

AD-A233 139

2

Multispectral Bathymetry Programs: A Users Guide

DTIC FILE COPY

DTIC
MAR 27 1981

H. V. Miller
T. Green-Douglas
C. L. Walker
R. K. Clark
T. H. Fay
Mapping, Charting, and Geodesy Division
Ocean Science Directorate

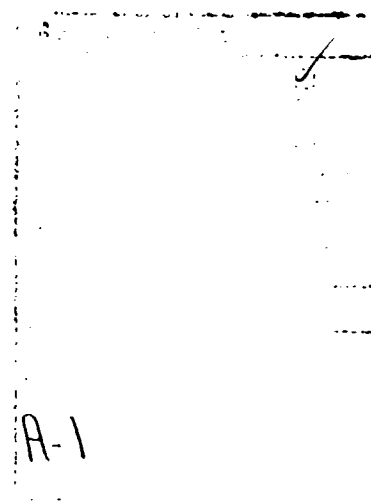


Approved for public release: distribution is unlimited. Naval Oceanographic and Atmospheric Research Laboratory, Stennis Space Center, Mississippi 39529-5004.

ABSTRACT

The Naval Ocean Research and Development Activity (NORDA)*, the Navy's lead laboratory in mapping, charting, and geodesy, is currently investigating the use of remotely sensed multispectral imagery as an accurate source for computing coastal-zone bathymetry. Because the Navy supports amphibious operations, special warfare, and coastal hydrographic surveying, knowledge of near-shore features is essential. The widespread availability, temporal sensitivity, and almost complete global coverage of most satellites' imagery make it an ideal way to collect water depth information from areas of limited or denied standard access. Bathymetry computations are done through software designed specifically for the ongoing research in this field. The software applications and abilities are discussed in this technical note.

*Recently designated the Naval Oceanographic and Atmospheric Research Laboratory (NOARL)



ACKNOWLEDGMENTS

We wish to express our appreciation to Ms. Maura Lohrenz, Mr. Gregory E. Terrie, and Mr. Michael Trenchard of NORDA's MC&G Division for their contributions to the hardware documentation.

This work was sponsored by the Office of Naval Technology (ONT) Program Element 62435N, under the program management of Dr. Melbourne Briscoe and CDR Lee Bounds (ONT Code 228), and Dr. Herbert C. Eppert, Jr., Director, Ocean Science Directorate, Naval Oceanographic and Atmospheric Research Laboratory, Stennis Space Center, Mississippi.

- I. INTRODUCTION
- II. PREPARING THE CALIBRATION DATA
 - A. Reading the points
 - B. Conversion programs
 - 1. DSTLL2LE (NOS tape data)
 - 2. UTM2ST (NOAA chart data)
 - C. Sorting the data
 - D. Sieve programs
 - E. DSTMAKER
- III. USING THE DST FILES
 - A. MINMAX4
 - B. MINMAX7
 - C. LINF4
 - D. LINF7
 - E. Example MINMAX4 run
- IV. SUMMARY FILES
- V. MAKING BATHYMETRIC IMAGES
- VI. FILTERING THE BATHYMETRIC IMAGES
- VII. ERROR IMAGE CREATION
- VIII. FINAL NOTES
- IX. SUMMARY
- X. REFERENCES

- APPENDIX A: Georeferencing an image using ELAS
- APPENDIX B: Description of QUAD.DAT
- APPENDIX C: Menu-driven command file
- APPENDIX D: Steps for making matrix camera prints
- APPENDIX E: Hints for adjusting matrix camera

APPENDIX F: NOS format and available tapes
APPENDIX G: Datum Transformations
APPENDIX H: Source Code (FORTRAN)

Multispectral Bathymetry Programs

A Users Guide

I. INTRODUCTION

The Coastal Image Understanding (CIU) project is currently investigating the combined use of various types of multispectral imagery, calibration data sets, and regression algorithms as a source for bathymetry measurements to chart coastal-zone waters. Not all coastal areas are readily accessible for exploration; thus, bathymetry capability relies on alternate techniques. These techniques must involve fast, efficient methodology to support naval interest in amphibious operations and special warfare. Although satellite remote sensing has the disadvantage of low spatial resolution, it has shown some promising results, particularly in areas where the ocean bottom is highly visible. To create bathymetric charts from the remotely sensed data, both calibration data and image processing software had to be developed. The following outlines the abilities of the software developed by the CIU project for the creation of bathymetric charts from Landsat Thematic Mapper (TM) and Le Systeme Probatoire d'Observation de la Terre (SPOT) remotely sensed imagery.

The bathymetry process presented in this document can be classified into two phases. First, calibration data must be prepared and analyzed. Sections II-IV describe the processing of the calibration data, and appendices F and G give location and availability status of raw calibration data. The remaining sections of this technical note present steps on how to complete the bathymetry process; that is, how to create a bathymetric image (section V), a filtered bathymetric image (section VI), and a coded image showing error at each of the calibration sites (section VII). In addition, appendices A, B, C, and H contain all algorithms and FORTRAN software necessary to georeference an image, process the calibration data, and utilize the results to create

bathymetric images. Appendices D and E describe how to transfer these images from digital image files to camera prints.

II. PREPARING THE CALIBRATION DATA

Calibration data used by the software discussed here can be obtained from two sources: National Oceanic and Atmospheric Administration (NOAA) bathymetric charts and National Ocean Survey (NOS) digital bathymetry tapes. Appendix A describes in detail how to georeference an image to a chart so that calibration points can be digitized using ELAS (Earth Resources Laboratories Application Software) and soundings from charts. If the data are to be taken from NOS tapes, then the following steps are suggested (see Appendix F for a complete description of NOS tape format and a list of those tapes available in the Pattern Analysis Laboratory tape library). If the data are to be taken from charts, only step (2) and program EOF from step (4) are necessary:

- (1) Read the points from the tape.
- (2) Determine the desired points and their lines/elements relative to a particular image (using conversion programs).
- (3) Sort primarily by line and then by element.
- (4) "Sieve" the file of points (e.g., cut on depth, take every 10th point, etc.) and put end-of-file record on file (program EOF).
- (5) Create a DST (Data Summary Tape) file.

Each of these steps is covered in the following sections. Program/file names and VAX/VMS commands are given where needed. Immediately after the discussion are comments on and schematic diagrams of the entire calibration data preparation process.

A. READING THE POINTS

To read NOS data points from tape, use the following commands:

```
$ MOUNT/FOR/DENS-6250/BLOCKSIZE-5120/RECORDSIZE-40 device:
    (mounts the tape on tape drive "device:" with NOS parameter values)
$ SET MAGTAPE device:/SKIP-FILES:n
    (skips the first n files, if necessary)
$ FTCOPY device:/FILES-m/BLOCKSIZE-5120/RECORD_SIZE-40 -
    outfile/RECORD_SIZE-40
    (creates the output file in the current account directory)
```

The FILES=m option in the FTCOPY command specifies that the next m files are to be read. The "outfile" created will contain all of the NOS survey data in original NOS format, consisting of a registry number, julian date, latitude, longitude, depth, and a cartographic code for each point (see Appendix F).

If desired, the user may select points in a particular latitude/longitude range from this output file by running the program FINDLLI. This program prompts the user for maximum and minimum latitude/longitude values and outputs only those points within that range. The registry number, latitude, longitude, depth, and cartographic code are printed in the format of the original NOS file.

B. CONVERSION PROGRAMS

1. DSTLL2LE

To determine desired NOS depth points, the program DSTLL2LE ("DST Latitude/Longitude to Line/Element") takes NOS calibration data and computes a line and element, relative to an entire TM quad and SPOT scene. It uses the ELAS subroutine LLUTM ("Latitude/Longitude to UTM") to convert latitude/longitude values to UTM coordinates. In turn, these coordinates are

used, along with the georeferencing coefficients generated by ELAS, to compute each point's line/element values that correspond to both the TM and SPOT image. As mentioned, the georeferencing of TM or SPOT images (i.e., finding the correspondence between UTM easting/northing values from the chart and the line/element values in the image) using ELAS is discussed in detail in Appendix A. The georeferencing coefficients are usually kept in ELAS-generated files named COEFUT.LEL and are created in the georeferencing process.

In the FTCOPY command listed previously, the file called "outfile" (or the output of FINDLLI, as the case may be) is the file to be used as input for DSTLL2LE. Each line of the output file will correspond to a calibration point and will contain latitude, longitude, easting, northing, depth, and a line and element position for both TM and SPOT data.

While running DSTLL2LE it may be necessary to use "junk" files. For instance, if only TM imagery is available, then a junk SPOT coefficient file and image file are needed. The junk coefficient file (usually named "COEFUT.JUNK") should contain six lines with 0.0 as the entry on each line (these six zeros represent the georeferencing coefficients). The junk image file (usually called "JUNK.SPOT" or "JUNK.TM") has only one element and one line and may be created using the ELAS FMGR AL option (i.e., the allocate option AL in the FMGR module of ELAS) to allocate the necessary memory. In reality, the junk image file may be any ELAS readable image file with nonzero line/element limits. However, using junk image files with only one pixel, certainly avoids any confusion as to its purpose and saves a considerable amount of disk space.

Because many of the NOS files are large, it is possible that points selected by DSTLL2LE may be outside a desired image. Because of this, a sifting program, LESIEVE ("Line/Element Sieve"), is used to process the output from DSTLL2LE and select only those points within a user-supplied range. For example, the user may want only those points which have positions occurring on the TM image having initial element 2000, last element 2511, initial line

2471, and last line 2982. LESIEVE allows the user to input these limits interactively.

As a final remark, the program DSTLL2LE assumes that the latitude/longitude coordinates taken from the NOS files are referenced to the same geodetic datum as the datum of the chart used in georeferencing. If this is not the case, program CORLL2LE ("Corrected Latitude/Longitude to Line/Element") should be used in place of DSTLL2LE. CORLL2LE is virtually the same as DSTLL2LE, only it allows the user to interactively supply corrections to bring the NOS latitude/longitude values into conformance with the chart latitude/longitude values. See Appendix G for the procedure to determine the corrections for conversion from one datum to another.

2. UTM2ST

The program UTM2ST ("UTM to SPOT/TM") takes NOAA calibration data and computes a line and element relative to an entire TM quad and SPOT scene. The input file for UTM2ST is generated by the digitizing routines within ELAS. The output file contains easting, northing, depth and line and element positions for TM and SPOT for each point. As in the NOS situation, "junk" files may also be needed.

The reader is encouraged to observe the sounding depth units of the NOAA chart being used. For DSTMAKER (discussed later in this document) to process a depth properly, these units must be feet. If the chart values have other units, then a conversion is necessary when entering values into the digitizer keypad. For example, if the depth is 2 fathoms, input 12 (1 fathom = 6 feet) into the keypad (see Appendix A).

C. SORTING THE DATA

The output file from LESIEVE consists of points in a particular line/element range. Sorting the points in this output file serves two basic purposes. First, the VAX/VMS SORT command allows for the deletion of

duplicate points. This option is particularly useful when working with NOS data, since several NOS calibration points can lie within the same pixel. Also, sorting points (first by line, then by element) allows for more efficient and speedy image processing. This topic will be discussed further in the section on error image creation.

`SORT.COM` is the command file which invokes the VAX/VMS SORT utility. The main command in this file is given below:

```
$ SORT/NODUPPLICATES/WORK_FILES-2 -
    /KEY=(POSITION:63,SIZE:8,NUMBER=1) -
    /KEY=(POSITION:55,SIZE:8,NUMBER=2) -
    filename1 filename2
```

As can be seen from the command, another file is created ("filename2") in the sorting process and is equal in size or smaller than the input file ("filename1"). Usually, after the sort has completed, the unsorted file is deleted. The positions 63 and 55 are predetermined values from the NOS format and represent the position (in "filename1") of a point's line and element, respectively.

In addition to prompting the user for "filename1" and "filename2", `SORT.COM` also requests a "scratch" disk to be used for the sorting workfiles. This disk is assigned to `SYS$SCRATCH` and deassigned when the command file has SUCCESSFULLY completed (in other words, if the command file is interrupted before completion, the user must manually "`$ DEASSIGN SYS$SCRATCH`"). Again, because the NOS files are large, this assignment is needed so that the memory of a user's disk (normally, `USER$DISK`) is not consumed by the large workfiles. However, if the user has enough free memory on `USER$DISK`, `USER$DISK` may be input as the scratch disk.

D. SIEVE PROGRAMS

Some additional routines for the user are the following:

DEPTHSIEVE - makes a depth cut based on maximum depth.

MODSIEVE - takes every n^{th} point of the input file.

These programs are self-explanatory and accept interactively all needed information.

Before the next phase of data calibration processing is performed, the program EOF must be run on the final sieved file. EOF places end-of-file records on the input file accepted by the program DSTMAKER. EOF must be run after all sieving has occurred.

E. DSTMAKER

Program DSTMAKER creates a data summary file of all gray levels for the calibration points that are to be used in the regressions for calculation of the bathymetric model(s).

The program will access both TM and SPOT imagery of the same area and compile the appropriate gray levels for each point given in the input file. The gray levels written to the DST file will be from bands 1-5 of TM imagery and 1-3 of SPOT imagery. DSTMAKER also writes to the output file three header lines of user-supplied information and comments. In general, the output file from DSTMAKER should be named with a ".DST" extension.

A sample DSTMAKER interactive session follows. Program prompts are indicated by a "." and user responses are indented for clarity:

```
$ RUN DSTMAKER
```

```
-Enter calibration file name:
```

```
    PUERTO.DEPTH
```

```
-Enter 'N' if calibration from NOS tape
```

```
-Enter 'C' if calibration from NOAA chart
```

N

-Enter IM image file name:

DJBO:[THFAY.TM.RICO]PUERTOQUAD3.DAT

-Enter SPOT image file name:

DJBO:[THFAY.SPOT]JUNK.SPOT

-Enter comments:

DST comments: It is suggested that the user include calibration file name and date, and NOT go beyond 132 characters.

-Enter desired name of DST file:

PUERTO.DST

-Enter desired name of histogram file:

PUERTO.HIST

The histogram file contains histograms relating to the calibration depths. See Brun et al. (1979) for descriptions of these histograms.

Figure 1 is a generalized overview of the calibration data preparation, as well as a suggested naming scheme for intermediate files. PUERTOQUAD3.DAT is the raw image file containing a TM quad of the Puerto Rico area. PUERTO.NOS is the initial "outfile" created by the FTCOPY command and is in original NOS format (Note: PUERTO.NOS may be preprocessed by FINDLLI as described in a previous section.) It should also be noted that DSTLL2LE and LESIEVE must be performed first; SORT.COM, MODSIEVE, and DEPTHSIEVE can be run thereafter in any order, if necessary. The input file to DSTMAKER should have special end-of-file records, created by EOF.

In Figure 1, the name "PUERTO" was used to signify data from the Puerto Rico area. The "100" in PUERTO100.MOD means every 100th point was taken from the input file. The file NOAA.DAT is the file generated by ELAS when digitizing points by hand.

At this stage, all imagery information needed for regression analysis is contained in the DST files. The satellite imagery files are not accessed during regression.

III. USING THE DST FILES

Four programs use the DST files to implement a linear regression against the calibration data to calculate the bathymetric model coefficients. For a more in-depth description of these algorithms, see Clark et al. (1987) and Bevington (1969).

A. MINMAX4

MINMAX4 performs a linear regression on the data points in the DST file. Desired data points are selected based on a user-given depth range, minimum depth and maximum depth. The program is designed to utilize a maximum of four bands and can accommodate either TM or SPOT data from the DST file but not both. The program also requests values for the L infinities (average band values over deep water) but does not select data points based upon the L infinity values. These values are used only in conjunction with regression (Clark et al., 1987).

B. MINMAX7

MINMAX7 performs in the same manner as MINMAX4 with two exceptions. This program uses both TM and SPOT data simultaneously from the DST file and can utilize a maximum of seven bands.

C. LINF4

LINF4 selects calibration points from the DST file based on band signal values. It can accommodate either TM or SPOT data but not both and can utilize a maximum of four bands. If the number of bands to be used is N, then

the following check is performed to determine if a point is to be used in the regression (gv(i) = gray value in band i, Linf(i) = L infinity for band i).

<u>N = 1</u>	<u>N = 2</u>	<u>N = 3</u>	<u>N = 4</u>
gv(1) > Linf(1)	gv(1) > Linf(1)	gv(1) > Linf(1)	gv(1) > Linf(1)
gv(2) <= Linf(2)	gv(2) > Linf(2)	gv(2) > Linf(2)	gv(2) > Linf(2)
gv(3) <= Linf(3)	gv(3) <= Linf(3)	gv(3) > Linf(3)	gv(3) > Linf(3)
gv(4) <= Linf(4)	gv(4) <= Linf(4)	gv(4) >= Linf(4)	gv(4) > Linf(4)

For example, if N=2, Linf(1)=68, Linf(2)=17, Linf(3)=14, Linf(4)=7, gv(1)=69, gv(2)=18, gv(3)=14, and gv(4)=6, then the point will be selected for use in the regression.

D. LINF7

LINF7 selects points in the same manner as LINF4 but is designed to use both TM and SPOT data together and a maximum of seven bands.

E. EXAMPLE MINMAX4 RUN

An example of the interactive portion of the regression program MINMAX4 follows. All of the programs are similar in design and will prompt in the same manner. Computer prompts are noted by a "-" while responses have been indented here for clarity. Technical note comments are in parentheses.

```
$ RUN MINMAX4
```

```
-Enter name of DST file to use.
```

```
[THFAY.TERRI.EXEC]KEY100.DST
```

-Enter name of output file.

NOS4016 OUT

-Enter 'T' for TM imagery

-Enter 'S' for SPOT imagery

T

-Enter min and max depths to get from calibration file.

0 16

-Enter number of bands to use in fit.

3

-Enter the LINF's (band 1 to 3)

68 17 14

(At this point the program will display the line and element limits of the image file as stored on the DST file header and allow the user to access a smaller portion of the image if desired. An example of both responses is provided.)

-Do you wish to make changes? (Y/N)

N

-No changes made.

-Do you wish to make changes? (Y/N)

Y

-Enter Initial Elem and Last Elem:

2400 2911

-Enter Initial Line and Last Line:

2472 1983

(The program will then echo to the screen some of both the user-given data and the program-generated information.)

The output file created will contain a listing of all the user given information along with all the coefficients, histograms, and other values generated by the regression program. A portion of this data is echoed to summary files (see section IV). In this example, the naming device consists

of the calibration type (NOS), the number of bands (4), and the values for DMIN (0) and DMAX (16). If this had been an LINF4 run, then the output file might have been named L4681714.NOS ("L4" - LINF4, "681714" - L infinities, ".NOS" - NOS data).

IV. SUMMARY FILES

Each of the four regression programs accesses two previously created files, SUMMARY.LIS and SUMMARY.DBAS, and appends information from the current run to these files. The summary file SUMMARY.LIS can be examined from any terminal. Although SUMMARY.DBAS contains the same summary information as SUMMARY.LIS, it is used to store the information in a format recognized by DBASIII, thus allowing for a data base of all runs of the regression programs. Corresponding to each logical record of a SUMMARY file is a regression run. A complete description of the SUMMARY logical record format is given in Table 1.

TABLE 1. Description of SUMMARY Logical Record. Formats are given in FORTRAN notation.

<u>Record Contents</u>	<u>Data Type</u>	<u>Description</u>
date	char*9	date of regression run
time	char*8	time of regression run
calibration type	char*4	type of calibration (NOS, etc.)
image file name 1	char*40	image name from first sensor
image type 1	char*4	image type
initial element	integer*4	initial element of file
last element	integer*4	last element of file
initial line	integer*4	initial line of file
last line	integer*4	last line of file
image file name 2	char*40	image name from second sensor
image type 2	char*4	image type

initial element	integer*4	initial element of file
last element	integer*4	last element of file
initial line	integer*4	initial line of file
last line	integer*4	last line of file
bands used	F7.2	bands used in regression
Linf (1-7)	7(F7.2)	L infinities
dmin	F7.2	minimum depth
dmax	F7.2	maximum depth
(A-A7)	8(F7.2)	multiple regression coefficients
(EA-EA7)	8(F7.2)	uncertainty in coefficients
r's(1-7)	F7.2	correlation coefficient for each band
rmul	F7.2	multiple correlation coefficient
calib mean	F7.2	mean error from calib. points
calib rms	F7.2	residual RMS from calib. points
cal fitted mean	F7.2	mean of gaussian fit to resids.
ecal fit mean	F7.2	uncertainty in gaussian mean
cal fitted sigma	F7.2	sigma of gaussian fit to resids.
ecal fit sigma	F7.2	uncertainty in gaussian sigma
test mean	F7.2	mean of resids from test points
test rms	F7.2	RMS of resids from test points
test fitted mean	F7.2	mean of gaussian fit to test resids
etest fit mean	F7.2	uncertainty in gaussian mean
test fitted sigma	F7.2	sigma of gaussian fit to test resids
etest fit sigma	F7.2	uncertainty in gaussian sigma
* calib. pts.	F7.2	number of calibration points
* test pts.	F7.2	number of test points
avg. percent error	F7.2	average percent error

V. MAKING BATHYMETRIC IMAGES

Three programs create single-channel bathymetric images by using the coefficients generated during regression:

- (1) TMBATHY - creates a TM bathymetric image
- (2) SPOTBATHY - creates a SPOT bathymetric image
- (3) OVLBATHY - creates a bathymetric image from TM and SPOT data;
The input image file must be TM overlayed onto SPOT and
can be created with the ELAS module OVLA.

An example of the interactive portion of the bathymetry program TMBATHY follows. All three programs are similar in design and will prompt in the same manner. Computer prompts are noted by a "-" while responses have been indented for clarity.

```
$ RUN TMBATHY
```

```
-This program is designed to handle data  
-in the following format:
```

```
-For TM data the input file must be  
-bands 1, 2, 3, 4, 5, in that order.
```

```
-Enter TM input file:
```

```
DJB0:[THFAY.TM]KEYQUAD2.DAT
```

```
-Enter output file:
```

```
DJB0:[THFAY.TM]KEYQUAD2.BAT
```

- What comments would you like written to the
- output file? Please limit them to 132 characters.

input file: KEYQUAD2.DAT, 3aug88, coeffs. from L4681714 run.

- Enter L infinities in the following order:
- TM bands 1-5

- Enter L infinity for band 1
- (If no L infinity, enter 0)

68

- Enter L infinity for band 2
- (If no L infinity, enter 0)

17

- Enter L infinity for band 3
- (If no L infinity, enter 0)

14

- Enter L infinity for band 4
- (If no L infinity, enter 0)

7

- Enter L infinity for band 5
- (If no L infinity, enter 0)

10

- Enter coefficients in the following order:

-A0

-A1-A4 for TM bands 1-4

- Enter coefficient A(0)
- (If no coefficient, enter 0)

11.307

- Enter coefficient A(1)

```

-(If no coefficient, enter 0)
    8.35
-Enter coefficient A(2)
-(If no coefficient, enter 0)
    -3.607
-Enter coefficient A(3)
-(If no coefficient, enter 0)
    0
-Enter coefficient A(4)
-(If no coefficient, enter 0)
    0
-Enter coefficient A(5)
-(If no coefficient, enter 0)
    0

```

Once completed, the output image file will have two header records, the first containing the first record of the input image file and the second containing the user-supplied information (coefficients, L infinities, etc.) This information can be seen at any terminal screen by using the VAX/VMS command "\$ DUMP/DEC".

VI. FILTERING THE BATHYMETRIC IMAGES

By passing a $m \times m$ filter window over the image (where "m" is an odd positive integer), program EDGE implements a selective filter for filtering single-channel images. This program is designed to be an edge-preserving noise-smoothing filter. The preservation of edges is accomplished by symmetric nearest neighbor (SNN) logic.

The SNN routine is based upon the following algorithm (Harwood et al., 1987). Let b, c, and d be the gray values of three window pixels where c is the value of the center pixel and b and d values for a symmetric pair. (Given

in Figure 2 is an example of what is meant by "symmetric pairs" about the center pixel of a 5 x 5 filter window.) First calculate $cc = c + c$. Then, from each symmetric pair, select one pixel as follows:

```

if (b+d>cc) then
    if (b>d) select d
    else select b
else if (b+d<cc) then
    if (b>d) select b
    else select d
else select c

```

The center pixel is then assigned a value equal to a user-supplied statistic based on the $(m^2+1)/2$ selected pixels in the window. In general, from each symmetric pair, the algorithm selects the pixel which is nearer in gray value to the center pixel. Reasoning behind SNN logic can be seen in Figure 2: If the filter window has a straight edge separating two distinct gray value distributions, most of the pixels selected by the SNN criterion will fall on the same side as the center pixel. In the case of a tie during comparison, the center pixel itself is selected for that particular symmetric pair comparison.

The SNN logic is invoked only when a filter window contains a land pixel. On all other filter windows the entire contents of the window is used in the filtering process. By doing this, land and near-shore edge features are preserved.

The user may choose three filtering options:

- (1) Mean - replaces the value of the window center pixel with the mean value of selected pixels.

- (2) Median - replaces the value of the window center pixel with the median value of selected pixels.
- (3) Minimum - replaces the value of the window center pixel with the minimum value of selected pixels.

According to Harwood et al. (1987), the SNN-median filter produces slightly better results than the SNN-mean in terms of edge preservation; however, the SNN-mean is computationally more efficient.

Diagrammed in Figure 3 is a summary of bathymetric image creation and filtering (TM data), along with suggested filenames. VIEQUES.RAW is a five-channel TM image file and is named "VIEQUES" to signify data from the area near the island of Vieques.

A more generalized program, EDGEMULTI, uses the SNN selection criteria on each window, regardless of the pixels present; that is, no deference is made if the window doesn't contain a land pixel. In addition, this program accepts multichannel images and prompts the user for the number of channels and the channels to be filtered. The unfiltered channels are simply copied to the output file. As with EDGE, this program allows the user to enter one of the previously mentioned three filtering options. The main purpose of EDGEMULTI is to filter images which are largely comprised of land or shoreline pixels and very few water pixels.

VII. ERROR IMAGE CREATION

Once coefficients have been generated from a regression run (MINMAX4, MINMAX7, LINF4, or LINF7), the program ERROR can be used to generate an image displaying each calibration point color-coded by the error associated with that point's depth. The input image is a single-channel image containing exactly two classes: one for land pixels and one for water pixels. The

output image is the same single-channel image, with the calibration points given a value determined by their computed error. The error values are divided into bins, normally of 1-m width, as illustrated in Table 2. Each calibration point is assigned a class associated with the error bin containing the point's error value. A user-defined color is then assigned to each bin value.

TABLE 2. Example Error Bins and Associated Classes.

<u>Range</u>	<u>Class</u>
≤ -8	99
$(-8, -7]$	100
$(-7, -6]$	101
$(-6, -5]$	102
$(-5, -4]$	103
$(-4, -3]$	104
$(-3, -2]$	105
$(-2, -1]$	106
$(-1, 0]$	107
$(0, 1]$	108
$(1, 2]$	109
$(2, 3]$	110
$(3, 4]$	111
$(4, 5]$	112
$(5, 6]$	113
$(6, 7]$	114
$(7, 8]$	115
> 8	116

Before ERROR can be run, the user must first create the "land water" image by running LANDWATER. The input image must have five channels, and channel 5 values (usually TM band 5, but other infrared channels may be used)

are used to distinguish between land and water by simple thresholding (for example, if band 5 value > 10, pixel is land; else pixel is water). The output file created will be used as input for ERROR.

When ERROR is executed, the user will be prompted for a DST filename, land/water filename, error image filename, and output filename. The output file will contain the following summary information: total number of points with error <= -8 or error > 8, total number of points plotted, and total number of points out of range. Comments, up to 132 characters, may also be supplied by the user at this point.

For the next phase of interaction, a set of coefficients and L infinities are required. Similar to the programs which create bathymetric images, the prompts guide the user in entering these values. Once the values have been obtained, a depth is calculated for each point (if it is in line/element range) in the DST file by the formula:

$$\begin{aligned} \text{CDEPTH} = & \text{NINT}(\text{A}(0) + \text{A}(1)*\text{ALOG}(\text{MAX}(\text{FLOAT}(\text{GV}(1))-\text{L}(1)),1.0)) \\ & + \text{A}(2)*\text{ALOG}(\text{MAX}(\text{FLOAT}(\text{GV}(2))-\text{L}(2)),1.0)) \\ & + \text{A}(3)*\text{ALOG}(\text{MAX}(\text{FLOAT}(\text{GV}(3))-\text{L}(3)),1.0)) \\ & + \text{A}(4)*\text{ALOG}(\text{MAX}(\text{FLOAT}(\text{GV}(4))-\text{L}(4)),1.0)) \end{aligned}$$

where A(i) denotes a coefficient, L(i) denotes an L infinity value, GV(i) denotes the band i value of the calibration point, and CDEPTH denotes calculated depth. ALOG, NINT, MAX, FLOAT are the standard FORTRAN specific functions. The error for the calibration point is then found by subtracting this calculated depth from the actual depth obtained from the DST file. An appropriate value (determined by the above ranges) is then assigned to the pixel. When execution has completed, the error image can be displayed through the ELAS module COMD. With an appropriate color table, various error ranges can be highlighted and color-coded.

It should be noted that ERROR accepts DST files which are sorted, where the primary key is line and the secondary key is element. By having the

sorted information, an image line is read in only once, all appropriate pixels on that line are then "fixed," and finally the line is written out. Without a sorted DST file, ERROR is likely to read and write an entire line more than once, causing the execution time for ERROR to increase.

ERROR also catalogs most of the user-supplied information internal to each image it creates. The file header is record 1. The information is kept in record 2 of the file and can be seen by executing the following VAX/VMS command: \$ DUMP/DEC filename. All coefficients, L infinities, filenames, user-supplied comments, and the total point count are written to this record in a format that is easily readable from the terminal screen.

To summarize, Figure 4 gives the order of execution and suggested filenames to use when creating an error image (VIEQUES.RAW is a five-channel image file of TM data).

VIII. FINAL NOTES

At present, the source code for the programs discussed in this document is located in USER\$DISK:[BATHY.SOURCE] and the executable code is located in USER\$DISK:[BATHY.EXEC]. USER\$DISK is the user's disk on the VAX 11/780, node A35.

Appendix C contains an example of a menu-driven command file, BATHYMETRY.COM, which offers options to run most of the aforementioned programs. In addition, appendices D and E contain detailed instructions on how to obtain a hardcopy output of an image, using a Matrix Instruments camera.

IX. SUMMARY

The software discussed in this document is the result of an effort to calculate near-shore water depths using remotely sensed multispectral imagery. The technique can be summarized into two broad phases, that phase which deals with calibration data preparation and the phase which is associated with the processing of this data. In calibration data preparation, all software is geared toward the completion of a DST file that contains all information relative to the calibration depth points. This DST file is then used by the appropriate regression program to calculate the relationship between imagery digital values and depth, and also to compute the error associated with this relationship. Once this correspondence is determined, a bathymetric image can be generated and filtered to show coastal-zone bathymetry.

X. REFERENCES

Bevington, P. R. (1969) *Data Reduction and Error Analysis for the Physical Sciences*. New York, McGraw-Hill.

Brun, R., I. Ivanchenko, P. Palazzi (1979). *HBOOK Histogramming, Fitting and Data Presentation Package Users Guide*, version 3.

Clark, R. Kent, Temple H. Fay, and Charles L. Walker (1987). *Bathymetry Measurements from Landsat Imagery in the Vicinity of Isla de Vieques*. Naval Ocean Research and Development Activity, Stennis Space Center, Mississippi, NORDA Report 202.

Harwood, D., M. Subbarao, H. Hakalahti, and L. S. Davis (1987). "A New Class of Edge-Preserving Smoothing Filters." *Pattern Recognition Letters* 6(3):155-162.

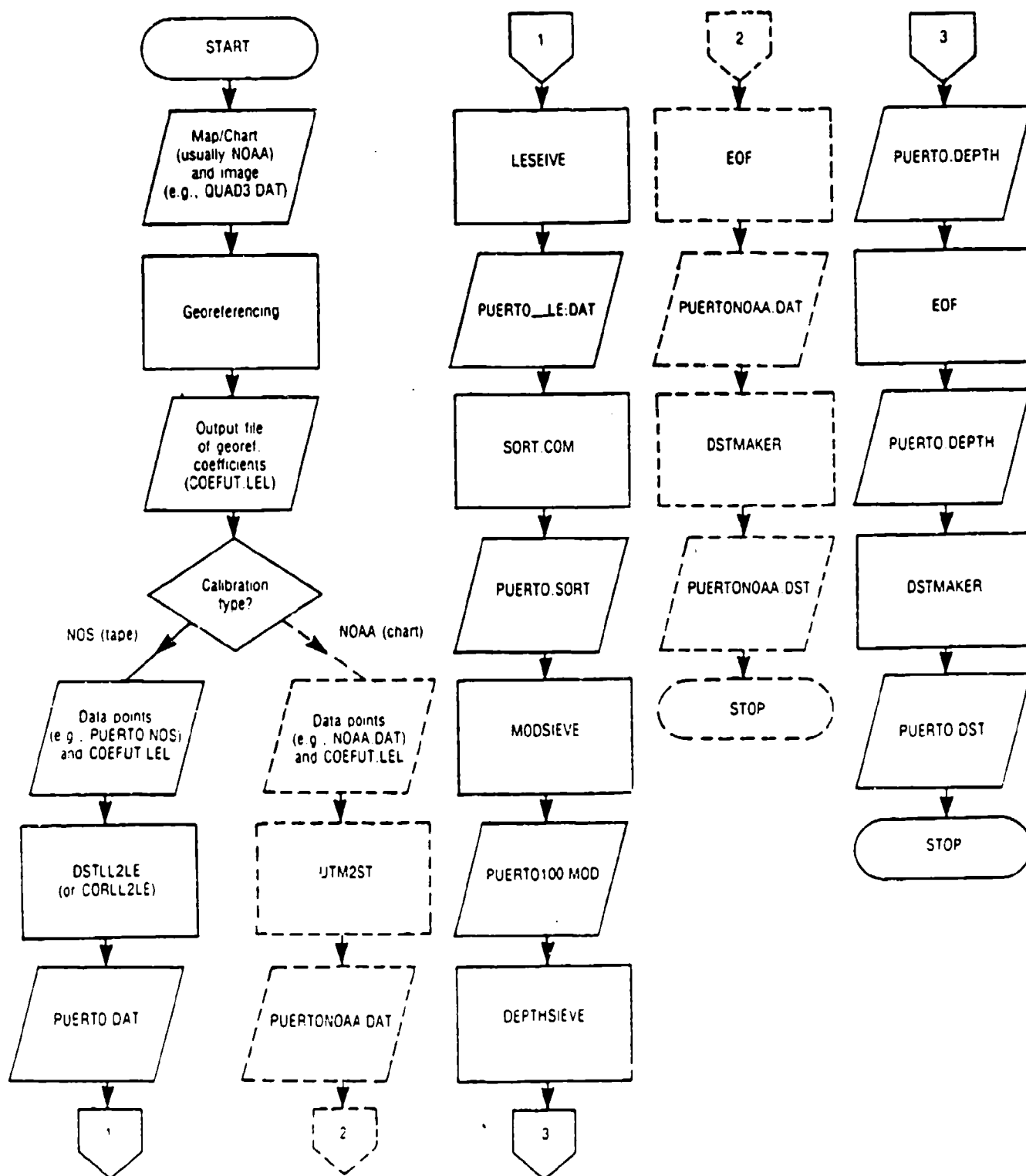


FIGURE 1. Summary of Calibration Preparation

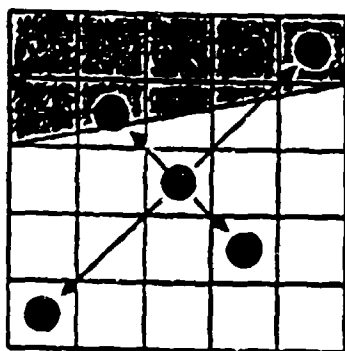


FIGURE 2. Sample 5 x 5 Window and Symmetric Pairs of Pixels

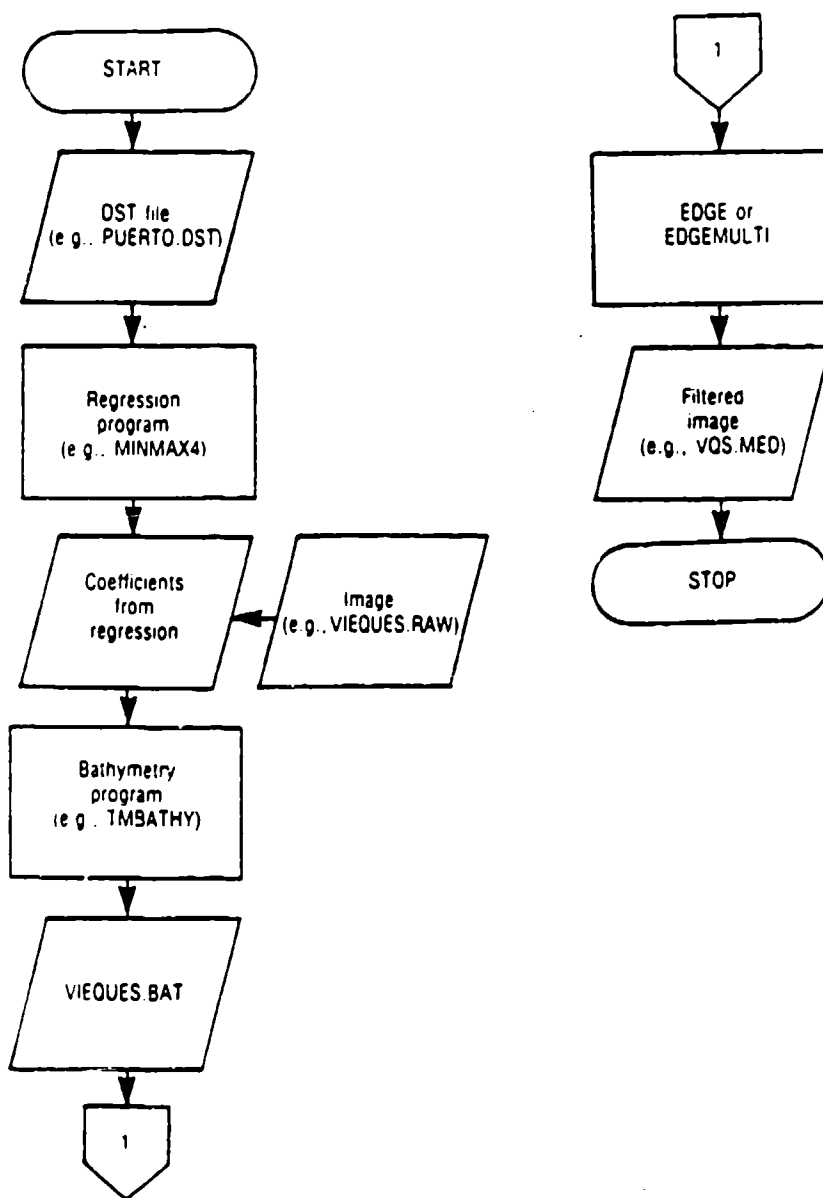


FIGURE 3. Summary of Bathymetry Calculation and Filtering

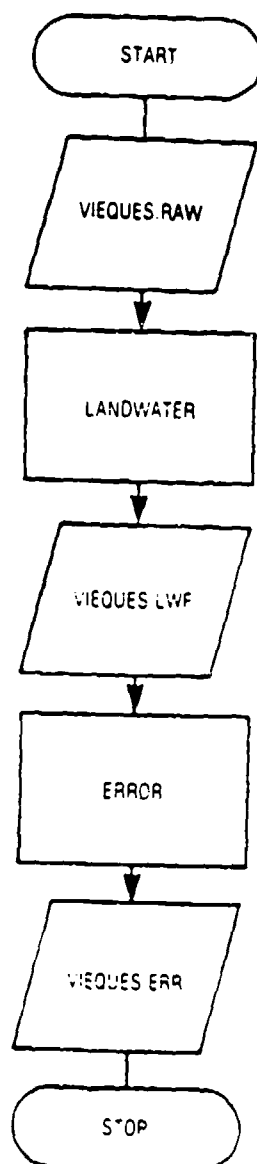


FIGURE 4. Summary of Error Image Creation

APPENDIX A: Georeferencing an Image Using ELAS

(By Gregory Terrie and Michael Trenchard,
updated by T. Green-Douglas and V. Miller)

INTRODUCTION - This appendix guides the reader through a step-by-step process to compute a set of georeferencing coefficients to convert Universal Transverse Mercator (UTM) easting/northing coordinates to line/element values and, if desired, to create a georeferenced image using the ELAS image processing package. It is assumed that the reader has already selected an image to process and that the reader has some knowledge of the use of ELAS.

