2mup

9Q1

2

IMAGE DISPLAY AND MANIPULATION SYSTEM

(IDAMS)

PROGRAM DOCUMENTATION

APPENDIXES A-D

(NASA-CR-132916) IMAGE DISPLAY AND MANIPULATION SYSTEM (IDAMS) PROGRAM DOCUMENTATION, APPENDIXES A-D (Computer Sciences Corp.) ~~385~~ 462 p HC $22.25 CSCL 09B

N74-17915

Unclas

G3/08 31544

SEPTEMBER 1972

CSC

COMPUTER SCIENCES CORPORATION

5006-04000-02TR

IMAGE DISPLAY AND MANIPULATION SYSTEM

(IDAMS)

PROGRAM DOCUMENTATION

Prepared by

COMPUTER SCIENCES CORPORATION

For

GODDARD SPACE FLIGHT CENTER

Prepared for

Lottie E. Brown
Code 563

Under

Contract No. NAS 5-11723
Task Assignment No. 040

Prepared by: Approved by:

R.W. Cecil	August 30, 1972	M. D. Vogel	August 30, 1972
R. W. Cecil	Date	M. D. Vogel	Date
R. A. White	30 Aug. 1972		
R. A. White	Date		
M R Szczur	August 30, 1972		
M. R. Szczur	Date		

Record of Revisions

Revision Number	Date	Revisions
Original	September 1972	
Revision 1	January 1973	ii, iii, v, vi, 2-2, 4-1, 4-4, 4-7, 4-9 4-14, 4-19, 4-20, 4-26, 4-31, 4-35, 4-45, 4-49, 4-52, 4-60, 4-86, 4-96, 4-97, 4-98, 4-104, 4-144, 4-147, 4-148, 4-149, 4-151, 4-152, 4-153, 4-158, 4-159, 4-160, 4-172, 4-174, 4-176, 4-177, 4-178, 4-179, C-79, and C-80.
Revision 2	December 1973	iii, vi, 2-2, 4-65, 4-65.1 through 4-65.18, 4-180 through 4-209, B-20, C-41, C-41.1 through C-41.7, C-81 through C-95

TABLE OF CONTENTS

Page

Section 1 - IDAMS System

1.1 Scope and Purpose 1-1
1.2 Problem Definition 1-1
1.3 Problem Analysis 1-3
1.4 Program Summary 1-4
1.5 Program Logic and Flow 1-4

Section 2 - Executive Program Description

2.1 Introduction 2-1
2.2 DRIVER-IDAMS Processor Executive Program 2-1

Section 3 - General Purpose Subroutine Descriptions

3.1 LBLRD - Tape Label Reading Subroutine 3-1
3.2 LBLWRT - Tape Label Writing Subroutine 3-3
3.3 IDAMSDSK - Disk Input/Output Subroutine 3-5
3.4 READRITE - Tape Input/Output Subroutine 3-9
3.5 UTMCON - UTM/Geographic Coordinate Interconversion Subroutine 3-11
3.6 TWOFIT - Two-Dimensional Least-Squares Fit Subroutine . . 3-16
3.7 MATINV - Matrix Inversion Subroutine 3-20
3.8 PERGEN - Inverted Bit-Order Permutation Generating Subroutine 3-23
3.9 TRIGGN - Sine Table Generating Subroutine 3-25
3.10 FFTONE - One-Dimensional Fast Fourier Transform Subroutine 3-26
3.11 CODE - Character Translation Subroutine 3-28
3.12 MOVE - Character Moving Subroutine 3-30
3.13 CODE8TO6 - Byte to Character Conversions Subroutine 3-32
3.14 ADDLINE - Message Array Update Subroutine 3-34
3.15 TTWLVE - 212 Display Station Interface Subroutine 3-36

Section 4 - Task Program Descriptions

4.1 BATCH - Task Entry Program 4-1
4.2 TESTGN - Test Pattern Generation Program 4-4
4.3 LIST - Tape to Printer Utility Program 4-7
4.4 CONTRAST - Radiometric Correction Program 4-9
4.5 CONVOLVE - Convolution Program 4-14
4.6 EXPAND - Image Expansion Program 4-20
4.7 SHADE - Photometric Correction Program 4-26

TABLE OF CONTENTS (Cont'd)

		Page
4.8	FFT - Fast Fourier Transform Program	4-31
4.9	FPCON - Floating-Point Conversion Program	4-35
4.10	SMOOTH - Floating-Point Smoothing Program	4-45
4.11	CXPACK - Complex Packing and Unpacking Program	4-49
4.12	ERROR - Message Processing Program	4-52
4.13	REDUCE - Image Reduction Program	4-54
4.14	HISTO - Histogram and Statistics Program	4-56
4.15	CHAROUT - Pixel Character Output Program	4-58
4.16	TEXTGN - Text Generator Program	4-60
4.17	NEIGHBOR - Nearest Neighbor Printer Listing Program . .	4-63
4.18	DISPLAY - Interactive Display Program	4-65
4.19	MODIFY - Image Editing Program	4-66
4.20	INSERT - Window Insertion and Mosaicking Program	4-70
4.21	GRID - Grid Overlay Program	4-73
4.22	GEOMTRAN - Geometric Transformation Program	4-75
4.23	CHIPGN - Reference Chip Generation and Update Program	4-87
4.24	RZOMAP - Reseau Mapping Program	4-94
4.25	CORREL - Image Correlation Program	4-99
4.26	RESECT - Spatial Resection Program	4-104
4.27	UTMGEO - UTM/Geographic Coordinate Conversion Program	4-125
4.28	FPMULT - Floating-Point Array Multiplication Program . .	4-128
4.29	FPSUM - Floating-Point Summation Program	4-131
4.30	FILTGN - Digital Filter Generating Program	4-134
4.31	RANDGRAY - Random Gray Level Generation Program . .	4-142
4.32	IMERGE - Bulk ERTS Tape Merging Program	4-144
4.33	PMERGE - Precision ERTS Tape Merging Program	4-150
4.34	PPUPDATE - Precision Processing Disk File Update Program	4-155
4.35	VPICIN - VICAR Image Reformatting Program	4-164
4.36	INCREASE - Image Enlarging Program	4-166
4.37	COLOR - False Color Coding Program	4-168
4.38	FPLIST - Floating-Point Listing Utility Program	4-172
4.39	DMDOUT - Digital Muirhead Reformatting Program	4-174
4.40	ADDPIX - Picture Addition Program	4-176
4.41	FORMAT - IDAMS Format Conversion Program	4-178
4.42	HISTCONT - Histogram-Contrast Correction Program . . .	4-180
4.43	JOYSTICK - Interactive Display Program	4-183
4.44	MSSCON - Special-Purpose Convolution Routine	4-205

TABLE OF CONTENTS (Cont'd)

Appendix A - Executive Program Flowcharts

Appendix B - General Purpose Subroutine Flowcharts

Appendix C - Task Program Flowcharts

Appendix D - Floating-Point Data Representation

LIST OF ILLUSTRATIONS

Figure

1-1 IDAMS Processor General Flow 1-5
4-1 Resolution Test Target 4-5
4-2 Contrast Conversions 4-12
4-3 TEXTGN Characters 4-61

LIST OF TABLES

Table

2-1 Task Program Overlay Assignments 2-2
4-1 FPCON Conversion and Transfers 4-37
4-2 FPCON Input Data 4-41
4-3 FPCON Output Data 4-42

LIST OF ILLUSTRATIONS (Cont'd)

Figure		Page
C-22	GEOMTRAN Program Flowchart	C-45
C-23	CHIPGN Program Flowchart	C-52
C-24	RZOMAP Program Flowchart	C-53
C-25	CORREL Program Flowchart	C-56
C-26	RESECT Program Flowchart	C-58
C-27	UTMGEO Program Flowchart	C-60
C-28	FPMULT Program Flowchart	C-61
C-29	FPSUM Program Flowchart	C-62
C-30	FILTGN Program Flowchart	C-63
C-31	RANDGRAY Program Flowchart	C-65
C-32	IMERGE Program Flowchart	C-67
C-33	PMERGE Program Flowchart	C-68
C-34	PPUPDATE Program Flowchart	C-72
C-35	VPICIN Program Flowchart	C-73
C-36	INCREASE Program Flowchart	C-74
C-37	COLOR Program Flowchart	C-76
C-38	FPLIST Program Flowchart	C-77
C-39	DMDOUT Program Flowchart	C-78
C-40	ADDPIX Program Flowchart	C-79
C-41	FORMAT Program Flowchart	C-80
C-42	HISCONT Program Flowchart	C-81
C-43	JOYSTICK Program Flowchart	C-83
C-44	MSSCON Program Flowchart	C-94

LIST OF TABLES

Table		Page
2-1	Task Program Overlay Assignments	2-2
4-1	FPCON Conversion and Transfers	4-37
4-2	FPCON Input Data	4-41
4-3	FPCON Output Data	4-42

LIST OF ILLUSTRATIONS

Figure		Page
1-1	IDAMS Processor General Flow	1-5
4-1	Resolution Test Target	4-5
4-2	Contrast Conversions	4-12
4-3	TEXTGN Characters	4-61
A-1	DRIVER Program Flowchart	A-1
B-1	Subroutine LBLRD Flowchart	B-1
B-2	Subroutine LBLWRT Flowchart	B-2
B-3	Subroutine IDAMSDSK Flowchart	B-3
B-4	Subroutine READRITE Flowchart	B-6
B-5	Subroutine UTMCON Flowchart	B-7
B-6	Subroutine TWOFIT Flowchart	B-8
B-7	Subroutine MATINV Flowchart	B-9
B-8	Subroutine PERGEN Flowchart	B-10
B-9	Subroutine TRIGGN Flowchart	B-11
B-10	Subroutine FFTONE Flowchart	B-12
B-11	Subroutine CODE Flowchart	B-15
B-12	Subroutine MOVE Flowchart	B-16
B-13	Subroutine CODEST06 Flowchart	B-17
B-14	Subroutine ADDLINE Flowchart	B-18
B-15	Subroutine TTWLVE Flowchart	B-19
C-1	BATCH Program Flowchart	C-1
C-2	TESTGN Program Flowchart	C-2
C-3	LIST Program Flowchart	C-3
C-4	CONTRAST Program Flowchart	C-4
C-5	CONVOLVE Program Flowchart	C-6
C-6	EXPAND Program Flowchart	C-11
C-7	SHADE Program Flowchart	C-15
C-8	FFT Program Flowchart	C-17
C-9	FPCON Program Flowchart	C-21
C-10	SMOOTH Program Flowchart	C-28
C-11	CXPACK Program Flowchart	C-32
C-12	ERROR Program Flowchart	C-33
C-13	REDUCE Program Flowchart	C-35
C-14	HISTO Program Flowchart	C-37
C-15	CHAROUT Program Flowchart	C-38
C-16	TEXTGN Program Flowchart	C-39
C-17	NEIGHBOR Program Flowchart	C-40
C-18	DISPLAY Program Flowchart	C-41
C-19	MODIFY Program Flowchart	C-42
C-20	INSERT Program Flowchart	C-43
C-21	GRID Program Flowchart	C-44

LIST OF ILLUSTRATIONS (Cont'd)

Figure		Page
C-22	GEOMTRAN Program Flowchart	C-45
C-23	CHIPGN Program Flowchart	C-52
C-24	RZOMAP Program Flowchart	C-53
C-25	CORREL Program Flowchart	C-56
C-26	RESECT Program Flowchart	C-58
C-27	UTMGEO Program Flowchart	C-60
C-28	FPMULT Program Flowchart	C-61
C-29	FPSUM Program Flowchart	C-62
C-30	FILTGN Program Flowchart	C-63
C-31	RANDGRAY Program Flowchart	C-65
C-32	IMERGE Program Flowchart	C-67
C-33	PMERGE Program Flowchart	C-68
C-34	PPUPDATE Program Flowchart	C-72
C-35	VPICIN Program Flowchart	C-73
C-36	INCREASE Program Flowchart	C-74
C-37	COLOR Program Flowchart	C-76
C-38	FPLIST Program Flowchart	C-77
C-39	DMDOUT Program Flowchart	C-78
C-40	ADDPIX Program Flowchart	C-79
C-41	FORMAT Program Flowchart	C-80

LIST OF TABLES

Table		Page
2-1	Task Program Overlay Assignments	2-2
4-1	FPCON Conversion and Transfers	4-37
4-2	FPCON Input Data	4-41
4-3	FPCON Output Data	4-42

SECTION 1 - IDAMS SYSTEM DESCRIPTION

1.1 SCOPE AND PURPOSE

The Image Display and Manipulation System (IDAMS) Processor provides the user with an open-ended capability for performing a variety of operations on a large-scale digitized pictorial image (up to 5000 lines of 5000 picture elements per line). It consists of a modular package of analytical tools (task programs) that can be used to enhance, manipulate, and analyze pictorial information from a satellite or other photographic source. Specifically the IDAMS Processor provides support for the Earth Resources and Technology Satellite (ERTS) program.

1.2 PROBLEM DEFINITION

The IDAMS Processor was developed to satisfy a requirement for the following types of picture processing:

1. Image degradation removal and simulation
    a. Convolution filter, including weighted sampling
    b. Position-independent radiometric correction
    c. Complex array multiplication and addition
    d. Filter weight table generation
    e. Gray level noise generation
2. Discrete Fourier transform and its inverse
3. Data conversion and scaling
4. Autocorrelation to power spectral density and its inverse
5. Smoothing, including Hamming and Hanning, of frequency space image representations
6. Position-dependent photometric correction

7. Image display and description

    a. Image reduction and expansion (linear and nonlinear)
    b. Histogram and statistics generation
    c. Bit masking
    d. Printed output
    e. Text generation
    f. False color conversion

8. Compositing and editing

    a. Image point and line editing
    b. Window insertion and mosaicking
    c. Grid overlay
    d. Intensity stretching

9. Geometric correction

10. Precision processing

    a. ERTS bulk and precision image reformatting
    b. Reference chip generation and update
    c. Reseau mapping
    d. Cross correlation
    e. Spatial resection
    f. UTM/lat-long coordinate transformation
    g. Disk file editing

1.3 PROBLEM ANALYSIS

The IDAMS Processor is meant to be a flexible picture processing package with a capability for the addition of new features, as needs arise. For this reason, the overall design must adhere to the following considerations:

1. Organization of a modular task structure
2. Overall job control by an executive, with user's choice of sequence and types of image processing to be performed
3. Standard task and parameter cards for each task, with identical formats for all tasks to minimize effort in structuring a run
4. Design of functions that are identical for more than one task, such as tape label processing and moving data within core, as general-purpose subroutines that can be called by any task program
5. Provision of a general system support capability, such as tape listing and test pattern generation programs, to assist in test and evaluation of image processing tasks.

1.4 PROGRAM SUMMARY

The IDAMS Processor is a package of task routines and support software that performs convolution filtering, image expansion, Fast Fourier transformation, and other operations on a digital image tape. A unique task control card for that program, together with any necessary parameter cards, selects each processing technique to be applied to the input image. A variable number of tasks can be selected for execution by including the proper task and parameter cards in the input deck. An executive maintains control of the run; it initiates execution of each task in turn and handles any necessary error processing.

1.5 PROGRAM LOGIC AND FLOW

A small core resident program, DRIVER, directs the execution of all tasks within the system. It initially passes control to the batch processor, BATCH, which reads one task card and any parameter cards for that task. The program performs a preliminary edit on the data and places the information in COMMON. Control then returns to DRIVER, which calls the task selected for execution. To save core, BATCH, all task programs, and the error processing routine reside on an overlay tape. As the execution of each task terminates, control returns to the main driver program for initiation of the next task. Execution continues until all control cards have been processed or a fatal error occurs. Figure 1-1 is a general flow diagram for the system. It shows the maximum configuration of input and output peripherals operating with a task routine, but one or more of these devices may not be used by an individual task.

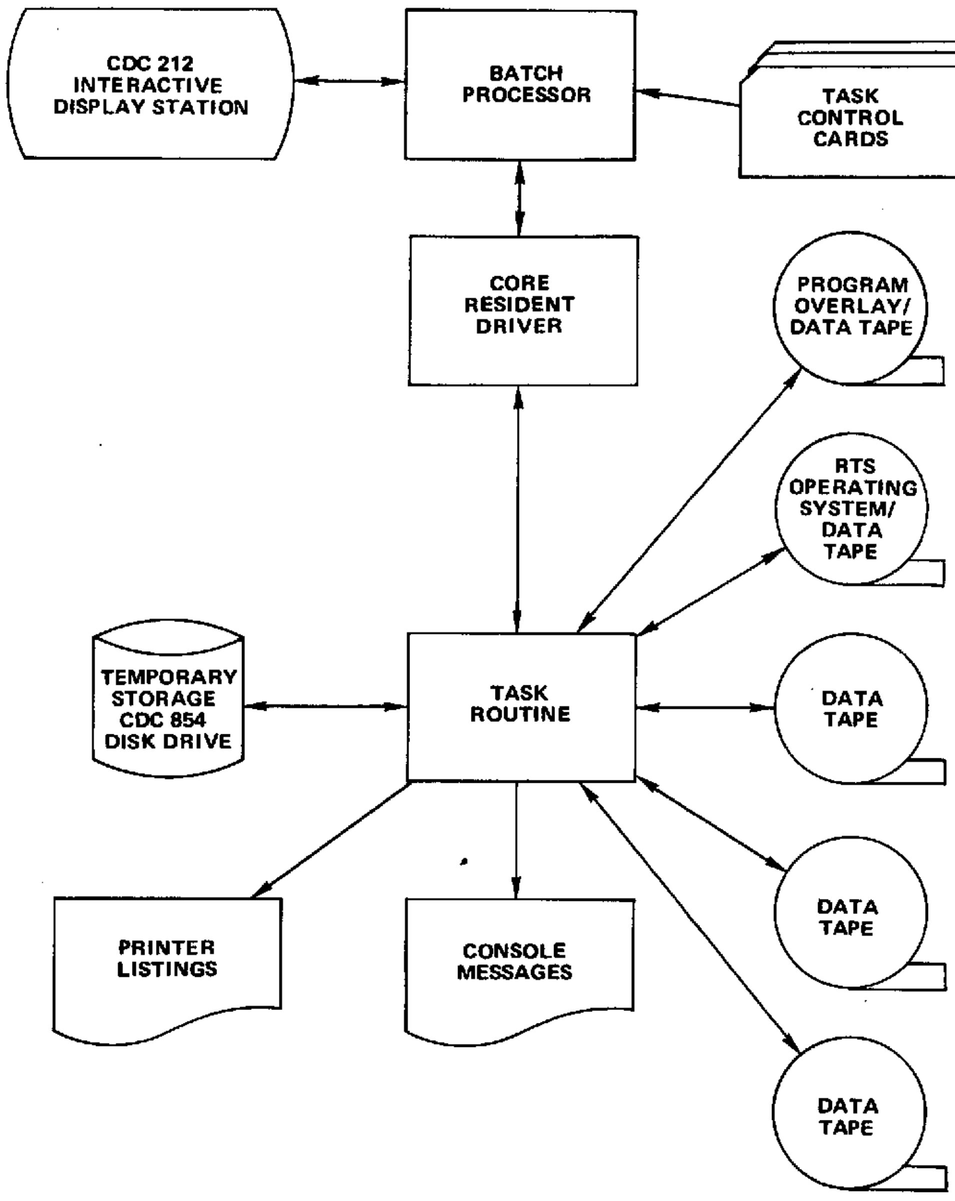

CDC 212 INTERACTIVE DISPLAY STATION
BATCH PROCESSOR
TASK CONTROL CARDS
CORE RESIDENT DRIVER
PROGRAM OVERLAY/ DATA TAPE
RTS OPERATING SYSTEM/ DATA TAPE
TEMPORARY STORAGE CDC 854 DISK DRIVE
TASK ROUTINE
DATA TAPE
DATA TAPE
DATA TAPE
PRINTER LISTINGS
CONSOLE MESSAGES

Figure 1-1. IDAMS Processor General Flow

SECTION 2 - EXECUTIVE PROGRAM DESCRIPTION

2.1 INTRODUCTION

The IDAMS executive consists of a core-resident routine (DRIVER), a task entry module (BATCH), and an error processor (ERROR). Since BATCH and ERROR reside on the overlay tape with task routines, they have been included in Section 4, Task Program Descriptions. Table 2-1 lists these routines together with their overlay assignments.

2.2 DRIVER - IDAMS PROCESSOR EXECUTIVE PROGRAM

2.2.1 Program Description

DRIVER is the main core resident program used to control execution of all IDAMS task and special-purpose routines. DRIVER first calls the batch processing overlay to read and edit the first/next set of task control cards. If the return is normal, it calls the overlay for the indicated task. This sequence--first calling BATCH followed by a call to the proper overlay task--continues until an END card is encountered or a fatal error occurs. If the ERROR overlay determines that recovery is to be attempted from a fatal error condition, DRIVER continues with the next task.

2.2.2 Parameters

No parameters are necessary.

2.2.3 Input

There is no input to DRIVER.

2.2.4 Output

There is no output from DRIVER.

2.2.5 Examples

None.

Table 2-1. Task Program Overlay Assignments

Overlay	Name	Function	Overlay	Name	Function
01	BATCH	TASK ENTRY MODULE	25	CORREL	IMAGE CORRELATION
02	TESTGN	TEST IMAGE GENERATION	26	RESECT	SPATIAL RESECTION
03	LIST	TAPE TO PRINT (INTEGER)	27	UTMGEO	COORDINATE CONVERSION
04	CONTRAST	RADIOMETRIC CORRECTION	28	FPMULT	FLOATING POINT MULTIPLICATION
05	CONVOLVE	CONVOLUTION FILTERING	29	FPSUM	FLOATING POINT ADDITION
06	EXPAND	INVERSE CONVOLUTION FILTERING	30	FILTGN	FILTER GENERATION
07	SHADE	PHOTOMETRIC CORRECTION	31	RANDGRAY	RANDOM NOISE GENERATOR
08	FFT	DISCRETE FOURIER TRANSFORMATION	32	IMERGE	BULK IMAGE MERGING
09	FPCON	DATA SCALING AND CONVERSION	33	PMERGE	PRECISION IMAGE MERGING
10	SMOOTH	HAMMING AND HANNING SMOOTHING	34	PPUPDATE	DISK FILE EDITING
11	CXPACK	COMPLEX DATA PACKING	35	VPICIN	VICAR TO IDAMS CONVERSION
12	ERROR	ERROR PROCESSING	36	INCREASE	LINEAR IMAGE EXPANSION
13	REDUCE	IMAGE REDUCTION	37	COLOR	FALSE COLOR CODING
14	HISTO	HISTOGRAM AND STATISTICS	38	FPLIST	FLOATING POINT LISTING
15	CHAROUT	TAPE TO PRINT (CHARACTER)	39	DMDOUT	IDAMS TO NOAA CONVERSION
16	TEXTGN	TEXT GENERATION	40	ADDPIX	PICTURE ADDITION
17	NEIGHBOR	POINT NEIGHBORHOOD LISTING	41	FORMAT	IDAMS FORMAT CONVERSION
18	DISPLAY	INTERACTIVE DISPLAY PACKAGE	42	HISTCONT	HISTOGRAM-CONTRAST CORRECTION PROGRAM
19	MODIFY	IMAGE TAPE EDITING	43	JOYSTICK	DISPLAY SYSTEM WITH JOYSTICK CONTROL
20	INSERT	WINDOW INSERTION	44	MSSCON	SPECIAL-PURPOSE CONVOLUTION PROGRAM
21	GRID	GRID OVERLAY	45-64	TEST 45-64	TEST OVERLAYS
22	GEOMTRAN	GEOMETRIC TRANSFORMATION	65-99	UNUSED	
23	CHIPGN	CHIP GENERATION			
24	RZOMAP	RESEAU DETECTION			

2.2.6 Messages

None.

2.2.7 Flowchart

See Appendix A, Figure A-1.

SECTION 3 - GENERAL PURPOSE SUBROUTINE DESCRIPTIONS

3.1 LBLRD - TAPE LABEL READING SUBROUTINE

3.1.1 Program Description

This general-purpose subroutine uses a tape name, logical unit number, and file number to locate the label record of an input tape file for further processing. It makes an initial search to locate the proper file number. If it cannot be found, LBLRD sets the system error code and terminates abnormally. When a match is found on the file number, LBLRD checks the name and, if no match is found, again takes an abnormal error exit. When file number and name both match, LBLRD returns the contents of the label record to the calling program and writes them on the line printer.

3.1.2 Parameters

The calling sequence for LBLRD is

```
CALL LBLRD (LABEL)
```

where the parameter is

LABEL - array into which label contents are to be read. The calling program passes tape file name, logical unit number, and file number to LBLRD.

3.1.3 Input

LBLRD requires an input tape in standard IDAMS format.

3.1.4 Output

LBLRD prints the label on the line printer.

3.1.5 Examples

A typical label record listing produced by LBLRD is

```
INPUT LABEL DATA - NAME = JAMESBLU, 4096 PIXELS, 3218 LINES,
                          LUN 49, FILE 1
```

3.1.6 Messages

Message	Explanation
LBLRD - WARNING, NO MATCH ON LUN nn, LUN CHANGED TO nn	The logical unit number found on tape did not match the one supplied; the one supplied is being used.
BAD FILE NO nnn	The file number requested could not be located on the input tape; execution terminates.
aaaaaaaa ON LUN nn NE aaaaaaaa	The tape file name passed to LBLRD did not match the one found in the label record; execution terminates.

3.1.7 Flowchart

See Appendix B, Figure B-1.

3.2 LBLWRT - TAPE LABEL WRITING SUBROUTINE

3.2.1 Program Description

This general-purpose subroutine uses a logical unit number and file number to position an output IDAMS tape to the first available record and then writes the label record into that location.

Upon entry, LBLWRT prints the contents of the label record and rewinds the tape. It makes a search based on the input file number until the tape is at the point where the label is to be written. A bad file number will cause abnormal program termination. Once the proper position is found, LBLWRT writes the label and returns to the caller.

3.2.2 Parameters

The calling sequence for LBLWRT is

```
CALL LBLWRT (LABEL)
```

where the parameter is

LABEL -	array containing the contents of the label to be written. Information in the array includes the tape file name, number of characters per record, number of records, logical unit number, and file number.

3.2.3 Input

There is no input other than the label array.

3.2.4 Output

LBLWRT writes the contents of the label record on the output tape and prints them on the line printer.

3.2.5 Examples

A typical label record listing produced by LBLWRT is

```
OUTPUT LABEL DATA - NAME = JAMESBLU, 4096 PIXELS, 3218 LINES,
                           LUN 47, FILE 1
```

3.2.6 Messages

Message	Explanation
BAD FILE NO OR BAD LUN	The file number or logical unit number passed to LBLWRT did not correspond to data found on tape; execution terminates.

3.2.7 Flowchart

See Appendix B, Figure B-2.

3.3 IDAMSDSK - DISK INPUT/OUTPUT SUBROUTINE

3.3.1 Program Description

This routine provides disk I/O capabilities for the IDAMS task routines using the CDC Model 3234 Mass Storage Controller and Model 854 disk drive. The Model 854 has 32,320 sectors of 64 24-bit words each, organized into 202 cylinders, each of which has 10 tracks of 16 sectors apiece. IDAMSDSK enables the user to reference these sectors, or cells, in blocks of arbitrary length using sequential location numbers from 0 to 32319.

Entry DPSEEK converts the specified cell location number into cylinder, track, and sector address, issues a seek instruction to the Control Data Model 3234 Peripheral Controller, and returns to the caller without waiting for the recording heads to reach the specified position.

Entry DPPUT waits until the heads are positioned as specified by the last seek, and then writes a specified number of complete 64-word cells from a specified full-word buffer. Upon completion, a seek is issued for the cell immediately following the last one written onto, and control is returned to the caller. Entry DPPUTF issues the write command and then returns control to the caller immediately.

Entry DPFETCH waits until the heads are positioned as specified by the last previous seek and then reads a specified number of complete 64-word cells into the specified buffer. Upon completion, a seek is issued for the cell immediately following the last one read, and control is returned to the caller.

Entry DPFETCHF issues the read command and then returns control to the caller immediately.

Entry DPCHECK checks the status of the last command and waits for completion before returning to the caller.

3.3.2 Parameters

The calling sequences for IDAMSDSK are

```
CALL DPSEEK (IDLOC)
CALL DPPUT (BUFF, NCELL)
CALL DPFETCH (BUFF, NCELL)
CALL DPPUTF(BUFF, NCELL)
CALL DPFETCHF (BUFF, NCELL)
CALL DPCHECK
```

where the parameters are

1. IDLOC - Location number (0 to 32319) of cell to be read or written
2. BUFF - First core memory location of integer or real buffer for disk data transfer
3. NCELL-Number of 64-word cells to be read or written

3.3.3 Input

For DPFETCH only, disk cells are specified by the call parameters.

3.3.4 Output

For DPPUT only, disk cells as specified by the call parameters.

3.3.5 Example

Copy ten consecutive blocks of 500 core words each, located in BUFF (1) through BUFF (5000), onto the disk, starting at cell number 4000. The necessary sequence of FORTRAN instructions is

```
DIMENSION IBUFF (5000)
COMMON IDUMMY (5), IEROR
....
CALL DPSEEK (4000)
```

```
          IPOINT = 1
          DO 320 I = 1, 10
          CALL DPPUT (IBUFF (IPOINT), 8)
          IF (IEROR-4) 320, 320, 11111
    320   IPOINT = IPOINT + 500
  11111   (error processing routine)
```

Note that for this example the eighth cell in each block is not entirely filled by the 500 words, since 8 X 64 = 512; the extra 12 words in the cell are skipped over before starting to write the next block of data.

3.3.6 Messages

IDAMSDSK generates the following non-fatal messages:

Message	Explanation
DISK NOT READY - WHEN READY, PRESS FINISH	Disk power not turned on, not up to speed or Control A on front panel of controller may not be set to 2.
DISK CHECKWORD ERROR CYL nnnn TRK nnn	During disk read a checkword error was encountered; processing will continue using the doubtful data unless cancelled by operator.

The following fatal error messages may also be generated:

Message	Explanation
SEL, OUTW, OR INPW REJECTED REPEATEDLY	After accepting CON instruction, disk failed to accept subsequent instructions.
DISK OVERFLOW	The number of cells specified by DPPUT or DPFETCH would have required reading or writing beyond cell 32319.

Message	Explanation
DISK ADDRESS ERROR	Disk cell location specified by DPSEEK was not between 0 to 32319 (program error) or was refused by Disk Controller (hardware error).
LOST DATA	Channel transferred data faster than disk read or write rate (hardware error).
WRITE LOCKOUT	Unable to write on disk.
BAD DISK TRACK NO nn ON CYLINDER nnn	Specified track has been tagged as bad using Disk Controller switches.
RESERVED TRACK NO nn ON CYLINDER nnn	Specified track has been reserved using Disk Controller switches.
TOO MANY CHECKWORD ERRORS	Repeated checkword errors have occurred causing execution to terminate.
UNKNOWN DISK ERROR, STATUS CODE = nnnn	The specified status code does not indicate any of the errors above.

3.3.7 Flowchart

See Appendix B, Figure B-3.

3.4 READRITE - TAPE INPUT/OUTPUT SUBROUTINE

3.4.1 Program Description

This subroutine performs tape input/output processing for the IDAMS task routines. Entry points are provided to read or write a magnetic tape record with or without unit status checking. If necessary, recovery from parity errors is attempted.

Entry READ causes one logical record to be read, followed by a unit status check.

Entry READF causes one logical record to be read, followed by an immediate return to the calling program.

Entry WRITE will write one logical record and check the unit status.

Entry WRITEF will write one logical record and return immediately to the calling program.

Entry CHECK tests the status of the last tape I/O operation on the unit specified.

If a parity error is detected on one of the tape units READRITE checks to see whether or not a warning message is to be printed based on a preset counter. A second check is then made to terminate the run if a preset limit on the number of acceptable errors is passed. The program then takes the appropriate action and returns to the caller.

3.4.2 Parameters

The calling sequences for READRITE are

```
CALL READ (LUN, IOBUFF, LENGTH)
CALL READF (LUN, IOBUFF, LENGTH)
CALL WRITE (LUN, IOBUFF, LENGTH)
CALL WRITEF (LUN, IOBUFF, LENGTH)
CALL CHECK (LUN, IOBUFF, IOPT)
```

where the parameters are

LUN	- Tape logical unit number
IOBUFF	- Input/Output buffer starting location
LENGTH	- Number of words to read or write
IOPT	- Not currently used

3.4.3 Input

For READ and READF only, one tape record is read.

3.4.4 Output

For WRITE and WRITEF only, one tape record is written.

3.4.5 Example

None.

3.4.6 Messages

READRITE may generate the following messages:

Message	Explanation
PAR ERR, LINE nnnn, LUN nn	A parity error has been detected on the indicated tape unit or on disk, execution continues.
nn PARITY ERRORS ON LUN nn, RUN WILL ABORT AT nn	More than an arbitrary number of tape parity errors has been detected, execution will terminate if a threshold number is exceeded.

3.4.7 Flowchart

See Appendix B, Figure B-4.

3.5 UTMCON - UTM/GEOGRAPHIC COORDINATE INTERCONVERSION SUBROUTINE

3.5.1 Program Description

UTMCON converts Universal Transverse Mercator (UTM) grid coordinates to geographic latitude-longitude coordinates or, as specified by the calling program, carries out the inverse transformation.

UTMCON begins by examining the zone designator, IZONE. If it is non-zero, latitude and longitude values are converted to UTM northing and easting using as central meridian longitude

$$L_{c.m.} = -183^{\circ} + 6^{\circ} \times |\text{IZONE}|$$

where negative values of $L_{c.m.}$ are west of Greenwich, and positive values are east. The difference λ between the input longitude and $L_{c.m.}$ is computed; if this value exceeds 4-1/2°, the value of IEROR in COMMON is set and control is returned to the calling program without carrying out the conversion. Similar action is taken if the latitude is not between -90° and +90°, where a negative value is interpreted as south latitude.

The conversion is then carried out using the formulae

$$E = r \sin^{-1} (\sin\lambda \cos\phi)$$

$$N = s + r\left[\sin^{-1}\left(\frac{\sin\phi}{\cos\frac{E}{r}}\right) - \phi\right]$$

$$s = a\,(c_1\phi - c_2\sin 2\phi + c_3\sin 4\phi)$$

$$r = a\,(1 - e^2 \sin^2 \phi)^{-1/2}$$

where

a = .9996 x semi-major axis of spheroid

e^2 = eccentricity squared

E = grid meters east of central meridian

N = grid meters north of equator

ϕ = latitude (negative for southern hemisphere)

λ = longitude relative to central meridian of zone (negative for west of c.m.)

and the expansion for meridian distances s, accurate to < 0.2 meters, has coefficients

$$c_1 = 1 - \frac{1}{4}e^2 - \frac{3}{64}e^4$$

$$c_2 = \frac{3}{8}e^2 + \frac{3}{32}e^4$$

$$c_3 = \frac{15}{256}e^4$$

In order to produce a positive easting value and record the UTM zone, 500000 + IZONE x 10^6 is added to the easting value. The easting and northing values are then returned to the calling routine.

If IZONE = 0, a conversion from UTM to latitude-longitude is carried out. The value of easting passed by the calling routine is divided by 10^6; the quotient represents the zone number, and the remainder the UTM easting. The false easting of 500,000 is subtracted from the easting to obtain the value of E.

Because the equation relating N and ϕ cannot be solved explicitly for ϕ, an iterative technique is used. An initial estimate of ϕ is computed as

$$\phi_1 = \frac{N}{a_s\left(1 - e^2 + \frac{1}{2}\frac{E^2}{a^2}\right)}$$

Successive approximations for ϕ are then computed by first computing the value N (ϕ_i) obtained using the i-th estimate, and then using

$$\phi_{i+1} = \frac{N - N(\phi_i)}{a\left[1 + \frac{1}{2\cos^2\phi_i}\,\frac{E^2}{a^2} + e^2\left(\frac{3}{2}\sin^2\phi_i - 1\right)\right]}$$

When $N\text{-}N(\phi_i) < 1$ meter, the estimate ϕ_{i+1} is taken as the value of ϕ; normally only two iterations are required to obtain an accuracy of better than 0.5 meters. If adequate convergence is not obtained within five iterations, IEROR is set and control is returned to the calling program without performing the conversion. Otherwise, the longitude relative to the central meridian is computed using

$$\lambda = \sin^{-1}\left(\sin\frac{E}{r}/\cos\phi\right)$$

and added to the central meridian longitude for the specified zone. The latitude and longitude values are then returned to the calling program.

3.5.2 Parameters

The calling sequence for UTMCON is

CALL UTMCON (XLAT, XLONG, UTME, UTMN, IZONE)

where the parameters are

XLAT -	Latitude in degrees and decimal fraction; south latitude is denoted by a minus sign
XLONG -	Longitude in degrees and decimal fraction; longitude west of Greenwich is denoted by a minus sign
UTME -	UTM easting, with central meridian assigned a false easting of 500,000 meters. The value is preceded by 10^6 x zone number.

UTMN - UTM northing, with negative values representing southern hemisphere northing less false northing of 10^7 meters; i.e., negative values are distances from equator

IZONE - UTM zone if conversion from lat/long to UTM is desired; 0 if conversion from UTM to lat/long is required

For either type of conversion, one pair of coordinate parameters are dummy locations into which UTMCON will store the results of converting the values passed in the other pair of coordinate locations.

3.5.3 Input

The only input to UTMCON is through the parameters.

3.5.4 Output

The output from UTMCON is stored into the locations specified in the calling sequence parameters.

3.5.5 Examples

Find the latitude and longitude corresponding to UTM easting of 648320 and UTM northing of 6423880, in UTM zone 19. The call is

```
CALL UTMCON (XLAT, XLONG, 19648320., 6423880., 0)
```

Find the UTM easting and northing for 38.447203 degrees north latitude and 96.266667 deg. west longitude. The result is to be referred to UTM zone 15 (central meridian 93° W). The call is

```
CALL UTMCON (38.447203, -96.266667, UTME, UTMN, 15)
```

3.5.6 Messages

UTMCON sets the value of IEROR in COMMON to indicate three error conditions, as follows:

Message	Explanation
LONGITUDE MORE THAN 4.5 DEG FROM C. M.	Input longitude too far from central meridian of specified zone.
LATITUDE GREATER THAN 90 DEG	Error in input parameter.
NO CONVERGENCE FOR LATITUDE	Input northing too large; corresponding latitude cannot be found.

In each case, control is returned to the calling program without carrying out the requested conversion.

3.5.7 Flowchart

See Appendix B, Figure B-5.

3.6 TWOFIT - TWO-DIMENSIONAL LEAST-SQUARES FIT SUBROUTINE

3.6.1 Program Description

This general-purpose subroutine determines the coefficients of the two-dimensional polynomial(s) which yield(s) the least-squares best fit(s) to one or more sets of values determined at an arbitrary set of points. The variance between the values obtained from the least-squares polynomial coefficients and the input data is also computed.

TWOFIT begins by checking that the degree I_{deg} of the required polynomial is between 1 and 5, and that the number of points N_{pt} for which values are specified satisfies

$$N_{pt} \geq \frac{1}{2}\left(I_{deg} + 1\right)\left(I_{deg} + 2\right)$$

If either condition is not satisfied, an error code is set and control is returned immediately to the calling program.

Otherwise, TWOFIT computes the coefficients b_{kl} of the approximating polynomial

$$z\,(x,y) = \sum_{k=0}^{I_{deg}} \sum_{l=0}^{I_{deg-k}} b_{kl}\, x^k y^l$$

by finding the values of b_{kl} for which the variance between actual values z_i at the points (x_i, y_i) and the computed values $z(x_i, y_i)$ is a minimum; this variance is given by

$$V = \frac{1}{N_{pt}} \sum_{i=1}^{N_{pt}} \left[z_i - z(x_i, y_i)\right]^2$$

The computation is carried out by calculating the coefficient matrix and right-hand side column vector(s) of the system of linear equations

$$\sum_{i=1}^{N_{pt}} \sum_{k=0}^{I_{deg}} \sum_{l=0}^{I_{deg-k}} x_i^{m+k}\, y_i^{n+l}\, b_{kl} = \sum_{i=1}^{N_{pt}} z_i\, x_i^m\, y_i^n$$

$$0 \leq m \leq I_{deg}, \quad 0 \leq n \leq I_{deg-m}$$

The general-purpose subroutine MATINV is then called to obtain the solutions for the coefficients b_{kl} by matrix inversion.

In the event that the matrix is found to be singular, an error code is set and no solutions are returned; otherwise, TWOFIT returns one set of coefficients b_{kl} for each set of values z_i passed in the call.

3.6.2 Parameters

The calling sequence for TWOFIT is

```
CALL TWOFIT (IDEG, NPT, NSET, X, Y, Z, A, B)
```

where the parameters are

IDEG	- Degree of polynomial desired ($1 \le \text{IDEG} \le 5$)
NPT	- Number of points for which values are supplied
NSET	- Number of sets of values to be fit
X	- NPT floating point values of x-coordinates of data points
Y	- NPT floating-point values of y-coordinates
Z	- NPT * NSET floating-point observed values at points (x, y). The first set of NPT values are given in the first NPT locations; the second set in the next NPT locations, and so on
A	- A dummy array containing at least $\frac{1}{4}(\text{IDEG}+1)^2(\text{IDEG}+2)^2$ floating point locations for intermediate computation, contains values of variance on return

B Contains floating-point polynomial coefficients on return; must provide at least $\frac{1}{2}$ NSET*(IDEG + 1)*(IDEG + 2) locations. First set of coefficients occupy first $\frac{1}{2}$ (IDEG +1)*(IDEG + 2) locations, and so on.

<u>NOTE</u>: The points at which values are specified should include at least IDEG + 1 different x-values and IDEG + 1 different y-values.

3.6.3 <u>Input</u>

Input to TWOFIT is entirely through the calling sequence, as specified above.

3.6.4 <u>Output</u>

The results of TWOFIT are returned via the calling sequence and the error code IEROR in COMMON.

3.6.5 <u>Example</u>

Two sets of displacements, DELTAX and DELTAY, have been determined for 69 points whose coordinates are located in arrays X and Y. It is desired to obtain the coefficients for the least-squares third-degree polynomials and place them in arrays COEFFX and COEFFY. Suitable FORTRAN statements are

```
      DIMENSION COEFFX (10), COEFFY (10), A (100), DELTAX (69),
     1  DELTAY (69), DELTA (138), X (69), Y (69), COEFF (20)
      EQUIVALENCE (DELTA (1), DELTAX (1)), (DELTA (70),
     1  DELTAY (1)), (COEFF (1), COEFFX (1)),
     2  (COEFF (11), COEFFY (1))

      CALL TWOFIT (3, 69, 2, X, Y, DELTA, A, COEFF)
```

On return, the variance between the actual values and those computed from the polynomials will be located in A(1) for DELTAX and A(2) for DELTAY.

3.6.6 Messages

TWOFIT sets error codes for three abnormal conditions, as follows:

Message	Explanation
POLYNOMIAL DEGREE LT 1 OR GT 5	Requested polynomial degree not between 1 and 5
TOO FEW KNOWN POINTS	Number of points with known values less than $\frac{1}{2}\left(I_{deg} + 1\right)\left(I_{deg} + 2\right)$
MATINV GAVE SINGULAR SOLUTION	MATINV gave singular solution: known points did not occupy at least $\left(I_{deg} + 1\right)$ different x positions $\left(I_{deg} + 1\right)$ different y positions

3.6.7 Flowchart

See Appendix B, Figure B-6.

3.7 MATINV - MATRIX INVERSION SUBROUTINE

3.7.1 Program Description

This general-purpose subroutine inverts a square matrix of up to 21 x 21 and/or finds the solutions to one or more sets of simultaneous linear equations whose coefficient array is represented by the matrix. The determinant of the matrix is also computed.

MATINV uses the Gauss-Jordon method of pivotal condensation, with total pivot searching and row interchange. It is a modification of a routine developed by D. C. Knight of the Commonwealth Scientific and Industrial Research Organization, Adelaide, Australia.

Execution begins by searching the matrix for the element having the largest absolute value. This element is moved onto the principal diagonal and the other elements in the pivot column reduced to zero by subtracting multiples of the pivot row from the other rows. The search and reduction continues until all columns have been reduced, giving the solution vectors, if requested. The inverse of the matrix is then computed, if requested, by interchanging columns of the matrix so as to reverse the effects of the row interchange. The determinant is computed as the product of the pivotal elements. If at any time the pivot value falls below 1.0×10^{-90}, the matrix is treated as being singular, and the determinant value is set to zero.

3.7.2 Parameters

The calling sequence is

```
CALL MATINV (A, B, N, L, DET)
```

where the parameters are

A -	Array to be inverted; contains inverse matrix on return

B	-	Array of right-hand side vectors for sets of simultaneous linear equations of the form AX = B; on return, contains solution vectors X
N	-	Order of A
L	-	$\|L\|$ is number of vectors in B. The sign of L is used to specify the desired combination of solutions and/or inverse, as follows: $L > 0$, only solutions are given $L = 0$, only matrix inverse is given $L < 0$, both solutions and inverse are given
DET	-	Contains determinant of A on return, DET = 0 for singular matrix

3.7.3 Input

The only input to MATINV is via the calling sequence.

3.7.4 Output

The only output from MATINV is returned through the subroutine arguments.

3.7.5 Example

It is desired to obtain solutions to the two sets of simultaneous linear equations

$$\sum_{j=1}^{8} a_{ij} x_j = y_i, \quad i = 1, 8$$

$$\sum_{j=1}^{8} a_{ij} x_j = z_i, \quad i = 1, 8$$

These equations can be represented in matrix notation as

$$AX = B$$

where B is a two-column matrix whose first column contains the vector $\{y_i\}$ and whose second column is $\{z_i\}$, and X is a column vector composed of the unknowns $\{x_i\}$. The solutions can be obtained by calling MATINV as follows:

```
CALL MATINV (A, B, 8, 2, DET)
```

On return, the two sets of solutions will be in the two columns of B. Since $L > 0$, the inverse of A is not computed. The results should be checked for the possibility of a singular matrix by testing whether $DET = 0$.

3.7.6 Messages

MATINV generates no messages.

3.7.7 Flowchart

See Appendix B, Figure B-7.

3.8 PERGEN - INVERTED BIT-ORDER PERMUTATION GENERATING SUBROUTINE

3.8.1 Program Description

This general-purpose subroutine generates a table of integer index values which may then be used to place a complex data array in permuted order for use by FFTONE (one-dimensional Fast Fourier transform subroutine). PERGEN computes the table values by reversing the bit-order of each binary number between 0 and the table size minus one. It modifies the results to take into account FORTRAN indexing methods, including the use of two floating-point numbers to represent each complex value in the array. All odd-numbered entries are calculated and then copied into even-numbered entries before exiting.

3.8.2 Parameters

The calling sequence for PERGEN is

```
CALL PERGEN (MX, PERTBL)
```

where the parameters are

MX	- Number of binary digits in index
PERTBL	- Buffer for storing final results

3.8.3 Input

PERGEN needs only the parameter, MX, and a table area of sufficient size.

3.8.4 Output

PERGEN produces the table, PERTBL.

3.8.5 Examples

None.

3.8.6 Messages

None.

3.8.7 Flowchart

See Appendix B, Figure B-8.

3.9 TRIGGN - SINE TABLE GENERATING SUBROUTINE

3.9.1 Program Description

This general purpose subroutine generates a table of sines of angles between 0 and $\pi/2$ in steps of $2\pi/NX$, where NX is the number of columns in the array.

3.9.2 Parameters

The calling sequence for TRIGGN is

```
CALL TRIGGN (MX, TRGTBL)
```

where the parameters are

MX	- Size of table, 2**(MX - 1) values are generated.
TRGTBL	- Storage location for sine values.

3.9.3 Input

TRIGGN requires the parameter, MX, and a table area of sufficient size.

3.9.4 Output

TRIGGN produces the sine table, TRGTBL.

3.9.5 Examples

None.

3.9.6 Messages

None.

3.9.7 Flowchart

See Appendix B, Figure B-9.

3.10 FFTONE - ONE-DIMENSIONAL FAST FOURIER TRANSFORM SUBROUTINE

3.10.1 Program Description

FFTONE carries out a one-dimensional fast Fourier transform on a line of data. It begins by computing the appropriate normalization factor. FFTONE then permutes the complex values to reverse the bit-order of their binary indices. Before they are stored into their new locations, each value is multiplied by the normalizing factor. The line is then transformed using the one-dimensional fast Fourier transform algorithm, and control returns to the calling program.

3.10.2 Parameters

The calling sequence for FFTONE is as follows:

```
CALL FFTONE (MX, DATA, TRGTBL, PERTBL)
```

where the parameters are

MX	- Number of passes for FFT, log base 2 of N
DATA	- Location of complex array to be transformed
TRGTBL	- Table of sines for first quadrant
PERTBL	- Bit-order reversed table created by PERGEN

3.10.3 Input

FFTONE needs only the input parameters.

3.10.4 Output

FFTONE creates one transformed line of data in the output buffer.

3.10.5 Examples

None.

3.10.6 Messages

None.

3.10.7 Flowchart

See Appendix B, Figure B-10.

3.11 CODE - CHARACTER TRANSLATION SUBROUTINE

3.11.1 Program Description

The general-purpose COMPASS subroutine CODE provides high-speed character translation by table lookup. CODE loads each character of the input array in turn, beginning with the last one, and uses it to index one of 64 entries in the translation table. It stores this entry into the output array.

CODE requires about 11 microseconds for each character translated.

3.11.2 Parameters

The calling sequence for CODE is

```
CALL CODE (N, DATAIN, DATOT, TABLE)
```

where the parameters are

N	- Number of consecutive characters to be translated
DATAIN	- Location of first character of input array
DATOT	- Location of first character of output array. DATOT might coincide with DATAIN; overlapping is permitted only if DATOT $\geq$ DATAIN.
TABLE	- Full-word array of translation values. Only low-order six bits of each word are used; first word contains translation for input character 00_8, next word for input 01_8, and so on to input 77_8

3.11.3 Input

CODE passes the number of characters to be translated, an array containing the data to be translated by the calling sequence and a translation table.

3.11.4 Output

CODE returns an array containing the translated data through the calling sequence.

3.11.5 Example

Not applicable.

3.11.6 Messages

CODE generates no messages.

3.11.7 Flowchart

See Appendix B, Figure B-11.

3.12 MOVE - CHARACTER MOVING SUBROUTINE

3.12.1 Program Description

The general-purpose COMPASS subroutine MOVE moves data within core. It moves the data in blocks of 128 characters, with the exception of a smaller first block, using the CDC 3200 Block Control MOVE instruction.

MOVE requires about 4 1/2 microseconds for each character moved; if the source and destination addresses are full-word boundaries and the count is a multiple of 4, however, this time is reduced by a factor of 4.

3.12.2 Parameters

The calling sequence for MOVE is

```
CALL MOVE (N, S, D)
```

where the parameters are

N - Number of characters to be moved

S - Character address of start of source array

D - Character address of start of destination array

If D = S + 1, MOVE will propagate the first character of S through all of D; this may be used for zeroing out an array.

3.12.3 Input

MOVE passes the number of characters to be moved and the character addresses of the origin and destination of the data being moved through the calling sequence.

3.12.4 Output

MOVE moves data into the user-specified location.

3.12.5 Examples

Not applicable.

3.12.6 Messages

MOVE generates no messages.

3.12.7 Flowchart

See Appendix B, Figure B-12.

3.13 CODE8TO6 - BYTE TO CHARACTER CONVERSIONS SUBROUTINE

3.13.1 Program Description

This general-purpose subroutine uses table look-up to translate 8-bit bytes, packed three per word, into 6-bit characters, packed four per word.

CODE8TO6 begins by accessing the character count, input and output buffer addresses, and table location and storing them into the main loop. Then one input word is loaded at a time. The bytes are accessed one at a time by shifting them, transferring one byte to an index register, and using it to load a value from the translation table; the resulting character is then stored into the output buffer. When the required number of characters have been translated in this manner, control is returned to the caller.

3.13.2 Parameters

The calling sequence is

```
CALL CODE8TO6 (N, IBUFFIN, CBUFFOUT, ITABLE)
```

where the parameters are

N	-	Number of characters to translate
IBUFFIN	-	Integer array containing input
CBUFFOUT	-	Character array for output
ITABLE	-	Integer translation table containing 256 entries

3.13.3 Input

The only input is via the calling sequence.

3.13.4 Output

The translated output is in the array specified by the calling sequence.

3.13.5 Example

It is desired to translate 24 words (72 bytes) in a buffer INBUFF and place them into a character array CBUFF starting with character position 15. The translation is to be as specified by a table ITAB. The required calling sequence is

```
      CALL CODE8TO6 (72, INBUFF, CBUFF(15), ITAB)
```

3.13.6 Messages

CODE8TO6 generates no messages.

3.13.7 Flowchart

See Appendix B, Figure B-13.

3.14 ADDLINE - MESSAGE ARRAY UPDATE PROGRAM

3.14.1 Program Description

This is a general-purpose subroutine which adds one or more lines of BCD data to the permanent 212 message array in COMMON. If the array is already full, enough lines are made available for the new data by rolling the array from bottom to top, truncating the oldest data.

Upon entry, ADDLINE calculates the number of lines needed for the new message and makes the necessary room. Old data is blanked out and new data is moved in. The program then returns to the caller.

3.14.2 Parameters

The calling sequence is

```
CALL ADDLINE (ICHARS, CLINE)
```

where the parameters are

ICHARS	- Character count for the message to be added.
CLINE	- Beginning location of the new message (character address).

3.14.3 Input

The only input is via the calling parameters.

3.14.4 Output

Output is to the message array in COMMON.

3.14.5 Examples

None.

3.14.6 Messages

None.

3.14.7 Flowchart

See Appendix B, Figure B-14.

3.15 TTWLVE - CDC 212 INPUT/OUTPUT SUBROUTINE

3.15.1 Program Description

TTWLVE is the input/output driver for the CDC 212 Display Station. It has four entry points; TTWCON, CDCON, STORE, and INTER.

TTWCON is the CDC 212 initialization routine which sets up an address in the CIT table in core, and clears and connects channel 4. It must be called before any attempt is made to read or write on the 212. Upon entry to TTWCON, interrupt control is disabled, channel 4 is cleared, and the connect instruction is issued. If rejected repeatedly, a message is written to the operator and the program waits for a ready condition. When successful, the address of the interrupt processor is stored in the CIT table, interrupt control is enabled, and the program returns to the caller.

CDCON is the 212 display output routine. When this routine is called, up to 250 words are presented on the 212 screen. Upon entry to CDCON, the input buffer address and word count parameters are picked up and stored into instructions later in the program. A loop is then entered which converts the input data from internal BCD to 212 external code. The screen is blanked and data is output a line at a time until done. If output fails because of a channel busy condition, a message is written to the operator. Before returning, the converted data is restored to its original code.

STORE is the 212 input routine. This routine transfers up to 250 data words from the display screen into core. Upon entry to STORE, the return data buffer address and word count parameters are stored where needed in the program. A loop is then entered to wait for an interrupt from the 212. After the interrupt is made and serviced by INTER, STORE returns to the caller.

INTER is the 212 interrupt processor which reads data from the 212 into the area defined by the buffer passed when a call to STORE is executed. This routine is entered when the SEND switch is depressed on the 212 keyboard. Upon entry to INTER, initialization is performed and the input area is filled by executing an INPW instruction. A channel busy message is generated if necessary. After a successful data transfer, an interrupt switch is set to 1 in COMMON location IPARIT(10). The data is then converted to internal BCD code and the entire array is shifted left two characters. The program then returns control to the central Interrupt Processor.

3.15.2 Parameters

The calling sequences for TTWLVE are as follows:

```
CALL TTWCON
CALL CDCON (IWRDBUFF, COUNT)
CALL STORE (IWRDBUFF, COUNT + 1)
```

where the parameters are

IWRDBUFF	-	Array address of first data word
COUNT	-	Integer word count for amount of data to be transferred

3.15.3 Input

TTWLVE reads an array from the 212 when a call to STORE is executed followed by an external interrupt generated at the 212 keyboard.

3.15.4 Output

TTWLVE writes an array to the 212 when a call to CDCON is executed.

3.15.5 Example

None

3.15.6 Messages

TTWLVE may generate the following advisory messages:

Message	Explanation
212 CONNECT FAILURE, NOW LOOPING UNTIL READY	Could not connect channel 4, hardware not ready.
CHANNEL 4 BUSY OVER 3 SECONDS	A channel 4 input/output operation has failed continuously for three seconds, hardware not ready.

3.15.7 Flowchart

See Appendix B, Figure B-15.

SECTION 4 - TASK PROGRAM DESCRIPTIONS

4.1 BATCH - TASK ENTRY PROGRAM

4.1.1 Program Description

This is a task entry module which receives task and parameter data either from cards or the 212 display station. BATCH decodes and uses this information to set up conditions for executing the various user tasks within the IDAMS processor. DRIVER calls BATCH before execution of each task to bring in from cards or the 212 display the parameter information necessary for the execution of one task program. BATCH interprets the free format input data, edit-checks tape name, logical unit and file numbers, and input picture coordinates, then stores these values in COMMON. In addition, special parameters required for an individual task are transferred to COMMON.

Upon entry, BATCH performs initialization of constants and arrays. Messages are written out indicating card or interactive mode. For interactive mode, the message "READY FOR INPUT" is written to the 212 followed by a read from that device. A call to CDCON performs the 212 write function while a call to STORE does the read. For card mode, data is read directly from cards.

After the data has been initially read into the program, BATCH decodes fields and subfields on the task card. An undefined task name will cause a fatal error, as will improper use of field delimiters.

BATCH next performs a limits check on logical unit and file numbers, then calls PARAMS to decode the parameter data. This is followed by further limits checks on starting line and pixel and number of lines and pixels. The primary input label is read and both the number of pixels and number of lines are reduced if necessary.

BATCH then returns control to DRIVER to execute the next task.

4.1.2 Parameters

BATCH functions with DRIVER as part of the executive, and requires no parameters.

4.1.3 Input

BATCH reads the user-supplied task and data from cards or the 212 display, and transmits the information to the indicated task program through COMMON.

4.1.4 Output

BATCH lists each input card on the printer and 212 display station as it is processed.

4.1.5 Examples

Not applicable.

4.1.6 Messages

BATCH can generate the following non-fatal messages:

```
IDAMS PROCESSOR - CARD MODE
THIS IS taskname (console only)

                     or

variable task and parameter data    { Each user-supplied set of task and
(printer and 212 display only)      { parameter information is written
                                    { to the printer and 212 display.
```

When a SWITCH task card is encountered, these additional messages will be written

```
NOW SWITCHING MODES
IDAMS PROCESSOR - INTERACTIVE MODE
READY FOR INPUT (printer and 212 display only)
```

If the number of pixels or lines must be reduced, BATCH will write one of the following messages:

NUMBER OF PIXELS REDUCED TO nnnn

or

NUMBER OF LINES REDUCED TO nnnn

BATCH may also generate a number of fatal error messages, as follows:

Message	Explanation
ILLEGAL CARD TYPE	The task name read did not match any name kept in an internal table.
MISSING DELIMITER ON TASK CARD	Parentheses, commas, or fields were incorrect on the task card.
2NDARY INPUT LUN LT 1 OR GT 55	The secondary input tape logical unit number specified was outside the limits defined for programmer units.
2NDARY INPUT FILE NO LT 1 OR GT 999	The secondary input file number did not fit within the arbitrary limits of 1 to 999.
OUTPUT LUN LT 1 OR GT 55	The output logical unit number specified was outside the system limits defined for programmer units.

4.1.7 Flowchart

See Appendix C, Figure C-1.

4.2 TESTGN - TEST PATTERN GENERATION PROGRAM

4.2.1 Program Description

This task subroutine generates a test image (Figure 4-1) having 270 lines of 340 pixels each, as follows:

1. Standard resolution bars in left half
2. Single blip at line 45, pixel 240
3. 8-x-8 array of 20-x-20 uniform gray squares in lower right.

Upon entry, TESTGN initializes pointers necessary to define boundaries for the resolution bars. It writes a label in IDAMS format on the first record of the output tape. The bars are in an internal array of 230 lines, each containing 120 pixels. After 20 lines of zeros are written, TESTGN moves the resolution bar data into the left half of lines 21 through 250. The right half of these lines is all zero except line 45, pixel 240, which contains the single point image. Lines 91 through 250 have bars in the left half and uniformly increasing 20-x-20 blocks of gray levels in the right half. The last 20 lines and the first and last 20 pixels in every line are all zero, forming a border around the test image.

4.2.2 Parameters

No special parameters are required.

4.2.3 Input

There is no input other than the TESTGN task control card.

4.2.4 Output

TESTGN creates a test image of 270 lines and 340 pixels containing resolution bars, a single point image, and uniform gray blocks in standard IDAMS format on an output tape.

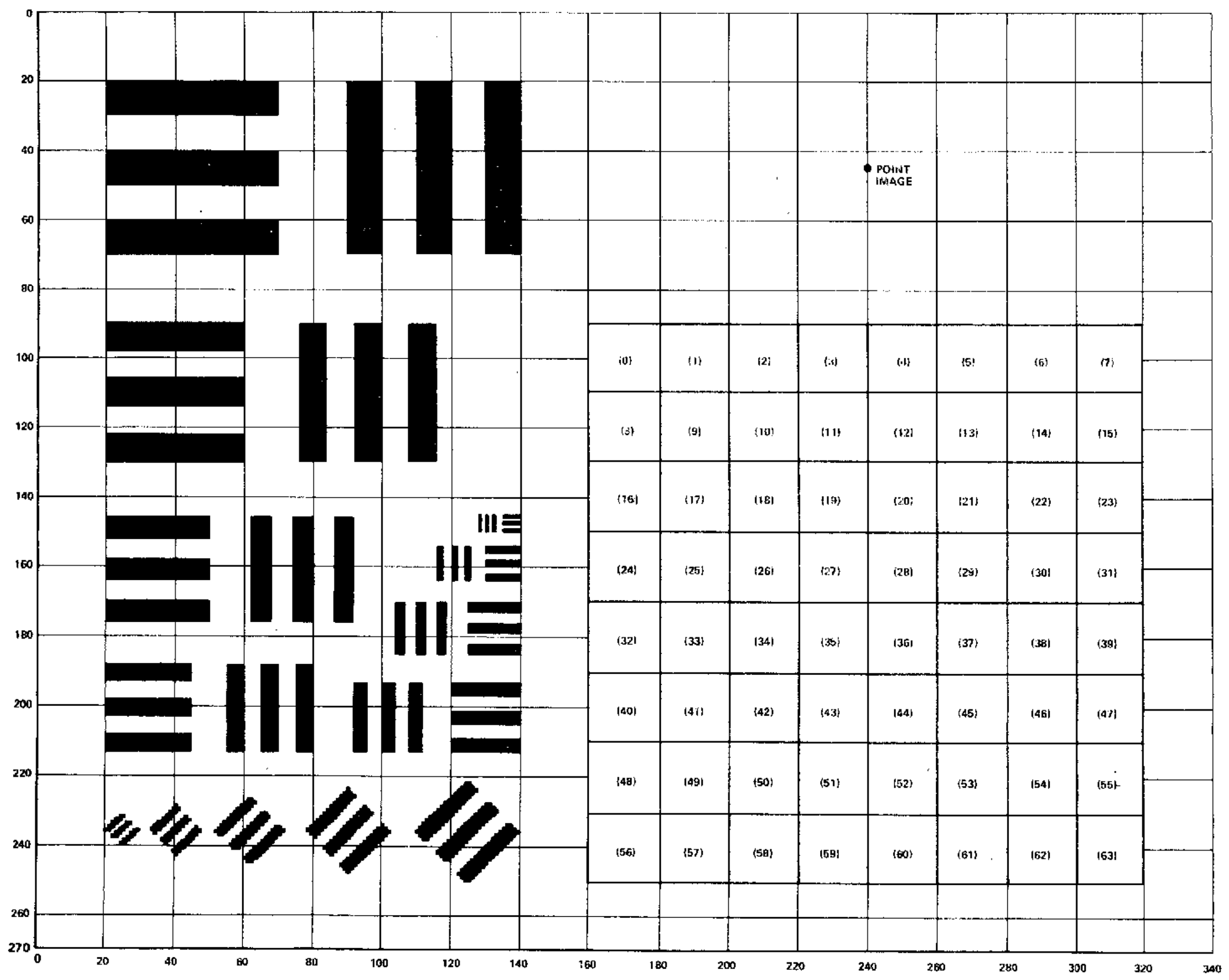


Figure 4-1. Resolution Test Target

4.2.5 Examples

To create the standard test image of 270 lines and 340 pixels each, the following task control card is necessary:

```
TESTGN,,,(TEST1,49,1)
```

This will create the previously described pattern on the tape mounted on logical unit 49. The IDAMS label record will contain the name "TEST1" for future reference.

NOTE: Card format specifications are defined in the IDAMS User's Guide.

4.2.6 Messages

There are no messages.

4.2.7 Flowchart

See Appendix C, Figure C-2.

4.3 LIST - TAPE TO PRINTER UTILITY PROGRAM

4.3.1 Program Description

LIST is a task routine used to generate a printed listing of selected sections of selected lines of images. It is used primarily for checkout and evaluation of other tasks.

Based on input data, LIST accesses the proper tape unit and reads and prints the tape label for the specified file. It compares the number of lines and pixels requested with the actual values found on tape, and reduces one or both of the requested values, if necessary. It then prints a line at a time in integer format, preceded by the line number, until it has listed all requested data. The user has the option of specifying packed or spaced output via an input parameter.

4.3.2 Parameters

LIST requires the following special parameters:

1.	SKIP -	interval between successive input lines. SKIP = 1 or default if all input is to be printed
2.	IBLOCK -	block list option. If IBLOCK = 1, horizontal and vertical spaces are suppressed. If this field is defaulted, normal spacing is used.

4.3.3 Input

A tape containing integer data in standard IDAMS format is necessary.

4.3.4 Output

This routine lists the portion of the input tape specified through input parameters on the printer.

4.3.5 Examples

A previous program has created a tape, which is now mounted on LUN 47. Its overall data dimensions are 4096 pixels and 3218 lines. Its label name is JAMESBLU and the file number is 1. To list every 100th line beginning at pixel 20 and line 50, the following control cards are necessary:

```
LIST, (JAMESBLU, 47, 1), (20, 50, 4077, 3169), ,1
```

```
100
```

NOTE: Card format specifications are defined in the IDAMS User's Guide. Parameters must be supplied in the order shown in paragraph 4.3.2.

4.3.6 Messages

None.

4.3.7 Flowchart

See Appendix C, Figure C-3.

4.4 CONTRAST - RADIOMETRIC CORRECTION PROGRAM

4.4.1 Program Description

The task program CONTRAST provides for converting the gray level values of an image using table look-up. The table can be obtained from any of three sources: a standard table of radiometric corrections stored in the program; a complete conversion table supplied by the user; or a table generated by linear interpolation between user-supplied pairs of old and new gray-level values, which define a piecewise linear relation between old and new values. CONTRAST can also perform a bit masking operation on an IDAMS formatted data tape where the number of bits to be set to zero is determined by the first parameter.

Upon entry, CONTRAST examines the first input parameter, N. If $2 \leq N \leq 11$, N pairs of coordinate points must follow which define the piecewise linear graph to be used for computing the translation table. If $N = 1$, the translation table is input as parameters 2 - 65. If $N = 0$, an internal translation table is used. If $-5 \leq N \leq -1$, $|N|$ bits are masked by computing an appropriate translation table. The lowest order (rightmost) bit will be set to zero in each pixel if $N = -1$. In any case, after generating or storing the required table, CONTRAST reads in one line of input data at a time. Unwanted lines are not processed. General purpose subroutine CODE translates one character at a time to new gray-level values specified by the table. The finished line is written on output, and successive lines are produced in the same manner until the specified region of the image has been processed.

4.4.2 Parameters

The contrast conversion table can be specified in one of four ways. If the conversion is to follow a linear or piecewise linear relation, the parameters are shown in the following page.

1. N = number of pairs of coordinate points that follow ($2 \leq N \leq 11$)

2,3, Old and new values, respectively, for point at left-hand end of left-most line segment

4,5, Old and new values, respectively, for point at left-hand end of next line segment

.

.

.

2N, 2N + 1. Old and new values, respectively, for point at right-hand end of last (right-most) line segment

The first pair of values should include at least one zero; the last pair at least one 63. If the first old value is nonzero, all values less than it will be assigned a new value of zero. If the last old value is not 63, all values greater than it will be assigned the last new value. At most, 11 pairs of coordinates can be specified, corresponding to 10 contiguous line segments. In addition, the old values must be strictly increasing.

If a nonlinear conversion is required, the parameters are:

1. N = 1 Use table of new values entered as parameters 2 to 65 for old values 0 to 63, respectively

2-65. New values to which the old values 0 to 63, in that order, are to be converted; 64 values must be supplied

or,

1. N = 0 Use standard table, stored internally

If bit masking is desired, $-5 \leq N \leq -1$ Mask $|N|$ low order bits to zero in each pixel.

4.4.3 Input

CONTRAST requires a single input image tape in standard IDAMS format.

4.4.4 Output

CONTRAST generates a single output image tape in standard IDAMS format.

4.4.5 Examples

It is desired to: increase the contrast in the low-density regions of a picture, TEST1; decrease the contrast in the medium-density region; and not change contrast at higher densities, in accord with the function depicted in Figure 4-2. The entire picture of 3000 lines of 3600 pixels each is to be processed. Appropriate IDAMS source statements are:

```
CONTRAST, (TEST1, 49, 1), (1, 1, 3600, 3000), (CONTRAST, 47, 1),

4, 0, 0, 20, 40, 55, 55, 63, 63
```

NOTE: Card format specifications are defined in the User's Guide. Parameters must be supplied in the order shown in paragraph 4.4.2.

4.4.6 Messages

CONTRAST generates the following fatal error messages:

Message	Explanation
N NOT LE 11 AND GE -5	The first special parameter contained a value out of range.
COORDINATE VALUE GT 63 OR LT 0	An old or new gray level value was beyond the range 0-63.
OLD COORD NOT STRICTLY INCREASING	A specified value of the old intensity was less than or equal to the preceding value.
OLD COORD NOT STRICTLY INCREASING	A specified value of the old intensity was less than or equal to the preceding value; execution terminates.

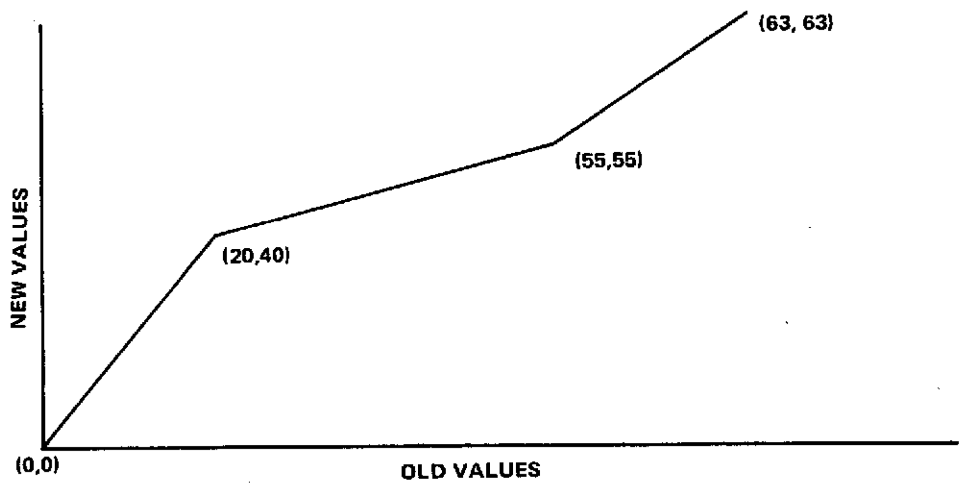


Figure 4-2. Contrast Conversions

4.4.7 Flowchart

See Appendix C, Figure C-4.

4.5 CONVOLVE - CONVOLUTION PROGRAM

4.5.1 Program Description

The task program CONVOLVE provides capabilities for convolving an image data set with a user-supplied weight matrix. Applications include simulation of sampling and blurring processes and digital filtering for edge enhancement and blur reduction. The program can generate output values for each input pixel, or at larger, user-specified increments between pixels and/or lines. The weight table can have either odd or even dimensions, it can either be symmetric about the x and y axes, in which case only one quadrant needs to be specified by the input parameters, or it can be nonsymmetric.

CONVOLVE begins by accessing the input parameters and computing the amount of COMMON required for storing the weights. The remainder of COMMON is dynamically allocated for picture data, to maximize processing efficiency. The program computes the sums of the positive weights and the negative weights separately, and checks that neither sum, after normalization, exceeds 32.5 in magnitude; a larger value could produce an uncorrectable overflow during computation. The program then normalizes the weights, with 12-bit fractional part, to make their sum equal unity; optionally, the user can specify alternative normalization.

The program then compares the dimensions of the specified region of the input image with the size of the entire input image. If the specified region exceeds the available input data, CONVOLVE reduces the specified numbers of lines and pixels to fit the available data, and writes an advisory message on the printer. If the specified region extends to or near the edge of the available data, the program makes provision for copying the boundary pixels outward to minimize edge effects by ensuring that each element of the weight matrix will always have a corresponding pixel value.

The program then compares the dimensions of the specified region with the available COMMON size. If the entire input region will not fit into core at one time, the program makes provision for breaking the image into horizontal strips. If each such strip contains fewer lines of data than the number of lines of weights, the program also segments the image into vertical strips. Next, it computes the remaining constants required for reading, writing, shifting, and convolving the data. It passes the constants required by the COMPASS subroutine ADDWTS by calling ADDPRM, which stores the parameters and modifies ADDWTS logic to provide maximum efficiency for the particular set of parameters.

The program reads input data into core until the available space is filled, and copies data on the edges of the input image outwards, if required. If segmentation into vertical strips is required, the program first transfers data from tape to disk, and then reads back into core from there.

The program calls subroutine ADDWTS to carry out the convolution to generate one line of output. To compute each output pixel, ADDWTS first resets the variable SUM to zero. For each weight, from one to four input pixels, depending on the symmetry of the weight array, are loaded and added together, and the sum is multiplied by the weight. This product is added to SUM. When all weights have been used, SUM is divided, with rounding, by 4096 to eliminate the 12-bit fractional part. If the result is negative, it is replaced by 0, the minimum gray-level value; if the result is greater than the maximum allowed value of 63, it is reduced to 63. The result is stored into the output buffer, and the input pixel addresses incremented as specified by the user-supplied parameter, and the next output pixel is computed.

The program writes each output line onto the output tape as soon as it is computed. When vertical segmentation is required, the program stores the segments of each line temporarily on disk until a complete output line has been assembled. When all output lines have been computed for one block of input data, the program reads an additional block of data into core, after first moving to the top of core any lines from the bottom of the previous input block that are needed for computing additional output. Processing continues one block at a time until the entire output image is complete.

When segmentation into vertical strips is required, the program processes the first segments of all lines on the disk first, then the second, and so on. If the input data does not exceed about 4 million characters, the program will process the entire image from one loading of the disk. For larger images, the disk is reloaded as many times as necessary.

Execution time has three components: tape I/O, computation time, and disk I/O (if any). Tape I/O is normally a small fraction of the total, because the input and output tapes are read or written once without intermediate rewinds. Computation time is about 20 microseconds per output pixel and per weight for symmetric weight arrays, and about four times as long for nonsymmetric arrays; for increments other than one, the numbers of output lines and pixels per line will equal the input numbers divided by the increments.

When segmentation into vertical strips is required, disk I/O will require an additional 1 to 6 minutes for each 1 million input pixels, depending on how many strips are required. Example: Processing a 3200-x-4100 input image with a 20-x-20 symmetric weight array and output increments of 11 and 16 requires approximately 1 hour and 20 minutes.

4.5.2 Parameters

CONVOLVE requires six special parameters and a table of weights, in addition to the standard parameters that define the input image. These special parameters are:

1. NX - number of columns in full weight matrix
2. NY - number of rows in full weight matrix
3. INCRX - increment between output pixels

4. INCRY - increment between output lines

5. IDIV - quantity by which input weights are to be divided for normalization. If IDIV = 0, weights are divided by their sum.

6. ISYM - symmetry of weights
   0 = Nonsymmetric
   1 = Symmetric

7. Weights, beginning with top line of array and left-hand end of line. For ISYM = 0, NX times NY values must be supplied. For ISYM =1, only the upper (NY + 1)/2 rows and left-hand (NX + 1)/2 values in each row are entered.

NX can have a maximum value of 256. The product of NX and NY may not exceed about 2000 for a nonsymmetric matrix or 3500 for a symmetric matrix; these values correspond to square arrays approximately 45 x 45 and 60 x 60, respectively.

4.5.3 Input

CONVOLVE requires a single input image tape in standard IDAMS format.

4.5.4 Output

CONVOLVE generates a single output image in standard IDAMS format. For large images and weight arrays, the program requires disk storage for temporary output.

4.5.5 Examples

1. Simulation is desired of the averaging characteristics of a detector that weights the central 36 points of an 8-x-8 array equally, and weights the 28 boundary points only half as much. The distance between successive sampling centers along the line is five elements,

and that between lines is eight elements. The corresponding weight matrix can be represented as:

1	1	1	1	1	1	1	1
1	2	2	2	2	2	2	1
1	2	2	2	2	2	2	1
1	2	2	2	2	2	2	1
1	2	2	2	2	2	2	1
1	2	2	2	2	2	2	1
1	2	2	2	2	2	2	1
1	1	1	1	1	1	1	1

An entire input image named LARGEPIC, having 4000 lines of 4000 pixels each, is to be processed. Appropriate IDAMS task and parameter cards are:

```
CONVOLVE, (LARGEPIC, 49, 1), (1, 1, 4000, 4000), (SMALLPIC, 47, 1), 2
```

```
8  8  5  8  0  1  1  1  1  1  1  2  2  2  1  2
```

```
2  2  1  2  2  2
```

2. It is desired to sharpen edges and enhance high-frequency details of the upper right quarter of a 3000-x-3000 image named CHEBAY using a symmetric 11-x-11 filter described by the matrix whose upper left quadrant is:

2	0	-1	-4	-6	-6
0	-3	-6	4	15	19
-1	-6	12	20	13	5
-4	4	20	-10	-76	-60
-6	15	13	-76	-20	10
-6	19	5	-60	10	844

Appropriate IDAMS task and parameter cards are:

```
CONVOLVE, (CHEBAY, 49, 1), (1501, 1, 1500, 1500), (SHARPBAY, 47, 1), 3

11  11   1   1   0   1   2   0  -1  -4  -6  -6   0  -3  -6   4

15  19  -1  -6  12  20  13   5  -4   4  20 -10 -76 -60  -6  15

13 -76 -20  10  -6  19   5 -60  10 844
```

NOTE: Card format specifications are defined in the User's Guide. Parameters must be supplied in the order shown in paragraph 4.5.2.

4.5.6 Messages

Message	Explanation
SUM OF WEIGHTS = 0	User has specified weight normalization by dividing by sum of weights (IDIV parameter = 0) and this sum = 0, fatal error.
NY TOO LARGE FOR AVAILABLE CORE	Insufficient core to hold both weight table and NY data segments of minimum possible length, fatal error.
WEIGHT VALUES TOO LARGE	Sum of either positive or negative weights, after normalization, exceeded 32.5, making possible uncorrectable overflow, fatal error.

4.5.7 Flowchart

See Appendix C, Figure C-5.

4.6 EXPAND - IMAGE EXPANSION PROGRAM

4.6.1 Program Description

The task program EXPAND converts an input image into an expanded output image using a user-supplied table of weights. These weights can be chosen to simulate the spreading associated with a raster-type scanning or recording device, or to provide interpolation between input points.

The weight table represents the fraction of the value of an input pixel located at a position corresponding to the center of the table that will be contributed to output values having relative positions equivalent to the relative positions of the weights. In other words, an input image array containing all zeros, except for a single element equal to unity, would (ignoring round-off effects) give an output image containing a single copy of the weight table surrounded by zeros. If the dimensions NX, NY of the table exceed magnification factors MX, MY, there will be overlapping between the weight distributions centered on adjacent input points, so the value of an output pixel can be a sum of the weighted contributions of several input pixels.

When NX or NY is even, the center of the weight table is taken to be on column NX/2 or row NY/2, respectively; i.e., for an even-even weight table, the center is taken to be the upper left one of the four values that surround the geometric center. Similarly, the first pixel of the input array is taken to be at column (MX + 1)/2 or MX/2 and row (MY + 1)/2 or MY/2, according to whether MX and MY are odd or even, respectively.

EXPAND begins by dividing each user-supplied integer weight value by the specified divisor using 12-bit fractional precision. If a symmetric array is specified, the program generates the complete weight array by copying from the upper left quadrant supplied by the user. Because different output points will use different subsets of the weights, the program compares the sum of the positive members and negative members of each possible subset with 32.5 to ensure that no uncorrectable overflow can occur during computation of the output values.

The program then examines magnification factors MX and MY and the table dimensions NX and NY to determine whether the points on the edges of the output image will need to reference input data beyond the edges of the input image. If so, the program computes parameters to permit extending the input image by copying the edge points outwards; if the center of the weight array falls on a point outside the input image, however, the program reduces user-supplied dimensions NL and NP, and writes an advisory message.

The program then computes the remaining parameters required for reading in and processing the data. In particular, if the required input image is too large to fit into core, the program computes parameters to permit dividing the input into horizontal strips to be processed one at a time. A call to PXLPRM passes constants required by the COMPASS subroutine PXLBLD: PXLPRM also modifies PXLBLD logic, as required, by the relative values of MX and NY.

The program then reads data from the input tape until core storage is filled. It then calls PXLBLD to compute one line of output data at a time. To compute one output pixel, PXLBLD loads each input pixel that will contribute to the output value, multiplies it by the appropriate weight, and adds the results together. This sum is divided, with rounding, by 4096 to eliminate the 12-bit fractional part. If the result is negative, it is replaced by the minimum gray-level value, zero. If the result exceeds the maximum value of 63, it is replaced by 63. The result is stored into the output buffer, and the index register is initialized for computing the next output pixel.

After each output line is generated, the program writes it on the output tape. After all data in core have been used, the program reads additional strips of the image, if any, into core and processes them in the same manner.

Execution time is roughly proportional to the number of output pixels generated and the average number of input pixels that contribute to each output value; this number is roughly equal to (NX*NY)/(MX*MY). For 3000 output lines of 4000 pixels each, using about three input pixels for each output value, EXPAND requires about 30 minutes.

4.6.2 Parameters

EXPAND requires the following special parameters:

1. NX - number of columns of weight matrix
2. NY - number of rows of weight matrix
3. MX - factor by which image is to be expanded along the line, in x direction
4. MY - magnification factor in direction perpendicular to lines (y direction)
5. IDIV - integer by which each integer weight parameter is to be divided for normalization
6. ISYM - symmetry of weight table
   0 = nonsymmetric
   1 = symmetric about x and y axes
7. Weight parameters, beginning with top row and left-most value on row. For ISYM = 1, only (NX + 1)/2 or NX/2 values per row and (NY + 1)/2 or NY/2 rows need be supplied for NX and NY, respectively, odd or even.

The parameters are restricted by the requirements that: the length of the output line, MX*NP, cannot exceed 5000; and there must be room for at least 1 + (NY - 1)/MY lines of input in core at one time. Hence,

$$MX \leq 5000/NP$$

$$NY \lesssim (20000/NP - 1)*MY + 1$$

where NP is the number of pixels per line of input to be processed.

4.6.3 Input

EXPAND requires one input tape in standard IDAMS format.

4.6.4 Output

EXPAND generates one output tape file in standard IDAMS format.

4.6.5 Examples

1. Simulation is desired of the output of a moving-spot recorder that produces an intensity distribution, on 5-x-5 centers, described by the weight matrix.

1	3	5	7	9	10	9	7	5	3	1
2	5	10	15	18	20	18	15	10	5	2
3	7	12	18	23	25	23	18	12	7	3
2	5	10	15	18	20	18	15	10	5	2
1	3	5	7	9	10	9	7	5	3	1

Because this table is symmetric about the X and Y axes, appropriate task and parameter cards for processing a 500-x-500 image SMALLPIC would be:

```
EXPAND, (SMALLPIC, 49, 1), (1, 1, 500, 500), (XPND1, 48, 1), 2
```

```
11   5   5   5   31   1   1   3   5   7   9   10   2   5   10   15
```

```
18   20   3   7   12   18   23   25
```

2. Magnification is desired of a 1000-x-1000 region of an image PINERUST, by three in each direction, using bilinear interpolation. The required weight matrix is:

1/9	2/9	1/3	2/9	1/9
2/9	4/9	2/3	4/9	2/9
1/3	2/3	1	2/3	1/3
2/9	4/9	2/3	4/9	2/9
1/9	2/9	1/3	2/9	1/9

If the upper left corner of the desired region is at line 501 and pixel 351, appropriate task and parameter cards would be:

```
EXPAND, (PINERUST, 47, 1), (351, 501, 1000, 1000), (PRX, 49, 1), 1
```

```
5   5   3   3   9   1   1   2   3   2   4   6   3   6   9
```

NOTE: Card format specifications are defined in the User's Guide. Parameters must be supplied in the order shown in paragraph 4.6.2.

4.6.6 Message

Message	Explanation
WEIGHT DIVISOR IDIV EQUALS ZERO	Weight divisor parameter of zero cannot be processed, fatal error.
WGHT SUBSET NORMALIZED SUM TOO BIG	An uncorrectable overflow (output value more than 32.5 times input value) could occur with weights normalized as specified by the weight divisor parameter, fatal error.

Message	Explanation
NY AND NP ARE TOO LARGE FOR CORE	Insufficient core available to hold enough input lines for the size of weight table defined, fatal error.

4.6.7 Flowchart

See Appendix C, Figure C-6.

4.7 SHADE - PHOTOMETRIC CORRECTION PROGRAM

4.7.1 Program Description

The task program SHADE makes position-dependent gray-level corrections on an image using user-supplied calibration data. Assuming the relationship between the true gray level T and measured value M to be linear, at any point the true value can be computed from the observed value using the relationship

$$T = \text{Slope} * M + \text{Intercept} \tag{1}$$

Measuring the values M_1 and M_2 for two known gray levels T_1 and T_2 determines the slope and intercept. In terms of these data, the slope and intercept are given by

$$\text{Slope} = (T_2 - T_1) / (M_2 - M_1) \tag{2a}$$

$$\text{Intercept} = (M_2 T_1 - M_1 T_2) / (M_2 - M_1) \tag{2b}$$

In practice, the measurements of M_1 and M_2 are made conveniently at each point of a rectangular calibration grid using the same values of T_1 and T_2 at each point.

SHADE begins by checking that the user-specified region of the input image to be corrected is entirely enclosed by the calibration grid. It then uses the values of T_1 and T_2, and of M_1 and M_2 for each point on the calibration grid; they are entered as parameters to compute the slope and intercept at each grid point. It then computes the constants used to control the input and output of data and the processing loops; a call to the entry point SHAPRM sets constants required by the COMPASS subroutine SHADIT.

SHADE then processes the input data lying between the first two rows of the calibration grid. To compute the values of slope and intercept for points not coinciding with a calibration grid point, SHADE uses bilinear interpolation, by computing, for each set of four neighboring calibration points, the coefficients a, b, c, d, e, f, g, and h in the formulae.

$$\text{Slope} = a + bx + cy + dxy \qquad (3a)$$

$$\text{Intercept} = e + fx + gy + hxy \qquad (3b)$$

where x and y denote distances from the upper-left corner of the rectangle defined by the four neighboring grid points.

SHADE reads in input data one line at a time. It then computes the values of the combinations $(a + cy)$, $(b + dy)$, $(e + gy)$, and $(f + hy)$ for that line for each calibration rectangle and converts them to fixed-point representation with 16-bit fractional precision. SHADIT then processes the line of input data using equation (1); the values of slope and intercept for successive points are obtained by incrementing the initial values of $(a + cy)$ and $(e + gy)$ in steps of $(b + dy)$ and $(f + hy)$, respectively; the initial values are reset for a new rectangle each time a vertical column of the calibration grid is reached. The intensity-corrected line of data is then written on the output tape.

When all input data within one row of calibration rectangles have been corrected, SHADE computes the interpolation coefficients for the next row and processes data as described above. Processing continues until all input data have been corrected, after which control returns to the monitor program.

4.7.2 Parameters

SHADE requires the following special parameters:

1. INITX - x-coordinate (pixel number) of upper left point in calibration grid
2. INITY - y-coordinate (line number) of upper left point in calibration grid
3. INCRX - spacing (in pixels) between columns of calibration grid
4. INCRY - spacing (in lines) between rows of calibration grid
5. NX - number of columns in calibration grid (not greater than 21)

6.	NY	- number of rows in calibration grid (not greater than 441/NX)
7.	LEVEL1	- lower (lighter) true value of gray level used for calibration
8.	LEVEL2	- upper (darker) true value of gray level used for calibration: LEVEL2 > LEVEL1
9.	LMEAS	- NX times NY pairs of measured calibration values, beginning with top line of grid and left-most calibration point on line. First value in each pair corresponds to LEVEL1. Data are four-digit integers (0000 to 6300) representing 100 times measured values; i.e., with two implied decimal places.

The calibration grid must entirely enclose region of the input image to be processed; this means that the relations

$$\begin{aligned}
&SP \geq INITX \\
&SL \geq INITY \\
&NP \leq INITX + NX*INCRX - (SP - 1) \\
&NL \leq INITY + NY*INCRY - (SL - 1)
\end{aligned}$$

must be satisfied; execution will be aborted otherwise. However, SHADE will simply ignore additional rows or columns of calibration data not required for processing the specified input image, and will carry out processing normally.

The 16-bit fractional precision used by SHADIT will keep round-off errors to less than one-half gray level as long as INCRX is less than 512; because incrementing between lines is done with 36-bit floating-point precision, there is no restriction on the value of INCRY.

The computed values of slope and intercept for all calibration points must be between -127 and +127; execution will be terminated otherwise.

4.7.3 Input

SHADE requires one input tape in standard IDAMS format.

4.7.4 Output

SHADE generates one output tape in standard IDAMS format.

4.7.5 Example

Calibration measurements have been made at pixels 50, 300, 550, and 800, and lines 100, 400, 700 and 1000 using true gray levels of 16 and 48. The resulting measured values were:

(10,42)	(13,44)	(15,46)	(17,47)
(11,42)	(14,45)	(16,48)	(17,49)
(12,43)	(15,46)	(16,49)	(17,51)
(11,43)	(15,47)	(15,50)	(16,52)

Appropriate task and parameter cards for processing the region of an input image, CB207, between pixels 100 and 800 and lines 100 and 900 are

```
SHADE, (CB207, 49, 1), (100, 100, 701, 801), (CB207CR, 47, 1), 3
```

```
50  100  250  300  4  4  16  48  1000  4200  1300  4400  1500  4600  1700  4700
```

```
1100 4200 1400 4500 1600 4800 1700 4900 1200 4300 1500 4600 1600 4900 1700 5100
```

```
1100 4300 1500 4700 1500 5000 1600 5200
```

NOTE: Card format specifications are defined in the User's Guide. Parameters must be supplied in the order shown in paragraph 4.7.2.

4.7.6 Messages

SHADE generates the following diagnostic messages:

Message	Explanation
NX EXCEEDS 21	More than 21 columns of calibration points were specified; execution terminates.

Message	Explanation
NX*NY EXCEEDS 441	More than 441 calibration points were specified; execution terminates.
SOME DATA LIES OUTSIDE CALIB GRID	Specified region of input image was not entirely enclosed by calibration grid; execution terminates.
INVALID CALIBRATION DATA	INCRX or INCRY was not positive, LEVEL2 was not greater than LEVEL1, or second number of a pair of measured values was not greater than first number; execution terminates.
SLOPE/INTERCEPT OVFL INPT VALUE NNNN	Value of slope or intercept computed from pair NNNN of calibration data was outside the range -127 to +127; execution terminates.
COEFF OVFL, CAL GRID ROW MMMM COL NNNN	An interpolation coefficient associated with row MMMM and column NNNN of the calibration rectangles was less than -127 or greater than +127; execution terminates.

4.7.7 Flowchart

See Appendix C, Figure C-7.

4.8 FFT - FAST FOURIER TRANSFORM PROGRAM

4.8.1 Program Description

This task routine performs a one- or two-dimensional Fourier transform on a complex array of not more than 512 rows and 512 columns, which is stored on disk.

FFT uses four subroutines. Subroutine TRIGGN generates a sine-cosine table. Subroutine PERGEN generates a table of integers to place the complex data array in permuted order, such that the transformed values will be in normal sequence. Subroutine FFTONE performs a one-dimensional Fast Fourier transform on the permuted complex data array. Subroutine FLIP transposes the transformed values of a one-dimensional FFT to prepare the complex data for a two-dimensional transform. It flips the final result of the second transform again to place the transformed values in the original array order.

FFT begins by calling subroutine TRIGGN to generate a table of sines of angles between 0 and $\pi/2$ in steps of $2\pi/NX$, where NX is the number of columns in the array. If the user has specified a negative sign for the complex exponential, negatives replace the resulting values.

A table of index permutations is then generated by calling PERGEN. This routine basically computes the numbers obtained by reversing the bit-order of each binary number between 0 and N-1; it modifies the results to take into account FORTRAN indexing methods, including the use of two floating-point numbers to represent each complex value in the array.

The data are then read into core one line at a time. FFT calls the subroutine FFTONE to carry out the one-dimensional fast Fourier transform on the line. FFTONE begins by computing the appropriate normalization factor. It then permutes the complex values to reverse the bit-order of their binary indices. Before they are stored into their new locations, they are each multiplied by the normalizing factor. The line is then transformed using the one-dimensional

Fast Fourier transform algorithm, and control returns to the main routine, which writes the line back onto the disk.

After all lines have been processed, the routine examines the user-supplied parameter IDIM to determine whether a two-dimensional transform is requested. If so, the routine interchanges the rows and columns of the array on the disk by calling FLIP. If the number of rows and columns are not the same, FLIP interchanges these parameters and computes new tables of sines and permutations. FFTONE then processes the rotated array in the same way as before. When the entire array has been transformed, another call to FLIP interchanges the rows and columns again. When processing is complete, control returns to the monitor.

FFT execution time for an N*M complex array is approximately N*M $\log_2$ N*M milliseconds for a two-dimensional transform. For example, a 64*64 array requires about one minute for transformation.

4.8.2 Parameters

1. MX - $\log_2$ NX, where NX is the number of columns in the data array (NX must be a power of 2)

2. MY - $\log_2$ NY, where NY is the number of rows in the data array (NY must be a power of 2)

3. IDIM - dimension of FFT required

   1 = perform FFT along rows only

   2 = perform FFT along rows and columns

4. ISIGN - sign of exponential function in transform

   -1 = use negative sign (normally used for transform from image space to frequency space)

   +1 = use positive sign (normally used for the inverse transform)

4.8.3 Input

FFT requires a complex data array on disk, 16 complex words/cell, beginning in cell 1.

4.8.4 Output

FFT places the transformed values on disk.

4.8.5 Examples

An array of 32-x-32 complex data words has been stored on disk. A two-dimensional inverse Fourier transform is to be performed on this data. The parameters required are MX = $\log_2$ NX = 5, MY = $\log_2$ NY = 5, IDIM = 2, ISIGN = -1 as shown in the following card layout:

```
FFT,,,,1
```

```
5    5    2    -1
```

NOTE: Card format specifications are defined in the User's Guide. Parameter must be supplied in the order shown in paragraph 4.8.2.

4.8.6 Messages

FFT may generate the following message:

Message	Explanation
ARRAY SIZE TOO LARGE	The complex data array is larger than 512 x 512, execution terminates.

NOTES: 1. FFT assumes that the origin of coordinates is in the upper left corner of the input array (the y axis is positive downwards), and leaves the origin of coordinates in the same place in the input. The user must specify the necessary conversions when transferring data between disk and tape if the image on tape is to have the origin at the center of the array.

2. A symmetric normalization has been assumed, whereby the Fourier transform is

$$a_{k\ell} = \frac{1}{\sqrt{NX*NY}} \sum_{m=0}^{NX-1} \sum_{n=0}^{NY-1} f_{mn} e^{-2\pi i(km/NX+\ell n/NY)}$$

and the inverse transform is

$$f_{mn} = \frac{1}{\sqrt{NX*NY}} \sum_{k=0}^{NX-1} \sum_{\ell=0}^{NY-1} a_{k\ell} e^{+2\pi i(km/NX+\ell n/NY)}$$

4.8.7 Flowchart

See Appendix C, Figure C-8.

4.9 FPCON - FLOATING-POINT CONVERSION PROGRAM

4.9.1 Program Description

The task program FPCON carries out conversions between the various floating-point and scaled (six-bit character) representations of image data, the Fourier components of an image, and associated power spectra and autocorrelation arrays.

The following floating-point representations can occur:

1. Full array of real image
2. Full array of real values representing modulus or squared modulus (power spectrum) for the Fourier transform of an image or an associated autocorrelation function; two arrangements are possible:
   a. Origin (zero frequency) at corner of array; this is the normal format for output from or input to the Fast Fourier transform
   b. Origin at center of array
3. One-half array of complex Fourier components for an image (or other real array) in real-plus-imaginary form; the remaining half of the array can be reconstructed using symmetry properties, as described in Appendix D
4. One-half array of complex Fourier components in modulus-plus-phase form
5. Packed complex representation of Fourier components, as described in Appendix D

The routines generating them normally place these representations on disk, but they can be transferred to and from tape. The program uses the first four floating-point words on the disk cell immediately following the last data record to store the maxima (words 1 and 3) and minima (words 2 and 4) of the real and imaginary parts, respectively (or modulus and phase), of the complex arrays. For real arrays the program sets words 3 and 4 to zero. FPCON does not provide conversions from packed representation (5.); instead, the routine CXPACK must be used to convert between (3.) and (5.).

The scaled (six-bit) representations handled by FPCON are:

6. Real image array

7. Full array with origin (zero frequency) at center representing first part (modulus or real part) of complex values for the Fourier transform; two types of scaling are possible

   a. Linear

   b. Logarithmic

8. Full array with origin at center representing second part (phase or imaginary part) of complex values for Fourier components; the same two types of scaling are possible

The scaled representations do not reside on disk; hence, conversions to or from floating point are required for transferring scaled data from or to tape.

Table 4-1 lists the possible conversion and transfers provided by FPCON.

Main routine FPCON accesses the size parameters, reads a requested transfer/conversion code ranging from 1 to 22, and passes control to a multiple-entry subroutine to carry out the required processing. When control returns to FPCON, it reads the next code. If a valid code is found, it again passes control to the appropriate subroutine. If the code is a zero, or if it is invalid, FPCON terminates and returns control to the monitor.

Entries F01CN to F06CN set parameters to identify the entry point and then enter a subroutine that handles all data transfers from tape to disk. For the first five entries, the program creates a conversion table from the 64 different

Table 4-1. FPCON Conversion and Transfers

CODE	CONVERSION OR TRANSFER
1	6-BIT IMAGE DATA ON TAPE TO FLOATING POINT ON DISK
2	6-BIT LINEAR SCALED DATA ON TAPE TO FIRST PART OF COMPLEX FLOATING-POINT VALUES FOR SYMMETRIC HALF ARRAY ON DISK
3	6-BIT LINEAR SCALED TO SECOND PART OF COMPLEX VALUES
4	6-BIT LOGARITHMIC SCALED DATA ON TAPE TO FIRST PART OF COMPLEX VALUES FOR SYMMETRIC HALF ARRAY ON DISK
5	6-BIT LOGARITHMICALLY SCALED TO SECOND PART OF COMPLEX VALUES
6	FLOATING POINT ARRAY TRANSFERRED FROM TAPE TO DISK
7	COMPLEX MODULUS-PLUS-PHASE SYMMETRIC HALF ARRAY ON DISK TO REAL-PLUS-IMAGINARY SYMMETRIC HALF ARRAY ON DISK
8	FULL ARRAY OF MODULUS VALUES ON DISK TO SQUARED MODULUS VALUES ON DISK
9	FULL ARRAY ON DISK WITH ORIGIN AT CENTER TO FULL ARRAY ON DISK WITHIN ORIGIN AT CORNER
10	SYMMETRIC COMPLEX HALF ARRAY ON DISK IN MODULUS-PLUS-PHASE REPRESENTATION TO FULL ARRAY ON DISK OF MODULUS VALUES WITH CORNER ORIGIN
11-19	INVERSE OF 1-9, RESPECTIVELY
20	SYMMETRIC COMPLEX HALF ARRAY ON DISK IN MODULUS-PLUS-PHASE REPRESENTATION TO FULL ARRAY ON DISK OF SQUARED MODULUS VALUES WITH CORNER ORIGIN
21	DISK TO PRINTER FLOATING POINT LISTING
22	TAPE TO PRINTER FLOATING POINT LISTING

six-bit values to floating point, using constants stored in the label records for codes 2 through 5. The program then computes constants controlling the number of characters to be read from each tape and the manner of storage into the floating point line to be stored on disk. Then the program reads in data, one line at a time, and converts them to floating points for codes 1 through 5. For codes 2 through 5, the program reads in the existing line on disk before conversion begins, so both the first and second parts of the complex word can be filled in by two successive calls to FPCON, using different input data files. For these codes the program shifts the origin (zero frequency point) of the array from the center to the corner, and stores the data as one-half a symmetric array. The program then writes the floating-point line onto disk before processing the next input line.

Entires F07CN, F08CN, F17CN, and F18CN set parameters to identify the entry point, and then enter a subroutine that converts one line of disk data at a time between real-plus-imaginary and modulus-plus-phase complex floating-point format or between modulus and squared-modulus real floating-point values. The complex data are in the format of one-half a symmetric array; the conversion takes into account the special packing of the values along the symmetry axes, as described in Appendix D.

F09CN and F19CN enter a subroutine that reads pairs of lines from disk, one each in the upper and lower half of the array, interchanges the right-hand and left-hand ends of each line, and then stores the lines back on disk in interchanged positions. After all lines have been so processed, the program has transferred the origin of the array from corner to center, or vice-versa.

Entries F10CN and F20CN set parameters to identify the entry point and then enter a subroutine for converting a complex modulus-plus-phase array to a full real array, with corner origin, of modulus or squared modulus values. The program reads one pair of symmetrically located lines from disk at a time, and constructs two lines of the full real array from them using the symmetry of the complex array. The program processes additional pairs of lines until the entire array has been converted.

Entries F11CN to F16CN set parameters to identify the entry point and then enter a subroutine for transferring data from disk to tape. For entry code 16, the program checks to see whether maxima and minima of the array on disk have been previously determined; if not, it sets parameters for examining each floating-point value as it is transferred to find maxima and minima. For entry codes 11 through 15, the program computes a table of threshold values for converting from floating-point to six-bit scaled values. It then computes parameters for accessing the required words on disk and storing them in output lines for tape. The program reads data from disk one line at a time. For codes 11 through 15, the program carries out a set of six comparisons of each floating-point value against threshold values to determine the six bits of its character representation. For codes 11 and 16 the program writes the output line directly onto tape; for codes 12 through 15 it stores the output line on a scratch region of disk. After all floating-point data have been processed, the program reads back the scratch data for codes 12 through 15 into core in pairs of symmetrically located lines. The program generates a full array with center origin one line at a time, and writes it onto disk. For codes 11 to 15, it writes the floating-point value for gray level 0 and the floating-point increment between levels on the tape level record; for code 16, it records maximum and minimum values instead.

FPDUMP is entered for codes 21 and 22. If code 21 is specified, NY lines of NX floating-point values each is read from disk cell 1 and listed on the line printer. For entry code 22, NY records of NX floating-point values are read from tape, after reading the label, and dumped to the line printer.

4.9.2 Parameters

In addition to the parameters (SP, SL, NP, NL) specifying the region of the input image, if any, to be used, the program requires the following special parameters:

1. NX - number of complex values per line of packed array on disk. NX is one-half the number of pixels per line of a real image, scaled (six-bit) array, and power-spectrum or autocorrelation array. NX must be a power of 2, $2^2 \leq NX \leq 2^9$.

2. NY - number of lines in array. NY must be a power of 2; $2^0 \leq NY \leq 2^9$.

3. ICODE - one to five integers, each specifying a transfer/conversion step.

4.9.3 Input

Table 4-2 lists the input data required for the various transfer/conversion options.

4.9.4 Output

Table 4-3 lists the output data created by the various transfer/conversion options.

4.9.5 Examples

A section of input tape, TEST1, consisting of the first 256 lines and 512 pixels per line is to be converted to floating-point representation on disk, then back to six-bit image data on an output tape. The task and parameter cards required are:

```
FPCON, (TEST1, 49, 1), (1, 1, 512, 256), (FPCON1, 47, 1), 1
```

```
256     256        1       11
```

Table 4-2. FPCON Input Data

CODE	INPUT
1	IMAGE TAPE IN STANDARD IDAMS FORMAT
2-5	IMAGE TAPE, REPRESENTING SCALED FLOATING-POINT DATA, IN STANDARD IDAMS FORMAT
6	TAPE FILE CONTAINING A FLOATING-POINT ARRAY, LABELLED IN STANDARD IDAMS FORMAT
7,10,20	DISK FILE CONTAINING ONE-HALF OF SYMMETRIC COMPLEX ARRAY IN MODULUS-PLUS-PHASE REPRESENTATION
8	DISK FILE CONTAINING FULL ARRAY OF REAL VALUES, NORMALLY REPRESENTING MODULUS
9	DISK FILE CONTAINING FULL ARRAY OF REAL VALUES, NORMALLY WITH ORIGIN AT CENTER
11	DISK FILE CONTAINING FULL ARRAY OF REAL IMAGE VALUES
12-15	DISK FILE CONTAINING ONE-HALF OF SYMMETRIC COMPLEX ARRAY IN EITHER MODULUS-PLUS-PHASE OR REAL-PLUS-IMAGINARY REPRESENTATION
16	DISK FILE CONTAINING A FLOATING-POINT ARRAY
17	DISK FILE CONTAINING ONE-HALF OF SYMMETRIC COMPLEX ARRAY IN REAL-PLUS-IMAGINARY REPRESENTATION
18	DISK FILE CONTAINING FULL ARRAY OF REAL VALUES, NORMALLY REPRESENTING SQUARED MODULUS
19	DISK FILE CONTAINING FULL ARRAY OF REAL VALUES, NORMALLY WITH ORIGIN AT UPPER LEFT CORNER
21	DISK FILE CONTAINING A FLOATING-POINT ARRAY
22	TAPE FILE CONTAINING A FLOATING-POINT ARRAY, LABELLED IN STANDARD IDAMS FORMAT

Table 4-3. FPCON Output Data

CODE	OUTPUT
1	DISK FILE CONTAINING FULL ARRAY OF REAL IMAGE VALUES
2-5	DISK FILE CONTAINING ONE-HALF OF SYMMETRIC COMPLEX ARRAY IN EITHER MODULUS-PLUS-PHASE OR REAL-PLUS-IMAGINARY REPRESENTATION
6	DISK FILE CONTAINING A FLOATING-POINT ARRAY
7	DISK FILE CONTAINING ONE-HALF OF SYMMETRIC COMPLEX ARRAY IN REAL-PLUS-IMAGINARY REPRESENTATION
8	DISK FILE CONTAINING FULL ARRAY OF REAL VALUES, NORMALLY REPRESENTING SQUARED MODULUS
9	DISK FILE CONTAINING FULL ARRAY OF REAL VALUES, NORMALLY WITH ORIGIN AT UPPER LEFT CORNER
10	DISK FILE CONTAINING FULL ARRAY OF REAL MODULUS VALUES WITH ORIGIN AT UPPER LEFT CORNER
11	IMAGE TAPE IN STANDARD IDAMS FORMAT
12-15	IMAGE TAPE, REPRESENTING SCALED FLOATING-POINT DATA, IN STANDARD IDAMS FORMAT
16	TAPE FILE CONTAINING A FLOATING-POINT ARRAY, LABELLED IN STANDARD IDAMS FORMAT
17	DISK FILE CONTAINING ONE-HALF OF SYMMETRIC COMPLEX ARRAY IN MODULUS-PLUS-PHASE REPRESENTATION
18	DISK FILE CONTAINING FULL ARRAY OF REAL VALUES, NORMALLY REPRESENTING MODULUS
19	DISK FILE CONTAINING FULL ARRAY OF REAL VALUES, NORMALLY WITH ORIGIN AT CENTER
20	DISK FILE CONTAINING FULL ARRAY OF SQUARED MODULUS VALUES WITH ORIGIN AT UPPER LEFT CORNER
21	DISK FILE TO PRINTER LISTING, FLOATING POINT
22	TAPE FILE TO PRINTER LISTING, FLOATING POINT

C8

Convert an array of complex real-plus-imaginary values on disk to complex modulus-plus-phase representation on disk. Control cards are:

```
FPCON,,,,1
```

```
64    16    17
```

Convert six-bit image data to floating point on disk. Treat it as a full real array with center origin and shift to corner origin. Control cards are:

```
FPCON, (TEST1, 49, 1), (1, 1, 128, 16),, 1
```

```
64    16    1    9
```

Assume six-bit log scaled data is stored on tape and is to be converted to the second half-word of a complex array on disk. This step is then to be followed by another conversion to six-bit linear scaled data on tape. The following control cards are necessary:

```
FPCON, (LOGDATA, 49, 1), (1, 1, 128, 16), (LINDATA, 47, 1), 1
```

```
64    16    5    13
```

NOTE: Card format specifications are defined in the User's Guide. Parameters must be supplied in the order shown in paragraph 4.9.2.

