO = 0
    GO TO 220
200 IOO = ID(4)
220 ISTAT = OPTNS(3)
    L = OPTNS(5)
    IP = ID(6) * (L - 1) + IOO + ISTAT
    L = 0
240 IADD = 2 * (IP + (ID(6) * ID(5))) * LINRC)
    IOO = MOD (IADD, 3)
    ISTAT = 3 - IOO
    IOO = MOD (ISTAT, 3)
    IADD = IADD/3
    I = IADD + 1
    IF (IOO - 2) 260, 250, 260
250 I = I - 1
260 IF (I - 4079) 270, 270, 350
270 IF (ID(8) - 1) 280, 410, 410
280 IN(1) = IA(I)
    IN(2) = IA(I + 1)
    CALL CON79 (IN, IB)
    N = IOO + 1
    DO 340 J = N, 3
    NWDS = NWDS + 1
    L = L + 1
    IF (L - 16) 300, 300, 290

```

LIST LINES - 33

READY

```
290 L = 1
    KT = KT + 1
300 IF (ICLS - IB(J)) 340, 320, 340
320 CALL ISET (IS(KT), L)
340 CONTINUE
    IOO = 0
    I = I + 2
    IF (NWDS - NOL) 260, 260, 360
350 IFLG1 = MOD (IFLG1, 10)
    IF (IFLG1 - 5) 360, 360, 355
355 IFLG1 = 5 - IFLG1
360 LNCNT = LNCNT + 1
    LINRC = LINRC + 1
    ID (13) = KT
    IBYT = 2 * KT
    CALL FWTF (1, IS, IBYT, IER)
    IF (LINRC - IX(9)) 362, 365, 365
362 KT = 1
    GO TO 240
365 ISTAT = OPTNS (2) - OPTNS(1) + 1.05
    IF (LNCNT - ISTAT) 380, 370, 370
370 IEND = 1
380 IF (LNCNT - 200) 50, 390, 390
390 CONTINUE
    CALL FRWFL (1, IER)
    CALL FCLFL (1, IER)
    IF (IER) 385, 395, 385
385 WRITE (10, 885) IER
885 FORMAT (1X, " ERROR SET AT", I5, " FROM WRITE, REWIND OR CLOSE")
    STOP
395 XSC=0.1
    YSC=0.1
    NLINES=OPTNS(2) - OPTNS(1) + 1.1
```

~~3-32~~

34

ORIGINAL PAGE IS  
OF POOR QUALITY

```
NPX =OPTNS(4) - OPTNS(3) + 1.1
XMAX=NPX
YMAX=NLINES
CALL FRAME(XMAX,YMAX,XSC,YSC)
CALL FRNOU (NAME, IER)
PAUSE FRNOU IN READLINE FAILED.
400 IEND = 1
WRITE (10,900) ID(8)
900 FORMAT (1X," IRREGULAR FORMAT INDICATED BY", I4)
READ (11) I
IF (I) 370, 370, 410
410 ID(8) = -1
IEND = 0
GO TO 170
END
```

READY

~~3-33~~  
35

### 3.2.7 SOFTWARE COMPONENT NO. 7 (ISET)

#### 3.2.7.1 Linkage

Subroutine ISET is called by RDLIN in overlay T2.

#### 3.2.7.2 Interface

RDLIN communicates with subroutine ISET via two calling arguments.

#### 3.2.7.3 Input

The argument IS(KL) is the KLth word in vector IS.

The argument L is the bit number in IS(KL) which needs to be set to 1.

#### 3.2.7.4 Output

The argument IS(KL) is returned with the Lth bit set to 1.

#### 3.2.7.5 Storage Requirements

Subroutine ISET requires 26 words in core.

#### 3.2.7.6 Description

Subroutine ISET sets the appropriate bit in a 16-bit word to indicate a pixel belonging to the class being examined. These words are the packed data which RDLIN packs into the temporary data disk file TDATA.

#### 3.2.7.7 Flowchart

#### 3.2.7.8 Listing



(ISET)

START

SET THE  
APPROPRIATE BITS  
IN A 16-BIT  
WORD TO INDICATE  
WHICH PIXELS  
BELONG TO THE  
CLASS BEING EXAM.

RETURN

~~3-25~~

37

ORIGINAL PAGE IS  
OF POOR QUALITY

LIST LINES - 60

READY

	TITL	ISET
	ENT	ISET
	EXTD	CPYL . . FRET
	NREL	
	2	
ISET	JSR	@.CPYL
	STA	3.SAVE
	LDA	0.@FTSTR,3
	STA	0.VALU
	NEG	0.1
	ADD	0.1
	MOVOR	1.1
	LDA	0.@FTSTR+1,3
	STA	0.@FTSTR+1,3
	STA	0.CONS
LOOP:	LDA	0.CONS
	DSZ	CONS
	JMP	RITS
	LDA	0.VALU
	ADD	0.1
	STA	1.VALU
	JMP	END
RITS	MOVOR	1.1
	JMP	LOOP
END	LDA	1.VALU
	STA	1.@FTSTR,3
	LDA	3.SAVE
	JSR	@.FRET
SAVE	0	
CONS	0	
VALU	0	
	END	

~~3-36~~  
38

### 3.2.8 SOFTWARE COMPONENT NO. 8 (FRAME)

#### 3.2.8.1 Linkage

Subroutine FRAME is called by RDLIN, and calls subroutine LINIT.

#### 3.2.8.2 Interface

FRAME receives format and scaling information through four input parameters.

#### 3.2.8.3 Input

Four calling arguments are input to subroutine FRAME reflecting format and scaling constraints.

#### 3.2.8.4 Output

None

#### 3.2.8.5 Storage Requirements

Subroutine FRAME requires 347 words in core.

#### 3.2.8.6 Description

Subroutine FRAME computes the output frame size, generates four corner ticks for the plot file, and calls subroutine LINIT to write these ticks in the plot file.

#### 3.2.8.7 Flowchart

#### 3.2.8.8 Listing

(FRAME)

START

SET UP X,Y  
ARRAY FOR  
LOWER LEFT TICK

LIMIT  
PLOT LOWER LEFT  
TICK IN  
DRAWING FILE

SET UP X,Y  
ARRAY FOR LOW-  
ER RIGHT TICK

LIMIT  
PLOT LOWER  
RIGHT TICK IN  
DRAWING FILE

SET UP X,Y  
ARRAY FOR  
UPPER RIGHT TICK

LIMIT  
PLOT UPPER  
RIGHT TICK IN  
DRAWING FILE

SET UP X,Y ARRAY  
FOR UPPER  
LEFT TICK

LIMIT  
PLOT UPPER  
LEFT TICK  
IN DRAWING FILE

RETURN

~~3-38~~

40

```

                                READY
SUBROUTINE FRAME(XMAX,YMAX,XSC,YSC)
DIMENSION X(3),Y(3)
ARM=0.5
X(1)=0.0
Y(1)=ARM * YSC
X(2)=0.0
Y(2)=0.0
X(3)=ARM * XSC
Y(3)=0.0
CALL LINIT(X,Y,3,0)
X(1)=(XMAX-ARM) * XSC
Y(1)=0.0
X(2)=XMAX * XSC
Y(2)=0.0
X(3)=X(2)
Y(3)=ARM * YSC
CALL LINIT(X,Y,3,0)
X(1)=XMAX * XSC
X(2)=X(1)
Y(1)=(YMAX-ARM) * YSC
Y(2)=YMAX * YSC
X(3)=(XMAX-ARM) * XSC
Y(3)=Y(2)
CALL LINIT(X,Y,3,0)
X(1)=ARM * XSC
Y(1)=YMAX * YSC
Y(2)=Y(1)
X(2)=0.0
X(3)=0.0
Y(3)=(YMAX-ARM) * YSC
CALL LINIT(X,Y,3,0)
RETURN
END

```

### 3.2.9 SOFTWARE COMPONENT NO. 9 (LINIT)

#### 3.2.9.1 Linkage

In overlay T2 subroutine LINIT is called by subroutine FRAME.  
In overlay T3 subroutine LINIT is called by BDT3, ENDTST, CONALL, CLSTST, CONECT, and FINDAR.

#### 3.2.9.2 Interface

Subroutine LINIT receives control information through the user common block ICONS, and through the System 100 common blocks BLK and MENU1.

#### 3.2.9.3 Input

LINIT receives x,y plot arrays through its calling arguments.

#### 3.2.9.4 Output

Subroutine LINIT transfers registered boundary plot string arrays to a System 100 drawing file.

#### 3.2.9.5 Storage Requirements

Subroutine LINIT requires 313 words in core.

#### 3.2.9.6 Description

LINIT accepts as input plot string arrays. Data registration is accomplished at this point by transforming the x,y coordinates of the plot arrays using either the eight coefficients input by the user or the default (no change) coefficients. The standard expression for the data transformation is:

$$X_t = (A_1X_0 + A_2Y_0 + A_3)/(1 + A_4Y_0 + A_5Y_0)$$

$$Y_t = (A_6X_0 + A_7Y_0 + A_8)/(1 + A_4X_0 + A_5Y_0)$$

where

$A_1-A_9$  are the eight coefficients

$X_0, Y_0$  = Initial or observed coordinates

$X_t, Y_t$  = Transformed coordinates

After transformation, these registered plot string arrays are transferred to a standard System 100 drawing file.

3.2.9.7 Flowchart

3.2.9.8 Listing

(LINIT)

START

Perform 8 parameter transformation using either coefficients input by user or default coefficients

Send Arrays to Drawing File

Return

3-12  
44



LIST LINES -

END