PREPARATION - The following steps should be followed before entering the ELAS georeferencing modules:

(1) Obtain all the maps for the area of interest and number them using the method outlined in this documentation. If NOS calibration data are to be used in connection with the coefficients produced in the georeferencing process, then particular attention must be given to the datum associated with the maps. Most NOS data sets have latitude/longitude coordinates relative to North American datum, 1927; see NOS documentation for a point-of-contact to which questions regarding the datum for a specific area should be addressed.

(2) Determine the UTM coordinates of the four corners of each map. The corners should be well defined, such as intersections of latitude/longitude lines, and are not necessarily the corners of the entire map. If the coordinates are in latitude/longitude, then use ELAS module CVRT to convert these values to UTM coordinates.

(3) Store the map numbers and the four corner coordinates for each map in an external file using an edit command (i.e., EDT filename in VMS). Name the file QUAD.DAT and then run TSK\$ELAS:QUADBLD.EXE. It will ask for an output file; respond with "QUAD.". See Appendix B for details on building the QUAD.DAT file.

(4) Turn on the digitizer (toggle switch is located on small black console under the digitizer). Press the red reset button (next to the on/off switch) and the RS232 button on the digitizer cursor keypad.

(5) Enter ELAS and set up the control file so that usage ID1 is assigned to the data file from which control points are to be picked (ex: VIEQUES.DAT). Assign usage DG to the digitizer (ex: AC TXB3: DG, where TXB3: is the device number for the digitizer) and assign DISP to the display device (ex: EPA1).

NUMBERING THE MAPS - The map numbering system was designed so that the user could easily catalog a large number of orthophotoquads. The maps, however, can be of various types, and all maps are not required to be orthophotoquads. These maps should be numbered by groups of four (quads) and the subsets numbered in a counterclockwise direction beginning with the upper-left quadrangle.

READING AND RECORDING THE CORNER COORDINATES - Enter the ELAS module CPPP by typing "CPPP" at the "FMGR?" prompt. Be sure that the map being used is stretched tightly and mounted securely to the digitizer. Type "IM" (initialize map) at the computer terminal; ELAS will prompt the user for a quad sheet number and quadrant number. If only one map is being used in the georeferencing process, as is usually the case, respond with "1" to each of these prompts. ELAS will then request input of the first corner point. Place the digitizer cursor keypad on the point of the desired upper-left corner of

the map, centering this point inside the cross hairs in the small clear lens of the keypad, and press "3" on the digitizer keypad. This command sends the point's coordinates to ELAS. Rotate counterclockwise and enter the other three corner points in a similar manner, being as accurate as possible. When finished, RMS values (UTM and LS) will be returned. It is a good idea to attempt several of these "map tie-downs" (i.e., retype "IM" and repeat) and stop when the RMS values are smaller than the imagery pixel resolution.

SELECTING CONTROL POINTS - A variety of commands may be used to display the image from which the control points are to be taken. First, type "RST" (reset to display overall scene) to view the entire scene. Move the screen cursor to an area in which control points can be chosen and use the "CTR" command to select a new image center. A 512 x 512 portion will appear on the screen. To display a particular channel (for instance, channel 5 of Landsat TM imagery), use the command "RI 5". Typing "XF 2" (expansion factor set to 2) before displaying an image will enlarge the image.

Enter the "pick point mode" by typing "PP" at the "CPPP?" prompt. (Pick points that have a relatively clear position on both the map and the image, such as distinctive geographical features. In general, try to emphasize natural features such as coastlines rather than roads, airports, etc.) ELAS will then request a point from the digitablet. Refer to the map on the digitablet and digitize the desired point (press "3" on the keypad), move display screen cursor to the corresponding point and enter a carriage return. ELAS will respond with the element and line for that point and then request another point. Continue to pick points in this manner until a desired number of well-distributed points have been chosen (choose at least 15). While picking points, be sure to record (on paper) a general location for each point: for example,

point 1 - elem:2300 line:2378 - Pier, west of Fleming Key

point 2 - elem:2250 line:2536 - southern tip of Archer Key

Press "5" on the digitizer keypad to get out of the "pick point mode". Type "RMS" (calculate root mean square). A listing of the points that have been selected will be given along with the RMS error. If the RMS error for point n is not acceptable (e.g., 30 meters or more), use "DEP n" to delete point n. Note that, after a point is deleted, a resequencing of point numbers occurs. Thus, if deleting more than one point, be sure to delete the highest numbered point first (e.g., DEP 15, DEP 7, DEP 3 <carriage return>). RMS can be rerun to recalculate the RMS error. If desired, "PP" can be used again to add more points.

It is wise to choose control points that are well distributed throughout the image. Selecting points in a small area may yield "acceptable" RMS values, but the georeferencing coefficients, which are used to convert UTM easting/northing values to line/element values, may not yield satisfactory results away from this area. When finished selecting points, a good way to test the coefficients is to use the CPPP directive DCTR. A point can be selected from the map on the digitizer (press "3" on the keypad) and the corresponding scene, with that point as its center, will scroll onto the screen.

CREATING COEFFICIENTS AND GEOREFERENCED IMAGE - Enter the ELAS module PMGC by typing "PMGC" at the "CPPP?" prompt. Type "C1" (compute mapping coefficients interactively). If you wish the georeferencing constants to be output to an external file, respond with "Y" to the ELAS prompt "OUTPUT CONSTANTS TO COEFUT.LEL?". Enter ELAS module PMGE and type "SP LP" to set parameters, list parameters. Input the appropriate values for each of the parameters MAXE (maximum easting), MINE (minimum easting), MAXN (maximum northing), MINN (minimum northing), referring to the four corner coordinates of the map. Parameters HT and WID refer to the pixel size of the image (e.g., 30 x 30).

Parameter ODF is the output filename. After the parameters have been set, type "EX" and "RUN".

This completes the creation of the warped, georeferenced image (the ODF file mentioned above).

SELECTING DEPTH POINTS - Once the georeferencing process is completed, depth points can be chosen for calibration purposes. To select depth points, enter the DGTZ module and type "IN" to initialize map. ELAS will respond with the prompt "DIGITIZE POINT 1 THEN INPUT X,Y VALUES". The user then places the digitizer keypad on the upper left corner of the chart and presses "3", centering the cross hairs in a similar manner as with the CPPP map tie-down. The easting (X) and northing (Y) values for that corner are then typed at the terminal. Continue similarly in a counterclockwise direction for the next two corner points (the upper right corner point is not needed).

Next, type "PPD" to enter the "pick depth point" mode and respond with a name to the prompt "WHAT POINT NAME ? (POINT NUMBER WILL BE APPENDED)". A single letter, such as "A", will suffice for this purpose. Center the digitizer keypad's cross hairs on a chart sounding and, with a steady hand, enter the following key sequence: depth --> DP --> STR --> EN --> 3. The key "DP" represents the decimal point of the depth, so a number key may be punched after this key and before the "STR" key. If "5" rather than "3" is pressed after "EN", the user exits the pick point mode. Below are some examples.

<u>chart sounding</u>	<u>key sequence</u>
10.0	1, 0, DP, STR, EN, 3
5.2	5, DP, 2, STR, EN, 3
13.7	1, 3, DP, 7, STR, EN, 5 (last point taken)

A file, FOR025.DAT, will be created and will contain the following information. in this order, for each point: easting, northing, depth, element, line.

NOTE: For more information about the Numonics digitablet and keypad, see the 2300 USERS MANUAL, NUMONICS DIGITABLET in the Pattern Analysis Lab.

APPENDIX B: Description of QUAD.DAT

The quad file QUAD.DAT contains the map group numbers, the quad sheet numbers, and the four corner points for each map recorded in UTM coordinates. These data are contained in an external file called QUAD.DAT which is read by the georeferencing routines of ELAS for tying down images to corresponding maps.

QUAD.DAT should be built with the map group numbers located in columns 1-3, the individual quad sheet numbers in column 4, and the UTM coordinates in columns 24 through 78. The coordinates should be in sets of two with easting first, followed by the northing. The coordinates of the upper left hand corner of the quad sheet will be the input for the first set. The remaining corner points must then be entered in a counterclockwise direction.

An example of a typical QUAD.DAT file is shown below. Column numbers have been added for clarity on data format but should not be written in the file.

TYPICAL QUAD.DAT FILE

```
c
c 00          2          3          5          6          7
1 34          4          8          2          6          8

11          2374694237604 2370204223730 2479804223383 2484104237257
```

After the file QUAD.DAT has been created, the program QUADBLD.EXE should be run. QUADBLD.EXE reads in the QUAD.DAT file created above and outputs another quad file, which is readable by ELAS. This output file must be named "QUAD." because that is the filename searched for by the ELAS georeferencing modules. QUADBLD.EXE will prompt the user for the output filename.

APPENDIX C: Menu-Driven Command File

The following is an example of a command file, BATHYMETRY.COM, which allows the user to select options from the areas of bathymetry calculation (options 1- 4) or image processing (options 5-7).

```
$ SET NOVERIFY
$ ASSIGN SYS$COMMAND: SYS$INPUT
$ INQUIRE :-- INQUIRE/NOPUNCT
$ WS :-- WRITE SYS$OUTPUT
$MENU
$ WS " "
$ WS " "
$ WS " OPTIONS "
$ WS "-----"
$ WS "1 - Paredes & Spero model fit, TM or SPOT, cut on depths"
$ WS "2 - Paredes & Spero model fit, TM and SPOT, cut on depths"
$ WS "3 - Paredes & Spero model fit, TM or SPOT, cut on L infinities"
$ WS "4 - Paredes & Spero model fit, TM and SPOT, cut on L infinities"
$ WS "5 - Create Bathymetric image, TM"
$ WS "6 - Create Bathymetric image, SPOT"
$ WS "7 - Create Bathymetric image, TM overlaid onto SPOT"
$ WS "8 - END"
$ WS " "
$ WS " "
$!
$ INQUIRE OP " Enter option: "
$ IF OP .EQS. "1" THEN GOTO OP1
$ IF OP .EQS. "2" THEN GOTO OP2
$ IF OP .EQS. "3" THEN GOTO OP3
```

```
$ IF OP .EQS. "4" THEN GOTO OP4
$ IF OP .EQS. "5" THEN GOTO OP5
$ IF OP .EQS. "6" THEN GOTO OP6
$ IF OP .EQS. "7" THEN GOTO OP7
$ IF OP .EQS. "8" THEN GOTO ENDIT
$!
$OP1:
$ RUN USER$DISK:[BATHY.EXEC]MINMAX4
$ GOTO MENU
$!
$OP2:
$ RUN USER$DISK:[BATHY.EXEC]MINMAX7
$ GOTO MENU
$!
$OP3:
$ RUN USER$DISK:[BATHY.EXEC]LINF4
$ GOTO MENU
$!
$OP4:
$ RUN USER$DISK:[BATHY.EXEC]LINF7
$ GOTO MENU
$!
$OP5:
$ RUN USER$DISK:[BATHY.EXEC]TMBATHY
$ GOTO MENU
$!
$OP6:
$ RUN USER$DISK:[BATHY.EXEC]SPOTBATHY
$ GOTO MENU
$!
$OP7:
```

\$ RUN USER\$DISK:[BATHY.EXEC]OVLBATHY

\$ GOTO MENU

\$!

\$ENDIT:

\$ WS " THE END "

\$ DEASSIGN SYS\$INPUT:

\$ EXIT

APPENDIX D: Steps for Making Matrix Camera Prints

CAMERA AND DEVELOPER OPERATION:

- (1) Fix camera control panel settings: color mode — CLR COMP
operation — POS AUTO
- (2) Loading negative into film cassette and film cassette into camera:
Hold cassette, blue buttons up;
Press buttons to open cassette;
Place negative, with white letters facing up, on right side of
cassette;
Center negative between blue lines;
Catch cassette's orange tab under the negative paper;
Close cassette so that it snaps shut;
Pull white arrows to remove the negative paper covering;
Turn cassette over so that yellow arrow is facing up;
Open camera door;
Insert cassette in camera drawer, yellow arrow pointing toward the
user;
Remove negative protector (a blue slide under yellow arrow tab);
Close camera door.

- (3) Fix camera settings: image position — FWD (Note: "1" should be displayed; if not, hit RST FWD)

exposure thumb wheel — 1200 for photo prints 809
— 2800 for transparencies
891

exposure — EXP (Camera will then cycle RGB)

- (4) Removing cassette from camera:

Open camera door;

RE-INSERT FILM PROTECTOR (BLUE SLIDE) INTO CASSETTE UNDER YELLOW ARROW TAB (If this is not done before cassette is removed, the negative is destroyed);

Lift cassette slightly and remove from camera.

- (5) Developing the film:

Insert positive in developer under silver tab (Thumb symbols facing up);

Insert film cassette (which contains the negative) on top of silver tab (yellow arrow facing up and put in first);

Set timer: photo prints 809 - 1.0-1.5 minutes

transparencies 891 - 4.5 minutes;

Press white button;

After buzzer sounds, lift developer door and remove the print;

Peel the negative from the print;

Clean developer rollers with premoistened cleaning pads.

APPENDIX E: Hints for Adjusting the Matrix Camera

(by Maura C. Lohrenz)

NOTE: Computer prompts are indicated by "-" and comments are given in parentheses.

(1) Display a 16-level gray scale on the screen:

\$ ELAS	(Enter ELAS)
file.CTL	(Use any control file)
-FILE MGR?	
DU "number" DISP	(Designate usage EPA2 as DISP, where "number" is the control file number for EPA2. If EPA2 is not in the control file, type AC EPA2 DISP and proceed.)
-FILE MGR?	
COMD	(Run the display module)
-DISPLAY?	
BW IF LF	(Initialize to linear function)
-DISPLAY?	
BT	(Build color table)

-MANUAL (M), AUTO (A), LIPS TABLE (L), HUES (H) ?

M

(Enter values manually)

-START, STOP, B, R, G

0,	15,	0,	0,	0
16,	31,	15,	15,	15
32,	47,	31,	31,	31
48,	63,	47,	47,	47
64,	79,	63,	63,	63
80,	95,	79,	79,	79
96,	111,	95,	95,	95
112,	127,	111,	111,	111
128,	143,	127,	127,	127
144,	159,	143,	143,	143
160,	175,	159,	159,	159
176,	191,	175,	175,	175
192,	207,	191,	191,	191
208,	223,	207,	207,	207
224,	239,	223,	223,	223
240,	255,	239,	239,	239

DP

(Done entering color values)

-DISPLAY?

DT

(Display color table)

-DISPLAY?

END

(Exit ELAS)

- (2) Open door to camera controls (pull handle under main control panel).
Write down the current settings for RGB Exposure, Contrast and
Brightness. For example.

R G B

3.18	2.50	5.01	Exposure
6.00	5.80	6.00	Contrast
2.30	3.00	2.70	Brightness

Write down current exposure time on main control panel; for example, 1100.

- (3) Take a picture. Write current settings on back of developed print.
- (4) If image is not as desired, remove bellows from lens plate to view image on lens. Adjust above settings to correct image as you watch. The following correlations hold:

White area is affected most by contrast settings

Black area is affected most by brightness settings

Gray/overall area is affected most by exposure settings

For example:

If white area looks pink, reduce red contrast setting;

If black area looks too blue, reduce blue brightness;

If whole picture (gray area included) looks too red, reduce red exposure.

Change settings carefully. Don't over-compensate!

- (5) To maximize use of film, change number of images per print to 4 (bottom left of control panel, in FORMAT section). This will allow 4 images per picture. Try adjustments and write down adjustments made for each image. Develop picture and adjust more, if needed, following the above rules. When finished, reset number of images per print to 1.

APPENDIX F: NOS Format and Available Tapes

Each file on an NOS tape has a logical record length of 40 bytes. The following data format describes each record, where fields may contain either leading blanks or leading zeros (FORTRAN format notation is used).

<u>Columns</u>	<u>Format</u>	<u>Description</u>
1-5	I5	Survey registry number
6-8	I3	Julian day of data collection
9-11	I3	Calendar year of survey completion (last 3 digits)
12-13	I2	Degrees of latitude of data point (north only)
14-15	I2	Minutes of latitude of data point
16-19	F4.2	Seconds of latitude of data point (to hundredths)
20-22	I3	Degrees of longitude of data point (west only)
23-24	I2	Minutes of longitude of data point
25-28	F4.2	Seconds of longitude of data point (to hundredths)
29-33		Depth (format dependent on cartographic code)
34-36	I3	Cartographic code
37-40	A4	Blank

The user can obtain a list of the possible cartographic codes from any of the available documentation relating to the NOS tapes. For further information on NOS data, contact

Lt. Bruce F. Hillard

NOAA/NGDC E/GC3

325 Broadway

Boulder, CO 80303

Phone: 303 497-6376

Listed below are the NOS tapes available in the Pattern Analysis Laboratory tape library, along with the current library position and comments. Documentation describing the specific areas available on these tapes is located in the Pattern Analysis Laboratory.

<u>Tape ID</u>	<u>Rack No.</u>	<u>Comments</u>
PA979	RB216	NOS data, Gulf of Mexico, tape 1/2
PA980	RB217	NOS data, Gulf of Mexico, tape 2/2
PA981	RB218	NOS data, Caribbean, tape 1/1
PA982	RB219	NOS data, Atlantic, tape 1/5
PA983	RB220	NOS data, Atlantic, tape 2/5
PA984	RB221	NOS data, Atlantic, tape 3/5
PA985	RB222	NOS data, Atlantic, tape 4/5
PA986	RB223	NOS data, Atlantic, tape 5/5
PA987	RB224	NOS data, Pacific, tape 1/2
PA988	RB225	NOS data, Pacific, tape 2/2
PA989	RB226	NOS data, Alaska, tape 1/4
PA990	RB227	NOS data, Alaska, tape 2/4
PA991	RB228	NOS data, Alaska, tape 3/4
PA992	RB229	NOS data, Alaska, tape 4/4
PA993	RB230	NOS data, Hawaii, tape 1/1
PA994	RB231	NOS data, Great Lakes, tape 1/1

NOTE: Most NOS calibration points have latitude/longitude coordinates relative to NORTH AMERICAN DATUM 1927.

APPENDIX G: Datum Transformations

Two programs, STANDARD and ABRIDGED, may be used to calculate the adjustments to correct from a local geodetic system to World Geodetic System (WGS) 1984, the datum most commonly associated with the NOAA charts mentioned in this guide. These programs implement the standard and abridged Molodensky formulas which calculate the latitude/longitude corrections in seconds. Although the abridged formulas do not depend on geodetic height, both programs run essentially in the same manner: the user is prompted for a sample point in the local geodetic system coordinates (degrees, minutes, seconds), and the adjustments are printed to the screen.

Subroutine LOCAL_WGS84 must be adapted to the particular local geodetic system under consideration (for the most part, this will be the most common datum related to the NOS data; i.e., North American Datum, 1927). In this subroutine, the parameters (used in the correction formulae) relating to the specific local geodetic system are defined, and may be found in the DMA Technical Report 8350.2, 30 Sept. 87, DoD World Geodetic System 1984. The adjustments are given in seconds and are applied as follows in the program CORLL2LE:

$$\text{WGS latitude} = \text{Local latitude} + (\text{latitude correction}/3600)$$
$$\text{WGS longitude} = \text{Local longitude} + (\text{longitude correction}/3600)$$

For a more in-depth discussion on datum transformations, see the technical report mentioned above.

APPENDIX H: Source Code (FORTRAN)

PROGRAMS

ABRIDGED.FOR
BATH.INCLUDE (code included in other programs)
CORLL2LE.FOR
DEPTHSIEVE.FOR
DSTLL2LE.FOR
DSTMAKER.FOR
EDGE.FOR
EDGEMULTI.FOR
EOF.FOR
ERROR.FOR
FINDLL.FOR
FINDLLI.FOR
LANDWATER.FOR
LESIEVE.FOR
LINF4.FOR
LINF7.FOR
MINMAX4.FOR
MINMAX7.FOR
MODSIEVE.FOR
OVLBATHY.FOR
SPOTBATHY.FOR
STANDARD.FOR
TMBATHY.FOR
UTM2ST.FOR

PROGRAM ABRIDGED

C.....This program calculates the corrections needed to convert from
C.....local geodetic system lat/lon's to World Geodetic System 1984 (WGS84)
C.....lat/lon's. The abridged Molodensky formulas are used.

C.....Formulae and data are from DMA Technical Report 8350.2, 30 Sept 87
C.....DoD World Geodetic System 1984 (WGS84).

```
INTEGER  LATD,LATM,LOND,LONM
REAL     RLATS,RLONS      !Fractional seconds
REAL     DELLAT,DELLON    !Lat/lon corrections, in seconds
REAL     DLAT,DLON        !lat/lon to be corrected
REAL     DELH             !Geodetic height correction
REAL     WGSLAT,WGSLON    !Corrected lat/lon
```

```
WRITE(6,*) '"Local GS" to WGS 1984 conversion -'
WRITE(6,*) 'ABRIDGED MOLODENSKY FORMULAS'
WRITE(6,*)
WRITE(6,*) 'ENTER A SAMPLE POINT (IN "LOCAL GS")'
WRITE(6,*) 'Enter lat. degree (N+/S-), minutes, REAL seconds:'
READ(5,*)  LATD,LATM,RLATS
WRITE(6,*) 'Enter lon. degree (E+/W-), minutes, REAL seconds:'
READ(5,*)  LOND,LONM,RLONS
```

C.....Convert to degrees in decimal:
DLAT = DECDEG(LATD,LATM,RLATS)
DLON = DECDEG(LOND,LONM,RLONS)

C.....Echo print:
WRITE(6,*)
WRITE(6,*) 'Lat (deg,min,sec): ',LATD,LATM,RLATS
WRITE(6,*) 'Lon (deg,min,sec): ',LOND,LONM,RLONS
WRITE(6,*)
WRITE(6,*) 'Lat (dec. degree): ',DLAT
WRITE(6,*) 'Lon (dec. degree): ',DLON
WRITE(6,*)

C.....Calculate the corrections:
CALL LOCAL_WGS84(DLAT,DLON,DELLAT,DELLON,DELH)

C.....Write the corrections.
WRITE(6,*) 'Lat/lon corrections, in seconds: ',DELLAT,DELLON

WGSLAT = DLAT + (DELLAT/3600.0)
WGSLON = DLON + (DELLON/3600.0)

C.....Convert decimal degrees to degrees, minutes, REAL seconds:
CALL DMS(WGSLAT,LATD,LATM,RLATS)
CALL DMS(WGSLON,LOND,LONM,RLONS)

C.....Echo print:
WRITE(6,*)
WRITE(6,*) 'WGS lat (degree): ',WGSLAT
WRITE(6,*) 'WGS lon (degree): ',WGSLON
WRITE(6,*)
WRITE(6,*) 'WGS lat (deg,min,sec): ',LATD,LATM,RLATS
WRITE(6,*) 'WGS lon (deg,min,sec): ',LOND,LONM,RLONS
WRITE(6,*)

END

C*****

SUBROUTINE DMS(L,D,M,S)

C.....Subroutine DMS converts from decimal lat/lon L to degree D,
C.....minute M, REAL seconds S.

INTEGER D,M
REAL L,S

D = INT(L)
DIFF = ABS(L - FLOAT(D))
REALMINUTES = DIFF * 60.0
M = INT(REALMINUTES)
DIFF = REALMINUTES - FLOAT(M)
S = DIFF * 60.0

RETURN
END

C*****

REAL FUNCTION DECDEG(D,M,S)

C.....Convert latitude/longitude D,M,S (in deg,min,sec) to REAL decimal degrees.
C.....Only D should carry the sign.

REAL S, RD
INTEGER D,M

RD = FLOAT(D)
IF (RD .LT. 0.) THEN
DECDEG = -1.0 * (ABS(RD) + FLOAT(M)/60.0 + S/3600.0)
ELSE
DECDEG = RD + FLOAT(M)/60.0 + S/3600.0
ENDIF

RETURN
END

C*****

SUBROUTINE LOCAL_WGS84(DLAT, DLON, DELLAT, DELLON, DELH)

REAL DLAT, DLON, DELLAT, DELLON, DELH

C.....Subroutine to calculate corrections used to convert from LOCAL GS to
C.....WGS 1984. THE ABRIDGED MOLODENSKY FORMULAS ARE USED.

C.....NOTE: this has not been checked for elevation corrections.

C

C.....Formulae and data from DMA Technical Report 8350.2, 30 Sept 87

C.....DoD World Geodetic System 1984 (WGS84)

C

C.....To use correction factors DEL* (in seconds), etc., use these formulae

C.....where DLAT, etc., are in LOCAL and WGS LAT, etc., is in WGS84:

C

C WGS LAT = DLAT + (DELPHI/3600.0)

C WGS_LON = DLON + (DELLAMBDA/3600.0)

REAL DELX,DELY,DELZ,DELA,DELF,RN,RM,E2,A,B,H,E,F
REAL DELPHI,DELLAMBDA !Corrections, in seconds.
REAL DELHEIGHT

DATA H / 0.0 / !Ignore geodetic height.

C.....The following is for NAD27 Bahamas (should be good for Puerto Rico):

C DATA DELX,DELY,DELZ / -4.0,154.0,178.0 / !p. 7-22.

C.....The following is Clark 1866 spheroid:

C DATA DELA,DELF / -69.4,-.000037264639 / !p. 7-22.

C DATA A,F_INVERSE / 6378206.4,294.9786982 / !p. 7-12.

C.....The following data is for Puerto Rico datum (p. 7-26) for check:

DATA DELX,DELY,DELZ / 11.,72.,-101./

DATA DELA,DELF / -69.4, -0.37264639E-4/

DATA A,F_INVERSE / 6378206.4,294.9786982 / !p. 7-12.

F = 1./F_INVERSE

S1 = SIND(1.0/3600.0)

B = A*(1-F)

E2 = F*(2-F)

DENOM = SQRT(1-E2*(SIND(DLAT)**2))

RN = A / DENOM

RM = A*(1-E2) / (DENOM**3)

DELPHI = (-DELX*SIND(DLAT)*COSD(DLON) -
 DELY*SIND(DLAT)*SIND(DLON) +
 DELZ*COSD(DLAT) +
 (A*DELF + F*DELA)*SIND(2*DLAT))/
 (RM*S1)

DELLAMBDA = (-DELX*SIND(DLON) + DELY*COSD(DLON)) /
 (RN*COSD(DLAT)*S1)

DELHEIGHT = DELX*COSD(DLAT)*COSD(DLON) +
 DELY*COSD(DLAT)*SIND(DLON) + DELZ*SIND(DLAT) +
 (A*DELF + F*DELA)*(SIND(DLAT)**2) - DELA

DELLAT = DELPHI

DELLON = DELLAMBDA

DELH = DELHEIGHT

RETURN

END

C.....BATH.INCLUDE

```
COMMON // B(30000)
COMMON /DATASET/ XT(4000,7),IMAGE(4000,13)
COMMON /MISC/ DMIN,DMAX,NTERMS,LINF(7),IBAND(7),INFO,IMAGEFILE1,
+           IMAGEFILE2,IMAGETYPE,CALTYPE,IMAGETYPE1,IMAGETYPE2,
+           AVC,AVT,SDC,SDT,SIGC(3),SIGT(3),NTM,NSPOT,INTYPE,
+           IE,LE,IL,LL,IET,LFT,ILT,LLT,IES,LES,ILS,LLS,ICHAN,
+           IFIELD
CHARACTER*130 INFO
CHARACTER*40 IMAGEFILE1, IMAGEFILE2
CHARACTER*4  IMAGETYPE,CALTYPE,IMAGETYPE1,IMAGETYPE2,CALTYPE2
CHARACTER*1  INTYPE
INTEGER      IE,LE,IL,LL,IET,LET,ILT,LLT,IES,LES,ILS,LLS,ICHAN
```

```
DATA IMAGETYPE1/' TM'/
DATA IMAGETYPE2/' SPOT'/
```

```

PROGRAM CORILL2LE
C.....This program reads an original NOS file and COEFUT.LEL files
C.....to convert lat/lon's to line/elements.  If corrections are to be
C.....used, these may be entered interactively.  The depth from the NOS file
C.....is also converted to meters.
C.....The input file has the following format:
C      I5  SURVEY REGISTRY NUMBER
C      I3  JULIAN DAY
C      I3  CALENDAR YEAR
C      I2  LATITUDE DEGREE
C      I2  LATITUDE MINUTE
C      F4.2 LATITUDE SECOND
C      I3  LONGITUDE DEGREE
C      I2  LONGITUDE MINUTE
C      F4.2 LONGITUDE SECOND
C      I5  DEPTH
C      I3  CARTOGRAPHIC CODE
C      A4  BLANK

```

```

INTEGER*4      TMSCAN,TMELEM,SPOTSCAN,SPOTELEM
DOUBLE PRECISION SLTM(3),ELTM(3),SLSPOT(3),ELSPOT(3),
+              EAS,NOR,XLAT,XLON
INTEGER        LATD,LATM,LGD,LGM,ICODE
REAL           DEPTH,RLATS,RLGS,DSIGN
REAL           DELLAT,DELLON
CHARACTER*40    INFILE,OUTFILE,TCOEFUT,SCOEPUT

```

```
DATA LUCOEF1/24/,LUCOEFF2/25/,LUOUT/26/,LUSURV/11/
```

```

WRITE(6,*) 'Enter the original NOS file:'
ACCEPT 200, INFILE
OPEN(LUSURV,FILE=INFILE,STATUS='OLD',READONLY)

```

```

WRITE(6,*) 'Enter +1 or -1 as follows:'
WRITE(6,*) 'Latitude (N +1, S -1)'
ACCEPT *, LATSIGN
WRITE(6,*) 'Enter +1 or -1 as follows:'
WRITE(6,*) 'Longitude (E +1, W -1)'
ACCEPT *, LONSIGN

```

```

WRITE(6,*) 'The correction formulas are as follows:'
WRITE(6,*) '"NEW LAT" = "OLD LAT" + "CORRECTION"'
WRITE(6,*) '"NEW LON" = "OLD LON" + "CORRECTION"'
WRITE(6,*)
WRITE(6,*) 'Enter lat correction (in seconds):'
ACCEPT *, DELLAT
WRITE(6,*) 'Enter lon correction (in seconds):'
ACCEPT *, DELLON
WRITE(6,*)

```

```

WRITE(6,*) 'Enter the output file name:'
ACCEPT 200, OUTFILE
OPEN(LUOUT,FILE=OUTFILE,STATUS='NEW')

```

```

WRITE(6,*) 'Enter the TM COEFUT.LEL file:'
WRITE(6,*) '(Be certain to give full path name)'
ACCEPT 200, TCOEFUT
OPEN(LUCOEF1,FILE=TCOEFUT,STATUS='OLD',READONLY)

```

```

READ(LUCOEF1,'(1X,D60.40)') (SLTM(I),I=1,3),(ELTM(I),I=1,3)

WRITE(6,*) 'Enter the SPOT COEFUT.LEL file:'
WRITE(6,*) '(Be certain to give full path name)'
ACCEPT 200, SCOEUFUT
OPEN(LUCOEF2,FILE=SCOEUFUT,STATUS='OLD',READONLY)
READ(LUCOEF2,'(1X,D60.40)') (SLSPOT(I),I=1,3),(ELSPOT(I),I=1,3)

```

200 FORMAT(A)

```

WRITE(LUOUT,10C1) !Writes heading line to output.

```

```

DO I = 1,1000000 !There are at most 1,000,000 lines in input file.

```

```

      READ(LUSURV,500,END=100) LATD,LATM,RLATS,LGD,LGM,
+                               RLGS,DEPTH,ICODE

```

C.....Each point in the input file has a depth code format label.
C.....Only check those points with codes indicating depths in fathoms,
C.....feet, fathoms and tenths, feet and tenths, meters, or meters and tenths.
C.....These depth codes are furnished with the original NOS data tape.

```

      IF ((ICODE.EQ.126).OR.(ICODE.EQ.127).OR.
+        (ICODE.EQ.129).OR.(ICODE.EQ.130).OR.
+        (ICODE.EQ.710).OR.(ICODE.EQ.711)) THEN

```

C.....Convert degree signs appropriately and convert to decimal:

```

      XLAT = LATSIGN*DECDEG(LATD,LATM,RLATS)
      XLON = LONSIGN*DECDEG(LGD,LGM,RLGS)

```

C.....Correct according to DELLAT and DELLON:

```

      XLAT = XLAT + (DELLAT/3600.0) !DELLAT in seconds.
      XLON = XLON + (DELLON/3600.0) !DELLON in seconds.

```

C.....Convert both degree signs back to positive:

```

      XLAT = LATSIGN*XLAT
      XLON = LONSIGN*XLON

```

C.....Call to an ELAS subroutine to convert decimal lat/lon to UTM.

C.....The flag "1" signifies decimal degree input.

```

      CALL LLUTM(LATD,LATM,LATS,LGD,LGM,LGS,IZONE,NOR,EAS,
+              XLAT,XLON,1)

```

C.....The georeferencing coefficients are now used to generate scan lines
C.....and elements (relative to the corrected data) for points in the input
C.....file. Depth is converted to meters by subroutine CODES.

```

      TMSCAN = SLTM(1) + SLTM(2)*EAS + SLTM(3)*NOR +.5
      TMELEM = ELTM(1) + ELTM(2)*EAS + ELTM(3)*NOR +.5
      SPOTSCAN = SLSPOT(1) + SLSPOT(2)*EAS + SLSPOT(3)*NOR +.5
      SPOTELEM = ELSPOT(1) + ELSPOT(2)*EAS + ELSPOT(3)*NOR +.5
      CALL CODES(DEPTH,ICODE)

```

C.....Finally, convert decimal degrees back to degrees,minutes.seconds
C.....for output file.

```

      CALL DMS(XLAT,LATD,LATM,RLATS)
      CALL DMS(XLON,LGD,LGM,RLGS)

```

C.....All information writt. . out is relative to corrected data.

```

      WRITE(LUOUT,1000) LATD,LATM,RLATS,LGD,LGM,RLGS,EAS,
+                       NOR,DEPTH,TMELEM,TMSCAN,

```


RETURN
END

C*****

REAL FUNCTION DECDEG(D,M,S)

C.....Convert latitude/longitude degree,minute,second to REAL decimal

C.....degrees. NO SIGNS ARE NEEDED ON DEGREE.

INTEGER D,M

REAL S

DECDEG = FLOAT(D) + (FLOAT(M)/60.0) + (S/3600.0)

RETURN

END

PROGRAM DEPTHSIEVE

C.....This program performs a depth cut on an NOS data file.

```

INTEGER          LATD,LATM,LGD,LGM
INTEGER*4        TMSCAN,TMELEM,SPOTSCAN,SPOTELEM
DOUBLE PRECISION EAS,NOR
REAL             DEPTH,RLATS,RLGS,MAXD
CHARACTER*132    HEADING
CHARACTER*40     INFILE,OUTFILE

WRITE(6,*)
WRITE(6,*)
WRITE(6,*) '*****'
WRITE(6,*) 'This program performs a depth sieve '
WRITE(6,*) 'on an NOS data file. '
WRITE(6,*) '*****'
WRITE(6,*)
WRITE(6,*) 'Enter NOS input file name:'
ACCEPT 20, INFILE
WRITE(6,*) 'Enter output file name:'
ACCEPT 20, OUTFILE
20  FORMAT(A)
WRITE(6,*) 'Enter maximum depth (in meters):'
READ(5,*)  MAXD

OPEN(11,FILE=INFILE,STATUS='OLD',READONLY)
OPEN(12,FILE=OUTFILE,STATUS='NEW')

READ(11,500)  HEADING
WRITE(12,500) HEADING

DO I=1,1000000

    READ(11,1000,END=100) LATD,LATM,RLATS,LGD,LGM,RLGS,
+                          EAS,NOR,DEPTH,TMELEM,TMSCAN,
+                          SPOTELEM,SPOTSCAN

    IF (DEPTH .LE. MAXD) THEN
        WRITE(12,1000) LATD,LATM,RLATS,LGD,LGM,RLGS,
+                      EAS,NOR,DEPTH,TMELEM,TMSCAN,
+                      SPOTELEM,SPOTSCAN
    END IF

END DO

100  CONTINUE

500  FORMAT(A132)
1000 FORMAT(1X,I2,I2,F5.2,1X,I3,I2,F5.2,2X,F10.0,2X,F10.0,2X,
+          F7.1,2X,I6,2X,I6,2X,I6,2X,I6)
END

```

PROGRAM DSTLL2LE

C.....This program reads an original NOS file and COEFUT.LEL files
C.....to convert lat/lon's to line/element's. The depth from the
C.....NOS file is also converted to meters.

C.....The input file has the following format:

```
C      I5      SURVEY REGISTRY NUMBER
C      I3      JULIAN DAY
C      I3      CALENDAR YEAR
C      I2      LATITUDE DEGREE
C      I2      LATITUDE MINUTE
C      F4.2    LATITUDE SECOND
C      I3      LONGITUDE DEGREE
C      I2      LONGITUDE MINUTE
C      F4.2    LONGITUDE SECOND
C      I5      DEPTH
C      I3      CARTOGRAPHIC CODE
C      A4      BLANK
```

```
INTEGER*4      TMSCAN,TMELEM,SPOTSCAN,SPOTELEM
DOUBLE PRECISION SLTM(3),ELTM(3),SLSPOT(3),ELSPOT(3),
+              EAS,NOR,XLAT,XLON
INTEGER        LATD,LATM,LGD,LGM,ICODE
REAL           DEPTH,RLATS,RLGS
CHARACTER*40    INFILE,OUTFILE,TCOEFUT,SCOEFUT
```

```
DATA LUCOEF1/24/,LUCOEFF2/25/,LUOUT/26/,LUSURV/11/
```

```
WRITE(6,*) 'Enter the original NOS file:'
```

```
ACCEPT 200, INFILE
```

```
OPEN(LUSURV,FILE=INFILE,STATUS='OLD',READONLY)
```

```
WRITE(6,*) 'Enter the output file name:'
```

```
ACCEPT 200, OUTFILE
```

```
OPEN(LUOUT,FILE=OUTFILE,STATUS='NEW')
```

```
WRITE(6,*) 'Enter the TM COEFUT.LEL file:'
```

```
WRITE(6,*) '(Be certain to give full path name)'
```

```
ACCEPT 200, TCOEFUT
```

```
OPEN(LUCOEF1,FILE=TCOEFUT,STATUS='OLD',READONLY)
```

```
READ(LUCOEF1,'(1X,D60.40)') (SLTM(I),I=1,3),(ELTM(I),I=1,3)
```

```
WRITE(6,*) 'Enter the SPOT COEFUT.LEL file:'
```

```
WRITE(6,*) '(Be certain to give full path name)'
```

```
ACCEPT 200, SCOEFUT
```

```
OPEN(LUCOEF2,FILE=SCOEFUT,STATUS='OLD',READONLY)
```

```
READ(LUCOEF2,'(1X,D60.40)') (SLSPOT(I),I=1,3),(ELSPOT(I),I=1,3)
```

```
200 FORMAT(A)
```

```
WRITE(LUOUT,1001)      !Writes heading line to output.
```

```
DO I = 1,1000000      !There are at most 1,000,000 lines in input file.
```

```
READ(LUSURV,500,END=100) LATD,LATM,RLATS,LGD,LGM,
+                          RLGS,DEPTH,ICODE
```

C.....Each point in the input file has a depth code format label.

