NOTICE

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
"AS-BUILT" SPECIFICATION FOR CCITT
PROCESSOR PROGRAM

Job Orders 73-783 and 71-475

(TIRF 78-0022)

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

National Aeronautics and Space Administration

MAY 1980
RECEIVED
NASA STI FACILITY,
ACCESS DEPT.
"AS-BUILT" SPECIFICATION FOR CCIT7 PROCESSOR PROGRAM

Job Orders 73-783 and 71-475
(TIRF 78-0022)

PREPARED BY
W. P. White

APPROVED BY

[Signature]
P. L. Krumm, Acting Supervisor
Scientific Applications Software Section

[Signature]
E. M. Hsu, Supervisor
Accuracy Assessment Section

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

November 1978
The program CCIT 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 7). 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).
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABBREVIATIONS.</td>
<td>vii</td>
</tr>
<tr>
<td>1. SCOPE.</td>
<td>1-1</td>
</tr>
<tr>
<td>2. APPLICABLE DOCUMENTS</td>
<td>2-1</td>
</tr>
<tr>
<td>3. SYSTEM DESCRIPTION</td>
<td>3-1</td>
</tr>
<tr>
<td>3.1 HARDWARE DESCRIPTION</td>
<td>3-1</td>
</tr>
<tr>
<td>3.2 MODULE DESCRIPTION</td>
<td>3-1</td>
</tr>
<tr>
<td>3.3 SOFTWARE DESCRIPTION</td>
<td>3-6</td>
</tr>
<tr>
<td>3.3.1 MODULE CCIT7</td>
<td>3-7</td>
</tr>
<tr>
<td>3.3.2 SUBROUTINE READH</td>
<td>3-13</td>
</tr>
<tr>
<td>3.3.3 SUBROUTINE READRC</td>
<td>3-17</td>
</tr>
<tr>
<td>3.3.4 SUBROUTINE HEADER</td>
<td>3-20</td>
</tr>
<tr>
<td>3.3.5 SUBROUTINE BIASC7</td>
<td>3-24</td>
</tr>
<tr>
<td>3.3.6 SUBROUTINE CLUST7</td>
<td>3-28</td>
</tr>
<tr>
<td>3.3.7 SUBROUTINE RITE7</td>
<td>3-32</td>
</tr>
<tr>
<td>3.3.8 SUBROUTINE TURNON</td>
<td>3-35</td>
</tr>
<tr>
<td>3.3.9 SUBROUTINE DOTS7</td>
<td>3-38</td>
</tr>
<tr>
<td>3.3.10 SUBROUTINE STCOD7</td>
<td>3-44</td>
</tr>
<tr>
<td>3.3.11 SUBROUTINE PRINT7</td>
<td>3-47</td>
</tr>
<tr>
<td>4. OPERATIONS</td>
<td>4-1</td>
</tr>
<tr>
<td>4.1 OPERATORS GUIDE</td>
<td>4-1</td>
</tr>
<tr>
<td>4.1.1 HARDWARE CONFIGURATION</td>
<td>4-1</td>
</tr>
<tr>
<td>4.1.2 PROGRAM EXECUTION</td>
<td>4-1</td>
</tr>
</tbody>
</table>
Section | Page
---|---
4.2 USERS GUIDE | 4-2
4.3 MAINTENANCE DOCUMENTATION | 4-2
APPENDIX — FORMAT FOR .CLO FILE. | A-1