```
SUBROUTINE LINIT(ARX,ARY,N,ITYREADY)
DIMENSION ARX(50),ARY(50)
COMMON /ICONS/ IZ(14),O(16),IFLG1
COMMON /BLK/X(30),Y(30),A(10),K(30),KP,ID(80)
COMMON /MENU1/KODE,MRFLG,SFACT,LNMOD,LNWID
EQUIVALENCE (X(1),X1),(Y(1),Y1),(K(11),K11),(K(12),K12)
EQUIVALENCE (K(14),K14),(K(15),K15),(A(1),A1),(A(2),A2)
C      DESIGNATE FILE NUMBER
      K4=1
      IF(ITYP.EQ.99) GO TO 20
      DO 1 I=1,N
      RX = ARX(I)
      RY = ARY(I)
      D = 1 + O(12) * RX + O(13) * RY
      ARX(I) = ( O(9)*RX + O(10)*RY + O(11) ) / D
1     ARY(I) = ( O(14)*RX + O(15)*RY + O(16) ) / D
C     PEN UP COMMAND
      X1=ARX(1)
      Y1=ARY(1)
      K11=1
      CALL RWCON(K4,2)
C     PEN DOWN COMMAND
      K11=6
      DO 10 I=2,N
      X1=ARX(I)
      Y1=ARY(I)
      CALL RWCON(K4,2)
10    CONTINUE
      GO TO 90
20    K11=31
C     WRITE END-OF-FILE COMMAND
      CALL PWCON(K4,2)
90    RETURN
```

~~3-43~~

45

### 3.2.10 SOFTWARE COMPONENT NO. 10 (BDT3)

#### 3.2.10.1 Linkage

Subroutine BDT3 is the principal routine in overlay T3, and calls the following user subroutines: FINDAR, FILL, READAT, CONALL, ENDTST, CLSTST, and LINIT. In addition, subroutine BDT3 utilizes the following system subroutines for drawing file manipulation: FOPFL, FCLFL, FDLFL, and FCNOT.

#### 3.2.10.2 Interface

Subroutine BDT3 receives control information through the common block ICONS. BDT3 communicates with its associate subroutines via the common blocks Z, ZZ, and MAXFIL.

#### 3.2.10.3 Input

Pixel data is brought in, line by line, from the disk file TDATA by subroutine READAT and placed in common block ZZ for processing in BDT3.

#### 3.2.10.4 Output

While overlay T3 is operating, BDT3 outputs a status message on the display device after each ten lines of requested data has been processed. In addition, BDT3 actuates the audible tone on the output device after processing is complete.

#### 3.2.10.5 Storage Requirements

Subroutine BDT3 requires 9224 words in core.

#### 3.2.10.6 Description

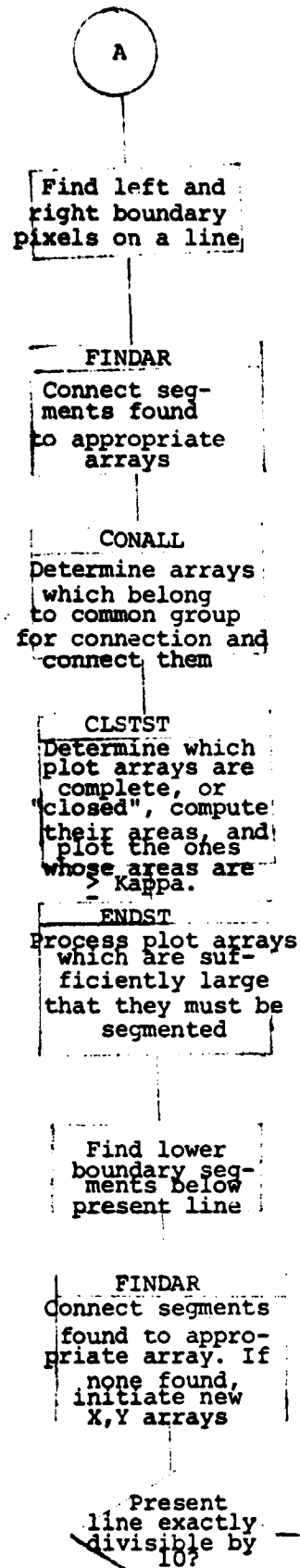
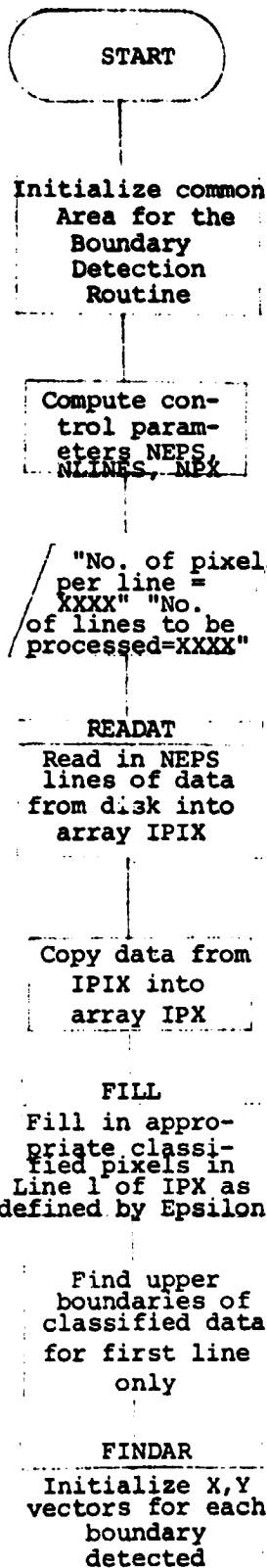
BDT3 is the routine which identifies, from the input pixel information, boundaries of a classified area or areas within a classified image. It processes the input data one line at a time,

identifying border pixels, and building plot string arrays which describe the limits of the classified areas. These plot strings are introduced into a standard system 100 drawing file through subroutine LINIT, which also accomplishes data transformation as specified by eight user-input coefficients, for registration onto any desired base. Ultimately, this drawing file is output on a magnetic tape which is used as input to the Gerber plotter, which creates the final registered boundary plot.

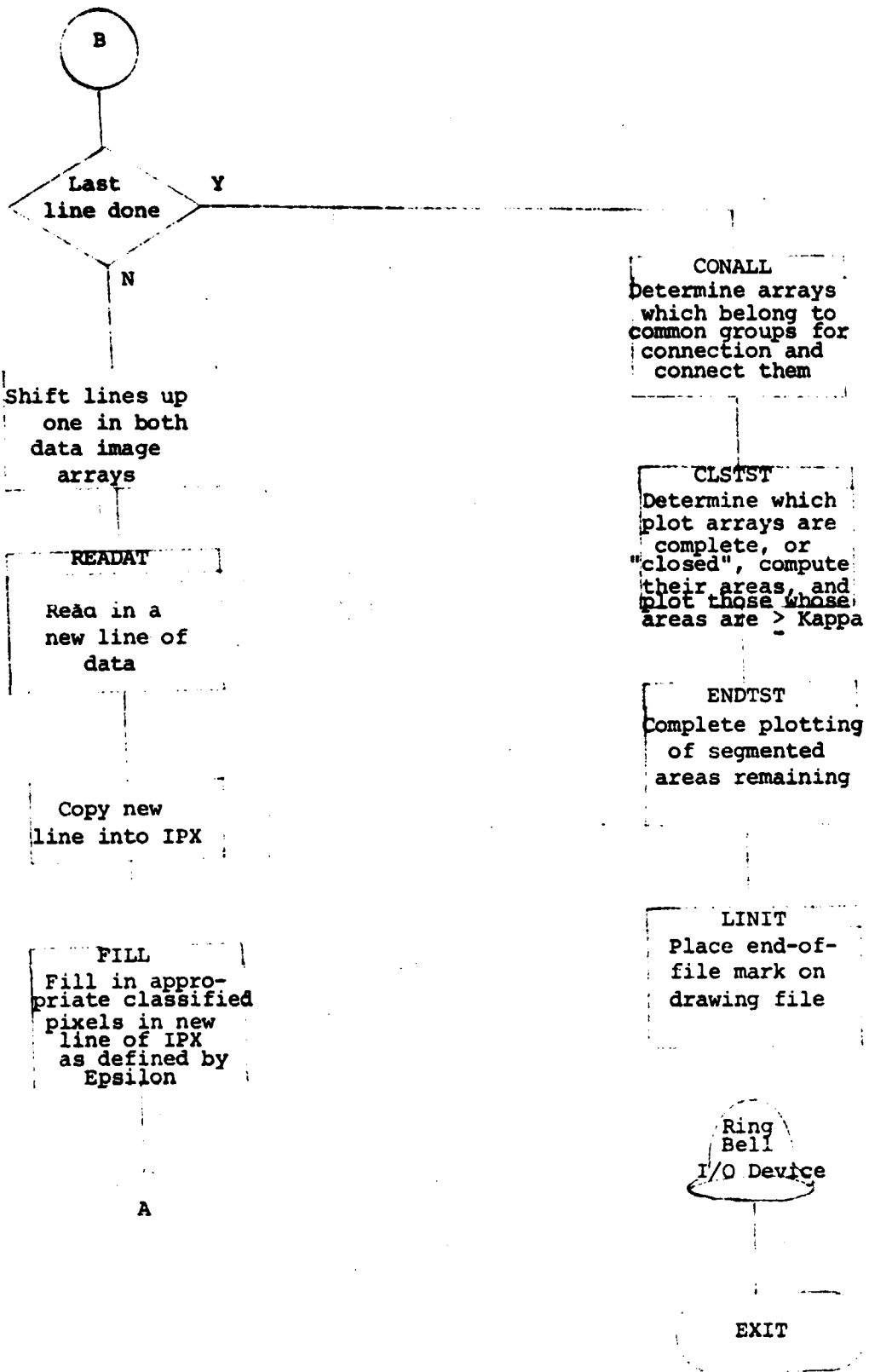
3.2.10.7 Flowchart

3.2.10.8 Listing

(BDT3)



~~346~~  
48



3-AT  
49

READY

```
COMMON /MAXFIL/ MXA
COMMON /Z/ NGRUP,ARRAYX(50,50),ARRAYY(50,50),ISIZE(50),ASIZE(50),
,ARX(50),ARY(50),LINE,YMAX,XSC,YSC,KAPPA
COMMON /ZZ/ IPIX(4,256),IPX(4,256),NPX,EPS
COMMON /ICONS/ ID(14),OPTNS(16),IFLG1
DIMENSION IAZ(256)
INTEGER ARRAYX,ARRAYY
MAXGRP=50
MXA = 0
EPS=OPTNS(7)
KAPPA=OPTNS(8)+0.1
IBYTE=2*ID(13)
XSC=0.1
YSC=0.1
KZ=1
NEPS=EPS + 1.0
NLINES=OPTNS(2) - OPTNS(1) + 1.1
NPX =OPTNS(4) - OPTNS(3) + 1.1
WRITE (10,3)NPX,NLINES
XMAX=NPX
YMAX=NLINES
C CALL FRAME(XMAX,YMAX,XSC,YSC)
HELEM=0
IF(NEPS.GT.4) WRITE(10,2) EPS
2 FORMAT(' EPSILON VALUE OF',F7.3,' EXCEEDS PRESENT PROGRAM CONSTRAI
NTS')
3 FORMAT(' NO. OF PIXELS PER LINE =",I4,/,/,
" NO. OF LINES TO BE PROCESSED =",I4,/,/)
CALL FOPFL("TDATA",2,0,IE)
IF(IE.EQ.0) GO TO 5
WRITE(10,599) IE
599 FORMAT(1X," IE=",I4)
PAUSE ERROR IN OPENING TDATA IN BOT3
```

READY

```
GO TO 990
5 CONTINUE
DO 7 I=1,NEPS
CALL READAT(IAZ,IBYTE)
DO 4 JZ=1,NPX
4 IPIX(I,JZ)=IAZ(JZ)
7 CONTINUE
DO 8 I=1,NEPS
DO 8 J=1,NPX
8 IPX(I,J)=IPIX(I,J)
LREAD=NEPS
CALL FILL
N=0
NGRUP=0
IFL=0
DO 10 IA=1,MAXGRP
ASIZE(IA)=0.
10 ISIZE(IA)=0.
C FIND UPPER BOUNDARIES ABOVE FIRST LINE ONLY
DO 100 IA=1,NPX
IF(IPX(1,IA)) 100,100,20
20 AX=IA-1
AY=0.
BX=IA
BY=0.
CALL FINDAR(AX,AY,BX,BY,0)
100 CONTINUE
LINE=0
C FIND LEFT AND RIGHT BOUNDARY PIXELS ON A LINE
C STORE LINE SEGMENTS IN APPROPRIATE ARRAYS
110 LINE=LINE+1
IT1=1
115 N1=IT1
```

READY

```
DO 200 IA=N1,NPX
  IF< IPX(1,IA)>> 200,200,120
120 AX=IA-1
  AY=LINE-1
  BX=IA-1
  BY=LINE
  CALL FINDAR<AX,AY,BX,BY,0>
  DO 150 IB=IA,NPX
  IF< IB.EQ.NPX> GO TO 125
  IF< IPX(1,IB)>> 130,130,150
125 IF< IPX(1,IB)>> 130,130,126
126 AX=IB
  AY=LINE-1
  BX=IB
  BY=LINE
  GO TO 135
130 AX=IB-1
  AY=LINE-1
  BX=IB-1
  BY=LINE
135 CALL FINDAR<AX,AY,BX,BY,0>
  IF< IB.EQ.NPX> GO TO 210
  IT1=IB
  GO TO 115
150 CONTINUE
200 CONTINUE
210 CONTINUE
C TEST FOR ARRAYS NOT CONTAINING ARRAY<1,MAX>=LINE
  IF< MOD<LINE,KZ>.NE.0> GO TO 290
  NNN=0
220 IPR0B=0
  DO 223 IG=1,NGRUP
  IG1=ISIZE<IG>
```

~~3-50~~

52



READY

```
223 CONTINUE
    CALL CONALL(IPROB)
    CALL CLSTST
    IF(IPROB.EQ.1) GO TO 220
    CALL ENDTST
290 CONTINUE
C   FIND LOWER BOUNDARIES BELOW A LINE
    ISKP=0
    DO 300 IA=1,NPX
    IF(ISKP) 291,291,401
291 IF(IPX(1,IA).EQ.IPX(2,IA)) GO TO 300
    IF(IA.EQ.NPX) GO TO 299
    IPX1=IPX(1,IA)
    IPX2=IPX(2,IA)
    IPX3=IPX(1,IA+1)
    IPX4=IPX(2,IA+1)
    IF(IPX2.EQ.IPX4) GO TO 299
    IF(IPX1.EQ.IPX3)GO TO 299
    IF((IA-1).EQ.NPX) GO TO 402
    IPX5=IPX(1,IA+2)
    IPX6=IPX(2,IA+2)
    IF(IPX6.EQ.IPX4) GO TO 402
    IF(IPX3.EQ.IPX5) GO TO 402
    IF(IPX1.EQ.1.AND.EPS.GE.1.414) GO TO 410
    IF(IPX1.EQ.0.AND.EPS.LT.1.414) GO TO 410
    AX=IA
    AY=LINE
    BX=IA-1
    BY=LINE
    CALL FINDAR(AX,AY,BX,BY,2)
    AX=IA+1
    AY=LINE
    BX=IA+2
```

READY

```
BY=LINE
CALL FINDAR(AX,AY,BX,BY,2)
AX=IA
AY=LINE
BX=IA+1
BY=LINE
CALL FINDAR(AX,AY,BX,BY,4)
ISKP=2
GO TO 300
402 IF(IPX1.EQ.1.AND.EPS.LT.1.414) GO TO 415
    IF(IPX1.EQ.0.AND.EPS.GE.1.414) GO TO 415
410 AX=IA
    AY=LINE
    BX=IA+1
    BY=LINE
    CALL FINDAR(AX,AY,BX,BY,2)
    AX=IA-1
    AY=LINE
    BX=IA
    BY=LINE
    CALL FINDAR(AX,AY,BX,BY,2)
    ISKP=1
    GO TO 300
415 AX=IA
    AY=LINE
    BX=IA-1
    BY=LINE
    CALL FINDAR(AX,AY,BX,BY,2)
    AX=IA
    AY=LINE
    BX=IA+1
    BY=LINE
    CALL FINDAR(AX,AY,BX,BY,3)
```

~~52~~  
54

READY

