LARS Contract Report 041880

FOREST RESOURCE INFORMATION SYSTEM

Phase III Quarterly Report

for the period

1 January 1980 to 31 March 1980

Prepared for

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Johnson Space Center
Earth Observations Division
Houston, Texas 77058

Contract: NAS 9-15325
Technical Monitor: R. E. Joosten/SF5

Submitted by:

The Laboratory for Applications of Remote Sensing
Purdue University
West Lafayette, Indiana 47906

Principal Investigator: R. P. Mroczynski

LARS Contract Report 041880

FOREST RESOURCE INFORMATION SYSTEM

Phase III Quarterly Report

for the period

1 January 1980 to 31 March 1980

Prepared for

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Johnson Space Center
Earth Observations Division
Houston, Texas 77058

Contract: NAS 9-15325
Technical Monitor: R. E. Joosten/SF5

Submitted by:

The Laboratory for Applications of Remote Sensing
Purdue University
West Lafayette, Indiana 47906

Principal Investigator: R. P. Mroczynski

(E80-10242) FOREST RESOURCE INFORMATION SYSTEM, PHASE 3 Quarterly Report, 1 Jan. - 31 Mar. 1980 (Purdue Univ.) 121 p
HC A06/MF A01 CSCL 02F
N80-30825
Unclas
G3/43 00242

LARS Contract Report 041880

FOREST RESOURCE INFORMATION SYSTEM

Phase III Quarterly Report

for the period

1 January 1980 to 31 March 1980

Prepared for

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Johnson Space Center
Earth Observations Division
Houston, Texas 77058

Contract: NAS 9-15325
Technical Monitor: R. E. Joosten/SF5

Submitted by:

The Laboratory for Applications of Remote Sensing
Purdue University
West Lafayette, Indiana 47906

Principal Investigator: R. P. Mroczynski

INDEX

Star Information Form

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

4. Title and Subtitle

Forest Resource Information System
Phase III Quarterly Report

5. Report Date

18 April 1980

6. Performing Organization Code

7. Author(s)

R.P. Mroczynski, S. Schwingendorf, D. Freeman,
C. Kozlowski, C. Smith, C. Peterson

8. Performing Organization Report No.

041880

9. Performing Organization Name and Address

Laboratory for Applications of Remote Sensing
Purdue University
West Lafayette, IN 47906

10. Work Unit No.

11. Contract or Grant No.

NAS 9-15325

12. Sponsoring Agency Name and Address

NASA/Johnson Space Center
Earth Observation Division
Houston, TX 77058

13. Type of Report and Period Covered

Quarterly
1 Jan 80 to 31 Mar 80

14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

This report covers the fourth quarter of the fifteen-month System Transfer Phase of the Forest Resource Information System Application Pilot Test. The principal Activities during this quarter revolved around transferring software systems, and training St. Regis staff in Landsat analysis procedures. Results of an Applications Pilot Test Project which involved the preparation and classification of two Landsat scenes and the production of a resulting change map are also reported this period.

17. Key Words (Suggested by Author(s))

LARSYS	Technology Transfer
Preprocessing	Remote Terminal
Management	Ratios
Change Map	Classification Evaluations

18. Distribution Statement

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

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

NASA - JSC

FRIS PROJECT OVERVIEW

The Forest Resource Information System Project (FRIS) is a cooperative effort between the National Aeronautics and Space Administration (NASA) and St. Regis Paper Co. (STR). Purdue University's Laboratory for Applications of Remote Sensing (LARS), under contract to NASA, will supply technical support to the project.

FRIS is an Application Pilot Test (APT) Project funded by NASA. The project is interdisciplinary in nature involving expertise from both the public and private sectors. FRIS also represents the first APT to involve a large broad base forest industry (STR) in a cooperative with the government and the academic communities.

Purpose

The goal of FRIS is to demonstrate the feasibility of using computer-aided analysis techniques applied of Landsat Multispectral Scanner Data to broaden and improve the existing STR forest data base, thereby creating the foundation of a dynamic information system. The successful demonstration of this technology during the first half of the project will lead to the establishment by STR of an independently controlled operational forest resource information system in which Landsat data is expected to make a significant contribution. FRIS can be viewed by the user community as a model of NASA's involvement in practical application and effective use of space technology. Additionally, FRIS will serve to demonstrate the capability of Landsat MSS data and machine-assisted analysis technology to private industry by:

- o Determining economic potentials,
- o Providing visibility and documentation, and

o The ability to provide timely information and thus serve management needs.

The ultimate long term successfulness of FRIS can be measured through future development of remote sensing technology within the forest products industry.

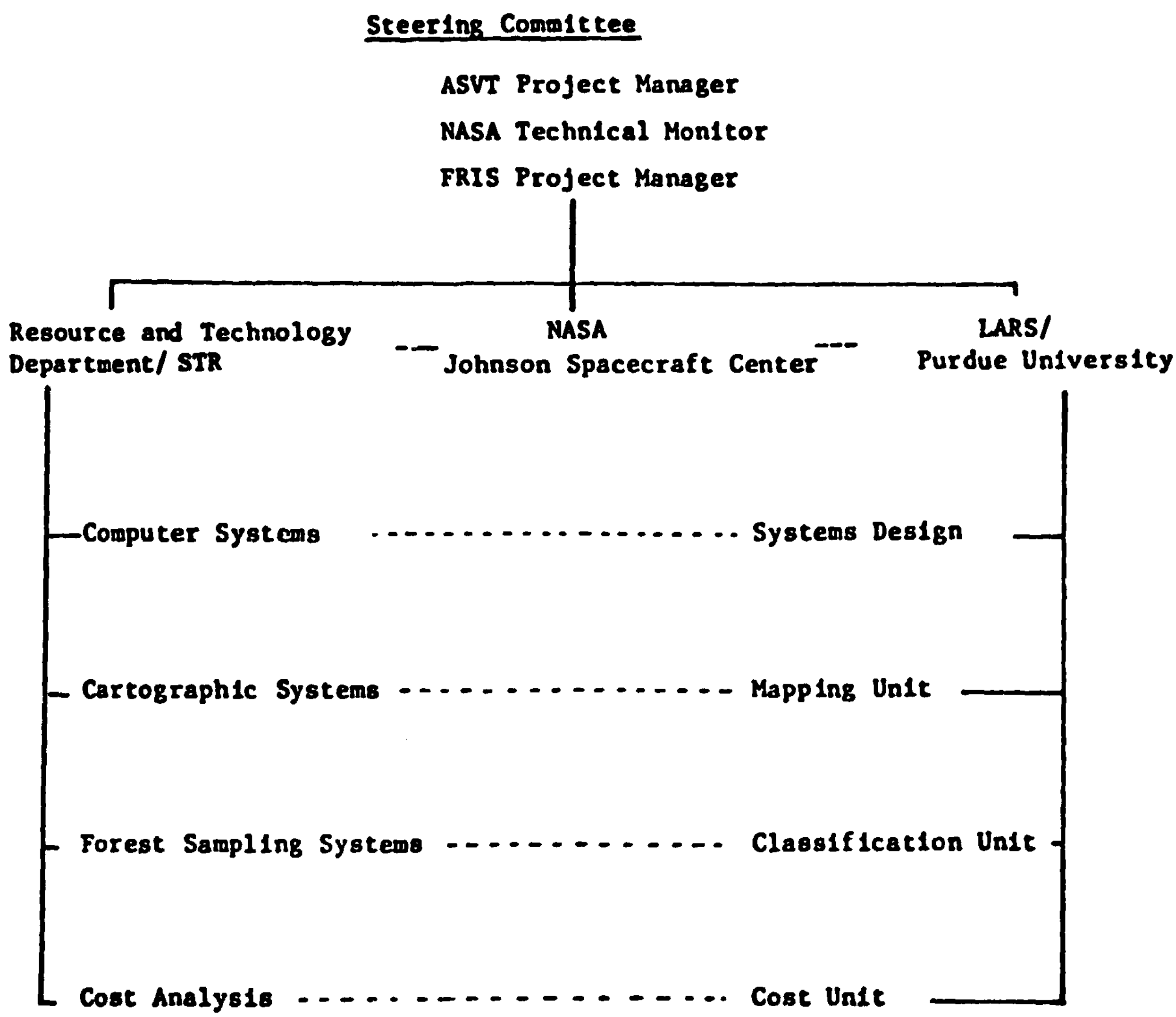
Scope

FRIS is funded as a modular or Phase project with an anticipated duration of three years. The original project concepts were developed in 1973, and a formal project plan was submitted to NASA by STR in 1976. The project officially began in October 1977 after the signing of a cooperative agreement between NASA and STR; and after the compeltion of contractual arrangements with Purdue University.

Organization

The organization of FRIS is depicted in the chart that follows. Since FRIS is a cooperative involving three independent agencies, a steering committee consisting of a project manager from each institution was formed to provide for overall guidance and coordination. Operationally, both STR and LARS have project managers and project staff to insure for the timely completion of activities within the project. The NASA technical coordinator monitors project activities and provides a liaison between the STR and LARS staffs. The solid lines on the chart indicate the flow of management responsibility. The dash lines reflect the technical and scientific inter-changes between operating units.

FRIS Organization



1.0 INTRODUCTION

The material which appears in this report is a reflection of the FRIS Project Staff activities for the period 1 January 1980 to 31 March 1980. This time frame encompasses the fourth quarterly reporting period for Phase III of the Forest Resource Information System (FRIS) Applications Pilot Test (APT). Phase III or the System Transfer Phase of FRIS is directed at meeting the overall Project goal:

> To document and transfer remote sensing technology developed throughout the project that will provide St. Regis with an independent operational system, having Landsat data as a significant and viable contributor.

The most significant Project-wide advance came in late January. At this time the St. Regis Corporate management announced that they had accepted the FRIS concept, and planned to make available financial resources for its support. Corporate acceptance authorized the necessary acquisition of equipment, space, and personnel within the Southern Timberlands Division to make FRIS a viable entity. This milestone also marked that juncture in the APT where St. Regis personnel assumed the greater burden of the lead activity and LARS staff assumed a consulting role.

The unavoidable nine-month delay in the announcement of this positive decision may impact the timely completion of the APT. However, delay in Project completion is not anticipated to be greater than three months, since most System Transfer tasks associated with Phase III were undertaken with the premise that Corporate approval of FRIS was forthcoming.

Noteworthy project accomplishments for this last quarterly reporting period include:

- Successful testing of the Landsat 3 reformatting programs at NCC.
- Transmission of assembler routines to improve the Landsat 3 program operating efficiency.

- o Completion of programming for the SMOOTH and CHANGEDETECTION subroutines.
- o Semi-annual project review at Jacksonville.

The remaining sections of this report describe these activities in more complete detail. Appendix B contains update timeline charts for all tasks.

2.0 TASK AREA ACTIVITIES

2.1 Technology Transfer Task

Technology transfer activities during the past quarter have been directed at system installation and operations. The only formal technology transfer activity scheduled during this period was a photo-interpretation workshop at Jacksonville. The workshop has been postponed because of time constraints imposed on the St. Regis staff associated with the acquisition of FRIS hardware.

With the transfer of two LARS staff to St. Regis during this quarter a smooth transfer of the image processing technology is insured. Although this could be a radical departure from traditional technology transfer activities, it does underscore the committment of the user to remote sensing. Knowledgable individuals within the organization will be more successful in disseminating the remote sensing technology than would an outsider using more classical approaches. Formal classroom sessions, workshops and consulting services are ineffective in answering day-to-day operational problems. We are confident that the new FRIS employees will be the cornerstone of remote sensing technology within St. Regis.

2.2.1 General System Transfer

The ultimate operational version of FRIS will be a relatively complex system composed of three unique subsystems. The three subsystems are:

1) A tabular data base which contains extensive information on forest stand conditions,

2) A Geographic Information Systems (GIS) which contains cartographic, cadastral, and other collateral information, and
3) An image data base which will be able to both process and display information collected by aerospace remote sensors.

FRIS will be unique in that it will combine digital information from all three subsystems in a geographically referenced data base system. The system will allow resource managers to interactively monitor and update conditions of their land base.

The ability to develop, and make a system like FRIS work is inherent in the level of development of the three subsystems. The tabular data base has been in existence and use for more than ten years. This data is resident at the corporate computer facility and accessible via remote terminal at Jacksonville.

The geographic information system consists of vendor provided software and hardware. The GIS will be resident on a PDP 11/70 minicomputer located at the Jacksonville FRIS center. In addition to its use for creating digitized maps and collateral data, it will be the chief means of integrating all forms of digital data. Communications between the mini and the mainframe will provide resource managers with the capability to access the tabular and image data bases from the mainframe. These data will be transferred by land line to the mini where they will be combined with graphic map data. Interaction and analysis will occur on the mini which will have the capability to produce high quality map output.

The core of the image data base will be the LARSYS image processing software developed at Purdue. Although this software was developed to support research and development programs, it was effectively proven during the Demonstration Phase to meet the FRIS objectives. The LARS staff objective during this Phase of FRIS is to assist St. Regis staff implement the software at the corporate computer facility. The remaining subsections of this report addresses the status of this activity.

2.2.2 LARSYS Preprocessing Software Development

LARSYS preprocessing software development is a task which includes three major pieces of software and appropriate documentation. The three processors convert digital Landsat data to LARSYS format, perform systematic geometric corrections of Landsat data and register two images of Landsat data. Software documentation will include listing documentation, traditional program abstracts and a user's guide.

The first preprocessing system of programs converts digital Landsat data to LARSYS format. The process used to complete this program was to develop; a) functional specification, b) design specification, c) implementation plans, and then execution of plans and testing. Functional specification refer to program expectations. Design specifications refer to program execution. The implementation plan documents the who, when and what that relate to tasks required to accomplish the programming of the software. Finally, the actual implementation plan that will be followed, work to be done, and how the results will be tested. Documentation in effect takes place throughout this process. The Landsat processor is complete in that it includes documentation, with the exception of the users guide.

The geometric correction is the second major system of pgorams. Aside from its multiple corrections of Landsat I and II data, this system will be used to rotate the new "P" formatted data to true north. Both functional and design specification have been completed. Implementation will begin during May, 1980. Completion will include most documentation and testing and is expected to be wrapped up during July, 1980.

The last major processor is the image registration system. The primary purpose of this system is to register two coincident digital images as two Landsat digital image data sets. The secondary purpose is to provide for the registration of any known two dimensional grid to another known or defined two dimensional grid. The status of this software is that functional specifications have been completed. Functional specifications that define what the image registration system will be able to do may be found in Appendix A-1. The overall goal of this activity is to produce a maintainable, efficient, system which is modulized, well documented, and easy to use. Both cubic and nearest neighbor interpolation will be available. Locations may be approximated by up to a third order bicubic polynomial.

Design specifications and implementation plans are to be completed by mid-June, 1980. Some implementation has already begun. Cubic interpolation will follow the algorithm described in Appendix A-2. Control cards have been carefully constructed to cover all functional requirements as well as simplicity of use. Other main image registration section elements and design specifications are still being finalized. The multifit least squares analysis section has been initiated. The cross-correlation section will be started in June, 1980.

Documentation is the last major effort of the LARSYS preprocessing software implementation. The three types of documentation will be program listing documentation, program module abstracts, and user documentation. The first type, listing documentation, is most important to system analyst personnel who will maintain these programs. Because of th this key concern for maintainability, a draft of a standard for listing documentation has been generated for this project. Features are that the leading comments in a listing will contain such key information what the program does, what the inputs and outputs are, and all global and local variable descriptions. Legibility of code is specifically emphasized (refer to Appendix A-3).

The total LARSYS preprocessing effort is progressing at a steady pace. The Landsat reformatting is virtually complete while geometric correction is next closest to completion. Image registration will have the most effort applied to it during the last two quarters of the project.

Another important preprocessing transfer activity involves the future potential use of fully corrected, P-format, data available from EDC. The availability of P-type data to FRIS will eliminate much of the front-end preprocessing currently required prior to image classification. The discussion that follows gives preliminary results on the use of fully corrected Landsat 3 data from the Picayune test site in southern Mississippi.

The fully geometrically corrected Landsat MSS frames acquired for the forest resource data base are placed in a specific projection and orientation. This makes possible a one-to-one correspondence between earth coordinates and row column pixel locations in the data. Having such a relationship for each frame will enable resource polygons on maps to be automatically related to row column locations in the data. Visual searching in the imagery would then be unnecessary once corner latitude, longitude, or UTM coordinates were known. A program is being developed to enable user conversion of coordinates and some of the details are included here.

The fully corrected MSS data are placed in a Hotine Oblique Mercator (HOM) projection and in the future they will be placed in the Space Oblique Mercator (SOM) projection. These projections are discussed in Appendix D of the new Landsat User's Handbook. The scale distortions of these projections is very small (1:10,000); thus a linear transformation can accurately be used to relate points in the frame. The earth is divided into zones of latitude and within each zone the corrected frames have a constant azimuth. The zone covering the areas of interest here is zone 2 with latitude range 23° N to 48° N and the zone azimuth is 14.3394993°. The pixel scale of the fully corrected data is 57 meters in both directions.

The software in its present form utilizes a latitude-longitude to Universal Transverse Mercator conversion program to transform user input latitude-longitude coordinates first to UTM. Then a linear conversion is made to line column using the expressions:

LINE = CLINE ÷ DLINE

COL = CCOL + DCOL

DLINE = (-DELEAS . SIN(ALPHA) - DELNOR . COS(ALPHA))/57.

DCOL = (DELEAS . COS(ALPHA) - DELNOR . SIN(ALPHA)//57.

DELEAS = EAST - CEAST

DELNOR = NOR - CNOR

where: CLINE, CCOL are the center line and column of the frame.

CEAST, CNOR is the UTM easting and northing of the center point.

EAST, NOR are the UTM easting and northing of the point to be transformed to LINE, COLUMN.

The conversion program (LOCPNT) is being developed for interactive terminal use and will require typing in the frame center latitude and longitude; then the user enters any number of latitude-longitude points in the frame he wants to convert. Problems are currently being encountered in testing the program on the Picayune frame with inaccurate results. Four test points were taken from the Nicholson and Dead Tiger Creek quadrangles in the Picayune frame and the latitude-longitude coordinates were input and the output line and column were observed. The input parameters are a part of the problem. A latitude and longitude are given as the frame format center; however, it was uncertain what exact line-and-column number corresponded to this. The bias observed at one of the test points was removed and the resulting center line column was taken as the format center. Thus, there is no error at this point. At the other three points, errors were observed. The results are listed in Table 2.2.2.1 .

TABLE 2.2.2.1 Coordinate Conversation Tests for Picayune Frame. Format Center: 30.18° N., 89.52° W. Center Line, Col: 1518,1796.

Test Point				Estimated Point		Error	
Lat.	Long.	Line	Col.	Line	Col.	Line	Col.
30.375	89.625	1189	1518	1189	1518	0	0
30.5	89.75	987	1265	999	1285	12	-7
30.375	89.5	1150	1725	1141	1724	-9	-1
30.5	89.625	948	1473	952	1463	4	-10

Causes for these errors are being investigated.

2.2.3 LARSYS Transfer

The LARSYS transfer task involves only those processors associated with image classification. The bulk of this software was transferred during the previous two quarters. The software transferred contains elements of LARSYS ver. 3.1 and LARSYSDV. LARSYSDV includes new programs which are under development and not part of the ver. 3.1 documentation. The programs were transferred in card image format on 9-track computer compatible tapes. Copies of tape listings and user documentation were also provided.

The tasks facing St. Regis personnel are to convert the programs which currently run on an IBM 3031 operating under VM to an IBM 3033 operating under MVS. That is the LARS computer operates as a virtual machine while the St. Regis computer operates as a batch machine. This means that the LARSYS programs are not directly compatible between the two IBM machines.

St. Regis staff are required to make certain changes to the LARSYS software. A summary of the necessary changes to the software appears below:

A. Add the function COPYTAP, this allows the data to be read from tape to disk and stored on disk. St. Regis has a disk based system while LARS is tape based.

B. Replace command language with ROSCOE. Due to the operating system differences between the two machines, the command language has to be modified. ROSCOE is a software package that permits St. Regis users to initiate jobs from remote sites. This will replace CMS currently used in LARSYS to perform similar functions.

C. LARSYS contains some bookkeeping routines that will be deleted because these functions are already handled by St.rRegis.

D. All non-standard file handling routines in LARSYS will be replaced to meet St. Regis computer software conventions.

E. All tape handling routines will be modified to deal with disks.

F. Machine dependent assembly language routines will be eliminated where feasible.

Implementation tasks facing St. Regis staff include:

1. Program compilation from tape.
2. Creation of disk files.
3. Modification of software for compatibility to St. Regis machine, including elimination of bookkeeping, assembler routines and modification of tape callable routines.
4. Creation of ROSCOE modules,
5. Development of links to GIS, and
6. Develop St. Regis/LARS user documentation.

LARS staff are available for consultation and debugging as needed. Our experience during this past quarter was that very little assistance was requested by St. Regis personnel. Implementation of these software are progressing with very few problems. This is most likely due to, a) the level of documentation provided with the LARSYS software, and b) the knowledge of the staff involved with the implementation.

2.2.4 Programming Additions

The LARSYS software packages were originally designed to process digital multispectral scanner data in a research environment. Periodically, modifications and embellishments have been added to LARSYS support packages to improve interaction with the human component of the analysis activity. Since FRIS is a user oriented, operational system there were certain additions required to improve user efficiency. There have been a number of additions to the LARSYS software since the midpoint of Phase II. The two newest additions reported this quarter are significant because they directly affect FRIS requirements. The two new program additions are SMOOTHRESULTS and CHANGEDETECTION.

SMOOTHRESULTS is a post classification processor designed to emulate the human action of creating a mapping cell. Mapping cells are the basic component of timber type or operating area maps. The theory behind the mapping cell is simply that areas less than a minimum size, say five acres, are ignored for map drawing purposes and included as part of a larger population. Therefore, a two or three acre inclusion in a type would be ignored when the map is created.

The human quickly handled these small inclusions when making a type map. A Landsat classification, however, will display most inclusions that fall within the scanner resolution. These will result in a salt and pepper effect on classification output. A situation that may accurately portray the cover composition uut which is often not appealing to land managers who are used to working with "clean" (no salt and pepper) maps.

SMOOTHRESULTS allows the analyst to define a mapping unit and produce a classification results map which does not exhibit a salt and pepper pattern. The processor scans a LARSYS Classification Results File and replaces groups of classified points (cells) with the dominant class from that group. The analyst has the option to specify the size of the cell (CELLSIZE card), class numbers which are to be replaced (PRIORITY card) and weighting factors for each class (WEIGHTS card). The output from this function is to tape or disk in LARSYS Classification Results File format. Figure 2.2.4.1 is an example of a classification result which shows output both before and after use of the SMOOTHRESULTS processor.

SMOOTHRESULTS was debugged and tested during this quarter. An additional option which allows the analyst to define new classes which are mixtures of old classes was developed and is being tested. The control card reference file and program abstracts are included in Appendix A-4.

The other processor that was upgraded for addition to the FRIS package of software which is being transferred to St. Regis is CHANGEDETECTION. This is another post classification processor, and is designed to make comparisons between classification results. This processor is intended to be used to compare two anniversary Landsat classifications which have similar class structures. The resulting product of this comparison is a LARSYS results tape containing "change" classes.

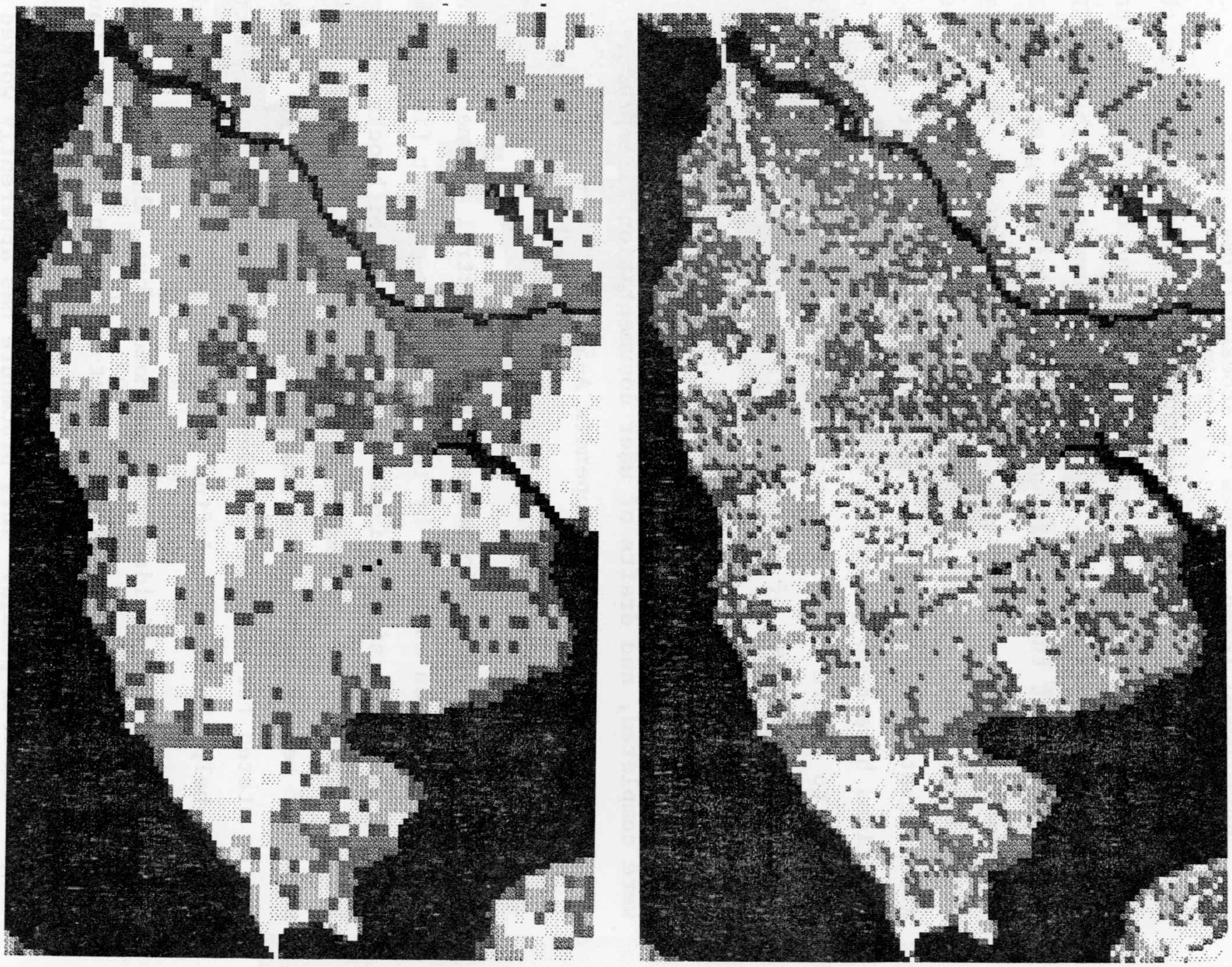


Figure 2.2.4.1 Example of classification output both before and after SMOOTHRESULTS processor implementation.

