N O T I C E

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

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

JSC-14268
80-10168
NASA CR-
160597

"AS-BUILT" SPECIFICATION FOR CCIT6A PROCESSOR PROGRAM

Job Order 73-783

(TIRF 78-0022)

(E80-10168) AS-BUILT SPECIFICATION FOR CCIT6A PROCESSOR PROGRAM (Lockheed Electronics Co.) 68 p HC A04/MF A01 CSCL 05B G3/43 N80-28769 Unclas 00168

Prepared By

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

Contract NAS 9-15200

For

EARTH OBSERVATIONS DIVISION

SPACE AND LIFE SCIENCES DIRECTORATE

NOV 1980
RECEIVED
NASA STI FACILITY
ACCESS DEPT.

National Aeronautics and Space Administration

LYNDON B. JOHNSON SPACE CENTER

Houston, Texas

August 1978

LEC-12303

JSC-14368

"AS-BUILT" SPECIFICATION FOR CCIT6A PROCESSOR PROGRAM

Job Order 73-783

(TIRF 78-0022)

PREPARED BY

W. P. White

APPROVED BY

Elmer M. Lore for

B. L. Carroll, Manager
EO Development and
Evaluation Department

Philip L. Krumm

P. L. Krumm, Manager
Data Systems Department

Prepared By

Lockheed Electronics Company, Inc.

For

Earth Observations Division

Space and Life Sciences Directorate

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

August 1978

LEC-12303

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
JSC-14368		

4. Title and Subtitle	5. Report Date
"As-Built" Specification for CCIT6A Processor Program	August 1978
	6. Performing Organization Code
7. Author(s) W. P. White Lockheed Electronics Company, Inc.	8. Performing Organization Report No. LEC-12303
9. Performing Organization Name and Address Lockheed Electronics Company, Inc. 1830 NASA Road 1 Houston, Texas 77058	10. Work Unit No. 11. Contract or Grant No. NAS 9-15200
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Lyndon B. Johnson Space Center Houston, Texas 77058 Technical Monitor: J. D. Erickson	13. Type of Report and Period Covered Technical Report 14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

The program CCIT6A is a utility module of the Accuracy Assessment Software System of the Large Area Crop Inventory Experiment. This program accesses data originating on the Classification and Mensuration Subsystem/Crop Assessment Subsystem interface tapes (CCIT's) of the Earth Resources Interactive Processing System (Version 6A). The data items needed for subsequent Accuracy Assessment processing are written into three disk files. The data extracted consist of a summary of the classification stratified areal estimate, cluster-dot match, and analyst-labeled dots (types 1 and 2).

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Accuracy Assessment CCIT	

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price*
Unclassified	Unclassified	68	

*For sale by the National Technical Information Service, Springfield, Virginia 22161

JSC Form 1424 (Rev Nov 75)

NASA — JSC

 PRECEDING PAGE BLANK NOT FILMED

CONTENTS

Section		Page
	ACRONYMS	ix
1.	SCOPE	1-1
2.	APPLICABLE DOCUMENTS	2-1
3.	SYSTEM DESCRIPTION	3-1
	3.1 HARDWARE DESCRIPTION	3-1
	3.2 MODULE DESCRIPTION	3-1
	3.3 SOFTWARE DESCRIPTION	3-5
	3.3.1 MODULE CCIT6A	3-6
	3.3.2 SUBROUTINE INPUT	3-12
	3.3.3 SUBROUTINE READH	3-15
	3.3.4 SUBROUTINE READRC	3-19
	3.3.5 SUBROUTINE HEADER	3-22
	3.3.6 SUBROUTINE BIASC	3-25
	3.3.7 SUBROUTINE CLUST	3-28
	3.3.8 SUBROUTINE RITEON	3-32
	3.3.9 SUBROUTINE TURNON	3-35
	3.3.10 SUBROUTINE DOTS	3-38
	3.3.11 SUBROUTINE STCODE	3-44
	3.3.12 SUBROUTINE PRNTIT	3-47
4.	OPERATIONS	4-1
	4.1 OPERATORS GUIDE	4-1
	4.1.1 HARDWARE CONFIGURATION	4-1
	4.1.2 PROGRAM EXECUTION	4-1

Section Page

4.2 USERS GUIDE. 4-2

4.3 MAINTENANCE DOCUMENTATION. 4-2

APPENDIX — FORMATS FOR .CLO FILE. A-1

TABLES

Table		Page
1	TASK-BUILDER COMMAND FILE FOR CCIT6A PROCESSOR PROGRAM.	3-5
2	BATCH RUN DECK SETUP.	4-3
A-1	FORMAT OF FIRST RECORD OF .CLO FILE	A-1
A-2	FORMAT OF CLUSTER-DOT MATCH IN .CLO FILE.	A-2

FIGURES

Figure		Page
1	Data flow of the CCIT6A processor program	3-2
2	Functional flow of the CCIT6A processor program	3-3
3	Flow diagram for the CCIT6A program	3-8
4	Listing for the CCIT6A program	3-10
5	Flow diagram for subroutine INPUT	3-13
6	Listing for subroutine INPUT	3-14
7	Flow diagram for subroutine READH	3-17
8	Listing for subroutine READH	3-18
9	Flow diagram for subroutine READRC	3-20
10	Listing for subroutine READRC	3-21
11	Flow diagram for subroutine HEADER	3-23
12	Listing for subroutine HEADER	3-24
13	Flow diagram for subroutine BIASC	3-26
14	Listing for subroutine BIASC	3-27
15	Flow diagram for subroutine CLUST	3-30
16	Listing for subroutine CLUST	3-31
17	Flow diagram for subroutine RITEON	3-33
18	Listing for subroutine RITEON	3-34
19	Flow diagram for subroutine TURNON	3-36
20	Listing for subroutine TURNON	3-37
21	Flow diagram for subroutine DOTS	3-40
22	Listing for subroutine DOTS	3-42
23	Flow diagram for subroutine STCODE	3-45

Figure		Page
24	Listing for subroutine STCODE	3-46
25	Flow diagram for subroutine PRNTIT.	3-48
26	Listing for subroutine PRNTIT	3-50

ACRONYMS

AA	Accuracy Assessment
CAMS	Classification and Mensuration Subsystem
CAS	Crop Assessment Subsystem
CCIT	CAMS/CAS interface tape
DO	Designated other
DPR	Data processing request
DTL	Data Techniques Laboratory
DTRM	Data terminal
DU	Designated unidentifiable
EOD	Earth Observations Division
ERIPS	Earth Resources Interactive Processing System
LACIE	Large Area Crop Inventory Experiment
Pixel	Picture element
SAE	Stratified areal estimate
TIRF	Transmittal Information Request Form
UIC	User identification code

1. SCOPE

This document specifies the detailed design for a software module to manipulate and extract data from Accuracy Assessment (AA) data base files previously derived from Large Area Crop Inventory Experiment (LACIE), version 6A, Classification and Mensuration Subsystem/Crop Assessment Subsystem (CAMS/CAS) interface tapes (CCIT's). It is called the CCIT6A module. The data extracted are output into three new data base files for direct input to AA analytical programs.

2. APPLICABLE DOCUMENTS

The following documents, of the exact issue shown, form parts of the specification to the extent specified herein.

a. "As-Built" Design Specification for PDP 11/45 Accuracy Assessment System Using Disk Data File. JSC-13893 (LEC-11881), February 1978 (and references therein).

b. Implementation of CCIT6A Processor Program. Transmittal Information Request Form (TIRF) 78-0022, May 11, 1978.

c. CAM/CAS Interface Tape Interface Control Document. LACIE-C00708, revision A (JSC-09866), July 1976.

d. Classification and Mensuration Subsystem (CAMS) Requirements. LACIE-C00200, volume II, revision D (JSC-11330), August 1977.

3. SYSTEM DESCRIPTION

The CCIT6A processor module accomplishes the data manipulations shown in figure 1. Basically, the CCIT data for a particular segment number, SSSS, and classification date, YYDDD, contained in file SSSSYYDDD.CCO are processed to obtain three output files required as input to existing or planned AA programs. The SSSSYYDDD.CLO file contains data needed for future programs. The SSSSYYDDD.AI1 and SSSSYYDDD.AI2 files are required for input to existing modules SPATL and MLTCRP.

3.1 HARDWARE DESCRIPTION

The PDP 11/45, with the following peripherals, is required.

a. Card reader

b. Line printer

c. Two disk units

3.2 MODULE DESCRIPTION

The CCIT6A module is implemented on the PDP 11/45 for background processing of CCIT data files into three output data files: an unformatted file of character data and two formatted files of analyst-labeled dots. See the functional flow diagram (fig. 2).

The LACIE CCIT is a universal nonimaging tape containing extensive statistical and ancillary data for a series of Earth Resources Interactive Processing System (ERIPS) runs. Using the AA CCIT program, all data for a relevant segment are transferred to a Files-11 disk file named SSSSYYDDD.CCO,•••, where SSSS is the segment number, YY is the year, and DDD is the day of the year. This .CCO file contains three 80-byte header records and a large number (>20) of 720-byte data records.

The first step of the process is to read the name of the input .CCO file and open this file for reading. Then the three CCIT header records are read

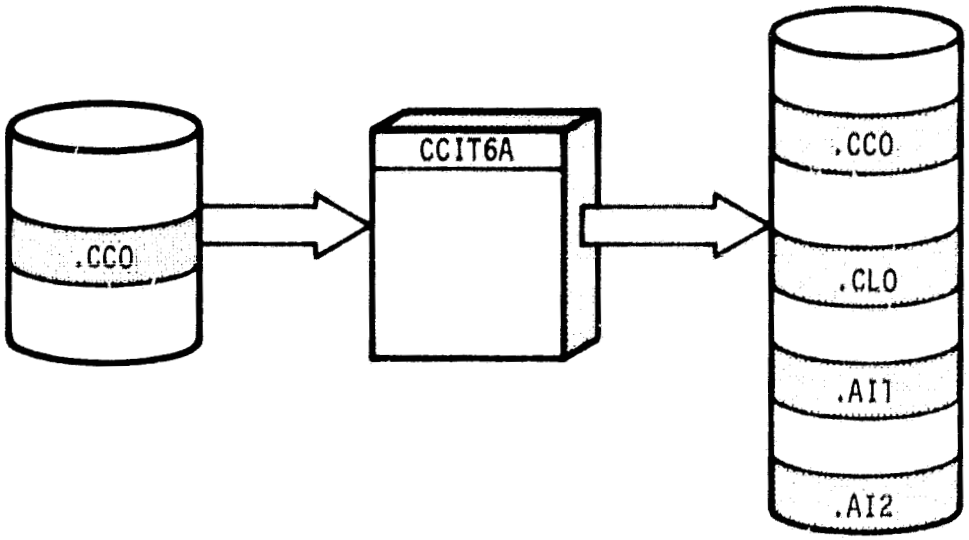

Figure 1.— Data flow of the CCIT6A processor program.

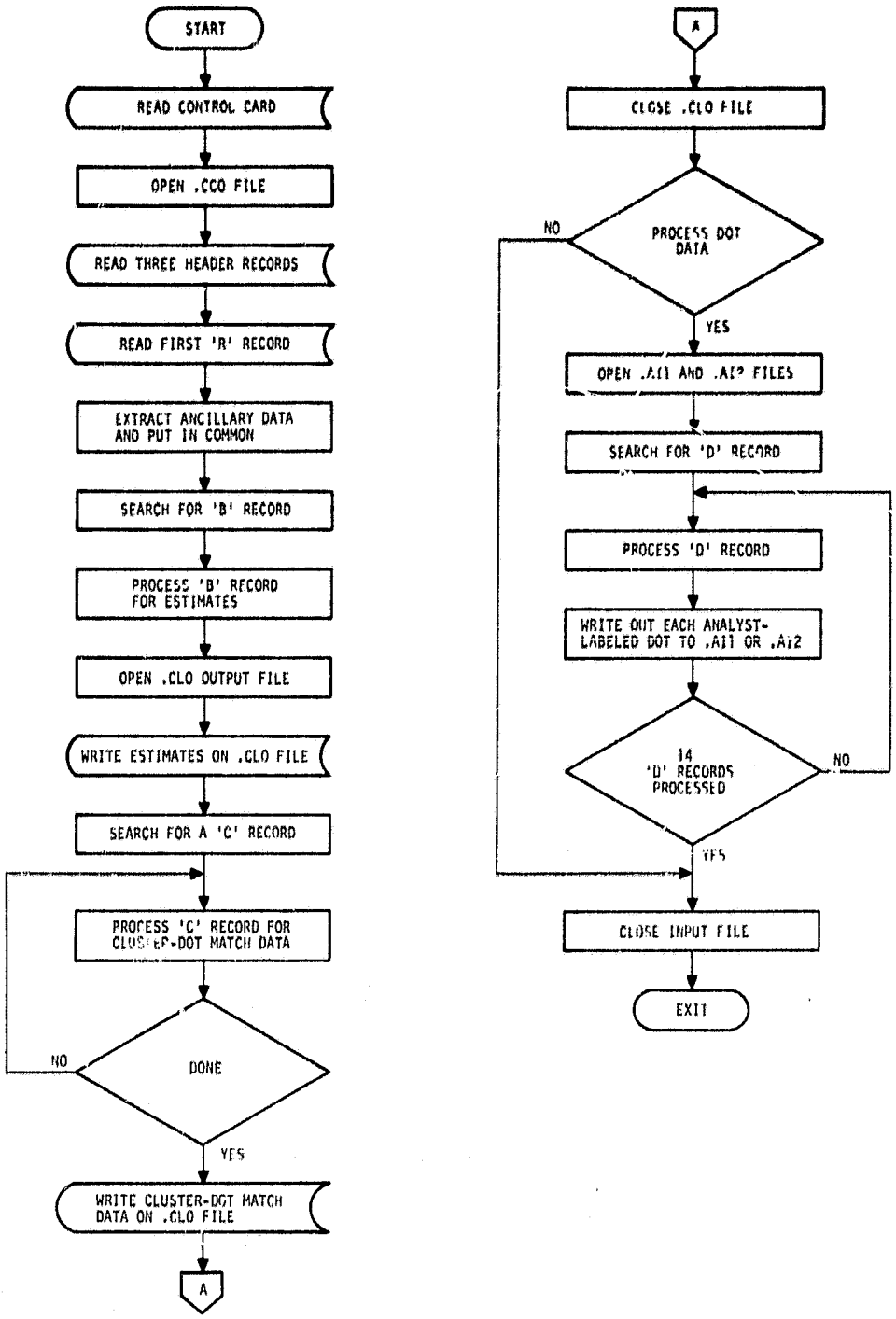

Figure 2.— Functional flow of the CCIT6A processor program.