```
      ISKP=1
      GO TO 300
299  AX=IA-1
      AY=LINE
      BX=IA
      BY=LINE
      CALL FINDAR(AX,AY,BX,BY,1)
      GO TO 300
401  ISKP=ISKP-1
300  CONTINUE
      IF(MXA LT.20) GO TO 300
      WRITE(10,307)
307  FORMAT(1X," THIS CLASS TOO DENSE TO PROCESS A SECTOR THIS LARGE",
/,2X,"RETRY PROGRAM USING A SMALLER SPAN OF PIXELS/LINE")
      GO TO 990
308  CONTINUE
      IF(MOD(LINE,10).NE.0) GO TO 305
      WRITE(10,306) LINE
306  FORMAT(1X,I4," LINES PROCESSED")
305  CONTINUE
      IF(LINE.GE.NLINES) GO TO 999
C    SHIFT LINES UP ONE IN BOTH ARRAYS
      NEP=NEPS-1
      DO 310 I=1,NEP
      DO 310 J=1,NPX
      IPIX(I,J)=IPIX(I+1,J)
      IPX(I,J)= IPX(I+1,J)
310  CONTINUE
C    READ IN NEW LINE
      IF(LREAD.GE.NLINES) GO TO 500
      LREAD=LREAD+1
      CALL READAT(IAZ,IBYTE)
      DO 320 JZ=1,NPX
```

```

320 IPIX(NEPS,JZ)=IAZ(JZ)
GO TO 600
500 DO 501 J=1,NPX
501 IPIX(NEPS,J)=0
600 CONTINUE
DO 700 J=1,NPX
700 IPX(NEPS,J)=IPIX(NEPS,J)
CALL FILL WILL BE INSERTED HERE
CALL FILL
GO TO 110
999 NNN=1
1000 IPROB=0
CALL CONALL(IPROB)
CALL CLSTST
CALL ENDTST
DO 800 J=1,NGRUP
IZ=ISIZE(J)
IF(IZ) 800,800,750
750 IF(IPROB.EQ.1) GO TO 1000
GO TO 801
800 CONTINUE
801 CONTINUE
CALL LINIT(ARX,ARY,11,99)
990 CONTINUE
CALL FCLFL(2,IE)
CALL FDLFL("TDATA",IE)
CALL FCNOT("<?>")
CALL OURLY(1,IER) ; RETURN TO PROG1 OF SYSTEM 101
PAUSE OURLY ERROR-NO RETURN TO SYSTEM 101
END

```

READY

~~54~~  
56

### 3.2.11 SOFTWARE COMPONENT NO. 11 (READAT)

#### 3.2.11.1 Linkage

Subroutine READAT is called by subroutine BDT3, and calls subroutine IGET.

#### 3.2.11.2 Interface

READAT transmits pixel information through the following two calling arguments:

IA - vector containing one line of classified pixel indicators, unpacked to one pixel per word.

IBYTE - number of bytes/line to be read from the temporary data file TDATA.

#### 3.2.11.3 Input

Subroutine READAT reads in bit images of line data from the data disk file, TDATA.

#### 3.2.11.4 Output

An error message may be displayed if a disk read error is encountered.

#### 3.2.11.5 Storage Requirements

Subroutine READAT requires 125 words in core.

#### 3.2.11.6 Description

Subroutine READAT reads in one line of packed pixel data from TDATA, unpacks the data into array IA using subroutine IGET, and transfers this line of data to subroutine BDT3.

3.2.11.7 Flowchart

3.2.11.8 Listing

```
SUBROUTINE READAT(IA,IBYTE)
DIMENSION IRAY(16),IA(256)
IWDS = IBYTE / 2
CALL FROFL(2,IRAY,IBYTE,IBYTR,IE)
IF(IE.EQ.0) GO TO 12
PAUSE DISK READ ERROR IN SUBROUTINE READAT
12 CONTINUE
IPT=0
DO 50 I=1,IWDS
DO 40 J=1,16
L=J
IT=IRAY(I)
CALL IGET(IT,L)
IPT=IPT + 1
IA(IPT)=IT
40 CONTINUE
50 CONTINUE
RETURN
END
```

READY

~~3-58~~

59

### 3.2.12 SOFTWARE COMPONENT NO. 12 (IGET)

#### 3.2.12.1 Linkage

Subroutine IGET is called exclusively by subroutine READAT.

#### 3.2.12.2 Interface

Communication with READAT is accomplished through two calling arguments.

#### 3.2.12.3 Input

Subroutine READAT requests the status of the Lth bit of word I from subroutine IGET.

#### 3.2.12.4 Output

Subroutine IGET outputs the status of the Lth bit for READAT.

#### 3.2.12.5 Storage Requirements

Subroutine IGET unpacks the bit data read from the disk data file TDATA into subroutine READAT.

#### 3.2.12.6 Description

Subroutine IGET unpacks the bit data read from the disk data file TDATA into subroutine READAT.

#### 3.2.12.7 Flowchart

#### 3.2.12.8 Listing

3-59  
60



(IGET)

START

Unpack the 16-bit data words which READAT extracted from the temporary data file TDATA

RETURN

~~3-60~~  
61

READY

	TITL	IGET
	.ENT	IGET
	.EXTD	.CPYL, .FRET
	.NREL	
	2	
IGET:	JSR	@.CPYL
	STA	3.RETN
	LDA	0,@FTSTR+1,3
	STA	0,CONS
	NEG	0,1
	ADD	0,1
	MOUR	1,1
POOL:	DSZ	CONS
	JMP	STIR
	JMP	MSK
STIR:	MOUR	1,1
	JMP	POOL
MSK:	LDA	0,@FTSTR,3
	STA	1,@FTSTR+1,3
	AND	0,1,SZR
	JMP	DOUT
	LDA	1,ZERO
	STA	1,@FTSTR,3
	JMP	BACK
DOUT:	LDA	1,ONE1
	STA	1,@FTSTR,3
BACK:	LDA	3.RETN
	JSR	@.FRET
PETN	0	
CONS	0	
ZERO	0	
ONE1	1	
	.END	

~~3-61~~  
022

### 3.2.13 SOFTWARE COMPONENT NO. 13 (FILL)

#### 3.2.13.1 Linkage

Subroutine FILL is called exclusively by subroutine BDT3.

#### 3.2.13.2 Interface

Communication of data between subroutines FILL and BDT3 is accomplished through the common block ZZ.

#### 3.2.13.3 Input

The data block IPIX enters subroutine FILL via ZZ.

#### 3.2.13.4 Output

The data block IPX exits subroutine FILL via ZZ.

#### 3.2.13.5 Storage Requirements

Subroutine FILL requires 584 words of core.

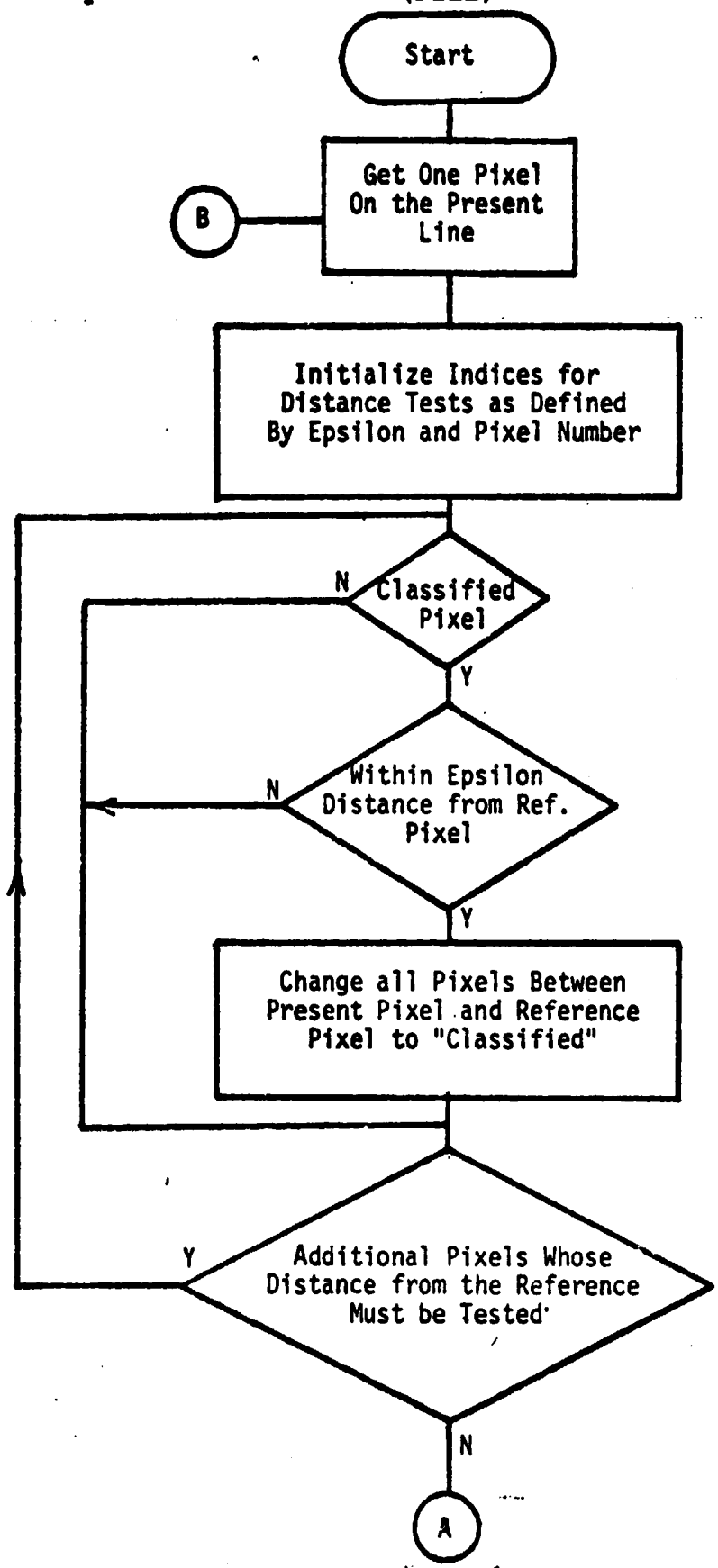
#### 3.2.13.6 Description

Subroutine FILL redefines appropriate pixels as classified to facilitate connectivity of "close" groups. The user defines the criteria for "closeness" via the input parameter Epsilon.

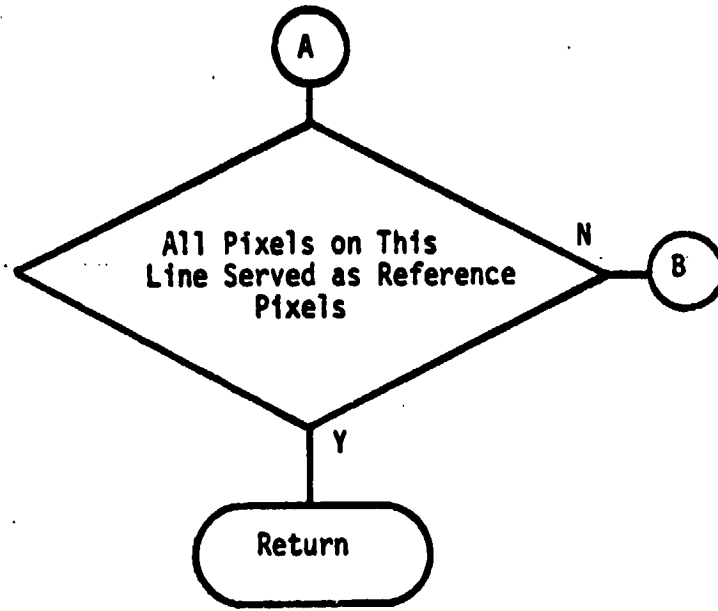
#### 3.2.13.7 Flowchart

#### 3.2.13.8 Listing

(FILL)



~~3-63~~  
64



~~3-64~~  
65

READY

```
SUBROUTINE FILL
COMMON /ZZ/ IPIX(4,256), IPX(4,256), NPX, EPS
N=EPS
L=1
  DO 100 IP=1, NPX
  IF( IPIX(1, IP).EQ.0 ) GO TO 100
  LFP=IP-N
  IF( LFP.LT.1 ) LFP=1
  IRP=IP+N
  IF( IRP.GT.NPX ) IRP=NPX
  IBR=L+N
  DO 20 J=L, IBR
  DO 10 I=LFP, IRP
  IF( J.EQ.L.AND.I.LE.IP ) GO TO 10
  IF( IPIX(J, I) ) 10, 10, 11
11 JPIX=( I-IP)**2 + ( J-L)**2
  PIXDST=SQRT(FLOAT(JPIX))
  IF( PIXDST.GT.EPS ) GO TO 10
  IF( IABS( I-IP ).EQ. IABS( J-L ) ) GO TO 10
  IF( I-IP ) 12, 14, 13
12 IPLUS=I+1
  DO 15 II=IPLUS, IP
15 IPX( J, II )=1
  GO TO 14
13 IMIN=I-1
  DO 16 II =IP, IMIN
16 IPX( J, II )=1
14 IF( ( J-L ).LE.1 ) GO TO 10
  LPLUS=L+1
  DO 17 JJ=LPLUS, J
17 IPX( JJ, IP )=1
  GO TO 10
18 IF( I-IP ) 19, 14, 21
```