C.....Only check those points with codes indicating depths in fathoms,

C.....feet, fathoms and tenths, feet and tenths, meters, or meters and tenths.
 C.....These depth codes are furnished with the original NOS data tape.

```

      IF ((ICODE.EQ.126).OR.(ICODE.EQ.127).OR.
+       (ICODE.EQ.129).OR.(ICODE.EQ.130).OR.
+       (ICODE.EQ.710).OR.(ICODE.EQ.711)) THEN

```

C.....Convert to decimal degrees and call ELAS subroutine to convert
 C.....lat/lon to UTM. The flag "1" signifies decimal degree input.

```

      XLAT = DECDEG(LATD,LATH,RLATS)
      XLON = DECDEG(LGD,LGM,RLGS)
      CALL LLUTH(LATD,LATH,LATS,LGD,LGM,LGS,IZONE,NOR,EAS,
+       XLAT,XLON,1)

```

C.....The georeferencing coefficients are now used to generate scan lines
 C.....and elements for points of the input file.

```

      TMSCAN = SLTH(1) + SLTH(2)*EAS + SLTH(3)*NOR +.5
      TMELEM = ELTH(1) + ELTH(2)*EAS + ELTH(3)*NOR +.5
      SPOTSCAN = SLSPOT(1) + SLSPOT(2)*EAS + SLSPOT(3)*NOR +.5
      SPOTELEM = ELSPOT(1) + ELSPOT(2)*EAS + ELSPOT(3)*NOR +.5
      CALL CODES(DEPTH,ICODE)
      WRITE(LUOUT,1000) LATD,LATH,RLATS,LGD,LGM,RLGS,EAS,
+       NOR,DEPTH,TMELEM,TMSCAN,
+       SPOTELEM,SPOTSCAN

```

```

      ENDIF
      ENDDO
100 CONTINUE

```

```

500 FORMAT(11X,I2,I2,F4.2,I3,I2,F4.2,F5.0,I3)
1000 FORMAT(1X,I2,I2,F5.2,1X,I3,I2,F5.2,2X,F10.0,2X,F10.0,2X,
+       F7.1,2X,I6,2X,I6,2X,I6,2X,I6)
1001 FORMAT(1X,'LAT,LON,EAS,NOR,METERS,TMELEM,TMSCAN,SPELEM,SPSCAN:')
      END

```

C*****

SUBROUTINE CODES(DEPTH,ICODE)

```

      REAL DEPTH
      INTEGER ICODE

```

C.....This subroutine converts units to meters accordingly:

```

C      CODE 126 represents whole feet;
C      CODE 127 represents feet and tenths;
C      CODE 129 represents whole fathoms;
C      CODE 130 represents fathoms and tenths;
C      CODE 710 represents whole meters; (no conversion necessary)
C      CODE 711 represents meters and tenths;
C      For example, a depth of 234 with code 127 would represent 23.4 feet.

```

```

      IF (ICODE.EQ.126) THEN
        DEPTH = DEPTH*(.3048)
      ELSE IF (ICODE.EQ.127) THEN
        DEPTH = (DEPTH/10.)*( .3048)
      ELSE IF (ICODE.EQ.129) THEN
        DEPTH = DEPTH*6.*( .3048)
      ELSE IF (ICODE.EQ.130) THEN
        DEPTH = (DEPTH/10.)*6.*( .3048)
      ELSE IF (ICODE.EQ.711) THEN

```

DEPTH = (DEPTH/10.)
ENDIF

RETURN
END

C*****

REAL FUNCTION DECDEG(D,M,S)
C.....Convert latitude/longitude degree,minute,second to REAL decimal
C.....degrees. NO SIGNS ARE NEEDED ON DEGREE.

INTEGER D,M
REAL S

DECDEG = FLOAT(D) + (FLOAT(M)/60.0) + (S/3600.0)

RETURN
END

PROGRAM DSTMAKER

C...This program finds gray levels for pixels in imagery corresponding to calibration depths. A Data Summary File is created containing the following data in the format as indicated

RECORD #	BYTE #	CONTENTS
1	1 - 4	'BOTH'
	5 - 8	'NOS' if calib. data from NOS tape
		'NOAA' if calib. data from NOAA chart
	9 - 48	TM image file name
	49 - 88	SPOT image file name
2	1 - 3	'IET'
	4 - 7	integer*4, initial element, TM
	8 - 10	'LET'
	11 - 14	integer*4, last element, TM
	15 - 17	'ILT'
	18 - 21	integer*4, initial line, TM
	22 - 24	'LLT'
	25 - 28	integer*4, last line, TM
	29 - 31	'IES'
	32 - 35	integer*4, initial element, SPOT
	36 - 38	'LES'
	39 - 42	integer*4, last element, SPOT
	43 - 45	'ILS'
	46 - 49	integer*4, initial line, SPOT
	50 - 52	'LLS'
	53 - 56	integer*4, last line, SPOT
3	1 - 130	A130, comments
4 - NRECOUT	1 - 8	integer*8, latitude (HHMMSS)
	9 - 16	integer*8, longitude (HHMMSS)
	17 - 24	integer*8, easting
	25 - 32	integer*8, northing
	33 - 36	integer*4, tm element number
	37 - 40	integer*4, tm row number
	41 - 44	integer*4, depth (meters*10)
	45 - 48	integer*4, band 1 intensity
	49 - 52	integer*4, band 2 intensity
	53 - 56	integer*4, band 3 intensity
	57 - 60	integer*4, band 4 intensity
	61 - 64	integer*4, band 5 intensity
	65 - 68	integer*4, spot element number
	69 - 72	integer*4, spot row number
	73 - 76	integer*4, spot band 1 intensity
	77 - 80	integer*4, spot band 2 intensity
	81 - 84	integer*4, spot band 3 intensity
NRECOUT+1	1 - 8	integer*8 = 0
	9 - 16	integer*8 = 0
	17 - 24	integer*8 = 0 (flag for end of file)
	25 - 32	integer*8 = 0
	33 - 36	integer*4 = 0
	37 - 40	integer*4 = 0
	41 - 44	integer*4, no. of data records (NRECOUT)
NRECOUT+2	1 - 44	same format as NRECOUT+1

C...Good luck and good hunting.

COMMON // B(20000)
CALL HLIMIT(20000)

```

CALL HBOOK1(1,' Band 1 Intensity$',64,0.,256.,0)
CALL HCOPY(1,2,' Band 2 Intensity$')
CALL HCOPY(1,3,' Band 3 Intensity$')
CALL HCOPY(1,4,' Band 4 Intensity$')
CALL HBOOK1(5,' Band 5 Intensity$',50,0.,50.,0)
CALL HBOOK1(6,' Calibration Depths$',100,0.,100.,0)
CALL HTITLE(' USA, USM, NORDA Remotely Sensed Bathymetry, DST$')
CALL HBLACK(0)
CALL MAIN
STOP
END

```

SUBROUTINE MAIN

```

COMMON // B(20000)
COMMON /LETTERS/ IMAGEFILE1,CALFILE,INFO,IMAGEFILE2
CHARACTER*130 INFO
CHARACTER*40 IMAGEFILE1,IMAGEFILE2,CALFILE,DSTNAME,PSFILE
INTEGER INTENSET(5),INTENSES(3)
CHARACTER*1 CHAR,CALT
CHARACTER*4 CALTYPE

```

```

DATA REAST,RNORTH,RDEPTH /3*0./
DATA INTENSET,INTENSES,NEOF,NOUT,NRECIN,NRECOUT /5*0,3*0,4*0/
DATA LUIMAGE1/8/,LUIMAGE2/10/
DATA LAT,LON /2*0/

```

C...Get name of calibration file and image file

```

PRINT 300

```

```

300 FORMAT(' Enter calibration file name:')

```

```

ACCEPT 200, CALFILE

```

```

305 PRINT 310

```

```

310 FORMAT( ' Enter''N'' If calibration from NOS tape',/,

```

```

+ ' Enter''C'' If calibration from NOAA chart')

```

```

ACCEPT 200, CALT

```

```

IF (CALT .EQ. 'N' .OR. CALT .EQ. 'n') THEN

```

```

    CALTYPE = ' NOS'

```

```

ELSE IF (CALT .EQ. 'C' .OR. CALT .EQ. 'c') THEN

```

```

    CALTYPE = 'NOAA'

```

```

ELSE

```

```

    GO TO 305

```

```

END IF

```

```

PRINT 400

```

```

400 FORMAT(' Enter TM image file name:')

```

```

ACCEPT 200, IMAGEFILE1

```

```

PRINT 420

```

```

420 FORMAT(' Enter SPOT image file name:')

```

```

ACCEPT 200, IMAGEFILE2

```

```

PRINT 450

```

```

450 FORMAT(' Enter comments:')

```

```

ACCEPT 200, INFO

```

```

PRINT 460
460 FORMAT(' Enter desired name of dst file:')
ACCEPT 200, DSTNAME
PRINT 461
461 FORMAT(' Enter desired name of histogram file:')
ACCEPT 200, PSFILE

200 FORMAT(A)

C...Open calibration various input and output files
OPEN(UNIT=7,FILE=CALFILE,STATUS='OLD',READONLY)           !calib. file
OPEN(UNIT=LUIMAGE1,FILE=IMAGEFILE1,STATUS='OLD',READONLY,FORM=
+ 'UNFORMATTED',ACCESS='DIRECT',RECL=128)                 !imagery file
OPEN(UNIT=LUIMAGE2,FILE=IMAGEFILE2,STATUS='OLD',READONLY,FORM=
+ 'UNFORMATTED',ACCESS='DIRECT',RECL=128)                 !imagery file
OPEN(UNIT=9,FILE=DSTNAME,STATUS='NEW')                     !DST file
OPEN(UNIT=2,FILE=PSFILE,STATUS='new')

C...Get image size
READ(LUIMAGE1,REC=1) NBIHT,NBPRT,ILT,LLT,IET,LET,NCHANNELT
WRITE(6,411) IET,LET,ILT,LLT,NCHANNELT,NBIHT,NBPRT
READ(LUIMAGE2,REC=1) NBIHS,NBPRS,ILS,LLS,IES,LES,NCHANNELS
WRITE(6,411) IES,LES,ILS,LLS,NCHANNELS,NBIHS,NBPRS
411 FORMAT('OINITIAL ELEMENT =',I5,'      LAST ELEMENT =',I5,/,
+ '  INITIAL LINE      =',I5,'      LAST LINE      =',I5,/,
+ '  NUMBER OF CHANNELS =',I3,/,
+ '  O# OF BYTES IN ELAS HEADER =',I10,/,
+ '  # OF BYTES PER RECORD      =',I10)

C...Print headers
CALL HEADEROUT(NRECOUT,CALTYPE,IET,LET,ILT,LLT,IES,LES,ILS,LLS)

C...Read in calibration data from disk...Big DO loop
READ(7,200) JUNK           !Skip header on calib. file
DO WHILE (NEOF .EQ. 0)
  IF (CALTYPE .EQ. 'NOAA') THEN
    READ(7,10) REAST,RNORTH,RDEPTH,NCT,NRT,NCS,NRS
    10  FORMAT(4X,F10.0,2X,F10.0,2X,F7.3,4(2X,I6))
    RDEPTH = (RDEPTH*6.*0.3048)*10. !depth in fathoms, convert to m*10
  ELSE IF (CALTYPE .EQ. 'NOS') THEN
    READ(7,11) LATH,LATM,SLAT,LONH,LONM,SLON,REAST,RNORTH,
    + RDEPTH,NCT,NRT,NCS,NRS
    11  FORMAT(1X,I2,I2,F5.2,1X,I3,I2,F5.2,2X,F10.0,2X,F10.0,2X,
    + F7.1,2X,I6,2X,I6,2X,I6,2X,I6)
    LAT = 10000*LATH + 100*LATM + INT(SLAT)
    LON = 10000*LONH + 100*LONM + INT(SLON)
    RDEPTH = RDEPTH*10 !depth in meters, convert to m*10
  END IF
  IEAST = REAST
  NORTH = RNORTH
  NRECIN = NRECIN + 1           !increment input counter

C...Check for eof of calibration file
IF (NCT .EQ. 1 .AND. NRT .EQ. 1 .AND. RDEPTH .EQ. 0.) THEN
  NEOF = 1

C...Check that calibration point is in the image
ELSE IF ((NCT .GE. LET .AND. NCT .GE. IET .AND.
+ NRT .LE. LLT .AND. NRT .GE. ILT) .AND.
+ (NCS .LE. LES .AND. NCS .GE. IES .AND.
+ NRS .LE. LLS .AND. NRS .GE. ILS)) THEN
  IPLACES = 5

```

```

      CALL IMAGEIN(LUIMAGE1,NBPRT,NCHANNELT,LET,INTENSET,NCT,
+               NRT,IPLACES,ILT,IET)  !inside image, get gray level
      IPLACES = 3
      CALL IMAGEIN(LUIMAGE2,NBPRS,NCHANNELS,LES,INTENSES,NCS,
+               NRS,IPLACES,ILS,IES)  !inside image, get gray level

C...Write out the data record to the DST
      CALL IMAGEOUT(LAT,LON,IEAST,NORTH,RDEPTH,NCT,NRT,INTENSET,
+               NCS,NRS,INTENSES)
      NRECOU = NRECOU + 1
C...Periodically, let user know that something is being accomplished
      IF (MOD(NRECOU,100) .EQ. 0) WRITE(6,910) NRECOU
910    FORMAT(I7,' records written to DST file.')
      ELSE
        NOUT = NOUT + 1  !outside of at least one image
        IF ((NCT .LE. LET .AND. NCT .GE. IET) .AND.
+         (NRT .LE. LLT .AND. NRT .GE. ILT)) THEN
          IPLACES = 5
          CALL IMAGEIN(LUIMAGE1,NBPRT,NCHANNELT,LET,INTENSET,NCT,
+         NRT,IPLACES,ILT,IET)  !inside image, get gray level
        ELSE
          NCT = 0
          NRT = 0
          DO I = 1,5
            INTENSET(I) = 0
          END DO
        END IF
        IF ((NCS .LE. LES .AND. NCS .GE. IES) .AND.
+         (NRS .LE. LLS .AND. NRS .GE. ILS)) THEN
          IPLACES = 3
          CALL IMAGEIN(LUIMAGE2,NBPRS,NCHANNELS,LES,INTENSES,NCS,
+         NRS,IPLACES,ILS,IES)  !inside image, get gray level
        ELSE
          NCS = 0
          NRS = 0
          DO I = 1,3
            INTENSES(I) = 0
          END DO
        END IF
        CALL IMAGEOUT(LAT,LON,IEAST,NORTH,RDEPTH,NCT,NRT,INTENSET,
+         NCS,NRS,INTENSES)
        NRECOU=NRECOU+1
C...Periodically, let user know that something is being accomplished
      IF (MOD(NRECOU,100) .EQ. 0) WRITE(6,910) NRECOU
      END IF
    END DO

C...NRECIN counts eof flag on end of calibration file.  Correct that.
      NRECIN = NRECIN - 1
C...Record number of output records in dst header
      CALL HEADEROUT(NRECOU,CALTYPE,IET,LET,ILT,LLT,IES,LES,ILS,LLS)

C...End of Job Routine
      WRITE(6,500) NRECIN, NOUT, NRECOU
500  FORMAT('Number of calibration points read in:',I10,/,
+         'Number of calib. pts. out of range: ',I10,/,
+         '(i.e., off at least one image)',/,
+         'Number of data records written out: ',I10)
      CALL HISTDO
      RETURN

```

END

C*****

SUBROUTINE IMAGEIN(LUNIT,NBPR,NCHANNEL,LE,INTENSE,NC,NR,
+ IPLACES,IL,IE)

COMMON // B(20000)

BYTE AIM(5000,7)

INTEGER INTENSE(*), Z

C...Get gray levels of calibration pixels

C...Set NBAND according to TM or SPOT image

NINC = NBPR/512 !# of 512 byte physical records/image line

NBAND = NINC*NCHANNEL !# of bands * # of records/band

NEND = NINC - 1 !loop limit

NELEM = LE-IE+1

C...Read each 512-byte block of the input file and store it in the

C appropriate byte array...

C loop over each physical record in the logical record

NREC = 3 + NBAND*(NR-IL) !first record to read

DO Z = 0,NEND

N1 = (512*Z)+1 !first byte to read

N2 = 512*(Z+1) !last byte to read

IF (N2 .GT. NELEM) N2 = NELEM

DO K = 1,NCHANNEL !read in each channel

NUMRE = NREC + Z + NINC*(K-1) !get record number

READ(LUNIT,REC=NUMRE) (AIM(N,K),N=N1,N2)

END DO

END DO

C...Store desired gray levels

NPIX = NC-IE+1

DO K = 1,IPLACES

INTENSE(K) = AIM(NPIX,K)

IF (INTENSE(K) .LT. 0)

+ INTENSE(K) = INTENSE(K) + 256

END DO

RETURN

END

C*****

SUBROUTINE IMAGEOUT(LAT,LON,IEAST,NORTH,DEPTH,NCT,NRT,
+ INTENSET,NCS,NRS,INTENSES)

C...This subroutine writes out data records to the DST file and fills

C...histograms for info on the data set.

INTEGER INTENSET(5),INTENSES(3) !gray levels

IDEPH = DEPTH !convert to integer

WRITE(9,10) LAT,LON,IEAST,NORTH,IDEPH,NCT,NRT, !write out data record

+ (INTENSET(N),N=1,5),NCS,NRS,(INTENSES(N),N=1,3)

10 FORMAT(5I8,12I4)

C...Do some histogramming of output data

DO NH = 1,5

XX = FLOAT(INTENSET(NH))

CALL HFILL(NH,XX,0.,1.)

END DO

```

XDEPTH = DEPTH/10.
CALL HFILE(6,XDEPTH,0.,1.)
RETURN
END

```

C*****

```

SUBROUTINE HEADTOUT(NRECOUT,CALTYPE,IET,LET,ILT,LLT,
+                   IES,LES,ILS,LLS)
C...This subroutine prints out header and trailer records on DST file
COMMON /LETTERS/ IMAGEFILE1,CALFILE,INFO,IMAGEFILE2
CHARACTER*130 INFO
CHARACTER*4 CALTYPE
CHARACTER*40 IMAGEFILE1,IMAGEFILE2,CALFILE
DATA ILAT,ILON,NIEAST,INORTH,INC,INR,ICS,IRS /8*0/

C...If NRECOUT is zero, print a header record
IF (NRECOUT.EQ. 0) THEN
  WRITE(9,15) CALTYPE,IMAGEFILE1,IMAGEFILE2
  WRITE(9,40) IET,LET,ILT,LLT,IES,LES,ILS,LLS
  WRITE(9,10) INFO
ELSE
C...Otherwise print trailer records. Two are needed. Trailer records have
C...same format as data records, save a bunch of zeros and depth field is
C...filled with number of data records.
  WRITE(9,30) ILAT,ILON,NIEAST,INORTH,NRECOUT,INC,INR,ICS,IRS
  WRITE(9,30) ILAT,ILON,NIEAST,INORTH,NRECOUT,INC,INR,ICS,IRS
  CLOSE(9)
END IF

10 FORMAT(A130)
15 FORMAT(' BOTH',A4,2(A40))
30 FORMAT(5I8,4I4)
40 FORMAT(' IET',I4,' LET',I4,' ILT',I4,' LLT',I4,
+         ' IES',I4,' LES',I4,' ILS',I4,' LLS',I4)

RETURN
END

```

PROGRAM EDGE

C.....THIS PROGRAM FILTERS A ONE-CHANNEL IMAGE USING A SYMMETRIC NEAREST
C.....NEIGHBOR ROUTINE ON AREAS CLOSE TO OR CONTAINING LAND VALUES IN AN
C.....EFFORT TO PRESERVE EDGES AND FEATURES.

```

PARAMETER (N=512)           ! NUMBER OF BYTES PER BLOCK IN AN ELAS FILE
PARAMETER (MAXW=9)          ! MAXIMUM WINDOW SIZE
PARAMETER (MAXE=4500)       ! MAXIMUM NUMBER OF ELEMENTS PER LINE
PARAMETER (MAXP=MAXW*MAXW)  ! MAXIMUM NUMBER OF PIXELS IN THE WINDOW
PARAMETER (MAXC=(MAXP+1)/2) ! MAXIMUM NUMBER OF PIXELS CHOSEN
                             ! IN THE NEAREST NEIGHBOR ROUTINE SNN

```

```

BYTE          AIM(MAXE,MAXW),BIM(MAXE)
INTEGER       IMAGE(MAXP)
INTEGER       CHOSEN(MAXC)
CHARACTER*4500 C(MAXW)

```

```

INTEGER       M,NREC,M2,NREC2,NREC3,NOC,K,K1
INTEGER       NL,NE,NM1,MAX2,MAXR
INTEGER       NBIH,NBPR,IL,LL,IE,LE,NC,IDESC
CHARACTER*1   KEY
CHARACTER*40  INFILE
CHARACTER*40  OUTFILE
CHARACTER*132 COMMENT
LOGICAL       FOUND ! VARIABLE USED TO FLAG THAT A LAND PIXEL WAS FOUND

```

EQUIVALENCE (AIM,C)

```

WRITE(6,*) '*****'
WRITE(6,*) '*'
WRITE(6,*) '*'          SNN FILTER
WRITE(6,*) '*'
WRITE(6,*) '*****'
WRITE(6,*)

```

```

WRITE(6,*) 'Enter input file name: '
READ(5,200) INFILE
WRITE(6,*) 'Enter output file name: '
READ(5,200) OUTFILE
WRITE(6,*)

```

```

WRITE(6,*) 'Enter size of the filter window,'
WRITE(6,*) 'max allowed is 9:'
READ(5,*) M
IF (M.GT.MAXW) THEN
  STOP 'WINDOW IS TOO LARGE'
END IF
IF (MOD(M,2) .EQ. 0) THEN
  STOP 'WINDOW SIZE MUST BE ODD'
END IF
WRITE(6,*)

```

```

M2=M*M
NOC=(M2+1)/2 ! NUMBER OF DIAMETRICALLY OPPOSED PAIRS IN THE WINDOW
K = (M2+1)/2 ! POSITION IN ARRAY "IMAGE" OF THE CENTER PIXEL OF THE
              ! FILTER WINDOW

```

```

K1 = (M-1)/2 ! LOOP CONTROL VARIABLE TO SKIP APPROPRIATE

```

! NUMBER OF PIXELS AT BEGINNING AND END OF LINE

```
WRITE(6,*) 'Do you wish to do a'
WRITE(6,*) 'Mean - Enter 1'
WRITE(6,*) 'or a Median - Enter 2'
WRITE(6,*) 'or a Min Depth - Enter 3'
WRITE(6,*) 'filter?'
READ(5,100) KEY
IF ((KEY.NE.'1').AND.(KEY.NE.'2').AND.(KEY.NE.'3')) THEN
  STOP 'KEY IS INCORRECT'
END IF
100 FORMAT(A1)
200 FORMAT(A40)
WRITE(6,*)

OPEN(11,FILE=INFILE,ACCESS='DIRECT',
+   STATUS='OLD',READONLY,Iostat=IOS1)
OPEN(12,FILE=OUTFILE,
+   STATUS='NEW',ACCESS='DIRECT',
+   FORM='UNFORMATTED',RECL=128)

IF (IOS1.NE.0) STOP 'ERROR IN INPUT FILE'

C.....READ HEADER OF INPUT FILE.
READ(11,REC=1) NBIH,NBPR,IL,LL,IE,LE,NC,IDESC

C.....WRITE HEADER OF OUTPUT FILE.
WRITE(12,REC=1) NBIH,NBPR,IL,LL,IE,LE,NC,IDESC

C.....WRITE A COMMENT TO THE OUTFILE HEADER.
WRITE(6,*) 'Enter comments, up to 132 characters:'
READ(5,300) COMMENT
WRITE(12,REC=2) COMMENT
300 FORMAT(A80)

NREC = 3           ! SKIP HEADER OF INFILE

NL = LL - IL + 1    ! TOTAL NUMBER OF LINES
NE = LE - IE + 1    ! TOTAL NUMBER OF ELEMENTS
MAXR = INT((NBPR*NL)/N + 2) ! NUMBER OF BLOCKS IN INFILE
MAXZ = INT(NBPR/N)    ! NUMBER OF BLOCKS PER INPUT LINE (ONE CHANNEL)
MAXREC = MAXR-(M*MAXZ)+1 ! MAXIMUM VALUE THAT NREC CAN OBTAIN

C.....READ IN GRAY VALUES FOR THE FIRST M-1 LINES.
DO KOUNT = 1,M-1
  DO IZ = 0,MAXZ-1
    N1 = (N*IZ)+1
    N2 = N*(IZ+1)
    IF (N2 .GT. NE) N2 = NE
    NREC2=NREC+IZ+(KOUNT-1)*MAXZ
    READ(11,REC=NREC2) (AIM(1,KOUNT), I=N1,N2)
  ENDDO
ENDDO

DO WHILE (NREC .LE. MAXREC) ! START OF MAIN LOOP TO PROCESS ONE LINE
```

C..... THIS READ COMPLETES THE WINDOW BY READING THE Mth LINE TO BE USED.

```
DO IZ=0,MAXZ-1
  N1=(N*IZ)+1
  N2= N*(IZ+1)
  IF (N2.GT.NE) N2=NE
  NREC3 = NREC + ((M-1)*MAXZ) + IZ
  READ(11,REC=NREC3) (AIM(I,M),I=N1,N2)
END DO
```

```
NM1 = NE-M+1
DO KC = 1,NM1    ; LOOP TO PROCESS AS WINDOW MOVES ALONG LINE
```

C.....READ IN ONE WINDOW, STORE IN "IMAGE".

```
  KOUNT = 1
  DO I = 1,M
    DO J = KC,KC+M-1
      IMAGE(KOUNT) = AIM(J,I)
      IF (IMAGE(KOUNT) .LT. 0)
+      IMAGE(KOUNT) = IMAGE(KOUNT) + 256
      KOUNT = KOUNT + 1
    ENDDO
  ENDDO
```

C.....A SEARCH IS NOW PERFORMED FOR LAND PIXELS
CALL SEARCH(IMAGE,M2,FOUND)

C.....CHECK THE CENTER PIXEL. IF IT'S WATER AND THE WINDOW CONTAINS A LAND
C.....PIXEL THEN USE THE SNN FILTER; IF IT'S WATER AND THERE IS NO LAND
C.....IN THE WINDOW USE A REGULAR FILTER; OTHERWISE THE CENTER PIXEL IS
C.....LAND AND USE THE SNN FILTER.

```
IF (IMAGE(K).NE.250) THEN
  IF (FOUND) THEN
    DO L2=0,NOC-1
      CALL SNN(IMAGE(K-L2),IMAGE(K),IMAGE(K+L2),CHOSEN(L2+1))
    END DO
    IF (KEY.EQ.'1') CALL MEAN(CHOSEN,NOC,IMAGE(K))
    IF (KEY.EQ.'2') CALL MEDIAN(CHOSEN,NOC,IMAGE(K))
    IF (KEY.EQ.'3') CALL MINDEPTH(CHOSEN,NOC,IMAGE(K))
  ELSE
    IF (KEY.EQ.'1') CALL MEAN(IMAGE,M2,IMAGE(K))
    IF (KEY.EQ.'2') CALL MEDIAN(IMAGE,M2,IMAGE(K))
    IF (KEY.EQ.'3') CALL MINDEPTH(IMAGE,M2,IMAGE(K))
  END IF
ELSE
  DO L2=0,NOC-1
    CALL SNN(IMAGE(K-L2),IMAGE(K),IMAGE(K+L2),CHOSEN(L2+1))
  END DO
  IF (KEY.EQ.'1') CALL MEAN(CHOSEN,NOC,IMAGE(K))
  IF (KEY.EQ.'2') CALL MEDIAN(CHOSEN,NOC,IMAGE(K))
  IF (KEY.EQ.'3') CALL MINDEPTH(CHOSEN,NOC,IMAGE(K))  !makes no diff.
END IF

IF (IMAGE(K).GE.128) IMAGE(K) = IMAGE(K) - 256
BIM(KC+K1) = IMAGE(K)

ENDDO    ! KC LOOP
```

C.....WRITE TO OUTPUT FILE.

```
DO IZ = 0,MAXZ-1
  N1 = (N*IZ)+1
  N2 = N*(IZ+1)
  IF (N2 .GT. NE) N2 = NE
  NREC2=NREC+IZ+((K1)*MAXZ)
  WRITE(12,REC = NREC2) (BIM(I),I=N1,N2)
END DO
```

C.....THE LAST M-1 LINES OF THE WINDOW ARE COPIED TO THE FIRST M-1 LINES
C.....AND THE WINDOW IS COMPLETED BY THE READ AT THE BEGINNING OF THE LOOP.

```
DO I=1,M-1
  C(I) = C(I+1)
END DO
```

NREC = NREC+MAXZ ! INCREMENT NREC TO SKIP TO NEXT LINE

ENDDO . ! END "WHILE" LOOP

C....."BLANK OUT" THE REMAINING BLOCKS USING NREC2

```
NREC2 = NREC2 + 1
DO WHILE (NREC2 .LE. MAXR)
  WRITE(12,REC=NREC2)
  NREC2 = NREC2 + 1
END DO
```

```
WRITE(6,*)
WRITE(6,*) '***** FILTER COMPLETED *****'
WRITE(6,*)
```

END

C*****

SUBROUTINE SNN(B,C,D,A)

C.....COMPARES DIAMETRICALLY OPPOSED PAIRS AND SELECTS THE NEAREST NEIGHBOR

C.....B AND D ARE THE PAIR MEMBERS, C IS THE CENTER PIXEL AND

C.....A IS THE SELECTED VALUE

INTEGER B,C,D,A

CC = C + C

IF ((B + D) .GT. CC) THEN

IF (B .GT. D) THEN

A = D

ELSE

A = B

ENDIF

ELSE IF ((B + D) .LT. CC) THEN

IF (B .GT. D) THEN

A = B

ELSE

A = D

ENDIF

ELSE

A = C

ENDIF

RETURN

END

```

C*****
      SUBROUTINE MEAN(GVALUES,NEL,MIDPIX)
C.....CALCULATES AVERAGE OF THE VALUES IN GVALUES, AN ARRAY OF SIZE NEL,
C.....AND ASSIGNS THIS AVERAGE TO MIDPIX
      INTEGER GVALUES(*), NEL, MIDPIX
      REAL      SUM

      SUM=0.0
      DO I=1,NEL
          SUM = SUM + GVALUES(I)
      END DO
      MIDPIX = NINT((SUM/NEL))

      RETURN
      END

```

```

C*****
      SUBROUTINE MEDIAN(GVALUES,NEL,MIDPIX)
C.....CALCULATES MEDIAN OF THE VALUES IN GVALUES, AN ARRAY OF SIZE NEL,
C.....AND ASSIGNS THIS MEDIAN IN MIDPIX
      INTEGER GVALUES(*), NEL, MIDPIX, TEMP, ENDVAL
      LOGICAL SORTED

      SORTED = .FALSE.
      ENDVAL = NEL-1
      DO WHILE(.NOT.(SORTED))
          SORTED = .TRUE.
          DO 200 I=1,ENDVAL
              IF (GVALUES(I) .GT. GVALUES(I+1)) THEN
                  TEMP = GVALUES(I)
                  GVALUES(I) = GVALUES(I +1)
                  GVALUES(I+1) = TEMP
                  SORTED = .FALSE.
              END IF
          200 CONTINUE
          ENDVAL = ENDVAL-1
      END DO

      IF (MOD(NEL,2) .EQ. 0) THEN
          TEMPPIX = (FLOAT(GVALUES(NEL/2) + GVALUES((NEL/2)+1)))/2
      ELSE
          TEMPPIX = FLOAT(GVALUES((NEL+1)/2))
      END IF

      MIDPIX = NINT(TEMPPIX)
      RETURN
      END

```

```

C*****
      SUBROUTINE SEARCH(GVALUES,NEL,FOUND)
C.....SEARCHES FOR LAND PIXELS IN WINDOW
      INTEGER GVALUES(*), NEL
      LOGICAL FOUND

      FOUND = .FALSE.
      I = 1
      DO WHILE ((.NOT. FOUND) .AND. (I .LE. NEL))
          IF (GVALUES(I) .EQ. 250) THEN
              FOUND = .TRUE.
          END IF
          I = I + 1
      END DO

```

```
ELSE
  I = I + 1
END IF
END DO
```

```
RETURN
END
```

```
C*****
```

```
  SUBROUTINE MINDEPTH(GVALUES,NEL,MIDPIX)
```

```
C.....SEARCHES FOR THE MINIMUM DEPTH OF THE WATER PIXELS IN THE WINDOW.
```

```
  INTEGER  GVALUES(*),NEL,MIDPIX
```

```
  INTEGER  MIN
```

```
  MIN = 500
```

```
  IF (MIDPIX.NE.250) THEN
```

```
    DO I = 1,NEL
```

```
      IF ((GVALUES(I) .LT. MIN).AND. (GVALUES(I).NE.250)) THEN
```

```
        MIN = GVALUES(I)
```

```
      END IF
```

```
    END DO
```

```
    MIDPIX = MIN
```

```
  END IF
```

```
  RETURN
```

```
  END
```

PROGRAM EDGEMULTI

C.....This program filters an image file (one or more channels) using a
C.....symmetric nearest neighbor routine in an effort to preserve edges and
C.....features.

```

PARAMETER (N=512)           !Number of bytes per block.
PARAMETER (MAXW=9)          !Maximum window size.
PARAMETER (MAXE=4500)        !Maximum number of elements per line.
PARAMETER (MAXCH=7)          !Maximum number of channels in input file.

```

```

PARAMETER (MAXP=MAXW*MAXW)    !Maximum number of pixels in window.
PARAMETER (MAXC=(MAXP+1)/2)    !Maximum number of pixels chosen
                                ! in the nearest neighbor routine SNN.

```

```

BYTE          AIM(MAXE,MAXW),BIM(MAXE)
INTEGER        IMAGE(MAXP),CHOSEN(MAXC)
INTEGER        CHAN(MAXCH)
CHARACTER*4500 C(MAXW)

```

```

INTEGER        M,NREC,M2,NREC2,NREC3,NOC,K,K1
INTEGER        NL,NE,NM1,MAXZ,MAXR,ISKIP,IC,NUMCHAN
INTEGER        NBIH,NBPR,IL,LL,IE,LE,NC,IDESC
CHARACTER*1    KEY
CHARACTER*40    INFILE,OUTFILE
CHARACTER*132   COMMENT
LOGICAL        FOUND

```

EQUIVALENCE (AIM,C)

```

WRITE(6,*) '*****'
WRITE(6,*) '*'
WRITE(6,*) '*'          SNN FILTER          '*'
WRITE(6,*) '*'
WRITE(6,*) '*****'
WRITE(6,*)

```

```

WRITE(6,*) 'Enter input file name: '
READ(5,100) INFILE
WRITE(6,*) 'Enter output file name: '
READ(5,100) OUTFILE
WRITE(6,*)

```

```

OPEN(11,FILE=INFILE,ACCESS='DIRECT',
+      STATUS='OLD',READONLY,IOSTAT=IOS1)
OPEN(12,FILE=OUTFILE,
+      STATUS='NEW',ACCESS='DIRECT',
+      FORM='UNFORMATTED',RECL=128)
IF (IOS1.NE.0) STOP '** Something's screwy about input file **'

```

C.....Read header of input file.
READ(11,REC=1) NBIH,NBPR,IL,LL,IE,LE,NC,IDESC

C.....Write header of output file.
WRITE(12,REC=1) NBIH,NBPR,IL,LL,IE,LE,NC,IDESC

```

WRITE(6,50) NC
50 FORMAT(1X,'Input file has ',I2,' channel(s).')
WRITE(6,*) 'Enter the number of channels to be filtered:'
READ(5,*) NUMCHAN

```

```
WRITE(6,*) 'Enter the channel(s) to be filtered:'
READ(5,*) (CHAN(I),I=1,NUMCHAN)
```

```
WRITE(6,*) 'Enter size of the filter window,'
WRITE(6,*) 'maximum allowed is 9:'
READ(5,*) M
IF (M.GT.MAXW) STOP '*** Window is too large ***'
IF (MOD(M,2).EQ.0) STOP '*** Window size must be odd ***'
WRITE(6,*)
```

```
M2=M*M
NOC=(M2+1)/2      !Number of diametrically opposed pairs in a window.
K = (M2+1)/2      !Position in array IMAGE of the window's center pixel.
K1 = (M-1)/2      !Loop control variable used to skip appropriate
                  ! number of pixels at beginning and end of line.
```

```
WRITE(6,*) 'Do you wish to do a'
WRITE(6,*) 'Mean - Enter 1'
WRITE(6,*) 'or a Median - Enter 2'
WRITE(6,*) 'or a Minimum Value - Enter 3'
WRITE(6,*) 'filter?'
READ(5,100) KEY
IF ((KEY.NE.'1').AND.(KEY.NE.'2').AND.(KEY.NE.'3')) THEN
  STOP '*** Key is incorrect ***'
END IF
WRITE(6,*)
```

```
C.....Write comment to the output file header.
WRITE(6,*) 'Enter comments, up to 132 characters:'
READ(5,100) COMMENT
WRITE(12,REC=2) COMMENT
WRITE(6,*)
```

```
100 FORMAT(A)
```

```
NL = LL - IL + 1      !Total number of lines.
NE = LE - IE + 1      !Total number of elements.
MAXR = INT((NBPR*NL*NC)/N + 2) !Number of blocks in input file.
MAXZ = INT(NBPR/N)     !Number of N-byte blocks per line.
```

```
C.....Perform the filter by channel...BIG do loop!
DO IC = 1,NUMCHAN
```

```
  !Skip to appropriate channel, line 1:
  NREC = 3 + (CHAN(IC)-1)*MAXZ
  WRITE(6,150) CHAN(IC)
150  FORMAT(1X,'Processing channel ',I2)
  !Compute the maximum value NREC can obtain:
  MAXREC = MAXR - (M*NC*MAXZ) + (CHAN(IC)-1)*MAXZ + 1
```

```
C.....Read in gray values for the first M-1 lines.
```

```
  DO KOUNT = 1,M-1
    DO IZ = 0,MAXZ-1
      N1 = (N*IZ)+1
      N2 = N*(IZ+1)
      IF (N2 .GT. NE) N2 = NE
```

```

      NREC2 = NREC + IZ + NC*(KOUNT-1)*MAXZ
      READ(11,REC=NREC2) (AIM(I,KOUNT), I=N1,N2)
    ENDDO
  ENDDO

```

DO WHILE (NREC .LE. MAXREC) !Start of main loop to process one line.

C.....This read completes the MxM window by reading the Mth line.

```

    DO IZ=0,MAXZ-1
      N1=(N*IZ)+1
      N2= N*(IZ+1)
      IF (N2.GT.NE) N2=NE
      NREC3 = NREC + ((M-1)*MAXZ*NC) + IZ
      READ(11,REC=NREC3) (AIM(I,M),I=N1,N2)
    END DO

```

```

    NM1 = NE-M+1
    DO KC = 1,NM1 !Loop to process as window moves along line.

```

C.....Read in one window, store in IMAGE.

```

      KOUNT = 1
      DO I = 1,M
        DO J = KC,KC+M-1
          IMAGE(KOUNT) = AIM(J,I)
          IF (IMAGE(KOUNT) .LT. 0)
            + IMAGE(KOUNT) = IMAGE(KOUNT) + 256
          KOUNT = KOUNT + 1
        ENDDO
      ENDDO

```

C=====

C.....This section of code deals with window processing.

```

    DO L2=0,NOC-1
      CALL SNN(IMAGE(K-L2),IMAGE(K),IMAGE(K+L2),CHOSEN(L2+1))
    END DO
    IF (KEY.EQ.'1') CALL MEAN(CHOSEN,NOC,IMAGE(K))
    IF (KEY.EQ.'2') CALL MEDIAN(CHOSEN,NOC,IMAGE(K))
    IF (KEY.EQ.'3') CALL MINVAL(CHOSEN,NOC,IMAGE(K))

    IF (IMAGE(K).GE.128) IMAGE(K) = IMAGE(K) - 256
    BIM(KC+K1) = IMAGE(K)

```

C=====

ENDDO !KC loop

C.....Write to output file.

```

    DO IZ = 0,MAXZ-1
      N1 = (N*IZ)+1
      N2 = N*(IZ+1)
      IF (N2 .GT. NE) N2 = NE
      NREC2 = NREC + IZ + (K1*NC*MAXZ)
      WRITE(12,REC = NREC2) (BIM(I),I=N1,N2)
    END DO

```

C.....The last M-1 lines of the window are copied to the first M-1 lines

C.....and the window is completed by the read at the beginning of the loop.

```

    DO I=1,M-1
      C(I) = C(I+1)

```

END DO

NREC = NREC + NC*MAXZ !Increment NREC to skip to the next line.

ENDDO !End "WHILE" loop.

C....."Blank out" the remaining blocks in ICHAN using NREC2.

ISKIP = (NC-1)*MAXZ !To skip appropriate channels.