and ignored. The next record (720 bytes) is read, checked to verify that it is a recognition (R) record, and processed to extract the data processing request (DPR) number and acquisition dates used in the classification.

Next a 'B' record is searched for and processed. The program extracts 44 bytes of data on picture element (pixel) populations, the ERIPS estimate, the stratified areal estimate (SAE), and the variance for each class. An output file SSSSYYDDD.CLO (where SSSS is the segment number and YYDDD is the classification date derived from the DPR number) is opened, and the estimate data are written as the first record.

In the next step a 'C' record is searched for and processed. The total number of clusters (q) and cluster-dot match data are extracted as q-groups of 12 characters. (Generally, there are more than 40 clusters, so some of these data appear in a second 'C' record.) When all the cluster-dot data are assembled into a buffer, the number of clusters is written out as the second record of file SSSSYYDDD.CLO and the match data as the third record of this file. The SSSSYYDDD.CLO file is then closed.

The final step is a test to determine if analyst-labeled dot output files are required; this is the default condition. If this condition exists, the output files SSSSYYDDD.AI1 and SSSSYYDDD.AI2 are opened, and a search is made for the first dot record. There are 14 dot records listing all 209 dots. The program examines each dot to determine if it has been labeled by the analyst. If so, it is written out (line, sample, and label) to the proper file, depending on the dot type (1 or 2). The first dot record in each output file also contains ancillary information on the segment (number and state code), classification date, acquisition dates, data terminal (DTRM) tape number, and type of label.

When all dots have been processed, the SSSSYYDDD.AI1 and SSSSYYDDD.AI2 output files and the input file are closed. The program then exits.

3.3 SOFTWARE DESCRIPTION

The CCIT6A processor program consists of 12 user-supplied routines: CCIT6A (main program), INPUT, READH, READRC, HEADER, BIASC, CLUST, RITEON, TURNON, DOTS, STCODE, and PRNTIT. The program makes use of a card-image-formatted file, CCIT6A.DAT, for program control and the line printer and user disk for output. The following sections provide a detailed description of each of the 12 routines. The recommended task-build command file (CCIT6A.CMD), used to create the load module (CCIT6A.TSK), is given in table 1.

TABLE 1.- TASK-BUILDER COMMAND FILE FOR CCIT6A PROCESSOR PROGRAM

```
CCIT6A,LP:/SH=RCCIT6,INPUT,READH,READRC,HEADER,BIASC,CLUST,
RITEON,TURNON,DOTS,STCODE,PRNTIT
/
FMTBUF=132
UNITS=6
ACTFIL=6
ASG=SY:1
ASG=SY:2
ASG=SY:3
ASG=SY:5
ASG=LP:6
PRI=50
//
```

For simplicity, the definition of arrays carried in COMMON blocks, the definition of COMMON blocks, and the description of COMMON blocks are not repeated for each routine. Instead, each of these elements is described in the routine of origin. Reference to the Interfaces subsections and to the compiler listings of each routine provides sufficient information to follow the data flow throughout the program.

3.3.1 MODULE CCIT6A

3.3.1.1 Linkage

The CCIT6A program is the main program. It calls user subroutines INPUT, READH, READRC, HEADER, BIASC, CLUST, DOTS, and PRNTIT. Subroutines CLUST, DOTS, and PRNTIT are called using multiple entry points.

3.3.1.2 Interface

Communication with the user routines is handled via COMMON blocks, except for a single integer parameter passed on call to READH which indicates the number of CCIT header records to be read.

3.3.1.2.1 COMMON Block BUF

BUF contains a 720-byte array, A, which is used to hold one CCIT logical record for processing.

3.3.1.2.2 COMMON Block FNAME

FNAME contains a 24-byte array, FILNAM, and an integer variable, SKIP. FILNAM contains the input file name read from CCIT6A.DAT. The value of SKIP determines whether the dot records are to be processed. If SKIP is nonzero, the dots are not processed.

3.3.1.3 Input

The CCIT6A program receives all input via subroutines INPUT and READRC.

3.3.1.4 Output

The CCIT6A program provides all output via subroutines PRNTIT, RITEON, BIASC, and DOTS.

3.3.1.5 Storage

The CCIT6A program requires 968 words of storage.

3.3.1.6 Description

The CCIT6A routine provides the control function for the program. Flow is controlled via tests on the first bytes (descriptive characters) of each logical record in the CCIT input file.

3.3.1.7 Flow Chart

The flow chart for CCIT6A is given in figure 3.

3.3.1.8 Listing

The listing for this subroutine is given in figure 4.

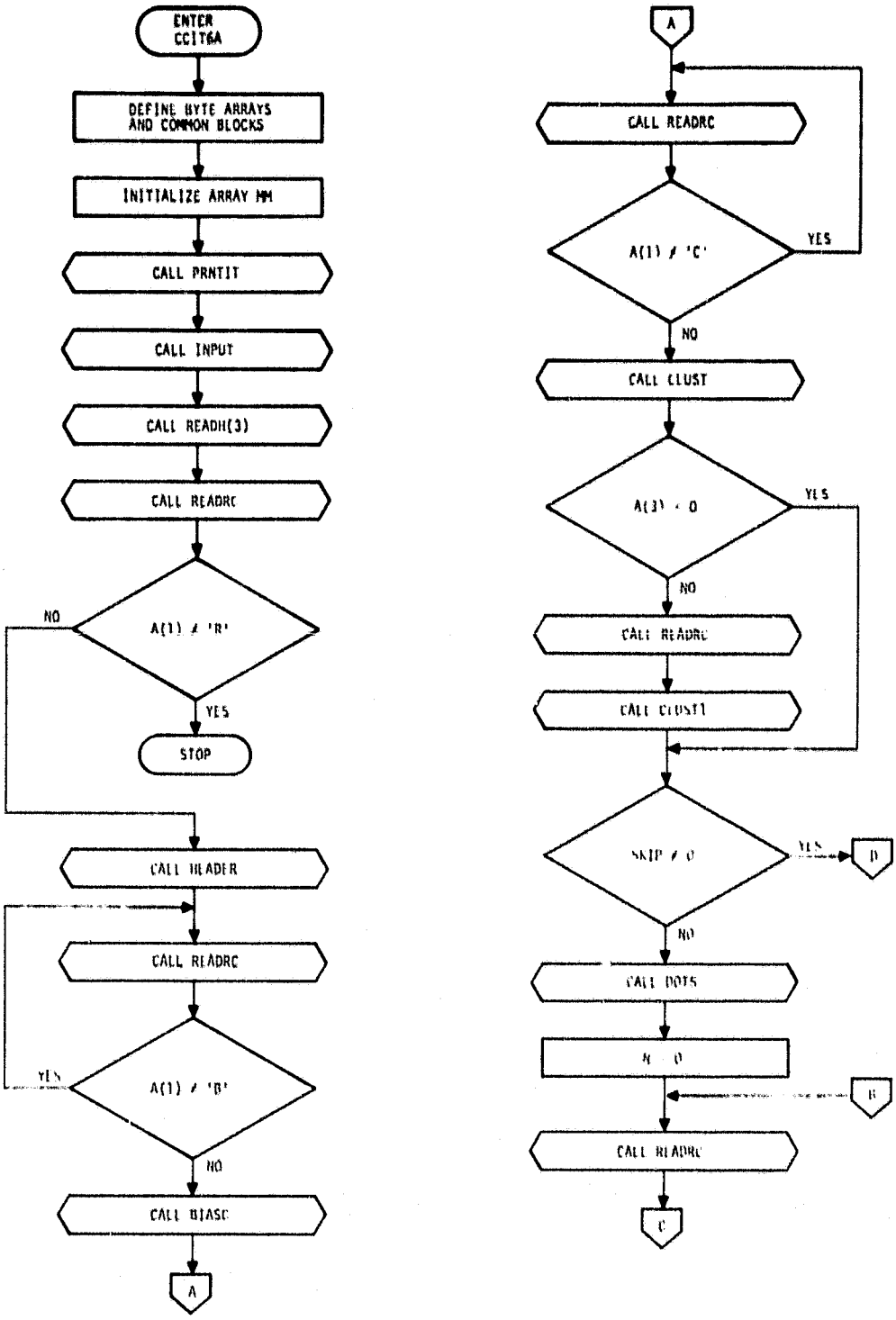

Figure 3.— Flow diagram for the CCIT6A program.