4.9.6 Messages

FPCON generates the following diagnostic messages:

Message	Explanation
TRANSFER/CONVERSION CODE LT 1 OR GT 22	An invalid option was specified on input; execution terminates.

Message	Explanation
NL OR NP NOT AVAILABLE ON INPUT	The number of lines or number of pixels to be processed is not available on the input tape; execution terminates.

4.9.7 Flowchart

See Appendix C, Figure C-9.

4.10 SMOOTH - FLOATING-POINT SMOOTHING PROGRAM

4.10.1 Program Description

This task routine convolves an array of real floating-point values stored on disk with a user-supplied, symmetric 3-x-3 or 5-x-5 set of smoothing weights, and leaves the smoothed array in the same place on disk.

SMOOTH begins by converting the input weights from integer to floating-point format. It then computes a series of constants required for processing. In particular, it checks whether a sufficient number of full lines can be held in core at one time to permit efficient processing; if not, it arranges to divide the lines into segments of 128 values each.

SMOOTH then reads data into core until it is filled. For data at the edge of the input array, it obtains an extension of one or two values all around by copying the boundary values outwards. It then convolves the data with the smoothing weights, using the extension values as necessary to provide an output array of the same size as the input. It writes the smoothed data back on disk.

Some of the data already in core will contribute to the next set of output; for this reason, SMOOTH moves these data to the top of the core region before it reads in more data. If segmentation is required, it stores some lines temporarily on a scratch area of the disk, and reads lines stored previously back into core. Then it reads additional data from the input array to again fill core, or until the last input line has been used. It again convolves the data in core with the smoothing weights, and writes the results back onto disk. This process continues until the entire image has been processed.

Execution time is about 5 milliseconds for each floating-point value convolved using a 3-x-3 weight array and about 10 milliseconds for 5-x-5 weights; for example, convolving 32 lines of 512 values each with a 5-x-5 weight array requires about three minutes.

4.10.2 Parameters

1.	NX	- a positive integer, the number of real values per line of data on disk
2.	NY	- a positive integer, the number of lines of data on disk
3.	IDIM	- dimension of square weight array, where 3 = 3-x-3 weights 3 = 5-x-5 weights
4.	IDIV	- number by which IWGHTS values will be divided to create floating-point weighted array. This value can never be zero.
4.	IWGHTS	- Four (IDIM = 3) or nine (IDIM = 5) values representing weights in upper-left quadrant of array. These are integers that will be divided by IDIV and converted to floating-point numbers.

NOTE: The value of NX for SMOOTH is the number of floating-point values per line, that is, twice the value defined for FFT and FPCON, which refers to the number of complex values per line.

4.10.3 Input

Input for SMOOTH is a floating-point array stored on disk beginning in cell number 1. If NX is not an exact multiple of 32, the remainder of the last cell for each line is not used; i.e., each line begins in a new cell.

4.10.4 Output

SMOOTH returns the smoothed values to the same location on disk from which they were input.

4.10.5 Example

It is desired to smooth a floating-point array containing 32 lines of 512 points each using a symmetric 5-x-5 weight matrix. The matrix will be weighted as follows:

$$\begin{array}{ccccc} 1 & 2 & 3 & 2 & 1 \\ 2 & 4 & 6 & 4 & 2 \\ 3 & 6 & 9 & 6 & 3 \\ 2 & 4 & 6 & 4 & 2 \\ 1 & 2 & 3 & 2 & 1 \end{array}$$

Appropriate IDAMS task and parameter cards are:

```
SMOOTH,,,,1
```

```
512    32    5    81    1    2    3    2    4    6    3    6    9
```

NOTE: Card format specifications are defined in the User's Guide. Parameters must be supplied in the order shown in paragraph 4.10.2.

4.10.6 Messages

SMOOTH generates the following diagnostic messages:

Message	Explanation
TABLE DIMENSION NOT 3 OR 5	The parameter defining the dimension of the matrix in the smoothing process was not a 3 or 5; execution terminates.
NX OR NY ZERO OR NEGATIVE	The parameter NX or NY, defining the number of points per line or number of lines, was input as zero or a negative value; execution terminates.

Message	Explanation
WEIGHT TABLE DIVISOR LE ZERO	The parameter IDIV by which the weight matrix will be divided was input as zero; execution terminates.

4.10.7 Flowchart

See Appendix C, Figure C-10.

4.11 CXPACK - COMPLEX PACKING AND UNPACKING PROGRAM

4.11.1 Program Description

This task routine provides packing and unpacking of complex data arrays using the formulae given in Appendix D.

After accessing the input parameters, CXPACK calls subroutine TRIGGN (described as part of the FFT package) to generate a table of sines of arguments $2\pi\ I/N$ for $0 \le I \le N/4$, where $N = 2 * NX$ is the number of columns in the complete unpacked array. The routine then computes constants required for accessing the disk file and processing the lines of data; if unpacking is requested, it initializes values of minimum and maximum for both the real and imaginary parts.

The routine reads in the first line of data, corresponding to $k = 0$ (see Appendix D). Because the arrays are periodic in k with period M, the line for M - k is identical, and need not be read. CXPACK calls a special routine to fill the first complex word and also word NX/2 + 1. It then initializes, indexes, and computes the remaining words in the output line, using the symmetry relationships. For each word, the routine obtains the corresponding sine and cosine values by table lookup in the table previously generated by TRIGGN; if unpacking has been requested, the routine reverses the sign of the cosine. It then completes the computation of each symmetric pair of complex values using the same coding for either packing or unpacking. It then writes the line back onto disk.

CXPACK reads successive lines of data in symmetric pairs. After a special step to compute the first complex word of each output line, it initializes indexes so the remaining values can be computed using the same coding used for the first line.

When all symmetric pairs of lines have been processed, CXPACK reads in and processes the last line in the same manner as the first line; control returns to the monitor.

If data are being unpacked, CXPACK searches each output line for new maxima and minima before writing it onto the disk. After all lines have been processed, it writes the two maxima and minima for the entire array onto the disk cell, immediately following the last data line, before it returns control to the monitor.

4.11.2 Parameters

1.	MX	- $\log_2$ of number of complex words per line of packed array
2.	MY	- $\log_2$ of number of lines in array
3.	IUNPCK	- specifies conversion required: 0 = pack; 1 = unpack

4.11.3 Input

Input for CXPACK is a floating-point complex array on disk, beginning in cell number 1.

4.11.4 Output

CXPACK puts the packed or unpacked result back onto the disk in the same location as the original data.

4.11.5 Examples

An array of floating-point data packed into the real and imaginary parts of complex words is now stored on disk. The complex array size is 512 lines of 256 complex words per line. The following control cards will unpack this data, leaving one-half the resultant array in the location of the original input (the second half of the array is symmetric with the first and, therefore, discarded):

```
CXPACK,,,,1
```

```
8  9  0
```

NOTE: Card format specifications are defined in the User's Guide. Parameters must be supplied in the order shown in paragraph 4.11.2.

4.11.6 Messages

CXPACK generates no special diagnostic messages.

4.11.7 Flowchart

See Appendix C, Figure C-11.

4.12 ERROR - MESSAGE PROCESSING PROGRAM

4.12.1 Program Description

ERROR is always called at the end of a run for normal terminations, or when a fatal error occurs during execution of a sequence of IDAMS tasks. Its primary function is to interpret the system error code and use the result to build a message array for display on an appropriate output device.

When entered, ERROR initializes constant and task names needed for creating the message. Because all messages are stored in subroutine GETMSG, this program is called to pick up the first line: "FATAL ERROR IN AAAAAAAA." ERROR inserts the proper task name in place of AAAAAAAA. If the job has ended normally, the program replaces this line with NORMAL END OF JOB, and ERROR then writes out the message and returns.

If a subroutine had been called from the task program, the program puts the message "AAAAAAAA WAS IN EXECUTION WHEN ERROR OCCURRED" into the message array, and replaces AAAAAAAA with the actual subroutine name. It then generates the next two lines, containing the messages "ERROR CODE AND MEANING FOLLOW" and "IEROR = nnnn." The last line is an interpretation of what the error code means. Breaking the error code down into a task pointer and a displacement generates this line. ERROR then calls GETMSG to pick up the proper message, writes out the message array, and returns to DRIVER.

4.12.2 Parameters

ERROR requires no parameters.

4.12.3 Input

There is no external input to ERROR.

4.12.4 Output

ERROR writes a message block on the console typewriter and line printer.

4.12.5 Examples

A typical fatal error message produced by ERROR is:

```
FATAL ERROR IN EXPAND
ERROR CODE AND MEANING FOLLOW
IEROR = 6003
END OF FILE ON TAPE
```

4.12.6 Messages

Error produces no messages other than the error message array.

4.12.7 Flowchart

See Appendix C, Figure C-12.

4.13 REDUCE - IMAGE REDUCTION PROGRAM

4.13.1 Program Description

The task program REDUCE, will reduce a standard IDAMS image by an integral factor computed from input parameters. Blank fill characters are provided on both sides and/or top and bottom, if needed to complete the requested output image. The results are then written on the specified output tape.

Upon entry, the program computes the largest integral reduction factor. An output line buffer is set up with edge fill characters, if any, and the output label record is written. If any fill lines are required at the top of the output picture, they are written out at this time. The program then enters a main processing loop to read an input line, reduce its length by the reduction factor, and store it in an internal array. This process continues until enough input lines are collected to form one output line. Averaging is then performed between lines, the completed line is moved to the output buffer, and the results are written at the bottom of the output picture, if needed. The program then returns to the monitor.

4.13.2 Parameters

1. NPO = Number of pixels to be output
2. NLO = Number of lines to be output
3. IFILL = Gray level for edge fill (default = 0)

4.13.3 Input

REDUCE requires a single input image tape in standard IDAMS format.

4.13.4 Output

REDUCE generates a single output image tape in standard IDAMS format.

4.13.5 Example

A 400 line by 1000 sample portion of input image BIGPIC is to be reduced to fit a 500 by 500 output requirement. This implies a reduction factor of 2 in both the number of data lines and pixels. Therefore, the resultant output image will contain a 200 line by 500 pixel image data area preceded and followed by 150 lines of fill characters. No fill is necessary along the left and right edges. The following IDAMS source statements would be appropriate:

```
REDUCE, (BIGPIC, 49, 1), (1, 1, 1000, 400), (OUTPIC, 47, 1), 1
```

```
500     500
```

4.13.6 Messages

None.

4.13.7 Flowchart

See Appendix C, Figure C-13.

4.14 HISTO - HISTOGRAM AND STATISTICS PROGRAM

4.14.1 Program Description

This task program reads an IDAMS format image tape and produces a printed listing of both a numeric table of intensity frequencies and a histogram in graphic form. The mean, median, and standard deviation are also provided.

The program first examines standard input parameters to determine the portion of the input picture for which statistics are to be gathered. A set of 64 counters, one for each possible gray-level value, is initialized with zero counts. A line of data is then read, each value is examined, and the corresponding counter incremented by 1. Additional lines are similarly processed until the specified input data have been exhausted. The array of counters is then examined to determine the maximum value, and a graphing interval is determined such that the tallest bar in the histogram will just fit on the printer page.

After the data has been normalized, a header line is printed. Next, the bin values are printed as columns of X's where each X represents a percentage of the total. The value assigned to each X is given in the header line.

Following the histogram, a table of the exact frequency counts for each of the 64 bins will be printed. In addition, the mean, median, and standard deviation about the mean will be provided.

4.14.2 Parameters

There are no special parameters. Required standard system parameters are task name, input file name, input file number, starting line, starting pixel, number of lines, and number of pixels.

4.14.3 Input

HISTO requires a single input image tape in standard IDAMS format.

4.14.4 Output

HISTO prints out a histogram of gray levels contained in an input image, followed by a table of actual frequency counter values for the same data. The mean, median, and standard deviation are also printed.

4.14.5 Examples

None.

4.14.6 Messages

HISTO generates no special messages.

4.14.7 Flowchart

See Appendix C, Figure C-14.

4.15 CHAROUT - PIXEL CHARACTER OUTPUT PROGRAM

4.15.1 Program Description

The task program CHAROUT converts a selected portion of an IDAMS input image to alphanumeric format. Pixel data is translated into characters using a table look-up technique. The 64-position conversion table to be used is stored internally or, optionally, input as parameters. The standard internal table contains 32 unique characters, providing a different output character for every two gray levels. There are no restrictions on the user-supplied table values. For printed output, data is listed in block format. If the length of the requested output line exceeds the maximum printer line length, the program will print the data as a sequence of vertical strips of the specified picture. If an output tape is named on the task card, data will go to tape instead of the printer.

When entered, the program generates the necessary conversion table from either input parameters or the stored data. A line of input picture data is then read into core and CODE is called to translate the line into the output character representation. The converted line is then written either on tape or the line printer. Lines are read, translated, and written until the requested portion of the input picture has been processed.

4.15.2 Parameters

The alphanumeric conversion table can be specified in one of two ways. If only a task card is supplied, use of the internally stored table is assumed. If a parameter card is provided, the first 64 columns are used to fill the table of translation values.

4.15.3 Input

CHAROUT expects a single input image tape in standard IDAMS format.

4.15.4 Output

CHAROUT either prints a character representation of the input data on the line printer or writes an output IDAMS format tape.

4.15.5 Messages

None.

4.15.6 Example

Test tape TEST1 is to be printed on the line printer in a character format. The following control cards are typical:

```
CHAROUT, (TEST1, 49, 1), (1, 1, 340, 270), ,1

ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ01
```

4.15.7 Flowchart

See Appendix C, Figure C-15.

4.16 TEXTGN - TEXT GENERATOR PROGRAM

4.16.1 Program Description

The task program TEXTGN is used to generate an alphanumeric text for output directly onto a blank tape or to be combined with an input image tape for output.

The character set of the text consists of the letters A through Z, numerals 0 through 9, and special characters = + - / * : ( ) , . and blank. The text is input from cards. Each card represents a line of the text. Each character in the card is decoded into a 5-character by 7-line (8 x 11 counting spacing) matrix of gray and zero levels. (See Figure 4-3.)

Seven image lines are written one at a time to output after one text card is processed. Where a text is to be combined with an image tape, the original image is preserved as much as possible.

The program terminates after all cards are processed.

4.16.2 Parameters

The task card parameters for starting line, starting pixel, number of lines, and number of pixels are used to select a portion of the input picture for output. If no input picture is specified, the number of lines and number of pixels fields are used to define the size of the output image.

In addition, the following special parameters are necessary:

1. Starting pixel for left edge of test line(s)
2. Starting line for top edge of first line of text
3. Multiplication factor for text data (default = 1)
4. ,... Variable number of text lines

4.16.3 Input

Text cards and an IDAMS image tape if text is to be combined for output.

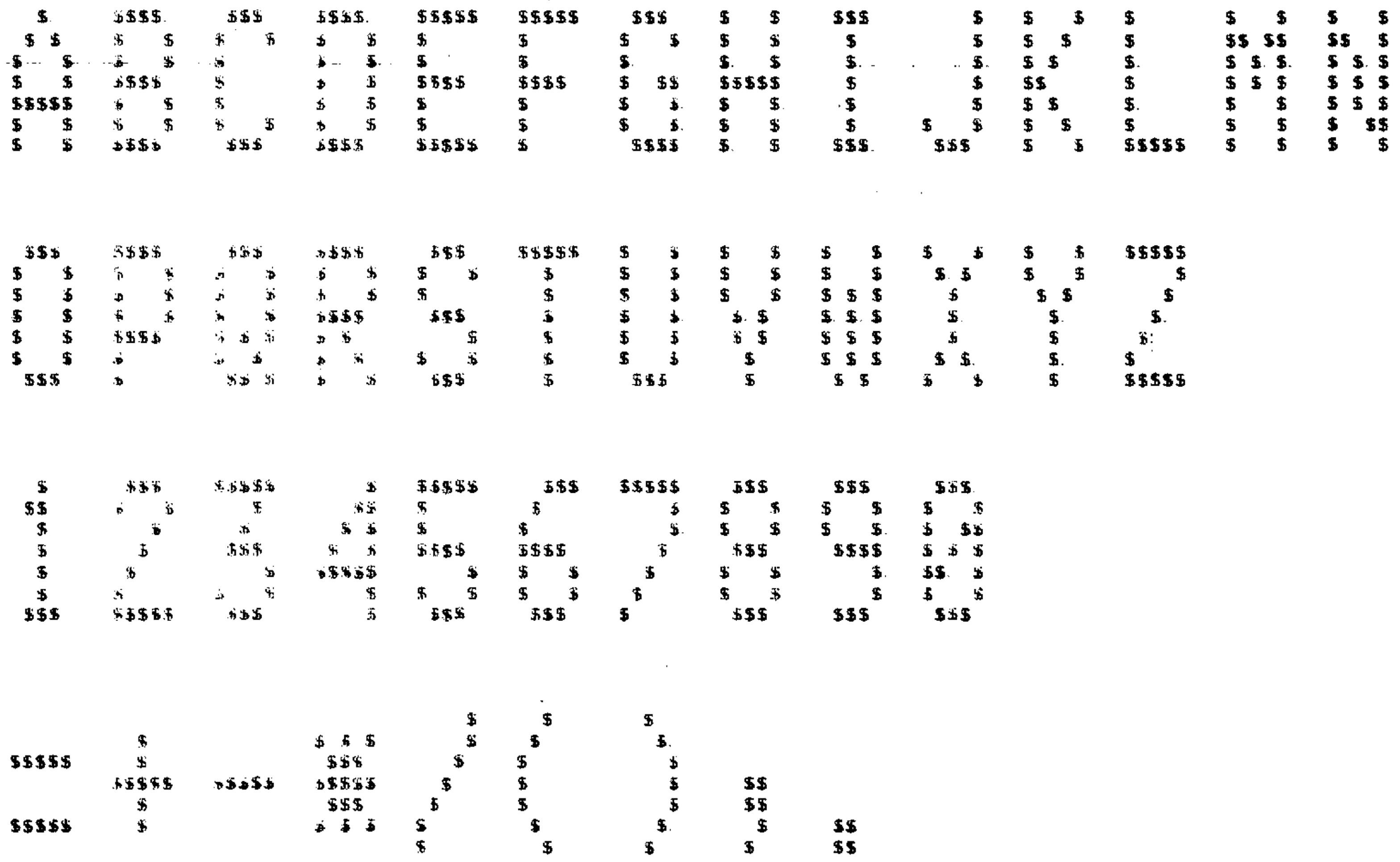


Figure 4-3. TEXTGN Characters

4.16.4 Output

An IDAMS image tape with imbedded text with just the generated text as data.

4.16.5 Example

A 1000 by 1000 input image, DATAPIC, is to be marked for identification with the header "JAMES RIVER - BLUE SEPARATION" starting at line 1, pixel 250. The following control cards could be used:

```
TEXTGN, (DATAPIC, 49, 1), (1, 1, 1000, 1000), (OUTPIC, 47, 1), 2
```

```
250      1
```

```
JAMES RIVER – BLUE SEPARATION
```

4.16.6 Messages

TEXTGN generates no special messages.

4.16.7 Flowchart

See Appendix C, Figure C-16.

4.17 NEIGHBOR - NEAREST NEIGHBOR PRINTER LISTING PROGRAM

4.17.1 Program Description

The task program NEIGHBOR accepts as input an IDAMS image tape, a point location, and an output array size. A printer listing of intensity values of neighbors surrounding the input point is produced as a square array with the input point at the center.

When entered, NEIGHBOR checks the input parameters to ensure the printed array will fit on one page. Next the input label is read and unwanted records are skipped. After a header line is printed, the program prints an array of pixel values with the input point location at the center. When the requested array is completely printed, the program returns control to the monitor.

4.17.2 Parameters

The following three special parameters are necessary:

1. Number of central pixel
2. Number of central line
3. Array size

4.17.3 Input

NEIGHBOR requires a single input image tape in standard IDAMS format.

4.17.4 Output

NEIGHBOR produces a printed array of points with a selected pixel at the center.

4.17.5 Messages

Message	Explanation
ARRAY SIZE REDUCED TO 40 X 40	The requested array will not fit on one page, processing continues with a 40 by 40 output array.

4.17.6 Example

It is desired to print a 5 x 5 portion of an image called TEST1 surrounding a point located at line 190, pixel 240. Appropriate IDAMS control cards would be:

```
NEIGHBOR, (TEST1, 49, 1), ,1
```

```
240       190       5
```

4.17.7 Flowchart

See Appendix C, Figure C-17.

4.18 DISPLAY - INTERACTIVE DISPLAY PROGRAM

4.18.1 Program Description

The task program DISPLAY provides the user with numerous image display and manipulation functions. The program displays the available capabilities on the CDC 212 and the user interactively inputs the desired function codes via the 212 keyboard. The task is subdivided into a main driver routine and two segments. The first segment handles all of the functions except the reduce/increase phase of the ZOOM capability, which resides in the second segment.

Once control is passed to the program DISPLAY1 in the first segment, the initial reseau coordinates and box coordinates are set. The subroutines TVCON and TTWCON are called, which connect the TV, and CDC 212 hardware, respectively. A call to the subroutine CDCON enables the following function code table to be displayed on the 212.

IDAMS
FUNCTION CODES

BOXGEN	01	ENLARGE	04	LEFT	06
RESEAU	02	SHRINK	05	RIGHT	07
ERASE	03			UP	08
LOCATE	10			DOWN	09
DATA	11	ZOOM	17		
DATA1	12	EXIT	18	REWIND	13
SELECT	16			FORWARD	14
				REVERSE	15

The program calls the subroutine STORE, which waits until a function code has been input to the 212. The program converts the code from a BCD number to an integer value and returns the integer as an argument. DISPLAY branches to the appropriate subsection depending upon the value of the function code.

4.18.1.1 BOXGEN Function

If the BOXGEN function code (01) was requested, the program checks to see if a reseau mark is already displayed on the TV. When no reseau mark is on the TV, the program computes the box coordinates such that the box will be centered on the TV and will be 138 pixels by 100 lines in size. However, if a reseau mark is displayed on the TV, the box coordinates are computed such that the 138 by 100 box will appear centered around the reseau mark location. The reseau mark is then erased by calling the subroutine KILLIN. Once the box coordinates have been set, the program branches to a subsection of the program which converts the computer image coordinates to a TV format and sends them to the TV hardware. To convert the Y coordinates to TV format, the values are divided by 2 and if the original coordinates were even, then 255 is added to the halved values. This is done to accommodate the TV feature of having a main level and an interlace level. The lines alternate between the main and interlace levels, and therefore, line 1 is equal to the TV line 0, line 2 equals TV line 256, ..., line 511 equals TV line 255, and line 512 equals TV line 511.

The x coordinates must also be converted to a TV format. The TV hardware counts pixels across the line in increments of 11 and therefore, the hardware must know in which group of 11 the pixel resides (x/11) and the pixel position within the group (MOD(x, 11)). The x coordinates are converted and the group number is placed in bits 23-12 of the TV coordinate word and the remainder value resides in bits 11-0 of the word. The conversion subsection then calls the program, DISP, which sends the TV coordinates to the TV hardware. DISP is a COMPASS routine which sends a function code for a box or reseau mark, and then transfers the coordinates to the TV hardware. Control is then returned to the conversion subsection, which returns to the originating function subsection. In the case of the box generator function, control is returned to the point where the function code table is displayed on the 212 and the program is waiting for another function code.

4.18.1.2 RESEAU Function

When the RESEAU function code (02) is specified, the program checks to see if a box is already displayed on the TV. If no box exists on the screen, the program sets the reseau coordinates such that it will be centered on the TV. However, if a box is already displayed, the program computes the reseau coordinates such that it will be centered within the box area. The box is then erased with a call to KILLIN and the program branches to the previously described TV conversion subsection. After the reseau mark has been placed on the TV, the program redisplays the function code table and waits for the next code to be input.

4.18.1.3 ERASE, ENLARGE, and SHRINK Functions

The third function ERASE (03), clears any box or reseau marks from the TV by calling the subroutine, KILLIN. This subprogram sets the TV function register to 0, which removes all marks from the TV screen. The program returns from the erase subsection, to display the function code table again, and wait for the next input function.

The fourth and fifth functions, ENLARGE (04) and SHRINK (05), are processed in the same subsection. A check is made to see if a box is displayed. If not, then control returns to the program area which displays the function code table. If a box is displayed, the box coordinates are checked to see if the box can be enlarged or reduced, depending on which function was requested. If the box is the maximum (minimum) size, then control returns to the program area from which the function code table is displayed. Otherwise, the coordinates are appropriately reduced or increased by one in order to enlarge or reduce the box. A call is made to the delay routine, ICLOCK, which stalls processing a specified number of milliseconds. This is necessary in order to slow down the box enlarging (reducing) action so that the user has control over the box movement. Control is transferred to the subsection which converts the coordinates to TV format and sends the values to the TV hardware. The program continues to enlarge or reduce the box until either the SEND key on the 212 is depressed or the box reaches maximum (minimum) size. Control then returns to the program area which displays the function code table.

4.18.1.4 LEFT, RIGHT, UP, and DOWN Functions

One subsection handles the functions which manipulate the movement of the box or reseau: LEFT (06), RIGHT (07), UP (08), and DOWN (09). Depending upon the direction of the requested movement, the program checks if the coordinates have reached an edge of the image and, therefore, cannot be moved in the requested direction. If this test is positive, then, control returns to the program area which displays the function code table. Otherwise, the appropriate coordinates are reduced or increased by one and control transfers to ICLOCK and the TV conversion subsection, where the TV formatted coordinates are sent to the TV hardware. The program continues to move the box or reseau in the requested direction until either the SEND key is depressed or the box (or reseau) reaches an edge of the image. Control, then, returns to the code which displays the function code table.

4.18.1.5 LOCATE Function

When the function, LOCATE (10), is requested, the corresponding subsection writes the location of the box or reseau, which is presently displayed on the TV, onto the printer and the 212. The program checks to verify that a box or reseau is displayed on the TV. If no marks are on the TV, control returns to the program area which displays the function table. Otherwise, the program prints out the coordinates of the box or reseau on the printer. Before the values can be output to the 212, the integers must be converted to left justified BCD format. This process is done in the subroutine BINBCD, and the reformatted coordinates are output to the 212. Control returns to the program area which displays the function code table only after the user has depressed the SEND key on the 212 keyboard.

4.18.1.6 DATA and DATA1 Functions

The functions DATA (11) and DATA1 (12) are handled in the same subsection. This subsection transfers image data from tape files to the TV. The program first requests the tape unit number from the user by calling the subroutine CDCON, which prints out the request on the 212. The subroutine STORE is then called, which will return with the user's reply in BCD format. The program converts the value to an integer format. The program requests that the user input the color gun numbers. The color gun number, which is returned from a call to STORE, is converted to binary and checked to ensure that it is valid. If the number is not valid ($0 < N \leq 7$) the program will again request that the user input the color gun numbers. The program reads the label record and prints out the length of the record.

Next the program enters a loop which reads 32 lines of data and properly positions the data in a format necessary for the TV hardware. Because of the main and interlace structure of the TV, even lines are separated from the odd lines. Consequently, as the data lines are read in, pointers are set which direct the data into the appropriate buffer location. For instance, in buffer 1 the lines 1,3,5,...,31 are sequentially packed, and lines 2,4,6...,32 are sequentially packed in buffer 2. The tape reads are double buffered, and while the next line is being read, the last line's data is sent to the subroutine, FLIP. This subroutine reverses the pixel order of each word in the line (i.e., if the characters ABCD are input as a word, they would be returned as DCBA in the same word). This procedure is necessary in order to make the data compatible with the TV hardware's counting method. After a set of 32 lines have been read in and processed by FLIP, the 16 even lines and 16 odd lines are ready to be transferred to the TV.

The subroutine LINDIS is called for each set of 16 lines. This routine prepares the data for the transfer and then outputs it to the TV. LINDIS is a COMPASS subroutine which computes the function code depending upon whether the lines are main or interlace and sends the function code to channel 2 (TV hardware). The subroutine then checks to see if the requested function was DATA (11) or DATA1 (12). If the request was for DATA, then each word of the 16 line data block has the least significant bit shifted off. This is required because the TV hardware, which counts from left to right, can only handle five of the six bits per pixel value. Without shifting, the most significant bit would be lost, which is an undesirable result. Consequently, by shifting each word to the right one bit, the TV hardware will be picking up the most significant bit and only losing the least significant bit. However, the DATA1 (12) function does not shift the data words and sends the data words as they are input. This feature is available in case a user wishes to view the data without the shifting procedure.

Once the data words have been prepared for transfer, LINDIS sends two blocks of 16 lines to the TV, channel 2, and waits for the I/O to be completed before returning to DISPLAY1. This procedure of processing data in sets of 32 lines continues until the program senses an end-of-file mark on the input tape. If the program determines that the total number of lines read is not an even multiple of 32, it prints a message indicating that some data lines must have been lost. The program concludes this fact because the number of lines in a TV size image is 512, which is an even multiple of 32. Control then returns to the program area which displays the function code table.

4.18.1.7 REWIND, FORWARD, and REVERSE Functions

The functions REWIND (13), FORWARD (14), and REVERSE (15) are all handled in the same subsection. The program calls CDCON, which requests that the user input the appropriate magnetic tape logical unit number. The subroutine STORE returns the tape unit number in a BCD format and the program converts it into a binary integer value. If the request function was REWIND (13), the program rewinds the tape and returns to the program area which displays the function code table. If the FORWARD (14) or REVERSE (15) functions were specified, the program sends a request, for the number of files to be skipped, to the user via the subroutine CDCON. The reply is returned from STORE and is converted from BCD format to an integer value. The program then forward spaces or backspaces the appropriate number of files, and, if the backspace function is being executed, the end-of-file mark is skipped over before control returns to the program area which displays the function code table.

4.18.1.8 SELECT Function

The subsection which processes the SELECT (16) function enables the user to select the coordinates of a box which will be displayed on the TV screen. By referencing the subroutine CDCON and STORE, the program requests that the user input the coordinates of the desired box. The coordinates must be input in the following order; leftmost pixel value, rightmost pixel value, top line number, and bottom line number. Because the box figure appears in either the main level or the interlace level, the input line values must both be even or odd. If the line numbers are mixed, the program outputs a message to the 212 indicating the error, and then corrects the line numbers by forcing them to both be even or odd values.

The program receives the parameters from the subroutine STORE and scans the parameter list from the last word of the input array to the first word. The program ignores blanks and expects commas to be the separator between coordinates. The coordinates are converted to integer format and sent to the TV conversion subsection, which converts the values to TV format and sends them to the TV hardware. If the parameter list has not been correctly input, the program requests that the coordinates be input again. Once the box has appeared on the TV, control returns to the program section which displays the function code table.

4.18.1.9 ZOOM Function

The last display function is the ZOOM (17) function which takes the area bound within a box or the TV image and increases or reduces it into a TV size image. The phase of the function which requests the required parameters and determines if an increase or reduction of the master image is needed makes up a subsection of the DISPLAY1 program. The code which actually performs the reduction or enlargement of the image resides in a separate segment. By referencing CDCON and STORE, the DISPLAY1 subsection initially requests the name of the image which is displayed on the TV, the tape unit on which the TV image tape resides, and the file number of the image. These parameters are stored in the label array, LBLIN.

The label processing routine, LBLRD, is then called to read the TV image file label, and the information stored in LBLIN is used to verify that the tape is positioned at the requested image. The requested file label contains the name of the master tape (the image tape from which the TV image originated) which is used to send a message to the user, via the 212, reminding him that the specific master tape must be mounted. The reduction or enlargement factor that was used when creating the TV image from the master image is stored in word 11 of the TV file label. Using this factor, the box coordinates, and the dimensions of the master image, the program determines whether the master image must be increased or reduced to create the desired TV image. If the box enclosed area would not result in an optimum TV size image, the program sends a message to the user asking if the user still wants to create the requested TV image. If the user replies negatively, then the program branches back to the program area which displays the function code table. If the user replies positively, the program continues processing.

A request is made for the master tape logical unit number and file number and then the program reads in the master image file label. The program then requests, via the 212, that the user supply the output TV image name, tape unit number, and file number. The output label is written onto the specified tape. Before going to the segment which carries out the actual reduction or enlargement, the program requests that the user verify that all of the parameters are correct. If the response is negative, the program branches to the beginning of the ZOOM subsection and begins the requests for input parameters again. If the user replies positively, the program returns to the main driver which calls segment 2, REDINC, the program that reduces or increases the master image data.

Upon entry, REDINC computes the largest integral reduction or multiplication factor which will just permit the input to fit within a TV size image. An output line buffer is set up with edge fill characters, and the output label record is written. If any fill lines are required at the top of the TV picture, they are written out at this time. A main increase or reduce processing loop is entered. If no increase or reduction is necessary, the master image is just transferred to the output image. If a reduction is required, the program reads an input line, reduces its length by the reduction factor, and stores it in an internal array. This continues until enough input lines are collected to form one output line. Averaging is then performed between lines and the completed line is output. If an enlargement is involved, the program reads an input line, enlarges its length by the multiplication factor, and outputs the enlarged line the requested number of times. After all lines have been processed for either enlargements and reductions, the program writes out any remaining lines of bottom fill. Control then returns to the main driver which recalls segment 1, DISPLAY1, and the IDAMS function code table is again displayed.

4.18.1.10 EXIT Function

The user exits from the DISPLAY package by selecting the EXIT (18) function code. When the program receives this code it returns to the main driver, which then returns to the IDAMS system.

4.18.2 Parameters

DISPLAY calls the IDAMS display package whose parameters are provided interactively through the 212 Display Station. By specifying the name DISPLAY on the task card, the following function code table is displayed on the 212 screen.

IDAMS
FUNCTION CODES

BOXGEN	01	ENLARGE	04	LEFT	06
RESEAU	02	SHRINK	05	RIGHT	07
ERASE	03			UP	08
LOCATE	10			DOWN	09
DATA	11	ZOOM	17		
DATA1	12	EXIT	18	REWIND	13
SELECT	16			FORWARD	14
				REVERSE	15

In order to execute any one of the functions, the user must type in the corresponding numeric code and depress the SEND key. A description of each function is given below.

Code	Function	Description
01	BOXGEN	Generates a box which, if no reseau mark displayed on the TV, is centered on the TV screen and has the following (pixel, line) coordinates: upper left corner (283,206), upper right corner (421,206), lower left corner (283,306), and lower right corner (421,306). If a reseau mark is already displayed on the TV, a box (138 pixels by 100 lines) which will be located around the reseau coordinates, will

Code	Function	Description
01 (Cont'd)		replace the reseau mark. To alter the location and size of the box, refer to the functions LEFT, RIGHT, UP, DOWN, SHRINK, and ENLARGE.
02	RESEAU	Places a reseau mark on the TV screen which, if a box is not presently displayed on the TV, has the following (pixel, line) coordinates at its center: (357, 256). If a box is already displayed on the TV, a reseau mark which is located at the center coordinates of the box replaces the box. To manipulate the reseau's location, refer to the functions LEFT, RIGHT, UP, and DOWN.
03	ERASE	Removes a box or reseau mark from the TV.
04	ENLARGE	Increases the size of the box which is displayed on the TV. To stop the enlarging action, the user must depress the SEND key.
05	SHRINK	Reduces the size of the box which is displayed on the TV. To halt the shrinking action, the user must depress the SEND key.
06	LEFT	Moves the box or reseau mark to the left. The left action is halted by depressing the SEND key.

Code	Function	Description
07	RIGHT	Moves the box or reseau mark to the right. The right action is halted by depressing the SEND key.
08	UP	Moves the box or reseau mark upward. The upward action is halted by depressing the SEND key.
09	DOWN	Moves the box or reseau mark downward. The downward action is halted by depress-ing the SEND key.
10	LOCATE	Returns the coordinates of the box or reseau which is presently displayed on the TV screen. When the user wants to clear the coordinates from the 212 and have the function code table reappear, the SEND key must be depressed.
11	DATA	Drops an image tape file, which contains 64 gray level data, onto the TV. The pro-gram requests two input parameters. The tape unit on which the image tape is mounted must be keyed in after the re-quest appears on the 212. After the SEND key is depressed, a request for the color gun number will appear. The TV has three TV refresher disk files available for image data storage, and each disk can be assigned to one of the three available color guns (red, green, or blue). The color gun parameter

Code	Function	Description
11 (Cont'd)		is a value which determines which disk file(s) the user wants for storing an image. The parameter is an octal representation of a three-digit binary number, in which each digit corresponds to one of the disk files and the "on-off" conditions are rep- sented by ones and zeros, respectively. The following table shows the correspond- ence between the color gun number, the disk assignments and the binary number from which the parameter value was de- rived.

Color Gun Number	Disk File(s)	Binary Representation Disk 3	Disk 2	Disk 1
1	1	0	0	1
2	2	0	1	0
3	1 and 2	0	1	1
4	3	1	0	0
5	1 and 3	1	0	1
6	2 and 3	1	1	0
7	1, 2, and 3	1	1	1

Once the user has specified the disk file(s) into which the image data are to be stored and the SEND key has been depressed, the image will be dropped to the disk, and dis- played on the TV. The user can define which color is to be associated with each

Code	Function	Description
11 (Cont'd)		disk by manually setting the three color wheels switches on the IDAMS Control Panel. The three wheels, from left to right, represent the color guns of red, green, blue, respectively. By setting the wheels to the appropriate disk number, the user has complete control over the color assignment of any image stored in the TV disk files.
12	DATA1	Drops an image tape file onto the TV. The required parameters are described above under function code 11. DATA1 differs from DATA in that the data contains 32 gray level values and, therefore, the data words do not have the least significant bit shifted off. However, the most significant lot will be truncated since only five bits of data can be displayed at one time.
13	REWIND	Rewinds a requested tape. The program requests the logical unit number on which the required tape is mounted. After keying in the tape unit number and depressing the SEND key, the tape is rewound to loadpoint.
14	FORWARD	Forward spaces a tape a specified number of files. The program requests the logical unit number on which the required tape is mounted. After keying in the tape unit number, the program requests the number

Code	Function	Description
14 (Cont'd)		of files over which the tape is to space forward. Once the SEND key is depressed, the tape is forward spaced the specified number of files.
15	REVERSE	Backspaces a tape a specified number of files. The program requests the logical unit number on which the required tape is mounted. After keying in the tape unit number, the program requests the number of files over which the tape is to be backspaced. Once the SEND key is depressed, the tape is backspaced the specified number of files.
16	SELECT	Enables the user to select the coordinates of a box which is to be displayed on the TV. The program requests that the coordinates be input in the following order: leftmost pixel, rightmost pixel, top line, lower line. The line number should be paired even, or odd, but not mixed. This is a display hardware requirement. The parameters must be separated by commas and the final parameter must be followed by a blank. Any blanks placed between parameters are ignored.
17	ZOOM	Takes the area bound within a box on the TV image and increases or reduces it into a TV size image. The program requests

Code	Function	Description
17 (Cont'd)		the name of the image which is presently displayed on the TV. After entering in the name and depressing the SEND key, the TV image's tape unit is requested, and is followed by a request for the file number (be sure to specify the file number with two digits). A message reminding the user that the master tape must be mounted is displayed on the 212, and is followed by a request for the master tape's unit number and file number. Before the program begins the ZOOM procedure, information about the new output tape is requested. The user is asked to supply the output tape name, the unit number and the file number. The program requests that the user specify if the input parameters are believed to be correct. If a "Y" is returned, the program continues with the ZOOM process. However, if an "N" is returned, the program begins the input parameter requests again. This gives the user, who is aware of an input parameter error, another chance to supply the correct input. (Note: After keying in the proper response to all requests, remember to press the SEND key.) The program will display on the TV whether an increase or reduction of the master image was necessary and the multiplication factor

Code	Function	Description
17 (Cont'd)		involved. The "ZOOM" image will reside on the output tape when the program is completed, and the user must reference DATA when he wishes to drop the image onto the TV screen.
18	EXIT	Returns control to the IDAMS main DRIVER program.

4.18.3 Input

DISPLAY has variable inputs depending upon which functions are requested. If the functions DATA, DATA1, REWIND, FORWARD, or REVERSE are requested, DISPLAY requires a single input image tape in standard IDAMS format. If the ZOOM function is requested, two input image tapes in standard IDAMS format are required (a TV size image tape and the master image tape).

4.18.4 Output

Display drops image data onto the TV and displays a box and reseau mark on the TV screen. If ZOOM is referenced, a single output image tape in standard IDAMS format is produced.

4.18.5 Examples

The fourth file on the image tape which is mounted on tape unit 49 is to be dropped onto all three TV disks. A reseau mark is to be placed on the image and moved to a desired point, where a box replaces the reseau mark. The box is increased slightly and the coordinates are printed out. The enclosed area is increased to a TV size image and the new image is dropped onto the first TV disk. The following communication would be required to achieve the above operations.

1. The following single control card would be submitted.

```
DISPLAY
```

2. The code 14 (FORWARD) would be entered on the 212 and the SEND key depressed. The user would specify the number 49 to the magnetic tape unit request and then would specify the number 03 to the request of number of files to be skipped.

3. The code 11 (DATA) would drop the fourth file on the TV after the user has specified the logical unit number 49 and color gun number 7 (binary representation indicating all three disk files).

4. The code 02 (RESEAU) would be entered on the 212 and a reseau mark would appear on the TV, once the SEND key has been depressed.

5. By using codes 06-09 (LEFT, RIGHT, UP, DOWN) the user would locate the area of interest.

6. The code 01 (BOX) would replace the reseau mark with a box.

7. The box would be increased by using code 04 (ENLARGE).

8. The code 10 (LOCATE) would display the box coordinates on the 212.

9. The code 17 (ZOOM) would increase the enclosed area using the master tape image data and outputting the area as a TV size image.

10. The new image is dropped onto the color gun 1 disk by referencing code 11 (DATA) again. The output tape unit from the ZOOM phase is entered as the input tape unit and the color gun number is 1.

4.18.6 Messages

DISPLAY generates no special messages.

4.18.7 Flowchart

See Appendix C, Figure C-18.

4.19 MODIFY - IMAGE EDITING PROGRAM

4.19.1 Program Description

The task program, MODIFY, will recreate a missing or destroyed data line on an IDAMS tape by performing a pixel by pixel linear interpolation between the two existing data lines which bracket the one to be replaced. Alternatively, one or more consecutive lines may be deleted from a tape image. It also permits the user to modify individual pixels on an IDAMS tape where necessary. Modifications performed are based on sets of keywords followed by parameters needed for the function indicated. For efficiency, these input parameter sets are first sorted on line number to permit the updating process to be accomplished in one pass through the input tape.

When entered, MODIFY checks the input parameters for keywords, grouping them by sets and supplying default values where necessary. The parameter sets are then sorted by line number. After writing an IDAMS label on the output tape, a major processing loop is entered which reads a line, modifies it where necessary, and writes the resulting line on the output tape.

For function code DEL, the requested number of input lines is skipped.

For function code ADD or MODL, data values for the new line are calculated by averaging between bracketing input lines.

For function code MODP, the specified number of pixels is replaced by the supplied value.

When all parameter sets have been processed, MODIFY returns to the monitor.

4.19.2 Parameters

MODIFY requires the following special parameters in sets, the first of which must be a keyword.

1. Function code - ADD, DEL, MOD, or MODP only.

   a. ADD will insert one or more new lines starting at the specified line number.

   b. DEL will delete one or more lines starting at the specified line number.

   c. MODL will modify all or part of a line by averaging between the preceding and following lines.

   d. MODP will replace one or more pixels in a line with an input value.

2. Starting line number - This indicates the first line to be added, deleted, or modified.

3. Number of lines/value - For ADD and DEL, this field indicates the number of lines affected (default = 1). For MODL, this field is always set to 1. For MODP, this field contains an input value for the pixels being replaced (default = 0).

4. Starting pixel - For MODL and MODP only, this field may be used to specify a starting pixel other than the beginning of the input line (default = SP). Not used for ADD or DEL.

5. Number of pixels - For MODL and MODP only, this field is used to specify a number of pixels less than or equal to the number in the input line (MODP default = 1, MODL default = NP). Not used for ADD or DEL.

4.19.3 Input

MODIFY requires an input tape in standard IDAMS format.

4.19.4 Output

MODIFY produces an output tape in standard IDAMS format.

4.19.5 Example

1. An image tape, TEST1, is to be modified as follows:

   a. Delete lines 21 - 40

   b. Add four lines after line 230

   c. Change line 150

   The following control cards would carry out the desired modifications and create a new tape, MODTEST:

```
MODIFY, (TEST1, 49, 1), (1, 1, 340, 270), (MODTEST, 47, 1), 1
```

```
DEL, 21, 20, ADD, 231, 4, MODL, 150
```

2. Assume that points in an image tape, TEST1, are to be altered as follows:

Pixel	Line	Value
240	45	63
30	125	0
200	245	10

   Control cards which would cause the results to be written onto a tape called TEST2 are shown below:

```
MODIFY, (TEST1, 49, 1), (1, 1, 270, 340), (TEST2, 47, 1), 1
```

```
MODP, 45, 240, 63, MODP, 125, 30, 0, MODP, 245, 200, 10
```

4.19.6 Messages

Bad data values, if any, are printed on the line printer. In addition, the following messages may be generated:

Message	Explanation
BAD LINE NUMBER	A conflicting or illegal line number was defined as an input parameter.
BAD FUNCTION CODE	An illegal function code was defined on input.

4.19.7 Flowchart

See Appendix C, Figure C-19.

4.20 INSERT - WINDOW INSERTION AND MOSAICKING PROGRAM

4.20.1 Program Description

Task program INSERT provides a capability for superimposing a portion of one image upon another. Two IDAMS image tapes are accepted as input with the primary input tape (as defined on the task control card) considered as base and the secondary input tape the "window." A single composite IDAMS image is output. A mosaicked output tape can be created by repeating the INSERT function as many times as needed.

INSERT initially picks up the input parameters and calculates the size of the output image, the number of lines to be copied directly to output before and after the window, and the pixel position of the window relative to the primary input line. Unaffected lines are then copied directly to output. A loop is entered next to process the window portion of the image. A line of the primary input and one of the windows' is read, the lines are merged, and the composite is written to output. Looping continues until the entire window area has been processed. Any remaining base image lines are then copied to the output tape after which INSERT returns to the system.

NOTE: Fill characters are supplied for cases where the window extends beyond the boundaries of the base image.

4.20.2 Parameters

INSERT requires the following parameters:

1. INSPW - Starting pixel of window position in secondary input
2. INSLW - Starting line of window position in secondary input
3. NPWNDW - Number of pixels in window
4. NLWNDW - Number of lines in window

5. IOTSPW - Pixel position in output of upper left corner of window (default = 1)

6. IOTSLW - Line position in output of upper left corner of window (default = 1)

7. IFILL - Fill character gray level value (default = 0)

A negative value for parameter 5 or 6 indicates that the upper left corner of the window is the specified number of pixels or lines to the left or above, respectively, the upper left corner of the base image.

In addition, the task name, primary and secondary input, size, output, and cards fields must be defined on the task control card.

4.20.3 Input

INSERT requires a primary (base) input tape and may have an optional secondary (window) input tape, both in standard IDAMS format.

4.20.4 Output

INSERT produces an output tape in standard IDAMS format.

4.20.5 Example

A 100*100 window from tape, WINDOW, starting at pixel 50, line 75 is to be inserted into a base image called TEST1 beginning at pixel 125, line 150. The result will be called COMPOSIT. The following control cards would accomplish the desired result:

```
INSERT, (TEST1, 49, 1, WINDOW, 47, 1), (1,,1, 340, 270), (COMPOSIT, 48, 1), 1
```

```
50, 75, 100, 100, 125, 150
```

4.20.6 Messages

INSERT may generate the following messages:

Message	Explanation
2NDARY INPUT STARTING PIXEL TOO BIG	The starting pixel value specified was greater than the highest pixel number in the image, execution terminates.
ONLY nnnn WINDOW INPUT PIXELS AVAILABLE	The line length defined for the secondary input exceeded the available length, execution continues with pixels reduced.
2NDARY INPUT STARTING LINE TOO BIG	The starting line number specified was greater than the last line in the image, execution terminates.
ONLY nnnn WINDOW INPUT LINES AVAILABLE	The number of lines defined for the secondary input exceeded the available line count, execution continues with number of lines reduced.

4.20.7 Flowchart

See Appendix C, Figure C-20.

4.21 GRID - GRID OVERLAY PROGRAM

4.21.1 Program Description

The task program, GRID, superimposes a reference grid on an IDAMS image tape. The grid block size and intersection points are controlled by input parameters. In addition, width and gray level value assigned to the grid line may be input as parameters or defaulted to 1 and 63, respectively.

GRID begins execution by accessing the input parameters and calculating which pixels and lines are to be replaced by grid values. After writing an output label, the program enters a major processing loop. Within this loop, a data line is read and characters are either inserted for vertical grid lines or the entire line is replaced by the grid line value. The completed line is then written on the output tape. Looping continues until all lines are processed and GRID then returns to the system.

4.21.2 Parameters

GRID requires the following special parameters:

1.	JUNCP	- Pixel number of first junction (default = grid block width/2)
2.	JUNCL	- Line number of first grid junction (default = grid block length/2)
3.	NPGRID	- Grid block width in pixels
4.	NLGRID	- Grid block length in lines
5.	LINSIZ	- Grid line size (defaults to 1 pixel width)
6.	LINLVL	- Grid line gray level (default = 0 or 63 if IFILL = 0)
7.	IFILL	- Fill character to use if an output grid only is desired (default = 0)

4.21.3 Input

GRID requires an input tape in standard IDAMS format, unless an output grid with fill data for background is desired.

4.21.4 Output

GRID generates an output tape in standard IDAMS format.

4.21.5 Example

A grid is to be superimposed on an image, TEST1. There are to be 30 pixels and 20 lines per grid block, and the first intersection is to be at pixel 5, line 10. Each grid line is to be two pixels wide with an assigned value of 32. If the output is called GRIDTEST, the following control cards would be appropriate:

```
GRID, (TEST1, 49, 1), (1, 1, 340, 270), (GRIDTEST, 47, 1), 1
```

```
5, 10, 30, 20, 2, 32
```

4.21.6 Messages

The following fatal error message may be generated:

Message	Explanation
BAD GRID SIZE PARAMETER	The number of pixels or lines defined on input for grid block size was 0 or negative, execution terminates.

4.21.7 Flowchart

See Appendix C, Figure C-21.

4.22 GEOMTRAN - GEOMETRIC TRANSFORMATION PROGRAM

4.22.1 Program Description

GEOMTRAN is a general purpose geometric transformation program that will allow the user to approximate a wide range of non-linear geometric transformations with a piecewise linear transformation. This program will not only allow for simple scaling and rotation operations, but will allow the user to map specific control points onto corrected control points in such a way as to map other points of the image linearly with respect to near control points.

This program, because of its generality and complexity, is divided into three phases:

1. The first phase reads the input and constructs pieces of output lines (merge strings) which are written to disk.
2. The second phase merges the line pieces constructed by the first phase and writes longer "merge strings" to tape.
3. The final phase merges the strings on to tape strings until the final image has been constructed.

Each of these phases will now be explained in detail.

4.22.1.1 Phase I

The first phase of the geometric transformation routine begins by reading the input parameters which consist of the following:

1. A sequence of points (X_i, Y_i) $i = 1, \cdots, n$ contained in the input image. For example these points would generally be points of interest or of known geographic location (see Diagram 1).

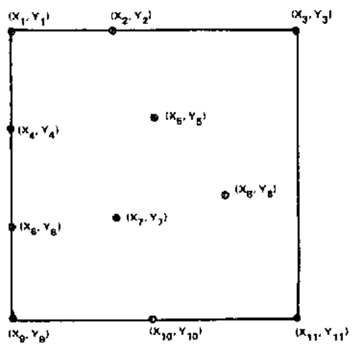


Diagram 1

2. A corresponding sequence of points (X_i', Y_i') i = 1, $\cdots$, n in the the output image. These points would generally be the "corrected" points of those points specified in the input image (see Diagram 2).

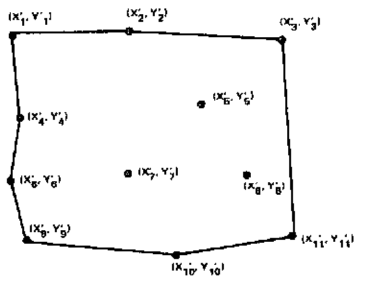


Diagram 2

3. A list of segment pairs (p_i, q_i) i = 1, $\cdots$, m that indicate line segments between pairs of points. These line segments serve to divide the image into topologically distinct regions (see diagrams 3 and 4) so that the applicable linear transformation can be applied to intermediate points.

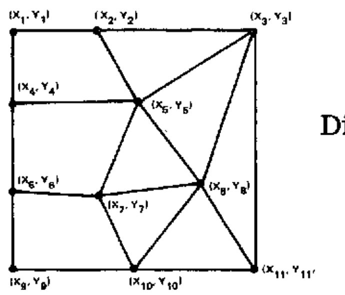


Diagram 3

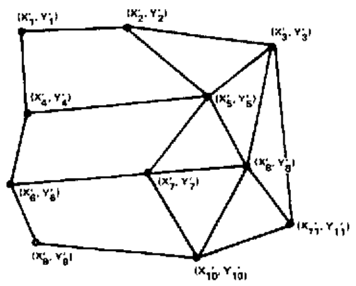


Diagram 4

These segments are input to the program as index pairs. In Diagrams 3 and 4 these pairs would be:

(1,2), (2,3), (1,4), (2,5), (3,5), (4,5), (4,6), (5,7), (5,8), (3,8), (3,11), (6,7), (7,8), (8,11), (8,10), (6,9), (9,10), (10,11), (7,10).

4. The portion of the input image to be transformed is specified (see Diagram 5).

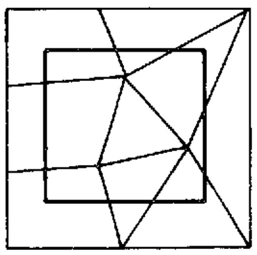


Diagram 5

5. The portion of the output image to be displayed will be specified (see Diagram 6).

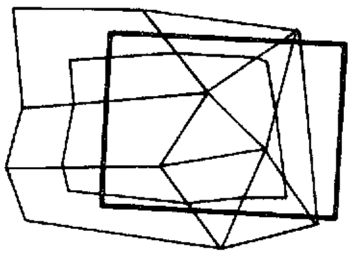


Diagram 6

6. The tape units for both input and output will be specified.

The next portion of Phase I of the Geometric Transformation Program performs the actual transformation requested. It accomplishes this in the following way:

1. Memory allocation parameters are computed for output buffers, input buffers, and image storage.

2. The input image is read and the portion of the input image to be transformed is stored in memory until available memory is filled.

3. The line segments that make up the rectangular boundary of the portion of the image held in memory are intersected with the line segments of the input segment list. These intersections will define a new segment list and a new point list in the input image (see Diagram 7). (Parametric functions are generated to create the new line segment list.)

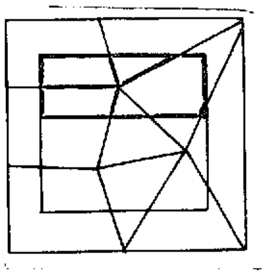


Diagram 7

4. The corresponding point list to the updated point list will be calculated for the output image.

5. Each of the segments in the updated list in the output image will be intersected with the segments that make up the rectangular boundary of the portion of the output image to be displayed (see Diagram 8).

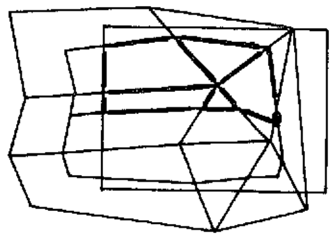


Diagram 8

These points of intersection will define a new segment and point list in the output image (see Diagram 9).

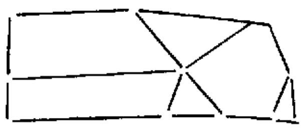


Diagram 9

6. The corresponding point list to the new point list will be calculated for the input image (see Diagram 10).

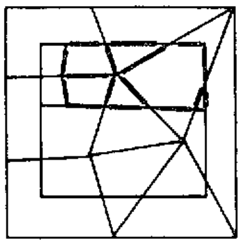


Diagram 10

7. The updated segment list in the output image is now ordered in the following way. The two end-points of each segment are ordered so that the top left point index is first in the pair. The entire list of segments is now ordered by the first point so that the list is in left to right within top-to-bottom order. This sorting of the segments will allow a great deal of time to be saved as the raster segments are generated later in the program.

8. The first output line will start at the first point of the first segment in the ordered list. This line will intersect a specific number of segments that will be put into a "current segment list".

   The current segment list will contain the following for each segment:

   a. The index pairs (p_i, q_i) indicating the segment

   b. The X' coordinate of the intersection of the current output line with the segment

   c. The X and Y coordinates of the corresponding point in the input segment

   d. The change in the output intersection $\Delta X'_y$ for a change by one of the output line

e. The change $(\Delta X_Y, \Delta Y_Y)$ in the input intersection for each change, by one, of the output line

f. The change $(\Delta X_X, \Delta Y_X)$ along the input segment for each change, by one, of the pixel along the output line

g. The change $(\Delta\Delta X_{XY} \Delta\Delta Y_{XY})$ of $(\Delta X_X \Delta Y_X)$ for each change, by one, of the output line

9. The points of intersection (X_i) of the segments in the current line list with the current output line are in ascending order with respect to X . This is because of the order of the output line segments.

The first intersection p of the output line m corresponds to the intersection (X, Y) in the corresponding input segment. This point in the input segment lies within the area defined by four adjacent pixels in the input image. The four pixel values, and the position (X, Y) are used to interpolate an output pixel that will correspond to output point (p, m).

NOTE: If a given raster point (X_m, Y_n) on the output image maps into (X', Y') under T^{-1} in the input image; i.e., $(X_m, Y_n)T^{-1} = (X', Y')$, and if (X', Y') is in the area bounded by the pixels (X'_i, Y'_j), (X'_{i+1}, Y'_j), (X'_i, Y'_{j+1}) and (X'_{i+1}, Y'_{j+1}) where the pixel values are Z'_{ij}, Z_{j+1j}, Z_{ij+1} and Z'_{i+1j+1}, respectively, then the output pixel value associated with (X_m, Y_n) is:

$$\begin{aligned} Z_{mn} = {} & (X' - X'_i)(Y' - Y'_j)(Z'_{ij} + Z'_{i+1j+1} - Z'_{ij+1} - Z'_{i+1j}) \\ & - (Y' - Y'_i)(Z'_{ij} - Z'_{ij+1}) - (X' - X'_i)(Z'_{ij} - Z'_{i+1j}) \\ & + Z'_{ij} \end{aligned}$$

10. Two types of output buffers are maintained by the Phase I part of the Geometric Transformation Program. The first buffer type contains the following:

a. The line number and pixel number of each section of output line

b. The number of pixels in the section of output line

c. The sector of disk on which this section is found

d. The word within the sector where section starts

e. The character within the word where section starts

This buffer is of fixed length and is written to disk when full. The second buffer contains raw pixel values, is of fixed length, and is emptied to disk whenever it becomes full.

The point (X, Y) is stored in buffer 1, and the value of the point (p, m) is stored into buffer 2. The location of this store is stored in buffer 1.

11. One is added to the X value of the current pixel position and this value is checked against the next X intersection value in the current line list. If the pixel position is less than the intersection value, then ΔX_X and ΔY_X are added to (X, Y), an interpolation is done and the resulting pixel value is stored in buffer 2 and step 11 repeated. If the pixel is equal to the next intersection, then a pointer to the segment in the current line list is advanced and step 11 is repeated.

12. When the pixel position in the current line is equal to the last X value in the current line list, then the current line number is incremented by 1.

13. When the line number is equal to one or more segment ends in the current line list, then these segments are deleted from the list and replaced by segments from the segment list whose start values match the deleted end value. (List process is employed here.)

14. If new segments are added then X', (X, Y), ΔX_X, ΔY_X, ΔX_Y, ΔY_Y, $\Delta\Delta X_{XY}$, $\Delta\Delta Y_{XY}$, and $\Delta X'$ are computed for those segments. The segments that were not deleted from the list are processed in the

following way. $\Delta\Delta X_{XY}$ and $\Delta\Delta Y_{XY}$ are added to ΔX_X and ΔY_X, respectively, ΔX_Y and ΔY_Y are added to X and Y, respectively, and $\Delta X'$ is added to X.

15. Step 9 is entered and this entire process continues until no more segments can be entered into the current line list. At this point Step 2 is entered and memory loads are processed until the input data is exhausted or until one of the disk files is full, at which time Phase II is entered.

4.22.1.2 Phase II

The second phase of the Geometric Transformation program has as its input the three files and one reference word generated by Phase I of the program. These files are stored on the disk.

The reference word, which is always located in word one of cell number 32170, contains the total number of core loads from disk.

The first file, which will be called the "core load table" will contain the following information for each core load:

word 1:	cell number of start of "index file" (2nd file)
word 2:	start word number within cell number
word 3:	number of index entries for core load

The second file, which is the "index file", contains the following data for each string:

word 1:	The line number of the beginning of the string
word 2:	The pixel number of the beginning of the string
word 3:	The number of pixels in the string
word 4:	The cell number of start of string
word 5:	The word *100 + character number of start of string

The third file contains the strings of pixel values to which the index entries refer.

The output from Phase II consists of three tapes that contain merge strings distributed among the three tapes in a Fibonacci series, so that Phase III can achieve a polyphase merge. When the number of actual merge strings on a tape is less than the number specified in a Fibonacci series, the differences are stored in common, where they will be passed to Phase III.

Phase II sorts from disk to tape in the following way:

1. It uses the core load table to read in index buffers from the disk, which refer to the first strings of each core load.

2. The indexes are sorted by pixel number within line number.

3. After the indexes are ordered, strings are read from disk in the order of the indexes and stored in an output buffer along with their associated start line, start pixel, and number of pixels. When required, strings are merged together, and redundant pixel values are omitted.

4. When output buffers are full they are emptied to specified output units in accordance with the Fibonacci algorithm.

4.22.1.3 Phase III

This phase has as its input the three merge tapes that were the output from Phase II. Phase III assumes that the merge strings are arranged on the three tape drives in a Fibonacci series so that the image string can be merged by the poly-phase sort-merge method. In the case where the number of required strings specified for each tape exceeds the actual number of strings residing on a tape, a set of numbers is supplied from Phase II which represents the number of strings lacking on each tape.

Two of the tapes are defined as input and have been rewound, while the third tape is used as output and is positioned at the record following the last image merge string output from Phase II.

Phase III processes tapes in the following way:

1. Merge strings are read from the two input tapes and merged onto a single string on the output file. In the case where the dummy string counter is greater than one, no tape processing is done, but the counter is decremented and processing is continued.
2. This process continues until one of the input tapes is exhausted.
3. The exhausted tape is rewound and becomes the new output tape.
4. The old output tape is rewound and becomes one of the input tapes.
5. This process continues until all of the merge strings have been merged into one complete image string. At this time, the tape is rewound and copied into IDAMS format. Any gaps in the image are filled with zeros.

4.22.2 Parameters

1.	SPO	-	Starting pixel of output window
2.	SLO	-	Starting line of output window
3.	NPO	-	Number of pixels of output window
4.	NLO	-	Number of lines of output window
5.	NOSEG	-	Number of line segments
6.	NOPOT	-	Number of points
7.	ICODE	=	0, if parameters on card
		=	1, if parameters on disk

8. Starting point number of a line segment

9. Ending point number of the same line segment

.
.
.
.
.
.
.
.

(8 + NOSEG * 2). Starting pixel number of the first input image point

(9 + NOSEG * 2). Starting line number of the first input image point

.
.
.
.
.
.
.
.

(8 + NOSEG * 2 + NOPOT * 2). Starting pixel number of the first corrected (output) image point

(9 + NOSEG * 2 + NOPOT * 2). Stating line number of the first corrected (output) image point

4.22.3 Input

An input tape in standard IDAMS format is necessary.

4.22.4 Output

An output tape in standard IDAMS format is generated.

4.22.5 Example

A selected portion of the test image, TEST1, is to be rotated through an angle of 45 degrees after it has been increased to TV size. The resulting output will be another TV size image (700*512). Four control points will be defined in the input and output images in a manner which will achieve the desired 45-degree rotation. These points are:

Point	Input (pixel, line)	Output (pixel, line)
1	134,40	350,1
2	565,40	700,256
3	565,471	350,512
4	134,471	1,256

Connecting line segments will be defined to define a mapping rectangle. Points 1-4, 1-2, 2-3, and 3-4 will be connected. The necessary parameters are:

```
INCREASE, (TEST1, 49, 1), (1, 1, 340, 270), (TVTEST, 48, 1), 1
```

```
700, 512
```

```
GEOMTRAN, (TVTEST, 48, 1), (134, 40, 432, 432), (ROTATE, 45, 1), 3
```

```
1, 1, 700, 512, 4, 4, 0, 1, 4, 1, 2, 2, 3, 4, 3
```

```
134, 40, 565, 40, 565, 471, 134, 471
```

```
350, 1, 700, 256, 350, 512, 1, 256
```

4.22.6 Messages

The geometric transformation program produces no messages.

4.22.7 Flowchart

See Appendix C, Figure C-22.

4.23 CHIPGN - REFERENCE CHIP TAPE GENERATION AND UPDATE PROGRAM

4.23.1 Program Description

The task program CHIPGN extracts one or more reference chips from an image tape and writes them onto a new reference tape or adds them to an existing reference tape. If no new chips are requested, CHIPGN simply prints out the directory of an existing reference chip tape.

If a new tape is to be generated (specified by an entry in the output field of the task card) a blank directory block is generated, the directory label record is set to specify zero chips, and the directory file is written onto the output tape. When an existing reference chip tape is specified as an input (secondary input for update mode and primary input for directory listing mode), CHIPGN begins by reading the existing directory into core. For update mode, the label data are transferred to the output label block and the tape is advanced to the end of the last chip file. For directory listing mode, control is transferred directly to the directory listing procedure.

Otherwise, CHIPGN accesses the parameters specifying new chips one at a time. The keywords are scanned, parameters are converted to suitable formats, and a check is made to see that the specified reference chip lies within the boundaries of the input image. If not, an advisory message is printed, and the next set of parameters is scanned. For each valid chip specification, the chip count in the directory label is incremented by one and the chip name is generated. The general purpose subroutine UTMCON is called to convert UTM to Lat-Long or vice versa, depending on which set of coordinates were input as parameters. Then the chip label is generated and written out. The label data, together with the source image location and any memorandum, are also entered in the appropriate line of the directory block. The input tape is then advanced to the starting line for the chip, and image data are transferred to the output tape one line at a time. When the chip file is complete, an end of file is written.

After the last chip file has been written, the output (reference) tape is rewound, the directory is printed out, and the new or updated directory file is written onto the output tape.

NOTE: CHIPGN can write a maximum of 98 reference chips onto a single reference chip tape.

4.23.2 Parameters

A separate set of parameters is used to specify each chip to be generated. Because the user may enter either Lat/Long or UTM coordinates, CHIPGN requires that parameters be identified by keywords, successive parameters must be separated by commas. The parameters are:

1.	CPP	=	nnnn	- Pixel location on input image of chip center
2.	CPL	=	nnnn	- Line location of chip center. Note: processing time will be minimized if successive chips are specified in order of increasing CPL.
3.	LAT	=	dd, mm, ss. s_{S}^{N}	- Latitude of chip center. If N/S designation is omitted, north latitude is assumed unless latitude is preceded by a minus sign, in which case south latitude is assumed.
4.	LONG	=	ddd, mm, ss. s_{W}^{E}	- Longitude of chip center. If E/W designator is omitted, north is assumed unless value is negative, in which case west longitude is assumed.
5.	ZONE	=	nn_{S}^{N}	- UTM Zone. If N/S designator is omitted, north is assumed unless number is preceded by a minus sign.

6.	UTME	=	nnnnnn	-	UTM easting in meters (=500,000 at zone central meridian)
7.	UTMN	=	nnnnnnn	-	UTM northing in meters
8.	ELEV	=	nnnn	-	Elevation above sea level in meters
9.	MEMO	=	'cc...cc'	-	Up to 24 alphanumeric characters of identifying memorandum enclosed in quotes (optional parameter)

NOTE: Either, but not both, Lat/Long or UTM coordinates must be supplied.

The use of the input and output specifications on the task card determines whether a new tape, update, or only the directory listing is to be generated, as follows:

Mode	Input/Output Specifications
List Directory Only	Primary Input = Reference Chip Tape No secondary input or output
New Reference Tape	Primary input = Source Image Output = Reference chip tape
Update existing tape	Primary input = Source Image Secondary input = Reference chip tape No output specified

In all cases the input size specification is ignored, and may be defaulted.

NOTE: All reference tapes are identified by the name of the directory file, which consists of five alphanumeric characters followed by 01D; for a new tape, only the first five characters of the specified output name are used. Reference chip files have the same initial five characters as the directory file, followed by a two-digit file number and the letter R as shown below:

Specified output name:	EASTBAY
Directory file name:	EASTB01D
First chip file name:	EASTB02R
Sixth chip file name:	EASTB07R

4.23.3 Input

CHIPGN requires one or two input tapes in standard IDAMS format, depending on the mode selected. They have the following significance:

Mode	Input(s)
List Directory only	Single input = reference chip tape
New Reference tape	Single input = source image
Update existing tape	Primary input = source image Secondary input = reference chip tape

4.23.4 Output

Output from CHIPGN comprises a printer listing of the reference chip directory and a tape containing a directory file and one or more reference chip files in standard IDAMS format; the number of chips is specified by word 7 of the directory file label. Output specification depends on the mode selected, as follows:

Mode	Output
List directory only	No output tape specified
New reference tape	Output - tape specified on task card Secondary input tape specified
Update existing tape	No output specified on task card; output = secondary input tape

Each reference chip file has an extended label, as follows:

Word(s)	Contents
1-2	Name of chip: XXXXXnnR where nn is file number and R is always present as an identifier
5	LUN: Logical unit number when generated (value does not affect subsequent use)
6	File Number (between 2 and 99)

Words	Contents
7-8	Latitude: Floating point, degrees and decimal fraction, with sign indicating north (+) and south (-)
9-10	Longitude: Floating point, degrees and decimal fraction, sign indicates east (+) and west (-)
11-12	UTM easting: Floating point, preceded by 10^6 times zone number and sign indicating north (+) or south (-)
13-14	UTM northing: Actual value in northern hemisphere; value -10^7 (giving negative result) in southern hemisphere.
15-16	Elevation

The directory file has a standard label, except that word 7 specifies the number of chip files on the tape. The directory comprises a single 2156-word (98 by 22) record containing for each chip its name, latitude, longitude, UTM easting and northing, elevation above sea level, source name, position of chip center in source, and 24-character identifying memorandum.

The printer listing gives the directory data for each chip: latitude and longitude are converted to degree, minute, and second form and UTM coordinates are expressed in standard format.

4.23.5 Example

1. A reference tape is to be created using two areas from an image tape named FLGHTQ16. Task and parameter cards would be as follows:

```
CHIPGN, (FLIGHTQ16, 48, 1),, (WILDWEST, 47, 1), 2
```

```
CP = 369, CL=426, LAT=33, 18, 56.0, LONG=105, 16, 24.2W, ELEV=987, MEMO='DRY GULCH JUNCTION',
```

```
CP = 861, CL=3248,LAT=32, 09, 16.4, LONG=105, 01, 48.8W, ELEV=1126, MEMO='TEXACO TANK FARM'
```

Note that only the first five characters of the output name will be kept, to make the directory file name WILDW01D and the chip names WILDW02R and WILDW03R.

2. An additional chip is to be added to tape WILDW01D, generated above, from image tape FLGHTQ18. Task and parameter cards are:

```
CHIPGN, (FLIGHTQ18, 48, 1, WILDW01D, 47, 1),,, 1
```

```
CP = 3241, CL=1892, LAT=33, 06, 10.0, LONG=-105, 17, 18.6, ELEV=874, MEMO='THOMPSON POND'
```

3. At a subsequent time it is desired to check the contents of this tape by examining the directory. The required task card is:

```
CHIPGN, (WILDW01D, 47, 1)
```

4.23.6 Messages

Message	Explanation
aaaa IS NOT A KEYWORD	An illegal keyword was detected, execution continues.
CHIP AT CPP = nnnn, CPL = nnnn NOT WITHIN RANGE	The chip coordinates will not permit a full chip to be extracted from the input image, parameters are discarded and execution continues.
ATTEMPT TO ADD nnTH CHIP HAS EXCEEDED CAPACITY	The chip tape is full, no more chips may be added unless others are removed.

4.23.7 Flowchart

See Appendix C, Figure C-23.

4.24 RZOMAP - RESEAU MAPPING PROGRAM

4.24.1 Program Description

This task routine locates reseaus on RBV imagery using a digital filter which is highly selective for vertical and horizontal bars, fits a least-squares polynomial to the corresponding displacements from the nominal positions, and generates a table containing nominal and actual locations, using the polynomial to fill in any missing reseaus by interpolation.

RZOMAP begins by calling the subprogram GETRZO to find as many of the reseaus as possible. GETRZO begins by setting up the output table. If the user has entered a table of nominal locations as parameters, these are transferred to the output table. Otherwise, an internally stored table is used. The remaining parameters are then checked for defaulted values, which are replaced by values appropriate to the table of nominal reseau locations.

The reseau search is carried out one row of reseaus at a time. For each row, the estimated position of the leftmost reseau is determined, and the approximate positions of the remaining eight reseaus on the row is computed using the estimated increment. The tape is advanced to the 64th line before the estimated line position of the leftmost reseau, and two lines of image data are read in. Two sets of 128 accumulators for image column and row sums are initialized for each one of the nine search regions.

Then additional data are read in one at a time, using a total of three buffers. As a new line is being read, the COMPASS subroutine XYGRAD is called for each search region in turn.

It uses the 129 pixels centered on the estimated pixel position of the reseau in each of the two lines already in core to compute the quantities

$$G_x(x,y) = \left[f(x,y) - f(x-1,y)\right] \left|f(x,y) - f(x-1,y)\right|$$

$$G_y(x,y) = \left[f(x,y) - f(x,y-1)\right] \left|f(x,y) - f(x,y-1)\right|$$

for each pixel location, and adds the gradient terms to the appropriate accumulator for that row or column of the search area. In these formulae, f(x, y) represents the gray level at pixel position x of line y of the input image, and the gradients G_x and G_y have been weighted by the absolute values of the differences in order to give extra emphasis to the large gradients associated with the edges of the reseau marks.

Lines are processed in this manner until 128 row sums (requiring 129 input lines) have been computed for each of the nine reseau search regions. For each reseau, the means and standard deviations about the means are computed separately for the set of row sums and the set of column sums. Each set is scanned for local minima at least 1.5 standard deviations below the mean, and local maxima at least 1.5 standard deviations above the mean. A bar of the reseau mark will be identified by a minimum which is followed by a maximum within two to eight locations; the mean bar position will be computed as the average of maximum and minimum positions plus 0.5 (since the nth gradient value is derived from pixels or lines n-1 and n). If one and only one pair of pixel and line positions is found, these will be stored in the reseau location table as "actual" locations; if no pair is found (or, very occasionally, more than one possible pair), zeros will be stored in the location table to show that the search for that particular reseau was unsuccessful.

GETRZO searches for the remaining eight rows of reseau marks in the same way as for the first row. However, the estimated pixel positions for reseaus in succeeding rows are taken equal to the actual position of the reseaus in the previous row, except for the reseaus which were not located. Similarly, after the second row, the leftmost reseau's line position is estimated using the actual separation between the two previous rows of reseaus in place of the original estimate. When all 81 reseaus have been searched for, GETRZO returns control to RZOMAP.

The subprogram POLY2 is then called to prepare a condensed location table containing only those reseaus whose actual locations were obtained by GETRZO. This table is passed to the subroutine TWOFIT, which fits third or fifth degree polynomials, as specified by the user, to the x displacements and y displacements of the actual reseaus relative to the nominal reseau positions. The coefficients for the two polynominals are then stored in COMMON by POLY2, and control is returned to RZOMAP.

The final step is performed by the subprogram NTRP2 which uses the polynomial coefficients to compute the locations of the missing reseaus and store them in the table. The table is copied onto the reseau location file on disk, and also written on the printer. Control is then returned to RZOMAP, which terminates the program and returns control to the IDAMS monitor.

4.24.2 Parameters

RZOMAP requires the following parameters:

1. IDEGR - 3 or 5 - desired degree of polynomial fit used for interpolation. Default = 3
2. MP - Estimated pixel location of midpoint of upper left reseau
3. ML - Estimated line location of upper left reseau
4. IP - Estimated spacing between reseaus along lines (pixel increment)
5. IL - Estimated spacing between reseaus perpendicular to scan lines (line increment)
6. NCOL - Number of columns of reseaus. Default = 9; maximum = 9
7. NROW - Number of rows of reseaus. Default = 9, maximum = 9
8. ICODE - Nominal reseau table indicator. Format is n + 10*m where m is the spectral band number (1, 2, or 3) and n = 0 if using stored

table, n = 1 if table is to be computed, n = 2 if user is supplying table.

9. LP(i), LL(i) - Pixel and line coordinates of nominal reseau locations, beginning with i = 1 for upper left, and proceeding from left to right along row and then to successively lower rows. *Default:* use stored table.

NOTE: A zero for any of the first eight parameters or for LP(1) is interpreted as a defaulted value. For MP, ML, IP, and IL, default values are obtained from upper left corner of table of nominal values.

4.24.3 Input

RZOMAP requires one input image, representing RBV sensor output, in standard IDAMS format.

4.24.4 Output

RZOMAP generates a table containing nominal and actual reseau locations which is stored in the reseau location file on disk and also listed on the printer. The disk file occupies disk cells (sectors) 5 to 10 and comprises one header record and 81 data records of four integer words each.

The header record format is

Word	Contents
1-2	Image name
3	Number of reseaus from GETRZO
4	Number of reseaus from NTRP2

Each data record corresponds to one reseau mark, starting with the top row and going from left to right within each row. The record format is

Word	Symbol	Contents
1	NOMP	Nominal pixel position
2	NOML	Nominal line position
3	IMP	Actual pixel position in image
4	IML	Actual line position in image

4.24.5 Example

An image JAMESBLU is to be scanned for reseau marks. The nominal locations are those of the standard stored table, and represent a 9 by 9 grid. Missing reseaus are to be filled in using a fifth degree polynomial fit. The upper left reseau is estimated to be at pixel 290 and line 340, and the spacing between reseau columns is approximately 405 pixels, and between rows, about 330 lines. Suitable IDAMS task and parameter cards are:

```
RZOMAP, (JAMESBLU, 49, 1),,, 1
```

```
5, 290, 340, 405, 330, 9, 9
```

4.24.6 Messages

In addition to messages generated by the general-purpose subroutines, RZOMAP generates the following fatal-error messages:

Message	Explanation
ESTIMATED RESEAU LOCATION OUTSIDE IMAGE	Specified line or pixel increment between reseaus too large, so that some reseaus lie outside input image
TOO FEW RESEAUS FOUND BY GETRZO	Not enough reseaus were identified by GETRZO to permit finding least-squares fit polynomial of specified degree.

4.24.7 Flowchart

See Appendix C, Figure C-24.

4.25 CORREL - IMAGE CORRELATION PROGRAM

4.25.1 Program Description

This program determines the relative positioning of a reference chip and an image segment for which the variance between the gray-level values, normalized to equal average gray level and standard deviation about the mean, is a minimum.

For each relative position of the reference image within a search area specified by the user, the variance is computed using the formula

$$V = 2 + 2\frac{\bar{x}\,\bar{r}}{\sigma_x \sigma_r} - \frac{2}{N}\frac{\Sigma x\,r}{\sigma_x \sigma_r}$$

where x and r represent gray-level values for the input image and reference image, respectively; $\bar{x}$ and $\bar{r}$ are the averages over N points defined by the dimensions of the reference image, and the standard deviations are computed by

$$\sigma_x^2 = \frac{1}{N}\Sigma x^2 - \bar{x}^2$$

$$\sigma_r^2 = \frac{1}{N}\Sigma r^2 - \bar{r}^2$$

In all three equations, the summation extends over the area of the reference image.

CORREL begins by reading in the reference image and computing $\bar{r}$ and σ_r, if $\sigma_r = 0$, an error exit is taken, since no correlation is possible. Lines of the specified segment of the input image are then read in until there are as many input lines in core as there are reference lines. The values of x and x^2 are added up for each column. To generate one line of the variance array, the COMPASS subroutine CROSS is called to add up the cross products (r x) for each horizontal position of the reference image relative to the input image. The column sums for x and x^2 are then added together for those columns overlaid by the first position of the reference image, and the first variance in the line computed. The x and x^2 sums are then modified to correspond to each successive position of the reference image and the corresponding variance values computed. If $\sigma_x = 0$ (perfectly uniform image), the variance is assigned the value 2.0. Each variance value is tested to see whether it is the smallest one so far; if so, its position in the matrix is recorded. The variance line is then saved on disk. The next line of the input image is then read in, and the column sums of x and x^2 modified to include the new line and exclude the former top line. The next line of the variance is then computed in the same way as before.

When the entire variance array has been computed, the 15 x 15 submatrix with the minimum value at its center is read into core. (The submatrix size is reduced if minimum is less than eight locations from edge of variance matrix.) If the minimum is within two locations of the edge, an advisory message is written. A test is also made for sharpness of the minimum by computing the average of the eight variance values adjacent to the minimum. This average is compared with the values of the 16 next-adjacent elements; if any is smaller than the average, a warning message is written.

The position of the center of the reference image relative to the input image for minimum variance is printed out, together with the variance value. The variance submatrix is also printed. If the user specified that the data should be stored, the image coordinates of the reference chip control point and the geographic and UTM coordinates read from the reference chip label are stored in the specified position on disk.

4.25.2 Parameters

CORREL requires the following parameters:

1.	MCP	- Central pixel location in reference (mask) image
2.	MCL	- Central line in mask
3.	NPM	- Number of pixels in mask - preferably odd
4.	NLM	- Number of lines in mask - preferably odd
5.	ISAVE	- 0 - correlation results are not saved on disk
		1 - the control point file on disk is cleared and correlation results written into record 1
		2 - correlation results are written into next available record (maximum 11)
		-n - results of this correlation replace record n of control point file

Note that the search area (variance matrix) size is defined by:

1. NCOL = NP - NPM + 1
2. NROW = NL - NLM + 1

4.25.3 Input

CORREL requires two input images. The secondary image is used as a reference image (mask) for correlation against the primary input image.

4.25.4 Output

CORREL prints out the minimum variance value, the corresponding position of the mask center point relative to the input image, and 15 x 15 submatrix of variance values around the minimum.

If ISAVE > 0, the position of the center point of the mask and its corresponding geographic and UTM coordinates (taken from the reference chip label record) are written onto record ISAVE of the control point file on disk.

4.25.5 Example

It is desired to correlate an 81 x 81 segment of reference chip WILDW23R, centered at pixel 51 and line 51, against an image FLGHTQ49. The control point is located at approximately pixel 2480 and line 210, and the search is to be carried out over an area up to 25 pixels and lines on all sides of this point. Then the total image segment to be searched has dimensions of 131 x 131 (81 + 2 x 25), with its upper left corner at pixel 2415 and line 145 (2480 - 131/2, etc.). Hence the task and parameter cards are:

```
CORREL, (FLIGHTQ49, 47, 1, WILDW23R, 48, 23), (2415, 145,131, 131),, 1
```

```
51,   51,   81,   81,   -4
```

The value ISAVE = - 4 will cause the result to be saved in record 4 of the control point file on disk.

4.25.6 Messages

CORREL may generate fatal error messages, as follows:

Message	Explanation
CORREL MASK OR SEARCH AREA TOO LARGE	The specified mask and search areas are too large for available core
CORREL SEARCH AREA NOT POSITIVE	NCOL and/or NROW computed from parameters are negative
MASK EXTENDS OUTSIDE REFERENCE IMAGE	Mask parameters not entered correctly
MASK IS ENTIRELY UNIFORM	Mask has no features at all; correlation not possible.

In addition, the following advisory messages may also be generated:

Message	Explanation
WARNING - VARIANCE MINIMUM IS AT OR VERY NEAR EDGE OF SEARCH AREA	The correlation may not be reliable due to edge effects.
WARNING - VARIANCE MINIMUM IS NOT SHARP	A well-defined minimum could not be found, results are questionable.
WARNING - THIS IS NOT A REFERENCE CHIP. RESULTS CANNOT BE STORED ON DISK SINCE CENTRAL POINT COORDINATES ARE NOT AVAILABLE.	The mask used was not in reference chip format, disk file data is incomplete.
WARNING - NAME OF CONTROL POINT FILE ON DISK, NOT NAME OF INPUT IMAGE. RESULTS HAVE NOT BEEN STORED.	Conflicting file names, data is ignored.
WARNING - SPECIFIED RECORD NUMBER GREATER THAN 1. NEW DATA HAVE NOT BEEN STORED.	Only ten control point files are permitted.
WARNING - CONTROL POINT FILE ON DISK ALREADY FULL. DATA HAVE NOT BEEN STORED.	Disk file is already full and space must be made available before more chips can be added.

4.25.7 Flowcharts

See Appendix C, Figure C-25.

4.26 RESECT - SPATIAL RESECTION PROGRAM

4.26.1 Program Description

This routine uses an iterative differential correction procedure to obtain corrections to the nominal spacecraft attitude and altitude. These minimize the variance between the observed image locations of selected control points, the locations computed using the corrected attitude and altitude parameters, and the known geographic coordinates of the control points. The corrected parameters are used to compute a grid of displacement values by means of which GEOMTRAN can transform the image to a UTM projection. When the resection has already been carried out for one spectral band of an RBV image, the precision parameters for that band may be used to go directly to computation of GEOMTRAN coordinates for additional bands. RESECT begins by accessing the parameters to determine the type of sensor being used, the scale of the final projection, and whether the UTM projection is to be aligned north-south or in the direction of the satellite heading. Tables of ephemeris and attitude data, control point data, and, for RBV images, reseau locations are then copied from the files stored on disk by IMERGE, RZOMAP, and CORREL (including modifications made by PPUPDATE or previous execution of RESECT), after first checking that the image name on disk agrees with the image specified as input. Since normal label checking cannot be carried out by BATCH, the window parameters (sp, sl, np, nl) are then checked for default, and if so, the full image is specified. A check is also made that enough control points are supplied to permit solving for all the resection parameters. Beyond this point, after certain constants have been computed, the procedure is sensor-dependent. For RBV images, RESECT continues as described below.

The latitude Φ, longitude L, and elevation H for each control point are then converted to coordinates (x_e, y_e, z_e) relative to the satellite nadir point $(\Phi_n, L_n, 0)$, where x_e is positive in the direction of satellite heading and z_e is positive upwards normal to the plane tangent to the earth ellipsoid at the nadir point (the <u>nadir plane</u>). The transformation equations are

$$\begin{pmatrix} x_e \\ y_e \\ z_e \end{pmatrix} = \left(b_{ij}\right) \begin{pmatrix} U - U_n \\ V - V_n \\ W - W_n \end{pmatrix}$$

where

$$U = (r + H) \cos \Phi \cos (L-L_n)$$

$$V = (r + H) \cos \Phi \sin (L-L_n)$$

$$W = [r (1-e^2) + H] \sin \Phi$$

and (U_n, V_n, W_n) are the values of the nadir point. The constants describing the earth ellipsoid are

$$e^2 = \text{eccentricity squared}$$
$$r = a (1 - e^2 \sin^2 \Phi)^{-1/2}$$
$$a = \text{semimajor axis of ellipsoid}$$

The elements of the rotation matrix are given by

$$(b_{ij}) = \begin{pmatrix} \cos \alpha' \sin \Phi_n & -\sin\alpha' & -\cos \alpha' \cos \Phi_n \\ \sin \alpha' \sin \Phi_n & \cos \alpha' & -\sin\alpha' \cos \Phi_n \\ \cos \Phi_n & 0 & \sin \Phi_n \end{pmatrix}$$

where α' = satellite heading from south (compass bearing -180°).

If a secondary RBV image has been specified, only the transformation coefficients are computed, and control then jumps to the procedures for generating the parameter grid for GEOMTRAN. Otherwise, RESECT proceeds to compute corrected attitude. For RBV images from the ERTS satellites, for which the small uncertainties in ϕ_n and L_n can be adequately taken into account by lumping them with the roll and pitch errors, the corrections are computed in the following manner.

The observed image coordinates (x_i, y_i) of each control point (normally obtained by CORRELating against a reference chip) are converted to face-place coordinates (x_f, y_f) using the table of actual and observed reseau positions, where the actual positions are assumed to include corrections for lens distortions. This is accomplished by determining the image coordinates $(\bar{x}_i, \bar{y}_i)$ of the nearest reseau mark by finding the reseau mark for which $(x_i - \bar{x}_i)^2 + (y_i - \bar{y}_i)^2$ is a minimum. The next nearest reseaus (x_i', y_i') on the same row as $(\bar{x}_i, \bar{y}_i)$ and (x_i'', y_i'') on the same column are then found (using next reseau to right if $x_i - \bar{x}_i$ is positive, and so on). Defining the vectors

$$\Delta_i = (x_i - \bar{x}_i, y_i - \bar{y}_i)$$

$$D_i' = (x_i' - x_i, y_i' - y_i)$$

$$D_i'' = (x_i'' - x_i, y_i'' - y_i)$$

and, similarly, Δ_f, D_f', and D_f'' for the corresponding points in faceplace coordinates, the location of the control point in faceplate coordinates is computed by linear interpolation, using the components of Δ_i normal to D_i' and D_i'', as

$$\begin{pmatrix} x_f \\ y_f \end{pmatrix} = \begin{pmatrix} x_f \\ y_f \end{pmatrix} + \frac{\Delta_i \times D_i''}{D_i' \times D_i''} D_f' + \frac{\Delta_i \times D_i'}{D_j'' \times D_i'} D_f''$$

where the symbol x denotes the vector cross-product.

The transformation of the true locations of the control points from the earth nadir-point coordinates (x_e, y_e, z_e) to faceplate coordinates is then carried out in two steps using the nominal attitude and altitude parameter values. The first step obtains the coordinates (x_s, y_s, z_s) of the control point relative to the sensor, where the sensor axes, initially parallel to the x_e, y_e, z_e axes, have been rotated through yaw angle κ, roll angle ω, and pitch angle φ. Applying the rotations gives

$$\begin{pmatrix} x_s \\ y_s \\ z_s \end{pmatrix} = R_\varphi \; R_\omega \; R_\kappa \begin{pmatrix} x_e \\ y_e \\ z_e - h \end{pmatrix}$$

where the product of the three rotation matrices is

$$R_\varphi R_\omega R_\kappa = \begin{pmatrix} \cos\varphi\cos\kappa - \sin\varphi\sin\omega\sin\kappa & \cos\varphi\sin\kappa + \sin\varphi\sin\omega\cos\kappa & -\sin\phi\cos\omega \\ -\cos\omega\sin\kappa & \cos\omega\cos\kappa & \sin\omega \\ \sin\varphi\cos\kappa + \cos\varphi\sin\omega\sin\kappa & \sin\varphi\sin\kappa - \cos\varphi\sin\omega\cos\kappa & \cos\varphi\cos\omega \end{pmatrix}$$

and

h = altitude of spacecraft above nadir plane.

The faceplate coordinates are then

$$x_f = \bar{x}_f + \frac{f}{|z_s|} y_s = \bar{x}_f - \frac{f}{z_s} y_s$$

$$y_f = \bar{y}_f + \frac{f}{|z_s|} x_s = \bar{y}_f - \frac{f}{z_s} x_s$$

where

f = focal length of RBV camera

($\bar{x}_f$, $\bar{y}_f$) = faceplate coordinates of point on camera axis in image and faceplate coordinates, the x_i or x_f axis is transverse to the satellite heading, rather than coincident with it, and the x_f and y_f axes are positive in the $+y_x$ and $+x_s$ directions, respectively.

RESECT then computes differential corrections to the values of κ, ω, φ, and h by requiring that the variance V between the faceplate coordinates $(\overset{o}{x}_f, \overset{o}{y}_f)$ computed from the observed image locations of the control points and the coordinates (x_f, y_f) computed using the nominal attitude and altitude parameters be a minimum; V is defined as

$$V = \frac{1}{N} \Sigma \left[(x_f - \overset{o}{x}_f)^2 + (y_f - \overset{o}{y}_f)^2 \right]$$

where the summation extends over the entire set of N control points.

The conditions for V to be a minimum are

$$\frac{\partial V}{\partial \kappa} = \frac{\partial V}{\partial \omega} = \frac{\partial V}{\partial \varphi} = \frac{\partial V}{\partial h} = 0$$

To simplify the computation of these derivatives, x_f and y_f are expanded around their values for the nominal parameter values $\bar{\kappa}$, $\bar{\omega}$, $\bar{\varphi}$, and $\bar{h}$, giving

$$x_f \doteq x_f(\bar{\kappa}, \bar{\omega}, \bar{\varphi}, \bar{h}) + \frac{\partial x_f}{\partial \kappa}(\kappa - \bar{\kappa}) + \frac{\partial x_f}{\partial \omega}(\omega - \bar{\omega}) + \frac{\partial x_f}{\partial \varphi}(\varphi - \bar{\varphi}) + \frac{\partial x_f}{\partial h}(h - \bar{h})$$

$$y_f \doteq y_f(\bar{\kappa}, \bar{\omega}, \bar{\varphi}, \bar{h}) + \frac{\partial y_f}{\partial \kappa}(\kappa - \bar{\kappa}) + \frac{\partial y_f}{\partial \omega}(\omega - \bar{\omega}) + \frac{\partial y_f}{\partial \varphi}(\varphi - \bar{\varphi}) + \frac{\partial y_f}{\partial h}(h - \bar{h})$$

For further simplifications, the derivatives of x_f and y_f are approximated at the average values of attitude and altitude rather than the nominal values for the instant at which the image was recorded, i.e., $\kappa = \omega = \varphi = 0$, $h = h_o$. Then using

$$dx_f = \frac{f}{z_s} dy_s + \frac{y_s f}{z_s^2}\left(dz_s\right)$$

$$dy_f = \frac{f}{z_s} dx_s + \frac{x_s f}{z_s^2}\left(dz_s\right)$$

and the infinitesimal rotation from (x_e, y_e, z_e) to (x_s, y_s, z_s) in the form

$$dx_s = y_e\, d\kappa + h_o\, d\varphi$$

$$dy_s = -x_e d\kappa - h_o d\omega$$

$$dz_s = -y_e d\omega + x_e d\varphi - dh$$

we obtain

$$dx_f = -\frac{f}{h_o} x_e d\kappa - (f + \frac{f}{h_o^2} y_e^2)\ d\omega + \frac{f}{h_o^2} x_e y_e d\varphi - \frac{f}{h_o^2} y_e dh$$

$$dy_f = +\frac{f}{h_o} y_e d\kappa - \frac{f}{h_o^2} x_e y_e\, d\omega + (f + \frac{f}{h_o^2} x_e^2)\ d\varphi - \frac{f}{h_o^2} x_e dh$$

from which the required (approximate) derivatives may be extracted. By introducing the notation

$$\alpha_1 = \kappa,\ \alpha_2 = \omega,\ \alpha_3 = \varphi,\ \alpha_4 = h$$

$$\Delta\alpha_j = \alpha_j - \bar{\alpha}_j\ ,\quad j = 1,4$$

we can write the conditions for minimum variance as

$$\frac{\partial V}{\partial \alpha_j} = 0 = 2\Sigma\left[(x_f - x_f^o)\frac{\partial x_f}{\partial \alpha_j} + (y_f - y_f^o)\frac{\partial y_f}{\partial \alpha_j}\right], \quad j = 1,4$$

which, with the aid of the expansions around the nominal values of the $\{\alpha_j\}$, becomes

$$\sum_{k=1}^{4}\left\{\sum_{\substack{\text{control}\\\text{points}}} \left(\frac{\partial x_f}{\partial \alpha_j}\frac{\partial x_f}{\partial \alpha_k} + \frac{\partial y_f}{\partial \alpha_j}\frac{\partial y_f}{\partial \alpha_k}\right)\right\} \Delta\alpha_k$$

$$= \sum_{\substack{\text{control}\\\text{points}}}\left\{\left[x_f^o - x_f(\bar{\kappa},\bar{\omega},\bar{\varphi},\bar{h})\right]\frac{\partial x_f}{\partial \alpha_j}\right.$$

$$\left. + \left[y_f^o - y_f(\bar{\kappa},\bar{\omega},\bar{\varphi},\bar{h})\right]\frac{\partial y_f}{\partial \alpha_j}\right\}, \quad j = 1,4$$

After all the terms have been evaluated, the set of four simultaneous linear equations for the four corrections $\{\Delta\alpha_j\}$ are solved by a call to the general purpose subroutine MATINV. The corrected values of $\{\alpha_j\}$ are used to compute new values of (x_f, y_f) for each control point. The variance is computed; if its square root has decreased by less than 0.1 resolution elements compared to the previous variance, and fewer than 5 iterations have been performed, a further set of corrections to the new values of $\{\alpha_j\}$ is computed.

Otherwise, the final values of κ, ω, φ, and h are printed out, together with a measure of goodness of fit defined by

$$m = \left[V/2N - 4)\right]^{1/2}$$

and the contributions to the variance by each control point. The corrected parameters are also enterd in the ephemeris file on disk.

Finally, RESECT generates a grid of image points and their corresponding UTM coordinates to be used as input parameters for the geometric transformation routine, GEOMTRAN.

This procedure begins by computing the image coordinates of a 10 by 10 grid of calibration points, in the form of 81 identical rectangles which are just large enough to completely cover the image subset specified by the window parameters.

The UTM coordinates of the corresponding point on the ground are then computed for each calibration point in turn. This procedure begins by computing the faceplate coordinates of the calibration point using the same procedure as before.

Because the values of z_s for these points are not known a priori, the corresponding values of (x_s, y_s) are then computed from

$$x_s = -\frac{z_s}{f}(y_f - \bar{y}_f)$$

$$y_s = -\frac{z_s}{f}(x_f - \bar{x}_f)$$

using $z_s = -h$ as a first approximation. The corresponding values of (x_e, y_e, z_e) are then computed using the inverse of the rotation $R\varphi\, R\omega\, R\kappa$; this inverse is given simply as the transpose of the matrix.

From this approximate set of (x_e, y_e, z_e), the corresponding values of U, V, and W are obtained by an inverse rotation through angles Φ_n and α'. Latitude and longitude are then computed using

$$\tan(L - L_n) = V/U$$

$$\tan\Phi = \frac{W}{(1-e^2)(U^2 + V^2)^{1/2}}$$

$$H = 0$$

Because of the apprixomate computation of (x_e , y_e , z_e), the values of L and ϕ are not exact. Instead, an iterative procedure is used in which this trial set of (L, ϕ, H) is transformed back to (x_f, y_f). The working values of (x_f, y_f) are incremented by the residual error between the true value and the value obtained from (ϕ, L, H), and the transformation from faceplate coordinates to approximate geographic and back to faceplate is repeated. This process continues until the residual errors are less than 0.1 resolution element.

Finally, the resulting geographic coordinates are converted to the UTM grid zone containing the nadir point by a call to UTMCON.

If the user has specified north-south alignment of the output grid, the value of α' is checked to see whether it is between 90° and 270°, in which case the grid will be oriented with line number increasing from south to north, or between 90° and -90°, in which case normal map orientation will be used. A warning message will be written if this requires rotation through more than 8.1 degrees, in which case GEOMTRAN execution time may be excessive.

The UTM value for points along the top and left-hand edge of the grid are scanned to find the largest northing and smallest easting value for any point (for normal map orientation) or smallest northing and largest easting (for inverted orientation). These values are rounded to the nearest multiples of 1000 meters which lie outside the entire grid. The rounded values are taken as the origin of output coordinates, corresponding to pixel 1 of line 1.

The UTM coordinates for each grid point are then converted to output image coordinates by determining x and y displacements from the origin in meters, dividing by the user-supplied scale factor, and rounding to the nearest integer.

If the user has instead specified alignment along the spacecraft line of flight, rotated easting and northing values are obtained by

$$E' = -(E - E_n) \cos \alpha' - (N - N_n) \sin \alpha'$$

$$N' = (E - E_n) \sin \alpha' - (N - N_n) \cos \alpha'$$

where (E_n, N_n) are the coordinates of the nadir point (obtained, if necessary, by a call to UTMCON). The topmost and leftmost values in rotated coordinates are rotated back to standard UTM, rounded to a multiple of 1000 meters, and rotated back to the coordinates aligned along the track. Output image coordinates are then computed from the rotated grid coordinates. The output and input image coordinates of each grid point are then entered in the table. A table of linkages, specifying all pairs of grid points which are adjacent along either rows or columns, is also generated. These tables are printed out, and also stored on disk cells 13 to 24 for access by GEOMTRAN. The alignment angle of the output grid, the UTM coordinates of its origin of coordinates, the scaling factor, and the UTM coordinates of the nadir point are also printed out.

For MSS images, the procedures are quite similar to those for RBV. However, account must be taken of the motion of the satellite, and hence of the nadir point, during the time required to scan one frame. In addition, the rotation of the earth and the non-linear relation between the scanner mirror angle and pixel number (i.e., elapsed time from the start of the scanning of the line) must be allowed for.

For each control point, the time difference between control point acquisition and format center time

$$\Delta t = T_{scan}(l_{cp} - l_{fc})$$

is computed, where the average interval between line scans is $T_{scan} = (73.42/6)$ millisec, l_{cp} is the line number, and l_{fc} is the line number of the format center. The geographic coordinates (Φ_s, L_s, H_s) for the subsatellite point at the time at which the control point was scanned are computed from the format-center nadir point coordinates (Φ_n, L_n, 0) by

$$\Phi_s = \Phi_n - \frac{1 + (1 - e^2) \tan^2 \Phi_n}{r(1 - e^2)(1 + \tan^2 \Phi_n)} \cos \alpha' V_{track} \Delta t$$

$$L_s = L_n - \frac{1}{r \cos \Phi_n} \sin \alpha' V_{track} \Delta t - \Omega_e \Delta t$$

$$H_s = 0$$

where r, e^2, and α' have the same meanings as for the RBV resection, V_{track} is the velocity of the subsatellite point, and $\Omega_e = 2\pi/86164$ rad/sec is the earth's rotational angular velocity.

Using these values of (Φ_s, L_s) in place of (Φ_n, L_n), transformations to geocentric coordinates (U, V, W) and subsatellite-point coordinates, (x_e, y_e, z_e) are carried out in the same manner as for RBV control points; however, a new set of (b_{ij}) must be computed for each control point.

The measured image position for each control point is then converted to apparent coordinates (x_p^o, y_p^o) of the point when projected into the plane $z_s = h_o$ in the coordinate system (x_s, y_s, z_s) which rotates with the satellite (the $-z_s$ axis is the line of sight of the scanner at mid scan, and the x_s axis lies in the plane of the z_s axis and the spacecraft velocity, roughly coinciding with the latter). The conversion equations are

$$x_p^o = 0$$

$$y_p^o = h_o \tan \theta$$

where the non-linear relationship between time (pixel position p_p) and scan-mirror angle θ (measured from mid-scan) is expressed as

$$\theta = \frac{c_1}{c_2} \sin(c_2 p_p + e_3) + c_4$$

The coordinates computed from the actual position relative to the subsatellite point are then computed by

$$\begin{pmatrix} x_s \\ y_s \\ z_s \end{pmatrix} = R_\psi R_\omega R_\kappa \begin{pmatrix} x_e \\ y_e \\ z_e - h \end{pmatrix}$$

in the same way as for the RBV sensor, with the exception that now the attitude and altitude parameters are time varying. The approximate values of each of these 4 parameters at 9 different times, representing time intervals of 3.45 sec from 13.8 sec before until 13.8 sec past the format-center time, are input via the ephemeris file on disk as the parameters A_{ij}, $i = 1.4$, $j = 1,9$. In addition, after the first iteration of the resection procedure each of the four attitude/altitude parameters has a constant correction a_i and an additional time-dependent correction $b_i \Delta t$ associated with it. Thus the nominal values of the attitude/altitude parameters at time Δt relative to format center are given, using linear interpolation between A_{ij} and $A_{i,j+1}$ by

$$\bar{\alpha}_i = A_{ij} \frac{\Delta t - 3.45(j-5)}{3.45} + A_{i,j+1} \frac{3.45(j-4) - \Delta t}{3.45} + a_i + b_i \Delta t$$

where the value of j is computed as

$$j = \frac{\Delta t + 17.25}{3.45} \qquad \text{(integer part only)}$$

The projected coordinates are then computed by

$$x_p = -h_o x_s / z_s$$

$$y_p = -h_o y_s / z_s$$

where the minus signs are required because z_s is negative.

The actual resection will be carried out by finding those corrections to the attitude and altitude parameters which will minimize the variance between the values (x_p^o, y_p^o) derived from the control-point image coordinates and the values (x_p^o, y_p^o) derived from the geographic coordinates by means of the α_i, i.e.,

$$V = \sum_{\substack{\text{control}\\\text{points}}} \left[(x_p - x_p^o)^2 + (y_p - y_p^o)^2 \right]$$

is to be minimized by requiring

$$\frac{\partial V}{\partial a_i} = \frac{\partial V}{\partial b_i} = 0, \quad i = 1, 4$$

The required derivatives of V are obtained by expanding x_p and y_p as

$$x_p \doteq \bar{x}_p + \sum_{i=1}^{4} \frac{\partial x_p}{\partial a_i} \Delta a_i + \sum_{i=1}^{4} \frac{\partial x_p}{\partial b_i} \Delta b_i$$

$$y_p \doteq \bar{y}_p + \sum_{i=1}^{4} \frac{\partial y_p}{\partial a_i} \Delta a_i + \sum_{i=1}^{4} \frac{\partial y_p}{\partial b_i} \Delta b_i$$

where

$$\bar{x}_p = x_p (A_{ij}, \Delta t; \bar{a}_i, \bar{b}_i)$$
$$\bar{y}_p = y_p (A_{ij}, \Delta t; \bar{a}_i, \bar{b}_i)$$
$$\Delta a_i = a_i - \bar{a}_i$$
$$\Delta b_i = b_i - \bar{b}_i$$

and $\bar{a}_i$, $\bar{b}_i$ represent the nominal values of the corrections.