Change classes are designated by the analyst and are in the form where:

Pine (time 1) goes to Non-Pine (time 2), and
Non-stocked (time 1) goes to Stocked (time 2).

Optimally, Landsat data is of an anniversary nature, that is the data of collection for both dates is nearly coincident but chronologically a year or more apart in time. Present requirements of the CHANGEDETECTION program are that the Landsat scenes be precision registered. Independent classifications are generated for time 1 and time 2. The analyst is careful to insure that class structure, that is the various spectral groups that comprise the information classes is similar. Once the classifications have been generated, CHANGEDETECTION is ran, and an output similar to figure 2.2.4.2 is produced. Tabular information which indicates the amount of change in acres percent of area by class can also be produced.

During this past quarter, program abstracts for CHANGEDETECTION were completed, and drafts of user documentation were prepared. Copies of this material appear in Appendix A-5.

2.3 Classification Evaluation Review

Traditionally, performance of multispectral scanner data classifications have been assessed by the evaluation of test fields. Test fields vary in size from single pixels to multiple pixel blocks which are located randomly or systematically throughout the classification. The number of test fields can be statistically determined so that results can be stated for a given confidence range. A priori information is used to help the analyst define the number of sample test fields needed.

Ideally, test fields should be homogenous, that is they should represent a 'pure' cover type. Recent trends in evaluating Landsat classifications have been toward selecting single pixels or blocks no larger than four pixels. This procedure has apparently worked well when agricultural or general land use classifications are being evaluated. In fact, one may go so far as to assume that the quality of the ground

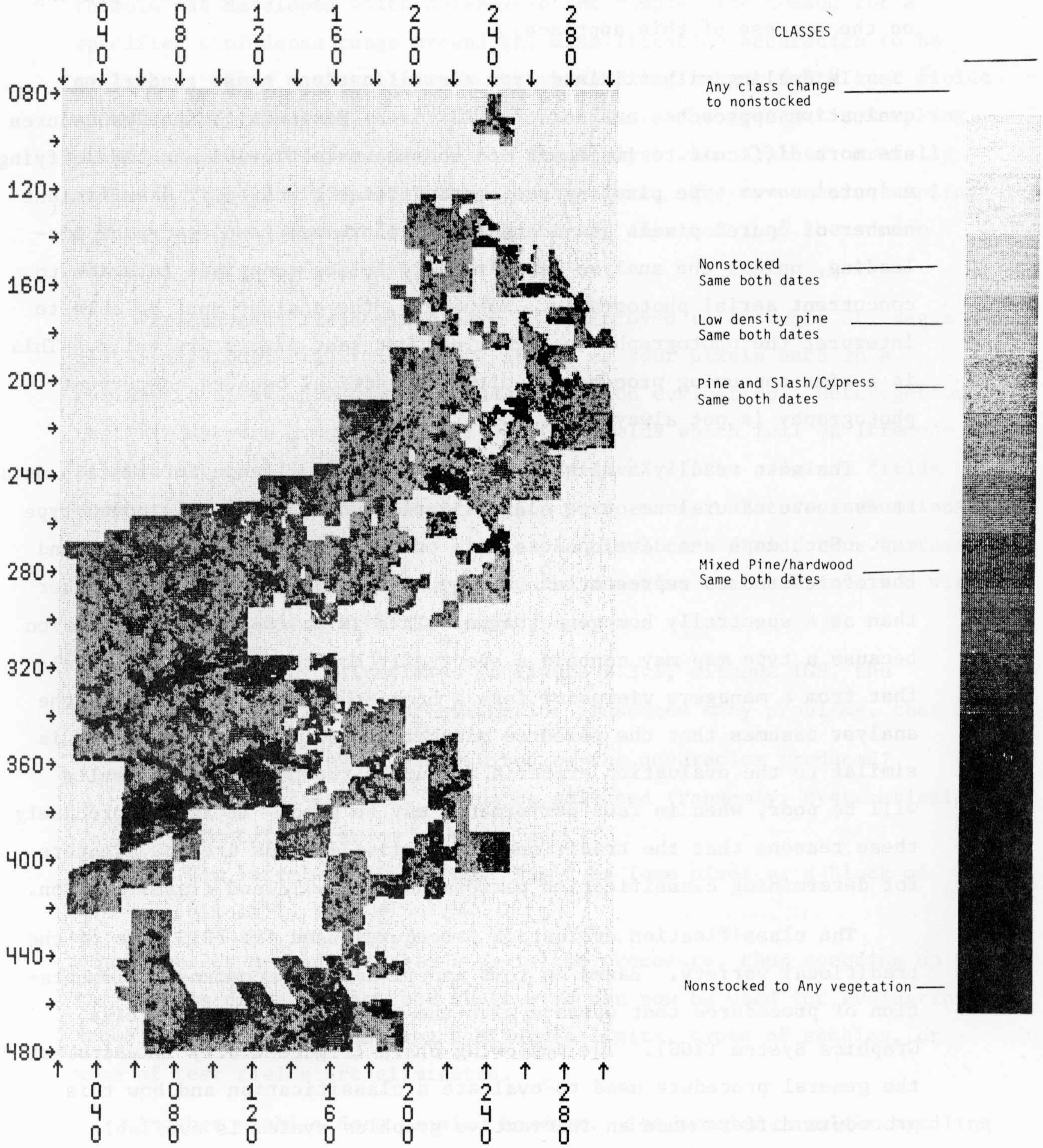


Figure 2.2.4.2 Output of CHANGEDETECTION program; classes shown are change classes.

reference data for these types of classifications has a direct bearing on the success of this approach.

In dealing with wildland type classifications these traditional evaluation approaches are not as effective. Minimally, these procedures are more difficult to implement and consume more time because identifying a 'pure' cover type pixel becomes more difficult. Even if a sufficient number of 'pure' pixels are identified performance results can be misleading, unless the analyst can carefully relate the pixel location to concurrent aerial photography. Naturally, the analyst must be able to interpret the photography to determine what test pixels are valid. This is a time consuming process and often impractical because concurrent photography is not always available.

The most readily available form of ground reference information used to evaluate natural resource classification performance is a cover type map. Such maps are developed as aids to natural resource managers and therefore tend to represent a cover type as a management entity rather than as a spectrally homogenous area. This is an important distinction because a type map may contain a spectrally heterogenous mix of pixels that from a managers viewpoint form a homogenous mapping unit. If the analyst assumes that the resource managers definition of homogenous is similar to the evaluation criteria of purity the performance results will be poor, when in fact performance may be good. It is for precisely these reasons that the traditional evaluation methods are not adequate for determining classification performance for wildland classification.

The classification evaluation procedures used for FRIS were of the traditional variety. Based on this experience we recommend implementation of procedures that operate with the benefit of an Interactive Graphics System (IGS). The following chart (Figure 2.3.1) illustrates the general procedure used to evaluate a classification and how this procedure differs when an interactive graphics system is available.

Previous Methodologies

Various strategies have been employed to sample St. Regis Ownerships for classification accuracy evaluations. During the FRIS demonstration a formula was developed which determined the sample size needed for a specified confidence range around the classification accuracies to be evaluated. Then a systematic or random sampling scheme (with test fields of one pixel) was applied to three test sites. The systematic sampling method was the preferred method for the analyst when applied carefully and with full knowledge of its cyclical nature. The systematic sampling method was precise and had less human error involved than the random sampling approach.

Various test field sizes have been employed to evaluate St. Regis classifications. Initially, test fields of four pixels each in a systematic array were used for classification evaluation. Heterogeneous test fields were dropped as well as test fields which fell on irreconcilable map boundaries. Ultimately only 42.7% of the original fields were utilized to assess classification performance. Future evaluations utilized a single rather than a multiple pixel test field, thus increasing the number of samples and decreasing the man-time that was involved with the multiple pixel systematic sample.

Previously, as illustrated in Figure 2.3.1, without IGS, the selection of test fields for evaluation presented many problems, that is:

1) What confidence was required on the accuracies produced?
2) How would the test fields be selected (randomly, systematically, stratified random samples, etc.)?
3) How large would each test field be (one pixel or a block of pixels)?

Since the IGS mechanises this step in the procedure, thus speeding up the process dramatically, the whole area can now be used for evaluation. Hence any decisions involving confidence limits, types of samples, or size of test fields are eliminated.

Another problem does remain, however. The problem involves handling border pixels in classification evaluations.

The rest of the steps involved in evaluating classifications can all be mechanised within the interactive graphics system employed at FRIS.

Evaluation Procedure without IGS	Evaluation Procedure with IGS
(confidence required) => determine number of test fields; select test fields a) randomly b) systematically	all fields can be included automatically
*PRINTRESULTS map test fields	not necessary
OVERLAY test fields	can be automatically done
RECORD Landsat and Inventory categories; Classification Accuracy and confusion table	automatically recorded and tallied for classification accuracy

Figure 2.3.1 Comparison of Classification Evaluation Procedures with and without an Interactive Graphics System (IGS).

Thus, many man hours and the resulting human errors can be eliminated.

As stated earlier and illustrated in Figure 2.3.1 the IGS improves classification evaluations in two key areas:

- o The whole area or a very large sample of the area can be utilized for evaluation.
- o The system can be virtually, completely, atuomatic, eliminating many man-hours of work.

With these improvements come new and in some respects different problems.

First of all, since polygons, not single pixels or blocks of pixels, will be used for evaluation, criteria must be developed for labeling an heterogeneous polygon. The proposed criteria for FRIS is:

- o 75% or more pixels pine => "pine" polygon
- o 75% or more pixels hardwood => "hardwood" polygon
- o otherwise the polygon is labeled as "other" or "nonstocked"

These criteria may change as the spectral characteristics of these cover types are studied further and/or more classes or subtypes become separable.

Another problem involves the polygons' border pixels and how they are to be treated with respect to the classification evaluation. In other words, if a pixel lies on the boundary between two or more polygons (Figure 2.3.2) of what polygon should it be considered a member, or should it be considered a member of any polygon? There are three possible solutions for handling this problem:

1) Include all border pixels in whichever polygon the IGS overlay has assigned them.
2) Exclude all border pixels from the analyses.
3) Exclude only some border pixels from the evaluation.

The first solution, including all border pixels, would give a complete, unbiased evaluation of the area. In this case, however, any small error in registration could substantially effect the evaluation, causing border pixels to be assigned to the incorrect polygon. Also, this method

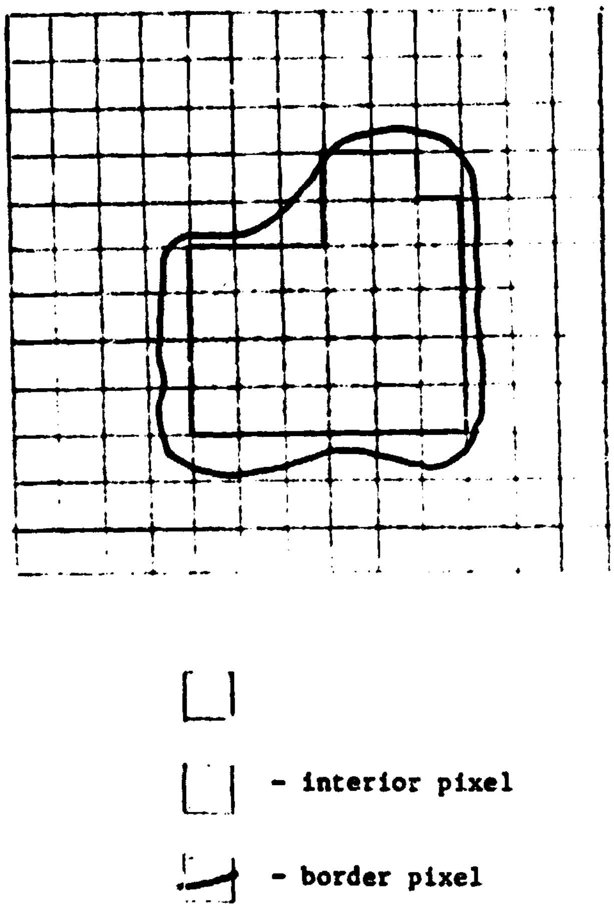


Figure 2.3.2 Overlayed area illustrating border pixels and interior pixels.

assumes that spectrally a pixel should fall in the class which covers the majority of the pixel. In fact, however, this is not necessarily the case. This method would be extremely fast and require very little programming and hence could serve as a first cut at evaluating a classification.

The second solution, excluding all border pixels, would give an incomplete, (not all pixels included), biased evaluation of the area. If, however, the assumption can be made that the border pixels of any polygon have the same distribution of classes as the internal pixels, then this solution becomes unbiased. Also, this method reduces the amount of registration and mapping error measured in the evaluation. Some programming, supplementing the IGS will be necessary to implement this solution.

The third solution is essentially the same as the second solution except that those pixels which are on a border between two spectrally undifferentiable polygon types would be included in the evaluation. That is, if a pixel lies on the boundary between two pine plantations which differ by a spectrally unmeasurable criterion (i.e., 2 years age difference), that pixel would be included in the evaluation. This solution has the same criticisms as the second solution (excluding all the border pixels) except that more of the area is being evaluated thus giving a somewhat better total evaluation. With this solution, however, the assumption that the remaining border pixels of any polygon reflect the distribution of those included, may not be as reasonable.

Another problem involved with using an interactive graphic system for classification evaluation concerns the size of the polygons evaluated. If a small number of pixels (e.g., 1, 2 or 3) are used to assign a polygon's class, the assigned class could easily be in error. Thus a criterion, such as 10 internal pixels, should be developed, indicating the smallest polygon to be evaluated.

The interactive graphics system which FRIS will employ for classification evaluation will facilitate and improve this procedure greatly.

The new procedure, once it is implemented, will become virtually automatic. Many previous sampling considersations will become unnecessary due to the graphics and overlay capabilities of the new system.

2.4 MANAGEMENT

The FRIS management activity oversees day-to-day project operations and is responsible for all technical and fiscal project reports. Status of all major System Transfer Tasks are shown in Exhibits 1 through 4 of Appendix B. A brief discussion of these status charts follows.

Technology Transfer Task

All computer-aided analysis workshops and short courses have been successfully completed. Activity A3, the photo-interpretation short course has been extended twice and in all probability will be eliminated as a Phase III activity. Time constraints on St. Regis staff involved with the installation of FRIS systems prohibit selection of a time during the remainder of Phase III when this course can be given. The possibility of scheduling this activity during Phase IV will be investigated.

Consultation (activity B) associated with LARS software implementation will be continued through the end of Phase III. Onforseen delays associated with St. Regis' acquisition and installation of vendor hardware and software at Jacksonville are the reasons for this shift.

User documentation, especially activity C2, is approximately 66% complete. Part of this activity will carry over to Phase IV.

Remote Terminal activities will cease on 30 June as projected.

LARSYS Transfer Task

The first two activities of this Task, Planning and Transfer are complete. Activity C, Consultation and Debugging is extended to 30 June. At this time all LARSYS processors should be operational at NCC. St. Regis staff will be primarily responsible for debugging activities with LARS staff assisting as required.

Documentation (activity D) will be extended into Phase IV. Supplemental timeline charts to Exhibit 2 in Appendix B which will identify specific documentation tasks are included.

Activity E (Test and Evaluation) is anticipated to occur during the last quarter of Phase III. St. Regis staff will be primarily responsible for this activity.

Preprocessing Transfer Task

The Planning and most of the Program Refinement activities (B1 and B2) are complete. Activity B3, Image Registration, will carry over to Phase IV, as will that portion of activity C, Program Transfer, and activity D, Consultation and Debugging. The image registration software will be transferred early during Phase IV.

Documentation, activity E, is well advanced for the reformatting and geometric correction software. The image registration documentation is being prepared in parallel with the software programming. Because of this, documentation will be completed during Phase IV.

Test and Evaluation, activity F, is dependent on St. Regis staff and their use of the systems. The activity will not be entirely completed until transfer and implementation of the image registration software.

Support, activities G, will be completed by the end of Phase III, with the possible exception of G3. Reformatting Operations Procedures are closely associated with communications capabilities between the mainframe and the mini. LARS staff will assist St. Regis personnel in a consulting role as needed to support this activity.

Management Task

Management activities are up to date as indicated by the status chart. Noteworthy among these activities is a new task being developed under Information Dissemination. Specifically, we are developing a FRIS color brochure for wide dissemination by the three agencies. Content and format of this product have been discussed and a cost proposal to cover this work is being prepared.

Appendix A-1

Image Registration Functional Specifications

An image registration capability has been determined to be a necessary part of the FRIS III image preprocessing software. Image registration is general enough to mean grid to grid transformation. Thus, while the system is designed to register two coincident Landsat scenes, registration to alternate grid systems may be accomplished with this software as well. Functional specifications will be as follows:

I. Purpose

A. Primary: Registration of two coincident digital images as two Landsat digital image data sets.

B. Secondary: Provide for the registration of any known two-dimensional grid to another known or defined two-dimensional grid.

II. Input images are assumed to be in LARSYS format.

III. Checkpoint Acquisition

A. Manual checkpoint acquisition is possible.

B. Cross-correlation of two coincident digital images may be accomplished by implementation of a numerical integration image correlator.

C. Control may be by set line and column intervals.

D. Alternate control will be from a set of inputted control correlation point locations where a cross correlation is desired, i.e., arbitrary point by point correlation.

IV. Registration transformation

A. Coefficient determination will be calculated for affine, biquad, and bicubic transformation.

B. Transformations through bicubic will be implemented for the registration transformation.

C. Block registration technique will be utilized.
   1. Optimum rectangular block size will be determined for biquadratic and bicubic registrations.

D. Interiors of all blocks will be registered with an affine or linear transformation.

V. Radiometric interpolation

A. Nearest neighbor will be the default.

B. Cubic interpolation will be optimally implemented.

VI. Output images will be produced in LARSYS format.

Appendix A-2

Cubic Interpolation Used in the Image Registration System

The algorithm used in the current image registration system for cubic interpolation of data values is based on a thrid order Lagrange interpolation. The general Langrangian interpolating polynomial for three dimensions is:

$$P_{mn}(X,Y) = \sum_{i=0}^{m} \sum_{j=0}^{n} L_i(X) L_j(Y) f(X_i Y_j)$$

where

$$L_i(X) = \prod_{\substack{k=0 \\ k \neq i}}^{m} \frac{X-X_k}{X_i-X_k} \qquad i = 0,\ldots,m$$

and

$$L_j(Y) = \prod_{\substack{\ell=0 \\ \ell \neq j}}^{n} \frac{Y-Y_\ell}{Y_j-Y_\ell} \qquad j = 0,\ldots,n$$

The image registration system uses the above equations with m = 3, n = 3. Therefore, we need m+1=4 different X_i values and n+1=4 different Y_i values. The X_i's and Y_i's used are 0,1,2,3 and 0,1,2,3. Then the general equation reduces to:

$$\begin{aligned} P_{33}(X,Y) = {} & L_0(X)L_0(Y)f(0,0)+L_1(X)L_0(Y)f(1,0)+ \\ & \quad L_2(X)L_0(Y)f(2,0)+L_3(X)L_0(Y)f(3,0)+ \\ & L_0(X)L_1(Y)f(0,1)+L_1(X)L_1(Y)f(1,1)+ \\ & \quad L_2(X)L_1(Y)f(2,1)+L_3(X)L_1(Y)f(3,1)+ \\ & L_0(X)L_2(Y)f(0,2)+L_1(X)L_2(Y)f(1,2)+ \\ & \quad L_2(X)L_2(Y)f(2,2)+L_3(X)L_2(Y)f(3,2)+ \\ & L_0(X)L_3(Y)f(0,3)+L_1(X)L_3(Y)f(1,3)+ \\ & \quad L_2(X)L_3(Y)f(2,3)+L_3(X)L_3(Y)f(3,3) \end{aligned}$$

where:

$$L_0(X) = \frac{(X-1)(X-2)(X-3)}{(0-1)(0-2)(0-3)} = \frac{X^3 - 6X^2 + 11X - 6}{-6}$$

$$L_1(X) = \frac{(X-0)(X-2)(X-3)}{(1-0)(1-2)(1-3)} = \frac{X^3 - 5X^2 + 6X}{2}$$

$$L_2(X) = \frac{(X-0)(X-1)(X-3)}{(2-0)(2-1)(2-3)} = \frac{X^2 - 4X^2 + 3X}{-2}$$

$$L_3(X) = \frac{(X-0)(X-1)(X-2)}{(3-0)(3-1)(3-2)} = \frac{X^3 - 3X^2 + 2X}{6}$$

and $L_j(Y)$'s have the same equations with Y substituted for X

and f(X,Y) is the data value associated with pixel (X,Y).

To save computation time, the L_i's are calcualted according to the above equations for specific points in the (X,Y) grid. These points were chosen at quarter pixel intervals as shown in figure 1. The calucalted $L_i(X)$'s are;

	$L_0(X)$	$L_1(X)$	$L_2(X)$	$L_3(X)$
X = 1.00	0.0	1.0	0.0	0.0
X = 1.25	-0.0546875	0.8203125	0.2734375	-0.0390625
X = 1.50	-0.0625	0.5625	0.5625	-0.0625
X = 1.75	-0.0390625	0.2734375	0.8203125	-0.0546875
X = 2.00	0.0	0.0	1.0	0.0

The same table applies for Y=1.00, 1,25,1.50, 1.75, 2.00.

In the image registration process, an input point A (see Figure 1) is approximated to its nearest quarter pixel. To calculate the data value associated with A, the Lagrange polynomial coefficients for that quarter pixel location are used in the $P_{33}(X,Y)$ equation. To further save on computation, the products $L_i(X)L_j(Y)$ for all combinations of the quarter pixel locations ((1.0, 1.0), (1.25,1.0), (1.50,1.0), (1.75,1.0), (2.0,1.0), (1.0,1.25), (1.25,1.25), etc.) have been stored in a table. Then when $P_{33}(X,Y)$ is calculated, a table lookup locates the appropriate $L_i(X)L_j(X)$'s.

When this algorithm was implemented for cubic interpolation of data values, it was determined that the error introduced by this method of using discrete intervals versus continuous intervals was negligible. It was negligible because the intervals involved were quarter pixels and the final data values were integer values between 0 and 255.

References:

"Multitemporal Image Registrations of Multispectral LANDSAT Data of Finney and Ellis Co.'s, Kansas by Nearest-Neighbor and Third Order Lagrangian Interpolation Methods." Prepared by Charles R. Smith, LARS, September 20, 1976.

Source listing of OVERLA subroutine used in current Image Registration System.

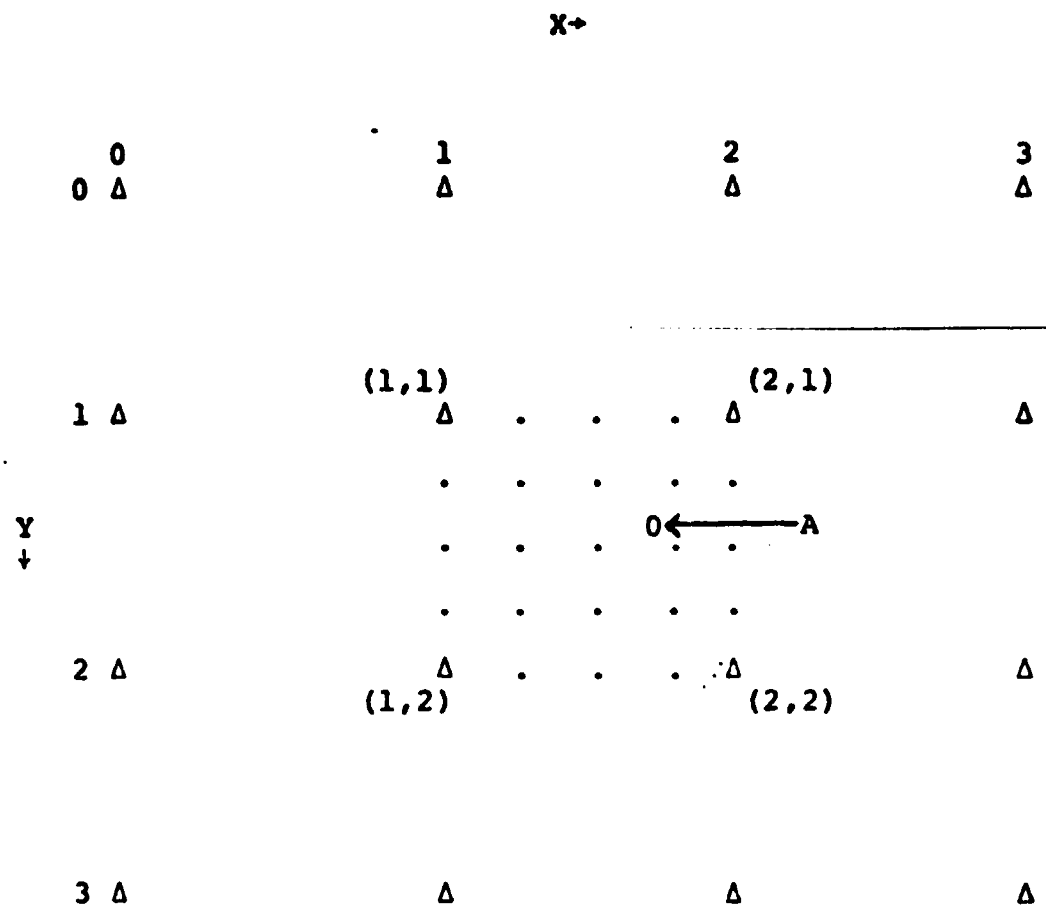


Figure 1. 4 x 4 Data matrix surrounding point to be interpolated (point A). Example: Since point A is nearest grid coordinates (1.5, 1.75), the Lagrange coefficients for this x and y are taken from the table and used in the interpolating polynomial.