NREC2 = NREC2 + 1 + ISKIP

DO WHILE (NREC2 .LE. (MAXR-(NC-CHAN(IC))*MAXZ))

DO IZ=0,MAXZ-1

NREC2 = NREC2 + IZ

WRITE(12,REC=NREC2)

ENDDO

NREC2 = NREC2 + 1 + ISKIP

ENDDO

ENDDO !BIG do loop on IC.

C.....Now, copy the unfiltered channel.

DO IC = 1,NC

CALL SEARCH(CHAN,IC,NC,FOUND)

IF (.NOT. FOUND) THEN !Copy the unfiltered channel.

WRITE(6,160) IC

160 FORMAT(1X,'Copying channel ',I2)

NREC = 3 + (IC-1)*MAXZ

MAXREC = MAXR - (NC-IC+1)*MAXZ + 1

DO WHILE (NREC .LE. MAXREC)

DO IZ=0,MAXZ-1

N1 = (N*IZ)+1

N2 = N*(IZ+1)

IF (N2 .GT. NE) N2 = NE

READ(11,REC=NREC+IZ) (BIM(I),I=N1,N2)

WRITE(12,REC=NREC+IZ) (BIM(I),I=N1,N2)

ENDDO

NREC = NREC + NC*MAXZ

ENDDO

ENDIF

ENDDO

WRITE(6,*)

WRITE(6,*) '***** FILTER COMPLETED *****'

WRITE(6,*)

END

C*****

SUBROUTINE SNN(B,C,D,A)

C.....SNN compares diametrically opposed pixel pairs and selects the nearest

C.....neighbor. B and D are the pair members, C is the center pixel and

C.....A is the selected value.

INTEGER B,C,D,A

CC = C + C

IF ((B + D) .GT. CC) THEN

IF (B .GT. D) THEN

A = D

```

ELSE
  A = B
ENDIF
ELSE IF ((B + D) .LT. CC) THEN
  IF (B .GT. D) THEN
    A = B
  ELSE
    A = D
  ENDIF
ELSE
  A = C
ENDIF
RETURN
END

```

C*****

```

SUBROUTINE MEAN(GVALUES,NEL,MIDPIX)
C....MEAN calculates the average of the values in GVALUES, an array of size
C....NEL, and assigns this average to MIDPIX.
INTEGER GVALUES(*), NEL, MIDPIX
REAL SUM

```

```

SUM=0.0
DO I=1,NEL
  SUM = SUM + GVALUES(I)
END DO
MIDPIX = NINT((SUM/NEL))

```

```

RETURN
END

```

C*****

```

SUBROUTINE MEDIAN(GVALUES,NEL,MIDPIX)
C....MEDIAN calculates the median of the values in GVALUES, an array of size
C....NEL, and assigns this median to MIDPIX
INTEGER GVALUES(*), NEL, MIDPIX, TEMP, ENDVAL
LOGICAL SORTED

```

```

SORTED = .FALSE.
ENDVAL = NEL-1
DO WHILE(.NOT.(SORTED))
  SORTED = .TRUE.
  DO 200 I=1,ENDVAL
    IF (GVALUES(I) .GT. GVALUES(I+1)) THEN
      TEMP = GVALUES(I)
      GVALUES(I) = GVALUES(I + 1)
      GVALUES(I+1) = TEMP
      SORTED = .FALSE.
    END IF
200  CONTINUE
  ENDVAL = ENDVAL-1
END DO

IF (MOD(NEL,2) .EQ. 0) THEN
  TEMPPIX = (FLOAT(GVALUES(NEL/2) + GVALUES((NEL/2)+1)))/2
ELSE
  TEMPPIX = FLOAT(GVALUES((NEL+1)/2))
END IF

MIDPIX = NINT(TEMPPIX)

```

```
RETURN
END
```

```
C*****
```

```
  SUBROUTINE MINVAL(GVALUES,NEL,MIDPIX)
```

```
C.....MINVAL searches for the minimum value pixel in the filter
```

```
C.....window, and assigns this value to MIDPIX.
```

```
  INTEGER GVALUES(*),NEL,MIDPIX
```

```
  INTEGER MIN
```

```
  MIN = GVALUES(1)
```

```
  DO I = 2,NEL
```

```
    IF (GVALUES(I) .LT. MIN) MIN = GVALUES(I)
```

```
  END DO
```

```
  MIDPIX = MIN
```

```
RETURN
```

```
END
```

```
C*****
```

```
  SUBROUTINE SEARCH(CHAN,ICHAN,NC,FOUND)
```

```
C.....SEARCH searches for channel ICHAN in CHAN.
```

```
  INTEGER CHAN(*),NC,I
```

```
  LOGICAL FOUND
```

```
  FOUND = .FALSE.
```

```
  I = 1
```

```
  DO WHILE ((.NOT. FOUND) .AND. (I .LE. NC))
```

```
    IF (CHAN(I) .EQ. ICHAN) THEN
```

```
      FOUND = .TRUE.
```

```
    ELSE
```

```
      I = I + 1
```

```
    END IF
```

```
  END DO
```

```
RETURN
```

```
END
```

```

PROGRAM EOF
C.....Program EOF writes an end-of-file flag to a .NOS file or NOAA file
C.....to be used as input for DSTMAKER.
CHARACTER*40 INFILE
CHARACTER*1  CALT

WRITE(6,*) 'This program accepts a file and writes an'
WRITE(6,*) 'end-of-file marker used by DSTMAKER.'
WRITE(6,*) 'This file must be the final, sieved file to be'
WRITE(6,*) 'used by DSTMAKER.'
WRITE(6,*)

WRITE(6,*) 'Enter file:'
READ(5,100) INFILE
OPEN(8,FILE=INFILE,STATUS='OLD',ACCESS='APPEND')
WRITE(6,*) 'Enter "N" if calibration from NOS tape'
WRITE(6,*) 'Enter "C" if calibration from NOAA chart:'
READ(5,100) CALT

IF (CALT .EQ. 'N' .OR. CALT .EQ. 'n') THEN
  WRITE(8,101)
ELSE IF (CALT .EQ. 'C' .OR. CALT .EQ. 'c') THEN
  WRITE(8,102)
ELSE
  WRITE(6,*) 'NO END-OF-FILE RECORD WRITTEN'
ENDIF

100 FORMAT(A)
101 FORMAT(51X,'0.0',7X,'1',7X,'1')
102 FORMAT(30X,'0.0',9X,'1',7X,'1')

END

```

PROGRAM ERROR

C.....This program takes a 1-channel land/water image and plots calibration
 C.....points taken from a sorted (or, inefficiently, from an unsorted) DST file.

```

BYTE          AIM(4220) !Max. number of elements = 4220.
INTEGER*4      ICCOUNT, IDEPTH, IMAGE, NCT, NRT, NV(5), L(5), LAT
REAL*4         A(0:4), ERR
CHARACTER*8     SPACE8
CHARACTER*40    DSTFILE, LWFILE, ERRORFILE, OUTFILE
CHARACTER*132   COMMENTS
LOGICAL         ONIMAGE
  
```

```

DATA SPACE8 / ' ' /
  
```

```

WRITE(6,*) '*****'
WRITE(6,*) '* ERROR-IMAGE CREATION *'
WRITE(6,*) '*****'
WRITE(6,*)
  
```

```

WRITE(6,*) 'Enter DST file:'
ACCEPT 12, DSTFILE
WRITE(6,*) 'Enter land/water image file (1-channel):'
ACCEPT 12, LWFILE
WRITE(6,*) 'Enter error-image output file:'
ACCEPT 12, ERRORFILE
WRITE(6,*) 'Enter output file (for summary information):'
ACCEPT 12, OUTFILE
WRITE(6,*) 'Enter comments (limit to 132 characters):'
ACCEPT 12, COMMENTS
12  FORMAT(A)
  
```

```

OPEN(20, FILE=DSTFILE, STATUS='OLD', READONLY)
  
```

```

OPEN(25, FILE=LWFILE, FORM='UNFORMATTED', READONLY,
+      ACCESS='DIRECT', STATUS='OLD', RECL=128)
  
```

C.....Read header from land/water image file.
 READ(25, REC=1) NBIH, NBPR, IL, LL, IE, LE, NC, IDESC
 IF (NC .NE. 1) STOP 'ERROR *** input image must be 1-channel.'

```

OPEN(30, FILE=ERRORFILE, FORM='UNFORMATTED',
+      ACCESS='DIRECT', STATUS='NEW', RECL=128)
  
```

C.....Write header to error-image file.
 WRITE(30, REC=1) NBIH, NBPR, IL, LL, IE, LE, NC, IDESC

```

OPEN(35, FILE=OUTFILE, STATUS='NEW')
  
```

```

CALL GETINFO(L, A) !To get L infinities and coefficients.
  
```

```

NE = LE-IE+1 !Number of elements.
NL = LL-IL+1 !Number of lines.
MAXZ = INT(NBPR/512) !Total number of 512-byte blocks per line.
MAXREC = MAXZ*NL + 2 !Total number of records in input file.
  
```

C.....Copy the land/water image file into the error-image file.

```

NREC = 3
DO WHILE (NREC .LE. MAXREC)
  DO I=0,MAXZ-1
    N1 = (512*I)+1
    N2 = 512*(I+1)
    IF (N2 .GT. NE) N2 = NE
    READ(25,REC=NREC+I) (AIM(N), N=N1,N2)
    WRITE(30,REC=NREC+I) (AIM(N), N=N1,N2)
  ENDDO
  NREC = NREC + MAXZ
ENDDO !"WHILE"
CLOSE(25)

```

C.....Skip header records on DST file.

```

READ(20,*)
READ(20,*)
READ(20,*)

```

C.....Read first record from DST file.

```

READ(20,1000) IEAST,IDEPTH,NCT,NRT,(NV(I),I=1,5)
SAVEROW = NRT
NPTS = 0
NTHROW = 0

```

```

DO WHILE (IEAST .NE. 0)

```

C.....Test to see if point is on the image, both line- and element-wise.

```

ONIMAGE = (NCT .GE. IE .AND. NCT .LE. LE) .AND.
+ (SAVEROW .GE. IL .AND. SAVEROW .LE. LL)

```

```

IF (ONIMAGE) THEN

```

C.....Read in the entire line.

```

NREC = 3 + MAXZ*(SAVEROW-IL)
DO Z = 0,MAXZ-1
  N1 = (512*Z)+1
  N2 = 512*(Z+1)
  IF (N2.GT.NE) N2=NE
  READ(30,REC=NREC+Z) (AIM(I),I=N1,N2)
ENDDO

```

```

DO WHILE ((NRT .EQ. SAVEROW) .AND. (IEAST .NE. 0))

```

```

IF (ONIMAGE) THEN

```

```

RDEPTH = FLOAT(IDEPTH)/10.

```

```

CDEPTH = NINT(A(0) +

```

```

A(1)*ALOG(MAX(FLOAT(NV(1))-L(1)),1.0)) +

```

```

A(2)*ALOG(MAX(FLOAT(NV(2))-L(2)),1.0)) +

```

```

A(3)*ALOG(MAX(FLOAT(NV(3))-L(3)),1.0)) +

```

```

A(4)*ALOG(MAX(FLOAT(NV(4))-L(4)),1.0)))

```

```

ERR = RDEPTH - CDEPTH !Actual depth - calculated depth.

```

```

CALL CHECK(ERR,ICOUNT,IMAGE) !Find the appropriate error range.

```

```

IF (IMAGE.GE.128)

```

```

IMAGE = IMAGE - 256

```

```

AIM(NCT-IE+1)=IMAGE

```

```

NPTS = NPTS + 1

```

```

ELSE

```

```

      NTHROW = NTHROW + 1
    ENDIF

    READ(20,1000) IEAST,IDEPTH,NCT,NRT,(NV(I),I=1,5)
    !Test to see if point is on image, element-wise.
    ONIMAGE = (NCT .GE. IE .AND. NCT .LE. LE)

    ENDDO !Inner while loop

    SAVEROW = NRT !A new row was encountered, so save the new row.

C.....Write out the completed line.
    DO Z = 0,MAXZ-1
      N1 = (512*Z)+1
      N2 = 512*(Z+1)
      IF (N2.GT.NE) N2=NE
      WRITE(30,REC=NREC+Z) (AIM(I),I=N1,N2)
    ENDDO

    ELSE !A point not on the image was found, so skip it.

      NTHROW = NTHROW + 1
      READ(20,1000) IEAST,IDEPTH,NCT,NRT,(NV(I),I=1,5)
      SAVEROW = NRT

    ENDIF

  ENDDO !Outer while loop

C.....Write comments, consisting of L infinities, coefficients, files, count,
C.....and COMMENTS.
    WRITE(30,REC=2) 'First 5 words ',
+                  'are L1-L5, next ',
+                  ' 5 words are ',
+                  'A0-A4 * 1000, ',
+                  'next word is ',
+                  'no. of points: ',
+                  (L(I),I=1,5),
+                  (INT(A(I)*1000.),I=0,4),NPTS,0,
+                  'DST file: ',
+                  DSTFILE,SPACE8,
+                  'input file: ',
+                  LWFILE,SPACE8,
+                  'summary file: ',
+                  OUTFILE,SPACE8,
+                  'comments: ',
+                  COMMENTS

    WRITE(35,1001) ICOUNT
    WRITE(6,1001) ICOUNT
    WRITE(35,1002) NPTS
    WRITE(6,1002) NPTS
    WRITE(35,1003) NTHROW
    WRITE(6,1003) NTHROW

    WRITE(6,*)
    WRITE(6,*) '*****'
    WRITE(6,*) '* COMPLETED *'
    WRITE(6,*) '*****'

```

```

1000  FORMAT(16X,I8,8X,I8,7I4)
1001  FORMAT(1X,'Total number of points with error < -8 or
+ error > 8 = ',I8)
1002  FORMAT(1X,'Total number of points plotted = ',I8)
1003  FORMAT(1X,'Total number of points out of range = ',I8)

```

END

C*****

SUBROUTINE CHECK(ERR,ICOUNT,IMAGE)

C.....This subroutine determines which range the error is in and

C.....assigns an appropriate value to the pixel.

```

INTEGER*4 ICOUNT,IMAGE
REAL*4     ERR

```

```

IF (ERR .LE. -8.) THEN
  ICOUNT = ICOUNT + 1
  IMAGE = 99
ELSE IF ((ERR .GT. -8.) .AND. (ERR .LE. -7.)) THEN
  IMAGE = 100
ELSE IF ((ERR .GT. -7.) .AND. (ERR .LE. -6.)) THEN
  IMAGE = 101
ELSE IF ((ERR .GT. -6.) .AND. (ERR .LE. -5.)) THEN
  IMAGE = 102
ELSE IF ((ERR .GT. -5.) .AND. (ERR .LE. -4.)) THEN
  IMAGE = 103
ELSE IF ((ERR .GT. -4.) .AND. (ERR .LE. -3.)) THEN
  IMAGE = 104
ELSE IF ((ERR .GT. -3.) .AND. (ERR .LE. -2.)) THEN
  IMAGE = 105
ELSE IF ((ERR .GT. -2.) .AND. (ERR .LE. -1.)) THEN
  IMAGE = 106
ELSE IF ((ERR .GT. -1.) .AND. (ERR .LE. 0.)) THEN
  IMAGE = 107
ELSE IF ((ERR .GT. 0.) .AND. (ERR .LE. 1.)) THEN
  IMAGE = 108
ELSE IF ((ERR .GT. 1.) .AND. (ERR .LE. 2.)) THEN
  IMAGE = 109
ELSE IF ((ERR .GT. 2.) .AND. (ERR .LE. 3.)) THEN
  IMAGE = 110
ELSE IF ((ERR .GT. 3.) .AND. (ERR .LE. 4.)) THEN
  IMAGE = 111
ELSE IF ((ERR .GT. 4.) .AND. (ERR .LE. 5.)) THEN
  IMAGE = 112
ELSE IF ((ERR .GT. 5.) .AND. (ERR .LE. 6.)) THEN
  IMAGE = 113
ELSE IF ((ERR .GT. 6.) .AND. (ERR .LE. 7.)) THEN
  IMAGE = 114
ELSE IF ((ERR .GT. 7.) .AND. (ERR .LE. 8.)) THEN
  IMAGE = 115
ELSE IF (ERR .GT. 8.) THEN
  ICOUNT = ICOUNT + 1
  IMAGE = 116
ENDIF

```

```

RETURN
END

```

C*****

SUBROUTINE GETINFO(L,A)

C.....This subroutine prompts user for L infinities and coefficients

C.....necessary for depth calculation.

INTEGER*4 L(5)
REAL*4 A(0:4)

WRITE(6,*)
WRITE(6,*) 'Enter L infinities in the following order:'
WRITE(6,*) 'TM bands 1-5'
WRITE(6,*)

DO M=1,5
WRITE(6,75) M
WRITE(6,*) '(If no L infinity, enter 0)'
READ(5,*) L(M)
END DO

75 FORMAT(1X,'Enter L infinity for band ',I1)

WRITE(6,*)
WRITE(6,*) 'Enter coefficients in the following order:'
WRITE(6,*) 'A0'
WRITE(6,*) 'A1-A4 for TM bands 1-4'
WRITE(6,*)

DO M=0,4
WRITE(6,76) M
WRITE(6,*) '(If no coefficient, enter 0)'
READ(5,*) A(M)
END DO

76 FORMAT(1X,'Enter coefficient A(',I1,')')

RETURN
END

```

PROGRAM FINDLL
C.....Program to test for NOS calibration points in a desired
C.....lat/lon area.
C.....Each record of the input file has the following format:
C      I5    SURVEY REGISTRY NUMBER
C      I3    JULIAN DAY
C      I3    CALENDAR YEAR
C      I2    LATITUDE DEGREE
C      I2    LATITUDE MINUTE
C      F4.2  LATITUDE SECOND
C      I3    LONGITUDE DEGREE
C      I2    LONGITUDE MINUTE
C      F4.2  LONGITUDE SECOND
C      I5    DEPTH
C      I3    CARTOGRAPHIC CODE
C      A4    BLANK

```

```

INTEGER      LATD,LATM,LOND,LONM,ICODE
REAL         DEPTH,RLATS,RLONS,DLAT,DLON
REAL         MAXLAT,MAXLON,MINLAT,MINLON
CHARACTER*40 INFILE,OUTFILE

```

```
DATA LUOUT/26/,LUSURV/11/
```

```

WRITE(6,*) '*****'
WRITE(6,*) 'This program searches an original '
WRITE(6,*) 'NOS file for points in a particular'
WRITE(6,*) 'latitude/longitude area '
WRITE(6,*) '*****'

```

```

WRITE(6,*) 'Enter the original NOS file:'
ACCEPT 200, INFILE
OPEN(LUSURV,FILE=INFILE,STATUS='OLD',READONLY)

```

```

WRITE(6,*) 'Enter the output file name:'
ACCEPT 200, OUTFILE
OPEN(LUOUT,FILE=OUTFILE,STATUS='UNKNOWN')

```

```

WRITE(6,*) 'Enter maximum latitude degree, minute, REAL seconds:'
READ(5,*) LATD,LATM,RLATS
WRITE(6,*) 'Enter maximum longitude degree, minute, REAL seconds:'
READ(5,*) LOND,LONM,RLONS
MAXLAT = DECDEG(LATD,LATM,RLATS)
MAXLON = DECDEG(LOND,LONM,RLONS)

```

```

WRITE(6,*) 'Enter minimum latitude degree, minute, REAL seconds:'
READ(5,*) LATD,LATM,RLATS
WRITE(6,*) 'Enter minimum longitude degree, minute, REAL seconds:'
READ(5,*) LOND,LONM,RLONS
MINLAT = DECDEG(LATD,LATM,RLATS)
MINLON = DECDEG(LOND,LONM,RLONS)

```

```
200 FORMAT(A)
```

```
DO I = 1,1000000 !There are at most 1,000,000 points in file.
```

```

READ(LUSURV,500,END=100) LATD,LATM,RLATS,LOND,LONM,
+                          RLONS,DEPTH,ICODE

```

```

DLAT = DECDEG(LATD,LATH,RLATS)
DLON = DECDEG(LOND,LONM,RLONS)
IF (DLAT .GE. MINLAT .AND. DLAT .LE. MAXLAT .AND.
+   DLON .GE. MINLON .AND. DLON .LE. MAXLON) THEN
    !Write out the info and "up" the counter.
    WRITE(LUOUT,600) LATD,LATH,INT(RLATS*100),LOND,LONM,
+   INT(RLONS*100),DEPTH,ICODE
    K = K + 1
ELSE
    L = L + 1
ENDIF

```

ENDDO

```

100 CONTINUE
WRITE(6,*) 'in range = ',K,' out of range = ',L
500 FORMAT(11X,I2,I2,F4.2,I3,I2,F4.2,A5,I3)
600 FORMAT(11X,I2,I2,I4,I3,I2,I4,A5,I3)

```

END

C*****

REAL FUNCTION DECDEG(D,M,S)

C.....Convert latitude/longitude degree,minute,second to REAL decimal
C.....degrees.

```

INTEGER D,M
REAL S

```

DECDEG = FLOAT(D) + (FLOAT(M)/60.0) + (S/3600.0)

```

RETURN
END

```

PROGRAM FINDLLI

C.....Program to test for NOS calibration points in a desired
C.....lat/lon area, with cartographic code 126, 127, 129, 130, 710, or 711.
C.....Each record of the input file has the following format:

```
C      I5  SURVEY REGISTRY NUMBER
C      I3  JULIAN DAY
C      I3  CALENDAR YEAR
C      I2  LATITUDE DEGREE
C      I2  LATITUDE MINUTE
C      F4.2 LATITUDE SECOND
C      I3  LONGITUDE DEGREE
C      I2  LONGITUDE MINUTE
C      F4.2 LONGITUDE SECOND
C      I5  DEPTH
C      I3  CARTOGRAPHIC CODE
C      A4  BLANK
```

```
INTEGER      LATD,LATM,LOND,LONM,ICODE
REAL         DEPTH,RLATS,RLONS,DLAT,DLO
REAL         MAXLAT,MAXLON,MINLAT,MINLO
CHARACTER*40 INFILE,OUTFILE
```

DATA LUOUT/26/,LUSURV/11/

```
WRITE(6,*) '*****'
WRITE(6,*) 'This program searches an original '
WRITE(6,*) 'NOS file for points in a particular'
WRITE(6,*) 'latitude/longitude area '
WRITE(6,*) '*****'
```

```
WRITE(6,*) 'Enter the original NOS file:'
ACCEPT 200, INFILE
OPEN(LUSURV,FILE=INFILE,STATUS='OLD',READONLY)
```

```
WRITE(6,*) 'Enter the output file name:'
ACCEPT 200, OUTFILE
OPEN(LUOUT,FILE=OUTFILE,STATUS='UNKNOWN')
```

```
WRITE(6,*)
WRITE(6,*) 'ENTER POSITIVE MAXIMUM/MINIMUM VALUES'
WRITE(6,*) 'Enter maximum latitude degree, minute, REAL seconds:'
READ(5,*) LATD,LATM,RLATS
WRITE(6,*) 'Enter maximum longitude degree, minute, REAL seconds:'
READ(5,*) LOND,LONM,RLONS
MAXLAT = DECDEG(LATD,LATM,RLATS)
MAXLON = DECDEG(LOND,LONM,RLONS)
```

```
WRITE(6,*) 'Enter minimum latitude degree, minute, REAL seconds:'
READ(5,*) LATD,LATM,RLATS
WRITE(6,*) 'Enter minimum longitude degree, minute, REAL seconds:'
READ(5,*) LOND,LONM,RLONS
MINLAT = DECDEG(LATD,LATM,RLATS)
MINLON = DECDEG(LOND,LONM,RLONS)
```

200 FORMAT(A)

DO I = 1,1000000 !There are at most 1,000,000 points in file.

```

      READ(LUSURV,500,END=100) ISV,LATD,LATM,RLATS,LOND,LONM,
+      RLONS,DEPTH,ICODE
+   IF (ICODE.EQ.126 .OR. ICODE.EQ.127
+      .OR. ICODE.EQ.129 .OR. ICODE.EQ.130
+      .OR. ICODE.EQ.710 .OR. ICODE.EQ.711) THEN
      DLAT = DECDEG(LATD,LATM,RLATS)
      DLON = DECDEG(LOND,LONM,RLONS)
      IF (DLAT .GE. MINLAT .AND. DLAT .LE. MAXLAT .AND.
+      DLON .GE. MINLON .AND. DLON .LE. MAXLON) THEN
        !Write out the info and "up" the counter.
        WRITE(LUOUT,600) ISV,LATD,LATM,INT(RLATS*100),LOND,
+      LONM,INT(RLONS*100),DEPTH,ICODE
      K = K + 1
    ELSE
      L = L + 1
    ENDIF
  ENDIF
ENDDO

100 CONTINUE
WRITE(6,*) 'in range = ',K,' out of range = ',L
500 FORMAT(I5,6X,I2,I2,F4.2,I3,I2,F4.2,A5,I3)
600 FORMAT(I5,6X,I2,I2,I4,I3,I2,I4,A5,I3)

```

END

C*****

```

      REAL FUNCTION DECDEG(D,M,S)
C.....Convert latitude/longitude degree,minute,second to REAL decimal
C.....degrees.  NO SIGNS ARE NEEDED ON DEGREE.

```

```

      INTEGER D,M
      REAL S

```

```

      DECDEG = FLOAT(D) + (FLOAT(M)/60.0) + (S/3600.0)

```

```

      RETURN
      END

```

PROGRAM LANDWATER
 C.....This program creates a two-class image (land/water) with one channel
 C.....so that calibration points (or errors at these points) can be
 C.....plotted.

```

PARAMETER      (N=512)      !Number of bytes per block.
BYTE           AIM(4500)
INTEGER        IMAGE(4500)
INTEGER        M,P,Q,I,J,N3,N1,N2,Z
CHARACTER*40    TMINFILE,OUTFILE
CHARACTER*132   COMMENT

```

```

WRITE(6,*) '*****'
WRITE(6,*) '*      LAND/WATER IMAGE CREATION      *'
WRITE(6,*) '*****'
WRITE(6,*)

```

```

LUN1 = 0
CALL FILE_OPEN(LUN1,TMINFILE)

```

```

WRITE(6,*)
WRITE(6,*) 'Enter output file:'
READ(5,50) OUTFILE
50  FORMAT(A)
OPEN(UNIT=15,FILE=OUTFILE,STATUS='NEW',
+    FORM='UNFORMATTED',ACCESS='DIRECT',RECL=128)

```

C.....Read header of input file.
 READ(LUN1,REC=1) NBIH,NBPR,IL,LL,IE,LE,NC,IDESC

C.....Write header of the output file.
 WRITE(15,REC=1) NBIH,NBPR,IL,LL,IE,LE,1,IDESC

```

WRITE(6,*) 'What comments would you like written to the'
WRITE(6,*) 'output file? (Limit to 132 characters.)'
READ(5,200) COMMENT
WRITE(15,REC=2) COMMENT
200  FORMAT(A)

```

```

NREC = 3                                ! Input data for Line 1 channel 1 begins
                                           !   in this block of the input file.
NREC3 = 3                               ! Control for writing to the correct
                                           !   position of the output file.
NL = LL-IL+1                            ! Total number of lines.
NE = LE-IE+1                            ! Total number of elements.
MAXR = INT((NBPR*NL*NC)/N+2)             ! Number of blocks in infile.
MAXZ = INT(NBPR/N)                       ! Number of blocks per line per channel.
MAXREC = MAXR-(NC*MAXZ)+1                ! Block MAXREC is the start of last line
                                           !   channel 1.

```

```

DO WHILE (NREC .LE. MAXREC)

```

C.....Read each 512-byte block of the input file and store it in a byte array.
 C.....All the information of one line of input data (channel 5) is read here.

```

DO Z = 0,MAXZ-1
  N1 = (512*Z)+1
  N2 = 512*(Z+1)
  IF(N2.GT.NE) N2 = NE

```

```

      NREC2 = NREC+Z+((NC-1)*MAXZ)  !Only band 5 is read in.
      READ(LUN1,REC=NREC2)  (AIM(N3), N3=N1,N2)
END DO

```

C.....Convert bytes to integer*4 and change negatives to positives.

```

      DO NCOL = 1,NE
        IMAGE(NCOL) = AIM(NCOL)
        IF (IMAGE(NCOL) .LT. 0)
+         IMAGE(NCOL) = IMAGE(NCOL) + 256
      END DO

```

```

      CALL BIGD(IMAGE,AIM,NE)

```

C.....Converts positives bigger than 128 to negatives for storage as bytes in
C.....byte array bathy.

```

      DO J = 1,NE
        .IF (AIM(J) .GE. 128)
+         AIM(J) = AIM(J) - 256
      END DO

```

C.....Write to output file the calculated values.

```

      DO Z = 0,MAXZ-1
        N1 = (512*Z)+1
        N2 = 512*(Z+1)
        IF(N2.GT.NE) N2 = NE
        WRITE(15,REC=NREC3+Z)  (AIM(N3), N3=N1,N2)
      END DO
      NREC3 = NREC3 + MAXZ  !Increment NREC3 to skip to first block
                           !of next line.

```

C.....Increment NREC so that the program reads the gray levels for the next
C.....scan line.

```

      NREC = NREC + (NC*MAXZ)

END DO

```

```

WRITE(6,*) '*****'
WRITE(6,*) '*'
WRITE(6,*) '*   COMPLETED   *'
WRITE(6,*) '*'
WRITE(6,*) '*****'
WRITE(6,*)
END

```

C*****

```

SUBROUTINE BIGD(IMAGE,AIM,NE)

```

C.....This subrounine calculates the value for each pixel.

```

      INTEGER IMAGE(4500),NE
      BYTE    AIM(4500)

      DO 400 J = 1,NE
        IF ((IMAGE(J) - 10) .GT. 0) THEN

```

```

        AIM(J) = 250 !Land
      ELSE
        AIM(J) = 255 !Water
      END IF
400    CONTINUE
      RETURN
    END

```

C*****

SUBROUTINE FILE_OPEN(LUN1,TMINFILE)

```

      INTEGER      LUN1
      CHARACTER*40  TMINFILE

```

```

      WRITE(6,*) 'Enter TM input file:'
      WRITE(6,*) '(must have 5 channels since value 10 from'
      WRITE(6,*) 'channel 5 is used as a land/water threshold)'
      READ(5,100) TMINFILE
      LUN1 = 10
      OPEN(UNIT=LUN1,FILE=TMINFILE,STATUS='OLD',IOSTAT=IOS1,
+        READONLY,FORM='UNFORMATTED',ACCESS='DIRECT',RECL=128)
100  FORMAT(A)

      RETURN
    END

```

PROGRAM LESIEVE

C.....This program performs a line/element sieve on an NOS data file.

```

INTEGER          LATD,LATH,LGD,LGM
INTEGER*4        TMSCAN,TMELEM,SPOTSCAN,SPOTELEM
DOUBLE PRECISION EAS,NOR
REAL             DEPTH,RLATS,RLGS
CHARACTER*132    HEADING
CHARACTER*40     INFILE, OUTFILE

```

```

WRITE(6,*)
WRITE(6,*)
WRITE(6,*) '*****'
WRITE(6,*) 'This program performs a line/element '
WRITE(6,*) 'sieve on an NOS data file. '
WRITE(6,*) '*****'
WRITE(6,*)
WRITE(6,*) 'Enter NOS input file name:'
ACCEPT 20, INFILE
WRITE(6,*) 'Enter output file name:'
ACCEPT 20, OUTFILE

```

20 FORMAT(A)

```

OPEN(11,FILE=INFILE,STATUS='OLD')
OPEN(12,FILE=OUTFILE,STATUS='NEW')

```

```

READ(11,500) HEADING
WRITE(12,500) HEADING

```

```

WRITE(6,*) 'Please enter TM Initial and Last Line:'
ACCEPT *,ILT,LLT

```

```

WRITE(6,*) 'Please enter TM Initial and Last Element:'
ACCEPT *,IET,LET

```

```

WRITE(6,*) 'Please enter SPOT Initial and Last Line:'
WRITE(6,*) '(if using junk coefut files, enter 1 1)'
ACCEPT *,ILS,LLS

```

```

WRITE(6,*) 'Please enter SPOT Initial and Last Element:'
WRITE(6,*) '(if using junk coefut files, enter 1 1)'
ACCEPT *,IES,LES

```

DO I=1,336000

```

      READ(11,1000,END=100) LATD,LATH,RLATS,LGD,LGM,RLGS,
+      EAS,NOR,DEPTH,TMELEM,TMSCAN,
+      SPOTELEM,SPOTSCAN

```

```

      IF (((TMELEM.GE.IET).AND.(TMELEM.LE.LET).AND.
+      (TMSCAN.GE.ILT).AND.(TMSCAN.LE.LLT)).OR.
+      ((SPOTELEM.GE.IES).AND.(SPOTELEM.LE.LES).AND.
+      (SPOTSCAN.GE.ILS).AND.(SPOTSCAN.LE.LLS))) THEN
        WRITE(12,1000) LATD,LATH,RLATS,LGD,LGM,RLGS,
+      EAS,NOR,DEPTH,TMELEM,TMSCAN,
+      SPOTELEM,SPOTSCAN

```

END IF

END DO

100 CONTINUE

500 FORMAT(A132)

1000 FORMAT(1X,I2,I2,F5.2,1X,I3,I2,F5.2,2X,F10.0,2X,F10.0,2X,
+ F7.1,2X,I6,2X,I6,2X,I6,2X,I6)

END

PROGRAM LINF4

C...This program does a paretas & spero model fit to the data
 C...It quizzes the user for the number of bands and bands to use, and
 C...bases the regression on L infinity slices.
 C...It reads in imagery data from the new DST file, asking the user for
 C...the file name.

```

  INCLUDE 'USER$DISK:[BATHY.SOURCE]BATH.INCLUDE'
  CALL HLIMIT(20000)
  CALL HBOOK2(4,' ACTUAL DEPTH VS. CALCULATED DEPTHS',
+           50,0.,50.,35,0.,35.,16)
  CALL HBOOK2(10,' ACT.-CALC. DEPTH VS. ACT. DEPTHS',
+           40,0.,20.,40,-10.,10.,256)
  CALL HBOOK1(15,' PER CENT ERROR, CALIB. PTS.$',
+           50,0.,100.,256)
  CALL HBOOK1(20,' RESIDUALS, ACT. DEPTH - CALC. DEPTHS',
+           60,-15.,15.,256)
  CALL HCOPY(15,31,' PER CENT ERROR, TEST PTS.$')
  CALL HCOPY(4,32,' TEST DEPTHS; ACT. DEPTH VS. CALC. DEPTHS',
+           50,0.,50.,35,0.,35.,16)
  CALL HCOPY(10,33,' TEST DEPTHS; ACT.-CALC. VS. ACT. $',
+           40,0.,20.,40,-10.,10.,16)
  CALL HCOPY(20,34,' TEST DEPTH RESIDUALS, ACT. - CALC.$',
+           60,-15.,15.,256)
  CALL HBLACK(0)
  CALL HTITLE(' USA, USM, NORDA.  Satellite Bathymetry$')
  CALL MAIN
  CALL EXIT
  END
  
```

SUBROUTINE MAIN

C...Subroutine to do Multiple Linear Regression driving

```

  CHARACTER*40 DSTFILE, PSFILE
  LOGICAL FOUND
  
```

```

  INCLUDE 'USER$DISK:[BATHY.SOURCE]BATH.INCLUDE'
  
```

C...Array IMAGE contains the following data

```

C      IMAGE(N,1) = column of Nth calibration point
C      ,2) = row of Nth calibration point
C      3 = depth * 10 in meters
C      4 = 1st band gray level
C      5 = 2nd band gray level
C      6 = 3rd band gray level
C      7 = 4th band gray level (if tm imagery used)
C      8 = 5th band gray level (if tm imagery used)
  
```

```

  DIMENSION X(4000), Y(4000), SIGMAY(4000), M(10), YFIT(4000),
+           A(10), SIGMAA(10), R(10), RSIG(10)
  
```

C...Define error mode for subroutine REGRESS

```

  DATA MODE/0/
  DATA DMIN/0./,DMAX/0./
  
```

C...Open needed files

```

  WRITE(6,3)
  
```

```

3 FORMAT('Enter name of DST file to use.')
  READ(5,4) DSTFILE
4 FORMAT(A)
  PRINT 7
7 FORMAT(' Enter name of output file.')
  ACCEPT 4, PSFILE
  OPEN(UNIT=7,FILE=DSTFILE,STATUS='OLD',READONLY)    !data file
  OPEN(UNIT=2,FILE=PSFILE,STATUS='NEW')              !hardcopy output file
  PRINT 110, DSTFILE, PSFILE
  WRITE(2,110) DSTFILE, PSFILE
110 FORMAT(1H1,' Input filename = ',A/,
+         ' Output bathy filename = ',A)

```

C...Go get some needed information from the user
CALL GETINFO

C...Get some more needed information from the user
CALL LINE_ELEM

```

IF (IMTYPE .EQ. 'T' .OR. IMTYPE .EQ. 't') THEN
  IE = IET
  LE = LET
  IL = ILT
  LL = LLT
ELSE IF (IMTYPE .EQ. 'S' .OR. IMTYPE .EQ. 's') THEN
  IE = IES
  LE = LES
  IL = ILS
  LL = LLS
END IF

WRITE(2,20) IE,LE
WRITE(2,21) IL,LL

```

C...Go read in calibration data and gray levels from disk

```

IF (IMTYPE .EQ. 'T' .OR. IMTYPE .EQ. 't') THEN
  CALL DATATM(1,NPTS)    !1 indicates data to be used in regression
ELSE IF (IMTYPE .EQ. 'S' .OR. IMTYPE .EQ. 's') THEN
  CALL DATASPOT(1,NPTS) !1 indicates data to be used in regression
END IF

```

C...Gather calibration and corresponding data points into one array

```

NE = 0
DO NK = 1,NPTS
  FOUND = .FALSE.
  CALL FINDIT(FOUND,NK)
  IF (FOUND) THEN
    NE = NE + 1
  END IF

```

C...Set up arrays for multiple linear regression

```

X(NE) = NE
DO K = 1,NTERMS
  XT(NE,K) = ALOG(FLOAT(MAX(IMAGE(NK,K+3)-LINF(K),1)))
END DO
Y(NE) = FLOAT(IMAGE(NK,3))/10.
ELSE
  NTHROW = NTHROW + 1
END IF

```

END DO

```
WRITE(6,555) NE,NTHROW
WRITE(2,555) NE,NTHROW
IF(NE .LT. NTERMS+2) THEN
  WRITE(6,556)
  WRITE(2,556)
  STOP
END IF
```

C...Go call the multiple linear regression stuff

```
CALL REGRESS(X,Y,SIGMAY,NE,NTERMS,M,O,YFIT,AO,A,SIGMAO,SIGMAA,
+           R,RMUL,CHISQR,FTEST)
```

C...Loop over calibration depths. Calculate residuals.

```
DO N = 1, NE
  CALCZ = YFIT(N)
  Z = Y(N)
  PCE = ABS(((CALCZ-Z)/Z)*100.)
  CALL HFILL(15,PCE,0.,1.)
  CALL HFILL(20,Z-CALCZ,0.,1.)
  CALL HFILL(10,Z,Z-CALCZ,1.)
  CALL HFILL(4,CALCZ,Z,1.)
END DO
```

C...End of Job Routine

C...Write fit info to screen

```
WRITE(6,200)
WRITE(6,205)
WRITE(6,210) AO,SIGMAO
WRITE(6,215) (IBAND(K),A(K),SIGMAA(K), K=1,NTERMS)
WRITE(6,218)
WRITE(6,220) (IBAND(K),R(K), K=1,NTERMS)
WRITE(6,225) RMUL
WRITE(6,230) CHISQR, FTEST
```

C...Write fit info to output file

```
WRITE(2,200)
WRITE(2,205)
WRITE(2,210) AO,SIGMAO
WRITE(2,215) (IBAND(K),A(K),SIGMAA(K), K = 1,NTERMS)
WRITE(2,218)
WRITE(2,220) (IBAND(K),R(K), K = 1,NTERMS)
WRITE(2,225) RMUL
WRITE(2,230) CHISQR, FTEST
```

C.. Let the user know about what's going on.

```
WRITE(6,240)
```

C...Loop to check resids of non calibration points

```
IF (IMTYPE .EQ. 'T' .OR. IMTYPE .EQ. 't') THEN
  CALL DATATM(2,NPTS) !get test calib. pts.
ELSE IF (IMTYPE .EQ. 'S' .OR. IMTYPE .EQ. 's') THEN
  CALL DATASPOT(2,NPTS) !get test calib. pts.
END IF
DO N = 1, NPTS
  FOUND = .FALSE.
  CD = FLOAT(IMAGE(N,3))/10.
  CALL FINDIT(FOUND,N)
  IF (FOUND) THEN
    DO MM = 1,NTERMS
```

```

        RSIG(MM) = FLOAT(MAX(IMAGE(N,MM+3)-LINF(MM),1))
    END DO
    ZT = AO
    DO MM =1, NTERMS
        ZT = ZT + A(MM)*ALOG(RSIG(MM))
    END DO
    PCE = ABS(((ZT-CD)/CD)*100.)
    CALL HFILL(31,PCE,0.,1.)
    CALL HFILL(32,ZT,CD,1.)
    CALL HFILL(33,CD,CD-ZT,1.)
    CALL HFILL(34,CD-ZT,0.,1.)
END IF
END DO

```

C...Let user know what is happening
 WRITE(6,260)

C...fit Gaussian distribution to residuals and then print the histograms
 CALL HFITGA(20,C3,AVC,SDC,CHI2C,12,SIGC) !calibration points
 CALL HFITGA(34,C3,AVT,SDT,CHI2T,12,SIGT) !test points
 CALL HISTDO

C....Go print summary information on fit
 WRITE(6,265)
 CALL SUMMARY(AO,A,SIGMAO,SIGMAA,R,RMUL)

C...Lets get out of here. Tell user we're done.
 WRITE(6,270)
 RETURN

C...FORMAT statements

```

    5 FORMAT(1X,I10,' Calibration points read in.',/,
      +      1X,I10,' Calibration points outside of image.')
    10 FORMAT(31X,F7.3,2X,F6.1,2X,F6.1)
    20 FORMAT(' IE ',I4,' LE ',I4)
    21 FORMAT(' IL ',I4,' LL ',I4)
    90 FORMAT(1X,I3,2X,I3,2X,F4.1)
    100 FORMAT(I3,2X,I3,F5.1)
    200 FORMAT(///'O      ---- RESULTS OF MULTIPLE LINEAR'
      +      ' REGRESSION ----'//)
    205 FORMAT('OFitted Parameter Values')
    210 FORMAT(' AO = 'F8.3,' +/- ',F8.4)
    215 FORMAT(' A',I1,' = ',F8.3,' +/- ',F8.4)
    218 FORMAT('OLinear Correlation Coefficients')
    220 FORMAT(' R',I1,' = ',F8.3)
    225 FORMAT(' Multiple Correlation Coefficient, RM = ',F8.3)
    230 FORMAT('OCHISQ = ',F8.3,' FTEST = ',F10.3)
    240 FORMAT('ONow gathering statistics using test points...')
    250 FORMAT('ODuplicate calibration point....NR, NC, OLD DEPTH, NEW',
      +      2I5,2F8.1)
    260 FORMAT('ONow Playing: Histograms and Scatter Plots!')
    265 FORMAT('OPrinting out summary.')
    270 FORMAT('OJob completed. I'm outta here.')
    555 FORMAT(1H0,I10,' Calibration points to be used in regression.',
      +      /,1H0,I10,' Calibration points out of range.')
    556 FORMAT(1H0,' INSUFFICIENT DATA FOR REGRESSION! STOPPING!!')
    2000 FORMAT(A1)
END

```

C*****

SUBROUTINE GETINFO

INCLUDE 'USER\$DISK:[BATHY.SOURCE]BATH.INCLUDE'

C...Read the header records from DST file.