**TABLES**

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TASK-BUILDER COMMAND FILE FOR CCITT PROCESSOR PROGRAM</td>
</tr>
<tr>
<td>2</td>
<td>BATCH RUN DECK SETUP.</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>------------------------------------------------</td>
</tr>
<tr>
<td>1</td>
<td>Data flow of the CCIT7 processor program.</td>
</tr>
<tr>
<td>2</td>
<td>Functional flow of the CCIT7 processor program.</td>
</tr>
<tr>
<td>3</td>
<td>Flow diagram for the CCIT7 processor program.</td>
</tr>
<tr>
<td>4</td>
<td>Listing for the CCIT7 processor program</td>
</tr>
<tr>
<td>5</td>
<td>Flow diagram for subroutine READH</td>
</tr>
<tr>
<td>6</td>
<td>Listing for subroutine READH.</td>
</tr>
<tr>
<td>7</td>
<td>Flow diagram for subroutine READR.C.</td>
</tr>
<tr>
<td>8</td>
<td>Listing for subroutine READR.C</td>
</tr>
<tr>
<td>9</td>
<td>Flow diagram for subroutine HEADER</td>
</tr>
<tr>
<td>10</td>
<td>Listing for subroutine HEADER</td>
</tr>
<tr>
<td>11</td>
<td>Flow diagram for subroutine BIAS.C7.</td>
</tr>
<tr>
<td>12</td>
<td>Listing for subroutine BIAS.C7</td>
</tr>
<tr>
<td>13</td>
<td>Flow diagram for subroutine CLUST7.</td>
</tr>
<tr>
<td>14</td>
<td>Listing for subroutine CLUST7.</td>
</tr>
<tr>
<td>15</td>
<td>Flow diagram for subroutine RITE7</td>
</tr>
<tr>
<td>16</td>
<td>Listing for subroutine RITE7.</td>
</tr>
<tr>
<td>17</td>
<td>Flow diagram for subroutine TURNON</td>
</tr>
<tr>
<td>18</td>
<td>Listing for subroutine TURNON</td>
</tr>
<tr>
<td>19</td>
<td>Flow diagram for subroutine DOTS7</td>
</tr>
<tr>
<td>20</td>
<td>Listing for subroutine DOTS7.</td>
</tr>
<tr>
<td>21</td>
<td>Flow diagram for subroutine STCOD7.</td>
</tr>
<tr>
<td>22</td>
<td>Listing for subroutine STCOD7</td>
</tr>
<tr>
<td>23</td>
<td>Flow diagram for subroutine PRINT7.</td>
</tr>
<tr>
<td>24</td>
<td>Listing for subroutine PRINT7.</td>
</tr>
</tbody>
</table>
### ABBREVIATIONS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AA</td>
<td>Accuracy Assessment</td>
</tr>
<tr>
<td>CAMS</td>
<td>Classification and Mensuration Subsystem</td>
</tr>
<tr>
<td>CAS</td>
<td>Crop Assessment Subsystem</td>
</tr>
<tr>
<td>CCIT</td>
<td>CAMS/CAS interface tape</td>
</tr>
<tr>
<td>DEC</td>
<td>Digital Equipment Corporation</td>
</tr>
<tr>
<td>DPR</td>
<td>Data processing request</td>
</tr>
<tr>
<td>DTL</td>
<td>Data Techniques Laboratory</td>
</tr>
<tr>
<td>DTRM</td>
<td>Data terminal</td>
</tr>
<tr>
<td>EOD</td>
<td>Earth Observations Division</td>
</tr>
<tr>
<td>ERIPS</td>
<td>Earth Resources Interactive Processing System</td>
</tr>
<tr>
<td>LACIE</td>
<td>Large Area Crop Inventory Experiment</td>
</tr>
<tr>
<td>Pixel</td>
<td>Picture element</td>
</tr>
<tr>
<td>SAE</td>
<td>Stratified areal estimate</td>
</tr>
<tr>
<td>TIRF</td>
<td>Transmittal Information Request Form</td>
</tr>
<tr>
<td>UIC</td>
<td>User identification code</td>
</tr>
</tbody>
</table>
1. SCOPE

This document specifies the detailed design for a software module called CCIT7, which will manipulate and extract data from Accuracy Assessment (AA) data base files. These files are derived from Large Area Crop Inventory Experiment (LACIE), version 7, Classification and Mensuration Subsystem/Crop Assessment Subsystem (CAMS/CAS) interface tapes (CCIT's). 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.