The derivatives of the projected coordinates are related to the satellite coordinates by

$$dx_p = -(h_o / z_s)\, dx_s + (h_o x_s / z_s^2)\, dz_s$$

$$dy_p = -(h_o / z_s)\, dy_s + (h_o x_s / z_s^2)\, dz_s$$

For small deviations of the attitude/altitude variables from their average values ($\kappa = \omega = \varphi = 0$, $h = h_o$), the derivatives of (x_s, y_s, z_s) are the same as for the RBV; allowing for the time dependence for the MSS case, this gives

$$dx_s \doteq y_e(da_1 + \Delta t db_1) + h_o(da_3 + \Delta t db_3)$$

$$dy_s \doteq -x_e(da_1 + \Delta t db_1) - h_o(da_3 + \Delta t db_2)$$

$$dz_s \doteq -y_e(da_2 + \Delta t db_2) + x_e(da_3 + \Delta t db_3) - (da_4 + \Delta t db_4)$$

Hence, for the average values of the α_i, the variations of (x_p, y_p) with the (a_i, b_i) are given by

$$dx_p = y_e(da_1 + \Delta t db_1) - \frac{x_e y_e}{h} (da_2 + \Delta t db_2)$$
$$+ (h_o + \frac{x_e^2}{h_o}) \; (da_3 + \Delta t db_3) - \frac{y_e}{h_o} (da_4 + \Delta t db_4)$$

$$dy_p = -x_e(da_1 + \Delta t db_1) - (h_o + \frac{y_e^2}{h_o})\ (da_2 + \Delta t db_2)$$
$$+ \frac{x_e y_e}{h_o}\ (da_3 + \Delta t db_3) - \frac{y_e}{h_e}\ (da_4 + \Delta t db_4)$$

Accordingly, these relations are used to compute the derivatives for each control point, which are then used to compute the coefficients of the system of eight linear equations in the eight variables (Δa_i, Δb_i, $i = 1, 4$), obtained from the conditions for minimum variance, in the form

$$\sum_{k=1}^{4} \left[\sum_{\substack{\text{control}\\ \text{points}}} \left(\frac{\partial x_p}{\partial a_i} \frac{\partial x_p}{\partial a_k} + \frac{\partial y_p}{\partial a_i} \frac{\partial y_p}{\partial a_k} \right) \right] \Delta a_k$$
$$+ \sum_{k=1}^{4} \left[\sum_{\substack{\text{control}\\ \text{points}}} \left(\frac{\partial x_p}{\partial a_i} \frac{\partial x_p}{\partial b_k} + \frac{\partial y_p}{\partial a_i} \frac{\partial y_p}{\partial b_k} \right) \right] \Delta b_k$$
$$= \sum_{\substack{\text{control}\\ \text{points}}} \left[\left(x_p^o - \bar{x}_p \right) \frac{\partial x_p}{\partial a_i} + \left(y_p^o - \bar{y}_p \right) \frac{\partial y_p}{\partial a_i} \right],\ i = 1, 4$$

and

$$\sum_{k=1}^{4} \left[\sum_{\substack{\text{control}\\ \text{points}}} \left(\frac{\partial x_p}{\partial b_i} \frac{\partial x_p}{\partial a_k} + \frac{\partial y_p}{\partial b_i} \frac{\partial y_p}{\partial a_k} \right) \right] \Delta a_k$$
$$+ \sum_{k=1}^{4} \left[\sum_{\substack{\text{control}\\ \text{points}}} \left(\frac{\partial x_p}{\partial b_i} \frac{\partial x_p}{\partial b_k} + \frac{\partial y_p}{\partial b_i} \frac{\partial y_p}{\partial b_k} \right) \right] \Delta b_k$$
$$= \sum_{\substack{\text{control}\\ \text{points}}} \left[\left(x_p^o - \bar{x}_p \right) \frac{\partial x_p}{\partial b_i} + \left(y_p^o - \bar{y}_p \right) \frac{\partial y_p}{\partial b_i} \right],\ i = 1, 4$$

The solutions are obtained by a call to MATINV.

For the first iteration, initial values of $\bar{a}_i = \bar{b}_i = 0$ are assumed. Before each subsequent iteration, these estimates are updated by the relations

$$\bar{a}_i + \Delta a_i \rightarrow \bar{a}_i$$
$$\bar{b}_i + \Delta b_i \rightarrow \bar{b}_i$$

where Δa_i , Δb_i , $i = 1 , 4$ are the solutions obtained during the previous iteration. After each repetition, the variance, expressed in terms of resolution elements, is compared with the variance from the previous iteration. When the difference is less than 0.01 squared resolution elements, or after five iterations, the results are written out, together with the quality measure defined for the RBV case. The values of a_i and b_i are also used to correct the sets of nine values for each parameter, A_{ij} , before they are written back onto the disk.

A 10 by 10 grid of image points is then created in the same manner as for the RBV resection to initiate the computation of GEOMTRAN coefficients. For each image point, the corresponding values of (x_p^o , y_p^o) are computed. These are used as test values (x_p, y_p) to compute approximate values of (x_s, y_s, z_s) using

$$x_s = x_p$$
$$y_s = y_p$$
$$z_s = -h_o$$

The inverse rotations to (x_e, y_e, z_e) and then to (Φ, L, H) are carried out using matrix values appropriate to the Δt computed from the image line number. H is set equal to zero, and the approximate geographic coordinates are transformed back to the satellite coordinate system, giving values $(\bar{x}_s, \bar{y}_s, \bar{z}_s)$ from which projected coordinate values $(\bar{x}_p, \bar{y}_p)$ are computed;

at each step, the same formulae are used that were specified for the control point transformation.

The squared distance between approximate and true projections,

$$d^2 = (\bar{x}_p - x_p^o)^2 + (\bar{y}_p - y_p^o)^2$$

is computed. If it exceeds 0.01 squared resolution elements, the estimated values of (x_p, y_p) are updated by

$$x_p - (\bar{x}_p - x_p^o) \rightarrow x_p$$
$$y_p - (\bar{y}_p - y_p^o) \rightarrow y_p$$

and the approximate transformation to geographic coordinates and exact transformation back to projected satellite coordinates is repeated.

When the distance is less than the test value, the values (Φ, L, 0) are transformed to UTM coordinates by calling UTMCON. After all 100 grid points have been mapped into UTM coordinates, the actual generation of the output grid is carried out in exactly the same way as for the RBV case.

4.26.2 <u>Parameters</u>

RESECT requires 4 parameters, followed by a table of boresight offsets, if required. The parameters are

1. ISENSOR - RBV, MSS, or other allowed sensor type of up to four characters, beginning with alphabetic
2. ISCALE - Number of meters per resolution element (pixel and line separation) in output grid; integer format
3. IALIGN - Alignment of output grid
   0 or default - along spacecraft track
   1 - along north-south axis

4. IBAND - For RBV only
Flag to indicate whether first band for this image or an additional band
0 or default - first band, do full resection
1-9 - Use results, on disk, of previous resection and stored table 1 to 9, respectively, for boresight offsets
10 - Use offsets entered via parameters

When boresight offsets are to be entered as parameters, three integer values are required, giving the corrections which must be added to the attitude parameters obtained by spatial resection on the initial band. The corrections are given in this order:

1. IYAW - Correction (micro radians) to be added to yaw

2. IROLL - Correction (micro radians) to be added to roll

3. IPITCH - Correction (micro radians) to be added to pitch

4.26.3 Input

Although RESECT does not access image data, it requires specification of the image name and the rectangle for which a geometric transformation grid is to be generated; the values of LUNIN and FILEIN may be arbitrary, since they are not used.

In addition, disk files of ephemeris data, control point data, and reseau data (RBV only) are read as input. Corrections and additions to these tables may be entered by means of PPUPDATE.

Note that these disk files may sometimes be saved temporarily on a tape; in this case, they must be reloaded onto disk by a preliminary execution of FPCON, using the following task and parameter cards:

```
FPCON, (tapename, LUNIN, FILEIN),,, 1
```

```
16, 24, 6
```

C3

4.26.4 Output

The principal output from RESECT is a table stored on disk cells 13 to 24 which describes the image coordinates and corrected UTM grid coordinates of a 10 by 10 grid of calibration points. This file consists of a four-word header record, 100 four-word location data records, and 180 two-word linkage records, as follows:

Header record

Word	Contents
1-2	Image name
3	Number of grid points (always 100 for RESECT output)
4	Number of linkages (always 180 for RESECT output)

Location records

Word	Contents
1	Grid pixel position
2	Grid line position
3	Image pixel position
4	Image line position

Linkage records

Word	Contents
1	Grid point (numbered in order of appearance in location table) at start of linkage line
2	Grid point at end of linkage line

For RESECT, the linkages are always the same: first, all pairs of adjacent points in the top grid line, going from left to right, are linked, and then all linkages from points in the top line to the points in the same column on the second line. Then the same linkages for the second line, and so on for successive grid lines. Hence the table will be:

1,2	2,3	3,4	4,5	5,6	6,7	7,8	8,9	9,10	
1,11	2,12	3,13	4,14	5,15	6,16	7,17	8,18	9,19	10,20
11,12	12,13	.							
.	.	.							
81,91	82,92	83,93	84,94	85,95	86,96	87,97	88,98	89,99	90,100
91,92	92,93	93,94	94,95	95,96	96,97	97,98	98,99	99,100	

The location records are also written out on the printer.

When the full resection is performed, the altitude and attitude values and an indication of their accuracy are listed on the printer. The precision attitude and altitude values are also entered on the ephemeris file on disk, in place of the approximate values.

4.26.5 Example

An RBV image WALLOPSI is to be processed by RESECT using the ephemeris data stored on disk by IMERGE, the control point data stored by CORREL, and the reseau data stored by RZOMAP. The task and parameter cards are:

```
RESECT, (WALLOPSI, 49, 1), (1, 1, 4500, 3600),, 1
```

```
RBV, 75, 1
```

where the last parameter has been defaulted.

Note that the scale distance for separation between picture elements (resolution elements) is given in meters; for ERTS images the following values, with their equivalents in feet, are likely to be appropriate:

Meters	Feet
75	246
80	262
85	279

4.26.6 Messages

RESECT generates the following warning message:

WARNING - OUTPUT MAY REQUIRE EXCESSIVE ROTATION TO ACHIEVE NORTH-SOUTH ALIGNMENT

RESECT generates the following fatal messages:

Message	Explanation
DISK TABLE FOR WRONG IMAGE	The image name in the header of one of the tables on disk is not the same as the specified input image.
INVALID SENSOR TYPE	Specified sensor not RBV.
TOO FEW CONTROL POINTS	Not enough control points were used to permit solving for the resection variables
SP, SL, SPECIFY POINT OUTSIDE IMAGE	Value of SP or SL less than 1 or greater than maximum image size.
COEFF MATRIX SINGULAR ON FIRST PASS	Most probable cause: control points too close together or three or more on same straight line.

4.26.7 Flowchart

See Appendix C, Figure C-26.

4.27 UTMGEO - UTM GEOGRAPHIC COORDINATE CONVERSION PROGRAM

4.27.1 Program Description

This routine provides a capability for transforming UTM grid coordinates into geographic latitude and longitude, and vice versa. The two types of conversions may be intermixed during a single execution of the program.

UTMGEO begins by scanning the parameter data for keywords, which indicate which type of coordinates are being input. The keywords are identified, and the coordinate values are converted to internal format. The general purpose subroutine UTMCON is then called to carry out the conversion. The results are then converted back to standard format, and both the latitude and longitude and the UTM grid coordinates are printed out. Additional pairs of coordinates are processed in the same manner until the specified number of coordinate pairs have been converted.

4.27.2 Parameters

In order to identify the type of coordinates, keyword identifiers must precede each numerical value; successive parameters must be separated by commas. The parameters are as follows:

1. LAT = dd, mm, ss. s_{S}^{N} - Geographic latitude. If the N/S designator is omitted, north latitude is assumed unless the value is preceded by a minus sign.

2. LONG = ddd, mm, ss. s_{W}^{E} - Geographic longitude. If the E/W designator is omitted, east longitude is assumed unless the value is preceded by a minus sign.

3.	ZONE	=	nn_{S}^{N}	- UTM zone number. If the N/S designator is omitted, north is assumed unless the number is preceded by a minus sign.
4.	UTME	=	nnnnnn	- UTM easting, with zone central meridian = 500,000 meters
5.	UTMN	=	nnnnnnn	- UTM northing, with equator = 0 meters in northern hemisphere. In souther hemisphere, a false northing of 10 000 000 is added, so that grid distance from equator = UTMN -10^7 meters.

4.27.3 Input

UTMGEO requires no input other than parameters.

4.27.4 Output

UTMGEO produces a printed listing of the geographic latitude and longitude and UTM zone, easting, and northing for each coordinate pair entered as parameters.

4.27.5 Example

It is desired to convert three sets of coordinates from Lat/Long to UTM and two sets from UTM to Lat/Long. Appropriate IDAMS batch processor task and parameters cards are:

```
UTMGEO,,,,5

LAT = 26, 16, 24.5N, LONG=127, 48, 12.4E,

ZONE = 33S, UTME=764329, UTMN=9784625,

ZONE = -33, UTME=765228, UTMN=9784007,

LAT = -00, 18, 11.4, LONG=33, 18, 12.0,

LAT = 56, 46, 58.8S, LONG=726, 12, 04.0W
```

4.27.6 Messages

UTMGEO generates one message:

AAAA IS NOT A KEYWORD	Keyword is not valid, processing skips to next parameter set.

4.27.7 Flowchart

See Appendix C, Figure C-27.

4.28 FPMULT - FLOATING-POINT ARRAY MULTIPLICATION PROGRAM

4.28.1 Program Description

This task routine forms the products of corresponding elements of two floating point arrays, one of them stored on tape and one on disk. The two arrays may be both real, both complex, or one (on tape) real and the other (on disk) complex. The complex arrays must be in IDAMS symmetric half-array format, and may be in real-plus-imaginary or modulus-plus-phase format. When both arrays are complex, they must be in the same format (a preliminary execution of FPCON option 7 or 17 may be used to convert one array). When a complex array is multiplied by a real array, the latter is assumed to have its origin at the corner (FPCON option 9 may be used for conversion from center-origin form) and to be symmetric, so that only the left-hand half array need be used. The product array is left on disk in place of the input array.

FPMULT begins by determining the types of arrays to be multiplied, and checking the label of the array on tape to ensure that it is the specified size. Minimum and maximum test values are initialized.

Data are read double-buffered from tape and single buffered from disk one line at a time. Corresponding elements are multiplied together using logic appropriate to the type of data, all result values are tested against the current maximum and minimum values, and the line of products written back onto disk in place of the last input line.

When all lines of the product array have been computed, the maximum and minimum values are written into the disk cell following the end of the last array line, and control is returned to the IDAMS monitor.

4.28.2 Parameters

FPMULT requires the following parameters:

1.	NX	-	Number of complex values or one-half number of real values per line of both input arrays
2.	NY	-	Number of lines in both input arrays
3.	ITYP	-	Types of arrays (integer value with following meanings)

1 - Both Real, both corner origin or both center origin

2 - Both complex, real-plus-imaginary format

3 - Both complex, modulus-plus-phase format

4 - One real array, corner origin and one complex, real-plus-imaginary format

5 - One real array, corner origin and one complex, modulus-plus-phase format

4.28.3 Input

FPMULT requires two floating-point input arrays, one on tape (primary input) in standard IDAMS format and one on disk. If the two arrays are not of the same type, the primary input (tape) array must be real. The SP, SL, NP, NL field on the task card is ignored, since the entire array is always used and its size is specified by the parameters NX, NY.

4.28.4 Output

FPMULT generates one floating-point output array on disk; it is in the same form as, and replaces, the secondary input array.

4.28.5 Example

A complex real-plus-imaginary symmetric half-array on disk, having 64 lines of 32 complex values each, is to be multiplied by a real array on tape named FLTR16B. Suitable IDAMS task and parameter cards would be:

```
FPMULT, (FLTR16B, 49, 1),,, 1
```

```
32       64    4
```

4.28.6 Messages

FPMULT generates the following fatal-error message:

Message	Explanation
PRIMARY INPUT ARRAY WRONG SIZE	The size parameters on the input tape label do not satisfy NP=16*NX, NL=NY

4.28.7 Flowchart

See Appendix C, Figure C-28.

4.29 FPSUM - FLOATING-POINT SUMMATION PROGRAM

4.29.1 Program Description

This task routine computes the sum or difference of two floating-point arrays. Either or both arrays may be multiplied by constant weighting factors. The floating point arrays may be real or complex real-plus-imaginary; the latter may be in packed or unpacked IDAMS complex format.

FPSUM begins by setting flags to indicate whether either array is to be multiplied by a weight other than unity, and whether either is to be taken with a negative sign. The label of the primary (tape) input is checked to see that it is the correct size, and establish whether one pair or two pairs of maximum and minimum values, corresponding to real and complex arrays, respectively, were required. Then a seek is issued for the first line of the secondary data set, located on disk, and the first data line is read from tape.

The summation is carried out one line at a time, as follows. The line of data is fetched from disk, a seek is issued to return the disk heads to the start of that line, the last read from tape is checked for completion, and a new read, into an alternate buffer, is issued. Then the line of tape data already in core is added to the line from disk, changing signs and multiplying by constants as specified by the flags, and each new value is tested as a possible minimum or maximum. The tape input buffer assignments are then interchanged and the completed line of data is written back onto disk. At the completion of this write, the disk head is in position to read the next line of input from the disk.

When all lines have been processed, the result array has replaced the secondary input on disk. The maximum and minimum values are then written into the cell following the last line of array data, and control is returned to the IDAMS monitor.

4.29.2 Parameters

FPSUM requires four parameters, as follows:

1. NX - Number of complex words per line or one-half number of real words per line
2. NY - Number of lines of data
3. X1 - Floating-point multiplier for all elements of primary input array (from tape)
4. X2 - Floating-point multiplier for secondary input (from disk)

4.29.3 Input

FPSUM requires two input data sets. The primary data set is a floating point array which has been previously copied to tape from disk by FPCON option 16. The values of SP, SL, NP, and NL on the task card are ignored. The values of NPI and NLI on the input tape label must equal 16 * NX and NY, respectively, for the summation to be carried out.

The secondary input resides on disk, beginning in cell 1, and is not referenced by the task card. It is the responsibility of the user to ensure that the values of NX and NY specified in the parameters are correct for the data set on disk.

4.29.4 Output

FPSUM produces a floating point array on disk, starting in cell 1, in the same mode as the two input arrays.

4.29.5 Example

The difference of a floating point array A206FT on tape and an array on disk is to be computed. The array contains 64 lines of 128 complex words each. Appropriate task and parameter cards are:

```
FPSUM, (A206FT, 49, 1),,, 1

128      64     1.0      -1,0
```

4.29.6 Messages

FPSUM generates the following fatal error message:

Message	Explanation
TAPE INPUT ARRAY NOT SPECIFIED SIZE	The size specified in the label (NP=16*NX, NL=NY) does not agree with values of NX and NY specified by parameters.

4.29.7 Flowchart

See Appendix C, Figure C-29.

4.30 FILTGN - DIGITAL FILTER GENERATING PROGRAM

4.30.1 Program Description

This task routine generates filters corresponding to a two-dimensional modulation transfer function (MTF). Output may be either a symmetric set of weights for convolution image-space filtering or an array of real values in corner-origin format for frequency-domain filtering by array multiplication. The input MTF may be specified as the product of two functions, each of which may be a linear, radially symmetric, or elliptically symmetric function, with principal axes oriented along the x or y axes. The actual MTF values along the principal axis may be entered as a complete point-by-point table, pairs of coordinates of the end points of segments for a piecewise linear function, or by specifying the 3 db point and ratio of high-to-low frequency point for a high or low emphasis filter. To avoid overenhancement of noise by a filter represented as the product of two MTF's, a maximum gain for the product filter may be specified. The logic can easily be extended to include additional options for generating MTF values internally.

FILTGN begins by calling the subprogram TABLEGN which accesses the parameters to determine the type of output filter requested, whether one or two input functions have been specified and the type for each, and the source of MTF values for each filter. If a complete table has been entered by the user, it is transferred directly to the appropriate specification, and the first gain value is copied into the table for all frequencies up to the first frequency value. Successive values are then generated by linear interpolation between adjacent pairs of coordinate values. For either of the preceding cases, the table is completed by copying the final gain value into any locations remaining.

If a high or low pass filter is specified, the parameter specifying the ratio of high-to-low frequency modulation is converted to a real number G_{hf} and tested to see whether high emphasis ($G_{hf} > 1$) or low emphasis ($G_{hf} < 1$) is required. For high emphasis, the MTF table is compiled by computing the gain

$$G(k) = 1 + (G_{hf} - 1)/(1 + k_o^2/k^2)$$

for integral values of k, where k_o is the user-specified 3 db frequency. For low emphasis, the formula used is

$$G(k) = G_{hf} + (1 - G_{hf})/(1 + k^2/k_o^2)$$

If an elliptically symmetric MTF is being used, it will be necessary to evaluate points off the coordinate axes by interpolation. If the minor axis of the ellipse is f times the major axis, then the effective distance of a point (k, l) from the origin is

$$\left(k^2 + \frac{l^2}{f^2}\right)^{\frac{1}{2}} = i + r$$

where i is the integral number of units from the origin, r is the remainder, and k is the distance along the major axis. In order to use efficiently the cubic interpolation formula for the Fourier amplitude a(k, l),

$$a(k, l) = G(i) + B(i)r + C(i)r^2 + D(i)r^3$$

TABLEGN computes tables of the coefficients using the formulae

$$B(i) = -1/3\ G(i-1) - 1/2\ G(i) + G(i+1) - 1/6\ G(i+2)$$

$$C(i) = 1/2\ G(i-1) - G(i) + 1/2\ G(i+1)$$

$$D(i) = -1/6\ G(i-1) + 1/2\ G(i) - 1/2\ G(i+1) + 1/6\ G(i+2)$$

These tables will be computed for

$$\left[\left(\frac{1}{2} N_{COL} + 1\right)^2 + \frac{1}{f^2}\left(\frac{1}{2} N_{ROW} + 1\right)^2\right]^{1/2}$$

values of i for an ellipse whose principal axes lie along the x-axis; for y-axis alignment, the number of rows N_{row} and columns N_{col} are interchanged in the above formula. However, due to core storage limitations, any required entries in the table beyond 600 will be supplied by duplicating the 600th value; this need could arise, for example, for $N_{col} = N_{row} = 572$ and $f < .47$.

When all required tables have been generated, FILTGN calls the subprogram MTFGN to generate the full two-dimensional MTF. When the final output is to be a set of convolution weights, a 32 by 32 MTF is generated, regardless of the specified size of the final filter (which may not exceed 33 by 33). MTFGN initializes test values for maximum and minimum and then generates one pair of symmetrically located lines at a time by computing the left-hand half of the line from the tables generated previously and then copying these values into the other, symmetrical, half of the line. When two functions have been entered, the values obtained from the first function are computed for the half-line, and then multiplied by the corresponding values for the second line. Each value is compared with the previous maximum and minimum values of the array and also checked to ensure it does not exceed the maximum gain specified by the user; if so, it is reduced to this maximum. The completed line is then written into two symmetrically located records on disk, and the computation continued until the MTF is complete. The final values of maximum and minimum are then copied onto the next disk cell following the last array line. If the final output is to be a frequency--space filter, FILTGN terminates processing and returns control to the monitor. To generate a set of convolution weights, FILTGN instead calls the subprogram INVERT. This program reads the top 17 rows of the 32 by 32 MTF back into core, inserts zero words after each MTF value to form a complex array, and calls PERGEN, TRIGGN, and FFTONE (which are described as General Purpose Subroutines) to carry out one-dimensional Fourier transforms on these 17 rows. The left-hand.17 columns are then interchanged with the first 17 complex words of each row, and the new (flipped) rows extended using the symmetry of the array. FFTONE is again used to Fourier transform the 17 flipped rows and the results are flipped back again to complete the two-dimensional Fourier transform.

The sum of the values which will be included in the final set of convolution weights is computed, and then the output array is generated in integer format, normalizing the values to produce a sum of 4096 and shifting the origin of coordinates from the corner to the center of the array. The filter array is then listed on the printer and copied onto the output tape for future reference.

4.30.2 Parameters

FILTGN requires the following parameters, all of which are in integer format except the identifiers X and Y:

1.	ITYPE	- Type of filter to be generated 1 - Frequency domain (MTF) 2 - Image domain (convolution weights)
2.	NCOL	- Number of columns in filter For frequency domain, must be power of 2:$2^1 \leq N_{col} \leq 2^9$ For image domain, must be odd and ≤ 33
3.	NROW	- Number of rows in filter. Same restrictions as for NCOL
4.	MAXGAIN	- Maximum gain in final MTF, as integer multiple of zero-frequency gain; this permits avoiding over-enhancement of high-frequency noise when filter is product of two high-emphasis filters Default or 0: No tests are made
5.	AXIS1	- X or Y (single letter) - axis along which first MTF component is aligned

6. ISYM1 - Symmetry of first MTF component:

1 - Linear - MTF value depends only on coordinate along AXIS1, and is constant in direction perpendicular to it

2 - 1000 - Elliptical (1000 = circular) with major axis along AXIS1 and minor axis = (ISYM1/1000) times major axis for each elliptical contour

7. INPUT1 - Mode of input for first set of MTF values

1 - Full table of MTF values

2 - Piecewise-linear representation by coordinate pairs

3 - High/Low emphasis filter

8. (Data) - Specification of MTF values along AXIS1. Format depends on mode of input, as follows:

INPUT1 = 1

N - Number of values in table

$K_1, K_2, \cdots, K_n$ - MTF values times 1000 (i.e., 333 represents .333) beginning with zero frequency

INPUT1 = 2

N - Number of pairs of coordinates

$M_1, K_1, M_2, K_2, \cdots, M_N, K_N$ - pairs of values, with M_i = integer frequency value followed by K_i = 1000 times MTF value. Values of M_i must be strictly increasing.

INPUT1 = 3

IHF - Ratio of high-frequency MTF to low frequency (D.C.) MTF, times 1000 (i.e., IHF = 200 means high frequency MTF is .200 times dc MTF).

I3DB - Frequency at which frequency-dependent part of filter is 50% of maximum (3 db point).

9.	AXIS2	-	X or Y - Alignment axis for second input MTF. If default or zero, only one MTF supplied
10.	ISYM2	-	Symmetry for second MTF, if any. Same codes as for ISYM1.
11.	INPUT2	-	Mode of input for second MTF, if any. Same codes as for INPUT2.
12.	(Data)	-	Specifications of MTF values along AXIS2, if any. Same formats as for first MTF.

4.30.3 Input

Input to FILTGN is entirely via parameters, as described above.

4.30.4 Output

FILTGN has two alternative types of output. If the user specifies a convolution (image-space) filter, the result is printed as a table of integer values normalized to a sum of 4096. Since the table is always symmetric, only the upper left quadrant, including the symmetry axes, is printed. The table is also stored, with standard IDAMS label, on the output tape, if any, specified by the user.

If the user specifies a real floating point array, it is output onto disk in standard IDAMS floating point format, beginning in cell 1. The symmetric array is in corner-origin format. Conversion to other formats or copying onto tape may be achieved using FPCON, specifying NX = NCOL/2, NY = NROW.

4.30.5 Example

It is desired to enhance the high-frequency detail of an image using a radially symmetric filter whose 3 db point is at 50% of maximum frequency and with a limiting gain of 3 times the dc gain. In addition, instrumental noise at approximately 25% of maximum frequency in the horizontal direction is to be suppressed. Output is to be a 21 by 21 convolution filter. Since a 32 by 32 MTF array is generated whenever a convolution filter is to be created, one-half the symmetric MTF will contain 16 points, so that 50% of maximum frequency is 8 and 25% is 4. Hence suitable IDAMS task and parameter specifications are:

```
FILTGN,,,,4

2,21,21,0

X,1000,3,3000,8,

Y,17,1000,1000,1000,0,1000,1000,1000,1000,1000,1000

1000,1000,1000,1000,1000 1000,
```

Notice that for the MTF table frequency 4 is the 5th MTF value given, since the table begins with the zero-frequency value; in this table, all frequencies are given unit modulation except the noise frequency, which is given modulation zero in order to suppress it completely. (The actual filter generated will involve approximations and roundings, so that the suppression will not be complete.) Since no output tape is specified, the convolution weights will be printed but not saved on tape.

4.30.6 Messages

FILTGN may generate the following fatal error messages:

Message	Explanation
INVALID OUTPUT TYPE	Parameter ITYPE not 1 or 2
INVALID AXIS SPECIFICATION	Alignment axis not specified as X or Y

Message	Explanation
ILLEGAL SYMMETRY CODE	Symmetry code ISYM not between 1 and 1000
ILLEGAL INPUT MODE	Input mode not 1, 2, or 3
ILLEGAL FREQUENCY VALUE	Frequency value not between 0 and 599 (piecewise-linear case) or not positive (high/low emphasis case)
FREQ VALUES NOT STRICTLY INCREASING	For piecewise-linear input mode, frequency coordinates not strictly increasing
SPECIFIED FILTER GAIN IS NEGATIVE	For high/low emphasis, high-frequency gain has illegal negative value
CONVOLUTION FILTER TOO LARGE	Dimensions specified for convolution filter exceed 33 by 33
CONVOLUTION FILTER DIMENSION NOT ODD	Dimensions specified are not both odd
SUM OF WEIGHTS NOT POSITIVE	Sum of unnormalized convolution weights is zero or negative; normalized filter cannot be generated

4.30.7 Flowchart

See Appendix C, Figure C-30.

4.31 RANDGRAY - RANDOM GRAY LEVEL GENERATION PROGRAM

4.31.1 Program Description

The Random Gray Level Generation Routine enables the user to either generate an image of random gray valued pixels with a given mean value and standard deviation, or to add random gray values with a mean of zero and a given standard deviation to each of the pixels of an input image. In each of the cases the distribution of the gray values is normal over the image.

Upon entry the program reads the parameters which initializes the random number generation routine. The IDAMS tape is then read (if one is given) and the random values are either added or stored into successive pixel positions in the lines of the image. The image is then written to an output tape.

4.31.2 Parameters

The random number generation routine requires the following parameters:

1. The standard deviation of the random numbers to be generated
2. The "seed" of the generator
3. The mean value of the random numbers if no input image is given

4.31.3 Input

RANDGRAY requires an input tape in standard IDAMS format (if applicable).

4.31.4 Output

RANDGRAY requires an output tape in standard IDAMS format.

4.31.5 Example

Assume that a random noise pattern is to be superimposed on an existing image and that the noise is to have a standard deviation of five gray levels. A "seed" value of 1875 is to be used. The following control cards would be required,

```
RANDGRAY, (TEST, 49, 1), (1, 1, 270, 340), (TEST2, 47, 1), 1
```

```
5, 1875
```

4.31.6 Messages

Illegal values of parameters are printed, if any occur.

4.31.7 Flowchart

See Appendix C, Figure C-31.

4.32 IMERGE - BULK ERTS TAPE MERGING PROGRAM

4.32.1 Program Description

This task routine unpacks the data from four 7-track ERTS computer compatible tapes (CCT's) representing the four strips of one spectral band of an RBV or MSS image and builds a single output image tape in standard IDAMS format. The annotation data from the CCTs are converted to CDC 3200 formats, stored in the annotation file on disk, and also printed out for reference.

IMERGE begins by determining whether data to be merged is RBV or MSS. If the data is MSS, control goes to a subroutine that builds a look-up table used for compressing the 8-bit characters from the Bulk CCT's to 6-bit characters for the IDAMS image tape. After the table is built, it is saved in core for use by the MSS merge program. Control then returns to the main driver program.

IMERGE sets up a message instructing the operator to dismount the SCOPE and IDAMS overlay tapes, and mount the four input tapes and the output tape on specified units; the program then pauses until the operator confirms that the mounting is complete. If RBV data are being processed, the spectral band numbers and associated output data set names are rearranged in reverse band sequence.

The header data on each CCT are then read. The date and time identifiers are compared to ensure that they are the same for all four tapes; if not, an error flag is set and execution aborts. The strip numbers are read, and a table created of the logical unit numbers for strips 1 through 4; if four different strips are not present, an error exit is taken. The data record length is read; for RBV it is checked to see whether it is less than 3456 bytes, and an error exit taken if not. For MSS, the adjusted line length and correction code are also decoded; if the correction code does not specify an acceptable data mode, an error exit is taken. The output label is then written, specifying 4608 pixels per line for RBV or the adjusted line length for MSS, and 4125 or 2340 lines for RBV or MSS, respectively (780 lines for MSS Band 5).

The annotation record from strip 1 is read next, while the corresponding records on the other three tapes are spaced over. The annotation data are decoded, converted to floating point format, and stored on the annotation file on disk. Cell 25 is filled with dummy maximum and minimum values of 1.0 to facilitate the use of FPCON for saving the disk tables on tape. The annotation data are also written out on the printer.

For RBV, the input tapes are advanced to the first data records for spectral band 2 or 1 if band 3 is not to be processed.

Then image data are read in one record from each tape at a time, and an appropriate entry point to the COMPASS subroutine REPACK called to generate full output lines one at a time. For RBV data, for which one input record contains three consecutive line segments, three output lines will be generated before the next input read. For ERTS-B MSS data, every fourth record, containing band 5 data, will be skipped, except when band 5 is being output, in which case only every fourth input record will be read.

When all lines have been processed, the input parameters will be checked to see whether any further bands are to be processed. If so, the MSS tapes will be rewound and RBV tapes will be advanced to the appropriate band.

The header and annotation records are skipped, and instructions issued to the operator to mount a new output tape. When the operator returns control, the new output label is written, and the image data processed in the same manner as for the first tape. When all output tapes have been generated, or a fatal error has been encountered, the operator is instructed to remount the SCOPE and IDAMS overlay tapes. When the operator returns control, IMERGE exits, and returns to the IDAMS monitor for error processing or initiation of the next task.

4.32.2 Parameters

IMERGE requires the following parameters:

1. ITYPE - Image Type: 1 = RBV
   2 = MSS Type IIa (4 bands)
   3 = MSS Type IIb (5 bands)

2. NBAND	-	Integer number of spectral bands to be processed
3. IBAND1	-	Integer band number for first spectral band
4. OUTNAME2	-	Five to eight alphanumeric characters specifying name to be given to second output tape (if any)
5. IBAND2	-	Integer band number for second band
6. OUTNAME3	-	Name and band number for additional band, if requested
7. IBAND3	-	

•

•

•

NOTE: The name specified for the output is applied only to the first spectral band requested. Additional spectral bands are named in the parameters. The input size field on the task card is ignored; the output image always represents the entire input image.

4.32.3 Input

Four input computer-compatible tapes are required, representing the four strips of the same ERTS frame.

NOTE: Do not specify any input on the task card, since the CCTs do not have IDAMS-compatible labels.

4.32.4 Output

IMERGE generates one or more image tapes in standard IDAMS format; each tape represents one spectral band for a single ERTS frame.

In addition to the standard data in words 1 to 6, the output label contains a summary of annotation and ID data from the computer compatible (input) tapes, as follows:

Word	Contents
7-9	Scene/Frame ID, BCD, in Form EDDDHHMMMSBNC where E = ERTS mission code DDDHHMMS = day and time of exposure B = Spectral band identifier N = Sequential subframe identifier (if needed) C = MSS data mode/correction code: 1s bit = 1 for decompression 2s bit = 1 for Hi gain on band 2 3s bit = 1 for Hi gain on band 3
10	Format center latitude, degrees with 12-bit fraction (..., to nearest 1/4096 degree)
11	Format center longitudes, degrees with 12-bit fraction
12	Nadir latitude, degrees with 12-bit fraction
13	Nadir longitude, degrees with 12-bit fraction
14	Sun elevation, degrees (no fraction)
15	Sun azimuth, degrees
16	Orbital heading (from North), degrees

A table of annotation data is written onto disk in cells 1 to 4, and also listed on the printer. The header record consists of ten words, as follows:

Word	Symbol	Meaning
1-2	OUTNAME1	Name of first IDAMS output data set generated for this ERTS frame

Word	Symbol	Meaning
3-4	OUTNAME2	Names of additional output tapes, if any; unused locations are filled with zeros
5-6	OUTNAME3	
7-8	OUTNAME4	
9-10	OUTNAME5	

The second record contains ten words of data. The first nine word are identical to words 7 through 15 of the output label. Word 10 contains the number of pixels per line, NPO.

The third record contains 12 words representing six floating-point entries for attitude and orbital data at mid-scan time, as follows:

Word	
1-2	Yaw, in radians
3-4	Roll, in radians
5-6	Pitch, in radians
7-8	Altitude, in meters
9-10	Velocity along track, meters/second
11-12	Orbital heading from north, degrees.

4.32.5 Example

Merge the four CCT tapes for an RBV image to create output tapes named TETONRED and TETONI-R from spectral bands 2 and 3, respectively. Suitable IDAMS task and parameter cards are:

```
IMERGE,,, (TETONRED, 47, 1), 1
```

```
1, 2, 2, TETONI-R, 3
```

4.32.6 Messages

The ERTS bulk image conversion routine prints out a table of annotation data associated with the image being converted to IDAMS format. The following operator messages are also typed out:

Message	Explanation
DEMOUNT OVERLAY AND SYSTEM TAPES MOUNT OUTPUT ON LUN nn AND SET AT 800 BPI MOUNT INPUT TAPES ON REMAINING DRIVES (IN ANY ORDER) WHEN READY, PRESS CLEAR	All tape drives are required to handle four input tapes and one output tape.
MOUNT OUTPUT TAPE ON LUN nn WHEN READY, PRESS CLEAR	A new output tape is needed for each spectral band.
REMOUNT RTS AND OVERLAY TAPES WHEN READY, PRESS CLEAR	System and program tapes must be restored when processing is complete.

In addition, one of these fatal errors may be produced:

Message	Explanation
STRIP NUMBER NOT BETWEEN 1 AND 4	Invalid strip number, tape cannot be identified.
INPUTS NOT ALL SAME ERTS FRAME	Frame ID (Date and Time) not same for all input tapes.
INPUTS NOT 4 DIFFERENT STRIPS	One or more strips has been duplicated on input.
ILLEGAL SPECTRAL BAND	Specified spectral band not allowed for specified sensor type.
WRONG LINE LENGTH FOR GIVEN SENSOR	Line length specified by annotation record not appropriate to sensor type.

4.32.7 Flowchart

See Appendix C, Figure C-32.

4.33 PMERGE - PRECISION ERTS TAPE MERGING PROGRAM

4.33.1 Program Description

This task routine unpacks the data from precision 7-track ERTS computer compatible tapes (CCTs) and builds a single output image tape in standard IDAMS format. The annotation data from the CCTs are converted to CDC 3200 formats, stored in the annotation file on disk, and also printed out for reference.

PMERGE begins by determining the type of precision tape being input. If the precision tapes are sixteenth frame tapes (two input tapes) or quarter frame tapes (four input tapes) the program PARMER is called to process the tapes. When the tapes are full frame tapes (four input tapes) the program FULMER is called to process the tapes.

PARMER and FULMER both read the header data on each CCT. The annotation records are read, the data decoded, converted to floating point format, and stored on the annotation file on disk.

PARMER determines if the input tapes are quarter frame or sixteenth frame. If sixteenth frame tapes the output label is written, specifying 1024 pixels per line and 1024 lines. In the case of quarter frame tapes, the specifications are 2048 pixels per line and 2048 lines. Image data are read in one record at a time from each tape at a time and the COMPASS subroutine REPACK called to generate 4096 6-bit characters. A record contains a portion of eight consecutive lines. Each line portion from one tape's record is merged with the corresponding line portion of the other input tape(s). The eight completed output lines are generated before the next set of input records are read.

FULMER's logic becomes more complicated because of the larger amount of input image data. FULMER begins by writing the output label, specifying 4096 pixels per line and 4096 lines. The image data is arranged on the four input tapes in eight strips in which the 1st and 5th strips reside on tape 1, the 2nd and 6th strips reside on tape 2, etc. Each strip contains 512 records and each record contains 8 pixel values for 512 consecutive lines. The scan line begins with the right-most pixel of the first line in a strip and scans down 512 lines before preceding to the next pixel value. Consequently, in the case of the first strip, the image data from the first record would be ordered as follows: pixel 4096 from line 1, pixel 4096 from line 2, ... pixel 4096 from line 512, pixel 4095 from line 1, pixel 4095 from line 2, ... pixel 4095 from line 512, ... pixel 4088 from line 1, pixel 4088 from line 2, ... pixel 4088 from line 512.

Due to the nature of the input data, it is not possible to output even the first line of imagery in IDAMS format until 512 records, which constitutes a strip, have been read. Because the core restrictions prevent 512 complete lines from being stored in core, PMERGE is forced to store portions of the 512 lines onto disk storage. The program reads in four consecutive lines, sorts and orders the pixels such that 8 words of pixel values from each line are linked together. These blocks of eight words are then stored on disk before the next set of four records is read in. This process continues until 512 records have been processed and stored onto disk. At this point, the program fetches the data back from disk until it can output onto tape four complete consecutive lines. The data is fetched from disk until 512 lines have been output. PMERGE then proceeds to carry out the same series of operations on the remaining strips of data until a full image has been processed.

When all lines have been processed, both PARMER and FULMER check the input parameters to see whether any further bands are to be processed. If so, instructions are issued to the operator to mount a new output tape, and the image data is processed in the same manner as for the first tape. When all output tapes have been generated, or a fatal error has been encountered, the operator is instructed to remount the SCOPE and IDAMS overlay tapes. When the operator returns control, PMERGE exits, and returns to the IDAMS monitor for error processing or initiation of the next task.

4.33.2 Parameters

PMERGE requires the following parameters:

1.	ITYPE	-	Input type: 1 = full frame tapes 2 = 16th frame tapes 3 = quarter frame tapes
2.	NBAND	-	Integer number of spectral bands to be processed
3.	IBAND1	-	Integer band number for first spectral band
4.	OUTNAME2	-	Five to eight alphanumeric characters specifying name to be given to second output tape (if any)
	IBAND2	-	Integer band number for second band
	OUTNAME3	-	Name and band number for additional band, if requested.
	IBAND3	-	
	•		
	•		
	•		

NOTE: The name specified for the output is applied only to the first spectral band requested. Additional spectral bands are named in the parameters. The input size field on the task card is ignored; the output image always represents the entire input image.

4.33.3 Input

Four (two) input computer compatible tapes are required, representing the four strips of the same ERTS frame.

NOTE: Do not specify any input on the task card, since the CCTs do not have IDAMS - compatible labels.

4.33.4 Output

IMERGE generates one or more image tapes in standard IDAMS format; each tape represents one spectral band for a single ERTS frame.

A table of annotation data is written onto disk in cells 1 to 4, and also listed on the printer. The header record consists of five double words, as follows:

Word	Symbol	Meaning
1-2	OUTNAME1	Name of first IDAMS output data set generated for this ERTS frame.
3-4	OUTNAME2	Names of additional output tapes, if any; unused locations are filled with zeros.
5-6	OUTNAME3	
7-8	OUTNAME4	
9-10	OUTNAME5	

The first data record contains seven floating point words, as follows:

Floating Point Word	Contents
1	Nadir latitude, degrees
2	Nadir longitude, degrees
3	Spacecraft heading at format center, radians
4	Spacecraft altitude at format center, meters
5	Yaw at format center, radians
6	Roll at format center, radians
7	Pitch at format center, radians

For MSS images, an additional record containing 38 floating-point words is also generated, as follows:

Floating Point Word(s)	Contents
1	Spacecraft velocity (component tangential to each) at frame center, meter/second
2	Average rate of change of tangential component of velocity during scanning of frame, meter/(second)^2
3-11	Spacecraft altitude at times -13.8, -10.3, -6.9, -3.4, 0.0, +3.4, +6.9, +10.3, and +13.8 seconds from format center time

Floating Point Word(s)	Contents
12-20	Yaw angles at corresponding times
21-29	Roll angles at corresponding times
30-38	Pitch angles at corresponding times

4.33.5 Example

Merge four CCT full frame tapes to create output tapes named PREC1, PREC2, and PREC3 from spectral bands 1, 2, and 3, respectively. Suitable IDAMS task and parameter cards are:

```
PMERGE,,, (PREC1, 47, 1), 1
```

```
1, 3, 1, PREC2, 2, PREC3, 3
```

4.33.6 Messages

PMERGE generates no messages.

4.33.7 Flowchart

See Appendix C, Figure C-33.

4.34 PPUPDATE - PRECISION PROCESSING DISK FILE UPDATE PROGRAM

4.34.1 Program Description

This task routine provides a capability for adding, deleting, changing, or listing records in the data files on disk containing ephemeris, reseau location, control point location, and geometric transformations information for use by the routines in the precision processing and geometric manipulation packages. It provides a means for entering test data and editing data stored on disk by IMERGE, RZOMAP, CORREL, and RESECT.

PPUPDATE is designed to carry out a sequence of updating and listing steps on a single loading of the overlay. It begins processing by reading disk cells 1 through 24 into core. The keywords designating successive task steps are then processed one at a time.

The first keyword for each step is scanned to determine whether ephemeris, reseau, control point, geometric grid, or geometric linkage data are to be processed. The second key word is then accessed to determine whether the required function is adding, deleting, or changing a data record or also listing the entire table. Control is then passed to the required module.

Each module begins by accessing the remaining parameters for that step, except that no additional parameters are needed for a listing step. For delete and change, the next parameter specifies the number of the data record to be deleted or changed; record zero is interpreted as the header record, which may be changed but not deleted. When a deletion is made, the record count in the header record is reduced by one, and the remaining data records are moved up to fill the gap. When a record is to be added, the number of the next available record is obtained from the count in the header record, which is incremented after first checking that the file is not already full.

The data for the new or changed record are then interpreted and converted to the required internal formats. The user is required to supply all entries in the record, and not merely those to be changed, with the exception that for control point records either latitude-longitude or UTM easting and northing may be entered.

When each update or listing step is completed, control is returned to the keyword-scanning module. When the keyword DONE is encountered, the updated files are written back onto disk, and control is returned to the IDAMS monitor.

4.34.2 Parameters

For each step to be executed, the following parameters are required:

1. File ID - keyword with following meanings:

EPHM	- Ephemeris and annotation file
RZO	- Reseau location file
CP	- Control-point location file
GRID	- Grid-point location file
LINK	- Linkage file for grid points
DONE	- Last step finished; always required as last parameter

2. Mode - keyword with following meanings:

INIT	- Initialize the designated file
LIST	- List designated file, including header record (except for LINK, whose header is printed with GRID)
ADD	- Add a data record, and update record count in header
DEL	- Delete a data record, close up gap, and update count in header
CHNG	- Change a record

3. Record Number (DEL or CHNG)

   0 - change header record; with DEL, 0 causes an error

   >0 - change or delete specified data record

   New Data Entry (ADD)

4. New Data Entry (CHNG only)

For EPHM, RZO, GRID, and LINK, the entire new record must be entered in the same format as specified for that table in the paragraphs under Input. A reference to LINK record 0 will be interpreted as the header file for GRID.

For CP, the header file has only three of eight words currently in use; only these three words should be entered.

For CP data files, the parameters should be entered in the same manner as for CHIPGN. This requires that either UTM or Lat/Long values be entered, that every parameter be preceded by a keyword, and that successive parameters be separated by commas. The parameters are:

1.	CPP	= nnnn	- Image pixel position of control point
2.	CPL	= nnnn	- Image line position of control point
3.	LAT	= dd, mm, ss. s_{S}^{N}	- Latitude of control point. If N/S designation is omitted, north latitude is assumed unless latitude is preceded by a minus sign, in which case south latitude is assumed.
4.	LONG	= ddd, mm, ss. s_{W}^{E}	- Longitude of control point. If E/W designator is omitted, north is assumed unless value is negative, in which case west longitude is assumed.

5.	ZONE = nn^{N}_{S}	-	UTM Zone. If N/S designator is omitted, north is assumed unless number is preceded by a minus sign
6.	UTME = nnnnnn	-	UTM easting in meters (=500,000 at zone central meridian)
7.	UTMN = nnnnnnn	-	UTM northing in meters
8.	ELEV = nnnn	-	Elevation above sea level in meters

4.34.3 Input

No input image is used. In addition to the parameters, input data for PPUPDATE are taken from the precision processing files contained in cells 1 to 24 on the disk. (Note that if these files have been copied temporarily onto tape, they may be reloaded onto disk using FPCON option 6 with NX = 16, NY = 24.)

The formats of these files are as follows:

1. Ephemeris and Annotation Data (Disk Cells 1 to 4)

Header Record (10 words)

Word	Contents
1-2	Name (8 BCD characters maximum) of first IDAMS output data set generated for this ERTS frame
3-4	Name of second data set, representing a different spectral band, for this ERTS frame
5-6	Name of third data set
7-8	Name of fourth data set (MSS only)
9-10	Name of fifth data set (ERTS-B MSS only)

Unused locations are filled with zeros.

Identification Record (10 words)

Word	Contents
1-3	Scene/Frame ID, BCD, in form:

EDDDHHMMMSBNC, where

E = ERTS mission code

DDDHHMMS = day and time of exposure

B = Spectral band identifier

N = Sequential subframe identifier (if needed)

C - MSS data mode/correction code

1's bit = 1 for decompression

2's bit = 1 for Hi gain on band 2

4's bit = 1 for Hi gain on band 3

4	Format center latitude, degrees with 12-bit fraction (i.e., to nearest 1/4096 degree)
5	Format center longitude, degrees with 12-bit fraction
6	Nadir latitude, degrees with 12-bit fraction
7	Nadir longitude, degrees with 12-bit fraction
8	Sun elevation, degrees (no fraction)
9	Sun azimuth, degrees
10	Length of scan line, pixels

Mid-Scan attitude and orbital data record (12 words)

Word	Contents
1-2	Yaw, in radians, floating point
3-4	Roll, in radians, floating point
5-6	Pitch, in radians, floating point
7-8	Altitude, in meters, floating point
9-10	Velocity along track, meters/second, floating point
11-12	Orbital heading from north, degrees, floating point

MSS attitude/altitude record (72 words)

Word	Contents
1-18	9 values of Yaw (radians, floating point) at -13.8, -10.35, -6.9, -3.45, 0.0, 3.45, 6.9, 10.35, and 13.8 seconds from mid-scan time

19-36	9 values of Roll, at same times
37-54	9 values of Pitch, at same times
55-72	9 values of altitude (meters, floating point), at same times

2. Reseau Location Table (Disk cells 5 to 10)

One header record and 81 data records of four integer words each. The header record format is:

Word	Contents
1-2	Image Name
3	Number of reseaus from GETRZO
4	Number of reseaus from NTRP2

The data records correspond to reseau marks starting with the top line and going from left to right within each line of reseaus. The record format is:

Word	Symbol	Contents
1	NOMP	Nominal pixel position
2	NOML	Nominal line position
3	IMP	Pixel position in image
4	IML	Line position in image

3. Control Point Location (Disk cells 11 and 12)

One 8-word header record and up to 10-12 word data records.

The header record format is:

Word	Contents
1-2	Image name
3	Number of data records
4-8	Unused

The data record format is:

Word	Symbol	Contents
1	ICPP	Image pixel position of control point
2	ICPL	Image line position of control point
3-4	XLAT	Latitude, degrees and decimal fraction
5-6	XLONG	Longitude, degrees and decimal fraction
7-8	UTME	UTM easting, proceeded by 10^6 times zone number
9-10	UTMN	UTM northing, negative for southern hemisphere
11-12	ELEV	Elevation above sea-level, in meters

4. Geometric Transformation Coordinates (Disk cells 13 to 24)

One four-word header record, 100 four-word location data records, and 180 two-word linkage data records; unused data records are ignored. The header record format is:

Word	Contents
1-2	Image name
3	Number of grid points
4	Number of linkages

The grid point location record format is:

Word	Contents
1	Grid pixel position
2	Grid line position
3	Image pixel position
4	Image line position

The linkage record format is:

Word	Contents
1	Grid point (numbered in order of appearance in location table) at start of linkage line
2	Grid point at end of linkage

4.34.4 Output

Output from PPUPDATE is an updated set of precision processing data files on disk and printer listings of those files which the user specifies.

4.34.5 Example

After examining the results of processing performed by RZOMAP and CORREL, the user has decided to modify the position data for the 19th reseau, delete the correlation results for the third and seventh control points, and add one new control point value. Suitable IDAMS control statements are:

```
PPUPDATE,,,,7
RZO, CHNG, 19, 3148, 582, 3117, 585,
RZO, LIST,
CP, CHNG, 3, CPP=2402, CPL=1681,
ZONE = 33, UTME=604280, UTMN=26400, ELEV=1180,
CP, DEL, 7,
CP, LIST,
DONE
```

Note that the user has obtained a listing of the updated reseau and control point location files. Instead of deleting CP record 3 and subsequently adding the new control point, he has saved one step by using CHNG to replace record 3 by the new data.

4.34.6 Messages

PPUPDATE may print one or more of the following warning messages:

Message	Explanation
aaaa MODE IS AN ILLEGAL OPERATION ON FILE aaaa	An illegal attempt was made to alter a portion of one of the files, probably a header record. Data is ignored and processing continues.
nnnn IS AN ILLEGAL RECORD NUMBER	The supplied record number is inconsistent with currently defined file limits. Data is ignored and processing continues.
aaaa FILE FULL, NEW RECORD CANNOT BE ADDED	No more available space exists in file aaaa, DEL or INIT must be performed before another record can be added. Data is ignored and processing continues.

4.34.7 Flowchart

See Appendix C, Figure C-34.

4.35 VPICIN - VICAR IMAGE REFORMATTING PROGRAM

4.35.1 Program Description

This task routine converts an image which has been generated in standard VICAR format on a 7-track tape using the IBM-360 tape conversion mode into an image in standard IDAMS format.

VPICIN begins by generating a table for converting the 8-bit grey-level values to 6-bit values by truncating the two low-order bits. The VICAR label records are then read in, and the EBCDIC data translated to BCD using the general-purpose subroutine CODE8TO6 and an internally stored conversion table. The size of the image is obtained from the labels, and the labels are written on the printer.

After the last label record has been read and identified as such, VPICIN writes the IDAMS label onto the output tape. The image data records are then read, using double buffering, and translated to 6-bit values one line at a time, again using CODE8TO6, with the translation table generated at the start of the routine. When all lines have been processed, control is returned to the IDAMS monitor.

4.35.2 Parameter

VPICIN requires a single parameter in integer format:

1. LUNV - logical unit number of tape drive on which VICAR tape is mounted

4.35.3 Input

Input to VPICIN is a single 7-track tape containing a VICAR image with standard VICAR labels generated using the IBM 360 conversion mode for tape output.

Since this image does not have an IDAMS label, it cannot be specified by the standard input field of the IDAMS task card. Instead, this field must be defaulted, and the logical unit number specified by the parameter LUNV, described above.

4.35.4 Output

VPICIN creates one output image in standard IDAMS format and a printer listing of the VICAR labels on the input image.

4.35.5 Example

It is desired to convert a VICAR image mounted on LUN 49 into an IDAMS output tape on LUN 47; the output image is to be name VICAR037. Appropriate IDAMS task and parameter cards are:

```
VPICIN,,, (VICAR037, 47, 1), 1
```

```
49
```

4.35.6 Messages

VPICIN generates one fatal error message, as follows:

Message	Explanation
END OF VICAR LABEL NOT FOUND	VICAR tape apparently not in standard format; start of image data could not be identified.

Warning messages are also written to inform the user when parity errors have been detected, these messages are:

PARITY ERROR ON VICAR LABEL

nnnn PARITY ERROR(S)IN DATA

No other action is taken by the program for parity errors.

4.35.7 Flowchart

See Appendix C, Figure C-35.

4.36 INCREASE - IMAGE ENLARGING PROGRAM

4.36.1 Program Description

The task program, INCREASE, will enlarge a standard IDAMS image by an integral factor computed from input parameters. Blank fill characters are provided on both sides and/or top and bottom, if needed to complete the requested output image. The results are then written on the specified output tape.

The program computes the largest integral multiplication factor which will just permit the input to fit within a specified output image. An output line buffer is set up with edge fill characters, if any, and the output label record is written. If any fill lines are required at the top of the output picture, they are written out at this time. The program then enters a main processing loop to read an input line, enlarge its length by the multiplication factor and output the enlarged line the requested number of times. This process continues until all of the input lines have been read and processed. Any remaining lines of bottom fill are written out, if necessary, and the program returns to the monitor.

4.36.2 Parameters

1. NPOUT = number of pixels to be output
2. NLOUT = number of lines to be output
3. IFILL = gray level for edge fill. default = 0

4.36.3 Input

INCREASE requires a single input image tape in standard IDAMS format.

4.36.4 Output

INCREASE generates a single output image tape in standard IDAMS format.

4.36.5 Example

A 200 line by 500 pixel sample portion of input image LITPIC is to be reduced to fit a 500 by 1000 output requirement. This implies a multiplication factor of 2 in both the number of data lines and pixels. Therefore, the resultant output image will contain a 400 line by 1000 pixel image data area preceded and followed by 50 lines of fill characters. No fill is necessary along the left and right edges.

The following IDAMS source statements would be appropriate:

```
INCREASE, (LITPIC, 49, 1), (1, 1, 500, 200), (OUTPIC, 47, 1), 1
```

```
1000      4000
```

4.36.6 Messages

None.

4.36.7 Flowcharts

See Appendix C, Figure C-36.

4.37 COLOR - FALSE COLOR CODING PROGRAM

4.37.1 Program Description

The task program COLOR provides for a "false-color" presentation of an image using table lookup. The program has the capability of generating several IDAMS - format image files from a single input image, using a different conversion code for each output. The translation tables can be obtained from any of three sources: several standard tables of color corrections stored in the program, a complete conversion table supplied by the user, or a table generated by linear interpolation between user-supplied pairs of old and new gray-level values, which define a piecewise linear relation between old and new values.

After interpreting the first keyword, which designates which spectral band is being represented, the required table is retrieved or generated and stored into the program area. COLOR reads in one line of input data at a time and stores the input line in a disk file for future reference. The COMPASS subroutine CODE translates one character at a time to the new gray level values specified by the table. The finished line is written on output, and successive lines are produced in the same manner until the specified region of the image has been processed. Then the next keyword is interpreted, the required table set up and the next output file, representing a different spectral band for a "false-color" presentation of the input image, is produced. Successive color keywords are processed until all the parameters have been digested and the output tape has been completed.

4.37.2 Parameters

Three color translation tables are required, one for each spectral band. Each table must be preceded by a keyword, which designates the color spectral band.

1. Keyword (RED, BLUE, GREEN)

The color translation tables can be specified in one of three ways.

If the translation is to follow a non-linear relation, then the remaining parameters are

2.	N = 2	Use table of new tables entered as parameters 3 - 66 for old values 0 to 63, respectively.
3.-66.		New values to which the old values 0 to 63, in that order, are to be converted; 64 values must be supplied

OR

2.	N = 1	Use standard table, stored internally
3.	M = 1	Use primary table for specified color
	= 2	Use secondary table for specified color

If the translation is to follow a linear or piecewise linear relation, the remaining parameters are:

2.	N = 3	Use pairs of coordinate points
3.	M	Number of pairs of coordinate points that follow $(2 \leq M \leq 11)$
4.-5.		Old and new values, respectively, for point at left-hand end of leftmost line segment
6.-7.		Old and new values, respectively, for point at left-hand end of next line segment
2N., 2N + 1.		Old and new values, respectively, for point at right-hand end of last (rightmost) line segment

The first pair of values should include at least one zero; the last pair at least one 63. If the first old value is nonzero, all values less than it will be assigned a new value of zero. If the last old value is not 63, all values greater than it will be assigned the last new value. At most, 11 pairs of coordinates can be specified, corresponding to ten contiguous line segments. In addition, the old values must be strictly increasing.

4.37.3 Input

COLOR requires a single input image in standard IDAMS format. The size of the image to be processed [N = ((NPI - 1)/4 + 2)/65 + 1 * NLI] must be less than 32300, so that the input image can be stored onto disk.

4.37.4 Output

COLOR generates a single output tape with multiple files in standard IDAMS format. For each spectral band requested, a file of a false color image is generated.

4.37.5 Examples

It is desired to generate three bands of a false-color image, (RED, GREEN, and BLUE). The input image has previously been reduced to TV size and it is desired to use the standard tables for each band. Appropriate IDAMS source statements are:

```
COLOR, (TVJAMES, 49, 1), (1, 1, 700, 512), (COLOR, 48, 1), 1

RED, 1, 1, GREEN, 1, 1, BLUE, 1, 1
```

The output tape will consist of three files, whose names are COLORR, which contains a "red" image, COLORG, which contains a "green" image, and COLORB, which contains a "blue" image.

4.37.6 Messages

COLOR generates the following diagnostic messages:

Message	Explanation
N NOT LE 11 AND GE 0	Either N was found to be negative or more than 11 pairs of coordinates were specified; execution terminates.

Message	Explanation
COORDINATE VALUE GT 63 OR LT 0	A parameter greater than 63 or less than 0 was specified; execution terminates.
OLD COORD NOT STRICTLY INCREASING	A specified value of the old intensity was less than or equal to the preceding value; execution terminates.
IMAGE SIZE TOO LARGE FOR DISK	The input image specified is too large for COLOR to process since the image will not fit on disk.

4.37.7 Flowchart

See Appendix C. Figure C-37.

4.38 FPLIST - FLOATING-POINT LISTING UTILITY PROGRAM

4.38.1 Program Description

The task program FPLIST provides a floating point formatted print for a user-supplied number of pixel values and number of lines.

FPLIST begins by determining if the input image resides on magnetic tape or on disk. Each line segment is fetched and the specified number of pixel values is output onto the printer. This process continues until all the requested lines have been printed.

4.38.2 Parameters

FPLIST requires the following two parameters:

1. NX - number of floating point values to be printed per line
2. NY - number of lines to be printed

4.38.3 Input

FPLIST requires one of the two following inputs:

1. an image tape in standard IDAMS format
2. a disk file containing image data

4.38.4 Output

FPLIST provides a formatted floating point print of the user-supplied input area.

4.38.5 Examples

An array of 32 x 16 floating point words has been stored on disk. FPLIST is to print out the array. The appropriate task and parameter cards are:

```
FPLIST,,,,1
```

```
32,16
```

4.38.6 Messages

None.

4.38.7 Flowchart

See Appendix C, Figure C-38.

4.39 DMDOUT - DIGITAL MUIRHEAD REFORMATTING PROGRAM

4.39.1 Program Description

This task program reformats IDAMS images into the format required by the Digital Muirhead Display (DMD) at the National Oceanic and Atmospheric Agency.

DMDOUT begins by determining whether the output image can fit within the 2500 by 2500 raster size or requires the 5000 by 5000 raster, and sets the mode in the header record accordingly. The operator is also instructed to mark the tape with the appropriate mode. If the image has more than 4860 lines, a bit is also set in the header record to suppress the gray-step wedge, which is otherwise generated automatically by the DMD system. The header record is then written onto the output tape.

For the 5000 by 5000 mode, the length of the output tape required is also estimated; if only one reel is required, the operator is instructed to mark it accordingly.

The requested segment of the image is then read in one line at a time, aligned for output, and written onto the output tape. On completion, the operator is reminded to mark both reels of a double-reel image. Control then returns to the IDAMS driver.

4.39.2 Parameters

DMDOUT requires no parameters.

4.39.3 Input

DMDOUT requires a single input image in standard IDAMS format.

4.39.4 Output

DMDOUT generates an output image in the format required by the Digital Muirhead Display.

4.39.5 Example

A 3000 by 3000 segment of an image called IMAGE6, starting at pixel 501 of line 101, is to be converted for DMD recording. The required IDAMS control statement is

```
DMDOUT,(IMAGE6,49,1),(501,101,3000,3000),(,47,1)
```

4.39.6 Messages

DMDOUT generates one fatal error message:

Message	Explanation
IMAGE TOO LARGE	The specified output image dimensions exceeded 5000 pixels by 4980 lines

A number of advisory messages are also generated to instruct the operator how to mark the output tape, as follows:

USE 556 BPI, MARK TAPE 'xxxx - BYTE MODE'

... AND 'REEL 1 OF 1'

IF SECOND OUTPUT REEL WAS REQUIRED, MARK TAPES '1 OF 2' AND '2 OF 2' - IF ONLY ONE REEL, MARK '1 OF 1'

4.39.7 Flowchart

See Appendix C, Figure C-39.

4.40 ADDPIX - PICTURE ADDITION PROGRAM

4.40.1 Program Description

This task routine merges two images in IDAMS format. This is achieved by adding the values of corresponding pixels and storing the sum into the corresponding pixel location of the output image.

Upon entry, the routine reads the label from the secondary input tape and verifies that the image size specified for the primary input image is the same or less than the size of the secondary tape. If the image sizes are compatible, then the program generates and writes the output label.

The two input tapes are positioned to their respective starting lines and the program begins the merging process. The routine reads in one record from each input tape at a time, computes the sum of each corresponding pixel value and outputs the merged line onto the output tape. This procedure continues until the specified number of records have been processed.

ADDPIX then returns control to DRIVER.

4.40.2 Parameters

ADDPIX requires the following parameters:

1. ISSP - secondary input tape start pixel
2. ISSL - secondary input tape start line

4.40.3 Input

ADDPIX requires two input tapes in standard IDAMS format.

4.40.4 Output

ADDPIX generates a single output tape in standard IDAMS format which contains the averaged sum of the two input tapes.

4.40.5 Example

It is desired to merge a portion of an image from two different input sources. The size of the portion is 64 pixels by 100 lines and, on the primary input tape, the area to be merged is located in the upper left-hand corner of the entire image, while on the secondary input tape, the area starts after line 100. Appropriate IDAMS source statements are:

```
ADDPIX, (IN1, 49, 1, IN2, 48, 1), (1, 1, 64, 100), (OUT, 47, 1), 1
1, 100
```

4.40.6 Messages

ADDPIX may generate the following message:

Message	Explanation
IMAGE SIZE EXCEEDS SECONDARY INPUT SIZE	The user specified an image size which is larger than the secondary input tape's image size.

4.40.7 Flowchart

See Appendix C, Figure C-40.

4.41 FORMAT - IDAMS FORMAT CONVERSION PROGRAM

4.41.1 Program Description

The task program, FORMAT, converts a rectangular image to an IDAMS-formatted tape and vice-versa. The record length and number of records on a rectangular image are unknown, while this information is supplied in the label record of an IDAMS image.

Format begins execution by determining the direction of the conversion. In the case where an IDAMS to rectangular conversion is requested, the program ignores the IDAMS label record and after deleting the line number from an IDAMS record writes the image line onto the output tape. This process continues until all of the requested lines have been output.

When a rectangular image is to be converted to IDAMS format, it becomes necessary to determine the record length. The program reads the first record from the rectangular image, determines the length of the record and sets the necessary variables. A dummy IDAMS label is written and as each record is read from the rectangular image it is output onto the IDAMS - formatted tape, after the line number has been placed into the first word of each record. This process continues until an end-of-file mark is sensed on the input tape. After writing an end-of-file mark on the output tape, the IDAMS tape is then rewound and the completed IDAMS tape label is written onto the tape. FORMAT then returns control to the system.

4.41.2 Parameters

FORMAT requires the following special parameter:

LUN = unit number of rectangular image tape

4.41.3 Input

FORMAT requires an input tape in either IDAMS format or in rectangular image format.

4.41.4 Output

FORMAT generates an output tape in either a rectangular image format or in IDAMS format.

4.41.5 Example

A rectangular image is to be converted to an IDAMS formated image. Appropriate IDAMS batch processor task and parameter cards are:

```
FORMAT,,, (IDAMSTAP, 47, 1), 1
48
```

4.41.6 Messages

FORMAT does not generate any messages.

4.41.7 Flowchart

See Appendix C, Figure C-41.

4.42 HISTCONT - HISTOGRAM-CONTRAST PROGRAM

4.42.1 Program Description

HISTCONT is a program which attempts to make an image's gray level appear more uniform between the six different ERTS/MSS scan lines. This is achieved by creating comprehensive histograms and performing a contrast conversion on a scan line basis.

The program has two modes of operation: (1) the entire process of histograms and contrasting is executed as a single task, or (2) the initial histograms and creation of the contrast look-up tables is run as a separate step from the contrast portion of the program. Initially the program determines which mode the user specified. If no contrast tables have been input, the program transfers control to the histogram program. This program checks to see if the number of specified input lines is an even multiple of 6 and if it is not, the number of lines is altered to force it to an even multiple of 6. The input tape is forward spaced to the first specified input line, if necessary. Each record is read and the subroutine TALLY stores each line's gray level values into the appropriate scan line bins.

Once all the required lines have been processed, the program computes and outputs six histograms in which each graph represents the intensity frequencies of a unique set of scan lines. Six numeric tables of the intensity frequencies are also printed out. The program then computes a cumulative histogram and tables using one of the scan line sets as the base detection line. A look-up table for each set of scan lines is computed by finding midpoints of input bins relative to the base detector line bins. The five look-up tables are punched out as parameter cards, in case the user wishes to run the contrast phase at a later time. If an output tape has not been specified, control returns to the IDAMS driver. If an output tape has been specified, and the contrast look-up tables have either been input by parameter cards or from the histogram phase, control is transferred to the contrast program. The contrast program calls the subroutine CODE, which applies the appropriate contrast conversion on each line. As a final program verification, the program generates new histograms for each of the six scan line sets. Control is then returned to the IDAMS driver.

4.42.2 Parameters

1. IBASE = The base detector line number (1 through 6)

2. ITAB = (a) If not specified, implies the histogram and contrast phases are to be executed

   (b) A table of 320 values (5 sets of 64 new values to be used in the contrast phase). When these values are present, only the contrast portion of the program is executed.

4.42.3 Input

HISTCONT requires one input tape in standard IDAMS format.

4.42.4 Output

HISTCONT generates numerous histograms, punched parameter cards, and either one or no output tapes in standard IDAMS format, depending on the mode selected. They have the following significance:

Mode	Output
Histogram only	Six scan line histograms, one cumulative histogram of base detector line, punched parameter cards of look-up tables
Contrast only	Single output tape, six scan line histograms
Histogram and contrast	All of the above mentioned outputs

4.42.5 Example

It is desired to make the image CONT1 gray levels appear more uniform between the six different ERTS/MSS scan lines. The entire 3000 by 3000 picture is to be processed. The base detector line is line 4 and the user specifies that the output image name be OUT1. Appropriate IDAMS source statements are:

```
CONTHIST, (CONT1,49,1), (1,1,3000,3000), (OUT1,48,1),1
4
```

4.42.6 Messages

None

4.42.7 Flowchart

See Appendix C, Figure C-42.

4.43 JOYSTICK - INTERACTIVE DISPLAY PROGRAM

4.43.1 Program Description

The task program JOYSTICK supplements the numerous user image display and manipulation functions provided by the 212 display package. The user controls the functions by depressing buttons on the joystick box, which have been predefined to perform specific functions. Also, several available capabilities are displayed on the CDC 212 and the user interactively inputs the desired function codes via the 212 keyboard. The task is subdivided into a main driver routine and two segments. The first segment handles all of the functions except the ZOOM capability, which resides in the second segment.

Once control is passed to the program JOYSTICK in the first segment, the initial reseau coordinates and box coordinates are set. The subroutines TVCON, TTWCON, and JOYCON are called which connect with the TV hardware, the CDC 212 hardware, and the DDI (analog-to-digital converter), respectively. A call to the subroutine CDCON enables the following function code table to be displayed on the 212.

IDAMS

FUNCTION CODES

(CODES 01 - 09 ARE FUNCTIONAL IN TASK DISPLAY)

LOCATE	10				
DATA	11	ZOOM	17	REWIND	13
DATA1	12	EXIT	18	FORWARD	14
SELECT	16			REVERSE	15

The program calls the subroutine STORE, which waits until a function code has been input to the 212 or a joystick button has been activated. The joystick box contains six buttons and the joystick control. The buttons and stick have been assigned the following functions:

Button	Function
1	Generates a box on the TV; if a box is already displayed on the TV, by activating button 1 the box is erased.
2	Generates a reseau on the TV; if a reseau is already displayed on the TV, by activating button 2 the reseau is erased.
3	Returns the coordinates of the box or reseau which is presently displayed on TV screen.
4	Enables the user to move the box or reseau left, right, up, or down, by moving the joystick left, right, up, or down.
5	Enables the user to enlarge and shrink the box by moving the joystick up and down, respectively.
6	Enables the user to vertically enlarge and shrink a box by moving the joystick up and down; and enables the user to horizontally enlarge and shrink a box by moving the joystick right and left, respectively.

If a function code has been input via the 212, the program converts the code from a BCD number to an integer value and branches to the appropriate subsection, depending upon the value of the function code. If one of the joystick buttons has been activated, STORE returns the button number, and JOYSTICK branches to the appropriate subsection.

4.43.1.1 Description of Joystick Function Buttons

4.43.1.1.1 Button 1

If the joystick function button 1 was depressed, the program checks to see if a box or a reseau is already displayed on the TV. If a box is on the TV, the program erases it by calling the subroutine KILLIN, and control returns to where the program is waiting for another function code to be input or for the joystick controls to be activated. When no reseau mark is on the TV, the program computes the box coordinates such that the box will be centered on the TV and will be 138 pixels by 100 lines in size. However, if a reseau mark is displayed on the TV, the box coordinates are computed such that the 138 by 100 box will appear centered around the reseau mark location. The reseau mark is then erased by calling the subroutine KILLIN.

Once the box coordinates have been set, the program branches to a subsection of the program which converts the image coordinates to a TV format and sends them to the TV hardware. To convert the Y coordinates to TV format, the values are divided by 2; if the original coordinates were even, then 255 is added to the halved value. This is done to accommodate the TV feature of having a main level and an interlace level. The lines alternate between the main and interlace levels. Therefore, line 1 is equal to the TV line 0, line 2 equals TV line 256, . . . , line 511 equals TV line 255, and line 512 equals TV line 511.

The x coordinates must also be converted to a TV format. Because the TV hardware counts pixels across the line in increments of 11, the hardware must know in which group of 11 the pixel resides (x/11) and the pixel position within the group (MOD(x,11)). The x coordinates are converted, the group number is placed in bits 23-12 of the TV coordinate word, and the remainder value resides in the bits 11-0 of the word. The conversion subsection then calls the program DISP, which sends the TV coordinates to the TV hardware. DISP is a COMPASS routine which connects with the TV (channel 2), sends a function code for a box or reseau mark, and transfers the coordinates to the TV hardware. Control is then returned to the conversion subsection, which returns to the originating function subsection. In the case of the box generator function, control is returned to where the function code table is displayed on the 212 and the program is waiting for another function or joystick interrupt.

4.43.1.1.2 Button 2

When the joystick button 2 is depressed, the program checks to see if a reseau or a box is already displayed on the TV. If a reseau is on the TV, the program erases it by calling the subroutine KILLIN, and control is transferred to where the program is waiting for another function code to be input or for the joystick buttons to be activated. If no box appears on the screen, the program sets the reseau coordinates such that it will be centered on the TV. However, if a box is already displayed, the program computes the reseau coordinates such that it will be centered within the box area. The box is then erased with a call to KILLIN and the program branches to the previously described conversion subsection. After the reseau mark has been placed on the TV, the program redisplays the function code table and waits for the next code to be input or the next button to be activated.

4.43.1.1.3 Button 3

When the third button is depressed, the corresponding subsection writes the location of the box or reseau which is presently displayed on the TV onto the printer and the 212. The program checks to verify that a box or reseau is displayed on the TV. If no marks are on the TV, the control returns to the program area which displays the function table. Otherwise, the program prints out the coordinates of the box or reseau onto the printer. Before the values can be output to the 212, the integers must be converted to left-justified BCD format. This process is done in the subroutine BINBCD, and the reformatted coordinates are printed onto the 212. Control returns to the program area which displays the function code table only after the user has depressed the SEND key on the 212 keyboard.

4.43.1.1.4 Button 4

The fourth button is functional only when used in conjunction with the joystick. When the program senses that the fourth button has been depressed, it transfers control to the CHKJOY subroutine. CHKJOY returns control to JOYSTICK only if the joystick has been activated while the fourth button is depressed or if the fourth button is no longer depressed. When the joystick has been moved, the program attempts to move the box or the reseau on the TV in the same direction that the joystick was moved (up, down, left, or right). Depending upon the direction of the requested movement, the program checks whether the coordinates have reached an edge of the image and, therefore, cannot be moved in the requested direction. If this test is positive, then control returns to the program area which displays the function code table. Otherwise, the appropriate coordinates are reduced or increased by one, and control transfers to the TV conversion subsection where the TV formatted coordinates are sent to the TV hardware.

The subroutine ICLOCK is used to control the speed at which the box or reseau is moved. The argument which is sent to ICLOCK and indicates the number of milliseconds to delay processing, is a function of the pressure applied to the joystick. If the joystick is lightly pushed in the desired direction, the ICLOCK variable is large and the box or reseau movement is slow. The speed of the box (reseau) movement increases as more pressure is applied to the joystick. The program continues to move the box or reseau until either the fourth button is no longer depressed, the joystick is no longer activated, or the box (reseau) reaches an edge of the image. Control is then returned to the code which displays the function code.

4.43.1.1.5 Button 5

The fifth button is also only functional when used in conjunction with the joystick. When the program senses that the fifth button has been depressed, it checks by calling the subroutine CHKJOY, to see if the joystick has been moved up or down. If the joystick has been moved up, the box is increased; and if the joystick has been moved down, the box is reduced. The box coordinates are checked to see if the box can be enlarged or reduced, depending on which function was requested. If the box is the maximum (minimum) size, then control returns to the program area from which the function code table is displayed. Otherwise, the coordinates are appropriately reduced or increased by one in order to enlarge or reduce the box.

A call to the delay routine, ICLOCK, which stalls processing a specified number of milliseconds, is used in order to make the movement of the enlarge/reduce process increase as the pressure on the joystick increases. Control is transferred to the subsection which converts the coordinates to TV format and sends the values to the TV hardware. The program continues to enlarge or reduce the box until either the fifth button is no longer depressed, the joystick is no longer activated, or the box reaches an edge of the image. Control is then returned to the program area, where the function codes are displayed and the program is waiting for the next command.

4.43.1.1.6 Button 6

The sixth and last button is used in conjunction with the joystick. When button 6 is depressed and the joystick is moved up and down, the box is vertically increased and reduced, respectively. When the joystick is moved right and left, the box is horizontally enlarged and reduced, respectively. The box coordinates are checked to see if the box can be enlarged (or reduced). If the box is the maximum (minimum) size, then control returns to the program area from which the function code table is displayed. Otherwise, the coordinates are appropriately increased or reduced by one in order to enlarge or reduce the box. The speed of the enlarging or reducing process, which is changed by applying more or less pressure to the joystick, is controlled by calls to the delay routine, ICLOCK. The program then transfers control to the subsection which converts the coordinates to TV format and sends the values to the TV hardware. The program continues to enlarge or reduce the box until either the sixth button is no longer depressed, the joystick is no longer activated, or the box reaches an edge of the image. Control is then returned to the program area where the program waits for the next command.

4.43.1.2 212 Functions

4.43.1.2.1 LOCATE Function

When the function LOCATE is requested from the 212, the corresponding subsection writes the location of the box or reseau which is presently displayed on the TV onto the printer and the 212. The program checks to verify that a box or reseau is displayed on the TV. If no marks are on the TV, the control returns to the program area which displays the function table. Otherwise, the program prints out the coordinates of the box or reseau onto the printer. Before the values can be output to the 212, the integers must be converted to left-justified BCD format. This process is done in the subroutine BINBCD, and the reformatted coordinates are printed onto the 212. Control returns to the program area which displays the function code table only after the user has depressed the SEND key on the 212 keyboard.

4.43.1.2.2 DATA Functions

The functions DATA (11) and DATA1 (12) are handled in the same subsection. This subsection transfers image data from tape files to the TV. The program first requests the tape unit number from the user by calling the subroutine CDCON, which prints out the request on the 212. The subroutine STORE is then called, which will return with the user's reply in BCD format. The program converts the value to an integer format. The program requests that the user input the color gun numbers, which will specify the TV disk(s) onto which the image is to be dropped.

The TV has three TV refresher disk files available for image data storage, and and each disk can be assigned to one of the three available color guns (red, green, or blue). The color gun parameter is a value which determines which disk file(s) the user wants to use for storing an image. The parameter is an octal representation of a three-digit binary number, in which each digit corresponds to one of the disk files and the "on-off" conditions are represented by ones and zeros, respectively. The following table shows the correspondence between the color gun number, the disk assignments, and the binary number from which the parameter value was derived.

		Binary Representation		
Color Gun Number	Disk File(s)	Disk 3	Disk 2	Disk 1
1	1	0	0	1
2	2	0	1	0
3	1 and 2	0	1	1
4	3	1	0	0
5	1 and 3	1	0	1
6	2 and 3	1	1	0
7	1, 2, and 3	1	1	1

Once the user has specified the disk file(s) into which the image data are to be stored and the SEND key has been depressed, the image will be dropped to the disk and displayed on the TV. The user can define which color is to be associated with each disk by manually setting the three color wheels switches on the IDAMS Control Panel. The three wheels, from left to right, represent the color guns of red, green, blue, respectively. By setting the wheels to the appropriate disk number, the user has complete control over the color assignment of any image stored in the TV disk files.

The color gun number, which is returned from a call to STORE, is converted to binary and checked to be sure it is valid. If the number is not valid ($0 < N \leq 7$) the program will again request that the user input the color gun numbers. The program reads the label record and prints out the length of the record. Next the program enters a loop to read 32 lines of data and properly position the data in a format necessary for the TV hardware. Because of the main and interlace structure of the TV, even lines are separated from the odd lines. Consequently, as the data lines are read in, pointers are set which direct the data into the appropriate buffer location. For instance, in buffer 1 the lines 2,3,4, ...,31 are sequentially packed, and lines 2,4,6,...,32 are sequentially packed in buffer 2. The tape reads are double buffered, and while the next line is being read, the last line's data is sent to the subroutine, FLIP. This subroutine reverses the pixel order of each word in the line (i.e., if the characters ABCD are input as a word, they would be returned as DCBA in the same word). This procedure is necessary in order to make the data compatible with the TV hardware's counting method.

After a set of 32 lines have been read in and processed by FLIP, the 16 even lines and 16 odd lines are ready to be transferred to the TV. The subroutine LINDIS is called for each set of 16 lines; this routine prepares the data for the transfer, and then outputs it to the TV. LINDIS is a COMPASS subroutine which computes the function code depending upon whether the lines are main or interlace and sends the function code to channel 2 (TV hardware). The subroutine then checks to see if the requested function was DATA (11) or DATA1 (12).

If the request was for DATA, then each word of the 16 line data block has the least significant bit shifted off. This is required because the TV hardware, which counts from left to right, can only handle five of the six bits per pixel value. Without shifting, the most significant bit would be lost, which is an undesirable result. Consequently, by shifting each word to the right one bit, the TV hardware will be picking up the most significant bit and only losing the least significant bit. The DATA1 (12) function however, does not shift the data words but sends them as they are input. This feature is available in case a user wishes to view the data without the shifting procedure.

Once the data words have been prepared for transfer, LINDIS sends two blocks of 16 lines to the TV, channel 2, and waits for the I/O to be completed before returning to DISPLAY1. This procedure of processing data in sets of 32 lines continues until the program senses an end-of-file mark on the input tape. If the program determines that the total number of lines read is not an even multiple of 32, it prints a message indicating that some data lines must have been lost. The program concludes this fact because the number of lines in a TV size image is 512, which is an even multiple of 32. Control then returns to the program area which displays the function code table.

4.43.1.2.3 Tape Functions

The functions REWIND (13), FORWARD (14), and REVERSE (15) are all handled in the same subsection. The program calls CDCON, which requests that the user input the appropriate magnetic tape logical unit number. The subroutine STORE returns the tape unit number in a BCD format and the program converts it into a binary integer value. If the request function was REWIND (13), the program rewinds the tape and returns to the program area which displays the function code table. If the FORWARD (14) or REVERSE (15) functions were specified, the program sends a request for the number of files to be skipped to the user via the subroutine CDCON. The reply is returned from STORE and is converted from BCD format to an integer value. The program then forward spaces or backspaces the appropriate number of files. If the backspace function is being executed, the end-of-file mark is skipped over before control returns to the program area which displays the function code table.

4.43.1.2.4 SELECT Function

The subsection which processes the SELECT (16) function enables the user to select the coordinates of the box to be displayed on the TV screen. By referencing the subroutine CDCON and STORE, the program requests that the user input the coordinates of the desired box. The coordinates must be input in the following order; leftmost pixel value, rightmost pixel value, top line number, and bottom line number. Because the box figure appears in either the main level or the interlace level, the input line values must both be even or odd. If the line numbers are mixed, the program outputs a message to the 212 indicating the error, and then corrects the line numbers by forcing them to both be even or odd values.

The program receives the parameters from the subroutine STORE and scans the parameter list from the last word of the input array to the first word. The program ignores blanks and expects commas to be the separator between coordinates. The coordinates are converted to integer format and sent to the TV conversion subsection, which converts the values to TV format and sends them to the TV hardware. If the parameter list has not been correctly input, the program requests that the coordinates be input again. Once the box has appeared on the TV, control returns to the program section to display the function code table.

4.43.1.2.5 ZOOM Function

The last display function is the ZOOM (17) function which takes the area bounded within a box on the TV image and increases or reduces it into a TV size image. This function is accomplished by the program ZOOM, which is in the second overlay segment. The ZOOM routine initially requests by referencing CDCON and STORE, the name of the image which is displayed on the TV, the tape unit on which the TV image tape resides, and the file number of the image. These parameters are stored in the label array, LBLIN.

The label processing routine, LBLRD, is then called to read the TV image file label and, the information stored in LBLIN is used to verify that the tape is positioned at the requested image. The requested file label contains the name of the master tape (the image tape from which the TV image originated) and, using this information, the program sends a message to the user via the 212 reminding him that a specific master tape must be mounted. The reduction or enlargement factor that was used when creating the TV image from the master image is stored in word 11 of the TV file label. Using this factor, the box coordinates, and the dimensions of the master image, the program determines whether the master image must be increased or reduced to create the desired TV image. If the box enclosed area would not result in an optimum TV size image, the program sends a message to the user asking if the user still wants to create the requested TV image. If the user replies negatively, the program branches back to the program area which displays the function code table. If the user replies positively, the program continues processing.

A request is made for the master tape logical unit number and file number, and then the program reads in the master image file label. The program then requests via the 212 that the user supply the output TV image name, tape unit number, and file number. The output label is written onto the specified tape. Before going to the segment which carries out the actual reduction or enlargement, the program requests that the user verify that all of the parameters are correct. If the response is negative, the program branches to the beginning of the ZOOM routine and begins the request for input parameters again. If the user replies positively, the program continues and reduces or increases the master image data.

ZOOM computes the largest integral reduction or multiplication factor which will just permit the input to fit within a TV size image. An output line buffer is set up with edge fill characters, and the output label record is written. If any fill lines are required at the top of the TV picture, they are written out at this time. A main increase or reduce processing loop is entered. If no increase or reduction is necessary, the master image is just transferred to the output image. If a reduction is required, the program reads an input line, reduces its length by the reduction factor, and stores it in an internal array. This continues until enough input lines are collected to form one output line. Averaging is then performed between lines, and the completed line is output. If an enlargement is involved, the program reads an input line, enlarges its length by the multiplication factor, and outputs the enlarged line the requested number of times. After all lines have been processed for both enlargements and reductions, the program writes out any remaining lines of bottom fill. Control then returns to the main driver which calls segment 1, JOYSTICK. The IDAMS function code table is again displayed.

4.43.1.2.6 EXIT Function

The user exits from the JOYSTICK package by selecting the EXIT (18) function code. When the program receives this code it returns to the main driver, which then returns to the IDAMS system.

4.43.2 Parameters

JOYSTICK calls the IDAMS joystick package whose parameters are provided interactively through the 212 Display Station or through the joystick control box. By specifying the name JOYSTICK on the task card, the following functions code table is displayed on the 212 screen:

IDAMS

FUNCTION CODES

(CODES 01 - 09 ARE FUNCTIONAL IN TASK DISPLAY)

LOCATE	10				
DATA	11	ZOOM	17	REWIND	13
DATA1	12	EXIT	18	FORWARD	14
SELECT	16			REVERSE	15

In order to execute any one of the functions, the user must type in the corresponding numeric code and depress the SEND key. A description of each function is given below:

Code	Function	Description
10	LOCATE	Returns the coordinates of the box or reseau which is presently displayed on TV screen. When the user wants to clear the coordinates from the 212 and have the function code table reappear, the SEND key must be depressed.
11	DATA	Drops an image tape file which contains 64 gray level data onto the TV. The program requests two input parameters. The tape unit on which the image tape is mounted must be keyed in after the request appears on the 212. After the SEND key is depressed, a request for the color gun number will appear. The TV has three TV refresher disk files available for image data storage and each disk can be assigned to one of the three available color guns (red, green, or blue). The color gun parameter is a value which determines which disk file(s) the user wants to use for storing an image. The parameter is an octal representation of a three-digit binary number, in which each digit corresponds to one of the disk

Code	Function	Description
11 (Cont'd)		files and the "on-off" conditions are represented by ones and zeros, respectively. The following table shows the correspondence between the color gun number, the disk assignments and the binary number from which the parameter value was derived.

Color Gun Number	Disk File(s)	Binary Representation Disk 3	Disk 2	Disk 1
1	1	0	0	1
2	2	0	1	0
3	1 and 2	0	1	1
4	3	1	0	0
5	1 and 3	1	0	1
6	2 and 3	1	1	0
7	1, 2, and 3	1	1	1

Once the user has specified the disk file(s) into which the image data are to be stored and the SEND key has been depressed, the image will be dropped to the disk, and displayed on the TV. The user can define which color is to be associated with each disk by manually setting the three color wheels switches on the IDAMS Control Panel. The three wheels, from left to right, represent the color guns of red,

Code	Function	Description
11 (Cont'd)		green, blue, respectively. By setting the wheels to the appropriate disk number, the user has complete control over the color assignment of any image stored in the TV disk files. Once the color gun number has been entered and the SEND key depressed, the image will be dropped on the TV.
12	DATA1	Drops an image tape file onto the TV. The required parameters are described above under function code 11. DATA1 differs from DATA in that the data contains 32 gray level values and, therefore, does not have the least significant bit shifted off. However, the most significant bit will be truncated since only five bits of data can be displayed at one time.
13	REWIND	Rewinds a requested tape. The program requests on which tape unit the required tape is mounted. After keying in the tape unit number and depressing the SEND key, the tape is rewound to loadpoint.
14	FORWARD	Forward spaces a tape a specified number of files. The program requests the logical unit number on which the required tape is mounted. After keying in the tape unit number, the program requests the number of files over which the tape is to be spaced

Code	Function	Description
14 (Cont'd)		forward. Once the SEND key is depressed, tape is forward spaced the specified number of files.
15	REVERSE	Backspaces a tape a specified number of files. The program requests the logical unit number on which the required tape is mounted. After keying in the tape unit number, the program requests the number of files over which the tape is to be backspaced. Once the SEND key is depressed, the tape is backspaced the specified number of files.
16	SELECT	Enables the user to select the coordinates of a box which is to be displayed on the TV. The program requests that the coordinates be input in the following order - leftmost pixel, rightmost pixel, top line, lower line. The line numbers should be paired even, or odd, but not mixed. This is a display hardware requirement. The parameters must be separated by commas and the final parameter must be followed by a blank. Any blanks placed between parameters are ignored.
17	ZOOM	Takes the area bounded within a box on the TV image and increases or reduces it into a TV size image. The program requests

Code	Function	Description
17 (Cont'd)		the name of the image which is presently displayed on the TV. After entering in the name and depressing the SEND key, the image's tape unit is requested, followed by a request for the file number (be sure to specify the file number with two digits). A message reminding the user that the master tape must be mounted is displayed on the 212, and is followed by a request for the master tape's unit number and file number. Before the program begins the ZOOM procedure, information about the new output tape is requested. The user is asked to supply the output tape name, the unit number, and the file number. The program requests that the user specify if the input parameters are believed to be correct. If a "Y" is returned, the program continues with the ZOOM process. However, if an "N" has been returned, the program begins the input parameter requests again. This gives the user, who is aware of an input parameter error, another chance to supply the correct input. (Note: After keying in the proper response to all requests, remember to press the SEND key.) The program will display on the TV whether an increase or reduction

Code	Function	Description
17 (Cont'd)		of the master image was necessary and the multiplication factor involved. The "ZOOM" image will reside on the output tape when the program is completed, and the user must reference DATA when he wishes to drop the image onto the TV screen.
18	EXIT	Returns control to the IDAMS main driver program.

For special instructions pertaining to the operational procedures involved when using JOYSTICK and the TV, refer to Section 3.4.3 (Special Instructions for the DISPLAY User) in the IDAMS User's Guide.

The other means of supplying parameters is by activating one of the six buttons on the joystick control box. The buttons and stick have been assigned the following functions:

Button	Function
1	Generates a box on the TV; if a box is already displayed on the TV, by activating button 1 the box is erased.
2	Generates a reseau on the TV; if a reseau is already displayed on the TV, by activating button 2 the reseau is erased.
3	Returns the coordinates of the box or reseau which is presently displayed on the TV screen.

Button	Function
4	Enables the user to move the box or reseau left, right, up, or down by moving the joystick left, right, up, or down.
5	Enables the user to enlarge and shrink the box by moving the joystick up and down, respectively.
6	Enables the user to vertically enlarge and shrink a box by moving the joystick up and down; enables the user to horizontally enlarge and shrink a box by moving the joystick right and left.

4.43.3 Input

JOYSTICK has variable inputs depending upon which functions are requested. If the functions DATA, DATA1, REWIND, FORWARD, or REVERSE are requested, JOYSTICK requires a single input image tape in standard IDAMS format. If the ZOOM function is requested, two input image tapes in standard IDAMS format are required (a TV size image tape and the master image tape).

4.43.4 Output

JOYSTICK drops image data onto the TV and displays a box and reseau mark on the TV screen. If ZOOM is referenced, a single output image tape in standard IDAMS format is produced.

4.43.5 Examples

The fourth file on the image tape which is mounted on tape unit 49 is to be dropped onto all three TV disks. A reseau mark is to be placed on the image and moved to a desired point, where a box replaces the reseau mark. The box is increased slightly and the coordinates are printed out. The enclosed area is increased to a TV size image and the new image is dropped onto the first TV disk. The following communication would be required to achieve the above operations:

1. The following single control card would be submitted.

```
JOYSTICK
```

2. The code 14 (FORWARD) would be entered on the 212 and the SEND key depressed. The user would specify the number 49 to the magnetic tape unit request and then would specify the number 03 to the request of number of files to be skipped.

3. The code 11 (DATA) would drop the fourth file on the TV after the user has specified the logical unit number 49 and color gun number 7 (binary representation indicating all three disk files).

4. The second button on the joystick control box would be depressed and a reseau mark would appear on the TV.

5. By depressing the fourth button and moving the joystick to the left, right, up, and down, the reseau would be moved left, right, up, and down, respectively.

6. Once the user has located the area of interest, the first button would be depressed which would replace the reseau mark with a box.

7. The box would be enlarged by depressing the fifth button and moving the joystick upward.

8. The box coordinates would be displayed on the 212 by either depressing the third button on the joystick control box or by referencing code 10 on the 212.

9. The code 17 (ZOOM) would increase the enclosed area using the master tape image data and output the area as a TV size image.

10. The new image would be dropped onto the color gun 1 disk by referencing code 11 (DATA) again. The output tape unit from the ZOOM phase would be entered as the input tape unit and the color gun number would be 1.

4.43.6 Messages

JOYSTICK generates no special messages.

4.43.7 Flowchart

See Appendix C, Figure C-43.

4.44 MSSCON - SPECIAL PURPOSE CONVOLUTION ROUTINE

4.44.1 Program Description

The task program MSSCON provides the capability for convolving an image data set with a special-purpose, user-supplied weight matrix consisting of six individual row matrices used cyclically in processing the image. In essence, the program is a modification of program CONVOLVE which allows the use of a particular type of weight matrix. Applications include all standard CONVOLVE applications: simulation of sampling and blurring processes; digital filtering for edge enhancement; and blur reduction. Output values can be generated for each input pixel, or can be specified at larger increments at the user's discretion. The weight table is a sequence of six row matrices and must be specified in its entirety.

Initially, the input parameters are accessed and the amount of COMMON required for storage of the weight tables is computed. In order to maximize processing efficiency, COMMON which is required for picture data is allocated dynamically. For each of the six row weight matrices, separate sums of positive and negative weights are made, and testing of each sum is done to ensure that none, after normalization, exceeds 32.5 in magnitude, since a larger value could cause an uncorrectable overflow. Each weight is then normalized so as to make the sum of the weights in each row equal to zero.

The program next compares the dimensions of the specified region of the input image with the size of the entire input image. If the specified region exceeds the available input data, CONVOLVE reduces the specified numbers of lines and pixels to fit the available data and writes an advisory message on the printer. If the specified region extends to or near the edge of the available data, the program makes provision for copying the boundary pixels outward to minimize edge effects by ensuring that each element of the weight matrix will always have a corresponding pixel value.

The program then compares the dimensions of the specified region with the available COMMON size. If the entire input region will not fit into core at one time, the program makes provision for breaking the image into horizontal strips. Next, it computes the remaining constants required for reading, writing, shifting, and convolving the data. It passes the constants required by the COMPASS subroutine ADDWTS by calling ADDPRM, which stores the parameters and modifies ADDWTS logic to provide maximum efficiency for the particular set of parameters.

The program reads input data into core until the available space is filled, and copies data on the edges of the input image outwards, if required. In order to generate one line of output, a call to subroutine ADDMSS is first required in order to set pointers to the appropriate row matrix in the weight table.

The program calls subroutine ADDWTS to carry out the convolution to generate one line of output. To compute each output pixel, ADDWTS first resets the variable SUM to zero. For each weight, from one to four input pixels, depending on the symmetry of the weight array, are loaded and added together, and the sum is multiplied by the weight. This product is added to SUM. When all weights have been used, SUM is divided, with rounding, by 4096 to eliminate the 12-bit fractional part. If the result is negative, it is replaced by 0, the minimum gray level value; if the result is greater than the maximum allowed value of 63, it is reduced to 63. The result is stored into the output buffer, and the input pixel addresses incremented as specified by the user-supplied parameter, and the next output pixel is computed.

The program writes each output line onto the output tape as soon as it is computed. When all output lines have been computed for one block of input data, the program reads an additional block of data into core, after first moving to the top of core any lines from the bottom of the previous input block that are needed for computing additional output. Processing continues one block at a time until the entire output image is complete.

Execution time has three components: tape I/O, computation time, and disk I/O (if any). Tape I/O is normally a small fraction of the total, because the input and output tapes are read or written once without intermediate rewinds. Computation time is about 20 microseconds per output pixel and per weight for symmetric weight arrays, and about four times as long for nonsymmetric arrays; for increments other than one, the numbers of output lines and pixels per line will equal the input numbers divided by the increments.

4.44.2 Parameters

MSSCON requires six special parameters and a table of weights, in addition to the standard parameters that define the input image. These special parameters are:

1. NX = Number of columns in full weight matrix.
2. NY = Number of rows in full weight matrix.
3. INCRX = Increment between output pixels.
4. INCRY = Increment between output lines.
5. INDIV = Quantity by which input weights are to be divided for normalization. If IDIV = 0, weights are divided by their sum.
6. ISYM = Symmetry of weights: 0 = nonsymmetric; 1 = symmetric.
7. Weights, beginning with top line of array and left-hand end of line. For ISYM = 0, NX times NY values must be supplied. For ISYM = 1, only the upper (NY + 1)/2 rows and left-hand (NX + 1)/2 values in each row are entered.

NX can have a maximum value of 256. The product of NX and NY may not exceed about 2000 for a nonsymmetric matrix or 3500 for a symmetric matrix; these values correspond to square arrays approximately 45 × 45 and 60 × 60, respectively.

4.44.3 Input

MSSCON requires a single input image tape in standard IDAMS format.

4.44.4 Output

MSSCON generates a single output image in standard IDAMS format. For large images and weight arrays, the program requires disk storage for temporary output.

4.44.5 Examples

A 100 × 100 section of a standard test pattern is convolved with a set of cyclical weights designed to incrementally shift each successive line to the right by one pixel further than the previous line, with the pattern repeating every six lines. The appropriate weight matrix is:

```
0, 5, 10, 15, 20, 15, 10, 5
1, 6, 11, 16, 19, 14, 9, 4
2, 7, 12, 17, 18, 13, 8, 3
3, 8, 13, 18, 17, 12, 7, 2
4, 9, 14, 19, 16, 11, 6, 1
5, 10, 15, 20, 15, 10, 5, 0
```

Since only a 100 × 100 section is being used, the appropriate IDAMS task and parameter cards are:

```
MSSCON, (TEST1,48,1) (1,1,100,100), (TEST2,47,1),7
```

```
8, 1, 1, 1, 0, 0,
```

```
0, 5, 10, 15, 20, 15, 10, 5
```

```
1, 6, 11, 16, 19, 14, 9, 4
```

```
2, 7, 12, 17, 18, 13, 8, 3,
```

```
3, 8, 13, 18, 17, 12, 7, 2
```

```
4, 9, 14, 19, 16, 11, 6, 1,
```

```
5, 10, 15, 20, 15, 10, 5, 0
```

NOTE: Card format specifications are defined in the User's Guide. Parameters must be supplied in the order shown in Section 4.5.2.

4.44.6 Messages

Message	Explanation
SUM OF WEIGHTS = 0	User has specified weight normalization by dividing by sum of weights (IDIV parameter = 0) and this sum = 0; fatal error.
NY TOO LARGE FOR AVAILABLE CORE	Insufficient core to hold both weight table and NY data segments of minimum possible length; fatal error.
WEIGHT VALUES TOO LARGE	Sum of either positive or negative weights, after normalization, exceeded 32.5, making possible uncorrectable overflow; fatal error.

4.44.7 Flowchart

See Appendix C, Figure C-44.