READ(7,5) IMAGETYPE,CALTYPE,IMAGEFILE1,IMAGEFILE2

READ(7,6) IET,LET,ILT,LLT,IES,LES,ILS,LLS

READ(7,7) INFO

5 FORMAT(4A)

6 FORMAT(8(3X,I4))

7 FORMAT(A130)

WRITE(6,10)

10 FORMAT(' SATELLITE BATHYMETRY!')

WRITE(6,20)

20 FORMAT(' Enter''T'' for TM imagery',/,

+ ' Enter''S'' for SPOT imagery')

ACCEPT 25,IMTYPE

25 FORMAT(A)

WRITE(6,30)

30 FORMAT('OEnter number of bands to use in fit.')

READ(5,*) NTERMS

WRITE(6,35)

35 FORMAT('OEnter band(s) to use in fit:')

READ(5,*) (IBAND(N),N=1,NTERMS)

WRITE(6,40) NTERMS

40 FORMAT('OEnter the LINF''s for the ',I1,' band(s):')

READ(5,*) (LINF(N),N=1,NTERMS)

C...Go write out info to output file

CALL INFOUT

RETURN

END

C*****

SUBROUTINE INFOUT

INCLUDE 'USER\$DISK:[BATHY.SOURCE]BATH.INCLUDE'

WRITE(2,10)

10 FORMAT(' SATELLITE BATHYMETRY')

WRITE(2,12) IMAGETYPE, CALTYPE, IMAGEFILE1,IMAGEFILE2

12 FORMAT('OImagery from the ',A,' sensor. Calibration from ',A,/,

+ ' Image names are ',A,A)

WRITE(2,14) INFO

14 FORMAT('OComments entered on this DST file are:',/,1H ,A)

WRITE(2,*)

WRITE(2,*) ' Regression done based on Linf slices.'

WRITE(2,*)

```

WRITE(2,20) NTERMS, (IBAND(N),N=1,NTERMS), DMIN, DMAX
20 FORMAT('0Using',I3,' bands of imagery',/,
+       ' Band(s) ',<NTERMS>(1X,I2),/,
+       ' Minimum calibration depth is',F4.0,/,
+       ' Maximum calibration depth is',F4.0)
WRITE(2,25)
25 FORMAT('0The L infinities are...')
DO N = 1,NTERMS
  WRITE(2,30) IBAND(N), LINF(N)
30  FORMAT(' LINF(',I1,') = ',I3)
END DO
RETURN
END

```

C*****

```

SUBROUTINE DATATM(LCALL,NP)
C...This subroutine reads in TM imagery data from the dst file and
C  stores it in array IMAGE.

```

```

  INCLUDE 'USER$DISK:[BATHY.SOURCE]BATH.INCLUDE'
  DIMENSION INTENSE(5)
  NP = 0

```

```

C...Rewind file and skip header records of DST file.
  REWIND(7)
  READ(7,*)
  READ(7,*)
  READ(7,*)

```

```

C...Read the first data record
  READ(7,10) LAT,LON,NEAST,NORTH,ID,NC,NR,
+           (INTENSE(N),N=1,5)
10  FORMAT(5I8,7I4)

  DO WHILE (NEAST .NE. 0)
    IF (LCALL .EQ. 1) THEN
      IF (INTENSE(1) .GT. 0) THEN
        NP = NP + 1
        IMAGE(NP,1) = NC
        IMAGE(NP,2) = NR
        IMAGE(NP,3) = ID
        DO J = 1,NTERMS
          IMAGE(NP,J+3) = INTENSE(IBAND(J))
        ENDDO
        IMAGE(NP,8) = INTENSE(5) !Band 5 is used as a land/water cut.
      END IF

```

```

C...Skip a record to use as a test point
  READ(7,10) LAT,LON,NEAST,NORTH,ID,NC,NR,
+           (INTENSE(N),N=1,5)
  READ(7,10) LAT,LON,NEAST,NORTH,ID,NC,NR,
+           (INTENSE(N),N=1,5)

```

```

  ELSE IF (LCALL .EQ. 2) THEN
C...First record used as a calib. point; skip it.
  READ(7,10) LAT,LON,NEAST,NORTH,ID,NC,NR,
+           (INTENSE(N),N=1,5)

```

```

      IF (INTENSE(1) .GT. 0) THEN
        NP = NP + 1
        IMAGE(NP,1) = NC
        IMAGE(NP,2) = NR
        IMAGE(NP,3) = ID
        DO J = 1, NTERMS
          IMAGE(NP,J+3) = INTENSE(IBAND(J))
        ENDDO
        IMAGE(NP,8) = INTENSE(5) !Band 5 is used as a land/water cut.
      END IF
C...Skip next next record as it was used as a calib. point, too.
      READ(7,10) LAT,LON,NEAST,NORTH,ID,NC,NR,
+      (INTENSE(N),N=1,5)
      END IF
    END DO

    RETURN
  END

```

C*****

```

      SUBROUTINE DATASPOT(LCALL,NP)
C...This subroutine reads in SPOT imagery data from the dst file and
C   stores it in array IMAGE.

```

```

      INCLUDE 'USER$DISK:[BATHY.SOURCE]BATH.INCLUDE'
      DIMENSION INTENSE(5)
      NP = 0

```

```

C...Rewind file and skip header records of DST file.
      REWIND(7)
      READ(7,*)
      READ(7,*)
      READ(7,*)

```

```

C...Read the first data record
      READ(7,10) LAT,LON,NEAST,NORTH,ID,NC,NR,
+      (INTENSE(N),N=1,3)

10   FORMAT(5I8,28X,5I4) !(The "28X" skips the TM values.)

```

```

      DO WHILE (NEAST .NE. 0)
        IF (LCALL .EQ. 1) THEN
          IF (INTENSE(1) .GT. 0) THEN
            NP = NP + 1
            IMAGE(NP,1) = NC
            IMAGE(NP,2) = NR
            IMAGE(NP,3) = ID
            DO J = 1, NTERMS
              IMAGE(NP,J+3) = INTENSE(IBAND(J))
            ENDDO
          END IF
C...Skip a record to use as a test point
          READ(7,10) LAT,LON,NEAST,NORTH,ID,NC,NR,
+          (INTENSE(N),N=1,3)
          READ(7,10) LAT,LON,NEAST,NORTH,ID,NC,NR,
+          (INTENSE(N),N=1,3)

          ELSE IF (LCALL .EQ. 2) THEN

```

```

C...First record used as a calib. point; skip it.
  READ(7,10) LAT,LON,NEAST,NORTH,ID,NC,NR,
    +      (INTENSE(N),N=1,3)
  IF (INTENSE(1) .GT. 0 ) THEN
    NP = NP + 1
    IMAGE(NP,1) = NC
    IMAGE(NP,2) = NR
    IMAGE(NP,3) = ID
    DO J = 1, NTERMS
      IMAGE(NP,J+3) = INTENSE(IBAND(J))
    ENDDO
  END IF
C...Skip next next record as it was used as a calib. point, too.
  READ(7,10) LAT,LON,NEAST,NORTH,ID,NC,NR,
    +      (INTENSE(N),N=1,3).
  END IF
END DO
RETURN
END

```

C*****

```

  SUBROUTINE REGRESS(X,Y,SIGMAY,NPTS,NTERMS,M,MODE,YFIT,AO,A,
    +      SIGMAO,SIGMAA,R,RMUL,CHISQR,FTEST)
  COMMON /DATASET/ XT(4000,7),IMAGE(4000,13)
  DIMENSION X(4000),Y(4000),SIGMAY(4000),M(10),YFIT(4000),A(10),
    +      SIGMAA(10),R(10)
  DIMENSION WEIGHT(4000), XMEAN(10), SIGMAX(10), ARRAY(10,10)
  DIMENSION INDEX(10) !scratch space for matrix inversion routine

```

C...INITIALIZE SUMS AND ARRAYS

```

11 SUM = 0.
  YMEAN = 0.
  SIGMA = 0.
  CHISQ = 0.
  RMUL = 0.
  DO 17 I = 1, NPTS
17 YFIT(I) = 0.
21 DO 28 J = 1, NTERMS
  XMEAN(J) = 0.
  SIGMAX(J) = 0.
  DO 28 K=1, NTERMS
28 ARRAY(J,K) = 0.

```

C...ACCUMULATE WEIGHTS

```

30 DO 50 I=1,NPTS
31 IF (MODE) 32,37,39
32 IF (Y(I)) 35, 37, 33
33 WEIGHT(I) = 1./(-Y(I))
  GO TO 41
35 WEIGHT(I) = 1./ (-Y(I))
  GO TO 41
37 WEIGHT(I) = 1.
  GO TO 41
39 WEIGHT(I) = 1./SIGMAY(I)**2
41 SUM = SUM + WEIGHT(I)
  YMEAN = YMEAN + WEIGHT(I)*Y(I)
  DO 44 J = 1, NTERMS
44 XMEAN(J) = XMEAN(J) + WEIGHT(I)*FCTN(X,I,J,M)

```

```

50 CONTINUE
51 YMEAN = YMEAN/SUM
   DO 53 J=1, NTERMS
53 XMEAN(J) = XMEAN(J)/SUM
   FNPTS = NPTS
   WMEAN = SUM / FNPTS
   DO 57 I=1, NPTS
57 WEIGHT (I) = WEIGHT (I) /WMEAN

C ACCUMULATE MATRICES R AND ARRAY
61 DO 67 I=1, NPTS
   SIGMA = SIGMA + WEIGHT(I)*(Y(I) - YMEAN)**2
   DO 67 J=1, NTERMS
   SIGMAX(J) = SIGMAX(J) + WEIGHT (I)*(FCTN(X,I,J,M) - XMEAN(J))**2
   R(J) = R(J) + WEIGHT(I)*(FCTN(X,I,J,M)-XMEAN(J))*(Y(I)-YMEAN)
   DO 67 K=1, J
67 ARRAY(J,K) = ARRAY(J,K)+WEIGHT(I)*(FCTN(X,I,J,M)-XMEAN(J))*
1 (FCTN(X,I,K,M)-XMEAN(K))
71 FREE1 = NPTS - 1
72 SIGMA = SQRT(SIGMA/FREE1)
   DO 78 J = 1, NTERMS
74 SIGMAX(J) = SQRT(SIGMAX(J)/FREE1)
   R(J) = R(J)/(FREE1*SIGMAX(J)*SIGMA)
   DO 78 K = 1, J
   ARRAY(J,K) = ARRAY(J,K) / (FREE1*SIGMAX(J)*SIGMAX(K))
78 ARRAY(K,J) = ARRAY(J,K)

C...INVERT SYMMETRIC MATRIX
81 CALL MATIN1(ARRAY,10,NTERMS,MDIM,0,INDEX,NERROR,DET)
   IF (DET) 101, 91, 101
91 AO = 0.
   SIGMAO = 0.
   RMUL = 0.
   CHISQR = 0.
   FTEST = 0.
   GO TO 150

C...CALCULATE COEFFICIENTS, FIT, AND CHI SQUARE
101 AO = YMEAN
102 DO 108 J=1, NTERMS
   DO 104 K=1, NTERMS
104 A(J) = A(J) + R(K) * ARRAY(J,K)
105 A(J) = A(J) * SIGMA/SIGMAX(J)
106 AO = AO - A(J)*XMEAN(J)
107 DO 108 I=1, NPTS
108 YFIT(I) = YFIT(I) + A(J)*FCTN(X,I,J,M)
111 DO 113 I=1, NPTS
   YFIT(I) = YFIT(I) + AO
113 CHISQ = CHISQ + WEIGHT(I)*(Y(I) - YFIT(I))**2
   FREEN = NPTS - NTERMS - 1
115 CHISQR = CHISQ*WMEAN/FREEN

C CALCULATE UNCERTAINTIES
121 IF (MODE) 122, 124, 122
122 VARNCE = 1./WMEAN
   GO TO 131
124 VARNCE = CHISQR
131 DO 133 J=1, NTERMS
132 SIGMAA(J) = ARRAY(J,J) * VARNCE / (FREE1*SIGMAX(J)**2)
133 RMUL = RMUL + A(J) * R(J) * SIGMAX(J)/SIGMA

```

```

FREEJ = NTERMS
135 FTEST = (RMUL/FREEJ) / ((1.-RMUL)/FREEM)
136 RMUL = SQRT (RMUL)
141 SIGMAO = VARNCE / FNPTS
DO 145 J=1, NTERMS
DO 145 K=1, NTERMS
145 SIGMAO = SIGMAO + VARNCE*XMEAN(J)*XMEAN(K)*ARRAY(J,K) /
1 (FREE1*SIGMAX(J)*SIGMAX(K))
146 SIGMAO = SQRT (SIGMAO)
150 RETURN
END

```

```

FUNCTION FCTN(X,I,J,M)
COMMON /DATASET/ XT(4000,7),IMAGE(4000,13)
DIMENSION X(1), M(1)
IF (J .LE. 4) THEN
FCTN=XT(I,J)
ELSE
WRITE(6,10) J
10 FORMAT('O!!!SCREW UP SOMEWHERE!!!',/,
+ 'In FCTN. J =',I3,' Check NTERMS.')
WRITE(2,10) J
STOP
END IF
RETURN
END

```

C*****

SUBROUTINE SUMMARY(AO,A,SIGMAO,SIGMAA,R,RMUL)
C...This subroutine prints out a 3 line summary to the analysis summary file
C according to the following format

C	record 1 contents	data type	1st byte	# bytes
C	date	char*9	1	9
C	time	char*8	10	8
C	calibration type	char*4	18	4
C	image file name 1	char*40	22	40
C	image type 1	char*4	62	4
C	initial element	integer*4	66	4
C	last element	integer*4	70	4
C	initial line	integer*4	74	4
C	last line	integer*4	78	4
C	image file name 2	char*40	82	40
C	image type 2	char*4	122	4
C	initial element	integer*4	126	4
C	last element	integer*4	130	4
C	initial line	integer*4	134	4
C	last line	integer*4	138	4
C	bands used	F8.0	142	8
C	Linf (1-7)	7(F7.2)	150	49
C	dmin	F7.2	199	7
C	dmax	F7.2	206	7
C	(A-A7)	8(F7.2)	213	56
C	(EA-EA7)	8(F7.2)	269	56
C	r's(1-7)	F7.2	325	7
C	rmul	F7.2	332	7
C	calib mean	F7.2	339	7

C	calib rms	"	346	7
C	cal fitted mean	"	353	7
C	ecal fit mean	"	360	7
C	cal fitted sigma	"	367	7
C	ecal fit sigma	"	374	7
C	test mean	"	381	7
C	test rms	"	388	7
C	test fitted mean	"	395	7
C	etest fit mean	"	402	7
C	test fitted sigma	"	409	7
C	etest fit sigma	"	416	7
C	# calib. pts.	"	423	7
C	# test pts.	"	430	7
C	avg. per cent error	"	437	7

```

INCLUDE 'USER$DISK:[BATHY.SOURCE]BATH.INCLUDE'
CHARACTER*9 ADATE
CHARACTER*8 ATIME, BLANK
DATA BLANK/' ' /
DIMENSION A(10),SIGMAA(10),R(10)
REAL*4 KREC(50),LREC(50)

```

```

C...Open the file for appending
OPEN(UNIT=15,FILE='DJA3:[THFAY.DSTRUNS]SUMMARY.DBAS',
+ STATUS='OLD',ACCESS='APPEND')
OPEN(UNIT=16,FILE='DJA3:[THFAY.DSTRUNS]SUMMARY.LIS',
+ STATUS='OLD',ACCESS='APPEND')

```

```

C...get date and time
CALL DATE(ADATE)
CALL TIME(ATIME)

```

```

C...Set "bands used" word
DO N = 0,NTERMS-1
KREC(1) = KREC(1) + IBAND(N+1)*(10**N)
END DO

```

```

C...Store the L infinities
DO N = 1,7
KREC(1+N) = LINF(')
END DO

```

```

C...Store min. and max. depth allowed
KREC(9) = DMIN
KREC(10) = DMAX

```

```

C...Save the fitted constants and their errors
DO N = 1,NTERMS
KREC(11+N) = A(N)
KREC(19+N) = SIGMAA(N)
END DO
KREC(11) = AO
KREC(19) = SIGMAO

```

```

C...Now fill up the third record
C...Store the correlation coefficients
DO N = 1,NTERMS
LREC(N) = R(N)
END DO

```

LREC(8) = RMUL

C...Get residual mean and rms from HBOOK and store

LREC(9) = HSTATI(20,1) !calib. resid. mean
LREC(10) = HSTATI(20,2) !calib. resid. rms
LREC(15) = HSTATI(34,1) !test resid. mean
LREC(16) = HSTATI(34,2) !test resid. rms

C...Store Gaussian params. to calib. residuals

LREC(11) = AVC !fitted mean
LREC(12) = SIGC(2) !std. dev. of mean
LREC(13) = SDC !fitted sigma
LREC(14) = SIGC(3) !std. dev. of sigma

C...Store Gaussian params. to test residuals

LREC(17) = AVT
LREC(18) = SIGT(2)
LREC(19) = SDT
LREC(20) = SIGT(3)

C...Extract number of calibration and test points from histo info

CALL HNOENT(20,L1) !# of entries in histo #20
CALL HNOENT(34,L2) !# of entries in histo #34
LREC(21) = FLOAT(L1)
LREC(22) = FLOAT(L2)

C...Per Cent error in test points

LREC(23) = HSTATI(31,1)

WRITE(15,10) ADATE,ATIME,CALTYPE,IMAGEFILE1,IMAGETYPE1,IET,LET,
+ ILT,LLT,IMAGEFILE2,IMAGETYPE2,IES,LES,
+ ILS,LLS,(KREC(N),N=1,26),(LREC(N),N=1,23)

WRITE(16,25) ADATE,ATIME,CALTYPE
WRITE(16,26) IMAGEFILE1,IMAGETYPE1,IET,LET,ILT,LLT
WRITE(16,26) IMAGEFILE2,IMAGETYPE2,IES,LES,ILS,LLS
WRITE(16,27) (KREC(N),N=1,13)
WRITE(16,28) (KREC(N),N=14,26)
WRITE(16,28) (LREC(N),N=1,13)
WRITE(16,29) (LREC(N),N=14,23)
WRITE(16,*)
WRITE(16,*)

10 FORMAT(1H ,5A,4I4,2A,4I4,F8.0,25F7.2,23F7.2)
25 FORMAT(/,3A)
26 FORMAT(2A,4I4)
27 FORMAT(F8.0,12F7.2)
28 FORMAT(13F7.2)
29 FORMAT(10F7.2)
RETURN
END

C*****

SUBROUTINE FINDIT(FOUND,N)

INCLUDE 'USER\$DISK:[BATHY.SOURCE]BATH.INCLUDE'
LOGICAL FOUND,F1

```

F1 = .TRUE.

DO I = 1,NTERMS
  IF (IMAGE(N,I+3) .GT. LINF(I) .AND. F1) THEN
    F1 = .TRUE.
  ELSE
    F1 = .FALSE.
  END IF
END DO

IF (IMTYPE .EQ. 'T' .OR. IMTYPE .EQ. 't') THEN

  IF ((IMAGE(N,1) .GE. IE .AND. IMAGE(N,1) .LE. LE).AND.
+   (IMAGE(N,2) .GE. IL .AND. IMAGE(N,2) .LE. LL).AND.
+   (IMAGE(N,3) .GT. 0) .AND.
+   (IMAGE(N,8) .LE. 10) .AND. F1) THEN
    FOUND = .TRUE.
  END IF

ELSE IF (IMTYPE .EQ. 'S' .OR. IMTYPE .EQ. 's') THEN

  IF ((IMAGE(N,1) .GE. IE .AND. IMAGE(N,1) .LE. LE).AND.
+   (IMAGE(N,2) .GE. IL .AND. IMAGE(N,2) .LE. LL).AND.
+   (IMAGE(N,3) .GT. 0) .AND. F1) THEN
    FOUND = .TRUE.
  END IF

END IF

RETURN
END

```

C+*****

SUBROUTINE LINE_ELEM

```

INCLUDE 'USERSDISK:[BATHY.SOURCE]BATH.INCLUDE'
CHARACTER*1 RESPONSE

```

```

IF (IMTYPE .EQ. 'T' .OR. IMTYPE .EQ. 't') THEN
  WRITE(6,411) IET,LET,ILT,LLT
  WRITE(6,*)
  WRITE(6,*) 'Do you wish to make changes? (Y/N)'
  ACCEPT 15, RESPONSE
  IF (RESPONSE .EQ. 'Y' .OR. RESPONSE .EQ. 'y') THEN
    WRITE(6,*) 'Enter Initial Elem and Last Elem:'
    ACCEPT *, IET,LET
    WRITE(6,*) 'Enter Initial Line and Last Line:'
    ACCEPT *, ILT,LLT
  ELSE
    WRITE(6,*) 'No changes made.'
  END IF
ELSE IF (IMTYPE .EQ. 'S' .OR. IMTYPE .EQ. 's') THEN
  WRITE(6,412) IES,LES,ILS,LLS
  WRITE(6,*) 'Do you wish to make changes? (Y/N)'
  ACCEPT 15, RESPONSE
  IF (RESPONSE .EQ. 'Y' .OR. RESPONSE .EQ. 'y') THEN
    WRITE(6,*) 'Enter Initial Elem and Last Elem:'
    ACCEPT *, IES,LES
  END IF
END IF

```

```

        WRITE(6,*) 'Enter Initial Line and Last Line:'
        ACCEPT *, ILS,LLS
    ELSE
        WRITE(6,*) 'No changes made.'
    END IF
END IF

15 FORMAT(A)
411 FORMAT('OInitial Element TM =',I5,'      Last Element TM =',I5,/,
+         ' Initial Line TM      =',I5,'      Last Line TM      =',I5,/)
412 FORMAT('OInitial Element SPOT =',I5,'    Last Element SPOT =',I5,/,
+         ' Initial Line SPOT    =',I5,'    Last Line SPOT    =',I5,/)

RETURN
END

```

PROGRAM LINF7

C...This program does a paretas & spero model fit to the data,
 C based on L infinity slices.
 C...It uses combined imagery from TM and SPOT
 C...It quizzes the user for the number of bands to use from each sensor
 C...and asks for the value of the L infinities in each band.
 C...Imagery data are taken from the combined DST files, asking the user for
 C the file name.
 C...A subroutine at the end will print a three line summary of the fit to
 C the output summary file.

C...Include file contains common blocks

```

INCLUDE 'USER$DISK:[BATHY.SOURCE]BATH.INCLUDE'
CALL HLIMIT(20000)
CALL HBOOK2(4,' CALCULATED DEPTH VS. ACTUAL DEPTHS',
+          50,0.,50.,35,0.,35.,16)
CALL HBOOK2(10,' MEAS. DEPTH VS. CALC. DEPTH - MEAS. DEPTHS',
+          40,0.,20.,40,-10.,10.,16)
CALL HBOOK1(15,' PER CENT ERROR, CALIB. PTS.S',
+          50,0.,100.,256)
CALL HBOOK1(20,' RESIDUALS, DEPTH - CALCULATED DEPTHS',
+          60,-15.,15.,256)
CALL HCOPY(15,31,' PER CENT ERROR, TEST PTS.S')
CALL HCOPY(4,32,' TEST DEPTHS; CALC. VS. ACTUALS')
CALL HCOPY(10,33,' TEST DEPTHS; ACT. VS. MEAS. - ACT. S')
CALL HCOPY(20,34,' TEST DEPTH RESIDUALS, ACT. - CALCS')
CALL HBOOK1(101,' NEAREST NEIGHBORSS',100,0.,1000.,2048)
CALL HBOOK1(102,' DEPTH DIFF. TM - SPOTS',60,-30.,30.,2048)
CALL HBOOK1(103,' NEIGHBORS, POINT 3S',100,0.,10000.,2048)
CALL HBLACK(0)
CALL HTITLE(' USA, USM, NORDA.  Satellite BathymetryS')
CALL MAIN
CALL EXIT
END
  
```

SUBROUTINE MAIN

C...Subroutine to do Multiple Linear Regression driving

C...include the common blocks

```

INCLUDE 'USER$DISK:[BATHY.SOURCE]BATH.INCLUDE'
CHARACTER*40 TMSPOTFILE, PSFILE
  
```

C...Array IMAGE contains the following data

```

C      IMAGE(N,1) = column of Nth calibration point-tm
C      ,2) = row of Nth calibration point-tm
C      3 = depth*10 in meters
C      4 = tm band 1 gray level
C      5 = tm band 2 gray level
C      6 = tm band 3 gray level
C      7 = tm band 4 gray level
C      8 = tm band 5 gray level
C      9 = column of Nth calib point-spot
C      10 = row of Nth calib point-spot
C      11 = spot band 1 gray level
C      12 = spot band 2 gray level
C      13 = spot band 3 gray level
  
```

```

        DIMENSION X(2000), Y(2000), SIGMAY(2000), M(10), YFIT(2000),
+           A(10), SIGMAA(10), R(10), RSIG(10)

```

C...Define error mode for subroutine REGRESS

```

        DATA MODE/0/
        DATA DMIN/0./,DMAX/0./

```

C...Open needed files

```

        WRITE(6,3)
        3 FORMAT('Enter name of TMSPOT DST file to use.')
        READ(5,4) TMSPOTFILE
        4 FORMAT(A)
        PRINT 7
        7 FORMAT(' Enter name of output file.')
        ACCEPT 4, PSFILE
        OPEN(UNIT=7,FILE=TMSPOTFILE,STATUS='OLD',READONLY) !tm-spot data file
        OPEN(UNIT=2,FILE=PSFILE,STATUS='NEW')
        PRINT 110, TMSPOTFILE, PSFILE
        WRITE(2,110) TMSPOTFILE,PSFILE
        110 FORMAT(1H1,' TMSPOT Input filename = ',A/,
+           '00Output bathy filename = ',A)

```

C...Go get some needed information from the user

```

        CALL GETINFO

```

C...Get some more needed information from the user

```

        CALL LINE_ELEM

        WRITE(2,20) IET,LET,ILT,LLT
        WRITE(2,21) IES,LES,ILS,LLS

```

C...Go read in calibration data and gray levels from disk

```

        CALL DATAIN(1,NPTS) !1 indicates data to be used in regression

```

C...Gather calibration and corresponding data points into one array

```

        NE = 0
        DO NK = 1,NPTS
            IF((IMAGE(NK,1) .GE. IET .AND. IMAGE(NK,1) .LE. LET).AND.
+           (IMAGE(NK,2) .GE. ILT .AND. IMAGE(NK,2) .LE. LLT).AND.
+           (IMAGE(NK,9) .GE. IES .AND. IMAGE(NK,9) .LE. LES).AND.
+           (IMAGE(NK,10) .GE. ILS .AND. IMAGE(NK,10) .LE. LLS).AND.
+           (IMAGE(NK,4) .GT. LINF(1)).AND.
+           (IMAGE(NK,5) .GT. LINF(2)).AND.
+           (IMAGE(NK,6) .GT. LINF(3)).AND.
+           (IMAGE(NK,7) .GT. LINF(4)).AND.
+           (IMAGE(NK,11) .GT. LINF(5)).AND.
+           (IMAGE(NK,12) .GT. LINF(6)).AND.
+           (IMAGE(NK,13) .GT. LINF(7)).AND.
+           (IMAGE(NK,8) .LE. .10).AND.
+           (IMAGE(NK,3) .GT. 0)) THEN
                NE = NE + 1

```

C...Set up arrays for multiple linear regression

```

        X(NE) = NE
        DO K = 1,NTM
            XT(NE,K) = ALOG(FLOAT(MAX(IMAGE(NK,K+3)-LINF(K),1)))
        END DO
        DO K = 1,NSPOT
            XT(NE,NTM+K)=ALOG(FLOAT(MAX(IMAGE(NK,K+10)-LINF(K+4),1)))
        END DO
        Y(NE) = FLOAT(IMAGE(NK,3))/10.

```

```

        ELSE
          NTHROW = NTHROW + 1
        END IF
      END DO

```

```

      WRITE(6,555) NE,NTHROW
      WRITE(2,555) NE,NTHROW
      IF(NE .LT. NTERMS+2) THEN
        WRITE(6,556)
        WRITE(2,556)
        STOP
      END IF

```

```

C...Go call the mulitple linear regression stuff
      CALL REGRESS(X,Y,SIGMAY,NE,NTERMS,M,O,YFIT,AO,A,SIGMAO,SIGMAA,
+               R,RMUL,CHISQ,FTEST)

```

```

C...Loop over calibration depths. Calculate residuals.

```

```

      DO N = 1, NE
        CALCZ = YFIT(N)
        Z = Y(N)
        PCE = ABS(((CALCZ-Z)/Z)*100.)
        CALL HFILL(15,PCE,0.,1.)
        CALL HFILL(20,Z-CALCZ,0.,1.)
        CALL HFILL(10,Z,Z-CALCZ,1.)
        CALL HFILL(4,CALCZ,Z,1.)
      END DO

```

```

C...End of Job Routine

```

```

C...Write fit info to screen

```

```

      WRITE(6,200)
      WRITE(6,205)
      WRITE(6,210) AO,SIGMAO
      WRITE(6,215) (K,A(K),SIGMAA(K), K=1,NTERMS)
      WRITE(6,218)
      WRITE(6,220) (K,R(K), K=1,NTERMS)
      WRITE(6,225) RMUL
      WRITE(6,230) CHISQR, FTEST

```

```

C...Write fit info to output file

```

```

      WRITE(2,200)
      WRITE(2,205)
      WRITE(2,210) AO,SIGMAO
      WRITE(2,215) (K,A(K),SIGMAA(K), K = 1,NTERMS)
      WRITE(2,218)
      WRITE(2,220) (K,R(K), K = 1,NTERMS)
      WRITE(2,225) RMUL
      WRITE(2,230) CHISQR, FTEST

```

```

C.. Let the user know about what's going on.

```

```

      WRITE(6,240)

```

```

C...Loop to check resids of non calibration points

```

```

      CALL DATAIN(2,NPTS) !get test calib. pts.
      DO N = 1, NPTS
        CD = FLOAT(IMAGE(N,3))/10. !depth in meters

```

```

        IF((IMAGE(N,1) .GE. IET .AND. IMAGE(N,1) .LE. LET).AND.
+        (IMAGE(N,2) .GE. ILT .AND. IMAGE(N,2) .LE. LLT).AND.
+        (IMAGE(N,9) .GE. IES .AND. IMAGE(N,9) .LE. LES).AND.
+        (IMAGE(N,10) .GE. ILS .AND. IMAGE(N,10) .LE. LLS).AND.

```

```

+ (IMAGE(N,4) .GT. LINF(1)).AND.
+ (IMAGE(N,5) .GT. LINF(2)).AND.
+ (IMAGE(N,6) .GT. LINF(3)).AND.
+ (IMAGE(N,7) .GT. LINF(4)).AND.
+ (IMAGE(N,11) .GT. LINF(5)).AND.
+ (IMAGE(N,12) .GT. LINF(6)).AND.
+ (IMAGE(N,13) .GT. LINF(7)).AND.
+ (IMAGE(N,8) .LE .10).AND.
+ (IMAGE(N,3) .GT. 0)) THEN
DO MM = 1, NTM
  RSIG(MM) = FLOAT(MAX(IMAGE(N,MM+3)-LINF(MM),1))
END DO
DO MM = 1, NSPOT
  RSIG(NTM+MM) = FLOAT(MAX(IMAGE(N,MM+10)-LINF(MM+4),1))
END DO
ZT = AO
DO MM = 1, NTERMS
  ZT = ZT + A(MM)*ALOG(RSIG(MM))
END DO
PCE = ABS(((ZT-CD)/CD)*100.)
CALL HFILL(31,PCE,0.,1.)
CALL HFILL(32,ZT,CD,1.)
CALL HFILL(33,CD,CD-ZT,1.)
CALL HFILL(34,CD-ZT,0.,1.)
END IF
END DO

```

C...Let user know what is happening
WRITE(6,260)

C...fit Gaussian distribution to residuals and then print the histograms
CALL HFITGA(20,C3,AVC,SDC,CHI2C,12,SIGC) !calibration points
CALL HFITGA(34,C3,AVT,SDT,CHI2T,12,SIGT) !test points
CALL HISTD0

C...Go print out summary information on fit
WRITE(6,265)
CALL SUMMARY(AO,A,SIGMAO,SIGMAA,R,RMUL)

C...Lets get out of here. Tell user we're done.
WRITE(6,270)
RETURN

C...FORMAT statements

```

5 FORMAT(1X,I10,' Calibration points read in.',/,
+ 1X,I10,' Calibration points outside of image.')
10 FORMAT(31X,F7.3,2X,F6.1,2X,F6.1)
20 FORMAT(' IET ',I4,' LET ',I4,' ILT ',I4,' LLT ',I4)
21 FORMAT(' IES ',I4,' LES ',I4,' ILS ',I4,' LLS ',I4)
90 FORMAT(1X,I3,2X,I3,2X,F4.1)
100 FORMAT(I3,2X,I3,F5.1)
200 FORMAT(///'0 ---- RESULTS OF MULTIPLE LINEAR'
+ ' REGRESSION ----'//)
205 FORMAT('OFitted Parameter Values')
210 FORMAT(' A0 = 'F8.3,' +/- ',F8.4)
215 FORMAT(' A',I1,' = ',F8.3,' +/- ',F8.4)
218 FORMAT('OLinear Correlation Coefficients')
220 FORMAT(' R',I1,' = ',F8.3)
225 FORMAT(' Multiple Correlation Coefficient, RM = ',F8.3)
230 FORMAT('OCHISQ = ',F8.3,' FTEST = ',F10.3)

```

```

240 FORMAT('ONow gathering statistics using test points...')
250 FORMAT('ODuplicate calibration point....NR, NC, OLD DEPTH, NEW',
+       2I5,2F8.1)
260 FORMAT('ONow Playing: Histograms and Scatter Plots!')
265 FORMAT('OPrinting out summary.')
270 FORMAT('OJob completed. I'm outta here.')
555 FORMAT(1H0,I10,' Calibration points to be used in regression.',
+       /,1H0,I10,' Calibration points out of range.')
556 FORMAT(1H0,' INSUFFICIENT DATA FOR REGRESSION! STOPPING!!!')
2000 FORMAT(A1)
      END

```

C*****

```

      SUBROUTINE GETINFO
      INCLUDE 'USER$DISK:[BATHY.SOURCE]BATH.INCLUDE'
C...Read the header records
      READ(7,5) IMAGETYPE,CALTYPE,IMAGEFILE1,IMAGEFILE2
      READ(7,6) IET,LET,ILT,LLT,IES,LES,ILS,LLS
      READ(7,7) INFO
      5 FORMAT(4A)
      6 FORMAT(8(3X,I4))
      7 FORMAT(A130)

      WRITE(6,10)
10  FORMAT('                                SATELLITE BATHYMETRY!')
      WRITE(6,30)
30  FORMAT('OEnter number of TM bands to use in fit.')
      READ(5,*) NTM
      PRINT 40
40  FORMAT('OEnter number of SPOT bands to use in fit.')
      READ(5,*) NSPOT
      NTERMS = NTM + NSPOT
      WRITE(6,50) NTM
50  FORMAT('OEnter the TM LINF''s (band 1 to ',I1,')')
      READ(5,*) (LINF(N),N=1,NTM)
      WRITE(6,60) NSPOT
60  FORMAT('OEnter the SPOT LINF''s (band 1 to ',I1,')')
      READ(5,*) (LINF(N),N=NTM+1,NTERMS)
C...Go write out info to output file
      CALL INFOUT

      RETURN
      END

```

C*****

```

      SUBROUTINE INFOUT
      INCLUDE 'USER$DISK:[BATHY.SOURCE]BATH.INCLUDE'

      WRITE(2,10)
10  FORMAT('                                SATELLITE BATHYMETRY')
      WRITE(2,12) IMAGETYPE, CALTYPE, IMAGEFILE1,IMAGEFILE2
12  FORMAT('OImagery from the ',A,' sensor. Calibration from ',A,/,
+       ' Image name is ',A,A)
      WRITE(2,14) INFO
14  FORMAT('OComments entered on this image are:',/,1H ,A)

```

```

WRITE(2,*)
WRITE(2,*) ' Regression based on Linfs slices.'
WRITE(2,*)

WRITE(2,20) NTERMS, DMIN, DMAX
20 FORMAT('OUsing',I3,' bands of imagery',/,
+       ' Minimum calibration depth is',F4.0,/,
+       ' Maximum calibration depth is',F4.0)
WRITE(2,25)
25 FORMAT('OThe L infinities are...')
DO N = 1,NTERMS
  WRITE(2,30) N, LINF(N)
30  FORMAT(' LINF(',I1,') = ',I3)
END DO
RETURN
END

```

C*****

```

SUBROUTINE DATAIN(LCALL,NP)
INCLUDE 'USER$DISK:[BATHY.SOURCE]BATH.INCLUDE'
DIMENSION INTENSET(5),INTENSES(3)
CHARACTER*80 JUNK
NP = 0