Appendix A-3

Reformatting Documentation Standards

Preface

This guide supplements the LARSYS Standards Report Section III Programming Standards. Programmers writing software for the Reformatting group should read the LARSYS report as well as this guide; wherever this guide conflicts with the LARSYS report, this guide should be followed. Programmers should take particular note of the paragraphs in the LARSYS Standards Report Section III on Assembler and EXEC organizations and comments, and on programming techniques.

The main emphasis of the guide is on the documentation of program source code. Program logic must flow downward, and comments must reflect that flow. Within the source code, all global and local variables must be identified in variable description lists. The source code also must contain a general description of the algorithm used and input/output requirements. Specific coding and commenting practices are recommended for improving the legibility of source code.

This guide contains the following information.

I. DOCUMENTATION OUTSIDE OF SOURCE CODE LISTING

A. Any program with a control card reader must have a separate description of its control cards. The description must include all keywords and all parameters with an indication of which keywords and parameters are required and which are optional. All default values must be indicated. It is also useful to include one or two sample control card decks. For an example of a control card description, see Appendix A.

B. Any program designed for use by non-reformatting staff should have a user's guide. This guide should include several example user sessions.

C. Any program using routines that depend on non-trivial algorithms, calculations, or data structures must have an abstract. The abstract may be for an entire system or for specific subroutines. The abstract must describe the algorithms, calculations, and/or data structures in sufficient detail for a person unfamiliar with the source code to understand the implementation. For a major program, it may be appropriate to have two levels of documentation abstracts. One abstract would be directed at the interested user, and the other at the programmer responsible for program maintenance. For an example of a program abstract, see Appendix B.

II. DOCUMENTATION WITHIN THE SOURCE CODE LISTINGS

A. Overall System Standards

1. Each Routine must flow logically downward. See Appendix C for examples of routines that flow logically downward.

2. The names of all routines for a specific software system must have the same three-letter prefix. The last three letters should be unique for each routine and represent the main function of the routine. See the example below.

```
MEAD   - main routine for processing a MEAD product.
MEACC  - read MEAN control cards.
MEAINT - initialize MEAD variables and common
         blocks.
MEAMTX - set up MEAD scaling matrix
MEATRA - translate one line of input values into
         one line of output values.
```

Example 1

3. Use variables for constants. In the example below, constants such as Fortran unit numbers and the buffer sizes are declared as variables. Such a convention facilitates program maintenance and revision.

```
C*****************************************************************************
C
C         LOCAL VARIABLES
C
C
      REAL * 4 TIME
C
      INTEGER * 4 BLANK/'    '/,      F1UNT/21/,   F2UNT/22/,   F3UNT /23/,
     1            HEX3F /Z3F/,        HEXFF /ZFF/, INBCNT /1008/,
     2            INUNT/12/,          MAXCHN/3/,   MAXIN/500/,
     3            MAXLC/10000/,       NO/'NC  '/,  CUTID(200),   CUTLNT/11/,
     4            TRK7 /'7TRK'/
C
      LOGICAL * 4 IDFLG
C
      INTEGER * 2 LARDAT(5000),    ROLL /Z7FFF/
C
      LOGICAL * 1 INBUF(1008),     ZERO/Z00/
C
C*****************************************************************************
C
C             LOCAL VARIABLE DESCRIPTIONS
C     BLANK   THE CONSTANT BLANK.
C     F1UNT   DISK UNIT WHERE FIRST TAPE FILE IS TRANSFERRED.
C     F2UNT   DISK UNIT WHERE SECOND TAPE FILE IS TRANSFERRED.
C     F3UNT   DISK UNIT WHERE THRID TAPE FILE IS TRANSFERRED.
C     HEXFLG  EQUALS THE ID FLAG FOR THE INPUT TAPE (DEPENDS ON
C             FORMAT OF INPUT TAPE).
C     HEX3F   CONSTANT EQUAL TO 3F HEXIDECIMAL.  AN INPUT RECORD IS
C             AN ID RECORD IF THE FIRST BYTE EQUALS HEX 3F (7 TRACK FORMAT)
C     HEXFF   CONSTANT EQUAL TO FF HEXDECIMAL.  AN INPUT RECORD IS
C             AN ID RECORD IF THE FIRST BYTE EQUALS HEX FF (9TRACK FORMAT)
C     INBCNT  NUMBER OF BYTES IN AN INPUT RECORD.
C     INUNT   UNIT NUMBER OF INPUT TAPE.
C     MAXCHN  MAXIMUM NUMBER OF DATA CHANNELS THIS ROUTINE CAN HANDLE
C     MAXIN   MAXIMUM NUMBER OF DATA VALUES IN ONE INPUT RECORD.
C     MAXLC   MAXIMUM NUMBER OF BYTES ALLOWED IN ONE LINE OF LARSYS DATA.
C     OUTUNT  UNIT NUMBER FOR OUTPUT TAPE.
C     NC      CONSTANT EQUAL TO 'NC'.
C     TRK7    CONSTANT EQUAL TO '7TRK'.
C     ZERO    CONSTANT BYTE EQUAL TO 00 HEXIDECIMAL.
C
```

Example 2

In the above example, local variable descriptions have been provided only for the "constant" variables. See the example software system in Appendix C for descriptions of all local variables.

4. Block commons must be named and they must have variables listed in the order:

REAL * 8
REAL * 4
INTEGER * 4
LOGICAL * 4
INTEGER * 2
LOGICAL * 1

Within each type of variable, the variables must be listed alphabetically. Large common blocks must be spaced for legibility.

```
C
C     VARIABLE NAMES FOR AGRONOMIC ID AND FOR SOILS IS
C
      COMMON /IDNAME/  AITE,      AZIR,      AZVI,      HAPR,      BRLE
      COMMON /IDNAME/  CARU,      CATN,      CLCO,      CLTY(4),   COB2
      COMMON /IDNAME/  COMM(37),DACO,        DADA,      DAPL,      CBBL
      COMMON /IDNAME/  DBFR,      DBGL
      COMMON /IDNAME/  DBST,      DBTO,      DBWE,      DBYL,      DEEO
      COMMON /IDNAME/  DENA(2),   DERA,      DIGR,      DIIN,      DQF1(2)
      COMMON /IDNAME/  DQF2(2),   DQF3(2),   DQF4(2),   DQF5(2),   DQF6(2)
      COMMON /IDNAME/  DQF7(2),   DRCL,      EPC1,      EP02,      EP03
      COMMON /IDNAME/  EP04,      EP05,      EPC6,      EP07,      EP08
      COMMON /IDNAME/  EP09,      EP10,      EXNA(4),   EXNU,      FANA(4)
      COMMON /IDNAME/  FIAC,      FINU,      FIVI,      FLLI(2),   FOCA
      COMMON /IDNAME/  FRBI,      FRCO,      GMCS,      GRLE,      HAWI
      COMMON /XDNAME/  HEIG,      HERF,      HISC,      HORI(2),   ILLU(2)
      COMMON /IDNAME/  ININ,      INNA(4),   INST,      JUDA
      COMMON /IDNAME/  LAID,      LEAR,      LEPL,      LF(6),     LF7
      COMMON /IDNAME/  LF8,       LOCA(4),   LODA,      LOLA(2),   LOLO(2)
      COMMON /IDNAME/  LOSO,      MATU(4),   MODA,      MOFI(4),   MOLA
      COMMON /IDNAME/  MOST,      MUCO(4),   NMAT,      NUDE,      NUSA
      COMMON /IDNAME/  NUSC,      OBNU,      OTST,      PECL,      PEGR
      COMMON /IDNAME/  PESA,      PESI,      PHFR(2),   PHRO,      PHSE(4)
      COMMON /IDNAME/  PLCO,      PLDA,      PLMC,      PLNU,      PMOW
      COMMON /IDNAME/  PRIN(4),   RATE
      COMMON /IDNAME/  RECA,      REDA,      REHU,      RENU,      RCDI
      COMMON /IDNAME/  ROWI,      RUSE,      SAGR,      SCRA,      SCTY(4)
      COMMON /IDNAME/  SENA(4),   SENU,      SPEC(4),   STCO(10),SUCO(4)
      COMMON /IDNAME/  TALE,      TATE,      TAWI,      TCR1,      TCR2
      COMMON /IDNAME/  TEXT(4),   TIDA,      TSWT,      UNCA,      VARI(4)
      COMMON /IDNAME/  VI
      COMMON /IDNAME/  WABA(4),   WBTE,      WEEL,      WIDI,      WISP
      COMMON /IDNAME/  YEDA,      YELD,      YELE,      ZEIR,      ZEVI
C
C     VARIABLE NAMES EXCLUSIVELY FOR SOILS ID
C
      COMMON /IDNAME/  ACTI,      ALUM,      ASHO(2),   AVPH,      AVPC
      COMMON /IDNAME/  BASA,      BUDE,      BUFH,      CAEX,      CALC
      COMMON /IDNAME/  CHRO,      CLAY,      COCO,      COIN,      COPA
      COMMON /IDNAME/  COSA,      COSI,      CSNU,      ELCO,      ELNU
      COMMON /IDNAME/  EP11,      EP12,      EP13,      ERFA,      EROS
      COMMON /IDNAME/  EXAC,      FINE,      FISA,      FISI,      FSAN
      COMMON /IDNAME/  GRGR(2),   HUE1,      HUE2,      IRON,      LIIN
      COMMON /IDNAME/  LILI,      LISH,      LUF(6),    MAGN,      MANG
      COMMON /IDNAME/  MESA,      MICL,      MOTE,      MCZC(2),   MSAN
      COMMON /IDNAME/  MSNU,      OMOD,      ORCA,      ORDR,      PAMA
      COMMON /IDNAME/  PASI,      PHYS,      PLIN,      PLLI,      PCTA
      COMMON /IDNAME/  SAND,      SAPO,      SHLI,      SHRA,      SILI
      COMMON /IDNAME/  SILT
      COMMON /IDNAME/  SLOP,      SODI,      SOEL,      SPGR,      STAB
      COMMON /IDNAME/  STLN,      SUBO,      SUDE(10),SUNA(4),     TERE(2)
      COMMON /IDNAME/  UNIF,      VALU,      VCSA,      VFSA,      VCSH
      COMMON /IDNAME/  WACO,      WAPH,      WIER,      YEAR
C
C     INTERMEDIATE VARIABLES USED IN CALCULATION OF ID VALUES
C
C       REAL * 4 VARIABLES
C
        COMMON /IDNAME/  METROW,   XFRCO,     XPLCO
C
C       INTEGER * 4  VARIABLES
C
        COMMON /IDNAME/  AGOC,     BIOPLT,    FWR,       GRG,       HEAD
        COMMON /IDNAME/  MODE,     MOZ,       ORD,       SIDE,      SUBR
        COMMON /IDNAME/  SURFST,   SWING,     TEMR,      TEX,       INTBUF(50)
C
C
        REAL * 4  METROW, XFRCO, XPLCO
```

Example 3

Although the common block above does not list the variables in the order REAL, INTEGER, LOGICAL, it is a good example of spacing for legibility. (The variables are arranged by usage in this common block).

Common block variables must be described in a BLOCK DATA routine or in an initialization subroutine. The variable descriptions must be alphabetic. See Appendix D for an example.

5. Do not use Fortran entry points unless the use of them is clearly the best solution to an implementation problem.

6. Information and error messages should be informative to the user as well as the programmer. Each message must include the name of the routine printing the message.

```
C                *ELSE   ERROR
 4200               OK = .FALSE.
                    WRITE (TYPWTR, 94210) I, ID(STRESS(I))
                    WRITE (PRNTR, 94210) I, ID(STRESS(I))
94210               FORMAT(' E00A0 STRESS(', I2, ') DOES NOT EQUAL ',
     1                 ' N, Y, OR BLANK. ',
     2              ' INSTEAD IS SET TO (', A4, ')',
     3              4X, '(XTKA)')
```

Example 4

It may be numbered either sequentially (1 to n) or for the labeled Fortran statement nearest the message in the code. In the example above the message is numbered sequentially. In the example software system in Appendix C the messages are numbered for labeled Fortran statements.

7. Labels for code statements must be assigned in ascending order within the body of each routine. For examples see Appendix C.

8. Labels for FORMAT statements must be assigned in ascending order within the body of each routine. The FORMAT labels should be sufficiently different from the code labels that they stand out. For example, code labels in a routine could range from 100 to 900 and FORMAT labels from 9100 to 9900. The FORMAT statements may be interspersed with the executable code or they may be just before the END statement. However, within one routine, they must be either all interspersed or all at the end. The software system shown in Appendix C is an example of FORMAT statements interspersed with executable code.

9. Do not use unnecessary EQUIVALENCE statements. However, there are some data structures for which EQUIVALENCE statements are necessary. For example, a LARSYS ID record contains real data values and integer data values. In order to correctly access both data types, the ID record must be declared as:

```
REAL * 4 RID(2000)
INTEGER * 4 ID(200)
EQUIVALENCE (ID(1), RID(1))
```

10. Use standard LARSYS and Reformatting routines whenever possible. For example, often used LARSYS routines are CTLWRD and BCDVAL (for interpreting control cards), and often used Reformatting routines are IDRITE and EOT (for mounting LARSYS data tapes, writing ID records, and writing end-of-tape records).

11. Document all revisions to routines by adding your name and date to the comments. Include a version number if appropriate. If the revision is appropriate for only a special application, add a comment near the revision comment stating exactly what the special applications is.

```
C
C    WRITTEN 07/19/79 BY CATHERINE KOZLOWSKI FOR FY70
C                           SR&T CONTRACT
C
C    REVISED 11/20/79 BY CATHERINE KOZLOWSKI FOR FY79
C                           SR&T CONTRACT
C
```

Example 5

12. Indent (horizontal) and space (vertical) the source code to improve readability and/or logical flow of each routine. See the software system in Appendix C for examples.

13. When reading or writing a long string of variables, space the variable names the same in the READ/WRITE statement as in the FORMAT statement.

```
C
C*********************************************************************
C
C READ AGRONOMIC RECORD SHEET NUMBER 2 OF THE GROUP OF 7
C
C*********************************************************************
  200     READ  (AGSHE, 9200, END=110)
     1        AGOC2, PAGE2, RID(HEIG), RID(LEPL), ID(GRLE),
     2        ID(YELE), ID(BRLE), XPLCO, XFRCO,
     3        METROW, ID(PEGR), RID(DBGL), RID(DBYL),
     4        RID(DBBL), RID(DBST), RID(DBFR), RID(DBTO),
     5        RID(FRBI), BIOPLT, RID(LEAR), ID(PLMO)
C
 9200     FORMAT(I3, I1, F3.0, 1X, F4.1, 1X, 3I2,
     1         F4.0,F3.0,
     2        F4.0, 1X, I2, 1X, 5F5.0,
     3        2F6.0, I1, F5.0, 1X, I2)
C
          PTRA  = PTRA - 1
          READ(AGSHE, 9201, END=110)
     1         BLKID(HEIG), BLKID(LEPL), BLKID(GRLE), BLKID(YELE),
     2         BLKID(BRLE), BLKID(PLCO), BLKID(FRCO), BLKID(PEGR),
     3         BLKID(DBGL), BLKID(DBYL), BLKID(DBBL), BLKID(DBST),
     4         BLKID(DBFR), BLKID(DBTO), BLKID(FRBI), BLKID(LEAR),
     5         BLKID(PLMO)
C
 9201     FORMAT(T5, A3, 1X, A2, T14, 3A2, 1X, 2A3, T32, A2,
     1              1X, 5(1X,A4), 2(2X,A4), T76, A2, 1X, A2)
C
```

Example 6

14. If possible, use the following convention for FILEDEFing and assigning tape units:

```
FILEDEF 11 TAP1
FILEDEF 12 TAP2
FILEDEF 13 TAP3
FILEDEF 14 TAP4
FILEDEF 10 TAP5
```

where Fortran unit 11 is the output tape and units 12-14, 10 are input tapes.

15. Several suggestions about labels and CONTINUE statements:

a. It is easier to revise routine if each DO loop has its own CONTINUE statement.

```
      DO 120 K = 1, 20
         DO 100 J = 1, 3
            ARRAY(J,K) = J + K
100      CONTINUE
126   CONTINUE
```

Example 7

b. It is easier to revise a routine if all of its non-FORMAT labels are on CONTINUE statements.

16. Debugging convention

B. Layout of Individual Routines

```
C   routine name
C
C*****************************************************************
C   routine name  one-line description
C
C   WRITTEN date BY name FOR CONTRACT name or number
C   REVISED data BY name
C
      SUBROUTINE name
C
      IMPLICIT INTEGER * 4 (A - Z)
C
C   detailed description
C
C   special features and/or limitations
C
C   input
C
C   output
C
C   subroutines used (include one-line description of
      each subroutine)
C
      COMMON /name/ declarations
                  .
                  .
                  .
      COMMON / name/ declarations
C
C*****************************************************************
C
C           LOCAL VARIABLES
C
      local variables declarations by type, then alphabetic
          (include parameters as necessary)
C
C   local variable descriptions including parameters, listed
       alphabetically
C
      body of routine
      END
```

Example 8

All routines should follow the general format outlined above. See Appendix C for a complete system following this layout.

1. The first several lines of the source code should identify the routine.

```
C    SPCSCN
C
C*****************************************************************************
C
C    SPCSCN    REFORMATS ONE SPECSCAN TAPE TO ONE LARSYS RUN.
C
C    WRITTEN 02/14/79 BY CATHERINE KOZLOWSKI    FOR SRET FY79 CONTRACT
C    REVISED 04/04/79 BY CATHERINE KOZLOWSKI
C    REVISED 07/02/79 BY CATHERINE KOZLOWSKI
C
C*****************************************************************************
C
```

Example 9

2. After the IMPLICIT INTEGER * 4 statement, there should be a detailed description of the routine.

```
C
C     SPCSCN
C       THIS PROGRAM AND ITS SUBROUTINES REFORMAT A 7-TRACK (MODE 3)
C     OR 9-TRACK 800 BPI SPECSCAN TAPE TO A 9-TRACK 1600 BPI LARSYS
C     DATA RUN.  THE ORIGINAL SPECSCAN TAPE FOR WHICH THIS SOFTWARE WAS
C     WRITTEN WAS RECEIVED FROM ROBERT A. GOODING, TECHNICOLOR GRAPHIC
C     SERVICES INC. (TEXAS OPERATIONS, LYNDON B. JOHNSON SPACE CENTER,
C     P.O. BOX 58863, HOUSTON, TEXAS 77058).
C     THE INPUT TAPE HAS ONE OR MORE FILES, EACH FILE CORRESPONDING TO 1
C     CHANNEL OF DATA ON THE LARSYS TAPE.  ALL RECORDS ON THE INPUT TAPE
C     ARE 1008 BYTES LONG.  THE FIRST INPUT RECORD OF EACH FILE MAY
C     BE AN ID RECORD -- IF IT IS, THE FIRST BYTE EQUALS HEXIDECIMAL
C     '3F' OR 'FF' AND THE RECORD IS SKIPPED DURING PROCESSING OF DATA.
C     IT IS ASSUMED EACH FILE HAS THE SAME NUMBER  OF RECORDS AND,
C     IF ONE FILE HAS AN ID, THEY ALL HAVE ID RECORDS.
C     IT IS ALSO ASSUMED THAT THE ONLY SIZE OF ONE SPECSCAN INPUT
C     RECORD IS 1008 BYTES.  HOWEVER, ONE LINE OF SPECSCAN INPUT MAY
C     BE SEVERAL INPUT RECORDS LONG.
                                   CONTINUE
C     THE PROGRAM REQUIRES ONE TEMPORARY DISK AND TWO TAPE DRIVES.
C     THE EXEC CALCULATES THE AMOUNT OF TEMP SPACE NEEDED FOR THE DATA
C     TO BE TRANSFERRED.  THE INPUT TAPE DRIVE IS ASSIGNED UNIT NUMBER 12
C     IT MAY BE A 7 OR 9 TRACK DRIVE.  THE OUTPUT TAPE DRIVE IS ASSIGNED
C     UNIT NUMBER 11  IT IS A 9 TRACK DRIVE.
                                   CONTINUE
C     THE PROGRAM FIRST TRANSFERS THE INPUT TAPE FILES TO DISK.  THEN
C     IT SETS UP THE LARSYS ID RECORD.  INPUT DATA IS REFORMATTED LINE
C     BY LINE.  EACH INPUT LINE GOES THROUGH THE FOLLOWING PROCESSINGO
C         1)  INPUT FOR CHANNEL N IS READ FROM DISK N INTO A TEMP BUFFER.
C         2)  THE DATA IN THE TEMP BUFFER IS REFORMATTED FROM 2 BYTES
C             PER VALUE TO 1 BYTE PER VALUE.  THE DATA IS INVERTED (BLACK
C             TO WHITE) AT THIS TIME.  IF NECESSARY, THE DATA IS ALSO
C             FLIPPED LEFT TO RIGHT.
C         3)  THE DATA IS MOVED TO A LARSYS-FORMAT DATA LINE
C             AND WRITEN TO TAPE.
```

Example 10

In the above example, special features and limitations of the routine have been noted. Special features are 1) the input can be on either a 7-track or 9-track tape, and 2) the data can be flipped left to right. Limitations are 1) if one input file has an id record, all input files must have ids, and 2) the routine requires two tape drives and one temporary disk.

3. Input requirements must be specified.

```
C       THE INPUT IS AN ARRAY OF DATA VALUES IN SPECSCAN TAPE
C       FORMAT.
C       EACH DATA VALUE IS ASSUMED TO BE ONE 3-BIT FIELD IN
C       AN 8-BIT BYTE AND ONE 6-BIT FIELD IN AN 8-BIT BYTE.
C       THESE TWO FIELDS REPRESENTING ONE DATA VALUE RANGING
C       FROM 0 TO 511.
```

Example 11

4. Output from a routine must be described.

```
C       THE OUTPUT IS AN ARRAY OF DATA VALUES WITH EACH 2 BYTES
C       REPRESENTING ONE 8-BIT DATA VALUE (THE FIRST BYTE IS SET TO ZERO
C       AND THE SECOND BYTE CONTAINS THE DATA).  OUTPUT VALUES RANGE
C       BETWEEN 0 AND 255.
```

Example 12

5. The source listing must include all non-system subroutines called.

```
C    THE NON-SYSTEM SUBROUTINES USED ARE:
C             EOT       WRITES END-OF-TAPE RECORD N OUTPUT TAPE.
C             GTDATE    RETURNS TODAYS'S DATE IN CHARACTER FORMAT.
C             IDRITE    MOUNTS OUTPUT TAPE AND WRITES OUTPUT RUN ID RECORD.
C             MOUNT     MOUNTS INPUT TAPE.
C             MOVBYT    MOVES BYTES FROM INPUT BUFFER TO OUTPUT BUFFER.
C             SPCDAT    TRANSLATES DATA FROM A 9-BIT FORMAT TO AN 8-BIT
C                       FORMAT.
C             SPCSAM    CALCULATES THE NUMBER OF SAMPLES PER CHANNEL IN THE
C                       OUTPUT.
C             TAPOP (ENTRY POINTS TOPEF, TOPFF,
C                       TOPRD, TOPWR)  PERFORMS TAPE I/O FUNCTIONS.
C
C
C*****************************************************************************
C
C
```

Example 13

6. **All local variables must be declared (as necessary) and described.**

```
C*******************************************************************************
C
      INTEGER * 4 BLFCNT /5C4/, MAXDAT /500/, NCNDAT /4/, RLIMIT /5/,
     1        ZERO /0/
C
      INTEGER * 2 BLFFER(504)
C
      LOGICAL * 4 IIFLG
C
      LOGICAL * 1 INBUF(10CE)
C
      EQUIVALENCE (EUFFER, INBLF)
C
C*******************************************************************************
C
C                   LOCAL VARIABLE DESCRIPTICNS
C
C    ACJUST TEMPORARY VARIABLE LSED TO ACJUST SAMPLE CCUNT TC BE EVENLY
C           DIVISIBLE BY 4.
C    BUFCNT NLMBER OF ELEMENTS IN INPUT BUFFER.
C    BUFFER INPUT BUFFER IN INTEGER * 2 FORMAT.
C    CISP   CISPLACEMENT INTC INPUT BUFFER.
C    ICFLG  FLAG INDICATIANG WHETHER FIRST INPUT RECORD CF THE FILE IS
C           AN ID RECORD.  IF ICFLG IS SET, FIRST RECCRC IS AN ID.
C    INEUF  INPUT BUFFER IN LOGICAL * 1 FCRMAT.
C    LSTVAL LAST CATA VALUE IN INPUT BUFFER.
C    MAXDAT MAXIMLM NUMBER OF DATA VALUES POSSIBLE IN CNE INPUT RECORD.
                                   CONTINUE
C    NCNCAT NUMBER OF NON-DATA VALLES IN INPLT BLFFER.
C    NREAD  NUMBER OF CONSECUTIVE RECORDS READ WHEN SEARCH FCR ZERO DATA
C           VALUES.
C    NSAMP  NUMBER OF SAMPLES PER CHANNEL THAT WILL BE IN LARSYS CUTPUT.
C    PREVAL PREVICUS DATA VALUE IN INPLT BUFFER.  USEC TC SEARCH
C           BACKWARCS IN INPLT RECORD.
C    RLIMIT UPPER LIMIT CN THE NUMBER OF CONSECUTIVE READS TC PERFORM
C           BEFORE TERMINATING SEARCH FOR ZERC DATA VALUES.
C    TCTAL  RUNNING TOTAL USED TC CALCULATE NUMBER CF SAMPLES.
C    UNIT   DISK UNIT FROM WHICH TO READ INPLT RECORDS.
C    ZERO   THE CCNSTANT ZERO.
C
C*******************************************************************************
C*******************************************************************************
C
```

Example 14

C. Comments Within The Body of a Routine

1. Highlight comments that describe large sections of code. See Appendix C for examples.

2. Comments by themselves should describe the flow of the routine in sufficient detail so a reader can understand the routine without looking at the code.