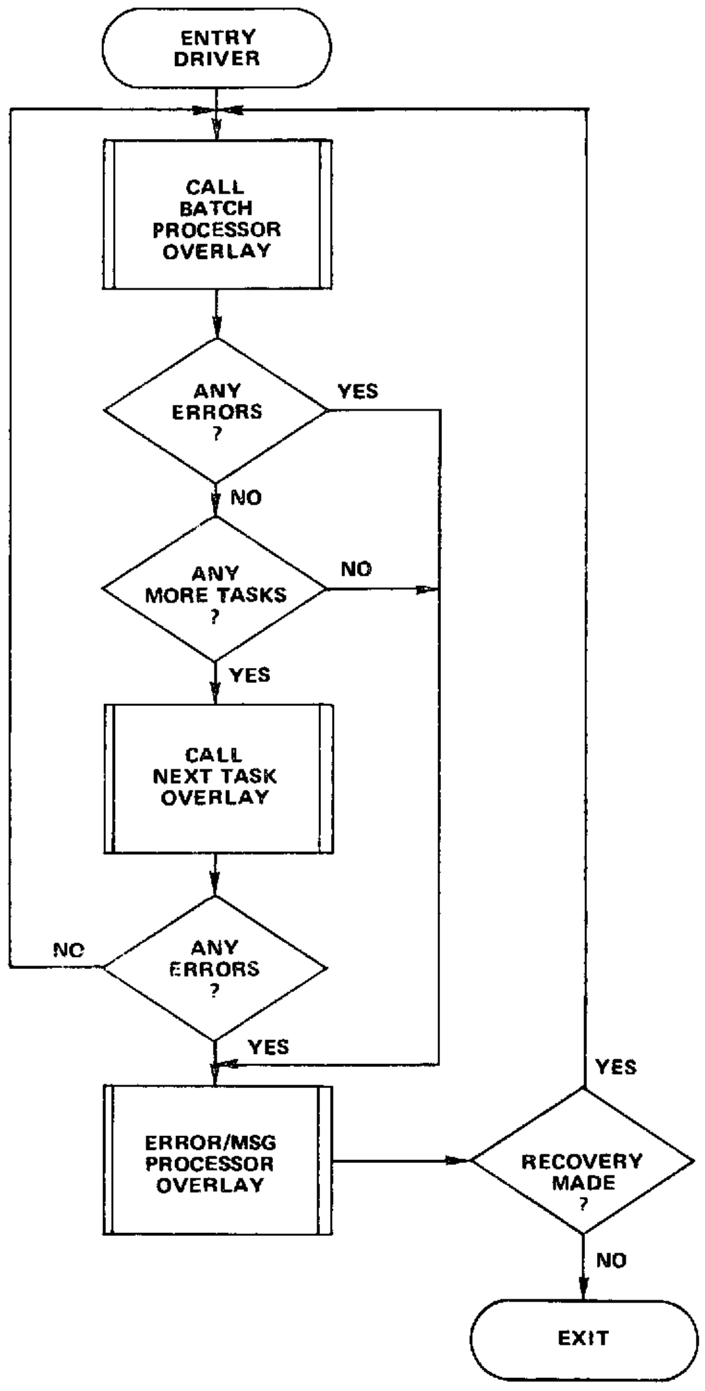


Figure A-1. DRIVER Program Flowchart

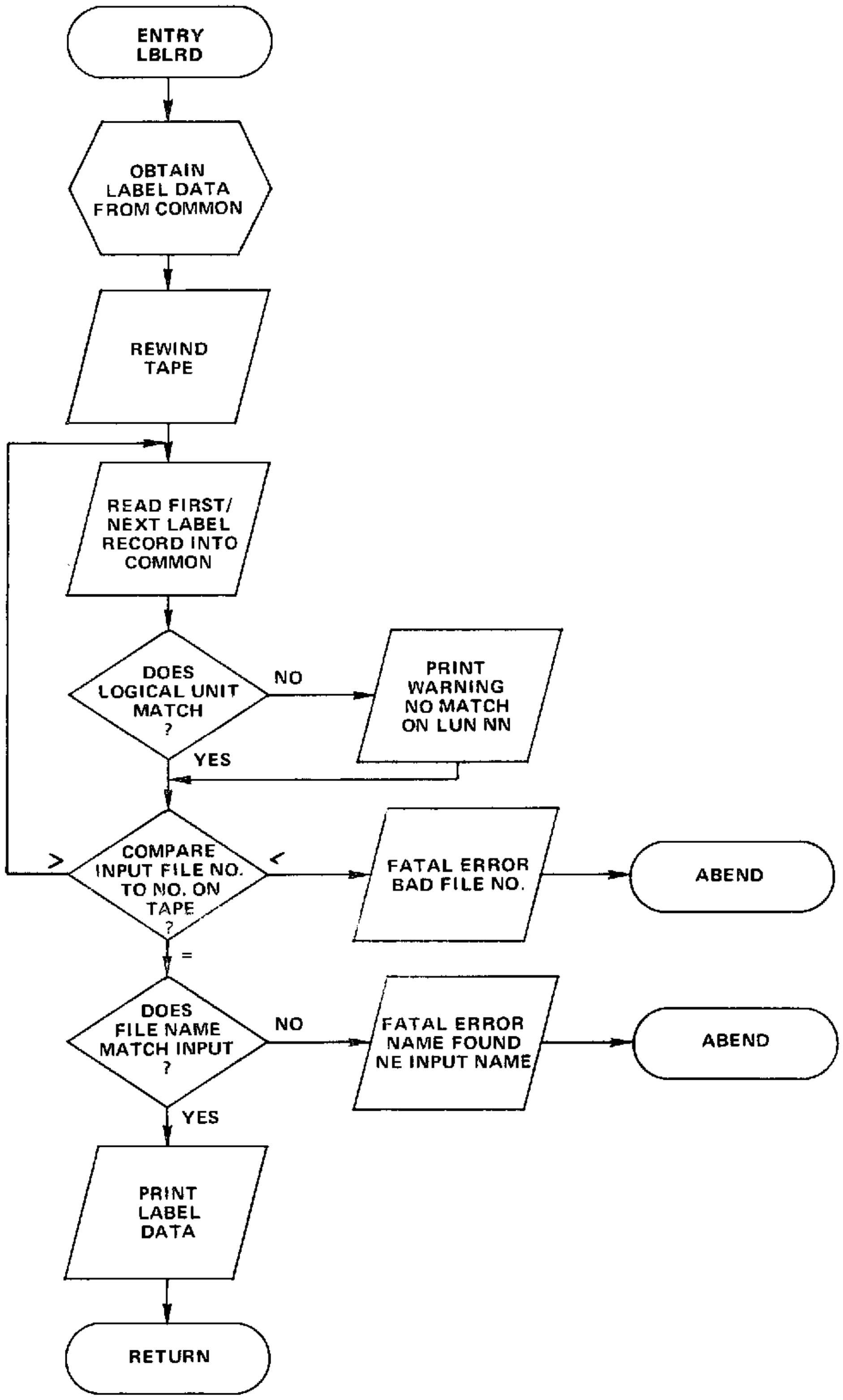

ENTRY LBLRD
OBTAIN LABEL DATA FROM COMMON
REWIND TAPE
READ FIRST/ NEXT LABEL RECORD INTO COMMON
DOES LOGICAL UNIT MATCH ?
NO
PRINT WARNING NO MATCH ON LUN NN
YES
COMPARE INPUT FILE NO. TO NO. ON TAPE ?
>
<
FATAL ERROR BAD FILE NO.
ABEND
=
DOES FILE NAME MATCH INPUT ?
NO
FATAL ERROR NAME FOUND NE INPUT NAME
ABEND
YES
PRINT LABEL DATA
RETURN

Figure B-1. Subroutine LBLRD Flowchart

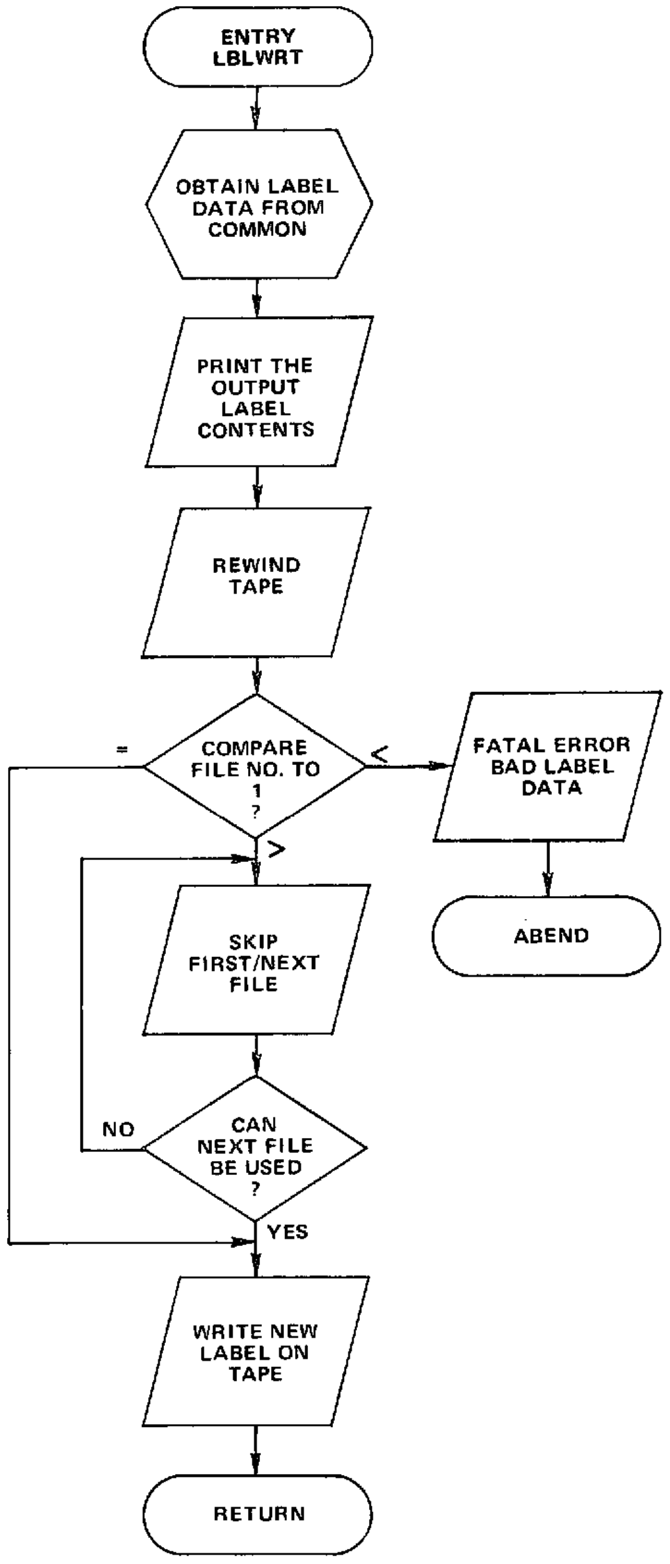


Figure B-2. Subroutine LBLWRT Flowchart

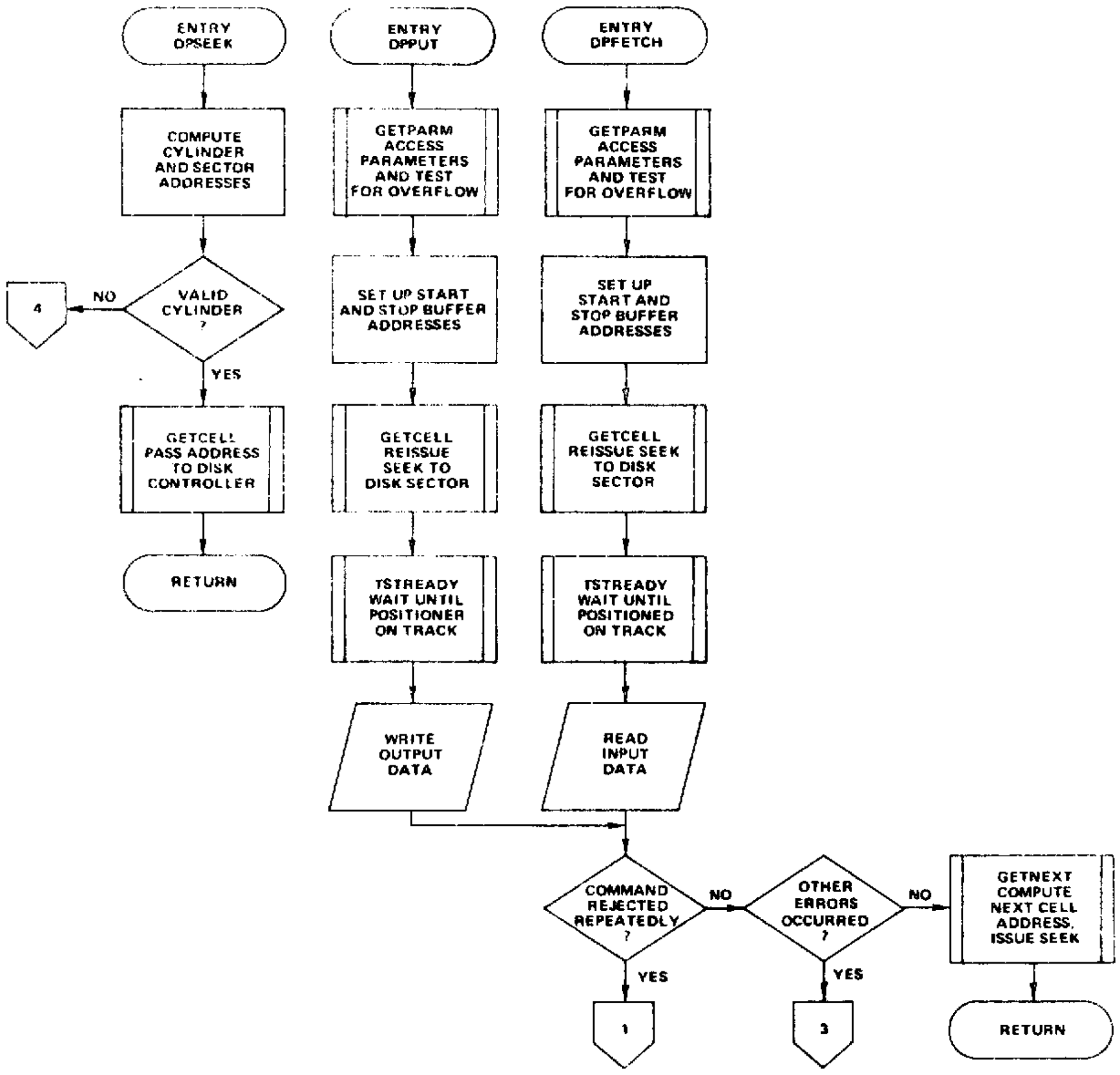

ENTRY OPSEEK
COMPUTE CYLINDER AND SECTOR ADDRESSES
VALID CYLINDER ?
NO
4
YES
GETCELL PASS ADDRESS TO DISK CONTROLLER
RETURN
ENTRY DPPUT
GETPARM ACCESS PARAMETERS AND TEST FOR OVERFLOW
SET UP START AND STOP BUFFER ADDRESSES
GETCELL REISSUE SEEK TO DISK SECTOR
TSTREADY WAIT UNTIL POSITIONER ON TRACK
WRITE OUTPUT DATA
ENTRY DPFETCH
GETPARM ACCESS PARAMETERS AND TEST FOR OVERFLOW
SET UP START AND STOP BUFFER ADDRESSES
GETCELL REISSUE SEEK TO DISK SECTOR
TSTREADY WAIT UNTIL POSITIONED ON TRACK
READ INPUT DATA
COMMAND REJECTED REPEATEDLY ?
NO
YES
1
OTHER ERRORS OCCURRED ?
NO
YES
3
GETNEXT COMPUTE NEXT CELL ADDRESS, ISSUE SEEK
RETURN

Figure B-3. Subroutine IDAMSDSK Flowchart (1 of 3)

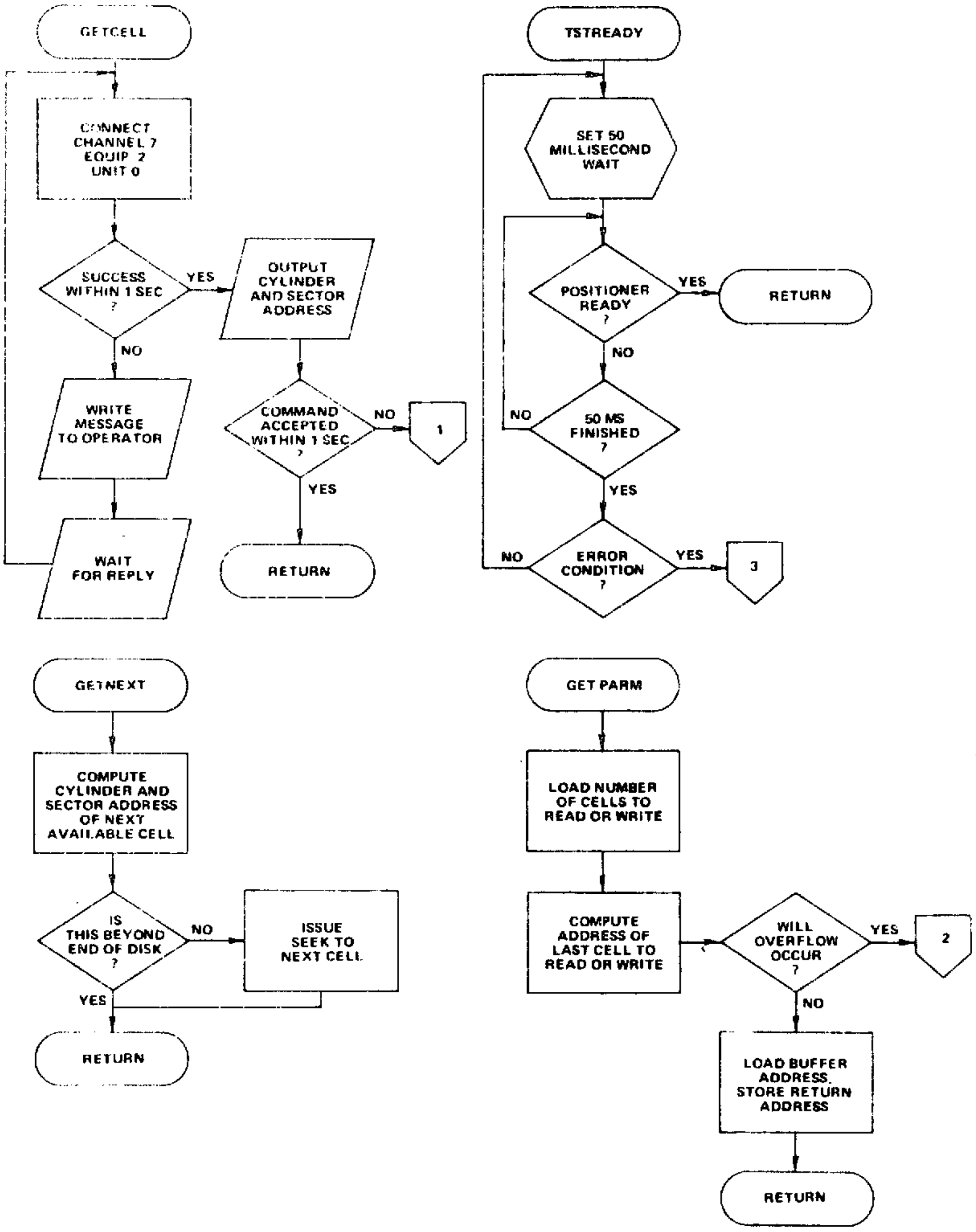

GETCELL
CONNECT CHANNEL 7 EQUIP 2 UNIT 0
SUCCESS WITHIN 1 SEC ?
YES
NO
OUTPUT CYLINDER AND SECTOR ADDRESS
WRITE MESSAGE TO OPERATOR
COMMAND ACCEPTED WITHIN 1 SEC ?
NO
1
YES
WAIT FOR REPLY
RETURN
TSTREADY
SET 50 MILLISECOND WAIT
POSITIONER READY ?
YES
RETURN
NO
50 MS FINISHED ?
NO
YES
ERROR CONDITION ?
NO
YES
3
GETNEXT
COMPUTE CYLINDER AND SECTOR ADDRESS OF NEXT AVAILABLE CELL
IS THIS BEYOND END OF DISK ?
NO
ISSUE SEEK TO NEXT CELL
YES
RETURN
GET PARM
LOAD NUMBER OF CELLS TO READ OR WRITE
COMPUTE ADDRESS OF LAST CELL TO READ OR WRITE
WILL OVERFLOW OCCUR ?
YES
2
NO
LOAD BUFFER ADDRESS. STORE RETURN ADDRESS
RETURN

Figure B-3. Subroutine IDAMSDSK Flowchart (2 of 3)

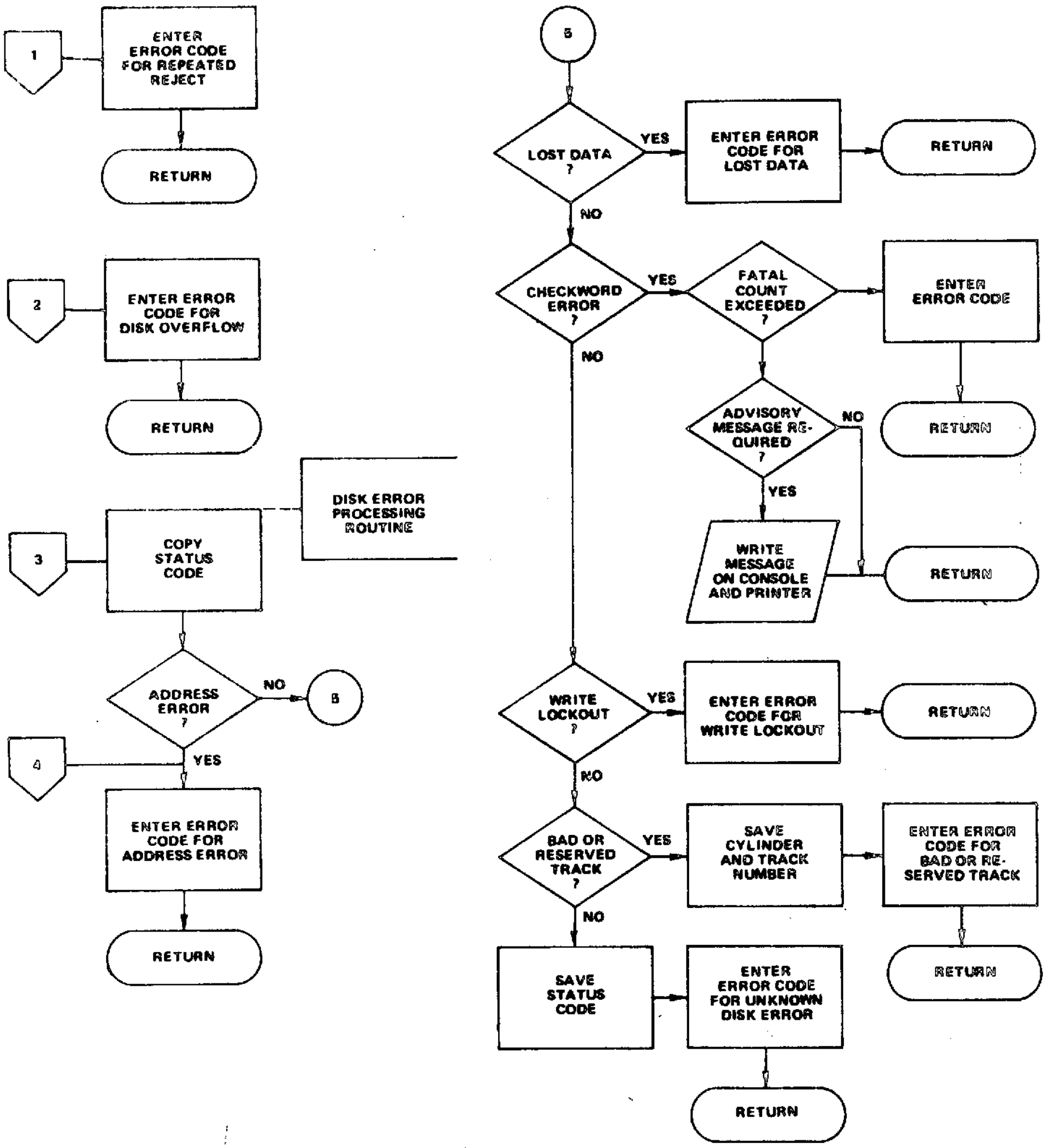

1
ENTER ERROR CODE FOR REPEATED REJECT
RETURN
2
ENTER ERROR CODE FOR DISK OVERFLOW
RETURN
DISK ERROR PROCESSING ROUTINE
3
COPY STATUS CODE
ADDRESS ERROR ?
NO
5
YES
4
ENTER ERROR CODE FOR ADDRESS ERROR
RETURN
5
LOST DATA ?
YES
ENTER ERROR CODE FOR LOST DATA
RETURN
NO
CHECKWORD ERROR ?
YES
FATAL COUNT EXCEEDED ?
ENTER ERROR CODE
RETURN
NO
ADVISORY MESSAGE REQUIRED ?
NO
YES
WRITE MESSAGE ON CONSOLE AND PRINTER
RETURN
WRITE LOCKOUT ?
YES
ENTER ERROR CODE FOR WRITE LOCKOUT
RETURN
NO
BAD OR RESERVED TRACK ?
YES
SAVE CYLINDER AND TRACK NUMBER
ENTER ERROR CODE FOR BAD OR RESERVED TRACK
RETURN
NO
SAVE STATUS CODE
ENTER ERROR CODE FOR UNKNOWN DISK ERROR
RETURN

Figure B-3. Subroutine IDAMSDSK Flowchart (3 of 3)

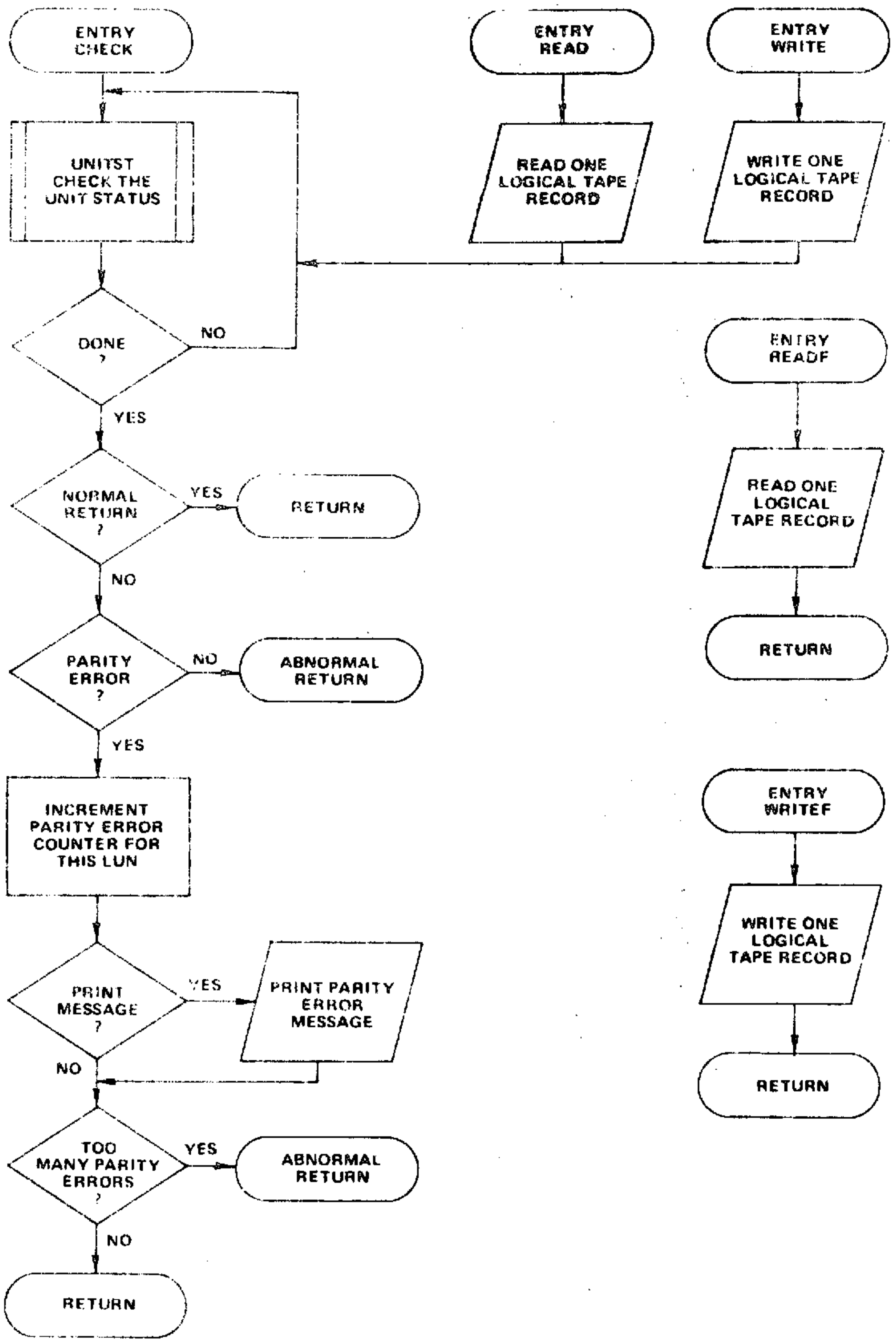

ENTRY CHECK
UNITST CHECK THE UNIT STATUS
DONE ?
NO
YES
NORMAL RETURN ?
YES
RETURN
NO
PARITY ERROR ?
NO
ABNORMAL RETURN
YES
INCREMENT PARITY ERROR COUNTER FOR THIS LUN
PRINT MESSAGE ?
YES
PRINT PARITY ERROR MESSAGE
NO
TOO MANY PARITY ERRORS ?
YES
ABNORMAL RETURN
NO
RETURN
ENTRY READ
READ ONE LOGICAL TAPE RECORD
ENTRY WRITE
WRITE ONE LOGICAL TAPE RECORD
ENTRY READF
READ ONE LOGICAL TAPE RECORD
RETURN
ENTRY WRITEF
WRITE ONE LOGICAL TAPE RECORD
RETURN

Figure B-4. Subroutine READRITE Flowchart

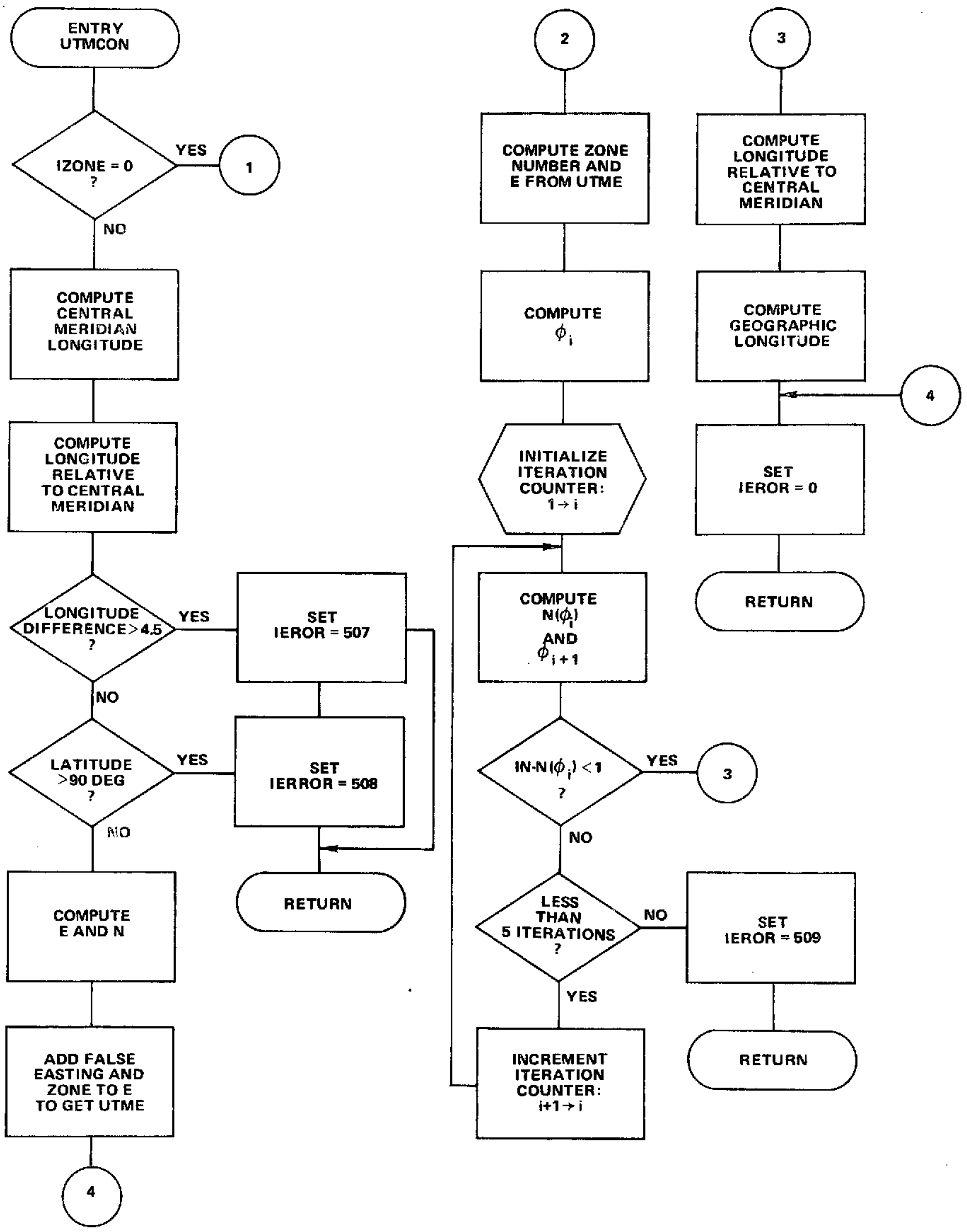

ENTRY UTMCON
IZONE = 0 ?
YES
1
NO
COMPUTE CENTRAL MERIDIAN LONGITUDE
COMPUTE LONGITUDE RELATIVE TO CENTRAL MERIDIAN
LONGITUDE DIFFERENCE > 4.5 ?
YES
SET IEROR = 507
NO
LATITUDE > 90 DEG ?
YES
SET IERROR = 508
NO
RETURN
COMPUTE E AND N
ADD FALSE EASTING AND ZONE TO E TO GET UTME
4
2
COMPUTE ZONE NUMBER AND E FROM UTME
COMPUTE ϕ_i
INITIALIZE ITERATION COUNTER: 1 → i
COMPUTE N(ϕ_i) AND ϕ_{i+1}
IN-N(ϕ_i) < 1 ?
YES
3
NO
LESS THAN 5 ITERATIONS ?
NO
SET IEROR = 509
RETURN
YES
INCREMENT ITERATION COUNTER: i+1 → i
3
COMPUTE LONGITUDE RELATIVE TO CENTRAL MERIDIAN
COMPUTE GEOGRAPHIC LONGITUDE
4
SET IEROR = 0
RETURN

Figure B-5. Subroutine UTMCON Flowchart

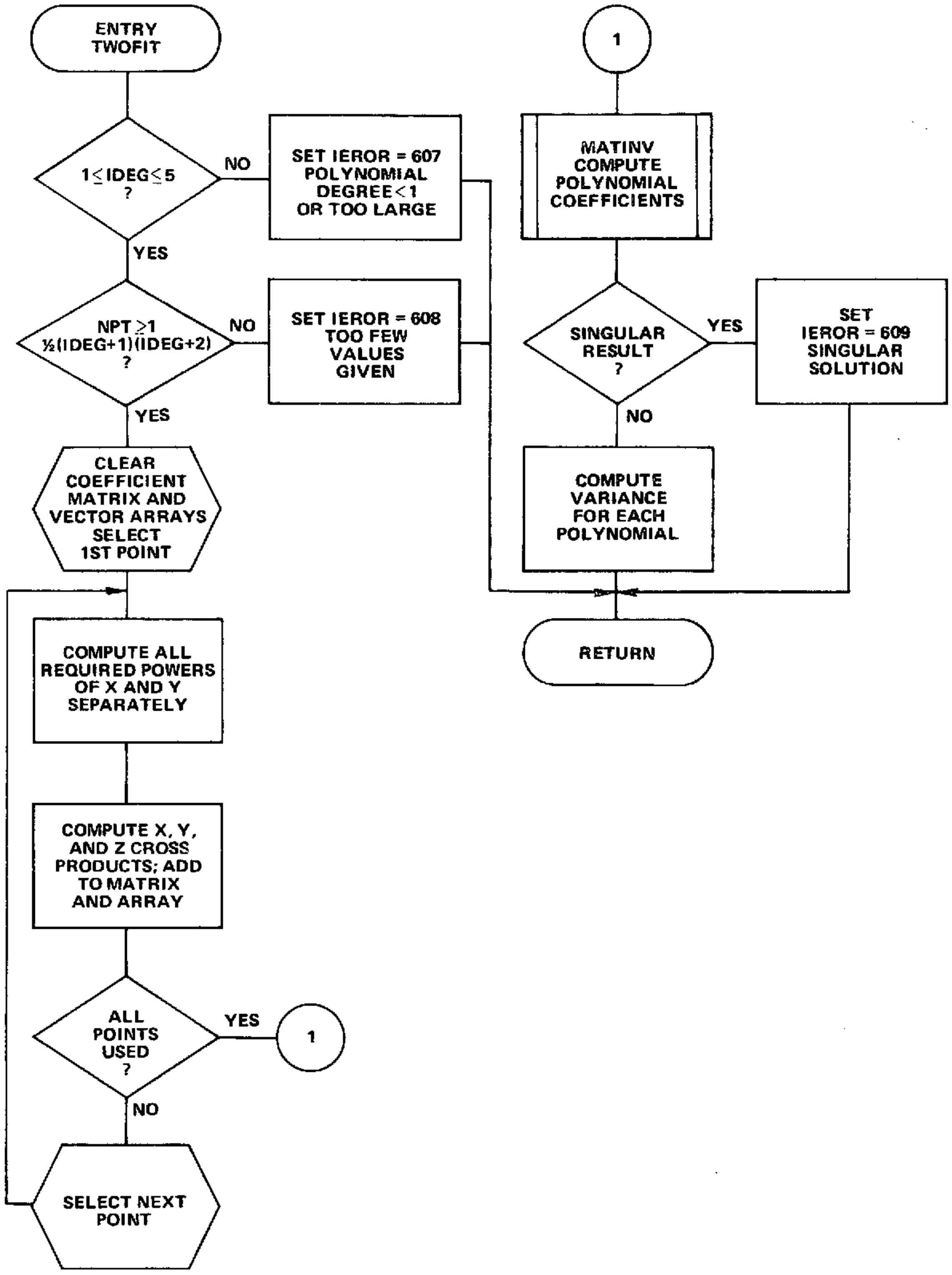

ENTRY TWOFIT
1≤IDEG≤5 ?
NO
SET IEROR = 607 POLYNOMIAL DEGREE<1 OR TOO LARGE
YES
NPT ≥1 ½(IDEG+1)(IDEG+2) ?
NO
SET IEROR = 608 TOO FEW VALUES GIVEN
YES
CLEAR COEFFICIENT MATRIX AND VECTOR ARRAYS SELECT 1ST POINT
COMPUTE ALL REQUIRED POWERS OF X AND Y SEPARATELY
COMPUTE X, Y, AND Z CROSS PRODUCTS; ADD TO MATRIX AND ARRAY
ALL POINTS USED ?
YES
1
NO
SELECT NEXT POINT
1
MATINV COMPUTE POLYNOMIAL COEFFICIENTS
SINGULAR RESULT ?
YES
SET IEROR = 609 SINGULAR SOLUTION
NO
COMPUTE VARIANCE FOR EACH POLYNOMIAL
RETURN

Figure B-6. Subroutine TWOFIT Flowchart

C4

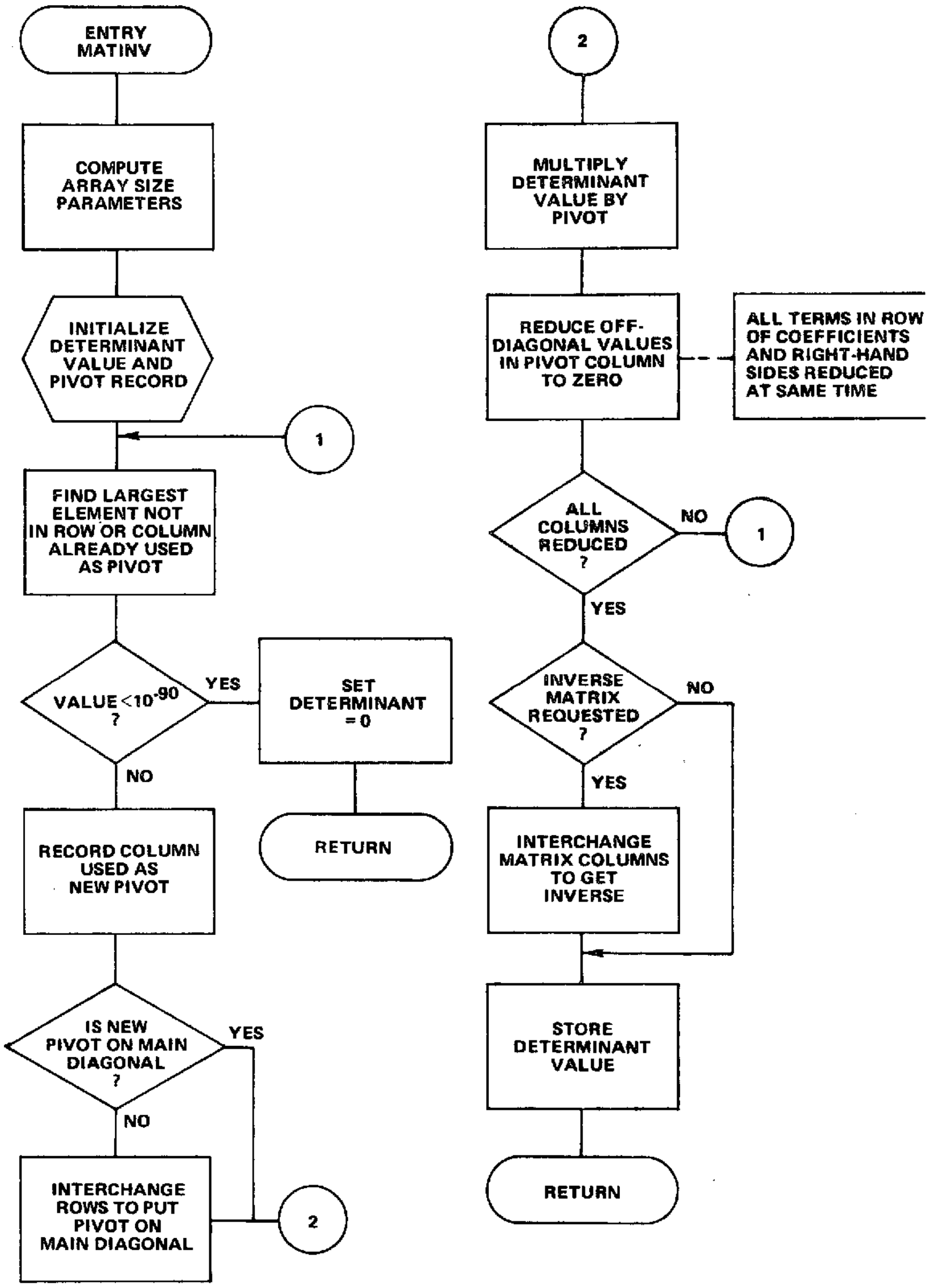

ENTRY MATINV
COMPUTE ARRAY SIZE PARAMETERS
INITIALIZE DETERMINANT VALUE AND PIVOT RECORD
1
FIND LARGEST ELEMENT NOT IN ROW OR COLUMN ALREADY USED AS PIVOT
VALUE < 10^{-90} ?
YES
SET DETERMINANT = 0
RETURN
NO
RECORD COLUMN USED AS NEW PIVOT
IS NEW PIVOT ON MAIN DIAGONAL ?
YES
NO
INTERCHANGE ROWS TO PUT PIVOT ON MAIN DIAGONAL
2
2
MULTIPLY DETERMINANT VALUE BY PIVOT
REDUCE OFF-DIAGONAL VALUES IN PIVOT COLUMN TO ZERO
ALL TERMS IN ROW OF COEFFICIENTS AND RIGHT-HAND SIDES REDUCED AT SAME TIME
ALL COLUMNS REDUCED ?
NO
1
YES
INVERSE MATRIX REQUESTED ?
NO
YES
INTERCHANGE MATRIX COLUMNS TO GET INVERSE
STORE DETERMINANT VALUE
RETURN

Figure B-7. Subroutine MATINV Flowchart

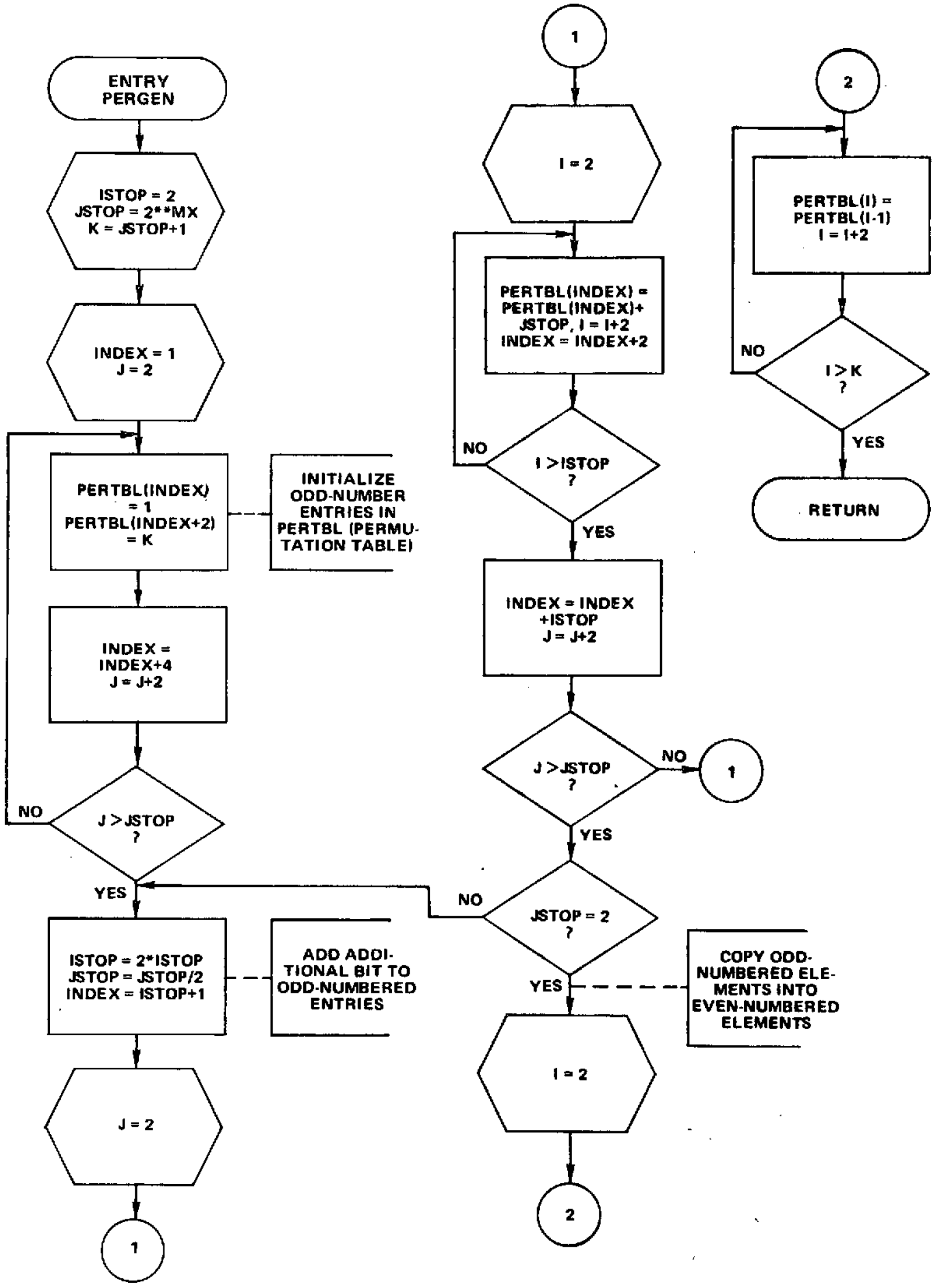


Figure B-8. Subroutine PERGEN Flowchart

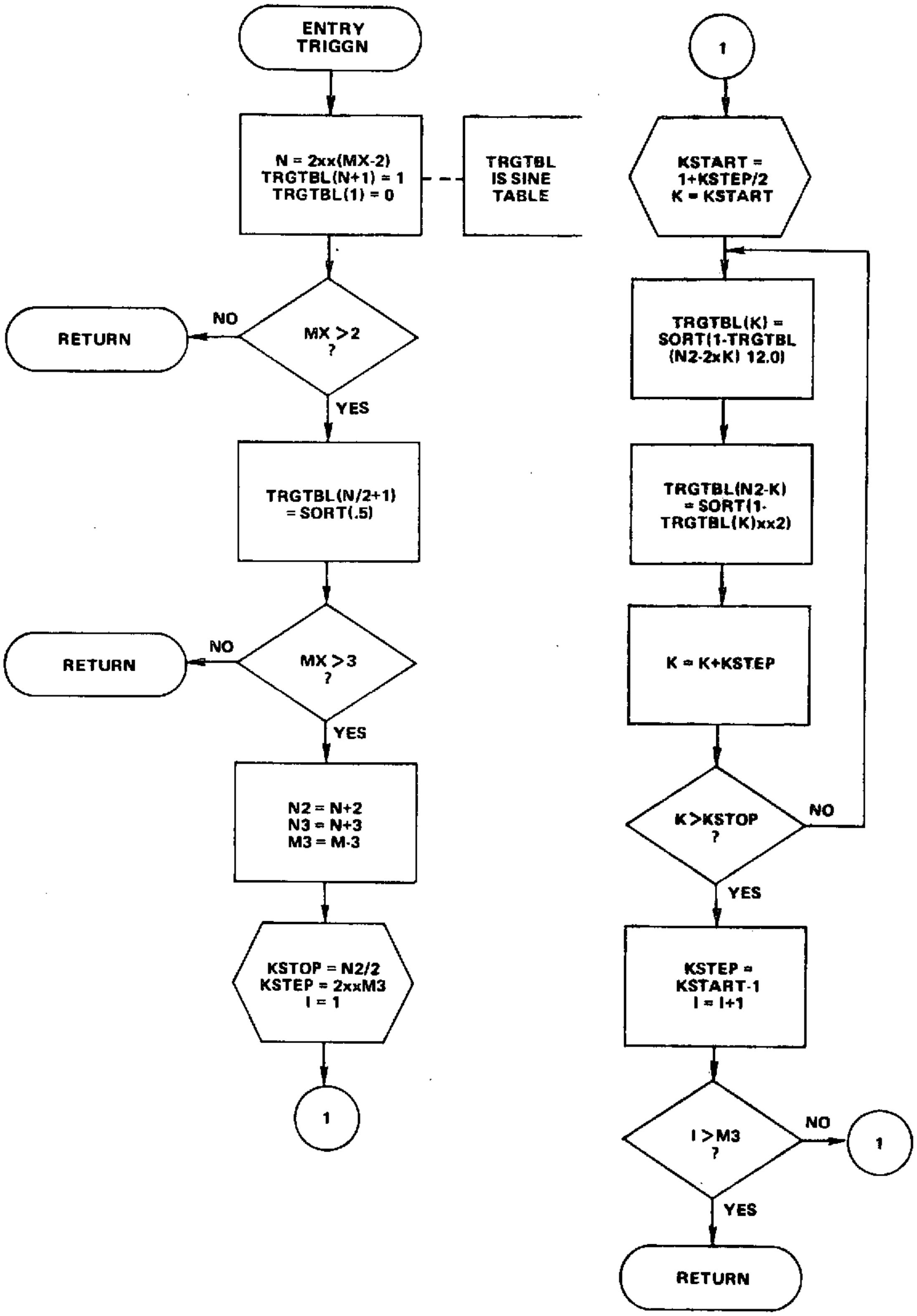


Figure B-9. Subroutine TRIGGN Flowchart

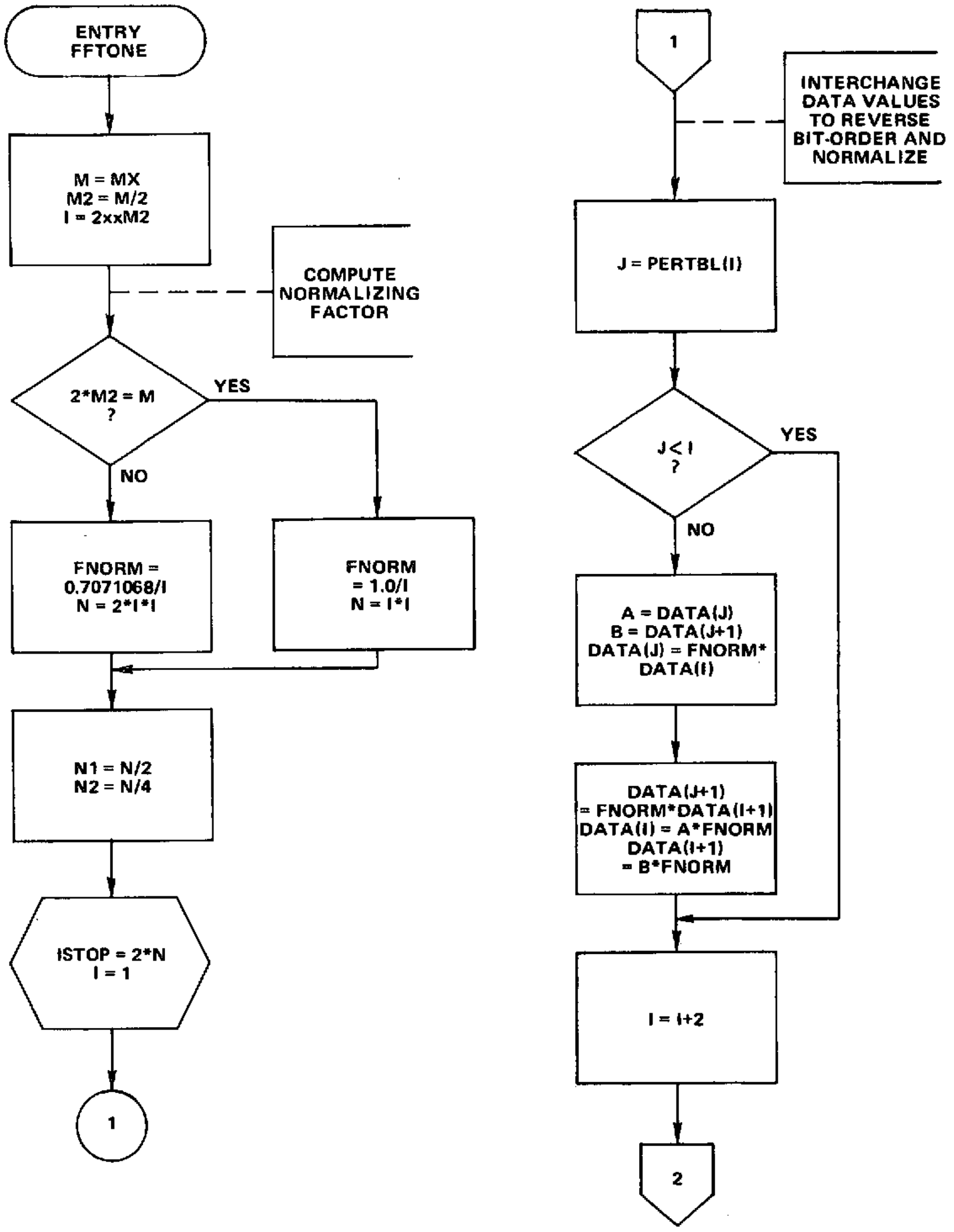


Figure B-10. Subroutine FFTONE Flowchart (1 of 3)

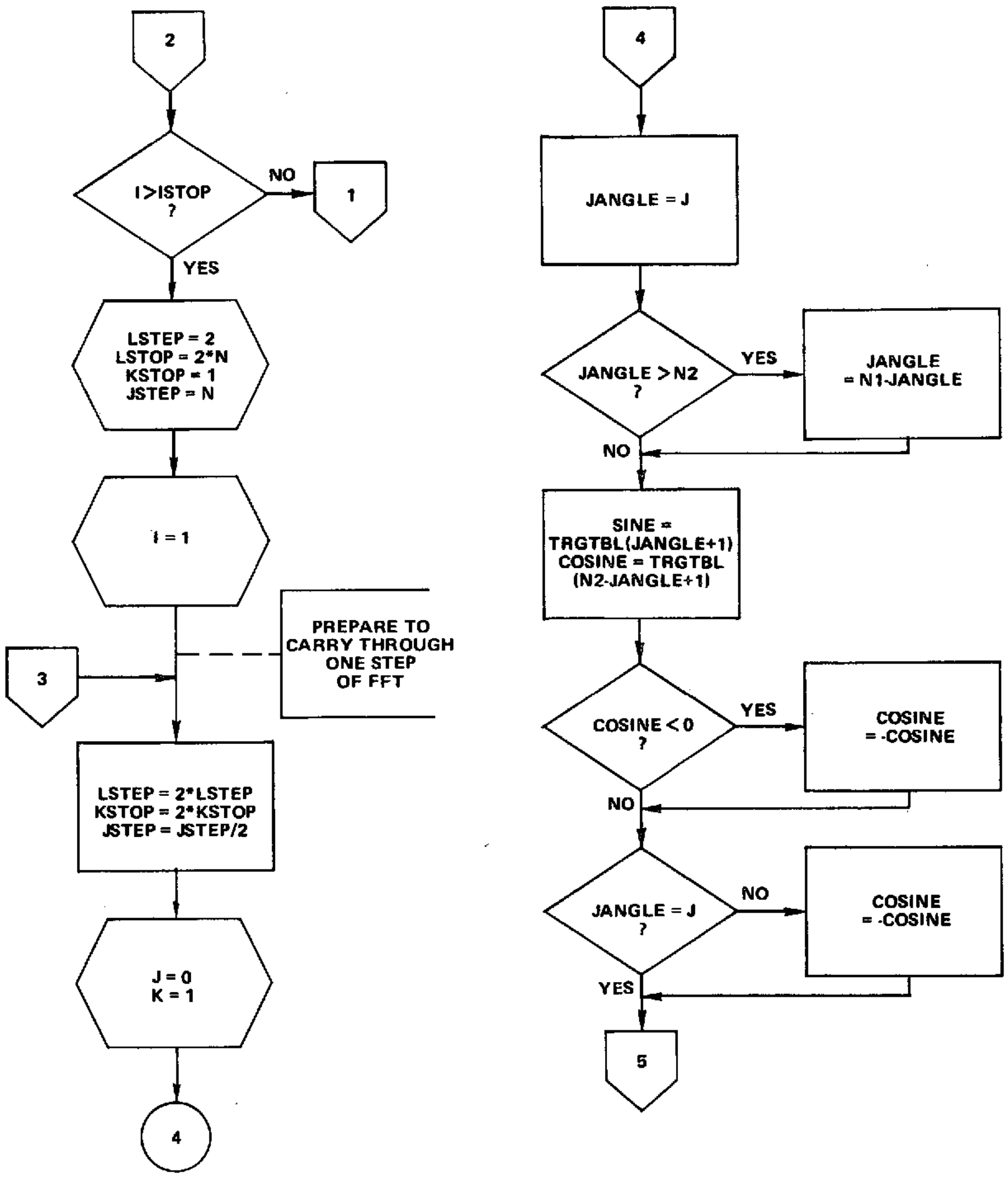


Figure B-10. Subroutine FFTONE Flowchart (2 of 3)

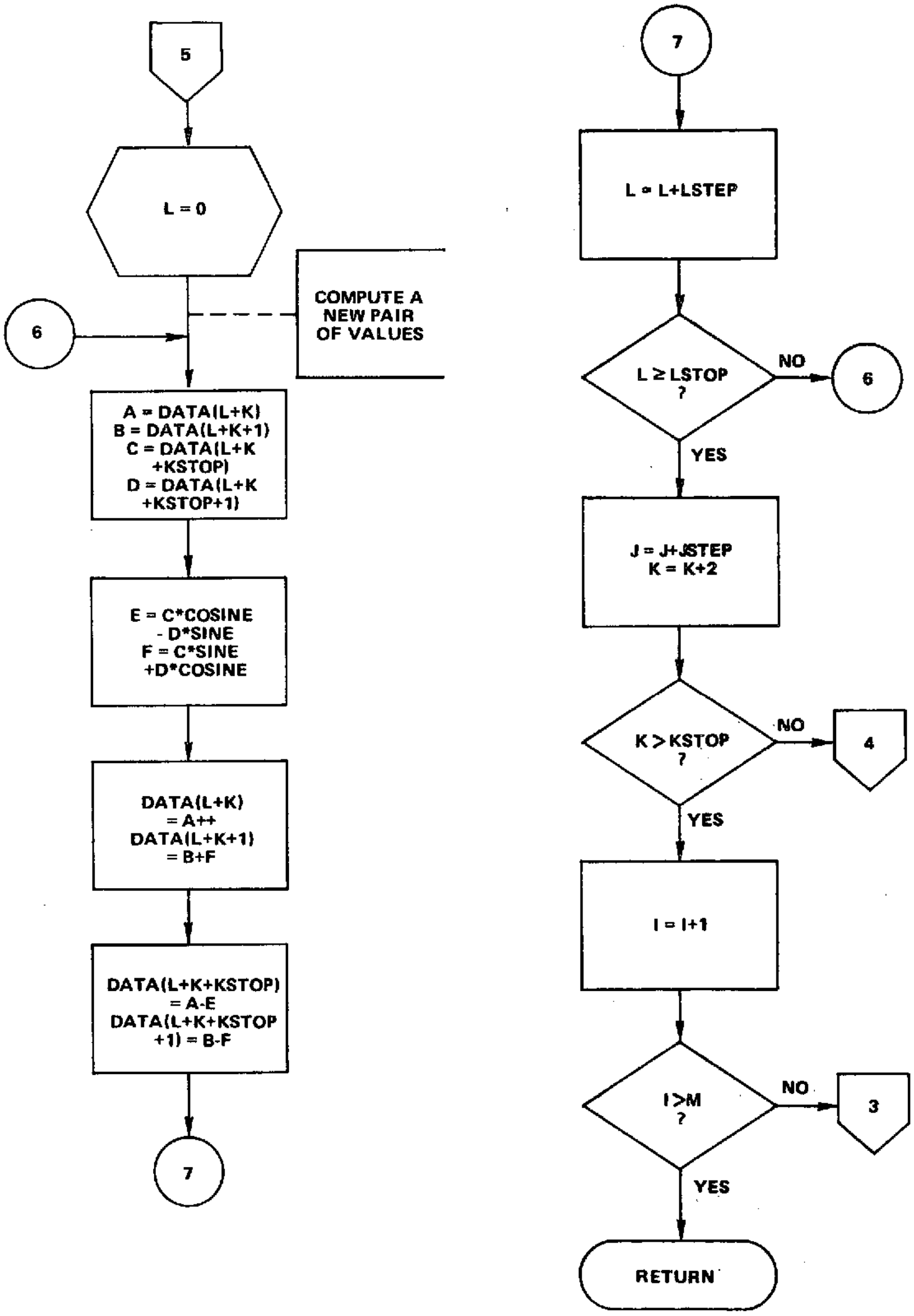


Figure B-10. Subroutine FFTONE Flowchart (3 of 3)

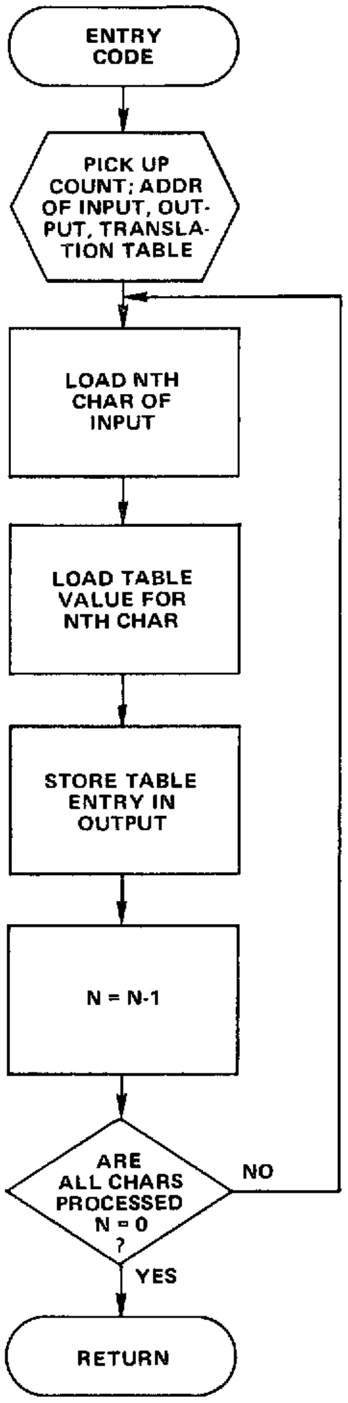

ENTRY CODE
PICK UP COUNT; ADDR OF INPUT, OUTPUT, TRANSLATION TABLE
LOAD NTH CHAR OF INPUT
LOAD TABLE VALUE FOR NTH CHAR
STORE TABLE ENTRY IN OUTPUT
N = N-1
ARE ALL CHARS PROCESSED N = 0 ?
NO
YES
RETURN

Figure B-11. Subroutine CODE Flowchart

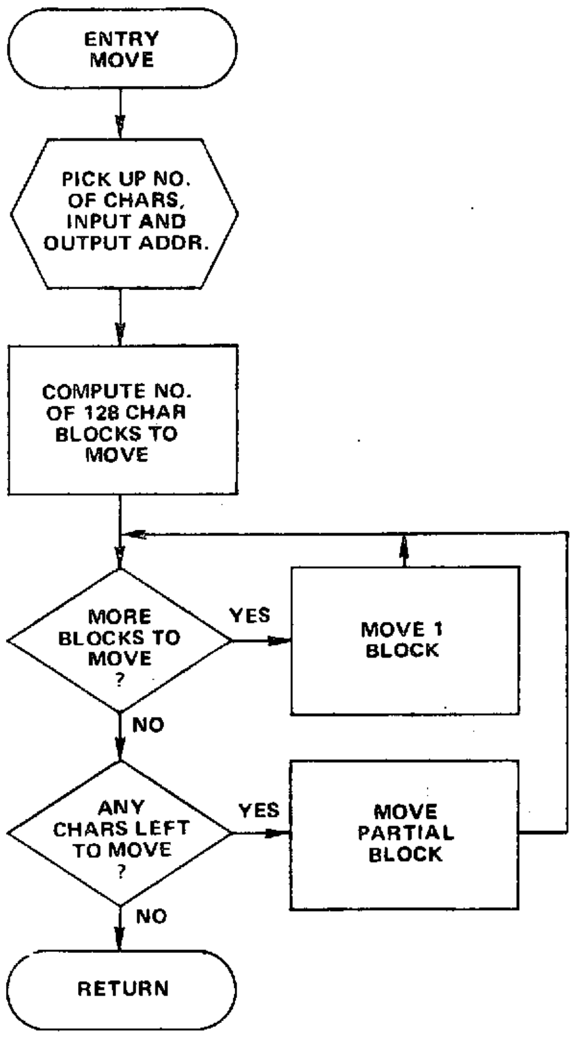
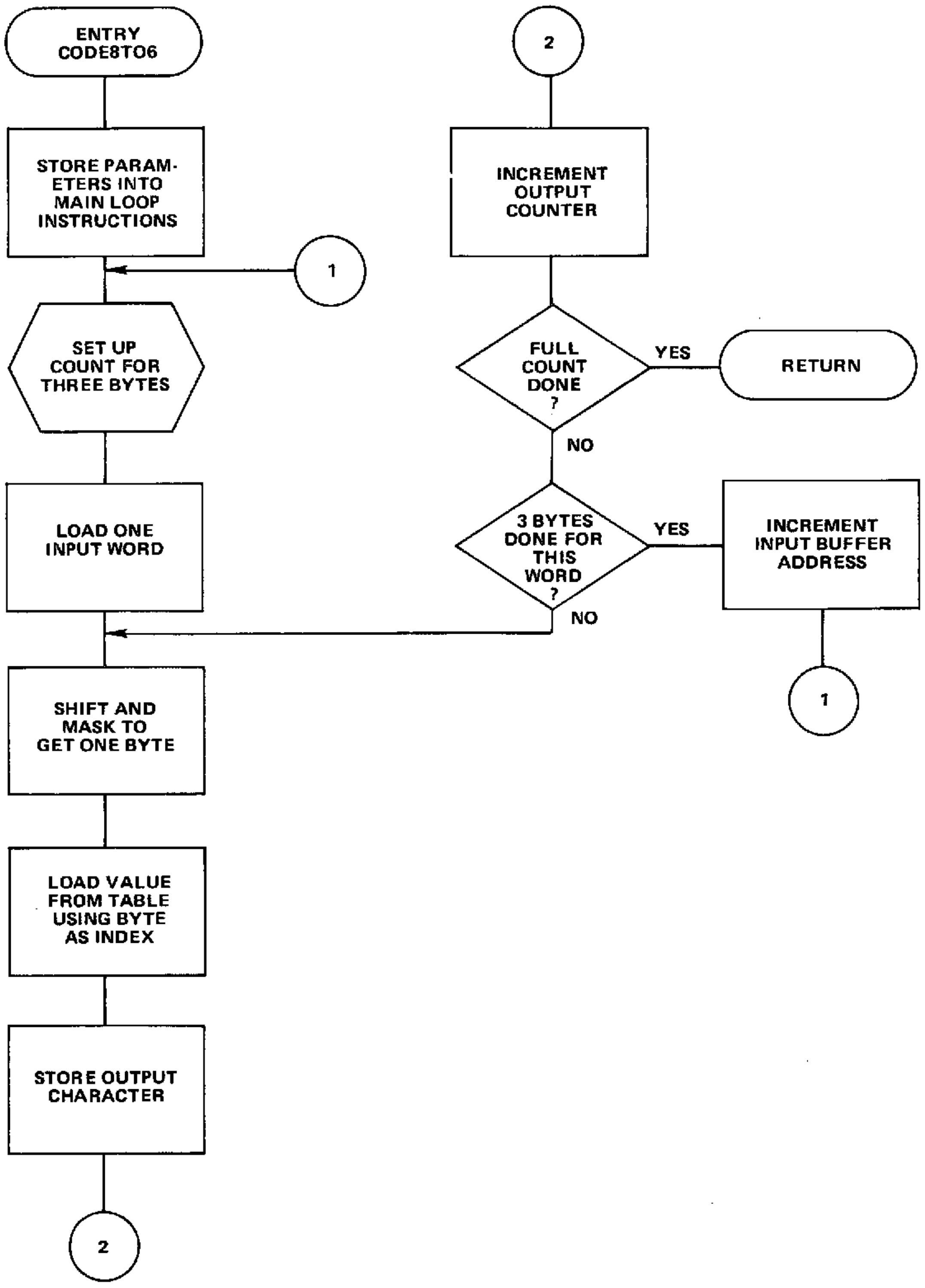


Figure B-12. Subroutine MOVE Flowchart



ENTRY CODE8TO6
STORE PARAMETERS INTO MAIN LOOP INSTRUCTIONS
1
SET UP COUNT FOR THREE BYTES
LOAD ONE INPUT WORD
SHIFT AND MASK TO GET ONE BYTE
LOAD VALUE FROM TABLE USING BYTE AS INDEX
STORE OUTPUT CHARACTER
2
2
INCREMENT OUTPUT COUNTER
FULL COUNT DONE ?
YES
RETURN
NO
3 BYTES DONE FOR THIS WORD ?
YES
INCREMENT INPUT BUFFER ADDRESS
NO
1

Figure B-13. Subroutine CODE8T06 Flowchart

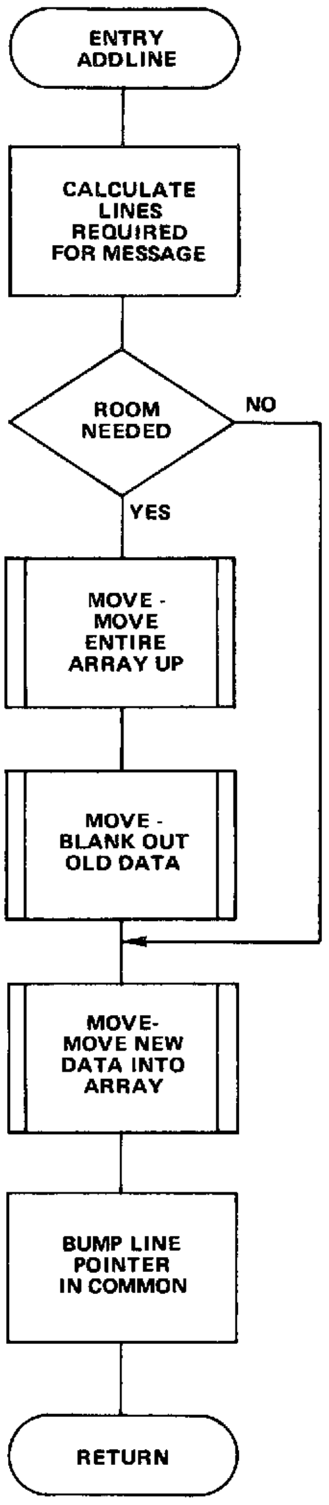

ENTRY ADDLINE
CALCULATE LINES REQUIRED FOR MESSAGE
ROOM NEEDED
NO
YES
MOVE - MOVE ENTIRE ARRAY UP
MOVE - BLANK OUT OLD DATA
MOVE- MOVE NEW DATA INTO ARRAY
BUMP LINE POINTER IN COMMON
RETURN

Figure B-14. Subroutine ADDLINE Flowchart

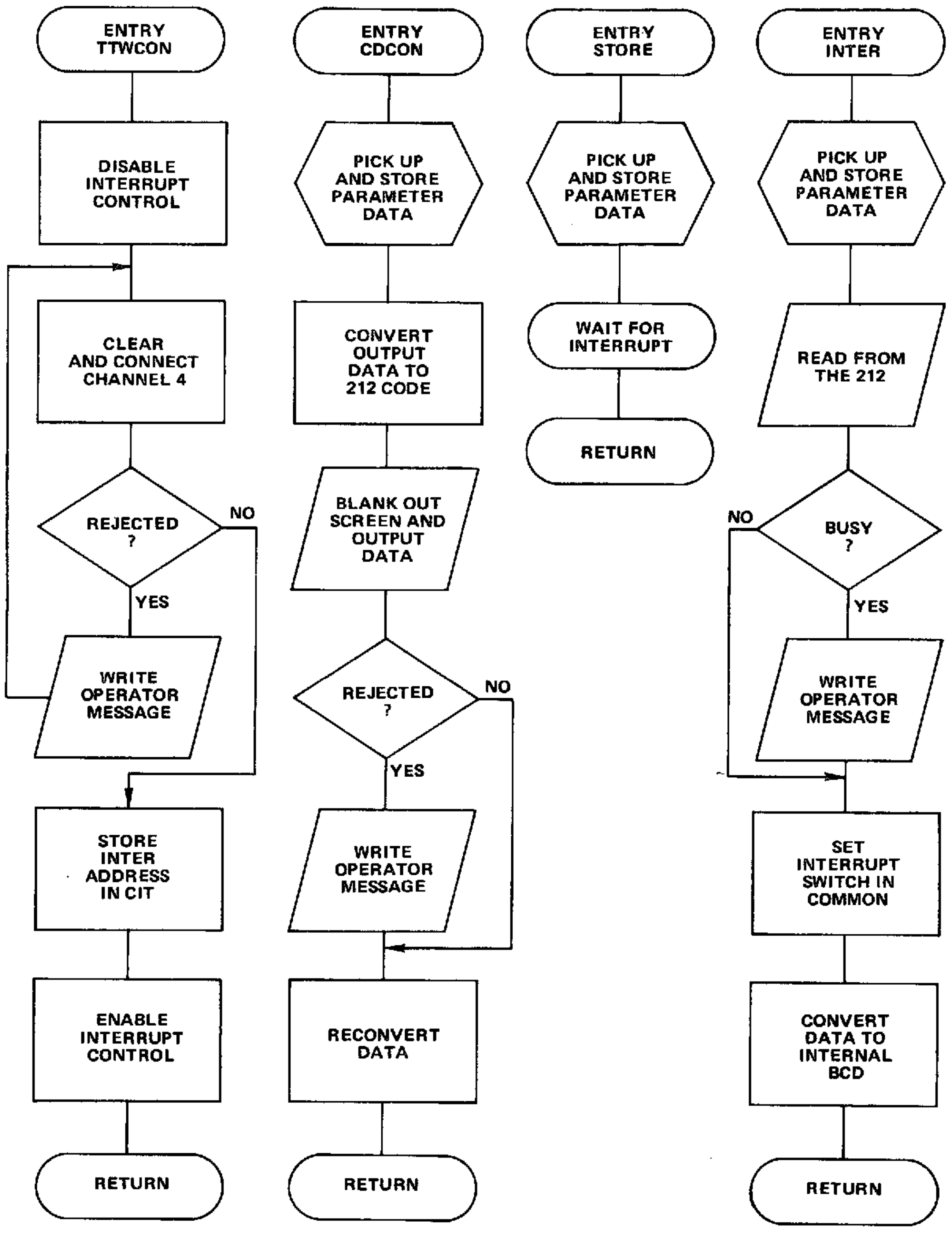
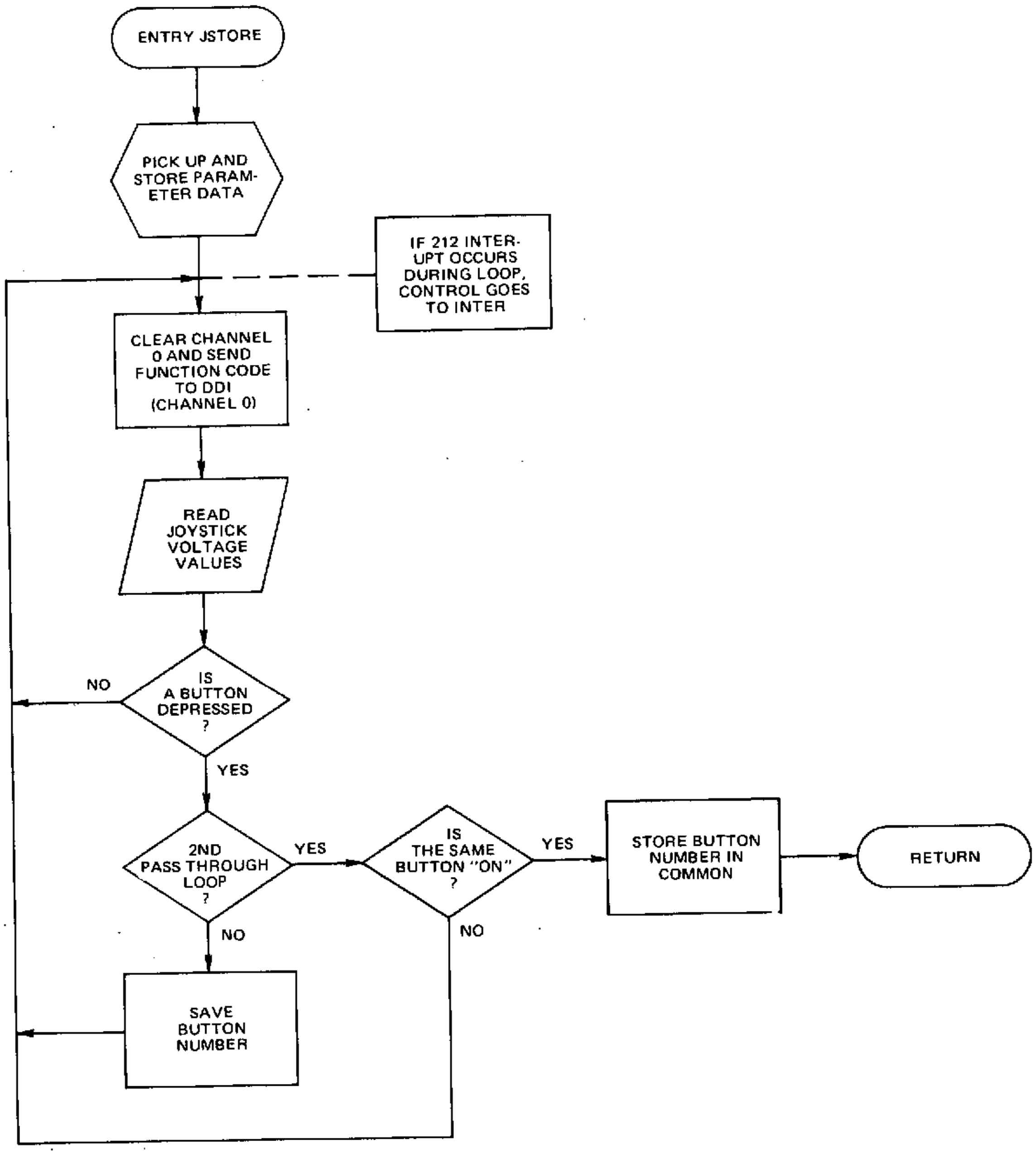

ENTRY TTWCON
DISABLE INTERRUPT CONTROL
CLEAR AND CONNECT CHANNEL 4
REJECTED ?
NO
YES
WRITE OPERATOR MESSAGE
STORE INTER ADDRESS IN CIT
ENABLE INTERRUPT CONTROL
RETURN
ENTRY CDCON
PICK UP AND STORE PARAMETER DATA
CONVERT OUTPUT DATA TO 212 CODE
BLANK OUT SCREEN AND OUTPUT DATA
REJECTED ?
NO
YES
WRITE OPERATOR MESSAGE
RECONVERT DATA
RETURN
ENTRY STORE
PICK UP AND STORE PARAMETER DATA
WAIT FOR INTERRUPT
RETURN
ENTRY INTER
PICK UP AND STORE PARAMETER DATA
READ FROM THE 212
BUSY ?
NO
YES
WRITE OPERATOR MESSAGE
SET INTERRUPT SWITCH IN COMMON
CONVERT DATA TO INTERNAL BCD
RETURN

Figure B-15. Subroutine TTWLVE Flowchart


ENTRY JSTORE
PICK UP AND STORE PARAMETER DATA
IF 212 INTERUPT OCCURS DURING LOOP, CONTROL GOES TO INTER
CLEAR CHANNEL 0 AND SEND FUNCTION CODE TO DDI (CHANNEL 0)
READ JOYSTICK VOLTAGE VALUES
IS A BUTTON DEPRESSED ?
NO
YES
2ND PASS THROUGH LOOP ?
YES
NO
IS THE SAME BUTTON "ON" ?
YES
NO
STORE BUTTON NUMBER IN COMMON
RETURN
SAVE BUTTON NUMBER

Note: JSTORE is present only in the TTWLVE program version which resides in the JOYSTICK overlay. It replaces the entry STORE.

Figure B-15. Subroutine TTWLVE Flowchart (2 of 2)

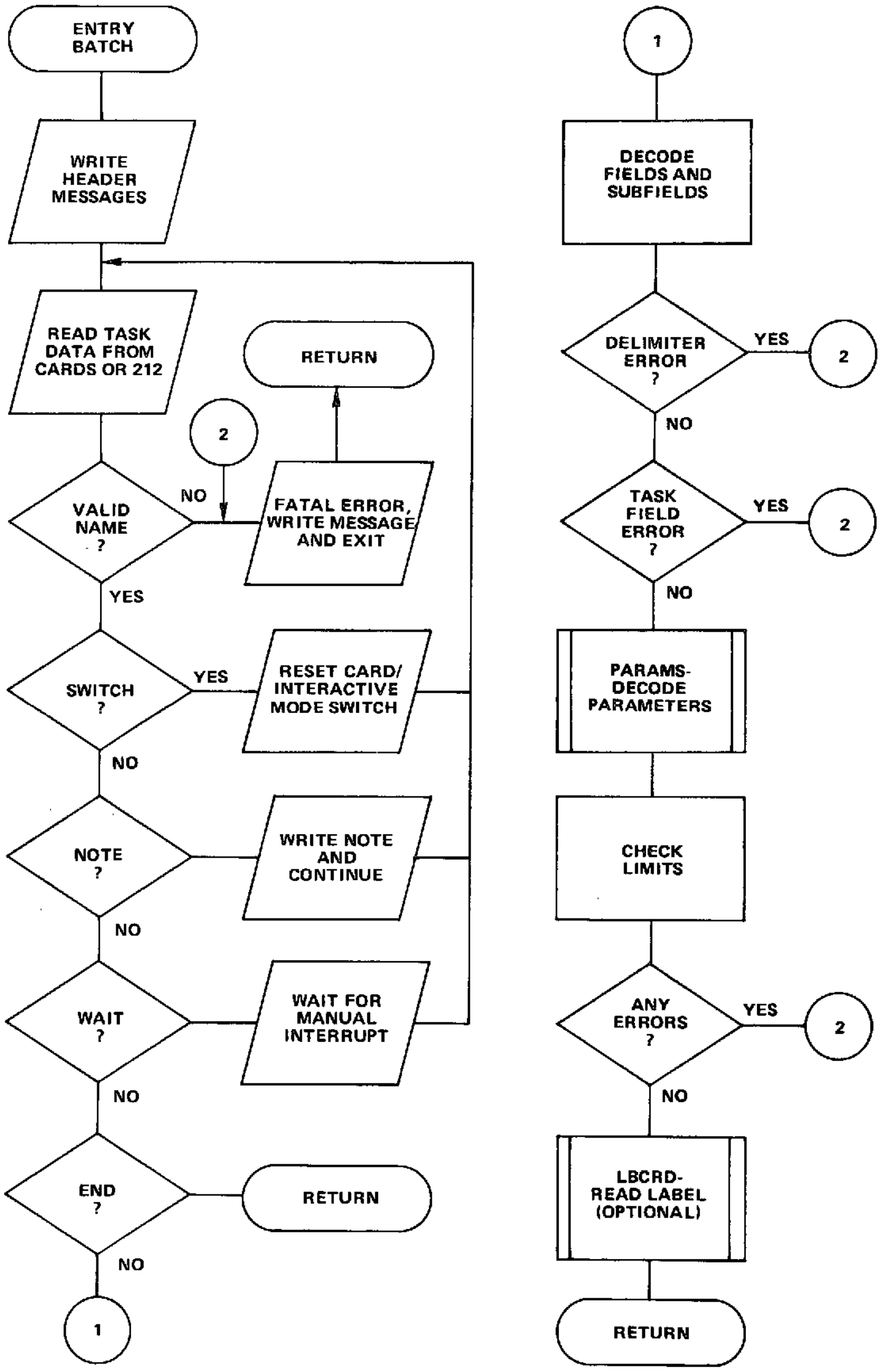


Figure C-1. BATCH Program Flowchart

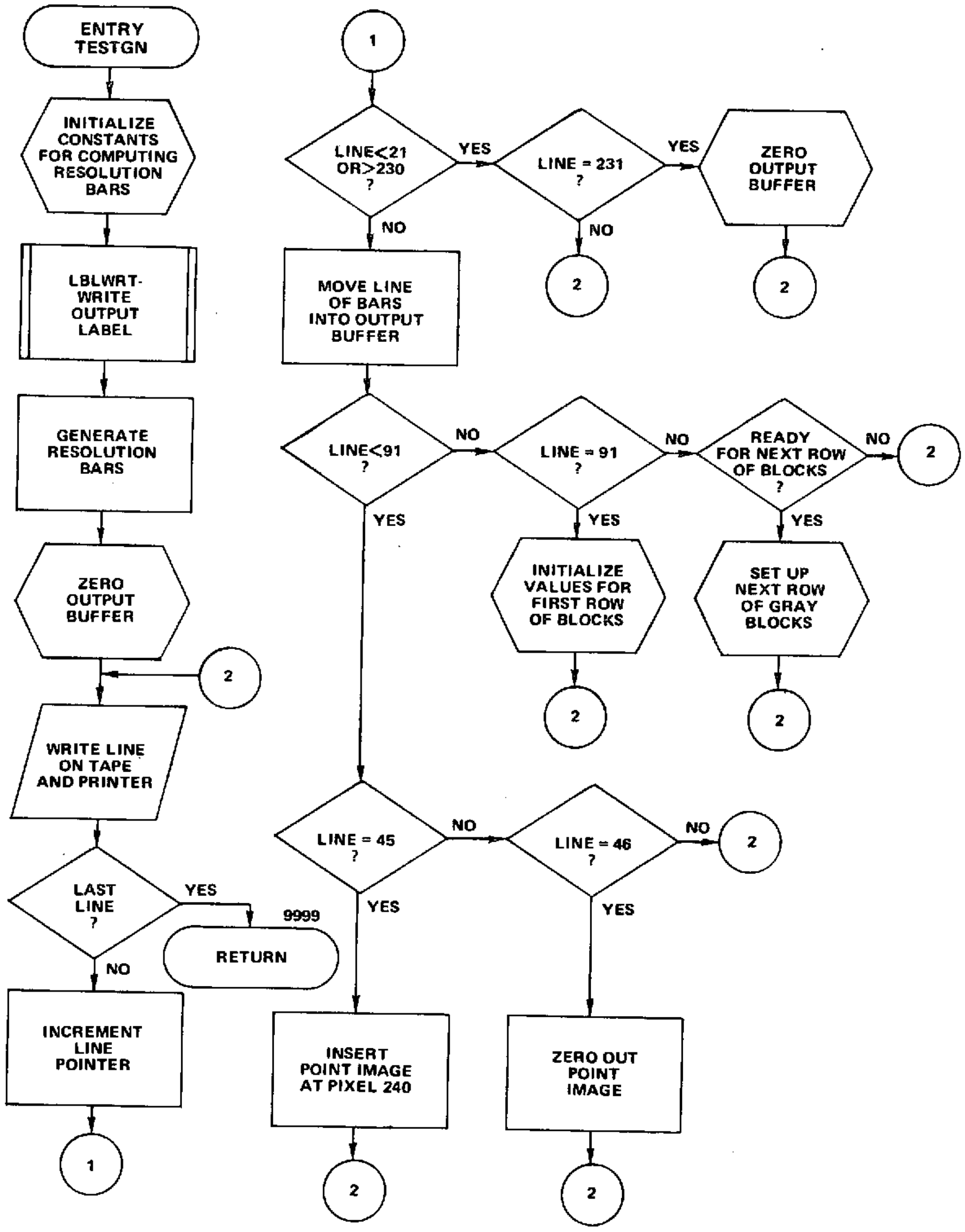

ENTRY TESTGN
INITIALIZE CONSTANTS FOR COMPUTING RESOLUTION BARS
LBLWRT- WRITE OUTPUT LABEL
GENERATE RESOLUTION BARS
ZERO OUTPUT BUFFER
2
WRITE LINE ON TAPE AND PRINTER
LAST LINE ?
YES
9999
RETURN
NO
INCREMENT LINE POINTER
1
1
LINE<21 OR>230 ?
YES
LINE = 231 ?
YES
ZERO OUTPUT BUFFER
2
NO
2
NO
MOVE LINE OF BARS INTO OUTPUT BUFFER
LINE<91 ?
NO
LINE = 91 ?
NO
READY FOR NEXT ROW OF BLOCKS ?
NO
2
YES
INITIALIZE VALUES FOR FIRST ROW OF BLOCKS
2
YES
SET UP NEXT ROW OF GRAY BLOCKS
2
YES
LINE = 45 ?
NO
LINE = 46 ?
NO
2
YES
INSERT POINT IMAGE AT PIXEL 240
2
YES
ZERO OUT POINT IMAGE
2

Figure C-2. TESTGN Program Flowchart

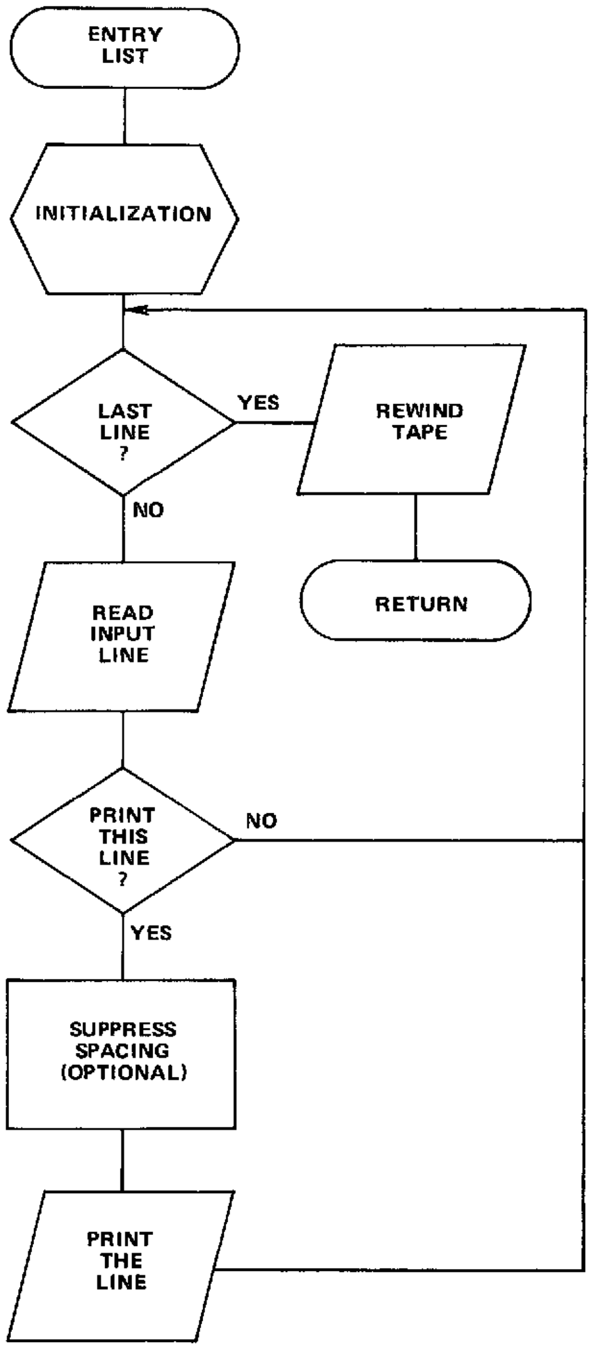


Figure C-3. LIST Program Flowchart

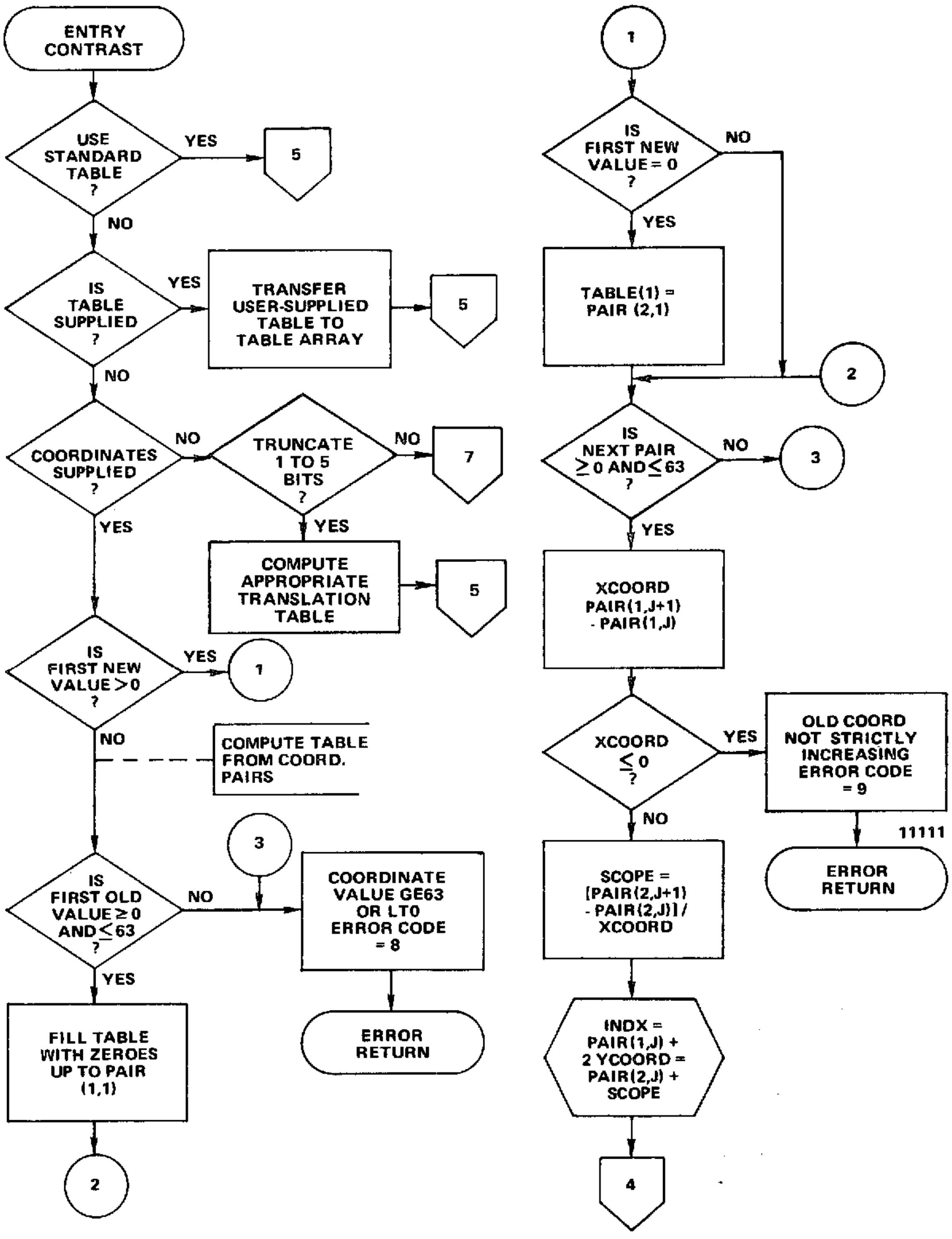

ENTRY CONTRAST
USE STANDARD TABLE ?
YES
5
NO
IS TABLE SUPPLIED ?
YES
TRANSFER USER-SUPPLIED TABLE TO TABLE ARRAY
5
NO
COORDINATES SUPPLIED ?
NO
TRUNCATE 1 TO 5 BITS ?
NO
7
YES
COMPUTE APPROPRIATE TRANSLATION TABLE
5
YES
IS FIRST NEW VALUE > 0 ?
YES
1
NO
COMPUTE TABLE FROM COORD. PAIRS
3
IS FIRST OLD VALUE ≥ 0 AND ≤ 63 ?
NO
COORDINATE VALUE GE63 OR LT0 ERROR CODE = 8
ERROR RETURN
YES
FILL TABLE WITH ZEROES UP TO PAIR (1,1)
2
1
IS FIRST NEW VALUE = 0 ?
NO
YES
TABLE(1) = PAIR (2,1)
2
IS NEXT PAIR ≥ 0 AND ≤ 63 ?
NO
3
YES
XCOORD PAIR(1,J+1) - PAIR(1,J)
XCOORD ≤ 0 ?
YES
OLD COORD NOT STRICTLY INCREASING ERROR CODE = 9
11111
ERROR RETURN
NO
SCOPE = [PAIR(2,J+1) - PAIR(2,J)] / XCOORD
INDX = PAIR(1,J) + 2 YCOORD = PAIR(2,J) + SCOPE
4

Figure C-4. CONTRAST Program Flowchart (1 of 2)

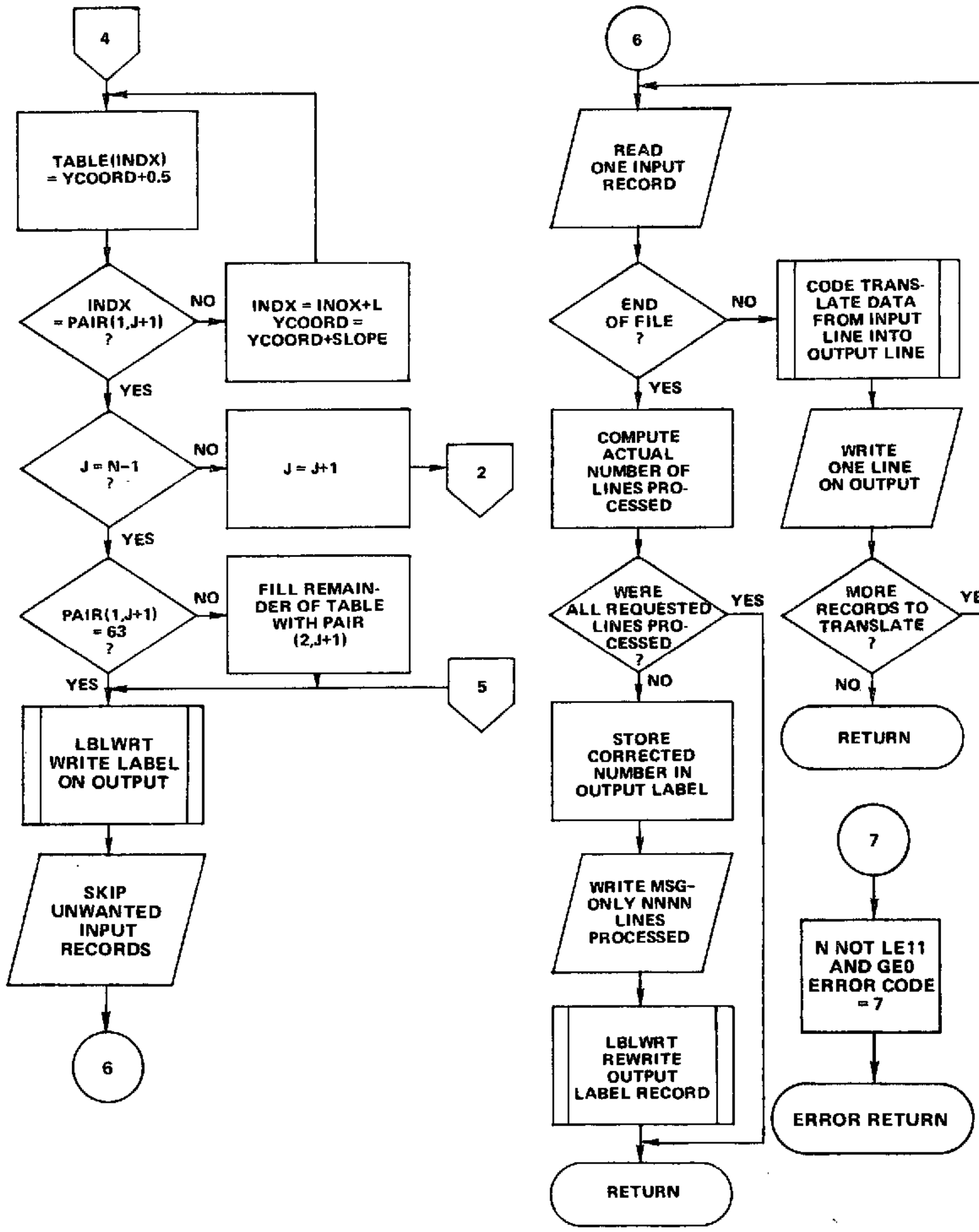
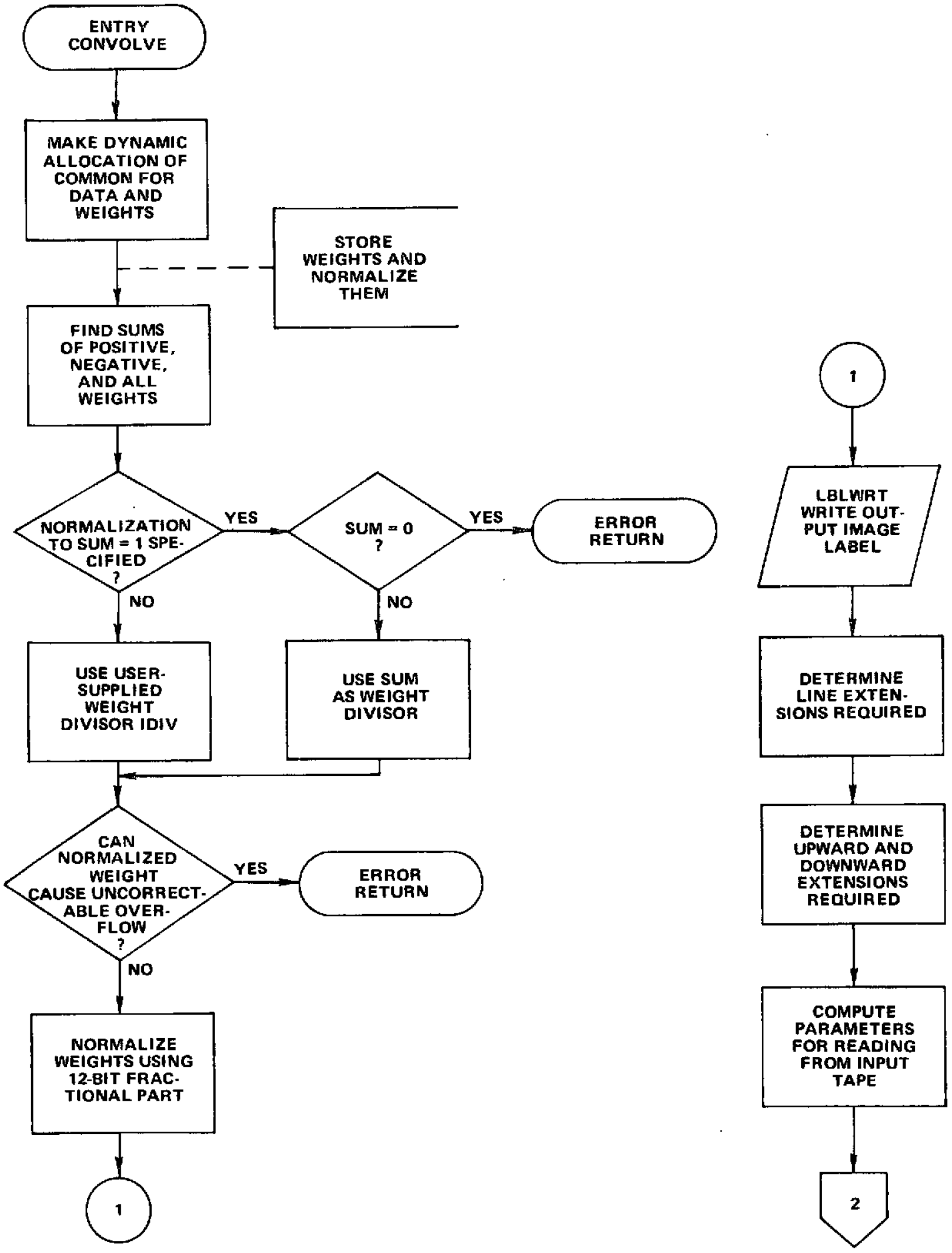

4
TABLE(INDX) = YCOORD+0.5
INDX = PAIR(1,J+1) ?
NO
INDX = INOX+L YCOORD = YCOORD+SLOPE
YES
J = N-1 ?
NO
J = J+1
2
YES
PAIR(1,J+1) = 63 ?
NO
FILL REMAINDER OF TABLE WITH PAIR (2,J+1)
YES
5
LBLWRT WRITE LABEL ON OUTPUT
SKIP UNWANTED INPUT RECORDS
6
6
READ ONE INPUT RECORD
END OF FILE ?
NO
CODE TRANSLATE DATA FROM INPUT LINE INTO OUTPUT LINE
YES
COMPUTE ACTUAL NUMBER OF LINES PROCESSED
WRITE ONE LINE ON OUTPUT
WERE ALL REQUESTED LINES PROCESSED ?
YES
MORE RECORDS TO TRANSLATE ?
YES
NO
NO
STORE CORRECTED NUMBER IN OUTPUT LABEL
RETURN
7
WRITE MSG-ONLY NNNN LINES PROCESSED
N NOT LE11 AND GE0 ERROR CODE = 7
LBLWRT REWRITE OUTPUT LABEL RECORD
ERROR RETURN
RETURN

Figure C-4. CONTRAST Program Flowchart (2 of 2)



ENTRY CONVOLVE
MAKE DYNAMIC ALLOCATION OF COMMON FOR DATA AND WEIGHTS
STORE WEIGHTS AND NORMALIZE THEM
FIND SUMS OF POSITIVE, NEGATIVE, AND ALL WEIGHTS
NORMALIZATION TO SUM = 1 SPECIFIED ?
YES
SUM = 0 ?
YES
ERROR RETURN
NO
NO
USE USER-SUPPLIED WEIGHT DIVISOR IDIV
USE SUM AS WEIGHT DIVISOR
CAN NORMALIZED WEIGHT CAUSE UNCORRECTABLE OVERFLOW ?
YES
ERROR RETURN
NO
NORMALIZE WEIGHTS USING 12-BIT FRACTIONAL PART
1
1
LBLWRT WRITE OUTPUT IMAGE LABEL
DETERMINE LINE EXTENSIONS REQUIRED
DETERMINE UPWARD AND DOWNWARD EXTENSIONS REQUIRED
COMPUTE PARAMETERS FOR READING FROM INPUT TAPE
2

Figure C-5. CONVOLVE Program Flowchart (1 of 5)