3. SYSTEM DESCRIPTION

The CCIT7 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 or user terminal
b. Line printer
c. Two disk units

3.2 MODULE DESCRIPTION

The CCIT7 module is implemented on the PDP 11/45 for time-sharing or 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 CCIT7 module represents a major revision of the CCIT6A module described in reference e (section 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.
Figure 1.— Data flow of the CCIT7 processor program.
Figure 2.— Functional flow of the CCIT7 processor program.
Figure 2.— Concluded.
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 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 'BA' record is searched for and processed. The program extracts classification data on class picture element (pixel) populations, the ERIPS estimate, the stratified areal estimate (SAE), and the variance for each class. An output file SSYYDDDD.CLO (where SSY is the segment number and YYDDD is the classification date derived from the DPR number) is opened. The first record of the .CLO file contains an integer word giving the number of crop classes for the classification. The estimate data are written as the second record. Sixteen bytes are required for each class; a LACIE version 7 CCIT can contain data on up to 26 classes. A line printer report is output after the disk files are written.

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 15 clusters, so some of these data appear in additional 'C' records.) When all the cluster-dot data are assembled into a buffer, the number of clusters is written out as the third record of file SSYYDDDD.CLO, and the match data are written as the second record of this file. A line printer report is then output, and the SSYYDDDD.CLO file is closed.

Then the program tests to determine if analyst-labeled dot output files are required; this is the default condition. If this condition exists, the output files SSYYDDDD.AI1 and SSYYDDDD.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 SSSSYYDDDAII and SSSSYYDDDAII2 output files and the input file are closed. No report is output detailing the .AI file dot data because these formatted files are easily examined using the Digital Equipment Corporation (DEC) PDP-11 utility program PIP.

3.3 SOFTWARE DESCRIPTION

The CCIT7 processor program consists of 11 user-supplied routines: CCIT7 (main program), READH, READRC, HEADER, BIASC7, CLUST7, RITE7, TURNON, DOTS7, STCOD7, and PRINT7. The program makes use of a card-image-formatted file, CCIT7.DAT, for program control and the line printer and user disk for output. The following sections provide a detailed description of each of the 11 routines. The recommended task-build command file (CCIT7.CMD), used to create the load module (CCIT7.TSK), is given in table 1.

TABLE 1. - TASK-BUILDER COMMAND FILE FOR CCIT7 PROCESSOR PROGRAM

| CCIT7,LP:/SH=CCIT7,READH,READRC,HEADER,BIASC7,CLUST7,RITE7,TURNON,DOTS7,STCOD7,PRINT7 |
| 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 Interface subsections and to the compiler listings of each routine provides sufficient information to follow the data flow throughout the program.

3.3.1 MODULE CCIT7

3.3.1.1 Linkage

The CCIT7 program is the main program. It calls user subroutines READH, READRC, HEADER, BIASC7, CLUST7, DOTS7, and PRINT7. Subroutines BIASC7, DOTS7, and PRINT7 are called using multiple entry points.

3.3.1.2 Interface

Most communication with the user routines is handled via COMMON blocks. A single integer parameter is passed on call to READH, which indicates the number of CCIT header records to be read. A single integer parameter Q is passed to CLUST7 as a flag.

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 CCIT7.DAT. The value of SKIP determines whether the dot records are to be processed. If SKIP is nonzero, the dots are not processed.

3.3.1.2.3 COMMON Block B7

B7 contains the number of categories, NCAT, for the classification.
3.3.1.2.4 COMMON Block CLUSTR

CLUSTR contains a 60-byte by 12-byte array, CNAME, and an integer variable CNUM. CLUSTR provides an interface between subroutine CLUST7 and subroutine RITE7.

3.3.1.3 Input

The CCIT7 program receives all input CCIT data via subroutine READRC. Control data are provided directly by reading card images from file CCIT7.DAT.

3.3.1.4 Output

The CCIT7 program provides all output via subroutines PRINT7, RITE7, BIASC7, DOTS7, and CLUST7.

3.3.1.5 Storage

The CCIT7 program requires 1078 words of storage.

3.3.1.6 Description

The CCIT7 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 and by counts based upon the required number of records of a given type.