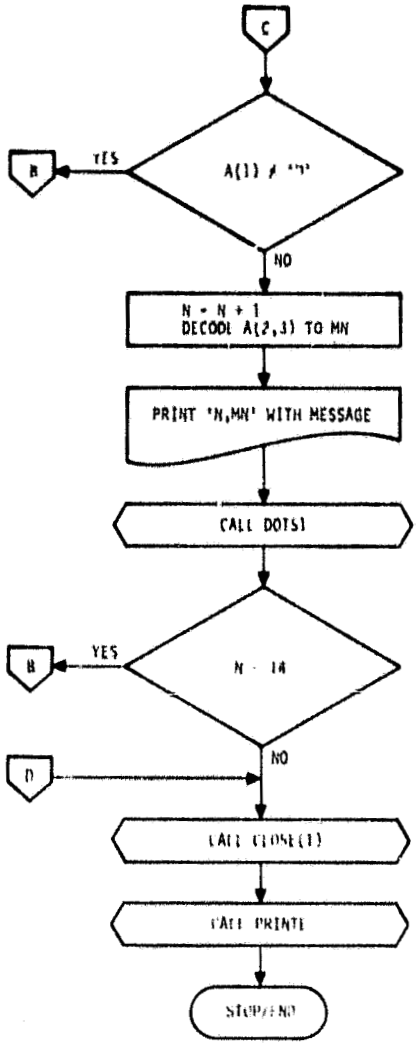

Figure 3.— Concluded.

```
FORTRAN IV-PLUS V02-51          13:14:57    04-MAY-78          PAGE 1
RCCIT6.FTN        /TR:BLOCKS/WR

0001              PROGRAM CCIT6A
0002              IMPLICIT INTEGER(A-Z)
0003              BYTE A(720),DPRNO(23),SNGNUM(4),ACD1(4),ACD2(4),ACD3(4),ACD4(4),
                1 CNAME(60,12),FILNAM(24),NM(25),MM(15)
          C*************************************************************************
          C       MAIN OF CCIT6A PROCESSOR - READ A CARD IMAGE FROM [110,6]CCIT6A.DAT
          C       WHICH GIVES THE FILE NAME OF THE LACIE6A AA CCIT FILE ON DISC
          C       IN THE FORM [131,2]SSSSYYDDD.CCD
          C       A(720) IS THE BUFFER WHICH CONTAINS ONE CCIT6A DATA RECORD
          C*************************************************************************
0004              COMMON/BUF/A
0005              COMMON/FNAME/FILNAM,SKIP
0006              DATA MM/1HR,1HB,1HC,1HD,1H0,1H1,1H2,8*1H /
0007              DATA SKIP/0/
0008              CALL PRNTIT
          C       READ INPUT FILNAME AND SKIP FACTOR
0009              CALL INPUT
          C       READ 3 HEADER RECORDS OF 80 BYTES EACH
0010              CALL READH(3)
          C       READ THE FIRST 'R' RECORD AND PROCESS FOR OPR NUMBER,ACQUISITION
          C       DATES ETC
0011              CALL READRC
0012              IF(A(1).NE.MM(1)) STOP
0013              CALL HEADER
          C       READ AND PROCESS BIAS CORRECTION SUMMARY RECORD
          C       AS FIRST RECORD OF FILE SSSSYYDDD.CLU
0014      210     CALL READRC
0015              IF(A(1).NE.MM(2)) GO TO 210
0016              CALL BIASC
          C       READ FIRST CLUSTER RECORD
          C       WHEN DONE SUBROUTINE 'RITEON' IS CALLED WHICH WRITES THE NUMBER
          C       OF CLUSTERS; AND THE CLUSTER-DOT MATCH DATA OUT AS TWO RECORDS
          C       OF FILE SSSSYYDDD.CLU
0017      220     CALL READRC
0018              IF(A(1).NE.MM(3)) GO TO 220
0019              CALL CLUST
0020              IF(A(3).EQ.MM(5)) GO TO 225
          C       READ SECOND CLUSTER RECORD IF PRESENT
0021              CALL READRC
0022              CALL CLUST1
          C       IF DOT DATA IS NOT NEEDED SKIP TO END OF PROGAM
0023              IF(SKIP.NE.0) GO TO 250
          C       INITIALIZE THE DOT OUTPUT FILES AND COUNTERS
0024      225     CALL DOTS
          C       READ DOTS RECORDS
0025      230     N = 0
          C       LOOK FOR 'D' (209 DOTS) RECORDS
0026      240     CALL READRC
0027              IF(A(1).NE.MM(4)) GO TO 230
0028              N = N + 1
0029              DECODE(2,100,A(2))MN
0030              PRINT 500,N,MN
          C       PROCESS A 'D' RECORD
0031              CALL DOTS1
0032              IF(N.LT.14) GO TO 240
          C       CLOSE INPUT FILE AND EXIT
```

ORIGINAL PAGE IS OF POOR QUALITY

Figure 4.— Listing for the CCIT6A program.

```
FORTRAN IV-PLUS V02-51          13:14:57    04-MAY-78          PAGE 2
RCCIT6.FTN        /TR:BLOCKS/WR

0033      250     CALL CLOSE(1)
0034              CALL PRINTE
0035              STOP
0036      100     FORMAT(I2)
0037      500     FORMAT(1H ,10X,'COUNT OF',I3,'  DOY RECORD NUMBER',I3)
0038              END
```

Figure 4.- Concluded.

3.3.2 SUBROUTINE INPUT

3.3.2.1 Linkage

Subroutine INPUT is called by the main program, CCIT6A. It calls subroutine PRNTIT via entry PRINTI.

3.3.2.2 Interface

Subroutine INPUT interfaces with CCIT6A via COMMON block FNAME, described in section 3.3.1.2.2.

3.3.2.3 Input

INPUT opens and reads file CCIT6A.DAT of user identification code (UIC) [110,6].

3.3.2.4 Output

INPUT calls entry PRINTI of subroutine PRNTIT to send a message to the line printer.

3.3.2.5 Storage

This subroutine requires 968 words of storage.

3.3.2.6 Description

INPUT opens file CCIT6A.DAT, reads a control card containing FILNAME and SKIP, closes the input file, calls the PRINTI entry of subroutine PRNTIT, and returns to CCIT6A.

3.3.2.7 Flow Chart

The flow diagram for subroutine INPUT is given in figure 5.

3.3.2.8 Listing

The subroutine listing is given in figure 6.

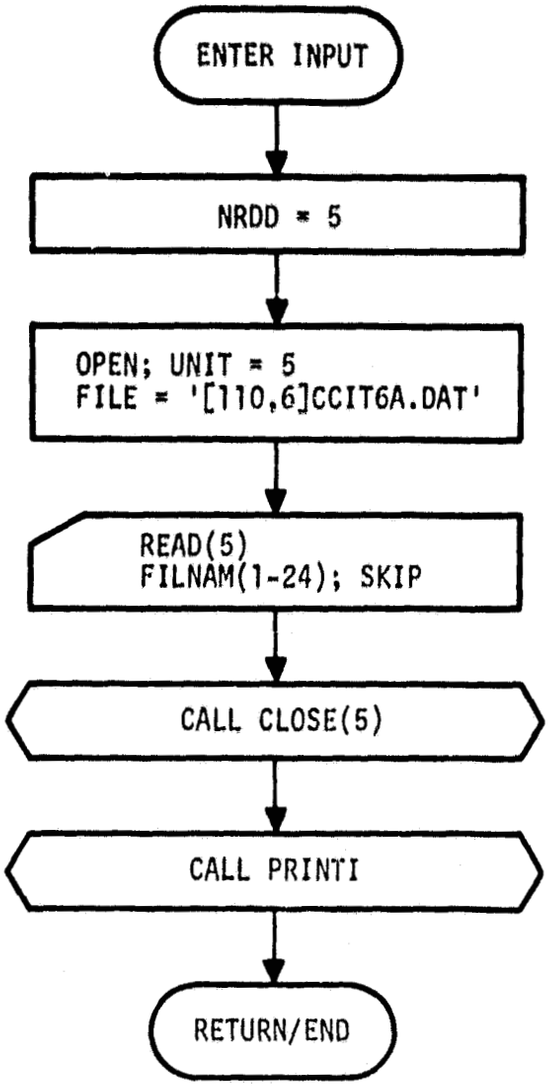

Figure 5.— Flow diagram for subroutine INPUT.

```
FORTRAN IV-PLUS V02-51          13:16:32    04-MAY-78          PAGE 1
INPUT.FTN        /TR:PLOCKS/WR

0001             SUBROUTINE INPUT
0002             IMPLICIT INTEGER(A-Z)
0003             BYTE FILNAM(24)
0004             COMMON/FNAME/FILNAM,SKIP
0005             NRDD=5
        C*****************************************************************
        C        THIS ROUTINE READS IN THE AA DATA BASE DISC FILE NAME FOR
        C        THE CCIT DESIRED
        C*****************************************************************
0006             OPEN(UNIT=NRDD,NAME=T[1,6]CCIT6A,DATA,TYPE='OLD',READONLY,
            1    ACCESS='SEQUENTIAL',FORM='FORMATTED',CARRIAGE CONTROL='NONE')
0007             READ(NRDD,100)(FILNAM(K),K=1,24),SKIP
0008             CALL CLOSE(NRDD)
0009             CALL PRINT1
0010             RETURN
0011     100     FORMAT(24A1,I2)
0012             END
```

Figure 6.— Listing for subroutine INPUT.

3.3.3 SUBROUTINE READH

3.3.3.1 Linkage

Subroutine READH calls subroutine TURNON.

3.3.3.2 Interface

READH interfaces with TURNON via an integer parameter (passed on call) giving the logical unit number to be opened and via COMMON block NAME containing the name of the file to be opened. COMMON block FNAME interfaces CCIT6A with READH. COMMON block BUF provides no true interfacing function for this routine.

3.3.3.2.1 COMMON Block NAME

NAME contains a 25-byte array, NM, which contains the complete name of a file to be opened by subroutine TURNON. NAME also interfaces several subroutines with subroutine PRNTIT. The last byte of array NM should contain the null (0) character.

3.3.3.3 Input

Header records from the CCIT input file are input.

3.3.3.4 Output

The only output is a read error message to the line printer.

3.3.3.5 Storage

READH requires 504 words of storage.

3.3.3.6 Description

Subroutine READH spaces past the three 80-byte CCIT header records, and the CCIT file name is written into the NM array. Subroutine TURNON opens the file on unit 1, the three records are read, and READH returns to CCIT6A.

3.3.3.7 Flow Chart

The flow diagram for subroutine READH is given in figure 7.

3.3.3.8 Listing

The listing for this subroutine is given in figure 8.

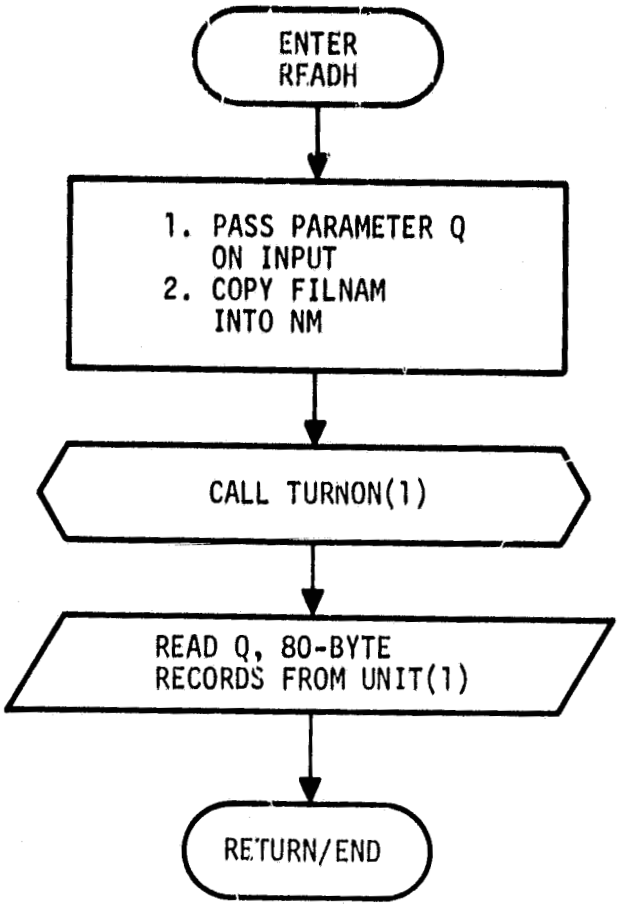

Figure 7.— Flow diagram for subroutine READH.

```
FORTRAN IV-PLUS V02-51          07144133    05-MAY-78          PAGE 1
READH,FT          /TR:0 FORS/W4

0001              SUBROUTINE READH(Q)
0002              IMPLICIT INTEGER(A-Z)
          C       READ UNIT READER RECORDS FROM DISC FOR FILE FILNAM
0003              BYTE NM(25),FILNAM(24),A(720)
0004              COMMON/BUF/A
0005              COMMON/NAME/NM
0006              COMMON/FNAME/FILNAM,SKIP
0007              DO 10 JJ=1,24
0008        10    NM(JJ) = FILNAM(JJ)
0009              NM(25) = 0
0010              CALL TURNON(1)
0011              DO 1 J=1,C
0012        1     READ(1,ERR=99,END=99)(A(K),K=1,80)
0013              RETURN
0014        99    PRINT 200,J
0015        200   FORMAT(1H ,' READ ERROR IN READH; ITERATION ',I2,//)
0016              STOP
0017              END
```

ORIGINAL PAGE IS OF POOR QUALITY

Figure 8.— Listing for subroutine READH.

3.3.4 SUBROUTINE READRC

3.3.4.1 Linkage

Subroutine READRC is called by CCIT6A.

3.3.4.2 Interface

Subroutine READRC interfaces with CCIT6A via COMMON block BUF.

3.3.4.3 Input

One data record read from the CCIT disk file is input.

3.3.4.4 Output

A read operation error message is output to the line printer.

3.3.4.5 Storage

This subroutine requires 441 words of storage.

3.3.4.6 Description

READRC reads one 720-byte logical data record from the CCIT input file into a buffer array, A.

3.3.4.7 Flow Chart

The flow diagram for subroutine READRC is given in figure 9.

3.3.4.8 Listing

The listing for this subroutine is given in figure 10.