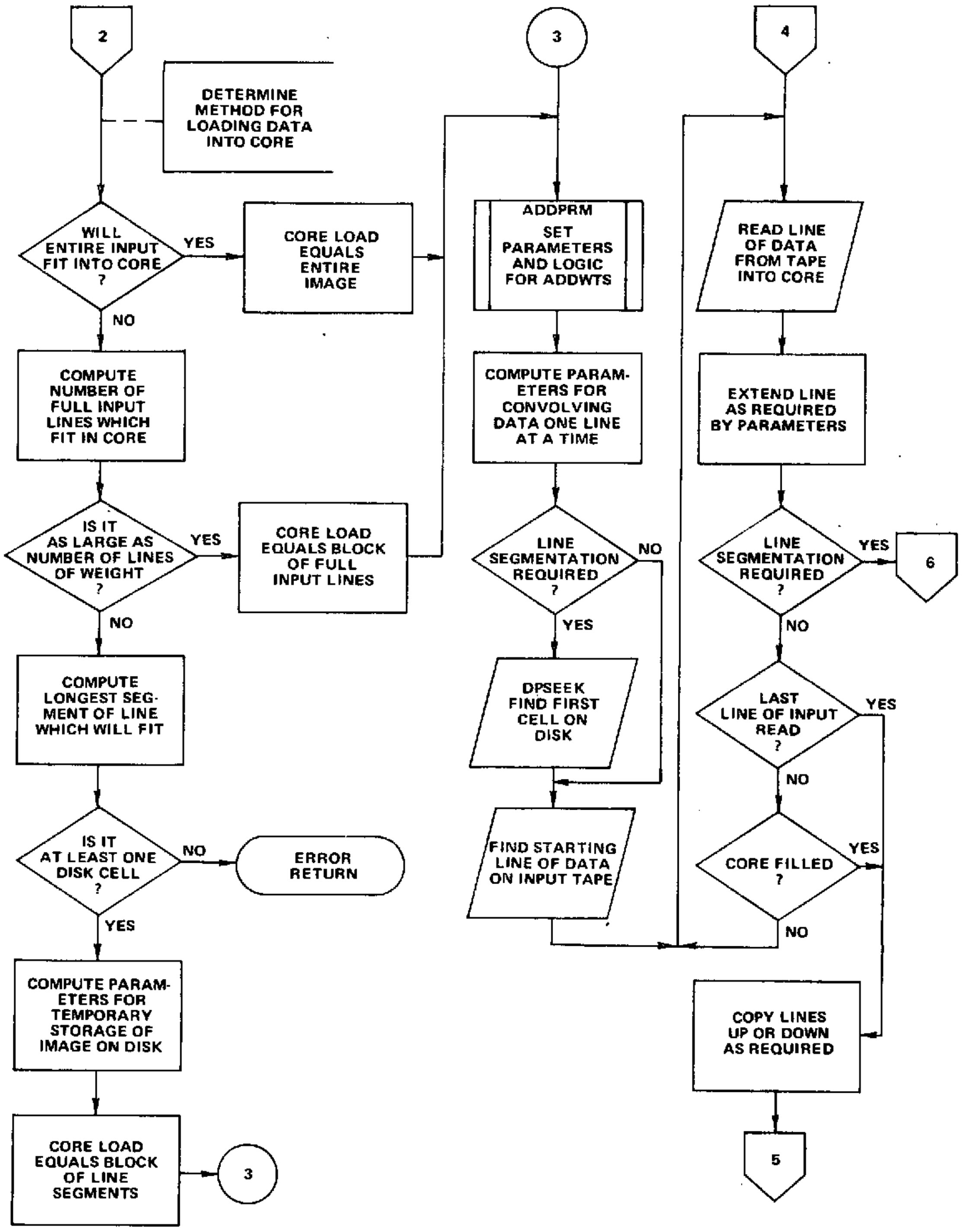

2
DETERMINE METHOD FOR LOADING DATA INTO CORE
WILL ENTIRE INPUT FIT INTO CORE ?
YES
CORE LOAD EQUALS ENTIRE IMAGE
NO
COMPUTE NUMBER OF FULL INPUT LINES WHICH FIT IN CORE
IS IT AS LARGE AS NUMBER OF LINES OF WEIGHT ?
YES
CORE LOAD EQUALS BLOCK OF FULL INPUT LINES
NO
COMPUTE LONGEST SEGMENT OF LINE WHICH WILL FIT
IS IT AT LEAST ONE DISK CELL ?
NO
ERROR RETURN
YES
COMPUTE PARAMETERS FOR TEMPORARY STORAGE OF IMAGE ON DISK
CORE LOAD EQUALS BLOCK OF LINE SEGMENTS
3
3
ADDPRM SET PARAMETERS AND LOGIC FOR ADDWTS
COMPUTE PARAMETERS FOR CONVOLVING DATA ONE LINE AT A TIME
LINE SEGMENTATION REQUIRED ?
NO
YES
DPSEEK FIND FIRST CELL ON DISK
FIND STARTING LINE OF DATA ON INPUT TAPE
4
READ LINE OF DATA FROM TAPE INTO CORE
EXTEND LINE AS REQUIRED BY PARAMETERS
LINE SEGMENTATION REQUIRED ?
YES
6
NO
LAST LINE OF INPUT READ ?
YES
NO
CORE FILLED ?
YES
NO
COPY LINES UP OR DOWN AS REQUIRED
5

Figure C-5. CONVOLVE Program Flowchart (2 of 5)

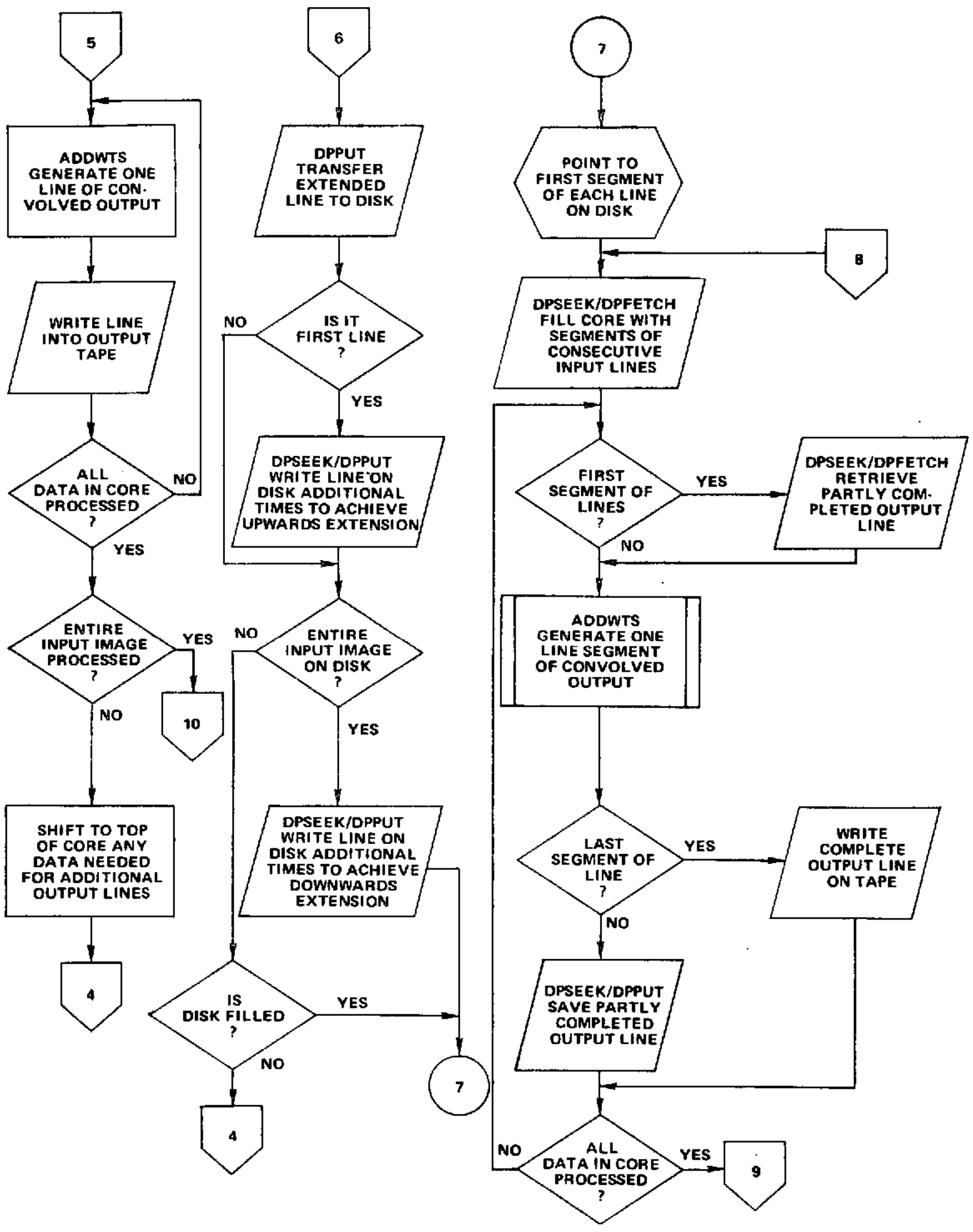
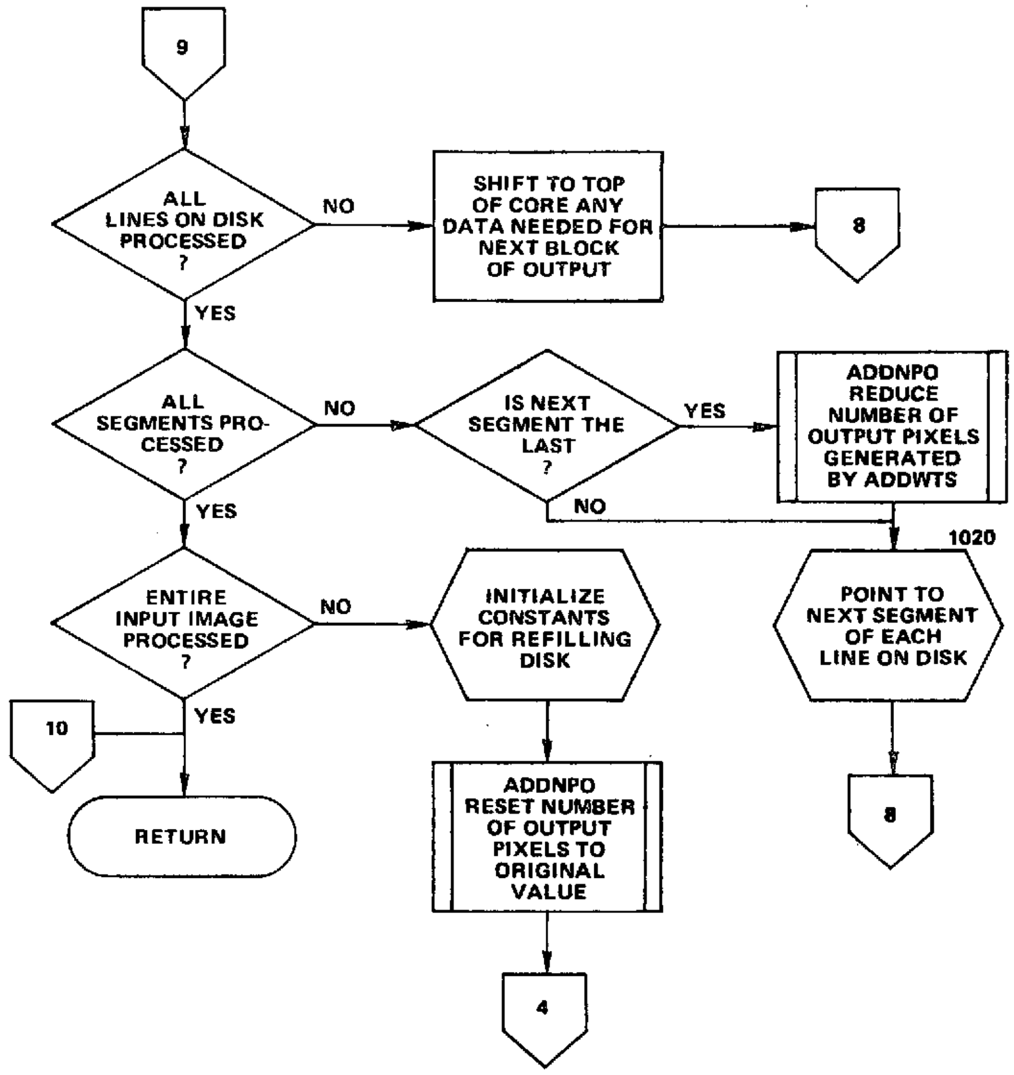
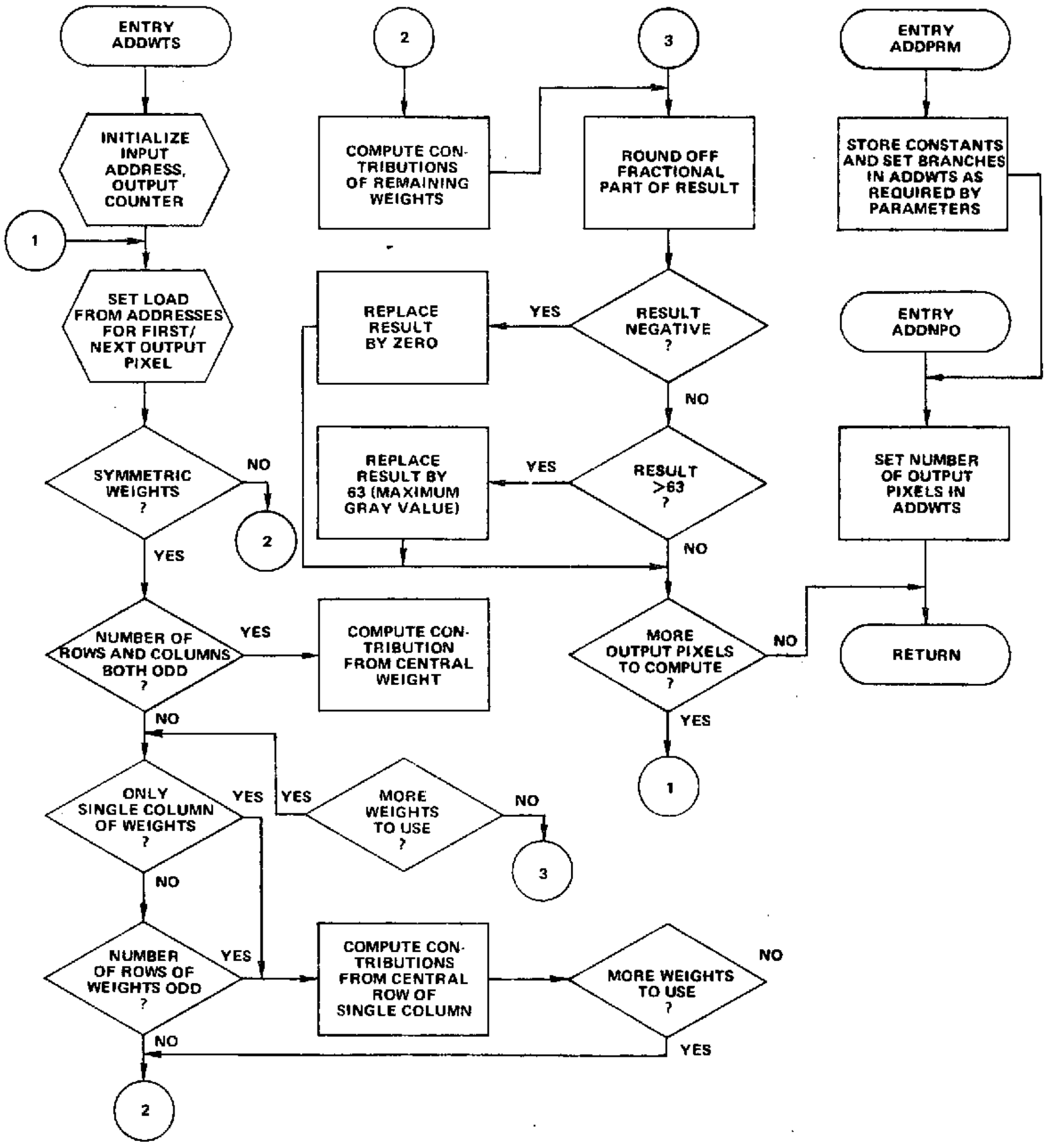

5
ADDWTS GENERATE ONE LINE OF CON-VOLVED OUTPUT
WRITE LINE INTO OUTPUT TAPE
ALL DATA IN CORE PROCESSED ?
NO
YES
ENTIRE INPUT IMAGE PROCESSED ?
YES
10
NO
SHIFT TO TOP OF CORE ANY DATA NEEDED FOR ADDITIONAL OUTPUT LINES
4
6
DPPUT TRANSFER EXTENDED LINE TO DISK
IS IT FIRST LINE ?
NO
YES
DPSEEK/DPPUT WRITE LINE ON DISK ADDITIONAL TIMES TO ACHIEVE UPWARDS EXTENSION
ENTIRE INPUT IMAGE ON DISK ?
NO
YES
DPSEEK/DPPUT WRITE LINE ON DISK ADDITIONAL TIMES TO ACHIEVE DOWNWARDS EXTENSION
IS DISK FILLED ?
YES
NO
4
7
7
POINT TO FIRST SEGMENT OF EACH LINE ON DISK
8
DPSEEK/DPFETCH FILL CORE WITH SEGMENTS OF CONSECUTIVE INPUT LINES
FIRST SEGMENT OF LINES ?
YES
NO
DPSEEK/DPFETCH RETRIEVE PARTLY COM-PLETED OUTPUT LINE
ADDWTS GENERATE ONE LINE SEGMENT OF CONVOLVED OUTPUT
LAST SEGMENT OF LINE ?
YES
NO
WRITE COMPLETE OUTPUT LINE ON TAPE
DPSEEK/DPPUT SAVE PARTLY COMPLETED OUTPUT LINE
ALL DATA IN CORE PROCESSED ?
NO
YES
9

Figure C-5. CONVOLVE Program Flowchart (3 of 5)



9
ALL LINES ON DISK PROCESSED ?
NO
SHIFT TO TOP OF CORE ANY DATA NEEDED FOR NEXT BLOCK OF OUTPUT
8
YES
ALL SEGMENTS PROCESSED ?
NO
IS NEXT SEGMENT THE LAST ?
YES
ADDNPO REDUCE NUMBER OF OUTPUT PIXELS GENERATED BY ADDWTS
NO
YES
1020
ENTIRE INPUT IMAGE PROCESSED ?
NO
INITIALIZE CONSTANTS FOR REFILLING DISK
POINT TO NEXT SEGMENT OF EACH LINE ON DISK
YES
10
RETURN
ADDNPO RESET NUMBER OF OUTPUT PIXELS TO ORIGINAL VALUE
8
4

Figure C-5. CONVOLVE Program Flowchart (4 of 5)



ENTRY ADDWTS
INITIALIZE INPUT ADDRESS, OUTPUT COUNTER
1
SET LOAD FROM ADDRESSES FOR FIRST/ NEXT OUTPUT PIXEL
SYMMETRIC WEIGHTS ?
NO
2
YES
NUMBER OF ROWS AND COLUMNS BOTH ODD ?
YES
COMPUTE CON-TRIBUTION FROM CENTRAL WEIGHT
NO
ONLY SINGLE COLUMN OF WEIGHTS ?
YES
YES
MORE WEIGHTS TO USE ?
NO
3
NO
NUMBER OF ROWS OF WEIGHTS ODD ?
YES
COMPUTE CON-TRIBUTIONS FROM CENTRAL ROW OF SINGLE COLUMN
MORE WEIGHTS TO USE ?
NO
YES
NO
2
2
COMPUTE CON-TRIBUTIONS OF REMAINING WEIGHTS
3
ROUND OFF FRACTIONAL PART OF RESULT
RESULT NEGATIVE ?
YES
REPLACE RESULT BY ZERO
NO
RESULT >63 ?
YES
REPLACE RESULT BY 63 (MAXIMUM GRAY VALUE)
NO
MORE OUTPUT PIXELS TO COMPUTE ?
NO
YES
1
ENTRY ADDPRM
STORE CONSTANTS AND SET BRANCHES IN ADDWTS AS REQUIRED BY PARAMETERS
ENTRY ADDNPO
SET NUMBER OF OUTPUT PIXELS IN ADDWTS
RETURN

Figure C-5. CONVOLVE Program Flowchart (5 of 5)

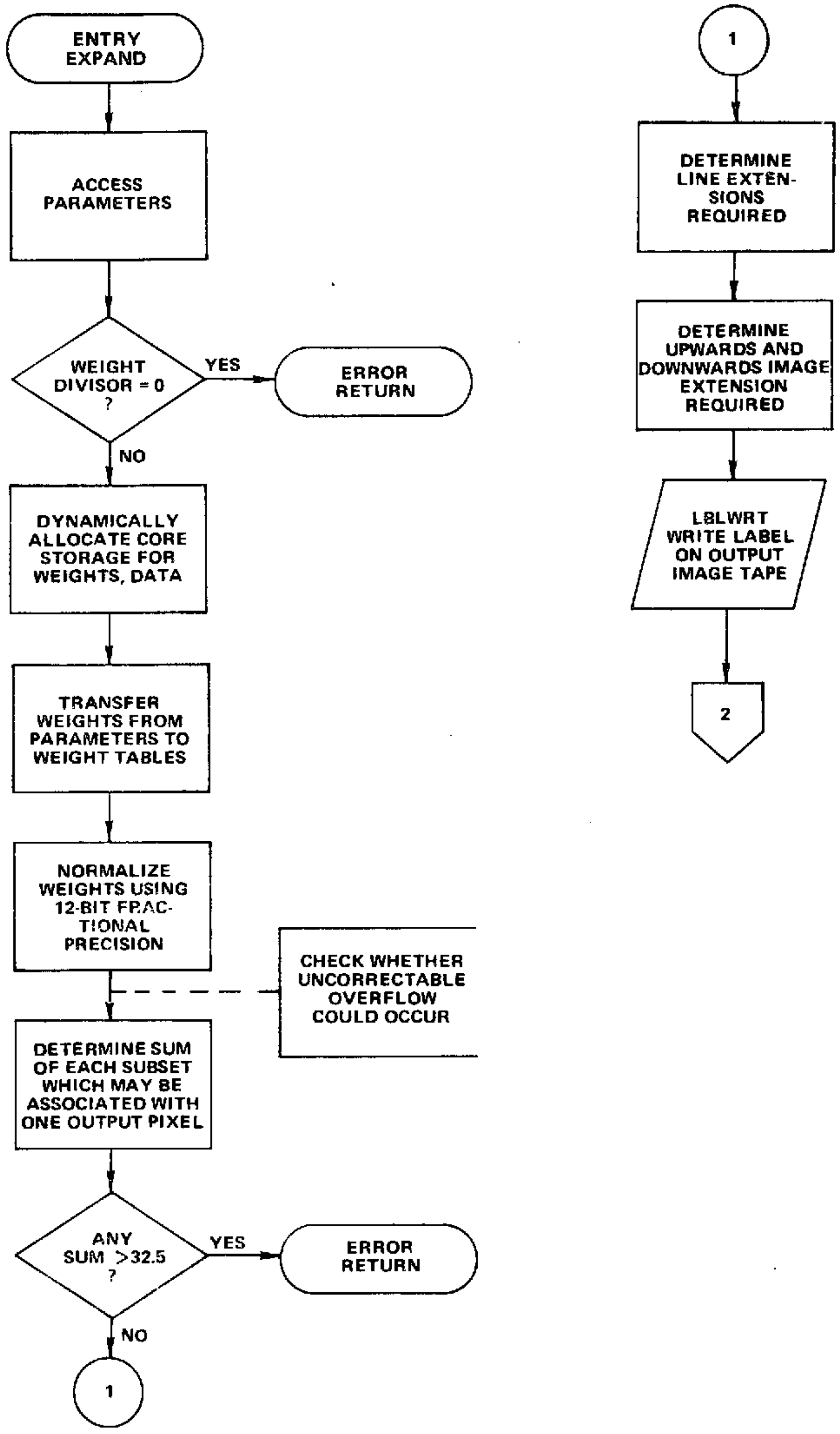

ENTRY EXPAND
ACCESS PARAMETERS
WEIGHT DIVISOR = 0 ?
YES
ERROR RETURN
NO
DYNAMICALLY ALLOCATE CORE STORAGE FOR WEIGHTS, DATA
TRANSFER WEIGHTS FROM PARAMETERS TO WEIGHT TABLES
NORMALIZE WEIGHTS USING 12-BIT FRAC-TIONAL PRECISION
CHECK WHETHER UNCORRECTABLE OVERFLOW COULD OCCUR
DETERMINE SUM OF EACH SUBSET WHICH MAY BE ASSOCIATED WITH ONE OUTPUT PIXEL
ANY SUM >32.5 ?
YES
ERROR RETURN
NO
1
1
DETERMINE LINE EXTEN-SIONS REQUIRED
DETERMINE UPWARDS AND DOWNWARDS IMAGE EXTENSION REQUIRED
LBLWRT WRITE LABEL ON OUTPUT IMAGE TAPE
2

Figure C-6. EXPAND Program Flowchart (1 of 4)

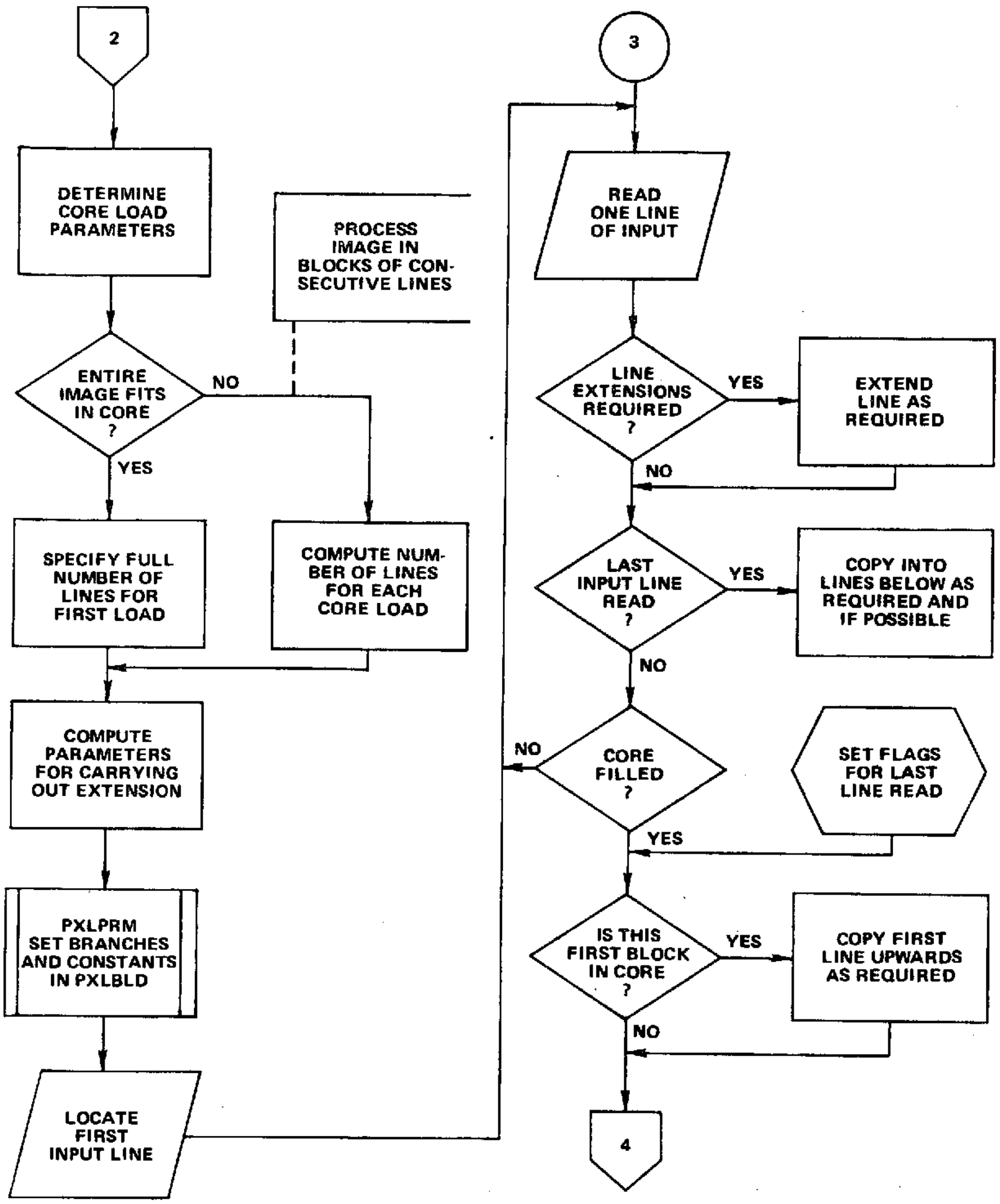

2
DETERMINE CORE LOAD PARAMETERS
PROCESS IMAGE IN BLOCKS OF CONSECUTIVE LINES
ENTIRE IMAGE FITS IN CORE ?
NO
YES
SPECIFY FULL NUMBER OF LINES FOR FIRST LOAD
COMPUTE NUMBER OF LINES FOR EACH CORE LOAD
COMPUTE PARAMETERS FOR CARRYING OUT EXTENSION
PXLPRM SET BRANCHES AND CONSTANTS IN PXLBLD
LOCATE FIRST INPUT LINE
3
READ ONE LINE OF INPUT
LINE EXTENSIONS REQUIRED ?
YES
EXTEND LINE AS REQUIRED
NO
LAST INPUT LINE READ ?
YES
COPY INTO LINES BELOW AS REQUIRED AND IF POSSIBLE
NO
CORE FILLED ?
NO
SET FLAGS FOR LAST LINE READ
YES
IS THIS FIRST BLOCK IN CORE ?
YES
COPY FIRST LINE UPWARDS AS REQUIRED
NO
4

Figure C-6. EXPAND Program Flowchart (2 of 4)

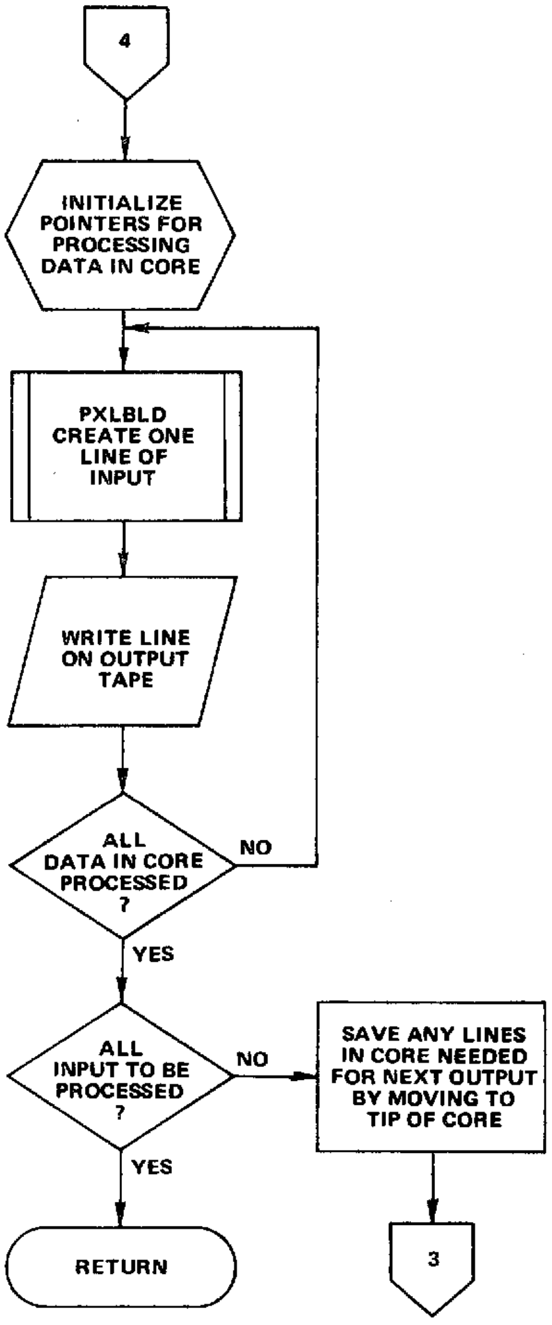

4
INITIALIZE POINTERS FOR PROCESSING DATA IN CORE
PXLBLD CREATE ONE LINE OF INPUT
WRITE LINE ON OUTPUT TAPE
ALL DATA IN CORE PROCESSED ?
NO
YES
ALL INPUT TO BE PROCESSED ?
NO
SAVE ANY LINES IN CORE NEEDED FOR NEXT OUTPUT BY MOVING TO TIP OF CORE
YES
RETURN
3

Figure C-6. EXPAND Program Flowchart (3 of 4)

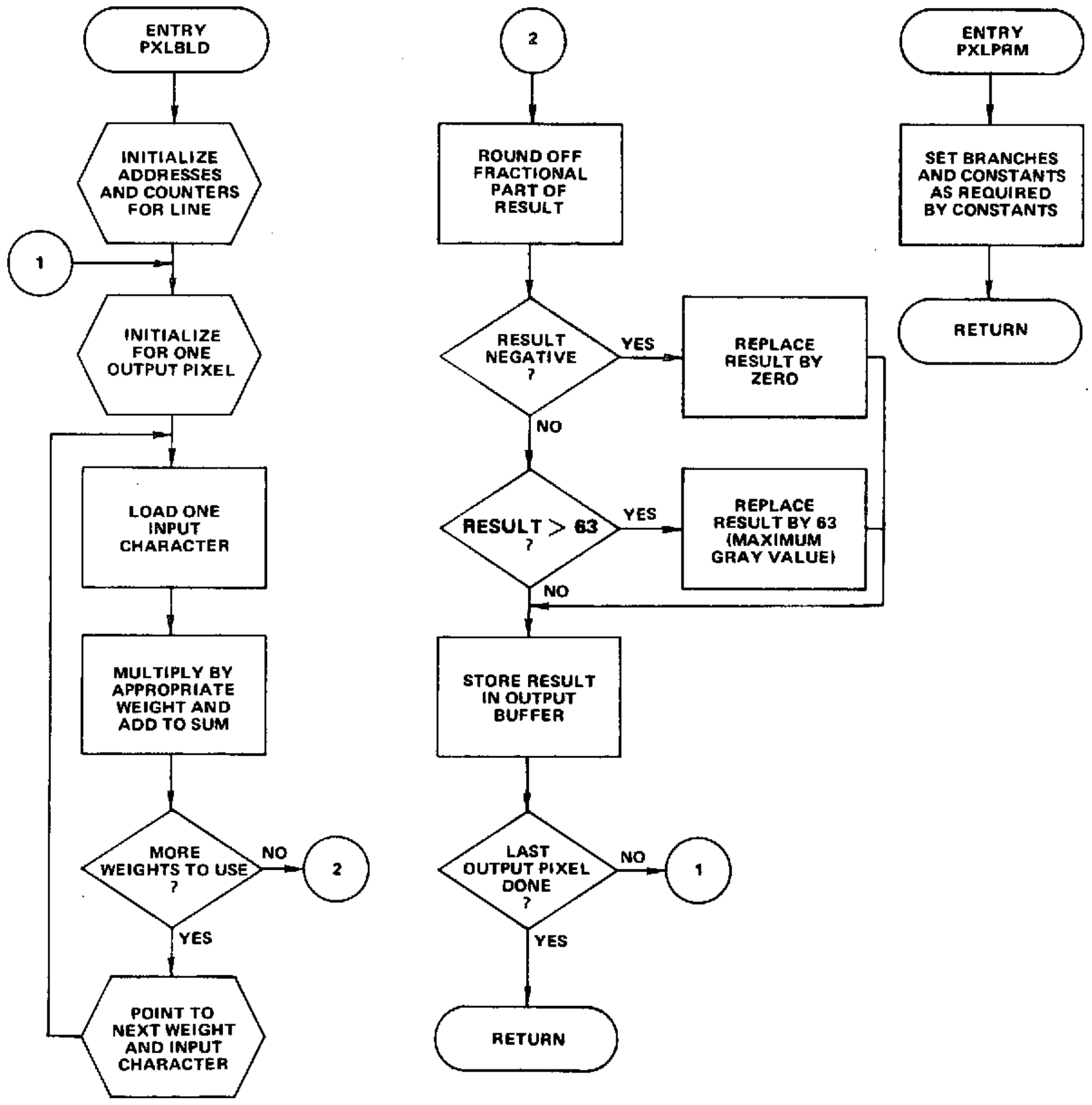

ENTRY PXLBLD
INITIALIZE ADDRESSES AND COUNTERS FOR LINE
1
INITIALIZE FOR ONE OUTPUT PIXEL
LOAD ONE INPUT CHARACTER
MULTIPLY BY APPROPRIATE WEIGHT AND ADD TO SUM
MORE WEIGHTS TO USE ?
NO
2
YES
POINT TO NEXT WEIGHT AND INPUT CHARACTER
2
ROUND OFF FRACTIONAL PART OF RESULT
RESULT NEGATIVE ?
YES
REPLACE RESULT BY ZERO
NO
RESULT > 63 ?
YES
REPLACE RESULT BY 63 (MAXIMUM GRAY VALUE)
NO
STORE RESULT IN OUTPUT BUFFER
LAST OUTPUT PIXEL DONE ?
NO
1
YES
RETURN
ENTRY PXLPRM
SET BRANCHES AND CONSTANTS AS REQUIRED BY CONSTANTS
RETURN

Figure C-6. EXPAND Program Flowchart (4 of 4)

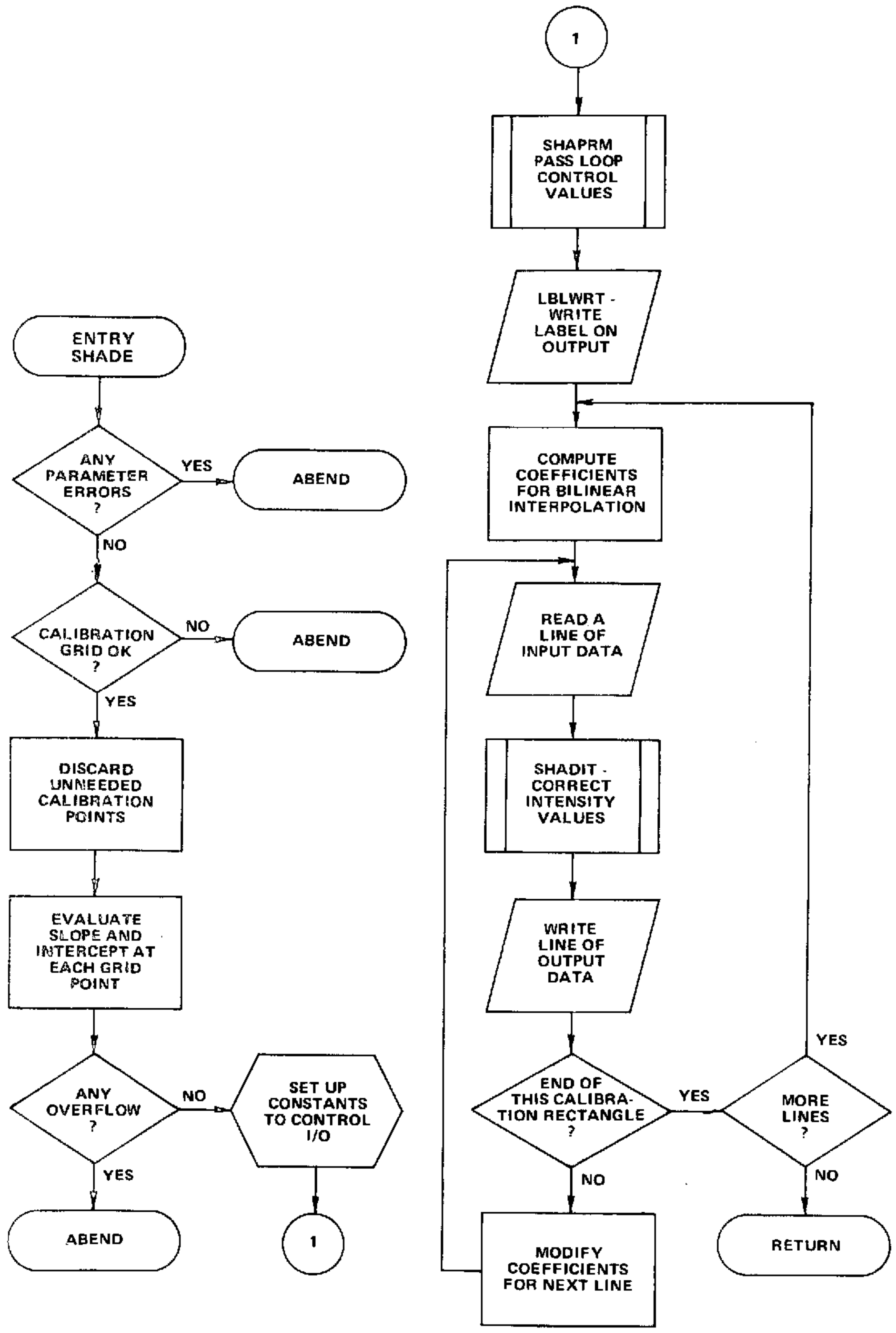

ENTRY SHADE
ANY PARAMETER ERRORS ?
YES
ABEND
NO
CALIBRATION GRID OK ?
NO
ABEND
YES
DISCARD UNNEEDED CALIBRATION POINTS
EVALUATE SLOPE AND INTERCEPT AT EACH GRID POINT
ANY OVERFLOW ?
NO
SET UP CONSTANTS TO CONTROL I/O
YES
ABEND
1
1
SHAPRM PASS LOOP CONTROL VALUES
LBLWRT - WRITE LABEL ON OUTPUT
COMPUTE COEFFICIENTS FOR BILINEAR INTERPOLATION
READ A LINE OF INPUT DATA
SHADIT - CORRECT INTENSITY VALUES
WRITE LINE OF OUTPUT DATA
END OF THIS CALIBRATION RECTANGLE ?
YES
MORE LINES ?
YES
NO
NO
MODIFY COEFFICIENTS FOR NEXT LINE
RETURN

Figure C-7. SHADE Program Flowchart (1 of 2)

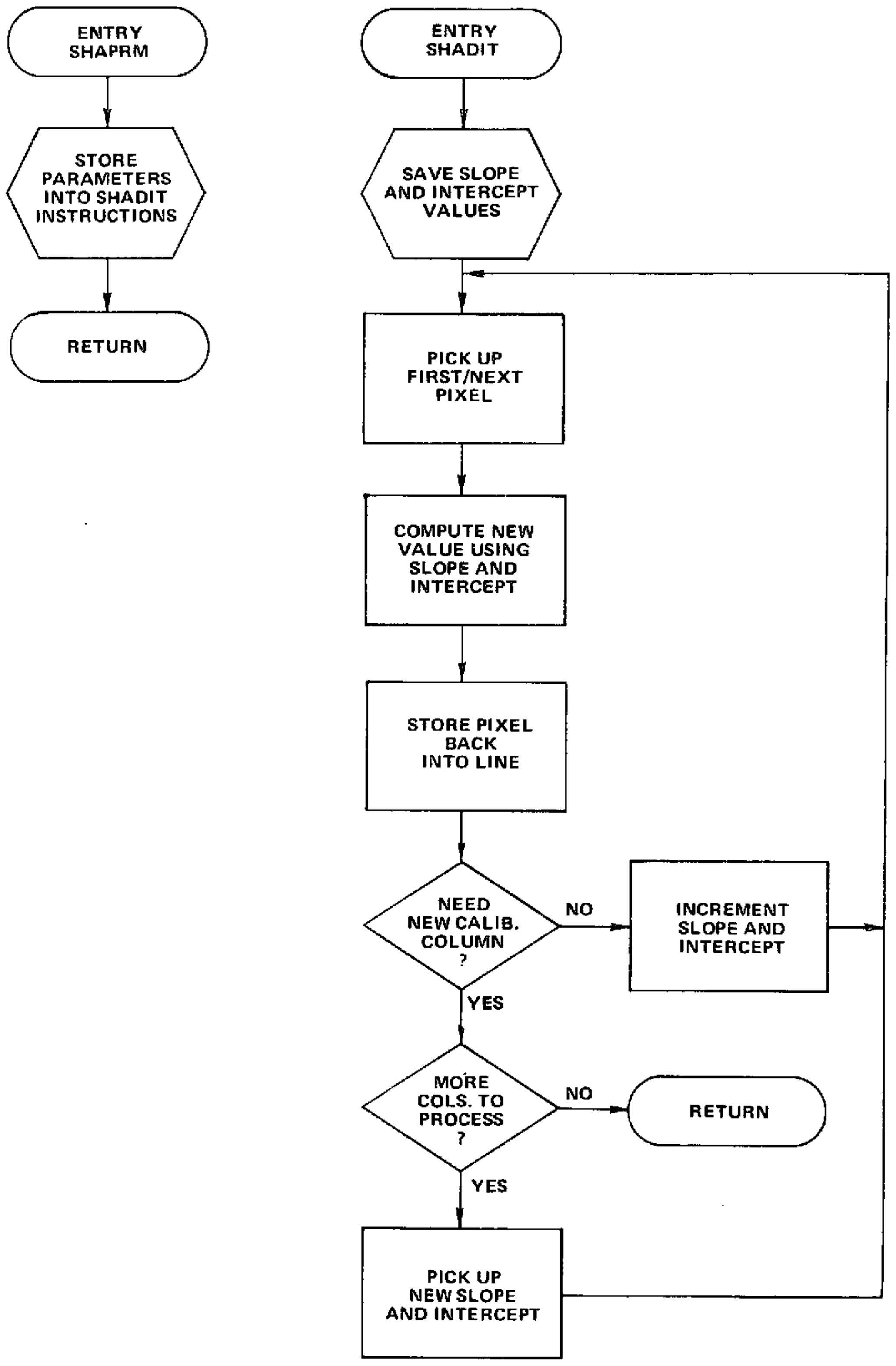

ENTRY SHAPRM
STORE PARAMETERS INTO SHADIT INSTRUCTIONS
RETURN
ENTRY SHADIT
SAVE SLOPE AND INTERCEPT VALUES
PICK UP FIRST/NEXT PIXEL
COMPUTE NEW VALUE USING SLOPE AND INTERCEPT
STORE PIXEL BACK INTO LINE
NEED NEW CALIB. COLUMN ?
NO
INCREMENT SLOPE AND INTERCEPT
YES
MORE COLS. TO PROCESS ?
NO
RETURN
YES
PICK UP NEW SLOPE AND INTERCEPT

Figure C-7. SHADE Program Flowchart (2 of 2)

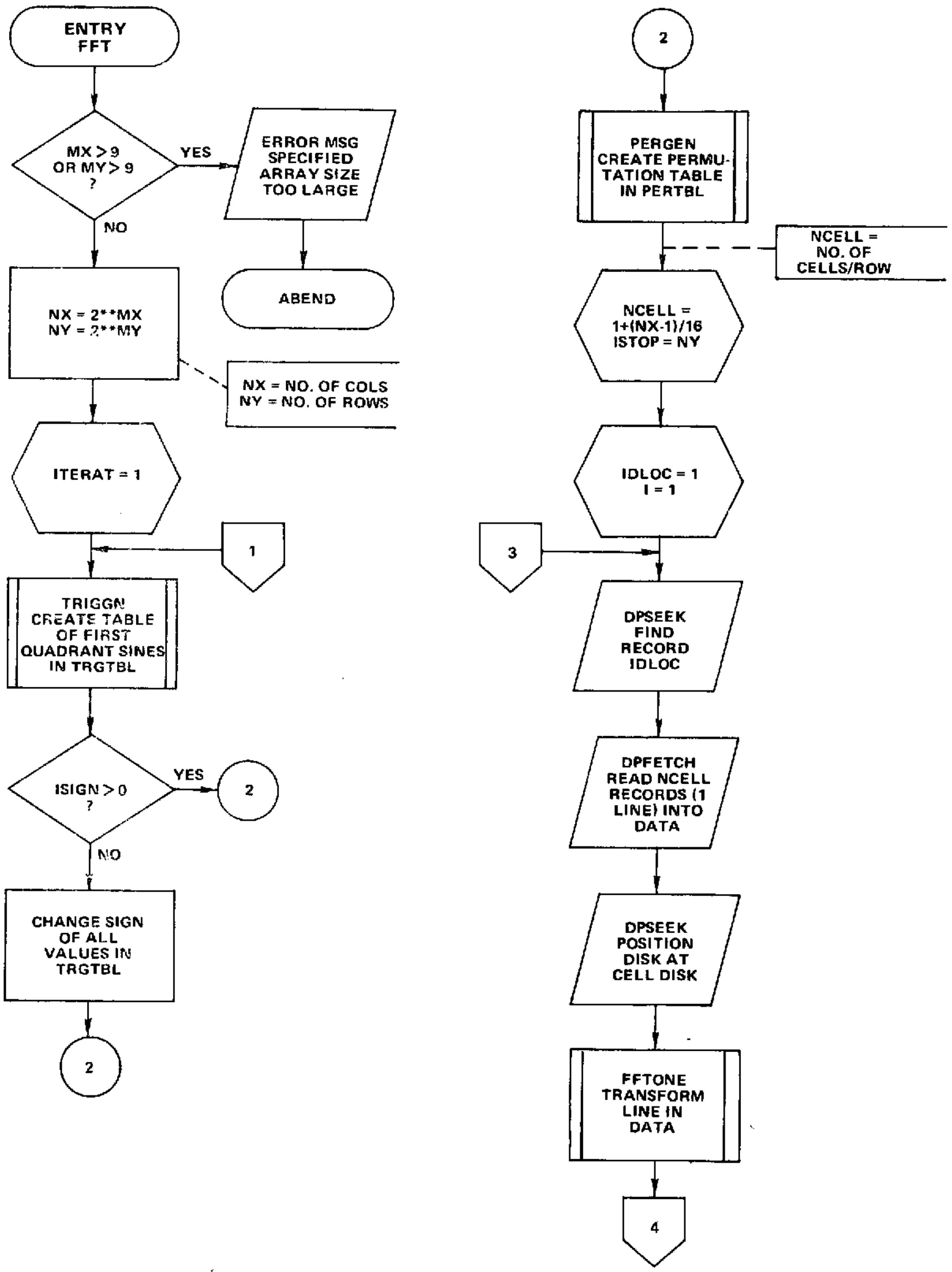

ENTRY FFT
MX > 9 OR MY > 9 ?
YES
ERROR MSG SPECIFIED ARRAY SIZE TOO LARGE
ABEND
NO
NX = 2**MX NY = 2**MY
NX = NO. OF COLS NY = NO. OF ROWS
ITERAT = 1
1
TRIGGN CREATE TABLE OF FIRST QUADRANT SINES IN TRGTBL
ISIGN > 0 ?
YES
2
NO
CHANGE SIGN OF ALL VALUES IN TRGTBL
2
2
PERGEN CREATE PERMUTATION TABLE IN PERTBL
NCELL = NO. OF CELLS/ROW
NCELL = 1+(NX-1)/16 ISTOP = NY
IDLOC = 1 I = 1
3
DPSEEK FIND RECORD IDLOC
DPFETCH READ NCELL RECORDS (1 LINE) INTO DATA
DPSEEK POSITION DISK AT CELL DISK
FFTONE TRANSFORM LINE IN DATA
4

Figure C-8. FFT Program Flowchart (1 of 4)

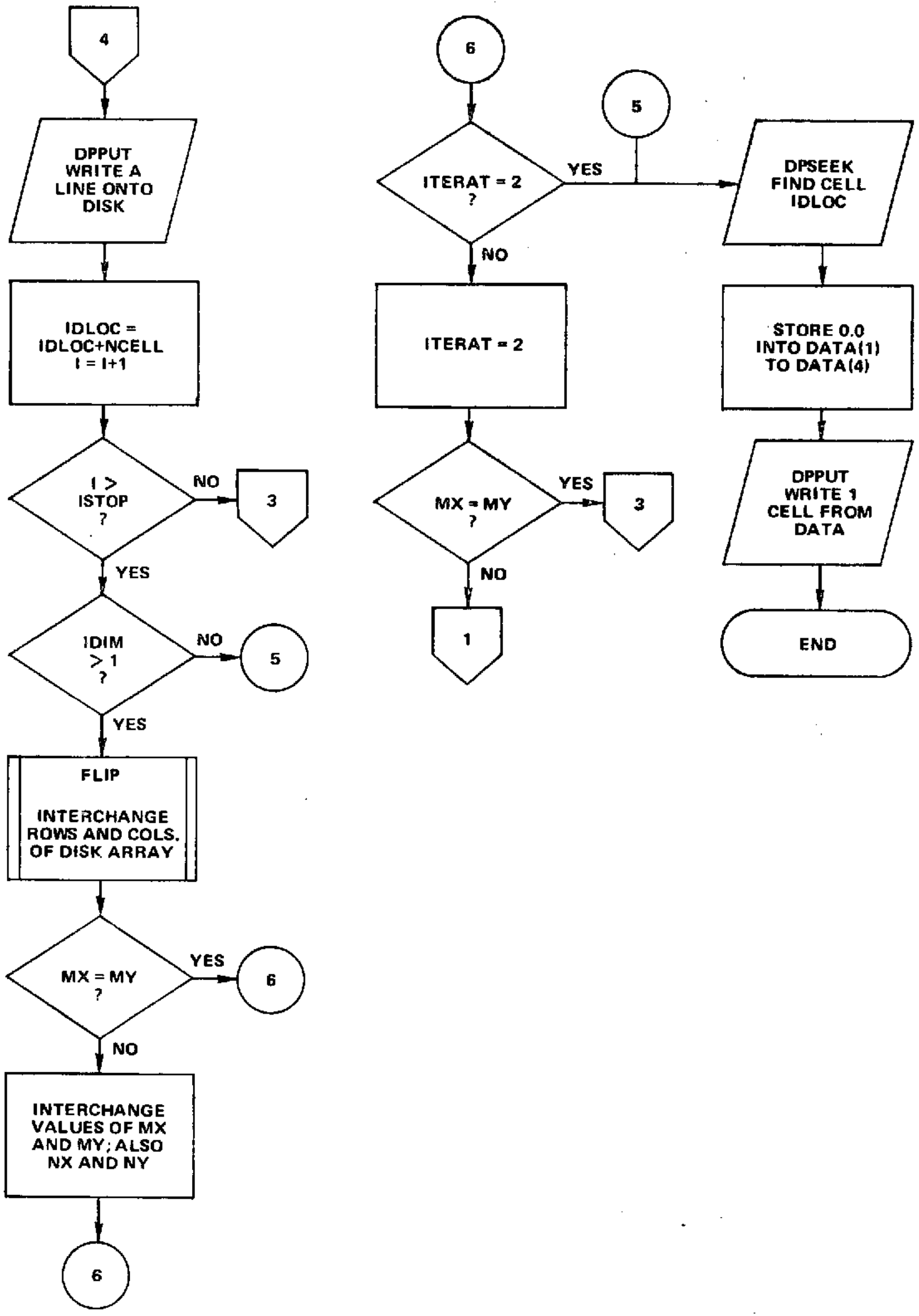

4
DPPUT WRITE A LINE ONTO DISK
IDLOC = IDLOC+NCELL
I = I+1
I > ISTOP ?
NO
3
YES
IDIM > 1 ?
NO
5
YES
FLIP
INTERCHANGE ROWS AND COLS. OF DISK ARRAY
MX = MY ?
YES
6
NO
INTERCHANGE VALUES OF MX AND MY; ALSO NX AND NY
6
6
ITERAT = 2 ?
YES
5
NO
ITERAT = 2
MX = MY ?
YES
3
NO
1
DPSEEK FIND CELL IDLOC
STORE 0.0 INTO DATA(1) TO DATA(4)
DPPUT WRITE 1 CELL FROM DATA
END

Figure C-8. FFT Program Flowchart (2 of 4)

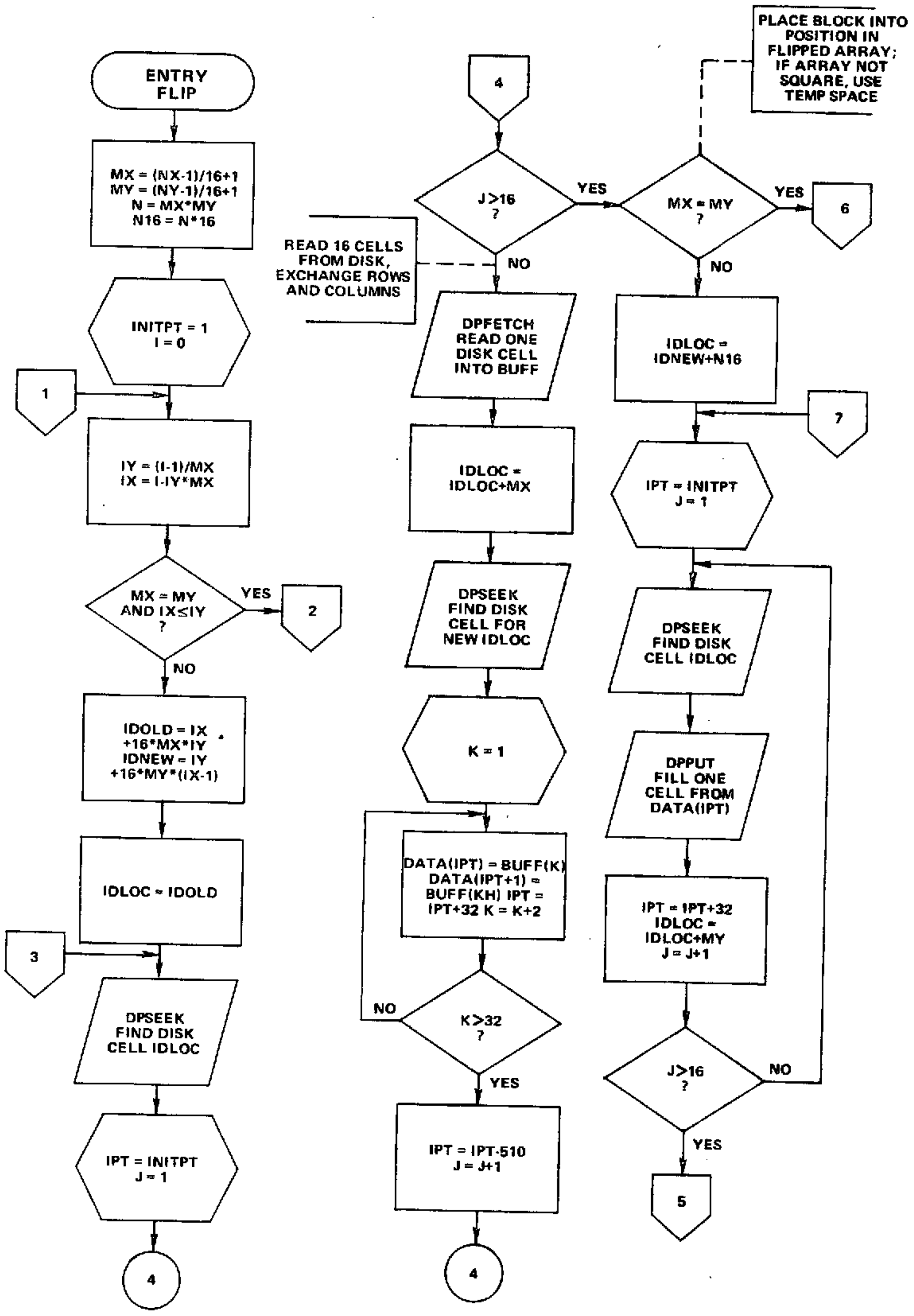


Figure C-8. FFT Program Flowchart (3 of 4)

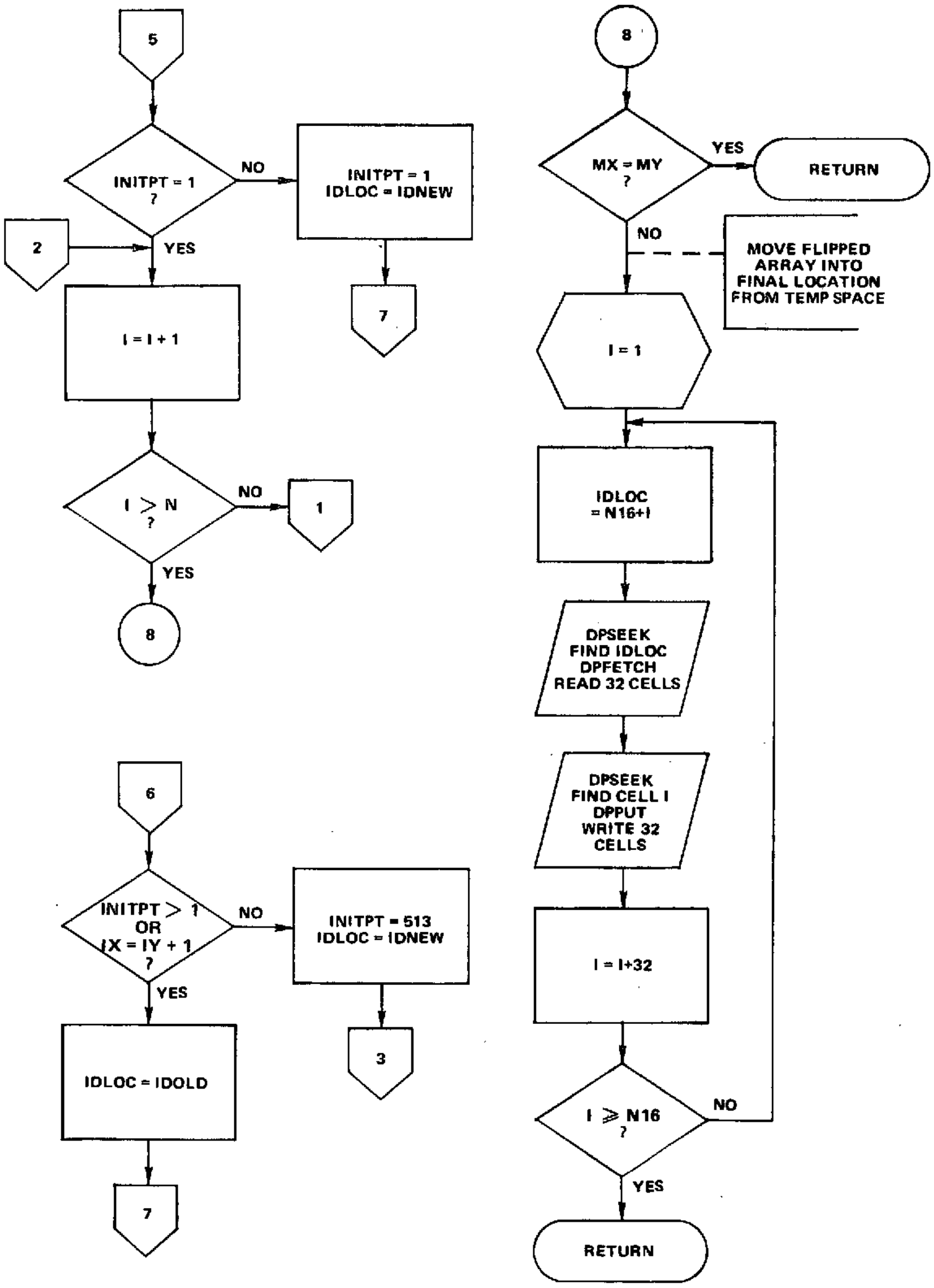


Figure C-8. FFT Program Flowchart (4 of 4)

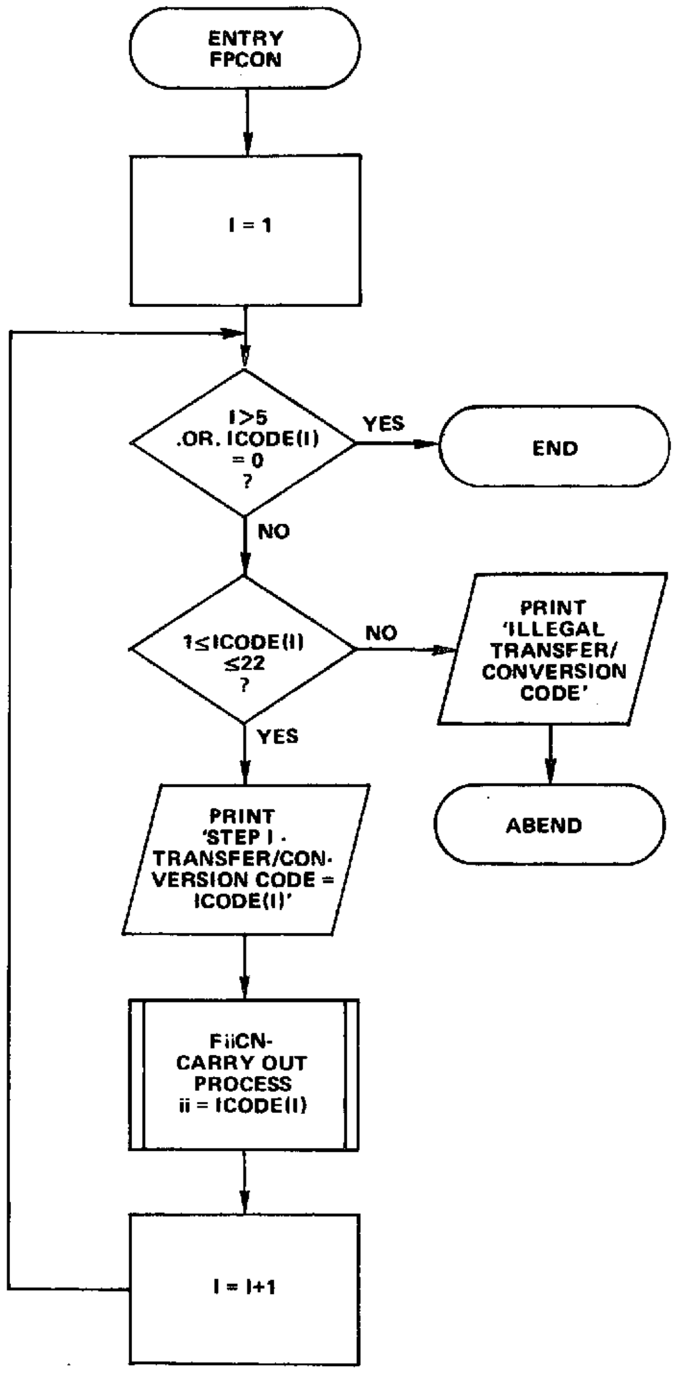

ENTRY FPCON
I = 1
I>5 .OR. ICODE(I) = 0 ?
YES
END
NO
1≤ICODE(I) ≤22 ?
NO
PRINT 'ILLEGAL TRANSFER/ CONVERSION CODE'
ABEND
YES
PRINT 'STEP I - TRANSFER/CON- VERSION CODE = ICODE(I)'
FiiCN- CARRY OUT PROCESS ii = ICODE(I)
I = I+1

Figure C-9. FPCON Program Flowchart (1 of 7)

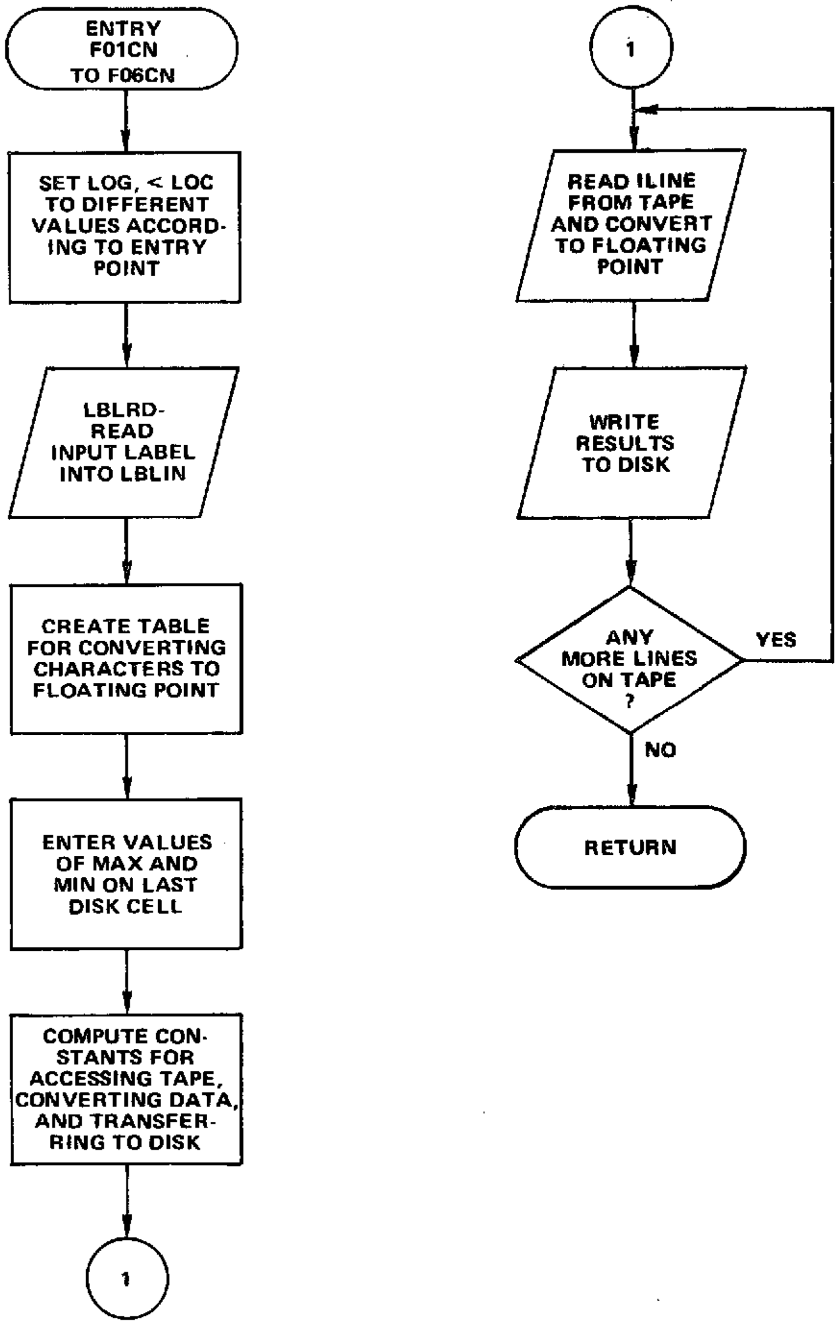

ENTRY F01CN TO F06CN
SET LOG, < LOC TO DIFFERENT VALUES ACCORDING TO ENTRY POINT
LBLRD- READ INPUT LABEL INTO LBLIN
CREATE TABLE FOR CONVERTING CHARACTERS TO FLOATING POINT
ENTER VALUES OF MAX AND MIN ON LAST DISK CELL
COMPUTE CONSTANTS FOR ACCESSING TAPE, CONVERTING DATA, AND TRANSFERRING TO DISK
1
1
READ ILINE FROM TAPE AND CONVERT TO FLOATING POINT
WRITE RESULTS TO DISK
ANY MORE LINES ON TAPE ?
YES
NO
RETURN

Figure C-9. FPCON Program Flowchart (2 of 7)

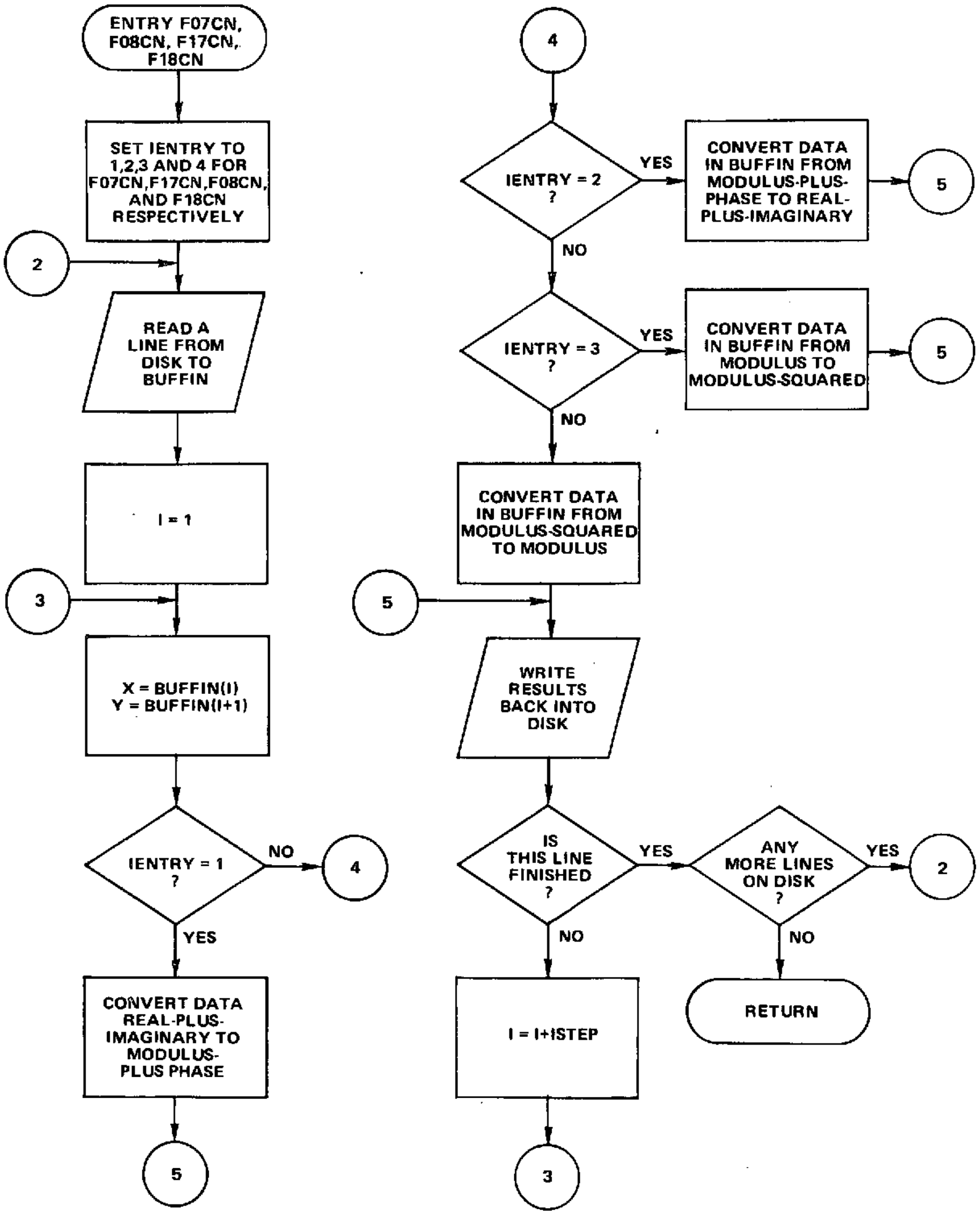

ENTRY F07CN, F08CN, F17CN, F18CN
SET IENTRY TO 1,2,3 AND 4 FOR F07CN,F17CN,F08CN, AND F18CN RESPECTIVELY
2
READ A LINE FROM DISK TO BUFFIN
I = 1
3
X = BUFFIN(I)
Y = BUFFIN(I+1)
IENTRY = 1 ?
NO
4
YES
CONVERT DATA REAL-PLUS-IMAGINARY TO MODULUS-PLUS PHASE
5
4
IENTRY = 2 ?
YES
CONVERT DATA IN BUFFIN FROM MODULUS-PLUS-PHASE TO REAL-PLUS-IMAGINARY
5
NO
IENTRY = 3 ?
YES
CONVERT DATA IN BUFFIN FROM MODULUS TO MODULUS-SQUARED
5
NO
CONVERT DATA IN BUFFIN FROM MODULUS-SQUARED TO MODULUS
5
WRITE RESULTS BACK INTO DISK
IS THIS LINE FINISHED ?
YES
ANY MORE LINES ON DISK ?
YES
2
NO
RETURN
NO
I = I+ISTEP
3

Figure C-9. FPCON Program Flowchart (3 of 7)

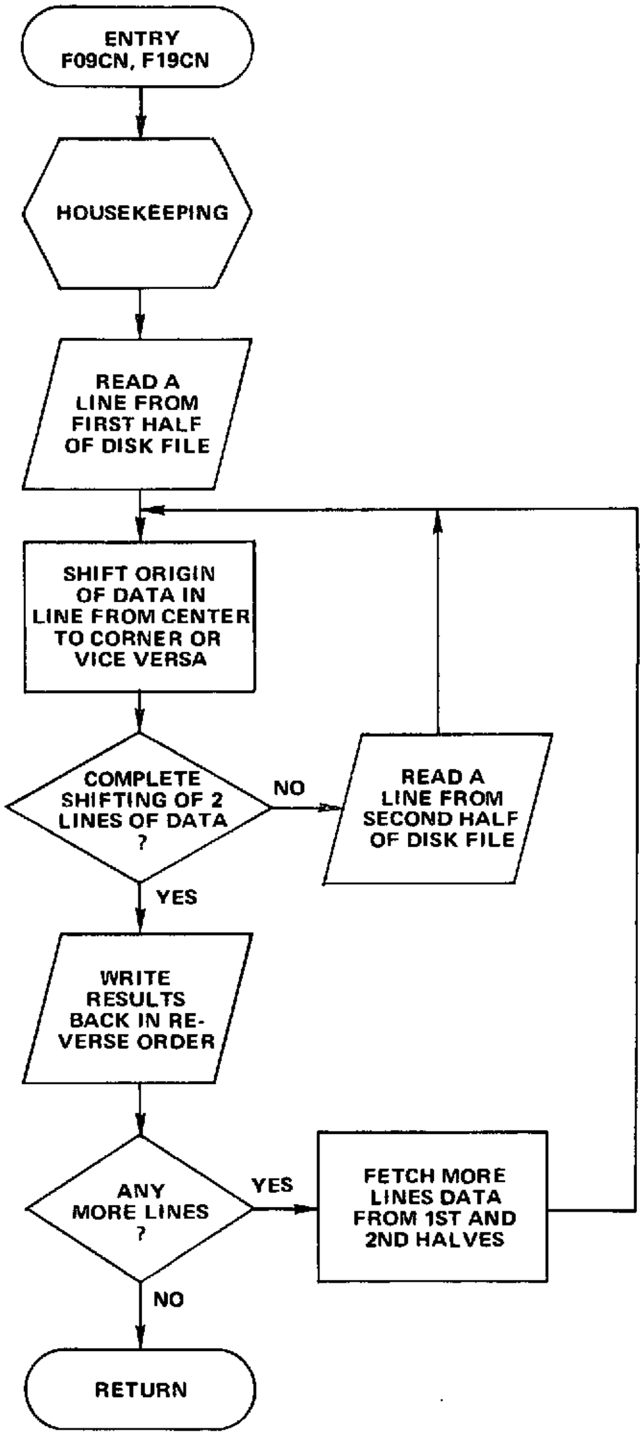

ENTRY
F09CN, F19CN
HOUSEKEEPING
READ A LINE FROM FIRST HALF OF DISK FILE
SHIFT ORIGIN OF DATA IN LINE FROM CENTER TO CORNER OR VICE VERSA
COMPLETE SHIFTING OF 2 LINES OF DATA ?
NO
READ A LINE FROM SECOND HALF OF DISK FILE
YES
WRITE RESULTS BACK IN RE- VERSE ORDER
ANY MORE LINES ?
YES
FETCH MORE LINES DATA FROM 1ST AND 2ND HALVES
NO
RETURN

Figure C-9. FPCON Program Flowchart (4 of 7)

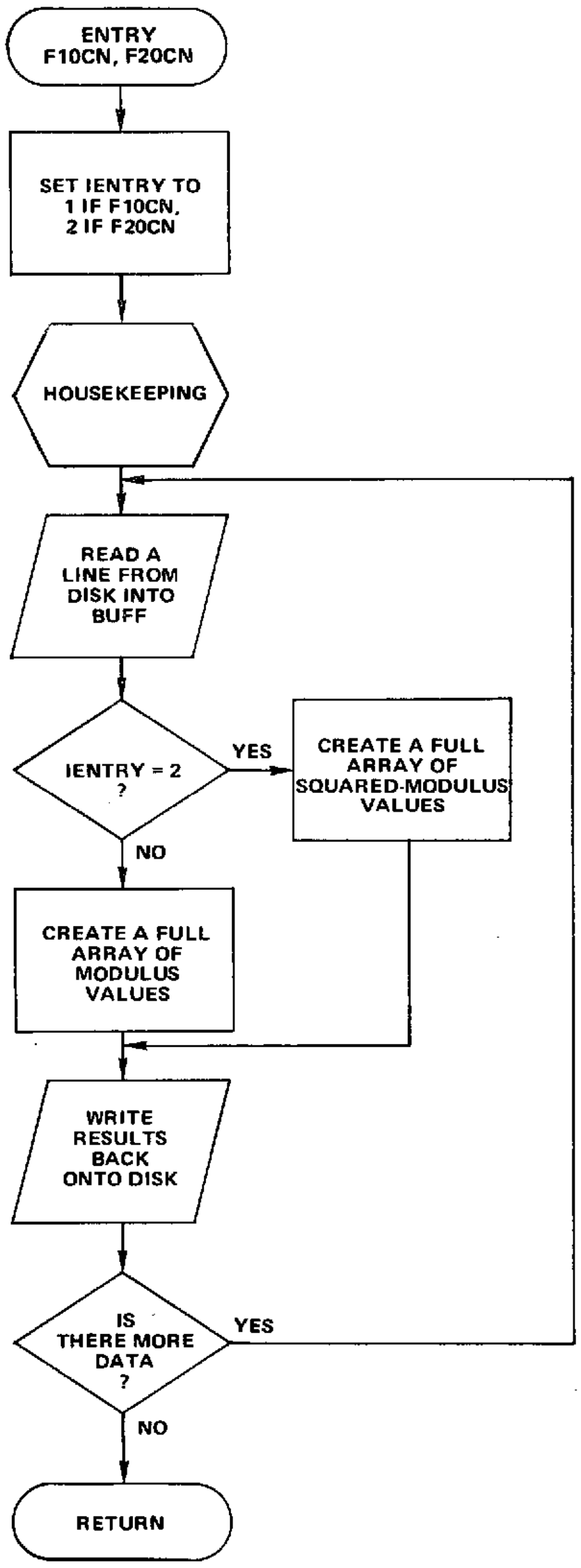


Figure C-9. FPCON Program Flowchart (5 of 7)

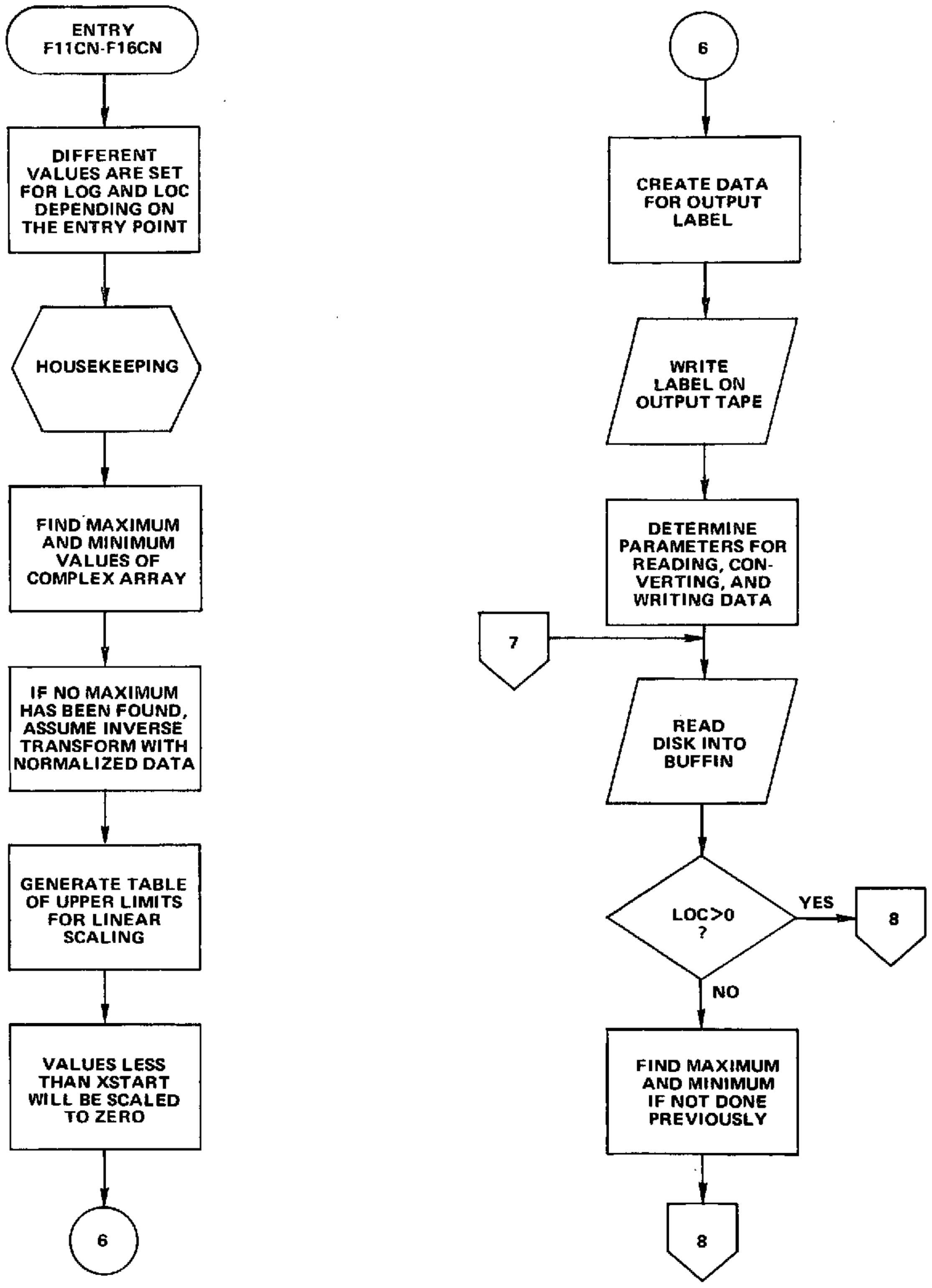

ENTRY F11CN-F16CN
DIFFERENT VALUES ARE SET FOR LOG AND LOC DEPENDING ON THE ENTRY POINT
HOUSEKEEPING
FIND MAXIMUM AND MINIMUM VALUES OF COMPLEX ARRAY
IF NO MAXIMUM HAS BEEN FOUND, ASSUME INVERSE TRANSFORM WITH NORMALIZED DATA
GENERATE TABLE OF UPPER LIMITS FOR LINEAR SCALING
VALUES LESS THAN XSTART WILL BE SCALED TO ZERO
6
6
CREATE DATA FOR OUTPUT LABEL
WRITE LABEL ON OUTPUT TAPE
DETERMINE PARAMETERS FOR READING, CONVERTING, AND WRITING DATA
7
READ DISK INTO BUFFIN
LOC>0 ?
YES
8
NO
FIND MAXIMUM AND MINIMUM IF NOT DONE PREVIOUSLY
8

Figure C-9. FPCON Program Flowchart (6 of 7)

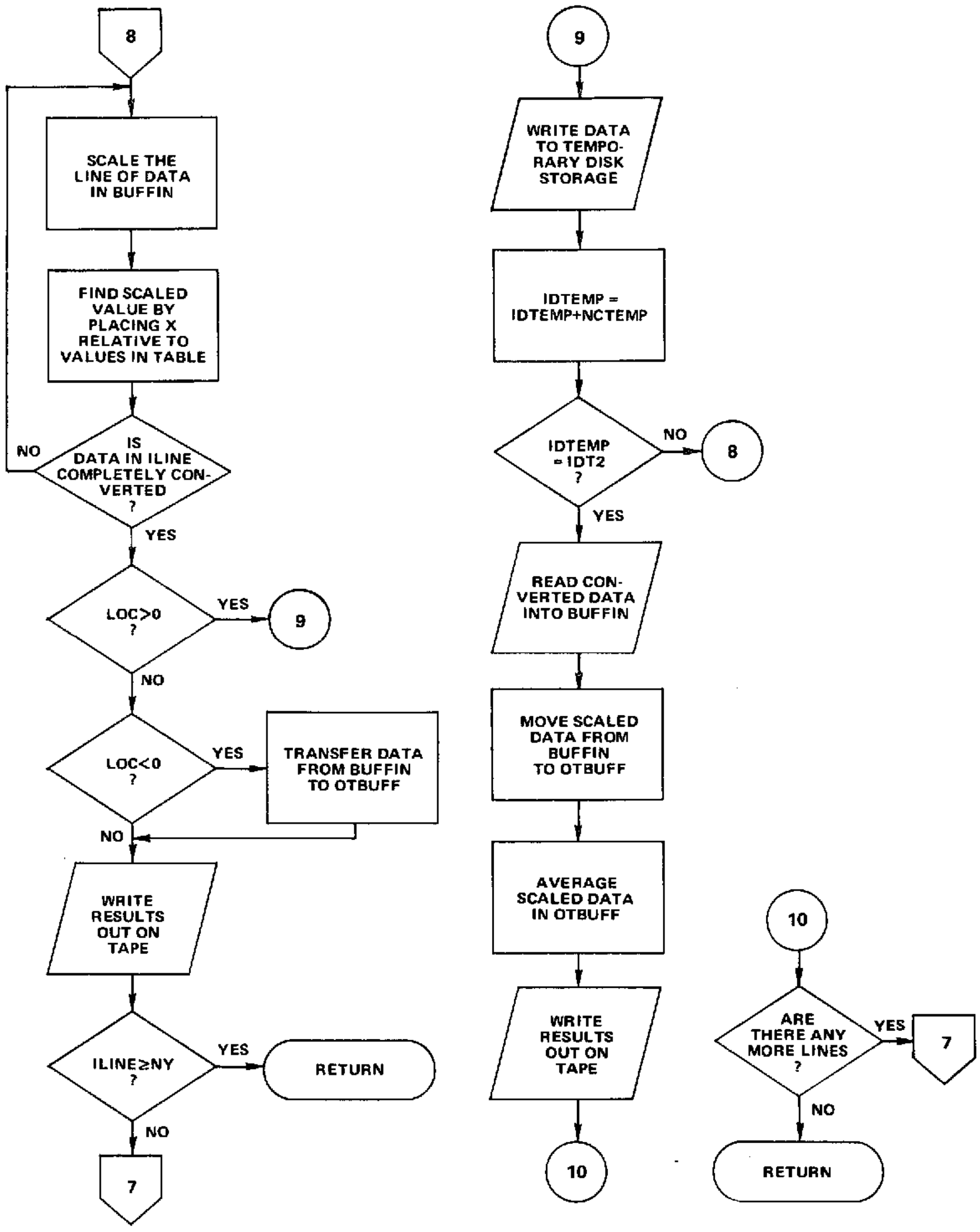

8
SCALE THE LINE OF DATA IN BUFFIN
FIND SCALED VALUE BY PLACING X RELATIVE TO VALUES IN TABLE
IS DATA IN ILINE COMPLETELY CONVERTED ?
NO
YES
LOC>0 ?
YES
9
NO
LOC<0 ?
YES
TRANSFER DATA FROM BUFFIN TO OTBUFF
NO
WRITE RESULTS OUT ON TAPE
ILINE≥NY ?
YES
RETURN
NO
7
9
WRITE DATA TO TEMPORARY DISK STORAGE
IDTEMP = IDTEMP+NCTEMP
IDTEMP = IDT2 ?
NO
8
YES
READ CONVERTED DATA INTO BUFFIN
MOVE SCALED DATA FROM BUFFIN TO OTBUFF
AVERAGE SCALED DATA IN OTBUFF
WRITE RESULTS OUT ON TAPE
10
10
ARE THERE ANY MORE LINES ?
YES
7
NO
RETURN

Figure C-9. PPCON Program Flowchart (7 of 7)

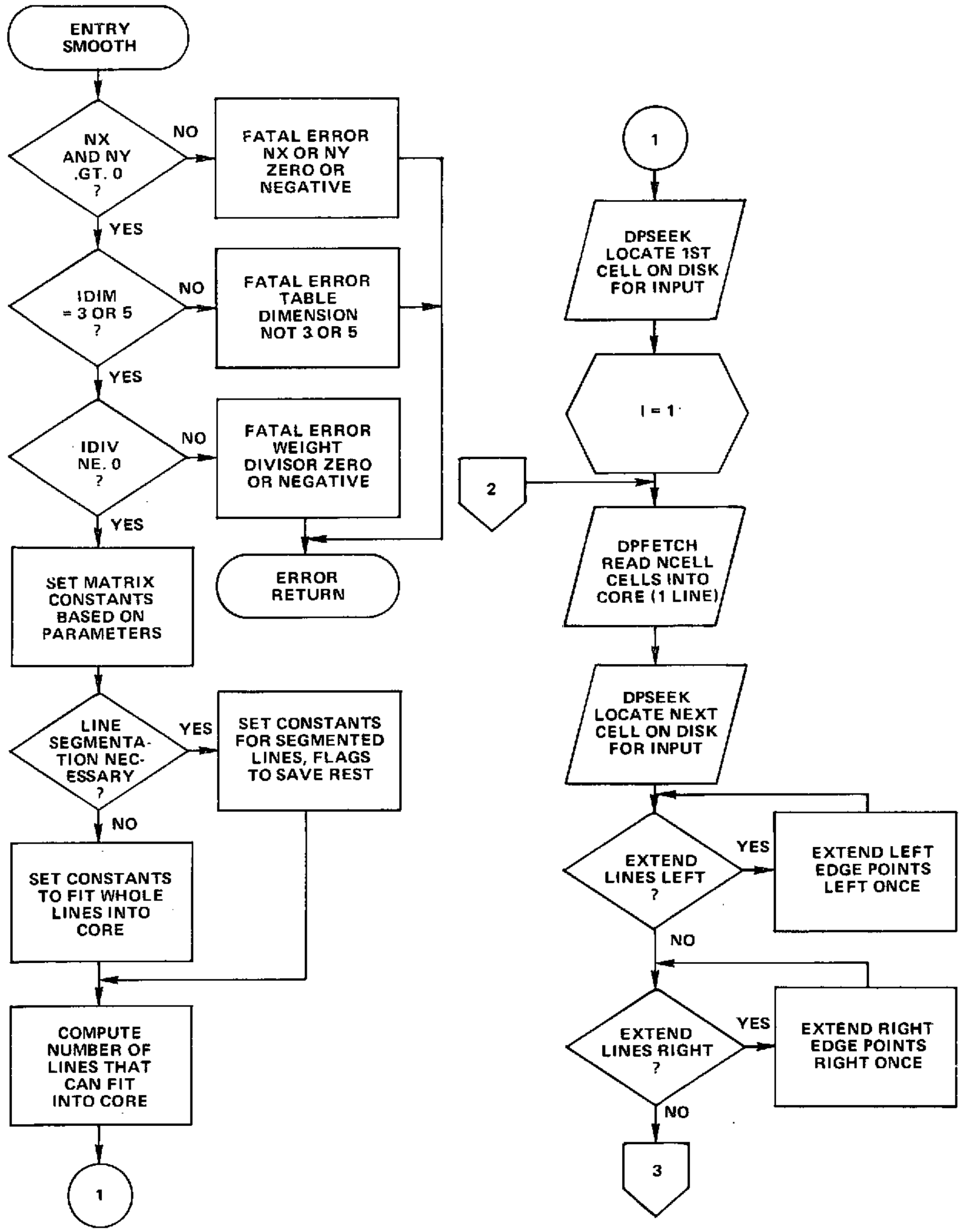

ENTRY SMOOTH
NX AND NY .GT. 0 ?
NO
FATAL ERROR NX OR NY ZERO OR NEGATIVE
YES
IDIM = 3 OR 5 ?
NO
FATAL ERROR TABLE DIMENSION NOT 3 OR 5
YES
IDIV NE. 0 ?
NO
FATAL ERROR WEIGHT DIVISOR ZERO OR NEGATIVE
YES
ERROR RETURN
SET MATRIX CONSTANTS BASED ON PARAMETERS
LINE SEGMENTATION NECESSARY ?
YES
SET CONSTANTS FOR SEGMENTED LINES, FLAGS TO SAVE REST
NO
SET CONSTANTS TO FIT WHOLE LINES INTO CORE
COMPUTE NUMBER OF LINES THAT CAN FIT INTO CORE
1
1
DPSEEK LOCATE 1ST CELL ON DISK FOR INPUT
I = 1
2
DPFETCH READ NCELL CELLS INTO CORE (1 LINE)
DPSEEK LOCATE NEXT CELL ON DISK FOR INPUT
EXTEND LINES LEFT ?
YES
EXTEND LEFT EDGE POINTS LEFT ONCE
NO
EXTEND LINES RIGHT ?
YES
EXTEND RIGHT EDGE POINTS RIGHT ONCE
NO
3

Figure C-10. SMOOTH Program Flowchart (1 of 4)

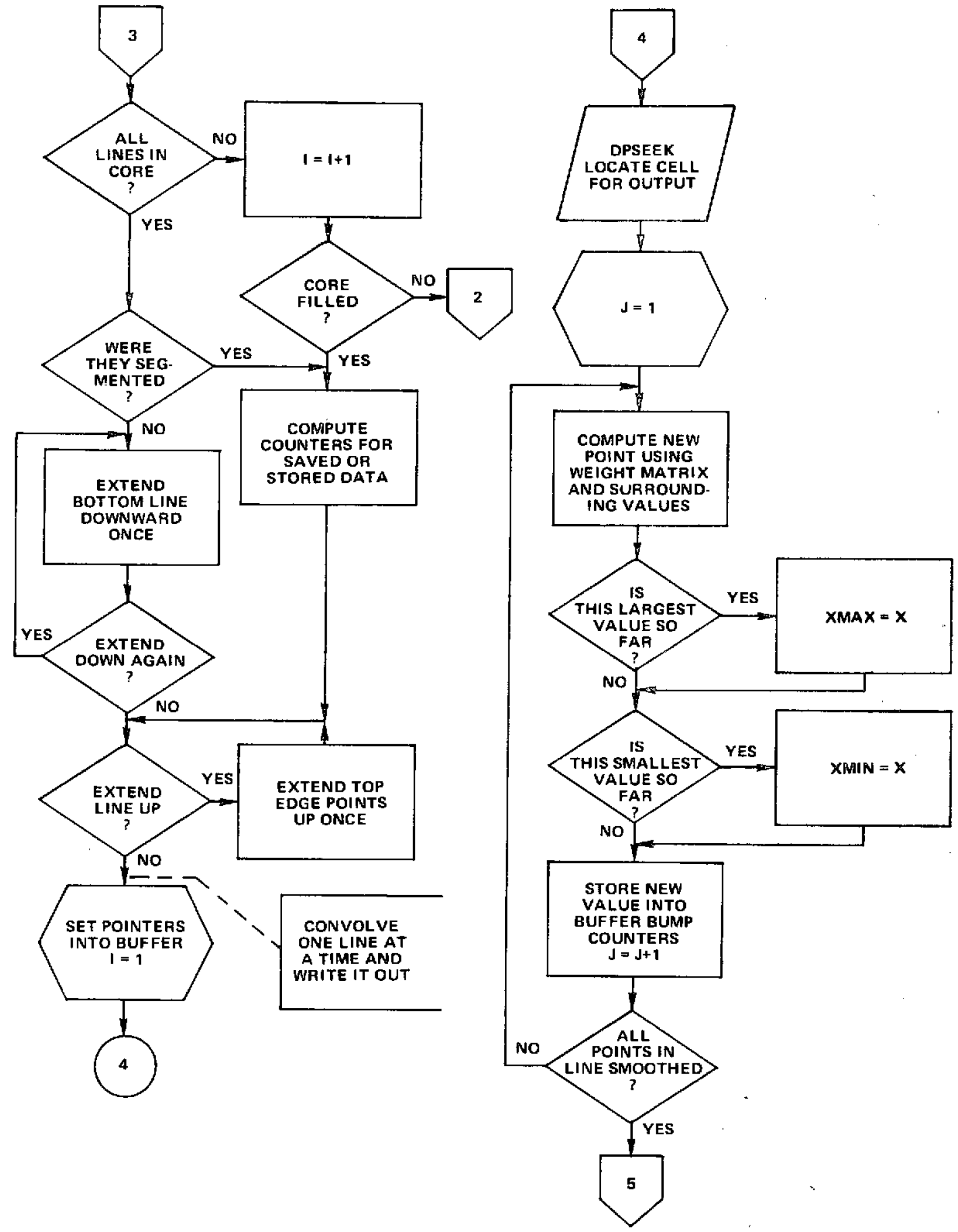

3
ALL LINES IN CORE ?
NO
I = I+1
YES
CORE FILLED ?
NO
2
WERE THEY SEG-MENTED ?
YES
YES
NO
COMPUTE COUNTERS FOR SAVED OR STORED DATA
EXTEND BOTTOM LINE DOWNWARD ONCE
YES
EXTEND DOWN AGAIN ?
NO
EXTEND LINE UP ?
YES
EXTEND TOP EDGE POINTS UP ONCE
NO
SET POINTERS INTO BUFFER I = 1
CONVOLVE ONE LINE AT A TIME AND WRITE IT OUT
4
4
DPSEEK LOCATE CELL FOR OUTPUT
J = 1
COMPUTE NEW POINT USING WEIGHT MATRIX AND SURROUND-ING VALUES
IS THIS LARGEST VALUE SO FAR ?
YES
XMAX = X
NO
IS THIS SMALLEST VALUE SO FAR ?
YES
XMIN = X
NO
STORE NEW VALUE INTO BUFFER BUMP COUNTERS J = J+1
NO
ALL POINTS IN LINE SMOOTHED ?
YES
5

Figure C-10. SMOOTH Program Flowchart (2 of 4)

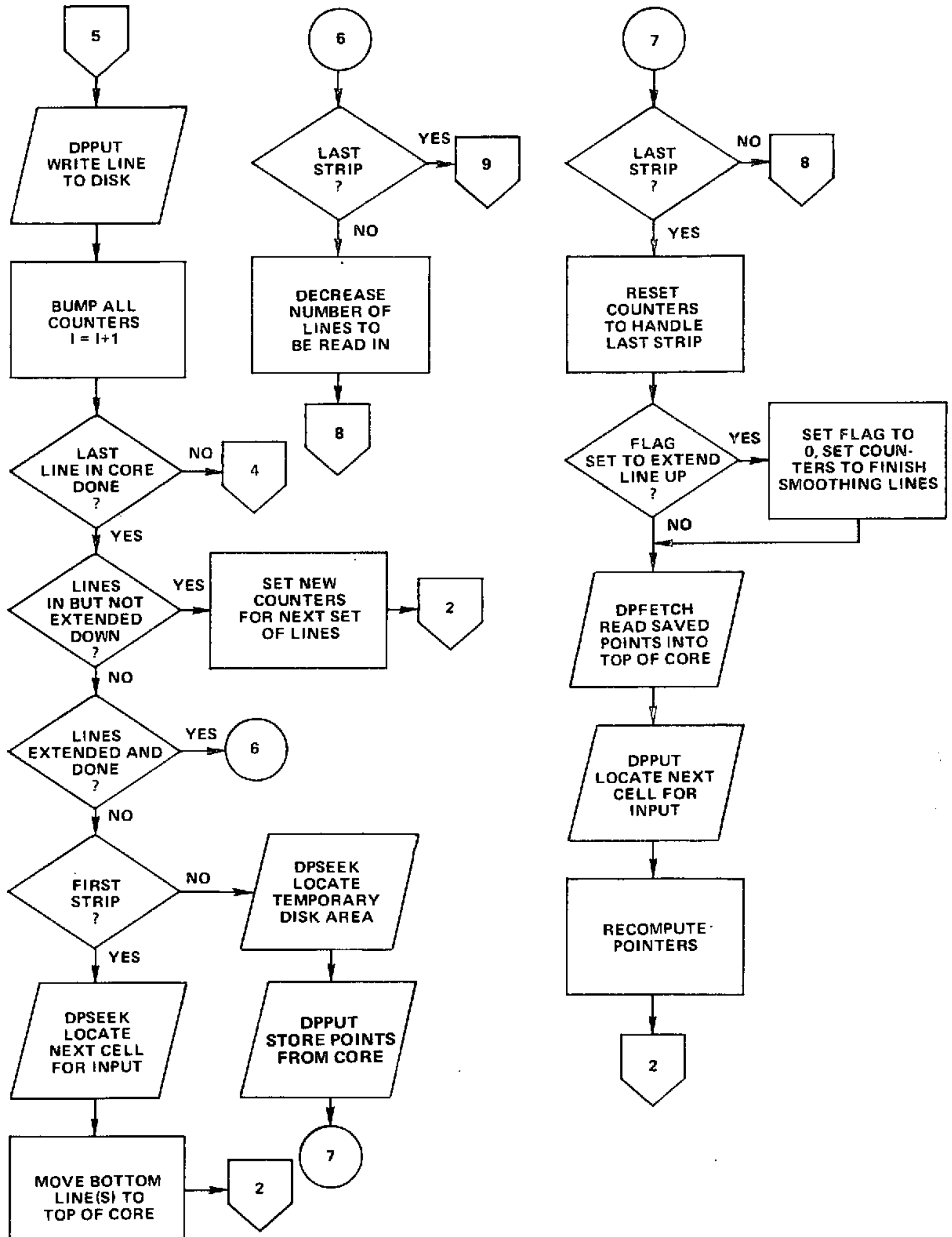

5
DPPUT WRITE LINE TO DISK
BUMP ALL COUNTERS I = I+1
LAST LINE IN CORE DONE ?
NO
4
YES
LINES IN BUT NOT EXTENDED DOWN ?
YES
SET NEW COUNTERS FOR NEXT SET OF LINES
2
NO
LINES EXTENDED AND DONE ?
YES
6
NO
FIRST STRIP ?
NO
DPSEEK LOCATE TEMPORARY DISK AREA
DPPUT STORE POINTS FROM CORE
7
YES
DPSEEK LOCATE NEXT CELL FOR INPUT
MOVE BOTTOM LINE(S) TO TOP OF CORE
2
6
LAST STRIP ?
YES
9
NO
DECREASE NUMBER OF LINES TO BE READ IN
8
7
LAST STRIP ?
NO
8
YES
RESET COUNTERS TO HANDLE LAST STRIP
FLAG SET TO EXTEND LINE UP ?
YES
SET FLAG TO 0, SET COUN-TERS TO FINISH SMOOTHING LINES
NO
DPFETCH READ SAVED POINTS INTO TOP OF CORE
DPPUT LOCATE NEXT CELL FOR INPUT
RECOMPUTE POINTERS
2

Figure C-10. SMOOTH Program Flowchart (3 of 4)

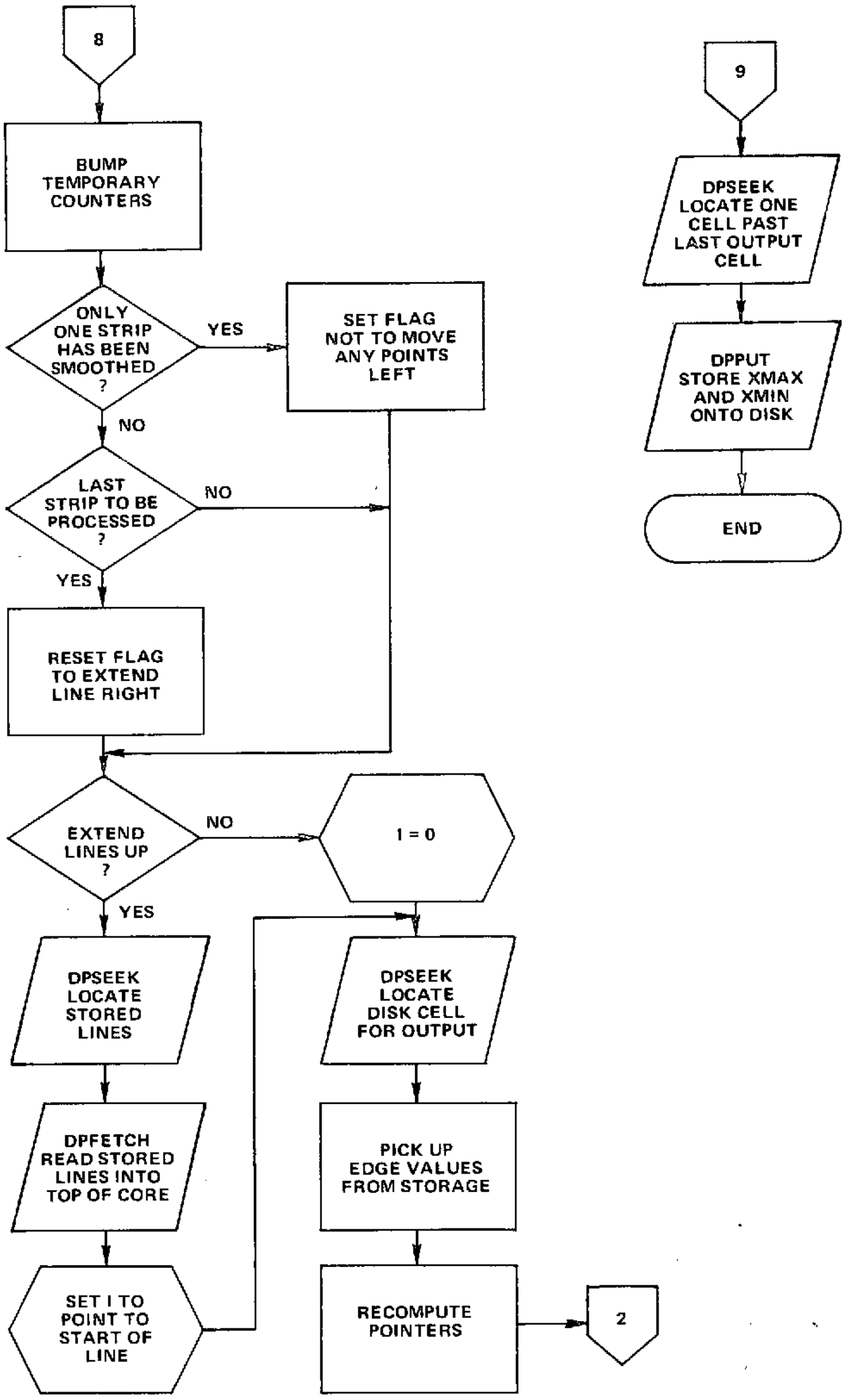

8
BUMP TEMPORARY COUNTERS
ONLY ONE STRIP HAS BEEN SMOOTHED ?
YES
SET FLAG NOT TO MOVE ANY POINTS LEFT
NO
LAST STRIP TO BE PROCESSED ?
NO
YES
RESET FLAG TO EXTEND LINE RIGHT
EXTEND LINES UP ?
NO
I = 0
YES
DPSEEK LOCATE STORED LINES
DPFETCH READ STORED LINES INTO TOP OF CORE
SET I TO POINT TO START OF LINE
DPSEEK LOCATE DISK CELL FOR OUTPUT
PICK UP EDGE VALUES FROM STORAGE
RECOMPUTE POINTERS
2
9
DPSEEK LOCATE ONE CELL PAST LAST OUTPUT CELL
DPPUT STORE XMAX AND XMIN ONTO DISK
END