3. Inobvious programming "tricks" must be explained in detail including the reason for the trick.

4. Specific suggestions:

   a. Comment a control card computed GO TO so that it is apparent which label corresponds to which key word.

```
      IMPLICIT INTEGER * 4 (A - Z)
C
      INTEGER * 4 KEYLST(7)
C
      DATA KEYLST /'*INP', 'REFO', 'INPL', 'SCRA', 'OUTP',
     1             'END',  '-COM'/
C
      DATA KEYSZ /7/
C
C
C
      CALL CTLWRD(CARD, COL, KEYLST, KEYSZ, CODE, READIN, ERROR)
C
C             *INP   REFO    INPL    SCRA    OUTP    END     -COM
      GOTO (1000,   2000,   3000,   4000,   5000,   6000,   7000), CODE
C
 1000 CONTINUE
 2000 CONTINUE
 3000 CONTINUE
 4000 CONTINUE
 5000 CONTINUE
 6000 CONTINUE
 7000 CONTINUE
      STOP
      END
```

Example 15

b. Comment logical program structures with statements such as:

```
C
C   WHILE NOT END-OF-FILE PROCESS DATA
C

C
C   REPEAT LINE PROCESSING UNTIL END-OF-FILE
C

C
C   IF GOOD DATA THEN PROCESS IT
C                ELSE PRINT ERROR MESSAGE
C
```

APPENDIX A

* * * CFMRP Control Cards * * *

Keyword	REQ	Option	Parameter	Function	Default
INPUT	*	TAPE	(xxxx)	Input tape number.	(none)
OUTPUT	*	TAPE	(yyyy)	Output tape number.	(none)
		FIRST		Indicates that output run sequencing starts at beginning of tape.[1]	(none)
		LAST		Indicates that output run sequencing starts after the last run sequence already on tape.[1]	(none)
		RUSE	(aaaa)	Start output run sequencing at output run sequence number aaaa.[1]	(none)
INRUS		FRUSE	(bbbb)	Indicates first input run to be CFMRPed.[2]	bbbb='FIRS'
		LRUSE	(cccc)	Indicates last input run sequence to be CFMRPed.[2]	cccc=999999999
END	*	(none)		Indicates end of a processing group.	
$END		(none)		Indicates end of all processing.	

NOTES:

1. Must have a FIRST, LAST, or RUSE option.

2. The defaults are set up so the entire input tape would be CFMRPed. Therefore, a deck setup of

```
INPUT TAPE(9900)
OUTPUT TAPE(9901),FIRST
END
$END
```

would CFMRP all the run sequences on tape 9900 and put the output on tape 9901 starting at run sequence 1.

* * * CFMRP Control Cards * * * (cont)

A deck setup of

```
INPUT TAPE(9902)
OUTPUT TAPE(9903),RUSE(5)
INRUSE FRUSE(7), LRUSE(15)
INRUSE FRUSE(20), LRUSE(25)
END
$END
```

would CFMRP run sequences 7 to 15 inclusive and 20-25 inclusive on tape 9902 and put the output on tape 9903 as run sequences 5 to 19 inclusive.

APPENDIX B

LARS Program Abstract 5475

MODULE IDENTIFICATION

Module Name: IDEDT Function Name:

Purpose: Edit ID record and modify data of MSS tape

System/Language: 360/FORTRAN

Author: Laura Wallace Date: 7/28/78

Latest Revisor: Laura Wallace Date: 10/05/78

MODULE ABSTRACT

Through control cards, the ID record of a LARSYS formatted MSS data tape (either 800 or 1600 BPI) is edited and output to another LARSYS formatted tape (1600 BPI). More than one input tape may be concatenated onto one output tape if the total line count comprises less than 95% of the output tape. There is an option to remove ancillary data in the data records (see MODTAP subroutine abstract). There is also an option to ROTATE the data 180 degrees, north-south (see ROTDAT subroutine abstract).

1. MODULE ABSTRACT:

A. Input:

Input is expected from the card reader (device 5).
Note: The FILEDEF in the EXEC allows input to be on disk.

B. EXEC Cards Needed

```
GETDISK LARSYS ADR 19A MODE A
LOAD IDEDT (CLEAR NOMAP)
USE CTLWRD BCDVAL CPFUNC
FILEDEF 5 DSK-P1 & 1 & 2
START IDEDT
FILEDEF * CLEAR
& EXIT
```

C. Running IDEDT from the Terminal

```
IPL REFORM
IDEDT filename filetype (RUN IDEDT FROM ABOVE EXEC)
```

D. Running IDEDT from Batch

Batch Cards:

```
BATCH MACHINE BATONITE
BATCH ID userid username
BATCH OUTPUT printloc punchloc
EXEC$$
GETDISK REFORM 19D D
GETDISK userid 191 P RR DETACH PASS diskpassword
GLOBAL T REFRMLIB SYSLIB
EXEC IDEDT filename filetype
$$
```

2. INTERNAL DESCRIPTION

All control cards are read. 'OPTION MMS' puts spectral band maximums and minimums into a temporary ID array (TEMPID). EDIT puts new information into the same TEMPID. When the 'END' is read all processing begins. The first input tape is mounted and the ID is read. The number of lines in each of the input tapes is counted and printed. The first N tapes are concatenated onto the output tape. N is the number of tapes whose sum of total line count comprise less than 95% of the output tape The ID record is then read from the input tape and edited using the TEMPID. IDRITE gets the output tape and writes out the ID. If the data is to be rotated 180 degrees then the tape is forward spaced the number of lines to be written. Each line of data is then read (using TOPRB if rotate option selected) from the input tape, modified if the MMS option is given, rotated if requested, and written out to the output tape. The LARS17 forms are printed. The tapes are rewound and detached.

3. SUBROUTINES CALLED

BCDFIL	MOUNT
CPFUNC	RCOUNT
CTLPRM	RINGIN
CTLWRD	ROTDAT
EOT	TOPBF
FVAL	TOPFF
GTDATE	TOPFS
IDRITE	TOPRB
IVAL	TOPRD
LARS17	TOPRW
MODTAP	TOPWR

4. INPUT DESCRIPTION

R E Q	KEY WORD(COL.1)	CONTROL PARAMETER	FUNCTION	DEFAULT
*	INPUT	TAPE(XXX)	MAX OF 5 INPUT TAPES	00000
		FILE(FF)	LOCATION OF FILES FOR INPUT	1
		RUN(XXXXXXXX)	TO BE IMPLEMENTED AT A LATER DATE	(NONE)
		DENSITY(XXXX)	DENSITY OF INPUT TAPE	800 BPI
	OUTPUT	TAPE(XXX)	OUTPUT TAPE (1600 BPI)	00000
		FILE(FF)	FILE LOCATION FOR OUTPUT	00000
		RUN(XXXXXXXX)	RUN NUMBER FOR OUTPUT TAPE	(NONE)
	OPTIONS	MMS	MODIFY DATA, ADD SPECTRAL BAND INFOR FOR MODULAR MULTISPECTRAL SCANNER (MSS)	(NONE)
		FORM(X)	NUMBER OF LARS17 FORMS TO PRINT	8
		DEBUG	PRINT ADDITIONAL INFORMATION AND VARIABLE VALUES	OFF
		ROTATE	ROTATE DATA 180 DEGREES. ONLY [redacted] VALID FOR 1 INPUT TAPE. ID(16) AUTOMATICALLY CHANGES BY 180 DEGREES.	DO NOT ROTATE
		NOCOUNT	DO NOT COUNT THE RECORDS ON THE TAPE	COUNTS THEM
	LARS17	MISSION(XXX)	MISSION NUMBER	0
		SITE(XXX)	SITE NUMBER	0
		LINE(X)	LINE NUMBER OF LINE-RUN-	0
		RUN(X)	RUN NUMBER OF LINE-RUN-	0
*	EDIT	X	ELEMENT OF ID TO BE EDITED	(NONE)
		NNNNN	NEW VALUE (CAN BE INTEGER, REAL, OR ALPHANUMERIC)	(NONE)
*	END		BEGIN PROCESSING	

NOTE: The run number for the output tape is the same run number as in the input tape id record unless the 'RUN' parameter is specified on the 'output' keyword or 'edit' for ID(3) is used. Since the keywords are operated on in order, the keyword which is later in the list of keywords input will take precedence over previous changes.

5. OUTPUT DESCRIPTION

Control cards will be printed, followed by the number of lines of data on the input tapes (unless NOCOUNT option specified) and which tapes were actually read and the new value for the number of data lines, as put in ID(20).
The previous values and new values for all edited elements of the ID record are printed. IDRITE prints the percentage the input will take and any information pertaining to the run number, followed by the IDPRINT of the tape. The LARS17 forms are printed (caused by LARS17 subroutine).

ERROR MESSAGES: (All errors cause the program to halt. The tapes are rewound and detached.)
ERROR ON CARD --- (the card is echoed)
JOB HALTED --- No output written on tape
INPUT TAPE NOT WRITE PROTECTED. HALTED.

```
** EOT DETECTED ON INPUT TAPE BEFORE ALL LINES READ
**     WITH NO 2ND INPUT TAPE GIVEN
**     NUMBER OF LINES ACTUALLY WRITTEN = XXXXX
**NO MORE ROOM ON OUTPUT TAPE.  LINES WRITTEN = XXX

***RUN NUMBER FOR OUTPUT TAPE IS ILL DEFINED -- JOB TERMINATED
ERROR DURING TOPRB.  PROGRAM TERMINATED.  OUTPUT TAPE PARTIALLY
  WRITTEN THRU XXX RECORDS.
ILLEGAL --- CANNOT HAVE MORE THAN ONE INPUT TAPE WITH ROTATE
  OPTION --- JOB TERMINATED.  NO OUTPUT.
```

6. SUBROUTINE AND ENTRY POINTS CALLED

```
RCOUNT
ROTDAT
CTLWRD
  CTLWRD
  CTLPRM
BCDVAL
  IVAL
  FVAL
  BCDFIL
TAPOP
  TOPWR
  TOPRD
  TOPRU
  TOPBF
EOT
CPFUNC
IDRITE
LARS17
MODTAP
REVERS
```

7. SAMPLE CONTROL CARD DECK

```
INPUT TAPE (26,27), FILE(1,1), DENSITY(800)
OPTIONS  MMS,FORM(4)
LARS17   MISSION(365),SITE(194),LINE(2),RUN(1)
EDIT     3,77007500
EDIT     7, LINE 2 RUN1 HAND
EDIT     21, 44.58
END
```