```

C...Rewind file and skip header records

```

REWIND(7)
READ(7,5) JUNK
READ(7,5) JUNK
READ(7,5) JUNK
5 FORMAT(A)

```

C...Read the first TMSPOT data record

```

READ(7,10) LAT,LON,NEAST,NORTH,ID,NCT,NRT,
+       (INTENSET(N),N=1,5),NCS,NRS,
+       (INTENSES(N),N=1,3)
10 FORMAT(5I8,12I4)
DO WHILE (NEAST .NE. 0)
  IF (LCALL .EQ. 1) THEN
    IF (INTENSET(1) .GT. 0 .AND. INTENSES(1) .GT. 0) THEN
      NP = NP + 1
      IMAGE(NP,1) = NCT
      IMAGE(NP,2) = NRT
      IMAGE(NP,3) = ID
      IMAGE(NP,4) = INTENSET(1)
      IMAGE(NP,5) = INTENSET(2)
      IMAGE(NP,6) = INTENSET(3)
      IMAGE(NP,7) = INTENSET(4)
      IMAGE(NP,8) = INTENSET(5)
      IMAGE(NP,9) = NCS
      IMAGE(NP,10) = NRS
      IMAGE(NP,11) = INTENSES(1)
      IMAGE(NP,12) = INTENSES(2)
      IMAGE(NP,13) = INTENSES(3)
    END IF
  END IF

```

C...Skip a record to use as a test point

```

READ(7,10) LAT,LON,NEAST,NORTH,ID,NCT,NRT,
+       (INTENSET(N),N=1,5),NCS,NRS,

```

```

+          (INTENSES(N),N=1,3)
      READ(7,10) LAT,LON,NEAST,NORTH,ID,NCT,NRT,
+          (INTENSET(N),N=1,5),NCS,NRS,
+          (INTENSES(N),N=1,3)

      ELSE IF (LCALL .EQ. 2) THEN
C...First record used as a calib. point; skip it.

      READ(7,10) LAT,LON,NEAST,NORTH,ID,NCT,NRT,
+          (INTENSET(N),N=1,5),NCS,NRS,
+          (INTENSES(N),N=1,3)
      IF (INTENSET(1) .GT. 0 .AND. INTENSES(1) .GT. 0) THEN
        NP = NP + 1
        IMAGE(NP,1) = NCT
        IMAGE(NP,2) = NRT
        IMAGE(NP,3) = ID
        IMAGE(NP,4) = INTENSET(1)
        IMAGE(NP,5) = INTENSET(2)
        IMAGE(NP,6) = INTENSET(3)
        IMAGE(NP,7) = INTENSET(4)
        IMAGE(NP,8) = INTENSET(5)
        IMAGE(NP,9) = NCS
        IMAGE(NP,10) = NRS
        IMAGE(NP,11) = INTENSES(1)
        IMAGE(NP,12) = INTENSES(2)
        IMAGE(NP,13) = INTENSES(3)
      END IF
C...Skip next next record as it was used as a calib. point, too.
      READ(7,10) LAT,LON,NEAST,NORTH,ID,NCT,NRT,
+          (INTENSET(N),N=1,5),NCS,NRS,
+          (INTENSES(N),N=1,3)

      END IF
      END DO
      RETURN
      END

C*****

      SUBROUTINE REGRESS(X,Y,SIGMAY,NPTS,NTERMS,M,MODE,YFIT,AO,A,
+          SIGMAO,SIGMAA,R,RMUL,CHISQR,FTEST)
      COMMON /DATASET/ XT(3000,7),IMAGE(3000,13)
      DIMENSION X(2000),Y(2000),SIGMAY(2000),M(10),YFIT(2000),A(10),
+          SIGMAA(10),R(10)
      DIMENSION WEIGHT(2000),XMEAN(10),SIGMAX(10),ARRAY(10,10)
      DIMENSION INDEX(10) !scratch space for matrix inversion routine

C...INITIALIZE SUMS AND ARRAYS
11 SUM = 0.
   YMEAN = 0.
   SIGMA = 0.
   CHISQ = 0.
   RMUL = 0.
   DO 17 I = 1, NPTS
17 YFIT(I) = 0.
21 DO 28 J = 1, NTERMS
   XMEAN(J) = 0.
   SIGMAX(J) = 0.
   DO 28 K=1, NTERMS

```

28 ARRAY(J,K) = 0.

C...ACCUMULATE WEIGHTS

```
30 DO 50 I=1,NPTS
31 IF (MODE) 32,37,39
32 IF (Y(I)) 35, 37, 33
33 WEIGHT(I) = 1./(-Y(I))
   GO TO 41
35 WEIGHT(I) = 1./ (-Y(I))
   GO TO 41
37 WEIGHT(I) = 1.
   GO TO 41
39 WEIGHT(I) = 1./SIGMAY(I)**2
41 SUM = SUM + WEIGHT(I)
   YMEAN = YMEAN + WEIGHT(I)*Y(I)
   DO 44 J = 1, NTERMS
44 XMEAN(J) = XMEAN(J) + WEIGHT(I)*FCTN(X,I,J,M)
50 CONTINUE
51 YMEAN = YMEAN/SUM
   DO 53 J=1,NTERMS
53 XMEAN(J) = XMEAN(J)/SUM
   FNPTS = NPTS
   WMEAN = SUM / FNPTS
   DO 57 I=1, NPTS
57 WEIGHT (I) = WEIGHT (I) /WMEAN
```

C ACCUMULATE MATRICES R AND ARRAY

```
61 DO 67 I=1, NPTS
   SIGMA = SIGMA + WEIGHT(I)*(Y(I) - YMEAN)**2
   DO 67 J=1, NTERMS
```

C...DEBUG...DEBUG...DEBUG

FCV = FCTN(X,I,J,M)

```
   SIGMAX(J) = SIGMAX(J) + WEIGHT (I)*(FCTN(X,I,J,M) - XMEAN(J))**2
   R(J) = R(J) + WEIGHT(I)*(FCTN(X,I,J,M)-XMEAN(J))*(Y(I)-YMEAN)
   DO 67 K=1, J
67 ARRAY(J,K) = ARRAY(J,K)+WEIGHT(I)*(FCTN(X,I,J,M)-XMEAN(J))*
   1 (FCTN(X,I,K,M)-XMEAN(K))
71 FREE1 = NPTS - 1
72 SIGMA = SQRT(SIGMA/FREE1)
   DO 78 J = 1,NTERMS
74 SIGMAX(J) = SQRT(SIGMAX(J)/FREE1)
   R(J) = R(J)/(FREE1*SIGMAX(J)*SIGMA)
   DO 78 K = 1,J
   ARRAY(J,K) = ARRAY(J,K) / (FREE1*SIGMAX(J)*SIGMAX(K))
78 ARRAY(K,J) = ARRAY(J,K)
```

C...INVERT SYMMETRIC MATRIX

```
81 CALL MATIN1(ARRAY,10,NTERMS,MDIM,0,INDEX,NERROR,DET)
   IF (DET) 101, 91, 101
91 AO = 0.
   SIGMAO = 0.
   RMUL = 0.
   CHISQR = 0.
   FTEST = 0.
   GO TO 150
```

C...CALCULATE COEFFICIENTS, FIT, AND CHI SQUARE

```

101 AO = YMEAN
102 DO 108 J=1, NTERMS
    DO 104 K=1, NTERMS
104 A(J) = A(J) + R(K) * ARRAY(J,K)
105 A(J) = A(J) * SIGMA/SIGMAX(J)
106 AO = AO - A(J)*XMEAN(J)
107 DO 108 I=1, NPTS
108 YFIT(I) = YFIT(I) + A(J)*FCTN(X,I,J,M)
111 DO 113 I=1, NPTS
    YFIT(I) = YFIT(I) + AO
113 CHISQ = CHISQ + WEIGHT(I)*(Y(I) - YFIT(I))**2
    FREEN = NPTS - NTERMS - 1
115 CHISQR = CHISQ*WMEAN/FREEN

```

C CALCULATE UNCERTAINTIES

```

121 IF (MODE) 122, 124, 122
122 VARNCE = 1./WMEAN
    GO TO 131
124 VARNCE = CHISQR
131 DO 133 J=1, NTERMS
132 SIGMAA(J) = ARRAY(J,J) * VARNCE / (FREE1*SIGMAX(J)**2)
133 RMUL = RMUL + A(J) * R(J) * SIGMAX(J)/SIGMA
    FREEJ = NTERMS
135 FTEST = (RMUL/FREEJ) / ((1.-RMUL)/FREEN)
136 RMUL = SQRT (RMUL)
141 SIGMAO = VARNCE / FNPTS
    DO 145 J=1, NTERMS
    DO 145 K=1, NTERMS
145 SIGMAO = SIGMAO + VARNCE*XMEAN(J)*XMEAN(K)*ARRAY(J,K) /
    1 (FREE1*SIGMAX(J)*SIGMAX(K))
146 SIGMAO = SQRT (SIGMAO)
150 RETURN
    END

```

```

FUNCTION FCTN(X,I,J,M)
COMMON /DATASET/ XT(3000,7),IMAGE(3000,13)
DIMENSION X(1), M(1)
IF (J .LE. 7) THEN
    FCTN = XT(I,J)
ELSE
    WRITE(6,10) J
10  FORMAT('O!!!SCREW UP SOMEWHERE!!!',/,
+      'In FCTN. J =',I3,' Check NTERMS.')
    WRITE(2,10) J
    STOP
END IF
RETURN
END

```

C*****~*****

SUBROUTINE SUMMARY(AO,A,SIGMAO,SIGMAA,R,RMUL)

C...This subroutine prints out a 3 line summary to the analysis summary file
C according to the following format

C	record 1 contents	data type	1st byte	# bytes
C	date	char*9	1	9
C	time	char*8	10	8

C	calibration type	char*4	18	4
C	image file name 1	char*40	22	40
C	image type 1	char*4	62	4
C	initial element	integer*4	66	4
C	last element	integer*4	70	4
C	initial line	integer*4	74	4
C	last line	integer*4	78	4
C	image file name 2	char*40	82	40
C	image type 2	char*4	122	4
C	initial element	integer*4	126	4
C	last element	integer*4	130	4
C	initial line	integer*4	134	4
C	last line	integer*4	138	4
C	bands used	F8.0	142	8
C	Linf (1-7)	7(F7.2)	150	49
C	dmin	F7.2	199	7
C	dmax	F7.2	206	7
C	(A-A7)	8(F7.2)	213	56
C	(EA-EA7)	8(F7.2)	269	56
C	r's(1-7)	F7.2	325	7
C	rmul	F7.2	332	7
C	calib mean	F7.2	339	7
C	calib rms	"	346	7
C	cal fitted mean	"	353	7
C	ecal fit mean	"	360	7
C	cal fitted sigma	"	367	7
C	ecal fit sigma	"	374	7
C	test mean	"	381	7
C	test rms	"	388	7
C	test fitted mean	"	395	7
C	etest fit mean	"	402	7
C	test fitted sigma	"	409	7
C	etest fit sigma	"	416	7
C	# calib. pts.	"	423	7
C	# test pts.	"	430	7
C	avg. per cent error	"	437	7

```

INCLUDE 'USER$DISK:[BATHY.SOURCE]BATH.INCLUDE'
CHARACTER*9 ADATE
CHARACTER*8 ATIME, BLANK
DATA BLANK/' '/
DIMENSION A(10),SIGMAA(10),R(10)
REAL*4 KREC(50),LREC(50)

```

```

C...Open the file for appending
OPEN(UNIT=15,FILE='USER$DISK:[THFAY.TERRI.EXEC]SUMMARY.DBAS',
+ STATUS='OLD',ACCESS='APPEND')
OPEN(UNIT=16,FILE='USER$DISK:[THFAY.TERRI.EXEC]SUMMARY.LIS',
+ STATUS='OLD',ACCESS='APPEND')

```

```

C...get date and time
CALL DATE(ADATE)
CALL TIME(ATIME)

```

```

C...Set "bands used" word, first for TM then for SPOT.
DO N = 0,NTM-1
  KREC(1) = KREC(1) + (N+1)*(10**N)
END DO

DO N = 0,NSPOT-1

```

```

        KREC(1) = KREC(1) + (N+1)*(10**(N+4))
    END DO

C...Store the L infinities
    DO N = 1,7
        KREC(1+N) = LINF(N)
    END DO

C...Store min. and max. depth allowed
    KREC(9) = DMIN
    KREC(10) = DMAX

C...Save the fitted constants and their errors
    DO N = 1,NTERMS
        KREC(11+N) = A(N)
        KREC(19+N) = SIGMAA(N)
    END DO
    KREC(11) = A0
    KREC(19) = SIGMA0

C...Now fill up the third record
C...Store the correlation coefficients
    DO N = 1,NTERMS
        LREC(N) = R(N)
    END DO
    LREC(8) = RMUL

C...Get residual mean and rms from HBOOK and store
    LREC(9) = HSTATI(20,1) !calib. resid. mean
    LREC(10) = HSTATI(20,2) !calib. resid. rms
    LREC(15) = HSTATI(34,1) !test resid. mean
    LREC(16) = HSTATI(34,2) !test resid. rms

C...Store Gaussian params. to calib. residuals
    LREC(11) = AVC !fitted mean
    LREC(12) = SIGC(2) !std. dev. of mean
    LREC(13) = SDC !fitted sigma
    LREC(14) = SIGC(3) !std. dev. of sigma

C...Store Gaussian params. to test residuals
    LREC(17) = AVT
    LREC(18) = SIGT(2)
    LREC(19) = SDT
    LREC(20) = SIGT(3)

C...Extract number of calibration and test points from histo info
    CALL HNOENT(20,L1) !# of entries in histo #20
    CALL HNOENT(34,L2) !# of entries in histo #34
    LREC(21) = FLOAT(L1)
    LREC(22) = FLOAT(L2)

C...Per Cent error in test points
    LREC(23) = HSTATI(31,1)

    WRITE(15,10) ACATE,ATIME,CALTYPE,IMAGEFILE1,IMAGETYPE1,IET,LET,
+               ILT,LLT,IMAGEFILE2,IMAGETYPE2,IES,LES,
+               ILS,LLS,(KREC(N),N=1,26),(LREC(N),N=1,23)

```

```

WRITE(16,25) ADATE,ATIME,CALTYPE
WRITE(16,26) IMAGEFILE1,IMAGETYPE1,IET,LET,ILT,LLT
WRITE(16,26) IMAGEFILE2,IMAGETYPE2,IES,LES,ILS,LLS
WRITE(16,27) (KREC(N),N=1,13)
WRITE(16,28) (KREC(N),N=14,26)
WRITE(16,28) (LREC(N),N=1,13)
WRITE(16,29) (LREC(N),N=14,23)
WRITE(16,*)
WRITE(16,*)

```

```

10 FORMAT(1H ,5A,4I4,2A,4I4,F8.0,25F7.2,23F7.2)
25 FORMAT(/,3A)
26 FORMAT(2A,4I4)
27 FORMAT(F8.0,12F7.2)
28 FORMAT(13F7.2)
29 FORMAT(10F7.2)
RETURN
END

```

C*****

SUBROUTINE LINE_ELEM

```

INCLUDE 'USER$DISK:[BATHY.SOURCE]BATH.INCLUDE'
CHARACTER*1 RESPONSE

```

```

WRITE(6,411) IET,LET,ILT,LLT
WRITE(6,*)
WRITE(6,*) 'Do you wish to make changes? (Y/N)'
ACCEPT 15, RESPONSE
IF (RESPONSE .EQ. 'Y' .OR. RESPONSE .EQ. 'y') THEN
    WRITE(6,*) 'Enter Initial Elem and Last Elem:'
    ACCEPT *, IET,LET
    WRITE(6,*) 'Enter Initial Line and Last Line:'
    ACCEPT *, ILT,LLT
ELSE
    WRITE(6,*) 'No TM line/element changes made.'
END IF
WRITE(6,412) IES,LES,ILS,LLS
WRITE(6,*) 'Do you wish to make changes? (Y/N)'
ACCEPT 15, RESPONSE
IF (RESPONSE .EQ. 'Y' .OR. RESPONSE .EQ. 'y') THEN
    WRITE(6,*) 'Enter Initial Elem and Last Elem:'
    ACCEPT *, IES,LES
    WRITE(6,*) 'Enter Initial Line and Last Line:'
    ACCEPT *, ILS,LLS
ELSE
    WRITE(6,*) 'No SPOT line/element changes made.'
END IF

```

```

15 FORMAT(A)
411 FORMAT('OInitial Element TM =',I5,' Last Element TM =',I5,/,
+ ' Initial Line TM =',I5,' Last Line TM =',I5,/)
412 FORMAT('OInitial Element SPOT =',I5,' Last Element SPOT =',I5,/,
+ ' Initial Line SPOT =',I5,' Last Line SPOT =',I5,/)

```

```

RETURN
END

```

PROGRAM MINMAX4

C...This program does a paretos & spero model fit to the data
 C...It quizzes the user for the number of bands to use, which bands to use,
 C...the value of DMIN and the value of DMAX to use. Cuts on the data are made
 C...according to the values of DMIN and DMAX. It also asks for the value of
 C...the L infinities in each band.
 C...It reads in imagery data from the new DST file, asking the user for
 C...the file name.
 C...A three line summary of the results are printed to the output
 C...summary file.

```

INCLUDE 'USER$DISK:[BATHY.SOURCE]BATH.INCLUDE'
CALL HLIMIT(20000)
CALL HBOOK2(4,' ACTUAL DEPTH VS. CALCULATED DEPTH$',
+           50,0.,50.,35,0.,35.,16)
CALL HBOOK2(10,' ACT.-CALC. DEPTH VS. ACT. DEPTH$',
+           40,0.,20.,40,-10.,10.,16)
CALL HBOOK2(15,' PER CENT ERROR, CALIB. PTS.$',
+           50,0.,100.,256)
CALL HBOOK2(20,' RESIDUALS, ACT. DEPTH - CALC. DEPTH$',
+           60,-15.,15.,256)
CALL HBOOK2(31,' PER CENT ERROR, TEST PTS.$')
CALL HBOOK2(32,' TEST DEPTHS; ACT. DEPTH VS. CALC. DEPTH$')
CALL HBOOK2(33,' TEST DEPTHS; ACT.-CALC. VS. ACT. $')
CALL HBOOK2(34,' TEST DEPTH RESIDUALS, ACT. - CALC.$')
CALL HBLACK(0)
CALL HTITLE(' USA, USM, NORDA.  Satellite Bathymetry$')
CALL MAIN
CALL EXIT
END
  
```

SUBROUTINE MAIN

C...Subroutine to do Multiple Linear Regression driving

CHARACTER*40 DSTFILE, PSFILE

C.."Kount" counts the number of points with calculated depth 0.
 integer kount(7)
 data kount/7*0/

```

INCLUDE 'USER$DISK:[BATHY.SOURCE]BATH.INCLUDE'
  
```

C...Array IMAGE contains the following data

```

C      IMAGE(N,1) = column of Nth calibration point
C      ,2) = row of Nth calibration point
C      3 = depth*10 in meters
C      4 = 1st band gray level
C      5 = 2nd band gray level
C      6 = 3rd band gray level
C      7 = 4th band gray level
C      8 = 5th band gray level (for land/water cut.)
C      DIMENSION X(4000), Y(4000), SIGMAY(4000), M(10), YFIT(4000),
+      A(10), SIGMAA(10), R(10)
  
```

C...Define error mode for subroutine REGRESS.
 DATA MODE/0/

C...Open needed files.

```

WRITE(6,3)
3 FORMAT('Enter name of DST file to use.')
READ(5,4) DSTFILE
4 FORMAT(A)
PRINT 7
7 FORMAT(' Enter name of output file.')
ACCEPT 4, PSFILE
OPEN(UNIT=7,FILE=DSTFILE,STATUS='OLD',READONLY) !data file
OPEN(UNIT=2,FILE=PSFILE,STATUS='NEW') !hardcopy output file
PRINT 110, DSTFILE, PSFILE
WRITE(2,110) DSTFILE, PSFILE
110 FORMAT(1H1,' Input filename = ',A/,
+ ' Output bathy filename = ',A)

```

C...Go get some needed information from the user
CALL GETINFO

C...Go get some needed information from the user
CALL LINE_ELEM

```

IF (IMTYPE .EQ. 'T' .OR. IMTYPE .EQ. 't') THEN
  IE = IET
  LE = LET
  IL = ILT
  LL = LLT
ELSE IF (IMTYPE .EQ. 'S' .OR. IMTYPE .EQ. 's') THEN
  IE = IES
  LE = LES
  IL = ILS
  LL = LLS
END IF

WRITE(2,20) IE,LE
WRITE(2,21) IL,LL

```

C...Go read in calibration data and gray levels from dst

```

IF (IMTYPE .EQ. 'T'.OR. IMTYPE .EQ. 't') THEN
  CALL DATATM(1,NPTS) !1 indicates data to be used in regression
ELSE IF (IMTYPE .EQ. 'S'.OP. IMTYPE .EQ. 's') THEN
  CALL DATASPOT(1,NPTS)
ELSE
  WRITE(6,120) IMTYPE
END IF

```

C...Gather calibration and corresponding data points into one array

```

NE = 0
DO NK = 1,NPTS
  IF ((IMAGE(NK,1) .GE. IE .AND. IMAGE(NK,1) .LE. LE).AND.
+ (IMAGE(NK,2) .GE. IL .AND. IMAGE(NK,2) .LE. LL).AND.
+ (IMAGE(NK,3) .GT. DMIN*10).AND.
+ (IMAGE(NK,3) .LT. DMAX*10).AND.
+ (IMAGE(NK,8) .LE. 10)) THEN
    NE = NE + 1

```

C...Set up arrays for multiple linear regression

```

X(NE) = NE
DO NN = 1,NTERMS
  XT(NE,NN) =
+   ALOG(FLOAT(MAX(IMAGE(NK,NN+3)-LINF(NN),1)))
+   if (image(nk,nn+3)-linf(nn) .le. 1) then

```

```

            kount(nn) = kount(nn) + 1
        endif
    END DO
    Y(NE) = FLOAT(IMAGE(NK,3))/10.
ELSE
    NTHROW = NTHROW + 1
END IF
END DO

C...ALOG stuff:
do i=1,nterms
    write(6,*) 'band =',iband(i),' "ALOG MAX=1" count =',kount(i)
enddo

WRITE(6,555) NE,NTHROW
WRITE(2,555) NE,NTHROW
IF(NE .LT. NTERMS+2) THEN
    WRITE(6,556)
    WRITE(2,556)
    STOP
END IF

C...Go call the mulitple linear regression stuff
CALL REGRESS(X,Y,SIGMAY,NE,NTERMS,M,0,YFIT,AO,A,SIGMAO,SIGMAA,
+           R,RMUL,CHISQR,FTEST)

C...Loop over calibration depths.  Calculate residuals.
DO N = 1, NE
    CALCZ = YFIT(N)
    Z = Y(N)
    PCE = ABS(((CALCZ-Z)/Z)*100.)
    CALL HFILL(15,PCE,0.,1.)
    CALL HFILL(20,Z-CALCZ,0.,1.)  !Act. - Calc.
    CALL HFILL(10,Z,Z-CALCZ,1.)  !Act. - Calc. vs Act.
    CALL HFILL(4,CALCZ,Z,1.)      !Act. vs Calc.
END DO

C...End of Job Routine
C...Write fit info to screen
WRITE(6,200)
WRITE(6,205)
WRITE(6,210) AO,SIGMAO
WRITE(6,215) (IBAND(K),A(K),SIGMAA(K), K=1,NTERMS)
WRITE(6,218)
WRITE(6,220) (IBAND(K),R(K), K=1,NTERMS)
WRITE(6,225) RMUL
WRITE(6,230) CHISQR, FTEST
C...Write fit info to output file
WRITE(2,200)
WRITE(2,205)
WRITE(2,210) AO,SIGMAO
WRITE(2,215) (IBAND(K),A(K),SIGMAA(K), K = 1,NTERMS)
WRITE(2,218)
WRITE(2,220) (IBAND(K),R(K), K = 1,NTERMS)
WRITE(2,225) RMUL
WRITE(2,230) CHISQR, FTEST

C...Let the user know about what's going on.
WRITE(6,240)

```

C...Loop to check resids of non calibration points

C...Loop to check resids of non calibration points. First get test data.

```
      IF (IMTYPE .EQ. 'T'.OR. IMTYPE .EQ. 't') THEN
        CALL DATATM(2,NPTS)  !2 indicates data to be used in test
      ELSE IF (IMTYPE .EQ. 'S' .OR. IMTYPE .EQ. 's') THEN
        CALL DATASPOT(2,NPTS)
      ELSE
        WRITE(6,120) IMTYPE
      END IF
      DO N = 1, NPTS
        CD = FLOAT(IMAGE(N,3))/10.  !depth in meters

        IF ((IMAGE(N,1) .GE. IE .AND. IMAGE(N,1) .LE. LE).AND.
+         (IMAGE(N,2) .GE. IL .AND. IMAGE(N,2) .LE. LL).AND.
+         (CD.GT.DMIN) .AND. (CD.LT.DMAX) .AND. IMAGE(N,8).LE.10) THEN
          ZT = A0
          DO MM = 1, NTERMS
            ZT = ZT +
+            A(MM)*ALOG(FLOAT(MAX(IMAGE(N,MM+3)-LINF(MM),1)))
          END DO
          PCE = ABS(((ZT-CD)/CD)*100.)
          CALL HFILL(31,PCE,0.,1.)
          CALL HFILL(32,ZT,CD,1.)      !Actual depth vs Calculated depth
          CALL HFILL(33,CD,CD-ZT,1.)  !Act. - Calc. vs Act.
          CALL HFILL(34,CD-ZT,0.,1.)  !Act. - Calc.
        END IF
      END DO
```

C...Let user know what is happening
 WRITE(6,260)

C...fit Gaussian distribution to residuals and then print the histograms
 CALL HFITGA(20,C3,AVC,SDC,CHI2C,12,SIGC) !calibration points
 CALL HFITGA(34,C3,AVT,SDT,CHI2T,12,SIGT) !test points
 CALL HISTDO

C...Go print out summary information on fit
 WRITE(6,265)
 CALL SUMMARY(A0,A,SIGMA0,SIGMAA,R,RMUL)

C...Lets get out of here. Tell user we're done.
 WRITE(6,270)
 RETURN

C...FORMAT statements

```
      5 FORMAT(1X,I10,' Calibration points read in.',/,
+      1X,I10,' Calibration points outside of image.')
     10 FORMAT(31X,F7.3,2X,F6.1,2X,F6.1)
     20 FORMAT(' IE ',I4,' LE ',I4)
     21 FORMAT(' IL ',I4,' LL ',I4)
     90 FORMAT(1X,I3,2X,I3,2X,F4.1)
    100 FORMAT(I3,2X,I3,F5.1)
    120 FORMAT('OScrewly IMAGETYPE: ',A)
    200 FORMAT(////'0      ---- RESULTS OF MULTIPLE LINEAR'
+      ' REGRESSION ----'//)
    205 FORMAT('OFitted Parameter Values')
    210 FORMAT(' A0 = 'F8.3,' +/- ',F8.4)
    215 FORMAT(' A',I1,' = ',F8.3,' +/- ',F3.4)
```

```

218 FORMAT('OLinear Correlation Coefficients')
220 FORMAT(' R',I1,' = ',F8.3)
225 FORMAT(' Multiple Correlation Coefficient, RM = ',F8.3)
230 FORMAT('OCHISQ = ',F8.3,' FTEST = ',F10.3)
240 FORMAT('ONow gathering statistics using test points...')
250 FORMAT('ODuplicate calibration point....NR, NC, OLD DEPTH, NEW',
+       2I5,2F8.1)
260 FORMAT('ONow Playing: Histograms and Scatter Plots!')
265 FORMAT('OPrinting out summary.')
270 FORMAT('OJob completed. I'm outta here.')
555 FORMAT(1H0,I10,' Calibration points to be used in regression.',
+       /,1H0,I10,' Calibration points out of range.')
556 FORMAT(1H0,' INSUFFICIENT DATA FOR REGRESSION! STOPPING!!')
2000 FORMAT(A1)
END

```

C*****

SUBROUTINE GETINFO

INCLUDE 'USER\$DISK:[BATHY.SOURCE]BATH.INCLUDE'

C...Read the header records from DST file.

```

READ(7,5) IMAGETYPE,CALTYPE,IMAGEFILE1,IMAGEFILE2
READ(7,6) IET,LET,ILT,LLT,IES,LES,ILS,LLS
READ(7,7) INFO
5 FORMAT(4A)
6 FORMAT(8(3X,I4))
7 FORMAT(A130)

WRITE(6,10)
10 FORMAT(' SATELLITE BATHYMETRY!')
WRITE(6,15)
15 FORMAT(' Enter''T'' for TM imagery',/,
+       ' Enter''S'' for SPOT imagery')
ACCEPT 16,IMTYPE
16 FORMAT(A)
WRITE(6,20)
20 FORMAT('OEnter min and max depths to get from calibration file.')
READ(5,*) DMIN, DMAX
WRITE(6,30)
30 FORMAT('OEnter number of bands to use in fit.')
READ(5,*) NTERMS
WRITE(6,35)
35 FORMAT('OEnter band(s) to use in fit:')
READ(5,*) (IBAND(N),N=1,NTERMS)
WRITE(6,40) NTERMS
40 FORMAT('OEnter the LINF''s for the ',I1,' band(s):')
READ(5,*) (LINF(N),N=1,NTERMS)

```

C...Go write out info to output file
CALL INFOUT

RETURN
END

C*****

SUBROUTINE INFOUT

```

INCLUDE 'USER$DISK:[BATHY.SOURCE]BATH.INCLUDE'
WRITE(2,10)
10 FORMAT('                SATELLITE BATHYMETRY')
WRITE(2,12) IMAGETYPE, ALTYPE, IMAGEFILE1, IMAGEFILE2
12 FORMAT('0Imagery from the ',A,' sensor.  Calibration from ',A/,
+       ' Image names are ',A,A)
WRITE(2,14) INFO
14 FORMAT('0Comments entered on this DST file are:',/,1H ,A)
WRITE(2,20) NTERMS, (IBAND(N),N=1,NTERMS), DMIN, DMAX
20 FORMAT('0Using',I3,' bands of imagery',/,
+       ' Band(s) ',<NTERMS>(1X,I2),/,
+       ' Minimum calibration depth is',F4.0,/,
+       ' Maximum calibration depth is',F4.0)
WRITE(2,25)
25 FORMAT('0The L infinities are...')
DO N = 1,NTERMS
    WRITE(2,30) IBAND(N), LINF(N)
30  FORMAT(' LINF(',I1,') = ',I3)
END DO
RETURN
END

```

C*****

SUBROUTINE DATATM(LCALL,NP)

C...This subroutine reads in TM imagery data from the dst file and
C...stores it in array IMAGE.

```

INCLUDE 'USER$DISK:[BATHY.SOURCE]BATH.INCLUDE'
DIMENSION INTENSE(5)
NP = 0

```

C...Rewind file and skip header records of DST file.

```

REWIND(7)
READ(7,*)
READ(7,*)
READ(7,*)

```

C...Read the first data record

```

READ(7,10) LAT,LON,NEAST,NORTH,ID,NCT,NRT,
+       (INTENSE(N),N=1,5)
10 FORMAT(5I8,7I4)

```

```

DO WHILE (NEAST .NE. 0)
    IF (LCALL .EQ. 1) THEN
        IF (INTENSE(1).GT.0) THEN
            NP = NP + 1
            IMAGE(NP,1) = NCT
            IMAGE(NP,2) = NRT
            IMAGE(NP,3) = ID
            DO J=1,NTERMS
                IMAGE(NP,J+3) = INTENSE(IBAND(J))
            ENDDO
            IMAGE(NP,8) = INTENSE(5) !Band 5 is used as a land/water cut.
        END IF
    END IF

```

C...Skip a record to use later as a test point

```

      READ(7,10) LAT,LON,NEAST,NORTH,ID,NCT,NRT,
+      (INTENSE(N),N=1,5)
      READ(7,10) LAT,LON,NEAST,NORTH,ID,NCT,NRT,
+      (INTENSE(N),N=1,5)
      ELSE IF (LCALL .EQ. 2) THEN
C...First record was used as a calib. point; skip it.
      READ(7,10) LAT,LON,NEAST,NORTH,ID,NCT,NRT,
+      (INTENSE(N),N=1,5)
      IF (INTENSE(1).GT.0) THEN
        NP = NP + 1
        IMAGE(NP,1) = NCT
        IMAGE(NP,2) = NRT
        IMAGE(NP,3) = ID
        DO J=1,NTERMS
          IMAGE(NP,J+3) = INTENSE(IBAND(J))
        ENDDO
        IMAGE(NP,8) = INTENSE(5) !Band 5 is used as a land/water cut.
      END IF
C...Skip next record as it was used as a calib. point, too.
      READ(7,10) LAT,LON,NEAST,NORTH,ID,NCT,NRT,
+      (INTENSE(N),N=1,5)
      END IF
    END DO

    RETURN
  END

```

```

      SUBROUTINE DATASPOT(LCALL,NP)
C...This subroutine reads in SPOT imagery data from the dst file and
C...stores it in array IMAGE.

```

```

      INCLUDE 'USER$DISK:[BATHY.SOURCE]BATH.INCLUDE'
      DIMENSION INTENS(3)
      NP = 0

```

```

C...Rewind file and skip header records of DST file.
      REWIND(7)
      READ(7,*)
      READ(7,*)
      READ(7,*)

```

```

C...Read the first data record.
      READ(7,10) LAT,LON,NEAST,NORTH,ID,
+      NCS,NRS,(INTENS(N),N=1,3)
10 FORMAT(5I8,28X,5I4)
      DO WHILE (NEAST .NE. 0)
        IF (LCALL .EQ. 1) THEN
          IF (INTENS(1) .GT. 0 ) THEN
            NP = NP + 1
            IMAGE(NP,1) = NCS
            IMAGE(NP,2) = NRS
            IMAGE(NP,3) = ID
            DO J=1,NTERMS
              IMAGE(NP,J+3) = INTENS(IBAND(J))
            ENDDO
            IMAGE(NP,7) = 0
            IMAGE(NP,8) = 0
          END IF

```

```

C...Skip a record to use as a test point
      READ(7,10) LAT,LON,NEAST,NORTH,ID,
+       NCS,NRS,(INTENSS(N),N=1,3)
      READ(7,10) LAT,LON,NEAST,NORTH,ID,
+       NCS,NRS,(INTENSS(N),N=1,3)
      ELSE IF (LCALL .EQ. 2) THEN
C...First record used as a calib. point; skip it.
      READ(7,10) LAT,LON,NEAST,NORTH,ID,
+       NCS,NRS,(INTENSS(N),N=1,3)
      IF (INTENSS(1) .GT. 0 ) THEN
        NP = NP + 1
        IMAGE(NP,1) = NCT
        IMAGE(NP,2) = NRT
        IMAGE(NP,3) = ID
        DO J=1,NTERMS
          IMAGE(NP,J+3) = INTENSS(IBAND(J))
        ENDDO
        IMAGE(NP,7) = 0
        IMAGE(NP,8) = 0
      END IF
C...Skip next record as it was used as a calib. point, too.
      READ(7,10) LAT,LON,NEAST,NORTH,ID,
+       NCS,NRS,(INTENSS(N),N=1,3)
      END IF
      END DO

      RETURN
      END

```

C*****

```

      SUBROUTINE REGRESS(X,Y,SIGMAY,NPTS,NTERMS,M,MODE,YFIT,AO,A,
+       SIGMAO,SIGMAA,R,RMUL,CHISQR,FTEST)
      COMMON /DATASET/ XT(4000,7),IMAGE(4000,13)
      DIMENSION X(4000),Y(4000),SIGMAY(4000),M(10),YFIT(4000),A(10),
+       SIGMAA(10),R(10)
      DIMENSION WEIGHT(4000), XMEAN(10), SIGMAX(10), ARRAY(10,10)
      DIMENSION INDEX(10) !scratch space for matrix inversion routine

```

C...INITIALIZE SUMS AND ARRAYS

```

11 SUM = 0.
   YMEAN = 0.
   SIGMA = 0.
   CHISQ = 0.
   RMUL = 0.
   DO 17 I = 1, NPTS
17 YFIT(I) = 0.
21 DO 28 J = 1, NTERMS
   XMEAN(J) = 0.
   SIGMAX(J) = 0.
   DO 28 K=1, NTERMS
28 ARRAY(J,K) = 0.