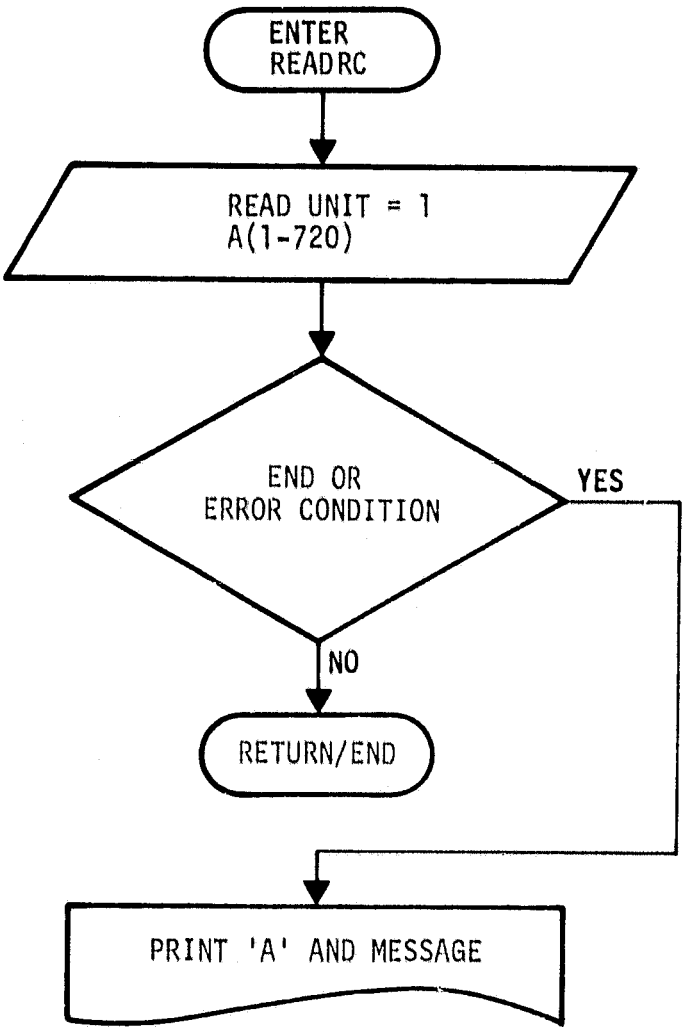

Figure 9.— Flow diagram for subroutine READRC.

```
FORTRAN IV-PLUS V02-51          07:44:49    05-MAY-78          PAGE 1
READRC.FTN        /TR:BLOCKS/XR

0001              SUBROUTINE READRC
0002              IMPLICIT INTEGER(A-Z)
        C         READ A CCIT DATA RECORD OF 720 BYTES INTO THE BUFFER A
0003              BYTE A(720)
0004              COMMON/BUF/A
0005              READ(1,ERR=99,END=99)(A(K),K=1,720)
0006              RETURN
0007      99      PRINT 100,A
0008      100     FORMAT(1H ,'ERROR IN READRC',//,' BUFFER CONTAINS',//,
             *1H ,'(12^A1)')
0009              END
```

Figure 10.— Listing for subroutine READRC.

ORIGINAL PAGE IS
OF POOR QUALITY

3.3.5 SUBROUTINE HEADER

3.3.5.1 Linkage

Subroutine HEADER is called by CCIT6A.

3.3.5.2 Interface

HEADER interfaces with CCIT6A via COMMON blocks BUF and DOTS and interfaces with PRNTIT (entry PRINTH) via COMMON block DOTS.

3.3.5.3 Input

There is no input to this subroutine.

3.3.5.4 Output

HEADER has no output.

3.3.5.5 Storage

This subroutine requires 445 words of storage.

3.3.5.6 Description

Subroutine HEADER selects byte data from the 'R' record of a CCIT (contained in buffer array A) and stores it into arrays in COMMON block DOTS. The data selected are the LACIE segment number [SMGNUM(1-4)], acquisition dates used for the ERIPS run [ACD1,ACD2,ACD3,ACD4], and ERIPS DPR number [DPRNO(1-23)]. The DPR number is printed in a message via a call to entry PRINTH of PRNTIT.

3.3.5.7 Flow Chart

The flow diagram for subroutine HEADER is given in figure 11.

3.3.5.8 Listing

The listing for this subroutine is given in figure 12.

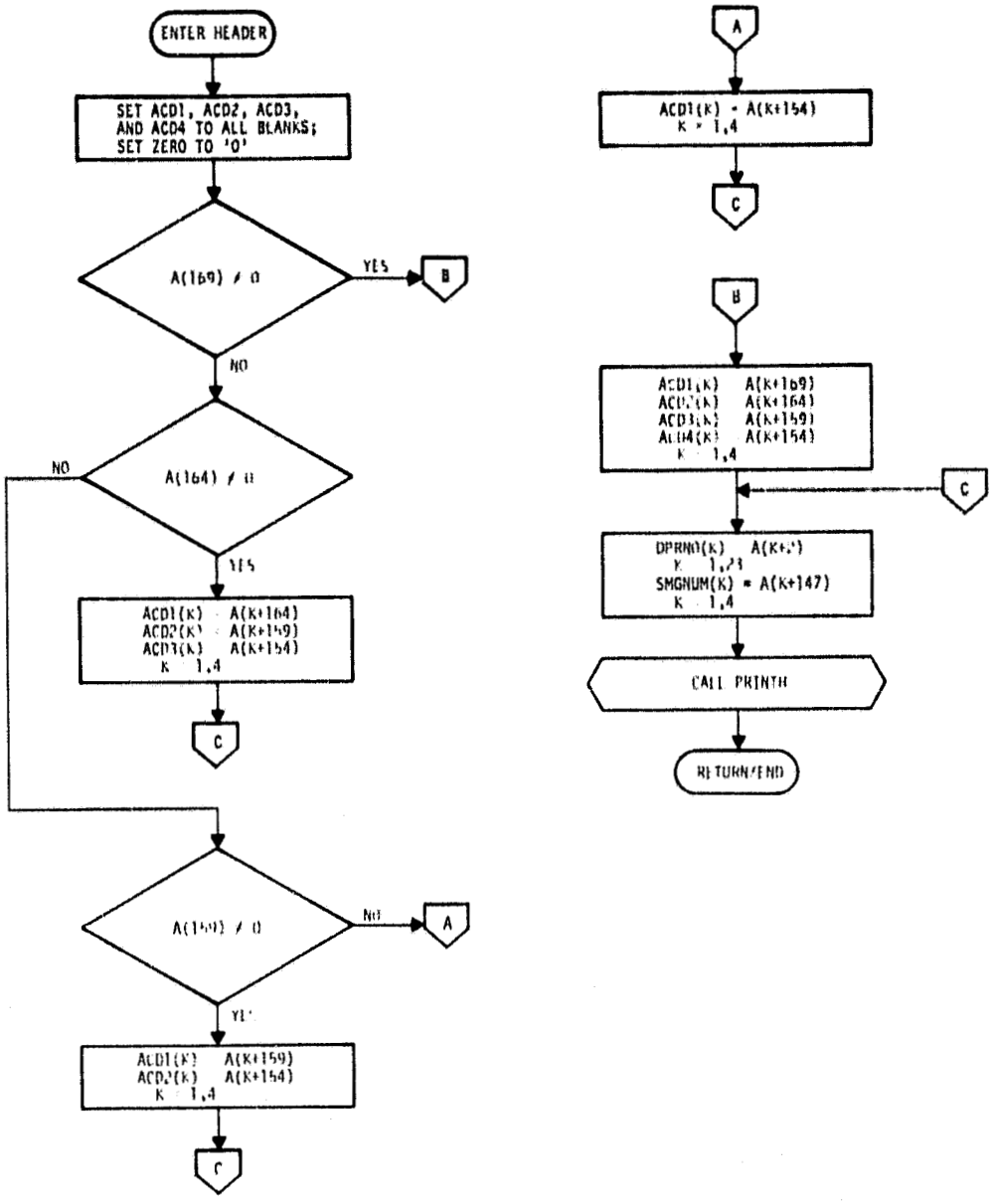

Figure 11.— Flow diagram for subroutine HEADER.

```
FORTRAN IV-PLUS V02-51          07:47:40     05-MAY-78          PAGE 1
HEADER.FTN        /TR:BLOCKS/WR

0001              SUBROUTINE HEADER
          C*****************************************************************
          C       THIS SUBROUTINE OBTAINS THE DPR NUMBER ,AND ACQUISITION
          C       DATES FROM A  1ST RECORD.
          C*****************************************************************
0002              IMPLICIT INTEGER(A-Z)
0003              BYTE A(720),ACD1(4),ACD2(4),ACD3(4),ACD4(4),
                1 DPR N(23),SHGN N(4)
0004              COMMON/BUF/A
0005              COMMON/DATS/DPR N,ACD1,ACD2,ACD3,ACD4,SHGNUM
0006              DO 1 K=1,4
0007        1     SHGN N(K)=A(K+147)
0008              DO 2 K=1,4
0009              ACD1(K) = A(154+K)
0010              ACD2(K) = A(159+K)
0011              ACD3(K) = A(164+K)
0012        2     ACD4(K) = A(170+K)
0013              DO 3 K=1,23
0014        3     DPR N(K) = A(K+2)
0015              CALL PRINTH
0016              RETURN
0017        100   FORMAT(I4)
0018              END
```

ORIGINAL PAGE IS OF POOR QUALITY

Figure 12.— Listing for subroutine HEADER.

3.3.6 SUBROUTINE BIASC

3.3.6.1 Linkage

BIASC is called by the CCIT6A program. It calls subroutine TURNON once.

3.3.6.2 Interface

BIASC interfaces with CCIT6A via COMMON block BUF and with TURNON via COMMON block NAME. Data are passed from subroutine HEADER to BIASC via COMMON block DOTS.

3.3.6.3 Input

There is no input to this subroutine.

3.3.6.4 Output

BIASC writes one 44-byte unformatted record onto unit 3. The data contained in this record are detailed in the appendix.

3.3.6.5 Storage

This subroutine requires 497 words of storage.

3.3.6.6 Description

BIASC codes the output file name as SSSSYYDDD.CLO, where SSSS is the segment number and YYDDD is the classification date. These data are obtained from COMMON block DOTS. Unit 3 is opened for output via a call to subroutine TURNON. Then a single 44-byte record of data from array A is written onto unit 3.

3.3.6.7 Flow Chart

The flow diagram for subroutine BIASC is given in figure 13.

3.3.6.8 Listing

The listing for this subroutine is given in figure 14.

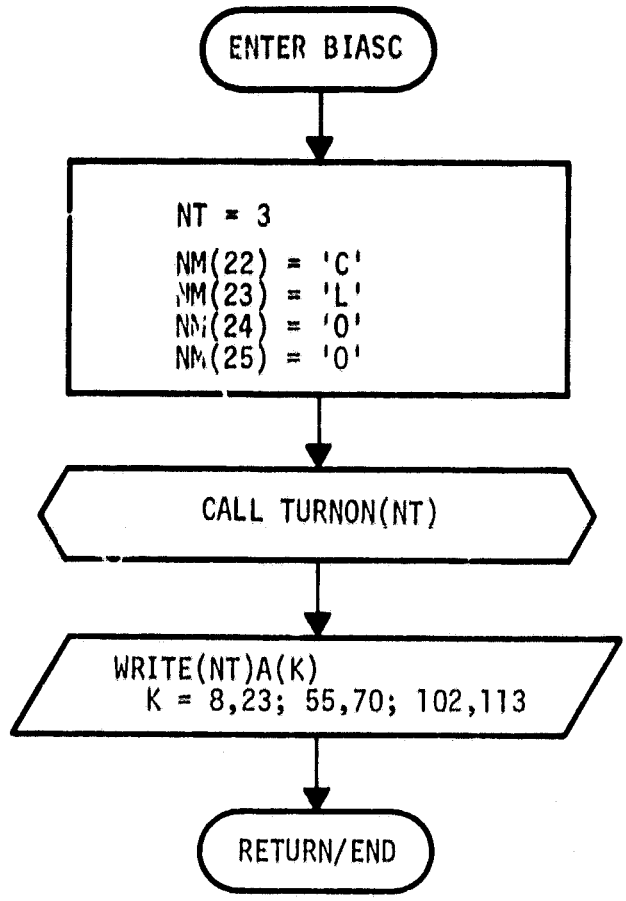

Figure 13.— Flow diagram for subroutine BIASC.

```
      ?
0001          SUBROUTINE BIASC
0002          IMPLICIT INTEGER(A-Z)
      C*************************************************************
      C        SES THE BIAS CORRECTION (I.E., STRATIFIED APEAL
      C        ESTIMATE DATA FROM A CCIT VET RECORD
      C        TO BE USED THE FIRST RECORD OF THE
      C        SESSYYDD.CLB OUTPUT FILE, LEAVES FILE OPEN FOR
      C        WRITING OF THE 2 WORD RECORDS NUMBER OF CLUSTERS,
      C        AND THE ALPHANUMERIC CLUSTER-DCT MATCH TABLE
      C        IN S SRV TIME OL ST
      C*************************************************************
0003          BYTE SW(25),CHRS(4),OPEN2(23),ACD1(4),ACD2(4),ACD3(4),
     1        ACD4(23),FMT(7)
0004          BYTE A(720)
0005          EQUIV (A,B,B,A)
0006          COMMON/NAME/ A
0007          EQUIV (A,ITS) PRN2,ACD1,ACD2,ACD3,ACD4,SWCHRM
0008          DATA CHR/1H0,1H1,1H /
0009          IT = 1
      C        GETS NAME FOR OUTPUT FILE IN FORM SSSSYYDD.CLB
0010          SW(12) = SWS SW(1)
0011          SW(13) = SWS SW(2)
0012          SW(14) = SWS SW(3)
0013          SW(15) = SWS SW(4)
0014          SW(16) = DRN A(8)
0015          SW(17) = DRN A(9)
0016          SW(18) = DRN A(9)
0017          SW(19) = DRN A(12)
0018          SW(20) = DRN A(13)
0019          SW(22) = CHR(1)
0020          SW(23) = CHR(2)
0021          SW(24) = CHR(3)
0022          SW(25) = 0
      C        OPEN OUTPUT FILE = UNFORMATTED
0023          CALL TURN2 ( IT)
      C
      C        WRITE FIRST RECORD OF .CLB FILE CONTAINING BIAS CORRECTION RESULTS
0024          WRITE(IT)(A(K),K=8,23),(A(K),K=55,70),(A(K),K=102,113)
0025          RETURN
0026          END
```

Figure 14.— Listing for subroutine BIASC.