APPENDIX C

```
*
*   WRITTEN 07/02/79 BY CATHERINE KOZLOWSKI FCR SR&T FY79 CONTRACT
*
&BEGTYPE
      HOW MANY FILES WILL BE TRANSFERRED TC DISK, HOW MANY RECCRDS
         PER FILE
&END
      &READ VARS &NFILE &NREC
*
*   CALCULATE HOW MANY CYLINDERS OF 3350 TEMP SPACE IS NEECED
*
      &TOT = &NFILE * &NREC  * 1008
      &NCYL = &TOT / 460000
      &NCYL = &NCYL + 1
      &TYPE &NCYL CYL OF 3350 TEMP SPACE WILL BE REQUESTD
      DEF T3350 250 &NCYL
      &IF &RETCODE NE 91 &GOTO -GOTDSK
         &TYPE TEMP DISK WITH &NCYL OF SPACE NOT AVAIL.
         &EXIT
-GOTDSK &IF &RETCODE EQ 92 &GOTO -FORMATD
         FORMAT 250 B
-FORMATD ACCESS 250 B (ERASE
      GLOBAL TXTLIB REFRMLIB CMSLIB FORTRAN
      LOAD SPCSCN TAPCP  ( NOMAP CLEAR
      FILEDEF 5 READER
      FILEDEF 6 PRINTER (RECFM FA
      FILEDEF 7 PUNCH
      FILEDEF 15 TERM (PERM
      FILEDEF 16 TERM (PERM
*  OUTPUT TAPE
      FILEDEF 11 TAP1 (PERM
*  INPUT TAPE
      FILEDEF 12 TAP2 (PERM 7TRACK DEN 800 TRTCH C
*  TEMPORARY DISK FILES
      FILEDEF 21 DISK SPEC FILE1 B1 (RECFM F LRECL 1008 BLKSIZE 1008
      FILEDEF 22 DISK SPEC FILE2 B1 (RECFM F LRECL 1008 BLKSIZE 1008
      FILEDEF 23 DISK SPEC FILE3 B1 (RECFM F LRECL 1008 BLKSIZE 1008
      START SPCSCN
      CP DET 181
      &EXIT
```

```
      C    SPCSCN
      C
      C**********************************************************************
      C
      C    SPCSCN   REFORMATS ONE SPECSCAN TAPE TO ONE LARSYS RUN.
      C
      C    WRITTEN 02/14/79 BY CATHERINE KOZLOWSKI   FOR SR&T FY79 CONTRACT
      C    REVISED 04/04/79 BY CATHERINE KOZLOWSKI
      C    REVISED 07/02/79 BY CATHERINE KOZLOWSKI
      C
      C**********************************************************************
      C
01           IMPLICIT INTEGER * 4 (A-Z)
      C
      C    SPCSCN
      C       THIS PROGRAM AND ITS SUBROUTINES REFORMAT A 7-TRACK (MODE 3)
      C    OR 9-TRACK 800 BPI SPECSCAN TAPE TO A 9-TRACK 1600 BPI LARSYS
      C    DATA RUN.  THE ORIGINAL SPECSCAN TAPE FOR WHICH THIS SOFTWARE WAS
      C    WRITTEN WAS RECEIVED FROM ROBERT A. GOODING, TECHNICOLOR GRAPHIC
      C    SERVICES INC. (TEXAS OPERATIONS, LYNDON B. JOHNSON SPACE CENTER,
      C    P.O. BOX 58863, HOUSTON, TEXAS 77058).
      C    THE INPUT TAPE HAS ONE OR MORE FILES, EACH FILE CORRESPONDING TO 1
      C    CHANNEL OF DATA ON THE LARSYS TAPE.  ALL RECORDS ON THE INPUT TAPE
      C    ARE 1008 BYTES LONG.  THE FIRST INPUT RECORD OF EACH FILE MAY
      C    BE AN ID RECORD -- IF IT IS, THE FIRST BYTE EQUALS HEXIDECIMAL
      C    '3F' OR 'FF' AND THE RECORD IS SKIPPED DURING PROCESSING OF DATA.
      C    IT IS ASSUMED EACH FILE HAS THE SAME NUMBER  OF RECORDS AND,
      C    IF ONE FILE HAS AN ID, THEY ALL HAVE ID RECORDS.
      C    IT IS ALSO ASSUMED THAT THE ONLY SIZE OF ONE SPECSCAN INPUT
      C    RECORD IS 1008 BYTES.  HOWEVER, ONE LINE OF SPECSCAN INPUT MAY
      C    BE SEVERAL INPUT RECORDS LONG.
02                                     CONTINUE
      C    THE PROGRAM REQUIRES ONE TEMPORARY DISK AND TWO TAPE DRIVES.
      C    THE EXEC CACULATES THE AMOUNT OF TEMP SPACE NEEDED FOR THE DATA
      C    TO BE TRANSFERRED.  THE INPUT TAPE DRIVE IS ASSIGNED UNIT NUMBER 12
      C    IT MAY BE A 7 OR 9 TRACK DRIVE.  THE OUTPUT TAPE DRIVE IS ASSIGNED
      C    UNIT NUMBER 11  IT IS A 9 TRACK DRIVE.
03                                     CONTINUE
      C    THE PROGRAM FIRST TRANSFERS THE INPUT TAPE FILES TO DISK.  THEN
      C    IT SETS UP THE LARSYS ID RECORD.  INPUT DATA IS REFORMATTED LINE
      C    BY LINE.  EACH INPUT LINE GOES THROUGH THE FOLLOWING PROCESSINGO
      C       1)  INPUT FOR CHANNEL N IS READ FROM DISK N INTO A TEMP BUFFER.
      C       2)  THE DATA IN THE TEMP BUFFER IS REFORMATTED FROM 2 BYTES
      C           PER VALUE TO 1 BYTE PER VALUE.  THE DATA IS INVERTED (BLACK
      C           TO WHITE) AT THIS TIME.  IF NECESSARY, THE DATA IS ALSO
      C           FLIPPED LEFT TO RIGHT.
      C       3)  THE DATA IS MOVED TO A LARSYS-FORMAT DATA LINE
      C           AND WRITEN TO TAPE.
04                                     CONTINUE
      C    THE NON-SYSTEM  SUBROUTINES USED ARE:
      C           EOT       WRITES END-OF-TAPE RECORD N OUTPUT TAPE.
      C           GTDATE    RETURNS TODAYS'S DATE IN CHARACTER FORMAT.
      C           IDRITE    MOUNTS OUTPUT TAPE AND WRITES OUTPUT RUN ID RECORD.
      C           MOUNT     MOUNTS INPUT TAPE.
      C           MOVBYT    MOVES BYTES FROM INPUT BUFFER TO OUTPUT BUFFER.
      C           SPCDAT    TRANSLATES DATA FROM A 9-BIT FORMAT TO AN 8-BIT
      C                     FORMAT.
      C           SPCSAM    CALCULATES THE NUMBER OF SAMPLES PER CHANNEL IN THE
      C                     OUTPUT.
      C           TAPOP (ENTRY POINTS TOPEF, TOPFF,
      C                   TOPRD, TOPWR)  PERFORMS TAPE I/O FUNCTIONS.
      C
      C
      C**********************************************************************
      C
      C
05          COMMON /SPCCOM/
      C            INTEGER * 2 VARIABLES
           1     INDAT, OUTDAT,
      C            LOGICAL * 1 VARIABLES
           2     FLIP
```

```
              C
              C
106                INTEGER * 2 INDAT(10000), OUTDAT(10000)
              C
107                LOGICAL * 1 FLIP
              C
              C
              C*************************************************************************
              C
              C          LOCAL VARIABLES
              C
              C
              C
108                REAL * 4 TIME
              C
109                INTEGER * 4 BLANK/'    '/,     F1UNT/21/,   F2UNT/22/,   F3UNT /23/,
                1             HEX3F /Z3F/,      HEXFF /ZFF/, INBCNT /1008/,
                2             INUNT/12/,        MAXCHN/3/,   MAXIN/500/,
                3             MAXLC/10000/,     NO/'NO  '/,  CUTID(200),  CUTLNT/11/,
                4             TRK7 /'7TRK'/
              C
10                 LOGICAL * 4 IDFLG
              C
11                 INTEGER * 2 LARDAT(5000),    ROLL /Z7FFF/
              C
12                 LOGICAL * 1 INBUF(1008),     ZERO/Z00/
              C
              C*************************************************************************
              C
              C             LOCAL VARIABLE DESCRIPTIONS
              C
              C
              C      ANSW    USED TO STORE ANSWER TO QUESTIONS ASKED AT THE TERMINAL.
              C      BLANK   THE CONSTANT BLANK.
              C      CHKID   FIRST BYTE OF FIRST RECORD OF A FILE.
              C      CCUNT   BYTE COUNT USED IN CALLS TO IOPRD AND TOPWR.
              C      CURCHN  CHANNEL CURRENTLY IN PROCESS.
              C      CURLIN  LINE NUMBER OF LARSYS LINE CURRENTLY IN PROCESS.
              C      CFILE   UNIT NUMBER OF DISK FILE CURRENTLY BEING PROCESSED.
              C      DISP    DISPLACEMENT INTO LARSYS LINE CURRENTLY BEING ASSEMBLED.
              C      FLIP    FLAG INDICATING WHETHER INPUT DATA NEEDS TO BE FLIPPED
              C              LEFT TO RIGHT ON OUTPUT.  IF FLIP IS SET, THE DATA SHOULD
              C              BE FLIPPED.
13                                          CONTINUE
              C      F1UNT   DISK UNIT WHERE FIRST TAPE FILE IS TRANSFERRED.
              C      F2UNT   DISK UNIT WHERE SECOND TAPE FILE IS TRANSFERRED.
              C      F3UNT   DISK UNIT WHERE THRID TAPE FILE IS TRANSFERRED.
              C      HEXFLG  EQUALS THE ID FLAG FOR THE INPUT TAPE (DEPENDS ON
              C              FORMAT OF INPUT TAPE).
              C      HEX3F   CONSTANT EQUAL TO 3F HEXIDECIMAL.  AN INPUT RECORD IS
              C              AN ID RECORD IF THE FIRST BYTE EQUALS HEX 3F (7 TRACK FORMAT)
              C      HEXFF   CONSTANT EQUAL TO FF HEXDECIMAL.  AN INPUT RECORD IS
              C              AN ID RECORD IF THE FIRST BYTE EQUALS HEX FF (9TRACK FORMAT)
              C      IDFLG   FLAG INDICATING WHETHER FIRST INPUT RECORD OF A FILE IS AN ID
              C            IF IDFLG IS SET, FIRST RECORD IS AN ID.
              C      INBCNT  NUMBER OF BYTES IN AN INPUT RECORD.
              C      INBUF   BUFFER FOR INPUT RECORDS.
              C      INDAT   INTERMEDIATE LINE BUFFER WITH THE VALUES IN SPECSCAN
              C              (THAT IS, INPUT) FORMAT.
14                                          CONTINUE
              C      INFIL   FILE NUMBER OF FIRST INPUT FILE.  THERE IS A TOTAL OF 3
              C              INPUT FILES.
              C      INTAP   INPUT TAPE NUMBER.
              C      INUNT   UNIT NUMBER OF INPUT TAPE.
              C      LARDAT  BUFFER FOR A PROCESSED LINE OF LARSYS DATA.
              C      LCNT    BYTE COUNT FOR ONE LINE OF LARSYS DATA.
              C      MAXCHN  MAXIMUM NUMBER OF DATA CHANNELS THIS ROUTINE CAN HANDLE
              C      MAXIN   MAXIMUM NUMBER OF DATA VALUES IN ONE INPUT RECORD.
              C      MAXLC   MAXIMUM NUMBER OF BYTES ALLOWED IN ONE LINE OF LARSYS DATA.
              C      MCCUNT  NUMBER OF BYTES TO TRANSFER FROM INPUT BEFFER TO
              C              INTERMEDIATE LARSYS LINE BUFFER.
              C      NCAL    NUMBER OF CALIBRATION BYTES PER CHANNEL IN INTERMEDIATE
              C              LARSYS LINE BUFFER.
```

```
      C     NCHAN   NUMBER OF CHANNELS OF DATA.
      C     NCAT    NUMBER OF DATA BYTES PER CHANNEL IN INTERMEDIATE LARSYS
      C             LINE BUFFER.
      C     NLINE   TOTAL NUMBER OF LARSYS DATA LINES.
      C     NC      CONSTANT EQUAL TO 'NC'.
                                      CONTINUE
      C     NSAMP   NUMBER OF SAMPLES PER CHANNEL PER LINE (INCLUDING CALIBRATION
      C     ONFOUT  NUMBER OF INPUT RECORDS THAT CORRESPOND TO ONE CHANNEL, ONE
      C             LINE OF OUTPUT.
      C     OUTCAT  INTERMEDIATE LARSYS DATA LINE BUFFER.  EACH DATA VALUE TAKES
      C             2 BYTES BUT ACTUAL DATA IS IN SECOND BYTE.
      C     OUTID   ID RECORD FOR OUTPUTTED LARSYS DATA.
                                      CONTINUE
      C     OUTUNT  UNIT NUMBER FOR OUTPUT TAPE.
      C     REC     NUMBER OF RECORDS IN ONE INPUT FILE.
      C     ROLL    ROLL PARAMETER FOR ONE LARSYS LINE.
      C     TAPMOD  TAPE MODE OF INPUT TAPE.
      C     TIME    TIME OF JOB IN SECONDS.
      C     TIME1   START TIME OF JOB IN MILLESECONDS.
      C     TIME2   STOP TIME OF JOB IN MILLESECONDS.
      C     TRK7    CONSTANT EQUAL TO '7TRK'.
      C     ZERO    CONSTANT BYTE EQUAL TO 00 HEXIDECIMAL.
      C
      C*****************************************************************************
      C*****************************************************************************
      C
      C     SET INITIAL CONDITIONS
      C
            TIME1 = TIMER(X)
            COUNT = MAXLC
            CALL MOVBYT(ZERO, 0, 0, LARDAT, 0, 1, COUNT)
      C
      C*****************************************************************************
      C
      C     MOUNT INPUT TAPE AND TRANSFER  DATA FILES TO DISK
      C
      C*****************************************************************************
      C
            WRITE(16, 9050)
       9050 FORMAT(' IS050 ENTER INPUT TAPE AND FILE NUMBERS (I5,I4)')
            READ(15,9060) INTAP, INFIL
       9060 FORMAT(I5,I4)
            WRITE(16,9070) INTAP, INFIL
       9070 FORMAT(' IS070 INPUT TAPE IS',I5,' AND INPUT FILE IS',I4)
            WRITE(16,9073)
       9073 FORMAT(' IS073 ENTER NUMBER OF INPUT FILES AND MODE OF',/,
           1       '         TAPE (I3,1X,A4)',/,
           2       ' FOR TAPE MODES:     800 MEANS 9TRACK 800 BPI',/,
           3       '                    1600 MEANS 9TRACK 1600 BPI',/,
           4       '                    7TRK MEANS 7TRACK MODE 3 800 BPI')
            READ(15,9074) NCHAN,TAPMOD
       9074 FORMAT(I3,1X,A4)
            WRITE(16,9075) NCHAN,TAPMOD
       9075 FORMAT(' IS075 THERE ARE',I3,' INPUT FILES AND INPUT TAPE',/,
           1       '            MODE IS ',A4,'   (SPCSCN)')
            IF (NCHAN .LE. MAXCHN) GO TO 79
               WRITE(6,9077) NCHAN, MAXCHN
               WRITE(16,9077)NCHAN, MAXCHN
       9077    FORMAT(' ES077 ',I4,' CHANNELS REQUESTED FOR PROCESSING BUT',/,
           1          '      SPCSCN IS SET UP TO HANDLE ONLY',I4,' CHANNELS',/,
           2          ' ----- PROCESSING ABORTED  (SPCSCN)')
               STOP
         79 CONTINUE
      C
      C     SET HEXIDECIMAL ID FLAG ACCORDING TO INPUT TAPE MODE
      C
            HEXFLG = HEXFF
            IF (TAPMOD .EQ. TRK7) HEXFLG = HEX3F
            CALL MOUNT(INTAP,INUNT, 'RO', TAPMOD)
            IF (INFIL .EQ. 1) GO TO 90
               DO 80 J = 2, INFIL
```

```
43                    CALL TOFFF(INUNT)
44          80     CONTINUE
45          90 CONTINUE
        C
        C     SET UP VARIABLES TO TRANSFER FIRST SPECSCAN FILE TO
        C     DISK FILE 'SPEC FILE1'
        C
46               DFILE = FILNT
47               IDFLG = .FALSE.
48               DO 150 J = 1, NCHAN
49                  REWIND DFILE
50                  REC = 0
51         100      CONTINUE
        C
        C     READ IN ONE PECCRD FROM TAPE
        C
52                     COUNT = INBCNT
53                     REC = REC + 1
54                     CALL TOFRD(INUNT, COUNT, ERR, INBUF)
        C
        C     CHECK FOR END OF TAPE FILE
        C
55                     IF (ERR .EQ. 1) GO TO 120
56                     IF (REC .GT. 1) GO TO 105
        C
        C     SET FLAG IF WE HAVE AN ID RECORD
        C
57                        CHKID = 0
58                        CALL MOVBYT(INBUF, 0, 1, CHKID, 3, 1, 1)
59                        IF (CHKID.EQ. HEXFLG) IDFLG = .TRUE.
60         105         CONTINUE
61                     IF (ERR .EQ. 0) GO TO 110
62                        WRITE(6,9100) ERR, REC, J
63                        WRITE(16,9100)ERR, REC, J
64        9100            FORMAT(' E9100 TOPRD ERR=',I5,' ON RECORD',I5,' OF FILE',
               1                 I5,'    (SPCSCN)')
65         110         CONTINUE
        C
        C     WRITE RECORD TO DISK
        C
66                     WRITE(DFILE,9110) INBUF
67        9110         FORMAT(10(100A1),8A1)
68                     GO TO 100
69         120      CONTINUE
        C
        C     TRANSFER COMPLETE
        C
70                  WRITE( 6,9120)J
71                  WRITE(16,9120)J
72        9120      FORMAT(' I9120 FILE',I5,' TRANSFERRED TO DISK (SPCSCN)')
        C
        C     END OF TAPE FILE WAS READ---SET UP VARIABLES TO
        C     TRANSFER NEXT SPECSCAN FILE TO DISK
        C
73                  DFILE = DFILE + 1
74         150 CONTINUE
        C
        C      DETACH INPUT TAPE DRIVE SINCE ALL INPUT NOW IS ON DISK
        C
75               CALL CPFUNC(7,'DET 182',CERR)
76               IF (CERR .EQ. 0) GO TO 170
77                  WRITE(6,9160) CERR
78                  WRITE(16,9160) CERR
79        9160      FORMAT(' E9160 ERRCR ',I3,' RETURNED FROM CPFUNC    (SPCSCN)')
80         170 CONTINUE
        C
        C*****************************************************************
        C
        C     SET UP OUTPUT LARSYS ID RECORD AND MOUNT OUTPUT TAPE
        C
        C*****************************************************************
```

```
              C
0081             200 CONTINUE
              C
              C     INITIALIZE LARSYS ID RECORD
              C
0082                    CALL MOVBYT(ZERO, 0, 0, OUTID, 0, 1, 800)
              C
              C     GET OUTPUT TAPE AND FILE NUMBERS
              C
0083                    WRITE(16,9200)
0084             9200   FORMAT(' I9200 ENTER OUTPUT TAPE AND FILE NUMBERS (I5,I4)',/,
                    1     'IF YOU ENTER      0    0 IDRITE WILL SELECT FOR YOU')
0085                    READ(15,9210) OUTID(1),OUTID(2)
0086             9210   FORMAT(I5,I4)
              C
              C     GET RUN NUMBER
              C
0087                    WRITE(16,9230)
0088             9230   FORMAT(' I9230 ENTER RUN NUMBER (I8)')
0089                    READ(15,9240) OUTID(3)
0090             9240   FORMAT(I8)
              C
              C     STORE CONTINUATION CODE AND NUMBER OF CHANNELS
              C
0091                    OUTID(4) = 0
0092                    OUTID(5) = NCHAN
              C
              C     CALCULATE NUMBER OF SAMPLES PER CHANNEL PER LINE
              C
0093                    CALL SPCSAM(NSAMP, FILNT, IDFLG)
0094                    OUTID(6) = NSAMP
              C
              C     GET FLIGHT LINE
              C
0095                    WRITE(16,9250)
0096             9250   FORMAT(' I9250 ENTER FLIGHT LINE (4A4)')
0097                    READ(15,9260) OUTID(7), OUTID(8), OUTID(9), OUTID(10)
0098             9260   FORMAT(4A4)
              C
              C     GET DATE DATA WAS TAKEN
              C
0099                    WRITE(16,9270)
0100             9270   FORMAT(' I9270 ENTER MONTH, DAY, YEAR DATA WAS TAKEN (3I4)')
0101                    READ(15,9275)OUTID(11), OUTID(12), OUTID(13)
0102             9275   FORMAT(3I4)
              C
              C     GET TIME DATA WAS TAKEN, ALTITUDE OF SENSOR PLATFORM,
              C     AND GROUND HEADING
              C
0103                 WRITE(16,9277)
0104             9277 FORMAT(' I9277 ENTER TIME TAKEN, ALT. OF PLATFORM, AND GROUND',/,
                    1      '      HEADING    (A4,I10,I5)')
0105                 READ(15,9278) OUTID(14), OUTID(15), OUTID(16)
0106             9278 FORMAT(A4,I10,I5)
              C
              C     GET DATE DATA WAS REFORMATTED
              C
0107                    CALL GTDATE(OUTID(17))
              C
              C     CALCULATE NUMBER OF LINES OF DATA THAT WILL BE PRODUCED IN THE
              C     LARSYS RUN --FIRST DETERMINE NUMBER OF INPUT DATA RECORDS
              C
0108                    OUTID(20) = REC - 1
0109                    IF (IDFLG) OUTID(20) = OUTID(20) - 1
              C
              C     THEN DETERMINE HOW MANY INPUT RECORDS MAKE UP ONE OUTPUT LINE
              C
0110                    ONEOUT = (NSAMP + (MAXIN -1)) / MAXIN
0111                    OUTID(20) = OUTID(20) / ONEOUT
              C
              C     GET FRAME CENTER LATITUDE AND LONGITUDE
```

```
            C
0112              WRITE(16,9279)
0113          9279 FORMAT(' I9279 ENTER FRAME CENTER LAT & LONG  (2F7.2)')
0114              READ(15,9281) OUTID(21), OUTID(22)
0115          9281 FORMAT(2F7.2)
            C
            C   PRINT OUT ASSEMBLED ID VALUES FOR REVIEW
            C
0116                 WRITE(16,9280) (OUTID(JJ),JJ=1,22)
0117          9280   FORMAT(//,' I9280 OUTPUT ID IS SET UP AS0    (SPCSCN)',/,
                 1          ' TAPE NO...',I5,'   FILE NO....',I4,/,
                 2          ' RUN NO....',I5,'   CONT. CODE...',I2,/,
                 3          ' NO. OF CHANNELS...',I3,'   NO. OF SAMPLES...',I10,/,
                 4          ' FLIGHT LINE...',4A4,/,
                 5          ' MONTH,DAY,YEAR...',I5,I3,I5,/,
                 6          ' TIME DATA TAKEN...',A4,/,
                 7          ' ALTITUDE AND GR. HEADING...',I10,I5,/,
                 8          ' DATE REFORMATTED...',3A4,/,
                 9          ' NUMBER OF DATA LINES...',I5,/,
                 A          ' LAT & LONG...',2F7.2)
0118                 WRITE(16,9290)
0119          9290   FORMAT(' I9290 IS OUTPUT ID OKAY    (YES OR NO)')
0120                 READ(15,9295) ANSW
0121          9295   FORMAT(A4)
0122              IF (ANSW .EQ. NO) GO TO 200
            C
            C   MOUNT OUTPUT TAPE AND WRITE OUT ID RECORD
            C
0123              CALL IDRITE(OUTID, OUTUNT, ERROR)
            C
            C*************************************************************************
            C
            C   PROCESS DATA --LINE BY LINE
            C
            C*************************************************************************
            C
            C   CHECK IF OUTPUT NEEDS TO BE FLIPPED LEFT TO RIGHT FROM INPUT DATA
            C
0124              WRITE(16,9296)
0125          9296 FORMAT(' I9296 SHOULD THE OUTPUT DATA BE FLIPPED RIGHT TO LEFT ',/
                 1    ,'         ANSWER YES OR NO     (SPCSCN)')
0126              READ(15,9295) ANSW
0127              FLIP = .TRUE.
0128              IF (ANSW .EQ. NO) FLIP = .FALSE.
0129              IF (FLIP) WRITE(6, 9297)
0130              IF (FLIP) WRITE(16,9297)
0131          9297 FORMAT(//,' I9297 DATA WILL BE FLIPPED LEFT TO RIGHT (SPCSCN)')
            C
            C   PREPARE FOR LINE PROCESSING
            C
0132              DFILE = FIUNT
0133              DO 296 J = 1, NCHAN
0134                 REWIND DFILE
0135                 DFILE = DFILE + 1
0136           296 CONTINUE
0137              LCNT = 4 + (NSAMP * NCHAN)
0138              NDAT = (NSAMP - 6) * 2
0139              NCAL = 12
            C
            C   CHECK FOR OUTPUT GREATER THAN SIZE OF OUTPUT BUFFER
            C
0140              IF (LCNT .GT. MAXLC) GO TO 420
0141              NLINE = OUTID(20)
            C
            C   SKIP SPECSPAN ID RECORDS IF THEY ARE PRESENT
            C
0142              IF (.NOT. IDFLG) GO TO 300
0143                 DFILE = FIUNT
0144                 DO 298 J = 1, NCHAN
0145                    READ(DFILE,9300,END=400) INBUF
0146                    DFILE = DFILE + 1
```

```
      298    CONTINUE
      300 CONTINUE
C
C       PROCESS DATA LINE BY LINE
C
          DO 390 CURLIN = 1,NLINE
C
C       READ IN ALL THE CHANNELS FOR ONE LINE
C
             DISP = 0
             DFILE = FIINT
             DO 320 CURCHN = 1, NCHAN
C
C       OBTAIN ALL THE INPUT RECORDS THAT MAKE UP ONE OUTPUT CHANNEL FOR
C       ONE LINE
C
                DO 310 J = 1, ONEOUT
                   READ(DFILE,9300,END=400) INBUF
      9300         FORMAT(10(100A1),8A1)
C
C       TRANSFER THE DATA VALUES TO INPUT DATA LINE BUFFER---
C       SKIP FIRST 8 BYTES SINCE THEY ARE X AND Y INTERVAL
C       OFFSETS AND NOT DATA.
C
                   MCOUNT = MAXIN * 2
                   IF (J .EQ. ONEOUT) MCOUNT = NDAT - (((J-1) * MAXIN) * 2)
                   CALL MOVBYT(INBUF, 8, 1, INDAT, DISP, 1, MCOUNT)
                   DISP = DISP + MCOUNT
      310       CONTINUE
C
C       SET CALIBRATION BYTES FOR THIS CHANNEL
C
                CALL MOVBYT(ZERO, 0, 0, INDAT, DISP, 1, NCAL)
                DISP = DISP + NCAL
C
C       GO TO NEXT CHANNEL
C
                DFILE = DFILE + 1
      320    CONTINUE
C
C       REPACK THE DATA FROM A 9-BIT FIELD IN 2 BYTES INTO A
C       8-BIT FIELD IN 1 BYTE
C
             CALL SPCDAT(NSAMP, NCHAN)
C
C       SET UP LINE AND ROLL PARAMETER
C
             LARDAT(1) = CURLIN
             LARDAT(2) = ROLL
C
C       TRANSFER DATA TO OUTPUT ARRAY, DELETING NON-DATA BYTES
C
             CALL MOVBYT(OUTDAT, 1, 2, LARDAT, 4, 1, LCNT-4)
C
C       OUTPUT ONE LINE
C
             COUNT = LCNT
             CALL TOPWR(OUTUNT, COUNT, ERR, LARDAT)
             IF (ERR .EQ. 0) GO TO 360
                WRITE( ,9350) ERR, CURLIN
                WRITE(16,9350) ERR, CURLIN
     9350       FORMAT(' E9350 TOPWR ERR=',I5,' WRITING OUTPUT LINE',I6,
         1      '   (SPCSCN)')
      360    CONTINUE
             IF (MOD(CURLIN,100) .EQ. 0) WRITE(6,9360) CURLIN
             IF (MOD(CURLIN,100) .EQ. 0) WRITE(16,9360) CURLIN
     9360    FORMAT(' I9360 ',I7,'LINES PROCESSED    (SPCSCN)')
      390 CONTINUE
          GO TO 450
C
C**********************************************************************
```

```
           C
           C   THE INPUT FILES ARE NOT THE SAME LENGTH SO LINE COUNT IS WRONG IN
           C   OUTPUT ID RECORD
           C
           C**********************************************************************
           C
0181        400     CONTINUE
0182                WRITE( 6,9400)INLINE, CURLIN
0183                WRITE(16,9400)INLINE, CURLIN
0184       9400     FORMAT(' E9400 THERE WAS A PREMATURE END-OF-FILE ON ONE OF THE'
                1          ,' INPUT FILE',/,
                2          ' SO THE LINE COUNT IN THE ID OF',I5,' IS PROBABLY',
                3         ' WRONG',/,
                4          ' PROBABLE LINE COUNT IS (',I5,' - 1) BUT CHECK ',
                5          'OUTPUT FILE AND EDIT THE OUTPUT ',/,
                6          ' ID IF NECESSARY      (SPCSCN)')
0185            GO TO 450
           C
           C**********************************************************************
           C
           C   OUTPUT DATA LINES TOO BIG FOR OUTPUT BUFFER
           C
           C**********************************************************************
           C
0186        420 CONTINUE
0187            WRITE(6, 9420) LCNT, MAXLC
0188            WRITE(16,9420) LCNT, MAXLC
0189       9420 FORMAT(/,' E9420 INPUT DATA NEEDS',I7,' BYTES OF BUFFER SPACE BUT'
                1     ,' OUTPUT BUFFER IS SET UP',/,
                2     '          TO HANDLE ONLY',I7,' BYTES--- PROCESSING',/,
                3     ' TERMINATED ABNORMALLY    (SPCSCN)')
           C
           C**********************************************************************
           C
           C   END THE JOB
           C
           C**********************************************************************
           C
0190        450 CONTINUE
           C
           C   PUT END-OF-FILE MARK AT THE END OF THE OUTPUT RUN AND
           C   PUT END-OF-TAPE RECORD AFTER THAT
           C
0191            CALL TOPEF(OUTUNT)
0192            CALL EOT(OUTTIC, OUTUNT, ERR)
           C
           C   PRINT OUT TIMING INFORMATION
           C
0193            TIME2 = TIMER(X)
0194            TIME = FLOAT(TIME2 -TIME1) / 1000.0
0195            WRITE( 6,9460) TIME
0196            WRITE(16,9460) TIME
0197       9460 FORMAT(/,' I9460 CPU TIME FOR THIS JOB WAS',F10.3,' SEC.    ',
                1      '(SPCSCN)')
0198            STOP
0199            END
```

ORTRAN IV G LEVEL 21 SPCSCN DATE = 80121 10/53/(

E SPCSCN PURDUE / LARS 3031

COMMON BLOCK /SPCCCM / MAP SIZE 9C41

MBCL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBCL	LOC.
DAT	0	OUTDAT	4E20	FLIP	9C40		

SUBPROGRAMS CALLED

MBCL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBCL	LOC.
MER	174	MOVBYT	178	IBCCM#	17C	MCUNT	
PRC	188	CPFUNC	18C	SPCSAM	190	GTDATE	
CDAT	19C	TOPWR	1A0	TOPEF	1A4	ECT	

SCALAR MAP

MBCL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBCL	LOC.
ANK	2E4	F1UNT	2E8	F2UNT	2EC	F3UNT	
XFF	2F8	INBCNT	2FC	INUNT	300	MAXCHN	
XLC	30C	NO	310	CUTCNT	314	TRK7	
	320	COUNT	324	INTAP	328	INFIL	
PMCD	334	HEXFLG	338	J	33C	OFILE	
C	348	ERR	34C	CHKID	350	CERR	
EOUT	35C	JJ	360	ANSW	364	ERRCR	
AT	370	NCAL	374	NLINE	378	CURLIN	
RCHN	384	MCOUNT	388	TIME2	38C	TIME	
RC	396						

ARRAY MAP

MBCL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBCL	LOC.
TIC	398	LARDAT	6B8	INBUF	20C8		

FORMAT STATEMENT MAP

MBCL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBCL	LOC.
9050	31C7	9060	31FB	9070	32C1	9073	3.
9075	3313	9077	3366	9100	33F3	9110	3.
9160	346E	9200	34A3	9210	35CC	9230	3.
9250	3536	926C	3558	9270	355C	9275	3.
9278	35EF	9279	35F7	9281	3628	9280	3.
9295	37B6	9296	37BA	9297	3819	9300	3.
9360	389A	940C	38C3	942C	3986	9460	3.

OPTIONS IN EFFECT ID,EBCDIC,SOURCE,NOLIST,DECK,NOLOAD,MAP
OPTIONS IN EFFECT NAME = SPCSCN , LINECNT = 75
STATISTICS SOURCE STATEMENTS = 159,PROGRAM SIZE = 18034
STATISTICS NO DIAGNOSTICS GENERATED

```
                C   SPCCAT
                C
                C*********************************************************************
                C
                C   SPCCAT   TRANSLATE 9-BIT FIELDS TO 8-BIT FIELDS
                C
                C         WRITTEN 02/15/79 BY CATHERINE KOZLCWSKI FCR SR&T FY79 CONTRACT
                C         REVISEC 04/03/79 BY CATHERINE KOZLCWSKI
                C
                C*********************************************************************
                C
 0001                 SUBRCUTINE SPCDAT(NSAMP, NCh.. )
                C
 0002                 IMPLICIT INTECER * 4 (A-Z)
                C
                C   SUBROUTINE TC TRANLATE 9-BIT FIELDS INTC 8-BIT FIELDS.
                C      THE INPUT IS AN ARRAY OF DATA VALUES IN SPECSCAN TAPE
                C      FORMAT.
                C      EACH DATA VALUE IS ASSUMED TO BE CNE 3-BIT FIELD IN
                C      AN 8-BIT BYTE AND ONE 6-BIT FIELC IN AN 8-BIT BYTE.
                C      THESE TWO FIELDS REPRESENTING CNE DATA VALUE RANGING
                C      FROM 0 TO 511.
                C      THE OUTPUT IS AN ARRAY CF DATA VALUES WITH EACH 2 BYTES
                C      REPRESENTING ONE 8-BIT DATA VALUE (THE FIRST BYTE IS SET TO ZERC
                C      ANC THE SECOND BYTE CONTAINS THE CATA).  CUTPUT VALUES RANGE
                C      BETWEEN 0 ANC 255.
                C
                C
                C   THIS ROUTINE CALLS NO SUBRCUTINES.
                C
                C*********************************************************************
                C
                C
 0003                 COMMON /SPCCOM/
                C                INTECER * 2 VARIABLES
                   1       INDAT, CUTDAT,
                C               LCGICAL * 1 VARIABLES
                   2       FLIP
                C
                C
 0004                 INTEGER * 2 INDAT(100C0), OUTDAT(1??0C)
                C
 0005                 LOGICAL * 1 FLIP
                C
                C*********************************************************************
                C
                C
                C          LOCAL VARIABLES
                C
 0006                 INTEGER * 4 NCAL/6/
                C
                C*********************************************************************
                C
                C
                C              LOCAL VARIABLE DESCRIPTIONS
                C
                C
                C   CCHAN  CHANNEL CURRENTLY IN PROCESSING.
                C   CSAMP  SAMPLE CURRENTLY IN PROCESSING.
                C   CURIN  DISPLACEMENT INTO INPUT BUFFER FCR CURRENT SAMPLE.
                C   CUROUT DISPLACEMENT INTC OUTPUT BUFFER FCR CURRENT SAMPLE.
                C   DISP   DISPLACEMENT INTC LINE BUFFER.  PCINTS TC BYTE JUST
                C          BEFORE FIRST BYTE OF A CHANNEL.
                C   FLIP   FLAG INDICATING WHETHER INPUT DATA NEEDS TC BE FLIPPEC
                C          LEFT TO RIGHT ON OUTPUT.  IF FLIP IS SET, THE DATA
                C          SHOULC BE FLIPPED.
 0007                                  CONTINUE
                C   HBIT   WORK SPACE FOR 3 MOST SIGNIFICANT BITS CF INPUT CATA.
                C   LBIT   WORK SPACE FOR 6 LEAST SIGNIFICANT BITS CF INPUT CATA VALUES.
                C   INCAT  LINE BUFFER WITH DATA IN INPUT FCRMAT.
                C   NCAL   NUMBER OF CALIBRATICN VAUES PER CHANNL.
                C   NCHAN  NUMBER OF CHANNELS CF DATA PER LINE.
                C   NDATA  NUMBER OF DATA SAMPLES PER CHANNEL.
                C   NSAMP  NUMBER OF SAMPLES PER CHANNEL.
```

```
      C    OUTDAT LINE BUFFER WITH DTA IN OUTPUT FORMAT.
      C    TEMP   WORK SPACE USED IN CALCULATION OF HBIT AND LBIT.
      C
      C***********************************************************************
      C***********************************************************************
      C
      C    SET NUMBER OF NON-CALIBRATION DATA VALUES
      C
0008         NDATA = NSAMP - NCAL
      C
      C    PROCESS THE INPUT ARRAY CHANNEL BY CHANNEL
      C
0009         DO 400 CCHAN = 1, NCHAN
      C
      C    CALCULATE POINTER TO BEGINNING OF CURRENT CHANNEL
      C
0010            DISP = (CCHAN - 1) * NSAMP
      C
      C    PROCESS THE DATA CHANNEL SAMPLE BY SAMPLE
      C
0011            DO 300 CSAMP = 1, NDATA
      C
      C    CALCULATE POINTERS TO CURRENT INPUT DATA VALUE AND
      C    ITS CORRESPONDING OUTPUT DATA VALUE
      C       NOTE THAT OUTPUT DATA WILL BE REVERSED LEFT TO
      C       RIGHT FROM INPUT DATA IF FLIP FLAG IS SET
      C
0012               CURIN = DISP + CSAMP
0013               CUROUT = CURIN
0014               IF (FLIP) CUROUT = DISP + (NDATA - CSAMP + 1)
      C
      C    PUT THE 3-BIT INPUT FIELD INTO THE 3 MOST SIGNIFICANT BITS OF
      C    TEMPORARY OUTPUT 9-BIT FIELD
      C
0015               TEMP = INDAT(CURIN)
0016               HBIT = TEMP / 256
0017               HBIT = HBIT * 64
      C
      C    PUT THE 6-BIT INPUT FIELD INTO THE 6 LEAST SIGNIFICANT BITS OF THE
      C    TEMPORARY OUTPUT 9-BIT FIELD
      C
0018               LBIT = MOD(TEMP, 64)
0019               TEMP = HBIT + LBIT
      C
      C    STORE THE TEMPORARY FIELD IN THE OUTPUT ARRAY.
      C    TRANSLATE THE 9-BIT FIELD INTO AN 8-BIT FIELD BY DROPPING
      C    THE RIGHT-MOST BIT.
      C    SINCE THE INPUT IS A NEGATIVE IMAGE, THE OUTPUT IMAGE
      C    IS INVERTED TO MAKE A POSITIVE IMAGE.  (THAT IS, A TEMPORARY
      C    FIELD OF 0 IS OUTPUTTED AS 255 AND A TEMPORARY FIELD
      C    OF 255 IS OUTPUTTED AS 0.)
      C
0020               OUTDAT(CUROUT) = 255 - (TEMP / 2)
0021     300    CONTINUE
0022     400 CONTINUE
0023         RETURN
0024         END
```

```
              C   SPCSAM
              C
              C*****************************************************************
              C
              C   SPCSAM   CLACULATE NUMBER OF SAMPLES IN LARSYS OUTPUT
              C
              C   WRITTEN 04/03/79 BY CATHERINE KOZLOWSKI FOR SR&T FY79 CONTRACT
              C
              C*****************************************************************
              C
0001                SUBROUTINE SPCSAM(NSAMP, UNIT, IDFLG)
              C
0002                IMPLICIT INTEGER * 4 (A - Z)
              C
              C*****************************************************************
              C
              C      SPCSAM DETERMINES THE NUMBER OF SAMPLES PER LINE PER CHANNEL
              C   IN THE LARSYS FORMATTED OUTPUT RUN.  THE ROUTINE READ INPUT
              C   RECORDS OFF DISK UNTIL AN INPUT RECORD ENDS WITH ZEROES.
              C   THE ROUTINE THEN COUNTS THE NUMBER OF NON-ZERO INPUT VALUES THAT
              C   HAVE BEEN READ AND RETURNS THAT COUNT PLUS CALIBRATION BYTES AS THE
              C   NUMBER OF SAMPLES PER CHANNEL.  THE INPUT TO THE ROUTINE IS THE UNIT
              C   NUMBER OF A DISK FILE CONTAINING INPUT DATA RECORDS AND AN INPUT
              C   ID FLAG.  THE OUTPUT FROM THE ROUTINE IS THE NUMBER OF SAMPLES.
              C   THIS ROUTINE ASSUMES THE DISK FILE HAS ALREADY BEEN SET UP.
              C
              C   THIS ROUINTE CALLES ONLY SYSTEM SUBROUTINES.
              C
              C*****************************************************************
              C
0003                INTEGER * 4 BUFCNT /504/, MAXDAT /500/, NONDAT /4/, RLIMIT /5/,
                   1        ZERO /0/
              C
0004                INTEGER * 2 BUFFER(504)
              C
0005                LOGICAL * 4 IDFLG
              C
0006                LOGICAL * 1 INBUF(1008)
              C
0007                EQUIVALENCE (BUFFER, INBUF)
              C
              C*****************************************************************
              C
              C                 LOCAL VARIABLE DESCRIPTIONS
              C
              C   ADJUST TEMPORARY VARIABLE USED TO ADJUST SAMPLE COUNT TO BE EVENLY
              C          DIVISIBLE BY 4.
              C   BUFCNT NUMBER OF ELEMENTS IN INPUT BUFFER.
              C   BUFFER INPUT BUFFER IN INTEGER * 2 FORMAT.
              C   DISP   DISPLACEMENT INTO INPUT BUFFER.
              C   IDFLG  FLAG INDICATIANG WHETHER FIRST INPUT RECORD OF THE FILE IS
              C          AN ID RECORD.  IF IDFLG IS SET, FIRST RECORD IS AN ID.
              C   INBUF  INPUT BUFFER IN LOGICAL * 1 FORMAT.
              C   LSTVAL LAST DATA VALUE IN INPUT BUFFER.
              C   MAXDAT MAXIMUM NUMBER OF DATA VALUES POSSIBLE IN ONE INPUT RECORD.
0008                                      CONTINUE
              C   NONDAT NUMBER OF NON-DATA VALUES IN INPUT BUFFER.
              C   NREAD  NUMBER OF CONSECUTIVE RECORDS READ WHEN SEARCH FOR ZERO DATA
              C          VALUES.
              C   NSAMP  NUMBER OF SAMPLES PER CHANNEL THAT WILL BE IN LARSYS OUTPUT.
              C   PREVAL PREVIOUS DATA VALUE IN INPUT BUFFER.  USED TO SEARCH
              C          BACKWARDS IN INPUT RECORD.
              C   RLIMIT UPPER LIMIT ON THE NUMBER OF CONSECUTIVE READS TO PERFORM
              C          BEFORE TERMINATING SEARCH FOR ZERO DATA VALUES.
              C   TOTAL  RUNNING TOTAL USED TO CALCULATE NUMBER OF SAMPLES.
              C   UNIT   DISK UNIT FROM WHICH TO READ INPUT RECORDS.
              C   ZERO   THE CONSTANT ZERO.
              C
              C*****************************************************************
              C*****************************************************************
              C
```

```
      C     INITIALIZE VARIABLES
      C
              TOTAL = 0
              NREAD = 0
      C
      C*********************************************************************
      C
      C     READ DATA FROM FIRST DATA FILE UNTIL DATA VALUES EQUAL ZERO
      C
      C*********************************************************************
      C
              REWIND UNIT
      C
      C     SKIP ID RECORD IF THERE IS ONE
      C
              IF (IDFLG) READ(UNIT,9100)INBUF
       9100 FORMAT(10(100A1),8A1)
        120 CONTINUE
                 READ(UNIT,9100) INBUF
                 LSTVAL = BUFFER(BUFCNT)
                 TOTAL = TOTAL + MAXDAT
                 NREAD = NREAD + 1
      C
      C     STOP READING RECORDS IF ZERO VALUES FOUND OR RLIMIT RECORDS HAVE
      C     BEEN READ IN A ROW
      C
              IF ((LSTVAL .NE. ZERO) .AND. (NREAD .LT. RLIMIT)) GO TO 120
      C
      C
              IF (LSTVAL .EQ. ZERO) GO TO 200
      C
      C     THERE WERE NO ZERO DATA VALUES IN RLIMIT CONSECUTIVE DATA RECORDS
      C     --- ASSUME 1 SPECSCAN RECORD EQUALS 1 LARSYS CHANNEL PER LINE
      C
                 TOTAL = MAXDAT
                 WRITE(6, 9150)
                 WRITE(16,9150)
       9150      FORMAT(/,' IT IS ASSUMED THAT 1 SPECSCAN RECORD EQUALS',
           1            ' 1 CHANNEL PER LINE OF LARSYS DATA   (SPCSAM)')
                 GO TO 290
      C
      C     COUNT NUMBER OF ZERO DATA VALUES AND DELETE FROM TOTAL COUNT
      C
        200      CONTINUE
                 DISP = BUFCNT
                 PREVAL = ZERO
        220      CONTINUE
                 IF ((DISP .LE. NONDAT) .OR. (PREVAL .NE. ZERO)) GO TO 240
                    PREVAL = BUFFER(DISP)
                    IF (PREVAL .EQ. ZERO) TOTAL = TOTAL - 1
                    DISP = DISP - 1
                    GO TO 220
        240      CONTINUE
                 IF (DISP .GT. NONDAT) GO TO 270
      C
      C     THERE ARE INPUT RECORDS WITH ONLY ZEROS IN THEM---
      C     INCLUDE SOME OF THESE ZERO DATA VALUES IN THE OUTPUT SO
      C     THERE WILL BE PROPER CORRESPONDENCE BETWEEN INPUT
      C     RECORDS AND LINES OF OUTPUT.
      C
                    WRITE(6, 9240)
                    WRITE(16,9240)
       9240         FORMAT(/,' INPUT INCLUDES DATA RECORDS THAT CONTAIN',
           1               ' ONLY ZEROS (SPCSAM)')
                    TOTAL = TOTAL + 10
        270      CONTINUE
        290 CONTINUE
      C
      C*********************************************************************
      C
      C     SET UP OUTPUT SAMPLE COUNT
```

```
               C
               C***************************************************************************
               C
               C    INCLUDE CALIBRATION
               C
0043                 TOTAL = TOTAL + 6
               C
               C    ADJUST COUNT TO BE DIVISIBLE BY 4
               C
0044                 ADJUST = MOD(TOTAL, 4)
0045                 TOTAL = TOTAL - ADJUST
0046                 NSAMP = TOTAL
0047                 RETURN
0048                 END
```