3.3.1.7 Flow Chart

The flow chart for CCIT7 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 CCIT7 processor program.
Figure 3.— Concluded.
Figure 4.— Listing for the CCIT7 processor program.
<table>
<thead>
<tr>
<th>Requirement</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C204</td>
<td>C200</td>
<td>C200</td>
</tr>
<tr>
<td>C201</td>
<td>C200</td>
<td>C200</td>
</tr>
<tr>
<td>C202</td>
<td>C200</td>
<td>C200</td>
</tr>
<tr>
<td>C203</td>
<td>C200</td>
<td>C200</td>
</tr>
<tr>
<td>C204</td>
<td>C200</td>
<td>C200</td>
</tr>
</tbody>
</table>

Figure 4.— Concluded.
3.3.2 SUBROUTINE READH

3.3.2.1 Linkage

Subroutine READH calls subroutine TURNON.

3.3.2.2 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 (see section 3.3.1.2.2) interfaces CCIT7 with READH. COMMON block BUF provides no true interfacing function for this routine.

3.3.2.2.1 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 PRINT7. The last byte of array NM should contain the null (0) character.

3.3.2.3 Input

Header records from the CCIT input file are input.

3.3.2.4 Output

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

3.3.2.5 Storage

READH requires 504 words of storage.

3.3.2.6 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 CCIT7.
3.3.2.7 Flow Chart

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

3.3.2.8 Listing

The listing for this subroutine is given in figure 6.
1. PASS PARAMETER Q ON INPUT
2. COPY FILNAM INTO NM

CALL TURNON(1)

READ Q, 80-BYTE RECORDS FROM UNIT(1)

RETURN/END

Figure 5.— Flow diagram for subroutine READH.
3.3.3 SUBROUTINE READRC

3.3.3.1 Linkage
Subroutine READRC is called by CCIT7.

3.3.3.2 Interface
Subroutine READRC interfaces with CCIT7 via COMMON block BUF (see section 3.3.1.2.1).

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

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

3.3.3.5 Storage
This subroutine requires 441 words of storage.

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

3.3.3.7 Flow Chart
The flow diagram for subroutine READRC 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 READRC.
<table>
<thead>
<tr>
<th>Page 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAGE 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>0001</th>
<th>0002</th>
<th>0003</th>
<th>0004</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>100</td>
<td>200</td>
<td>300</td>
</tr>
</tbody>
</table>

Figure 8.— Listing for subroutine READC.
3.3.4 SUBROUTINE HEADER

3.3.4.1 Linkage
Subroutine HEADER is called by CCIT7.

3.3.4.2 Interface
HEADER interfaces with CCIT7 via COMMON blocks BUF (see section 3.3.1.2.1) and DOTS and interfaces with PRINT7 (entry PRINTH) via COMMON block DOTS.

3.3.4.2.1 COMMON Block DOTS
DOTS contains a 23-byte array (DPRNO) that is used to hold the ERIPS DPR number, four 4-byte arrays (ACD1, ACD2, ACD3, and ACD4) that are used to store the acquisition dates used for the ERIPS run, and a 4-byte array (SMGNUM) that is used to store the LACIE segment number.

3.3.4.3 Input
There is no input to this subroutine.

3.3.4.4 Output
HEADER has no output.

3.3.4.5 Storage
This subroutine requires 445 words of storage.

3.3.4.6 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 PRINT7.
3.3.4.7 Flow Chart

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

3.3.4.8 Listing

The listing for this subroutine is given in figure 10.
ENTER HEADER

SET ACD1, ACD2, ACD3, AND ACD4 TO ALL BLANKS;
SET ZERO TO 'O'

A(169) ≠ 0

YES

B

NO

A(164) ≠ 0

NO

A(164) ≠ 0

YES

ACD1(k) = A(k+164)
ACD2(k) = A(k+159)
ACD3(k) = A(k+154)
K = 1,4

C

A

ACD1(k) = A(k+159)
ACD2(k) = A(k+154)
K = 1,4

C

ACD1(k) = A(k+169)
ACD2(k) = A(k+164)
ACD3(k) = A(k+159)
ACD4(k) = A(k+154)
K = 1,4

C

DPRND(k) = A(k+2)
SMGNM(k) = A(k+147)
K = 1,4

CALL PRINTH

RETURN/END

Figure 9.— Flow diagram for subroutine HEADER.
Figure 10.— Listing for subroutine HEADER.
3.3.5 SUBROUTINE BIASC7

3.3.5.1 Linkage

BIASC7 is called once by the CCIT7 program. It calls subroutine TURNON once. An additional entry point, BIAS, is called one or more times by CCIT7.