```

RETURN
END
19 IPLU=I+1
   IPM=IP-1
   IF((IPM-IPLU).LT.0) GO TO 10
   JJ=J
   DO 30 II=IPLU,IPM
   JJ=J-1
30  IPX(JJ,II)=1
   GO TO 10
21  IPPL=IP+1
   IM=I-1
   IF((IM-IPPL).LT.0) GO TO 10
   JJ=L
   DO 40 II=IPPL,IM
   JJ=JJ+1
40  IPX(JJ,II)=1
10  CONTINUE
20  CONTINUE
100 CONTINUE
   IF(N.LT.2) GO TO 300
   NPXX=NPX-N
   DO 200 IP=1,NPXX
   IF(IPIX(2,IP).EQ.0) GO TO 200
   IRB=IP+N
   IPP=IP+2
   DO 220 I=IPP,IRB
   IF(IPIX(2,I).EQ.0) GO TO 220
   IP1=IP+1
   IR1=I-1
   DO 240 J=IP1,IR1
240 IPX(2,J)=1
220 CONTINUE
200 CONTINUE
300 CONTINUE

```

READY

### 3.2.14 SOFTWARE COMPONENT NO. 14 (FINDAR)

#### 3.2.14.1 Linkage

Subroutine FINDAR is called by subroutine BDT3, and calls subroutines CONECT, AREAL, and LINIT.

#### 3.2.14.2 Interface

Subroutine FINDAR receives control information via common blocks Z and MAXFIL (see Appendix A), and via five calling arguments.

#### 3.2.14.3 Input

None

#### 3.2.14.4 Output

None

#### 3.2.14.5 Storage Requirements

Subroutine FINDAR requires 693 words in core.

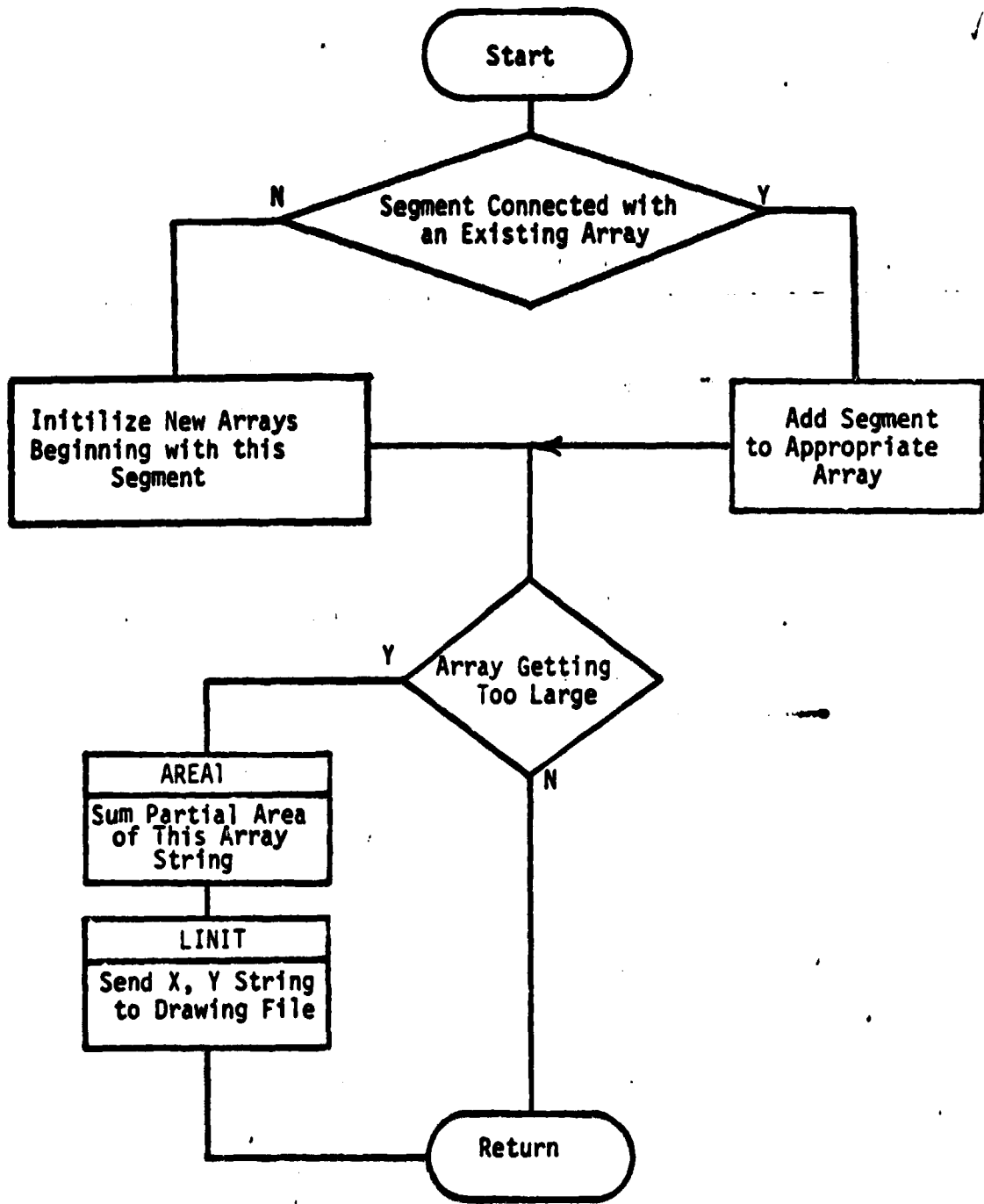
#### 3.2.14.6 Description

Subroutine FINDAR accepts as input a boundary line segment, finds the plot string array, if any, to which the segment connects, and adds it. If no such array exists, new arrays are formed initializing on this segment.

#### 3.2.14.7 Flowchart

#### 3.2.14.8 Listing





3-88

69

```

                                READY
SUBROUTINE FINDAR(XA, YA, XB, YB, ITYPE)
COMMON /MAXFIL/ MXA
COMMON /Z/ NGRUP, ARRAYX(50,50), ARRAYY(50,50), ISIZE(50), ASIZE(50),
      . ARX(50), ARY(50), LINE, YMAX, XSC, YSC, KAPPA
INTEGER ARRAYX, ARRAYY
MAXGRP=50
IPASS=0
IF(ITYPE-2) 9,9,210
9 IF(NGRUP) 200,200,10
10 DO 100 IA=1,NGRUP
    I1=ISIZE(IA)
    IF(I1-1) 100,100,11
11 DIF1=XA-ARRAYX(IA,I1)
    IF(ABS(DIF1).GT.0.01) GO TO 100
    DIF2=YA-ARRAYY(IA,I1)
    IF(ABS(DIF2).GT.0.01) GO TO 100
    IHOLD=IA
    IF(I1-1) 45,45,12
12 I1M=I1-1
    XO=ARRAYX(IA,I1M)
    YO=ARRAYY(IA,I1M)
    IF(ABS(XO-XA).GT.0.01) GO TO 25
    IF(ABS(XA-XB).GT.0.01) GO TO 25
    GO TO 46
25 IF(ABS(YO-YA).GT.0.01) GO TO 45
    IF(ABS(YA-YB).GT.0.01) GO TO 45
    GO TO 46
45 ISIZE(IA)=ISIZE(IA) + 1
    I1=ISIZE(IA)
46 ARRAYX(IA,I1)=XB
    ARRAYY(IA,I1)=YB
    IF(ITYPE-1) 902,902,900
100 CONTINUE

```

READY

```
101 IF( IPASS ) 101,101,200
    IPASS=1
    IF( ITYPE-1 ) 110,110,200
110 TEMP=XB
    XB=XA
    XA=TEMP
    TEMP=YB
    YB=YA
    YA=TEMP
    GO TO 10
200 IF( IPASS ) 210,210,201
201 TEMP=XA
    XA=XB
    XB=TEMP
    TEMP=YA
    YA=YB
    YB=TEMP
210 DO 300 IA=1,MAXGRP
    IF( ISIZE(IA) ) 220,220,300
220 ARRAYX(IA,1)=XA
    ARRAYX(IA,1)=YA
    ARRAYX(IA,2)=XB
    ARRAYX(IA,2)=YB
    ISIZE(IA)=2
    IF( IA.GT.NGRUP ) NGRUP=IA
    IHOLD=IA
    GO TO 900
300 CONTINUE
    MXA = MXA + 1
    WRITE(10,5) LINE
    5 FORMAT(20X,'ALL ARRAYS FILLED AT LINE',I4)
    GO TO 990
900 IF( ITYPE-3 ) 901,901,990
```

~~3-70~~

71

```
901 CALL CONECT(IHOLD)
902 DO 909 IA=1,NGRUP
    NU=ISIZE(IA)
    IF(NU.LT.49) GO TO 909
    DO 920 JK=1,NU
        ARX(JK)=ARRAYX(IA,JK) * XSC
920  ARY(JK)=(YMAX-ARRAYY(IA,JK)) * YSC
        CALL AREA1(IA,AREA)
        CALL LINIT(ARX,ARY,NU,0)
        ASIZE(IA)=ASIZE(IA)+AREA
        ARRAYX(IA,1)=ARRAYX(IA,NU)
        ARRAYY(IA,1)=ARRAYY(IA,NU)
        ISIZE(IA)=1
909  CONTINUE
990  CONTINUE
    RETURN
    END
```

READY

~~3-71~~  
72

### 3.2.15 SOFTWARE COMPONENT NO. 15 (CONECT)

#### 3.2.15.1 Linkage

Subroutine CONECT is called by subroutine FINDAR, and calls subroutines LINIT, AREAL, and JOIN.

#### 3.2.15.2 Interface

Subroutine CONECT receives control information through common block Z (see Appendix A).

#### 3.2.15.3 Input

None

#### 3.2.15.4 Output

None

#### 3.2.15.5 Storage Requirements

Subroutine CONECT requires 435 words in core.

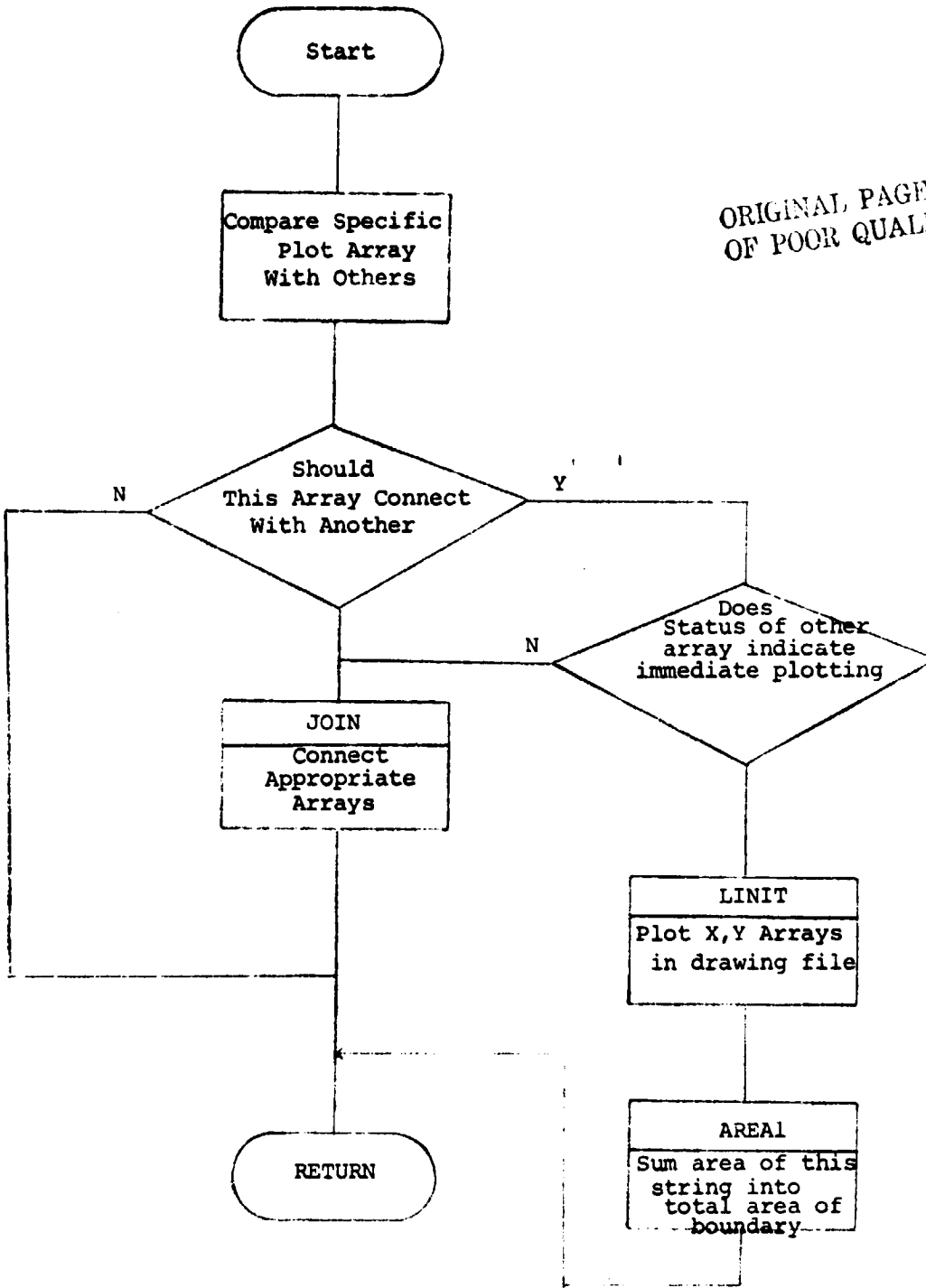
#### 3.2.15.6 Description

Subroutine CONECT accepts as input a particular plot string array and forces immediate connection with the appropriate other plot string array.

#### 3.2.15.7 Flowchart

#### 3.2.15.8 Listing

(CONNECT)



ORIGINAL PAGE IS OF POOR QUALITY

READY