ORIGINAL PAGE IS
OF POOR QUALITY

3.3.7 SUBROUTINE CLUST

3.3.7.1 Linkage

Subroutine CLUST is called once by CCIT6A. When there are more than 40 clusters, CCIT6A calls CLUST a second time via entry CLUST1. CLUST calls subroutine RITEON once and calls subroutine PRNTIT via entry PRINTC once for each 'C' record processed.

3.3.7.2 Interface

CLUST interfaces with CCIT6A via COMMON block BUF, with RITEON via COMMON block CLUSTR, and with PRNTIT via passing parameter RCNUM on call.

3.3.7.2.1 COMMON Block CLUSTR

CLUSTR contains a 60- by 12-byte array, CNAME, and an integer variable CNUM. CLUSTR provides an interface between subroutine CLUST and subroutine RITEON.

3.3.7.3 Input

There is no input to this subroutine.

3.3.7.4 Output

Subroutine CLUST has no output.

3.3.7.5 Storage

This subroutine requires 871 words of storage.

3.3.7.6 Description

CLUST processes the CCIT 'C' records to provide the total number of clusters and the identity of the analyst-labeled (type 1) dot used to name each.

CLUST decodes bytes 4 and 5 to obtain the total number of clusters (CNUM) and bytes 6 and 7 to obtain the number of clusters contained on the record

(RCNUM). Then for each cluster, the 12 bytes representing the cluster name (6 bytes) and dot name (6 bytes) used in labeling the cluster are copied into the array CNAME. PRNTIT is called via entry PRINTC to print a message containing the parameter RCNUM. If additional data are available on the next record, there is a return to the main program which calls CLUST via CLUST1. When CNUM sets of data are written into CNAME, CLUST calls subroutine RITEON for output.

3.3.7.7 Flow Chart

The flow diagram for subroutine CLUST is given in figure 15.

3.3.7.8 Listing

The listing for this subroutine is given in figure 16.

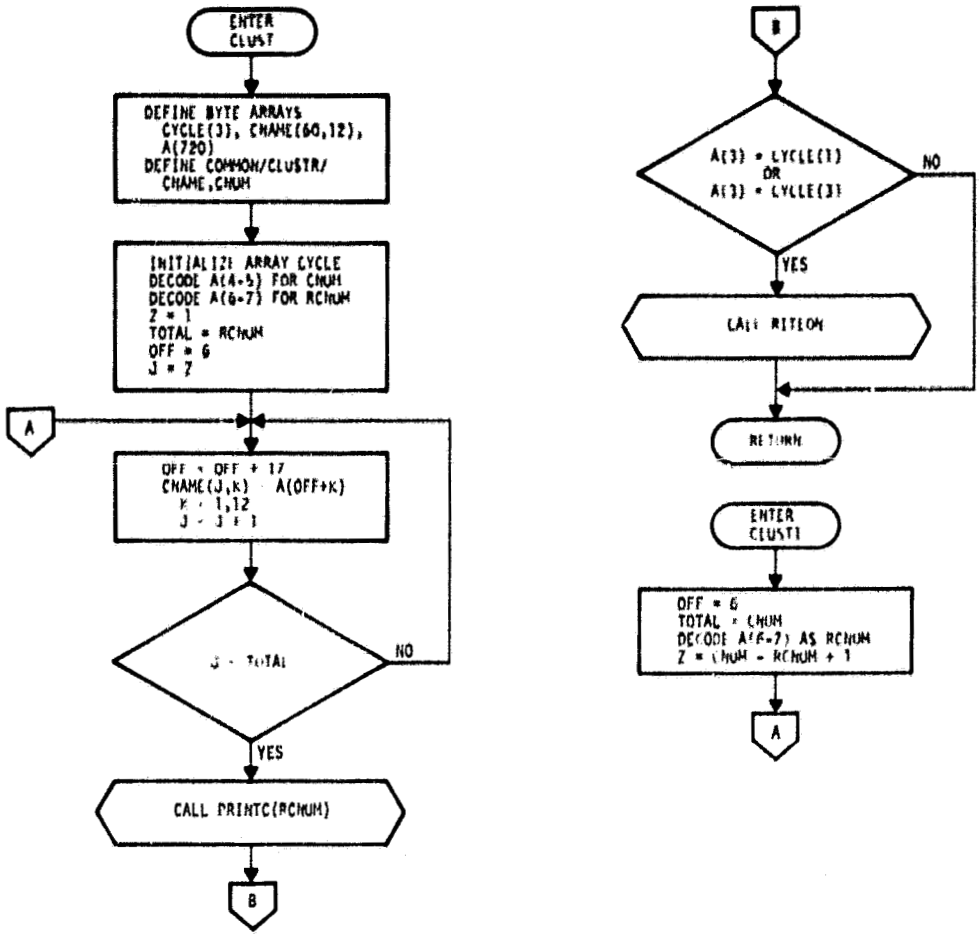

Figure 15.— Flow diagram for subroutine CLUST.

```
FORTRAN IV-PLUS V02-51          13:15:41     04-FEB-78          PAGE 1
CLUST.FTN          /TR:BLOCKS/-W

0001               SUBROUTINE CLUST
0002               IMPLICIT INTEGER(A-Z)
          C        READS CCIT 'C' RECORDS AND OUTPUTS CLUSTER LABEL AND CLUSTER-
          C        POT MATCH DATA TO A DISC FILE NAMED SSSSYYDDD.CLB.
          C        ENTER HERE FOR THE FIRST 'C' RECORD
          C        ENTER AT CLUST1 FOR A SECOND 'C' RECORD IF PRESENT
0003               BYTE A(720),CYCLE(3),CNAME(60,12)
          C*****************************************************************
          C        CNAME IS A STORAGE ARRAY TO PERMIT WRITING THE DATA IN ONE
          C        RECORD TO DISC IN A FILE NAMED SSSSYYDDD.CLB
          C        CNUM IS THE TOTAL NUMBER OF CLUSTERS
          C        OFF IS THE BYTE OFFSET WITHIN A 'C' RECORD
          C        RCNUM IS THE NO. OF CLUSTERS IN THE CURRENT RECORD
          C        IF CNUM > 40, THE FIRST 40 CLUSTERS ARE ON THE FIRST RECORD
          C        AND THE REMAINDER ARE ON THE SECOND RECORD
          C*****************************************************************
0004               COMMON/BUFF/A
0005               COMMON/CLUSTR/CNAME,CNUM
0006               DATA CYCLE/1H0,1H1,1H2/
          C
          C        DECODE TO FIND TOTAL NO. OF CLUSTERS,CNUM; AND NO. OF
          C        CLUSTERS ON THIS RECORD,RCNUM
0007               DECODE(2,100,A(4))CNUM
0008               DECODE(2,100,A(6))RCNUM
0009               Z = 1
0010               TOTAL = RCNUM
0011               OFF = 6
0012      5        DO 1 J = 2,TOTAL
0013               OFF = OFF + 17
0014               DO 1 K=1,12
0015      1        CNAME(J,K) = A(OFF+K)
0016               CALL PRTSTA(RCNUM)
          C        IF NONE GO TO DISC WRITE ROUTINE 'RITEON'
0017               IF(A(3).EQ.CYCLE(1).OR.A(3).EQ.CYCLE(3)) CALL RITEON
          C
0018               RETURN
          C        IF THERE ARE 2 'C' RECORDS, THE SECOND IS PROCESSED VIA
          C        THIS ENTRY.
0019               ENTRY CLUST1
0020               DECODE(2,100,A(6))RCNUM
0021               TOTAL = CNUM
0022               Z = CNUM - RCNUM + 1
0023               OFF = 6
0024               GO TO 5
0025      100      FORMAT(I2)
0026               END
```

Figure 16 - Listing for subroutine CLUST.

ORIGINAL PAGE IS OF POOR QUALITY

3.3.8 SUBROUTINE RITEON

3.3.8.1 Linkage

Subroutine RITEON is called once by subroutine CLUST.

3.3.8.2 Interface

Subroutine RITEON interfaces with CLUST via COMMON block CLUSTR (see section 3.3.7.2.1).

3.3.8.3 Input

There is no input to RITEON.

3.3.8.4 Output

RITEON writes two records onto a previously opened file (unit 3). This unit is opened in subroutine BIASC as an unformatted FORTRAN disk file.

3.3.8.5 Storage

This subroutine requires 427 words of storage.

3.3.8.6 Description

RITEON writes two records onto unit 3. The first record is a single integer, CNUM. The second record consists of the array CNAME as CNUM 12-byte elements. The output file is closed via a call to the system routine CLOSE.

3.3.8.7 Flow Chart

The flow diagram for subroutine RITEON is given in figure 17.

3.3.8.8 Listing

The listing for this subroutine is given in figure 18.

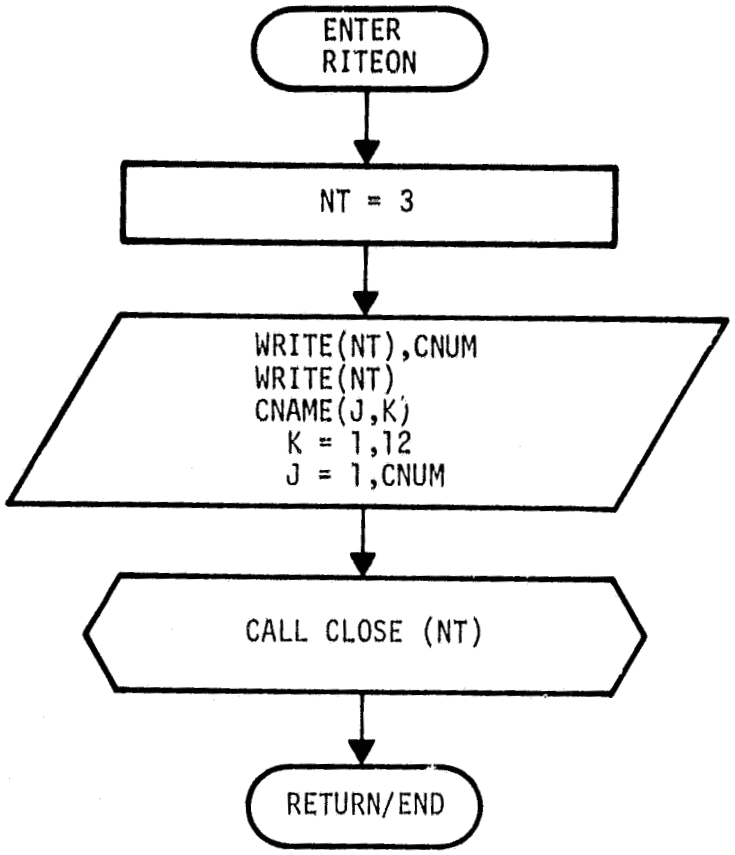

Figure 17.— Flow diagram for subroutine RITEON.

```
FORTRAN IV-PLUS V02-51          13:15:22    04-MAY-78          PAGE 1
RITEON.FTN      /TRIBLOCKS/WR

0001            SUBROUTINE RITEON
0002            IMPLICIT INTEGER(A-Z)
        C       WRITES AN UNFORMATTED FILE OF CLUSTER NAME AND CLUSTER # AT DOT MATCH
        C       DATA FROM DISC FOR AN PROCESSING...LACIEGA VERSION
        C       OUTPUT WAS CREATED IN SUBROUTINE BIASC
0003            BYTE CNAME(6,12)
0004            COMMON/CLUSTR/CNAME,CNUM
0005            NT = 3
        C       WRITE TWO RECORDS.  FIRST,THE NUMBER OF CLUSTERS. THEN, THE ALPHANUMERIC
        C       DATA...12 CHARACTERS PER CLUSTER
0006            WRITE(NT)CNUM
0007            WRITE(NT)((CNAME(J,K),K=1,12),J=1,CNUM)
        C       CLOSE OUTPUT FILE
0008            CALL CLOSE(NT)
0009            RETURN
0010            END
```

ORIGINAL PAGE IS
OF POOR QUALITY

Figure 18.— Listing for subroutine RITEON.

3.3.9 SUBROUTINE TURNON

3.3.9.1 Linkage

Subroutine TURNON is called by subroutines READH, BIASC, and DOTS.

3.3.9.2 Interface

TURNON interfaces with its calling routines via COMMON block NAME (see section 3.3.3.2.1) and a passed parameter, NT.

3.3.9.3 Input

There is no input to this subroutine.

3.3.9.4 Output

TURNON has no output.

3.3.9.5 Storage

This subroutine requires 162 words of storage.