3.3.5.2 Interface

BIASC interfaces with CCIT7 via COMMON blocks BUF (see section 3.3.1.2.1) and B7 (see section 3.3.1.2.3), and with TURNON via COMMON block NAME (see section 3.3.2.2.1).

3.3.5.3 Input

There is no input to this subroutine.

3.3.5.4 Output

BIASC7 writes two unformatted records onto unit 3. The data contained in this record are detailed in the appendix.

3.3.5.5 Storage

This subroutine requires 837 words of storage.

3.3.5.6 Description

BIASC7 codes the output file name as SSSSYYDDD.CLO, where SSSS is the segment number and YYDDD is the classification date. Unit 3 is opened for output via a call to subroutine TURNON. Then a record containing the integer NCAT (the number of categories or classes) is written on unit 3, and 12 times NCAT bytes of character data are output as the second record.

3.3.5.7 Flow Chart

The flow diagram for subroutine BIASC7 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 BIASC7.
Figure 12.- Listing for subroutine BIASC7.
3.3.6 SUBROUTINE CLUST7

3.3.6.1 Linkage
Subroutine CLUST7 is called one to four times by CCIT7. CLUST7 calls RITE7 for data output and calls the PRINTC entry to PRINT7 for an output message.

3.3.6.2 Interface
CLUST7 interfaces with CCIT7 via COMMON block BUF (see section 3.3.1.2.1) and passed parameter Q, with RITE7 via COMMON block CLUSTR (see section 3.3.1.2.4), and with PRINT7 via passing parameter RCNUM on call.

3.3.6.3 Input
There is no input to this subroutine.

3.3.6.4 Output
Subroutine CLUST7 has no output.

3.3.6.5 Storage
This subroutine requires 836 words of storage.

3.3.6.6 Description
CLUST7 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 cluster.

On the first record processed, CLUST7 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(1-15)]. 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. CLUST7 is called via entry PRINTC to print a message containing the parameter RCNUM. When CNUM sets of data are written into CNAME, CLUST7 calls subroutine RITE7 for output.
3.3.6.7 Flow Chart
The flow diagram for subroutine CLUST7 is given in figure 13.

3.3.6.8 Listing
The listing for this subroutine is given figure 14.
Figure 13.— Flow diagram for subroutine CLUST7.
Figure 14.— Listing for subroutine CLUST7.
3.3.7 SUBROUTINE RITE7

3.3.7.1 Linkage

Subroutine RITE7 is called once by subroutine CLUST7.

3.3.7.2 Interface

Subroutine RITE7 interfaces with CLUST7 via COMMON block CLUSTR (see section 3.3.1.2.4).

3.3.7.3 Input

There is no input to RITE7.

3.3.7.4 Output

RITE7 writes two records onto a previously opened file (unit 3). This unit is opened in subroutine BIASC7 as an unformatted FORTRAN disk file. In addition, RITE7 writes a line printer report of the cluster-dot match data for AA evaluation.

3.3.7.5 Storage

This subroutine requires 484 words of storage.

3.3.7.6 Description

RITE7 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.

3.3.7.7 Flow Chart

The flow diagram for subroutine RITE7 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 RITE7.
Figure 16.— Listing for subroutine RITE7.
3.3.8 SUBROUTINE TURNON

3.3.8.1 Linkage
Subroutine TURNON is called by subroutines READH, BIASC7, and DOTS7.

3.3.8.2 Interface
TURNON interfaces with its calling routines via COMMON block NAME (see section 3.3.2.2.1) and a passed parameter, NT.