```
SUBROUTINE CONECT(IH)
COMMON /Z/ NGRUP,ARRAYX(50,50),ARRAYY(50,50),ISIZE(50),ASIZE(50),
,ARX(50),ARY(50),LINE,YMAX,XSC,YSC,KAPPA
INTEGER ARRAYX,ARRAYY
IB=ISIZE(IH)
XA=ARRAYX(IH,IB)
YA=ARRAYY(IH,IB)
IF(NGRUP) 909,909,10
10 DO 100 I=1,NGRUP
IF(I.EQ.IH) GO TO 100
I1=ISIZE(I)
IF(I1.LT.1) GO TO 100
DIF1=XA-ARRAYX(I,I1)
IF(ABS(DIF1).GT.0.01) GO TO 100
DIF2=YA-ARRAYY(I,I1)
IF(ABS(DIF2).GT.0.01) GO TO 100
IH2=I
GO TO 102
100 CONTINUE
GO TO 909
102 IB2=ISIZE(IH2)
IF(IB2-1) 909,200,900
200 J=IB+1
DO 300 I=1,IB
J=J-1
ARX(I)=ARRAYX(IH,J)
300 ARY(I)=ARRAYY(IH,J)
DO 400 I=1,IB
ARRAYX(IH,I)=ARX(I)
400 ARRAYY(IH,I)=ARY(I)
DO 500 I=1,IB
ARX(I)=ARRAYX(IH,I) * XSC
500 ARY(I)=(YMAX-ARRAYY(IH,I)) * YSC
```

```
CALL LINIT(ARX,ARY,IB,0)
CALL AREA1(IH,AREA)
ASIZE(IH2)=ASIZE(IH2) + AREA
ARRAYX(IH2,1)=ARRAYX(IH,IB)
ARRAYY(IH2,1)=ARRAYY(IH,IB)
ISIZE(IH)=0
GO TO 909
900 CALL JOIN(IH,IB,IH2,IB2,3)
909 RETURN
END
```

READY



### 3.2.16 SOFTWARE COMPONENT NO. 16 (CONALL)

#### 3.2.16.1 Linkage

Subroutine CONALL is called by subroutine BDT3, and calls subroutines JOIN, AREAL, and LINIT.

#### 3.2.16.2 Interface

Subroutine CONALL receives control information through common block Z (see Appendix 1) and one calling argument.

#### 3.2.16.3 Input

None

#### 3.2.16.4 Output

None

#### 3.2.16.5 Storage Requirements

Subroutine CONALL requires 513 words in core.

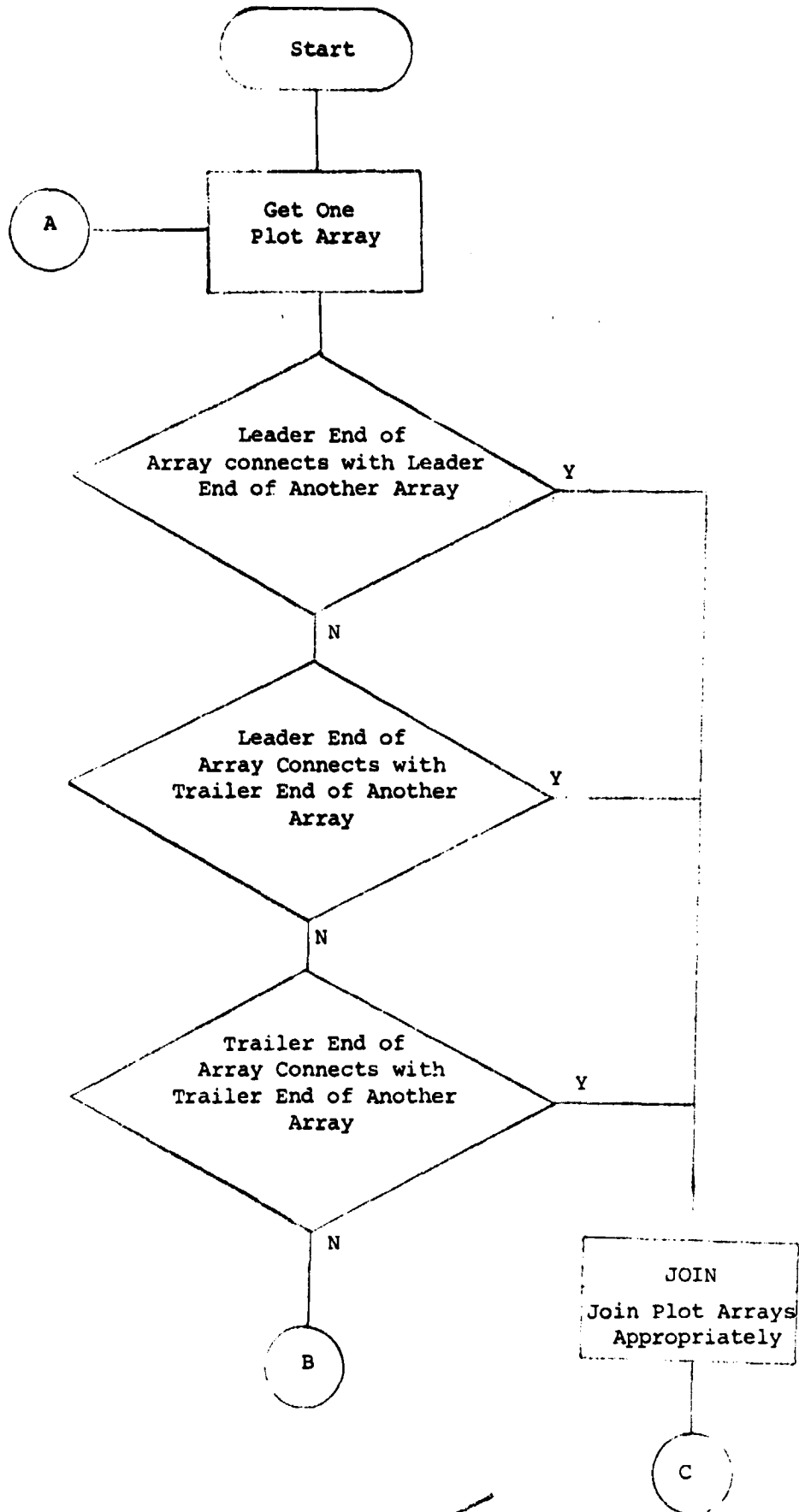
#### 3.2.16.6 Description

Subroutine CONALL determines which plot string arrays should be linked or connected, and how they should be joined (ordering).

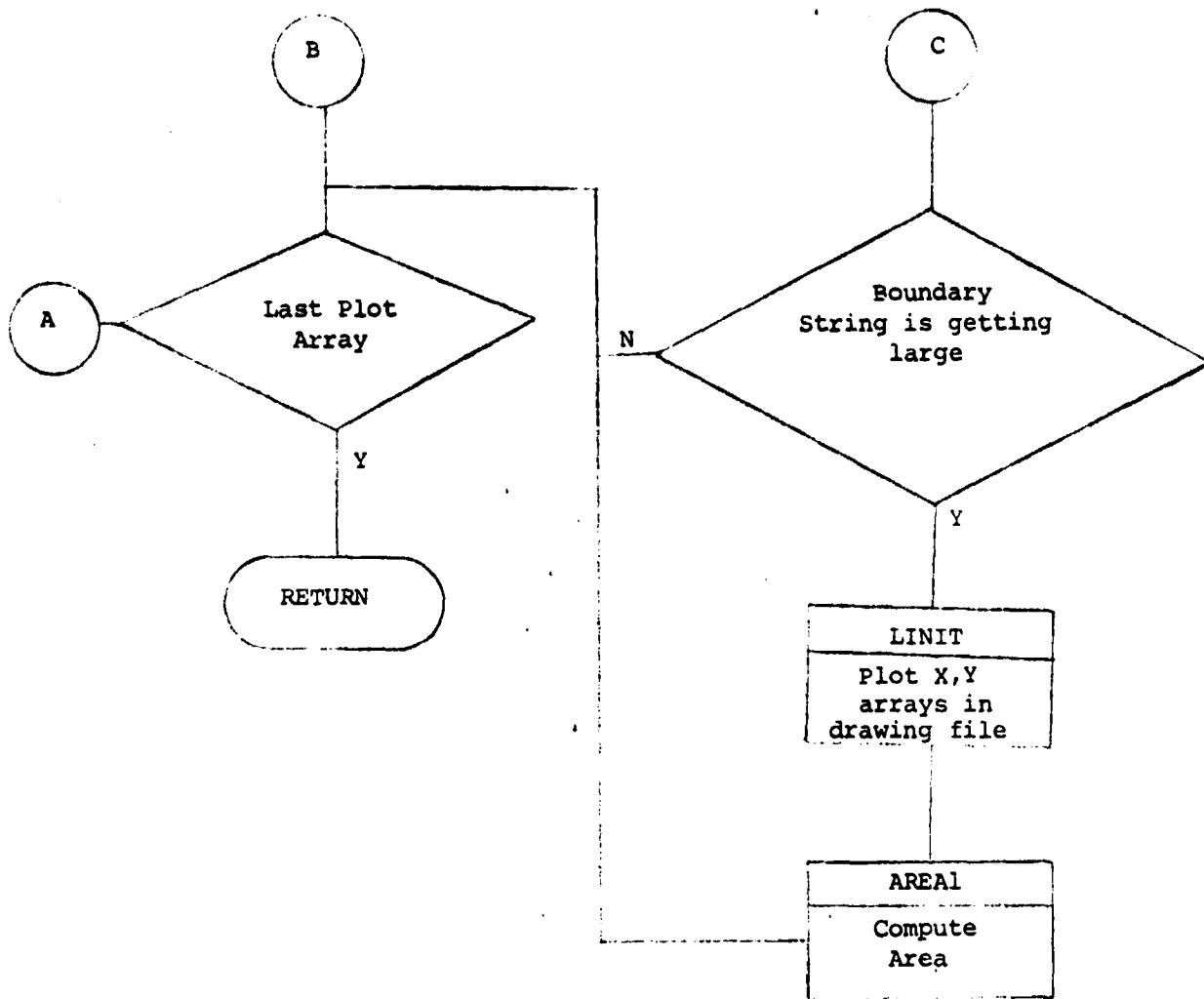
#### 3.2.16.7 Flowchart

#### 3.2.16.8 Listing

(CONALL)



~~3-27~~  
78



ORIGINAL FILED  
 1978 OCT 11 AM 10:00  
 FEDERAL BUREAU OF INVESTIGATION  
 U.S. DEPARTMENT OF JUSTICE

~~3-78~~  
 79

READY

```
SUBROUTINE CONALL(IPROB)
COMMON /Z/ NGRUP,ARRAYX(50,50),ARRAYY(50,50),ISIZE(50),ASIZE(50)
,ARX(50),ARY(50),LINE,YMAX,XSC,YSC,KAPPA
INTEGER ARRAYX,ARRAYY
DO 200 J=1,NGRUP
IS=ISIZE(J)
IF(IS-1) 200,200,10
10 DO 100 I=1,NGRUP
IF(I.EQ.J) GO TO 100
IS2=ISIZE(I)
IF(IS2-1) 100,100,20
20 IND=1
XA=ARRAYX(J,1)
YA=ARRAYY(J,1)
XB=ARRAYX(I,1)
YB=ARRAYY(I,1)
22 DIF1=XA-XB
IF(ABS(DIF1).GT.0.01) GO TO 25
DIF2=YA-YB
IF(ABS(DIF2).GT.0.01) GO TO 25
IPROB=1
JA=J
ISA=IS
IA=I
IS2A=IS2
CALL JOINK(JA,ISA,IA,IS2A,IND)
IS=ISIZE(JA)
J=JA
IF(IS.LT.24) GO TO 24
DO 23 IJ=1,IS
ARX(IJ)=ARRAYX(J,IJ) * XSC
23 ARY(IJ)=(YMAX-ARRAYY(J,IJ)) * YSC
CALL LINIT(ARX,ARY,IS,0)
```

ORIGINAL PAGE IS  
OF POOR QUALITY