```

C...ACCUMULATE WEIGHTS

```

30 DO 50 I=1,NPTS
31 IF (MODE) 32,37,39
32 IF (Y(I)) 35, 37, 33
33 WEIGHT(I) = 1./(-Y(I))
   GO TO 41

```

```

35 WEIGHT(I) = 1./ (-Y(I))
   GO TO 41
37 WEIGHT(I) = 1.
   GO TO 41
39 WEIGHT(I) = 1./SIGMAY(I)**2
41 SUM = SUM + WEIGHT(I)
   YMEAN = YMEAN + WEIGHT(I)*Y(I)
   DO 44 J = 1, NTERMS
44 XMEAN(J) = XMEAN(J) + WEIGHT(I)*FCTN(X,I,J,M)
50 CONTINUE
51 YMEAN = YMEAN/SUM
   DO 53 J=1,NTERMS
53 XMEAN(J) = XMEAN(J)/SUM
   FNPTS = NPTS
   WMEAN = SUM / FNPTS
   DO 57 I=1, NPTS
57 WEIGHT (I) = WEIGHT (I) /WMEAN

```

C ACCUMULATE MATRICES R AND ARRAY

```

61 DO 67 I=1, NPTS
   SIGMA = SIGMA + WEIGHT(I)*(Y(I) - YMEAN)**2
   DO 67 J=1, NTERMS
   SIGMAX(J) = SIGMAX(J) + WEIGHT (I)*(FCTN(X,I,J,M) - XMEAN(J))**2
   R(J) = R(J) + WEIGHT(I)*(FCTN(X,I,J,M)-XMEAN(J))*(Y(I)-YMEAN)
   DO 67 K=1, J
67 ARRAY(J,K) = ARRAY(J,K)+WEIGHT(I)*(FCTN(X,I,J,M)-XMEAN(J))*
   1 (FCTN(X,I,K,M)-XMEAN(K))
71 FREE1 = NPTS - 1
72 SIGMA = SQRT(SIGMA/FREE1)
   DO 78 J = 1,NTERMS
74 SIGMAX(J) = SQRT(SIGMAX(J)/FREE1)
   R(J) = R(J)/(FREE1*SIGMAX(J)*SIGMA)
   DO 78 K = 1,J
   ARRAY(J,K) = ARRAY(J,K) / (FREE1*SIGMAX(J)*SIGMAX(K))
78 ARRAY(K,J) = ARRAY(J,K)

```

C...INVERT SYMMETRIC MATRIX

```

81 CALL MATIN1(ARRAY,10,NTERMS,MDIM,0,INDEX,NERROR,DET)
   IF (DET) 101, 91, 101
91 AO = 0.
   SIGMAO =0.
   RMUL = 0.
   CHISQR = 0.
   FTEST = 0.
   GO TO 150

```

C...CALCULATE COEFFICIENTS, FIT, AND CHI SQUARE

```

101 AO = YMEAN
102 DO 108 J=1, NTERMS
   DO 104 K=1, NTERMS
104 A(J) = A(J) + R(K) * ARRAY(J,K)
105 A(J) = A(J) * SIGMA/SIGMAX(J)
106 AO = AO - A(J)*XMEAN(J)
107 DO 108 I=1, NPTS
108 YFIT(I) = YFIT(I) + A(J)*FCTN(X,I,J,M)
111 DO 113 I=1, NPTS
   YFIT(I) = YFIT(I) + AO
113 CHISQ = CHISQ + WEIGHT(I)*(Y(I) - YFIT(I))**2
   FREEN = NPTS - NTERMS - 1
115 CHISQR = CHISQ*WMEAN/FREEN

```

```

C      CALCULATE UNCERTAINTIES
121 IF (MODE) 122, 124, 122
122 VARNCE = 1./WMEAN
    GO TO 131
124 VARNCE = CHISQR
131 DO 133 J=1, NTERMS
132 SIGMAA(J) = ARRAY(J,J) * VARNCE / (FREE1*SIGMAX(J)**2)
133 RMUL = RMUL + A(J) * R(J) * SIGMAX(J)/SIGMA
    FREEJ = NTERMS
135 FTEST = (RMUL/FREEJ) / ((1.-RMUL)/FREEN)
136 RMUL = SQRT (RMUL)
141 SIGMA0 = VARNCE / FNPTS
    DO 145 J=1, NTERMS
    DO 145 K=1, NTERMS
145 SIGMA0 = SIGMA0 + VARNCE*XMEAN(J)*XMEAN(K)*ARRAY(J,K) /
    1 (FREE1*SIGMAX(J)*SIGMAX(K))
146 SIGMA0 = SQRT (SIGMA0)
150 RETURN
    END

```

```

FUNCTION FCTN(X,I,J,M)
COMMON /DATASET/ XT(4000,7),IMAGE(4000,13)
DIMENSION X(1), M(1)
IF (J .LE. 4) THEN
    FCTN = XT(I,J)
ELSE
    WRITE(6,10) J
10  FORMAT('0!!!SCREW UP SOMEWHERE!!!',/,
+      'In FCTN. J =',I3,' Check NTERMS.')
    WRITE(2,10) J
    STOP
END IF
RETURN
END

```

C*****

SUBROUTINE SUMMARY(A0,A,SIGMA0,SIGMAA,R,RMUL)

C...This subroutine prints out a 3 line summary to the analysis summary file
C according to the following format

C	record 1 contents	data type	1st byte	# bytes
C	date	char*9	1	9
C	time	char*8	10	8
C	calibration type	char*4	18	4
C	image file name 1	char*40	22	40
C	image type 1	char*4	62	4
C	initial element	integer*4	66	4
C	last element	integer*4	70	4
C	initial line	integer*4	74	4
C	last line	integer*4	78	4
C	image file name 2	char*40	82	40
C	image type 2	char*4	122	4
C	initial element	integer*4	126	4
C	last element	integer*4	130	4
C	initial line	integer*4	134	4
C	last line	integer*4	138	4
C	bands used	F8.0	142	8

C	Linf (1-7)	7(F7.2)	150	49
C	dmin	F7.2	199	7
C	dmax	F7.2	206	7
C	(A-A7)	8(F7.2)	213	56
C	(EA-EA7)	8(F7.2)	269	56
C	r's(1-7)	F7.2	325	7
C	rmul	F7.2	332	7
C	calib mean	F7.2	339	7
C	calib rms	"	346	7
C	cal fitted mean	"	353	7
C	ecal fit mean	"	360	7
C	cal fitted sigma	"	367	7
C	ecal fit sigma	"	374	7
C	test mean	"	381	7
C	test rms	"	388	7
C	test fitted mean	"	395	7
C	etest fit mean	"	402	7
C	test fitted sigma	"	409	7
C	etest fit sigma	"	416	7
C	# calib. pts.	"	423	7
C	# test pts.	"	430	7
C	avg. per cent error	"	437	7

```

INCLUDE 'USER$DISK:[BATHY.SOURCE]BATH.INCLUDE'
CHARACTER*9 ADATE
CHARACTER*8 ATIME, BLANK
DATA BLANK/' '/
DIMENSION A(10),SIGMAA(10),R(10)
REAL*4 KREC(50),LREC(50)

```

```

C...Open the file for appending
OPEN(UNIT=15,FILE='DJA3:[THFAY.DSTRUNS]SUMMARY.DBAS',
+ STATUS='OLD',ACCESS='APPEND')
OPEN(UNIT=16,FILE='DJA3:[THFAY.DSTRUNS]SUMMARY.LIS',
+ STATUS='OLD',ACCESS='APPEND')

```

```

C...get date and time
CALL DATE(ADATE)
CALL TIME(ATIME)

```

```

C...Set bits in "bands used" word
DO N = 0,NTERMS-1
  KREC(1) = KREC(1) + IBAND(N+1)*(10**N)
END DO

```

```

C...Store the L infinities
DO N = 1,7
  KREC(1+N) = LINP(N)
END DO

```

```

C...Store min. and max. depth allowed
KREC(9) = DMIN
KREC(10) = DMAX

```

```

C...Save the fitted constants and their errors
DO N = 1,NTERMS
  KREC(11+N) = A(N)
  KREC(19+N) = SIGMAA(N)
END DO
KREC(11) = A0

```

KREC(19) = SIGMAO

C...Now fill up the thiid record

C...Store the correlation coefficients

```
DO N = 1, NTERMS
  LREC(N) = R(N)
END DO
LREC(8) = RMUL
```

C...Get residual mean and rms from HBOOK and store

```
LREC(9) = HSTATI(20,1) !calib. resid. mean
LREC(10) = HSTATI(20,2) !calib. resid. rms
LREC(15) = HSTATI(34,1) !test resid. mean
LREC(16) = HSTATI(34,2) !test resid. rms
```

C...Store Gaussian params. to calib. residuals

```
LREC(11) = AVC !fitted mean
LREC(12) = SIGC(2) !std. dev. of mean
LREC(13) = SDC !fitted sigma
LREC(14) = SIGC(3) !std. dev. of sigma
```

C...Store Gaussian params. to test residuals

```
LREC(17) = AVT
LREC(18) = SIGT(2)
LREC(19) = SDT
LREC(20) = SIGT(3)
```

C...Extract number of calibration and test points from histo info

```
CALL HNOENT(20,L1) !# of entries in histo #20
CALL HNOENT(34,L2) !# of entries in histo #34
LREC(21) = FLOAT(L1)
LREC(22) = FLOAT(L2)
```

C...Per Cent error in test points

```
LREC(23) = HSTATI(31,1)
```

```
WRITE(15,10) ADATE, ATIME, CALTYPE, IMAGEFILE1, IMAGETYPE1, IET, LET,
+            ILT, LLT, IMAGEFILE2, IMAGETYPE2, IES, LES,
+            ILS, LLS, (KREC(N), N=1, 26), (LREC(N), N=1, 23)
```

```
WRITE(16,25) ADATE, ATIME, CALTYPE
WRITE(16,26) IMAGEFILE1, IMAGETYPE1, IET, LET, ILT, LLT
WRITE(16,26) IMAGEFILE2, IMAGETYPE2, IES, LES, ILS, LLS
WRITE(16,27) (KREC(N), N=1, 13)
WRITE(16,28) (KREC(N), N=14, 26)
WRITE(16,28) (LREC(N), N=1, 13)
WRITE(16,29) (LREC(N), N=14, 23)
WRITE(16,*)
WRITE(16,*)
```

```
10 FORMAT(1H ,5A,4I4,2A,4I4,F8.0,25F7.2,23F7.2)
25 FORMAT(/,3A)
26 FORMAT(2A,4I4)
27 FORMAT(F8.0,12F7.2)
28 FORMAT(13F7.2)
29 FORMAT(10F7.2)
RETJRN
END
```

C*****

SUBROUTINE LINE_ELEM

INCLUDE 'USERSDISK:[BATHY.SOURCE]BATH.INCLUDE'
CHARACTER*1 RESPONSE

IF (IMTYPE .EQ. 'T' .OR. IMTYPE .EQ. 't') THEN
WRITE(6,411) IET,LET,ILT,LLT
WRITE(6,*)
WRITE(6,*) 'Do you wish to make changes? (Y/N)'
ACCEPT 15, RESPONSE
IF (RESPONSE .EQ. 'Y' .OR. RESPONSE .EQ. 'y') then
WRITE(6,*) 'Enter Initial Elem and Last Elem:'
ACCEPT *,IET,LET
WRITE(6,*) 'Enter Initial Line and Last Line:'
ACCEPT *,ILT,LLT

ELSE
WRITE(6,*) 'No changes made.'

END IF

ELSE IF (IMTYPE .EQ. 'S' .OR. IMTYPE .EQ. 's') THEN
WRITE(6,412) IES,LES,ILS,LLS
WRITE(6,*) 'Do you wish to make changes? (Y/N)'
ACCEPT 15, RESPONSE
IF (RESPONSE .EQ. 'Y' .OR. RESPONSE .EQ. 'y') THEN
WRITE(6,*) 'Enter Initial Elem and Last Elem:'
ACCEPT *,IES,LES
WRITE(6,*) 'Enter Initial Line and Last Line:'
ACCEPT *,ILS,LLS

ELSE
WRITE(6,*) 'No changes made.'

END IF

END IF

15 FORMAT(A)
411 FORMAT('OInitial Element TM =',I5,' Last Element TM =',I5,/,
+ ' Initial Line TM =',I5,' Last Line TM =',I5,/)
412 FORMAT('OInitial Element SPOT =',I5,' Last Element SPOT =',I5,/,
+ ' Initial Line SPOT =',I5,' Last Line SPOT =',I5,/)

RETURN
END

PROGRAM MINMAX7

C...This program does a paretos & spero model fit to the data
C...It uses combined imagery from TM and SPOT
C...It quizzes the user for the number of bands to use from each sensor,
C the value of DMIN and the value of DMAX to use. It also asks for
C the value of the L infinities in each band.
C...Imagery data is taken from the DST file, asking the user for
C the file name.
C...A subroutine at the end will print a three line summary of the fit to
C the output summary file.

C...Include file contains common blocks
INCLUDE 'USER\$DISK:[BATHY.SOURCE]BATH.INCLUDE'
CALL HLIMIT(20000)
CALL HBOOK2(4,' CALCULATED DEPTH VS. ACTUAL DEPTHS',
+ 50,0.,50.,35,0.,35.,16)
CALL HBOOK2(10,' MEAS. DEPTH VS. CALC. DEPTH - MEAS. DEPTHS',
+ 40,0.,20.,40,-10.,10.,16)
CALL HBOOK1(15,' PER CENT ERROR, CALIB. PTS.\$',
+ 50,0.,100.,256)
CALL HBOOK1(20,' RESIDUALS, DEPTH - CALCULATED DEPTHS',
+ 60,-15.,15.,256)
CALL HCOPY(15,31,' PER CENT ERROR, TEST PTS.\$')
CALL HCOPY(4,32,' TEST DEPTHS; CALC. VS. ACTUALS\$')
CALL HCOPY(10,33,' TEST DEPTHS; ACT. VS. MEAS. - ACT. \$')
CALL HCOPY(20,34,' TEST DEPTH RESIDUALS, ACT. - CALC\$')
CALL HBOOK1(101,' NEAREST NEIGHBORSS\$,100,0.,1000.,2048)
CALL HBOOK1(102,' DEPTH DIFF. TM - SPOTS\$,60,-30.,30.,2048)
CALL HBOOK1(103,' NEIGHBORS, POINT 3\$,100,0.,10000.,2048)
CALL HBLACK(0)
CALL HTITLE(' USA, USM, NORDA. Satellite Bathymetry\$')
CALL MAIN
CALL EXIT
END

SUBROUTINE MAIN

C...Subroutine to do Multiple Linear Regression driving

C...include the common blocks

INCLUDE 'USER\$DISK:[BATHY.SOURCE]BATH.INCLUDE'
CHARACTER*40 TMSPOTFILE, PSFILE
C...Array IMAGE contains the following data
C IMAGE(N,1) = column of Nth calibration point-tm
C ,2) = row of Nth calibration point-tm
C 3 = depth*10 in meters
C 4 = tm band 1 gray level
C 5 = tm band 2 gray level
C 6 = tm band 3 gray level
C 7 = tm band 4 gray level
C 8 = tm band 5 gray level
C 9 = column of Nth calib. point-spot
C 10 = row of Nth calib. point-spot
C 11 = spot band 1 gray level
C 12 = spot band 2 gray level
C 13 = spot band 3 gray level

```

        DIMENSION X(2000), Y(2000), SIGMAY(2000), M(10), YFIT(2000),
+           A(10), SIGMAA(10), R(10), RSIG(10)

C...Define error mode for subroutine REGRESS
DATA MODE/0/

C...Open needed files
WRITE(6,3)
3 FORMAT('0Enter name of TMSPOT DST file to use.')
ACCEPT 4, TMSPOTFILE
4 FORMAT(A)
PRINT 7
7 FORMAT(' Enter name of output file.')
ACCEPT 4, PSFILE
OPEN(UNIT=7,FILE=TMSPOTFILE,STATUS='OLD',READONLY) !tmspot data file
OPEN(UNIT=2,FILE=PSFILE,STATUS='NEW') !hardcopy output file
PRINT 110, TMSPOTFILE, PSFILE
WRITE(2,110) TMSPOTFILE, PSFILE
110 FORMAT(1H1,' TMSPOT Input filename = ',A,/,
+         '0Output bathy filename = ',A)

C...Go get some needed information from the user
CALL GETINFO

C...Get some more needed information from the user
CALL LINE_ELEM

WRITE(2,20) IET,LET,ILT,LLT
WRITE(2,21) IES,LES,ILS,LLS

C...Go read in calibration data and gray levels from disk
CALL DATAIN(1,NPTS) !1 indicates data to be used in regression

C...Gather calibration and corresponding data points into one array
NE = 0
DO NK = 1,NPTS
  IF((IMAGE(NK,1) .GE. IET .AND. IMAGE(NK,1) .LE. LET).AND.
+   (IMAGE(NK,2) .GE. ILT .AND. IMAGE(NK,2) .LE. LLT).AND.
+   (IMAGE(NK,9) .GE. IES .AND. IMAGE(NK,9) .LE. LES).AND.
+   (IMAGE(NK,10) .GE. ILS .AND. IMAGE(NK,10) .LE. LLS).AND.
+   (IMAGE(NK,3) .GT. DMIN*10).AND.
+   (IMAGE(NK,3) .LT. DMAX*10).AND.
+   (IMAGE(NK,8) .LE. 10)) THEN
    NE = NE + 1
C...Set up arrays for multiple linear regression
X(NE) = NE
DO K = 1,NTM
  XT(NE,K) = ALOG(FLOAT(MAX(IMAGE(NK,K+3)-LINF(K),1)))
END DO
DO K = 1,NSPOT
  XT(NE,NTM+K) = ALOG(FLOAT(MAX(IMAGE(NK,K+10)-LINF(K+4),1)))
END DO
Y(NE) = FLOAT(IMAGE(NK,3))/10.
ELSE
  NTHROW = NTHROW + 1
END IF
END DO

WRITE(6,555) NE,NTHROW

```

```

WRITE(2,555) NE,NTHROW
IF(NE .LT. NTERMS+2) THEN
  WRITE(6,556)
  WRITE(2,556)
  STOP
END IF

```

```

C...Go call the multiple linear regression stuff
CALL REGRESS(X,Y,SIGMAY,NE,NTERMS,M,O,YFIT,AO,A,SIGMAO,SIGMAA,
+           R,RMUL,CHISQ,FTEST)

```

```

C...Loop over calibration depths. Calculate residuals.
DO N = 1, NE
  CALCZ = YFIT(N)
  Z = Y(N)
  PCE = ABS(((CALCZ-Z)/Z)*100.)
  CALL HFILL(15,PCE,0.,1.)
  CALL HFILL(20,Z-CALCZ,0.,1.)
  CALL HFILL(10,Z,Z-CALCZ,1.)
  CALL HFILL(4,CALCZ,Z,1.)
END DO

```

```

C...End of Job Routine
C...Write fit info to screen
  WRITE(6,200)
  WRITE(6,205)
  WRITE(6,210) AO,SIGMAO
  WRITE(6,215) (K,A(K),SIGMAA(K), K=1,NTERMS)
  WRITE(6,218)
  WRITE(6,220) (K,R(K), K=1,NTERMS)
  WRITE(6,225) RMUL
  WRITE(6,230) CHISQR, FTEST

```

```

C...Write fit info to output file
  WRITE(2,200)
  WRITE(2,205)
  WRITE(2,210) AO,SIGMAO
  WRITE(2,215) (K,A(K),SIGMAA(K), K = 1,NTERMS)
  WRITE(2,218)
  WRITE(2,220) (K,R(K), K = 1,NTERMS)
  WRITE(2,225) RMUL
  WRITE(2,230) CHISQR, FTEST

```

```

C.. Let the user know about what's going on.
  WRITE(6,240)

```

```

C...Loop to check resids of non calibration points
CALL DATAIN(2,NPTS) !get test calib. pts.
DO N = 1, NPTS
  CD = FLOAT(IMAGE(N,3))/10. !depth in meters

  IF((IMAGE(N,1) .GE. IET .AND. IMAGE(N,1) .LE. LET).AND.
+   (IMAGE(N,2) .GE. ILT .AND. IMAGE(N,2) .LE. LLT).AND.
+   (IMAGE(N,9) .GE. IES .AND. IMAGE(N,9) .LE. LES).AND.
+   (IMAGE(N,10) .GE. ILS .AND. IMAGE(N,10) .LE. LLS).AND.
+   (CD.GT.DMIN) .AND. (CD.LT.DMAX) .AND. IMAGE(N,8).LE.10) THEN
    DO MM = 1, NTM
      RSIG(MM) = FLOAT(MAX(IMAGE(N,MM+3)-LINF(MM),1))
    END DO
    DO MM = 1, NSPOT
      RSIG(NTM+MM) = FLOAT(MAX(IMAGE(N,MM+10)-LINF(MM+4),1))
    END DO
  END IF
END DO

```

```

      END DO
      ZT = AO
      DO MM = 1, NTERMS
        ZT = ZT + A(MM)*ALOG(RSIG(MM))
      END DO
      PCE = ABS(((ZT-CD)/CD)*100.)
      CALL HFILL(31,PCE,0.,1.)
      CALL HFILL(32,ZT,CD,1.)
      CALL HFILL(33,CD,CD-ZT,1.)
      CALL HFILL(34,CD-ZT,0.,1.)
    END IF
  END DO

```

C...Let user know what is happening
 WRITE(6,260)

C...fit Gaussian distribution to residuals and then print the histograms
 CALL HFITGA(20,C3,AVC,SDC,CHI2C,12,SIGC) !calibration points
 CALL HFITGA(34,C3,AVT,SDT,CHI2T,12,SIGT) !test points
 CALL HISTDO

C...Go print out summary information on fit
 WRITE(6,265)
 CALL SUMMARY(AO,A,SIGMAO,SIGMAA,R,RMUL)

C...Lets get out of here. Tell user we're done.
 WRITE(6,270)
 RETURN

C...FORMAT statements

```

5  FORMAT(1X,I10,' Calibration points read in.',/,
+        1X,I10,' Calibration points outside of image.')
10 FORMAT(31X,F7.3,2X,F6.1,2X,F6.1)
20 FORMAT(' IET ',I4,' LET ',I4,' ILT ',I4,' LLT ',I4)
21 FORMAT(' IES ',I4,' LES ',I4,' ILS ',I4,' LLS ',I4)
90 FORMAT(1X,I3,2X,I3,2X,F4.1)
100 FORMAT(I3,2X,I3,F5.1)
200 FORMAT(///'0      ---- RESULTS OF MULTIPLE LINEAR'
+        ' REGRESSION ----'//)
205 FORMAT('OFitted Parameter Values')
210 FORMAT(' AO = ',F8.3,' +/- ',F8.4)
215 FORMAT(' A ',I1,' = ',F8.3,' +/- ',F8.4)
218 FORMAT('OLinear Correlation Coefficients')
220 FORMAT(' R ',I1,' = ',F8.3)
225 FORMAT(' Multiple Correlation Coefficient, RM = ',F8.3)
230 FORMAT('OCHISQ = ',F8.3,'          FTEST = ',F10.3)
240 FORMAT('ONow gathering statistics using test points...')
250 FORMAT('ODuplicate calibration point....NR, NC, OLD DEPTH, NEW',
+        2I5,2F8.1)
260 FORMAT('ONow Playing: Histograms and Scatter Plots!')
265 FORMAT('OPrinting out summary.')
270 FORMAT('OJob completed. I'm outta here.')
555 FORMAT(1H0,I10,' Calibration points to be used in regression.',
+        /,1H0,I10,' Calibration points out of range.')
556  FORMAT(1H0,' INSUFFICIENT DATA FOR REGRESSION! STOPPING!!')
2000 FORMAT(A1)
      END

```

C*****

```

SUBROUTINE GETINFO
  INCLUDE 'USER$DISK:[BATHY.SOURCE]BATH.INCLUDE'
C...Read the header record
  READ(7,5) IMAGETYPE,CALTYPE,IMAGEFILE1,IMAGEFILE2
  READ(7,6) IET,LET,ILT,LLT,IES,LES,ILS,LLS
  READ(7,7) INFO
  5 FORMAT(4A)
  6 FORMAT(8(3X,I4))
  7 FORMAT(A130)

  WRITE(6,10)
10 FORMAT('                                SATELLITE BATHYMETRY!')
  WRITE(6,20)
20 FORMAT('OEnter min and max depths to get from calibration file.')
  READ(5,*) DMIN, DMAX
  WRITE(6,30)
30 FORMAT('OEnter number of TM bands to use in fit.')
  READ(5,*) NTM
  PRINT 40
40 FORMAT('OEnter number of SPOT bands to use in fit.')
  READ(5,*) NSPOT
  NTERMS = NTM + NSPOT
  WRITE(6,50) NTM
50 FORMAT('OEnter the TM LINF''s (band 1 to ',I1,')')
  READ(5,*) (LINF(N),N=1,NTM)
  WRITE(6,60) NSPOT
60 FORMAT('OEnter the SPOT LINF''s (band 1 to ',I1,')')
  READ(5,*) (LINF(N),N=NTM+1,NTERMS)
C...Go write out info to output file
  CALL INFOUT

  RETURN
END

```

C*****

```

SUBROUTINE INFOUT
  INCLUDE 'USER$DISK:[BATHY.SOURCE]BATH.INCLUDE'

  WRITE(2,10)
10 FORMAT('                                SATELLITE BATHYMETRY')
  WRITE(2,12) IMAGETYPE, CALTYPE, IMAGEFILE1,IMAGEFILE2
12 FORMAT('OImagery from the ',A,' sensor.  Calibration from ',A,/,
+        ' Image names are ',A,A)
  WRITE(2,14) INFO
14 FORMAT('OComments entered on this image are:',/,1H ,A)
  WRITE(2,20) NTERMS, DMIN, DMAX
20 FORMAT('OUsing',I3,' bands of imagery',/,
+        ' Minimum calibration depth is',F4.0,/,
+        ' Maximum calibration depth is',F4.0)
  WRITE(2,25)
25 FORMAT('OThe L infinities are...')
  DO N = 1,NTERMS
    WRITE(2,30) N, LINF(N)
30  FORMAT(' LINF(',I1,') = ',I3)
  END DO
  RETURN
END

```

C*****

```
SUBROUTINE DATAIN(LCALL,NP)
INCLUDE 'USER$DISK:[BATHY.SOURCE]BATH.INCLUDE'
```

```
DIMENSION INTENSET(5),INTENSES(3)
CHARACTER*80 JUNK
NP = 0
```

C...Rewind file and skip header record

```
REWIND(7)
READ(7,5) JUNK
READ(7,5) JUNK
READ(7,5) JUNK
5 FORMAT(A)
```

C...Read the first TMSPOT data record

```
READ(7,10) LAT,LON,NEAST,NORTH,ID,NCTM,NRTM,
+          (INTENSET(N),N=1,5),NCSPOT,NRSPOT,
+          (INTENSES(N),N=1,3)
10 FORMAT(5I8,12I4)
```

```
DO WHILE (NEAST .NE. 0)
  IF (LCALL .EQ. 1) THEN
    IF (INTENSET(1) .GT. 0 .AND. INTENSES(1) .GT. 0) THEN
      NP = NP + 1
      IMAGE(NP,1) = NCTM
      IMAGE(NP,2) = NRTM
      IMAGE(NP,3) = ID
      IMAGE(NP,4) = INTENSET(1)
      IMAGE(NP,5) = INTENSET(2)
      IMAGE(NP,6) = INTENSET(3)
      IMAGE(NP,7) = INTENSET(4)
      IMAGE(NP,8) = INTENSET(5)
      IMAGE(NP,9) = NCSPOT
      IMAGE(NP,10) = NRSPOT
      IMAGE(NP,11) = INTENSES(1)
      IMAGE(NP,12) = INTENSES(2)
      IMAGE(NP,13) = INTENSES(3)
    END IF
  END IF
```

C...Skip a record to use as a test point

```
READ(7,10) LAT,LON,NEAST,NORTH,ID,NCTM,NRTM,
+          (INTENSET(N),N=1,5),NCSPOT,NRSPOT,
+          (INTENSES(N),N=1,3)
READ(7,10) LAT,LON,NEAST,NORTH,ID,NCTM,NRTM,
+          (INTENSET(N),N=1,5),NCSPOT,NRSPOT,
+          (INTENSES(N),N=1,3)
```

ELSE IF (LCALL .EQ. 2) THEN

C...First record used as a calib. point; skip it.

```
READ(7,10) LAT,LON,NEAST,NORTH,ID,NCTM,NRTM,
+          (INTENSET(N),N=1,5),NCSPOT,NRSPOT,
+          (INTENSES(N),N=1,3)
IF (INTENSET(1) .GT. 0 .AND. INTENSES(1) .GT. 0) THEN
  NP = NP + 1
  IMAGE(NP,1) = NCTM
  IMAGE(NP,2) = NRTM
  IMAGE(NP,3) = ID
  IMAGE(NP,4) = INTENSET(1)
```

```

        IMAGE(NP,5) = INTENSET(2)
        IMAGE(NP,6) = INTENSET(3)
        IMAGE(NP,7) = INTENSET(4)
        IMAGE(NP,8) = INTENSET(5)
        IMAGE(NP,9) = NCSPOT
        IMAGE(NP,10) = NRSPOT
        IMAGE(NP,11) = INTENSES(1)
        IMAGE(NP,12) = INTENSES(2)
        IMAGE(NP,13) = INTENSES(3)
    END IF
C...Skip next record as it was      as a calib. point, too.
    READ(7,10) LAT,LON,NEAST,      4,ID,NCTM,NRTM,
+           (INTENSET(N),N=1,5),NCSPOT,NRSPOT,
+           (INTENSES(N),N=1,3)
    END IF
END DO

RETURN
END

```

C*****

```

SUBROUTINE REGRESS(X,Y,SIGMAY,NPTS,NTERMS,M,MODE,YFIT,AO,A,
+           SIGMAO,SIGMAA,R,RMUL,CHISQR,FTEST)
COMMON /DATASET/ XT(3000,7),IMAGE(3000,13)
DIMENSION X(2000),Y(2000),SIGMAY(2000),M(10),YFIT(2000),A(10),
+           SIGMAA(10),R(10)
DIMENSION WEIGHT(2000), XMEAN(10), SIGMAX(10), ARRAY(10,10)
DIMENSION INDEX(10)      !scratch space for matrix inversion routine

```

C...INITIALIZE SUMS AND ARRAYS

```

11 SUM = 0.
   YMEAN = 0.
   SIGMA = 0.
   CHISQ = 0.
   RMUL = 0.
   DO 17 I = 1, NPTS
17 YFIT(I) = 0.
21 DO 28 J = 1, NTERMS
   XMEAN(J) = 0.
   SIGMAX(J) = 0.
   DO 28 K=1, NTERMS
28 ARRAY(J,K) = 0.

```

C...ACCUMULATE WEIGHTS

```

30 DO 50 I=1,NPTS
31 IF (MODE) 32,37,39
32 IF (Y(I)) 35, 37, 33
33 WEIGHT(I) = 1./(-Y(I))
   GO TO 41
35 WEIGHT(I) = 1./ (-Y(I))
   GO TO 41
37 WEIGHT(I) = 1.
   GO TO 41
39 WEIGHT(I) = 1./SIGMAY(I)**2
41 SUM = SUM + WEIGHT(I)
   YMEAN = YMEAN + WEIGHT(I)*Y(I)
   DO 44 J = 1, NTERMS
44 XMEAN(J) = XMEAN(J) + WEIGHT(I)*FCTN(X,I,J,M)
50 CONTINUE

```

```

51 YMEAN = YMEAN/SUM
   DO 53 J=1, NTERMS
53 XMEAN(J) = XMEAN(J)/SUM
   FNPTS = NPTS
   WMEAN = SUM / FNPTS
   DO 57 I=1, NPTS
57 WEIGHT (I) = WEIGHT (I) /WMEAN

C   ACCUMULATE MATRICES R AND ARRAY
61 DO 67 I=1, NPTS
   SIGMA = SIGMA + WEIGHT(I)*(Y(I) - YMEAN)**2
   DO 67 J=1, NTERMS

      FCV = FCTN(X,I,J,M)

      SIGMAX(J) = SIGMAX(J) + WEIGHT (I)*(FCTN(X,I,J,M) - XMEAN(J))**2
      R(J) = R(J) + WEIGHT(I)*(FCTN(X,I,J,M)-XMEAN(J))*(Y(I)-YMEAN)
      DO 67 K=1, J
67 ARRAY(J,K) = ARRAY(J,K)+WEIGHT(I)*(FCTN(X,I,J,M)-XMEAN(J))*
   1 (FCTN(X,I,K,M)-XMEAN(K))
71 FREE1 = NPTS - 1
72 SIGMA = SQRT(SIGMA/FREE1)
   DO 78 J = 1, NTERMS
74 SIGMAX(J) = SQRT(SIGMAX(J)/FREE1)
   R(J) = R(J)/(FREE1*SIGMAX(J)*SIGMA)
   DO 78 K = 1, J
   ARRAY(J,K) = ARRAY(J,K) / (FREE1*SIGMAX(J)*SIGMAX(K))
78 ARRAY(K,J) = ARRAY(J,K)

C...INVERT SYMMETRIC MATRIX
81 CALL MATIN1(ARRAY,10, NTERMS, MDIM, 0, INDEX, NERROR, DET)
   IF (DET) 101, 91, 101
91 AO = 0.
   SIGMAO =0.
   RMUL = 0.
   CHISQR = 0.
   FTEST = 0.
   GO TO 150

C...CALCULATE COEFFICIENTS, FIT, AND CHI SQUARE
101 AO = YMEAN
102 DO 108 J=1, NTERMS
   DO 104 K=1, NTERMS
104 A(J) = A(J) + R(K) * ARRAY(J,K)
105 A(J) = A(J) * SIGMA/SIGMAX(J)
106 AO = AO - A(J)*XMEAN(J)
107 DO 108 I=1, NPTS
108 YFIT(I) = YFIT(I) + A(J)*FCTN(X,I,J,M)
111 DO 113 I=1, NPTS
   YFIT(I) = YFIT(I) + AO
113 CHISQ = CHISQ + WEIGHT(I)*(Y(I) - YFIT(I))**2
   FREEN = NPTS - NTERMS - 1
115 CHISQR = CHISQ*WMEAN/FREEN

C   CALCULATE UNCERTAINTIES
121 IF (MODE) 122, 124, 122
122 VARNC = 1./WMEAN
   GO TO 131

```

```

124 VARNCE = CHISQR
131 DO 133 J=1, NTERMS
132 SIGMAA(J) = ARRAY(J,J) * VARNCE / (FREE1*SIGMAX(J)**2)
133 RMUL = RMUL + A(J) * R(J) * SIGMAX(J)/SIGMA
    FREEJ = NTERMS
135 FTEST = (RMUL/FREEJ) / ((1.-RMUL)/FREEN)
136 RMUL = SQRT (RMUL)
141 SIGMA0 = VARNCE / FNPTS
    DO 145 J=1, NTERMS
    DO 145 K=1, NTERMS
145 SIGMA0 = SIGMA0 + VARNCE*XMEAN(J)*XMEAN(K)*ARRAY(J,K) /
    1 (FREE1*SIGMAX(J)*SIGMAX(K))
146 SIGMA0 = SQRT (SIGMA0)
150 RETURN
    END

```

```

FUNCTION FCTN(X,I,J,M)
COMMON /DATASET/ XT(3000,7),IMAGE(3000,13)
DIMENSION X(1), M(1)
IF (J .LE. 7) THEN
    FCTN = XT(I,J)
ELSE
    WRITE(6,10) J
10  FORMAT('O!!!SCREW UP SOMEWHERE!!!',/,
+      'In FCTN. J =',I3,' Check NTERMS.')
    WRITE(2,10) J
    STOP
END IF
RETURN
END

```

C*****

```

SUBROUTINE SUMMARY(AO,A,SIGMA0,SIGMAA,R,RMUL)
C...This subroutine prints out a 3 line summary to the analysis summary file
C according to the following format

```

C	record 1 contents	data type	1st byte	# bytes
C	date	char*9	1	9
C	time	char*8	10	8
C	calibration type	char*4	18	4
C	image file name 1	char*40	22	40
C	image type 1	char*4	62	4
C	initial element	integer*4	66	4
C	last element	integer*4	70	4
C	initial line	integer*4	74	4
C	last line	integer*4	78	4
C	image file name 2	char*40	82	40
C	image type 2	char*4	122	4
C	initial element	integer*4	126	4
C	last element	integer*4	130	4
C	initial line	integer*4	134	4
C	last line	integer*4	138	4
C	bands used	F8.0	142	8
C	Linf (1-7)	7(F7.2)	150	49
C	dmin	F7.2	199	7
C	dmax	F7.2	206	7
C	(A-A7)	8(F7.2)	213	56
C	(EA-EA7)	8(F7.2)	269	56

C	r's(1-7)	F7.2	325	7
C	rmul	F7.2	332	7
C	calib mean	F7.2	339	7
C	calib rms	"	346	7
C	cal fitted mean	"	353	7
C	ecal fit mean	"	360	7
C	cal fitted sigma	"	367	7
C	ecal fit sigma	"	374	7
C	test mean	"	381	7
C	test rms	"	388	7
C	test fitted mean	"	395	7
C	etest fit mean	"	402	7
C	test fitted sigma	"	409	7
C	etest fit sigma	"	416	7
C	# calib. pts.	"	423	7
C	# test pts.	"	430	7
C	avg. per cent error	"	437	7

```

INCLUDE 'USER$DISK:[BATHY.SOURCE]BATH.INCLUDE'
CHARACTER*9 ADATE
CHARACTER*8 ATIME, BLANK
DATA BLANK/' '/
DIMENSION A(10),SIGMAA(10),R(10)
REAL*4 KREC(50),LREC(50)

```

```

C...Open the file for appending
OPEN(UNIT=15,FILE='USER$DISK:[THFAY.TERRI.EXEC]SUMMARY.DBAS',
+ STATUS='OLD',ACCESS='APPEND')
OPEN(UNIT=16,FILE='USER$DISK:[THFAY.TERRI.EXEC]SUMMARY.LIS',
+ STATUS='OLD',ACCESS='APPEND')

```

```

C...get date and time
CALL DATE(ADATE)
CALL TIME(ATIME)

```

```

C...Set "bands used" word, first for TM then for SPOT
DO N = 0,NTM-1
  KREC(1) = KREC(1) + (N+1)*(10**N)
END DO

```

```

DO N = 0,NSPOT-1
  KREC(1) = KREC(1) + (N+1)*(10**(N+4))
END DO

```

```

C...Store the L infinities
DO N = 1,7
  KREC(1+N) = LINF(N)
END DO

```

```

C...Store min. and max. depth allowed
KREC(9) = DMIN
KREC(10) = DMAX

```

```

C...Save the fitted constants and their errors
DO N = 1,NTERMS
  KREC(11+N) = A(N)
  KREC(19+N) = SIGMAA(N)
END DO
KREC(11) = A0
KREC(19) = SIGMA0

```

```

C...Now fill up the third record
C...Store the correlation coefficients
    DO N = 1,NTERMS
        LREC(N) = R(N)
    END DO
    LREC(8) = RMUL

C...Get residual mean and rms from HBOOK and store
    LREC(9) = HSTATI(20,1) !calib. resid. mean
    LREC(10) = HSTATI(20,2) !calib. resid. rms
    LREC(15) = HSTATI(34,1) !test resid. mean
    LREC(16) = HSTATI(34,2) !test resid. rms

C...Store Gaussian params. to calib. residuals
    LREC(11) = AVC !fitted mean
    LREC(12) = SIGC(2) !std. dev. of mean
    LREC(13) = SDC !fitted sigma
    LREC(14) = SIGC(3) !std. dev. of sigma

C...Store Gaussian params. to test residuals
    LREC(17) = AVT
    LREC(18) = SIGT(2)
    LREC(19) = SDT
    LREC(20) = SIGT(3)

C...Extract number of calibration and test points from histo info
    CALL HNOFNT(20,L1) !# of entries in histo #20
    CALL HNOENT(34,L2) !# of entries in histo #34
    LREC(21) = FLOAT(L1)
    LREC(22) = FLOAT(L2)

C...Per Cent error in test points
    LREC(23) = HSTATI(31,1)

    WRITE(15,10) ADATE,ATIME,CALTYPE,IMAGEFILE1,IMAGETYPE1,IET,LET,
+               ILT,LLT,IMAGEFILE2,IMAGETYPE2,IES,LES,
+               ILS,LLS,(KREC(N),N=1,26),(LREC(N),N=1,23)

    WRITE(16,25) ADATE,ATIME,CALTYPE
    WRITE(16,26) IMAGEFILE1,IMAGETYPE1,IET,LET,ILT,LLT
    WRITE(16,26) IMAGEFILE2,IMAGETYPE2,IES,LES,ILS,LLS
    WRITE(16,27) (KREC(N),N=1,13)
    WRITE(16,28) (KREC(N),N=14,26)
    WRITE(16,28) (LREC(N),N=1,13)
    WRITE(16,29) (LREC(N),N=14,23)
    WRITE(16,*)
    WRITE(16,*)

10  FORMAT(1H ,5A,4I4,2A,4I4,F8.0,25F7.2,23F7.2)
25  FORMAT(/,3A)
26  FORMAT(2A,4I4)
27  FORMAT(F8.0,12F7.2)
28  FORMAT(13F7.2)
29  FORMAT(10F7.2)
    RETURN
    END

```

C*****

SUBROUTINE LINE_ELEM

INCLUDE 'USER\$DISK:[BATHY.SOURCE]BATH.INCLUDE'
CHARACTER*1 RESPONSE

```
WRITE(6,411) IET,LET,ILT,LLT
WRITE(6,*)
WRITE(6,*) 'Do you wish to make changes? (Y/N)'
ACCEPT 15, RESPONSE
IF (RESPONSE .EQ. 'Y' .OR. RESPONSE .EQ. 'y') then
  WRITE(6,*) 'Enter Initial Elem and Last Elem:'
  ACCEPT *, IET,LET
  WRITE(6,*) 'Enter Initial Line and Last Line:'
  ACCEPT *, ILT,LLT
ELSE
  WRITE(6,*) 'No TM line/element changes made.'
END IF
WRITE(6,412) IES,LES,ILS,LLS
WRITE(6,*) 'Do you wish to make changes? (Y/N)'
ACCEPT 15, RESPONSE
IF (RESPONSE .EQ. 'Y' .OR. RESPONSE .EQ. 'y') THEN
  WRITE(6,*) 'Enter Initial Elem and Last Elem:'
  ACCEPT *, IES,LES
  WRITE(6,*) 'Enter Initial Line and Last Line:'
  ACCEPT *, ILS,LLS
ELSE
  WRITE(6,*) 'No SPOT line/element changes made.'
END IF
```

```
15 FORMAT(A)
411 FORMAT('OInitial Element TM =',I5,'    Last Element TM =',I5/,
+         ' Initial Line TM      =',I5,'    Last Line TM      =',I5/)
412 FORMAT('OInitial Element SPOT =',I5,' Last Element SPOT =',I5/,
+         ' Initial Line SPOT    =',I5,' Last Line SPOT    =',I5/)
```

RETURN
END

PROGRAM MODSIEVE

C.....This program performs a MOD sieve on an NOS data file;
C.....that is, every Nth point is accepted with a MOD of N.

```

INTEGER          LATD,LATM,LGD,LGM,MODR
INTEGER*4        TMSCAN,TMELEM,SPOTSCAN,SPOTELEM
DOUBLE PRECISION EAS,NOR
REAL             DEPTH,RLATS,RLGS
CHARACTER*132    HEADING
CHARACTER*40     INFILE, OUTFILE

```

```

WRITE(6,*)
WRITE(6,*)
WRITE(6,*) '*****'
WRITE(6,*) 'This program performs a MOD sieve on '
WRITE(6,*) 'an NOS data file.  For example, with a'
WRITE(6,*) 'MOD of N, every Nth point is accepted.'
WRITE(6,*) '*****'
WRITE(6,*)

```

```

WRITE(6,*) 'Enter NOS input file name:'
ACCEPT 20, INFILE
WRITE(6,*) 'Enter output file name:'
ACCEPT 20, OUTFILE

```

```

20  FORMAT(A)
WRITE(6,*) 'Enter the increment:'
WRITE(6,*) '(for example, an increment of 100'
WRITE(6,*) ' means every 100th point is accepted)'
READ(5,*) MODR

```

```

OPEN(11,FILE=INFILE,STATUS='OLD',READONLY)
OPEN(12,FILE=OUTFILE,STATUS='NEW')

```

```

READ(11,500) HEADING
WRITE(12,500) HEADING

```

```

DO I=1,336000

```

```

      READ(11,1000,END=100) LATD,LATM,RLATS,LGD,LGM,RLGS,
+                               EAS,NOR,DEPTH,TMELEM,TMSCAN,
+                               SPOTELEM,SPOTSCAN

```

```

      IF (MOD(I,MODR) .EQ. 1) THEN !Take every "MODR"th point.
        WRITE(12,1000) LATD,LATM,RLATS,LGD,LGM,RLGS,
+                               EAS,NOR,DEPTH,TMELEM,TMSCAN,
+                               SPOTELEM,SPOTSCAN

```

```

      END IF

```

```

    END DO

```

```

100  CONTINUE

```

```

500  FORMAT(A132)

```

```

1000  FORMAT(1X,I2,I2,F5.2,1X,I3,I2,F5.2,2X,F10.0,2X,F10.0,2X,
+          F7.1,2X,I6,2X,I6,2X,I6,2X,I6)
END

```

```

PROGRAM OVLBATHY
C.....Bathymetry on Overlay data (i.e., a coregistered image of TM and
C.....SPOT data).