APPENDIX D

```
          C    X10CM4
          C
          C*****************************************************************
          C
          C    X10CM4   CONTAINS ALL VARIABLES NEEDED TO REFORMAT EXOTECH MODEL
          C             100 DATA.  A SEPARATE COMMON BLOCK WAS SET UP TO KEEP THESE
          C             VARIABLES SEPARATE FROM THE VARIABLES USED FOR REFORMATTING
          C             OF EXOTECH MODEL 20C DATA.
          C
          C    WRITTEN 03/19/79 BY CATHERINE KOZLOWSKI
          C    REVISED 07/19/79 BY CATHERINE KOZLOWSKI
          C
          C*****************************************************************
          C
0001            BLOCK DATA
          C
0002            IMPLICIT INTEGER * 4 (A-Z)
          C
0003            COMMON /X10CM4/    TIMAFT,          TIMBEF,
          C     REAL * 4 VARIABLES
               1     DLRAFT(12),   DLRBEF(12),      DLRX10(12),     DLSX10(12),
               2     DRAFT(12),    DRBEF(12),       DRX10(12),      DSX10(12),
               3     GAIN(9),      GRAFT(12),       GRBEF(12),      GRX10(12),
               4     GSX10(12),    RRX10(12),
               5                   THRX10,          THSX10,         X10BND(12,2),
          C     INTEGER * 4 VARIABLES
               6     DLRCAL(4,2),  DRCAL(4,2),      CTYPE,          EXPARA(4),
               7     EXPARB(4),    EXPARC(4),       FNUMB,          INSTNA(4),
               8     INSUNT,       LNUMB,
               9     MAXBND,       MAXGAN,          NBAND,          NGAIN,
               A     NUMBER,       NUMINS,          PRESER,         PRT5(2),
               B     PTRR,         P5CONV,          RESSHE,
          C     LOGICAL * 1 VARIABLES
               C     BNDSKP(12),   DBGCIN,          DBGSBK,         DBGSCH,
               D     DBGSFW,       NEWCIN,          SKPOBS
          C
          C
          C
0004            REAL * 8           TIMAFT,          TIMBEF
          C
0005            REAL * 4           DLRAFT,
               1                   DLRBEF,          DLRX10,         DLSX10,
               2     DRAFT,        DRBEF,           DRX10,          DSX10,
               3     GAIN,         GRAFT,           GRBEF,          GRX10,
               4     GSX10,        REXPAA(4),       REXPAB(4),      REXPAC(4),
               5     RRX10,        THRX10,          THSX10,         X10BND
          C
0006            INTEGER * 4        DLRCAL,
               1                   DRCAL,           CTYPE,          EXPARA,
               2     EXPARB,       EXPARC,          FNUMB,          INSTNA,
               3     INSUNT,       LNUMB,
               4     MAXBND,       MAXGAN,          NBAND,          NGAIN,
               5     NUMBER,       NUMINS,          PRESER,         PRT5,
               6     PTRR,         P5CONV,          RESSHE
          C
0007            LOGICAL * 1        BNDSKP,
               1                   DBGCIN,          DBGSBK,         DBGSCH,
               2     DBGSFW,       NEWCIN,          SKPOBS
          C
0008            EQUIVALENCE (EXPARA(1), REXPAA(1)), (EXPARB(1),REXPAB(1)),
               1            (EXPARC(1), REXPAC(1))
          C
          C
          C
          C
          C*****************************************************************
          C*****************************************************************
          C
          C      X10CM4 VARIABLES
          C
          C*****************************************************************
```

```
      C
      C
      C     BNDSKP  FLAGS DATA VALUES TO SKIP WHEN PROCESSING AN
      C             OBSERVATION.  INITIALIZED HERE TO MEAN NO INFOR-
      C             MATION KNOWN YET.  RESET IN SUBROUTINE X10RCL FROM
      C             CALIBRATION SHEET.
      C
0009          DATA BNDSKP /12 * F/
      C
      C
      C     DBGCIN  DEBUGGING FLAG.  IF SET ON, DEBUG INFORMATION IS PRINTED
      C             FOR SUBROUTINE X10CIN.
      C
      C     DBGSBK  DEBUGGING FLAG.  IF SET ON, DEBUG INFORMATION IS PRINTED
      C             FOR SUBROUTINE X10SBK.
      C
      C     DBGSCH  DEBUGGING FLAG.  IF SET ON, DEBUG INFORMATION IS PRINTED
      C             FOR SUBROUTINE X10SCH.
      C
      C     DBGSFW  DEBUGGING FLAG.  IF SET ON, DEBUG INFORMATION IS PRINTED
      C             FOR SUBROUTINE X10SFW.
      C
      C     DLRCAL  DARK LEVEL CALIBRATION INSTRUCTIONS.  INITIALIZED
      C             HERE TO MEAN NO INFORMATION KNOWN YET.  RESET IN THE
      C             SUBROUTINE X10RC ACCORDING TO THE CALIBRATION SHEETS.
      C
0010          DATA DLRCAL /8*0/
      C
      C
      C     DLRAFT  DARK LEVEL REFERENCE FOR REFERENCE STANDARD.  USED ONLY
      C             FOR CALIBRATION CODE 23.  OBTAINED FROM SUBROUTINE
      C             X10REF ACCORDING TO CALIBRATION INSTRUCTIONS.
      C
      C     DLRBEF  DARK LEVEL REFERENCE FOR REFERENCE STANDARD.  USED ONLY FOR
      C             CALIBRATION CODE 23.  OBTAINED FROM SUBROUTINE X10REF
      C             ACCORDING TO CALIBRATION INSTRUCTIONS.
      C
      C     DLRX10  DARK LEVEL REFERENCE FOR REFERENCE STANDARD.  OBTAINED FROM
      C             SUBROUTINE X10REF ACCORDING TO CALIBRATION INSTRUCTIONS.
      C
      C     DLSX10  DARK LEVEL REFERENCE FOR SCENE.  OBTAINED FROM
      C             SUBROUTINE X10CLM.
      C
      C     DRCAL   REFERENCE STANDARD CALIBRATION INSTRUCTIONS.  INITIALIZED
      C             HERE TO MEAN NO INFORMATION KNOWN YET.  RESET IN THE
      C             SUBROUTINE X10RCL ACCORDING TO THE CALIBRATION SHEETS.
      C
0011          DATA DRCAL /8*0/
      C
      C
      C     DRAFT   REFERENCE STANDARD DATA RESPONSES.  USED ONLY FOR
      C             CALIBRATION CODE 23.  OBTAINED FROM X10REF ACCORDING TO
      C             CALIBRATION INSTRUCTIONS.
      C
      C     DRBEF   REFERENCE STANDARD DATA RESPONSES.  USED ONLY FOR
      C             CALIBRATION CODE 23.  OBTAINED FROM X10REF ACCORDING TO
      C             CALIBRATION INSTRUCTIONS.
      C
      C     DRX10   REFERENCE STANDARD DATA RESPONSES.  OBTAINED FROM
      C             SUBROUTINE X10REF ACCORDING TO CALIBRATION INSTRUCTIONS.
      C
      C     DSX10   SCENE DATA RESPONSES.  OBTAINED FROM RESPONSE
      C             SHEET.  (ALSO CONTAINS PRT-5, EXPARA, EXPARB, AND
      C             EXPARC RESPONSES.)
      C
      C     DTYPE   DATA TYPE FOR SCENE.  INITIALIZED HERE TO MEAN NO
      C             INFORMATION KNOWN YET.  RESET IN THE SUBROUTINE X10RRS
      C             FROM DATA RESPONSE SHEETS.
      C
0012          DATA DTYPE /'    '/
      C
```

```
            C
            C      EXPARA   ARRAY CONTAINING INFORMATION FOR AN EXPERIMENTER PARAMETER.
            C               EXPARA(1) CONTAINS THE NUMBER OF THE ID ELEMENT FOR THIS
            C               EXPERIMENTER PARAMETER.  EXPARA(2) CONTAIN THE NUMBER OF THE
            C               DATA RESPONSE CHANNEL FROM WHICH THE VALUE SHOULD COME.
            C               EXPARA(3) CONTAINS THE PRT-5 CONVERSION TABLE TO USE WITH
            C               THE REPONSE.  EXPARA(4) STORES THE CALCULATED
            C               VALUE IN REAL FORMAT.  EXPARA(1) THROUGH
            C               (3) OBTAINED FROM THE SUBROUTINE X1CPRS.  EXPARA(4)
            C               SET IN SUBROUTINE X1CCLM.
            C               NOTED      IF THE EXPERIMENTER PARAMETER VALUE IS ALSO IN THE
            C               AGRONOMY SHEETS THE AGRONOMY SHEET VALUE IS USED.
            C
            C      EXPARB   SAME AS EXPARA.
            C
            C      EXPARC   SAME AS EXPARA
            C
            C      FNUMB    NUMBER OF FIRST OBSERVATION TO WHICH THE CURRENT CALIBRATION
            C               INSTRUCTIONS APPLY.  OBTAINED FROM SUBROUTINE X1ORCL
            C               ACCORDING TO CALIBRATION SHEETS.
            C
            C      GAIN     GAINS TO USE IN REFORMATTING DATA, INDEXED BY GAIN CODE.
            C               INITIALIZED HERE TO MEAN NO INFORMATION KNOWN YET.
            C               RESET IN SUBROUTINE X1ORSS ACCORDING TO RESPONSE SHEETS.
            C
0013                DATA GAIN /9*0.0/
            C
            C      GRAFT    GAIN USED FOR REFERENCE STANDARD.  USED ONLY FOR CALIBRATION
            C               CODE 23.  OBTAINED FROM SUBROUTINE X1OREF ACCORDING TO
            C               CALIBRATION INSTRUCTIONS.
            C
            C      GRBEF    GAIN USED FOR REFERENCE STANDARD.  USED ONLY FOR CALIBRATION
            C               CODE 23.  OBTAINED FROM SUBROUTINE X1OREF ACCORDING TO
            C               CALIBRATION INSTRUCTIONS.
            C
            C      GRX10    GAIN USED FOR REFERENCE STANDARD.  OBTAINED FOR THE
            C               SUBROUTINE X1OREF ACCORDING TO CALIBRATION INSTRUCTIONS.
            C
            C      GSX10    GAIN USED FOR THE SCENE.  OBTAINED FROM DATA RESPONSE
            C               SHEET.
            C
            C      INSTNA   THE INSTRUMENT NAME OBTAINED FROM THE SUBROUTINE X1ORSS.
            C
            C      INSUNT   UNIT NUMBER FOR INSTRUMENT CODE TABLE.
            C
0014                DATA INSUNT /17/
            C
            C      LNUMB    NUMBER OF LAST OBSERVATION TO WHICH THE CURRENT CALIBRATION
            C               INSTRUCTIONS APPLY.  OBTAINED FROM X1ORCL ACCORDING TO
            C               CALIBRATION SHEETS.
            C
            C      MAXBND   MAXIMUM NUMBER OF BANDS THE SYSTEM CAN HANDLE.
            C
0015                DATA MAXBND /12/
            C
            C      MAXGAN   MAXIMUM NUMBER OF GAIN CODES.
            C
0016                DATA MAXGAN /9/
            C
            C      NBAND    NUMBER OF BANDS OF DATA RESPONSES.
            C
            C      NGAIN    NUMBER OF GAINS TO USE FOR THIS SET OF DATA.
            C
            C      NEWCIN   FLAG TO INDICATE A NEW SET OF CALIBRATION
            C               INSTRUCTIONS.  SET IN THE SUBROUTINE X1ORCL AND RESET IN
            C               THE SUBROUTINE X1OREF.
            C
```

```
            C
            C       NUMBER   NUMBER (8 DIGITS) CF THE OBSERVATICN CURRENTLY BEING
            C                REFORMATTEC.  OBTAINED FOR THE SUBRCLTINE X10RFR.
            C
            C       NUMINS   NUMBER OF DARK LEVEL OR REFERENCE STANDARC INSTRUCTICNS.
            C
)017                 CATA NUMINS /4/
            C
            C
            C       PRESER   CONTAINS THE PREFIX NUMBERS FOR THE ID SERIAL NUMBER
            C                OBTAINEE FROM THE SUBROUTINE X1CRSS ACCORDING TC THE DATA
            C                RESPCNSE SHEETS.
            C
            C       PRT5(2)  ARRAY CCNTAINING INFORMATION FOR PRT5 DATA VALUE.
            C                PRT(1) CONTAINS THE NUMBER CF DATA RESPCNSE CHANNELS.
            C                PRT(2) CONTAINS THE  PRT-5 CCNVERSICN TABLE.
            C
            C       PTRR     POINTER INTO RESPONSE SHEET FILE.
            C
            C       P5CONV   UNIT NUMBER FOR PRT5 X100 DISK FILE.
            C
)018                 DATA P5CONV /24/
            C
            C
            C       RESSHE   EISK UNIT NUMBER FCR RESPONSE SHEET FILE.
            C
)019                 CATA RESSHE /35/
            C
            C
            C       RRX10    SPECTRAL BIDIRECTICNAL REFLECTANCE FACTOR OF REFERENCE
            C                STANCARC.  OBTAINEC FROM SUBROUTINE X10TAB FROM
            C                DISK FILE RFLSTND X100.
            C
            C       SKPOBS   FLAG TO INDICATE OBSERVATION SHCULD NOT BE PRCCESSEC.
            C                SET IN X10REF.
            C
            C       THRX10   SOLAR ZENITH ANGLE AT TIME CF OBSERVATICN CF REFERENCE
            C                STANCARC.  OBTAINEC FROM SUBROUTINE X10REF.
            C
            C       THSX10   SOLAR ZENITH ANGLE AT TIME CF OBSERVATICN OF SCENE.
            C                OBTAINEE FROM SUBRCUTINE X1CREF.
            C
            C       TIMAFT   TIME OF REFERENCE RESPCNSE AFTER THE SCENE RESPCNSE.
            C                USED ONLY FOR CALIBRATICN CCDE 23.  INITIALIZEC
            C                HERE TO MEAN NO INFCRMATION KNOWN YET.  RESET
            C                IN THE SUBROUTINE X10REF.
            C
)020                 CATA TIMAFT /C.0/
            C
            C
            C       TIMBEF   TIME OF REFERENCE RESPCNSE BEFORE THE SCENE
            C                RESPCNSE.  USED ONLY FCR CALIBRATICN CCDE 23.  INITIALIZED
            C                HERE TO MEAN NO INFORMATION KNOWN YET.  RESET IN THE
            C                SUBRCUTINE X10REF.
            C
)021                 CATA TIMBEF /999.9/
            C
            C
            C       X10BND   LOWER AND UPPER LIMITS FOR EACH BAND CF EXCTECK
            C                MODEL 1C0 DATA.  OBTAINED IN THE SUBROUTINE X10RSS FROM
            C                THE INSTRUMENT CODE DISK TABLE.
            C
            C***********************************************************************
            C
            C
)022                 END
```

Appendix A-4

Control Card Reference File

and

Program Abstracts

for

SMOOTHRESULTS

and

CHANGEDETECTION

Programs

SMOOTHRESULTS

R E Q	KEY WORD(COL 1)	CONTROL PARAMETER	FUNCTION	DEFAULT
*	*SMOOTHRESULTS	(NONE)	SELECT THE CLASSIFICATION RESULTS POST-PROCESSOR THAT REPLACES GROUPS OF POINTS WITH THE DOMINANT CLASS.	(NONE)
*	INRESULTS		LOCATION OF INPUT RESULTS	(NONE)
		TAPE(XXX) FILE(FF)	LOCATED ON TAPE XXX FILE FF	
		DISK	USE RESULTS PLACED ON DISK BY CLASSIFYPOINTS IN CURRENT TERMINAL SESSION.	
	CELLSIZE	LL,CC	DEFINE THE CELL DIMENSIONS. LL IS THE NUMBER OF LINES. CC IS THE NUMBER OF COLUMNS. MAXIMUM SIZE IS 10 X 10.	(2X2)
*	OUTRESULTS		DESTINATION OF RESULTS	(NONE)
		TAPE(XXX) FILE(FF)	PUT ON TAPE XXX, FILE FF.	
		INITIALIZE	INITIALIZE FILE ONE OF A NEW RESULTS TAPE (REQUIRED WHEN USING A NEW TAPE).	SEE CONTROL CARD DICTIONARY
		DISK	RESULTS WILL BE STORED ON LARSYS DISK.	
	PRIORITY	G1,G2,...	PRIORITY GROUPS G1,G2,... WILL NOT BE REPLACED WHEN THE CELL IS MODIFIED.	(NONE)
	GROUP	NAME(G1/P1,P2/)	GROUP CLASSIFICATION POOLS P1,P2,... FOR CALCULATING CORRECT RECOGNITION. 'NAME,' IS THE GROUP NAME AND G1 IS THE GROUP NUMBER.	NO GROUPING
	WEIGHTS	W1,W2,...	ASSIGN WEIGHTS TO POOLS IN THIS ORDER.	EQUAL WEIGHTS USED
	BLOCK	RUN(XXXXXXXX)	DATA FROM RUN XXXXXXXX IS REQUESTED.	ENTIRE AREA
		LINE(X,Y,Z)	DATA FROM LINE X TO Y WITH INTERVAL Z	
		COL(X,Y,Z)	DATA FROM COLUMN X TO Y WITH INTERVAL Z.	
*	END	(NONE)	END OF FUNCTION	(NONE)

MODULE IDENTIFICATION

Module Name: SMOSUP **Function Name:** SMOOTHRESULTS

Purpose: Supervisor for SMOOTHRESULTS

System/Language: CMS/FORTRAN

Author: John Cain **Date:** 4/18/80

Latest Revisor: **Date:**

MODULE ABSTRACT

SMOSUP supervises the SMOOTHRESULTS function by calling two subroutines: one to read the control cards, and one to modify the input results file.

1. Module Usage

CALL SMOSUP

No arguments are used in the call to SMOSUP. This SMOOTHRESULTS supervisor routine is called from LARSMN. Upon completion of the function, control is returned to LARSMN.

2. Internal Description

SMOSUP contains two common blocks - GLOCOM and SMOCOM. This supervisory routine calls SMORDR to read in the function control cards. After the cards have been interpreted, SMOSUP calls SMOINT which initiates the modification of an area. When SMOINT returns control to SMOSUP, the supervisor indicates that the function is completed and returns control to LARSMN. Subroutines called by SMOSUP:

SMORDR
SMOINT

3. Input Description

Not applicable.

4. Output Description

Standard supervisor information messages.

5. Supplemental Information

Not applicable.

6. Flowchart

Not applicable.

LARS Program Abstract

MODULE IDENTIFICATION

Module Name: SMOCOM Function Name: SMOOTHRESULTS

Purpose: Block common for SMOOTHRESULTS module

System/Language: CMS/FORTRAN

Author: John Cain Date: 4/18/80

Latest Revisor: Date:

MODULE ABSTRACT

PURDUE UNIVERSITY
Laboratory for Applications of Remote Sensing
1220 Potter Drive
West Lafayette, Indiana 47906

77 LARS Program Abstract ____________

MODULE IDENTIFICATION

Module Name: SMOINT **Function Name:** SMOOTHRESULTS

Purpose: Modify the input Classification Results File

System/Language: CMS/FORTRAN

Author: John Cain **Date:** 4/18/80

Latest Revisor: ____________ **Date:** ____________

MODULE ABSTRACT

SMOINT reads the input Classification Results File "n" lines at a time (where n is the number of lines in a 'cell'), reassigns class numbers to the dominant class (depending on user-input parameters) and writes out a modified Classification Results File.

1. Module Usage

SMOINT

CALL SMOINT

There are no arguments to SMOINT, which is called by SMOSUP and returns control to SMOSUP when the function terminates.

2. Internal Description

SMOINT reads from the input classification Results File and modifies each record type in the following way:

RECORD TYPE 1 - Record type 1 is read and checked to see whether it is valid (i.e., not a startup file). The new tape and file numbers are written onto the output Classification Results File.