3.3.8.3 Input
There is no input to this subroutine.

3.3.8.4 Output
TURNON has no output.

3.3.8.5 Storage
This subroutine requires 162 words of storage.

3.3.8.6 Description
TURNON opens a file with the file name contained in byte array NM. If NT equals 1, the input file is opened as UNIT equals 1. If NT equals 2 to 6, an unformatted file is opened as unit NT. If NT is greater than 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.

3.3.8.7 Flow Chart
The flow diagram for subroutine TURNON 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 TURNON.
Figure 18.— Listing for subroutine TURNON.
3.3.9 SUBROUTINE DOTS7

3.3.9.1 Linkage

Subroutine DOTS7 is called by CCIT7 once via the main entry and 14 times via entry DOTS1. DOTS7 calls subroutine TURNON twice and subroutines PRINT7 (via entry PRINTD) and STCOD7 once.

3.3.9.2 Interface

Subroutine DOTS7 interfaces with TURNON via COMMON block NAME (see section 3.3.2.2.1) and with CCIT7 via COMMON blocks BUF (see section 3.3.1.2.1) and DOTS (see section 3.3.4.2.1).

3.3.9.3 Input

There is no input to this subroutine.

3.3.9.4 Output

Subroutine DOTS7 writes formatted, card-image records onto two disk-based output files opened on the initial call to the routine.

3.3.9.5 Storage

This subroutine requires 976 words of storage.

3.3.9.6 Description

DOTS7 processes CCIT 'D' records into two formatted files of analyst-labeled dots. When called as DOTS7, 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 STCOD7 is called to obtain the two-byte parameter ST, the alphabetic state code for the segment. Control then returns to CCIT7.
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 CCIT7), 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 the PRINT7 subroutine entry PRINTD.

3.3.9.7 Flow Chart
The flow diagram for subroutine DOTS7 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 DOTS7.
Figure 19.— Concluded.
Figure 20.— Concluded.
3.3.10 SUBROUTINE STCOD7

3.3.10.1 Linkage

STCOD7 is called once by subroutine DOTS7.

3.3.10.2 Interface

STCOD7 interfaces with DOTS7 via COMMON block FNAME (see section 3.3.1.2.2) and passed parameter ST.

3.3.10.3 Input

There is no input to this subroutine.

3.3.10.4 Output

Subroutine STCOD7 has no output.

3.3.10.5 Storage

This subroutine requires 1099 words of storage.

3.3.10.6 Description

STCOD7 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 U.S. Great Plains blind sites for the 1978 Transition Year.

3.3.10.7 Flow Chart

The flow diagram for subroutine STCOD7 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 STCOD7.
3.3.11 SUBROUTINE PRINT7

3.3.11.1 Linkage

PRINT7 is called by CCIT7 via entries PRINT7 and PRINTE, by HEADER via entry PRINTH, by DOTS via entry PRINTD, and by CLUST7 via entry PRINTC. All other called routines are Image Processor system routines.

3.3.11.2 Interface

PRINT7 interfaces with HEADER via COMMON block DOTS (see section 3.3.4.2.1), with CCIT7 via COMMON block FNAME (see section 3.3.1.2.2), with DOTS7 via passed parameters K1 and K2, and with CLUST7 via passed parameter RCNUM.

3.3.11.3 Input

There is no input to this subroutine.

3.3.11.4 Output

PRINT7 prints messages on the line printer.

3.3.11.5 Storage

This subroutine requires 783 words of storage.

3.3.11.6 Description

PRINT7 provides most line printer message output for the CCIT7 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.

3.3.11.7 Flow Chart

The flow diagram for subroutine PRINT7 is given in figure 23.

3.3.11.8 Listing

The program listing for this subroutine is given figure 24.
ENTRY PRINT7

CALL TIME(T)

CALL DATE(D)

PRINT MESSAGE:
'BEGIN RUN OF (NAME OF PROGRAM WITH TIME AND DATE)'

RETURN

ENTRY PRINTH

PRINT MESSAGE:
'THE DPR NUMBER IS ...', WITH [DPRNO(K), K=1,23] AND SEGMENT NUMBER [FILNAM(K), K=12,15]

RETURN

ENTRY PRINTI

PRINT MESSAGE CONTAINING INPUT FILE NAME

RETURN

ENTRY PRINTD(K1,K2)

PRINT MESSAGE WITH K1 AND K2

RETURN

ENTRY PRINTC(RCNUM)

PRINT MESSAGE WITH RCNUM

RETURN

ENTRY PRINTE

CALL TIME(T)

CALL DATE(D)

PRINT MESSAGE WITH TIME AND DATE

RETURN/END

Figure 23.— Flow diagram for subroutine PRINT7.
Figure 24.—Listing for subroutine PRINT7.
4. OPERATIONS

This section presents all the information necessary to obtain proper execution of the CCIT7 processor program.