Figure C-10. SMOOTH Program Flowchart (4 of 4)

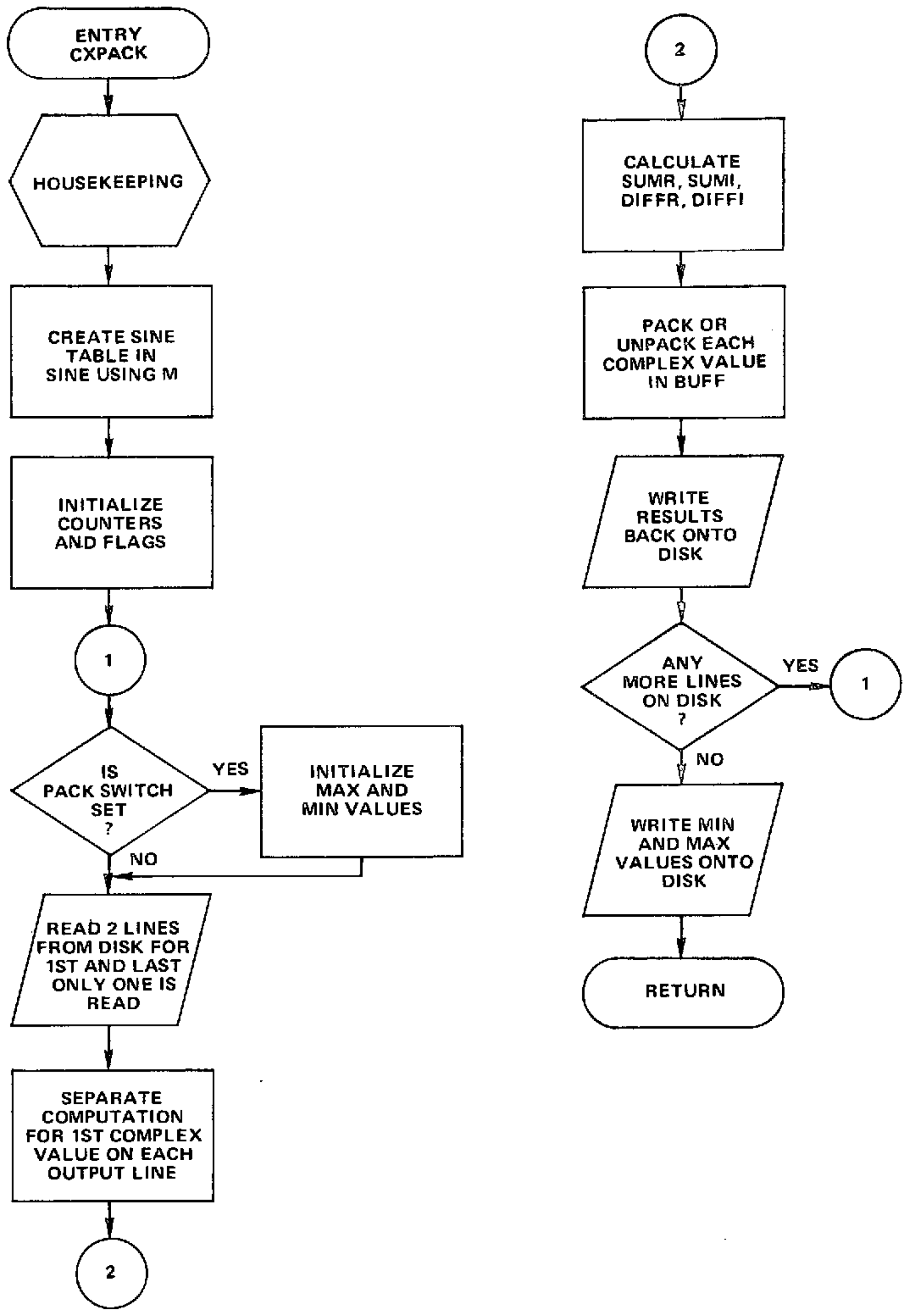

ENTRY CXPACK
HOUSEKEEPING
CREATE SINE TABLE IN SINE USING M
INITIALIZE COUNTERS AND FLAGS
1
IS PACK SWITCH SET ?
YES
INITIALIZE MAX AND MIN VALUES
NO
READ 2 LINES FROM DISK FOR 1ST AND LAST ONLY ONE IS READ
SEPARATE COMPUTATION FOR 1ST COMPLEX VALUE ON EACH OUTPUT LINE
2
2
CALCULATE SUMR, SUMI, DIFFR, DIFFI
PACK OR UNPACK EACH COMPLEX VALUE IN BUFF
WRITE RESULTS BACK ONTO DISK
ANY MORE LINES ON DISK ?
YES
1
NO
WRITE MIN AND MAX VALUES ONTO DISK
RETURN

Figure C-11. CXPACK Program Flowchart

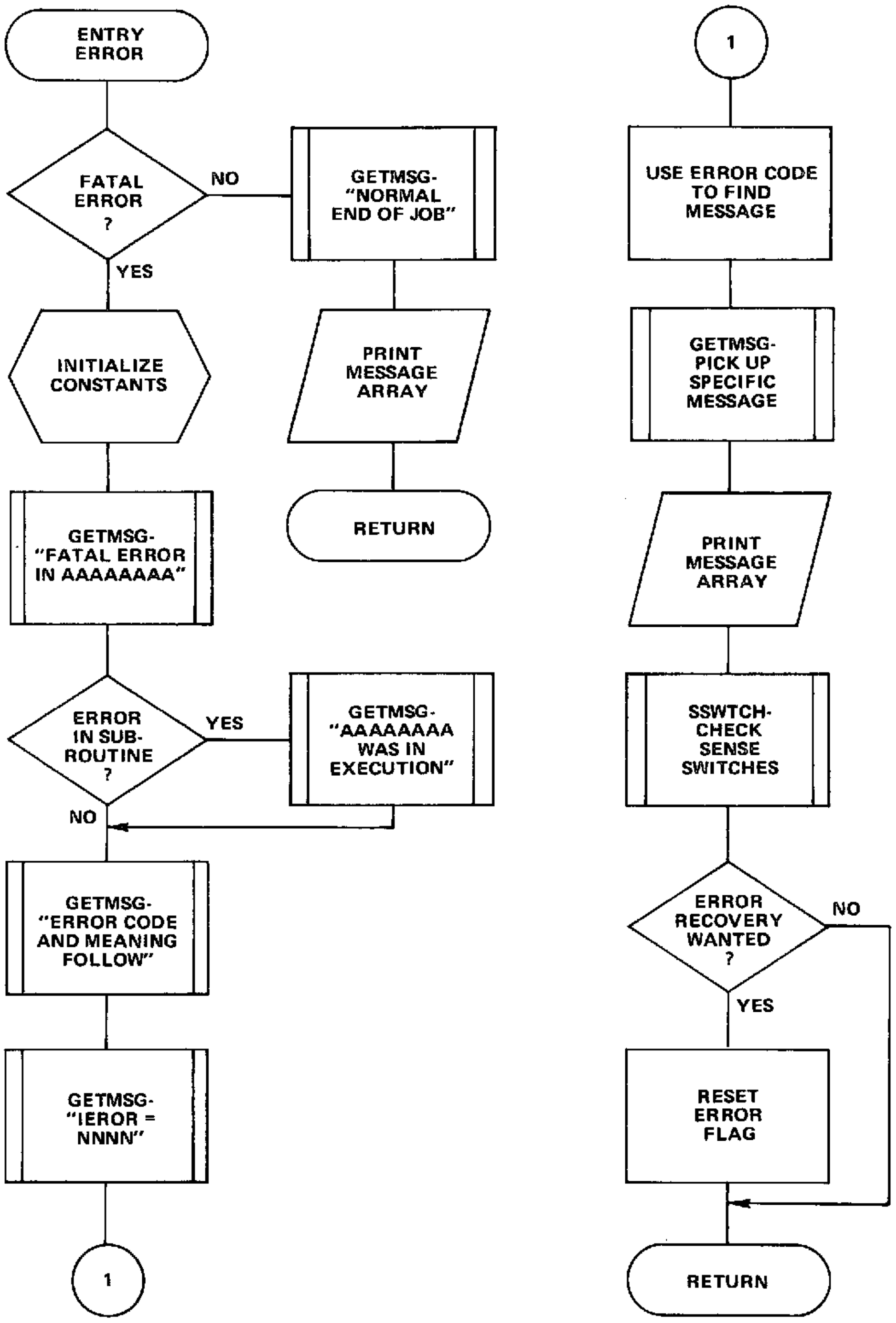

ENTRY ERROR
FATAL ERROR ?
NO
YES
GETMSG- "NORMAL END OF JOB"
PRINT MESSAGE ARRAY
RETURN
INITIALIZE CONSTANTS
GETMSG- "FATAL ERROR IN AAAAAAAA"
ERROR IN SUB-ROUTINE ?
YES
NO
GETMSG- "AAAAAAAA WAS IN EXECUTION"
GETMSG- "ERROR CODE AND MEANING FOLLOW"
GETMSG- "IEROR = NNNN"
1
1
USE ERROR CODE TO FIND MESSAGE
GETMSG- PICK UP SPECIFIC MESSAGE
PRINT MESSAGE ARRAY
SSWTCH- CHECK SENSE SWITCHES
ERROR RECOVERY WANTED ?
NO
YES
RESET ERROR FLAG
RETURN

Figure C-12. ERROR Program Flowchart (1 of 2)

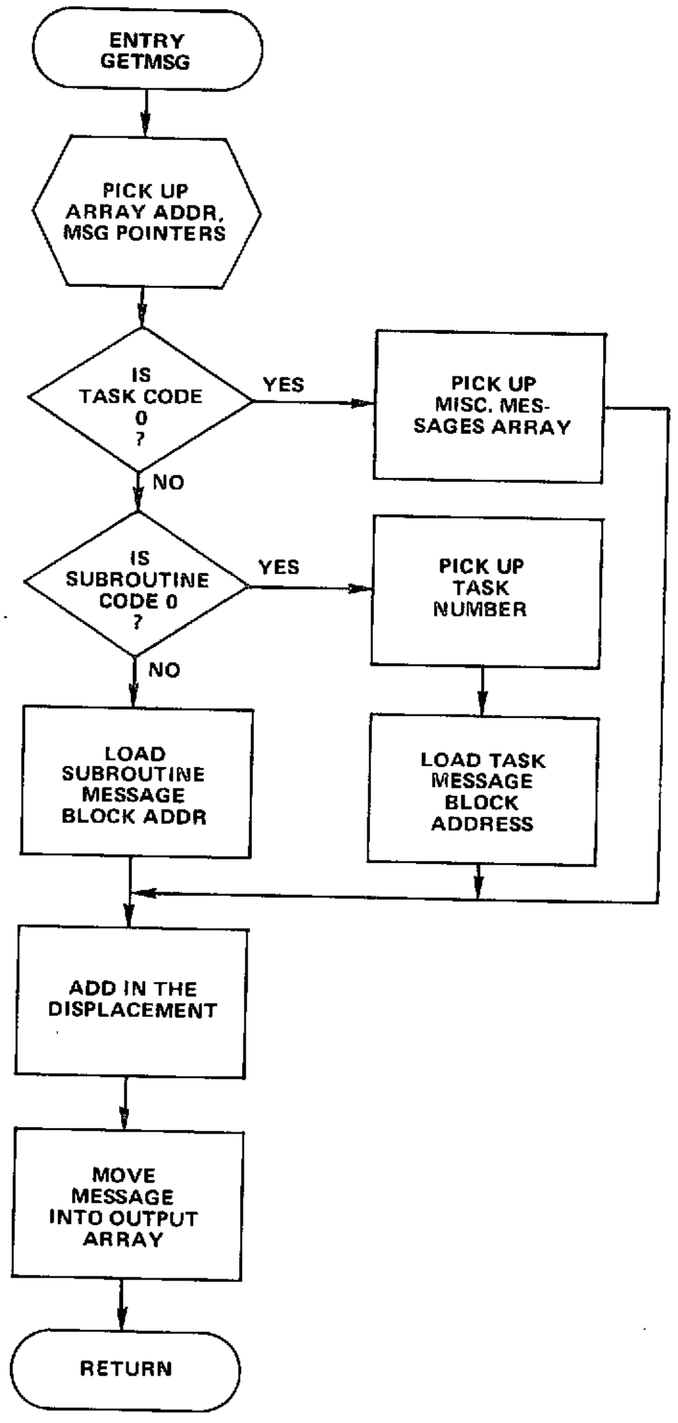

ENTRY GETMSG
PICK UP ARRAY ADDR, MSG POINTERS
IS TASK CODE 0 ?
YES
PICK UP MISC. MESSAGES ARRAY
NO
IS SUBROUTINE CODE 0 ?
YES
PICK UP TASK NUMBER
NO
LOAD SUBROUTINE MESSAGE BLOCK ADDR
LOAD TASK MESSAGE BLOCK ADDRESS
ADD IN THE DISPLACEMENT
MOVE MESSAGE INTO OUTPUT ARRAY
RETURN

Figure C-12. ERROR Program Flowchart (2 of 2)

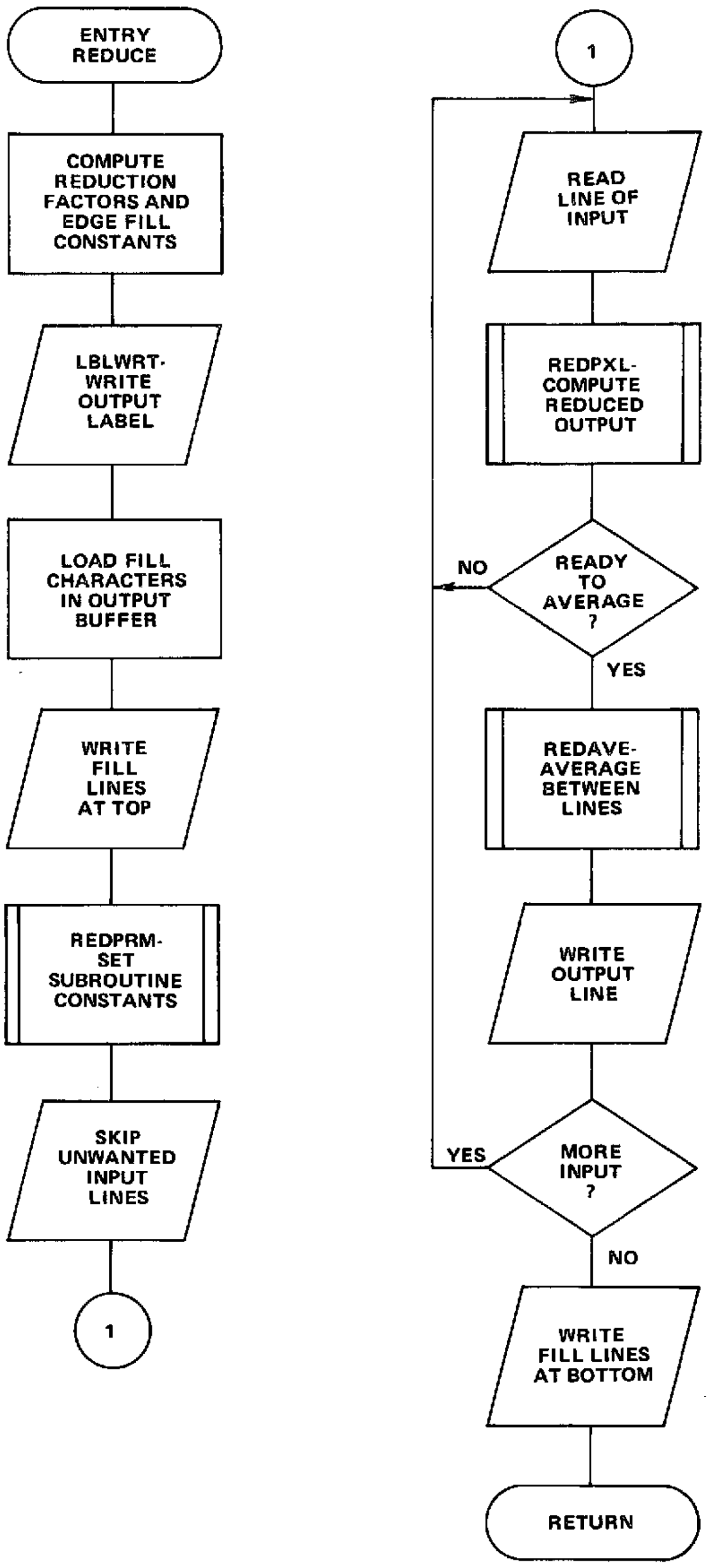

ENTRY REDUCE
COMPUTE REDUCTION FACTORS AND EDGE FILL CONSTANTS
LBLWRT-WRITE OUTPUT LABEL
LOAD FILL CHARACTERS IN OUTPUT BUFFER
WRITE FILL LINES AT TOP
REDPRM-SET SUBROUTINE CONSTANTS
SKIP UNWANTED INPUT LINES
1
1
READ LINE OF INPUT
REDPXL-COMPUTE REDUCED OUTPUT
READY TO AVERAGE ?
NO
YES
REDAVE-AVERAGE BETWEEN LINES
WRITE OUTPUT LINE
MORE INPUT ?
YES
NO
WRITE FILL LINES AT BOTTOM
RETURN

Figure C-13. REDUCE Program Flowchart (1 of 2)

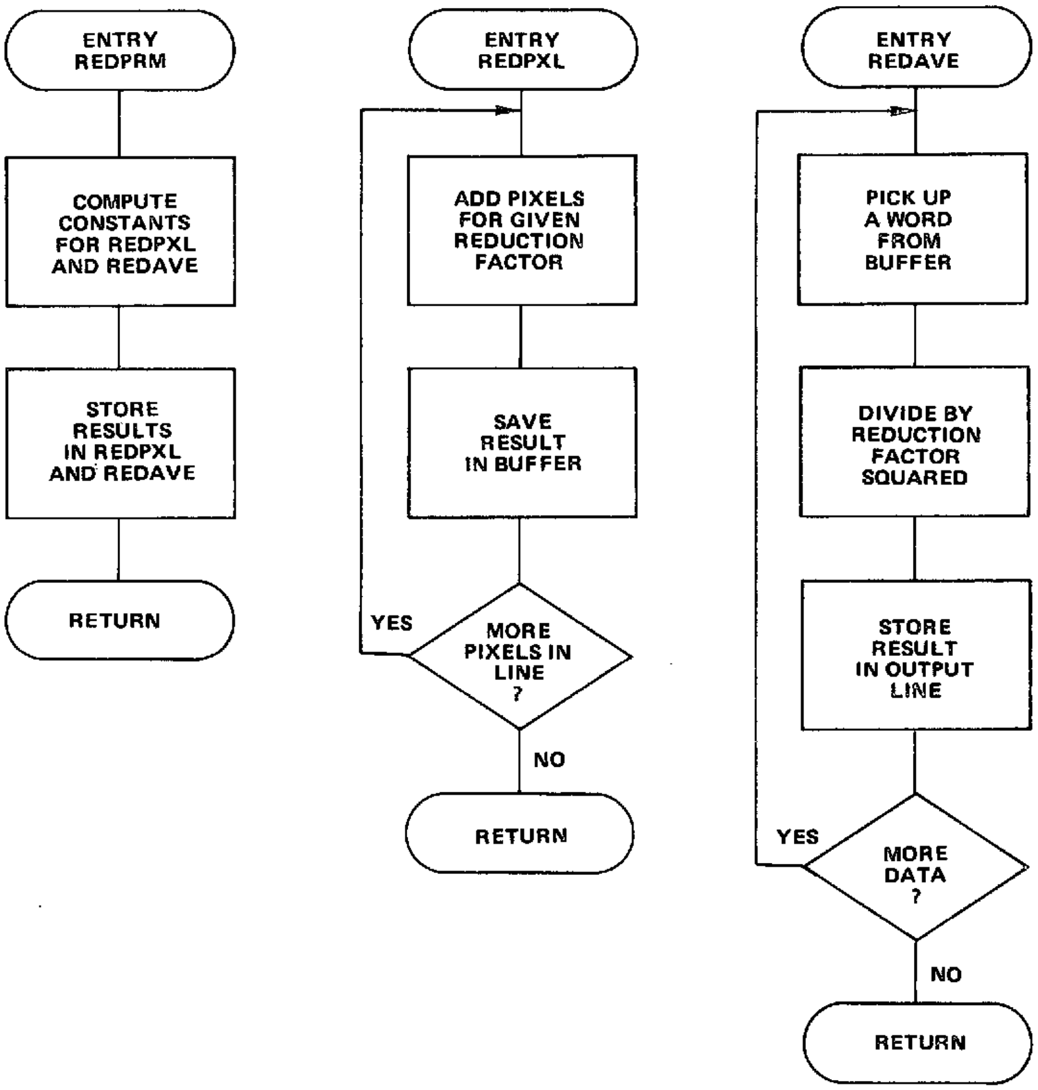

ENTRY REDPRM
COMPUTE CONSTANTS FOR REDPXL AND REDAVE
STORE RESULTS IN REDPXL AND REDAVE
RETURN
ENTRY REDPXL
ADD PIXELS FOR GIVEN REDUCTION FACTOR
SAVE RESULT IN BUFFER
MORE PIXELS IN LINE ?
YES
NO
RETURN
ENTRY REDAVE
PICK UP A WORD FROM BUFFER
DIVIDE BY REDUCTION FACTOR SQUARED
STORE RESULT IN OUTPUT LINE
MORE DATA ?
YES
NO
RETURN

Figure C-13. REDUCE Program Flowchart (2 of 2)

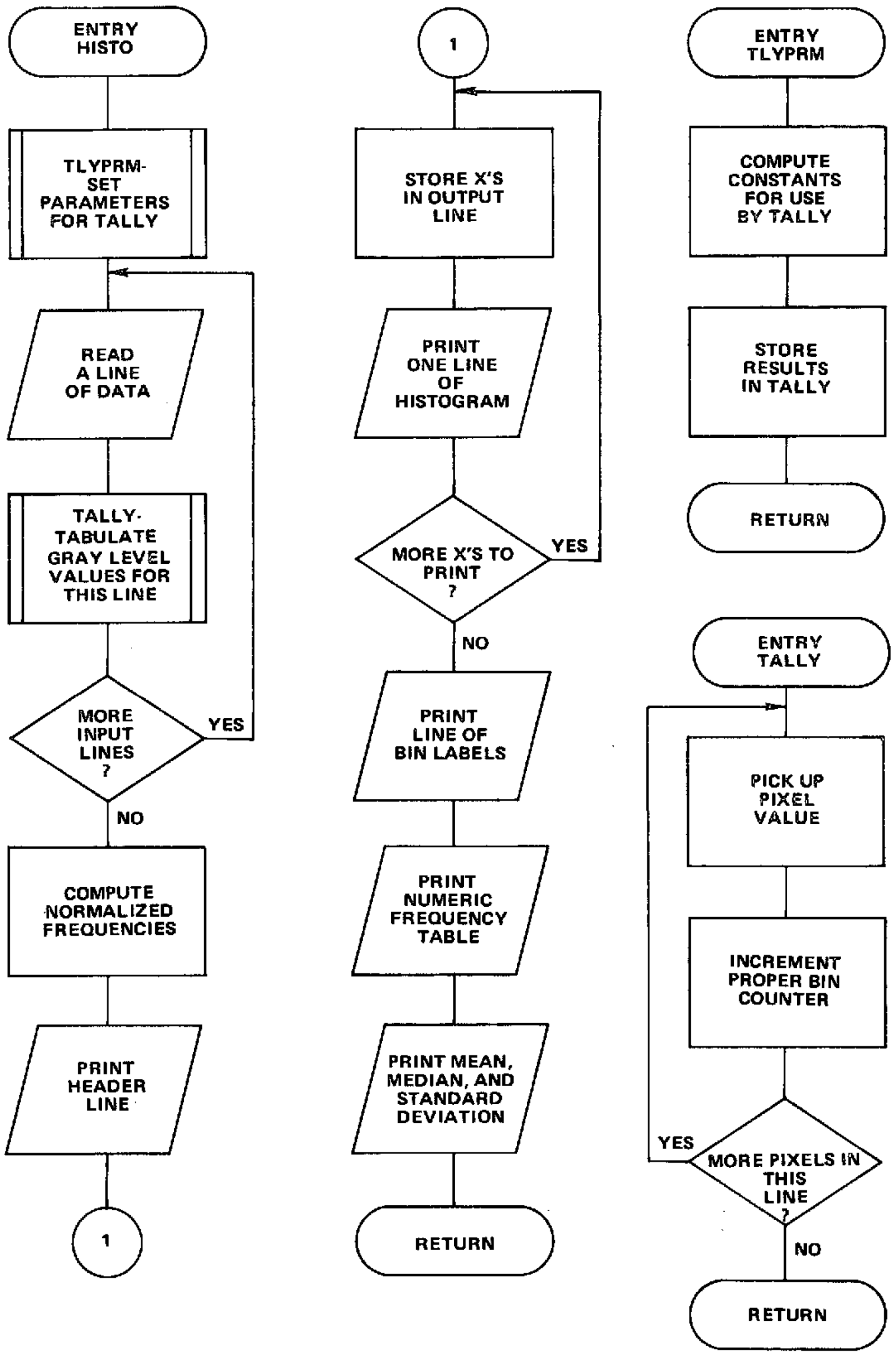

ENTRY HISTO
TLYPRM-SET PARAMETERS FOR TALLY
READ A LINE OF DATA
TALLY-TABULATE GRAY LEVEL VALUES FOR THIS LINE
MORE INPUT LINES ?
YES
NO
COMPUTE NORMALIZED FREQUENCIES
PRINT HEADER LINE
1
1
STORE X'S IN OUTPUT LINE
PRINT ONE LINE OF HISTOGRAM
MORE X'S TO PRINT ?
YES
NO
PRINT LINE OF BIN LABELS
PRINT NUMERIC FREQUENCY TABLE
PRINT MEAN, MEDIAN, AND STANDARD DEVIATION
RETURN
ENTRY TLYPRM
COMPUTE CONSTANTS FOR USE BY TALLY
STORE RESULTS IN TALLY
RETURN
ENTRY TALLY
PICK UP PIXEL VALUE
INCREMENT PROPER BIN COUNTER
MORE PIXELS IN THIS LINE ?
YES
NO
RETURN

Figure C-14. HISTO Program Flowchart

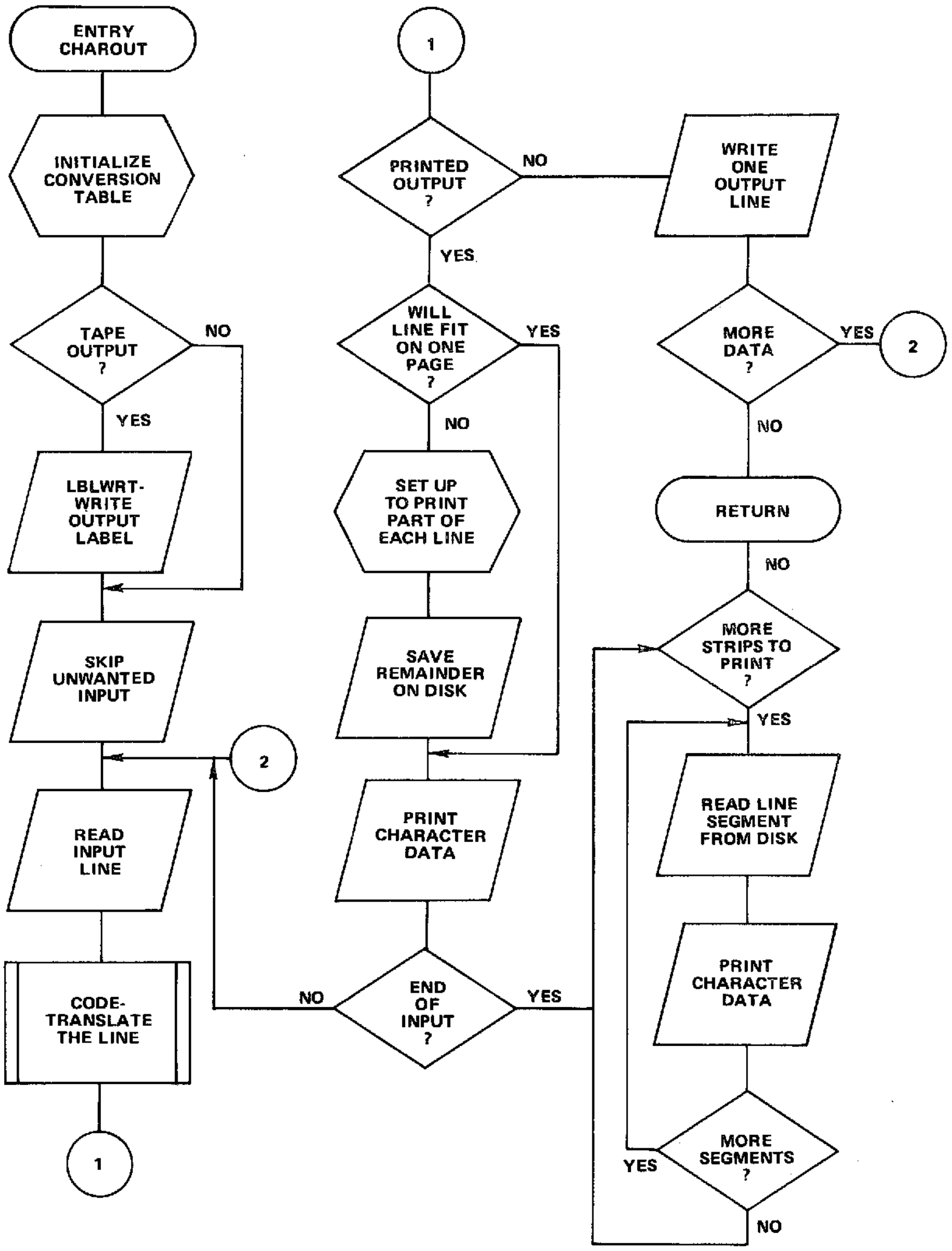


Figure C-15. CHAROUT Program Flowchart

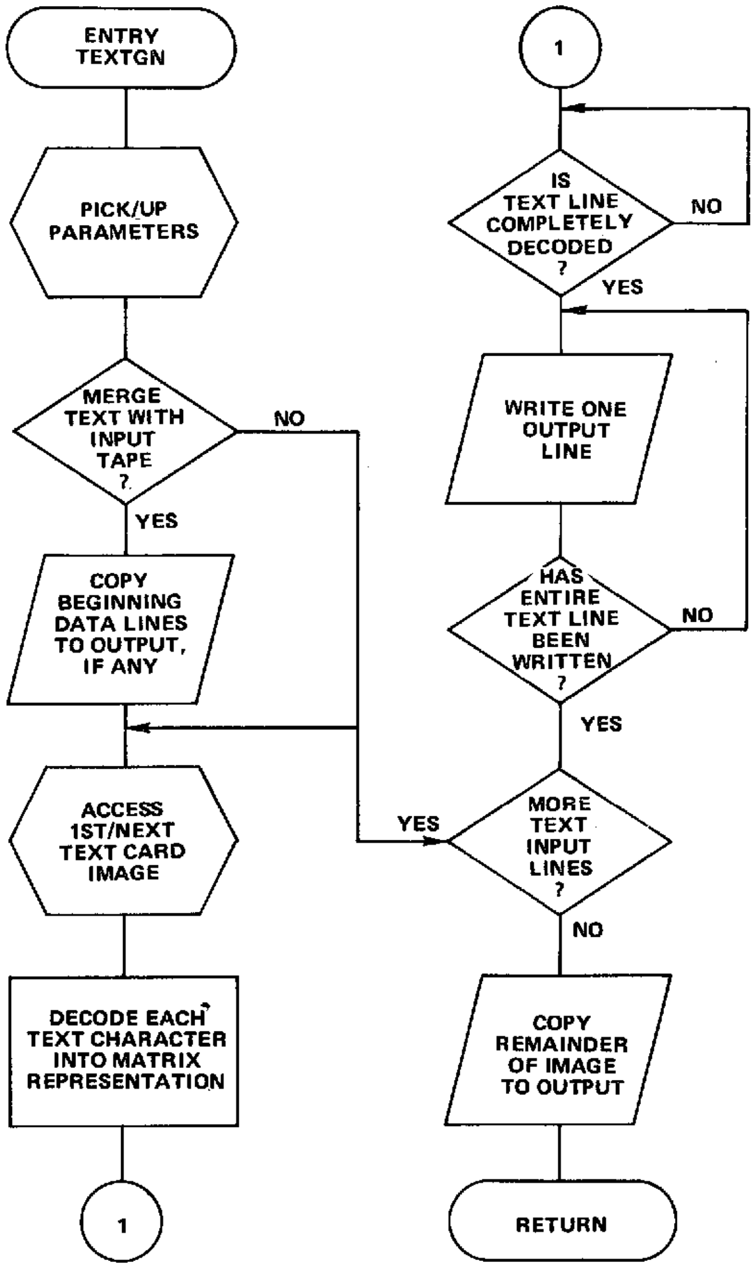

ENTRY TEXTGN
PICK/UP PARAMETERS
MERGE TEXT WITH INPUT TAPE ?
NO
YES
COPY BEGINNING DATA LINES TO OUTPUT, IF ANY
ACCESS 1ST/NEXT TEXT CARD IMAGE
DECODE EACH TEXT CHARACTER INTO MATRIX REPRESENTATION
1
1
IS TEXT LINE COMPLETELY DECODED ?
NO
YES
WRITE ONE OUTPUT LINE
HAS ENTIRE TEXT LINE BEEN WRITTEN ?
NO
YES
MORE TEXT INPUT LINES ?
YES
NO
COPY REMAINDER OF IMAGE TO OUTPUT
RETURN

Figure C-16. TEXTGN Program Flowchart

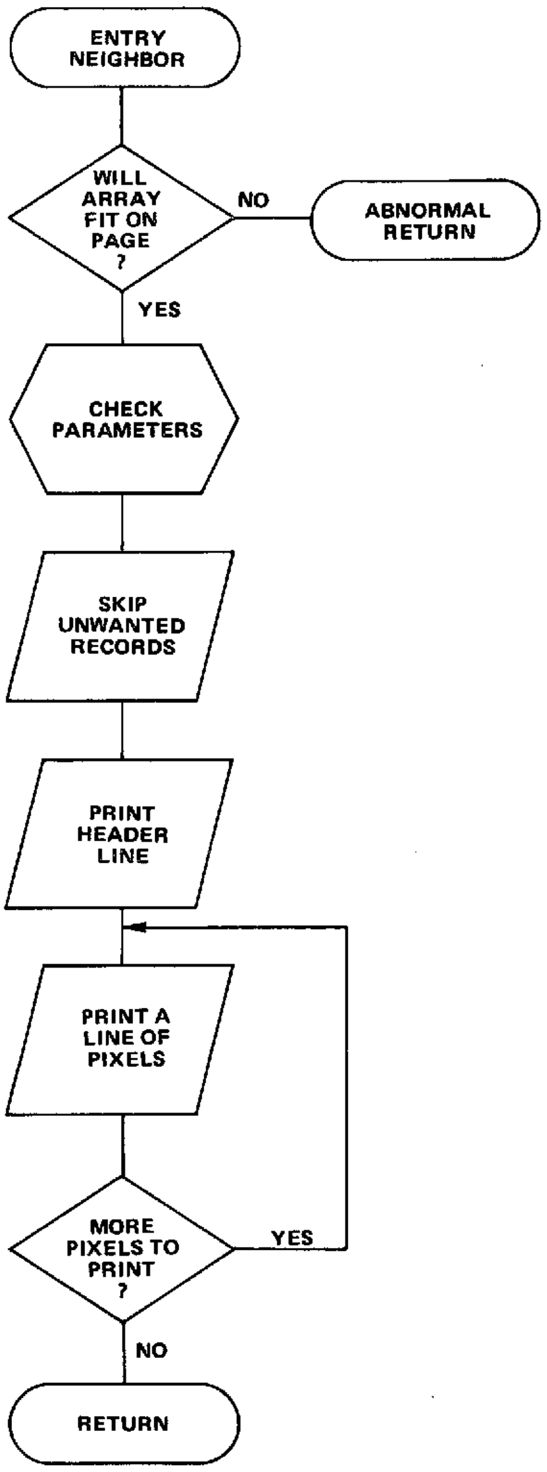
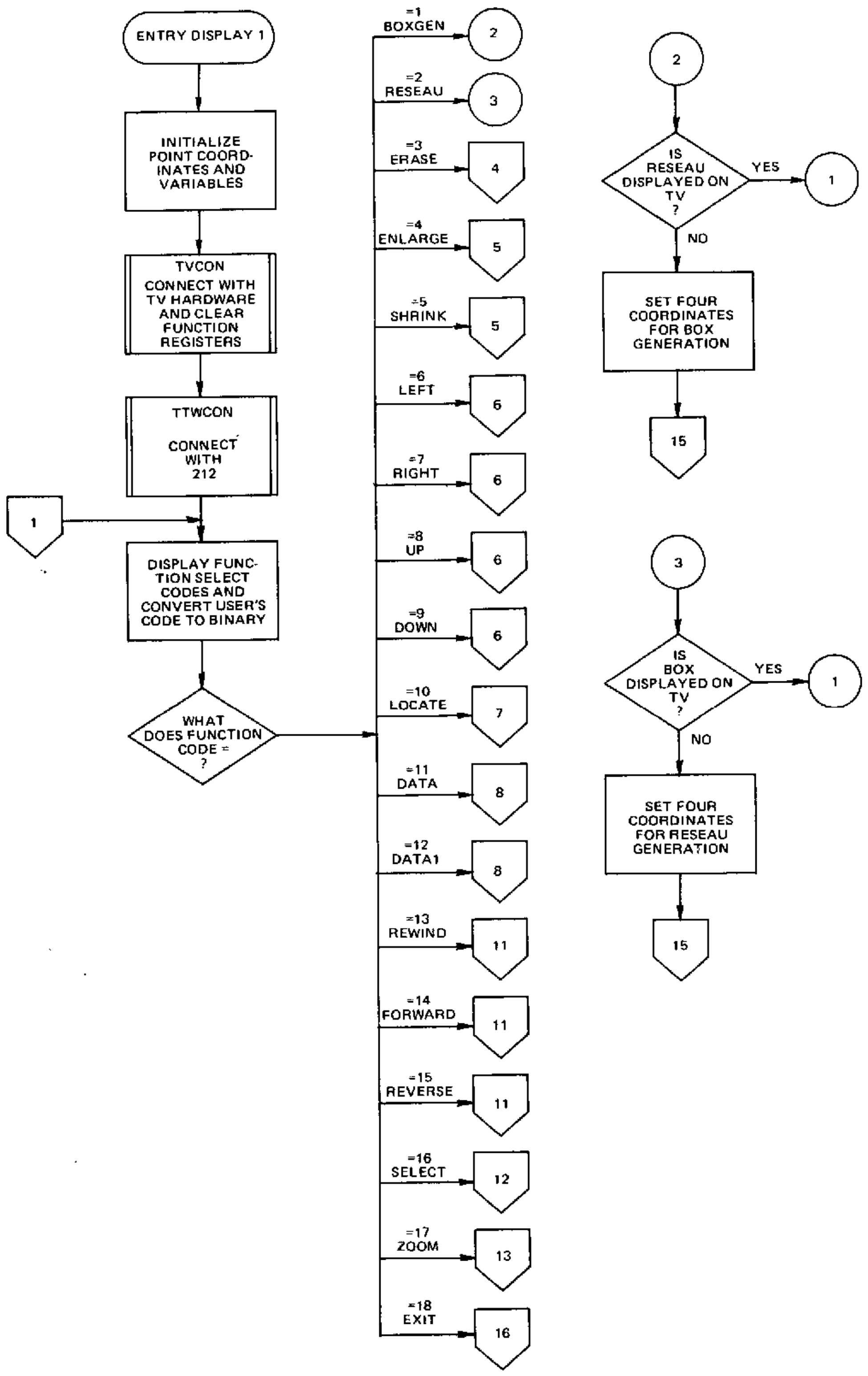
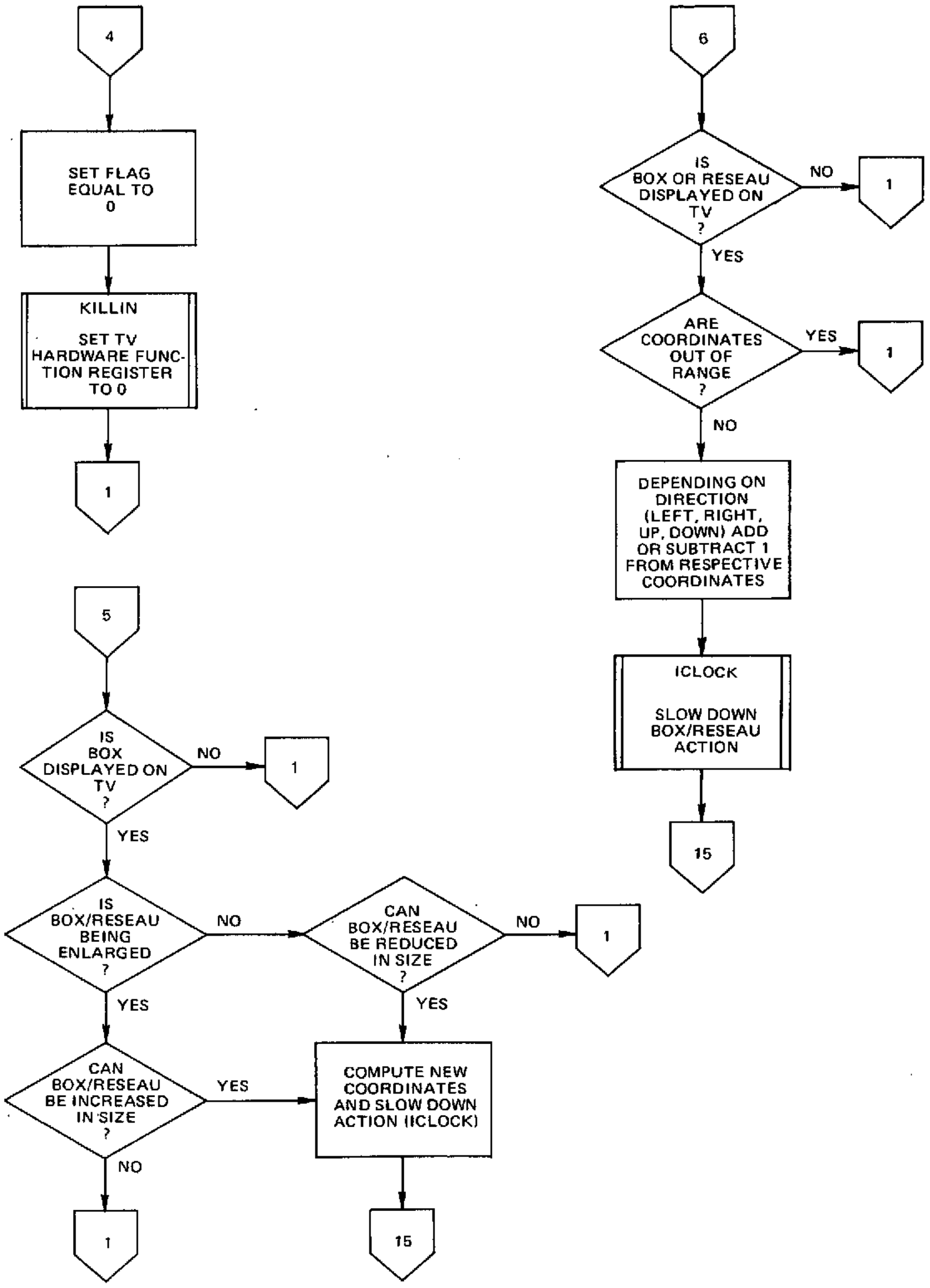


Figure C-17. NEIGHBOR Program Flowchart



ENTRY DISPLAY 1
INITIALIZE POINT COORDINATES AND VARIABLES
TVCON
CONNECT WITH TV HARDWARE AND CLEAR FUNCTION REGISTERS
TTWCON
CONNECT WITH 212
1
DISPLAY FUNCTION SELECT CODES AND CONVERT USER'S CODE TO BINARY
WHAT DOES FUNCTION CODE = ?
=1 BOXGEN
2
=2 RESEAU
3
=3 ERASE
4
=4 ENLARGE
5
=5 SHRINK
5
=6 LEFT
6
=7 RIGHT
6
=8 UP
6
=9 DOWN
6
=10 LOCATE
7
=11 DATA
8
=12 DATA1
8
=13 REWIND
11
=14 FORWARD
11
=15 REVERSE
11
=16 SELECT
12
=17 ZOOM
13
=18 EXIT
16
2
IS RESEAU DISPLAYED ON TV ?
YES
1
NO
SET FOUR COORDINATES FOR BOX GENERATION
15
3
IS BOX DISPLAYED ON TV ?
YES
1
NO
SET FOUR COORDINATES FOR RESEAU GENERATION
15

Figure C-18. DISPLAY Program Flowchart (1 of 8)



4
SET FLAG EQUAL TO 0
KILLIN
SET TV HARDWARE FUNCTION REGISTER TO 0
1
5
IS BOX DISPLAYED ON TV ?
NO
1
YES
IS BOX/RESEAU BEING ENLARGED ?
NO
CAN BOX/RESEAU BE REDUCED IN SIZE ?
NO
1
YES
YES
CAN BOX/RESEAU BE INCREASED IN SIZE ?
YES
COMPUTE NEW COORDINATES AND SLOW DOWN ACTION (ICLOCK)
NO
1
15
6
IS BOX OR RESEAU DISPLAYED ON TV ?
NO
1
YES
ARE COORDINATES OUT OF RANGE ?
YES
1
NO
DEPENDING ON DIRECTION (LEFT, RIGHT, UP, DOWN) ADD OR SUBTRACT 1 FROM RESPECTIVE COORDINATES
ICLOCK
SLOW DOWN BOX/RESEAU ACTION
15

Figure C-18. DISPLAY Program Flowchart (2 of 8)

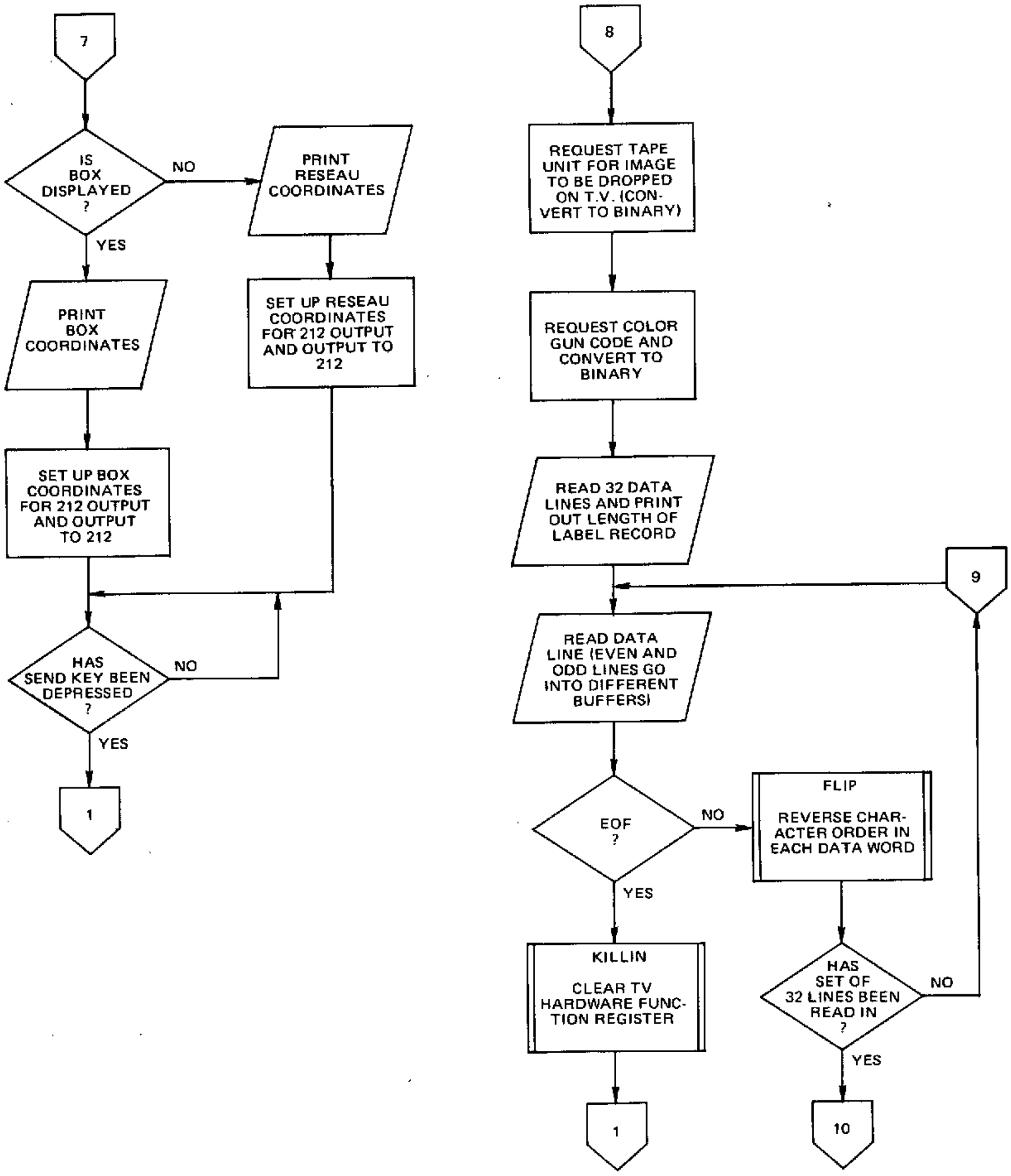

7
IS BOX DISPLAYED ?
NO
YES
PRINT RESEAU COORDINATES
PRINT BOX COORDINATES
SET UP RESEAU COORDINATES FOR 212 OUTPUT AND OUTPUT TO 212
SET UP BOX COORDINATES FOR 212 OUTPUT AND OUTPUT TO 212
HAS SEND KEY BEEN DEPRESSED ?
NO
YES
1
8
REQUEST TAPE UNIT FOR IMAGE TO BE DROPPED ON T.V. (CONVERT TO BINARY)
REQUEST COLOR GUN CODE AND CONVERT TO BINARY
READ 32 DATA LINES AND PRINT OUT LENGTH OF LABEL RECORD
9
READ DATA LINE (EVEN AND ODD LINES GO INTO DIFFERENT BUFFERS)
EOF ?
NO
YES
FLIP
REVERSE CHARACTER ORDER IN EACH DATA WORD
KILLIN
CLEAR TV HARDWARE FUNCTION REGISTER
HAS SET OF 32 LINES BEEN READ IN ?
NO
YES
1
10

Figure C-18. DISPLAY Program Flowchart (3 of 8)

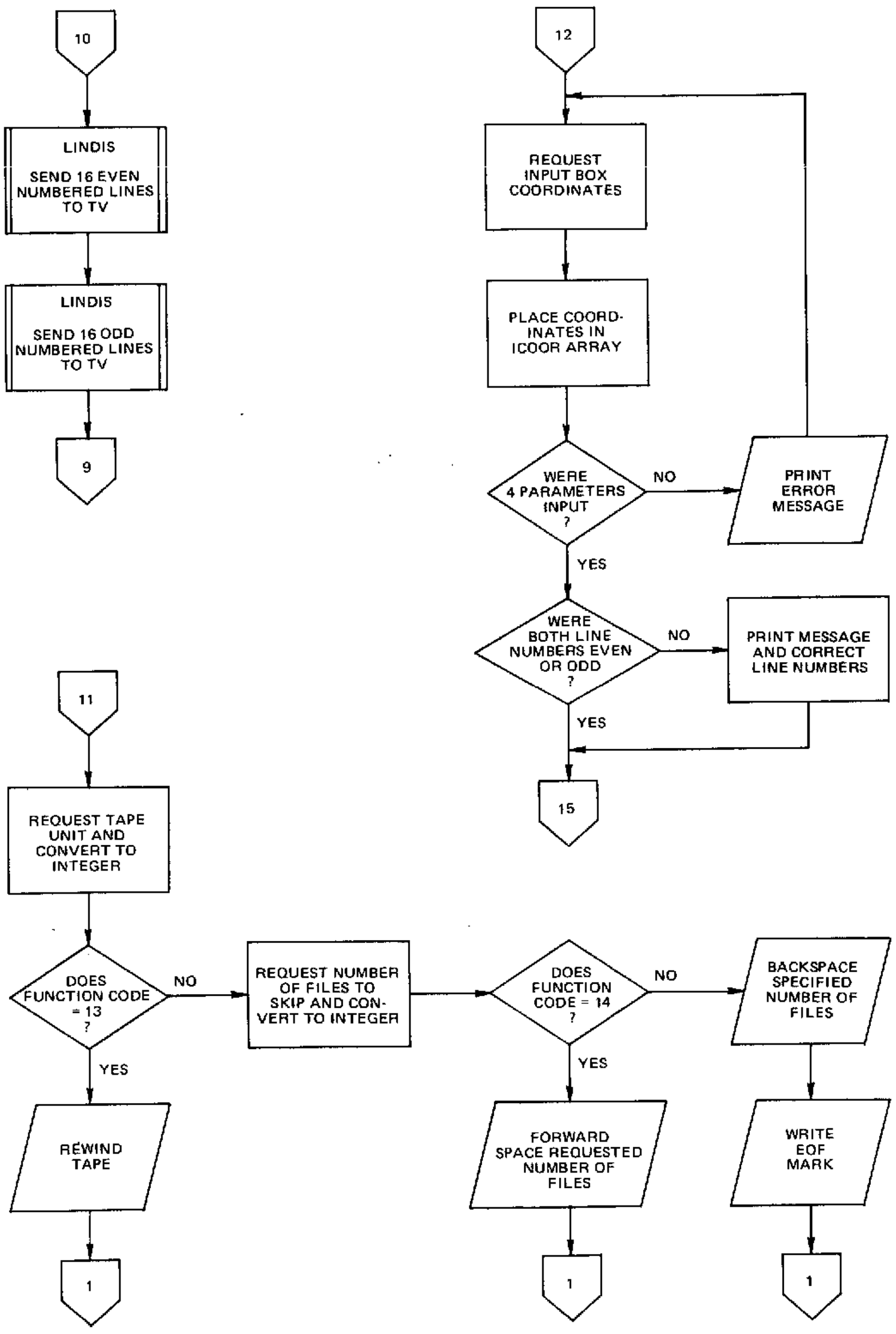
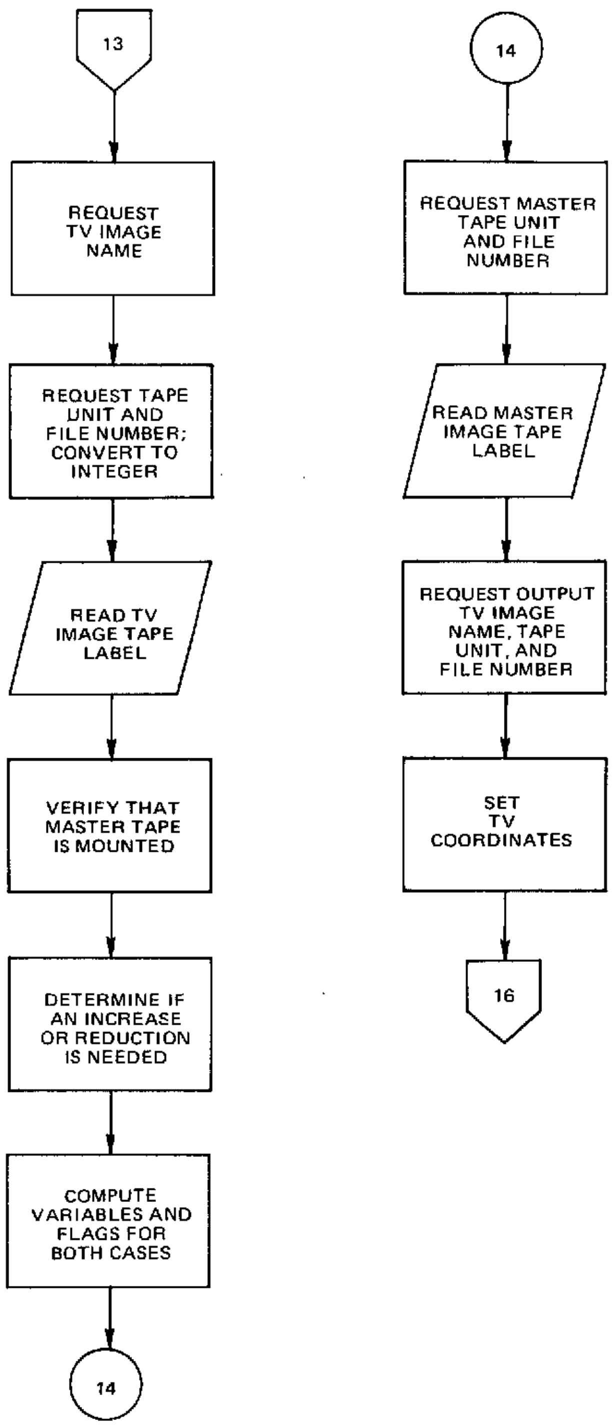

10
LINDIS
SEND 16 EVEN NUMBERED LINES TO TV
LINDIS
SEND 16 ODD NUMBERED LINES TO TV
9
12
REQUEST INPUT BOX COORDINATES
PLACE COORD-INATES IN ICOOR ARRAY
WERE 4 PARAMETERS INPUT ?
NO
PRINT ERROR MESSAGE
YES
WERE BOTH LINE NUMBERS EVEN OR ODD ?
NO
PRINT MESSAGE AND CORRECT LINE NUMBERS
YES
15
11
REQUEST TAPE UNIT AND CONVERT TO INTEGER
DOES FUNCTION CODE = 13 ?
NO
REQUEST NUMBER OF FILES TO SKIP AND CON-VERT TO INTEGER
DOES FUNCTION CODE = 14 ?
NO
BACKSPACE SPECIFIED NUMBER OF FILES
YES
REWIND TAPE
1
YES
FORWARD SPACE REQUESTED NUMBER OF FILES
1
WRITE EOF MARK
1

Figure C-18. DISPLAY Program Flowchart (4 of 8)



13
REQUEST TV IMAGE NAME
REQUEST TAPE UNIT AND FILE NUMBER; CONVERT TO INTEGER
READ TV IMAGE TAPE LABEL
VERIFY THAT MASTER TAPE IS MOUNTED
DETERMINE IF AN INCREASE OR REDUCTION IS NEEDED
COMPUTE VARIABLES AND FLAGS FOR BOTH CASES
14
14
REQUEST MASTER TAPE UNIT AND FILE NUMBER
READ MASTER IMAGE TAPE LABEL
REQUEST OUTPUT TV IMAGE NAME, TAPE UNIT, AND FILE NUMBER
SET TV COORDINATES
16

Figure C-18. DISPLAY Program Flowchart (5 of 8)

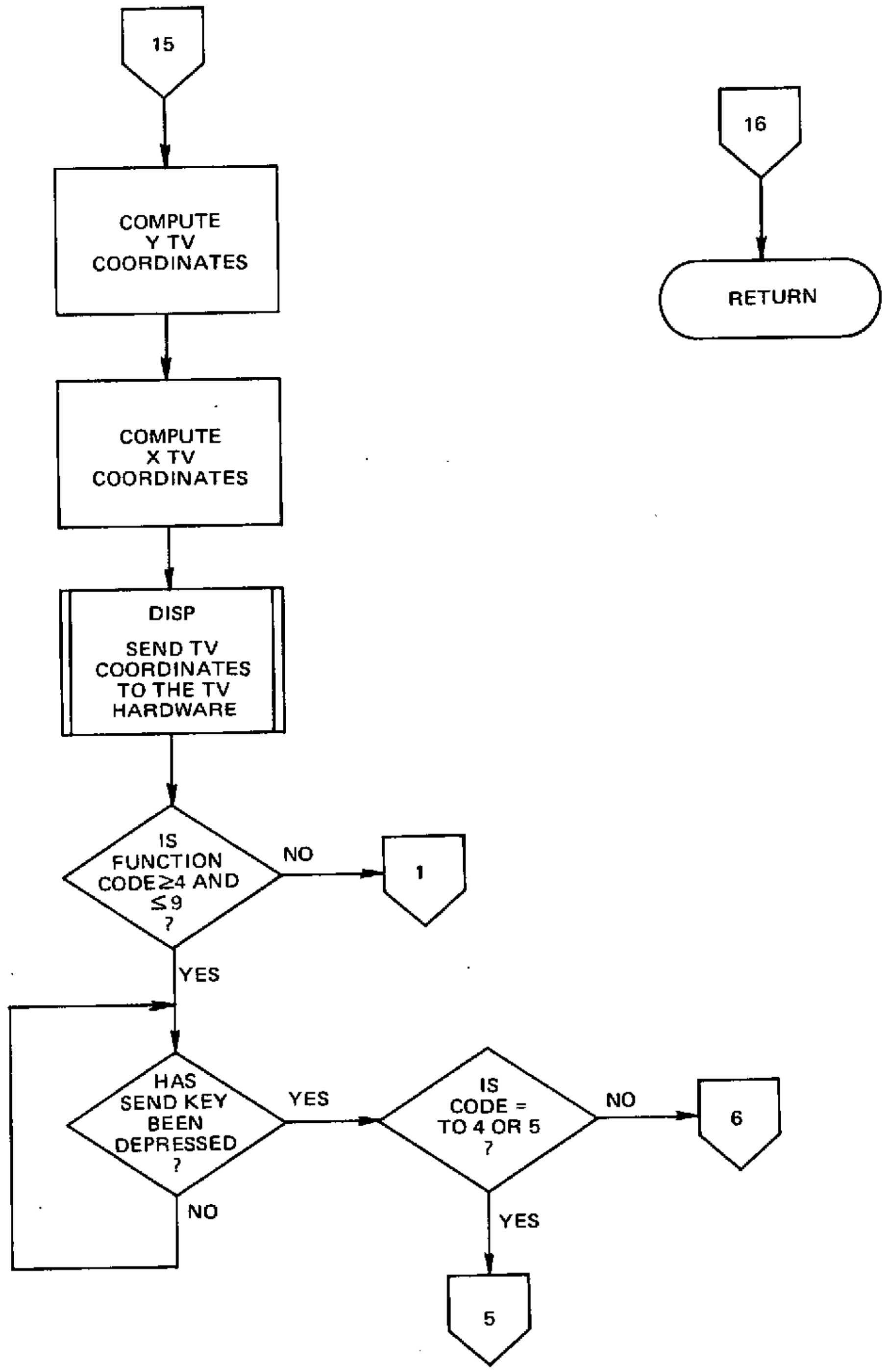


Figure C-18. DISPLAY Program Flowchart (6 of 8)

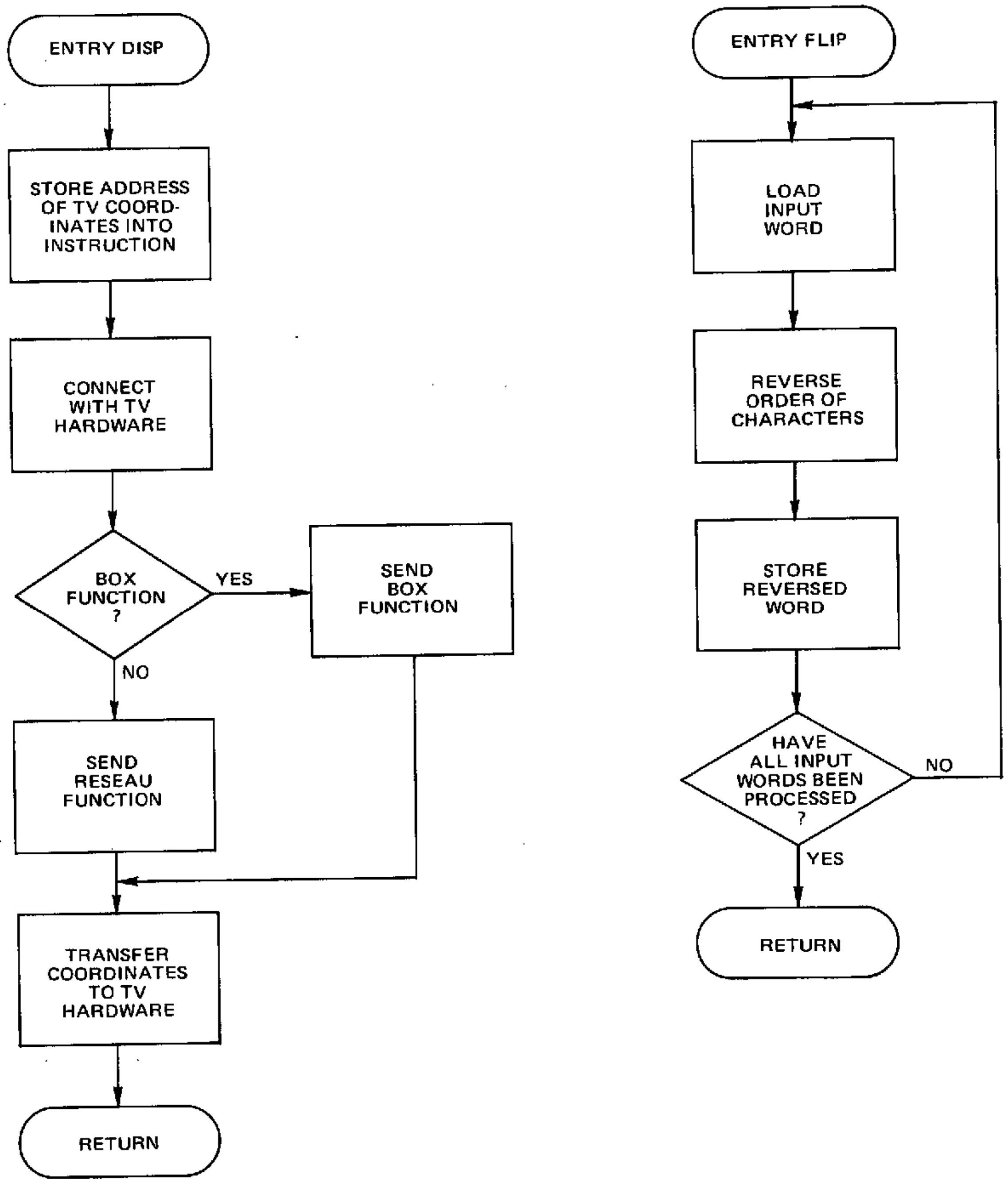


Figure C-18. DISPLAY Program Flowchart (7 of 8)

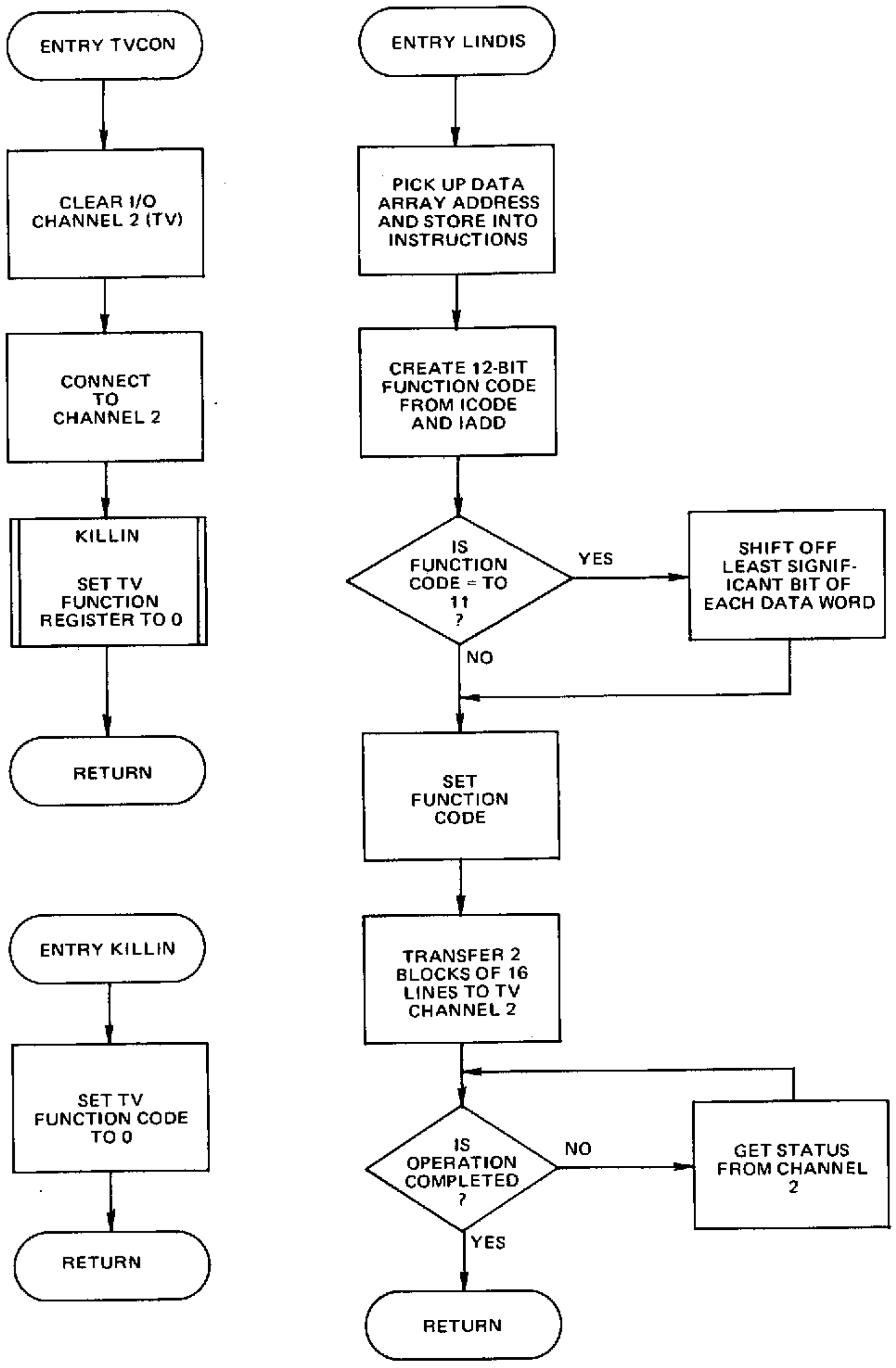

ENTRY TVCON
CLEAR I/O CHANNEL 2 (TV)
CONNECT TO CHANNEL 2
KILLIN
SET TV FUNCTION REGISTER TO 0
RETURN
ENTRY KILLIN
SET TV FUNCTION CODE TO 0
RETURN
ENTRY LINDIS
PICK UP DATA ARRAY ADDRESS AND STORE INTO INSTRUCTIONS
CREATE 12-BIT FUNCTION CODE FROM ICODE AND IADD
IS FUNCTION CODE = TO 11 ?
YES
SHIFT OFF LEAST SIGNIF-ICANT BIT OF EACH DATA WORD
NO
SET FUNCTION CODE
TRANSFER 2 BLOCKS OF 16 LINES TO TV CHANNEL 2
IS OPERATION COMPLETED ?
NO
GET STATUS FROM CHANNEL 2
YES
RETURN

Figure C-18. DISPLAY Program Flowchart (8 of 8)

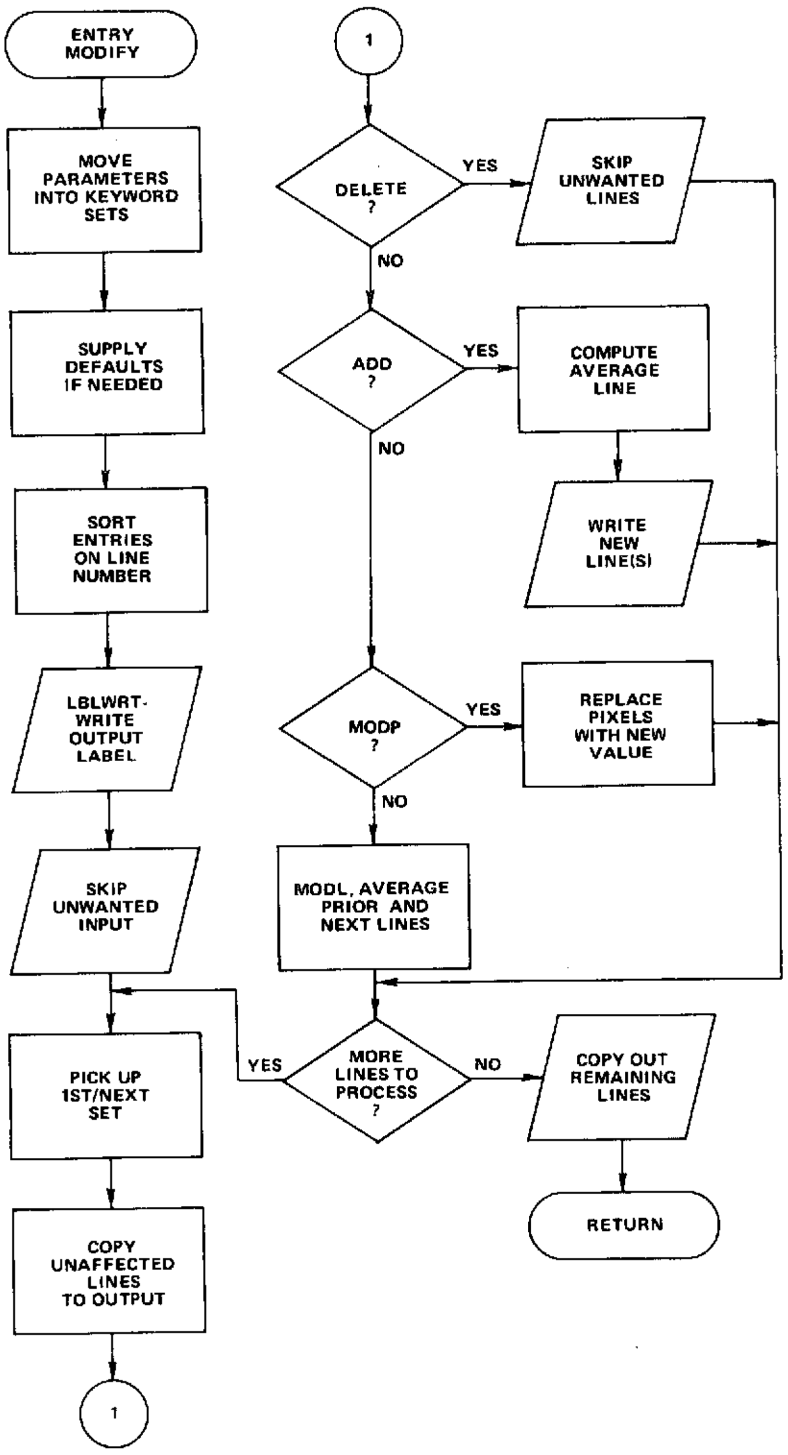

ENTRY MODIFY
MOVE PARAMETERS INTO KEYWORD SETS
SUPPLY DEFAULTS IF NEEDED
SORT ENTRIES ON LINE NUMBER
LBLWRT-WRITE OUTPUT LABEL
SKIP UNWANTED INPUT
PICK UP 1ST/NEXT SET
COPY UNAFFECTED LINES TO OUTPUT
1
1
DELETE ?
YES
SKIP UNWANTED LINES
NO
ADD ?
YES
COMPUTE AVERAGE LINE
WRITE NEW LINE(S)
NO
MODP ?
YES
REPLACE PIXELS WITH NEW VALUE
NO
MODL, AVERAGE PRIOR AND NEXT LINES
MORE LINES TO PROCESS ?
YES
NO
COPY OUT REMAINING LINES
RETURN

Figure C-19. MODIFY Program Flowchart

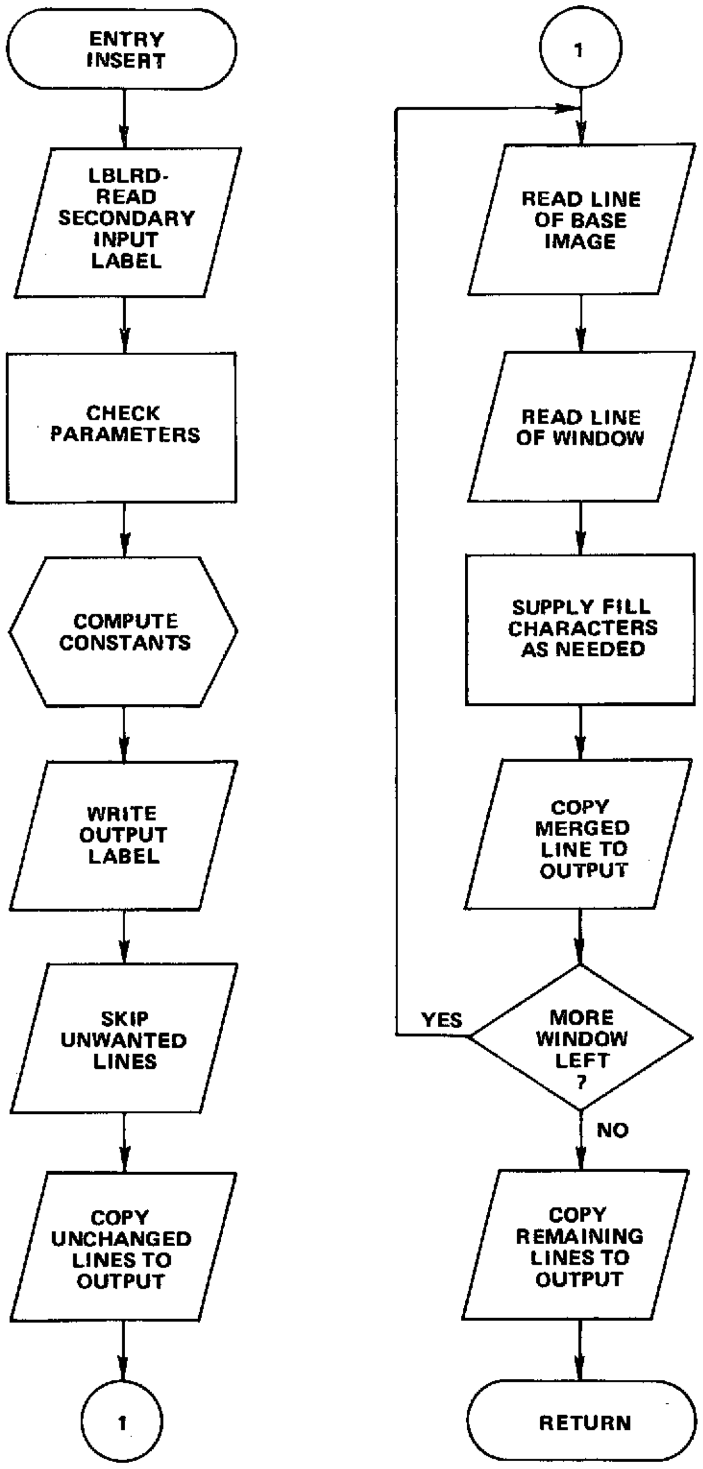


Figure C-20. INSERT Program Flowchart

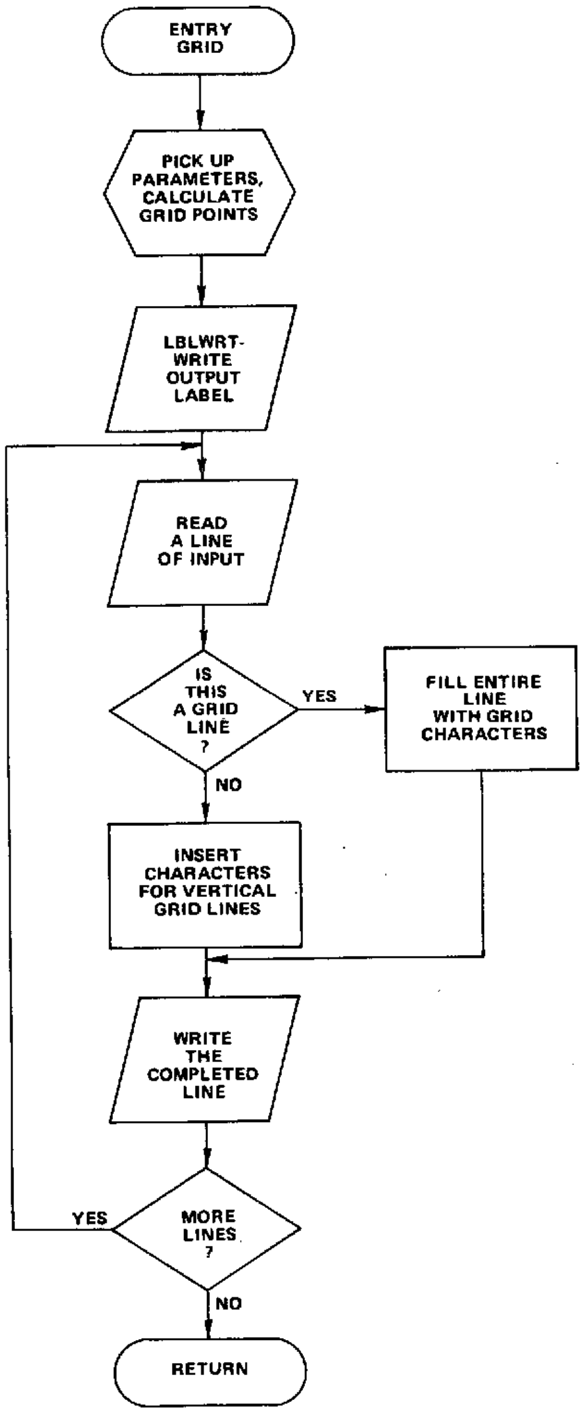

ENTRY GRID
PICK UP PARAMETERS, CALCULATE GRID POINTS
LBLWRT- WRITE OUTPUT LABEL
READ A LINE OF INPUT
IS THIS A GRID LINE ?
YES
FILL ENTIRE LINE WITH GRID CHARACTERS
NO
INSERT CHARACTERS FOR VERTICAL GRID LINES
WRITE THE COMPLETED LINE
YES
MORE LINES ?
NO
RETURN

Figure C-21. GRID Program Flowchart

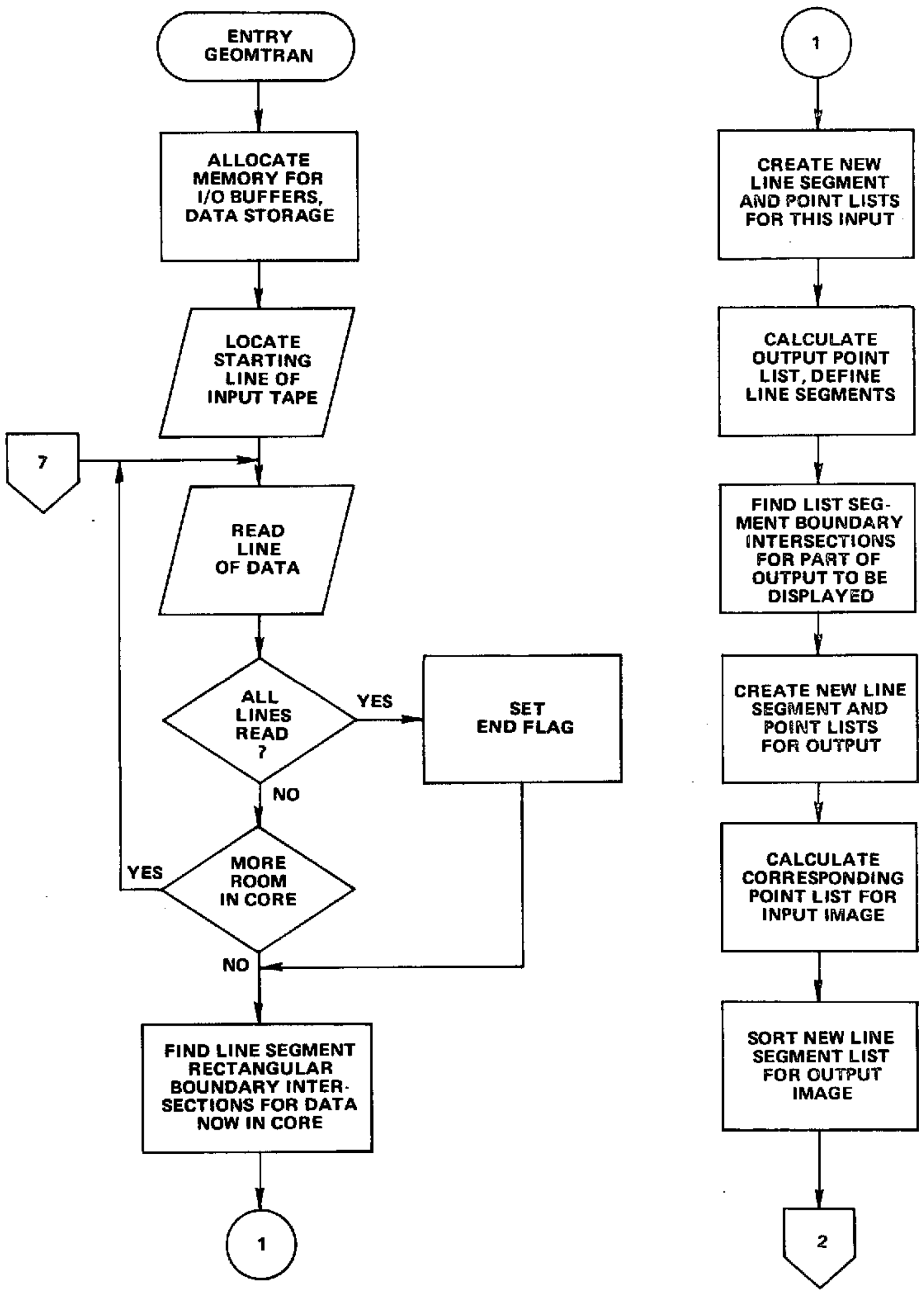

ENTRY GEOMTRAN
ALLOCATE MEMORY FOR I/O BUFFERS, DATA STORAGE
LOCATE STARTING LINE OF INPUT TAPE
7
READ LINE OF DATA
ALL LINES READ ?
YES
SET END FLAG
NO
MORE ROOM IN CORE
YES
NO
FIND LINE SEGMENT RECTANGULAR BOUNDARY INTER-SECTIONS FOR DATA NOW IN CORE
1
1
CREATE NEW LINE SEGMENT AND POINT LISTS FOR THIS INPUT
CALCULATE OUTPUT POINT LIST, DEFINE LINE SEGMENTS
FIND LIST SEG-MENT BOUNDARY INTERSECTIONS FOR PART OF OUTPUT TO BE DISPLAYED
CREATE NEW LINE SEGMENT AND POINT LISTS FOR OUTPUT
CALCULATE CORRESPONDING POINT LIST FOR INPUT IMAGE
SORT NEW LINE SEGMENT LIST FOR OUTPUT IMAGE
2

Figure C-22. GEOMTRAN Program Flowchart (1 of 7)

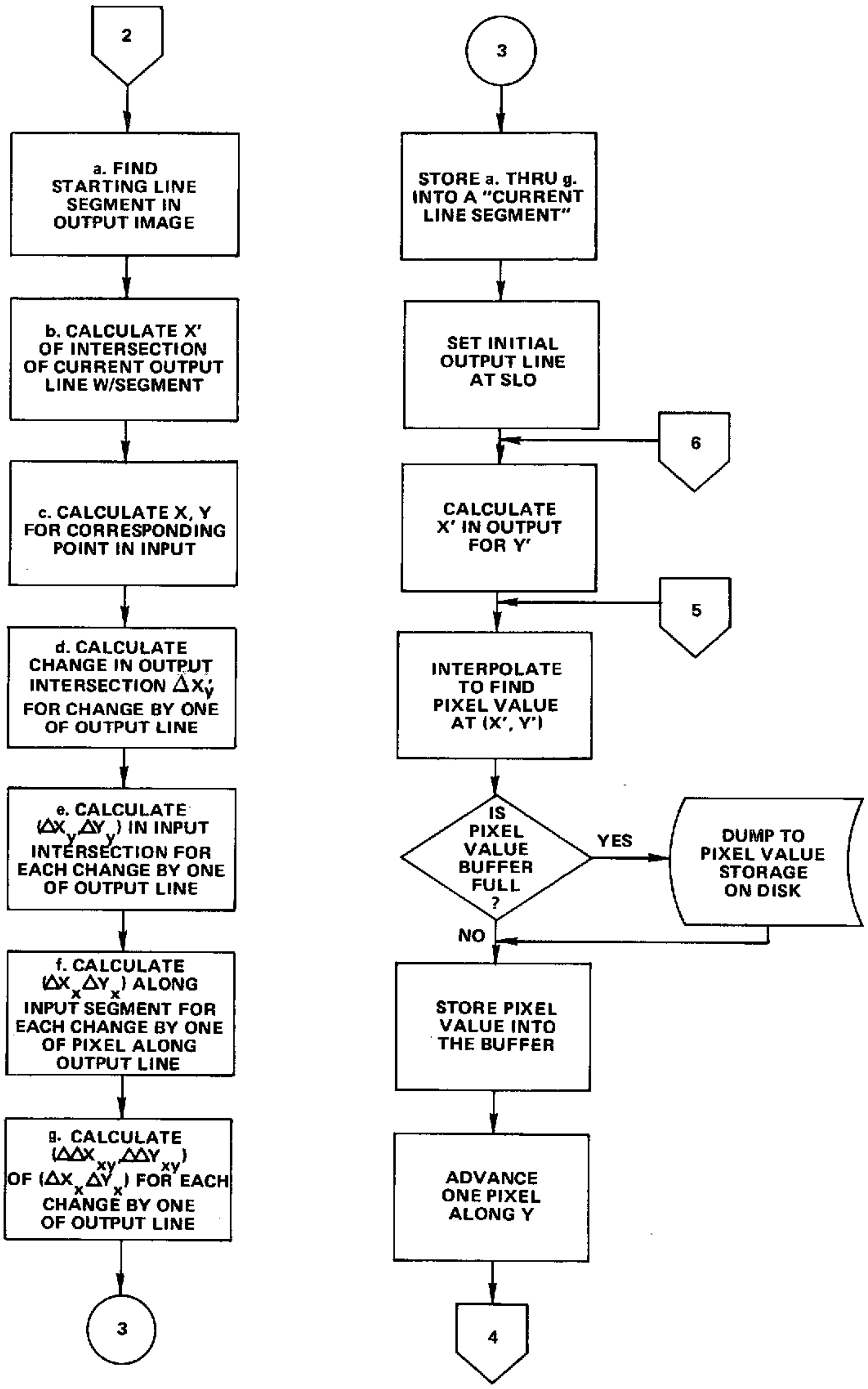

2
a. FIND STARTING LINE SEGMENT IN OUTPUT IMAGE
b. CALCULATE X' OF INTERSECTION OF CURRENT OUTPUT LINE W/SEGMENT
c. CALCULATE X, Y FOR CORRESPONDING POINT IN INPUT
d. CALCULATE CHANGE IN OUTPUT INTERSECTION $\Delta X'_Y$ FOR CHANGE BY ONE OF OUTPUT LINE
e. CALCULATE $(\Delta X_y, \Delta Y_y)$ IN INPUT INTERSECTION FOR EACH CHANGE BY ONE OF OUTPUT LINE
f. CALCULATE $(\Delta X_x, \Delta Y_x)$ ALONG INPUT SEGMENT FOR EACH CHANGE BY ONE OF PIXEL ALONG OUTPUT LINE
g. CALCULATE $(\Delta\Delta X_{xy}, \Delta\Delta Y_{xy})$ OF $(\Delta X_x, \Delta Y_x)$ FOR EACH CHANGE BY ONE OF OUTPUT LINE
3
3
STORE a. THRU g. INTO A "CURRENT LINE SEGMENT"
SET INITIAL OUTPUT LINE AT SLO
6
CALCULATE X' IN OUTPUT FOR Y'
5
INTERPOLATE TO FIND PIXEL VALUE AT (X', Y')
IS PIXEL VALUE BUFFER FULL ?
YES
DUMP TO PIXEL VALUE STORAGE ON DISK
NO
STORE PIXEL VALUE INTO THE BUFFER
ADVANCE ONE PIXEL ALONG Y
4

Figure C-22. GEOMTRAN Program Flowchart (2 of 7)

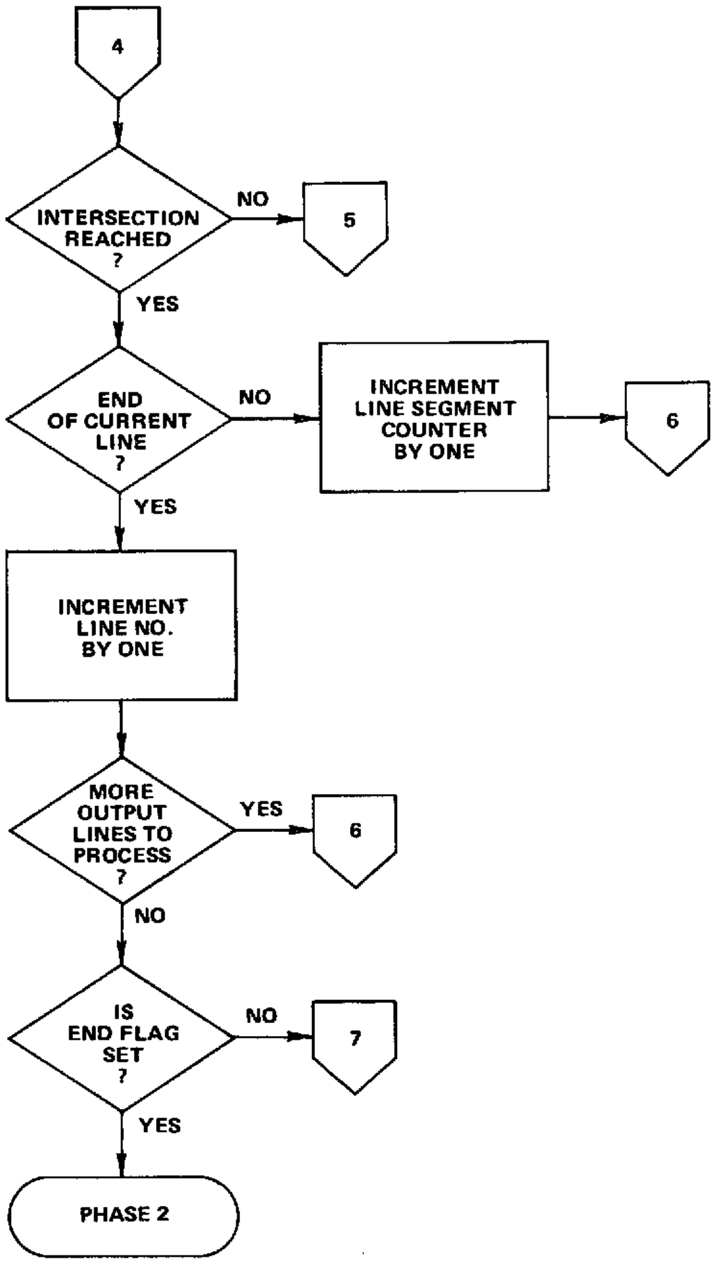

4
INTERSECTION REACHED ?
NO
5
YES
END OF CURRENT LINE ?
NO
INCREMENT LINE SEGMENT COUNTER BY ONE
6
YES
INCREMENT LINE NO. BY ONE
MORE OUTPUT LINES TO PROCESS ?
YES
6
NO
IS END FLAG SET ?
NO
7
YES
PHASE 2

Figure C-22. GEOMTRAN Program Flowchart (3 of 7)

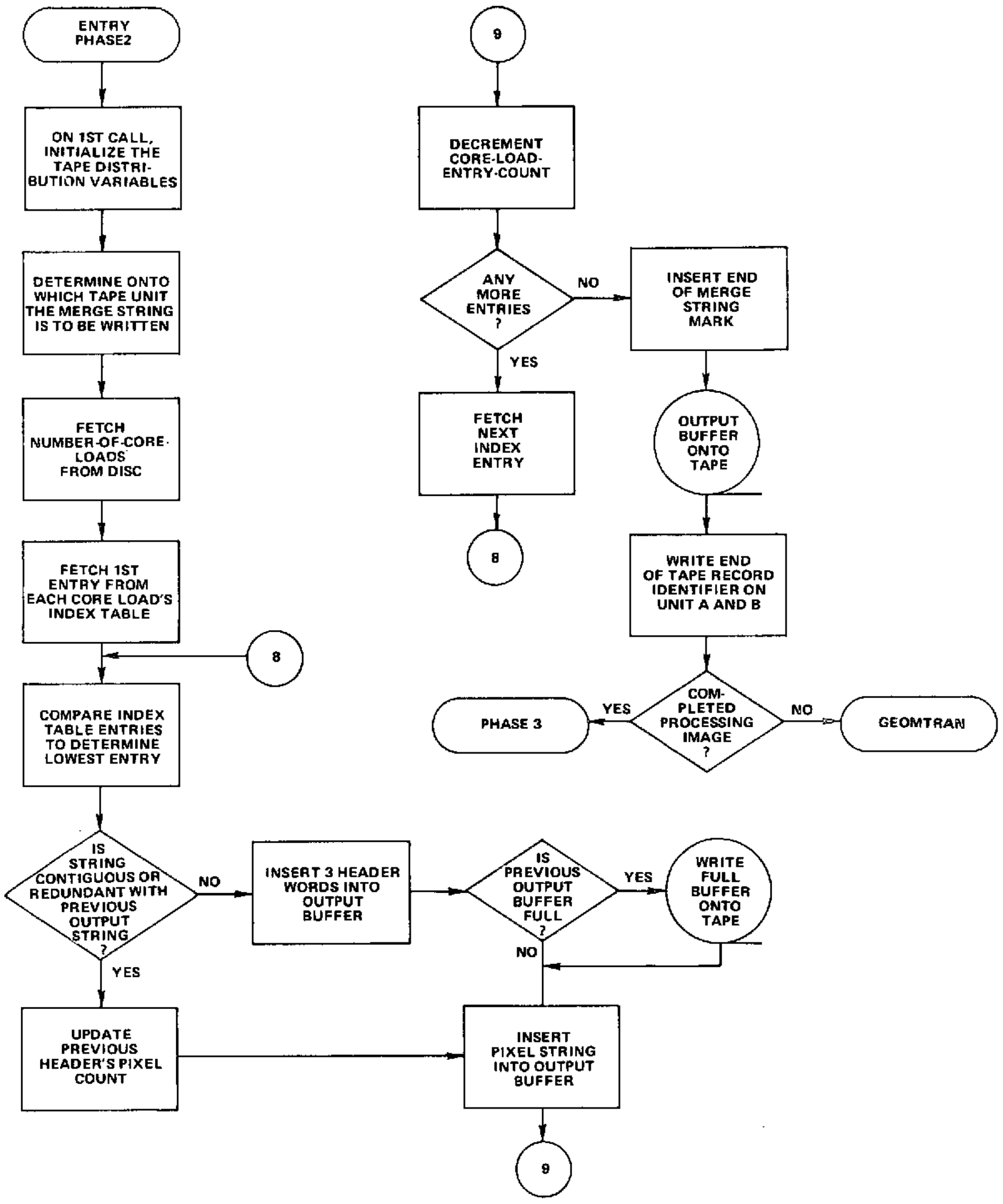
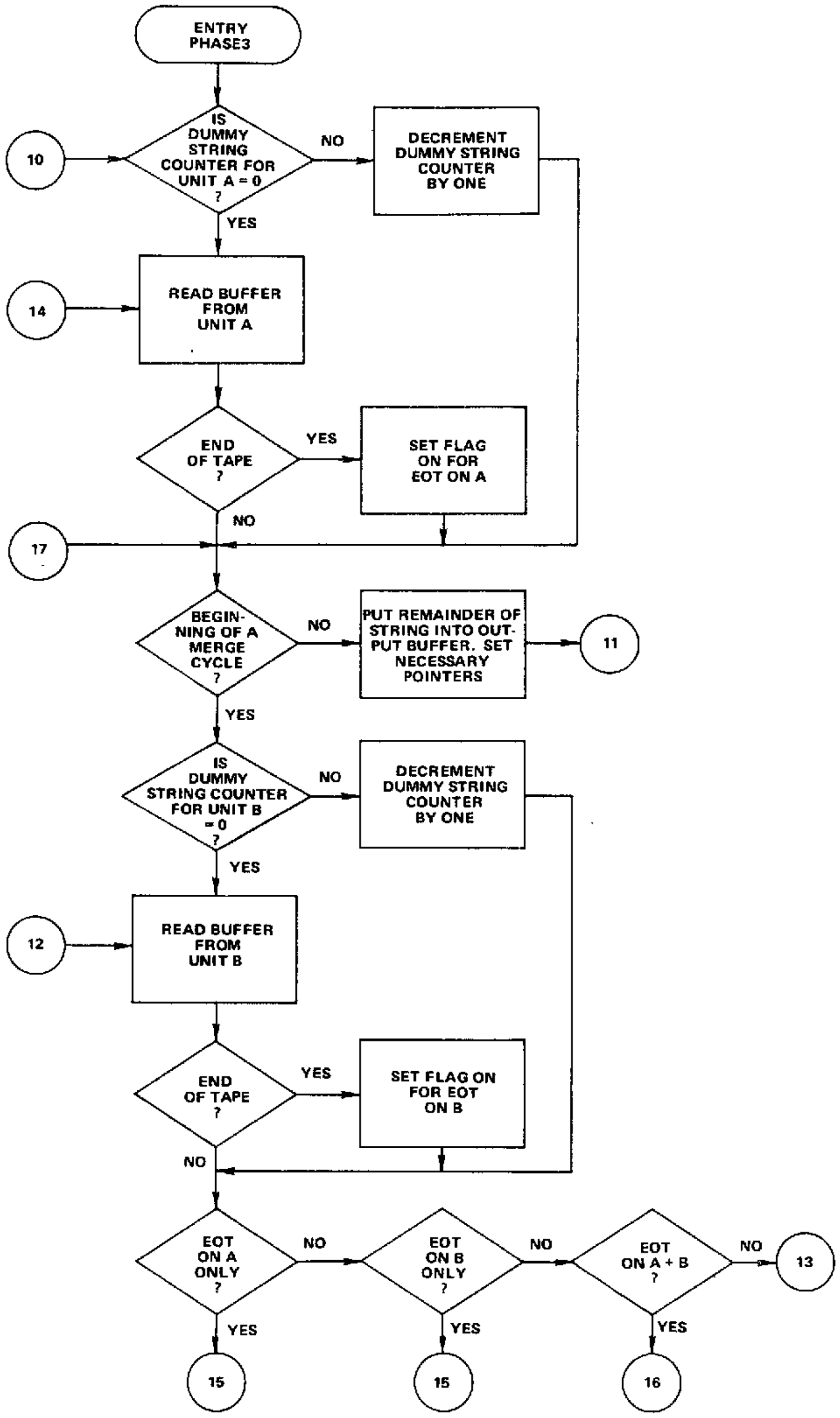

ENTRY PHASE2
ON 1ST CALL, INITIALIZE THE TAPE DISTRIBUTION VARIABLES
DETERMINE ONTO WHICH TAPE UNIT THE MERGE STRING IS TO BE WRITTEN
FETCH NUMBER-OF-CORE-LOADS FROM DISC
FETCH 1ST ENTRY FROM EACH CORE LOAD'S INDEX TABLE
8
COMPARE INDEX TABLE ENTRIES TO DETERMINE LOWEST ENTRY
IS STRING CONTIGUOUS OR REDUNDANT WITH PREVIOUS OUTPUT STRING ?
NO
YES
UPDATE PREVIOUS HEADER'S PIXEL COUNT
INSERT 3 HEADER WORDS INTO OUTPUT BUFFER
IS PREVIOUS OUTPUT BUFFER FULL ?
YES
NO
WRITE FULL BUFFER ONTO TAPE
INSERT PIXEL STRING INTO OUTPUT BUFFER
9
9
DECREMENT CORE-LOAD-ENTRY-COUNT
ANY MORE ENTRIES ?
NO
YES
FETCH NEXT INDEX ENTRY
8
INSERT END OF MERGE STRING MARK
OUTPUT BUFFER ONTO TAPE
WRITE END OF TAPE RECORD IDENTIFIER ON UNIT A AND B
COMPLETED PROCESSING IMAGE ?
YES
NO
PHASE 3
GEOMTRAN

Figure C-22. GEOMTRAN Program Flowchart (4 of 7)



ENTRY PHASE3
10
IS DUMMY STRING COUNTER FOR UNIT A = 0 ?
NO
DECREMENT DUMMY STRING COUNTER BY ONE
YES
14
READ BUFFER FROM UNIT A
END OF TAPE ?
YES
SET FLAG ON FOR EOT ON A
NO
17
BEGIN-NING OF A MERGE CYCLE ?
NO
PUT REMAINDER OF STRING INTO OUT-PUT BUFFER. SET NECESSARY POINTERS
11
YES
IS DUMMY STRING COUNTER FOR UNIT B = 0 ?
NO
DECREMENT DUMMY STRING COUNTER BY ONE
YES
12
READ BUFFER FROM UNIT B
END OF TAPE ?
YES
SET FLAG ON FOR EOT ON B
NO
EOT ON A ONLY ?
NO
EOT ON B ONLY ?
NO
EOT ON A + B ?
NO
13
YES
15
YES
15
YES
16

Figure C-22. GEOMTRAN Program Flowchart (5 of 7)