3.3.9.6 Description

TURNON opens a file with the file name contained in byte array NM. If NT = 1, the input file is opened as UNIT = 1. If NT = 2-6, an unformatted file is opened as unit NT. If NT > 6, a formatted file is opened as unit (NT-6). Prior to opening the file, the routine prints a message containing the passed unit number parameter, NT, and the file name, NM.

3.3.9.7 Flow Chart

The flow diagram for subroutine TURNON is given in figure 19.

3.3.9.8 Listing

The listing for this subroutine is given in figure 20.

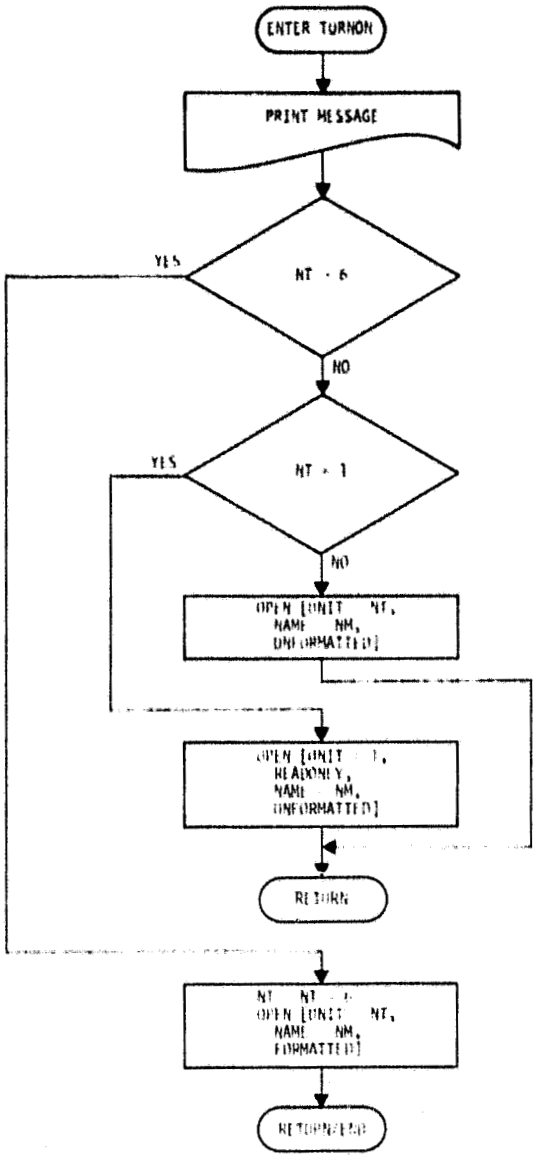

Figure 19.— Flow diagram for subroutine TURNON.

```
FORTRAN IV-PLUS V02-51          07:48:30    05-MAY-78          PAGE 1
TURNON.FTN      /TR:BLOCKS/WR

0001            SUBROUTINE TURNON(NT)
0002            IMPLICIT INTEGER(A-Z)
      C*****************************************************************
      C         ROUTINE OPENS DIRECT ACCESS FILES FOR PROGRAM CCIT6A
      C         IF UNIT NUMBER PASSED (NT) IS >6, OPEN A FORMATTED FILE
      C         ELSE OPEN AN UNFORMATTED FILE
      C         USE UNIT 1 FOR A FILE WHICH MUST BE 'OLD'
      C*****************************************************************
0003            BYTE NM(25)
0004            COMMON/NAME/NM
0005            PRINT 100,NT,(NM(K),K=1,24)
0006    100     FORMAT(1H0,'  OPENING UNIT ',I1,' ON DISC TO ACCESS FILE '
           *,24A1,//)
0007            IF(NT.GT.6) GO TO 2
0008            IF(NT.EQ.1) OPEN(UNIT=NT,NAME=NM,READONLY,
           *    TYPE='OLD',ACCESS='SEQUENTIAL',FORM='UNFORMATTED')
0009            IF(NT.EQ.1) RETURN
0010            OPEN(UNIT=NT,NAME=NM,TYPE='UNKNOWN',ACCESS='SEQUENTIAL',
           1    FORM='UNFORMATTED')
0011            RETURN
0012    2       NT = NT - 6
0013            OPEN(UNIT=NT,NAME=NM,TYPE='UNKNOWN',ACCESS='SEQUENTIAL',
           1    FORM='FORMATTED',CARRIAGE CONTROL='FORTRAN')
0014            RETURN
0015            END
```

Figure 20.— Listing for subroutine TURNON.

ORIGINAL PAGE IS
OF POOR QUALITY

3.3.10 SUBROUTINE DOTS

3.3.10.1 Linkage

Subroutine DOTS is called by CCIT6A once via the main entry and 14 times via entry DOTS1. DOTS calls subroutine TURNON twice and subroutines PRINTD and STCODE once.

3.3.10.2 Interface

Subroutine DOTS interfaces with TURNON via COMMON block NAME and with CCIT6A via COMMON blocks BUF and DOTS.

3.3.10.3 Input

There is no input to this subroutine.

3.3.10.4 Output

Subroutine DOTS writes formatted, card-image records onto two disk-based output files opened on the initial call to the routine.

3.3.10.5 Storage

This subroutine requires 976 words of storage.

3.3.10.6 Description

DOTS processes CCIT 'D' records into two formatted files of analyst-labeled dots. When called as DOTS, the routine initializes the unit parameters, NT and MT, and the dot counters, KOUNT1 and KOUNT2. Then the elements of the array NM are set to name the file to receive the type 1 analyst-labeled dot data, and TURNON is called to open this file. NM(24) is redefined (1 → 2) to provide the name of the type 2 dot output file, and TURNON is called to open this file. Subroutine STCODE is called to obtain the two-byte parameter ST, the alphabetic state code for the segment. Control then returns to CCIT6A.

When called as DOTS1, the routine processes one 720-byte 'D' record. For each analyst-labeled dot, one record is written. For type 1 dots, the data are written onto unit 2; for type 2 dots, the data are written onto unit 3. KOUNT1 is incremented for each type 1 dot, and KOUNT2 is incremented for each type 2 dot.

After processing all 209 dots (14 calls from CCIT6A), a blank record is written into each output file. Then both output files are closed, and a message listing KOUNT1 and KOUNT2 is printed via a call to PRNTIT subroutine entry PRINTD.

3.3.10.7 Flow Chart

The flow diagram for subroutine DOTS is given in figure 21.

3.3.10.8 Listing

The listing for this subroutine is given in figure 22.

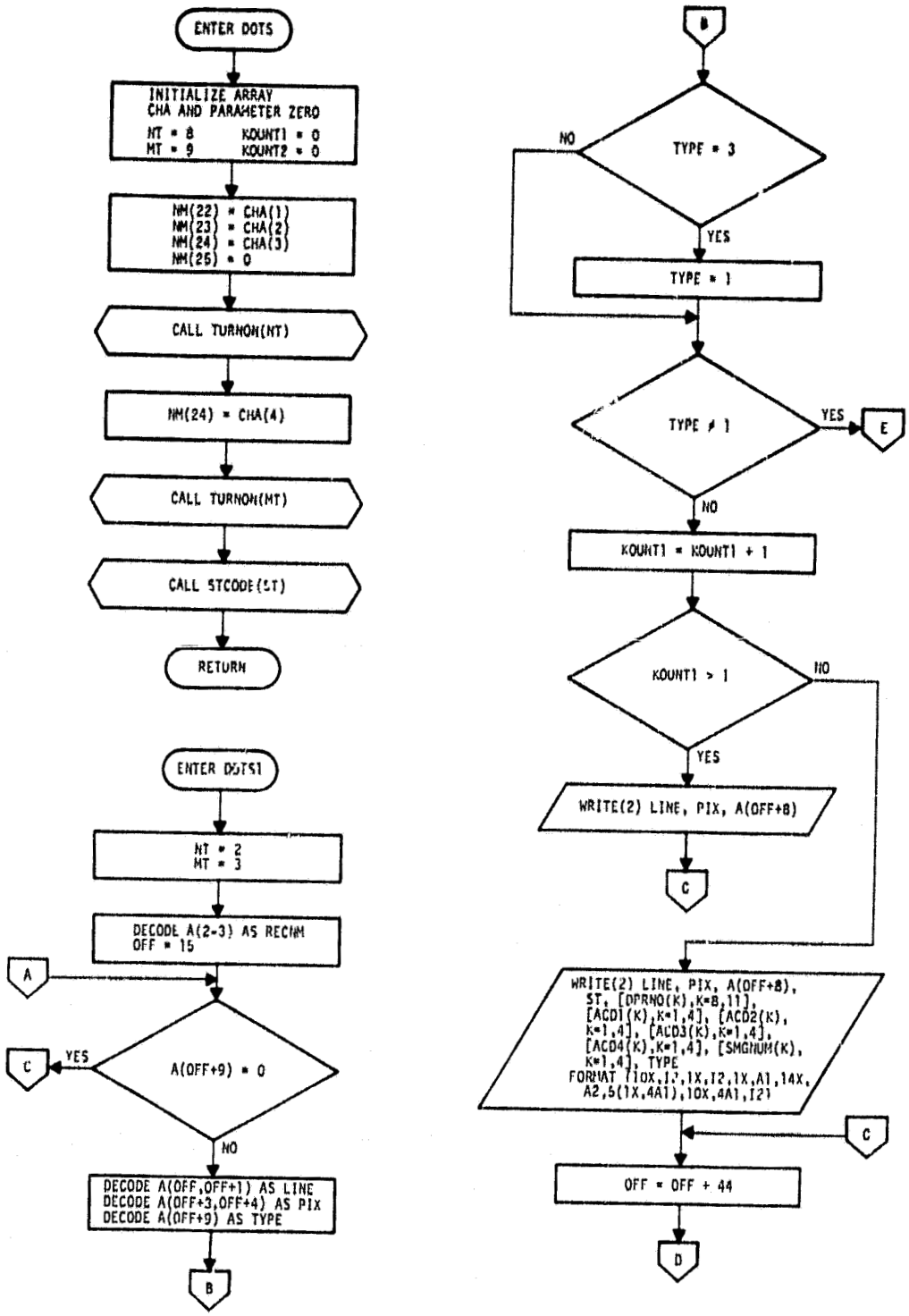

Figure 21.— Flow diagram for subroutine DOTS.