```

```

PARAMETER      (N=512)
BYTE            AIM(4500,8)
INTEGER        IMAGE(4500,8),LTM(5),LSPOT(3)
REAL           ATM(4),ASPOT(3),A0
BYTE           BATHY(4500)
INTEGER        M,P,Q,I,J,N3,N1,N2,Z
CHARACTER*40    OVLINEFILE,OUTFILE
CHARACTER*132   COMMENT

```

```

WRITE(6,*) '*****'
WRITE(6,*) '*'
WRITE(6,*) '*          BATHYMETRY          *'
WRITE(6,*) '*'
WRITE(6,*) '*****'
WRITE(6,*)

```

```

WRITE(6,*) 'This program is designed to handle data'
WRITE(6,*) 'in the following format: '
WRITE(6,*)
WRITE(6,*) '  For Overlay data, the input file must be '
WRITE(6,*) '  TM onto SPOT with bands in the following order:'
WRITE(6,*) '  SPOT 1,2,3, then TM 1,2,3,4,5. '
WRITE(6,*)

```

```

LUN3 = 0
CALL FILE_OPEN(LUN3,OVLINEFILE)

```

```

WRITE(6,*)
WRITE(6,*) 'Enter output file:'
READ(5,50) OUTFILE
50  FORMAT(A40)
OPEN(UNIT=15,FILE=OUTFILE,STATUS='NEW',
+    FORM='UNFORMATTED',ACCESS='DIRECT',RECL=128)

```

```

C..... READ HEADER OF OVLINEFILE
READ(LUN3,REC=1) NBIH,NBPR,IL,LL,IE,LE,NC,IDESC

```

```

C..... WRITE HEADER OF THE OUTPUT FILE. BATHY IMAGES WILL HAVE ONLY ONE CHANNEL.
WRITE(15,REC=1) NBIH,NBPR,IL,LL,IE,LE,1,IDESC

```

```

WRITE(6,*) 'What comments would you like written to the'
WRITE(6,*) 'output file? Please limit them to 132 characters.'
READ(5,200) COMMENT
WRITE(15,REC=2) COMMENT
200  FORMAT(A)

```

```

CALL GET_LINFS(LTM,LSPOT)

CALL GET_COEFF(ATM,ASPOT,A0)

```

```

NREC = 3      ! INPUT DATA FOR LINE 1 CHANNEL 1 BEGINS IN THE THIRD
              ! 512-BYTE BLOCK OF THE INPUT FILE
NREC3 = 3     ! CONTROL FOR WRITING TO THE CORRECT POSITION IN THE
              ! OUTPUT FILE
NL = LL-IL+1 ! TOTAL NUMBER OF LINES.
NE = LE-IE+1 ! TOTAL NUMBER OF ELEMENTS.
MAXR = INT((NBPR*NL*NC)/N+2) ! NUMBER OF BLOCKS IN INFILE.
MAXZ = INT(NBPR/N) ! NUMBER OF BLOCKS PER INPUT LINE (ONE CHANNEL).
MAXREC = MAXR-(NC*MAXZ)+1

```

```

DO WHILE (NREC .LE. MAXREC) !BLOCK MAXREC IS START OF LAST LINE
                          ! CHANNEL 1.

```

```

C.....READS EACH 512-BYTE BLOCK OF THE INPUT FILE AND STORES IT IN THE
C.....APPROPRIATE BYTE ARRAY. ALL THE INFORMATION FOR EVERY CHANNEL FOR ONE
C.....LINE OF INPUT DATA IS READ HERE.

```

```

DO I = 1,NC
  DO Z = 0,MAXZ-1
    N1 = (512*Z)+1
    N2 = 512*(Z+1)
    IF(N2.GT.NE) N2 = NE
    NREC2 = NREC+Z+((I-1)*MAXZ)
    READ(LUN3,REC=NREC2) (AIM(N3,I), N3=N1,N2)
  END DO
END DO

```

```

C.....CONVERTS BYTES TO INTEGER*4 AND CHANGES NEGATIVES TO POSITIVES.

```

```

DO I = 1,NC
  DO NCOL = 1,NE
    IMAGE(NCOL,I) = AIM(NCOL,I)
    IF (IMAGE(NCOL,I) .LT. 0)
+     IMAGE(NCOL,I) = IMAGE(NCOL,I) + 256
  END DO
END DO

```

```

CALL BIGD(IMAGE,BATHY,AO,ATM,ASPT,NE,LTM,LSPOT) !DOING BATHYMETRY

```

```

C.....CONVERTS POSITIVES BIGGER THAN 128 TO NEGATIVES FOR STORAGE AS BYTES IN
C.....BYTE ARRAY BATHY.

```

```

DO J = 1,NE
  IF (BATHY(J) .GE. 128)
+   BATHY(J) = BATHY(J) - 256
END DO

```

```

C.....WRITE TO OUTPUT FILE THE CALCULATED BATHYMETRIC VALUES.

```

```

DO Z = 0,MAXZ-1
  N1 = (512*Z)+1
  N2 = 512*(Z+1)
  IF(N2.GT.NE) N2 = NE
  WRITE(15,REC=NREC3+Z) (BATHY(N3), N3=N1,N2)
END DO
NREC3 = NREC3 + MAXZ      !Increment NREC3 to skip to the first
                          !block of next line.

```

C.....INCREMENT NREC SO THAT THE PROGRAM READS THE GRAY LEVELS FOR THE NEXT
 C.....SCAN LINE. REMEMBER, NREC MUST BE ADVANCED TO SKIP ALL RECORDS
 C.....CONTAINING CHANNELS FOR THE SCAN LINE BEING PROCESSED.

NREC = NREC + (NC*MAXZ)

END DO

```
WRITE(6,*) '*****'
WRITE(6,*) '*'
WRITE(6,*) '*   BATHYMETRY COMPLETED   *'
WRITE(6,*) '*'
WRITE(6,*) '*****'
WRITE(6,*)
END
```

C*****

SUBROUTINE BIGD(IMAGE,BATHY,AO,ATM,ASPOT,NE,LTM,LSPOT)

C.....THIS SUBROUTINE CALCULATES THE BATYMETRY VALUES FOR EACH ELEMENT.

```
INTEGER IMAGE(4500,8),LTM(5),LSPOT(3),NE
REAL    ATM(4),ASPOT(3),AO
BYTE BATHY(4500)
```

```
DO 400 J = 1,NE
  IF ((IMAGE(J,8) - LTM(5)) .GT. 0) THEN
    BATHY(J) = 250
  ELSE IF ((IMAGE(J,4) - LTM(1)) .GT. 0) THEN
    BATHY(J) = NINT(AO +
-      ASPOT(1)*ALOG(MAX(FLOAT(IMAGE(J,1))-LSPOT(1)),1.0)) +
-      ASPOT(2)*ALOG(MAX(FLOAT(IMAGE(J,2))-LSPOT(2)),1.0)) +
-      ASPOT(3)*ALOG(MAX(FLOAT(IMAGE(J,3))-LSPOT(3)),1.0)) +
-      ATM(1)*ALOG(MAX(FLOAT(IMAGE(J,4))-LTM(1)),1.0)) +
-      ATM(2)*ALOG(MAX(FLOAT(IMAGE(J,6))-LTM(2)),1.0)) +
-      ATM(3)*ALOG(MAX(FLOAT(IMAGE(J,7))-LTM(3)),1.0)) +
-      ATM(4)*ALOG(MAX(FLOAT(IMAGE(J,8))-LTM(4)),1.0)))
  ELSE
    BATHY(J) = 255
  END IF
400 CONTINUE
RETURN
END
```

C*****

SUBROUTINE FILE_OPEN(LUN3,OVLINFILE)

```
INTEGER LUN3
CHARACTER*40 OVLINFILE
```

```
WRITE(6,*) 'Enter the overlay input file:'
LUN3 = 10
READ (5,100) OVLINFILE
OPEN(UNIT=LUN3,FILE=OVLINFILE,STATUS='OLD',IOSTAT=IOS1,
+   READONLY,FORM='UNFORMATTED',ACCESS='DIRECT',RECL=128)
```

100 FORMAT(A40)

RETURN
END

C*****

SUBROUTINE GET_LINFS(LTM,LSPOT)

INTEGER LTM(5),LSPOT(3)

WRITE(6,*)
WRITE(6,*)
WRITE(6,*)
WRITE(6,*) 'Enter L infinities in the following order:'
WRITE(6,*) 'SPOT bands 1-3'
WRITE(6,*) 'TM bands 1-5'
WRITE(6,*)
WRITE(6,*)
DO M=1,3
WRITE(6,75) M
WRITE(6,*) '(If no L infinity, enter 0)'
READ(5,*) LSPOT(M)
END DO
DO M=1,5
WRITE(6,75) M
WRITE(6,*) '(If no L infinity, enter 0)'
READ(5,*) LTM(M)
END DO
WRITE(6,*)
WRITE(6,*)
WRITE(6,*)

75 FORMAT(1X,'Enter L infinity for band ',I1)

RETURN
END

C*****

SUBROUTINE GET_COEFF(ATM,ASPOT,A0)

REAL ATM(4),ASPOT(3),A0

WRITE(6,*)
WRITE(6,*)
WRITE(6,*)
WRITE(6,*) 'Please enter A0:'
READ(5,*) A0
WRITE(6,*)

WRITE(6,*) 'Enter remaining coefficients in the following order:'
WRITE(6,*) 'ATM(1)-ATM(4) for TM bands 1-4'
WRITE(6,*) 'ASPOT(1)-ASPOT(3) for SPOT bands 1-3'
WRITE(6,*)
WRITE(6,*)

```
      DO M=1,4
        WRITE(6,100) M
        WRITE(6,*) '(If no coefficient, enter 0)'
        READ(5,*) ATM(M)
      END DO
      DO M=1,3
        WRITE(6,200) M
        WRITE(6,*) '(If no coefficient, enter 0)'
        READ(5,*) ASPOT(M)
      END DO
    WRITE(6,*)
    WRITE(6,*)
```

```
100  FORMAT(1X,'Enter coefficient ATM(',I1,')')
200  FORMAT(1X,'Enter coefficient ASPOT(',I1,')')
      RETURN
      END
```

PROGRAM SPOTBATHY
C.....Bathymetry on SPOT data.

```
PARAMETER (N=512)
BYTE      AIM(4500,3)
INTEGER   IMAGE(4500,3),L(3)
REAL      A(0:2)
BYTE      BATHY(4500)
INTEGER   M,P,Q,I,J,N3,N1,N2,Z
CHARACTER*40 SPOTINFILE,OUTFILE
CHARACTER*132 COMMENT
```

```
WRITE(6,*) '*****'
WRITE(6,*) '*'
WRITE(6,*) '*          BATHYMETRY          *'
WRITE(6,*) '*'
WRITE(6,*) '*****'
WRITE(6,*)
```

```
WRITE(6,*) 'This program is designed to handle data'
WRITE(6,*) 'in the following format:'
WRITE(6,*)
WRITE(6,*) '  For SPOT data the input file must be'
WRITE(6,*) '  bands 1, 2, 3, in that order.'
WRITE(6,*)
```

```
LUN2 = 0
CALL FILE_OPEN(LUN2,SPOTINFILE)
```

```
WRITE(6,*)
WRITE(6,*) 'Enter output file:'
READ(5,50) OUTFILE
50  FORMAT(A40)
OPEN(UNIT=15,FILE=OUTFILE,STATUS='NEW',
+    FORM='UNFORMATTED',ACCESS='DIRECT',RECL=128)
```

C.....Read header of SPOTINFILE.
READ(LUN2,REC=1) NBIH,NBPR,IL,LL,IE,LE,NC,IDESC

C..... Write header of the output file, which will have only one channel.
WRITE(15,REC=1) NBIH,NBPR,IL,LL,IE,LE,1,IDESC

```
WRITE(6,*) 'What comments would you like written to the'
WRITE(6,*) 'output file? Please limit them to 132 characters.'
READ(5,200) COMMENT
WRITE(15,REC=2) COMMENT
200  FORMAT(A)
```

```
WRITE(6,*)
WRITE(6,*)
WRITE(6,*)
WRITE(6,*) 'Enter L infinities in the following order:'
WRITE(6,*) 'SPOT bands 1-3'
WRITE(6,*)
WRITE(6,*)
WRITE(6,*)
```

```

DO M=1,3
  WRITE(6,75) M
  WRITE(6,*) '(If no L infinity, enter 0)'
  READ(5,*) L(M)
END DO
75  FORMAT(1X,'Enter L infinity for band ',I1)

WRITE(6,*)
WRITE(6,*)
WRITE(6,*)
WRITE(6,*)
WRITE(6,*)
WRITE(6,*) 'Enter coefficients in the following order:'
WRITE(6,*) 'A0'
WRITE(6,*) 'A1-A2 for SPOT bands 1-2'
WRITE(6,*)
WRITE(6,*)

DO M=0,2
  WRITE(6,100) M
  WRITE(6,*) '(If no coefficient, enter 0)'
  READ(5,*) A(M)
END DO
100  FORMAT(1X,'Enter coefficient A(',I1,')')

NREC = 3      ! INPUT DATA FOR LINE 1 CHANNEL 1 BEGINS IN THE
               !   THIRD 512-BYTE BLOCK OF THE INPUT FILE
NREC3 = 3      ! CONTROL FOR WRITING TO THE CORRECT POSITION IN THE
               !   OUTPUT FILE
NL = LL-IL+1 ! TOTAL NUMBER OF LINES.
NE = LE-IE+1 ! TOTAL NUMBER OF ELEMENTS.
MAXR = INT((NBPR*NL*NC)/N+2) ! NUMBER OF BLOCKS IN INFILE.
MAXZ = INT(NBPR/N) ! NUMBER OF BLOCKS PER INPUT LINE (ONE CHANNEL).
MAXREC = MAXR-(NC*MAXZ)+1

DO WHILE (NREC .LE. MAXREC) !BLOCK MAXREC IS START OF LAST LINE
                             ! CHANNEL 1.

C.....READ EACH 512-BYTE BLOCK OF THE INPUT FILE AND STORE IT IN THE
C.....APPROPRIATE BYTE ARRAY. ALL THE INFORMATION FOR EVERY CHANNEL
C.....FOR ONE LINE OF INPUT DATA IS READ HERE.
  DO I= 1,NC
    DO Z = 0,MAXZ-1
      N1 = (512*Z)+1
      N2 = 512*(Z+1)
      IF(N2.GT.NE) N2 = NE
      NREC2 = NREC+Z+((I-1)*MAXZ)
      READ(LUN2,REC=NREC2) (AIM(N3,I), N3=N1,N2)
    END DO
  END DO

C.....CONVERTS BYTES TO INTEGER*4 AND CHANGES NEGATIVES TO POSITIVES.

DO I = 1,NC
  DO NCOL = 1,NE
    IMAGE(NCOL,I) = AIM(NCOL,I)
    IF (IMAGE(NCOL,I) .LT. 0)
      + IMAGE(NCOL,I) = IMAGE(NCOL,I) + 256
  
```

```

        END DO
    END DO

```

```

    CALL BIGD(IMAGE,BATHY,A,NE,L) !DOING BATHYMETRY

```

```

C.....CONVERTS POSITIVES BIGGER THAN 128 TO NEGATIVES FOR STORAGE AS BYTES IN
C.....BYTE ARRAY BATHY.

```

```

        DO J = 1,NE
            IF (BATHY(J) .GE. 128)
+           BATHY(J) = BATHY(J) - 256
        END DO

```

```

C.....WRITE TO OUTPUT FILE THE CALCULATED BATHYMETRIC VALUES.

```

```

        DO Z = 0,MAXZ-1
            N1 = (512*Z)+1
            N2 = 512*(Z+1)
            IF(N2.GT.NE) N2 = NE
            WRITE(15,REC=NREC3+Z) (BATHY(N3), N3=N1,N2)
        END DO
        NREC3 = NREC3 + MAXZ      !Increment NREC3 to skip to the first
                                !block of the next line.

```

```

C.....INCREMENT NREC SO THAT THE PROGRAM READS THE GRAY LEVELS FOR THE NEXT
C.....SCAN LINE.  REMEMBER, NREC MUST BE ADVANCED TO SKIP ALL RECORDS
C.....CONTAINING CHANNELS FOR THE SCAN LINE BEING PROCESSED.

```

```

        NREC = NREC + (NC*MAXZ)

```

```

    END DO

```

```

    WRITE(6,*) '*****'
    WRITE(6,*) '*'
    WRITE(6,*) '*   BATHYMETRY COMPLETED   *'
    WRITE(6,*) '*'
    WRITE(6,*) '*****'
    WRITE(6,*)
    END

```

```

C*****

```

```

    SUBROUTINE BIGD(IMAGE,BATHY,A,NE,L)

```

```

C.....THIS SUBROUTINE CALCULATES THE BATYMETRY VALUES FOR EACH ELEMENT.

```

```

    INTEGER IMAGE(4500,3),L(3),NE
    REAL    A(0:2)
    BYTE BATHY(4500)

```

```

        DO 400 J = 1,NE
            IF ((IMAGE(J,3) - L(3)) .GT. 0) THEN
                BATHY(J) = 250
            ELSE IF ((IMAGE(J,1) - L(1)) .GT. 0) THEN
                BATHY(J) = NINT(A(0) +
+                               A(1)*ALOG(MAX(FLOAT(IMAGE(J,1)-L(1)),1.0)) +
-                               A(2)*ALOG(MAX(FLOAT(IMAGE(J,2)-L(2)),1.0)))

```

```

        ELSE
          BATHY(J) = 255
        END IF
400    CONTINUE
        RETURN
      END

```

C*****

```

      SUBROUTINE FILE_OPEN(LUN2,SPOTINFILE)

      INTEGER    LUN2
      CHARACTER*40 SPOTINFILE

      WRITE(6,*) 'Enter SPOT input file:'
      READ(5,100) SPOTINFILE
      LUN2 = 10
      OPEN(UNIT=LUN2,FILE=SPOTINFILE,STATUS='OLD',IOSTAT=IOS1,
+        READONLY,FORM='UNFORMATTED',ACCESS='DIRECT',RECL=128)

100  FORMAT(A40)

      RETURN
      END

```

PROGRAM STANDARD

C.....This program calculates the corrections needed to convert from
C.....local geodetic system lat/lon's to World Geodetic System 1984 (WGS84)
C.....lat/lon's. The standard Molodensky formulas are used.

C.....Formulae and data are from DMA Technical Report 8350.2, 30 Sept 87
C.....DoD World Geodetic System 1984 (WGS84).

```
INTEGER  LATD,LATM,LOND,LONM
REAL     RLATS,RLONS      !Fractional seconds
REAL     DELLAT,DELLON    !Lat/lon corrections, in seconds
REAL     DLAT,DLON        !lat/lon to be corrected
REAL     DELH             !Geodetic height correction
REAL     WGSLAT,WGSLON    !Corrected lat/lon

WRITE(6,*) ' "Local GS" to WGS 1984 conversion - '
WRITE(6,*) ' STANDARD MOLODENSKY FORMULAS '
WRITE(6,*)
WRITE(6,*) ' ENTER A SAMPLE POINT (IN "LOCAL GS") '
WRITE(6,*) ' Enter lat. degree (N+/S-), minutes, REAL seconds: '
READ(5,*)  LATD,LATM,RLATS
WRITE(6,*) ' Enter lon. degree (E+/W-), minutes, REAL seconds: '
READ(5,*)  LOND,LONM,RLONS
```

C.....Convert to degrees in decimal:
DLAT = DECDEG(LATD,LATM,RLATS)
DLON = DECDEG(LOND,LONM,RLONS)

C.....Echo print:
WRITE(6,*)
WRITE(6,*) ' Lat (deg,min,sec): ',LATD,LATM,RLATS
WRITE(6,*) ' Lon (deg,min,sec): ',LOND,LONM,RLONS
WRITE(6,*)
WRITE(6,*) ' Lat (dec. degree): ',DLAT
WRITE(6,*) ' Lon (dec. degree): ',DLON
WRITE(6,*)

C.....Calculate the corrections:
CALL LOCAL_WGS84(DLAT,DLON,DELLAT,DELLON,DELH)

C.....Write the corrections.
WRITE(6,*) ' Lat/lon corrections, in seconds: ',DELLAT,DELLON

WGSLAT = DLAT + (DELLAT/3600.0)
WGSLON = DLON + (DELLON/3600.0)

C.....Convert decimal degrees to degrees, minutes, REAL seconds:
CALL DMS(WGSLAT,LATD,LATM,RLATS)
CALL DMS(WGSLON,LOND,LONM,RLONS)

C.....Echo print:
WRITE(6,*)
WRITE(6,*) ' WGS lat (degree): ',WGSLAT
WRITE(6,*) ' WGS lon (degree): ',WGSLON
WRITE(6,*)
WRITE(6,*) ' WGS lat (deg,min,sec): ',LATD,LATM,RLATS
WRITE(6,*) ' WGS lon (deg,min,sec): ',LOND,LONM,RLONS
WRITE(6,*)

END

C*****

 SUBROUTINE DMS(L,D,M,S)
C.....Subroutine DMS converts from decimal lat/lon L to degree D,
C.....minute M, REAL seconds S.

 INTEGER D,M
 REAL L,S

 D = INT(L)
 DIFF = ABS(L - FLOAT(D))
 REALMINUTES = DIFF * 60.0
 M = INT(REALMINUTES)
 DIFF = REALMINUTES - FLOAT(M)
 S = DIFF * 60.0

 RETURN
 END

C*****

 REAL FUNCTION DECDEG(D,M,S)
C.....Convert latitude/longitude D,M,S (in deg,min,sec) to REAL decimal degrees.
C.....Only D should carry the sign.

 REAL S, RD
 INTEGER D,M

 RD = FLOAT(D)
 IF (RD .LT. 0.) THEN
 DECDEG = -1.0 * (ABS(RD) + FLOAT(M)/60.0 + S/3600.0)
 ELSE
 DECDEG = RD + FLOAT(M)/60.0 + S/3600.0
 ENDIF

 RETURN
 END

C*****

 SUBROUTINE LOCAL_WGS84(DLAT, DLON, DELLAT, DELLON, DELH)

 REAL DLAT, DLON, DELLAT, DELLON, DELH

C.....Subroutine to calculate corrections used to convert from LOCAL GS to
C.....WGS 1984. THE STANDARD MOLODENSKY FORMULAS ARE USED.

C.....NOTE: this has not been checked for elevation corrections.

C

C.....Formulae and data from DMA Technical Report 8350.2, 30 Sept 87

C.....DoD World Geodetic System 1984 (WGS84)

C

C.....To use correction factors DEL* (in seconds), etc., use these formulae
C.....where DLAT, etc., are in LOCAL and WGS LAT, etc., is in WGS84:

C

C WGS LAT = DLAT + (DELPHI/3600.0)

C WGSLOD = DLON + (DELLAMBDA/3600.0)

REAL DELX,DELY,DELZ,DELA,DELF,RN,RM,E2,A,B,H,E,F
REAL DELPHI,DELLAMBDA !Corrections, in seconds.
REAL DELHEIGHT

DATA H / 0.0 / !Ignore geodetic height.

C.....The following is for NAD27 Bahamas (should be good for Puerto Rico):

DATA DELX,DELY,DELZ / -4.0,154.0,178.0 / !p. 7-22.

C.....The following is Clark 1866 spheroid:

DATA DELA,DELF / -69.4,-.000037264639 / !p. 7-22.

DATA A,F_INVERSE / 6378206.4,294.9786982 / !p. 7-12.

C.....The following data is for Puerto Rico datum (p. 7-26) for check:

C DATA DELX,DELY,DELZ /11.,72.,-101./

C DATA DELA,DELF /-69.4, -0.37264639E-4/

C DATA A,F_INVERSE / 6378206.4,294.9786982 / !p. 7-12.

F = 1./F_INVERSE

S1 = SIND(1.0/3600.0)

B = A*(1-F)

E2 = F*(2-F)

DENOM = SQRT(1-E2*(SIND(DLAT)**2))

RN = A / DENOM

RM = A*(1-E2) / (DENOM**3)

DELPHI = (-DELX*SIND(DLAT)*COSD(DLON) -
 DELY*SIND(DLAT)*SIND(DLON) +
 DELZ*COSD(DLAT) +
 DELA*(RN*E2*SIND(DLAT)*COSD(DLAT))/A +
 DELF*(RM*(A/B) + RN*(B/A))*SIND(DLAT)*COSD(DLAT))/
 ((RM + H)*S1)

DELLAMBDA = (-DELX*SIND(DLON) + DELY*COSD(DLON)) /
 ((RN + H)*COSD(DLAT)*S1)

DELHEIGHT = DELX*COSD(DLAT)*COSD(DLON) +
 DELY*COSD(DLAT)*SIND(DLON) + DELZ*SIND(DLAT) -
 DELA*A/RN + DELF*(B/A)*RN*SIND(DLAT)*SIND(DLAT)

DELLAT = DELPHI

DELLON = DELLAMBDA

DELH = DELHEIGHT

RETURN

END

PROGRAM TMBATHY
 C.....TMBATHY creates a bathymetric image from a raw TM image file.

```

PARAMETER      (N=512)

BYTE           AIM(4500,5)
INTEGER        IMAGE(4500,5),L(5)
REAL           A(0:4)
BYTE           BATHY(4500)
INTEGER        M,P,Q,I,J,N3,N1,N2,Z
CHARACTER*8    SPACE8
CHARACTER*40   TMINFILE,OUTFILE
CHARACTER*132  COMMENTS
  
```

```

WRITE(6,*) '*****'
WRITE(6,*) '*                      *'
WRITE(6,*) '*          BATHYMETRY          *'
WRITE(6,*) '*                      *'
WRITE(6,*) '*****'
WRITE(6,*)
  
```

```

WRITE(6,*) 'This program is designed to handle data'
WRITE(6,*) 'in the following format:'
WRITE(6,*)
WRITE(6,*) '  For TM data the input file must be'
WRITE(6,*) '  bands 1, 2, 3, 4, 5, in that order.'
WRITE(6,*)
  
```

```

WRITE(6,*)
WRITE(6,*) 'Enter TM input file:'
READ(5,50) TMINFILE
OPEN(UNIT=10,FILE=TMINFILE,STATUS='OLD',IOSTAT=IOS1,
+   READONLY,FORM='UNFORMATTED',ACCESS='DIRECT',RECL=128)
  
```

```

WRITE(6,*)
WRITE(6,*) 'Enter output file:'
READ(5,50) OUTFILE
OPEN(UNIT=15,FILE=OUTFILE,STATUS='NEW',
+   FORM='UNFORMATTED',ACCESS='DIRECT',RECL=128)
  
```

C.....Read header of TM input file.
 READ(10,REC=1) NBIH,NBPR,IL,LL,IE,LE,NC,IDESC

C.....Write header to the outut file. Bathymetry images will have 1 channel.
 WRITE(15,REC=1) NBIH,NBPR,IL,LL,IE,LE,1,IDESC

```

WRITE(6,*)
WRITE(6,*) 'What comments would you like written to the'
WRITE(6,*) 'output file? Limit to 132 characters.'
READ(5,50) COMMENTS
50  FORMAT(A)
  
```

```

WRITE(6,*)
WRITE(6,*)
WRITE(6,*)
WRITE(6,*) 'Enter L infinities in the following order:'
WRITE(6,*) 'TM bands 1-5'
  
```

```
WRITE(6,*)
WRITE(6,*)
WRITE(6,*)
```

```
DO M=1,5
  WRITE(6,75) M
  WRITE(6,*) '(If no L infinity, enter 0)'
  READ(5,*) L(M)
```

```
END DO
75 FORMAT(1X,'Enter L infinity for band ',I1)
```

```
WRITE(6,*)
WRITE(6,*)
WRITE(6,*)
WRITE(6,*)
WRITE(6,*)
WRITE(6,*) 'Enter coefficients in the following order:'
WRITE(6,*) 'A0'
WRITE(6,*) 'A1-A4 for TM bands 1-4'
WRITE(6,*)
WRITE(6,*)
```

```
DO M=0,4
  WRITE(6,100) M
  WRITE(6,*) '(If no coefficient, enter 0)'
  READ(5,*) A(M)
```

```
END DO
100 FORMAT(1X,'Enter coefficient A(',I1,')')
```

C.....Write comments, consisting of L infinities, coefficients, files, and
C.....COMMENTS.

```
WRITE(15,REC=2) 'First 5 words ',
+               'are L1-L5, and ',
+               'next 5 words are',
+               'A0-A4 * 1000: ',
+               (L(I),I=1,5),
+               (INT(A(I)*1000.),I=0,4),0,0,
+               'input file: ',
+               TMINFILE,SPACE8,
+               'comments: ',
+               COMMENTS
```

```
NREC = 3      !Input data for line 1, channel 1 begins in this block.
NREC3 = 3     !Control for writing to the correct position in output file.
NL = LL-IL+1 !Total number of lines.
NE = LE-IE+1 !Total number of elements
MAXR = INT((NBPR*NL*NC)/N+2) !Number of blocks in input file.
MAXZ = INT(NBPR/N) !Number of blocks per input line (one channel).
MAXREC = MAXR-(NC*MAXZ)+1 !Block MAXREC is start of last line channel 1.
```

```
DO WHILE (NREC .LE. MAXREC)
```

C.....Read each 512-byte block of the input file and store it in the
C.....appropriate byte array. All the information for every channel for one
C.....line of input data is read here.

```
DO I= 1,NC
  DO Z = 0,MAXZ-1
    N1 = (512*Z)+1
```

```

      N2 = 512*(Z+1)
      IF(N2.GT.NE) N2 = NE
      NREC2 = NREC+Z+((I-1)*MAXZ)
      READ(10,REC=NREC2) (AIM(N3,I), N3=N1,N2)
    END DO
  END DO

```

C.....Converts bytes to integer*4 and changes negatives to positives.

```

    DO I = 1,NC
      DO NCOL = 1,NE
        IMAGE(NCOL,I) = AIM(NCOL,I)
        IF (IMAGE(NCOL,I) .LT. 0)
+         IMAGE(NCOL,I) = IMAGE(NCOL,I) + 256
      END DO
    END DO

```

CALL BIGD(IMAGE,BATHY,A,NE,L) !Doing bathymetry

C.....Converts positives bigger than 128 to negatives for storage as bytes

C.....in byte array BATHY.

```

    DO J = 1,NE
      IF (BATHY(J) .GE. 128)
+       BATHY(J) = BATHY(J) - 256
    END DO

```

C.....Write to output file the calculated bathymetric values.

```

    DO Z = 0,MAXZ-1
      N1 = (512*Z)+1
      N2 = 512*(Z+1)
      IF(N2.GT.NE) N2 = NE
      WRITE(15,REC=NREC3+Z) (BATHY(N3), N3=N1,N2)
    END DO
    NREC3 = NREC3 + MAXZ !Increment NREC3 to skip to first block
                        !of next line.

```

C.....Increment NREC so that the program reads the gray levels for the next

C.....scan line. Note that NREC must be advanced to skip all records containing

C.....channels for the scan line being processed.

NREC = NREC + (NC*MAXZ)

END DO !"WHILE" loop.

```

WRITE(6,*) '*****'
WRITE(6,*) '*                                     *'
WRITE(6,*) '*   BATHYMETRY COMPLETED   *'
WRITE(6,*) '*                                     *'
WRITE(6,*) '*****'
WRITE(6,*)
END

```

C*****

SUBROUTINE BIGD(IMAGE,BATHY,A,NE,L)

C.....Subroutine BIGD calculates the bathymetry values for each element

```

INTEGER IMAGE(4500,5),L(5),NE
REAL      A(0:4)

```

BYTE BATHY(4500)

DO 400 J = 1,NE

IF (IMAGE(J,5) .GT. L(5)) THEN !element is land.

BATHY(J) = 250

ELSE IF (IMAGE(J,1) .GT. L(1)) THEN !element is water.

BATHY(J) = NINT(A(0) +

- A(1)*ALOG(MAX(FLOAT(IMAGE(J,1)-L(1)),1.0)) +

- A(2)*ALOG(MAX(FLOAT(IMAGE(J,2)-L(2)),1.0)) +

- A(3)*ALOG(MAX(FLOAT(IMAGE(J,3)-L(3)),1.0)) +

- A(4)*ALOG(MAX(FLOAT(IMAGE(J,4)-L(4)),1.0)))

ELSE

BATHY(J) = 255 !element is deep water.

END IF

400 CONTINUE

RETURN

END

``` PROGRAM UTM2ST ```

C....This program converts UTM's to element and scan in both the TM
C....and SPOT coordinates. Only NOAA calibration points may be used
C....in the CALFILE.

```

      INTEGER*4      TMSCAN,TMELEM,SPOTSCAN,SPOTELEM
      DOUBLE PRECISION SLTM(3),ELTM(3),SLSPOT(3),ELSPOT(3),
      +              EAS,NOR
      REAL           DEPTH
      CHARACTER*40    TCOEFUT,SCOEPUT,CALFILE,OUTFILE

```

```
DATA LUCOEF1/24/,LUCOEF2/26/,LUOUT/25/, LUSURV/11/
```

C....The COEFUT files contain the georeferencing coefficients
C....and were created by ELAS.

```

      WRITE(6,*) 'Enter TM COEFUT.LEL file:'
      WRITE(6,*) '(Be certain to give full path name)'
      ACCEPT 100, TCOEFUT
      OPEN(LUCOEF1,FILE=TCOEFUT,STATUS='OLD',READONLY)
      READ(LUCOEF1,'(1X,D60.40)') (SLTM(I),I=1,3),(ELTM(I),I=1,3)
      WRITE(6,*)

```

```

      WRITE(6,*) 'Enter SPOT COEFUT.LEL file:'
      WRITE(6,*) '(Be certain to give full path name)'
      ACCEPT 100, SCOEPUT
      OPEN(LUCOEF2,FILE=SCOEPUT,STATUS='OLD',READONLY)
      READ(LUCOEF2,'(1X,D60.40)') (SLSPOT(I),I=1,3),(ELSPOT(I),I=1,3)
      WRITE(6,*)

```

```

      WRITE(6,*) 'Enter the NOAA calibration file:'
      WRITE(6,*) '(NOTE: this file must have 0 depth'
      WRITE(6,*) ' in the last record to finish properly)'
      ACCEPT 100, CALFILE
      OPEN(LUSURV,FILE=CALFILE,STATUS='OLD',READONLY)
      WRITE(6,*)

```

```

      WRITE(6,*) 'Enter the output file:'
      ACCEPT 100, OUTFILE
      OPEN(LUOUT,FILE=OUTFILE,STATUS='NEW')

```

```
WRITE(LUOUT,1001)          !Writes heading line to output file.
```

```
READ(LUSURV,500) EAS,NOR,DEPTH
```

```
DO WHILE (DEPTH .NE. 0.)
```

C.....The georeferencing coefficients are now used to generate scanlines
C.....and elements for points in the input file.

```

      TMSCAN = SLTM(1) + SLTM(2)*EAS + SLTM(3)*NOR +.5
      TMELEM = ELTM(1) + ELTM(2)*EAS + ELTM(3)*NOR +.5
      SPOTSCAN = SLSPOT(1) + SLSPOT(2)*EAS + SLSPOT(3)*NOR +.5
      SPOTELEM = ELSPOT(1) + ELSPOT(2)*EAS + ELSPOT(3)*NOR +.5
      WRITE(LUOUT,1000) EAS,NOR,DEPTH,TMELEM,TMSCAN,
      +              SPOTELEM,SPOTSCAN
      READ(LUSURV,500) EAS,NOR,DEPTH

```

```
ENDDO
```

```
WRITE(6,*)  
WRITE(6,*) '***** COMPLETED *****'  
WRITE(6,*)  
  
100  FORMAT(A)  
500  FORMAT(7X,D6.0,5X,D7.0,5X,F7.3)  
1000 FORMAT(4X,F10.0,2X,F10.0,2X,F7.3,4(2X,I6))  
1001 FORMAT(7X,'EASTING',4X,'NORTHING',4X,'DEPTH',4X,'TMELEM',2X,  
+      'TMSCAN',2X,'SPELEM',2X,'SPSCAN')  
      END
```

Distribution List

Director
Office of Naval Research
800 N. Quincy St.
Arlington VA 22217-5000
ATTN: Code 10
ATTN: Code 228
ATTN: Code 20PD
ATTN: Code 1125

Commanding Officer
Naval Oceanographic and Atmospheric Research Laboratory
Stennis Space Center MS 39529-5004
ATTN: Code 100
Code 115
Code 125L (10)
Code 125P
Code 300
Code 350
Code 351

Director
Defense Intelligence Agency
ATTN: DC-5B (B. Gordon)
Washington DC 20340-0001

University of South Alabama
University Station
Mobile AL 36688
ATTN: Physics (Dr. Clark)

University of Southern Mississippi
Hardy St.
Hattiesburg MS 39401
ATTN: Mathematics (Dr. Fay)

REPORT DOCUMENTATION PAGE

Form Approved
OBM No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. Agency Use Only (Leave blank).

2. Report Date.
January 1991

3. Report Type and Dates Covered.
Final

4. Title and Subtitle.

Multispectral Bathymetry Programs: A Users Guide

5. Funding Numbers.

Program Element No. 62435N

Project No. 3585

Task No. MOG

Accession No. DN255031

6. Author(s).

H. V. Miller, T. Green-Douglas, C. L. Walker, R. K. Clark and T. H. Fay

7. Performing Organization Name(s) and Address(es).

Naval Oceanographic and Atmospheric Research Laboratory
Ocean Science Directorate
Stennis Space Center, Mississippi 39529-5004

8. Performing Organization
Report Number.

NOARL Technical Note 95

9. Sponsoring/Monitoring Agency Name(s) and Address(es).

Naval Oceanographic and Atmospheric Research Laboratory
Code 113
Stennis Space Center, Mississippi 39529-5004

10. Sponsoring/Monitoring Agency
Report Number.

NOARL Technical Note 95

11. Supplementary Notes.

*Recently designated the Naval Oceanographic and Atmospheric Research Laboratory (NOARL)

12a. Distribution/Availability Statement.

Approved for public release; distribution is unlimited.

12b. Distribution Code.

13. Abstract (Maximum 200 words).

The Naval Ocean Research and Development Activity (NORDA) *, the Navy's lead laboratory in mapping, charting, and geodesy, is currently investigating the use of remotely sensed multispectral imagery as an accurate source for computing coastal-zone bathymetry. Because the Navy supports amphibious operations, special warfare, and coastal hydrographic surveying, knowledge of near-shore features is essential. The widespread availability, temporal sensitivity, and almost complete global coverage of most satellites' imagery make it an ideal way to collect water information from areas of limited or denied standard access. Bathymetry computations are done through software designed specifically for the ongoing research in this field. The software applications and abilities are discussed in this technical note.

14. Subject Terms.

(U) Hydrography, (U) Bathymetry, (U) Optical Properties, (U) Remote Sensing

(U) Reverberation

15. Number of Pages.

139

16. Price Code.

17. Security Classification
of Report.

Unclassified

18. Security Classification
of This Page.

Unclassified

19. Security Classification
of Abstract.

Unclassified

20. Limitation of Abstract.

SAR