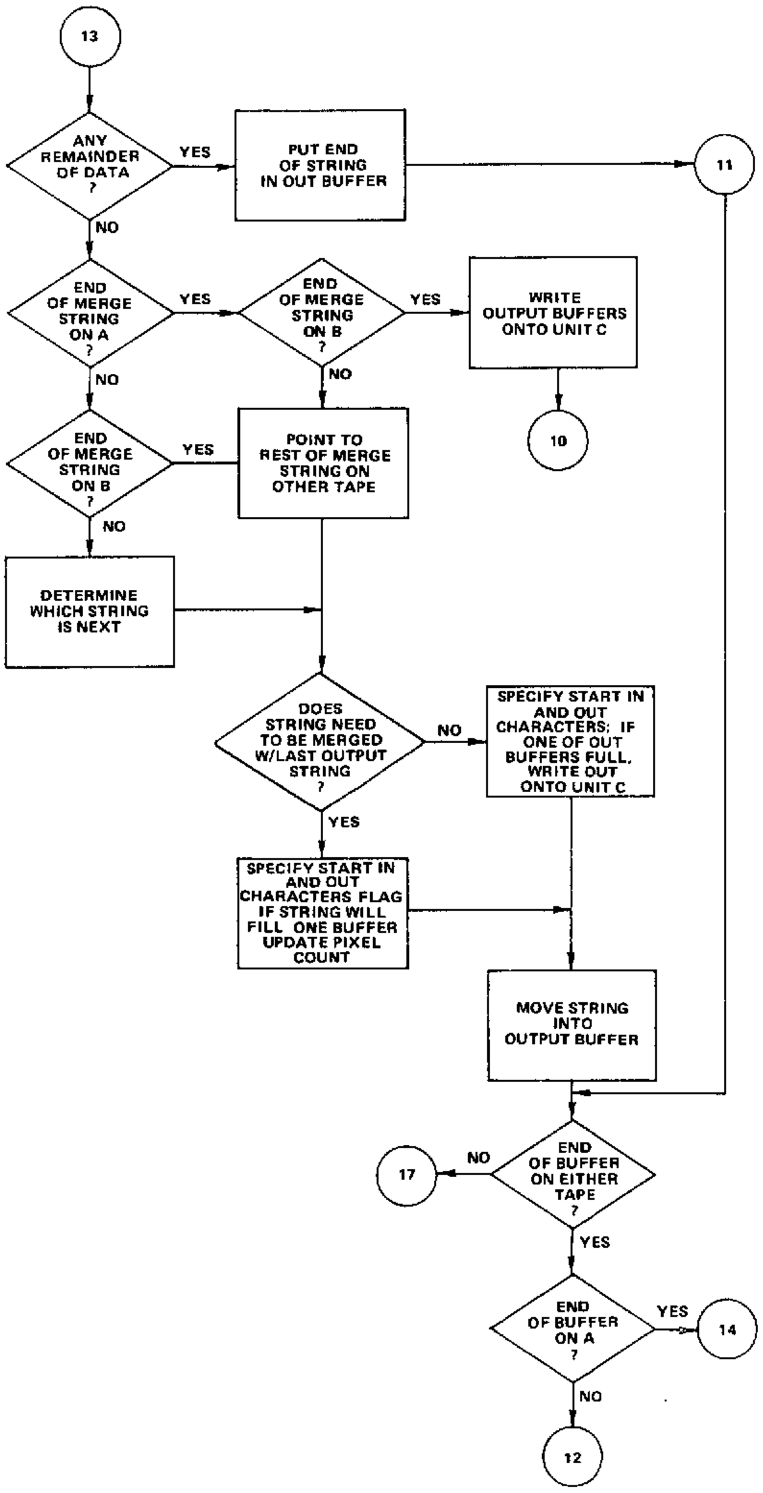

13
ANY REMAINDER OF DATA ?
YES
PUT END OF STRING IN OUT BUFFER
11
NO
END OF MERGE STRING ON A ?
YES
END OF MERGE STRING ON B ?
YES
WRITE OUTPUT BUFFERS ONTO UNIT C
10
NO
NO
END OF MERGE STRING ON B ?
YES
POINT TO REST OF MERGE STRING ON OTHER TAPE
NO
DETERMINE WHICH STRING IS NEXT
DOES STRING NEED TO BE MERGED W/LAST OUTPUT STRING ?
NO
SPECIFY START IN AND OUT CHARACTERS; IF ONE OF OUT BUFFERS FULL, WRITE OUT ONTO UNIT C
YES
SPECIFY START IN AND OUT CHARACTERS FLAG IF STRING WILL FILL ONE BUFFER UPDATE PIXEL COUNT
MOVE STRING INTO OUTPUT BUFFER
END OF BUFFER ON EITHER TAPE ?
NO
17
YES
END OF BUFFER ON A ?
YES
14
NO
12

Figure C-22. GEOMTRAN Program Flowchart (6 of 7)

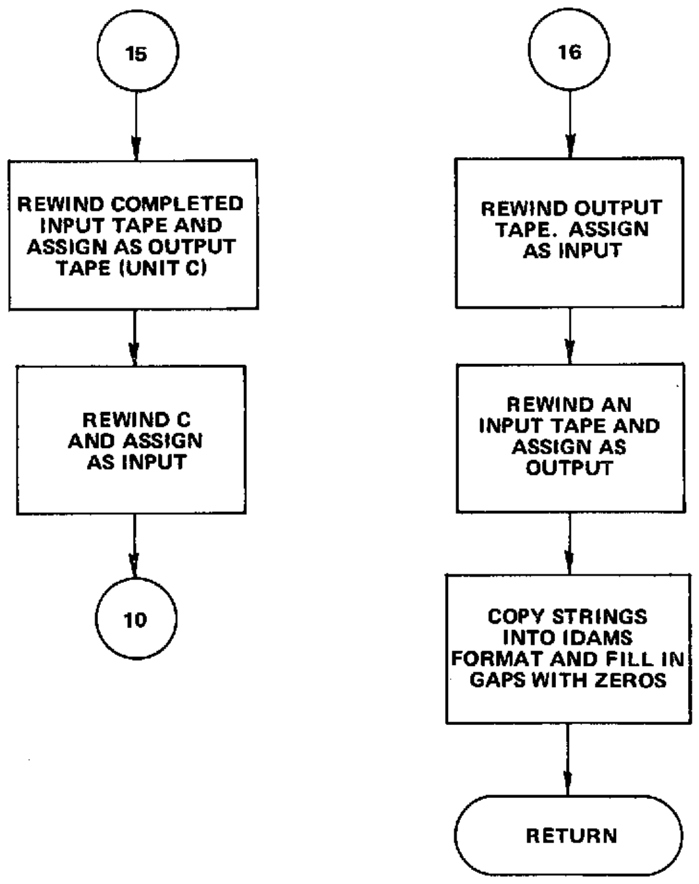

15
REWIND COMPLETED INPUT TAPE AND ASSIGN AS OUTPUT TAPE (UNIT C)
REWIND C AND ASSIGN AS INPUT
10
16
REWIND OUTPUT TAPE. ASSIGN AS INPUT
REWIND AN INPUT TAPE AND ASSIGN AS OUTPUT
COPY STRINGS INTO IDAMS FORMAT AND FILL IN GAPS WITH ZEROS
RETURN

Figure C-22. GEOMTRAN Program Flowchart (7 of 7)

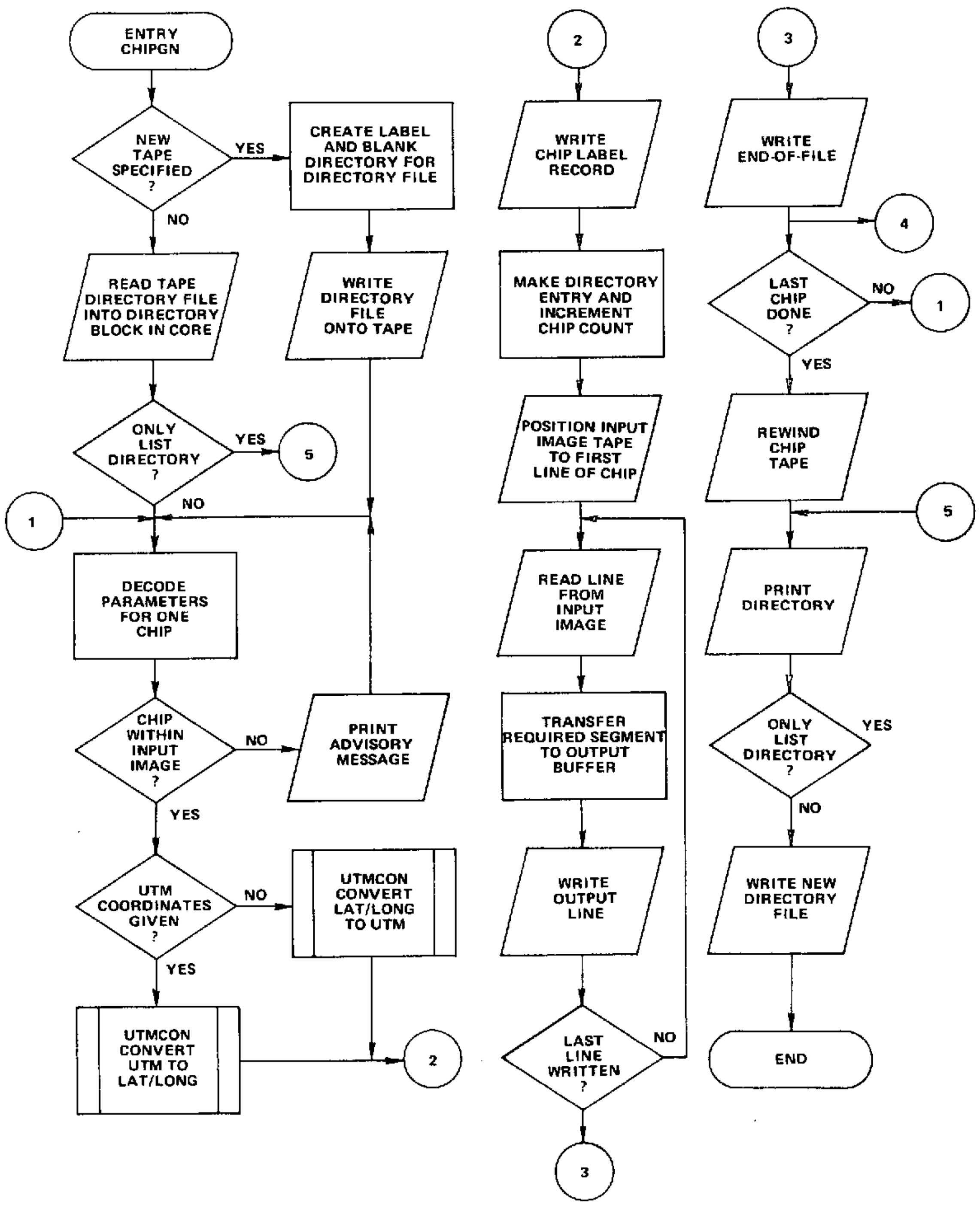

ENTRY CHIPGN
NEW TAPE SPECIFIED ?
YES
NO
CREATE LABEL AND BLANK DIRECTORY FOR DIRECTORY FILE
READ TAPE DIRECTORY FILE INTO DIRECTORY BLOCK IN CORE
WRITE DIRECTORY FILE ONTO TAPE
ONLY LIST DIRECTORY ?
YES
5
NO
1
DECODE PARAMETERS FOR ONE CHIP
CHIP WITHIN INPUT IMAGE ?
NO
PRINT ADVISORY MESSAGE
YES
UTM COORDINATES GIVEN ?
NO
UTMCON CONVERT LAT/LONG TO UTM
YES
UTMCON CONVERT UTM TO LAT/LONG
2
2
WRITE CHIP LABEL RECORD
MAKE DIRECTORY ENTRY AND INCREMENT CHIP COUNT
POSITION INPUT IMAGE TAPE TO FIRST LINE OF CHIP
READ LINE FROM INPUT IMAGE
TRANSFER REQUIRED SEGMENT TO OUTPUT BUFFER
WRITE OUTPUT LINE
LAST LINE WRITTEN ?
NO
3
3
WRITE END-OF-FILE
4
LAST CHIP DONE ?
NO
1
YES
REWIND CHIP TAPE
5
PRINT DIRECTORY
ONLY LIST DIRECTORY ?
YES
NO
WRITE NEW DIRECTORY FILE
END

Figure C-23. CHIPGN Program Flowchart

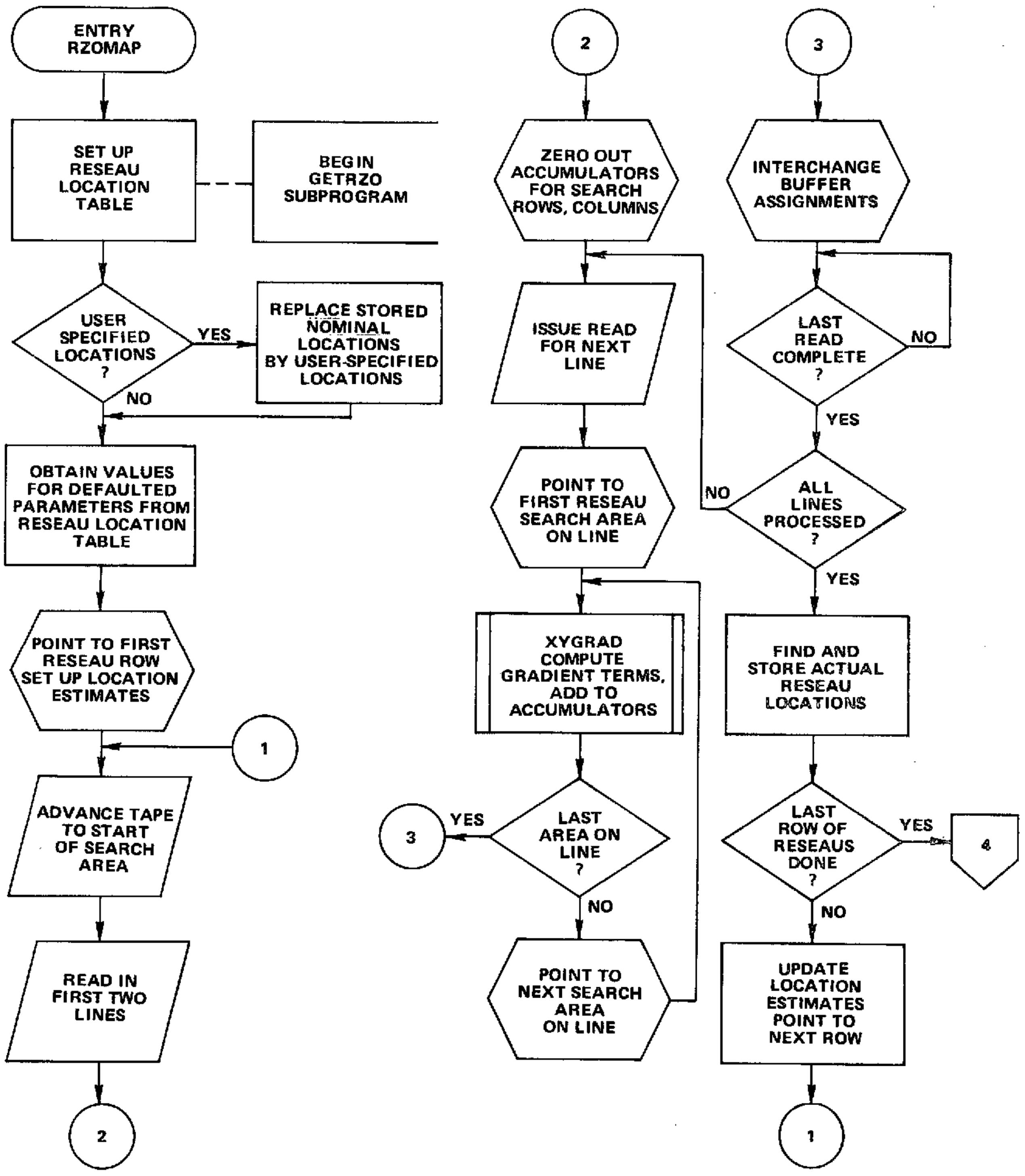

ENTRY RZOMAP
SET UP RESEAU LOCATION TABLE
BEGIN GETRZO SUBPROGRAM
USER SPECIFIED LOCATIONS ?
YES
NO
REPLACE STORED NOMINAL LOCATIONS BY USER-SPECIFIED LOCATIONS
OBTAIN VALUES FOR DEFAULTED PARAMETERS FROM RESEAU LOCATION TABLE
POINT TO FIRST RESEAU ROW SET UP LOCATION ESTIMATES
1
ADVANCE TAPE TO START OF SEARCH AREA
READ IN FIRST TWO LINES
2
2
ZERO OUT ACCUMULATORS FOR SEARCH ROWS, COLUMNS
ISSUE READ FOR NEXT LINE
POINT TO FIRST RESEAU SEARCH AREA ON LINE
XYGRAD COMPUTE GRADIENT TERMS, ADD TO ACCUMULATORS
LAST AREA ON LINE ?
YES
3
NO
POINT TO NEXT SEARCH AREA ON LINE
3
INTERCHANGE BUFFER ASSIGNMENTS
LAST READ COMPLETE ?
NO
YES
ALL LINES PROCESSED ?
NO
YES
FIND AND STORE ACTUAL RESEAU LOCATIONS
LAST ROW OF RESEAUS DONE ?
YES
4
NO
UPDATE LOCATION ESTIMATES POINT TO NEXT ROW
1

Figure C-24. RZOMAP Program Flowchart (1 of 3)

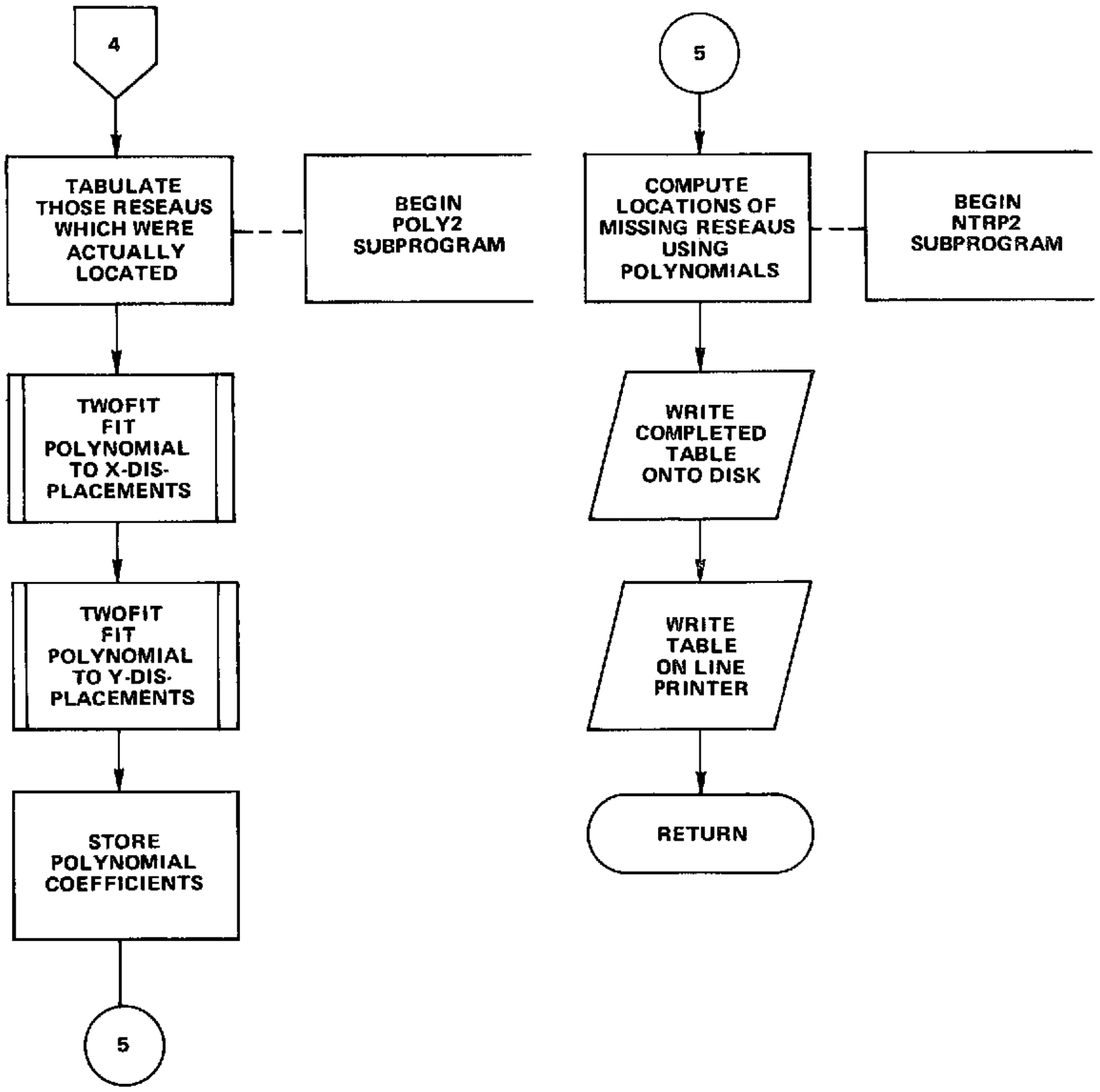

4
TABULATE THOSE RESEAUS WHICH WERE ACTUALLY LOCATED
BEGIN POLY2 SUBPROGRAM
TWOFIT FIT POLYNOMIAL TO X-DIS-PLACEMENTS
TWOFIT FIT POLYNOMIAL TO Y-DIS-PLACEMENTS
STORE POLYNOMIAL COEFFICIENTS
5
5
COMPUTE LOCATIONS OF MISSING RESEAUS USING POLYNOMIALS
BEGIN NTRP2 SUBPROGRAM
WRITE COMPLETED TABLE ONTO DISK
WRITE TABLE ON LINE PRINTER
RETURN

Figure C-24. RZOMAP Program Flowchart (2 of 3)

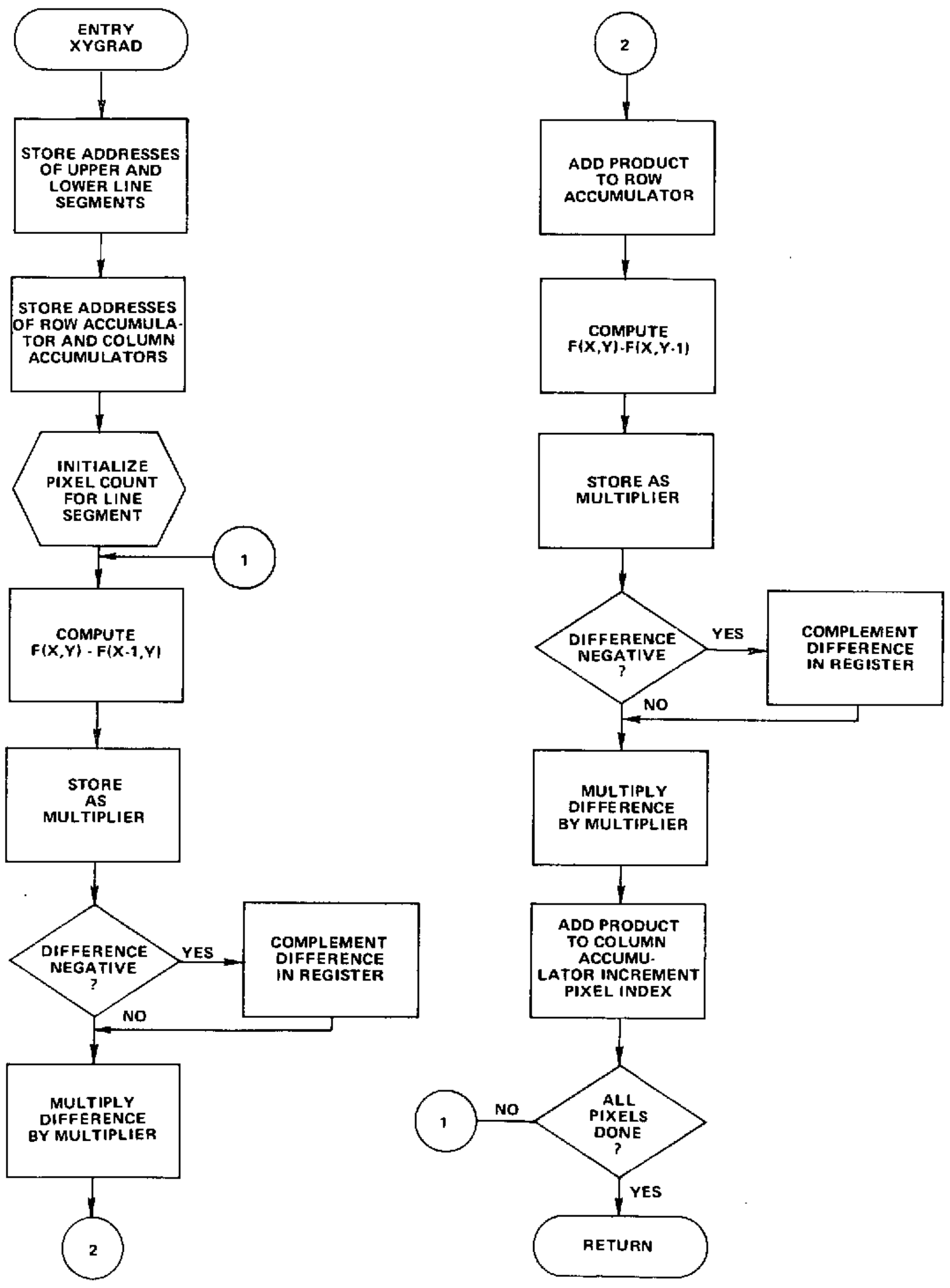

ENTRY XYGRAD
STORE ADDRESSES OF UPPER AND LOWER LINE SEGMENTS
STORE ADDRESSES OF ROW ACCUMULATOR AND COLUMN ACCUMULATORS
INITIALIZE PIXEL COUNT FOR LINE SEGMENT
1
COMPUTE F(X,Y) - F(X-1,Y)
STORE AS MULTIPLIER
DIFFERENCE NEGATIVE ?
YES
COMPLEMENT DIFFERENCE IN REGISTER
NO
MULTIPLY DIFFERENCE BY MULTIPLIER
2
2
ADD PRODUCT TO ROW ACCUMULATOR
COMPUTE F(X,Y)-F(X,Y-1)
STORE AS MULTIPLIER
DIFFERENCE NEGATIVE ?
YES
COMPLEMENT DIFFERENCE IN REGISTER
NO
MULTIPLY DIFFERENCE BY MULTIPLIER
ADD PRODUCT TO COLUMN ACCUMULATOR INCREMENT PIXEL INDEX
ALL PIXELS DONE ?
NO
1
YES
RETURN

Figure C-24. RZOMAP Program Flowchart (3 of 3)

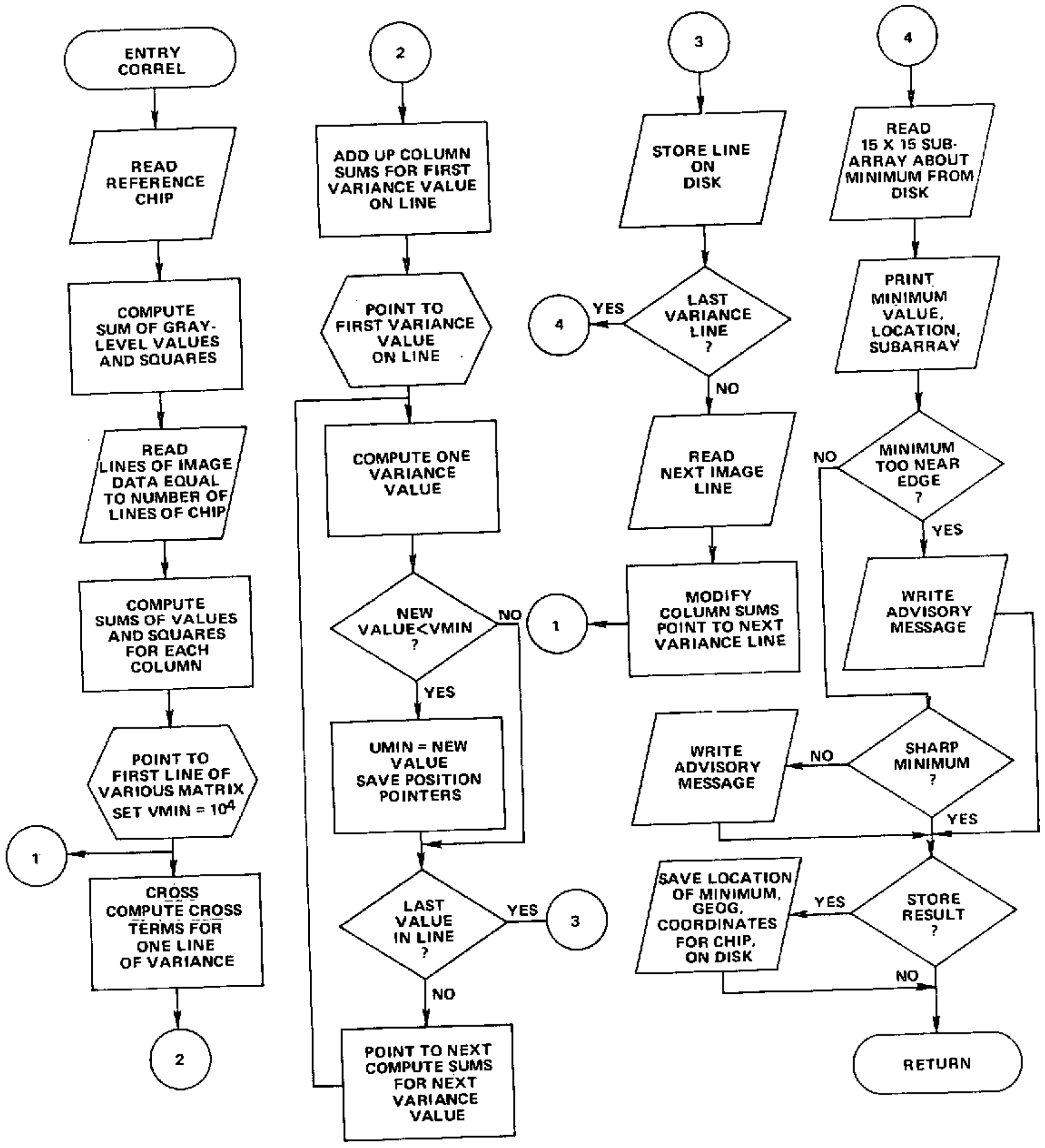

ENTRY CORREL
READ REFERENCE CHIP
COMPUTE SUM OF GRAY-LEVEL VALUES AND SQUARES
READ LINES OF IMAGE DATA EQUAL TO NUMBER OF LINES OF CHIP
COMPUTE SUMS OF VALUES AND SQUARES FOR EACH COLUMN
POINT TO FIRST LINE OF VARIOUS MATRIX SET VMIN = 10^4
1
CROSS COMPUTE CROSS TERMS FOR ONE LINE OF VARIANCE
2
2
ADD UP COLUMN SUMS FOR FIRST VARIANCE VALUE ON LINE
POINT TO FIRST VARIANCE VALUE ON LINE
COMPUTE ONE VARIANCE VALUE
NEW VALUE<VMIN ?
NO
1
YES
UMIN = NEW VALUE SAVE POSITION POINTERS
LAST VALUE IN LINE ?
YES
3
NO
POINT TO NEXT COMPUTE SUMS FOR NEXT VARIANCE VALUE
3
STORE LINE ON DISK
LAST VARIANCE LINE ?
YES
4
NO
READ NEXT IMAGE LINE
MODIFY COLUMN SUMS POINT TO NEXT VARIANCE LINE
4
READ 15 X 15 SUB-ARRAY ABOUT MINIMUM FROM DISK
PRINT MINIMUM VALUE, LOCATION, SUBARRAY
MINIMUM TOO NEAR EDGE ?
NO
YES
WRITE ADVISORY MESSAGE
SHARP MINIMUM ?
NO
WRITE ADVISORY MESSAGE
YES
STORE RESULT ?
YES
SAVE LOCATION OF MINIMUM, GEOG. COORDINATES FOR CHIP, ON DISK
NO
RETURN

Figure C-25. CORREL Program Flowchart (1 of 2)

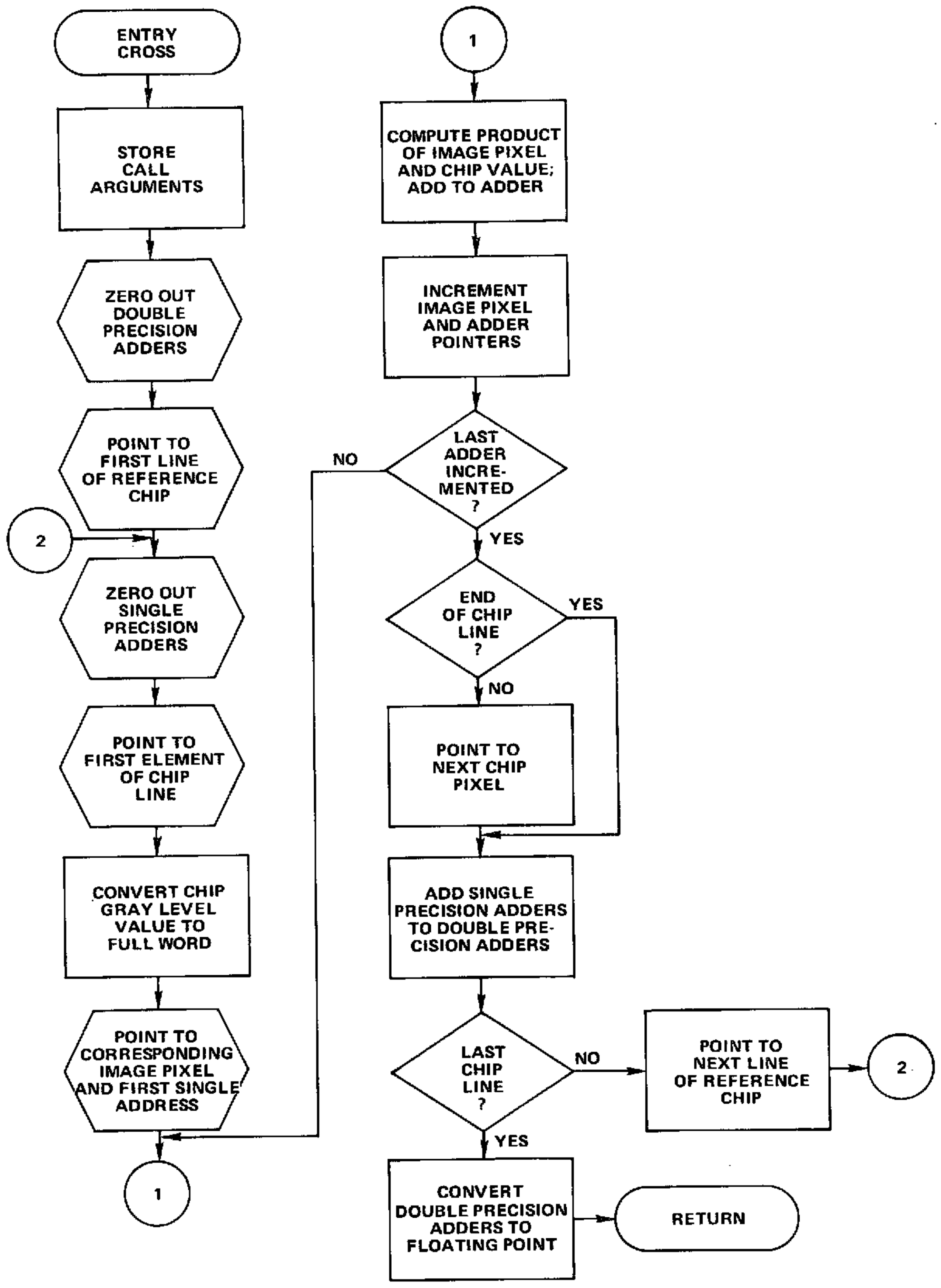

ENTRY CROSS
STORE CALL ARGUMENTS
ZERO OUT DOUBLE PRECISION ADDERS
POINT TO FIRST LINE OF REFERENCE CHIP
2
ZERO OUT SINGLE PRECISION ADDERS
POINT TO FIRST ELEMENT OF CHIP LINE
CONVERT CHIP GRAY LEVEL VALUE TO FULL WORD
POINT TO CORRESPONDING IMAGE PIXEL AND FIRST SINGLE ADDRESS
1
1
COMPUTE PRODUCT OF IMAGE PIXEL AND CHIP VALUE; ADD TO ADDER
INCREMENT IMAGE PIXEL AND ADDER POINTERS
LAST ADDER INCRE- MENTED ?
NO
YES
END OF CHIP LINE ?
YES
NO
POINT TO NEXT CHIP PIXEL
ADD SINGLE PRECISION ADDERS TO DOUBLE PRE- CISION ADDERS
LAST CHIP LINE ?
NO
POINT TO NEXT LINE OF REFERENCE CHIP
2
YES
CONVERT DOUBLE PRECISION ADDERS TO FLOATING POINT
RETURN

Figure C-25. CORREL Program Flowchart (2 of 2)

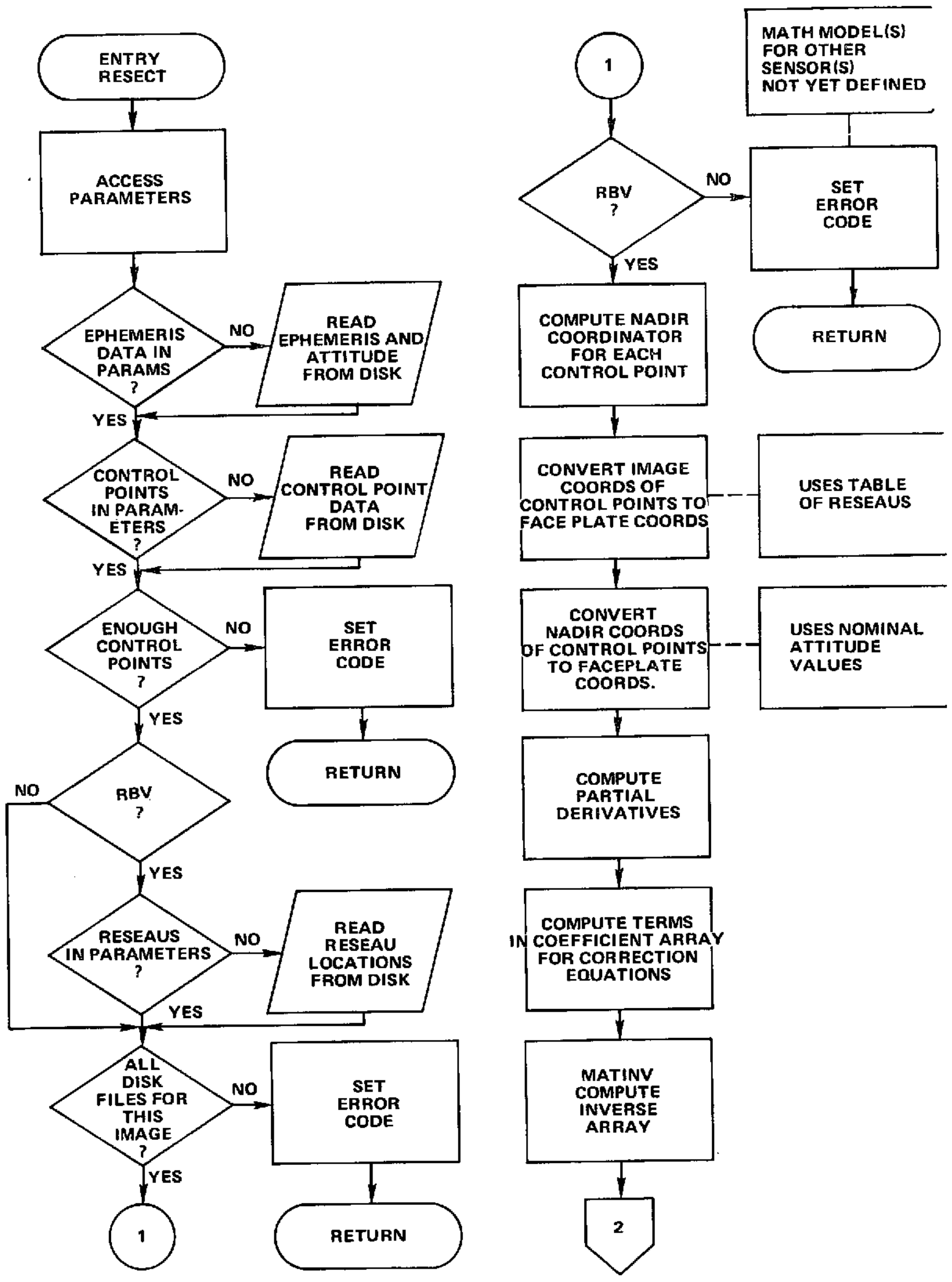
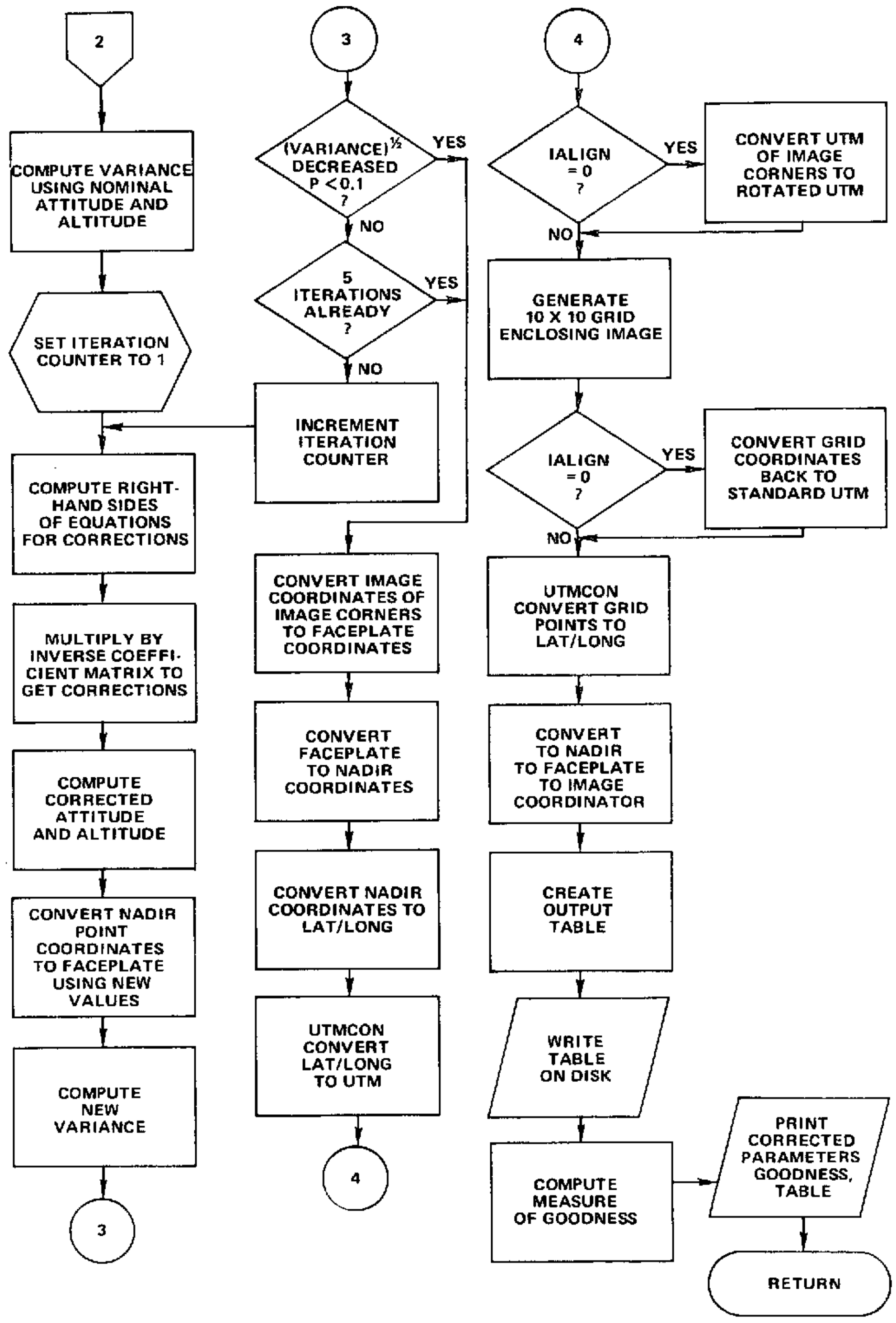

ENTRY RESECT
ACCESS PARAMETERS
EPHEMERIS DATA IN PARAMS ?
NO
READ EPHEMERIS AND ATTITUDE FROM DISK
YES
CONTROL POINTS IN PARAMETERS ?
NO
READ CONTROL POINT DATA FROM DISK
YES
ENOUGH CONTROL POINTS ?
NO
SET ERROR CODE
RETURN
YES
RBV ?
NO
YES
RESEAUS IN PARAMETERS ?
NO
READ RESEAU LOCATIONS FROM DISK
YES
ALL DISK FILES FOR THIS IMAGE ?
NO
SET ERROR CODE
RETURN
YES
1
1
RBV ?
NO
SET ERROR CODE
RETURN
MATH MODEL(S) FOR OTHER SENSOR(S) NOT YET DEFINED
YES
COMPUTE NADIR COORDINATOR FOR EACH CONTROL POINT
CONVERT IMAGE COORDS OF CONTROL POINTS TO FACE PLATE COORDS
USES TABLE OF RESEAUS
CONVERT NADIR COORDS OF CONTROL POINTS TO FACEPLATE COORDS.
USES NOMINAL ATTITUDE VALUES
COMPUTE PARTIAL DERIVATIVES
COMPUTE TERMS IN COEFFICIENT ARRAY FOR CORRECTION EQUATIONS
MATINV COMPUTE INVERSE ARRAY
2

Figure C-26. RESECT Program Flowchart (1 of 2)



2
COMPUTE VARIANCE USING NOMINAL ATTITUDE AND ALTITUDE
SET ITERATION COUNTER TO 1
COMPUTE RIGHT-HAND SIDES OF EQUATIONS FOR CORRECTIONS
MULTIPLY BY INVERSE COEFFICIENT MATRIX TO GET CORRECTIONS
COMPUTE CORRECTED ATTITUDE AND ALTITUDE
CONVERT NADIR POINT COORDINATES TO FACEPLATE USING NEW VALUES
COMPUTE NEW VARIANCE
3
3
(VARIANCE)½ DECREASED P < 0.1 ?
YES
NO
5 ITERATIONS ALREADY ?
YES
NO
INCREMENT ITERATION COUNTER
CONVERT IMAGE COORDINATES OF IMAGE CORNERS TO FACEPLATE COORDINATES
CONVERT FACEPLATE TO NADIR COORDINATES
CONVERT NADIR COORDINATES TO LAT/LONG
UTMCON CONVERT LAT/LONG TO UTM
4
4
IALIGN = 0 ?
YES
CONVERT UTM OF IMAGE CORNERS TO ROTATED UTM
NO
GENERATE 10 X 10 GRID ENCLOSING IMAGE
IALIGN = 0 ?
YES
CONVERT GRID COORDINATES BACK TO STANDARD UTM
NO
UTMCON CONVERT GRID POINTS TO LAT/LONG
CONVERT TO NADIR TO FACEPLATE TO IMAGE COORDINATOR
CREATE OUTPUT TABLE
WRITE TABLE ON DISK
COMPUTE MEASURE OF GOODNESS
PRINT CORRECTED PARAMETERS GOODNESS, TABLE
RETURN

Figure C-26. RESECT Program Flowchart (2 of 2)

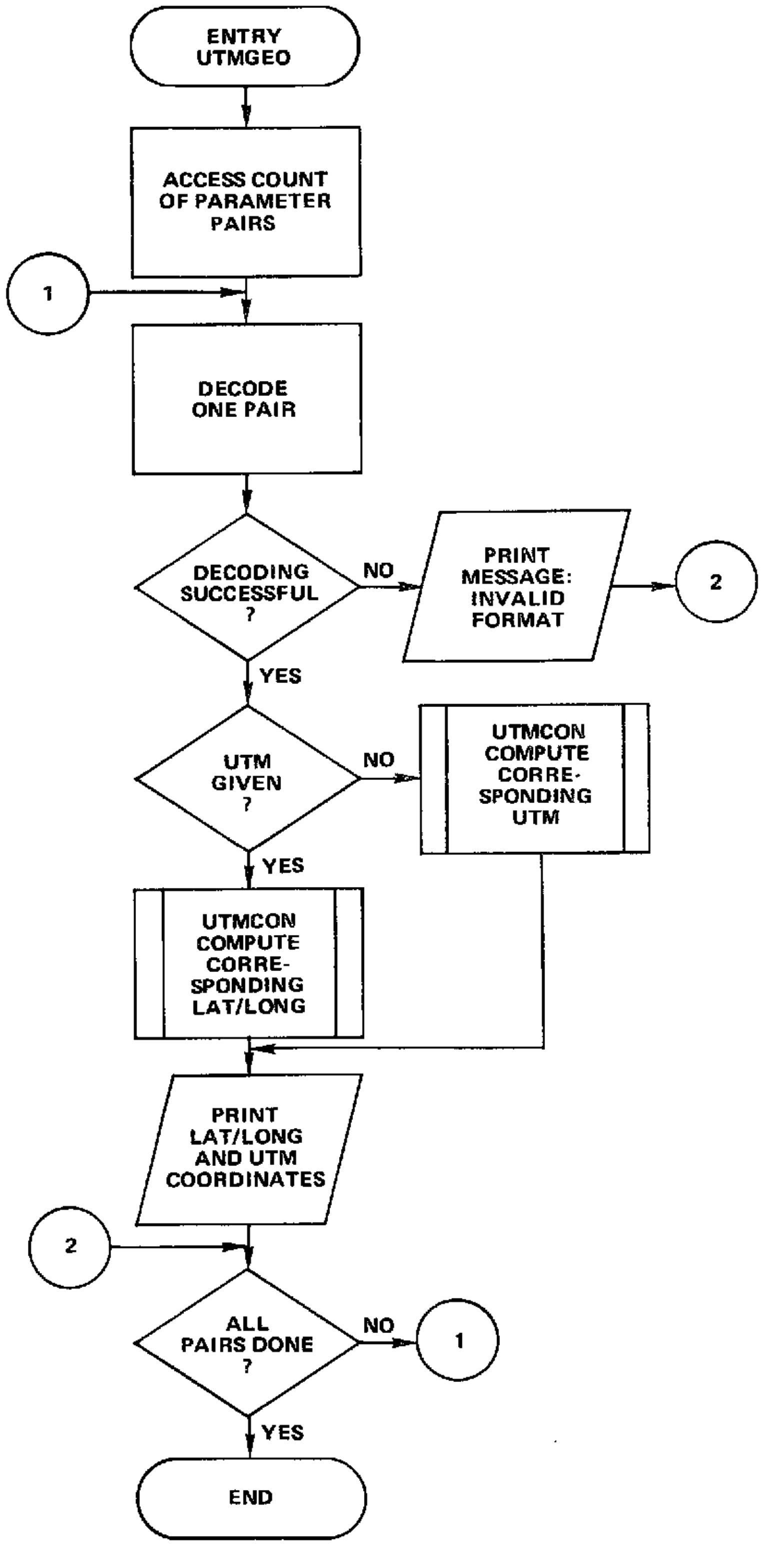

ENTRY UTMGEO
ACCESS COUNT OF PARAMETER PAIRS
1
DECODE ONE PAIR
DECODING SUCCESSFUL ?
NO
PRINT MESSAGE: INVALID FORMAT
2
YES
UTM GIVEN ?
NO
UTMCON COMPUTE CORRE-SPONDING UTM
YES
UTMCON COMPUTE CORRE-SPONDING LAT/LONG
PRINT LAT/LONG AND UTM COORDINATES
2
ALL PAIRS DONE ?
NO
1
YES
END

Figure C-27. UTMGEO Program Flowchart

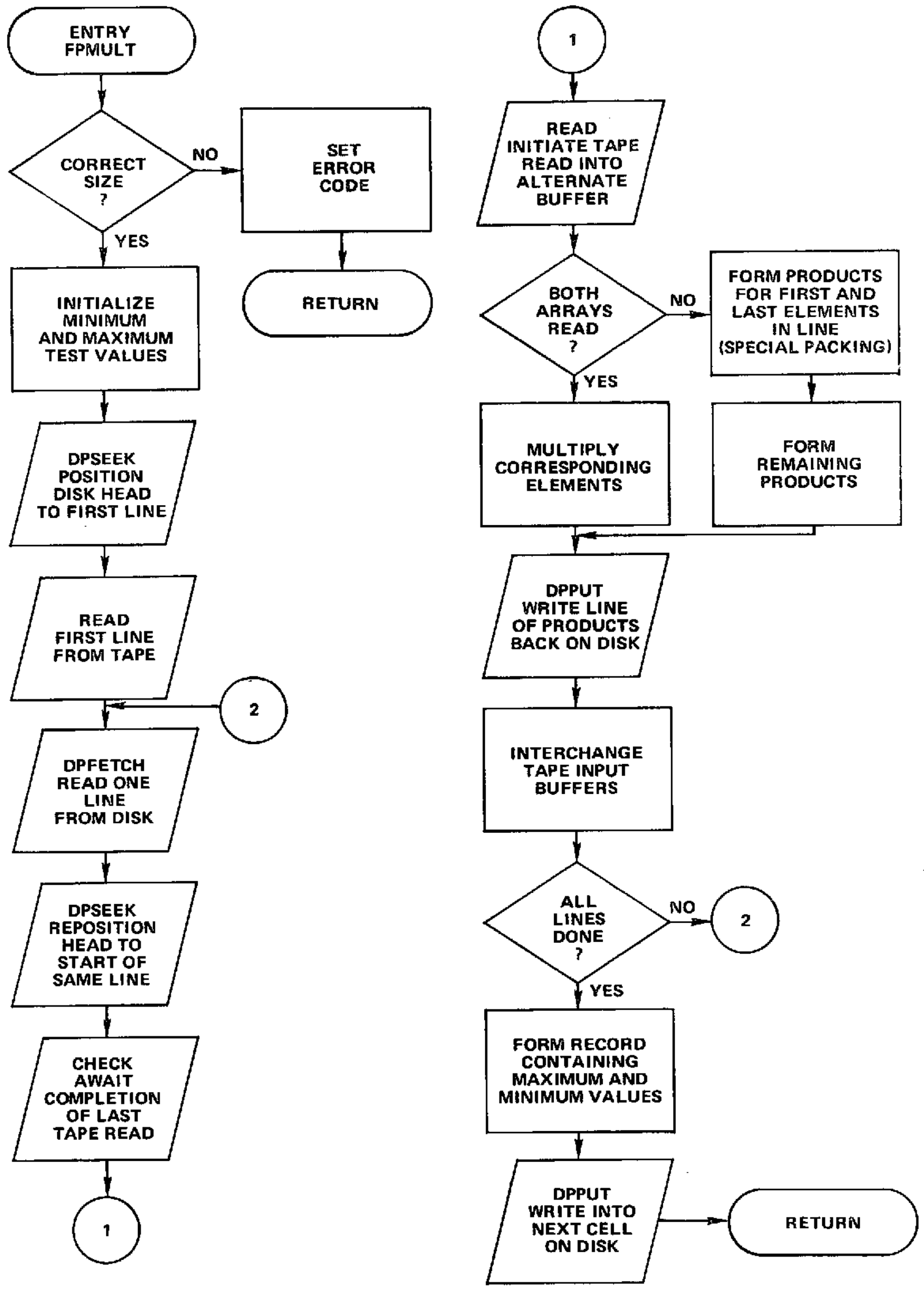

ENTRY FPMULT
CORRECT SIZE ?
NO
SET ERROR CODE
RETURN
YES
INITIALIZE MINIMUM AND MAXIMUM TEST VALUES
DPSEEK POSITION DISK HEAD TO FIRST LINE
READ FIRST LINE FROM TAPE
2
DPFETCH READ ONE LINE FROM DISK
DPSEEK REPOSITION HEAD TO START OF SAME LINE
CHECK AWAIT COMPLETION OF LAST TAPE READ
1
1
READ INITIATE TAPE READ INTO ALTERNATE BUFFER
BOTH ARRAYS READ ?
NO
FORM PRODUCTS FOR FIRST AND LAST ELEMENTS IN LINE (SPECIAL PACKING)
YES
MULTIPLY CORRESPONDING ELEMENTS
FORM REMAINING PRODUCTS
DPPUT WRITE LINE OF PRODUCTS BACK ON DISK
INTERCHANGE TAPE INPUT BUFFERS
ALL LINES DONE ?
NO
2
YES
FORM RECORD CONTAINING MAXIMUM AND MINIMUM VALUES
DPPUT WRITE INTO NEXT CELL ON DISK
RETURN

Figure C-28. FPMULT Program Flowchart

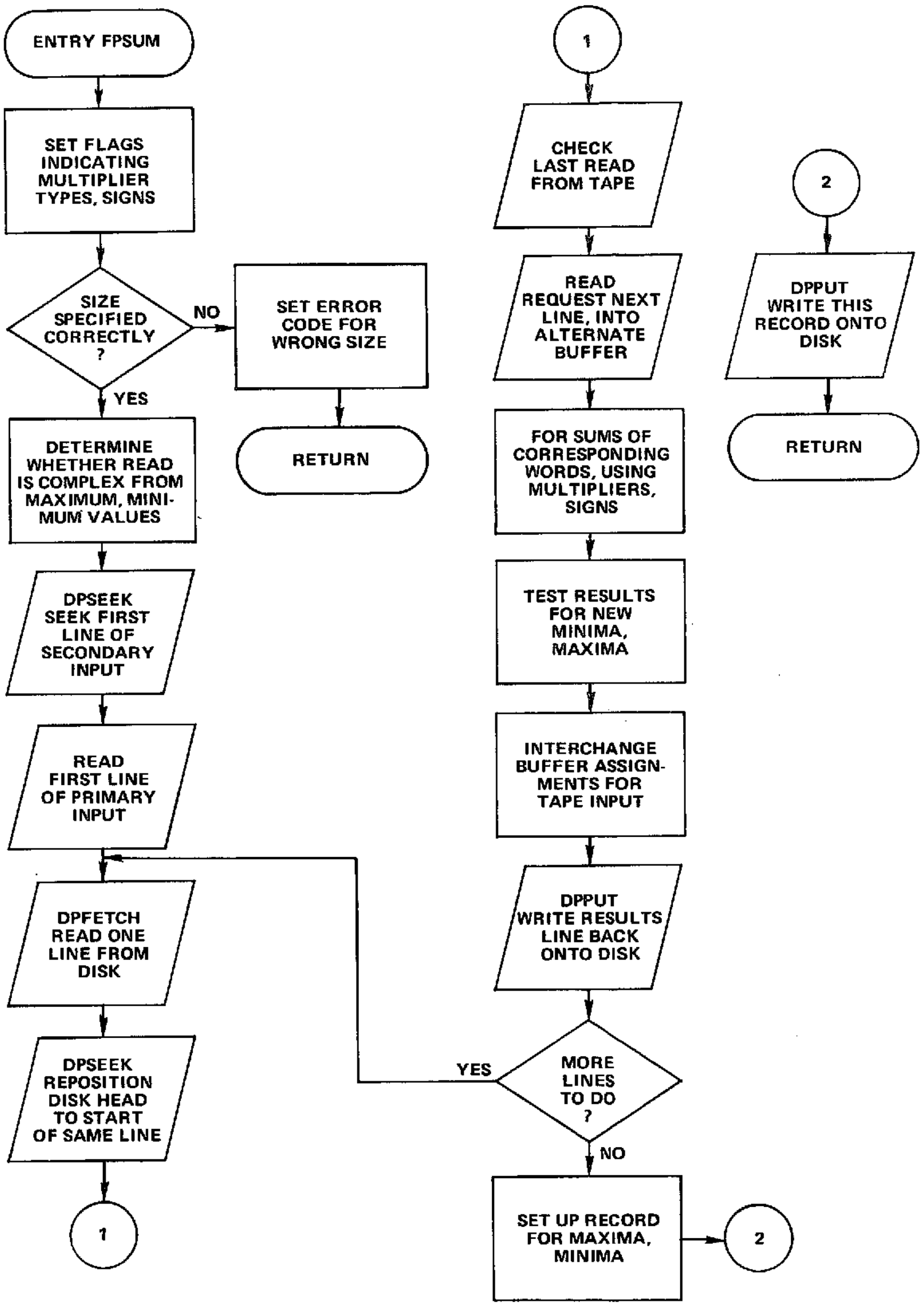

ENTRY FPSUM
SET FLAGS INDICATING MULTIPLIER TYPES, SIGNS
SIZE SPECIFIED CORRECTLY ?
NO
SET ERROR CODE FOR WRONG SIZE
RETURN
YES
DETERMINE WHETHER READ IS COMPLEX FROM MAXIMUM, MINIMUM VALUES
DPSEEK SEEK FIRST LINE OF SECONDARY INPUT
READ FIRST LINE OF PRIMARY INPUT
DPFETCH READ ONE LINE FROM DISK
DPSEEK REPOSITION DISK HEAD TO START OF SAME LINE
1
1
CHECK LAST READ FROM TAPE
READ REQUEST NEXT LINE, INTO ALTERNATE BUFFER
FOR SUMS OF CORRESPONDING WORDS, USING MULTIPLIERS, SIGNS
TEST RESULTS FOR NEW MINIMA, MAXIMA
INTERCHANGE BUFFER ASSIGNMENTS FOR TAPE INPUT
DPPUT WRITE RESULTS LINE BACK ONTO DISK
MORE LINES TO DO ?
YES
NO
SET UP RECORD FOR MAXIMA, MINIMA
2
2
DPPUT WRITE THIS RECORD ONTO DISK
RETURN

Figure C-29. FPSUM Program Flowchart

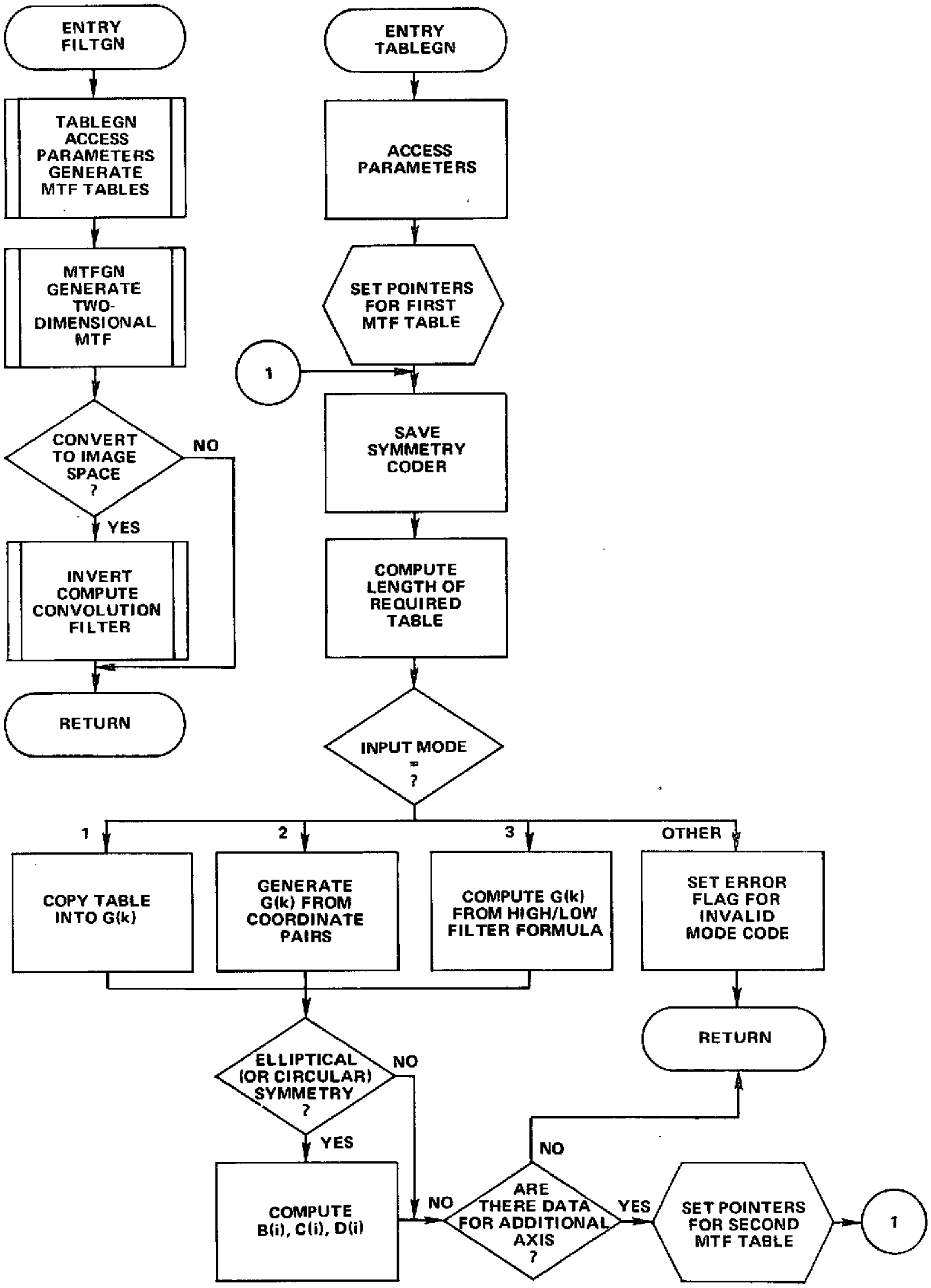

ENTRY FILTGN
TABLEGN ACCESS PARAMETERS GENERATE MTF TABLES
MTFGN GENERATE TWO-DIMENSIONAL MTF
CONVERT TO IMAGE SPACE ?
NO
YES
INVERT COMPUTE CONVOLUTION FILTER
RETURN
ENTRY TABLEGN
ACCESS PARAMETERS
SET POINTERS FOR FIRST MTF TABLE
1
SAVE SYMMETRY CODER
COMPUTE LENGTH OF REQUIRED TABLE
INPUT MODE = ?
1
2
3
OTHER
COPY TABLE INTO G(k)
GENERATE G(k) FROM COORDINATE PAIRS
COMPUTE G(k) FROM HIGH/LOW FILTER FORMULA
SET ERROR FLAG FOR INVALID MODE CODE
RETURN
ELLIPTICAL (OR CIRCULAR) SYMMETRY ?
NO
YES
COMPUTE B(i), C(i), D(i)
NO
ARE THERE DATA FOR ADDITIONAL AXIS ?
NO
YES
SET POINTERS FOR SECOND MTF TABLE
1

Figure C-30. FILTGN Program Flowchart (1 of 2)

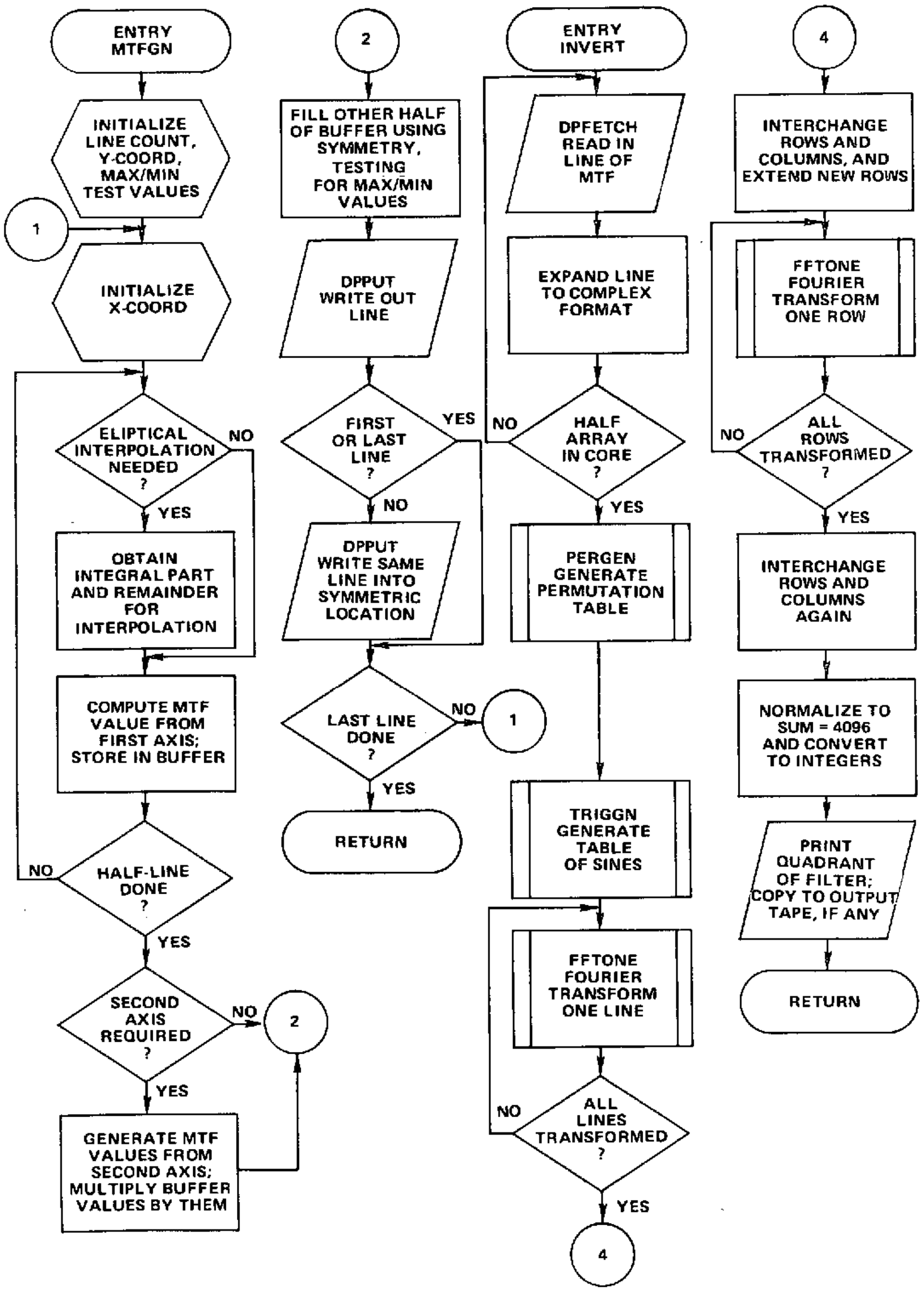

ENTRY MTFGN
INITIALIZE LINE COUNT, Y-COORD, MAX/MIN TEST VALUES
1
INITIALIZE X-COORD
ELIPTICAL INTERPOLATION NEEDED ?
NO
YES
OBTAIN INTEGRAL PART AND REMAINDER FOR INTERPOLATION
COMPUTE MTF VALUE FROM FIRST AXIS; STORE IN BUFFER
HALF-LINE DONE ?
NO
YES
SECOND AXIS REQUIRED ?
NO
2
YES
GENERATE MTF VALUES FROM SECOND AXIS; MULTIPLY BUFFER VALUES BY THEM
2
FILL OTHER HALF OF BUFFER USING SYMMETRY, TESTING FOR MAX/MIN VALUES
DPPUT WRITE OUT LINE
FIRST OR LAST LINE ?
YES
NO
DPPUT WRITE SAME LINE INTO SYMMETRIC LOCATION
LAST LINE DONE ?
NO
1
YES
RETURN
ENTRY INVERT
DPFETCH READ IN LINE OF MTF
EXPAND LINE TO COMPLEX FORMAT
HALF ARRAY IN CORE ?
NO
YES
PERGEN GENERATE PERMUTATION TABLE
TRIGGN GENERATE TABLE OF SINES
FFTONE FOURIER TRANSFORM ONE LINE
ALL LINES TRANSFORMED ?
NO
YES
4
4
INTERCHANGE ROWS AND COLUMNS, AND EXTEND NEW ROWS
FFTONE FOURIER TRANSFORM ONE ROW
ALL ROWS TRANSFORMED ?
NO
YES
INTERCHANGE ROWS AND COLUMNS AGAIN
NORMALIZE TO SUM = 4096 AND CONVERT TO INTEGERS
PRINT QUADRANT OF FILTER; COPY TO OUTPUT TAPE, IF ANY
RETURN

Figure C-30. FILTGN Program Flowchart (2 of 2)

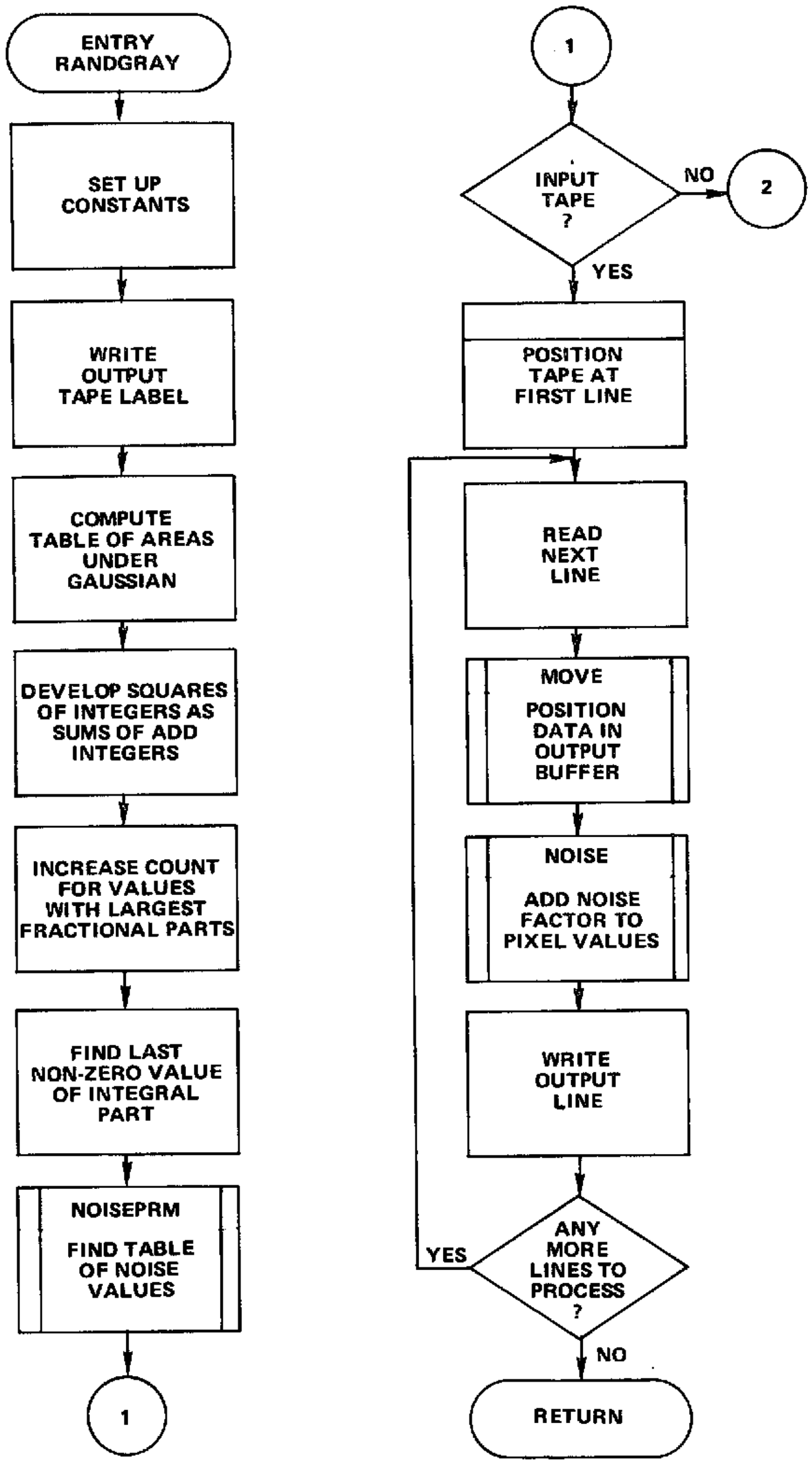

ENTRY RANDGRAY
SET UP CONSTANTS
WRITE OUTPUT TAPE LABEL
COMPUTE TABLE OF AREAS UNDER GAUSSIAN
DEVELOP SQUARES OF INTEGERS AS SUMS OF ADD INTEGERS
INCREASE COUNT FOR VALUES WITH LARGEST FRACTIONAL PARTS
FIND LAST NON-ZERO VALUE OF INTEGRAL PART
NOISEPRM
FIND TABLE OF NOISE VALUES
1
1
INPUT TAPE ?
NO
2
YES
POSITION TAPE AT FIRST LINE
READ NEXT LINE
MOVE
POSITION DATA IN OUTPUT BUFFER
NOISE
ADD NOISE FACTOR TO PIXEL VALUES
WRITE OUTPUT LINE
ANY MORE LINES TO PROCESS ?
YES
NO
RETURN

Figure C-31. RANDGRAY Program Flowchart (1 of 2)

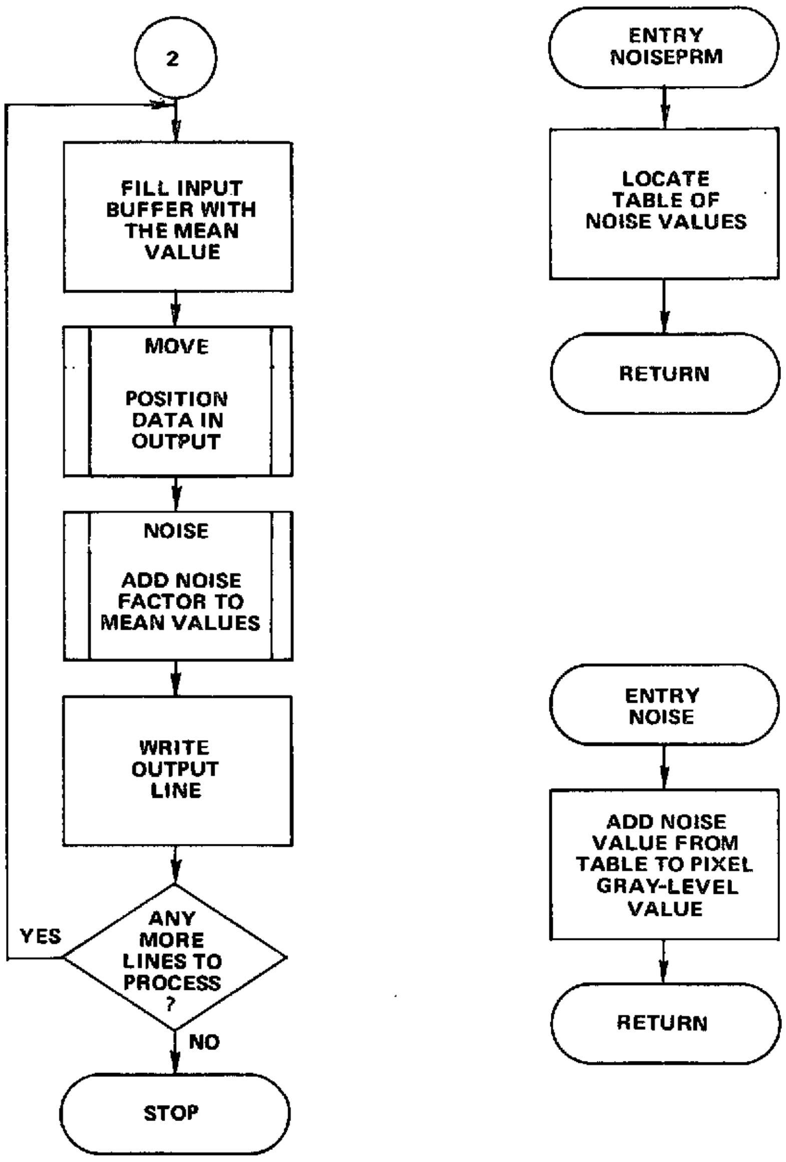
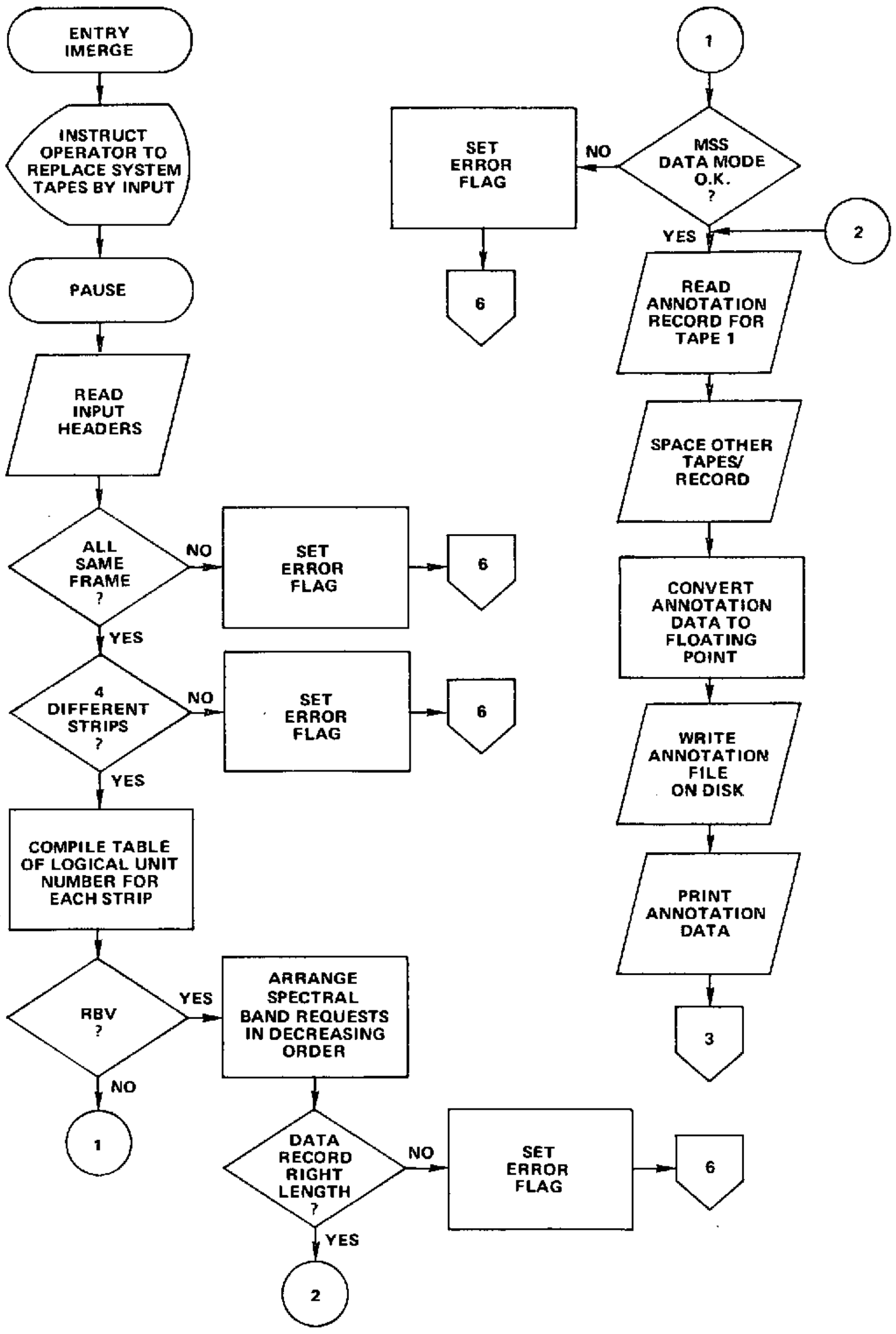

2
FILL INPUT BUFFER WITH THE MEAN VALUE
MOVE
POSITION DATA IN OUTPUT
NOISE
ADD NOISE FACTOR TO MEAN VALUES
WRITE OUTPUT LINE
ANY MORE LINES TO PROCESS ?
YES
NO
STOP
ENTRY NOISEPRM
LOCATE TABLE OF NOISE VALUES
RETURN
ENTRY NOISE
ADD NOISE VALUE FROM TABLE TO PIXEL GRAY-LEVEL VALUE
RETURN

Figure C-31. RANDGRAY Program Flowchart (2 of 2)



ENTRY IMERGE
INSTRUCT OPERATOR TO REPLACE SYSTEM TAPES BY INPUT
PAUSE
READ INPUT HEADERS
ALL SAME FRAME ?
NO
SET ERROR FLAG
6
YES
4 DIFFERENT STRIPS ?
NO
SET ERROR FLAG
6
YES
COMPILE TABLE OF LOGICAL UNIT NUMBER FOR EACH STRIP
RBV ?
YES
ARRANGE SPECTRAL BAND REQUESTS IN DECREASING ORDER
NO
1
DATA RECORD RIGHT LENGTH ?
NO
SET ERROR FLAG
6
YES
2
1
MSS DATA MODE O.K. ?
NO
SET ERROR FLAG
6
YES
2
READ ANNOTATION RECORD FOR TAPE 1
SPACE OTHER TAPES/ RECORD
CONVERT ANNOTATION DATA TO FLOATING POINT
WRITE ANNOTATION FILE ON DISK
PRINT ANNOTATION DATA
3

Figure C-32. IMERGE Program Flowchart (1 of 3)

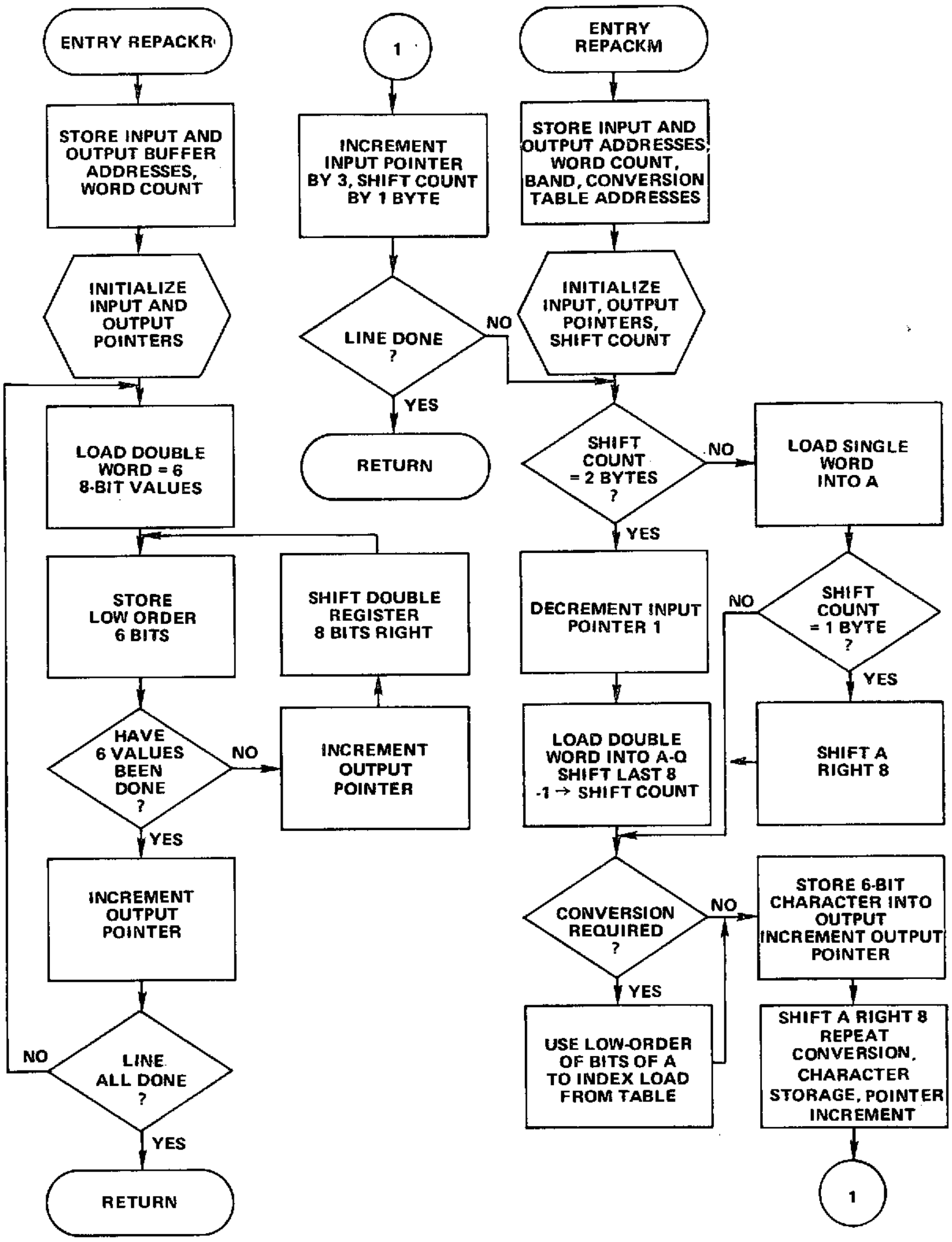
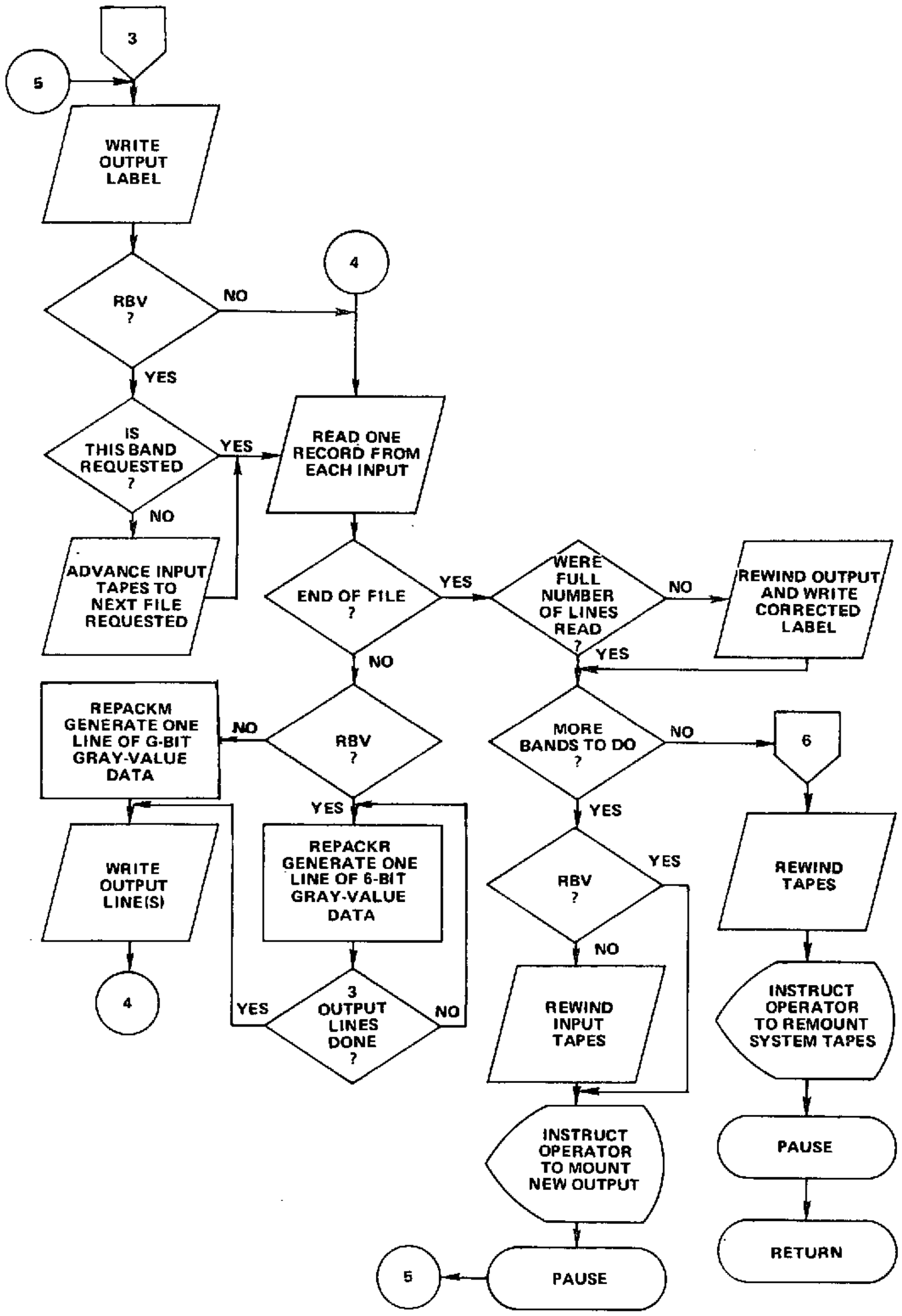

ENTRY REPACKR
STORE INPUT AND OUTPUT BUFFER ADDRESSES, WORD COUNT
INITIALIZE INPUT AND OUTPUT POINTERS
LOAD DOUBLE WORD = 6 8-BIT VALUES
STORE LOW ORDER 6 BITS
HAVE 6 VALUES BEEN DONE ?
NO
INCREMENT OUTPUT POINTER
SHIFT DOUBLE REGISTER 8 BITS RIGHT
YES
INCREMENT OUTPUT POINTER
LINE ALL DONE ?
NO
YES
RETURN
1
INCREMENT INPUT POINTER BY 3, SHIFT COUNT BY 1 BYTE
LINE DONE ?
NO
YES
RETURN
ENTRY REPACKM
STORE INPUT AND OUTPUT ADDRESSES, WORD COUNT, BAND, CONVERSION TABLE ADDRESSES
INITIALIZE INPUT, OUTPUT POINTERS, SHIFT COUNT
SHIFT COUNT = 2 BYTES ?
NO
LOAD SINGLE WORD INTO A
YES
DECREMENT INPUT POINTER 1
SHIFT COUNT = 1 BYTE ?
NO
YES
SHIFT A RIGHT 8
LOAD DOUBLE WORD INTO A-Q SHIFT LAST 8 -1 → SHIFT COUNT
CONVERSION REQUIRED ?
NO
YES
STORE 6-BIT CHARACTER INTO OUTPUT INCREMENT OUTPUT POINTER
USE LOW-ORDER OF BITS OF A TO INDEX LOAD FROM TABLE
SHIFT A RIGHT 8 REPEAT CONVERSION, CHARACTER STORAGE, POINTER INCREMENT
1

Figure C-32. IMERGE Program Flowchart (2 of 3)



3
5
WRITE OUTPUT LABEL
RBV ?
NO
4
YES
IS THIS BAND REQUESTED ?
YES
NO
ADVANCE INPUT TAPES TO NEXT FILE REQUESTED
READ ONE RECORD FROM EACH INPUT
END OF FILE ?
YES
NO
WERE FULL NUMBER OF LINES READ ?
NO
YES
REWIND OUTPUT AND WRITE CORRECTED LABEL
RBV ?
NO
YES
REPACKM GENERATE ONE LINE OF G-BIT GRAY-VALUE DATA
WRITE OUTPUT LINE(S)
4
REPACKR GENERATE ONE LINE OF 6-BIT GRAY-VALUE DATA
3 OUTPUT LINES DONE ?
YES
NO
MORE BANDS TO DO ?
NO
6
YES
RBV ?
YES
NO
REWIND INPUT TAPES
INSTRUCT OPERATOR TO MOUNT NEW OUTPUT
PAUSE
5
REWIND TAPES
INSTRUCT OPERATOR TO REMOUNT SYSTEM TAPES
PAUSE
RETURN

Figure C-32. IMERGE Program Flowchart (3 of 3)

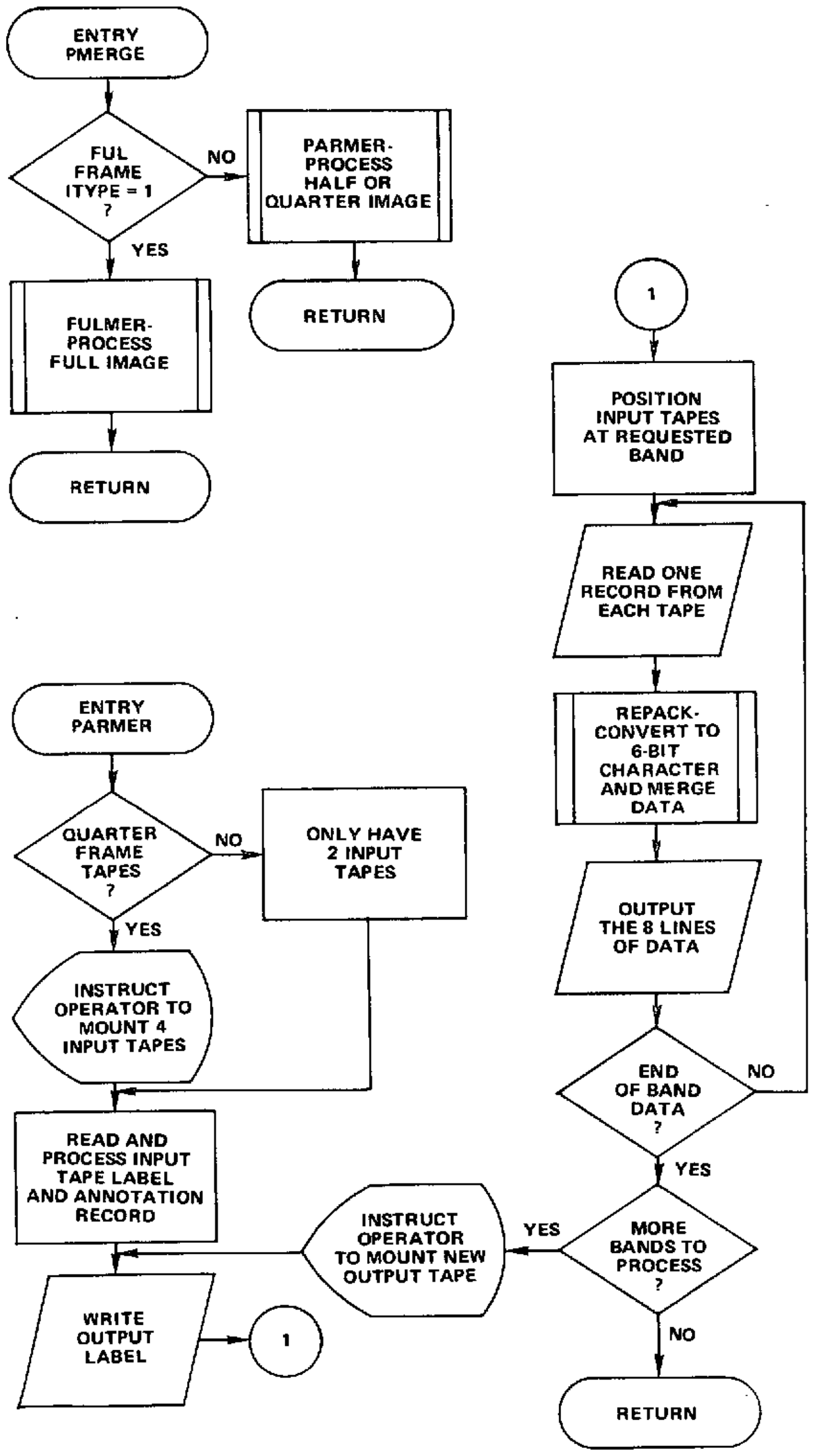
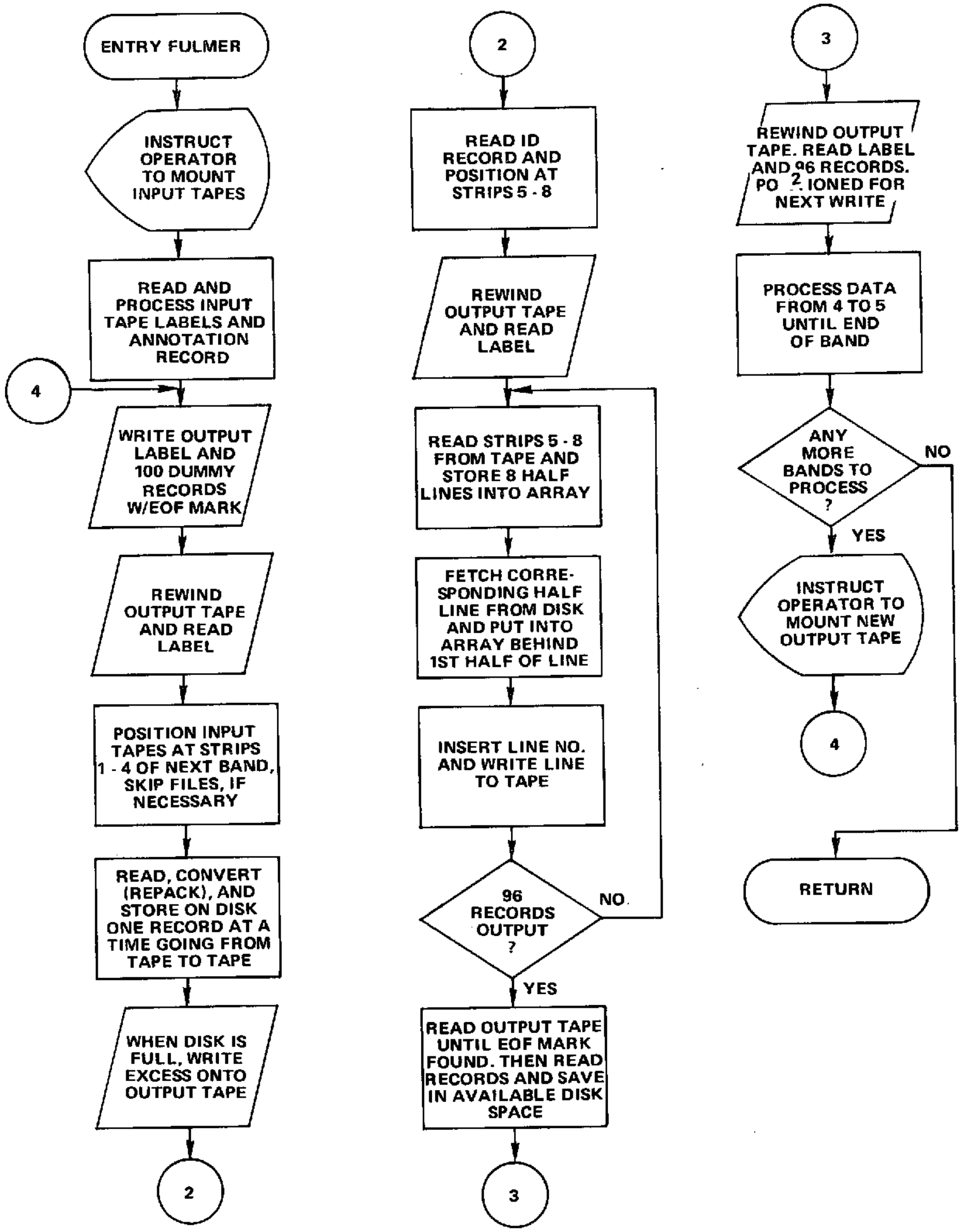

ENTRY PMERGE
FUL FRAME ITYPE = 1 ?
NO
PARMER-PROCESS HALF OR QUARTER IMAGE
RETURN
YES
FULMER-PROCESS FULL IMAGE
RETURN
ENTRY PARMER
QUARTER FRAME TAPES ?
NO
ONLY HAVE 2 INPUT TAPES
YES
INSTRUCT OPERATOR TO MOUNT 4 INPUT TAPES
READ AND PROCESS INPUT TAPE LABEL AND ANNOTATION RECORD
WRITE OUTPUT LABEL
1
1
POSITION INPUT TAPES AT REQUESTED BAND
READ ONE RECORD FROM EACH TAPE
REPACK-CONVERT TO 6-BIT CHARACTER AND MERGE DATA
OUTPUT THE 8 LINES OF DATA
END OF BAND DATA ?
NO
YES
MORE BANDS TO PROCESS ?
YES
INSTRUCT OPERATOR TO MOUNT NEW OUTPUT TAPE
NO
RETURN

Figure C-33. PMERGE Program Flowchart (1 of 2)



ENTRY FULMER
INSTRUCT OPERATOR TO MOUNT INPUT TAPES
READ AND PROCESS INPUT TAPE LABELS AND ANNOTATION RECORD
4
WRITE OUTPUT LABEL AND 100 DUMMY RECORDS W/EOF MARK
REWIND OUTPUT TAPE AND READ LABEL
POSITION INPUT TAPES AT STRIPS 1 - 4 OF NEXT BAND, SKIP FILES, IF NECESSARY
READ, CONVERT (REPACK), AND STORE ON DISK ONE RECORD AT A TIME GOING FROM TAPE TO TAPE
WHEN DISK IS FULL, WRITE EXCESS ONTO OUTPUT TAPE
2
2
READ ID RECORD AND POSITION AT STRIPS 5 - 8
REWIND OUTPUT TAPE AND READ LABEL
READ STRIPS 5 - 8 FROM TAPE AND STORE 8 HALF LINES INTO ARRAY
FETCH CORRE-SPONDING HALF LINE FROM DISK AND PUT INTO ARRAY BEHIND 1ST HALF OF LINE
INSERT LINE NO. AND WRITE LINE TO TAPE
96 RECORDS OUTPUT ?
NO
YES
READ OUTPUT TAPE UNTIL EOF MARK FOUND. THEN READ RECORDS AND SAVE IN AVAILABLE DISK SPACE
3
3
REWIND OUTPUT TAPE. READ LABEL AND 96 RECORDS. PO 2 IONED FOR NEXT WRITE
PROCESS DATA FROM 4 TO 5 UNTIL END OF BAND
ANY MORE BANDS TO PROCESS ?
NO
YES
INSTRUCT OPERATOR TO MOUNT NEW OUTPUT TAPE
4
RETURN

Figure C-33. PMERGE Program Flowchart (2 of 2)