RECORD TYPE 2 - The number of classification pools is changed to the number of group names and the pool pointer and stack arrays are changed accordingly. If no grouping was done, the pools are then considered classes in themselves. If there are any weights, then they are checked and normalized before written onto the output results file.

RECORD TYPE 3 - The first card of the statistics deck is marked indicating that the deck is invalid due to execution of the *SMOOTHRESULTS function. This deck is then copied onto the output results file.

RECORD TYPE 4 - unchanged.

RECORD TYPE 5 - SMOINT checks the area requested by the user to be sure it exists. If only part of the area requested exists, then the user is given the option to continue or terminate the function. If the user continues, the parameters are changed so that they are now valid. These are written to the output results RECORD TYPE 5. If no BLOCK card was used this record remains unchanged.

RECORD TYPE 6 - This record is read into buffers "n" lines at a time, (where "n" is the number of lines in a cell) and shifted until only the columns requested are considered. SMOINT then calls SMOOTH to modify the buffer data. Upon return, SMOINT writes the modified lines to the output Classification Results File.

RECORD TYPES 7, 8: These record types are created according to the required Classification Results File format. A final record TYPE 1 is written, all tapes are detached and control returns to SMOSUP. Subroutines called by SMOINT:

SMOOTH

3. Input Description

SMOINT reads the first 6 record types off of a LARSYS Classification Results File. It also checks the weights card and can read in a corrected version if required.

4. Output Description

Several information messages may be issued. They are as follows:

a) I*** This is a restart file -- Run Classifypoints first.
b) I*** File length is only one record -- Try running Listresults.
c) I*** Area requested only partially within this classification area. Do you wish to continue?
d) I*** Execution terminated by user. Do not consider partial area.

All eight record types are written to either tape or disk in the LARSYS Classification Results File format.

5. Supplemental Inforamtion

Not applicable.

6. Flowchart

Not applicable.

MODULE IDENTIFICATION

Module Name: SMORDR Function Name: SMOOTHRESULTS

Purpose: Read and interpret control cards for SMOOTHRESULTS

System/Language: CMS/FORTRAN

Author: John Cain Date: 4/18/80

Latest Revisor: ____________ Date: ____________

MODULE ABSTRACT

SMORDR reads and interprets all control cards for SMOOTHRESULTS and prints out a summary of the user's requests.

1. Module Usage

SMORDR

CALL SMORDR

There are no arguments to SMORDR. SMORDR interprets the following function control cards and takes the indicated actions:

CONTROL CARD	ACTION
INRESULTS	INRES is set to the correct DSRN for tape or disk. If it is a tape that is requested, MMTAPE is called to mount and position it.
CELLSIZE	CELROW is set equal to the number of lines per cell (length) and CELCOL is set to the number of columns (width).
OUTRESULTS	OUTRES is set to the correct DSRN for tape or disk. If it is a tape, MMTAPE is called to mount and position it.
PRIORITY	The vector PCLASS is filled with user defined priority classes.
GROUP	GRPNAM and GRPSTK are computed by a call to GRPSCN.
WEIGHTS	These weights are read into PROB, a REAL*4 vector of dimension GO.
BLOCK	The first 6 words of array BLOCK are used to contain this field boundary definition and the run number is put in RUNNUM.
MIXCLASS	The array MIXDAT is filled with the low and high range percentage values to be later compared to each group in each cell. The array is filled by a call to READMX.

2. Internal Description

SMORDR initializes necessary variables and then uses CTLWRD to interpret the keyword on each card. An unexpected end of file for the control card input causes a call to ERPRNT and termination of the function. Once the keyword has been interpreted, SMORDR uses CTLPRM, IVAL, and FVAL to further interpret the card. If a disk is requested the function TSPACE is used to calculate the amount of available space on the disk. Execution is terminated if this amount is not more than 20% greater than the amount required. READMX is called to read the MIXCLASS card and passes back the array of percentages.

After the END card is read, checks are made to determine whether user requests are valid and complete.

Subroutines called by SMORDR: MMTAPE READMX FVAL CTLWRD CTLPRM TSPACE ERPRNT IVAL

3. Input Description

Control cards are read by calls to CTLWRD. If a disk is used, it is first checked to make certain that the file exists.

4. Output Description

"Options chosen" messages are typed in addition to requests for more information.

5. Supplemental Information

Not applicable.

6. Flowchart

Not applicable.

 LARS Program Abstract ______

MODULE IDENTIFICATION

Module Name: READMX Function Name: SMOOTHRESULTS

Purpose: Interprets the MIXCLASS control card

System/Language: CMS/FORTRAN

Author: John Cain Date: 4/18/80

Latest Revisor: ______ Date: ______

MODULE ABSTRACT

READMX interprets the MIXCLASS control card(s) for the SMOOTHRESULTS function.

1. Module Usage

READMX

CALL READMX (LCARD, COL)

Input Arguments:

LCARD - L*1, Card image of the MIXCLASS card being interpreted.

COL - I*4, the column number preceding the first name on the MIXCLASS card (i.e., the first nonblank after the keyword).

Output Arguments: (passed in SMOCOM)

NCLNAM- I*4, NCLNAM is a two dimensional vector that is filled with the new class names from the MIXCLASS card.

NEWCLS- I*4, the number of new classes. NEWCLS is incremented as each new name is read.

MIXDAT- I*4, MIXDAT is filled with the percentage ranges specified on the MIXCLASS card. MIXDAT (I,1) is the lower boundary of the percentage range and MIXDAT (I,2) is the higher boundary.

2. Internal Description

READMX can be reduced to several functions--stripping off the name, scanning and modifying the user supplied ranges, and the reading and loading of the ranges into MIXDAT.

The first function is performed by using LOCATE to find the left parenthesis of the class being interpreted and then by a call to BCDFIL. The data for this particular name is then scanned and the positions of all hyphens are noted by setting a corresponding flag in a flag vector. After a hyphen is read it is changed to a comma so that once the data is finished being scanned it can be read by IVAL. Once IVAL has been called the flag vector is used to transfer the data read into the array MIXDAT. This is done by placing the first data value into MIXDAT (I,1) (i.e. the lower boundary of the user defined percentage range) and then checking the corresponding flag in the flag vector. If the flag was not set, then this lower boundary is simply copied into the upper boundary, MIXDAT (I,2) If the flag was set then the next data value is treated as the upper boundary. This is continued until all values have been loaded.

Subroutines called by READMX: LOCATE
BCDFIL
IVAL

3. Input Description

READMX checks for various syntax errors on the MIXCLASS card. It is therefore capable of requesting a new MIXCLASS card by using CTLWRD.

4. Output Description

READMX can write a syntax error message to the user. This gives the user the approximate location of the error and asks for the card to be retyped.

5. Supplemental Information

Not applicable.

6. Flowchart

Not applicable.

MODULE IDENTIFICATION

Module Name: SMOOTH Function Name: SMOOTHRESULTS

Purpose: Compute the dominant class in each cell and reclassify

System/Language: CMS/FORTRAN

Author: John Cain Date: 4/18/80

Latest Revisor: Date:

MODULE ABSTRACT

SMOOTH takes "n" lines of a Classification Results File (where n is the number of lines in a cell), determines which group or class makes up the largest weighted percentage of each cell, and reloads the cell with that class. User-defined priority classes remain unchanged.

1. Module Usage

CALL SMOOTH (NOCOLS)

Input Arguments:

NOCOLS - I*4, NOCOLS is the number of columns (i.e. the buffer length) that is to be segmented into individual cell widths.

CELCOL - I*4, the number of columns in the user defined cell.

CELROW - I*4, the number of rows in the user defined cell.

2. Internal Description

Upon entry, SMOOTH uses NOCOLS and the user defined parameters CELCOL and CELROW to determine the total number of cells to be processed. Using this information the main program loop is entered. The main loop begins by tallying the number of samples of each group in the cell. If the "weight" option is in effect then each group's tally is weighted. The total weight is then found by summing all the weighted values. Each group weight is then divided by the total weight to obtain a cell percentage for its group. These percentages are compared to user supplied group percentages from the MIXCLASS card. If the values fall within the MIXCLASS ranges, then all the samples of the cell are changed to this MIXCLASS group and rewritten into the main buffer. Any user defined priority samples (from the PRIORITY card) will not be altered.. If none of the MIXCLASS ranges correspond to the calculated percentages, then the nonpriority group with the greatest percentage must be determined. For efficiency the groups not found in the cell are removed from the percentage array (compressed). The array is then sorted so that the group with the greatest percentage is first and the one with the smallest is last. The number of the first group found that is not a priority class is then loaded into a pointer array. Groups defined as priority are loaded (overlayed) into their corresponding position in this array. The cell is then reloaded by using this pointer array for indexing. This loop is repeated for each CELCOL group of columns across the input Classification Results lines. When all cells have been modified, control is returned to SMOINT.

3. Input Description

Not applicable.

4. Output Description

Not applicable.

5. Supplemental Information

The buffer of samples modified by SMOOTH is passed through GLOCOM in ARRAY. It's dimensions are CELROW by NOCOLS.

6. Flowchart

Not applicable.

CHANGEDETECTION

NEW	KEY WORD(COL.1)	CONTROL PARAMETER	FUNCTION	DEFAULT
•	*CHANGEDETECT	(NONE)	SELECT CHANGE DETECTION FUNCTION	(NONE)
•	BASERESULTS		LOCATION OF RESULTS FROM FIRST DATE.	(NONE)
		TAPE(XXX)	LOCATED ON TAPE XXX.	(SEE CONTROL CARD DICTIONARY)
		FILE(FF)	FILE FF.	
		DISK	USE RESULTS PLACED ON DISK IN CURRENT TERMINAL SESSION.	
•	COMPARERESULTS		LOCATION OF RESULTS FROM SECOND DATE.	(NONE)
		TAPE(XXX)	LOCATED ON TAPE XXX.	
		FILE(FF)	FILE FF.	
		DISK	USE RESULTS PLACED ON DISK IN CURRENT TERMINAL SESSION.	
•	NEWRESULTS		LOCATION OF CHANGE (OUTPUT) RESULTS FILE.	(NONE)
		TAPE(XXX)	WRITE ON TAPE XXX.	
		FILE(FF)	FILE FF.	
		INIT	INITIALIZE TAPE AND WRITE RESULTS IN FILE 1	
		DISK	PLACE RESULTS ON DISK	
•	BLOCK	RUN(XXXXXXXX)	RUN NUMBER IS XXXXXXXX	(NONE)
		LINES(X,Y,Z)	DISPLAY LINES X TO Y WITH LINE INTERVAL Z	(NONE)
		COL(X,Y,Z)	DISPLAY COLUMNS X TO Y WITH COLUMN INTERVAL Z	(NONE)
•	DATA	-----START OF DATA DECK----- (see below)		
•	END	(NONE)	END OF FUNCTION.	(NONE)

DATA deck:

```
DEFINE CHANGE CLASSES OF INTEREST BY NAMING THE CLASS
  ON A 'CLASS' CARD, AND INDICATING WHICH CLASSES
  FROM THE FIRST CLASSIFICATION (ON 'BASE' CARD) AND
  WHICH CLASSES FROM THE SECOND CLASSIFICATION (ON
  'COMP' CARD) ARE PERMITTED FOR A POINT TO BELONG
  TO THIS CLASS.

CLASS NAME1
BASE N1, N2, N3, ....
COMP M1, M2, M3, ...
CLASS NAME2
      .
      .
      .
      ETC        (WHERE M1,M2,... AND N1,N2,... ARE CLASS OR
                 (POOL NUMBERS FROM CLASSIFYPOINTS.
```

LARS Program Abstract 1058

MODULE IDENTIFICATION

Module Name: CHASUP Function Name: CHANGEDETECTION

Purpose: Supervisor for the CHANGE DETECTION function.

System/Language: CMS/Fortran

Author: John Cain Date: 6/1/79

Latest Revisor: Date:

MODULE ABSTRACT

Supervisor for the CHANGEDETECTION function.

1. Module Usage

CALL CHASUP

There are no arguments to CHASUP. It is called from LARSMN when the CHANGEDETECTION function is requested. Control returns to LARSMN upon completion of the function.

2. Internal Description

CHASUP receives control from LARSMN to perform the CHANGEDETECTION processing. CHASUP calls CHARDR to read and interpret the control cards. Upon return from CHARDR, CHASUP calls CHANGE to finish the processing. Subroutines called by CHASUP: CHARDR
CHANGE

3. Input Description

Not applicable.

4. Output Description

Two informational messages, CHANGEDETECTION FUNCTION REQUESTED and CHANGE FUNCTION COMPLETED, are written at the typewriter.

5. Supplemental Information

Not applicable

6. Flowchart

Not applicable

 LARS Program Abstract 1059

MODULE IDENTIFICATION

Module Name: CHACOM **Function Name:** CHANGEDETECTION

Purpose: Block data

System/Language: CMS/Fortran

Author: John Cain **Date:** 6/1/79

Latest Revisor: **Date:**

MODULE ABSTRACT

This is the BLOCK DATA subroutine for the CHANGEDETECTION common block CHACOM

 LARS Program Abstract 1060

MODULE IDENTIFICATION

Module Name: CHARDR **Function Name:** CHANGEDETECTION

Purpose: Reads and interprets function control cards.

System/Language: CMS/Fortran

Author: John Cain **Date:** 6/1/79

Latest Revisor: **Date:**

MODULE ABSTRACT

CHARDR interprets all function control cards for CHANGEDETECTION. Checks are made for complete and valid specifications and the proper input-output devices are attached.

1. Module Usage

CALL CHARDR(Z,NAME)

Output Arguments:

Z-LOGICAL*1 each element initialized to .FALSE.

Z(i,1,j)=.TRUE. - if class i from the base classification is part of user defined class j.

Z(m,2,n)=.TRUE. - if class m from the 2nd (compared) classification is part of user-defined class n.

NAME-I*4 - contains the names of the user-defined classes

Listed below are the actions taken when the following control cards are read.

BASERESULTS (COMPARERESULTS)	TAPE - the variable TAPE1 (TAPE2) is set to the given tape number. FILE - the variable FILE1 (FILE2) is set to the given file number. DISK - DISKFG is checked to be sure that the DISK option is not already in effect, the tape and file numbers are checked to be sure that both the DISK option and TAPE option are not being used simultaneously. If they are, then an error message will be printed and the DISK will be used. RESLT1 (RESLT2) is set equal to CLASSR.
NEWRESULTS	TAPE - the variable TAPE3 is set equal to the given tape number. FILE - the variable FILE3 is set equal to the given file number. INIT - the variable INITFG is set equal to 1. DISK - the same checks are made as above in addition to a check to see whether the INIT and DISK option were used simultaneously. DISKFG is set equal to one and RESLT3 is set equal to CLASSR.
BLOCK	RUN - the variable RUNNUM is set equal to the given run number. LINE - STALIN is set equal to the first entry (the starting line of the area to be investigated). LASLIN (last line) is set equal to the second entry and finally LININT (line interval) is set equal to the last entry. COL - same as above where the variables are: STACOL - first entry, LASCOL - second entry and COLINT - final entry.
DATA	A check is made for the presence and validity of all information.
CLASS name	The name given is stored in the array NAME.

BASE N1,N2...	Using the given class numbers the appropriate locations in
COMP M1,M2...	the Z(64,2,64) array are set .TRUE.

(i.e. if these are the BASE and COMP cards for the j^{th} user-defined CLASS, then the following assignments are made for array Z:

Z(N1,1,j) = .TRUE.
Z(N2,1,j) = .TRUE.
:
and
Z(M1,2,j) = .TRUE.
Z(M2,2,j) = .TRUE.
:

2. Internal Description

CHARDR uses the standard card reader logic in using CTLWRD, CTLPRM and IVAL to read and interpret control cards.

CHARDR begins by initializing all flags and arrays that are used to convey control card information. It then goes into a loop of reading and interpreting the input specifications and the BLOCK card. When the DATA card is read CHARDR checks for the presence of all information and its validity. Another loop is entered and the CLASS cards and their corresponding BASE and COMP cards are read. The class numbers from the BASE and COMP cards are used to set appropriate values in the Z array to a logical .TRUE.

Z(i,1,j)=.TRUE. if the Ith class from the base results file is part of user-defined class j.

Z(k,2,m)=.TRUE. if class k from the compared results file is part of user-defined class m

This loop is exited when an END card is read. Once this card is read, CHARDR calls CHTAPE to mount the specified tapes. If a disk was specified as an input device, CHARDR first checks to be certain both a tape and disk were not specified for a single input. It then reads from the results file to be sure it exists on the disk. If a disk was specified as an output device, checks are made to be sure there is sufficient space for the output results. TSPACE makes a search for a larger disk if necessary. CHARDR finally returns control back to CHASUP. Subroutines called by CHARDR:

CTLWRD	CTLPRM	TSPACE
BCDFIL	CHTAPE	RTMAIN
IVAL	ERPRNT	

3. Input Description

Function control cards for CHANGEDETECTION are read via CTLWRD.

4. Output Description

Control card error messages are written via ERPRNT.

5. Supplemental Information

Not applicable.

6. Flowchart

Not applicable.

LARS Program Abstract 1061

MODULE IDENTIFICATION

Module Name: CHANGE **Function Name:** CHANGEDETECTION

Purpose: Compares two classification results files and outputs the compared results.

System/Language: CMS/Fortran

Author: John Cain **Date:** 6/1/79

Latest Revisor: **Date:**

MODULE ABSTRACT

CHANGE is the main subroutine for CHANGEDETECTION. It reads from two input tapes (or one disk and one tape), calls COMPAR, then outputs the data in standard LARSYS classification results file format to tape or disk.

1. Module Usage

CHANGE

CALL CHANGE(Z,NAME)

Input Arguemnts:

Z-Logical*1 - Z(i,1,j)=.TRUE. if class i from the base classification is part of user-defined class j.

Z(m,2,n)=.TRUE. if class m from the 2nd (compared) classification is part of user-defined class n.

NAME - I*4 - contains the names of the user-defined classes.

Output Arguments:

Not applicable.

2. Internal Description

CHANGE first reads the file numbers from the input tapes, and the tape numbers passed through the common block, and creates a code that takes the place of the CLASSIFICATION STUDY number. The code format is the base tape and file numbers followed by the compared tape and file numbers. CHANGE then reads the area identification record (record type 5) from both input sources and checks to see whether they are valid for the given BLOCK CARD; if not, appropriate error messages are printed and the function is terminated. Record types 1-5 are written to the output tape (DISK). The inputs are positioned to the correct line number and shifted to the correct column number. CHANGE then calls COMPAR to determine which class each point belongs to and this information is used to create file type 5. Finally record types 7 and 8 are written and control is returned to CHASUP. If the output device is a tape, then a final record type 1 and END OF FILE Mark are written before returning to the supervisor. Subroutines called by CHANGE:

COMPAR
RTMAIN
TAPOP

3. Input Description

Record types 1, 5, 6 of the LARSYS classification results files are read from the two input devices, RESLT1 and RESLT2. One of these may be a disk (DSRN CLASSR). Tape drives 181 (CPYOUT) and 182 (COMPTP defined in CHACOM) are used as inputs.

4. Output Description

The output device RESLT3 initially has a DSRN of MAPTAP. If a disk is used (only if one is not used for input), the DSRN is changed to CLASSR. Tape drive 180 is used for output to facilitate the run of PRINTRESULTS on the output data immediately after the CHANGEDETECTION run. The output is a classification results file in standard LARSYS format.

5. Supplemental Information

Not applicable.

6. Flowcnart

Not applicable.

LARS Program Abstract 1062

MODULE IDENTIFICATION

Module Name: COMPAR **Function Name:** CHANGE

Purpose: Compare 2 lines of classification results

System/Language: CMS/FORTRAN

Author: Susan Schwingendorf **Date:** 3/28/79

Latest Revisor: **Date:**

MODULE ABSTRACT

COMPAR compares two lines of classification results (presumably from two different classifications which are registered to each other) against user defined change classes in a logical array, and writes the output class number in the output vector.

1. Module Usage

CALL COMPAR (NCOLS, NCLASS, Z, BUFF1, BUFF2, BUFF3)

Input Arguments:

NCOLS - INTEGER*4, the number of columns of classified data.

NCLASS - INTEGER*4, the number of classes defined by the user in array Z.

Z - LOGICAL*1 (64, 2, 64) array containing user defined change classes. (Initialized to .FALSE.)

Z(I,1,K) = .TRUE.
Z(J,2,K) = .TRUE. } means a point in class

I from classification 1 (BUFF1) and in class J on classification 2 (BUFF2) should be assigned to class K in BUFF3.

BUFF1 - LOGICAL*1 (2*NCOLS + 4) vector containing classified data from first classification. First full word is line number. Then the second byte of each halfword contains the next class number

BUFF2 - LOGICAL*1 (2*NCOLS + 4) vector containing classified data from the second classification. First full word (4 bytes) is the line number. Then the second byte of each halfword contains the next class number.

Output Arguments:

BUFF3 - LOGICAL*1 (2*NCOLS + 4) vector of change classes for this line. The first full word contains the line number. Then the second byte of each halfword contains the assigned change class number.

2. Internal Description

The line number is written in the first word of BUFF3. The next class number is then extracted from BUFF1 and BUFF2 and assigned to integer variables CLASS1 and CLASS2. A loop through the logical array Z determines which output class to assign this point to. If Z (CLASS1,1,J) and Z (CLASS2,2,J) are true. then the point is assigned to class J. If it belongs to none of the defined output classes, then it is assigned a class number NCLASS+1. The output class numbers are written in BUFF3.

3. Input Description

Not applicable

4. Output Description

Not applicable

5. Supplemental Information

Not Applicable

6. <u>Flowchart</u>

Not Applicable

LARS Program Abstract 1063

MODULE IDENTIFICATION

Module Name: CHTAPE **Function Name:** CHANGEDETECTION

Purpose: Mounts and positions results tapes

System/Language: CMS/Fortran

Author: E.M. Rodd **Date:** 9/5/72

Latest Revisor: J. Cain **Date:** 6/1/79

MODULE ABSTRACT

CHTAPE mounts and positions the results tape (or a tape to be used as output for copying results files).

1. Module Usage

CHTAPE

CALL CHTAPE (RQTAPE,RQFILE,MODE,UNIT)

Input Arguments:

RQTAPE - I*4, Tape number of requested tape. A tape number of 0 is a request for a scratch tape.

RQFILE - I*4, File number of requested file. If RQFILE is = 0, then the tape will be initialized by writing a record type 1 on the results tape with filetype = 0.

MODE - I*4, Flag indicating usage of CHTAPE. MODE = -1 indicates CHTAPE has been called to mount and position a tape to be used for copying results files onto. Mode = 0 indicates that a results tape is being mounted for reading a results file. In this case, the tape is mounted ring out. Also, if MODE = 0, RQFILE = 0 is invalid and will cause an error when an attempt is made to write on the tape. MODE = 1 indicates a tape is being mounted for writing a new results file (or continuing a suspended classification). The difference between MODE = -1 and MODE = +1 is the DSRN used for the tape. For MODE = -1 DSRN is CPYOUT and for MODE = +1, DSRN is MAPTAP. (DSRN is MAPTAP for MODE = 0).

UNIT - I*4, DSRN of tape being mounted.

Output Arguments:

RQTAPE - I*4, When MODE = 0, set to -1 if requested tape file was full and user decided to use disk for results. Otherwise, remains unchanged.

RQFILE - I*4, When MODE = 1, set to -1 if requested tape file was full and user decided to use disk for results. Otherwise, sends back current file position of tape.

CHTAPE checks the validity of the tape by reading the record type 1 from the tape and verifying the tape and file number as well as checking for the correct type of file. Any attempt to overwrite an existing file causes CHTAPE to ask the user (via the typewriter) if he wishes to overwrite the file, respecify a new results card, or terminate the function. Note, however, that if a request has been made to initialize a tape, no checking is performed on previous contents.

2. Internal Description

See Output Description. Subroutines called by CHTAPE:

TAPOP	RINGIN	IVAL
MOUNT	CTLWRD	ERPRNT
CPFUNC	CTLPRM	RTMAIN

3. Input Description

The record type 1 of the results tape is read for each file up to and including the file needed. That is, if file 4 is requested the record type 1 is read from files 1-4.

4. Output Description

The following information messages are issued under the circumstances listed. The term filetype means the filetype code from record type 1 of a results file (the program uses variable CHECK for this number).

I0042 is typed when a tape has been mounted and before CHTAPE positions it. This message is not typed when the tape is being initialized or when the correct tape number was already mounted.

I0043 is typed when MODE = ± 1 and filetype of the requested file = 0.

I0044 is typed when MODE = ± 1 and filetype of the requested file = 1 and the restart flag from GLOCOM (RESTRT) is not = 1.

I0045 is typed when the tape is correctly positioned. This is not typed when initializing a tape.

After I0043 and I0044, the user is asked whether he wishes to overwrite the file, respecify a new results card with a new tape and/or file or disk option, or terminate the function.

I0100 is typed to allow entry of the new results card. This occurs when the user requests to respecify the results card.

I0101 is typed to confirm usage of disk for results and occurs whenever disk is specified on the results card.

The following error messages are typed under the conditions listed.

E361 is written when the tape is being filed forward and a file is encountered with filetype other than zero before the requested file is reached and MODE = 0.

E362 is written when the circumstance for E361 occurs and MODE = 1. It is also written when MODE = 1 and the filetype of the file requested is = -1.

E363 is written if the RESTRT flag is = 1 and the filetype of the requested file is not = 1.

E364 is written when MODE = 1 and the filetype of the fiel requested = 1.

E365 is written when an EOF is read on the results file. This should never occur with valid results files.

For message texts refer to the User's Manual.

5. Supplemental Information

This section deals with the handling of tapes by CHTAPE

Input:

If a tape is mounted on the device and it is the incorrect tape number (as noted from the appropriate status words in GLOCOM), TOPRU is called to unload the tape before the correct tape is mounted. If the correct tape is mounted, CHTAPE will check for the ring in if MODE = ±1. If the ring is not in, the tape is unloaded and MOUNT is called to mount the tape with the ring in. If the correct tape is mounted, CHTAPE assumes that the file number (as recorded in GLOCOM) is correct and moves the tape backwards or forwards to find the requested file.