```

CALL AREA1(J,AREA)
ASIZE(J)=ASIZE(J) + AREA
IF (ARRAYY(J,1).LT.LINE) GO TO 235
ASIZE(IA)=0
ARRAYX(IA,1)=ARRAYX(J,1)
ARRAYY(IA,1)=ARRAYY(J,1)
ISIZE(IA)=1
235 ISIZE(J)=1
    ARRAYX(J,1)=ARRAYX(J,IS)
    ARRAYY(J,1)=ARRAYY(J,IS)
24 GO TO 200
25 IND=IND+1
    GO TO (20,30,40,50,100),IND
30 XB=ARRAYX(I,IS2)
    YB=ARRAYY(I,IS2)
    GO TO 22
40 XA=ARRAYX(J,IS)
    YA=ARRAYY(J,IS)
    GO TO 22
50 XB=ARRAYX(I,1)
    YB=ARRAYY(I,1)
100 CONTINUE
200 CONTINUE
    RETURN
    END

```

ORIGINAL PAGE IS  
OF POOR QUALITY

READY

3.2.17 SOFTWARE COMPONENT NO. 17 (JOIN)

3.2.17.1 Linkage

Subroutine JOIN is called by subroutines CONECT and CONALL.

3.2.17.2 Interface

Subroutine JOIN receives control information through common block Z (see Appendix A).

3.2.17.3 Input

None

3.2.17.4 Output

None

3.2.17.5 Storage Requirements

Subroutine JOIN requires 358 words in core.

3.2.17.6 Description

Subroutine JOIN connects plot string arrays as determined by subroutines CONALL and CONECT. Arrays which are no longer needed, i.e., whose coordinates have been linked to another array, are flagged for reuse.

3.2.17.7 Flowchart

3.2.17.8 Listing

(JOIN)

Start

Join Designated  
Arrays Into  
One Array

Reset Second  
Array Area  
For Reuse

Return

~~3-82~~

83

READY

```
SUBROUTINE JOIN(J, IS, I, IS2, IND)
DIMENSION ATX(50), ATY(50)
COMMON /Z/ NGRUP, ARRAYX(50,50), ARRAYY(50,50), ISIZE(50), ASIZE(50)
, ARX(50), ARY(50), LINE, YMAX, XSC, YSC, KAPPA
INTEGER ARRAYX, ARRAYY
IF(IND.NE.2) GO TO 1
IT=J
J=I
I=IT
IT=IS
IS=IS2
IS2=IT
1 GO TO (10,30,20,30), IND
10 ISM=IS+1
DO 12 II=1, IS
ISM=ISM-1
ATX(ISM)=ARRAYX(J, II)
12 ATY(ISM)=ARRAYY(J, II)
DO 14 II=1, IS
ARRAYX(J, II)=ATX(II)
14 ARRAYY(J, II)=ATY(II)
GO TO 30
20 ISM=IS2+1
DO 22 II=1, IS2
ISM=ISM-1
ATX(ISM)=ARRAYX(I, II)
22 ATY(ISM)=ARRAYY(I, II)
DO 24 II=1, IS2
ARRAYX(I, II)=ATX(II)
24 ARRAYY(I, II)=ATY(II)
30 IJ=0
NEWE=IS+IS2-1
DO 50 II=IS, NEWE
```



```
IJ=IJ+1  
ARRAYX(J,I)=ARRAYX(I,IJ)  
50 ARRAYY(J,I)=ARRAYY(I,IJ)  
ISIZE(J)=NEW  
ISIZE(I)=0  
RETURN  
END
```

READY

### 3.2.18 SOFTWARE COMPONENT NO. 18 (CLSTST)

#### 3.2.18.1 Linkage

Subroutine CLSTST is called by subroutine BDT3, and calls subroutines AREAL and LINIT.

#### 3.2.18.2 Interface

Control information and data are communicated to subroutine CLSTST via common block Z (see Appendix A).

#### 3.2.18.3 Input

None

#### 3.2.18.4 Output

None

#### 3.2.18.5 Storage Requirements

Subroutine CLSTST requires 324 words in core.

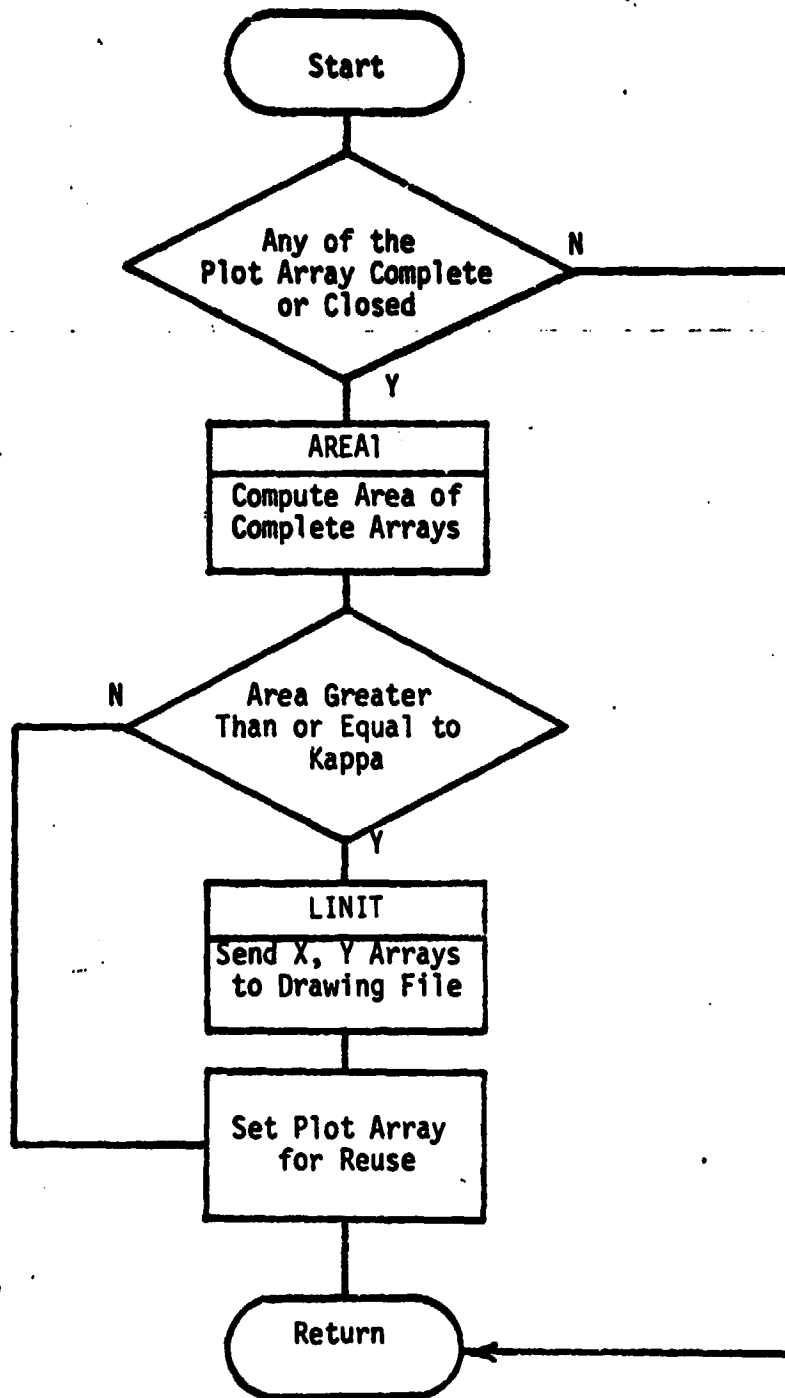
#### 3.2.18.6 Description

Subroutine CLSTST accepts as input plot string arrays and determines whether these strings are complete, or "closed". Arrays which are complete are sent to subroutine AREAL for area computation, and upon returning, are plotted if the area is  $\geq$  Kappa, a user-supplied constant.

#### 3.2.18.7 Flowchart

#### 3.2.18.8 Listing

(CLSTST)



~~3-86~~  
87

```

SUBROUTINE CLTST
COMMON /Z/ NGRUP,ARRAYX(50,50),ARRAYY(50,50),ISIZE(50),ASIZE(50)
,ARX(50),ARY(50),LINE,YMAX,XSC,YSC,KAPPA
INTEGER ARRAYX,ARRAYY
IF(NGRUP) 999,999,10
10 DO 100 I=1,NGRUP
A1=ASIZE(I)
IF(ABS(A1).GT.0.1) GO TO 100
I1=ISIZE(I)
IF(I1.LT.2) GO TO 100
DIF1=ARRAYX(I,1)-ARRAYX(I,I1)
IF(ABS(DIF1).GT.0.01) GO TO 100
DIF2=ARRAYY(I,1)-ARRAYY(I,I1)
IF(ABS(DIF2).GT.0.01) GO TO 100
CALL AREA1(I,AREA)
IF(ABS(AREA).LT.KAPPA) GO TO 90
DO 50 JK=1,I1
ARX(JK)=ARRAYX(I,JK) * XSC
50 ARY(JK)=(YMAX-ARRAYY(I,JK)) * YSC
CALL LINIT(ARX,ARY,I1,0)
ASIZE(I)=ASIZE(I) + AREA
LX=ARRAYX(I,1)
LY=ARRAYY(I,1)
C WRITE(10,60)LY,LX,ASIZE(I)
C 60 FORMAT(' AREA ',I3,' X ',I3,' )=',F8.2)
ASIZE(I)=0
90 ISIZE(I)=0
100 CONTINUE
999 RETURN
END

```

READY

### 3.2.19 SOFTWARE COMPONENT NO. 19 (AREAL)

#### 3.2.19.1 Linkage

Subroutine AREAL is called by subroutines CONECT, CONALL, FINDAR, CLSTST, and ENDTST.

#### 3.2.19.2 Interface

Control information and data information are communicated by means of common block Z (see Appendix A).

#### 3.2.19.3 Input

A plot string index is input to AREAL via a calling argument.

#### 3.2.19.4 Output

An area value is output via a calling argument.

#### 3.2.19.5 Storage Requirements

Subroutine AREAL requires 122 words in core.

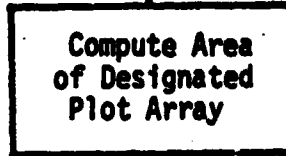
#### 3.2.19.6 Description

Subroutine AREAL accepts as input a plot string array, either partial or complete. AREAL computes the area or partial area in pixel units that this array represents.

#### 3.2.19.7 Flowchart

#### 3.2.19.8 Listing

(AREA1)



~~3-89~~  
90