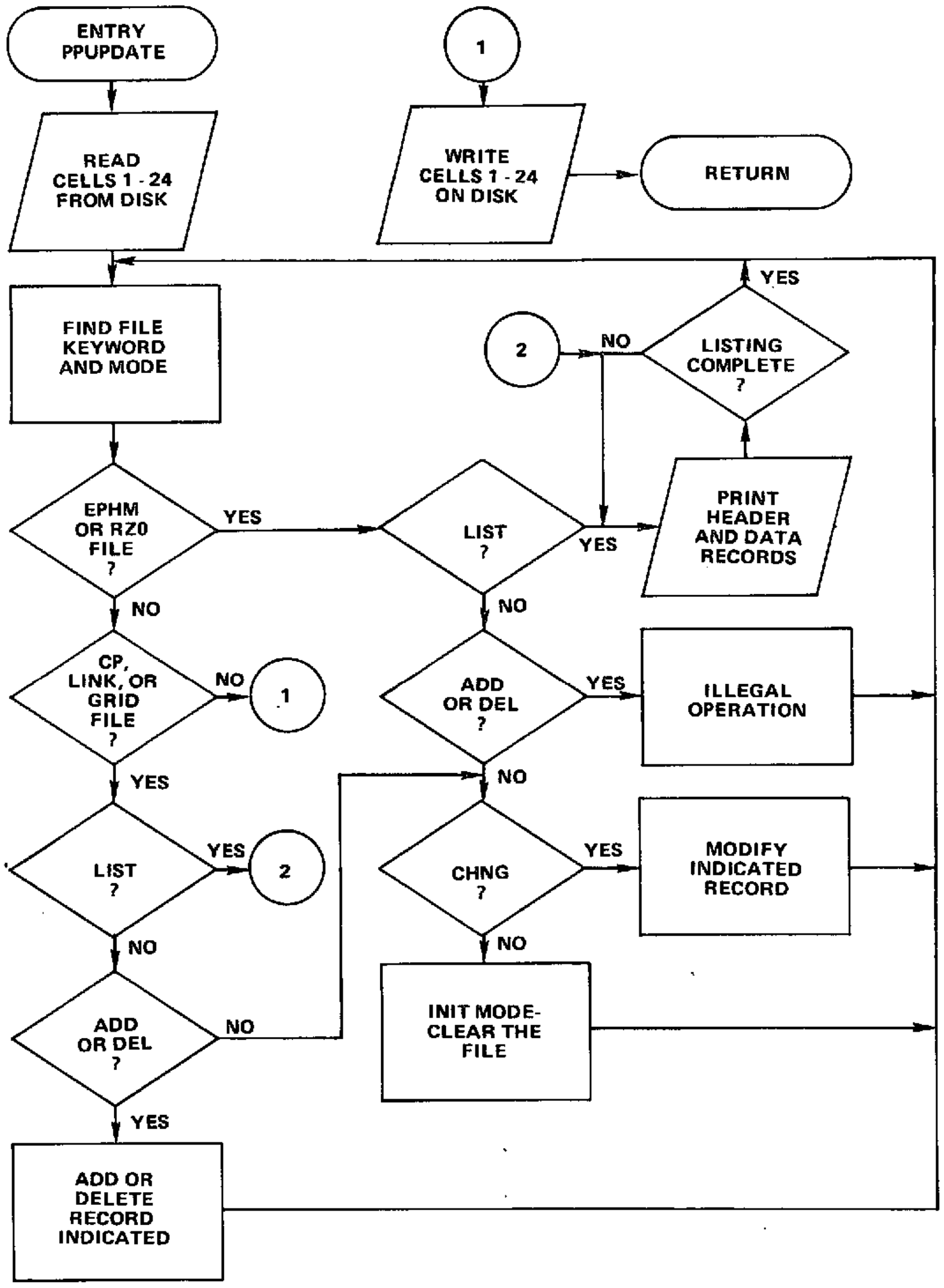


Figure C-34. PPUPDATE Program Flowchart

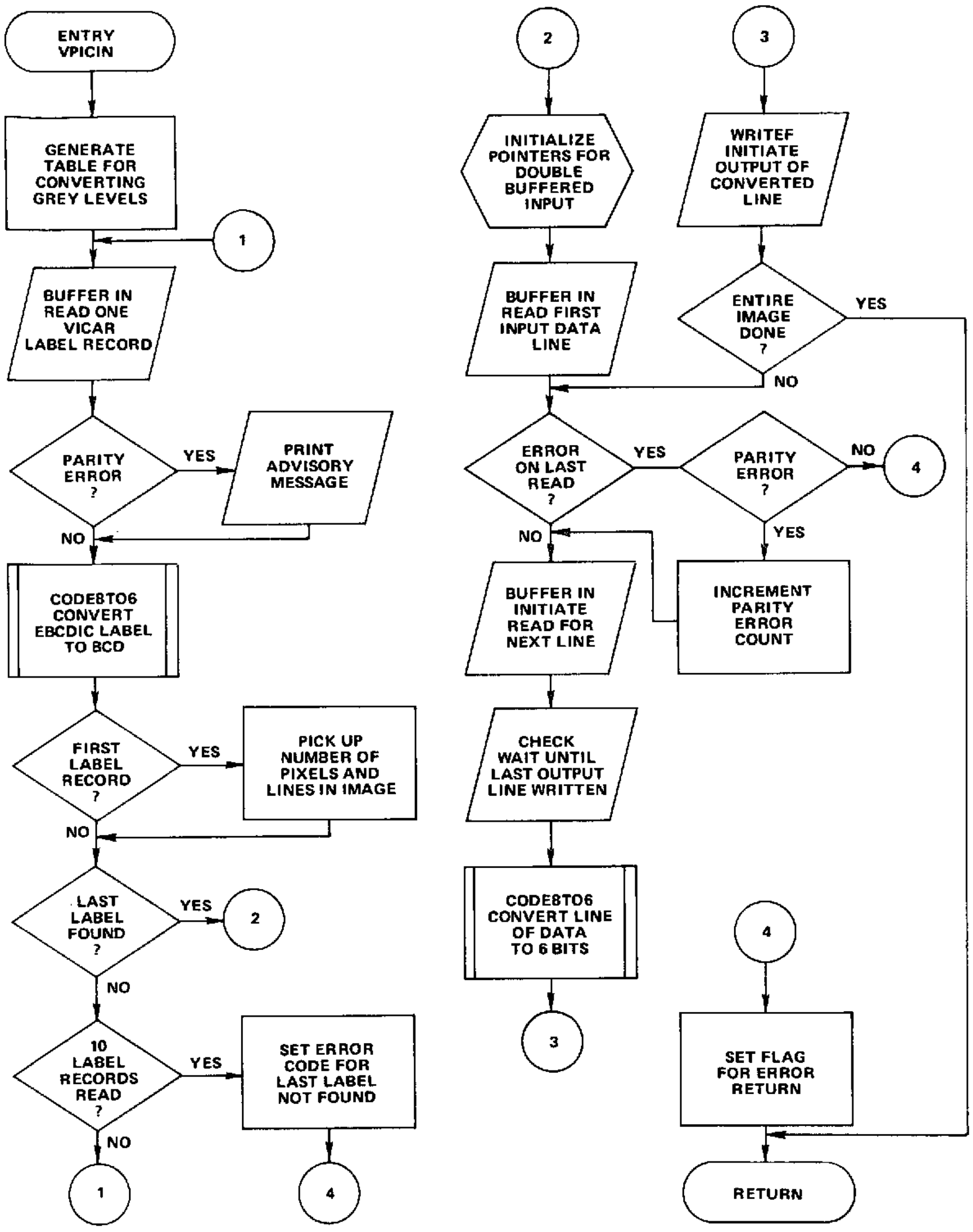

ENTRY VPICIN
GENERATE TABLE FOR CONVERTING GREY LEVELS
1
BUFFER IN READ ONE VICAR LABEL RECORD
PARITY ERROR ?
YES
PRINT ADVISORY MESSAGE
NO
CODE8TO6 CONVERT EBCDIC LABEL TO BCD
FIRST LABEL RECORD ?
YES
PICK UP NUMBER OF PIXELS AND LINES IN IMAGE
NO
LAST LABEL FOUND ?
YES
2
NO
10 LABEL RECORDS READ ?
YES
SET ERROR CODE FOR LAST LABEL NOT FOUND
NO
1
4
2
INITIALIZE POINTERS FOR DOUBLE BUFFERED INPUT
BUFFER IN READ FIRST INPUT DATA LINE
ERROR ON LAST READ ?
YES
NO
BUFFER IN INITIATE READ FOR NEXT LINE
CHECK WAIT UNTIL LAST OUTPUT LINE WRITTEN
CODE8TO6 CONVERT LINE OF DATA TO 6 BITS
3
3
WRITEF INITIATE OUTPUT OF CONVERTED LINE
ENTIRE IMAGE DONE ?
YES
NO
PARITY ERROR ?
NO
4
YES
INCREMENT PARITY ERROR COUNT
4
SET FLAG FOR ERROR RETURN
RETURN

Figure C-35. VPICIN Program Flowchart

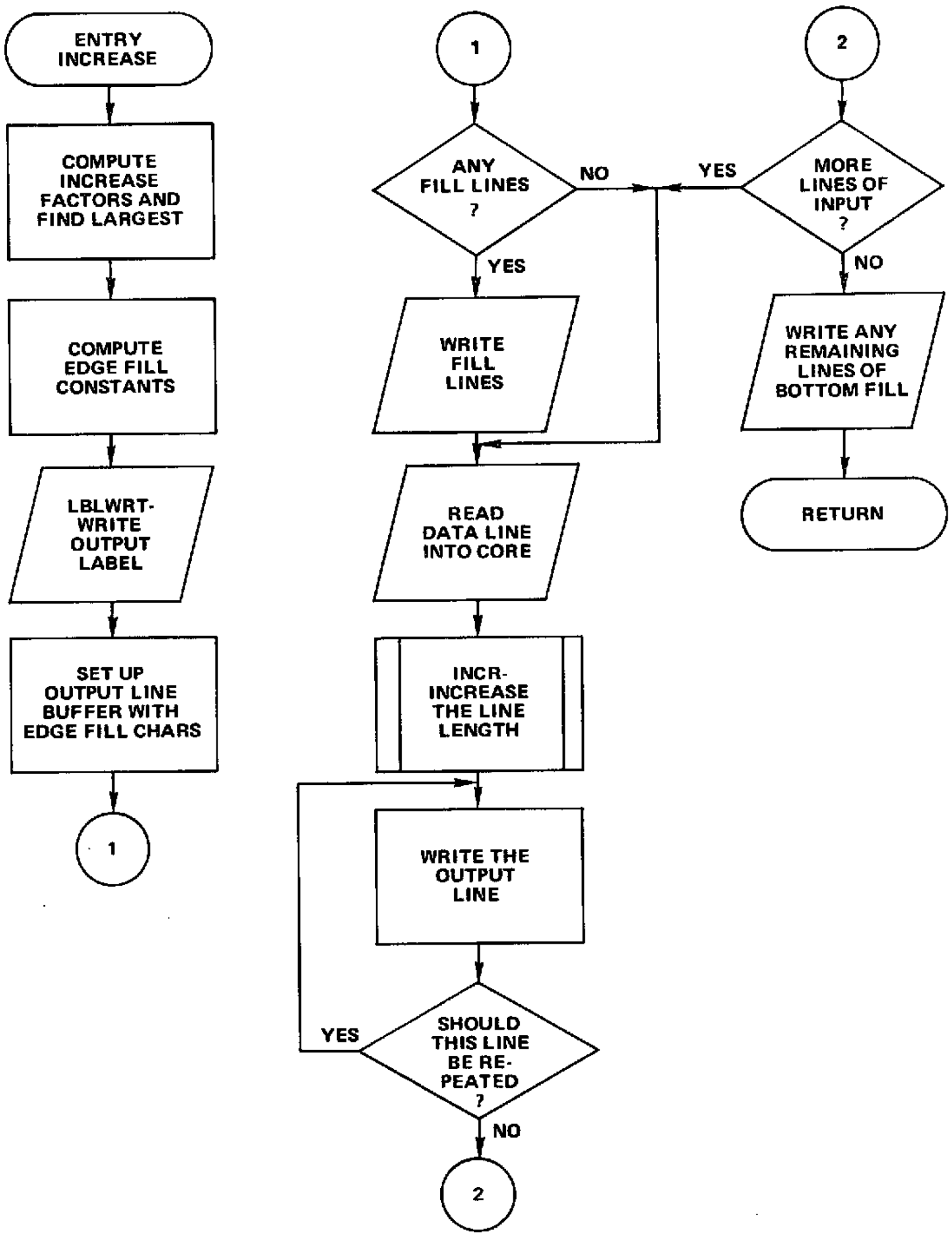

ENTRY INCREASE
COMPUTE INCREASE FACTORS AND FIND LARGEST
COMPUTE EDGE FILL CONSTANTS
LBLWRT-WRITE OUTPUT LABEL
SET UP OUTPUT LINE BUFFER WITH EDGE FILL CHARS
1
1
ANY FILL LINES ?
NO
YES
WRITE FILL LINES
READ DATA LINE INTO CORE
INCR-INCREASE THE LINE LENGTH
WRITE THE OUTPUT LINE
SHOULD THIS LINE BE RE-PEATED ?
YES
NO
2
2
MORE LINES OF INPUT ?
YES
NO
WRITE ANY REMAINING LINES OF BOTTOM FILL
RETURN

Figure C-36. INCREASE Program Flowchart (1 of 2)

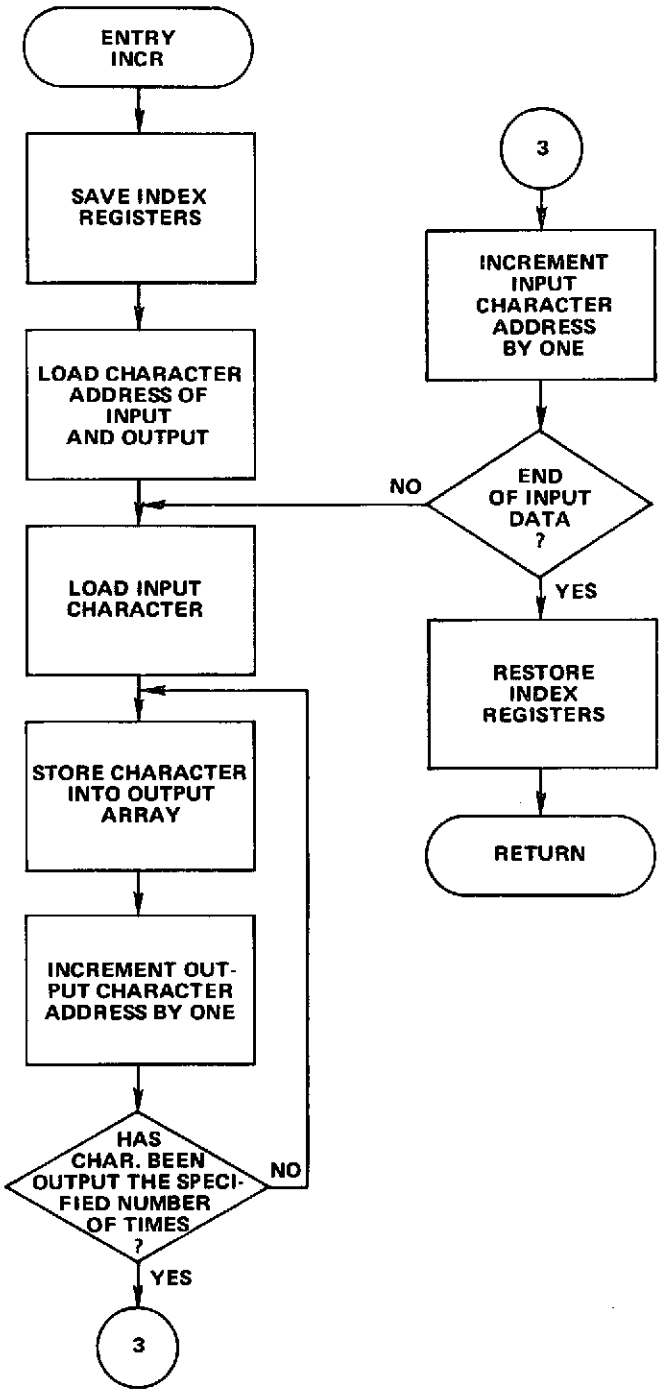
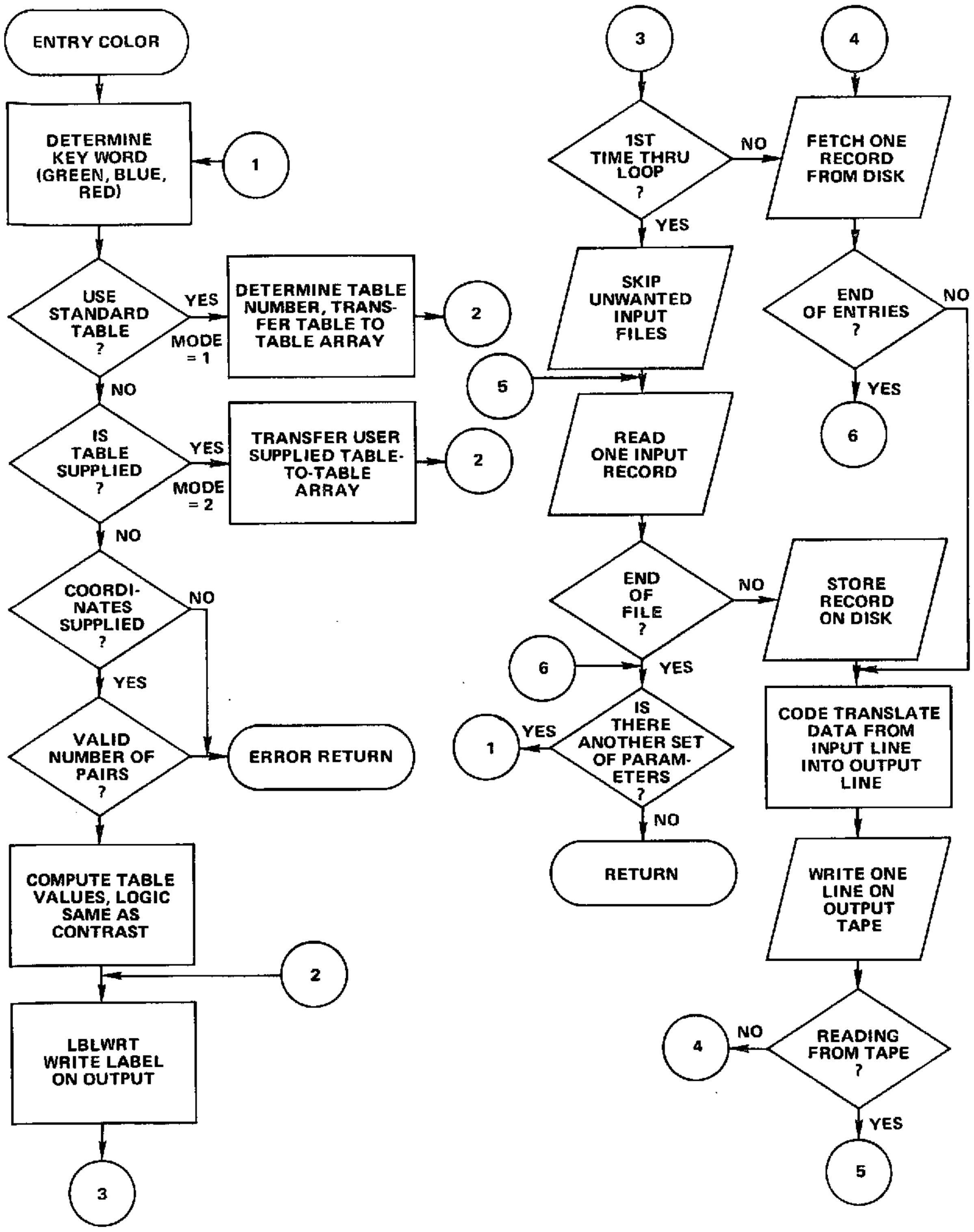

ENTRY
INCR
SAVE INDEX REGISTERS
LOAD CHARACTER ADDRESS OF INPUT AND OUTPUT
LOAD INPUT CHARACTER
STORE CHARACTER INTO OUTPUT ARRAY
INCREMENT OUTPUT CHARACTER ADDRESS BY ONE
HAS CHAR. BEEN OUTPUT THE SPECIFIED NUMBER OF TIMES ?
NO
YES
3
3
INCREMENT INPUT CHARACTER ADDRESS BY ONE
END OF INPUT DATA ?
NO
YES
RESTORE INDEX REGISTERS
RETURN

Figure C-36. INCREASE Program Flowchart (2 of 2)



ENTRY COLOR
DETERMINE KEY WORD (GREEN, BLUE, RED)
1
USE STANDARD TABLE ?
YES
MODE = 1
DETERMINE TABLE NUMBER, TRANSFER TABLE TO TABLE ARRAY
2
NO
IS TABLE SUPPLIED ?
YES
MODE = 2
TRANSFER USER SUPPLIED TABLE-TO-TABLE ARRAY
2
NO
COORDINATES SUPPLIED ?
NO
YES
VALID NUMBER OF PAIRS ?
ERROR RETURN
COMPUTE TABLE VALUES, LOGIC SAME AS CONTRAST
2
LBLWRT WRITE LABEL ON OUTPUT
3
3
1ST TIME THRU LOOP ?
NO
YES
SKIP UNWANTED INPUT FILES
5
READ ONE INPUT RECORD
END OF FILE ?
NO
YES
6
IS THERE ANOTHER SET OF PARAMETERS ?
YES
1
NO
RETURN
4
FETCH ONE RECORD FROM DISK
END OF ENTRIES ?
NO
YES
6
STORE RECORD ON DISK
CODE TRANSLATE DATA FROM INPUT LINE INTO OUTPUT LINE
WRITE ONE LINE ON OUTPUT TAPE
READING FROM TAPE ?
NO
4
YES
5

Figure C-37. COLOR Program Flowchart

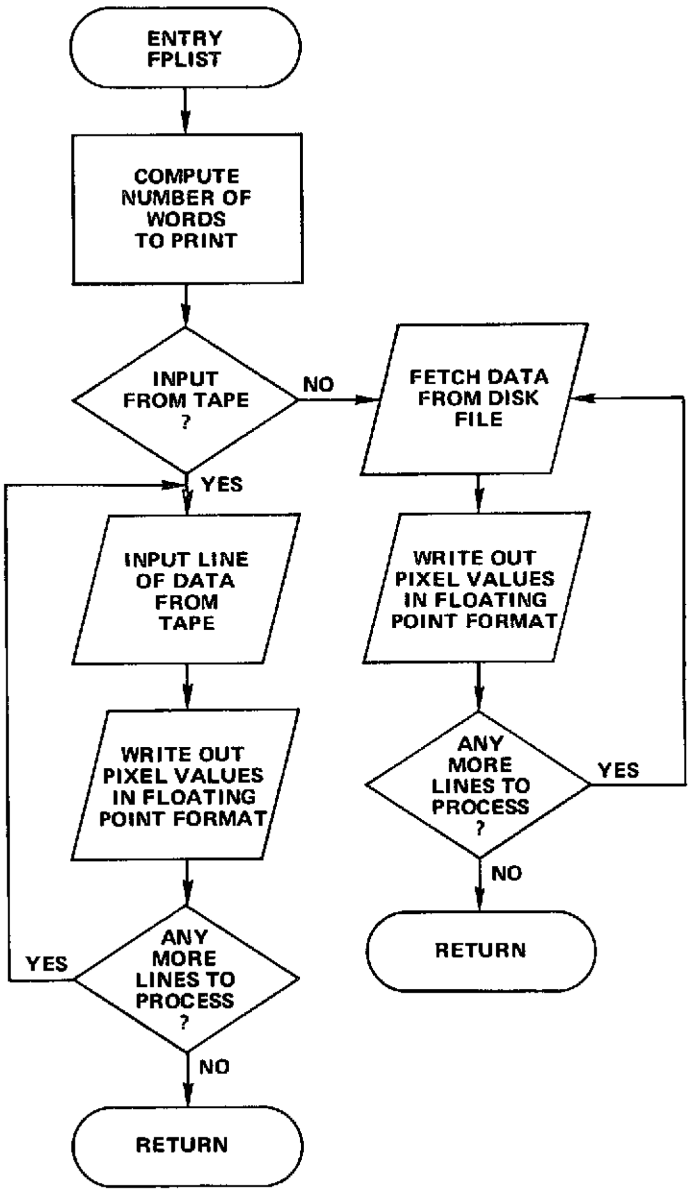

ENTRY FPLIST
COMPUTE NUMBER OF WORDS TO PRINT
INPUT FROM TAPE ?
NO
YES
FETCH DATA FROM DISK FILE
INPUT LINE OF DATA FROM TAPE
WRITE OUT PIXEL VALUES IN FLOATING POINT FORMAT
WRITE OUT PIXEL VALUES IN FLOATING POINT FORMAT
ANY MORE LINES TO PROCESS ?
YES
NO
RETURN
ANY MORE LINES TO PROCESS ?
YES
NO
RETURN

Figure C-38. FPLIST Program Flowchart

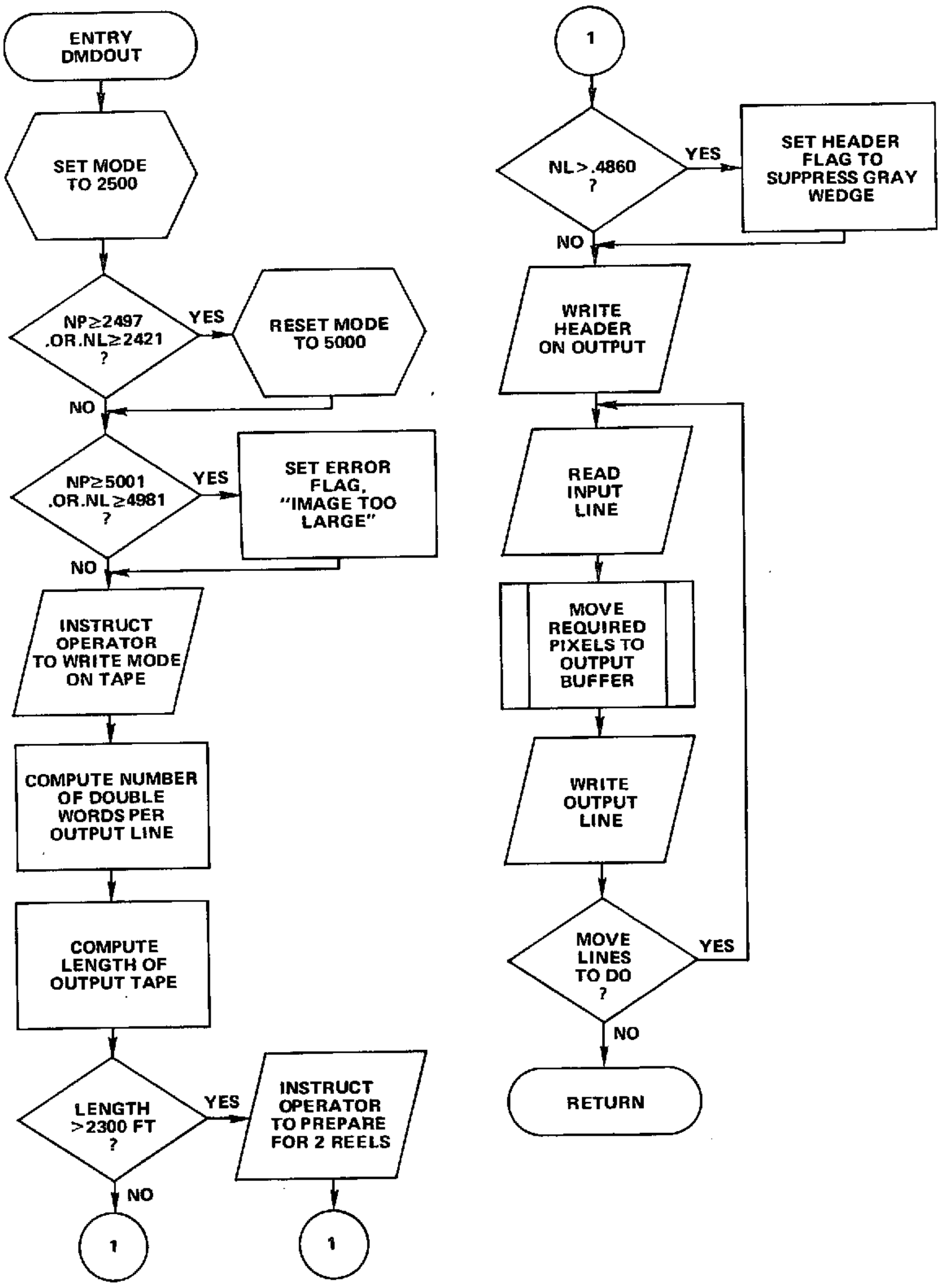
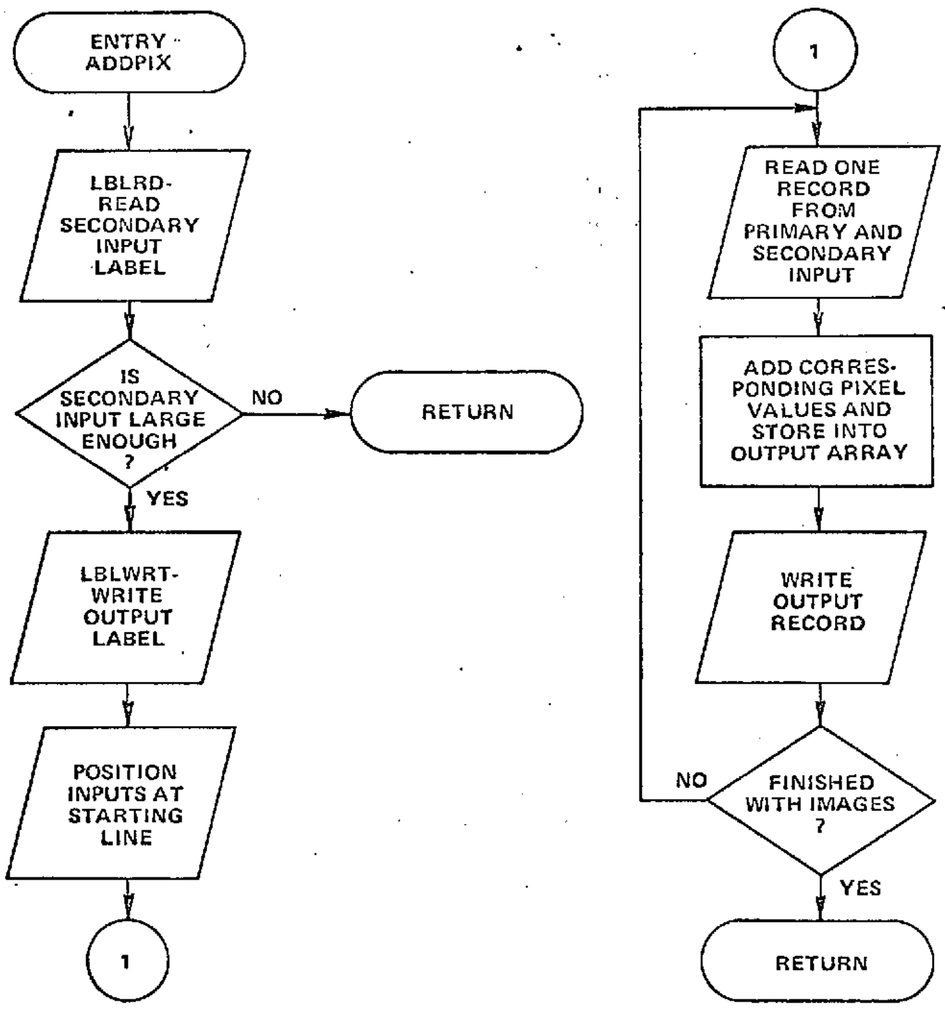

ENTRY DMDOUT
SET MODE TO 2500
NP≥2497 .OR.NL≥2421 ?
YES
RESET MODE TO 5000
NO
NP≥5001 .OR.NL≥4981 ?
YES
SET ERROR FLAG, "IMAGE TOO LARGE"
NO
INSTRUCT OPERATOR TO WRITE MODE ON TAPE
COMPUTE NUMBER OF DOUBLE WORDS PER OUTPUT LINE
COMPUTE LENGTH OF OUTPUT TAPE
LENGTH >2300 FT ?
YES
INSTRUCT OPERATOR TO PREPARE FOR 2 REELS
NO
1
1
1
NL>.4860 ?
YES
SET HEADER FLAG TO SUPPRESS GRAY WEDGE
NO
WRITE HEADER ON OUTPUT
READ INPUT LINE
MOVE REQUIRED PIXELS TO OUTPUT BUFFER
WRITE OUTPUT LINE
MOVE LINES TO DO ?
YES
NO
RETURN

Figure C-39. DMDOUT Program Flowchart

CS



ENTRY - ADDPIX
LBLRD- READ SECONDARY INPUT LABEL
IS SECONDARY INPUT LARGE ENOUGH ?
NO
RETURN
YES
LBLWRT- WRITE OUTPUT LABEL
POSITION INPUTS AT STARTING LINE
1
1
READ ONE RECORD FROM PRIMARY AND SECONDARY INPUT
ADD CORRES- PONDING PIXEL VALUES AND STORE INTO OUTPUT ARRAY
WRITE OUTPUT RECORD
FINISHED WITH IMAGES ?
NO
YES
RETURN

Figure C-40. ADDPIX Program Flowchart

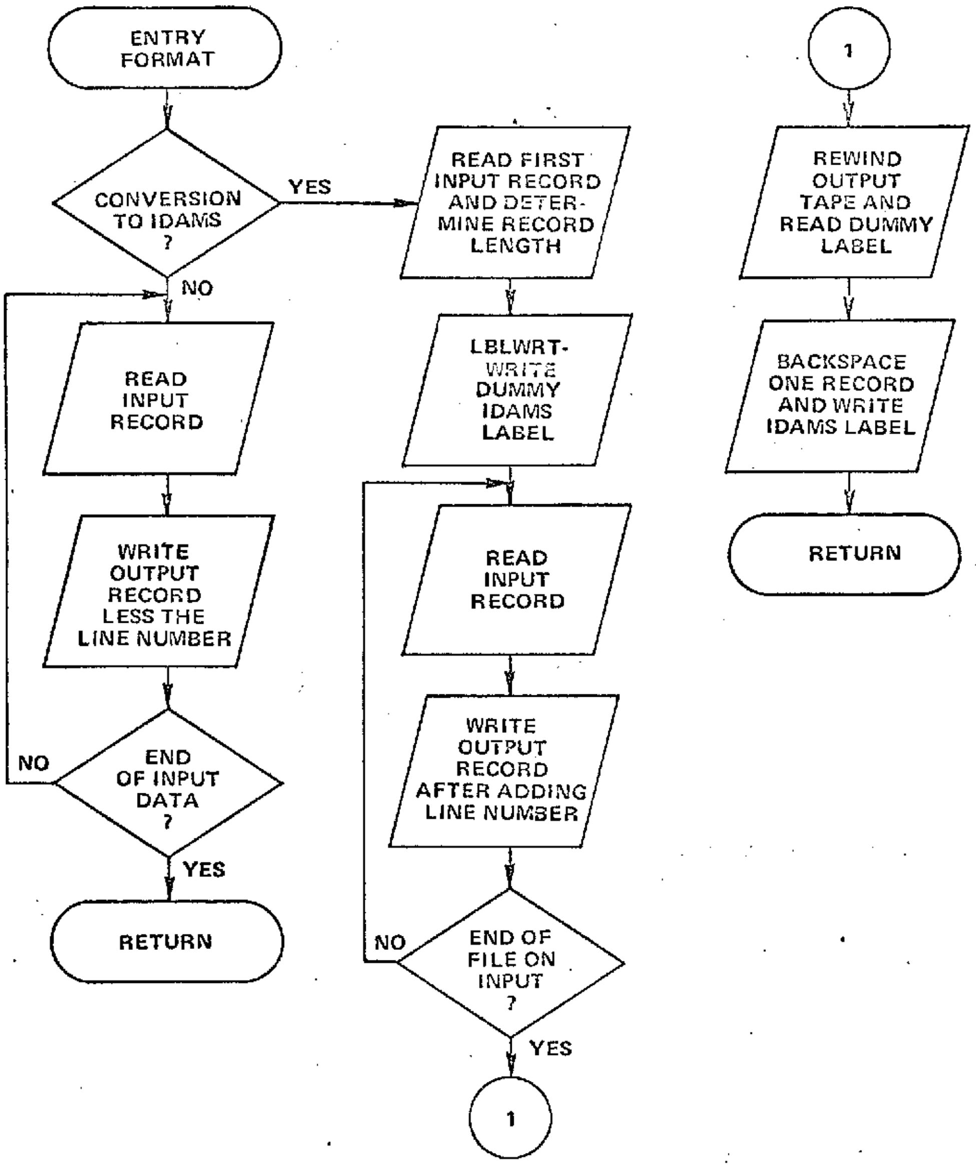

ENTRY FORMAT
CONVERSION TO IDAMS ?
YES
NO
READ INPUT RECORD
WRITE OUTPUT RECORD LESS THE LINE NUMBER
END OF INPUT DATA ?
NO
YES
RETURN
READ FIRST INPUT RECORD AND DETERMINE RECORD LENGTH
LBLWRT-WRITE DUMMY IDAMS LABEL
READ INPUT RECORD
WRITE OUTPUT RECORD AFTER ADDING LINE NUMBER
END OF FILE ON INPUT ?
NO
YES
1
1
REWIND OUTPUT TAPE AND READ DUMMY LABEL
BACKSPACE ONE RECORD AND WRITE IDAMS LABEL
RETURN

Figure C-41. FORMAT Program Flowchart

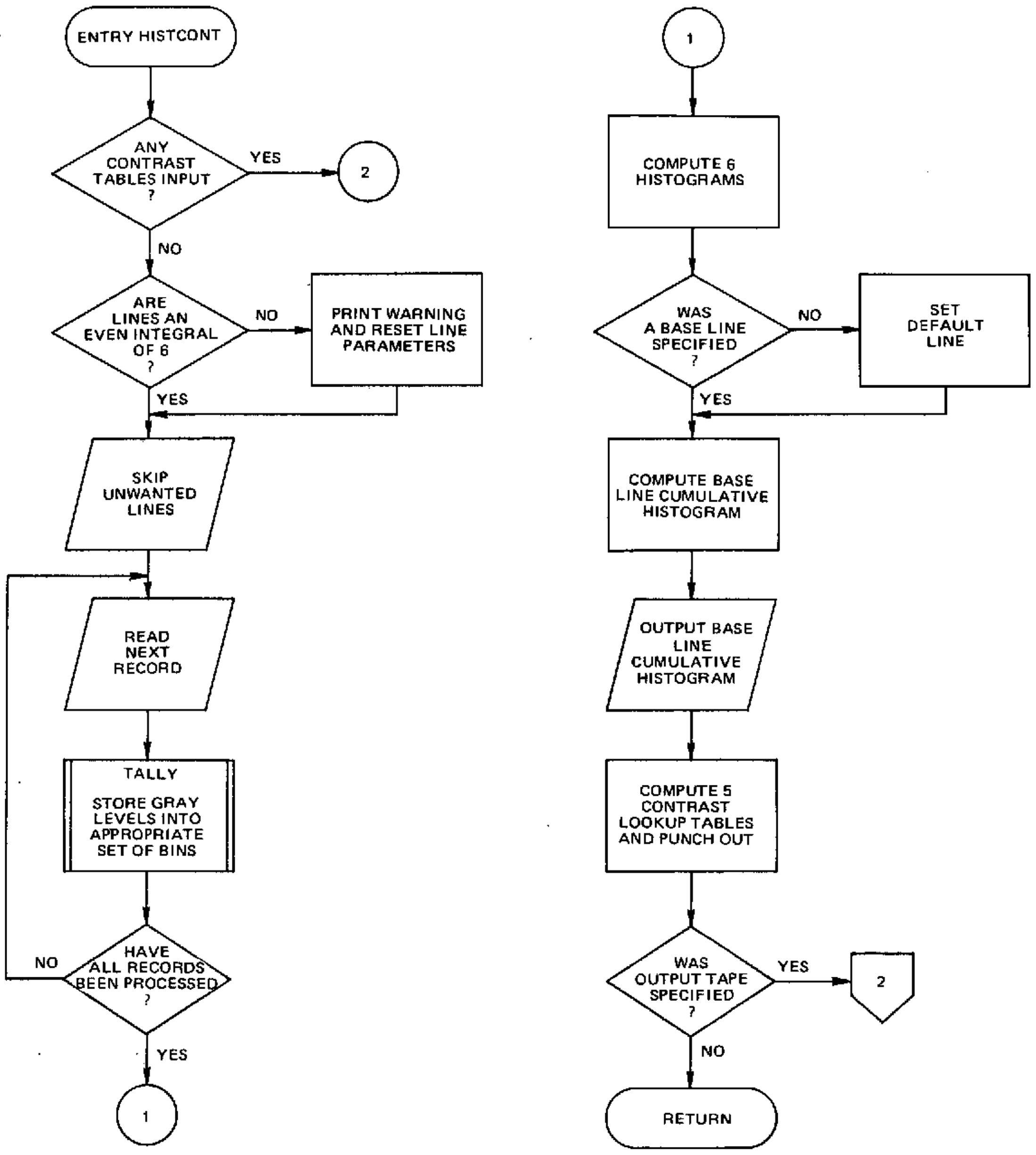

ENTRY HISTCONT
ANY CONTRAST TABLES INPUT ?
YES
2
NO
ARE LINES AN EVEN INTEGRAL OF 6 ?
NO
PRINT WARNING AND RESET LINE PARAMETERS
YES
SKIP UNWANTED LINES
READ NEXT RECORD
TALLY
STORE GRAY LEVELS INTO APPROPRIATE SET OF BINS
HAVE ALL RECORDS BEEN PROCESSED ?
NO
YES
1
1
COMPUTE 6 HISTOGRAMS
WAS A BASE LINE SPECIFIED ?
NO
SET DEFAULT LINE
YES
COMPUTE BASE LINE CUMULATIVE HISTOGRAM
OUTPUT BASE LINE CUMULATIVE HISTOGRAM
COMPUTE 5 CONTRAST LOOKUP TABLES AND PUNCH OUT
WAS OUTPUT TAPE SPECIFIED ?
YES
2
NO
RETURN

Figure C-42. HISTCONT Program Flowchart (1 of 2)

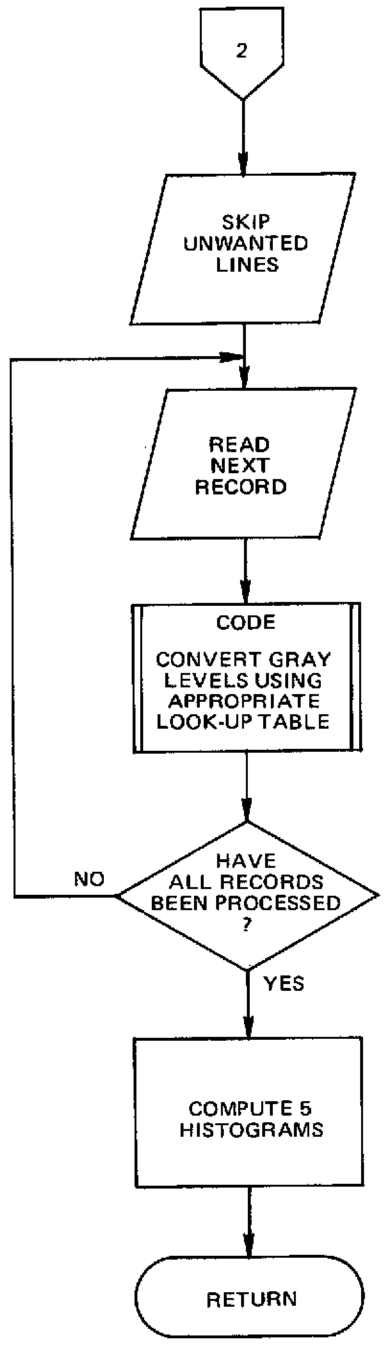

2
SKIP UNWANTED LINES
READ NEXT RECORD
CODE
CONVERT GRAY LEVELS USING APPROPRIATE LOOK-UP TABLE
HAVE ALL RECORDS BEEN PROCESSED ?
NO
YES
COMPUTE 5 HISTOGRAMS
RETURN

Figure C-42. HISTCONT Program Flowchart (2 of 2)

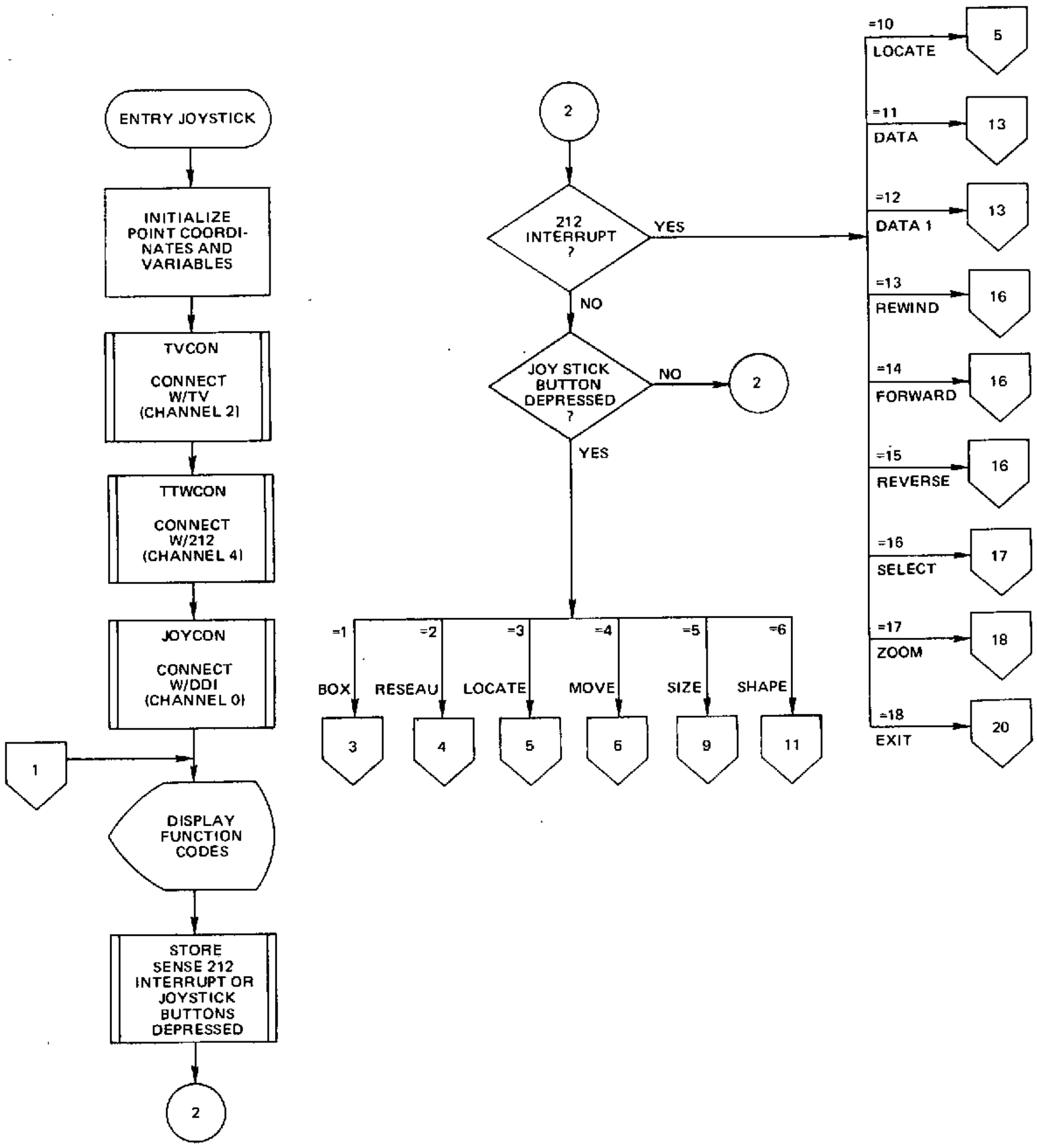
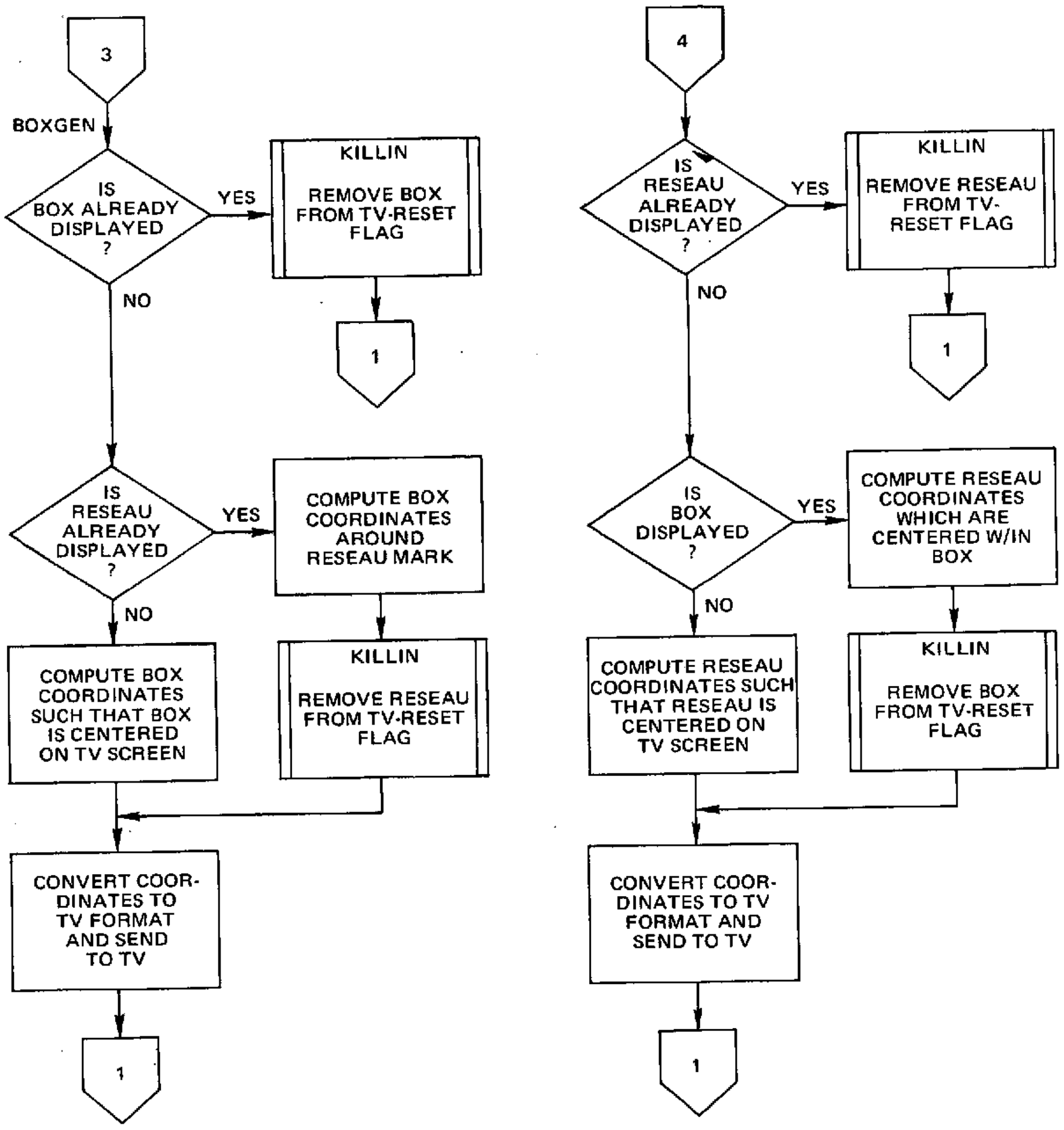


Figure C-43. JOYSTICK Program Flowchart (1 of 11)



3
BOXGEN
IS BOX ALREADY DISPLAYED ?
YES
KILLIN
REMOVE BOX FROM TV-RESET FLAG
1
NO
IS RESEAU ALREADY DISPLAYED ?
YES
COMPUTE BOX COORDINATES AROUND RESEAU MARK
KILLIN
REMOVE RESEAU FROM TV-RESET FLAG
NO
COMPUTE BOX COORDINATES SUCH THAT BOX IS CENTERED ON TV SCREEN
CONVERT COOR-DINATES TO TV FORMAT AND SEND TO TV
1
4
IS RESEAU ALREADY DISPLAYED ?
YES
KILLIN
REMOVE RESEAU FROM TV-RESET FLAG
1
NO
IS BOX DISPLAYED ?
YES
COMPUTE RESEAU COORDINATES WHICH ARE CENTERED W/IN BOX
KILLIN
REMOVE BOX FROM TV-RESET FLAG
NO
COMPUTE RESEAU COORDINATES SUCH THAT RESEAU IS CENTERED ON TV SCREEN
CONVERT COOR-DINATES TO TV FORMAT AND SEND TO TV
1

Figure C-43. JOYSTICK Program Flowchart (2 of 11)

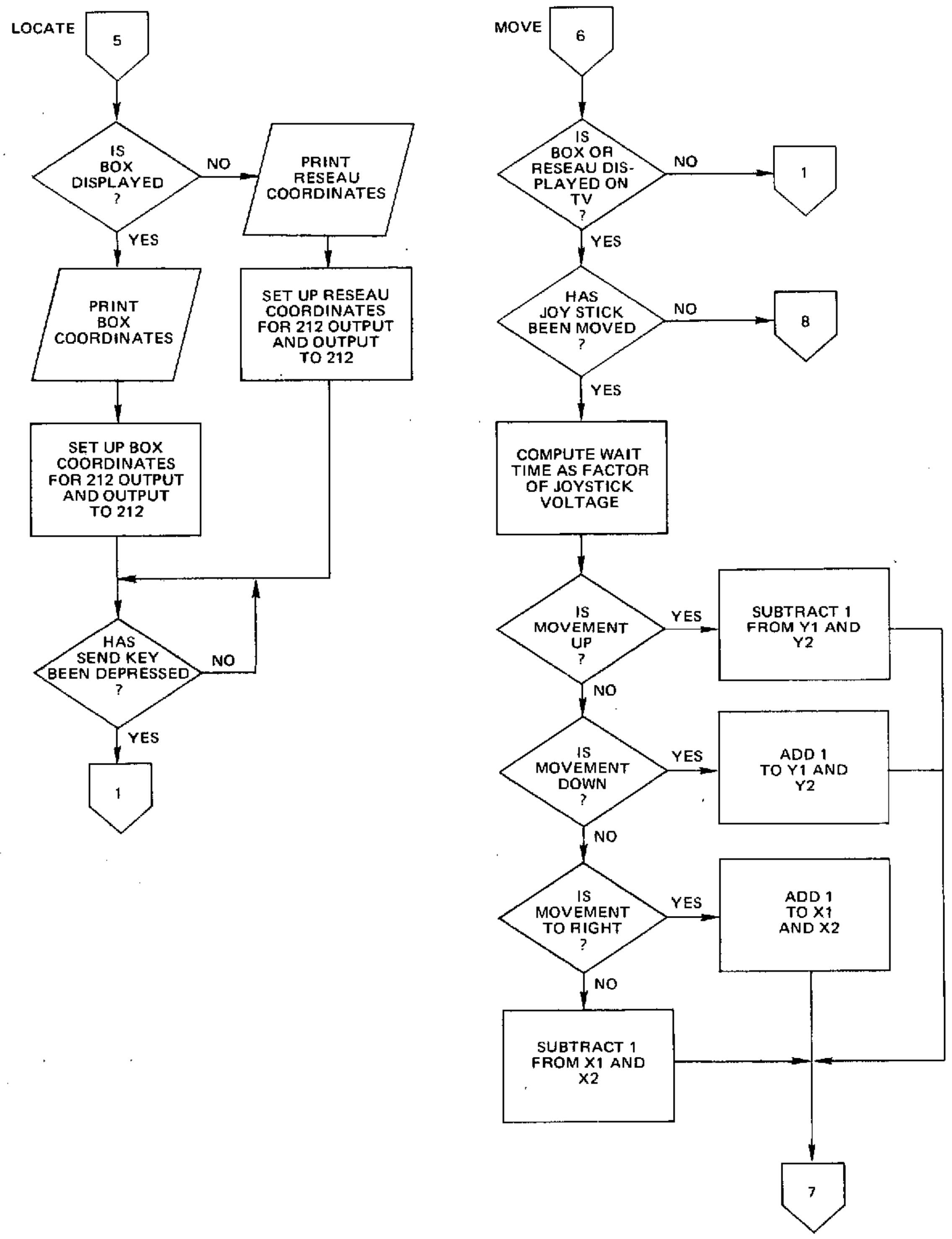

LOCATE
5
IS BOX DISPLAYED ?
NO
PRINT RESEAU COORDINATES
YES
PRINT BOX COORDINATES
SET UP RESEAU COORDINATES FOR 212 OUTPUT AND OUTPUT TO 212
SET UP BOX COORDINATES FOR 212 OUTPUT AND OUTPUT TO 212
HAS SEND KEY BEEN DEPRESSED ?
NO
YES
1
MOVE
6
IS BOX OR RESEAU DIS-PLAYED ON TV ?
NO
1
YES
HAS JOY STICK BEEN MOVED ?
NO
8
YES
COMPUTE WAIT TIME AS FACTOR OF JOYSTICK VOLTAGE
IS MOVEMENT UP ?
YES
SUBTRACT 1 FROM Y1 AND Y2
NO
IS MOVEMENT DOWN ?
YES
ADD 1 TO Y1 AND Y2
NO
IS MOVEMENT TO RIGHT ?
YES
ADD 1 TO X1 AND X2
NO
SUBTRACT 1 FROM X1 AND X2
7

Figure C-43. JOYSTICK Program Flowchart (3 of 11)

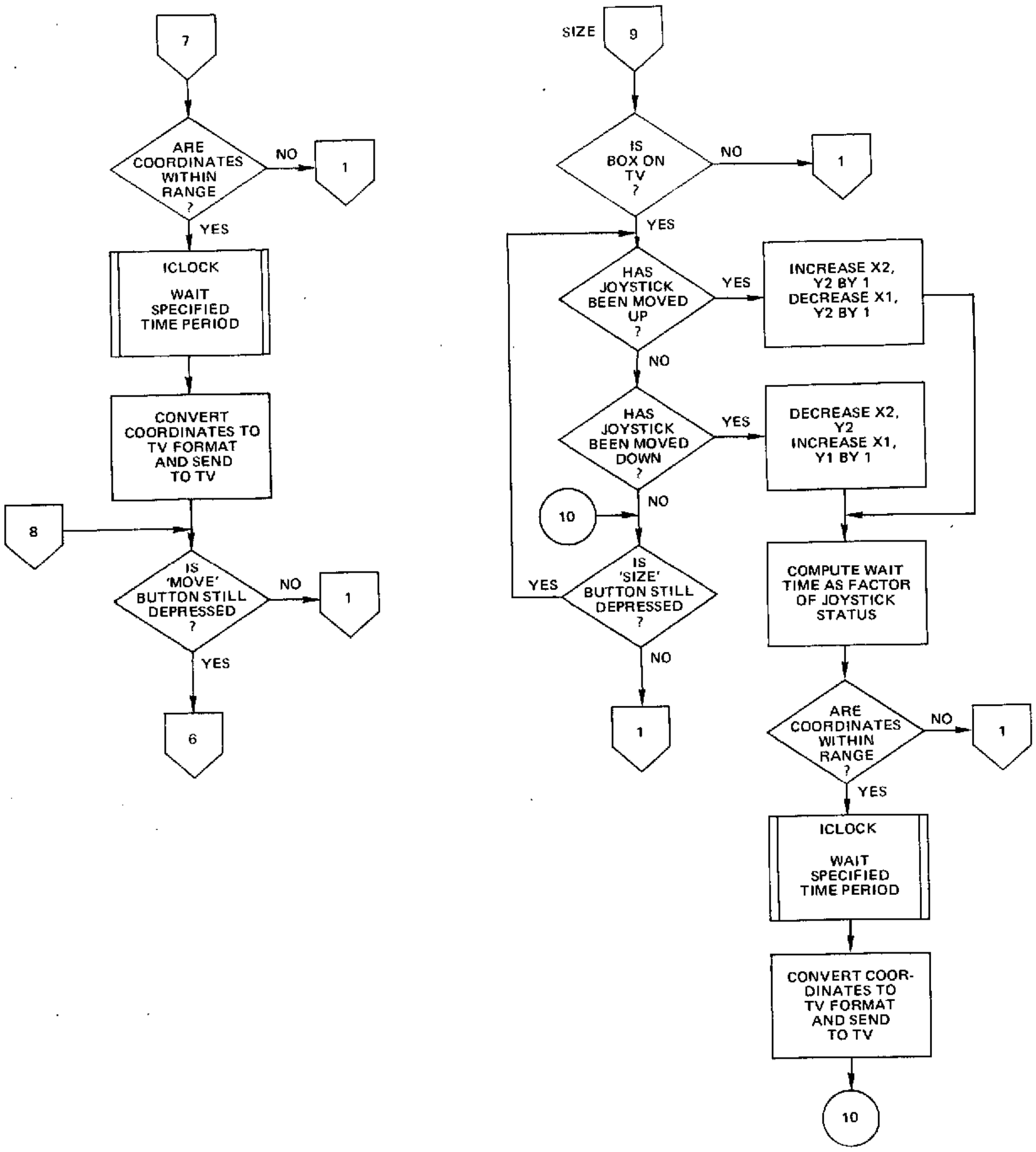

7
ARE COORDINATES WITHIN RANGE ?
NO
1
YES
ICLOCK
WAIT SPECIFIED TIME PERIOD
CONVERT COORDINATES TO TV FORMAT AND SEND TO TV
8
IS 'MOVE' BUTTON STILL DEPRESSED ?
NO
1
YES
6
SIZE
9
IS BOX ON TV ?
NO
1
YES
HAS JOYSTICK BEEN MOVED UP ?
YES
INCREASE X2, Y2 BY 1 DECREASE X1, Y2 BY 1
NO
HAS JOYSTICK BEEN MOVED DOWN ?
YES
DECREASE X2, Y2 INCREASE X1, Y1 BY 1
NO
10
IS 'SIZE' BUTTON STILL DEPRESSED ?
YES
NO
1
COMPUTE WAIT TIME AS FACTOR OF JOYSTICK STATUS
ARE COORDINATES WITHIN RANGE ?
NO
1
YES
ICLOCK
WAIT SPECIFIED TIME PERIOD
CONVERT COORDINATES TO TV FORMAT AND SEND TO TV
10

Figure C-43. JOYSTICK Program Flowchart (4 of 11)

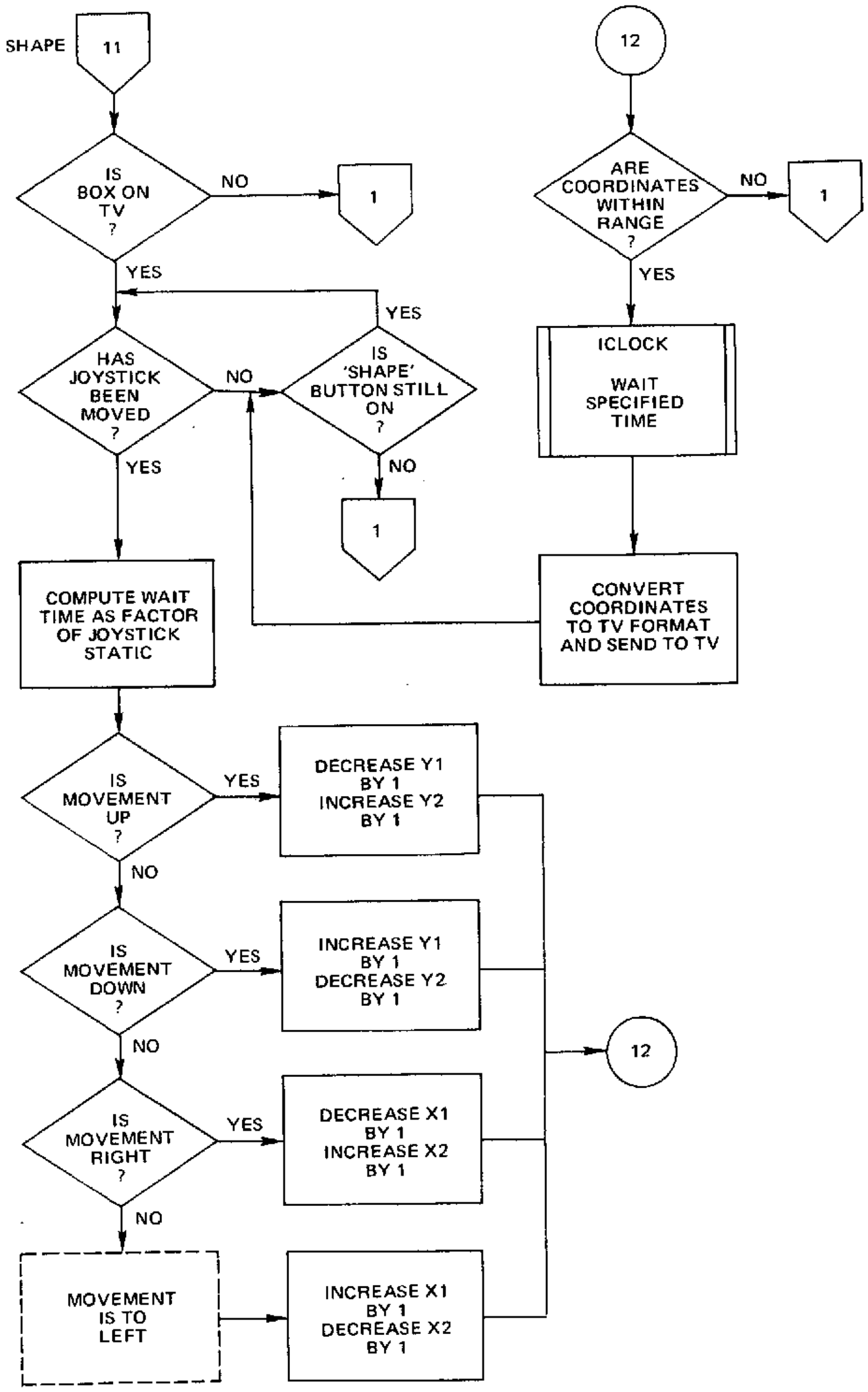
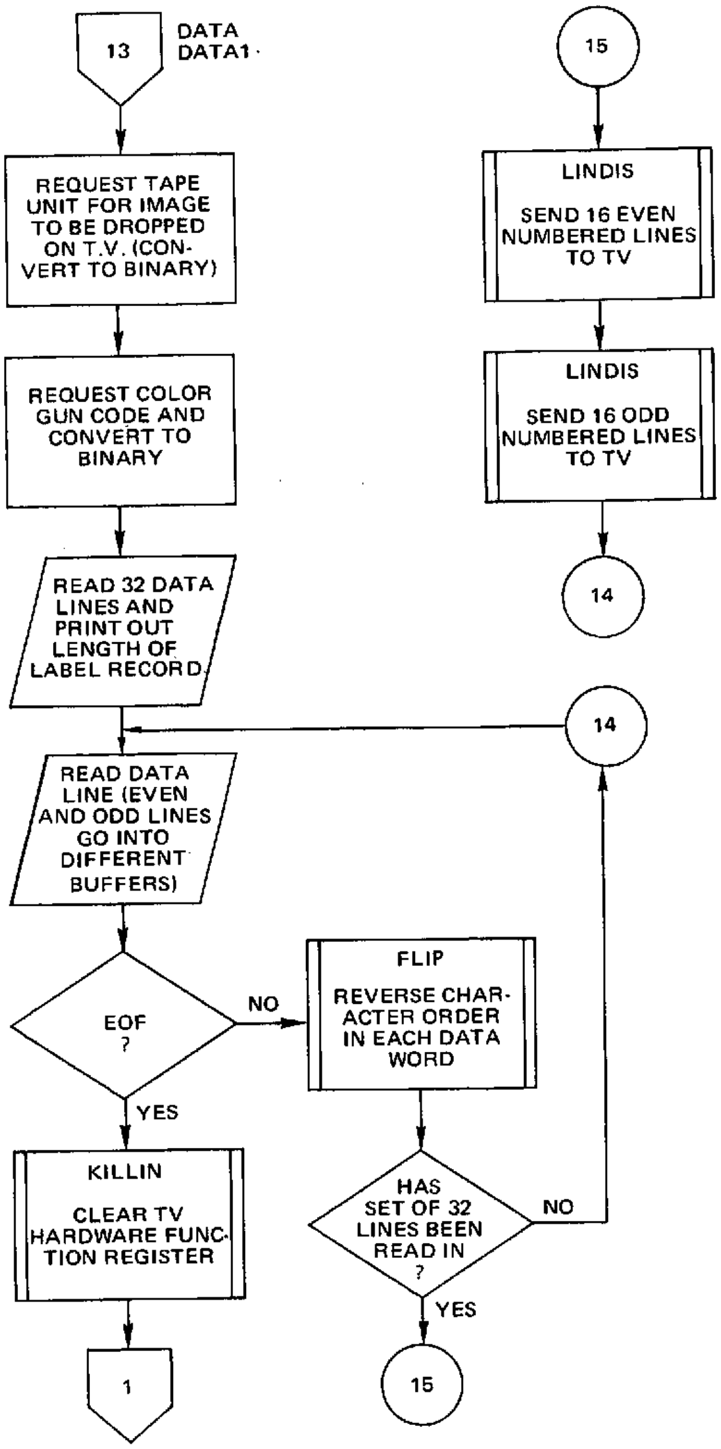

SHAPE
11
12
IS BOX ON TV ?
NO
1
ARE COORDINATES WITHIN RANGE ?
NO
1
YES
YES
HAS JOYSTICK BEEN MOVED ?
NO
IS 'SHAPE' BUTTON STILL ON ?
YES
ICLOCK
WAIT SPECIFIED TIME
YES
NO
1
COMPUTE WAIT TIME AS FACTOR OF JOYSTICK STATIC
CONVERT COORDINATES TO TV FORMAT AND SEND TO TV
IS MOVEMENT UP ?
YES
DECREASE Y1 BY 1 INCREASE Y2 BY 1
NO
IS MOVEMENT DOWN ?
YES
INCREASE Y1 BY 1 DECREASE Y2 BY 1
12
NO
IS MOVEMENT RIGHT ?
YES
DECREASE X1 BY 1 INCREASE X2 BY 1
NO
MOVEMENT IS TO LEFT
INCREASE X1 BY 1 DECREASE X2 BY 1

Figure C-43. JOYSTICK Program Flowchart (5 of 11)



13
DATA
DATA1
REQUEST TAPE UNIT FOR IMAGE TO BE DROPPED ON T.V. (CONVERT TO BINARY)
REQUEST COLOR GUN CODE AND CONVERT TO BINARY
READ 32 DATA LINES AND PRINT OUT LENGTH OF LABEL RECORD
14
READ DATA LINE (EVEN AND ODD LINES GO INTO DIFFERENT BUFFERS)
EOF ?
NO
YES
KILLIN
CLEAR TV HARDWARE FUNCTION REGISTER
1
FLIP
REVERSE CHARACTER ORDER IN EACH DATA WORD
HAS SET OF 32 LINES BEEN READ IN ?
NO
YES
15
15
LINDIS
SEND 16 EVEN NUMBERED LINES TO TV
LINDIS
SEND 16 ODD NUMBERED LINES TO TV
14

Figure C-43. JOYSTICK Program Flowchart (6 of 11)

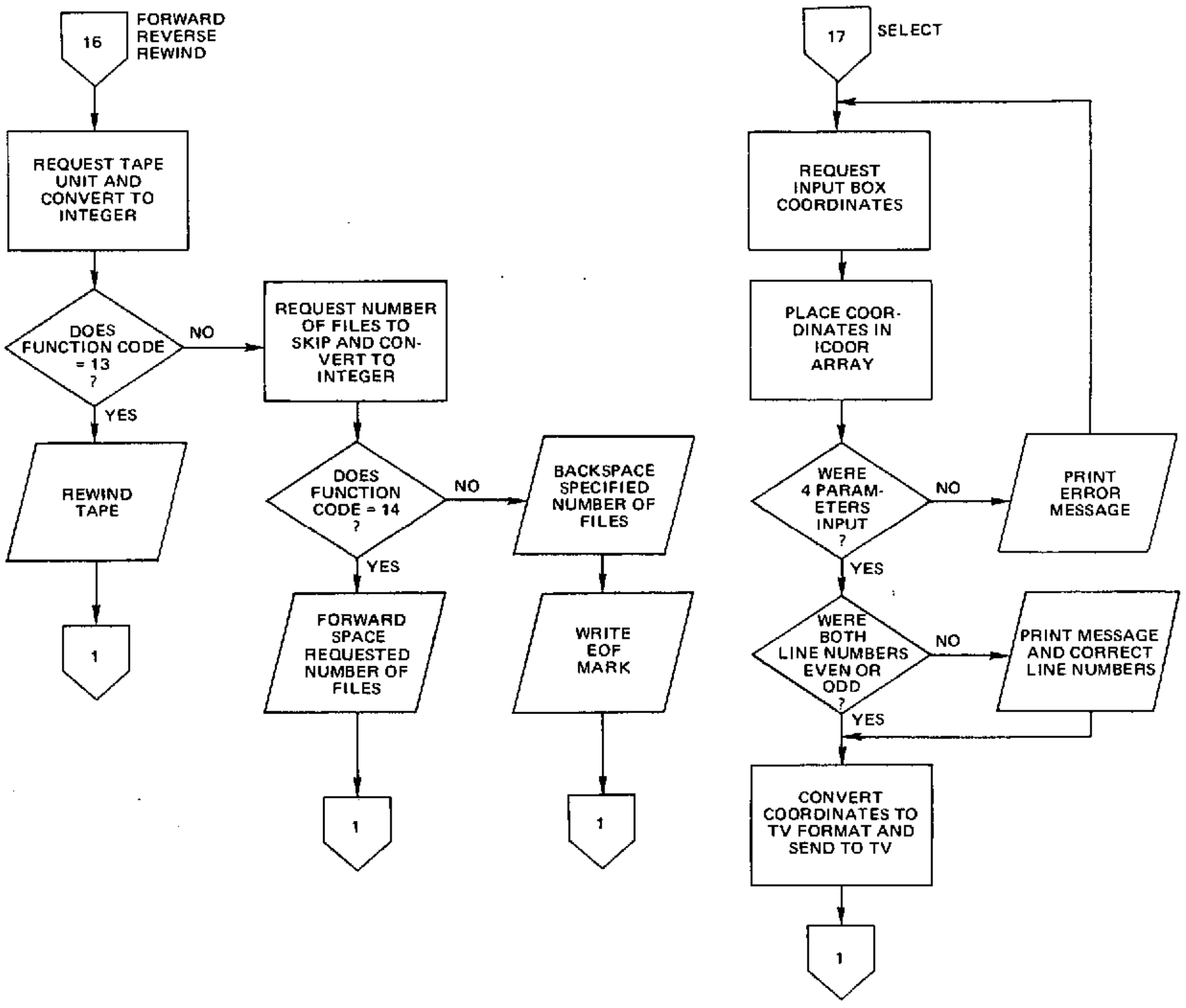

16
FORWARD
REVERSE
REWIND
REQUEST TAPE UNIT AND CONVERT TO INTEGER
DOES FUNCTION CODE = 13 ?
NO
YES
REWIND TAPE
1
REQUEST NUMBER OF FILES TO SKIP AND CONVERT TO INTEGER
DOES FUNCTION CODE = 14 ?
NO
YES
FORWARD SPACE REQUESTED NUMBER OF FILES
1
BACKSPACE SPECIFIED NUMBER OF FILES
WRITE EOF MARK
1
17
SELECT
REQUEST INPUT BOX COORDINATES
PLACE COORDINATES IN ICOOR ARRAY
WERE 4 PARAMETERS INPUT ?
NO
PRINT ERROR MESSAGE
YES
WERE BOTH LINE NUMBERS EVEN OR ODD ?
NO
PRINT MESSAGE AND CORRECT LINE NUMBERS
YES
CONVERT COORDINATES TO TV FORMAT AND SEND TO TV
1

Figure C-43. JOYSTICK Program Flowchart (7 of 11)

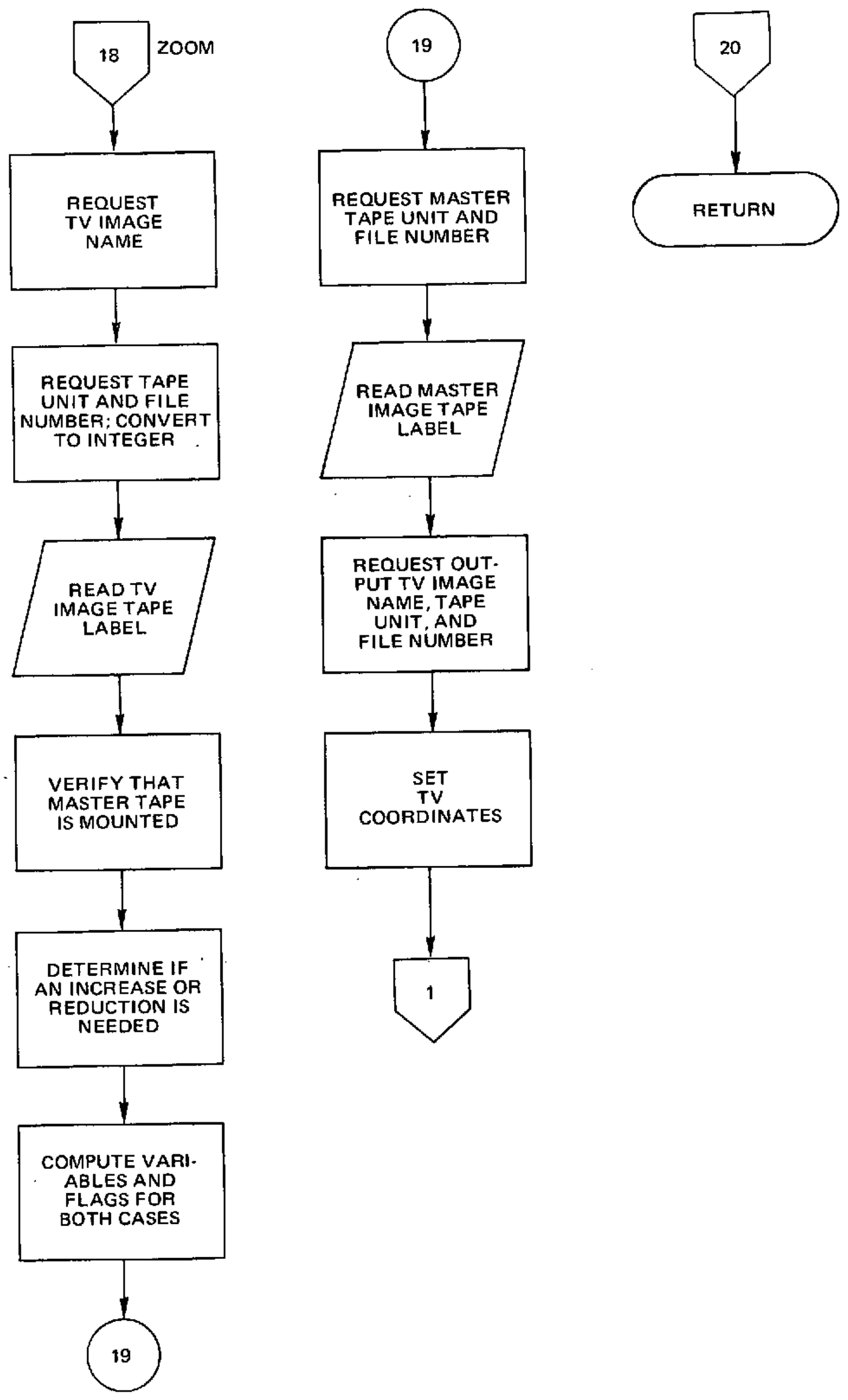

18
ZOOM
REQUEST TV IMAGE NAME
REQUEST TAPE UNIT AND FILE NUMBER; CONVERT TO INTEGER
READ TV IMAGE TAPE LABEL
VERIFY THAT MASTER TAPE IS MOUNTED
DETERMINE IF AN INCREASE OR REDUCTION IS NEEDED
COMPUTE VARIABLES AND FLAGS FOR BOTH CASES
19
19
REQUEST MASTER TAPE UNIT AND FILE NUMBER
READ MASTER IMAGE TAPE LABEL
REQUEST OUTPUT TV IMAGE NAME, TAPE UNIT, AND FILE NUMBER
SET TV COORDINATES
1
20
RETURN

Figure C-43. JOYSTICK Program Flowchart (8 of 11)

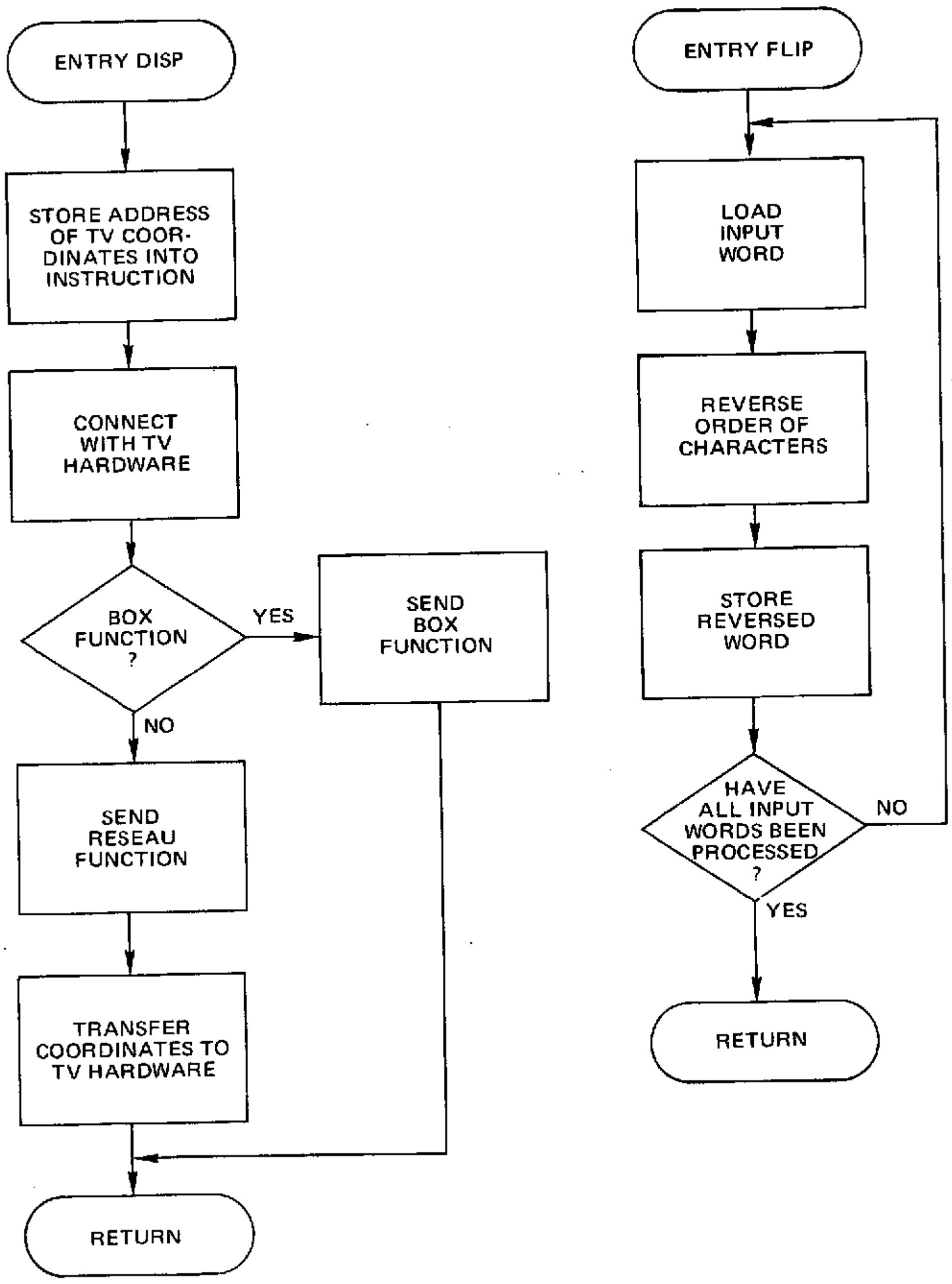


Figure C-43. JOYSTICK Program Flowchart (9 of 11)

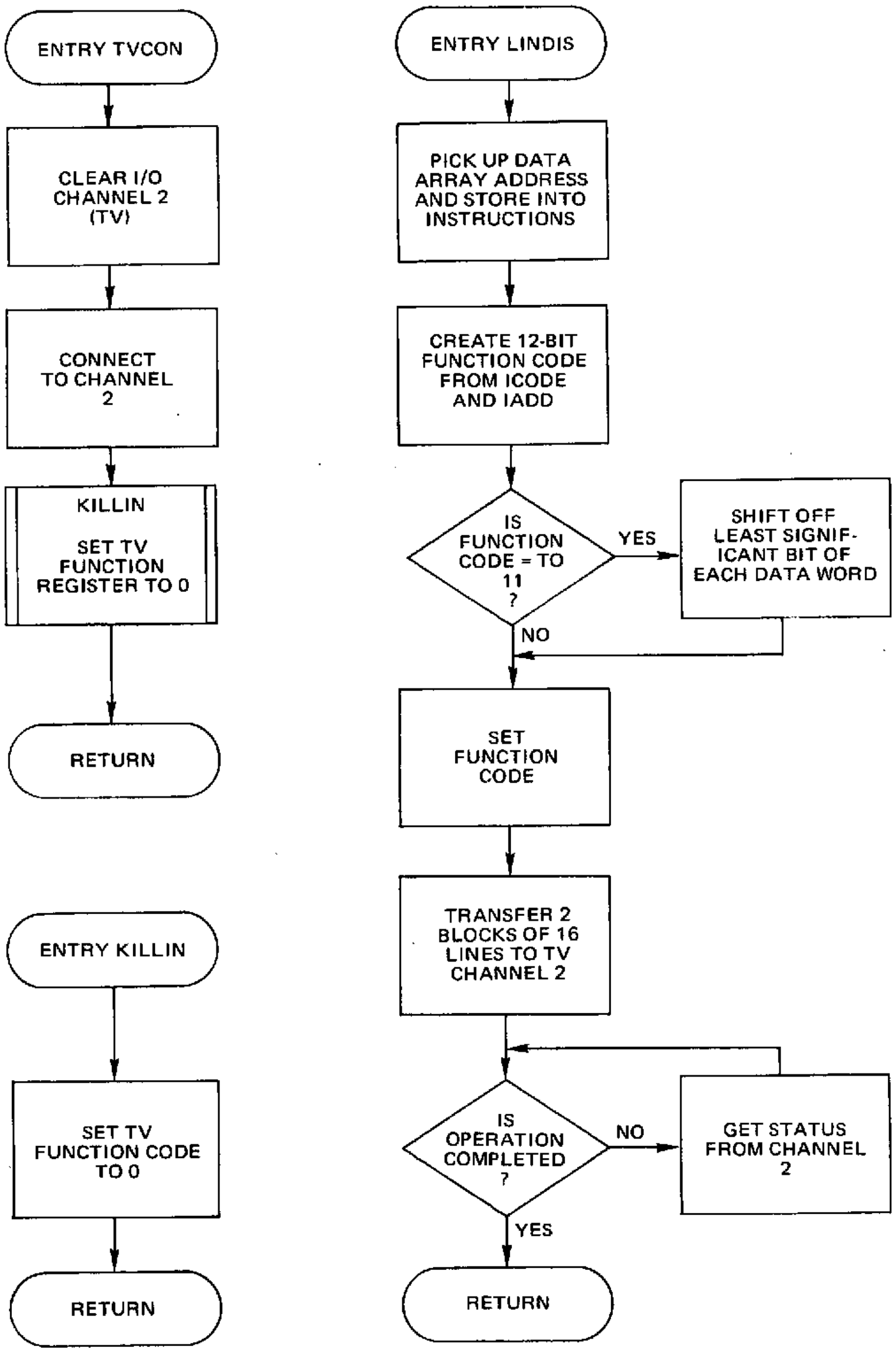
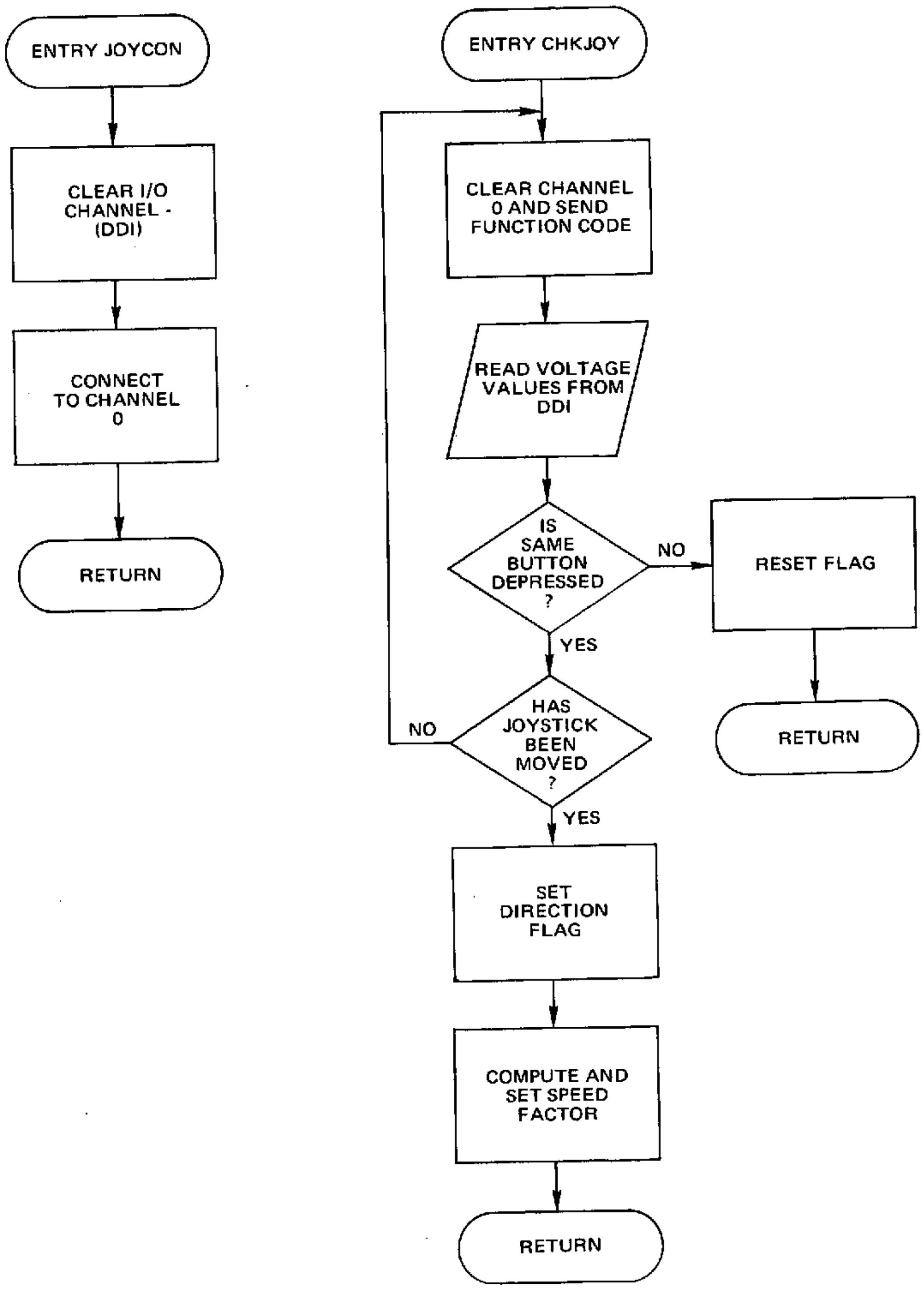

ENTRY TVCON
CLEAR I/O CHANNEL 2 (TV)
CONNECT TO CHANNEL 2
KILLIN
SET TV FUNCTION REGISTER TO 0
RETURN
ENTRY KILLIN
SET TV FUNCTION CODE TO 0
RETURN
ENTRY LINDIS
PICK UP DATA ARRAY ADDRESS AND STORE INTO INSTRUCTIONS
CREATE 12-BIT FUNCTION CODE FROM ICODE AND IADD
IS FUNCTION CODE = TO 11 ?
YES
SHIFT OFF LEAST SIGNIF-ICANT BIT OF EACH DATA WORD
NO
SET FUNCTION CODE
TRANSFER 2 BLOCKS OF 16 LINES TO TV CHANNEL 2
IS OPERATION COMPLETED ?
NO
GET STATUS FROM CHANNEL 2
YES
RETURN

Figure C-43. JOYSTICK Program Flowchart (10 of 11)



ENTRY JOYCON
CLEAR I/O CHANNEL - (DDI)
CONNECT TO CHANNEL 0
RETURN
ENTRY CHKJOY
CLEAR CHANNEL 0 AND SEND FUNCTION CODE
READ VOLTAGE VALUES FROM DDI
IS SAME BUTTON DEPRESSED ?
NO
RESET FLAG
RETURN
YES
HAS JOYSTICK BEEN MOVED ?
NO
YES
SET DIRECTION FLAG
COMPUTE AND SET SPEED FACTOR
RETURN

Figure C-43. JOYSTICK Program Flowchart (11 of 11)

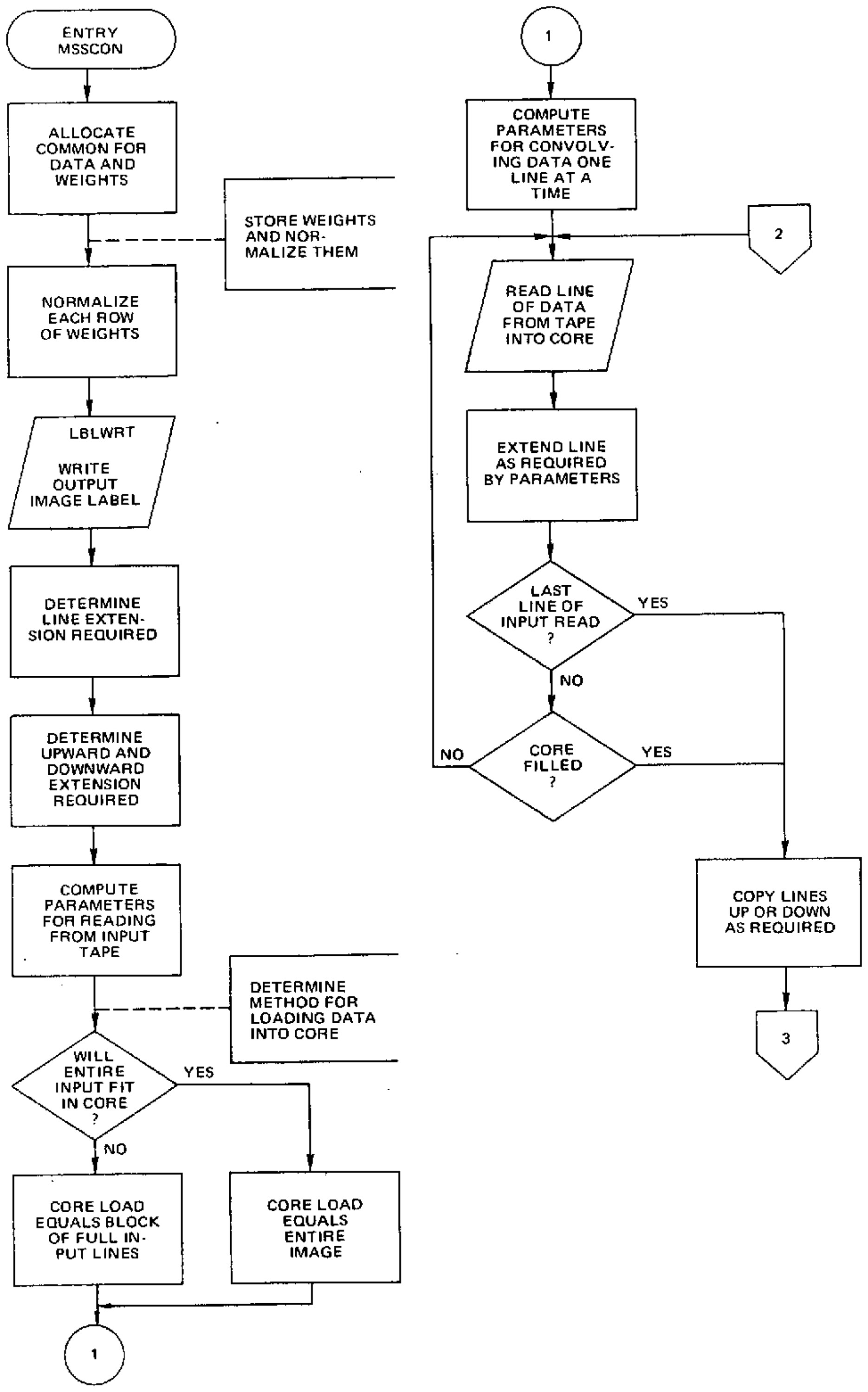

ENTRY MSSCON
ALLOCATE COMMON FOR DATA AND WEIGHTS
STORE WEIGHTS AND NOR-MALIZE THEM
NORMALIZE EACH ROW OF WEIGHTS
LBLWRT
WRITE OUTPUT IMAGE LABEL
DETERMINE LINE EXTEN-SION REQUIRED
DETERMINE UPWARD AND DOWNWARD EXTENSION REQUIRED
COMPUTE PARAMETERS FOR READING FROM INPUT TAPE
DETERMINE METHOD FOR LOADING DATA INTO CORE
WILL ENTIRE INPUT FIT IN CORE ?
YES
NO
CORE LOAD EQUALS BLOCK OF FULL IN-PUT LINES
CORE LOAD EQUALS ENTIRE IMAGE
1
1
COMPUTE PARAMETERS FOR CONVOLV-ING DATA ONE LINE AT A TIME
2
READ LINE OF DATA FROM TAPE INTO CORE
EXTEND LINE AS REQUIRED BY PARAMETERS
LAST LINE OF INPUT READ ?
YES
NO
CORE FILLED ?
NO
YES
COPY LINES UP OR DOWN AS REQUIRED
3

Figure C-44. MSSCON Program Flowchart (1 of 2)

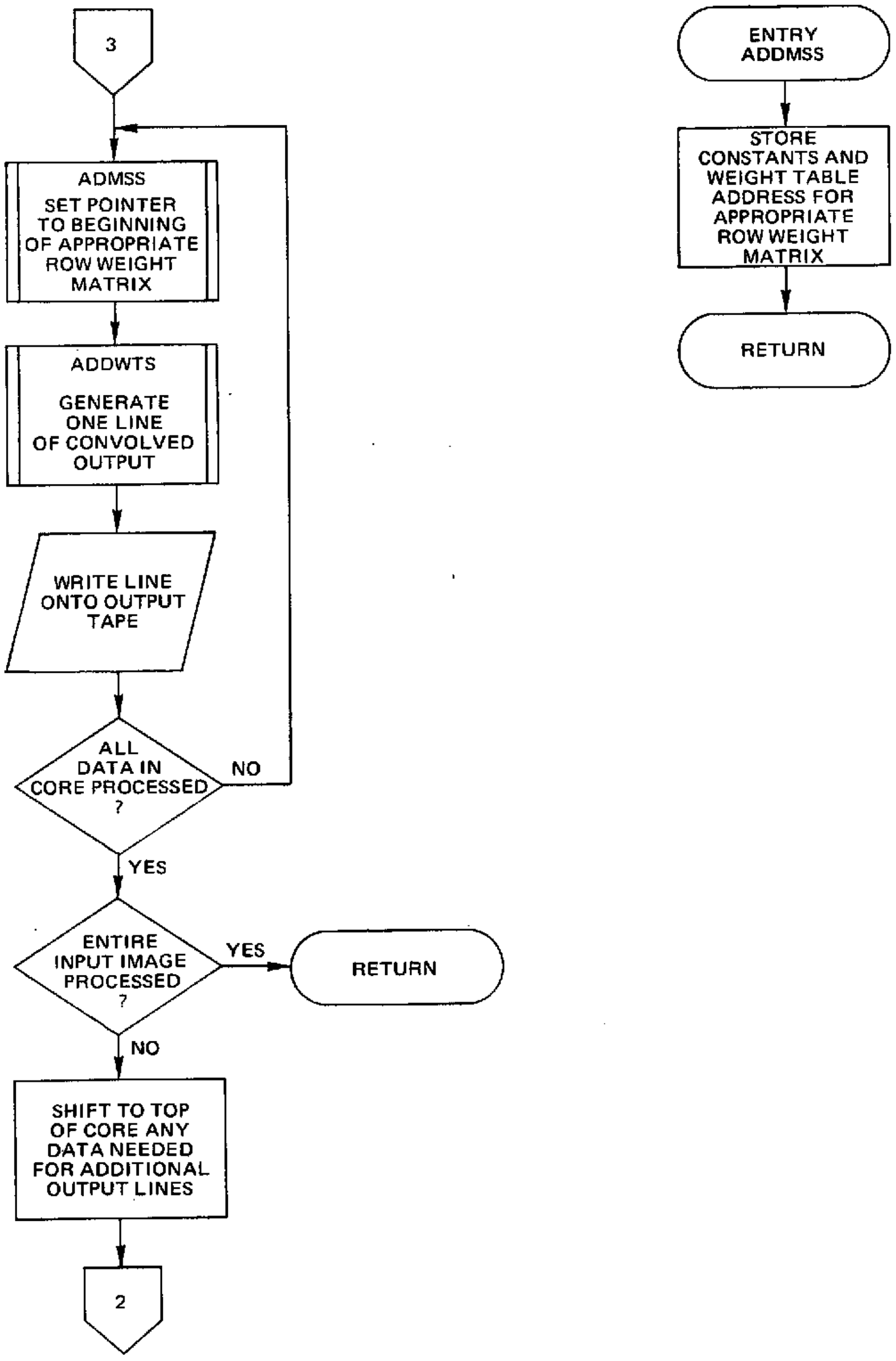

3
ADMSS
SET POINTER TO BEGINNING OF APPROPRIATE ROW WEIGHT MATRIX
ADDWTS
GENERATE ONE LINE OF CONVOLVED OUTPUT
WRITE LINE ONTO OUTPUT TAPE
ALL DATA IN CORE PROCESSED ?
NO
YES
ENTIRE INPUT IMAGE PROCESSED ?
YES
RETURN
NO
SHIFT TO TOP OF CORE ANY DATA NEEDED FOR ADDITIONAL OUTPUT LINES
2
ENTRY ADDMSS
STORE CONSTANTS AND WEIGHT TABLE ADDRESS FOR APPROPRIATE ROW WEIGHT MATRIX
RETURN

Figure C-44. MSSCON Program Flowchart (2 of 2)

APPENDIX D - FLOATING-POINT DATA REPRESENTATION

D.1 FLOATING-POINT ARITHMETIC

The complex quantities on which the fast Fourier transform (FFT) operates are processed using standard FORTRAN double-word floating-point arithmetic; this ensures straightforward adaptation of the programs to other computers. Substantial savings in computation time and storage space, without significant loss of accuracy, might have been achieved by using a nonstandard, single-word, floating-point representation rather than the standard CDC 3200 double-word software floating-point routines; however, extensive use of assembly-language coding would have been required. Therefore, this alternative was rejected.

D.2 MAXIMUM ARRAY SIZE

The FFT program uses an algorithm that requires array dimensions to be powers of 2, to simplify program logic. Because each complex quantity requires a total of four 24-bit computer words to represent both real and imaginary parts, approximately 6000 words available for storing arrays in core will accommodate a maximum of 1024 complex numbers, occupying 4096 computer words. Due to symmetries resulting from the absence of imaginary components in the original image data, these 1024 values normally will be packed into a 512-value line before performing the FFT or its inverse. The 32,300 cells (64 words/cell) available on disk storage are not quite sufficient to hold 1024 lines of 512 complex values each; therefore, the effective capacity of the disk is 512 lines.

For these reasons, an array of 512 lines of 512 values each is the largest that can be processed by FFT without requiring the addition of complicated and time-consuming provisions for frequent data transfer between disk and scratch tape. Because this size of array already requires more than an hour to transform on the CDC 3200, it was decided that there was little point in trying to provide for a larger array size.

Because image data is packed two pixels per complex word, an image with a maximum of 512 lines and 1024 pixels per line can be transformed.

D.3 THREE-STEP PROCESSING

Since the maximum-sized complex array can be accommodated on the disk, the FFT and smoothing routines store all intermediate results on the disk. To give the various Fourier transform and smoothing tasks maximum flexibility in input and output formats, any task involving complex data is subdivided into three steps:

1. Transfer input data from tape to disk, performing required packing and conversion during transmission so it is stored in standard format.
2. Carry out the FFT or smoothing and leave results on disk.
3. Transfer output data from disk to tape, performing any required unpacking, conversion, or scaling.

The FPCON routine transfers input data from tape to disk, carries out most format conversions of data already on disk, and transfers output data from disk to tape. However, a separate routine, CXPACK, packs and unpacks data on disk.

D.4 COMPLEX SYMMETRIC ARRAY PACKING AND UNPACKING: CXPACK

To halve FFT execution time, placement of alternate real values in the real and imaginary parts of the complex values converts a real input array to a pseudo-complex array. After Fourier transformation, the pseudo-complex array must be unpacked to obtain the actual Fourier components of the input array. The unpacked array of Fourier components is twice as large as the packed array. Because the input is real, however, only one-half the unpacked values need to be stored; the remainder can be obtained by using the symmetry properties of the Fourier components of a real array. The following paragraphs describe the mathematical details of this procedure.

Alternate elements of a real M by N array g_{mn} are packed into an M by N/2 complex array f_{mn} according to the relation

$$f_{mn} = g_{m,2n} + i\, g_{m,2n+1}, \quad m = 0, 1, \ldots, M-1; \; n = 0, 1, \ldots, \frac{N}{2} - 1$$

After carrying out the fast Fourier transform on f_{mn}, the Fourier coefficients $b_{k\ell}$ associated with g_{mn} can be computed from the Fourier coefficients $a_{k\ell}$ associated with f_{mn} by using the equation

$$b_{k\ell} = b^*_{M-k,N-\ell} = \frac{1}{2\sqrt{2}} \Big\{ (a_{k\ell} + a^*_{M-k, \frac{N}{2} - \ell})$$

$$- i\, e^{-2\pi i\ell/N} (a_{k\ell} - a^*_{M-k, \frac{N}{2} - \ell}) \Big\}$$

$$k = 0, 1, \ldots, M-1; \; \ell = 0, 1, \ldots, \frac{N}{2} - 1$$

Due to the symmetry relation between $b_{k\ell}$ and $b^*_{M-k,N-\ell}$, only the first N/2 + 1 columns of $b_{k\ell}$ need to be stored; the remaining N/2 - 1 can be computed using the symmetry equation. The packed array $a_{k\ell}$, however, requires only N/2 columns of storage. It would be very inconvenient to have to use a slightly larger disk array to accommodate the $b_{k\ell}$. However, the need for the extra column ($\ell = N/2$) can be eliminated as follows:

For $\ell = 0$, the symmetry principle simplifies to

$$b_{k,0} = b^*_{M-k,N} = b^*_{M-k,0}$$

because $b_{k\ell}$ is periodic in ℓ with period N; for $\ell = N/2$, this symmetry gives

$$b_{k,N/2} = b^*_{M-k,N/2}$$

Hence, by using the first column of the array to hold values of $b_{k,0}$ for $k < M/2$ and $b_{k,N/2}$ for $k \geq M/2$, all values of $b_{k\ell}$ may be reconstructed from an $M \times N/2$ array except for $b_{0,N/2}$ and $b_{m/2,0}$.

For these values, however, the symmetry principle gives

$$b_{0,N/2} = b^*_{M,N/2} = b^*_{0,N/2}$$

and,

$$b_{M/2,0} = b^*_{M/2,N} = b^*_{M/2,0}$$

where the periodicity of $b_{k\ell}$ has again been utilized. Hence, $b_{0,N/2}$ and $b_{M/2,0}$ are both real. A similar computation shows that $b_{0,0}$ and $b_{M/2,N/2}$ are also real. The four real values can, therefore, be stored into two complex words. The storage arrangement that has been employed is to store $b_{0,0}$ and $b_{0,N/2}$ into the real and imaginary parts, respectively, of the first complex value of the first line in the array; and $b_{M/2,0}$ and $b_{M/2,N/2}$, respectively, into the real and imaginary parts of the first complex value of line $M/2 + 1$ of the array. (This line corresponds to $k - M/2$, since $k = 0$ for the first line.)

When an inverse Fourier transformation is desired, the half array of b_k values described above must be packed back into the $a_{k\ell}$. This is achieved using the formula:

$$a_{k\ell} = \frac{1}{\sqrt{2}} \left\{ (1 + i\, e^{2\pi i\ell/N})\, b_{k\ell} + (1 - i\, e^{2\pi i\ell/N})\, b^*_{M-k', \frac{N}{2}-\ell} \right.$$

$$k = 0, 1 \ldots, M-1; \; \ell = 0, 1, \ldots, \frac{N}{2} - 1$$

After the inverse Fourier transformation has been carried out, the values of g_{mn} can be recovered simply by using

$$\left.\begin{aligned} g_{m,2n} &= \text{Re}\ (f_{mn}) \\ g_{m,2n+1} &= \text{Im}\ (f_{mn}) \end{aligned}\right\} \quad m = 0, 1, \ldots, M-1;\ n = 0, 1, \ldots, \frac{N}{2} - 1$$

where Re and Im denote real and imaginary parts, respectively.