4.1 OPERATORS GUIDE

This section explains the system hardware configuration and execution (run) setup for the CCIT7.

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 user identification code (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 CCIT7.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] = UIC for the input file
b. Mount the proper disk pack on the drive.

c. Type 'RUN CCIT7'.

d. When message CCIT7-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 CCIT7.DAT.
   - Create CCIT7.DAT with card images, as given in section 4.1.2.1.
   - Run CCIT7.TSK.

4.2 USERS GUIDE

The CCIT7 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 correctly for CCIT's other than those created under LACIE version 7. The approximate dates of valid CCIT's for version 7 are 77305 through the date of issuance of this specification.

4.3 MAINTENANCE DOCUMENTATION

Not applicable.
<table>
<thead>
<tr>
<th>TABLE 2.— BATCH RUN DECK SETUP</th>
</tr>
</thead>
<tbody>
<tr>
<td>$JOB/NAME=AA/MCR/LIMIT=99/ACCOUNT=1106</td>
</tr>
<tr>
<td>$MCR PIP</td>
</tr>
<tr>
<td>CCIT7.DAT;*/DE</td>
</tr>
<tr>
<td>$CREATE CCIT7.DAT</td>
</tr>
<tr>
<td>:</td>
</tr>
<tr>
<td>Card images for file name and SKIP parameter (24A1,I2)</td>
</tr>
<tr>
<td>:</td>
</tr>
<tr>
<td>Blank card</td>
</tr>
<tr>
<td>$EOD</td>
</tr>
<tr>
<td>$MCR REM RSXBAT</td>
</tr>
<tr>
<td>$RUN CCIT7.TSK</td>
</tr>
<tr>
<td>$EOJ</td>
</tr>
</tbody>
</table>
APPENDIX

FORMAT OF .CLO FILE
APPENDIX

FORMAT OF .CLO FILE

Record 1: The first record contains a single integer representing the number of classes detailed in the next record. Sixteen bytes of data follow for each class (minimum of 2 classes, maximum of 26 classes).

Record 2:

<table>
<thead>
<tr>
<th>Byte number</th>
<th>Data description (ASCII)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Class 1 label (W, S, G, N, etc.)</td>
</tr>
<tr>
<td>2-6</td>
<td>Pixel population; PPPPP</td>
</tr>
<tr>
<td>7-9</td>
<td>Uncorrected proportion; M.MM (implied decimal point)</td>
</tr>
<tr>
<td>10-12</td>
<td>Corrected proportion; N.NN (implied decimal point)</td>
</tr>
<tr>
<td>13-16</td>
<td>Variance; .VVVV (implied decimal point)</td>
</tr>
<tr>
<td>17</td>
<td>Class 2 label (W, S, G, N, etc.)</td>
</tr>
<tr>
<td>18-22</td>
<td>Pixel population; PPPPP</td>
</tr>
<tr>
<td>23-25</td>
<td>Uncorrected proportion; M.MM (implied decimal point)</td>
</tr>
<tr>
<td>26-28</td>
<td>Corrected proportion; N.NN (implied decimal point)</td>
</tr>
<tr>
<td>29-32</td>
<td>Variance; .VVVV (implied decimal point)</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
</tr>
</tbody>
</table>

Record 3: This record contains a single integer giving the number of clusters in the classification, CNUM.
Record 4: The fourth record contains 12 bytes of ASCII character data for each cluster; e.g., 12*CNUM bytes of data. The first 6 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 clusters; e.g., DOT103. Only type 1 dots are used to label clusters.