```
SUBROUTINE AREA1(I, AREA)
COMMON /Z/ NGRUP, ARRAYX(50, 50), ARRAYY(50, 50), ISIZE(50), ASIZE(50)
, ARX(50), ARY(50), LINE, YMAX, XSC, YSC, KAPPA
INTEGER ARRAYX, ARRAYY
COMPUTE AREA USING ARRAYX(I, ALL), ARRAYY(I, ALL)
I1=ISIZE(I)
AREA=0.
DO 100 J=2, I1
DX=ARRAYX(I, J)-ARRAYX(I, J-1)
AREA=AREA + DX* ARRAYY(I, J)
100 CONTINUE
RETURN
END
```

READY

3.2.20 SOFTWARE COMPONENT NO. 20 (ENDTST)

3.2.20.1 Linkage

Subroutine ENDTST is called by subroutine BDT3, and calls subroutines AREAL and LINIT.

3.2.20.2 Interface

Subroutine ENDTST receives control information through common block Z (see Appendix A).

3.2.20.3 Input

None

3.2.20.4 Output

None

3.2.20.5 Storage Requirements

Subroutine ENDTST requires 723 words in core.

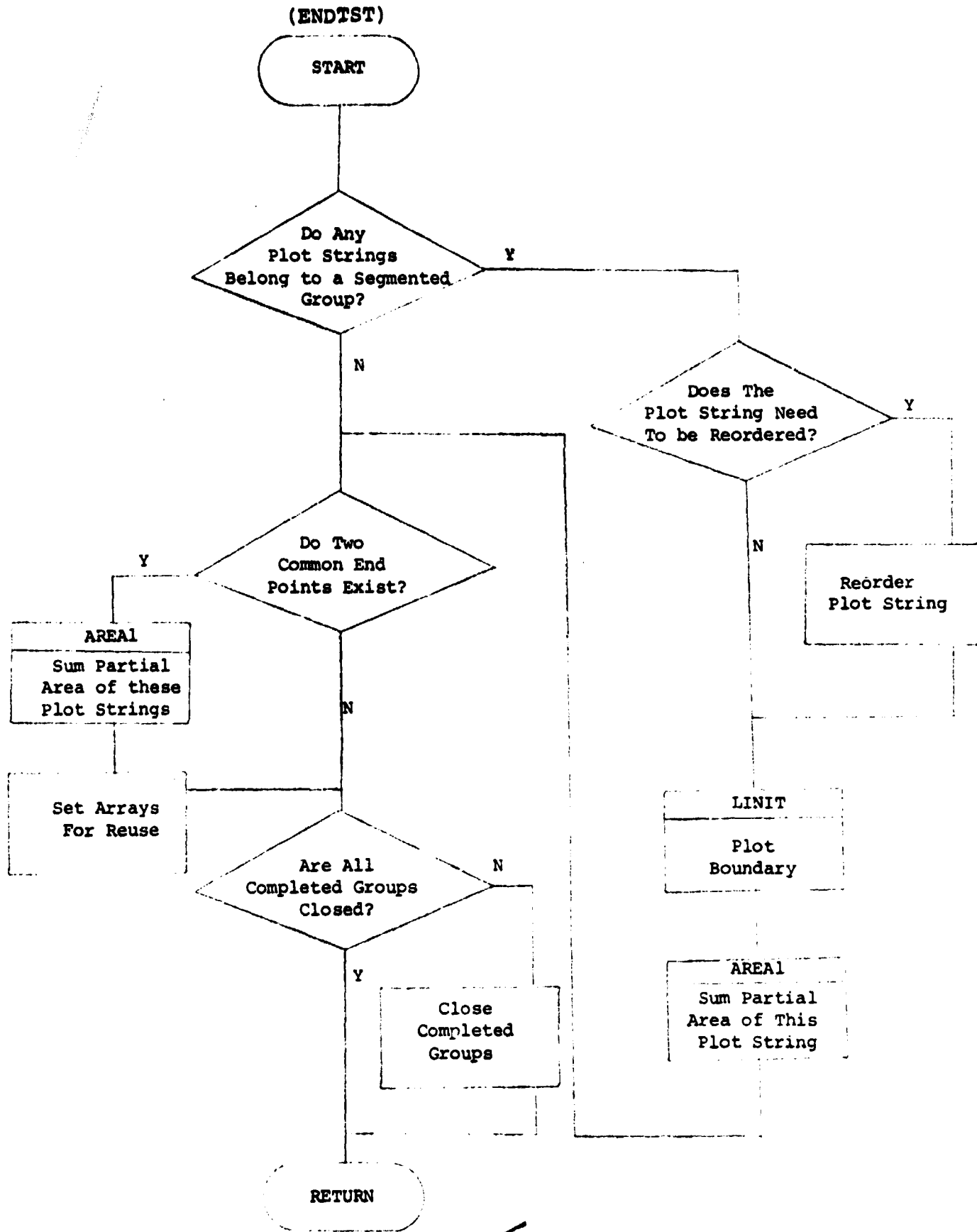
3.2.20.6 Description

Subroutine ENDTST handles, by segmentation, plotting and summation of area measurement for large plot strings which cannot be stored contiguously.

3.2.20.7 Flowchart

3.2.20.8 Listing





~~3-92~~  
93

READY

```
SUBROUTINE ENDTST
DIMENSION LRX(50), LRY(50)
COMMON /2/ NGRUP, ARRAYX(50,50), ARRAYY(50,50), ISIZE(50), ASIZE(50)
, ARX(50), ARY(50), LINE, YMAX, XSC, YSC, KAPPA
INTEGER ARRAYX, ARRAYY
IF(NGRUP) 99,99,1
1 DO 10 I=1,NGRUP
  I1=ISIZE(I)
  IF(I1-1) 10,2,10
2 DO 9 J=1,NGRUP
  I2=ISIZE(J)
  IF(I2-1) 9,9,3
3 N=1
  AX1=ARRAYX(I,1)
  AY1=ARRAYY(I,1)
4 AX2=ARRAYX(J,N)
  AY2=ARRAYY(J,N)
  IF(ABS(AX1-AX2).GT.0.01) GO TO 5
  IF(ABS(AY1-AY2).GT.0.01) GO TO 5
  IF(N-1) 8,8,7
5 IF(N-I2) 6,9,6
6 N=I2
  GO TO 4
7 L=I2+1
  DO 71 LL=1,I2
  L=L-1
  LRX(LL)=ARRAYX(J,L)
71 LRY(LL)=ARRAYY(J,L)
  DO 72 LL=1,I2
  ARRAYX(J,LL)=LRX(LL)
72 ARRAYY(J,LL)=LRY(LL)
8 DO 88 L=1,I2
  ARX(L)=ARRAYX(J,L) * XSC
```

~~3-93~~  
94

```

                                READY
88  ARY(L)=(YMAX-ARRAYY(J,L))*YSC
    CALL LINIT(ARX,ARY,I2,0)
    CALL AREA1(J,AREA)
    ASIZE(I)=ASIZE(I)+AREA
    ISIZE(I)=1
    ARRAYX(I,1)=ARRAYX(J,I2)
    ARRAYY(I,1)=ARRAYY(J,I2)
    ISIZE(J)=0
    ASIZE(J)=0
    GO TO 10
9   CONTINUE
10  CONTINUE
    DO 20 I=1,NGRUP
      I1=ISIZE(I)
      IF(I1-1) 20,11,20
11  DO 19 J=1,NGRUP
      IF(I-J) 12,19,12
12  I2=ISIZE(J)
      IF(I2-1) 19,13,19
13  AX1=ARRAYX(I,1)
      AY1=ARRAYY(I,1)
      AX2=ARRAYX(J,1)
      AY2=ARRAYY(J,1)
      IF(ABS(AX1-AX2).GT.0.01) GO TO 19
      IF(ABS(AY1-AY2).GT.0.01) GO TO 19
      LX=AX1
      LY=AY1
      AREA=ASIZE(I)-ASIZE(J)
C   WRITE(10,14)LY,LX,AREA
C 14  FORMAT(" AREA(",I3," , ",I3," )=",F8.2)
      ASIZE(I)=0
      ASIZE(J)=0
      ISIZE(I)=0

```