Output:

The tape is mounted with ring in for MODE = ±1 and with ring out for MODE = 0.

The tape is left positioned at the beginning of the requested file. When the tape is initialized a TOPRW is used to do this.

6. Flowchart

Not Applicable

Exhibit 1

FRIS III Timeline Chart

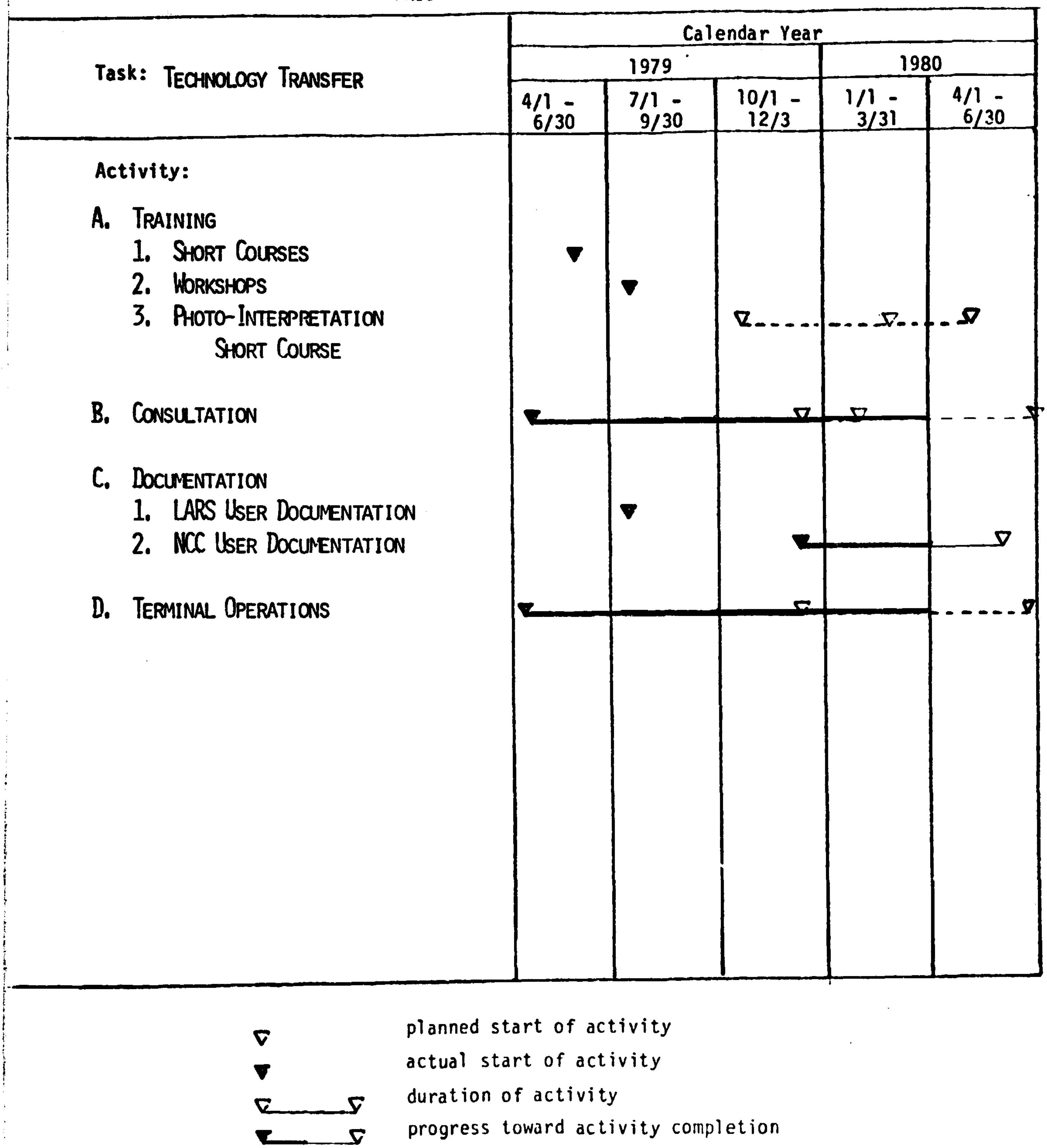
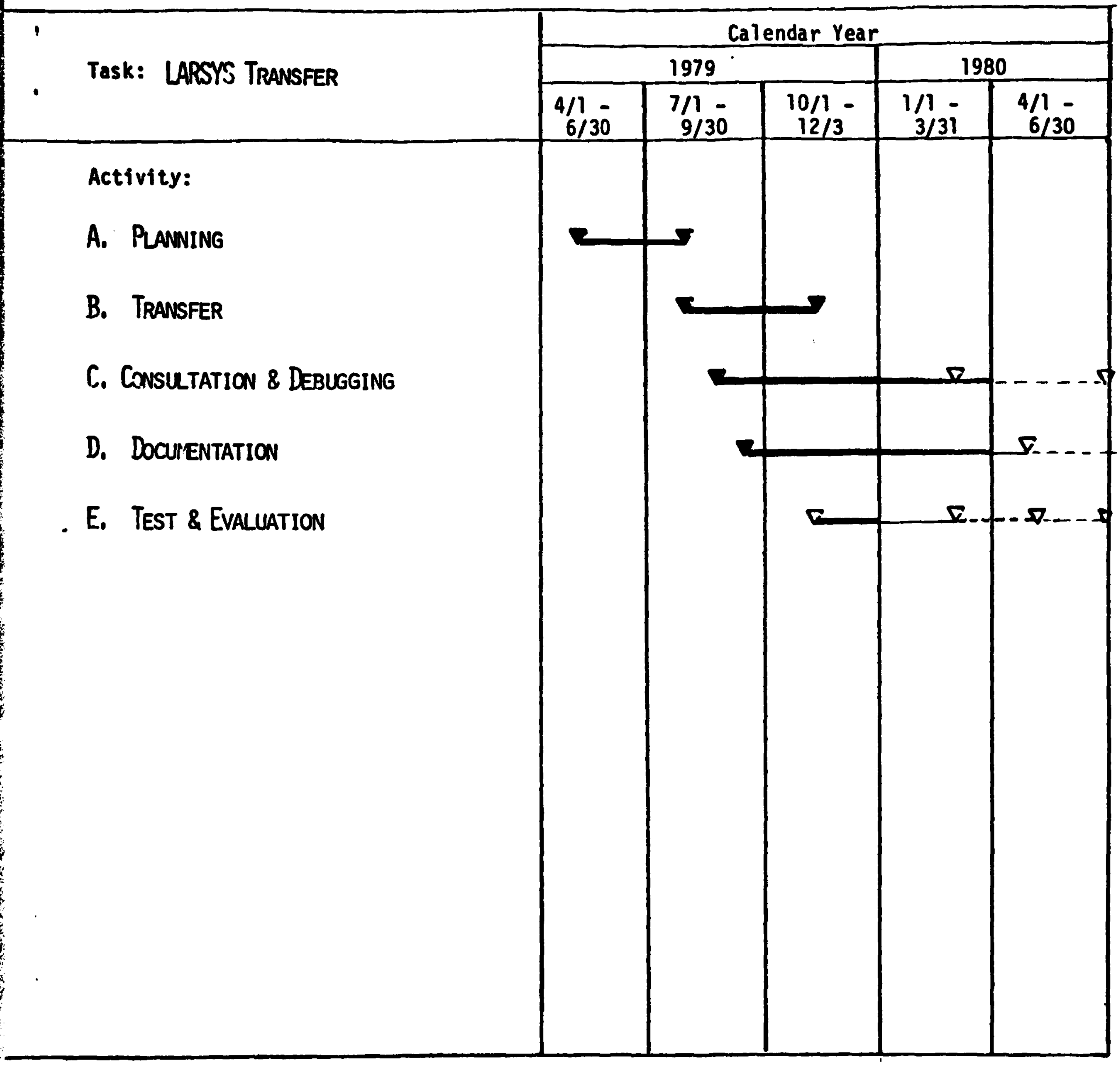


Exhibit 2

FRIS III Timeline Chart



planned start of activity

actual start of activity

duration of activity

progress toward activity completion

'T '|

FRIS LARSYS Software Documentation Task

	Apr	May	June	Jul	Aug	Sept

I. FRIS Software Development

A. CHANGEDETECTION
   1. Programming (completed 6/79)
   2. User Documentation
   3. Program Abstracts
   4. System Manual Flowchart
   5. Test Procedures

B. CLASSIFYPOINTS (MiniDistance)
   1. Programming (completed 3/79)
   2. User Documentation
   3. Program Abstracts
   4. System Manual Flowchart
   5. Test Procedures

C. SMOOTHRESULTS
   1. Programming
   2. User Documentation
   3. Program Abstracts
   4. System Manual Flowchart
   5. Test Procedures

D. PRINTRESULTS (Combine versions)
   1. Programming
   2. Program Abstracts
   3. User Documentation
   4. System Manual Flowchart
   5. Test Procedures

II. FRIS LARSYSDV Documentation

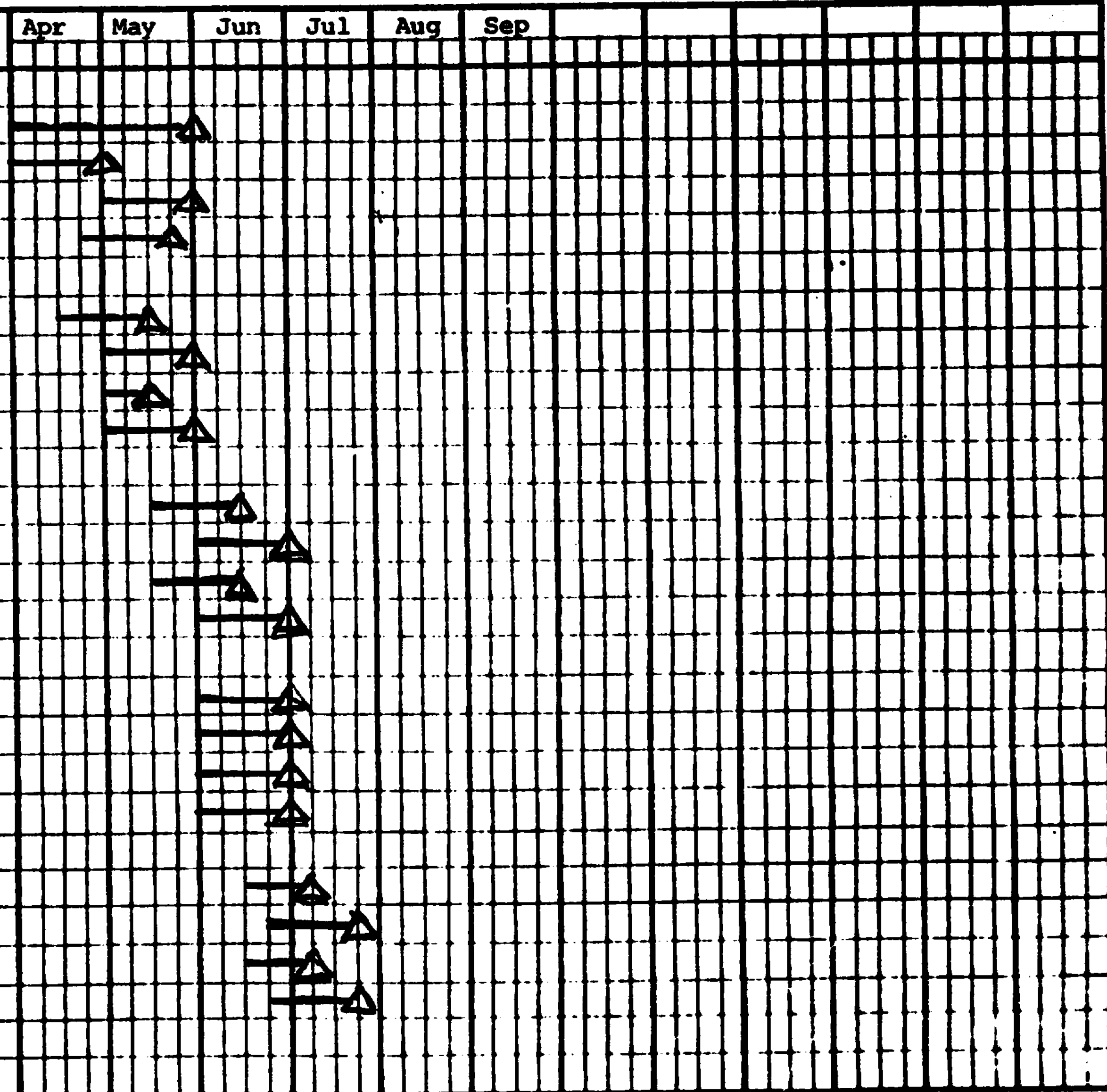
A. MERGESTATISTICS
1. Program Abstracts
2. User Documentation
3. System Flowchart
4. Test Procedures

B. BIPLOT
1. Program Abstracts
2. User Documentation
3. System Flowchart
4. Test Procedures

C. RATIO
1. Program Abstracts
2. User Documentation
3. System Flowchart
4. Test Procedures

D. BROWSE
1. Program Abstracts
2. User Documentation
3. System Flowchart
4. Test Procedures

E. ECHO
1. Program Abstracts
2. User Documentation
3. System Flowchart
4. Test Procedures



F. GDATA

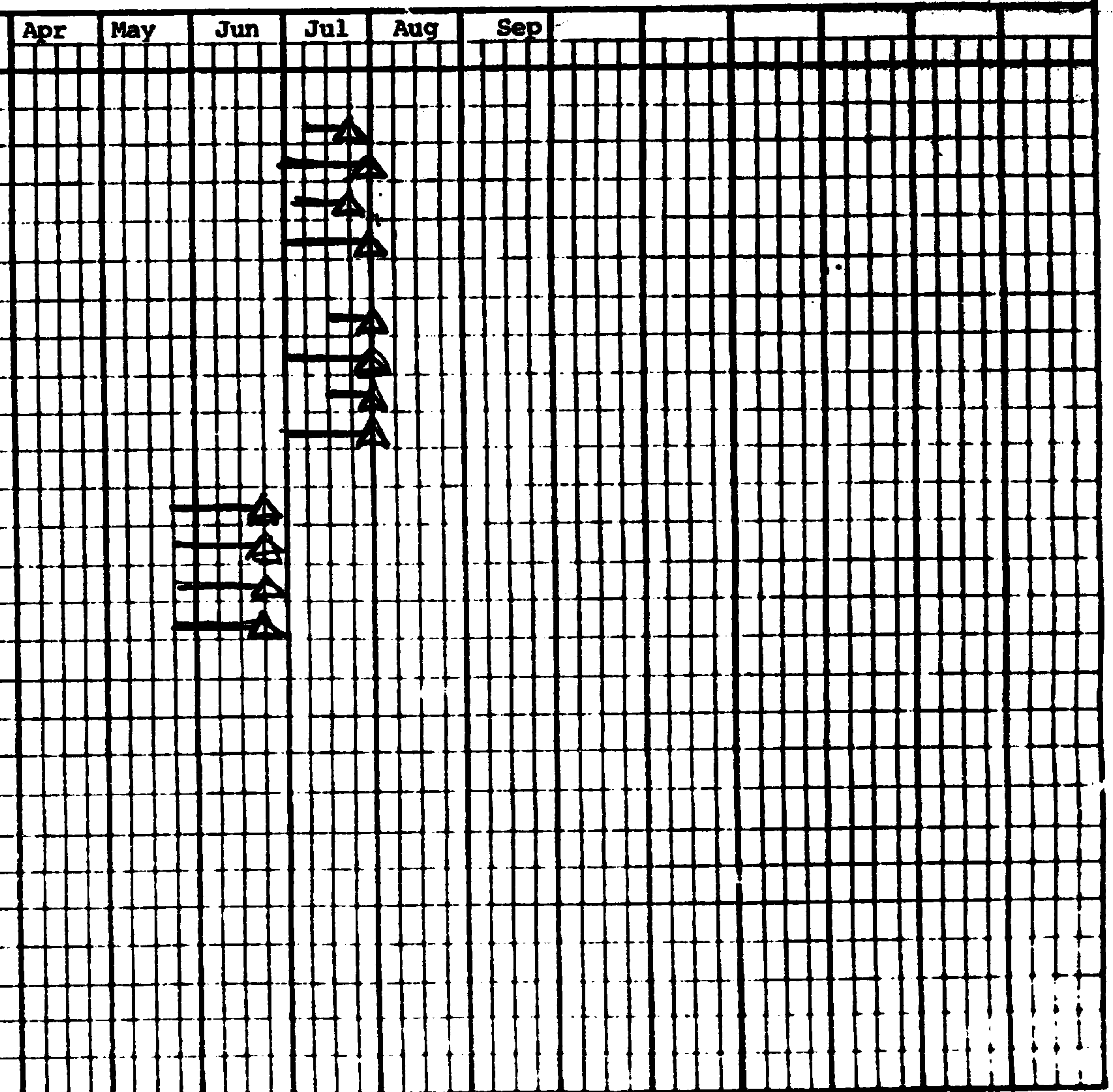
1. Program Abstracts
2. User Documentation
3. System Flowchart
4. Test Deck

G. GRESULTS

1. Program Abstracts
2. User Documentation
3. System Flowchart
4. Test Deck

H. Other DV Software (CLUSTER, SEP. DUPLICATERUN)

1. Program Abstracts
2. User Documentation'
3. System Flowchart
4. Test Deck



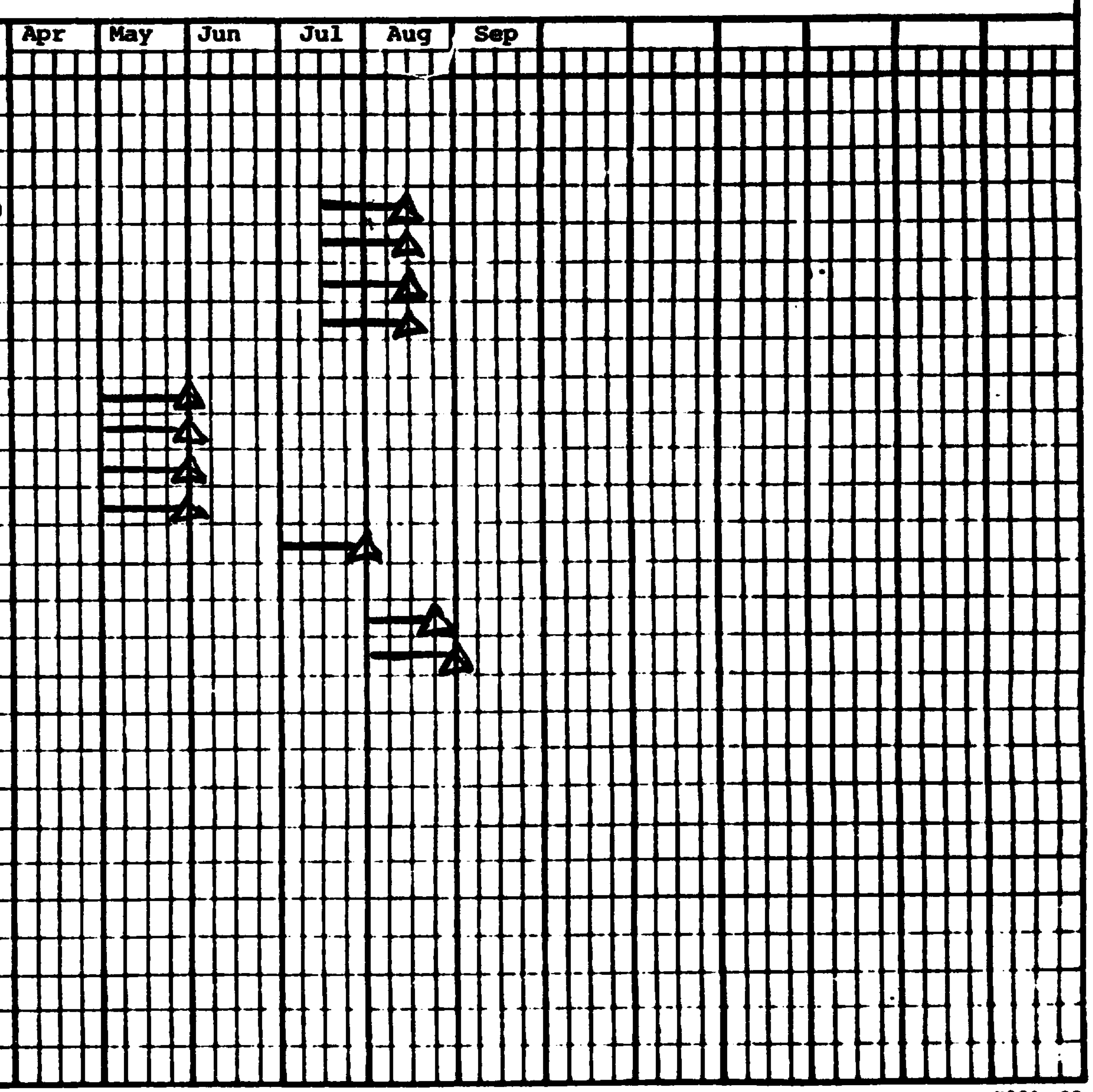
III. FRIS "LARSYS Documentation"

A. Remaining LARSYS Processors
*PIC, *STAT, *IDP, *LIST,*PUNCH,
*LINEGR, *COLUMNGR, *HIST, *GRAPH
1. Review User Manual (e.g. examples)
2. Review System Manual
3. Check Program Abstracts
4. Review Test Procedures

B. EXCOMD EXEC (+ related EXECs)
1. Abstracts
2. User Documentation
3. Flowchart
4. Test Procedures

C. System Manual (Section 2)

D. User Manual
1. Volume 1
2. Volume 3 - assign new error msg
#s and compile new list



IV. St. Regis COSMIC Package	Apr	May	Jun	Jul	Aug	Sep					
A. Compile module sizes, lines of code											
B. Create tape (document format)											
C. Duplicate "LARSYS" documentation											
D. Create New listing + Abstract books											
E. Send package to COSMIC											

JSC Form 1050C (Rev Jun 80)

NASA-JSC

Exhibit 3

FRIS III Timeline Chart

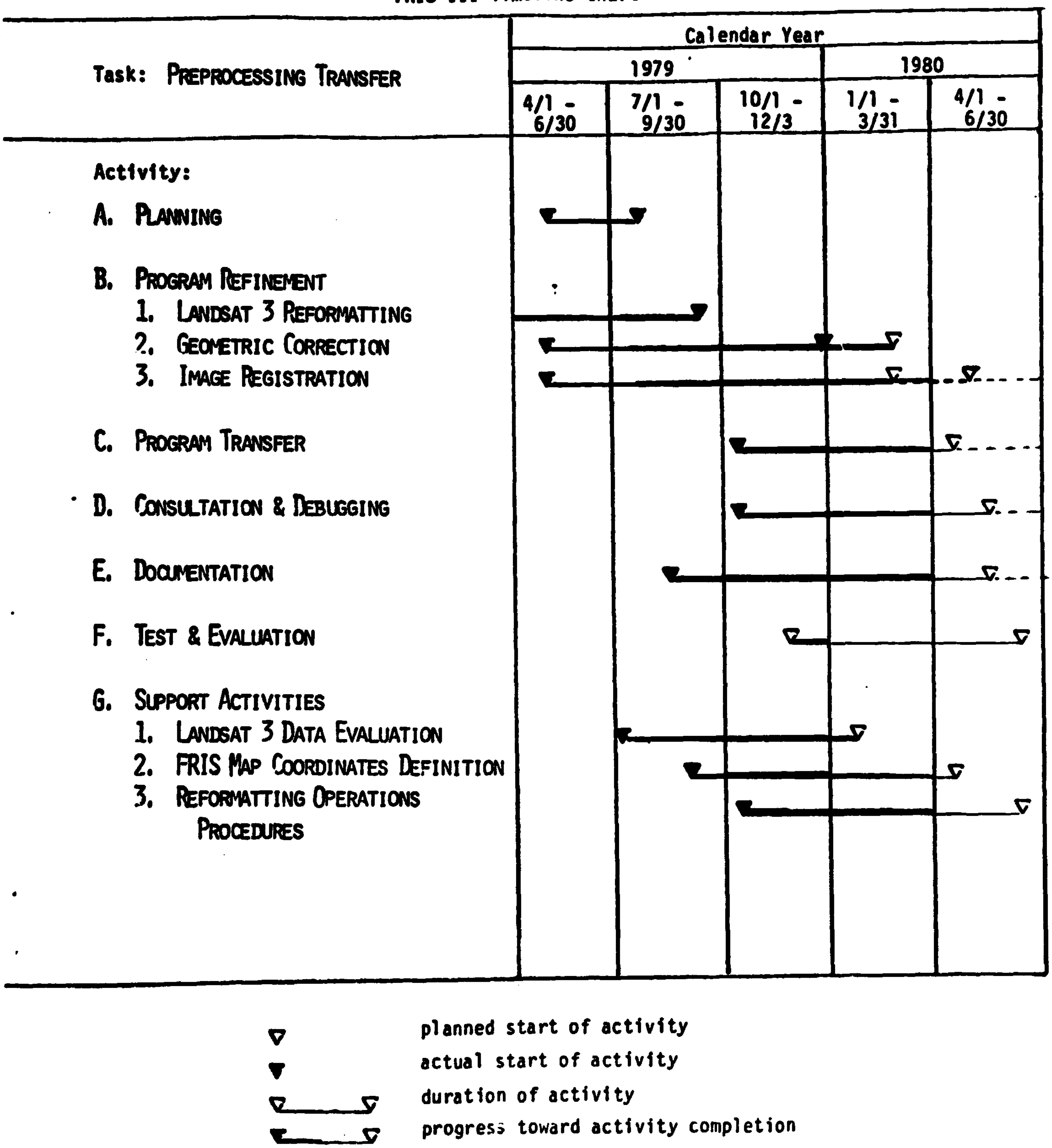


Exhibit 4

FRIS III Timeline Chart