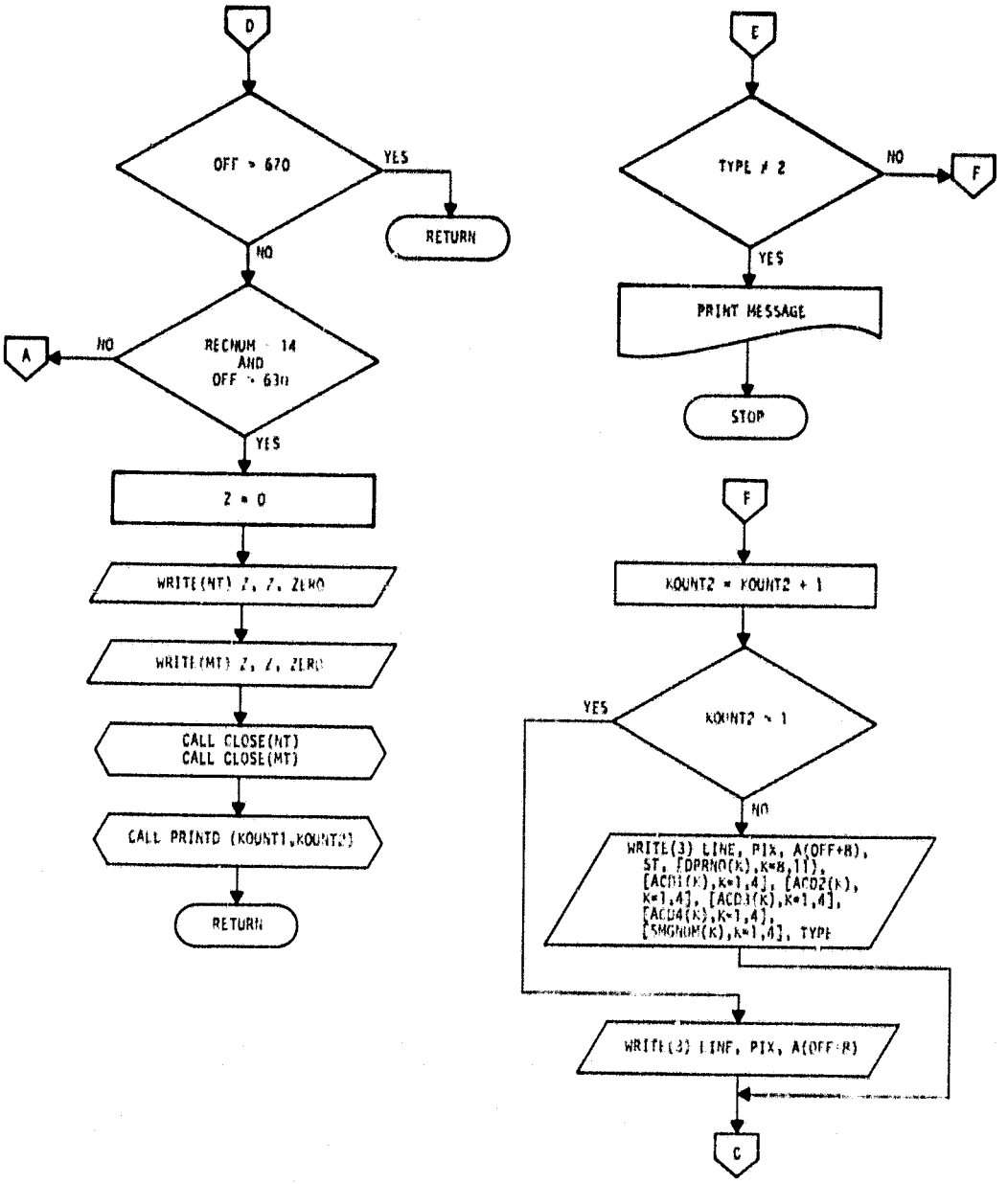

Figure 21.— Concluded.

```
DOTS.FTN        /VERIFY:CRAS/RE

0001            SUBROUTINE DOTS
0002            IMPLICIT INTEGER(A-Z)
        C       ***********************************************************
        C       ROUTINE REMOVES DATA FROM COIT DOTS, ITS, REFORMS INTO
        C       A FORM USEFUL FOR AN ANALYSIS PROGRAMS SUCH AS SPATL,
        C       ALLCOR, AND MLTCOR.
        C       LACIE64 VERSION
        C       TYPE 1 DOTS ARE SEPARATED FROM TYPE 2 DOTS AND BOTH ARE CHECKED TO
        C       SEE IF THEY WERE ALREADY LABELED. IF SO, THEY ARE WRITTEN
        C       TO DISC IN FILES SSSSYYDDD.AI1 AND SSSSYYDDD.AI2 RESPEC-
        C       TIVELY.
        C
        C       A SUBROUTINE STCODE REMOVES THE STATE CODE, ST(A5), FOR
        C       THE SEGMENT FROM A TABLE SO THAT IT CAN BE INCLUDED IN
        C       THE DOT FILES HERE, TO SAVE SPACE ONLY THE FIRST LINE OF
        C       DOTS 1 IN EACH FILE CONTAINS THE ANCILLARY DATA FOR THE
        C       SEGMENT AND CLASSIFICATION DATE.
        C***********************************************************
0003            BYTE IN(256),ACC1(4),ACC2(4),ACC3(4),ACC4(4),SHGANM(4),DPRNM(
                1 23),A(720),CHA(4),PERR
0004            DPRNM /ALLF/
0005            DATA/ALLF/
0006            DATA/DOTS/PRNT,ACC1,ACC2,ACC3,ACC4,SHGANM
0007            DATA CHA/1HA,1HI,1H1,1H2/
0008            DATA PER/1H?/
        C       SET UNIT NUMBERS
0009            NT = 8
0010            NT = 9
        C       SET COUNTERS FOR EACH CLASS OF DOT
0011            NT1 = 0
0012            NT2 = 0
0013            IN(22) = CHA(1)
0014            IN(23) = CHA(2)
0015            IN(24) = CHA(3)
0016            IN(25) = 0
        C       READ THE TYPE 1 DOTS FILE
0017            CALL RDT IN(NT)
0018            IN(24) = CHA(4)
        C       READ THE TYPE 2 DOTS FILE
0019            CALL RDT IN(NT)
        C       FIND AN ALPHANUMERIC STATE CODE
0020            CALL STCODE(ST)
0021            RETURN
0022            ENTRY DOTS1
        C       PROCESS DOT RECORDS
0023            NT = 7
0024            NT = 3
        C       FIND THE CURRENT RECORD NUMBER
0025            DECODE(2,100,A(2))NREC
0026            IFF = 15
        C       IF THIS DOT WAS NOT LABELED BY THE AITS SKIP TO THE NEXT DOT
0027      200   IF(A(IFF+9).EQ.PERR) GO TO 999
        C       DECODE THE LINE AND PIXEL NUMBERS OF A LABELED DOT
0028            DECODE(2,100,A(IFF))LINE
0029            DECODE(2,100,A(IFF+3))PIX
        C       FIND OUT WHAT TYPE DOT IT IS, A TYPE OF '3' MEANS A STARTING VECTOR
```

Figure 22.— Listing for subroutine DOTS.

ORIGINAL PAGE IS OF POOR QUALITY

```
C       TYPE 1 DATA SET BE ARRAY AS TYPE 1
        WRITE(1,11[illegible],A7,[illegible])TYPE
        IF(TYPE.EQ.3)TYPE=1
        IF(TYPE.EQ.1)GO TO 3[illegible]
C       WRITE OUT TYPE 1 DATA
        [illegible]
        IF(K.[illegible]T.1)GO TO 7[illegible]
        WRITE(6,14[illegible])L[illegible],PIX,4([illegible]),ST,(DPR[illegible](K),K
     1  =1,11),(ACD1(K),K=1,4),(ACD2(K),K=1,4),(ACD3(K),K=1,4),
     2  (ACD4(K),K=1,4),(SP[illegible](K),K=1,4),TYPE
        GO TO 7[illegible]
7[illegible]    WRITE(6,1[illegible])L[illegible],PIX,4([illegible])
7[illegible]    GO TO 999
3[illegible]    IF(TYPE.EQ.2)GO TO 666
C       WRITE OUT TYPE 2 DATA
        [illegible]
        IF(K.[illegible]T.1)GO TO 71[illegible]
        WRITE(6,14[illegible])L[illegible],PIX,4([illegible]),ST,(DPR[illegible](K),K=1,11),
     1  (ACD1(K),K=1,4),(ACD2(K),K=1,4),(ACD3(K),K=1,4),(ACD4(K),K=1,
     2  4),(SP[illegible](K),K=1,4),TYPE
        GO TO 999
7[illegible]    WRITE(6,1[illegible])L[illegible],PIX,4([illegible])
999     [illegible] = [illegible] + 46
        [illegible]
        IF([illegible].GT.63[illegible]) GO TO 888
        [illegible]
C       [illegible] WRITE [illegible] RECORDS CLOSE BOTH INPUT FILES AND PRINT
888     WRITE(6,14[illegible]) [illegible],[illegible],[illegible]
        WRITE(6,14[illegible]) [illegible],[illegible],[illegible]
        [illegible]
        [illegible]
        [illegible]
        [illegible] TYPE 2
C       [illegible] TYPE DATA WRITE ERROR MESSAGE
6[illegible]    WRITE(12,[illegible])
        [illegible]
1[illegible]    FORMAT(1H ,' ERROR IN DATS...IMPROPER TYPE ',A1,//)
1[illegible]    FORMAT([illegible])
1[illegible]    FORMAT([illegible])
14[illegible]   FORMAT([illegible],I7,1X,I7,[illegible],A1,I4X,A2,5(1X,4A1),1[illegible],4A1,I7[illegible]
        END
```

Figure 22.— Concluded.

3.3.11 SUBROUTINE STCODE

3.3.11.1 Linkage

STCODE is called once by subroutine DOTS.

3.3.11.2 Interface

STCODE interfaces with DOTS via COMMON block FNAME and passed parameter ST.

3.3.11.3 Input

There is no input to this subroutine.

3.3.11.4 Output

Subroutine STCODE has no output.

3.3.11.5 Storage

This subroutine requires 1095 words of storage.

3.3.11.6 Description

STCODE locates the correct two-character alphabetic state code, ST, for a given segment number via table lookup. Note: The table given is only valid for AA LACIE Phase III U.S. Great Plains blind sites.

3.3.11.7 Flow Chart

The flow diagram for subroutine STCODE is given in figure 23.

3.3.11.8 Listing

The listing for this subroutine is given in figure 24.

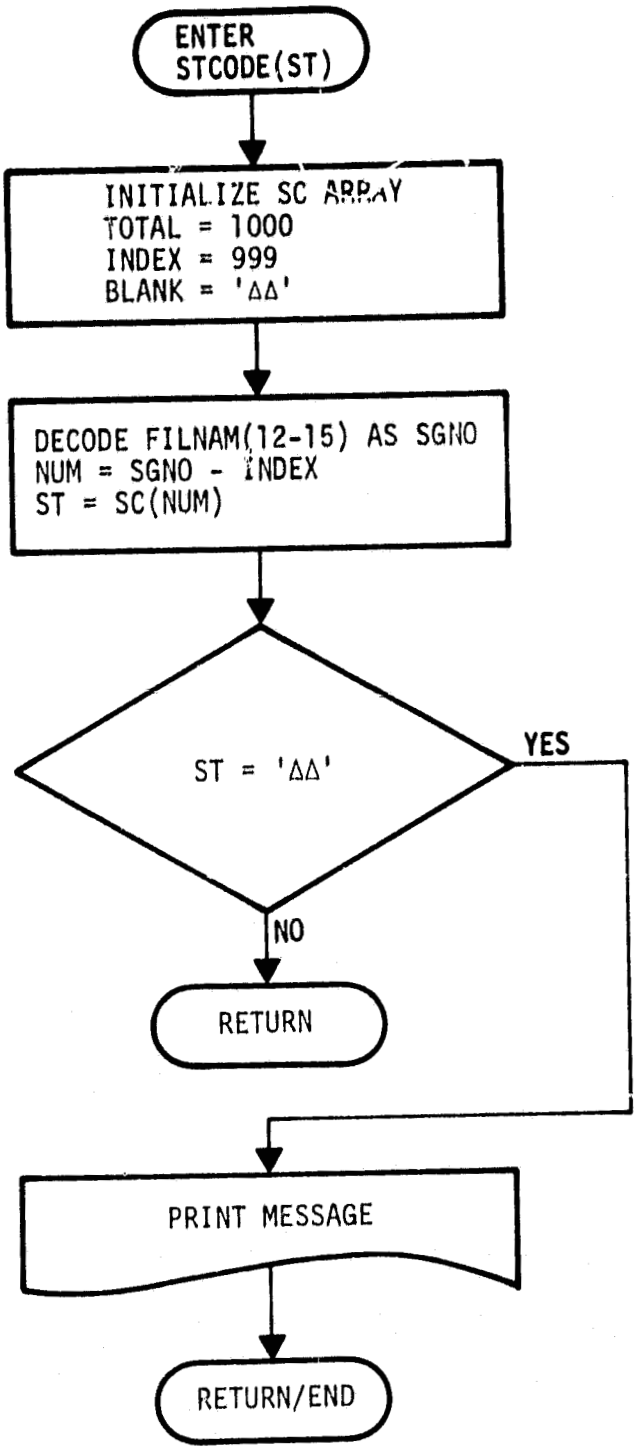

Figure 23.— Flow diagram for subroutine STCODE.

```
STCODE.FTN        /TR:PLUS/S/WR

0001                SUBROUTINE STCODE(ST)
0002                IMPLICIT INTEGER(A-Z)
          C         FIND THE TWO LETTER STATE CODE FOR THE LACIE PHASE III BLIND SITE
          C         SPECIFIED IN THE FILNAME
0003                DIMENSION SC(1000)
0004                BYTE FILNAM(24)
0005                COMMON/FNAME/FILNAM,SKIP
0006                DATA SC/1000*24  /
0007                DATA TOTAL/1000/
0008                DATA INDEX/999/
0009                DATA BLANK/2H  /
0010                DATA SC(1),SC(4),SC(8),SC(9),SC(12),SC(27),SC(33),SC(34),SC(49),
          1         SC(57),SC(59),SC(60),SC(61),SC(80),SC(87),SC(92),SC(95),SC(100),
          2         SC(103),SC(105),SC(154),SC(156),SC(159),SC(167),SC(171),SC(176),
          3         SC(191),SC(194),SC(206),SC(221),SC(223),SC(224),SC(225),SC(229),
          4         SC(232),SC(234),SC(237),SC(238),SC(240),SC(243),SC(245),SC(264),
          5         SC(267),SC(269),SC(272),SC(275),SC(280),SC(286),SC(291),SC(294),
          6         SC(295),SC(298),SC(325),SC(341),SC(344),SC(347),SC(356),SC(363),
          7         SC(365),SC(368),SC(370),SC(371),SC(372),SC(379),SC(399),SC(451),
          8         SC(452),SC(498),SC(499),SC(502),SC(503),SC(507),SC(508),SC(513),
          9         SC(514),SC(518),SC(521),SC(522),SC(523),SC(524),SC(530),SC(532),
          *         SC(533),SC(536),SC(545),SC(557),SC(561),SC(565),SC(567),SC(569),
          *         SC(572),SC(576),SC(578),SC(586),SC(593),SC(597),SC(599),SC(603),
          1         SC(595),SC(596),SC(613),SC(605),SC(607),SC(617),SC(620),SC(623),
          2         SC(628),SC(635),SC(639),SC(641),SC(645),SC(648),SC(652),SC(662)
          3         /5*2HCS,7*2HKS,2*2HPV,5*2HTX,5*2HCZ,2*2HWT,8*2HKS,13*2HCX,5*2HTX,
          4         4*2HKS,2HTX,3*2HWS,5*2HKS,2*2HTX,4*2HWS,2*2HSD,4*2HCW,7*2HMN,
          5         5*2HWT,14*2HNE,14*2HND/
0011                DATA SC(664),SC(666),SC(668),SC(675),SC(676),SC(678),SC(682),
          *         SC(687),
          4         SC(691),SC(695),SC(700),SC(726),SC(731),SC(735),SC(740),SC(742),
          5         SC(743),SC(748),SC(751),SC(754),SC(801),SC(803),SC(804),
          6         SC(806),SC(808),SC(812),SC(826),SC(831),SC(836),SC(840),SC(850),
          7         SC(851),SC(852),SC(854),SC(860),SC(862),SC(865),SC(870),SC(874),
          8         SC(882),SC(884),SC(911),SC(895),SC(898),SC(900),SC(903),SC(904),
          9         SC(914),SC(925),SC(928),SC(930),SC(938),SC(945),SC(949)/2HND,
          2         10*2HSD,10*2HWT,4*2HSD,5*2HMN,2HCE,5*2HKS,2*2HMN,3*2HKS,2HMN,
          3         7*2HWT,4*2HWT/
0012                READ(4,200,FILNAM(12))SGNO
0013                NUM = SGNO - INDEX
0014                ST = SC(NUM)
0015                IF(ST.EQ.BLANK) GO TO 900
0016                RETURN
0017      900       PRINT 100,SGNO
0018      100       FORMAT(1H ,10X,'NO STATE CODE ASSIGNED FOR SEGMENT',I5,/)
0019      200       FORMAT(I4)
0020                RETURN
0021                END
```