```
      ISIZE(J)=0
19  CONTINUE
20  CONTINUE
      DO 30 J=1,NGRUP
      J1=ISIZE(J)
      IF(J1-1)30,25,30
25  LX=ARRAYX(J,1)
      LY=ARRAYY(J,1)
      IF(LY.GE.LINE) GO TO 30
      ASIZE(J)=0
      ISIZE(J)=0
30  CONTINUE
99  RETURN
      END
```

READY

#### 4. OPERATION

The users of this software system are researchers and analysts who need a method of comparing classification results to ground truth and an accurate means of production display of classification results. The input to this software system is a 7-track, 800 BPI universally formatted classification data tape directly or indirectly obtained from the GE Interactive Multispectral Image Analyst System (IMAGE 100), the Earth Resources Interactive Processing System (ERIPS), and the UNIVAC 1100 Software (EOD-LARSYS). If any of the above-mentioned systems does not support 7-track, 800 BPI output tape, the user may use the conversion capability available in Bldg. 12 to meet this requirement.

##### 4.1 USER DOCUMENTATION

There is no formal user's document required in this phase implementation; the function of such a document is satisfied by the Technical Memorandum entitled "Software Specifications for Automated Thematic Plotting of Classified Digital Data", April 1976 (LEC 8289).

##### 4.2 OPERATION DOCUMENTATION

N/A

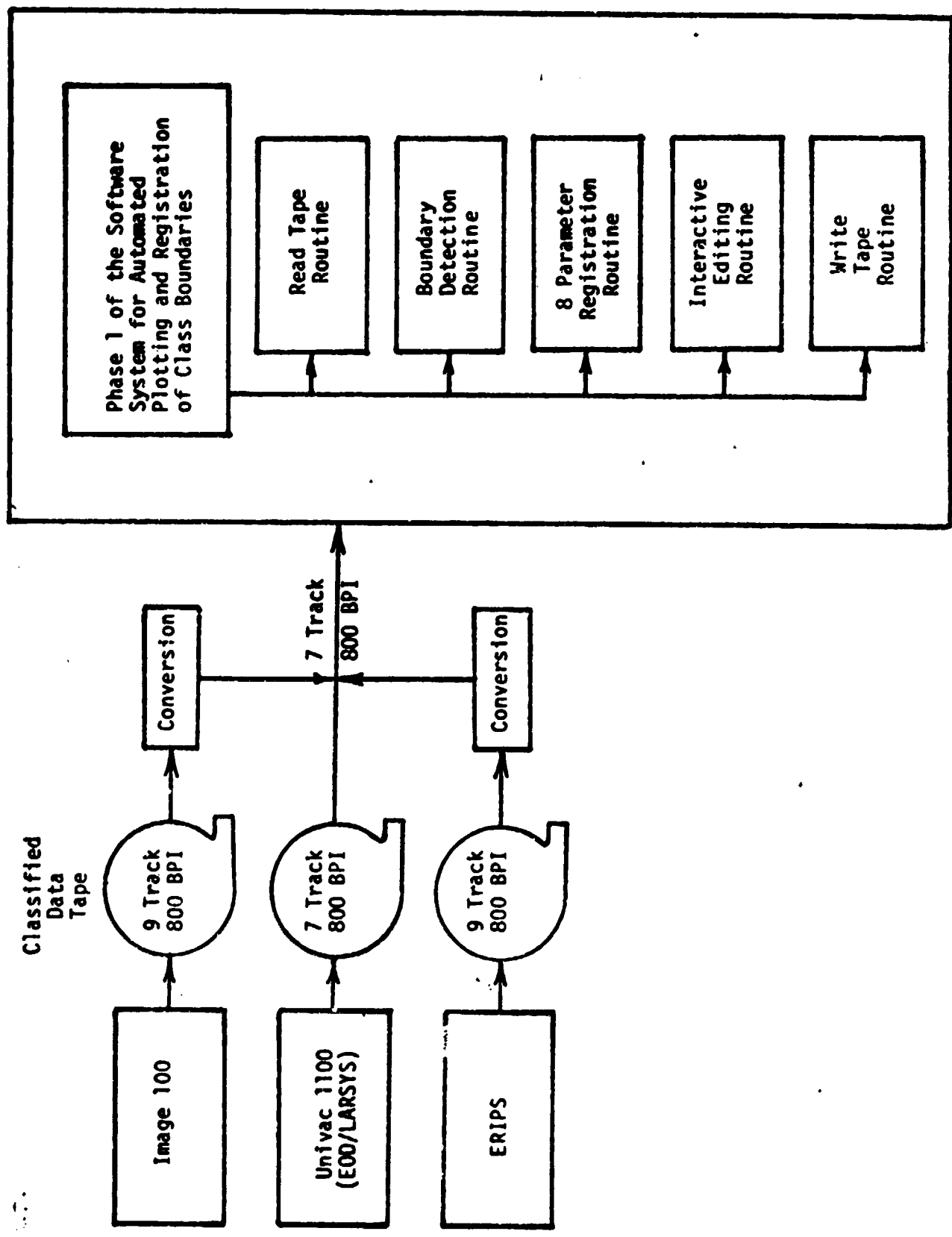


Figure 1: Functional Diagram of Phase 1 Implementation of the Software System

82

**STARTING  
OCTAL  
NUMBER**

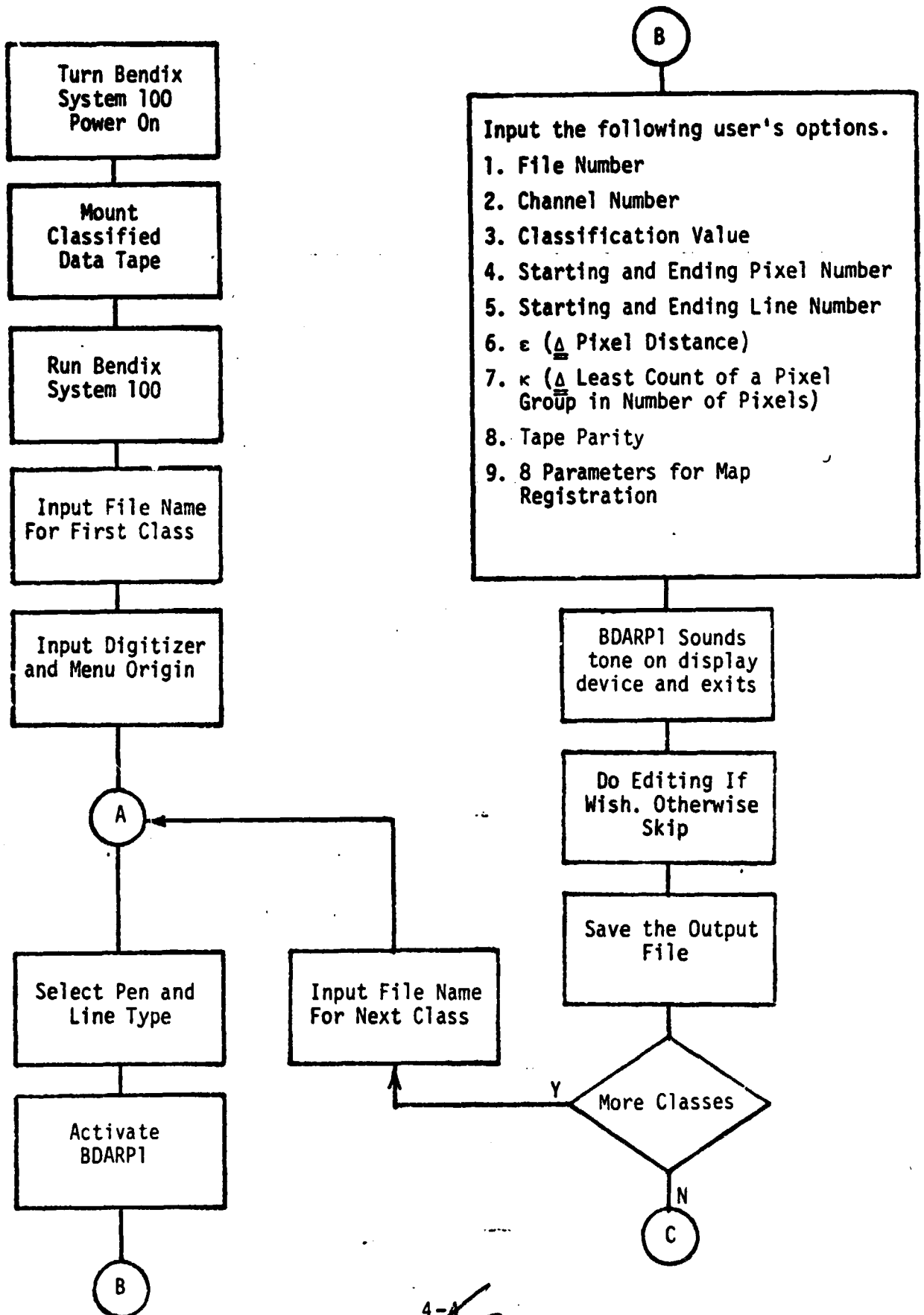
**CORE MAP (32K)**

0	Page 0 and Constants
440	1. Bendix System 100 labeled common 2. External references for Bendix System 100 in-core routines and plotter routines
3330	Fortran initialization routine - 1st routine executed by each overlay
4007	Fortran run-time linkage
4200	Fortran libraries
13651	Part 1 of Bendix System 100 subroutines
14234	Menu
16644	Part 2 of Bendix System 100 subroutines
30641	User's overlay
64000	Run time stack for main program
70704	1. Monitor 2. System loader 3. Paper tape loader 4. Key-in loader

} 13,919<sub>10</sub>

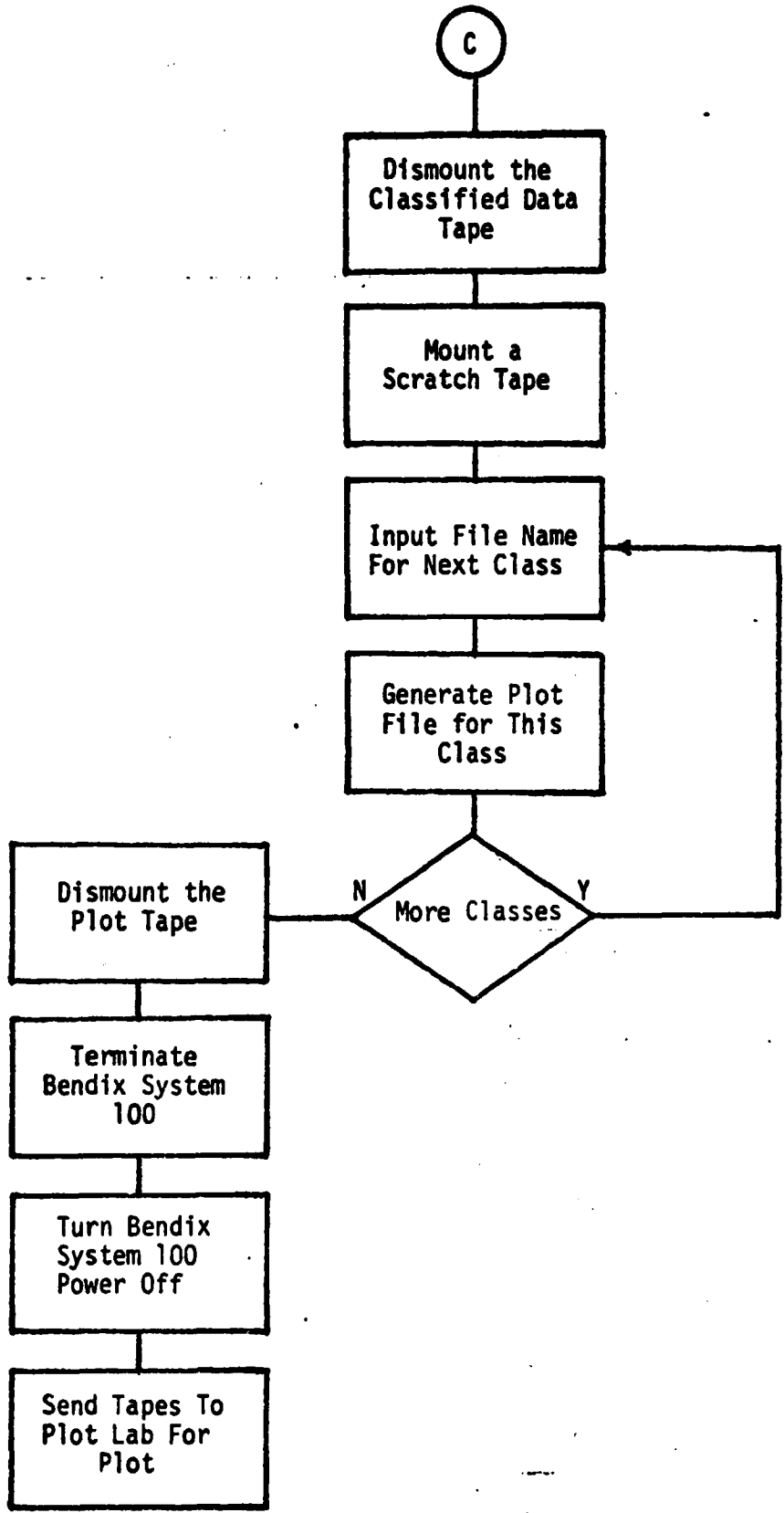
Figure 2: Bendix System 100 Core Utilization Map

4-5  
99



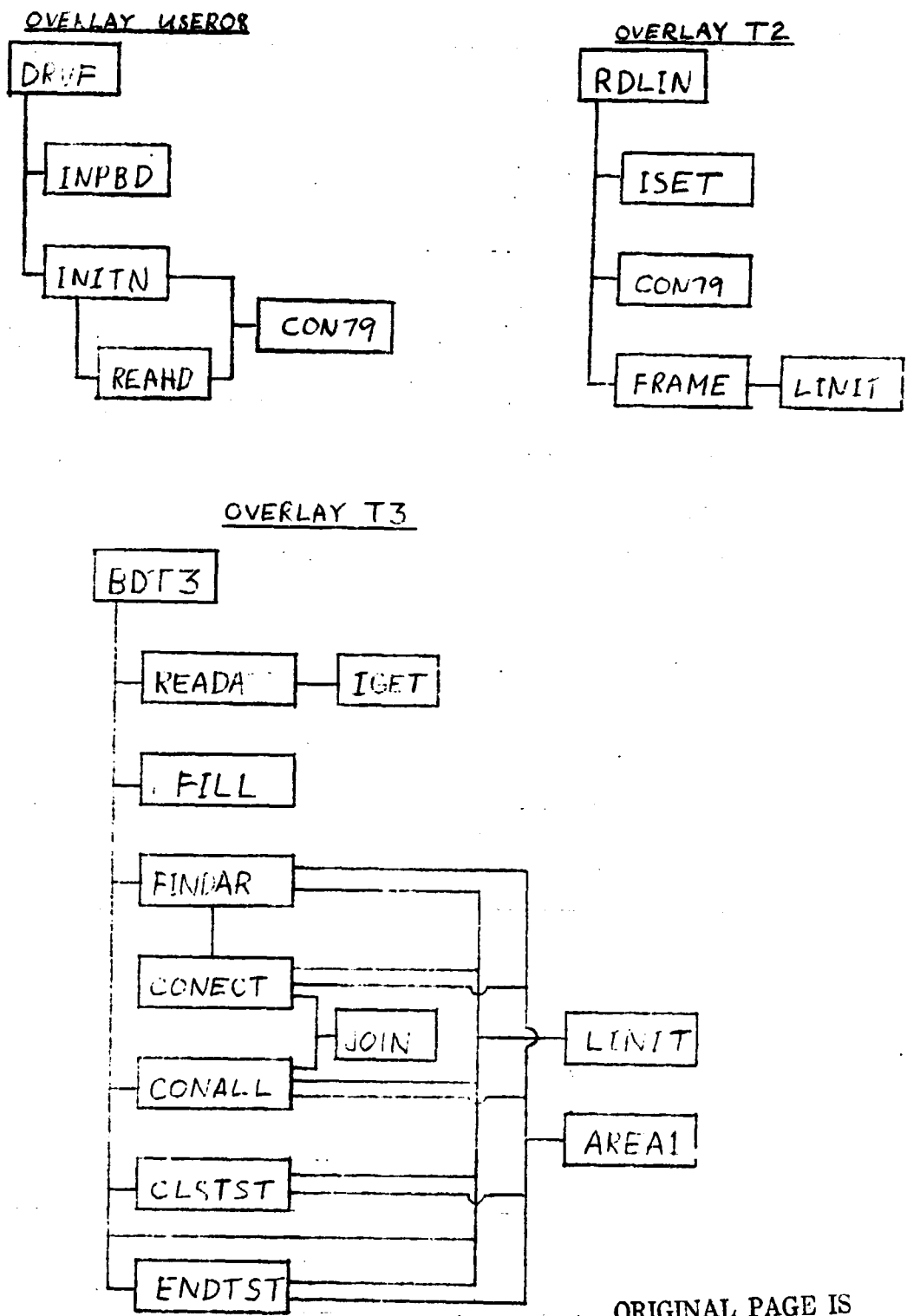
4-4  
110  
Figure 3: User's Procedure





4-8  
101

Figure 3: User's Procedure (continued)



ORIGINAL PAGE IS OF POOR QUALITY

FIGURE 4. The Functional Block Diagram

4/6/02

APPENDIX A  
BDARPI COMMON TABLE

BDARPI COMMON TABLE

No.	Title	Common Block	Initial Value	Initialized by	Referenced by	Modified by
1.	Header Record constants	ICONS	INPUT	REAH	REAH INITN RDLIN	REAH INITN
2.	Tape File Number	ICONS	INPUT	INPBD	REAH, INITN	--
3.	Tape Parity	ICONS	INPUT	INPBD	REAH, INITN	--
4.	Channel No.	ICONS	INPUT	INPBD	REAH, INITN	--
5.	Class Value	ICONS	INPUT	INPBD	RDLIN	--
6.	Starting/Ending Line	ICONS	INPUT	INPBD	RDLIN, BDT3	--
7.	Starting/Ending Pixel	ICONS	INPUT	INPBD	RDLIN, BDT3	--
8.	Epsilon Value	ICONS, ZZ	INPUT	INPBD	BDT3, FILL	--
9.	Kappa Value	ICONS, Z	INPUT	INPBD	CLSTST	--
10.	8 Registration coefficients	ICONS	INPUT	INPBD	LINIT	--
11.	Index on FINDAR failures due to all plot arrays being filled	MAXFIL	0	BDT3	BDT3, FINDAR	FINDAR
12.	No. of plot arrays in use	Z	0	BDT3	FINDAR	FINDAR
13.	X and Y arrays of current boundary plot strings	Z	0	FINDAR	*	*

A-2  
104

BDARP1 COMMON TABLE (cont)

No.	Title	Common Block	Initial Value	Initialized by	Referenced by	Modified by
14.	Vector de- scribing length of each boundary string	Z	0	FINDAR	*	*
15.	Vector de- scribing area of each boundary string	Z	0	FINDAR	*	*, AREAL
16.	X and Y array of plot string in drawing file format	Z	0	*	*	*
17.	Present line number	Z	1	BDT3	BDT3, FILL	BDT3
18.	Pixel scaling factors, X&Y	Z	0.1, 0.1	BDT3	*	--
19.	Block of "pixel" data, unchanged	ZZ	INPUT	BDT3	BDT, FILL	--
20.	Block of "pixel" data after fill	ZZ	COMPUTED	FILL	BDT3, FILL	FILL
21.	No. of pixels/line	ZZ	COMPUTED	BDT3	BDT3, FILL	--

\* denotes most of the following: BDT3, ENDTST, CLSTST, CONECT, CONALL, JOIN, and FINDAR.

~~A-55~~  
105