Figure 24.— Listing for subroutine STCODE.

ORIGINAL PAGE IS OF POOR QUALITY

3.3.12 SUBROUTINE PRNTIT

3.3.12.1 Linkage

PRNTIT is called by CCIT6A via entries PRNTIT and PRINTE, by HEADER via entry PRINTH, by INPUT via entry PRINTI, by DOTS via entry PRINTD, and by CLUST via entry PRINTC. All other called routines are Image Processor system routines.

3.3.12.2 Interface

PRNTIT interfaces with HEADER via COMMON block DOTS, with INPUT via COMMON block FNAME, with DOTS via passed parameters K1 and K2, and with CLUST via passed parameter RCNUM.

3.3.12.3 Input

There is no input to this subroutine.

3.3.12.4 Output

PRNTIT prints messages on the line printer.

3.3.12.5 Storage

This subroutine requires 783 words of storage.

3.3.12.6 Description

PRNTIT provides most line printer output for the CCIT6A processor. This output provides a processing record for AA status and tracking activity. The routine uses system routines TIME and DATE to obtain data for header and trailer line printer messages for each run.

3.3.12.7 Flow Chart

The flow diagram for subroutine PRNTIT is given in figure 25.

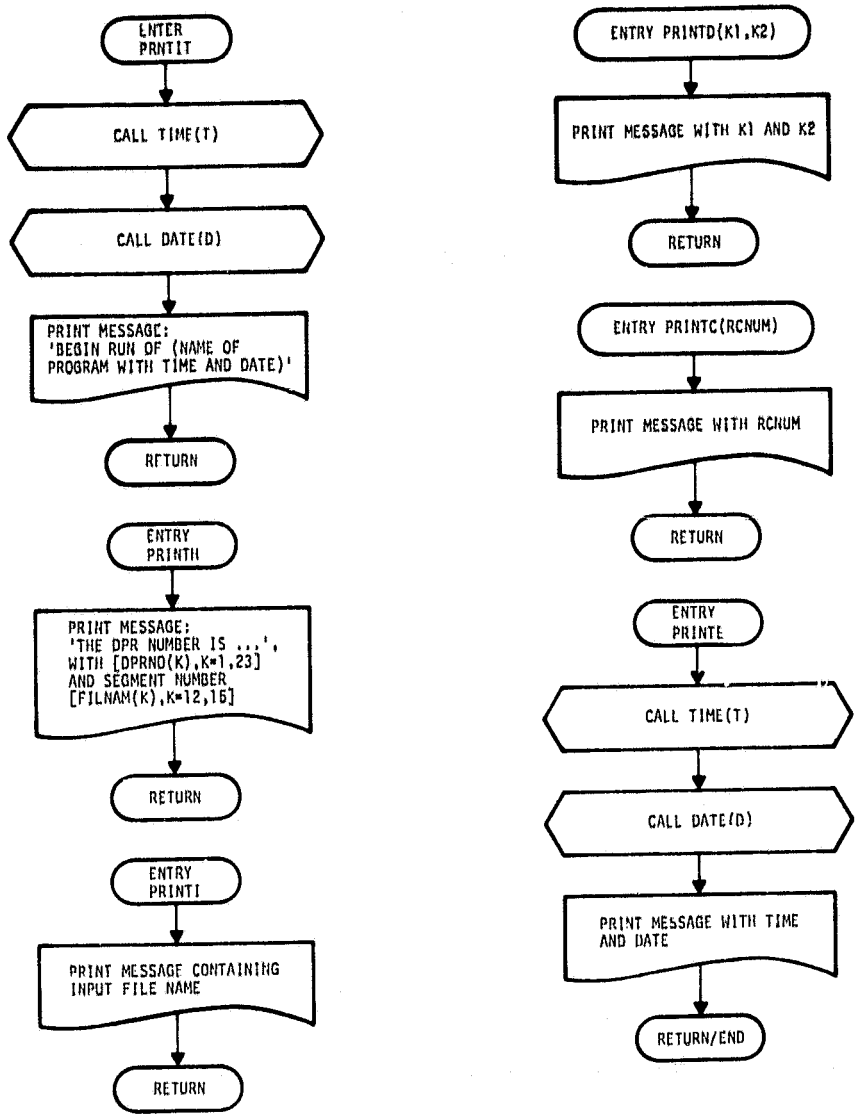

Figure 25.— Flow diagram for subroutine PRNTIT.

3.3.12.8 Listing

The program listing for this subroutine is given in figure 26.

```
0001          SUBROUTINE PRNTIT
0002          IMPLICIT INTEGER(A-Z)
        C     ***TEST SUBROUTINE TO PRINT HEADERS AND DATA OUTPUTS FOR CCIT
        C     PROCESSOR PROGRAM...LACIEGA VERSION
0003          BYTE A(72),FILNAM(24),DPR#(23),ACD1(4),ACD2(4),ACD3(4),
            * ACD4(4),SMGNUM(4),T(8),D(9)
0004          COMMON/FRAME/FILNAM,SKIP
0005          COMMON/DATS/DPR#,ACD1,ACD2,ACD3,ACD4,SMGNUM
0006          COMMON/BUF/A
0007          CALL TIME(T)
0008          CALL DATE(D)
0009          PRINT 100,D,T
0010    100   FORMAT(1H1,'    BEGIN RUN OF CCITGA PROCESSOR PROGRAM',//,
            *10X,'ON  ',9A1,'  AT  ',8A1)
0011          RETURN
0012          ENTRY PRINTH
0013          PRINT 500,(DPR#(K),K=1,23),(FILNAM(K),K=12,15)
0014    500   FORMAT(1H ,' THE DPR NUMBER IS ',23A1,' FOR SEGMENT NUMBER',
            *1X,4A1)
0015          RETURN
0016          ENTRY PRINTI
0017          PRINT 600,(FILNAM(K),K=1,24)
0018    600   FORMAT(1H1,' PROGRAM CCITGA',//,1H ,10X,'ON FILE ',24A1,//)
0019          RETURN
0020          ENTRY PRINTB(K1,K2)
0021          PRINT 700,K1,K2
0022    700   FORMAT(1H ,10X,' THERE WERE',I3,' TYPE 1, AND',
            *I3,' TYPE 2 PATS ASSIGNED')
0023          RETURN
0024          ENTRY PRINTC(RCOUN)
0025          PRINT 800,RCOUN
0026    800   FORMAT(1H ,10X,'PROCESSING',I3,' CLUSTERS FOR DOT MATCH')
0027          RETURN
0028          ENTRY PRINTE
0029          CALL TIME(T)
0030          CALL DATE(D)
0031          PRINT 200,D,T
0032          PRINT 200,D,T
0033    200   FORMAT(1H ,'    END RUN OF CCITGA PROCESSOR PROGRAM',//,10X,
            *'ON  ',9A1,'  AT  ',8A1)
0034          RETURN
0035          END
```

Figure 26.— Listing for subroutine PRNTIT.

ORIGINAL PAGE IS OF POOR QUALITY

4. OPERATIONS

This section presents all information necessary to obtain proper execution of the CCIT6A processor program.

4.1 OPERATORS GUIDE

This section explains the system hardware configuration and execution (run) setup for the CCIT6A.

4.1.1 HARDWARE CONFIGURATION

The nominal configuration is the Earth Observations Division/Data Techniques Laboratory (EOD/DTL) PDP 11/45 processor with the RSX 11-D operating system. The system must have the input CCIT files resident on either the system disk or a user disk. The output files are written onto the same disk and under the same UIC as the resident input data. The input files are created using program AACCIT, described in JSC-13893. (See section 2 of this specification.)

4.1.2 PROGRAM EXECUTION

4.1.2.1 INTERACTIVE SETUP

a. Edit file CCIT6A.DAT for the proper file name and the value of parameter SKIP (24A1,I2). The file name takes the form:

DBX:[abc,d]SSSSYYDDD.wxy

where

X = Disk unit number

SSSSYYDDD = Input file name

wxy = Input file type; i.e., CCO

[abc,d] = User UIC for the input file

b. Mount the proper disk pack on the drive.

c. Type 'RUN CCIT6A'.

d. When message CCIT6A-STOP appears on the monitor, collect a single-page report at the line printer, and check the listing to ensure that the ending message was printed and that the various steps were properly executed.

4.1.2.2 BATCH SETUP

a. Prepare a batch run request detailing the disk configuration required.

b. Set up a batch run deck as in table 2. The required steps follow:

- Delete CCIT6A.DAT.
- Create CCIT6A.DAT with a card image, as given in section 4.1.2.1.
- Run CCIT6A.TSK.

4.2 USERS GUIDE

The CCIT6A program is designed to obtain a small fraction of the data from a CCIT disk file and to reformat these data into a form directly used by several AA software modules. This program will not execute for CCIT's other than those created under LACIE version 6A. The approximate dates of valid CCIT's for 6A are 77225 through 77305. An upgrade of the program to process LACIE version 7 CCIT's is underway.

4.3 MAINTENANCE DOCUMENTATION

Not applicable.

TABLE 2.— BATCH RUN DECK SETUP

```
$JOB/NAME=AA/MCR/LIMIT=99/ACCOUNT=1106
$MCR PIP
CCIT6A.DAT;*/DE
$CREATE CCIT6A.DAT
   ⋮
Card images for file name and SKIP parameter (24A1,I2)
   ⋮
Blank card
$EOD
$MCR REM RSXBAT
$RUN CCIT6A.TSK
$EOJ
```

APPENDIX

FORMATS FOR .CLO FILE

TABLE A-1.— FORMAT OF FIRST RECORD OF .CLO FILE

.CLO file byte number	Data description (ASCII)	CCIT 'B' record byte number
1	Class 1 label (W, S, G, or blank)	8
2-6	Pixel population; PPPPP	9-13
7-9	Uncorrected proportion; M.MM (implied decimal point)	14-16
10-12	Corrected proportion; N.NN (implied decimal point)	17-19
13-16	Variance; .VVVV (implied decimal point)	20-23
17	Class 2 label (W, S, G, or blank)	55
18-22	Pixel population; PPPPP	56-60
23-25	Uncorrected proportion; M.MM (implied decimal point)*	61-63
26-28	Corrected proportion; N.NN (implied decimal point)	64-66
29-32	Variance; .VVVV (implied decimal point)	67-70
33	Nongrains class label (N)	102
34-38	Pixel population; PPPPP*	103-107
39-41	Uncorrected proportion; M.MM (implied decimal point)	108-110
42-44	Corrected proportion; N.NN (implied decimal point)	111-113

*Pixels in the classes designated other (DO) or designated unidentifiable (DU) are not included.

TABLE A-2.– FORMAT OF CLUSTER-DOT MATCH IN .CLO FILE

.CLO record	Contents
2	Single integer giving the number of clusters in the classification, CNUM.
3	12 bytes of ASCII character data for each cluster; e.g., 12*CNUM bytes of data. The first six bytes of each group of 12 are the cluster label; e.g., NOCL17. The last six bytes of each group are the identity of the dot used to label the cluster; e.g., DOT103. Only type 1 dots are used to label clusters.